DISSERTATION

submitted to the
Combined Faculties for the Natural Sciences and for Mathematics

of the Rupertus Carola University of
Heidelberg, Germany

for the degree of
Doctor of Natural Sciences

presented by

Diplom-Physicist: Jörg Langeheine
born in: Essen

Heidelberg, July 6, 2005

Intrinsic Hardware Evolution on the Transistor Level

Gutachter: Prof. Dr. Karlheinz Meier
Prof. Dr. Norbert Herrmann

Zusammenfassung

Intrinsische Hardware-Evolution von Transistorschaltungen -
Diese Arbeit stellt einen neuartigen Ansatz zur automatisierten Synthese analoger Schaltungen vor. Evolutionäre Algorithmen werden dazu benutzt, analoge Schaltungen zu generieren, die auf einem eigens dafür entworfenem Mikrochip, der als analoges Substrat fungiert, getestet werden. Die Tatsache, dass die Güte der von dem evolutionären Algorithmus generierten Schaltungen durch einen Test auf dem oben genannten Mikrochip erfolgt, hat zwei Vorteile. Einerseits kann es den künstlichen Evolutionsprozeß beschleunigen, da der Test einer Schaltung oft schneller erfolgen kann als seine Simulation. Andererseits garantiert diese Vorgehensweise, dass die evolutionierten Schaltungen tatsächlich auf einem echten Chip funktionieren. Die oben beschriebene Methode wird durch ein Hardware-Evolutionssystem realisiert, das aus den folgenden Komponenten besteht: Einem IBM kompatiblen Computer, auf dem der evolutionäre Algorithmus abläuft, einer FPGA basierten gemischt analogen-digitalen Messkarte und dem analogen Substrat. Dieses ist durch ein Field Programmable Transistor Array (FPTA) realisiert, dessen programmierbare Transistorzellen fast beliebig miteinander verschaltet werden können. Die Abmessungen des Transistorkanals können in jeder Transistorzelle aus 75 verschiedenen Kombinationen ausgewählt werden. Der Mikrochip ist in einem 0.6 µm CMOS Prozeß hergestellt worden und stellt vielfältige Möglichkeiten analoge Signale anzulegen und auszulesen zur Verfügung. Die Konfiguration des FPTA wird in SRAM Zellen gespeichert, die in die programmierbaren Transistorzellen eingebettet sind. In dieser Arbeit wird das Hardware-Evolutionssystem für die künstliche Evolution einer Vielzahl verschiedener Schaltungstypen verwendet. Die Zielschaltungen sind: Logische Gatter, Schaltkreise mit einer Gauß'schen Ausgangscharakteristik, D/A Wandler, Tief- und Hochpaßfilter, Tondiskriminatoren und Komparatoren. Die Ergebnisse der Experimente werden gründlich analysiert und mit anderen publizierten Arbeiten verglichen.

Abstract

Intrinsic Hardware Evolution on the Transistor Level -
This thesis presents a novel approach to the automated synthesis of analog circuits. Evolutionary algorithms are used in conjunction with a fitness evaluation on a dedicated ASIC that serves as the analog substrate for the newly bred candidate solutions. The advantage of evaluating the candidate circuits directly in hardware is twofold. First, it may speed up the evolutionary algorithms, because hardware tests can usually be performed faster than simulations. Second, the evolved circuits are guaranteed to work on a real piece of silicon. The proposed approach is realized as a hardware evolution system consisting of an IBM compatible general purpose computer that hosts the evolutionary algorithm, an FPGA-based mixed signal test board, and the analog substrate. The latter one is designed as a Field Programmable Transistor Array (FPTA) whose programmable transistor cells can be almost freely connected. The transistor cells can be configured to adopt one out of 75 different channel geometries. The chip was produced in a 0.6 µm CMOS process and provides ample means for the input and output of analog signals. The configuration is stored in SRAM cells embedded in the programmable transistor cells. The hardware evolution system is used for numerous evolution experiments targeted at a wide variety of different circuit functionalities. These comprise logic gates, Gaussian function circuits, D/A converters, low- and highpass filters, tone discriminators, and comparators. The experimental results are thoroughly analyzed and discussed with respect to related work.

Contents

Introduction **1**

I Foundations **7**

1 CMOS Analog Circuit Design **9**
1.1 CMOS Transistors . . . 10
1.1.1 Physical Definition . . . 10
1.1.2 Simple Low Frequency Transistor Model . . . 11
1.1.3 Capacitances of CMOS Transistors . . . 14
1.2 Linear Devices in CMOS Technology . . . 16
1.2.1 Resistors in CMOS technology . . . 16
1.2.2 Capacitors in CMOS . . . 17
1.3 CMOS Switches . . . 18
1.3.1 Switch Parasitics . . . 18
1.3.2 Dynamic Operation: Sample and Hold Circuits . . . 19
1.4 Simulation and Verification . . . 21
1.4.1 MOSFET Modeling . . . 22
1.4.2 Analysis Types . . . 22
1.4.3 Influence of Temperature and Device Mismatch . . . 23
1.4.4 Comment on Circuit Simulations in Evolutionary Algorithms . . . 24
1.5 Design Flow . . . 24

2 Evolutionary Algorithms in a Nutshell **27**
2.1 Biological Inspiration . . . 28
2.1.1 Darwinian Evolution . . . 28
2.1.2 Genetic Principles . . . 29
2.1.3 Ontogenesis . . . 31
2.2 Overview of Evolutionary Algorithms . . . 31
2.2.1 Operation Principle . . . 32
2.2.2 Components of an Evolutionary Algorithm . . . 32
2.2.3 Selection Schemes . . . 36
2.3 Dialects of Evolutionary Algorithms . . . 39
2.3.1 Genetic Algorithms . . . 39
2.3.2 Evolution Strategies . . . 40
2.3.3 Genetic Programming . . . 41
2.3.4 A Note on EA Parameters . . . 43
2.4 Evolutionary Algorithms as Global Optimizers . . . 45

2.4.1 Global Optimization . . . 45
2.4.2 Model-Free Heuristics . . . 46
2.4.3 No Free Lunch Theorem . . . 46
2.4.4 Implications of the Stochastic Nature of Evolutionary Algorithms . . . 47
2.5 Extensions and Refinements . . . 47
2.5.1 Distributed Populations . . . 47
2.5.2 Multi Objective Evolution . . . 48

II Evolution System 49

3 Implementation of the FPTA 51
3.1 Rationale . . . 52
3.1.1 The Very Idea of the Programmable Transistor Array . . . 52
3.1.2 Target Specifications . . . 53
3.2 Architecture of the FPTA . . . 55
3.3 Programmable Transistor Cell Array . . . 56
3.3.1 Transistor Cell Architecture . . . 56
3.3.2 Routing Concept . . . 57
3.3.3 Programmable Transistor . . . 58
3.3.4 Switch Dimensions . . . 60
3.3.5 Parasitic Devices . . . 61
3.3.6 Layout of the Programmable Transistor Cell . . . 67
3.4 SRAM for Configuration Storage . . . 70
3.4.1 SRAM Cell . . . 71
3.4.2 SRAM Architecture . . . 71
3.4.3 SRAM: IO-Circuitry . . . 73
3.4.4 SRAM: Concluding Remarks . . . 76
3.5 IO-Cells . . . 78
3.5.1 Functionality of the IO-cells . . . 78
3.5.2 Architecture of the IO-Cells . . . 80
3.5.3 Configuration of the IO-cells . . . 81
3.5.4 Sample and Hold Units: The Analog Perspective . . . 84
3.5.5 Layout of an IO-Cell . . . 87
3.6 Inner-Cell Signal Probing . . . 88
3.6.1 Inner-Cell Probing Concept . . . 88
3.6.2 Implementation of the Inner-cell Probing . . . 88
3.7 Family of Rail-to-Rail Operational Amplifiers . . . 92
3.7.1 IO-cell Buffer . . . 93
3.7.2 Global Output Buffer . . . 94
3.7.3 Cell Buffer . . . 95
3.7.4 Summary of Simulated Performance . . . 95
3.8 Measurement of Die Temperature . . . 99
3.9 Layout of the Complete Chip . . . 100
3.10 Yield Analysis . . . 102
3.11 Comparison of the Heidelberg and JPL FPTA Chips. . . . 102

4 Evolution System **105**
4.1 Overview of the Evolution System . . . 106
4.2 Mixed-Signal Test Environment . . . 107
4.2.1 Electrical Test System: Background . . . 107
4.2.2 Test Environment: Digital Part . . . 108
4.2.3 Analog Signal Path . . . 111
4.3 Hardware Control Software . . . 113
4.3.1 Overview . . . 114
4.3.2 Analog Test Engine . . . 116
4.4 DarkGAQT Software Package . . . 122
4.4.1 Overview . . . 122
4.4.2 Multithreading Architecture . . . 126
4.4.3 Implementation of the Evolutionary Algorithm . . . 128

III Experiments and Results **131**

5 Evolution of Quasi-DC Solutions **133**
5.1 Introduction . . . 133
5.1.1 Rationale . . . 133
5.1.2 Related Work . . . 134
5.2 Experimental Setup . . . 136
5.2.1 Problem Definition . . . 136
5.2.2 Geometrical Setup . . . 139
5.2.3 Overview of the Experiments . . . 139
5.2.4 GA Parameters . . . 141
5.2.5 Verification Tests . . . 141
5.3 Results: Evolution of Logic Gates I . . . 141
5.4 Results: Evolution of Logic Gates II . . . 144
5.4.1 Overview over the Results of all Runs . . . 144
5.4.2 Output Characteristic of the Best Evolved Gates . . . 146
5.4.3 Performance Comparison for Different Tests . . . 148
5.5 Results: Evolution of DC V-V Gaussian Circuits . . . 150
5.5.1 Overview over All Experiments . . . 150
5.5.2 Output Characteristic of the Best Evolved Circuits . . . 150
5.5.3 Verification Measurements . . . 153
5.6 Discussion . . . 153

6 Evolution of Digital-to-analog Converters **155**
6.1 Experimental Setup . . . 156
6.1.1 Problem Definition . . . 156
6.1.2 Overview of the Experiments . . . 157
6.1.3 Fitness Function . . . 158
6.1.4 Test Patterns . . . 159
6.1.5 GA Parameters . . . 160
6.2 Results for Series FW1 . . . 160
6.2.1 Root Mean Square Error . . . 161
6.2.2 Nonlinearity, Offset and Gain Error . . . 161

6.3 Comparison of the Five Different Series of Experiments . . . 164
6.3.1 Root Mean Square Error . . . 164
6.3.2 Offset, Gain and Nonlinearities . . . 167
6.3.3 Best per Series Results . . . 170
6.4 Generalizability of the Results of Series FW1 and FWB4 . . . 173
6.4.1 Verification at a Second Time Scale . . . 173
6.4.2 Performance on a Second Chip . . . 174
6.5 Discussion . . . 177

7 Evolution of Filters 181
7.1 Introduction . . . 181
7.1.1 Conventional Design of Analog VLSI Filters in a Nutshell . . . 182
7.1.2 Related Work . . . 189
7.2 Evaluation of the Magnitude Response . . . 194
7.2.1 M1: Transfer Function from the Step Response . . . 194
7.2.2 M2: Magnitude Response from the Mean Signal Power . . . 196
7.2.3 M3: Magnitude Response from Fourier Analyzed Sinusoidal Stimuli . . . 198
7.2.4 Noise Floor . . . 201
7.2.5 List of All Test Modes . . . 207
7.2.6 Randomization for Time-Dependent Experiments . . . 208
7.3 Evolving Low Pass Filters . . . 209
7.3.1 Experimental Setup . . . 209
7.3.2 Illustration and Discussion of the Different Test Modes . . . 212
7.3.3 Reproducibility . . . 215
7.3.4 Comparison of Different Experiments . . . 218
7.3.5 Comparison of Different Series of Experiments . . . 222
7.3.6 Migration to a Second Chip . . . 227
7.4 Evolving LPFs on Different Frequency Scales . . . 229
7.4.1 Experimental Setup . . . 229
7.4.2 Output Behavior for the Best-Of-Experiment Circuits . . . 229
7.4.3 Statistical Analysis . . . 231
7.4.4 Migration to a Second Chip . . . 232
7.5 High Pass Filter . . . 233
7.5.1 Experimental Setup . . . 233
7.5.2 Output Behavior for the Best-Of-Experiment Circuits . . . 235
7.5.3 Statistical Analysis . . . 237
7.5.4 Reproducibility . . . 239
7.5.5 Migration to a Second Chip . . . 240
7.6 Summary and Discussion . . . 240
7.6.1 Lessons Learned . . . 241
7.6.2 Comparison with Related Work . . . 243

8 Evolution Using Human Made Building Blocks 245
8.1 Methodology . . . 246
8.1.1 Rationale . . . 246
8.1.2 Related Work . . . 247
8.1.3 Building Block Concept . . . 248
8.1.4 Implementation . . . 248

8.2 Experimental Setup for Case Studies I and II . . . 250
8.2.1 Geometrical Setup for Case Studies I,II . . . 250
8.2.2 Building Block Library for Case Studies I,II . . . 251
8.2.3 Evolutionary Algorithm for Case Studies I,II . . . 252
8.3 Case Study I: XOR/XNOR Gate . . . 252
8.3.1 Problem definition . . . 253
8.3.2 Results . . . 254
8.4 Case Study II: Tone Discrimination . . . 257
8.4.1 Problem Definition . . . 257
8.4.2 Results . . . 259
8.5 Case Study III: Comparators . . . 264
8.5.1 Geometrical Setup . . . 268
8.5.2 Building Block Library . . . 270
8.5.3 Experimental Setup . . . 278
8.5.4 Overview of the Experiments . . . 282
8.5.5 Test of the Hand-Designed Comparator . . . 282
8.5.6 Evolution Results . . . 284
8.5.7 Summary of the Comparator Results . . . 293
8.6 Discussion . . . 295
8.7 Disclaimer and Acknowledgment . . . 296

Conclusion 297

A FPTA Chip: Configuration Details and Pad and Signal Description 305
A.1 Configuration Bit Assignment . . . 305
A.2 Bond Pads of the FPTA . . . 307
A.3 Bonding Diagram . . . 308
A.4 Chip Carrier Pins . . . 309
A.5 Signal Description . . . 310
A.6 Signal – Bond Pad – Carrier Pin Assignment . . . 313

B Rail-to-Rail Operational Amplifier: Simulation Results 315
B.1 Simulation of the Rail-to-Rail IO-cell Buffer . . . 316
B.2 Simulation of the Rail-to-Rail Output Buffer . . . 323
B.3 Simulation of the Rail-to-Rail Cell Buffer . . . 327

C Correlation Coefficient 335

D Signals and Systems Analysis 337
D.1 Discrete Fourier Transform . . . 337
D.1.1 Parseval's Relation for the DFT . . . 337
D.2 Around the Unit Circle on a Shoestring . . . 338
D.3 LTI System Response to Sinusoidal Inputs . . . 339

E Additional Series of Comparator Experiments: Large Settling Time 341
E.1 Comparison of the Different Experiments: Histograms . . . 341
E.2 Comparison of the Different Experiments: Convergence . . . 342
E.3 Comparison of the Different Experiments: Gain and Offset . . . 343
E.4 Best-of-Experiment Output Characteristics . . . 343

E.5 Test on a Second Chip 346

F List of Acronyms **347**

Introduction

> I am turned into a sort of machine for observing facts and grinding out conclusions.
>
> CHARLES DARWIN

The invention of the transistor in 1947 and the first successful integration of several transistors into one monolithic circuit in 1958 paved the way for a breathtaking development that has largely influenced and shaped today's life. Highly integrated microelectronic circuits are the cornerstone for the present era of information technology that brought radically new means of communications like cellular phones, e-mail or the internet. At the beginning of the third millennium, the aid of electronic devices like computers, microcontrollers or sensors seem to be indispensable for a vast variety of tasks ranging from office organization to space exploration as well as from medical applications to most of scientific research and design, control, and optimization in engineering.

As in recent years many signal processing tasks have been shifted from the analog to the digital domain, the aforementioned technological progress is often ascribed to the advances in digital circuit design. Yet, this is not the full truth: First, the nature of digital circuits is analog. On one hand, performance critical parts still require full custom design to push the technological limits. Second, prior to being used by digital synthesis tools, a library of logic primitives like gates, flipflops etc. must be generated so as to meet a set of analog specifications, as e.g. setup and hold times as well as acceptable input voltage ranges, noise immunities or gate delays. It is not until these devices are designed and verified that abstraction from their analog behavior can be achieved. Second, although nature exhibits quantized behavior according to last centuries ground breaking quantum theory, there is hardly any access thereof without analog electronics. In other words, even if most of the signal processing can indeed be done in digital hardware, any connection to the real word, be it to mass storage devices, be it wireless communication, a sound or graphics device, or the readout of a light intensity in an optical storage device, necessitates the conversion into an analog signal. Moreover, in a variety of consumer, engineering, and scientific products, sensors – in which a physical quantity is again first converted into an analog signal – play a crucial role. In summary, analog circuits will persist to be of great importance in future microelectronic systems.

Although the first integrated circuits consisted of bipolar junction transistors (BJT), CMOS[1] has become the prevalent integrated circuit technology during the last two decades. Its high integration density paired with low static currents have proven to be most appropriate for digital circuit. In case of analog circuits, both, BJT and CMOS technology, as well as the combination thereof, that is BiCMOS[2] technology are used. Yet, most often CMOS is also the choice for analog circuits. For one, this allows to benefit from the rapid advances in process technology driven by the larger digital market. For the other, analog subsystems are nowadays frequently integrated together with digital systems to form a

[1] Complementary Metal-Oxide Semiconductor

[2] Bipolar Complementary Metal-Oxide Semiconductor

system on a chip (SoC). Here again the requirements of the digital part dictate the process technology to be CMOS.

Unfortunately, the design of analog circuits is an intricate, tedious, time-consuming task, as it requires the designer to think about any single transistor. At first, a suited topology has to be found. In the next step the parameters of all the circuit components must be optimized, and finally, the resulting circuit has to be laid out. Even worse, because of parasitic resistors and capacitances introduced by the layout, it may be necessary to repeat the process of parameter optimization and layout several times. Furthermore, as the behavior of an analog circuit is inextricably connected to the device physics of the used circuit elements, great care must be taken to minimize an analog circuit's sensitivity to variances in the manufacturing process and its operating environment. Yet, even for a suited, fixed topology, a design of moderate size can comprise difficult relationships between some tens of design variables and a dozen performance goals.

Analog Design Automation

The efforts to automate the digital design process have led to sophisticated tools that greatly leverage productivity. In the best case, the digital design can be defined in a hardware description language. This abstract description is then automatically translated into a suited netlist, which is eventually mapped onto the target technology at hand. Here, the success of digital synthesis tools is largely due to the high level of abstraction achieved for the utilized digital building blocks in that they can be characterized relatively uniformly by a few timing constants. In comparison to digital synthesis, the field of analog design automation is still in its infancy, which is not surprising given the tight linkage of analog circuit behavior and device physics mentioned above. In the context of SoC designs, the vast difference in the current productivity in analog and digital design, which leads to situations in which the analog part only occupies 10% of the whole chip, yet requires 90% of the design time, is sometimes referred to as *the analog dilemma* [Tou02]. Accordingly, improved design methodologies as well as advances in analog design automation are believed to be of utmost importance to eliminate this bottleneck encountered in SoC designs [Gie00], [Sha02], [Phe00a].

Analog design, and therefore analog synthesis comprises to steps: First, an electrical design – the schematic – of a circuit has to be generated. Second, this design has to be translated into an appropriate layout that it used for the production of the circuit (physical synthesis). Here, the first task, that is, finding an appropriate analog circuit for the problem at hand can be divided into four levels of difficulty [Lia01]:

1. Local parameter optimization: Fine-tune the device sizes of a given circuit topology, starting with a working set of circuit parameters.

2. Global parameter optimization: Find optimal circuit parameters for a given topology.

3. Topology selection: Choose the best out of a given set of predefined topologies and optimize its parameters.

4. Circuit design: Invent a circuit that meets the desired specifications from scratch.

The majority of attempts to automatic synthesis of analog circuits attack at the second level, that is they try to find the optimal design variables for a given topology. Thereby, the description of the optimization problem will embody some principles of conventional analog circuit design, as for instance, that matched transistor pairs ought to possess the same channel dimensions. The reported approaches to this kind of parameter optimization can be classified in three groups [dMH02a], namely, knowledge

based, simulation based [Phe00a], [Alp03], and equation based methods [dMH01]. During the last couple of years, the latter two methods have been proven successful in optimizing high-performance analog cells, as e.g. operational amplifiers [Lia01], [Hen02], or an equalizer/filter block [Phe00b]. In case of the equation based method, even more complex problems have been tackled, for instance, the design of a pipeline analog-to-digital converter [dMH02a], or a phase locked loop [dMH02b]. Yet, it has only been very recently, that some of the attempts have matured into commercial products [Cad04b], [Syn05]. The same is true for one physical synthesis tool that allows the user to capture analog design constraints, which are then used to generate the layout [Cad04a].

Hardware Evolution

Although the tools described above are certainly an important help for the analog designer, they nevertheless fall short of finding new, more efficient circuit topologies. Yet, the incorporation of topological synthesis adds two new opportunities to analog design automation: First, it can provide new circuit topologies for problems for which a satisfactory design solution has not yet been found. It should be noted that this situation can also arise when the migration to a new circuit technology renders formerly successful circuit topologies useless, e.g. due to a decrease in power supply voltage. Second, analog synthesis including topology search may be able to exploit the actual transistor physics more effectively than today's conventional circuit topologies, because they are not restricted to the human way of thinking about circuit design [Tho99]. Thus, the resulting circuits may yield better or equal performance whilst consuming equal or less power and area.

Although the automatic invention of analog circuits is an intriguing idea, one must not forget that the inclusion of topology selection in the synthesis process vastly increases the complexity of the problem. In fact, to date, no heuristic algorithm is known that could deterministically solve the problem of analog circuit invention, which is not surprising if one takes into account that it has only been recently that a heuristic could be developed for analog synthesis on the parameter optimization level [dMH01].

On the other hand, evolutionary algorithms, a class of model-free heuristics, are often successfully used to solve hard optimization problems that cannot be solved deterministically. In analogy to natural evolution, evolutionary algorithms process a whole population of feasible solutions. In the course of an artificial evolution experiment, the members of the population are varied by genetic operators like mutation and crossover. A selection mechanism ensures that the probability for a member of the population to propagate its genetic information to the next generation depends on their ability to solve the problem at hand. Hence, probabilistic sampling of the search space is combined with a selection pressure towards better solutions.

The application of evolutionary algorithms to engineering design problems is one, if not the main topic of a relatively young research field referred to as hardware evolution or evolvable hardware. Generally speaking, the field comprises a wide variety of approaches including evolutionary algorithms, modeling of ontogenetic processes, and the design of electronic systems that mimic the behavior of a biological immune system. However, in the context of this thesis, the first approach, namely that to use evolutionary algorithms to solve synthesis problems, is most relevant. In principle, the field of hardware evolution lends itself to the evolutionary development of any kind of hardware. Practically, the target systems are almost exclusively restricted to be realizable as digital or analog circuits. Depending on the implementation process, hardware evolution is either referred to as *extrinsic* or *intrinsic*. It is *extrinsic* if the behavior of the phenotype is simulated, and *intrinsic* if hardware is used in the evolutionary loop to evaluate the fitness of candidate solutions produced by the algorithm.

Scope of this Thesis

On one hand, extrinsic, that is, simulation based hardware evolution may be more flexible, and its implementation requires less effort if existing circuit simulators can be used. On the other hand, intrinsic hardware evolution offers other beneficial properties. First, in case of intrinsic evolution experiments, the successfully evolved circuits are guaranteed to work at least on the particular substrate they are evolved on. In contrast, in extrinsic approaches, the evolutionary algorithm may not always find optimal circuit solutions but rather tend to exploit imperfections of the circuit simulator or the test setup used for the fitness evaluation. Second, hardware tests can speedup artificial evolution in that they are often faster than simulations. The potential speedup is most evident for time-consuming transient analyses of large designs, since simulation times scale with the size of the circuit, whereas a hardware test is independent of the circuit size for a given task. Moreover, intrinsic fitness evaluations inherently capture the real conditions the prospective circuits are to work in, as e.g. noise and the imperfections of the production process. The simulation of such effects dramatically increases the time necessary for a fitness evaluation. In fact, the verification of circuits invented by extrinsic hardware evolution will most probably have to be more elaborate than for circuits consisting of a well known topology whose design variables are optimized, as the sensitivity to variations of the circuit parameters or operating conditions are much more uncertain.

Yet, intrinsic hardware evolution also offers a completely new perspective conceivable as field evolvable hardware. During the last decade, the success of programmable logic arrays, as e.g. FPTAs[3], has inspired similar developments for the analog domain. These FPAAs[4] are general purpose analog devices that offer a variety of different analog functions. Typically, they are composed of analog cells like operational amplifiers [Zet99] and are targeted at filtering and signal conditioning applications [Lat01], [AN203].

Although recent FPAAs usually can be conveniently configured through dedicated software tools, field evolvable hardware can be advantageous in the following regards: First, here too hardware evolution may simplify the design task at hand. Second, field evolvable hardware can adapt to electrical and environmental conditions that cannot be known a priori. Such situations may arise e.g. in DSL[5] applications, where the precise electrical properties of the copper telephone wire used for connecting the subscriber to a broadband network are unknown, the calibration of intermediate frequency filters necessitated by the fabrication tolerances [Mur03], or space missions, in which the electronic circuitry has to cope with temperature variations of several hundred Kelvin [Sto04], [Zeb04].

The above considerations have motivated an intrinsic hardware evolution approach to analog synthesis. In particular, an analog substrate has been sought that lends itself to both, the synthesis of new transistor level circuits, and to field evolvable hardware applications. In order to support the search for new transistor level circuits, however, the analog substrate must offer the configurability of single transistors. As commercially available FPAAs are usually rather coarse-grained in that they only allow the configuration of high level building blocks, the quest for transistor level hardware evolution necessitated the design of a new analog substrate. This ASIC[6] provides an array of transistor cells that are programmable in their channel geometry as well as in their connectivity. It is dedicated to hardware evolution experiments and referred to as a field programmable transistor array (FPTA).

The goals of the Heidelberg FPTA project can be summarized as follows: First, the FPTA is designed as a search tool to find new analog transistor level circuits. Accordingly, the cells of the FPTA are used as a model for programmable CMOS transistors so that evolved circuits can be understood in

[3] field programmable transistor arrays
[4] Field Programmable Analog Arrays
[5] Digital Subscriber Line
[6] Application Specific Integrated Circuit

terms of simulation and human design experience. The use of hardware-in-the-loop may accelerate the evaluation of candidate solutions while avoiding the simplifications inherent to simulations. Second, the FPTA is a first step towards field evolvable hardware. The device may be used to perform analog tasks that cannot be a priori specified or need the analog circuit to adapt to changing environments. Third, the FPTA can be used as a research tool to learn how to use artificial evolution for the invention of systems with higher complexity from an algorithmic point of view.

Within the scope of this thesis, the FPTA as a fine-grained analog substrate dedicated to the intrinsic hardware evolution of transistor level circuits has been designed, and has been embedded into a suited hardware evolution system that consists of a host computer, a mixed-signal test environment, the FPTA itself, and a software front end that encapsulates the control and configuration of the system and also implements the evolutionary algorithm. The feasibility of the proposed intrinsic hardware evolution approach is tested on a variety of different circuit design problems. Most of these experiments concentrate on establishing and verifying appropriate test methods for the evaluation of candidate circuits rather than algorithmic sophistication. Nevertheless, the last chapter is dedicated to enhancing artificial evolution's success by providing small building blocks to the algorithm.

Outline

This thesis is divided into three main parts. As the central idea of this thesis is to apply evolutionary algorithms to the design of analog circuits, the most relevant aspects from both fields are summarized in part I: Chapter 1 reviews the characteristics of CMOS transistors as well as the analog design process, whereas chapter 2 gives a short introduction to and an overview of evolutionary algorithms.

The realization of the hardware evolution system is covered in part II. The FPTA chip represents the core of the evolution system. Since the FPTA is not only a product of this thesis, but also a subject of research itself, its implementation is described in detail in chapter 3. The hardware evolution system as a whole is surveyed in chapter 4. Although software as well as external hardware are of equal importance as the FPTA itself and although their design eventually may have consumed more effort than the chip, the discussion of these components is kept to a rather functional level for the following three reasons: First, external software and hardware are not of as much scientific relevance as the FPTA chip, at least not the details of their implementation. Second, as large parts of the auxiliary components of the evolution system are described as C++ or VHDL[7] code, a minimum of documentation is already provided in the code. Third, some of the components used in the hardware evolution system are shared with other projects of the Electronic Vision(s) group and are documented elsewhere.

Finally, part III presents a wide range of experiments that utilize the hardware evolution system presented in part II to evolve different types of analog circuits. The experiments are organized in four separate chapters to account for the respective nature of the target circuits as well as for different methodological approaches taken. Chapter 5 describes different attempts to evolve the analog dc behavior of logic gates and Gaussian function circuits. Here, the key question is how to test the dc behavior in a real measurement, that is in a finite amount of time. The experiments of chapter 6 are targeted at the artificial evolution of 6-bit digital to analog converters. Methodologically, these experiments are similar to those of chapter 5. Yet they exploit an improvement of the hardware evolution system to allow for real-time testing as well as for greater flexibility in the design of the circuit test. Chapter 7 discusses different methods to evaluate the frequency behavior of the circuit under test. The chapter contains a short review of LTI[8] systems as well as a theoretical account of the

[7] Verilog Hardware Description Language
[8] Linear Time-Invariant

noise contributions for the three test methods to be compared. The implemented test concepts are then used to evolve low- and highpass filters. While chapter 5 to 7 reflect the extensions and refinements achieved in the test of candidate circuits, chapter 8 proposes a new circuit representation scheme to support the synthesis of analog circuits. The hardware evolution system is extended such, that user-defined building blocks can be used as elementary cells instead of plain transistor cells. The concept is tested for two different building block libraries that are applied to three target functionalities. These are XOR/XNOR gates, tone discriminators that distinguish two frequencies of a square wave, and comparators.

Most of the chapters in this thesis are highly self-contained. For instance, all chapters in the last part are almost independent of each other[9]. In the rare cases in which a definition of a previous chapter should indeed be required, the reader is referred to the according section. It goes without saying, that the introductory chapters on CMOS technology and evolutionary algorithms are optional for readers that are not familiar with the respective topic. However, chapter 1 also includes some details that are used in the description of the FPTA chip in chapter 3. It also contains a brief discussion on possible advantages of a real circuit test over simulations in the context of hardware evolution experiments. Finally, chapters 3 and 4 are organized so that only the first sections are required to understand most of the experiments and results presented in part III as well as to appreciate the concept of the proposed hardware evolution system. In particular, these are sections 3.1 to 3.3, and 4.1 and 4.4.3.

[9] In particular, the numbering of experiments, series of experiments, and case studies is always limited to the scope of the respective chapter.

Part I

Foundations

Chapter 1

CMOS Analog Circuit Design

God runs electromagnetics by wave theory on Monday, Wednesday, and Friday, and the Devil runs them by quantum theory on Tuesday, Thursday, and Saturday.

SIR WILLIAM BRAGG

This chapter offers a short review of those topics in CMOS analog circuit design that are relevant to this thesis: The first part concentrates on the properties of the devices available in CMOS technology, especially the characteristics of CMOS transistors. As the array of programmable transistors proposed in chapter 3 does not provide any passive components, the implementation of resistors and capacitors by means of transistors is discussed. A short discussion of the intricacies of MOS switches is included, as they are extensively used in the FPTA design. The second part deals with the design process itself: This includes transistor models, simulation types and the overall design flow for analog circuit design.

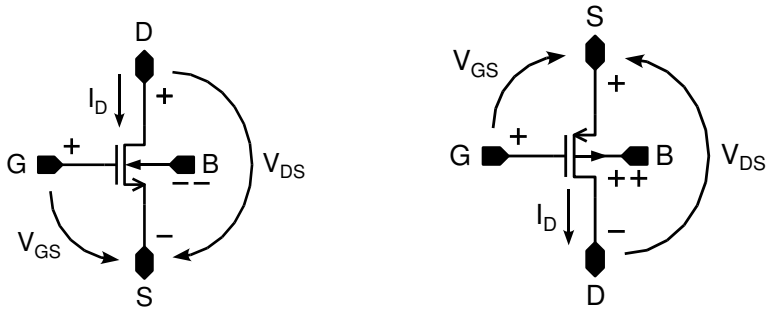
During the last 20 years, CMOS technology has become the prevalent integrated circuit technology. It is the technology of choice in the digital domain, where its combination of integration density and lack of static currents is still unrivaled, and also widely used in the analog world. Though offering some nice features, as for instance zero gate currents, the success of CMOS technology in analog designs is also partly owing to its popularity in the digital world, which renders CMOS technology the most advanced and most widely available integration technology [All02a]. The advent of systems on a chip (SoC) mixing analog and digital circuits on one chip has further increased the need for analog CMOS designs: Here, the quest for high integration densities for the digital part imposes the choice of CMOS technology.

As this thesis is focused on the automation of analog circuit design, a short introduction into the subject of this endeavor shall be given. This comprises a short account of the devices available in a CMOS process and some of their properties as well as brief sketch of the design process itself.

However, this chapter is by no means exhaustive and cannot replace the study of the text books relevant to the topic of analog circuit design in CMOS technology. As for this thesis, most of the information is taken from the first chapters of Allen and Holberg [All02b], Geiger, Allen and Strader [Gei90a] and Laker and Sansen [Lak94a].

1.1 CMOS Transistors

Electrically, MOS[1] transistors are four terminal devices that are usually either used as voltage controlled resistors or current sources. In CMOS technology, transistors come in two flavors, namely NMOS[2] and PMOS[3] transistors, depending on the type of charges carrying the current through the device. Transistors relying on electrons are denoted as NMOS and those using holes are called PMOS transistors. Their respective symbols as well as some of the electrical properties used for their characterization are defined in Fig. 1.1 For an NMOS transistor, gate voltage V_G and to a much smaller



(a) NMOS (b) PMOS

Figure 1.1: Terminal and terminal voltage definitions for an N- (a) and PMOS (b) transistor.

extent the bulk voltage V_B control the current I_D flowing from drain D to source S. The drain and source terminals are physically equivalent; yet electrically their influence on the transistor behavior is different. Thus, they are defined such that the terminal at the higher electrical potential is always the drain terminal. The voltage at the bulk terminal must always be smaller or equal to the source potential. PMOS transistors work similar to their NMOS counterparts. Yet, some of the electrical properties change their sign: The source potential by definition exceeds the drain potential and the gate-source voltage V_{GS} must be negative to allow for current I_D to flow. Proper operation of the PMOS transistor requires the potential at the bulk terminal to be at least as high as the source potential.

1.1.1 Physical Definition

The operation principle of MOS transistors is further explained by means of their physical implementation depicted in Fig. 1.2. The diagram applies to an n-well CMOS process and the layout of the n- and p-channel assume that both transistors are operated in the linear region. In terms of the n-channel transistor, the bulk, the source and drain terminals form pn junctions that are reverse biased according to the abovementioned constraint for V_B. Accordingly, if transistor gate and bulk are on the same potential, no significant current can be flowing between the source and drain terminals, at least as long as the break-down voltage of the pn junction is not exceeded. However, a positive gate voltage

[1] Metal-Oxide Semiconductor
[2] N-channel Metal-Oxide Semiconductor
[3] P-channel Metal-Oxide Semiconductor

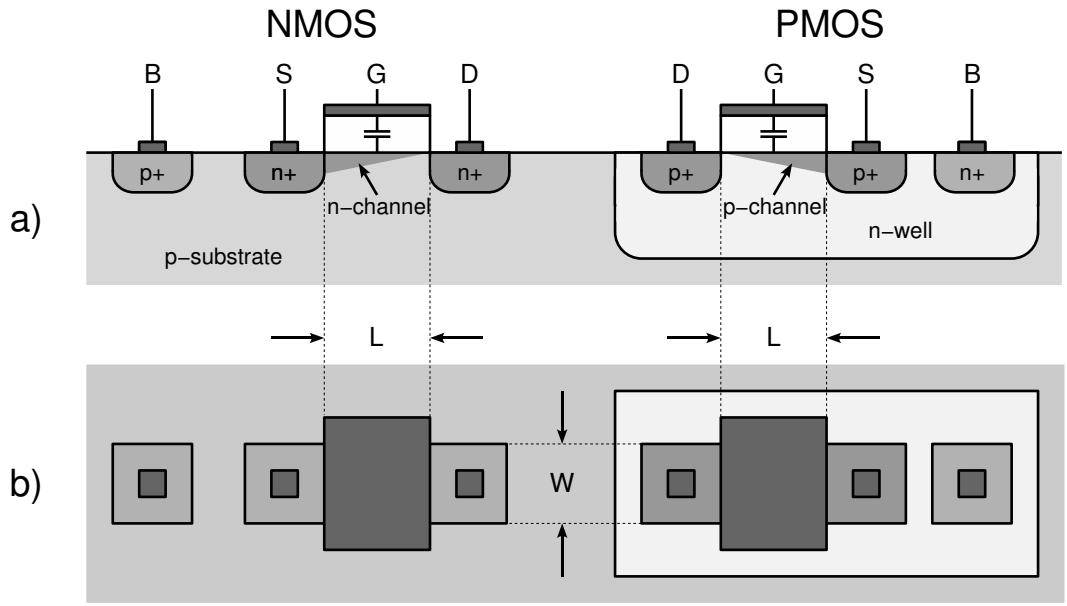


Figure 1.2: a) Cross section through an N- and PMOS transistor. b) Top view of the same N- and PMOS transistor.

can dramatically change this situation in that the gate and the bulk beneath the gate now form two plates of a capacitor with the gate oxide serving as the dielectric. Thus for a positive V_{GS}, a depletion region is generated in the bulk beneath the gate by means of electrostatic induction: Now source and drain are connected through this depletion region also called n-channel. Qualitatively, a larger voltage difference between source and gate, will cause a larger current to flow between source and drain for a given $V_{DS} > 0$. In an n-well process, the bulk terminal of a PMOS transistor is not connected to the substrate but to the n-well. In contrast to NMOS transistors, which all have to share the same bulk potential, one or a group of PMOS transistors can in principle have their own bulk potential. Yet, often there is neither need nor great benefit in an individual control of the bulk voltage, such that often it is simply connected to the highest possible voltage called vdd. In fact, all of the n-wells in the FPTA chip proposed in chapter 3 are connected to vdd.

1.1.2 Simple Low Frequency Transistor Model

A quantitative analysis of the physics of the semiconductor devices depicted in Fig. 1.2 leads to a quantitative model. A simple low frequency model, proposed similarly in [Lak94a], [Gei90a] and [All02b], is summarized in Table 1.1. The model describes the dependence of the terminal currents I_D and I_G on the respective voltage differences V_{GS} and V_{DS}, the channel geometry and the parameters of the fabrication process. As the gate is isolated from all other terminals by means of the gate oxide, there is no effective gate current, which is expressed by (1.1a) and (1.2a). The expressions describing the drain current I_D depend on the so called operation region, which is defined by V_{GS} and the relation of V_{DS} to V_{GS}. The different operation regions and the respective drain current dependence are exemplified below for an NMOS transistor.

Degrees of Substrate Inversion. The gate source voltage V_{GS} determines the amount of minority carriers gathered below the gate. If their density exceeds that of the majority carriers, the substrate underneath the gate will change its semiconductor type (e.g. from p-type to n-type in case of an NMOS transistor), which is called inversion. Consequently, for $V_{GS} = 0$, there is no inversion at all;

NMOS transistor

$$I_G = 0 \tag{1.1a}$$

$$I_D = 0 \quad \text{for} \quad V_{\mathrm{GS}} \cong 0\,,\ V_{\mathrm{DS}} \geq 0 \quad \text{cutoff} \tag{1.1b}$$

$$I_D = \frac{W}{L} I_{\mathrm{D0}}\, e^{-V_{\mathrm{BS}}\left(\frac{1}{nV_{\mathrm{th}}}-\frac{1}{V_{\mathrm{th}}}\right)} \left(1 - e^{-\frac{V_{\mathrm{DS}}}{V_{\mathrm{th}}}}\right) e^{\frac{V_{\mathrm{GS}}-V_T}{nV_{\mathrm{th}}}} \quad \text{for} \quad 0 < V_{\mathrm{GS}} < V_T + nV_{\mathrm{th}} \quad \text{weak inv.} \tag{1.1c}$$

$$I_D = K' \frac{W}{L}\left(V_{\mathrm{GS}} - V_T - \frac{1}{2}V_{\mathrm{DS}}\right) V_{\mathrm{DS}} (1 + \lambda V_{\mathrm{DS}}) \quad \text{for} \quad \begin{cases} V_{\mathrm{GS}} \geq V_T + nV_{\mathrm{th}} \\ 0 < V_{\mathrm{DS}} \leq V_{\mathrm{GS}} - V_T \end{cases} \quad \text{linear} \tag{1.1d}$$

$$I_D = K' \frac{W}{2L} (V_{\mathrm{GS}} - V_T)^2 (1 + \lambda V_{\mathrm{DS}}) \quad \text{for} \quad \begin{cases} V_{\mathrm{GS}} \geq V_T + nV_{\mathrm{th}} \\ V_{\mathrm{DS}} \geq V_{\mathrm{GS}} - V_T > 0 \end{cases} \quad \text{saturation} \tag{1.1e}$$

$$\text{where} \quad V_T = V_{\mathrm{T0}} + \gamma\left(\sqrt{2|\phi_b| - V_{\mathrm{SB}}} - \sqrt{2|\phi_b|}\right) \tag{1.1f}$$

PMOS transistor

$$I_G = 0 \tag{1.2a}$$

$$I_D = 0 \quad \text{for} \quad V_{\mathrm{GS}} \cong 0\,,\ V_{\mathrm{DS}} \leq 0 \quad \text{cutoff} \tag{1.2b}$$

$$I_D = \frac{W}{L} I_{\mathrm{D0}}\, e^{V_{\mathrm{BS}}\left(\frac{1}{nV_{\mathrm{th}}}-\frac{1}{V_{\mathrm{th}}}\right)} \left(1 - e^{\frac{V_{\mathrm{DS}}}{V_{\mathrm{th}}}}\right) e^{-\frac{V_{\mathrm{GS}}-V_T}{nV_{\mathrm{th}}}} \quad \text{for} \quad 0 > V_{\mathrm{GS}} > V_T - nV_{\mathrm{th}} \quad \text{weak inv.} \tag{1.2c}$$

$$I_D = K' \frac{W}{L}\left(V_{\mathrm{GS}} - V_T - \frac{1}{2}V_{\mathrm{DS}}\right) V_{\mathrm{DS}} (1 - \lambda V_{\mathrm{DS}}) \quad \text{for} \quad \begin{cases} V_{\mathrm{GS}} \leq V_T - nV_{\mathrm{th}} \\ 0 > V_{\mathrm{DS}} \geq V_{\mathrm{GS}} - V_T \end{cases} \quad \text{linear} \tag{1.2d}$$

$$I_D = K' \frac{W}{2L} (V_{\mathrm{GS}} - V_T)^2 (1 - \lambda V_{\mathrm{DS}}) \quad \text{for} \quad \begin{cases} V_{\mathrm{GS}} \leq V_T - nV_{\mathrm{th}} \\ V_{\mathrm{DS}} < V_{\mathrm{GS}} - V_T \end{cases} \quad \text{saturation} \tag{1.2e}$$

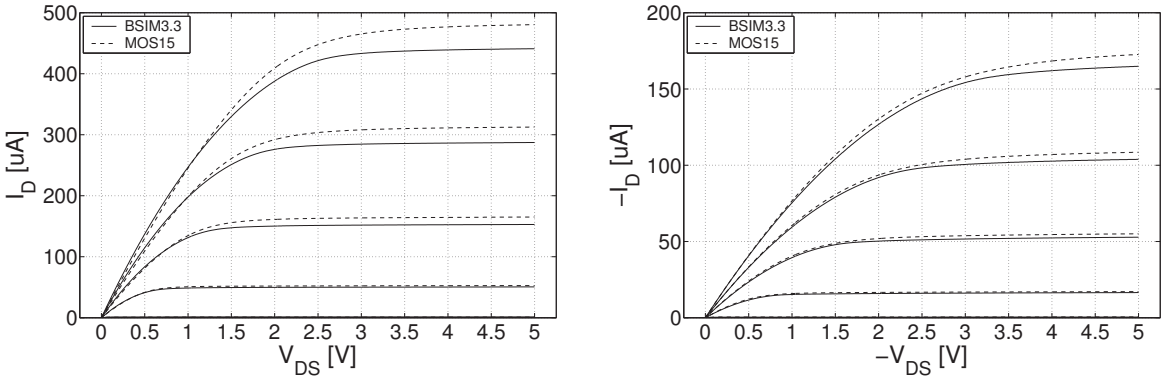
$$\text{where} \quad V_T = V_{\mathrm{T0}} - \gamma\left(\sqrt{2|\phi_b| + V_{\mathrm{SB}}} - \sqrt{2|\phi_b|}\right) \tag{1.2f}$$

			Typical Parameter Values		
Type	**Parameter**	**Description**	**NMOS**	**PMOS**	**Units**
design	W	channel width			µm
	L	channel length			µm
process	$K' = \mu C_{\mathrm{ox}}$	transconductance parameter	110	50	$\frac{\mu A}{V^2}$
	V_{T0}	threshold voltage ($V_{\mathrm{BS}} = 0$)	0.7	-0.7	V
	γ	bulk threshold parameter	0.4	0.57	$V^{1/2}$
	$2\lvert\phi_b\rvert$	strong inversion surface potential	0.7	0.8	V
	λ	channel length modulation parameter	0.04 ($L = 1\,\mu$m) 0.01 ($L = 2\,\mu$m)	0.05 ($L = 1\,\mu$m) 0.01 ($L = 2\,\mu$m)	V^{-1}
	n	subthreshold slope factor	$1 < n < 3$		–
	$I_{\mathrm{D0}} \propto K'$	subthreshold equivalent for K'	$15 - 20$		nA
ambient	$V_{\mathrm{th}} = \frac{kT}{q}$	thermal voltage equivalent	$V_{\mathrm{th}} \approx 26\,$mV for $T = 27\,^{\circ}$C		V

Table 1.1: Low frequency MOSFET model usually proposed for hand calculations: Top and middle part describe the gate and drain currents for N- and PMOS transistors, respectively. The bottom part summarizes the parameters employed in the above model equations. All values exemplified for the process parameters except for n and I_{D0} are typical for a 0.8 µm n-well CMOS process; they are taken from Allen and Holberg ([All02b], section 3.1).

the transistor is in the *cutoff* region and the resistance between source and drain is in the order of $10^{12}\,\Omega$. If, on the other hand, $V_{\mathrm{GS}} \geq V_T + nV_{\mathrm{th}}$, the substrate type is indeed inverted allowing for a conductive channel between the drain and source terminal. This condition is called *strong inversion*. For $0 < V_{\mathrm{GS}} < V_T + nV_{\mathrm{th}}$ the transistor is in an intermediate state called *weak inversion*, where V_T denotes the threshold voltage defined by (1.1f) and n the subthreshold slope factor (see Table 1.1).

Saturation Region. In *strong inversion*, the drain current I_D is expressed by (1.1d), which describes a parabola, if $V_{\mathrm{DS}} \leq V_{\mathrm{GS}} - V_T$ is satisfied. As $I_D = I_D(V_{\mathrm{DS}})$ is almost linear for small V_{DS}, this operation condition is denoted the *linear* or *ohmic* region. If, on the other, hand V_{DS} exceeds $V_{\mathrm{GS}} - V_T$, the drain current becomes almost independent of the drain source voltage, manifesting in (1.1e) ($\lambda \ll 1$). The $I_D - V_{\mathrm{DS}}$ characteristic of an NMOS and a PMOS transistor is displayed in Fig. 1.3(a) and 1.3(b), respectively for five different V_{GS} linearly spaced between 1 and 5 V. The characteristic family of $I_D -$



(a) NMOS (b) PMOS

Figure 1.3: $I_D - V_{\mathrm{DS}}$ curves for two different transistor models: BSIM3.3 and MOS15. V_S was tied to gnd and vdd for the simulated NMOS and PMOS transistor, respectively, whereas the gate voltage was varied between 1 and 5 V in equidistant steps. The nominal channel dimensions were set to $W = L = 2\,\mu\mathrm{m}$. Note that the curve for $V_{\mathrm{GS}} = 1\,\mathrm{V}$ almost coincides with the x-axis.

V_{DS} curves of transistors operated in *weak inversion* looks qualitatively similar to those depicted in Fig. 1.3. However, according to (1.1c), saturation occurs earlier at $V_{\mathrm{DS}} > 3V_{\mathrm{th}} \approx 80\,\mathrm{mV}$. Moreover, the absolute drain currents are much smaller in weak inversion and their dependence on the gate source voltage is exponential instead of the quadratic dependence found for saturation in strong inversion.

When operated in the linear region, the MOS transistor can be used as a voltage controlled resistor, albeit an imperfect one for larger V_{DS}. When operated in saturation, on the other hand, MOS transistors resemble a voltage controlled current source which are deteriorated only by a large resistor connected in parallel. This finite resistance is due the *channel length modulation* effect, which is caused by the extension of the depletion region around the pn junction of the drain diffusion. Accordingly, the parameter λ increases for shorter nominal channel lengths L.

Operation Regions. The influence of the different operation regions discussed above is not limited to the dc large signal model presented above. Dynamic as well as higher order effects also depend on the region the transistor is operated in. For instance, the different source, drain and gate capacitances depend on the degree of channel inversion as will be quantified below. Similarly, most of the important small signal properties as e.g. the transconductance $g_m = dI_D/dV_{\mathrm{DS}}$ largely depend on the transistor's region of operation. Even the device mismatch between equally designed transistors can

differ significantly between operation in weak and strong inversion, as threshold voltage variations will cause larger variations for smaller V_{GS}.

Model Parameters. Apart from the terminal voltages, the transistor behavior is determined by several parameters that are divided into three groups at the bottom of Table 1.1. The influence of the designer is limited to the choice of channel dimensions: First, the aspect ration W/L is taken as a multiplier for the transconductance at the respective operation point. Second, the length of the transistor channel influences the value of the channel length modulation parameter λ, such that the output resistance R_{DS} of a transistor in saturation can be varied. The different process parameters are defined by the fabrication technology. In effect, they can be traced back to the properties of the semiconductor materials, the doping profiles and the material and thickness of the gate oxide, as well as deviations between the drawn layout and the fabricated die, which are due to the fabrication process itself. These parameters are provided by the manufacturer. One of the characteristics of CMOS technology is the difference in the respective transconductances of PMOS and NMOS transistors due to the different mobilities of electrons and holes. Typically, the transconductance of NMOS devices exceeds that of their PMOS counterparts by a factor between 2 and 3. Although the die temperature enters the presented model only through the thermal voltage equivalent V_{th} in the equations describing the transistor in weak inversion, most if not all of the process parameters will exhibit a significant temperature dependence. As a result, the performance of a CMOS circuit will in general be sensitive to temperature variations.

1.1.3 Capacitances of CMOS Transistors

The model presented in the preceeding section concentrated on the transconductance properties of the channel induced by the potential on the gate terminal. However, the large signal model of a MOS transistor includes several other characteristics such as the diodes constituted by the source/drain bulk junctions, the ohmic resistance of the source and drain diffusion and various capacitors. While the leakage currents of the abovementioned pn junctions are considered in section 1.3, the capacitors inherent to MOS transistors that are relevant to this thesis are examined below. They are depicted in Fig. 1.4 for an NMOS transistor operated in the linear region.

The curve running below the source/drain diffusions and the channel marks the extension of the depletion regions of the respective pn junctions; the larger the reverse bias voltage, the larger the extension of the depletion regions. The boundaries of this depletion region form a capacitor. The value of the capacitors formed by the source/drain bulk diodes can – according to [Aus97c] – be calculated by

$$C_{SB} = \frac{WL_{diff}}{(1+V_S/\phi_b)^{M_j}} \cdot C_j + \frac{2(W+L_{diff})}{(1+V_S/\phi_b)^{M_{jsw}}} \cdot C_{jsw} \quad , \tag{1.3}$$

where W and L_{diff} are the width and length of the terminal diffusion and V_S denotes the actual terminal potential. While the first addend of (1.3) accounts for the area junction at the surface of the terminal diffusion, the second addend reflects the side wall junction at the circumference of the diffusion area. Typical values for the process parameters ϕ_b, M_j, M_{jsw} C_j, C_{jsw}, C_{ox} and C_{GSDO} are reported in [Lak94b] (section 1.8) and are presented in Table 1.2. For precise modeling, the channel bulk capacitance C_{BC} has to be added to C_{SB} and C_{DB} if the transistor is in strong inversion. It is split between drain and source in the same way the gate channel capacitance is split for the two cases of linear and saturation region, which is described below. However, as the effect is comparably small for the short transistors equation (1.3) is applied to, it is omitted for the hand calculations in Chapter 3 and consequently not included in (1.3).

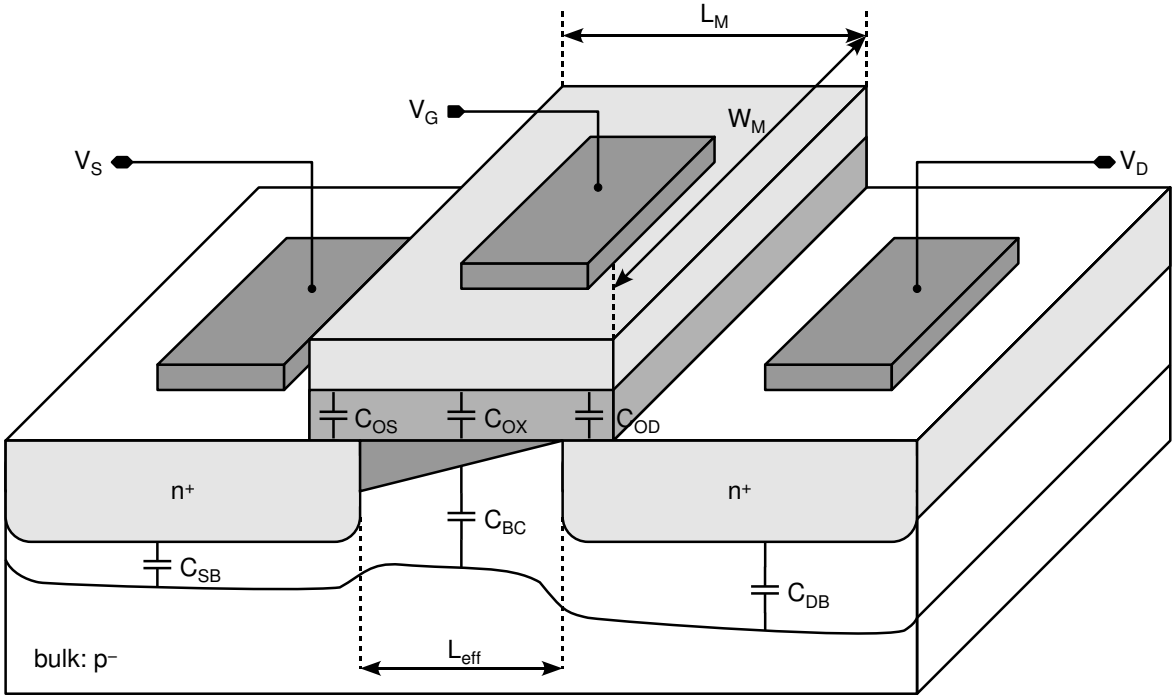


Figure 1.4: Capacitances in an n-channel MOST according to [Lak94b] (Section 1.8, pp 43). The transistor is assumed to be in the linear region.

Type	ϕ_b	M_j	M_{jsw}	C_j	C_{jsw}	C_{ox}	C_{GSDO}
NMOS	0.6 V	0.5	0.33	$0.45\,\frac{\text{fF}}{\mu\text{m}^2}$	$0.6\,\frac{\text{fF}}{\mu\text{m}}$	$0.68\,\frac{\text{fF}}{\mu\text{m}^2}$	$0.52\,\frac{\text{fF}}{\mu\text{m}}$
PMOS	0.6 V	0.5	0.33	$0.36\,\frac{\text{fF}}{\mu\text{m}^2}$	$0.6\,\frac{\text{fF}}{\mu\text{m}}$	$0.68\,\frac{\text{fF}}{\mu\text{m}^2}$	$0.4\,\frac{\text{fF}}{\mu\text{m}}$

Table 1.2: Typical values for the process parameters used in (1.3) (taken from [Lak94b]).

The gate-source capacitance can be calculated from the width W and length L of the transistor by:

$$C_{\text{GS}} = \begin{cases} WC_{\text{GSDO}} & \text{for} \quad V_{\text{GS}} \approx 0 & \text{transistor off} \\ WC_{\text{GSDO}} + \frac{1}{2}WLC_{\text{ox}} & \text{for} \quad V_{\text{DS}} \leq V_{\text{GS}} - V_T & \text{linear region} \\ WC_{\text{GSDO}} + \frac{2}{3}WLC_{\text{ox}} & \text{for} \quad V_{\text{DS}} > V_{\text{GS}} - V_T & \text{saturation region} \end{cases}, \tag{1.4}$$

whereas the gate-drain capacitance is described by

$$C_{\text{GD}} = \begin{cases} WC_{\text{GSDO}} & \text{for} \quad V_{\text{GS}} \approx 0 & \text{transistor off} \\ WC_{\text{GSDO}} + \frac{1}{2}WLC_{\text{ox}} & \text{for} \quad V_{\text{DS}} \leq V_{\text{GS}} - V_T & \text{linear region} \\ WC_{\text{GSDO}} & \text{for} \quad V_{\text{DS}} > V_{\text{GS}} - V_T & \text{saturation region} \end{cases}. \tag{1.5}$$

Here, the capacitors caused by the gate source and gate drain overlap, denoted as C_{OS} and C_{OD}, respectively are calculated from the one-dimensional capacitance density C_{GSDO} specified by the manufacturer. In the linear region, the gate channel capacitance WLC_{ox} is split equally between the source and the drain terminal. The difference between C_{GS} and C_{DS} in the saturation region is caused by the fact that in this case, different from Fig. 1.4, the channel does not reach the drain terminal. Hence,

drain and channel do not possess a direct connection and the gate-channel capacitance C_{GC} is not connected to the drain. Typical values for the process parameters C_{ox} and C_{GSDO} are again taken from [Lak94b] and are included in Table 1.2.

1.2 Linear Devices in CMOS Technology

Many interesting analog circuits, as for example a large variety of operational amplifier circuits or filters , rely on passive components featuring good linearity and matching properties. Unfortunately, for some ranges of desired component parameter values, such linear devices are not readily available in CMOS technology. In case of inductors, their performance is severely hampered by parasitic effects and limited to frequencies in the GHz range, where planar spiral inductors have been successfully used in filter applications (Abidi [Abi96] and Kuhn et al. [Kuh03]). As the frequencies of interest to the circuits discussed in this thesis will not exceed 100 MHz, inductors are not an issue here. However, a brief sketch of possible implementations of capacitors and resistors will be given below as a background information for the following design decision concerning the implementation of the FPTA chip presented in chapter 3. The *programmable transistor array* serving as the substrate for the test of candidate circuits bred by the evolutionary algorithm is deliberately restricted to CMOS transistors, that is, it does not provide any passive components.

1.2.1 Resistors in CMOS technology

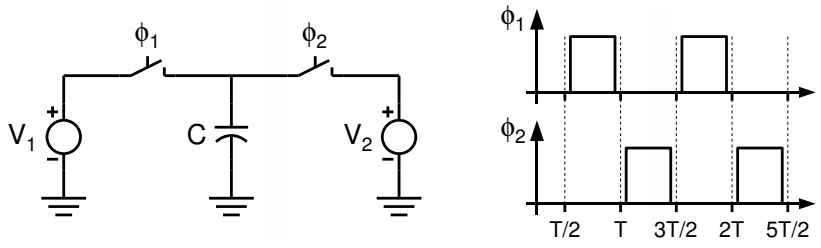
Passive Resistors. Passive monolithic resistors are usually implemented as strips of polysilicon or diffusion. Usually, polysilicon is preferable to diffusion as the latter one exhibits undesired nonlinearities. Yet, the sheet resistance of polysilicon in standard CMOS processes is quite low and resistors exceeding a few kΩ will become bulky if not infeasible. Fabrication technologies specialized for analog circuitry or four-transistor SRAM[4] cells ([Gei90a] section 9.10) offer an additional layer of high-resistive polysilicon, with which resistances up to MΩ can be realized spending only moderate amounts of silicon area. However, as these technologies are highly specialized, circuit designs that can do without this additional layer are preferable, as they can be implemented in cheaper and/or more advanced fabrication technologies.

Active Resistors. Since compact large passive resistors may be unavailable, they must be replaced by active resistors. In order to attain the desired linear $I-V$ curve from a CMOS transistor, it has to be operated in the linear region. Thus, most active resistor implementations achieve linearity by clever biasing of one or more transistors. The circuit examples presented e.g. in ([Gei90a] section 5.2) reveal that relatively good linearity over a large range of input voltages is easier to attain for differential circuits. An example for active resistors capable of very high resistance values, albeit a small range of differential voltages, is found in [Mea91]. For small differential voltages, CMOS switches – discussed in section 1.3 – can also serve as resistors, as can be observed from Fig. 3.5(a). Since large parts of the configurability of the FPTA are realized by means of switches, this may be exploited by an evolutionary algorithm if the task necessitates the use of resistors.

Switched Capacitors. In a typical CMOS process, capacitors offer the best relative precision, provided that they are laid out with sufficient care. Thus, for low and moderate frequency signals, resistors can be realized as switched capacitors, which offer a high degree of linearity as well as precise matching with capacitors allowing to form precise *RC* constants. The principle is depicted in Fig. 1.5;

[4] Static Random Access Memory

a detailed discussion can e.g. be found in ([All02b] chapter 9) or ([Gei90a] section 5.2). Supposed



(a) switched capacitor resistor (b) nonoverlapping clock

Figure 1.5: (a) switched capacitor realization of a resistor. (a) required nonoverlapping clock signals.

that V_1 and V_2 are varying much slower than the clock frequency $f_{\text{clk}} = 1/T$, the resistance of the switched capacitor circuit illustrated in 1.5(a) can be calculated as

$$R_{\text{SC}} = \frac{T}{C} = \frac{1}{C f_{\text{clk}}}. \tag{1.6}$$

In order to avoid any direct connection between V_1 and V_2, the respective clock signals ϕ_1 and ϕ_2 are not to overlap during their on phase. Such a nonoverlapping clocking scheme is shown in Fig. 1.5(b); a possible realization is discussed in ([Gei90a], section 9.5). Switched capacitor circuits are widely used in signal processing applications, for instance in temporal ([All02b], chapter 9) or spatial [Sch99] filtering. However, without supplying an external pair of nonoverlapping clocks, switched capacitor circuits are extremely unlikely to emerge from the kind of artificial evolution experiments discussed in this thesis.

1.2.2 Capacitors in CMOS

In fabrication technologies specified for analog designs monolithic capacitors can be conveniently realized as sandwiches of two polysilicon layers. Capacitor values up to few or even tens of pF can be implemented with moderate area consumption. If an additional second polysilicon layer is not available, there are two alternatives: Either sandwich capacitors using two metal and one polysilicon layer or MOS capacitors. The former structures are linear and can be used for the entire voltage range. Yet, they provide only a small fraction of the capacitance per square micrometer achieved by poly-poly capacitors. MOS capacitors exploit the gate source/drain capacitance described by (1.4) as illustrated in Fig. 1.6 for the case of an NMOS transistor. Their capacitance density usually

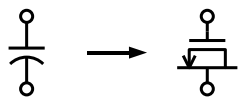


Figure 1.6: Capacitor symbol and its realization as an NMOS transistor.

exceeds that of poly-poly capacitors, due to the extremely thin gate oxide necessary to obtain high transconductance parameters. The major disadvantage of MOS capacitors is that the voltage drop across the capacitor must be larger than V_T (which also depends on V_B) to attain the desired gate channel capacitance. Depending on the signal level, this may not be too much of a problem if one end of the capacitor can be connected to a fixed voltage like vdd or gnd. In case of floating capacitors however, an appropriate biasing scheme must be able to ensure the required voltage drop across the capacitor

1.3 CMOS Switches

One, if not the salient feature of the FPTA chip proposed in chapter 3 is its ability to host a large variety of circuits whose topology is almost freely selectable. This kind of configurability is achieved by a large number of switches. In addition, the analog signals fed into the array of programmable transistors as well as those signals read from the array are processed in sample and hold stages. In this vein, CMOS switches are of utmost importance to the project proposed in this thesis. The simplest implementation of a CMOS switch would be a simple pass transistor. In case of an NMOS transistor, the switch is closed, if the gate is tied to vdd, and open, if the gate voltage is set to the gnd potential. The major drawback of pass transistors is their limited common mode range, which is due to the requirement $V_{GS} > V_T$. This limitation can be overcome by connecting an N- and a PMOS transistor in parallel and using signals of opposite polarities to control their respective gate voltages. The realization of a switch by means of a transmission gate is depicted in the upper part of Fig. 1.7.

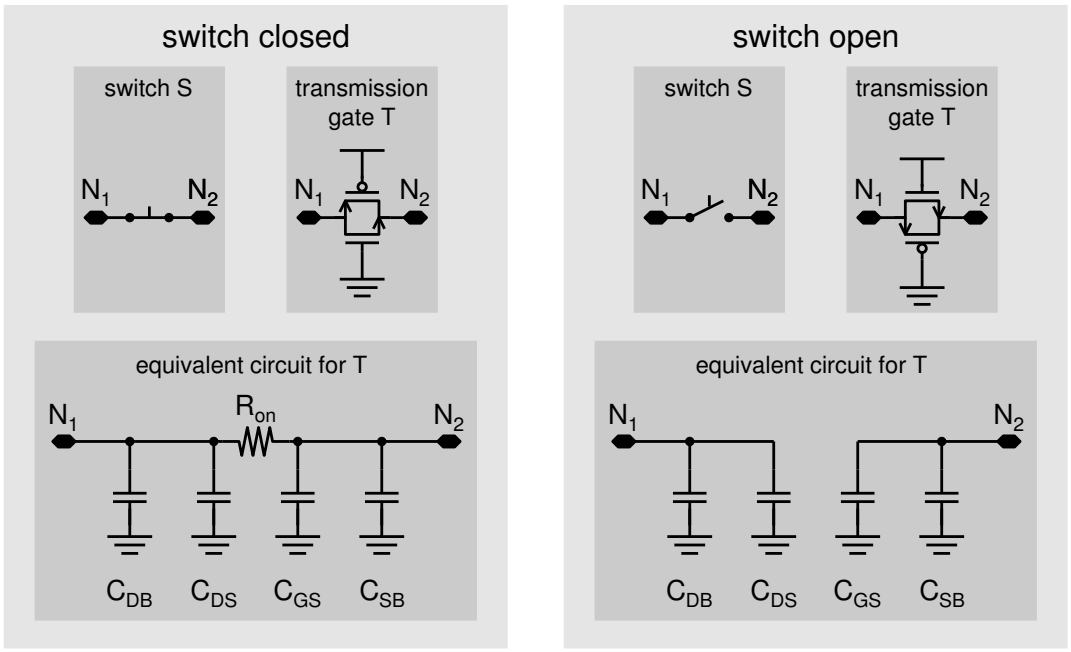


Figure 1.7: Top: Realization of a closed and an open switch as a transmission gate. **Bottom:** Small signal equivalent of the closed (left) and open (right) transmission gate.

1.3.1 Switch Parasitics

Though MOS switches are considered fairly good approximations of ideal switches, their nonidealities do matter in the context of this thesis: For one, the transistor possesses a finite on-resistance R_{on}. For the other, its two nodes N_1 and N_2 entail some parasitic capacitances. Both effects are included in the small signal model shown in the bottom part of Fig. 1.7. Since for a transmission gate, $|V_{GS}|$ is either zero, or equal to vdd, its transistors can be assumed to be in the off-state in the former case and to operate in the linear region in the latter case. For an NMOS transistor, the on-resistance can thus be calculated from (1.1d) to be:

$$R_{on} = R_{DS} = \left(\frac{\partial I_D}{\partial V_{DS}}\right)^{-1} = \left(K'\frac{W}{L}\left(vdd - V_S - V_T - V_{DS}\right)\right)^{-1}, \tag{1.7}$$

if the channel length modulation term is omitted. The simulation results for the on-resistance of two different transmission gates used in the FPTA implementation are shown in Fig. 3.5.

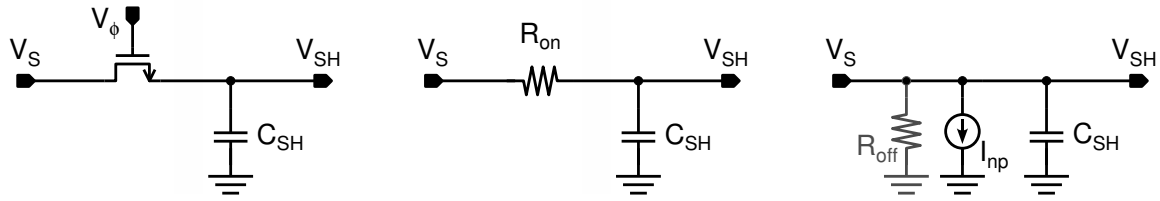
The parasitic capacitances entailed in the implementation of the transmission gate can be analyzed in terms of the parasitic transistor capacitances discussed in section 1.1.3: According to (1.3), (1.4) and (1.5) the capacitance at both nodes N_1 and N_2 depends solely on the respective node voltage V_{N_1} and V_{N_2} and can therefore be expressed by

$$C_{N_x} = C_{\mathrm{SB,P}}(W_\mathrm{P}) + C_{\mathrm{GS,P}}(W_\mathrm{P}) + C_{\mathrm{SB,N}}(W_\mathrm{N}) + C_{\mathrm{GS,N}}(W_\mathrm{N}) \quad . \tag{1.8}$$

Here, it is assumed that the length of the terminal diffusion L_{diff} is equal for both transistors types. The source-bulk and gate-source capacities C_{SB} and C_{GS} have to be calculated according to (1.3) and (1.4). The additional subscripts 'N' and 'P' indicate whether the process parameters for P- or NMOS transistors have to be chosen.

1.3.2 Dynamic Operation: Sample and Hold Circuits

So far, the discussion has concentrated on the *static* properties of CMOS switches in that the switch was either open *or* closed during the actual operation of the circuit at hand. However, a large variety of applications as e.g. switched current, switched capacitor or DRAM[5] circuits require their *dynamic* operation with respect to their state. Within this thesis, the paramount application that requires dynamic switching is the temporary storage of analog voltages on a sample and hold capacitor. This situation is depicted in Fig. 1.8(a). Usually, voltage V_{SH} is buffered by a suited operational amplifier



(a) S/H pass transistor (b) On-state model (c) Off-state model

Figure 1.8: (a) NMOS pass transistor used to sample voltage V_S on C_{SH}. (b) simple model used for the calculation of the time necessary to charge C_{SH}. (c) equivalent circuit if the NMOS transistor in (a) is turned off.

for further processing, which is omitted here for simplicity. Though in general transmission gates are often preferred for sample and hold units, the analysis can be simplified to an NMOS pass transistor without loss of generality.

1.3.2.1 Sample Phase

During the *sample* period, the S/H circuit of Fig. 1.8(a) can be approximated by the equivalent circuit shown in Fig. 1.8(b). The influence of any parasitic capacitances inherent to the switch as well as the switching operation itself are omitted in the analysis for simplicity. After the switch is closed, the capacitor C_{SH} is charged and V_{SH} converges exponentially towards the signal V_S at the input, which is described by

$$V_{\mathrm{SH}} = V_S \cdot \left(1 - e^{-t/R_{\mathrm{on}}C_{\mathrm{SH}}}\right) \quad . \tag{1.9}$$

[5]Dynamic Random Access Memory

This equation can be solved for the settling time necessary for V_{SH} to reach V_S with the desired accuracy Acc or resolution Res

$$T_{\text{settle}} = -R_{\text{on}} C_{\text{SH}} \ln(\text{Acc}) = R_{\text{on}} C_{\text{SH}} \ln(2) \text{Res [bit]} \ , \tag{1.10}$$

where the resolution Res is measured in bit. If, for instance, V_{SH} shall possess a resolution of 16 bit over the full power supply range, the switch must be closed for at least $T_{\text{settle}} = 11.1 R_{\text{on}} C_{\text{SH}}$ in the worst case in which V_{SH} and V_S differ by the full power supply voltage.

1.3.2.2 Hold Phase

As CMOS switches feature a relatively high off-resistance R_{off}, the voltage stored on C_{SH} will decay slowly and can be utilized for some time. The exact time depends on design, fabrication technology and the required precision for V_{SH}. In principle, leakage can occur at the input gates of the subsequent buffer and at the pn junction of the source/drain terminal of the switch transistor. Moreover, some current may still flow through the switch transistor according to [All02c]. Yet, since the CMOS process used for the proposed FPTA chip is not in the deep-submicron regime, the leakage through the CMOS gates and the transistor in the off-state are neglected in the following analysis, although they are included in the off-resistance R_{off} of the equivalent circuit depicted in Fig. 1.8(c). The leakage current through the pn junction of the source/drain terminal of the pass transistor can be calculated from

$$I_{\text{np}} = W \cdot L \cdot J_{\text{AN}} + 2(W + L) \cdot J_{\text{PN}} \ , \tag{1.11}$$

where J_{AN} describes the current density per drawn area and J_{PN} describes the current density per drawn perimeter of the reverse-biased pn junction in an NMOS source/drain terminal. In fact, the current through a reverse-biased diode may also exhibit voltage dependence, which can be considered by an adequate contribution to R_{off} and causes V_{SH} to decay exponentially (cf. e.g. [Lan98], chapter 2). Yet, as one is only interested in a worst-case estimate for the maximum allowable hold time, the pn junction leakage current can be approximated by its constant value at the maximum reverse bias voltage. The maximum hold time for a given resolution Res can thus be evaluated by

$$T_{\text{hold}} = \frac{C_{\text{SH}}}{I_{\text{np}}} \cdot \text{vdd} \cdot 2^{-\text{Res [bit]}} \ . \tag{1.12}$$

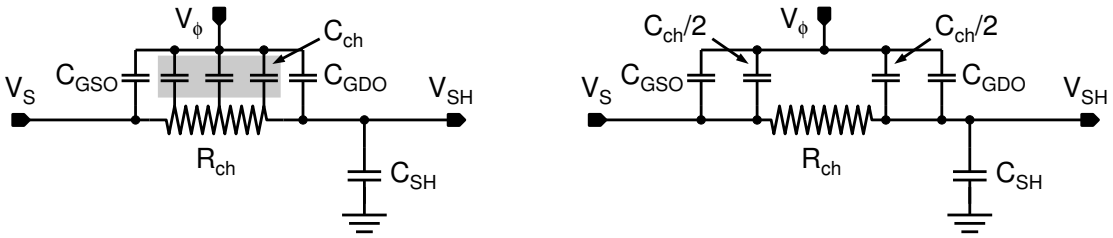
In case of a transmission gate, the proposed model has to be extended by an additional current source between vdd and node V_{SH} that accounts for the reverse biased pn junction of the PMOS drain/source transistor. The resulting leakage currents will partially cancel out.

1.3.2.3 Charge Injection

Unfortunately, the process of opening the transistor switch may severely deteriorate the voltage to be stored on C_{SH} due to capacitive feedthrough of the switched gate voltage V_ϕ and V_{SH}. The capacitive coupling between gate and drain/source through the gate-drain/source overlap capacitances $C_{\text{GS/DO}}$ and the gate-channel capacitance C_{ch} is depicted in Fig. 1.9(a). The latter capacitance is actually spread over the whole length of the channel. However, Sheu and Hu [She84] show that the distributed model can be replaced by the lumped model of Fig. 1.9(b) provided that $\frac{dV_\phi}{dt} \ll \frac{dV_{SH}}{dt}$, which is certainly fulfilled for the sampled voltage to be meaningful. It is worth noting that it is this lumped model on which (1.4) and (1.5) are based on.

To open the switch, V_ϕ changes from V_H to V_L. Thereby, the channel capacitance breaks down, when the transistor turns off, that is for

$$V_{\text{GS}} < V_{\text{HT}} \equiv V_H - V_S - V_T \ . \tag{1.13}$$



(a) distributed model (b) lumped model

Figure 1.9: (a) Distributed switch model for a transistor switch. (b) Lumped model of (a). Figures are similar to those presented in [All02c].

As long as V_ϕ is still larger than V_{HT} during the switching period, the channel is still conducting, such that a part of the charge injected onto node V_{SH} can partly be equalized through the channel resistance R_{ch}. The exact amount of charge deposited on node $\mathrm{V_{SH}}$ thus depends on the relation of the RC-constant defined by R_{ch} and $C_{\mathrm{ch}} + C_{\mathrm{GDO}}$ and the time during which V_ϕ changes from V_H to V_L.

Again, assuming $\frac{dV_\phi}{dt} \ll \frac{dV_{\mathrm{SH}}}{dt}$ Sheu and Hu [She84] have derived an analytical expression for the error voltage V_{err} injected on node V_{SH} for a linear transition of V_ϕ described by

$$V_\phi = V_H - \alpha t \quad . \tag{1.14}$$

To simplify matters, they distinguish two extreme cases of a *slow* and a *fast* transition which are determined by the critical slope

$$\alpha_{\mathrm{crit}} = \frac{\beta V_{\mathrm{HT}}^2}{2C_{\mathrm{L}}} \quad , \tag{1.15}$$

where $\alpha \ll \alpha_{\mathrm{crit}}$ refers to the *slow* and $\alpha \gg \alpha_{\mathrm{crit}}$ to the *fast* transition. In these cases, the error voltage V_{err} induced on node V_{SH} can be calculated by

$$V_{\mathrm{err}} = \begin{cases} \frac{WC_{\mathrm{GSDO}}+C_{\mathrm{ch}}/2}{C_{\mathrm{SH}}} \sqrt{\frac{\pi\alpha C_{\mathrm{SH}}}{2\beta}} + \frac{WC_{\mathrm{GSDO}}}{C_{\mathrm{SH}}}(V_S + V_T - V_L) & \text{if} \quad \frac{\beta V_{\mathrm{HT}}^2}{2C_{\mathrm{L}}} \gg \alpha \\ \frac{WC_{\mathrm{GSDO}}+C_{\mathrm{ch}}/2}{C_{\mathrm{SH}}} \left(V_{\mathrm{HT}} - \frac{\beta V_{\mathrm{HT}}^3}{6\alpha C_{\mathrm{SH}}}\right) + \frac{WC_{\mathrm{GSDO}}}{C_{\mathrm{SH}}}(V_S + V_T - V_L) & \text{if} \quad \frac{\beta V_{\mathrm{HT}}^2}{2C_{\mathrm{L}}} \ll \alpha \end{cases} . \tag{1.16}$$

If the slew rate of the clock signal was infinitely large, or the switching time infinitely short, no charge could be exchanged between the drain and source terminals of the pass transistor during the switching phase. Therefore, (1.16) simplifies to

$$V_{\mathrm{err},0} = \frac{C_{\mathrm{ch}}}{2C_{\mathrm{SH}}} \cdot V_{\mathrm{HT}} + \frac{WC_{\mathrm{GSDO}}}{C_{\mathrm{SH}}} \cdot (V_H - V_L) \quad . \tag{1.17}$$

1.4 Simulation and Verification

Optimization procedures in general and evolutionary algorithms in particular need a performance criterion determining the quality of the solution being optimized. Thus, it seems instructive to sketch the way in which the quality of analog CMOS designs is verified prior to their production. The peculiarity of the approach to analog design automation pursued in this thesis is that the candidate circuits are tested in hardware instead of being simulated. To fully appreciate the difference between those two approaches and to develop a feeling for the quality of different test criteria used in related work, a little insight in the simulation and verification of analog CMOS circuits may be beneficial.

Different simulators can be used for the simulation of a CMOS circuits. In principle, they should all be able to deal with the typical transistor models and support the analysis types described below,

but will probably differ in the actual implementation. This may manifest in different convergence properties, numeric precision or computation speed, rather than produce largely differing results. SPICE [Qua94] is the prototype of circuit simulators and probably the best known one. The information provided below is based on the SPICE simulator as this is the simulator the treatments of Allen and Holberg [All02b] and Geiger et al. [Gei90a] are based on. The simulations presented within this thesis however, rely on a simulator called SPECTRE [Cadb]. Apart from the design itself, the simulator needs models for all the device types present in the design and the process parameters characterizing the target technology (which must comply with the format and requirements of the device models).

1.4.1 MOSFET Modeling

In the course of improving fabrication technologies several transistor models of different levels of accuracy and complexity have been developed. Especially the trend to decreasing transistor lengths rendered new models accounting for the new short channel effects necessary. The dc model characterizing the I-V curves of a MOS transistor that are summarized in Fig. 1.1 are only good for hand calculations. For the exact simulation of submicron processes they are not sufficient. Except for the equations modeling the weak inversion, the model is equivalent to the dc part of the SPICE level 1 model, which is also referred to as Shichman-Hodges model. In the SPICE level 1 model, the behavior in weak inversion is dealt with as the cutoff region, that is by $I_D = 0$.

More accurate simulation of MOS transistors can be achieved by SPICE level 2 and 3 models, which ,for instance, provide more sophisticated means of calculating the transition between the linear and the saturation region and a model for MOS transistors operated in weak inversion. Moreover, the influence of short channel geometries is modeled more accurately. The high frequency model, which adds the capacitors described in section 1.1.3 to the dc model, refines some of the equations presented there. In particular, the calculation of capacitors connected to the channel are more elaborate, as they are essentially distributed capacitances that require some type of integration or approximation thereof. While the level 1 and level 2 models are derived from device physics, SPICE level 3 is a semi-empirical model. In order to improve the modeling of transistors in the submicron regime, a series of BSIM models have been developed. The BSIM3.3 has been the industry standard during recent years [All02b]. It addresses several effects occurring in deep submicron operation of MOSFETs and features 40 parameters just for the dc model. If not explicitely denoted otherwise, the simulations presented in this thesis are based on the BSIM3.3 model. The challenges of modeling most recent circuit technologies with feature sizes around 100 nm are met by the latest BSIM4.2 model.

The $I_D - V_{\mathrm{DS}}$ characteristics depicted in Fig. 1.3 serve as an example for the deviations between different transistor models: The dc characteristics of both transistor types are simulated once for the BSIM3.3 model and once using the MOS15 model. In both cases, those process parameters are used that belong to the fabrication technology in which the FPTA chip was produced in. The MOS15 model is an improved version of the SPICE level 2 model adapted by the manufacturer. Although the chosen channel length of $L = 2\,\mu\mathrm{m}$ is quite large, the resulting $I_D - V_{\mathrm{DS}}$ curves differ considerably for the two device models.

1.4.2 Analysis Types

Typically, a circuit simulator offers different analysis types allowing the designer to test different figures of merit for the design at hand. The most elementary types are dc, ac and transient analysis, which will be summarized below:

DC Analysis. For a dc analysis any potential time dependency of the circuit under simulation is eliminated by shortening all inductors and opening all capacitors. The resulting circuit is then simulated using the dc model chosen by the user. If successful, the simulation converges to an equilibrium solution yielding a set of node voltages and currents. Usually, circuit simulators allow to sweep either an input voltage, a design variable or the circuit temperature. To obtain a family of curves, the dc sweep has to be re-run for the desired set of parameters.

AC Analysis. At the beginning of an ac analysis the dc operation points of all circuit devices must be established by means of a dc analysis. Subsequently, the small signal model of the circuit is calculated by linearizing all of its devices at their respective operating points. The resulting system of linear equations is then typically solved for a set of different frequencies specified by the user. Strictly speaking, the results of an ac analysis are only valid for arbitrarily small signals and at the operation point defined in the respective simulation.

Transient Analysis. A transient analysis also starts by establishing a dc solution for the circuit under simulation. The dc solution is used to define the initial conditions for time $t = 0$. Subsequently, the analysis proceeds by numerically solving the system of differential equations for discrete time steps. The size of the time steps is controlled by a default resolution specified by the user and a mechanism that reduces the time steps if necessary. Here, necessary is defined by a threshold for the size of the relative and absolute changes of the involved node voltages and currents. The respective threshold values can also be changed by the user. In comparison, the transient analysis is the analysis type that is most closely related to an actual hardware test, because it includes all aspects of the device models and solves the full set of differential equations. On the other hand, solving the full set of differential equations also requires the largest amount of computation time.

1.4.3 Influence of Temperature and Device Mismatch

Temperature. The electrical properties of semiconductors depend on temperature. Consequently, the behavior of analog CMOS circuits is also subject to temperature variations. Therefore, the designer must ensure that the circuit at hand meets the desired specifications over the temperature range of interest. In simulations, this temperature dependency is modeled by defining a temperature dependence for the affected process parameters.

Process Parameter Variations. Due to tolerances and imperfections in the production process, the exact values of the process parameters cannot be guaranteed. Instead, the manufacturer usually defines upper and lower bounds and *typical mean* values for each process parameter (or at least those the device models are most sensitive to). To keep the subspace of allowed process parameters tractable, the parameter variations are summarized in a small set of *worst case* parameters. For example, differences in the relative doping densities may result in a change of the ratio of the transconductance parameters of n- and p-channel transistors, which is accounted for by a *worst case one* or *worst case zero*. This ratio of transconductances determines the set point of inverter-like circuits. To ensure that the circuit will meet the specifications for the full subspace of allowed process parameters, the designer usually verifies the design for the typical mean as well as for all worst case parameter sets, which is referred to as a *corner analysis*.

Device Mismatch. The imperfections inherent to the fabrication process also cause equally designed devices located on the same die to behave slightly different, which is referred to as *device*

mismatch or *device to device variations*. As a consequence, the designer has to verify that the circuit at hand still fulfills the specifications when the expected device variations are taken into account. This can e.g. be done by a sensitivity analysis, which calculates the influence of all sorts of circuit parameter variations on some output voltage or current [Qua94], or by means of a *Monte Carlo analysis*. The latter one can model the distribution of circuit behaviors based on a statistical model of the device to device and process parameter variations. As a consequence, a suited Monte Carlo analysis allows to predict the yield of circuits meeting the desired specifications.

1.4.4 Comment on Circuit Simulations in Evolutionary Algorithms

The above discussion revealed that proper circuit verification is quite complex and involves a considerable amount of computational effort. On top of that, the designer must ensure that the circuits are verified to work under all possible conditions, which also requires to use the correct load impedances at the output of the circuit under test. Despite all the effort and faith in the simulation, the authors of text books on CMOS design advise the designer to develop a good understanding of the circuit and to refrain from excessive simulations ([All02b], [Gei90a]). One of their possible reasons may be that it is impossible to cover all operating conditions of the circuit at hand with arbitrary resolution. As indicated in case of the ac analysis, the simulated results may only be valid in a very limited range of parameters. In case of a human designer, the circuit usually relies on well known design principles and its behavior can be roughly estimated by hand calculations and arguments. Therefore, the behavior of human designed circuits tends to extend smoothly under changing operation conditions. For circuits emerging from an evolutionary algorithm, this cannot be expected. The algorithm simply tries to find a solution that fulfills exactly those goals specified in its target function. As a consequence, the simulation of possibly unconventional circuits requires even more care than its human-designed counterpart.

1.5 Design Flow

Unlike their digital counterparts, for the design of which sophisticated CAD[6] tools exist[7], analog circuits and digital circuits whose analog behavior must be specified usually have to be designed by hand[8]. That is, the designer has to decide upon the topological position as well as the channel dimensions of each and every transistor belonging to the circuit. Even worse, subsequent to the electrical design of the circuit, it must be transferred into a suitable layout, which again requires the designer to place and route each single transistor. In between, the results of the electrical and later on the physical implementation have to be verified by simulations.

A viable design flow for an analog or mixed signal system is illustrated in Fig. 1.10. The system is assumed to comprise two levels of hierarchy in that the given target specification of the desired system can be broken down into a system of analog subcircuits that are dealt with on the transistor level. Faced with such a situation, the designer may choose to simulate the resulting architecture on an abstract level after actually dividing the system into smaller building blocks. This can be e.g. accomplished by using an AHDL[9] description to model the subcircuits. As a product of this possibly recurrent loop, the necessary specifications for all subcircuits emerge.

[6] Computer-Aided Design

[7] A sketch of a viable design flow used in digital design is given in section 4.3.

[8] It has only been very recently that some of the attempts to automate the analog design process did mature into available products that support the designer.

[9] Analog Hardware Description Language

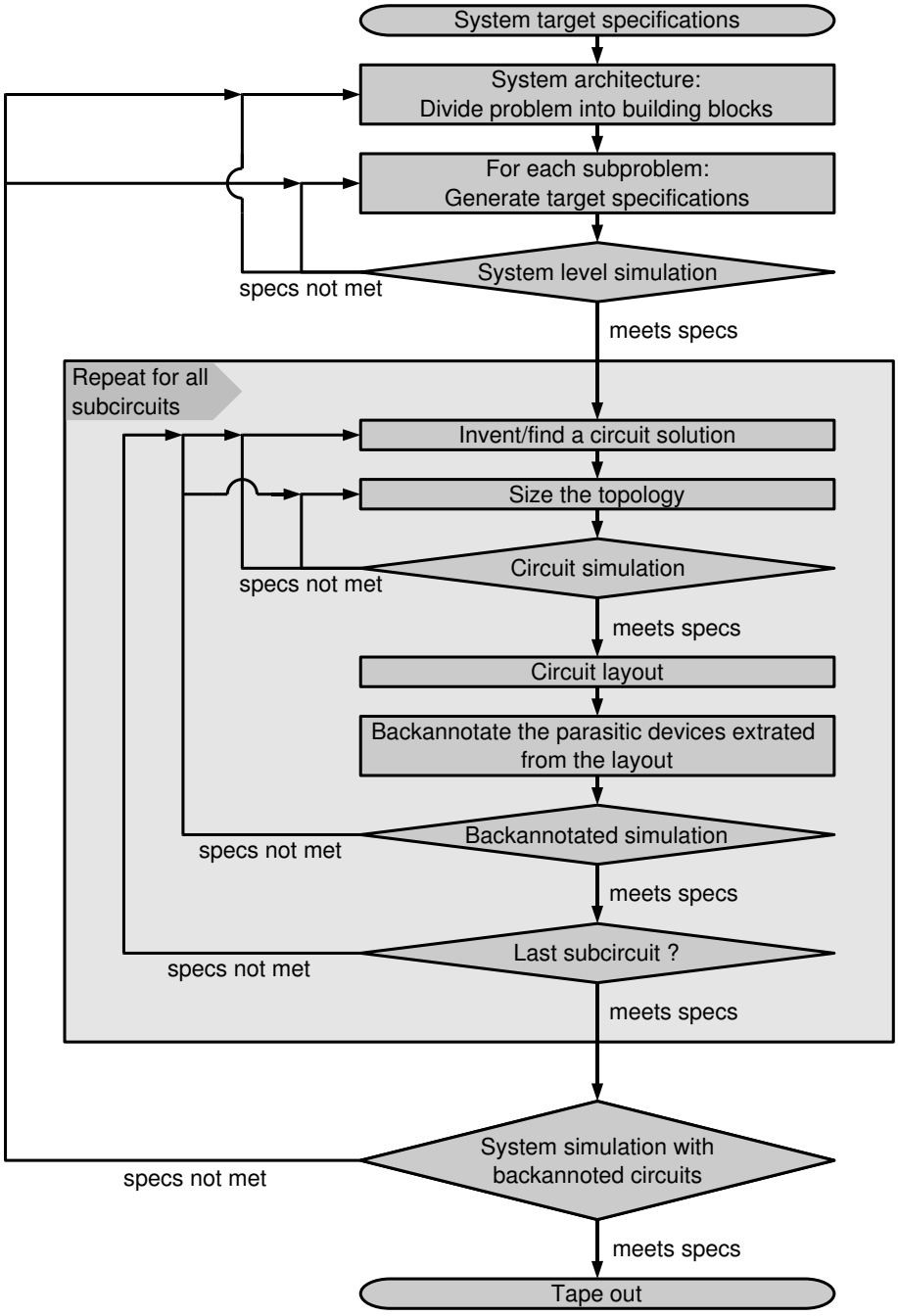


Figure 1.10: Design flow for an analog system featuring two levels of hierarchy.

Now, each of these subcircuits has to be designed according to the specifications determined above. Typically, the designer starts with picking a suited topology from a circuit library or from a text book. If no existing topology is available, a new one must either be invented or an existing must be adapted to the task. In the second step, the transistor dimensions of the chosen topology must be determined so that all of the target specifications can be met. This again recurrent procedure involves the different kinds of transistor level simulations described in the previous section. If the designer cannot satisfy the target specifications, he must either resort to a different topology and run through the same sizing procedure, or try to rearrange the set of specifications on the system level (this is omitted in Fig. 1.10 for clarity).

If the target specifications are eventually met, the designer has to do the layout of the circuit. As the layout adds parasitic devices, that is, capacitors formed by adjacent metal, polysilicon or diffusion strips, capacitors formed by layers that are in vertical proximity or resistors caused by finite sheet resistances, these devices must be extracted from the layout and added to the electrical circuit. The resulting circuit must again be verified. If it fails to meet the specifications the different methods of redesign described above apply. Finally, if all subcircuits have been designed up to their specifications, an overall transistor level simulation including all of the parasitics extracted from the top level layout must verify that the entire system satisfies the required specifications.

Chapter 2

Evolutionary Algorithms in a Nutshell

Birth, Copulation, and Death.
That's all the facts when you come
to brass tacks.

Sweeney Agonistes
T. S. ELLIOT

Most of the work in the field of hardware evolution as well as the evolution system described in this thesis rely on evolutionary algorithms (EAs) as an optimization engine or even as a source of invention. This chapter explains the general idea behind these algorithms based on its natural inspiration. The description continues with the main constituents of an EA and provides an overview over its different incarnations. The application of EAs is motivated by briefly discussing their position in the field of optimization as well as their abilities and limitations. The chapter concludes with a small selection of possible extension and refinements of EAs that may further enhance their performance in the proposed project.

Evolutionary algorithms (EAs) are widely used as general purpose automated problem solvers in science as well as in industry [Bug03] and are an active area of research[1] [Lan02],[Deb04]. The list of EA applications is quite long and diverse. It comprises different problem types, such as optimization of engineering designs, scheduling problems, timetabling problems, search, control, system identification or training of neural networks. Apart from the fascinating analogy to natural evolution, EC draws much of its popularity from its generality, in the sense that it can achieve good results without an exact understanding of the problem at hand. Although the roots of EC actually date back to the late 1950s [Fog98], [Bäc97], and although three of today's four founding dialects emerged separately in the 1960s, the field has received most attention during the last 15 years.

[1]The research area dealing with EA is actually referred to as evolutionary computation (EC).

2.1 Biological Inspiration

In 1859, Charles Darwin explained the enormous biological diversity found on our planet by a theory based on the principles of mutation and natural selection [Dar59]. To date, the core of this theory is still considered *the* scientific explanation for the evolution of different life forms, backed up by more evidence than ever [Got94]. The inventiveness of the natural evolution process is astounding, in the generated quantity of an estimated 2 million different species known to currently exist (not to mention those that have vanished again already) [Sto01e], as well as in the quality of the produced solutions. Accordingly, it comes as no surprise, that engineers attempt to imitate the principles of natural evolution to find new solutions to their problems. As such, it seems only natural to start with a brief description of the original, biological evolution process. More detailed information concerning the subject matter can e.g. be found in [Eib03a], [Got94], [Wei02], [Hes], [Czi92].

2.1.1 Darwinian Evolution

From a macroscopic point of view, the theory of evolutions can be summarized as follows: Typically, the members of a given species reproduce at a rate that causes their number to grow exponentially with time. Yet, at some point a population of such individuals will experience the limited resources provided by their environment. In this situation those individuals that are better adapted to their environment will be more likely to survive than the less adapted ones. This mechanism is termed natural selection or – in the words of Darwin himself – the *survival of the fittest*. The fitness of an individual is determined by its physiological, physical and behavioral features, in short: its phenotype. The phenotypical traits as well as the plan how they are developed is provided by the genetic information – the genotype – inherited from progenitors to their progeny. Since those individuals that are better adapted to the given environment will also be more likely to produce offspring, they are more likely to pass their genes to the next generation. A change that increases the fitness to survive and reproduce will be stored in the genetic code of of that fitter individual's offspring . The necessary changes are induced by small variations of the genetic information that occur during the reproduction process, which are denoted as mutations. They counterbalance the converging force created by the combination of selection and recombination. In summary, the fitness evaluation performed by the environment favors to propagate those changes in the genetic code to future generations that are either beneficial or neutral with respect to the rest of the population.

So far, the discussion referred to a population of competing individuals. Biologically, such a population is limited to members of one species, that is, creatures that members of the population can reproduce with. It is further limited by the geographical region, the members of one population have to coexist and therefore have to compete in. A new species can emerge if some successful mutation provides an advantage over the rest of an, for example, geographically isolated population. Given enough time, this population can be evolved to the point, where it cannot reproduce with members of its original species. It may be the case, that the new species adapted to one particular ecological niche, which makes it impossible for members of its old species to compete and hence dominate them. Another important aspect is the fact that the environment evaluating the respective fitness of a species is defined to a large extent by other evolving organisms that may, for example, be predators or serve as food. Hence, the environment is also constantly changing requiring an endless process of adaptations. The fact, that many species evolve together influencing the development of each other is described as coevolution.

2.1.2 Genetic Principles

The genetic information is typically[2] stored in the DNA[3]. DNA, in principle, consists of a long chain of base pairs (approximately 3 billions in case of the human genome [Got94]). The genetic information is stored by means of the alphabet comprised by the four base pairs G-C, T-A, C-G and A-T, where A denotes adenine, C cytosine, G guanine and T thymine. The entire genetic information is stored as DNA in the nucleus of each cell of one organism. It is called the genome, which is usually divided into several chromosomes (23 for human beings). Each chromosome comprises a large number of genes (e.g. in the order of 4,000 for E-coli bacteria [Got94] and between 20,000 and 30,000 for human beings [Nat04]). Opposite to chromosomes, which are (also) physical entities encoding part of the genetic information, genes are rather defined by the phenotypical trait they determine. That is, a gene can be one continuous part of a chromosome, but can also be distributed across the chromosome; it can occupy a larger or smaller fraction of the chromosome. The actual bit of information coded by the gene is referred to as an allele, while its position on the chromosome is called the locus. A cell can reproduce itself either by mitosis or meiosis. In mitosis, the cell first copies its nucleus (and thereby its genome) and then divides itself into two – genetically – identical halves. For organisms with sexual reproduction meiosis plays a crucial role in mixing the genetic information of two parents.

Meiosis. From an information processing point of view, meiosis can be simplified to the sequence depicted in Fig. 2.1: The diploid cell contains one paternal and one maternal genome. At the beginning, the paternal and maternal chromosomes are aligned in a way that the homologous chromosomes, i.e. those that contain the same genes, form pairs. Subsequently, the chromosomes are split into identical halves called chromatids, which is possible owing to the redundant structure of the DNA. The aligned pairs of homologous chromosomes thus become a tetrad. Pairs of the chromatids join now at several crossing points called chiasmata, typically between 1 and 8. The location of the chiasmata is chosen randomly. The exchange of parts of the chromosome between the chiasmata is called crossing over. It mixes maternal and paternal information on the intra chromosome level. At the end of the first stage called meiosis I, the mixed chromatids join to become chromosomes again, which are then aggregated to two complete genomes. As each chromosome is chosen at random from the two alternative chromatid pairs, recombination on an inter chromosome level is achieved here. Finally, in the second stage of meiosis, the chromosomes of the two genomes are divided separately into chromatids that are then completed to form four different sets of chromosomes in four separate cells. As each of these cells contains only one (mixed) genome, they are called haploid.

Haploidy and Diploidy. Although there are not only haploid and diploid, but also polyploid organisms and even those that switch between haploid and diploid with each generation, the discussion shall be limited to haploid and diploid organisms. In diploid creatures, as e.g. in human beings, all cells, except for the gametes, contain a maternal and a paternal genome. In the meiosis, gametes, that is haploid cells, are produced from diploid cells. In the mating process of two gametes, the two single genomes merge into a new diploid cell from which the new child organism grows. This latter development is called ontogenesis. In haploid organism, this scheme is reversed in that all body cells contain only one copy of the genome, but a meiosis is necessary to reduce the number of genomes subsequent to the mating of two gametes from two to one. Often, evolutionary algorithms implement the recombination in a rather haploid manner, that is, the genome that is responsible for the ontogenesis is passed to all of the children. Diploid inheritance differs from this scheme, in that the phenotype is determined by the way the information is read off from the pair of maternal and paternal genomes,

[2] At least in the case of eucaryotes, to which the description shall be limited.
[3] DesoxyriboNucleic Acid

whereas the gametes contain a mixture of the paternal and maternal genetic code that is completely uncorrelated with this former genome the ontogenesis is based on.

Mutations. In the previous section, the variations induced by mutations have been identified as being the driving force for evolution. The different types of mutations observed in living creatures can be classified by their scope: The smallest mutations are point mutations occurring at the gene level. Here, a base pair of the nucleotide sequence of the DNA is replaced by another base pair, which may or may not change the phenotypical trait related to the gene. On the chromosome level, four different types of mutation are known [Got94]. First, a part of the genome comprising one or more genes may simply be deleted. Second, such a part may be cut out of the genome and be inserted in the inverse order after being rotated by 180°. While the above two types of mutation are restricted to one single chromosome, the following two types require the presence of two chromosomes. These are duplication and translocation, where the pair of chromosomes must be homologous in the former and non-homologous in the latter case. In a duplication, a snippet of one chromosome is inserted into

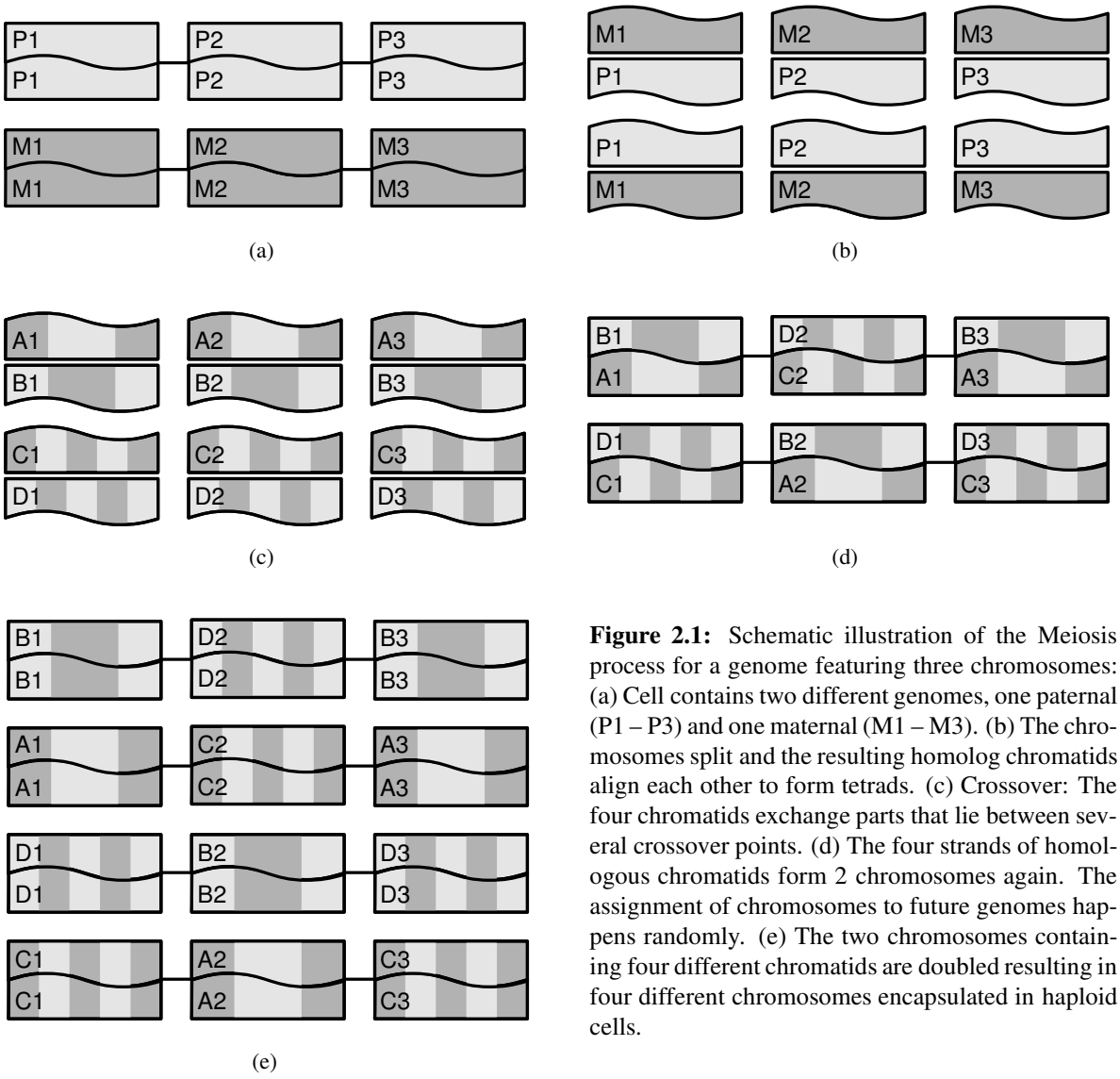


Figure 2.1: Schematic illustration of the Meiosis process for a genome featuring three chromosomes: (a) Cell contains two different genomes, one paternal (P1 – P3) and one maternal (M1 – M3). (b) The chromosomes split and the resulting homolog chromatids align each other to form tetrads. (c) Crossover: The four chromatids exchange parts that lie between several crossover points. (d) The four strands of homologous chromatids form 2 chromosomes again. The assignment of chromosomes to future genomes happens randomly. (e) The two chromosomes containing four different chromatids are doubled resulting in four different chromosomes encapsulated in haploid cells.

the other one, shortening the first and duplicating part of the second chromosome. An exchange of genetic material similar to crossing over that occurs between non-homologous chromosomes leads to a translocation of genetic material. Finally, there are also mutations on the genome level, which are either changing the number of one particular chromosome of one cell (e.g. trisomy 21 in case of humans) or changes the ploidy of the genome, that is the number or complete sets of chromosomes.

2.1.3 Ontogenesis

So far, only the storage on the DNA and the means of their variation and proliferation have been considered. However, as it is the phenotype that is evaluated by its environment, the interpretation of the genetic code, that is the ontogenesis from the genome to the final phenotype, is of utmost importance to natural evolution. Unfortunately, this highly complex process is far from being understood. The discussion below sheds a bit of light onto the complexity of the genotype phenotype mapping:

Depending on the type of organism, the genome contains large fractions of unused information. In fact, for humans it is estimated that only a few percent of the DNA are actually used to encode information. First, a process called transcription extracts the coding bits of information stored on the DNA, that is, it converts the DNA to messenger RNA[4], which contains only the coding snippets called exons, but none of the non-coding parts called introns. In the second phase referred to as translation, sequences of three base pairs called codons are translated into one of 20 different amino acids. A sequence of these codons makes up for one chain of amino acids forming a polypeptide (also called protein). These proteins may subsequently be further modified and usually change their structure, that is e.g. the chain morphs itself into a spherical shape.

A cell cannot at one time transcribe and translate the full genome. The mechanism that describes which quantities of which gene are expressed in what kind of environment is considered a self-organizing process referred to as gene regulation. Gene regulatory networks are believed to be responsible for the differentiation occurring during ontogenesis, that is the mechanism by which cells carrying the identical genetic information as the first zygote develops into very different cell types as, for instance, liver, skin or neural cells. In summary, the genotype phenotype mapping found in biological systems is highly complex, indirect and requires some sort of self-organization.

2.2 Overview of Evolutionary Algorithms

Evolutionary algorithms are a class of algorithms that imitate some of the features of natural evolution to solve optimization problems. Thereby, the environment providing the fitness evaluation in case of natural evolution is replaced by a cost function that describes the optimization task at hand. In contrast to many other optimization procedures, evolutionary algorithms usually employ not one, but a whole population of candidate solutions. Here, the population can be conceived as one species that is to adapt to the given problem specified by the user. An evolutionary algorithms may be used to constantly adapt a solution to a varying cost function. However, the genuine variants of evolutionary algorithms concentrate on one species and thereby discard the dynamics of coevolving species found in natural evolution. Another important difference between biological and artificial evolution is the complexity of the genotype phenotype mapping. Though the representation of the candidate solutions is indeed crucial to the success of evolutionary algorithms, it is usually kept much simpler than the – admittedly not yet understood – ontogenesis of living organisms. Yet, some researchers believe developmental processes linking genotype and phenotype to be the key in structural design optimization [Had01], [Dev03].

[4] RiboNucleic Acid

2.2.1 Operation Principle

The generic scheme of an evolutionary algorithm is depicted in Fig. 2.2 (cf. [Wei02]). At first, a population of candidate solutions must be created. Viable alternatives would be to start with individuals that are randomly generated, individuals that stem from a prior evolution experiment, or individuals obtained by guessing an approximate solution to the problem. Independent of the quality of the members of this first generation, they must comply with the syntax inherent to the fitness evaluation. Before entering the evolutionary loop, the individuals of the initial population must be evaluated. The next four operations in the evolutionary loop describe the generation of a new population: At first, this requires the selection of suited parent individuals that are allowed to inherit their genes to the next generation. Selection favors the fitter individuals as they attain a larger probability to be chosen for producing new offspring. In the next step, the selected genetic material is changed by the variation operators that realize the artificial analogies to natural recombination and/or mutation. At this point, the fitness of all of the child individuals is evaluated. The new generation can then be obtained by a second selection procedure that is based on the results of the previous two fitness evaluations. Here, a better fitness increases the probability to be part of the new generation, that is, to survive. The sequence described above is repeated until a termination criterion specified by the user is met. This could either be a (sufficiently) optimal solution, a fixed number of iterations, or a given amount of wall-clock or computation time.

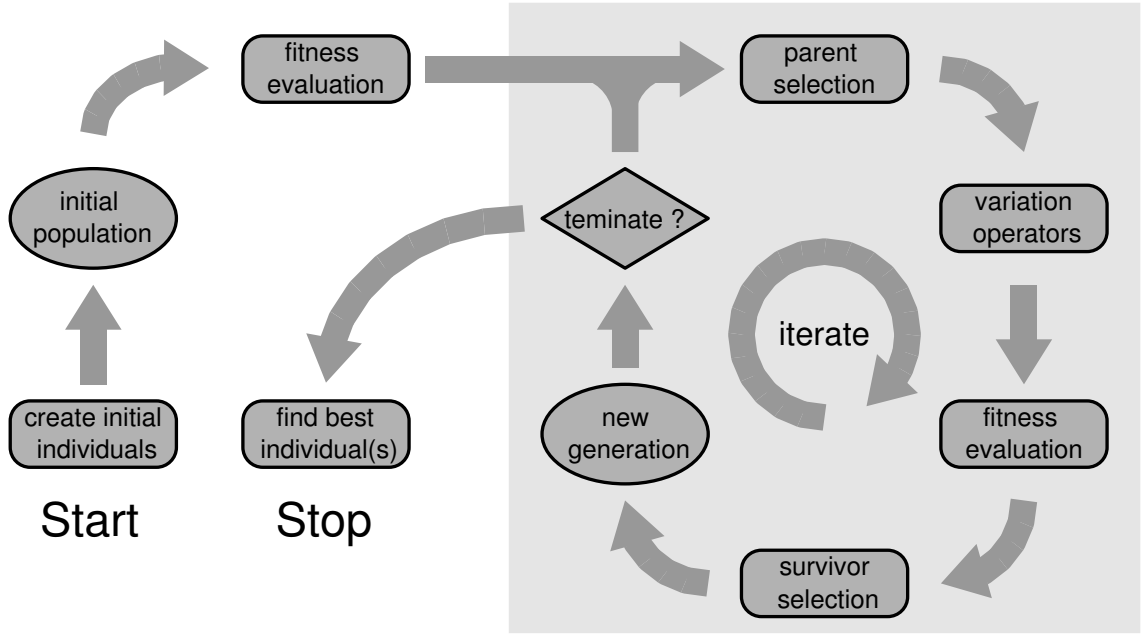


Figure 2.2: Generic scheme of an evolutionary algorithm. Its different constituents are discussed in the text. The graphical presentation is inspired by Hohmann [Hoh05].

2.2.2 Components of an Evolutionary Algorithm

In order to solve a concrete problem, the following components of an evolutionary algorithm must be specified for the algorithm to be applicable to the problem:

- representation
- variation operators

- fitness function
- selection scheme

The term representation accounts for the encoding of the candidate solutions, or more precisely, the encoding of their genotypes, to which the variation operators are applied. While the first three components are closely related to the actual problem to be solved, the selection scheme is a more general feature. Its discussion is therefore deferred to the next section.

The description of the different components will be clarified by means of a toy[5] problem, namely a least square fit. The optimization problem shall be stated as follows: For a given set of $N = 100$ data points $D[i] = (D_x[i], D_y[i])$, $0 < i < N$, find the seven coefficients α_j , $0 < j < 6$ of a polynomial of order 6, that minimize the mean square error

$$\mathrm{MSE} \equiv \frac{1}{N} \sum_{i=0}^{N-1} \left(f_{\mathrm{pol}}(i) - D[i] \right)^2 \quad \text{with} \quad f_{\mathrm{pol}}(x) = \sum_{j=0}^{6} \alpha_j x^j \ . \tag{2.1}$$

between the data points and the curve defined by the polynomial.

2.2.2.1 Representation

Analog to natural evolution, any individual of the populations possesses – in general – a genotypical and a phenotypical manifestation. The sought solution to the problem is defined in the phenotypical problem space. This could be, for instance, an electric circuit, a neural network, or the shape of an airfoil. In the case of the curve fitting example, the phenotype would be the mathematical function described by the polynomial. The variation operators, on the other hand, are bound to act in the genotype space – usually a subspace of data structure realizable with a digital computer. Prior to the fitness evaluation the genotype must be mapped to the phenotype, which is referred to as the decoding. The reverse mapping, the encoding is called representation in the context of evolutionary computation. It describes the data structure that encodes the phenotypical candidate solution. The terminology with which the genotype and its constituents are denoted is borrowed from the biological model, that is the whole genotype would be also the genome that may or may not be divided into chromosomes. Each chromosome in term possesses a set of genes, which are usually realized as variables or again defined data structures. Yet the gene would be the smallest entity whose value can be varied.

In case of the curve fitting example, a straightforward representation of the genotype could be as simple as a vector of the seven coefficients $\vec{\alpha} = (\alpha_0, \ldots, \alpha_6)$ encoded as 32-bit floating point variables. Another popular encoding uses bit strings. Independent of the actual mapping of bits and floating point variables, the choice will probably be disadvantageous. Although it looks like an unnecessary complication, a bit string representation can, however, also be beneficial and is much more natural to discrete, combinatorial problems. Both encodings are illustrated in Fig. 2.3. In case

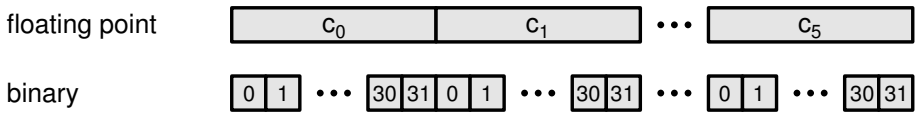


Figure 2.3: Floating point (top) and bit string (bottom) representation for the curve fitting example.

[5] First it should be noted that curve fitting in general is *not* a toy problem at all. Second, good mathematical curve fitting procedures do exist (e.g. the method of Levenberg and Marquardt); an evolutionary algorithm is by no means expected to beat these dedicated algorithms.

of the floating point representation, each α_j is considered one gene, whereas in the bit string it may be useful to separate the bits belonging to one floating point into chromosomes. Yet, this depends on the definition of the variation operators.

2.2.2.2 Variation Operators

The variation operators act on the genotypes and thus allows individuals to traverse the genotype space. Their implementation is closely linked to the chosen representation and the combination thereof decides upon the possible steps in the genotype space. In particular, the variation operators determine whether all points in the search space can be accessed, or, if not, which subset of them can be sampled. Analog to the biological model, variation operators are classified according to their arity, that is the number of input objects they act on. Unary operations are referred to as mutations, whereas binary operators are subsumed as recombination or crossover operators. Recombination operators acting on more than two genotypes are used rather infrequently, maybe due to the lack of biological precedence [Eib03a]. It is mainly the variation operators that are responsible for the stochastic nature of evolutionary algorithms. Hence, they must include some probabilistic element.

Mutation. A mutation is to apply a small random change to the chosen genotype it acts on. As mutations offer the smallest steps that can be taken, they sample the close vicinity of one point in the search space. The most fundamental mutation operations for bit strings and floating point variables can be explained in terms of the least square fit example: In case of a bit string, each bit is flipped with a probability $1/p$. This probability is referred to as the mutation rate; the default recommendation is to flip one bit per mutation on average, that is for a bit string of length L, choose $p = 1/L$ [Weg]. Apart from this point mutation other mutation involving two or more bits, such as swapping two bits, inserting a bit, inversion or randomization of the order of a group of bits are reported in [Eib03a]. For a vector of floating point variables, or integers taken from a virtually unlimited range, mutations are typically realized by adding a random vector, whose entries are randomly chosen from a normal distribution with mean μ. Hence, a mutation of the set of coefficients used for the least square fit example can be described by $\vec{\alpha} \rightarrow \vec{\alpha} + \vec{r}$. The variance σ of the normal distribution is the equivalent to the bit flip probability p.

Recombination. The crossover operator re-combines the information contained in two genotypes to generate new offspring. Often, two parents are used to create two children. The recombination of genetic material is again usually probabilistic in choosing which parts of the new child genome are taken from which parent. Yet, for the arithmetic crossover described below this is not the case. Referring back to the least square fit example, Fig. 2.4 illustrates possible crossover operations for the floating point vectors $\vec{a}$ and $\vec{b}$ denoting members of the respective population.

In single point crossover, the two vectors are split at one randomly drawn position; the resulting four parts are then joined with the complementary parts from the other parent, respectively. N-point crossover, depicted for $n = 2$ works similarly, albeit breaks the genome into n contiguous segments that are exchanged between the two original genotypes. In uniform crossover, the allele for each gene locus is chosen randomly from either of the two parents. While each of these three crossover variants results in a complementary pair of child genomes, that is, only the combination of genes is altered, uniform arithmetic crossover (also denoted as *Gaussian perturbation*) produces linear combinations of the parental genomes that do not contain the original genes any more. In general, arithmetic crossover can be described by $\vec{c} = s\vec{a} + (1-s)\vec{b}$ for the first and $\vec{d} = (1-s)\vec{a} + s\vec{b}$ for the second child genome. In Fig. 2.4 $s = 1/2$ is used, which yields only one new genome. One and n-point crossover

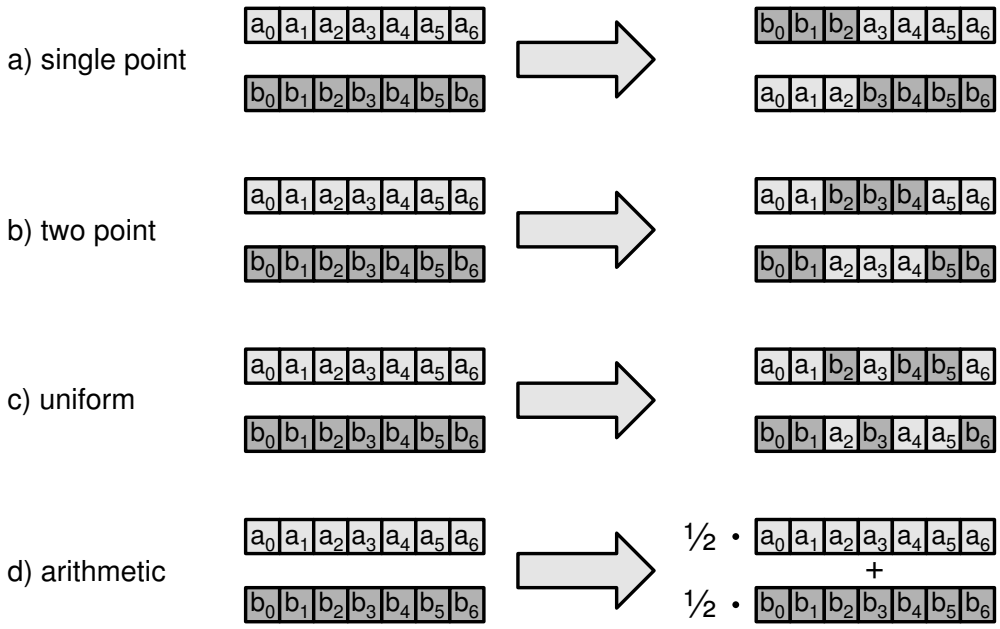


Figure 2.4: Different crossover operators applied to the genotype of the least fit square optimization example.

as well as uniform crossover are equally defined for bitstrings, but arithmetic crossover is limited to floating point representations.

2.2.2.3 Fitness evaluation

The evaluation or fitness function is the artificial analogon to the environment in biological evolution. It must capture the optimization task at hand in that individuals that are closer to the sought optimum attain a higher fitness. This is nontrivial. On the contrary, it is easy to construct problems that exhibit the opposite behavior. Consider for example the ONEMAX problem: For a bit string of given length, the fitness is defined as the sum of all zeroes. If one reformulates the goal such as to assign the highest fitness to the bit pattern containing all zeroes, the problem becomes a *deceptive* problem and does not satisfy the above requirement. Admittedly though, the problem is sort of pathological. Mathematically, the fitness evaluation is defined as a function that maps the genotype space to a scalar[6], that is $f : \mathscr{G} \rightarrow \mathbb{R}$. In case of the least square fit example, the fitness function could be simply defined by (2.1): $f : \mathbb{R}^7 \rightarrow \mathbb{R},\ f = \text{MSE}$, although other mappings may be conceivable.

Traditionally, evolution problems are set up as maximization problems. However, the above curve fitting example as well as all artificial evolution experiments presented within this thesis try to minimize fitness. This owes to the fact, that all of these performance criteria measure the deviation from a specific desired behavior. In such a context it is more convenient to state the task as a minimization problem in which some error function like the MSE is minimized. Nevertheless, a minimization problem can, in principle, be easily reformulated to a maximization problem and vice versa. Care must only be taken if absolute values of the objective function are of importance, which would be the case in the fitness proportional selection scheme described below. It is not within this thesis, as fitness proportional selection is not used for any of the presented experiments.

[6]The subject of multiobjective optimization will be addressed in section 2.5.2. For the rest of the thesis, fitness is understood to be a scalar.

It can be instructive to think about the optimization task in terms of the so-called *fitness landscape*, which is defined as the hyperplane formed by all genotype fitness value pairs in $\mathscr{G} \times \mathbb{R}$. To foster imagination, consider the case $f : \mathbb{R}^2 \to \mathbb{R}$: Here, the fitness landscape features peaks and valleys, whose altitude is defined by the respective fitness values. In this picture, the landscape is defined by the combination of the representation and the fitness function; the variation operators enable the algorithm to move the population of points through this search space. The goal it then to find either a maximum or a minimum in this landscape. Accordingly, the nature of this fitness landscape is tightly coupled to the complexity of solving the task at hand. If the fitness landscape is smooth featuring one single global optimum, say a smooth valley, the task is trivial in that a simple hill climbing algorithm is bound to find the optimum. Other scenarios are not: Imagine for example a rugged landscape featuring many local minima. Simple hill climbing algorithms will be trapped in the very next local optimum. Another very difficult task is given by the needle in a haystack problem, in which the entire fitness landscape is perfectly flat except for one single peak. A search algorithm can only find the optimum by a lucky choice, or exhaustive search, since none of the sampled points in the search space provides any information about the optimum, except the optimal point itself.

In summary, this leads to the following interpretation. The shape of the fitness landscape determines the level of difficulty in finding optimal or near optimal solutions. The fitness landscape in turn is determined by the fitness function and the chosen representation. Hence, the design of these two EA components provide the most freedom for improvements. If fitness function and representation can be chosen such that the resulting fitness landscape features one global minimum that smoothly ascends in all directions, the problem is, in principle, solved. The design of suited fitness functions and representations, however, does not only possess the highest potential for increasing the chances for success, it is also the most difficult part. As the fitness criterion is to a good extent determined by the sought optimization goal, it is the representation that is believed to be the most prospective candidate for improving an evolutionary algorithm's performance. This probably can only be achieved by including problem-specific knowledge, which in turn puts into perspective the claim that evolutionary algorithms are general purpose problem solvers.

2.2.3 Selection Schemes

In contrast to the components discussed in the previous section, the selection mechanism is a more general feature of evolutionary algorithms that can be applied relatively independent of the problem at hand. In particular, the selection scheme governs the population dynamics in the following sense: It is commonly perceived (see e.g. [Eib03a], [Wei02], [Mic99]) that the course of evolutionary algorithms can be divided into two phases, that of *exploration* and that of *exploitation*. At first, the members of the population are sampling distant points in the search space. While mutation helps the individuals to explore their local neighborhood, recombination and selection move the individuals on a larger scale. Finally, the population will converge to small mutational variations of one single individual due to the selection procedure. In other words, optimization occurs only locally, as the population is restricted to a small region of the search space. The selection scheme is liable for the extension of the two different phases of exploration and exploitation and as such influences the chances for the algorithm to succeed in solving a particular type of problem. The case, in which the population converges to a significantly suboptimal region in the search space is referred to as *premature convergence*.

2.2.3.1 Population Models

The generic evolutionary algorithm scheme depicted in Fig. 2.2 distinguishes between parent and survivor selection, which are also referred to as selection and replacement [Eib03a], which again

bears some resemblance to the natural evolution process of higher life forms: First, new offspring has to prove successful in surviving. Later on, fitter individuals are often accredited higher chances in being selected as a mate to create the children of the next generation. The scheme of Fig. 2.2 however, rather models the situation in which parents and offspring have to compete for resources and mates for some time.

In artificial evolution, the treatment of parental and survivor selection manifests itself in different replacement strategies. First, two different population models, namely the *generational* and the *steady-state model* are discerned [Eib03a]: In the former one underlying the above description of the generic evolutionary algorithm scheme illustrated in Fig. 2.2, an entire generation is renewed within each iteration. On the other hand, in the *steady-state model* only a fraction of the population is replaced. If μ defines the size of the population at the starting point denoted as *new generation* in Fig. 2.2 and λ the number of offspring generated from the μ parents, then $\lambda < \mu$ individuals are replaced in each iteration. The second option concerns the pool of possible survivors: They may be selected exclusively from the $\lambda \geq \mu$ children or take also the μ parents into account. In the latter case, the selection can either be based on the fitness of the intermediate $\mu + \lambda$ individuals or on their respective age.

2.2.3.2 Selection Schemes

A variety of different selection schemes have been reported. Usually, they can be used for both, parent and survivor selection. Yet, not all combinations thereof make sense [Wei02]. Moreover, one selection may even be omitted. The most popular selection schemes are briefly discussed below. They mainly differ in their selection pressure, which can be quantified either by the expectation value for takeover time needed for the best individual to replace all other individuals in the absence of genetical variation, or by means of the selection intensity defined as

$$I_{\text{sel}} = \frac{\overline{f_{\text{sel}}} - \overline{f}}{\sigma_f} \quad .$$

Here, $\overline{f_{\text{sel}}}$ denotes the mean fitness subsequent to the selection and σ_f the standard deviation of the fitness values prior to selection. A large value of selection intensity entails a strong preference for better individuals. Yet, both measures are strongly dependent on the actual fitness distribution and, according to Wegener, difficult to interprete [Weg]. Hence, it is not attempted to quantify the selection pressure of the respective scheme, but the different schemes are rather presented in descending order of selection pressure [dJ04].

Truncation Selection. The offspring/survivors are selected from the best P_{trunc} percent of individuals. Often, truncation selection is meant to be deterministic, that is that indeed all of the best P_{trunc} individuals are selected [Rud97]. This is the case for the standard selection schemes used in *evolution strategies*, which are abbreviated by (μ, λ) and $(\mu + \lambda)$. The former scheme selects the best μ individuals exclusively from the λ $(> \mu)$ offspring, whereas the latter one extends the selection to include the μ parents, too. However, the hardware evolution experiments presented within this thesis refer to truncation selection as a probabilistic scheme in that as many individuals are drawn from the best P_{trunc} percent of individuals as required to fill the population, even though Wegener [Weg] denotes this as threshold selection.

Rank Based Selection. Let $0 \leq r < \mu$ denote the rank of an individual in the population, so that the fittest individual possesses the highest rank. The probability of being selected can now be calculated

as a function of this rank. In principle, a large variety of functional dependencies are conceivable; the most popular ones are [Eib03a] [Weg]:

$$P_{\mathrm{lr}} = \frac{2-s}{\mu} + \frac{2r(s-1)}{\mu(\mu-1)} \quad \text{for} \quad 1 \leq s \leq 2 \qquad \text{linear} \tag{2.2a}$$

$$P_{\mathrm{qr}} = \frac{1}{C}(\alpha + \beta r^2) \qquad \text{quadratic} \tag{2.2b}$$

$$P_{\mathrm{er}} = \frac{1}{C}\left(1 - e^{r/\gamma}\right) \qquad \text{exponential} \ , \tag{2.2c}$$

$$\text{with} \quad C = \sum_{r=0}^{\mu-1} P_{\mathrm{xr}} \ .$$

Equ. (2.2a) is already normalized. The selection pressure can be adjusted from zero (s=1) to a linear scheme in which the worst individual cannot be selected, whereas the probability for the best one amounts to $2/\mu$ (s=2). The probability P_{lr} is defined such as to assign a medium probability $1/\mu$ to an individual with rank $(\mu-1)/2$. To achieve a higher selection pressure, a quadratic or exponential rank based selection must be used as described by (2.2b) and (2.2c), respectively. The constants α, β and γ can be used to further adjust the rank based probability distribution.

Tournament Selection. In a q-ary tournament selection q randomly drawn individuals have to compete against each other. Only the best of those q individuals is selected and the process repeated until the desired number of individuals is determined. The selection intensity can be adjusted by the number of competitors q participating in the tournament: More competitors cause a larger selection pressure. As the result of each competition does only depend on the rank of the participating individuals, tournament selection can be interpreted as a variant of rank based selection [Weg]. In contrast to truncation and rank based selection, tournament selection does not require the population to be sorted prior to selection.

Fitness Proportional Selection. The selection probability $P_{\mathrm{fp}}(x)$ is defined to be proportional to the fitness $f(x)$ of each individual x. Owing to the necessary normalization, this yields:

$$P_{\mathrm{fp}}(x) = \frac{f(x)}{\sum_{y=0}^{\mu-1} f(y)} \ . \tag{2.3}$$

In the traditional form of (2.3) fitness proportional selection can only be applied to problems where the fitness has to be maximized. Hence, for minimization problems, an appropriate transformation must be used. Weicker [Wei02] identifies two additional drawbacks of the fitness proportional selection scheme: First, $P_{\mathrm{fp}}(x)$ depends on the offset of the fitness value distribution: If the offset is large compared to the difference between highest and lowest fitness values, fitness proportional selection will treat the whole population almost uniformly. This can be mitigated by rescaling the fitness values so that only the range of fitness values of the last w generations is taken into account. A more advanced method referred to as *sigma scaling* [Eib03a] uses mean and variance of the fitness value distribution to achieve a more appropriate selection. However, the second disadvantage, namely that individuals with an outstanding fitness will take over the whole population within a few generations, remains. In other words, if large changes in the fitness occur, evolutionary algorithms using fitness proportional selection are prone to premature convergence. As large changes (drops in the case of problems where fitness is minimized) are found to be the rule rather than the exception for the kind of structural design optimization tasks presented in this thesis, fitness proportional selection has not been used for the proposed hardware evolution experiments.

Elitism. Of the four schemes described above, only the (μ, α) and $(\mu + \alpha)$ strategies are deterministic, whereas all others randomly choose the individuals according to the underlying probability distribution. Moreover, only if the $(\mu + \alpha)$ strategy is used for survivor selection, it ensures that the best individual is propagated to the next generation. Accordingly, if it is desired to keep the best individual, this must be done independently of the selection process defined by all other schemes. The unconditional propagation of the best individual is referred to as *elitism*.

2.3 Dialects of Evolutionary Algorithms

Evolutionary computation can be divided into four different approaches. Three of them, namely *evolutionary programming*, *evolution strategies* and *genetic algorithms*, have been developed independently in the second half of the 1960's. A survey of their historical development is presented e.g. in [Fog94] and [Bäc97]. The fourth dialect, *genetic programming*, was founded around 1990 by Koza [Koz90], [Koz99a]. It was only at this time, that the field of evolutionary computation emerged as one research field that encompasses all four different research avenues. A summary of the entire research area as well as summaries of the particular dialects can be found in [Eib03a], [Wei02]. In the following, the main features of *genetic algorithms*, *evolution strategies* and *genetic programming* shall be briefly summarized. The approach defined by *evolutionary programming* is foregone, because its original version was devised to the evolution of finite state machines, and modern evolutionary programming bears strong resemblance to evolution strategies [Wei02].

2.3.1 Genetic Algorithms

Simple Genetic Algorithm. Genetic algorithms are not only the most popular evolutionary algorithm, they are also the least concisely defined concept in that they comprise the largest variety of different representations, operators and selection schemes. In fact, almost any of the features of evolutionary computation that have been discussed so far can be part of an contemporary evolutionary algorithm. Nevertheless, in a narrower and more historical sense, genetic algorithms (GAs) are variations of the *simple* GA that is defined by the following instantiations of the respective components of an EA: The genotype is encoded as a bit string in exactly the way suggested as an alternative representation for the curve fitting example introduced in section 2.2.2. The simple GA is restricted to 1-point crossover and point mutations realized as bit flips. It uses a generational population model and a fitness proportional scheme for parent selection, yet foregoes an extra survivor selection.

Viable Representations. In a more general perception, GAs are amenable to wide variety of representations. Popular alternatives to the bit string and floating point number representations discussed in section 2.2.2.1 are integer and permutation representations. The former ones account for ordinal attributes, that is subsets of $\mathbb{Z}$ that encode an order and cardinal attributes as, for example, the four cardinal points North, East, South and West. The latter ones are used to model a set of possible orders in which different events are occur. Permutation representations are usually supported by a set of special variation operators adapted to their needs [Eib03a]. Again, the representation is to be chosen so that it reflects the nature of the problem. Accordingly, it may deem necessary to use a mixed representation featuring for instance a binary section as well as a vector of real-valued numbers.

Variation Operators. Different variation operators applicable to bit string and floating point representations have already been discussed in section 2.2.2.2. In case of integer variables, mutations can be either similar to a bit flip or add a relatively small, integer random number resembling the mutation used for a real-valued representation. The former alternative is referred to as *random resetting*,

whereas the latter one is denoted as *creep mutation* and may be more beneficial in case of ordinal integer variables [Eib03a]. Crossover operators of integer representations can take any of the four forms described in section 2.2.2.2. However, an arithmetic (also referred to as *intermediate*) crossover will only make sense for a set of ordinal integer values.

Selection Schemes and Population Models. Both, generational as well as steady-state population models are encountered in modern GAs. In principle, the whole variety of selection schemes presented in 2.2.3 lends itself to being used in GAs. However, the deterministic replacement strategies denoted as $(\mu,\lambda$ and $(\mu+\lambda)$ have originally been developed within the context of evolution strategies and are therefore found less frequently in GAs.

2.3.2 Evolution Strategies

In contrast to genetic algorithms evolution strategies (ES) are much more confined to one specific type of algorithm. Their distinct features can be summarized as follows: First, their representation is restricted to vectors of real values or quasi-continuous integers, that is a relatively large set of or ordinal attributes [Bäc04]. Second, ES use a uniform parent selection, usually creating $\lambda > \mu$ offspring and select the survivors deterministically according to the $(\mu+\lambda)$ or preferably to the (μ,λ) scheme. In comparison to GAs, where in general both variation operators are considered equally important, put a stress on the mutation operation, which is realized as a Gaussian perturbation. The knack of modern ES however, is their implicit self-adaptation mechanism that will be briefly discussed below for one kind of ES implementation.

Self-Adaptation. In its most sophisticated form, the individual is represented not only by a vector $\vec{x}$ of object variables that are evaluated by the fitness function at hand, but also comprises two types of strategy parameters that describe the mutation step size:

$$(\underbrace{x_1,\ldots,x_n}_{\vec{x}},\underbrace{\sigma_1,\ldots,\sigma_{n_\sigma}}_{\vec{\sigma}},\underbrace{\alpha_1,\ldots,\alpha_{n_\alpha}}_{\vec{\alpha}}) \quad \text{with} \quad n_\alpha = \left(n-\frac{n_\sigma}{2}\right)(n_\sigma-1) \ . \tag{2.4}$$

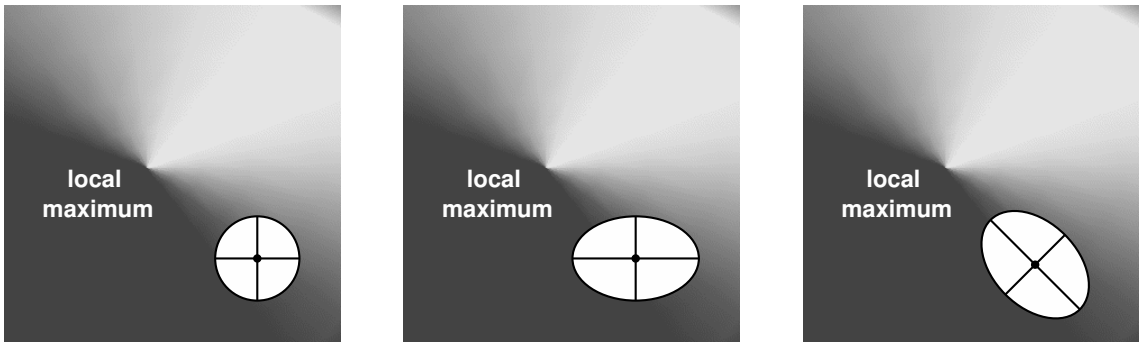
To fully appreciate the above representation it is necessary to recall the principle of Gaussian perturbation. Given an object vector $\vec{x}$, the current components x_i are altered by adding a relatively small value Δx_i, which is drawn from a normal distribution. Hence, the probability for adding the specific value Δx_i is given as:

$$p(\Delta x_i) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{(\Delta x_i-\xi)^2}{2\sigma^2}\right) \ . \tag{2.5}$$

While ξ, the mean of the distribution, is usually set to zero, the standard deviation σ defines the probable step sizes for a given mutation. Optimal step sizes depend on the problem, as this defines the range of useful values x_i. Even worse, the optimal step sizes depend on the shape of the fitness landscape and the respective position therein. Therefore, either one step size for all n directions ($n_\sigma = 1$), for some directions ($1 < n_\sigma < n$), or each direction ($n_\sigma = n$) can be included in the representation of each individual.

The impact of the strategy parameters $\vec{\sigma}$ and $\vec{\alpha}$ is elucidated in Fig. 2.5: The point in the ellipses denotes the location of the individual at hand within the two-dimensional fitness landscape. The ellipse itself describes step sizes of equal probability according to the normal distributions for both directions. In (a) the step size is identical for all directions, because it is defined by one single parameter σ. Although the introduction of one step size per direction can increase the step sizes towards the local optimum, it is still sub-optimal, as depicted in (b). Finally, (c) illustrates the situation

in which the rotation angles $\vec{\alpha}$ are also taken into account[7]: The largest step sizes point into the direction of the local maximum which can thus be reached in less steps. In summary, the strategy parameters $\vec{\sigma}$ and $\vec{\alpha}$ can be conceived as an internal model of the local topology [Bäc04].



(a) $n = 2, n_\sigma = 0, n_\alpha = 0$ (b) $n = 2, n_\sigma = 1, n_\alpha = 0$ (c) $n = 2, n_\sigma = 2, n_\alpha = 1$

Figure 2.5: Two-dimensional fitness landscape featuring one local maximum. The points within the ellipses represent a single individual. The ellipse itself contains all step sizes that possess the same probability. While (a) depicts the situation in which only one self-adapting step size σ is used for both, x- and y-direction, (b) and (c) correspond to two independent step sizes. In (c), the rotation angle α is liable for granting a step towards the local maximum a higher step size as a step orthogonal to the gradient.

Variation Operation. Typically, a new generation is created by attaining $\lambda > \mu$ offspring from the μ parent individuals, which subsequently undergo mutation and (μ, λ) selection. Although object variables $\vec{x}$ and strategy parameters $\vec{\sigma}$ and $\vec{\alpha}$ are evolved together, they are treated differently with respect to the used variation operations: While uniform crossover is used for the object variables, the strategy parameters are mixed by an intermediate recombination [Bäc04]. Both, $\vec{x}$ and $\vec{\alpha}$ are mutated by Δ's drawn from a normal distribution. However, the mutation on $\vec{\sigma}$ is based on a lognormal distribution [Bäc04], [Eib03a].

Benefits. First, ES with self-adaptation have been demonstrated to outperform ES without self-adaptation, experimentally. This observation is also backed up by theoretical results [Eib03a]. Second, ES have been proven to be successful for some problems with changing fitness landscapes [Bäc04]. Finally, the approach of ES seems to be promising in the context of parameter optimization problems with real-valued or quasi-continuous parameters. However, for applications that rely on high fitness evaluation rates and involve a large number of parameters, the computational complexity inherent to the self-adaptation process may become an issue.

2.3.3 Genetic Programming

The paramount feature that distinguishes genetic programmings (GPs) from the other variants of evolutionary computation is the representation of the genotype. In contrast to the direct encodings presented so far, GP considers the genotype as a program that has to be executed to attain the desired phenotype, which can only subsequently be evaluated. In principle, any structure that allows an execution could be used to host the evolving program. While traditional GP uses a tree representation

[7] Mathematically, these correlated mutations require an adaptation of the probability distribution in (2.5), which involves the calculation of the covariance matrix from the angles α_i.

[Koz90], [Koz99a], [Eib03a], linear representations are also reported [Wei02]. Genetic programming is closely related to genetic algorithms. Accordingly, apart from the constraints imposed by the genotype representation onto the variation operators, there are no strict limitations to the implementation of the remaining EA components. However, the variation operations are typically dominated by crossover, which is counterbalanced by large population sizes to ensure a sufficient amount of diversity within the population.

Tree Representation. The difference between standard GA implementations and genetic programming can be illustrated by returning to the curve fitting example introduced in section 2.2.2. In the previous GA like formulation, the genotype was encoded as a vector of coefficients of a polynomial described by f_{pol} in (2.1). In the context of genetic programming, the genotype must be specified by defining the syntax of the symbolic expressions (*s-expressions*). Following the tree representations introduced by Koza [Koz90], [Koz99a], the syntax is defined by requiring that the tree has to be parsed in a depth-first manner and by defining a *function set* and a *terminal set*. While the elements of the terminal set describe the possible entries chosen for the leaves at the bottom of the tree, the function set contains all entries allowed in the internal nodes. An example tree is illustrated in Fig. 2.6. If parsed, the tree results in the following candidate function for the curve fitting task:

$$f_{\text{pol}}(x) = \sin^2(x) + x^4 + (x-2)^3 \quad . \tag{2.6}$$

As the resulting candidate solutions are to approximate a one-dimensional function $f : \mathbb{R} \to \mathbb{R}$, the terminal set must comprise the variable x and real numbers as constants, that is $\mathbb{R} \cup \{x\}$. The function set provides the standard binary operators as well as power and the trigonometric functions sine and cosine: $\{+, -, \cdot, /, \wedge, \sin, \cos\}$. Although the latter two may mislead the algorithm to find non-polynomial solutions, they demonstrate the potential of the approach: Depending on the function set, the GP implementation can model virtually any function and therefore offers a solution to a much wider variety of problems than the previously discussed GA implementation. As the data points are to be fitted by an arbitrary instead of a parameterized function, the problem is also referred to as a symbolic regression.

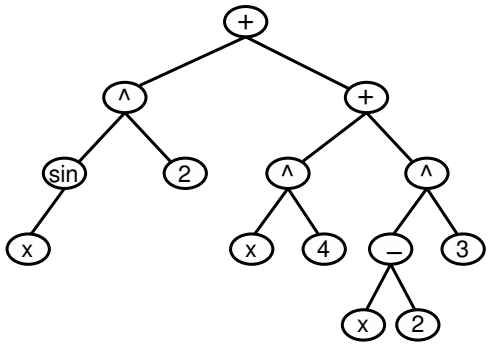


Figure 2.6: Tree representation as commonly used in genetic programming. The associated phenotype is stated in 2.6.

Variation Operators. To further elucidate the parse tree representation introduced above, possible mutation and recombination operations are exemplified in Fig. 2.7 and 2.8, respectively. Typically, a mutation can change one node value, that is a member of the function or terminal set, or replace a whole subtree by a new randomly grown subtree. However, a variety of further alterations are conceivable, as e.g. duplication, deletion or swapping of subtrees. The default crossover operation usually swaps two subtrees of the two parent individuals. That is, in each parental tree, one node is randomly chosen; the subtrees below these two nodes are then swapped. The variation operators

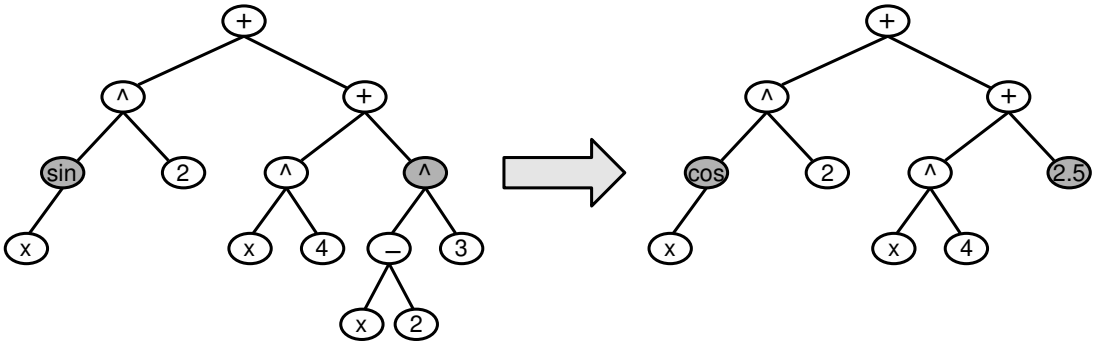


Figure 2.7: Two possible mutations in one GP parse tree: In the first mutation, the gray node containing the sin function is mutated in that its symbol is changed to cos. The second mutation exchanges the whole subtree starting at the gray node containing an $\wedge$ by a single terminal node containing the value 2.5.

mentioned above will probably have to be refined in a nontrivial application: First, it must be ensured that all resulting trees can be parsed. For instance, in the examples detailed in Fig. 2.7 and 2.8 there are function nodes taking one and others that take two input arguments. Second, careful design of variation operators may also consider the effect of different kind of alterations on the phenotype. Some effects may be undesired and should thus be avoided.

Final Remarks. There is much more to genetic programming than could be included in this short summary. However, a few additional remarks seem to be in place: First, the concept can nicely be extended, for example by *automatically defined functions* (ADFs) [Koz04a]. In this case, the algorithm evolves one result producing branch and a predefined number of branches that host functions that can be called from the result producing branch. In effect, this introduces the concept of modularity known from several programming languages to GP. Second, the GP representations differ fundamentally from the GA representations considered thus far in that the length of the genotype is variable and therefore under the control of the algorithm. This, as was indicated above, allows for a much larger degree of freedom in the genotypical description of the candidate solutions. At its downside, this freedom causes a phenomenon denoted as *bloat*, which describes the fact that, in the course of evolution, program representations tend to steadily increase in size. Apart from wasting computation power and possibly breeding inefficient phenotypical solutions, bloat may cause large parts of the genome to become irrelevant to the parsed solution. This, in turn, alters the impact of the variation operators, which may frequently change, what could be viewed as the artificial equivalent of *introns* in a natural chromosome. Though possibly useful, bloat usually necessitates the introduction of countermeasures as, for instance, penalty terms that provide the necessary *parsimony pressure*. Nevertheless, GP provides a scheme for developmental growth that may be used as a starting point for implementing an artificial analogon to biological ontogenesis. At any rate, the developmental process seems to be a good candidate for structural optimization that involves the invention of topologies rather than a mere parameter optimization. Hence, GP is believed to be a prospective candidate for enhancing the performance of the evolution system proposed within this thesis, even though its implementation has been beyond the scope of this thesis.

2.3.4 A Note on EA Parameters

Evolutionary algorithms are highly flexible: They allow for a large variety of choices for their components and also offer several parameter that may be tweaked to enhance their performance. While representation, variation operators, and fitness function are – to a large extent – prescribed by the

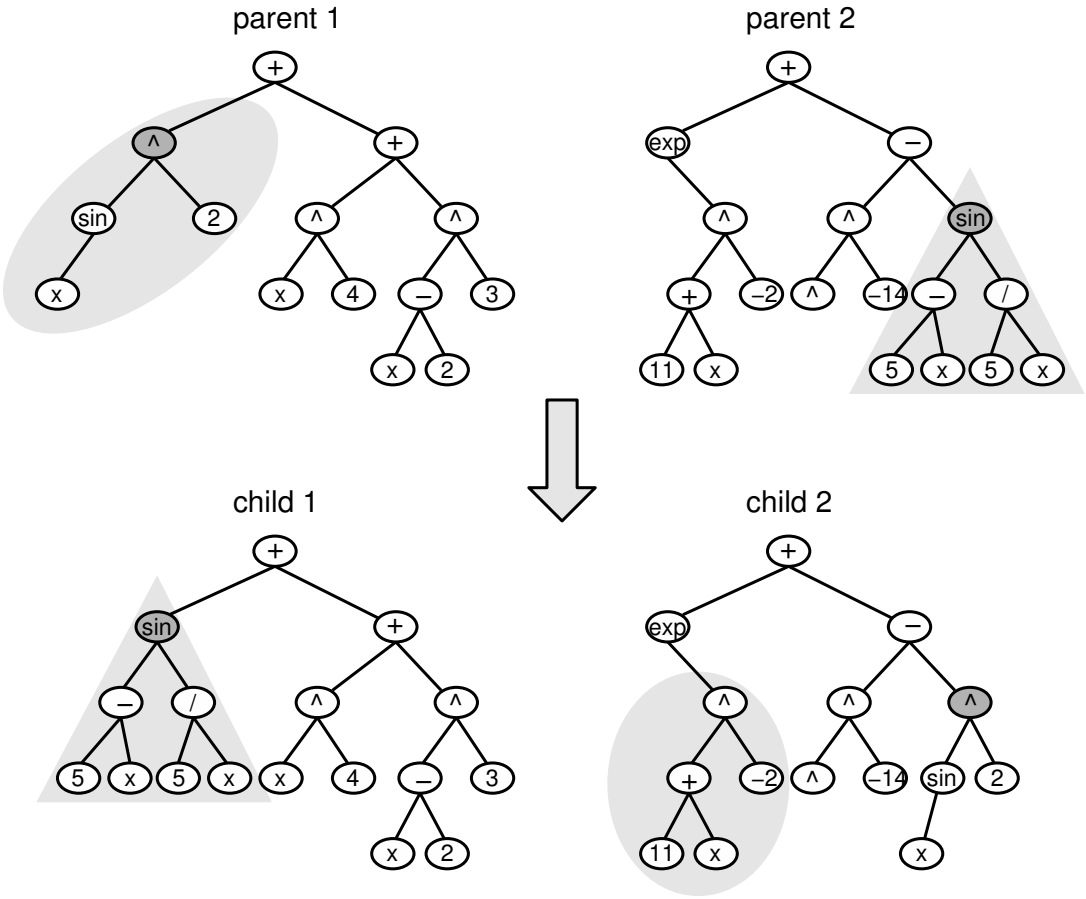


Figure 2.8: Crossover between two GP parse trees: The gray-shaded subtrees of the two parents are swapped to generate two new child trees.

task at hand, other choices have to be made independently. Those comprise several parameters like the population size, the crossover and mutation rates as well as the selection scheme, which possibly entail further parameters as, for example, the tournament size. Clearly, these parameters referred to as *strategy parameters* will affect the performance and efficiency of the evolutionary algorithm. Whether their impact is "crucial for good performance" [Eib03a] and will "greatly determine whether the algorithm will find an optimal solution or near-optimal solution" [Eib03a], however, is arguable and will depend on the problem at hand. In fact, in many of the applications contained in this thesis, the performance of the algorithm was found to depend significantly, but weakly on the strategy parameters. In this vein, EAs are robust, in that their reaction to modest parameter changes is well behaved. This observation, which has not been quantified for lack of time, is supported by the parameter sweeps presented in [Hoh02a], where the task has been to train the weights of a hardware neural network.

Apart from some general rules of thumb, the optimal choice of strategy parameters cannot be predicted independently of the problem at hand [Eib03a]. Some of these general recommendations would be to use approximately one effective point mutation per individual and generation, or to use a $(\mu, \lambda) = (15, 100)$ selection scheme in self-adaptive evolution strategies. While the former insight is rather plausible, the latter choice is motivated by empirically [Bäc04]. While typical population sizes for GAs usually range between 10 and 1000, larger populations are often used in GP; population sizes exceeding half a million are indeed reported [Koz99a].

Two general approaches in finding an optimal set of strategy parameters can be discerned [Eib03a]: *parameter tuning* and *parameter control*. The former term refers to procedures in which the optimal parameters are determined prior to the actual EA run. Unfortunately, this may require a large number of suboptimal EA runs prior to doing the optimal run, in which the parameters are to be found empirically. On the other hand, parameter control denotes procedures that adjust the strategy parameters during the run itself, which can be either done deterministically according to a prescribed schedule time, e.g. by the number of generation, or adaptive in that the performance of the algorithm is used as a feedback information. Here, one can further distinguish between adaptive methods, in which the feedback mechanism is determined by the algorithm designer, and self-adaptive approaches, in which the adaptation itself is subject to an optimization procedure. The latter situation is achieved in the self-adaptive evolutionary strategy presented above. As for this thesis, most of the experiments presented are preceded by some preliminary tests allowing for a reasonable amount of parameter tuning. However, due to the limited nature of these pre-studies, the chosen parameters are likely to be sub-optimal, not to speak of other algorithmic design decisions as selection or representation. An adaptive parameter control is used in the experiments presented in section 8.5, where the mutation rate is adjusted to the current fitness to sustain larger changes during the exploration and smaller changes in the exploitation phase.

2.4 Evolutionary Algorithms as Global Optimizers

Thus far, evolutionary algorithms have been advertised as robust problem solvers applicable to a multitude of different tasks. This section attempts to put EAs into the context of the admittedly wide and complex field of optimization. Naturally, this attempt allows only for a small glimpse on the subject matter and can only scratch at the surface. Yet, it may provide an idea in which situations evolutionary algorithms may prove useful and what they can be expected to achieve (and what not).

2.4.1 Global Optimization

Even though evolutionary algorithms are often referred to as problem solvers, these problems can usually be stated as an optimization problem of the following form: Find the set of parameters $(x_1,\ldots,x_n)$ that minimize the cost function $f(x_1,\ldots,x_n)$, where $(x_1,\ldots,x_n) \in \mathscr{P}$, where $\mathscr{P}$ denotes the space of all feasible parameters. This is actually the motivation for the metaphor of the fitness landscape , which is the n-dimensional hyperplane defined by $f(x_1,\ldots,x_n)$ in $\mathscr{P} \times \mathbb{R}$. Though in some special cases, the problem can be solved exactly, that is the global optimum can be found deterministically in a reasonable amount of time, this often turns out to be infeasible. Global optimization in the general case can only be guaranteed in the linear case in which the cost function depends linearly on the input parameters $(x_1,\ldots,x_n)$, e.g. using the simplex method, or by a virtually exhaustive search (exhaustive search or branch and bound, dynamic programming for discrete problems) [Hes]. The remaining majority of problems may be solved exactly by specialized algorithms that are tailored for this particular kind of problem. Yet, according to complexity theory, some of these problems will scale badly, that is, non-polynomial with the size of the search space; they are NP-hard.

If the computational intensity involved in finding the global optimum is prohibitively high, either due to lack of a suited algorithm, or because of the product of search space size and computational complexity, one has to resort to finding a good local optimum. In this case, often a heuristic is used, where, according to Hromkovič [Hro04], heuristic can be defined as an optimization method, "for which one is not able to guarantee at once the efficiency and the quality of the computed feasible solutions, even not with any bounded constant probability p > 0". In this sense, evolutionary algorithms are heuristics.

2.4.2 Model-Free Heuristics

However, often heuristics refer to algorithms that apply task specific knowledge to solve the problem. In other words they are based on an underlying model of the actual problem at hand. Evolutionary algorithms differ from those heuristics, in that they a priori do not rely on such a model, which renders them applicable to a wider variety of problems. There are several other model-free heuristics, as, for example, random search, local search, tabu search and simulated annealing. While local search algorithms can only find local optima, since they follow the steepest ascent in their direct neighborhood, random search does not utilize any information present in the respective fitness landscape. The idea behind simulated annealing as well as behind evolutionary algorithms is to combine probabilistic elements with a deterministic local search component. Thereby, the stochastic element shall prevent the found solution(s) from getting stuck in local optima (too early) [Hro04].

On one hand, simulated annealing can be conceived as a special evolutionary algorithm that features only one individual and whose survivor selection is based on the metropolis criterion. The according cooling schedule can then be interpreted as a deterministically varied change in selection pressure [Eib03a]. On the other hand, the implicit parallelism of most evolutionary algorithms allows to sample points that are more evenly spaced in the search space at hand and the crossover operator adds a unique way of exploiting the information gathered at these distributed points. However, the actual efficiency of the respective two types of algorithms will depend on the actual shape of the fitness landscape. As simulated annealing and evolutionary algorithms are the most popular model-free heuristics, and evolutionary algorithms can, in principle, mimic the behavior of simulated annealing, evolutionary algorithms appear to be the most important model-free heuristic available to date.

Additionally, evolutionary algorithms possess two further beneficial properties: First they can easily be parallelized, which suits their computational intensity well. They are amenable to a relative convenient implementation on computer clusters [Koz99a]. Second, evolutionary algorithms are modular, in that their main components, that is representation, fitness function and selection scheme, are independent of each other. Therefore, some of the components may be reused, while other can be exchanged to test different setups.

2.4.3 No Free Lunch Theorem

For a relatively long time, evolutionary algorithms were perceived as general problem solvers that perform relatively good on all possible problems, whereas problem specific methods would only perform well for the narrow subrange of problems they are tailored for, where they were assumed to clearly outperform evolutionary algorithms [Weg], [Mic99], [Gol89], [Eib03a], [Wei02]. However, in 1995 and 1997 Wolpert and MacReady [Wol97] proved them wrong by what became known as the *No Free Lunch theorem*. Put informally, the theorem states that all non-revisiting algorithms perform equally well, when one averages over the space of all possible problems. Here, non-revisiting means that none of the points in the search space is sampled twice, which can – at least theoretically – be ensured by combining the algorithm at hand with a tabu search.

Even though the No Free Lunch theorem contradicts the assumption that evolutionary algorithms are general purpose problem solvers for all possible problems, they may still perform comparably well on all problems one is usually interested in. Put more general, the No Free Lunch theorem raises the question, which classes of problems can be solved effective- and efficiently by which kind of (evolutionary) algorithms. Unfortunately, at least to the limited sight of the author's practitioner perspective, the field of evolutionary computing has not yet provided an answer to this question. Typically, the theoretical results are too general, or can only be derived for too simple problems or algorithms, as to be useful, whereas the scope of empirical evidence is limited to the concrete situation it emerged of.

2.4.4 Implications of the Stochastic Nature of Evolutionary Algorithms

Evolutionary algorithms are stochastic Heuristics. Therefore, they are bound to yield different results for each run. As a consequence, performance can only be determined statistically. Moreover, EAs cannot be expected to find the global optimum. Eiben [Eib03a] suggests the following performance measures: (1) Success rate, (2) mean best fitness, and (3) average number of evaluations to a solution. While the former two measures capture the performance achieved given a fixed amount of fitness evaluations, that is, measure the effectiveness of the algorithm, the latter measure account for the efficiency of the algorithm, that is, the speed with which it converges to a solution. Since the evolved circuits presented in chapters 5 to 8 are often not perfectly matching the ambitious target behavior (and due to the unavoidable noise inherent to the measurements cannot achieve perfect matching anyway), only measures (1) and (2) are used. In addition to the success rate and the mean best fitness, the distribution of fitness values is usually also presented by means of histograms, which can be conceived as a differential success rate. Histograms allow to illustrate the actual distribution, which in many cases is found to be far from any normal distribution. This in turn renders the concept of mean values and standard deviations rather problematic and hence motivates the illustration by means of histograms.

2.5 Extensions and Refinements

The above introduction to evolutionary computing covers only its core concepts. In fact, there are numerous extensions and refinements to these concepts as, for example, cooperative and competitive coevolution, linkage learning, or constraint handling. However, this last section shall address those two possible extensions to the basic concept, that are believed to be most relevant to the work presented in this thesis and that are considered to be the most prospective candidates for improving the performance of the proposed evolution system.

2.5.1 Distributed Populations

As has been mentioned in section 2.2.3, evolutionary algorithms may fail to find good local optima due to premature convergence. On one hand, maintaining a high level of diversity can only be achieved by lowering the selection pressure. On the other hand, it is the selection pressure that drives the population towards successful solutions. To escape this dilemma, different methods have been proposed to limit the recombination between subgroups of individuals. These mating restrictions can either be realized by introducing a spatial distribution (island model, distrubuted EAs) or by automatic speciation [Eib03a]. As the latter approach requires a metric to decide whether geno- or phenotype of two individuals are similar enough to be allowed to mate, the former two alternatives seem to be easier to realize in the general case. In the island model several sub-populations are evolved in parallel. These remain isolated for most of the time, albeit exchange individuals after a fixed number of generations or if some prescribed behavior is observed (e.g. stagnating fitness). On the other hand, distributed EAs place the individual on a grid and prescribe the neighborhood from which a mate can be acquired. Another interesting alternative is proposed by Hu et al. [Hu02a], [Hu03a], [Hu03b], namely that of hierarchical fair competition. The key idea is to have separate populations that differ in their fitness level. However, the individuals exceeding a certain fitness threshold can migrate to the population with the next higher fitness level. Thereby, good solutions can be fine-tuned in the high level populations while new genetic raw material can be bred in the lower levels without getting extinct due to better individuals. As the fitness thresholds segregating the different sub-populations

are crucial to the success, Hu et al. also suggest a version that includes automatic threshold adaptation [Hu02b].

2.5.2 Multi Objective Evolution

So far, the discussion has been limited to optimization problems with only one single objective. In fact, most problems encountered in the real world involve multiple, partially conflicting target specifications [Bug03], and analog circuits are no exception here: In general, microelectronic circuits are sought to exhibit as good a performance as possible consuming as little power and area as necessary. A more concrete example would be the specifications of operational amplifiers that entail several conflicting objectives as e.g. high gain bandwidth product and large phase margin or low quiescent currents and low distortion.

Within the framework of EAs, there are two possibilities to deal with this situation: First, the fitness contributions of all single objectives can be aggregated into one single fitness value by means of a weighted sum. This is advantageous in that the evolutionary algorithm must not be changed, but requires a careful choice of the respective weighting factors. Moreover, the particular choice of weighting factors may cause the population to get stuck in local optima, so that the weighting factors may have to be adapted in the course of the evolution run. Nevertheless, this aggregating scheme referred to as *scalarization* [Eib03a] is used to cope with different objectives in chapters 7 and 8 for two reasons. First, the different test modes used there can also be seen as covering different test cases for one desired behavior rather than describing conflicting objectives and second, for a lack of implementation and test of viable alternatives described below; this would have been beyond the scope of this thesis, yet currently entering the project [Tre05].

The second approach to multiobjective optimization extends the concept of fitness values to an n-dimensional vector accounting for the n conflicting objectives. Instead of seeking one (near-) global optimum, one is rather interested in a set of feasible tradeoffs located within the n-dimensional fitness space. The set of optimal tradeoffs is referred to as Pareto set, or nondominated front. It is defined as the set of all nondominated solutions, where the condition that solution A dominates solution B is expressed as [Eib03a]

$$A \text{ dominates } B \iff \forall i \in \{1,\ldots,n\}\ a_i \geq b_i, \text{ and } \exists\, i \in \{1,\ldots,n\},\ a_i > b_i \ . \tag{2.7}$$

Here, $\vec{a}$ and $\vec{b}$ denote the fitness vectors belonging to solution A and B, respectively. The Pareto front describes an (n-1)-dimensional hyperplane in the n-dimensional fitness space. From the viewpoint of an analog circuit designer, a tool that provides a set of solutions describing different tradeoffs between conflicting objectives is extremely attractive, as it allows the designer to choose a suitable tradeoff. This is particularly useful in the situation described in section 1.5, where a system consists of different building blocks whose target specifications may have to be adjusted several times to reach the overall goal. In addition, tradeoff curves that closely approximate the theoretical limit also characterize the nature of the underlying problem. A large variety of different approaches to multiobjective optimization have been proposed that, for example, address how to populate the tradeoff curves evenly. A comprehensive overview thereof can be found in [CC02].

Part II

Evolution System

Chapter 3

Implementation of the FPTA

To invent, you need a good
imagination and a pile of junk.

THOMAS ALVA EDISON

This chapter presents the concept and implementation of the analog substrate that is used for the evaluation of candidate circuits within the hardware evolution system proposed in this thesis. This analog substrate is realized as an ASIC that provides 256 programmable transistor cells and is hence referred to as a field programmable transistor array (FPTA). After a brief specification of the requirements for the FPTA chip, the principle idea as well as its realization are presented, which includes the implementation of the programmable transistor cells themselves. In particular, the deviations between the electrical properties of the programmable transistor cell and a plain transistor is addressed. Subsequently, the circuitry necessary to realize this programmable transistor array is discussed. These more technical sections include the storage of the configuration in SRAM cells together with the according interface circuitry as well as a thorough discussion of the analog circuitry necessary to apply input voltages to and read out the output voltages from the transistor cell array. Further features and some considerations on layout, power management, and yield complete the description of the FPTA implementation. The chapter concludes with a short comparison of the FPTA developed in Heidelberg and a family of similar FPTA devices designed by a different group at the Jet Propulsion Laboratory (JPL).

3.1 Rationale

3.1.1 The Very Idea of the Programmable Transistor Array

A close look at the work published on hardware evolution of analog circuits and on the work published on analog design automation reveals the following situation[1]: Research work from the analog design automation community has been focused on parameter optimization of hand-designed topologies. Within the field of evolvable hardware, artificial evolution of transistor level circuits is limited to extrinsic approaches. Intrinsic approaches on the other hand, usually employ devices of a coarse granularity that are based on high-level subsystems as for instance operational amplifiers or rely on discrete devices connected via multiplexers. This is in particular due to the lack of fine-grained configurable analog devices. Historically, two attempts to close this gap have independently emerged in two research groups, namely the group of Adrian Stoica at the JPL and the project presented within this thesis. While the former work, whose idea was first published in [Sto96] and [Sto98], has led to three FPTA implementations of increasing complexity, the FPTA chip that was developed at the University of Heidelberg shall be described below. A comparison of the two FPTA concepts will be given at the end of this chapter.

3.1.1.1 Motivation for Intrinsic Hardware Evolution

The development of a fine-grained analog array that is dedicated to the evolution of analog circuits was particularly inspired by the success of the tone discriminator experiments conducted by Thompson [Tho97], [Tho98b]. Thompson used a relatively simple genetic algorithm to evolve circuits that distinguish between square waves of two different frequencies on a digital FPGA[2]. As Thompson's first evolved tone discriminator apparently worked in an analog fashion exploiting parasitic effects of the FPGA. Thus, even better results were expected to arise from experiments with an inherently analog substrate. Moreover, artificial evolution experiments that rely on a hardware-in-the-loop setup offer several advantages compared to their extrinsic counterparts:

Intrinsic Realism. Every circuit that is successfully evolved on a real chip is bound to work in reality at least on that particular substrate it is evolved on. Although trivial at first sight, this statement possesses some deeper truth: First, although quite elaborate, the transistor and device models used in simulations are not necessarily perfect. Even if the weaknesses in describing the transistor in the weak and moderate inversion region attributed to earlier models by Allen and Holberg ([All02b], pp. 99) as well as by Geiger Allen and Strader ([Gei90a], pp. 177) do not persist in state of the art simulators, they are still models of real transistors. As new physical effects come up due to changes in process technology that enable further shrinking of devices sizes, the according models are bound to track these novelties and thus may fall short of correctly describing all of the properties of the most recent transistors. Moreover, simulators and transistor models are developed for human-designed circuits that follow some generic conventions in how they are designed. In case of unconventional designs found by means of EAs that tend to ruthlessly exploit any loopholes [Ben03] of a given simulation setup, this may lead to circuits that fail to work in reality, which was also found by Stoica et al. [Sto01b]. Second, the algorithms are also prone to exploiting/abusing artefacts in the task description as e.g. unlimited resources in terms of device dimensions, the precision in controlling those device

[1] For a short summary of the attempts in both research fields, the reader is referred to either the introduction or to [Lan03]. A more comprehensive overview of the field of analog design automation is given in [Gie00]. Further information on hardware evolution of analog circuits and inparticular on possibly suited configurable devices is available in [Zeb00b] and [Gor02].

[2] Field Programmable Gate Array

dimensions and the absence of adequate parasitics and loads as well as limited output impedances of the signal sources. However, this type of realism can, in principle, be described by the simulation setup defined by the respective user, yet requires a thorough design of the experiments and may require more complex test cases and therefore lead to increased evaluation times. Third, a hardware substrate naturally introduces the imperfections inherent to fabricated circuits as for example noise or device-to-device variations. Although these kind of effects can also be simulated, this usually requires Monte Carlo simulations that can easily increase the necessary simulation time by a factor of 100.

Evaluation Speed. As evolutionary algorithms usually rely on a large number of fitness evaluations, a fast test of the circuit behavior of interest is crucial to successful hardware evolution experiments. Accordingly, testing the candidate circuits in hardware may be advantageous in that it is often faster than a circuit simulation. Moreover, while simulation time scales badly with circuit size, the hardware tests are independent of the circuit size as long as the circuit can be realized on the hardware substrate. Finally, the simulation of the aforementioned imperfections of real circuits can be achieved more easily and may be parallelized in a hardware-in-the-loop system, which greatly accelerates the evaluation procedure.

Field Evolvable Hardware Applications. While the previous arguments motivated the usage of a fine-grained analog device as a replacement for time-consuming circuit simulations that can enhance the automatic synthesis of analog circuits, the successful evolution of analog circuits in the field has some benefits in its own right: Evolvable hardware can adapt to the environment, either once for one given environment as e.g. the particular electrical properties of a digital subscriber line, or continously to optimize the system's performance in a changing environment. This may be as extreme as some conditions encountered in space missions requiring reliable operations for temperature variations of several hundred Kelvin together with considerable amounts of radiation (cf. e.g. [Sto04], [Zeb04]. Moreover, reconfigurable devices allow for self-repair that can be exploited by the fault-tolerance inherent to population-based synthesis algorithms as was suggested by Layzell [Lay01].

3.1.2 Target Specifications

The architecture and implementation of the actual FPTA[3] chip is inspired by the arguments of the discussion above. Ideally, the prospective FPTA chip provided a freely configurable array of *virtual* transistors, that allows for full flexibility in their connectivity and channel dimensions. As it is desired to transfer the evolved circuits or design principles to future circuits, the substrate must allow for an understanding of the hosted circuits. Therefore, the electrical properties of the circuits implemented on this analog substrate are to be dominated by the chosen transistors, that is, any influence of the circuitry realizing the configurability should be negligible. This is opposed to a different sea-of-transistor like FPTA scheme that does not distinguish between configuration switches and active transistors. In addition, the FPTA implementation is to avoid any bias towards traditional design conventions. The desired features of the FPTA as well as the desired specifications and constraints its design has to meet are summarized below:

Protection against self-destruction The transistor array is to be be safe against self-destruction for all possible configuration. In case of analog transistors, the most probable hazards are metal lines affected by electro-migration and overheating.

[3]Unless denoted otherwise, the term FPTA refers to the chip designed within the proposed thesis and not to the transistor arrays designed by the group of Adrian Stoica et al. at the JPL.

Fast reliable static reconfiguration The analog substrate must endure a virtually endless number of reconfigurations. In order to avoid any interference of the configuration process and the analog functionality of the respective circuit the chip is to store its configuration statically. As a high rate of circuit evaluations is crucial for hardware evolution experiments, the time for the test of each individual and thus the time for the reconfiguration of the FPTA must be kept to a minimum. Since the actual measurements are expected to take between 100 µs and 10 ms, the time for each configuration is not to exceed 100 µs. A means to read out the actual configuration is required for two reasons: First, to facilitate the procedure of debugging and testing the fabricated tests and second, to provide a means of detecting bit flips caused by an unforeseen behavior of the circuit configuration under test.

Analog interface to signals at array boundaries The circuit tests require analog test signals that can be applied at the boundaries of the transistor cell array. Conversely, it must be possible to record the circuit response at yet another set of transistor cells beloning to the array boundary. In order to handle more than one in- and output signal with only one DAC[4] and one ADC[5], these signals must be stored in sample and hold units, which must meet the following specifications: rail-to-rail in- and outputs, high analog precision and high large-signal bandwidth ≥ 1 MHz.

Temperature measurement A device for determining the die temperature is sought for two reasons: First, to allow for a shut down of the chip to prevent it from overheating. Second, the experiments of Thompson [Tho97], [Tho98b], [Tho00] revealed that it may be necessary to take temperature into account as a design variable to evolve circuits working throughout a practical temperature range. Accordingly, the ability to precisely measure the die temperature is helpful in generating the necessary selection pressure towards circuits insensitive to temperature variations.

Inner cell probing The solutions discovered by EAs are often fairly difficult to understand (cf. e.g. [Tho97], [Tho99], [Tho00]). A method to read out the node voltages inside the transistor array and to estimate the according currents can alleviate the circuit analysis subsequent to the artificial evolution.

Inter die connectivity All signals available at the array boundaries shall be accessible via bond pads. On one hand, this allows for a direct test of the transistor array and thereby for a characterization thereof. On the other hand, the probepads can be used to connect different FPTAs to form a larger substrate.

Power management The power consumed by the non-transistor array circuitry should be kept to a minimum to avoid unnecessary heat generation, which would be counterproductive to possible substrate cooling. In additon, separate power nets should be provided for the transistor array to allow for monitoring and control of the current consumption of the circuit under test. This allows for the artificial evolution of low-power and low-voltage circuits.

Number of digital in-/outputs The number of digital signals is limited to 33 by the PCI[6]-interface card to be used.

Target technology The FPTA chip is fabricated in a double poly, triple metal 0.6 µm CMOS process qualified for analog circuits. Although the concept could benefit from a more advanced circuit

[4] Digital-to-Analog Converter
[5] Analog-to-Digital Converter
[6] Peripheral Component Interconnect

technology, the chosen technology was considered to be sufficient for this first proof-of-concept FPTA and yet less costly compared to fabrication technologies offering smaller feature sizes.

3.2 Architecture of the FPTA

At its core, the chip provides an array of 16×16 programmable transistor cells. As MOS transistors come in two flavors, half of the cells provide P- and the other half NMOS transistors. P- and NMOS cells interchange in a checkerboard pattern. On one hand, this pattern ensures an equal and fair distribution of P- and NMOS cells. On the other hand, the division into two cell types avoids that half of an aggregated P/NMOS cell remains unused. The chip features one analog in- and output, respectively. In order to drive the resistive and capacitive loads present outside of the chip, the analog output signal is buffered by an extra unity gain buffer depicted in the upper left corner of Fig. 3.1.

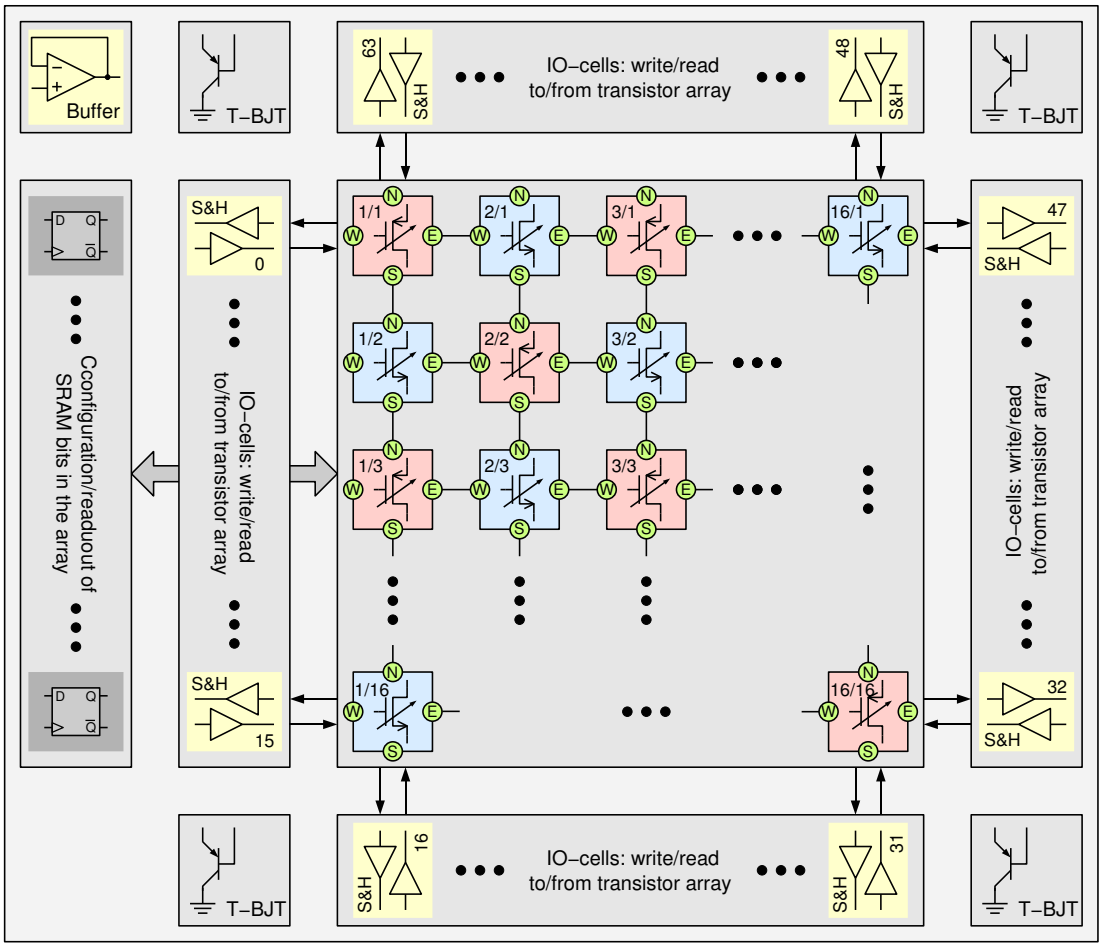


Figure 3.1: Simplified architecture of the FPTA chip.

The above list of specifications and constraints lead to the implementation of the FPTA chip discussed throughout the remainder of this chapter. The overall architecture of the chip is depicted in Fig. 3.1:

The application and readout of analog signals is encapsulated in 64 *IO-cells* that are positioned next to the edges of the transistor cell array. They are enumerated counterclockwise starting at the

upper left corner of the array. The paramount functionality of the IO-cells is to sample and hold the input signals for and output signal of their respective boundary transistor cell.

Four vertical pnp transistors located next to the corners of the transistor array serve as temperature sensors. Discrete chips that can read out the die temperature from measuring the collector and base currents through these bipolar transistors exist. The fourfold placement can be either used to get a more precise temperature measurement by means of averaging, or can be used to test for gradients across the transistor array.

The configuration of the transistor array is stored in SRAM cells embedded in the transistor cells themselves. The necessary circuitry for writing the information to these SRAM cells as well as for the readout thereof is located left to the IO-cells on the left hand side of the transistor array (IO-cells 0 to 15).

3.3 Programmable Transistor Cell Array

The programmable transistor array itself, depicted in the center of Fig. 3.1 is organized as follows: The complete functionality that includes the programmable transistor, all means of signal routing as well as the necessary auxiliary circuitry to realize these functionalities are encapsulated in the transistor cells themselves. The transistor cells provide four *border terminals* connected to their nearest neighbors in the array, which are labeled N,W,S and E after the four cardinal points (Throughout the schematic drawings of this chapter, the terminals will be marked by circles around their respective labels.) Note, that the terminals that are connected by a line in Fig. 3.1 are indeed directly connected without any further switches. Those terminals of the cells located at the array boundary that point outward of the array are connected to the adjacent IO-cells via several switches.

3.3.1 Transistor Cell Architecture

The architecture of an NMOS transistor cell is illustrated in Fig. 3.2. Yet, N- and PMOS cells merely differ in the *programmable transistor* located at the center of Fig. 3.2, which happens to emulate a PMOS-transistor for the latter type. Again, note that all border terminals of the same cardinal directions are connected, even though these connections are omitted in the schematic. The same is true for the three *generalized transistor terminals* TD, TG and TS which indicate the drain, gate and source terminals of the programmable transistor and, of course, for the power supply voltages vdd and gnd of the programmable transistor array (PTA).

Each of the generalized terminals of the programmable transistor can be connected to either of the four border terminals, vdd or gnd, which is realized by three 3-bit analog multiplexers located next to the programmable transistor. The channel width and length of the programmable transistor itself can be adjusted to $W = 1,\ldots,15\,\mu\text{m}$ and $L = 0.6,1,2,4,8\,\mu\text{m}$, respectively. Six independent routing switches implemented as transmission gates allow to connect each pair of border terminals with one another. A unity gain buffer included in the transistor cell can be used to probe some of the inner-cell node voltages as well as to attain an estimate for some of the according currents through these nodes. The concept of inner-cell probing will be detailed in section 3.6. The configuration information of the cell is stored in 4 SRAM blocks providing 6 bits each. As can be seen from Table 3.1, only 22 of the 24 bits are actually used.

A more detailed description of the assignment of configuration bits is enclosed in Table A.1 on page 306. Architecture and implementation of the SRAM and the necessary auxiliary IO-circuitry are presented in section 3.4.

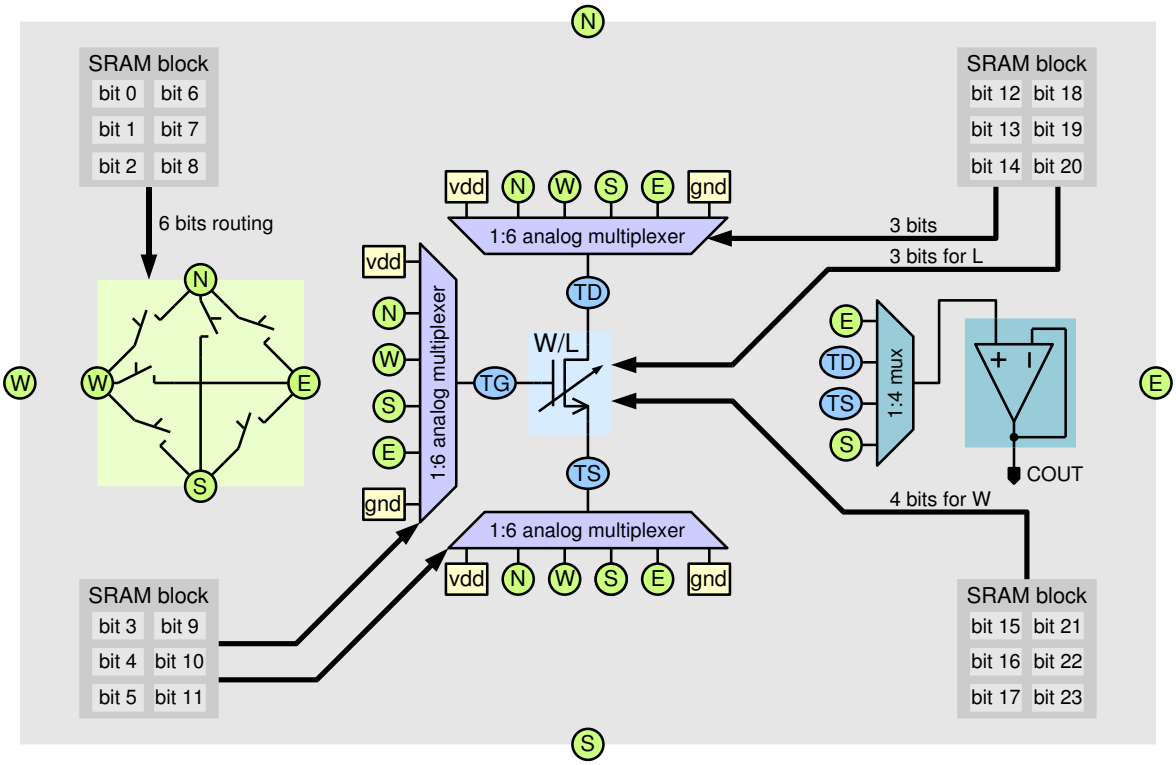


Figure 3.2: Simplified architecture of the complete NMOS transistor cell.

Property	Bit
gate connection	0–2
drain connection	6–8
source connection	9–11

Property	Bit
transistor length	3–5
transistor width	15, 21–23
routing	12–14, 18–20

Table 3.1: Allocation of the 22 configuration bits.

3.3.2 Routing Concept

Owing to the combination of analog multiplexers and routing switches any possible one-transistor circuit can be realized: While circuits a) to c) in Fig. 3.3 capture all of the conventional ways in which transistors are used, circuits d) and e) can also be realized by means of a transistor cell, even though they lack any practical usefulness. This freedom in configuring each of the transistor cells leaves room for an unbiased search for novel circuits beyond conventional human design knowledge.

Due to the distinction between P- and NMOS transistor cells, the implementation of practical circuits on the PTA substrate will often necessitate allocating some of the available cells exclusively to signal routing. While the chosen checkerboard pattern supports the implementation of subcircuits composed of adjacent P- and NMOS transistors like inverters, it lends itself less easily to forming subcircuits requiring transistor pairs of the same type, as e.g. differential pairs or current mirrors.

Although this problem could be mitigated by providing additional routing channels, of which some may even serve as medium distance or global connections, this extension was foregone in the described FPTA implementation for three reasons: First, additional routing switches inevitably

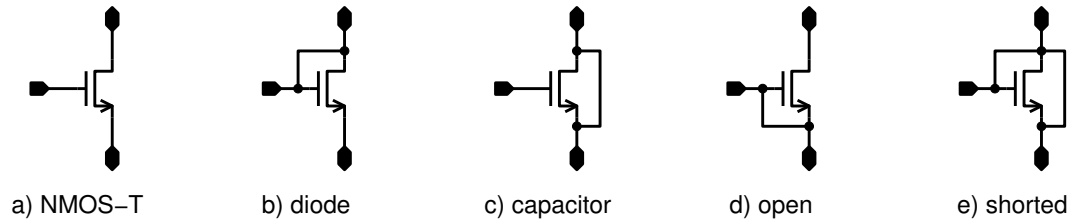


Figure 3.3: Different configurations of a single transistor.

involve further parasitic capacitances. Thus, a tradeoff between the variety of routing capabilities and the unwanted parasitic effects has to be found. Second, additional routing channels also increase the die area necessary to implement the PTA in[7]. Apart from the increase in production costs, this also deteriorates the analog properties of any circuit implemented by means of the PTA. Finally, being the first prototype of its kind, the chip was to provide a concept of moderate complexity facilitating not only its design, but also its usage in the evolution system as well as the understanding of any of circuits evolved on it.

3.3.3 Programmable Transistor

The ideal *programmable transistor* is to provide a quasi-continuous variety of transistor gate dimensions typically used in analog designs without affecting the integrity of the analog signals by any parasitic effects whilst occupying as little die area as possible. In reality, the first and the latter two objectives must be traded off against each other. In the proposed implementation, the selectable gate lengths are (almost) logarithmically spaced and can take on the values $0.6, 1, 2, 4, 8\,\mu m$ while the gate widths can be chosen from 15 linearly spaced values between 1 and $15\,\mu m$. While most analog circuits can indeed be designed from a limited set of few different transistor lengths covered by the selection offered by the programmable transistor (see e.g. the op amp described in section 3.5, the restriction to transistor widths $W \leq 15\,\mu m$ precludes output stages capable of driving large output impedances. However, as the outputs of the transistor array will typically be sampled by unity-gain buffers, this shortcoming does restrict the possible design space, yet does not limit the set of signal processing applications that can be implemented on the PTA.

The above concept is realized by a matrix of 20 single transistors depicted in Fig. 3.4 for the case of an NMOS transistor cell. Each of the five rows contains four transistors with the same gate length, but varying gate widths. Owing to a 3-bit decoder, exactly one row corresponding to one transistor length can thus be chosen by means of a three bit code. Any combination of the four transistors of one row can be selected in parallel. Since the transistor widths correspond to the first four powers of 2, this allows to choose any integer transistor width between 0 and $15\,\mu m$. Although this concept greatly reduces the number of necessary transistors for the proposed variety of selectable transistor widths, a combined transistor will behave slightly different than a single transistor with the same nominal length. This is due to the fact, that the difference between the nominal (drawn) and the effective (fabricated) transistor widths do not scale linearly. Yet, compared to the deterioration caused by the configuration overhead discussed in section 3.3.5, this effect is assumed to be of negligible impact.

While all of the source and drain terminals of the 20 single transistors are directly connected to form the generic terminals TD and TS, respectively, the selection of the desired transistors is restricted

[7] The question by how much additional routing resources add to the required die area depends on the chosen fabrication process. The effect will be less severe for more advanced technologies that provide more metal layers and allow stacking of the via connections between these metal layers. The design rules of the chosen production process are documented in [Aus97b].

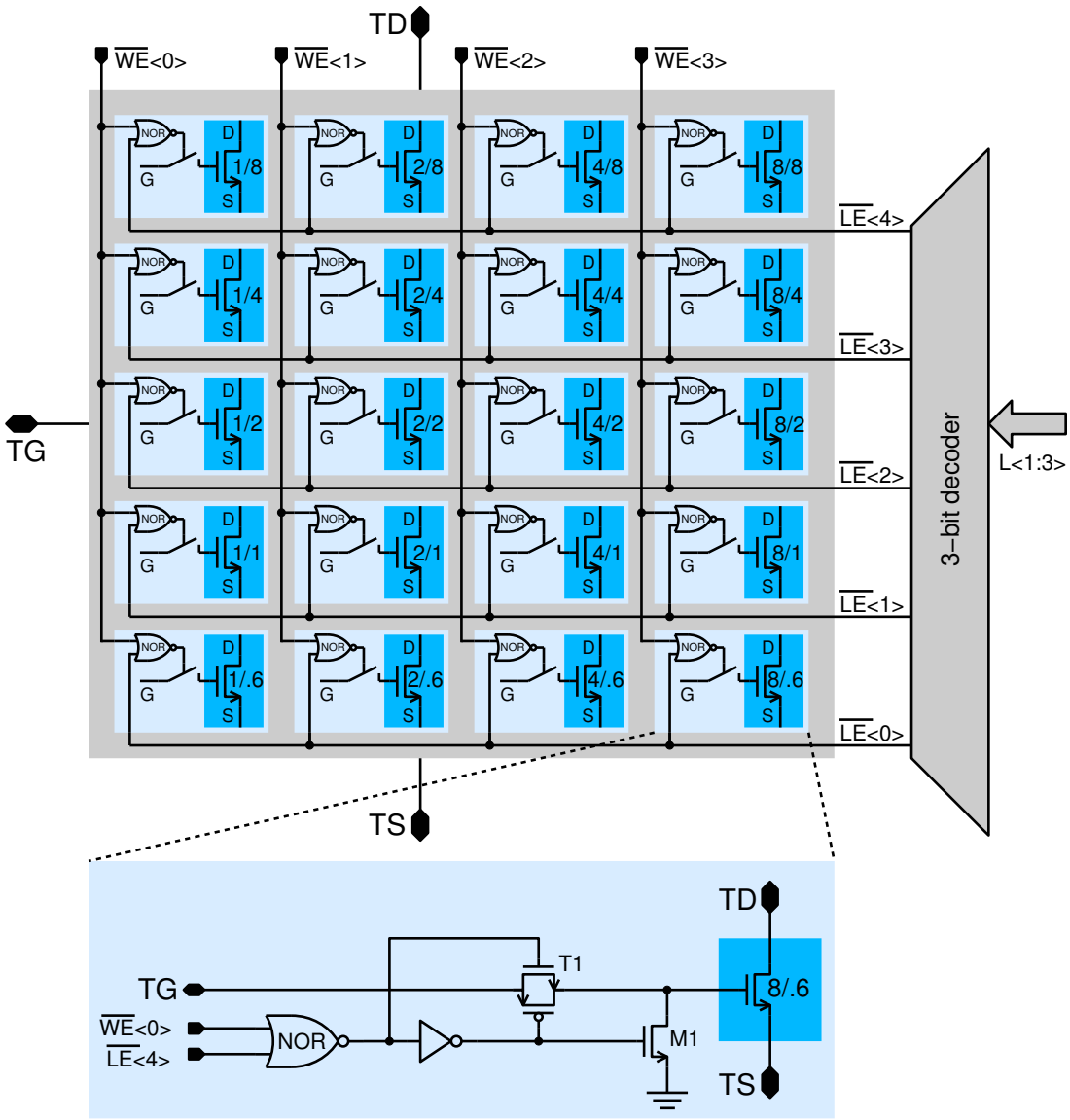


Figure 3.4: Gate selection for a programmable NMOS transistor. The transistor terminals D and S of all 20 transistors are connected to one generic drain TD and source TS terminal, respectively. The generic gate terminal TG is only connected to the gates of the selected transistors. The circuit shown at the bottom of the figure depicts the circuitry contained in the lower right sub-cell of the array of selectable transistors displayed in the upper part of the figure.

to connecting the respective gates to the generic gate terminal TG. The required selection circuitry is depicted in the cut-out in the lower fourth of Fig. 3.4: If the transistor in question is selected, the transmission gate T_1 connects its gate to TG and transistor M_1 is turned off. If, on the other hand, the transistor is not selected, the transmission gate is opened and transistor M_1 ties the gate of the actual transistor to gnd, thus effectively shutting it off. In case of a PMOS transistor cell, the matrix contains 20 PMOS transistors. Transistor M_1 is then also realized as a PMOS transistor and ties the transistor gate to vdd if the transistor is not selected, which again shuts it off. Although the gain factor for the NMOS transistors K'_N is three times larger than its PMOS counterpart K'_P [Aus97c], the arguments for symmetric N- and PMOS cells outweighed those for a set of channel dimensions that

counterbalance this discrepancy in gain factors. The necessary adaptation of the aspect ratios of P- and NMOS transistors has to be accomplished by the optimization process. In conventional design this would imply that the largest aspect ratios of NMOS transistor cells may remain unused in some of the circuit at hand.

The proposed selection mechanism is superior to its straight-forward alternative of switching all three transistor terminals D, G, and S in several ways: First, all of the inactive transistors are forced into a well-defined condition instead of being left floating. Second, the number of necessary transmission gates is greatly reduced. In particular, the transmission gates that would be required to switch the drain and source terminals would not only insert parasitic on-resistances into the current paths, but also – due to their required size – considerably increase the amount of parasitic capacitances added to the generic terminals TD and TS. The transmission gates T_1 connecting the gate terminals can be of relatively small dimensions, since they only need to pass the currents required to charge the transistor gates; apart from negligible leakage currents, no static current is drawn.

3.3.4 Switch Dimensions

Apart from the topology of the programmable transistor cell, the chosen channel dimensions of the transistors constituting the transmission gates used as switches have to be determined. As to-date, no perfect switch was invented for highly integrated production processes (cf. [Edw01], [Kim03][8]) that combine negligible on-resistance with low capacity and small area consumption, any chosen set of transistor dimensions has to trade off these conflicting goals against each other. The set of transistor dimensions chosen for the proposed FPTA implementation are summarized in Table 3.2.

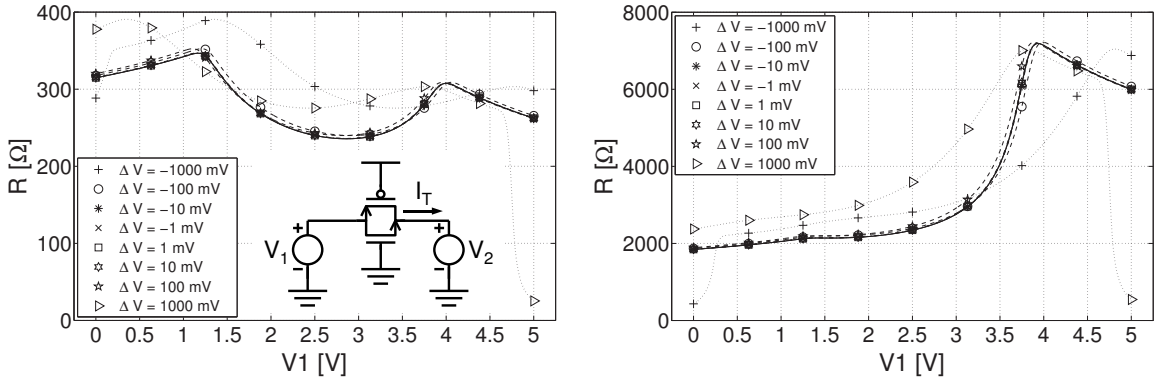
Task	$W_P[\mu m]$	$W_N[\mu m]$	$L_P[\mu m]$	$L_N[\mu m]$
routing	30	10	0.6	0.6
TD, TS multiplexer	30	10	0.6	0.6
TG multiplexer	1.4	1.4	0.6	0.6
programmable transistor selection	1.4	1.4	0.6	0.6

Table 3.2: Transistor dimensions of the different transmission gates used as switches to configure the programmable transistor cell. The subscripts 'P' and 'N' denote the P- and NMOS devices, respectively.

Since the six switches used for connecting the four border terminals with each other as well as those used in the analog multiplexers that connect the generic transistor terminals TD and TS to the border terminals are possibly located within the current conducting path of the prospective circuits, they employ fairly large transistors whose channel widths amount to 30 and 10 µm for the N- and PMOS transistors, respectively. The ratio of W_P and W_N is set to the inverse of the ratios of the respective gain factors K'_P and K'_N documented in [Aus97c] in order to counterbalance the different current gains of P- and NMOS transistors. As a result, the resistance of these switches is almost independent of the common mode voltage, which is depicted in Fig. 3.5(a). On one hand, this flat response makes the on-resistance more amenable to understanding and modeling, thus facilitating the analysis of evolved circuits. On the other hand, one may object, that the PMOS transistors of the switches are designed too large with respect to the programmable PMOS transistors, whose channel dimensions are identical to their NMOS counterparts. However, the design follows the former argument.

Although the switch dimensions are fairly large in terms of the occupied silicon area ($5 \times 10\,\mu m^2$) and their parasitic capacitances (see section 3.3.5 below), their channel width is exceeded by the

[8] The mercury droplet microswitches proposed here are too bulky (diameter of 200 µm), too slow (switching times of approximately 1 ms) and require excessive programming voltages (80 V and beyond).



(a) large transmission gate

(b) small transmission gate

Figure 3.5: Simulation results for the resistance of the large (a) and small (b) transmission gate as a function of the common mode voltage defined by V_1. The testbench used for the dc simulations are shown in the inset of (a). The simulation is performed using a SpectreS simulator [Cadb] in conjunction with the BSIM3.3 transistor model [Lak94b]. For each value of ΔV voltage V_1 is swept from 0 to 5 V while voltage V_2 is forced to $V_1 + \Delta V$. The resistance is then calculated by $R = \frac{\Delta V}{I_T}$.

largest programmable transistors for the NMOS type. In effect, the on-resistance of these switches will have a considerable influence on the circuit behavior, if the largest possible aspect ratios are used in conjunction with large gate-source voltages V_{GS}, which again defers the evolution of output stages suited for large output loads. Nevertheless, the voltage drop across a closed switch with an average on-resistance of approximately 300 Ω (inferred from Fig. 3.5(a)) remains below 0.1 V for up to 300 µA.

As the switches used for the gate selection in the programmable transistor as well as those used for multiplexing the generic gate terminal TG are to switch signals onto CMOS gates, they do not have to conduct any static current. Therefore, the width of both, N- and PMOS transistors of these transmission gates, are chosen to minimize the occupied silicon area[9]. Since the on-resistance of those switches only affects the time in which the gate capacity of the programmable transistor can be charged, an on-resistance independent of the common mode voltage is not required in this case. The limitation of the signal bandwidth caused by the on-resistance can be estimated by

$$f_{\text{-3 dB}} = \frac{1}{2\pi R_{\text{on,max}} C_{\text{gate,max}}} = \frac{1}{2\pi \cdot 7.2\,\text{k}\Omega \cdot 331\,\text{fF}} = 66.7\,\text{MHz} \tag{3.1}$$

3.3.5 Parasitic Devices

While the finite on-resistance of the switches has already been discussed in the previous section, this section shall discuss their parasitic capacitances as well as the capacitances introduced by the metal lines carrying the electrical signals. The resistance of these metal lines can be omitted as it is orders of magnitude smaller than the on-resistance of the configuration switches. The parasitic capacities of the transmission gates are calculated according to the discussion of sections 1.1.3 and 1.3.1. The parasitic capacitances of the metal lines are attained by extracting the layout of the NMOS transistor cell using the *capall* option (see [Cada], [Aus97d]). Although this extraction process yields capacities between all pairs of neighboring layers, all these capacities are lumped into one overall

[9] According to the design rules [Aus97b] of the FPTA's fabrication process the diffusion width of the gate and source terminals must not be less than 1.4 µm. A smaller gate width will narrow the area occupied by the necessary overlap of the gate's silicon area, albeit increases the overall length of the transistor.

capacitance between the respective signal and ground. The parasitic capacitances encountered in the programmable transistor cell are discussed first, followed by an analysis of those parasitic devices formed by the rest of the transistor cell excluding the programmable transistor itself. Finally, the results are merged to develop one small signal equivalent circuit for the complete transistor cell.

3.3.5.1 Programmable Transistor

The most important parasitic devices entailed in the auxiliary circuitry providing the configurability of the programmable transistor are depicted in Fig. 3.6. The small signal equivalent circuit is based on the following scenario: Exactly one – 0 to 4 are conceivable – of the array of 20 selectable transistors M_S is selected. The closed transmission gate connecting the generic drain terminal TD and the particular gate G of the selected transistor is replaced by its equivalent circuit as described in section 1.3.1 (the assignment of source and drain is not a priori clear and therefore defined arbitrarily). In addition, the node TD experiences the parasitic capacitances C_{SB} and C_{GS} of the 19 open switches T_1. To the right hand side of the closed transmission gate T_1 only the gate node of the selected transistor is shown. Accordingly, only the open transistor M_1 connected to this particular gate is included in Fig. 3.6 by its equivalent circuit, which follows the description of section 1.7. In contrast, the generic drain and source terminals TD and TS are shared among the selected transistor M_S and the 19 unselected transistors M_S which are in the off-state; the latter ones are displayed to the right of the selected transistor. As the capacitances inherent to the selected transistor are not contributed by the auxiliary circuitry, they must not be considered here. Therefore, drain and source terminals can be treated in the same way: Their capacitances differ only due to the bulk-source/drain dependency on the respective terminal voltages.

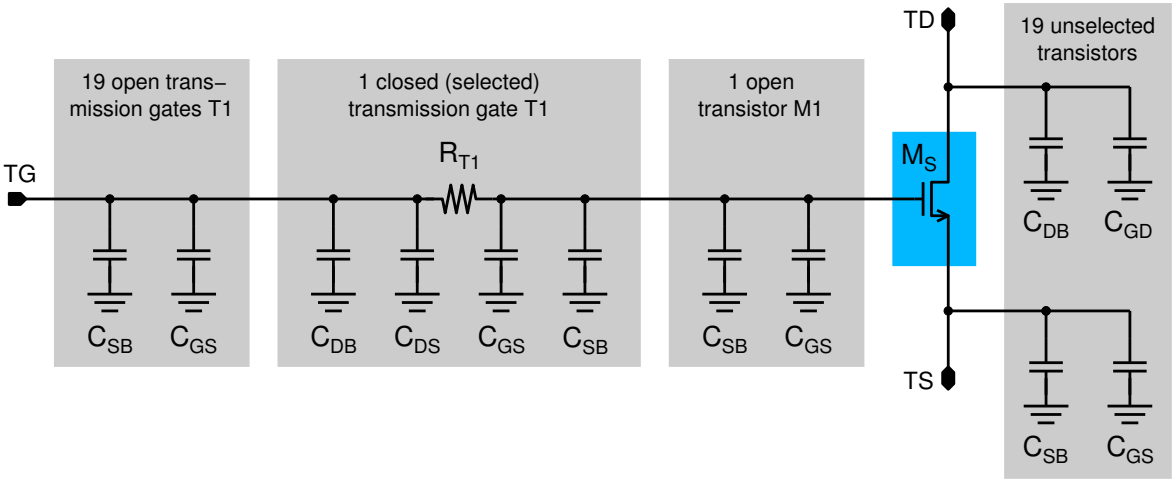


Figure 3.6: Small signal circuit equivalent of one programmable transistor regarding all parasitic devices introduced by the configuration circuitry. The capacitance values are calculated in Table 3.3.

Table 3.3 provides the necessary equations used to calculate all the parasitic capacitances illustrated in Fig. 3.6 and summarizes their numerical values. Its four meta-rows if read from top to bottom correspond to the four gray-shaded boxes of Fig. 3.6 when read from left to right. The formulas for C_{GS} are based on (1.4) and (1.5), respectively; the term describing the operation region at hand is already selected. To account for the source-bulk capacitance's dependency on the actual terminal voltages the minimum and maximum values for C_{SB} are calculated according to (1.3). Thus, the actual value of C_{SB} is bound by these minimum and maximum values. Since it is not a priori clear, which transistors are to be shut down and which are going to be selected, the parasitic capacitances

Device	Capa-citor	Formula[a,b]	W [µm]	C_{SB}[c], C_{GS} separately		C_{SB}[c] + C_{GS}	
				Min [fF]	Max [fF]	Min [fF]	Max [fF]
19 T1 off	C_{SB}	$19(C_{SB,P}+C_{SB,N})\big\vert_{off}$	1.4	42.59	95.39	60.68	113.48
	C_{GS}	$19W_{T1}C_{GSDO}$	1.4	18.09	18.09		
T1 on	C_{SB}	$(C_{SB,P}+C_{SB,N})\big\vert_{sat}$	1.4	2.24	5.02	4.95	7.73
	C_{GS}	$2\cdot 1/2C_{ox}W_{T1}L_{T1}$[d]	1.4	2.70	2.70		
M_1 off	C_{SB}	$C_{SB,N}\vert_{off}$[e]	1.4	1.00	2.42	1.48	2.90
	C_{GS}	$W_{M_1}C_{GSDO}$	1.4	0.48	0.48		
20 M_S off[f]	C_{SB}	$C_{SB,N}\vert_{off}$[e]	75[g]	36.63	87.59	62.13	113.09
	C_{GS}	$W_{M_S}C_{GSDO}$	75[g]	25.50	25.50		

[a] C_{SB} is calculated from (1.3).
[b] The additional subscripts 'N' and 'P' indicate whether the process parameters for P- or NMOS transistors have to be chosen.
[c] Minimum and maximum values account for C_{SB}'s dependency on the transistor's gate and drain potential described by (1.3).
[d] Widths and lengths for the P- and NMOS transistors of transmission gate T1 are identical (see Table 3.2).
[e] In case of M_1 and M_S the values for C_{SB} are calculated for the NMOS case only. The capacitance values are slightly higher for PMOS transistors.
[f] Since in case of the array of programmable transistors itself it is not a priori clear which transistors will be open, all 20 transistors are considered here to overestimate the worst case.
[g] The widths of the 20 transistors of M_S are summarized to one width by $5\cdot 15 = 75\,\mu m$. Yet, the computation of C_{SB} includes the correct circumference of the diffusion area of the source and drain terminals.

Table 3.3: Table of parasitic capacitors introduced by the auxiliary circuitry providing the configurability of the programmable transistor. The location of the capacitors listed here are depicted in Fig. 3.6. The diffusion lengths of the source and drain terminals of all transistors are assumed to be minimal, i.e. $L_{diff} = 1.4\,\mu m$ according to the design rules [Aus97b]. All capacitor values are calculated from [Aus97c] for the typical mean case.

at the generic drain and source terminals TD and TS are calculated for all 20 selectable transistors being shut off. As C_{SB} and C_{GS} of the selected transistor are inherent to any transistor and not caused by the configurability, these capacitances are thus overestimated. The last two columns of Table 3.3 aggregate the respective C_{SB} and C_{GS} values of the according nodes.

A closer look at the data compiled in Table 3.3 yields the following insights. First, C_{SB} exceeds C_{GS} by a factor of 1.5 to 5 for those transistors that are turned off. Thus, the parasitic capacitances introduced by switches in the off-state can be reduced by choosing layout structures that minimize the source and drain diffusion areas as, for instance, achieved by finger structures. Second, the total capacitances present at each of the three generic transistor terminals TD, TG and TS are almost identical. Again, this is achieved by the turning of the selectable transistors M_S themselves. If the drain and source terminals of all 20 transistors were also switched by transmission gates, these had to be larger than those for the gate terminals for at least some of those 20 transistors. Hence, the resulting parasitic capacitances would exceed those introduced at the generic gate terminal TD. Third, the parasitic capacitances at the gate terminal of the selected transistor, which are introduced by M_1

and the right hand side of the closed transmission gate T_1 are orders of magnitude smaller than those at the generic gate terminal and can thus be neglected.

3.3.5.2 Transistor Cell without the Programmable Transistor

In this subsection, only those parasitic devices shall be discussed that are entailed in those parts of the transistor cell that do not belong to the programmable transistor. The scenario is illustrated in Fig. 3.7, where the programmable transistor itself is included as a black box without any parasitic properties. According to Fig. 3.2, the configuration of the transistor cell is achieved exclusively by using transmission gates as analog switches. In order to attain an insight into the effects of the configuration circuitry, these transmission gates are replaced by their small signal circuit equivalents proposed in section 1.3.1. Since the illustration of Fig. 3.7 takes on the perspective of the nodes themselves, the equivalent circuits cover only half of the transmission gate relevant to the respective node. They are a split version of Fig. 1.7; the difference between the on- and off-state is indicated by an ideal switch. As summarized in Table 3.2, only two types of transmission gates are employed, large ones featuring $W_P = 30\,\mu\text{m}$ and $W_N = 10\,\mu\text{m}$ and small ones used for multiplexing the generic gate terminal TG featuring only one single gate width $W_P = W_N = 1.4\,\mu\text{m}$.

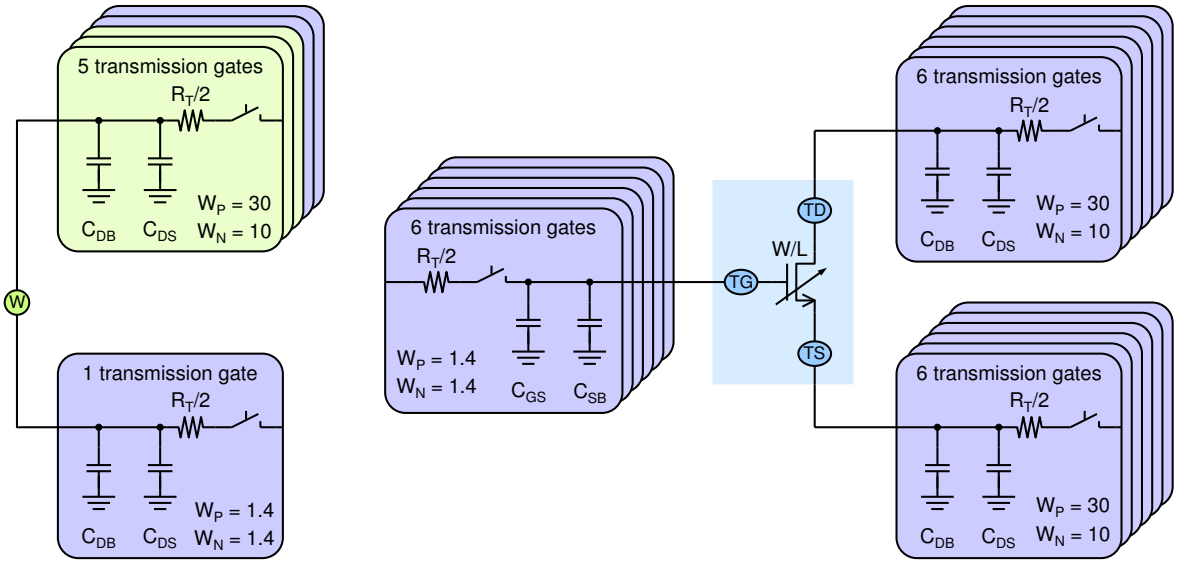


Figure 3.7: Small signal equivalent of the nodes W, TD, TG and TS in the programmable transistor cell due to its transmission gates. The switches are ideal; the on-resistance of the transmission gates is divided in to equal halves. The parasitics of the programmable transistor are omitted.

The right part of Fig. 3.7 accounts for the three analog multiplexers responsible for connecting the generic transistor terminals TD, TS and TG to the array power or either of the four cell borders N,W,S or E (cf. Fig. 3.2). The transistor cell's border terminals, on the other hand, are connected to 3 routing switches, and 3 transmission gates belonging to the analog multiplexers. Of the latter ones, two are large transmission gates and the third one, possibly connecting to the generic gate terminal TG of the programmable transistor, is kept smaller. The situation at the border cell terminals is exemplified on the left hand side of Fig. 3.7 for the *West* terminal.

The node capacitances of the two types of transmission gates – henceforth denoted as *large* and *small* – are compiled in Table 3.4. The calculation is based on (1.8), that is the contribution of C_{SB} and C_{GS} are already lumped together for both transistor types. Thereby, the transistors constituting the transmission gates are assumed to be either in the off-state or to be operated in the linear region.

Transmission gate	$W_P[\mu m]$	$W_N[\mu m]$	open		closed	
			Min[fF]	Max[fF]	Min[fF]	Max[fF]
TG multiplexer	1.4	1.4	3.19	5.97	4.56	7.34
TS/TD multiplexer	30	10	30.55	52.02	50.07	71.54
routing switches						

Table 3.4: Minimum and maximum parasitic capacitances for the different types of transmission gates used in the programmable transistor cell. The respective values are calculated according to (1.8) for both, the open and the closed configuration of the switch. For all transistors the length of the source/drain diffusion is assumed to be of minimum size, i.e. $L_{\text{diff}} = 1.4\,\mu m$.

In fact, for closed switches, the transistors will only reach the saturation region in two cases: Either for the pathological case of very large voltage drops across the switch, in which the transistor cell will be far from acting like a single transistor anyway, or for one of the two transistors, whenever the switch is operated close to either of the power supply rails. Apart from these exceptions, both nodes of the switch N_1, N_2 can be treated in the same way. In Table 3.4 the node capacitances C_{N_x} are reported in Table 3.4 for open and closed switches. The influence of the actual terminal voltages on C_{SB} are again covered by calculating the range of possible capacitance values. The capacitance values for the large transmission gates already take into account that their PMOS transistors are laid out using a finger structure, which slightly reduces their source-bulk capacitance.

The results of Table 3.4 are used to calculate the parasitic capacitances of the four cell border terminals N, W, S, E and the three generic transistor terminals TD, TG, TS except for those caused by the programmable transistor cell which were discussed in the last subsection. The results are summarized in Table 3.5. The number of *small* and *large* switches that have to be considered correspond to the situation depicted in Fig. 3.7 and are repeated in the third and fourth column of Table 3.5. The last four columns are calculated by adding the capacitances introduced by the respective switches

Node	C_{ML}[fF] for metal lines	# small switches	# large switches	open		closed[a]	
				Min[fF]	Max[fF]	Min[fF]	Max[fF]
TG	188.2	6	0	207.36	224.03	208.73	225.40
TD	419.9	0	6	603.18	732.04	622.70	751.56
TS	360.2	0	6	543.48	672.34	563.00	691.86
N	439.1	1	5	595.03	705.19	693.99	804.16
W	531.8	1	5	687.73	797.89	786.69	896.86
S	417.7	1	5	573.63	683.79	672.59	782.76
E	425.1	1	5	581.03	691.19	679.99	790.16

[a] In case of the programmable transistor terminals TG, TD and TS that are connected to the switches of the analog multiplexer, only one switch can be closed at a time. For the border terminals N, W, S and E all shared switches are assumed to be closed (which is probably not advisable for useful circuits).

Table 3.5: Parasitic capacitances inherent to the border terminals as well as for the generic transistor terminals of the transistor cell. For the generic terminals of the programmable transistor only those switches contained in Fig. 3.2 are considered, that is, not the internal parasitic capacitances of the programmable transistor that are presented in Table 3.3.

counted in columns three and four to the capacitances of the according metal lines C_{ML}. The C_{ML} are attained from the aforementioned extraction procedure that lumps all of the capacitances between the node at hand to all other layers into one grounded capacitor. In case of the three generic transistor terminals, the complete C_{ML} is exclusively considered in Table 3.5 although it also belongs to the programmable transistor cell. The last two columns account for the situation in which as many switches are closed as possible. In case of the three generic transistor terminals TD, TG, TS this is only one of the six switches belonging to the analog multiplexers. In case of the cell border terminals N, W, S, E all switches are assumed to be closed, because it is theoretically possible, even though improbable to find in practical circuits. Nevertheless, a connection to all three remaining border terminals and two generic transistor terminals – e.g. to form a capacitor or diode connected configuration – is conceivable.

The data compiled in Table 3.5 lends itself to a couple of observations: First, a comparison with the results in Table 3.3 yields that the capacitances inherent to the configuration switches of the programmable transistor are smaller by roughly a factor of four for all current conducting nodes. They are also smaller than the capacitances caused by the metal wires by a factor of 4 (2 in case of TD). Second, the capacitance of the physical realization C_{ML} dominates the overall capacitance of each of the six current conducting nodes considered here in that it exceeds the sum of capacitances introduced by the according switches by a factor between 1.1 and 3.5 depending on configuration, terminal voltage and node. In case of the generic gate TD, C_{ML}'s contribution is even more dominating. A comparison of the maximum values, on the other hand, together with the inclusion of the capacitances attained for the programmable transistor listed in Table 3.3, reveals that metal line and switch capacitances are nicely balanced in this case: On one hand, reducing the switch size to the absolute minimum could reduce the total capacitance by a factor of two at most. Larger switches, on the other hand, inevitably lead to larger parasitic switch capacitances and entail larger C_{ML} due to the increase in die area. Finally, from a comparison of C_{ML} for TD and all other nodes the influence of the width of the used metal lines can be estimated: As explained in section 3.3.6 all current conducting nodes are laid out fairly wide to protect them against self-destruction. Since this is not the case for the generic gate TD, it can be inferred that the metal capacitance of all other nodes could be reduced to approximately $150\ldots200\,\mathrm{fF}$ if they were narrowed down to widths close to the minimum size. Hence, abandoning this precaution against self-destruction would reduce the overall node capacitances by approximately 30%, which was not found worthwhile running the increased risk.

3.3.5.3 Small Signal Equivalent Circuit for the Complete Transistor Cell

The results of the above analyses are assembled into one small signal circuit equivalent for the analog signal path of one complete transistor cell, which is shown in Fig. 3.8. The node capacitances of the four cell border terminals N, E, S and W are directly taken from Table 3.5, those of the generic transistor terminals TD, TG and TS are calculated from the values listed in Table 3.5 and 3.3. The node capacitance at the node of the selected transistor M_S is taken from Table 3.3. For all of the parasitic capacitances the absolute minimum and maximum values including the variation of terminal voltages affecting C_{SG} as well as the state of the switches are added to the schematic. The two types of resistors R_1 and R_2 refer to the small and large transmission gates. The range of possible resistance values is read off from Fig. 3.5. All of the switches are assumed to be ideal, that is free of further parasitic effects.

In general, the particular combination of resistors and capacitors that limits the maximum achievable bandwidth will depend on the circuit implemented on the programmable transistor array: Due to the necessary signal routing, an analog signal will usually have to traverse through a few to tens of R_2 resistors, before it reaches another gate terminal. Yet, R_1 exceeds R_2 by a factor of 6 to 24, and C_{TG}

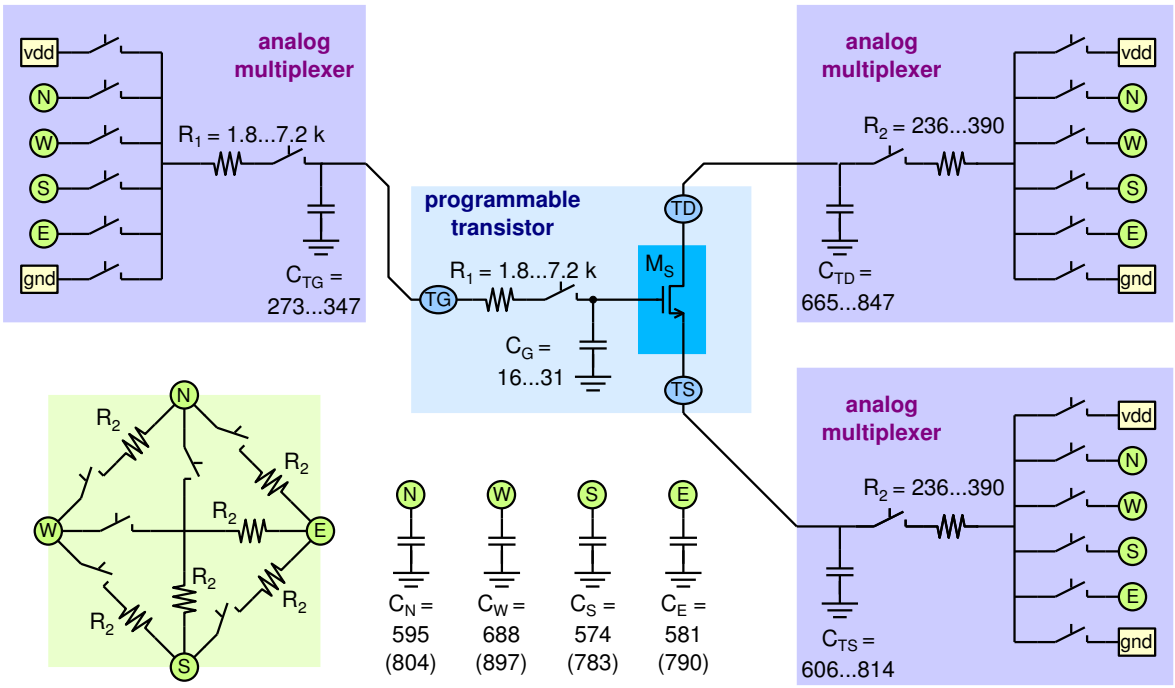


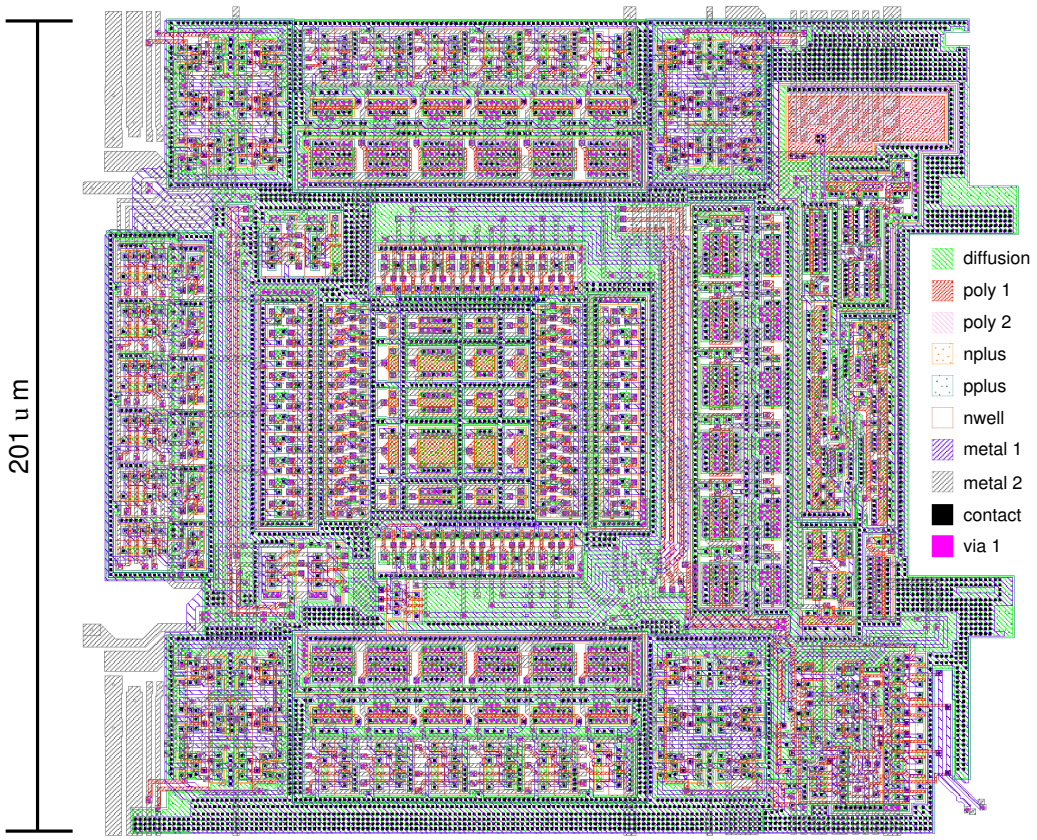
Figure 3.8: Small signal equivalent circuit model of the complete programmable transistor cell. All switches are assumed to be ideal. The resistor and capacitor values are understood to be quantified in Ω and fF, respectively.

amounts to almost half of all other node capacitances. Consequently, for designs that get along with short signal paths (in terms of traversed transistor cells), the pole formed by R_1 and C_{TG} is likely to be the dominating one. Its corner frequency is calculated analogous to (3.1) yielding $f_{\text{-3 dB}} = 63.7\,\mathrm{MHz}$. Although this is similar to the result in (3.1) attained for the pole introduced by R_1 and the largest possible gate available for M_S, the latter pole is assumed to be less critical, because time-critical designs are not expected to rely on extremely long transistors with $L \geq 4\,\mu\mathrm{m}$. As an increase in the transistor width of the 20 transmission gates T_1 belonging to the programmable transistor is more expensive in terms of silicon area than increasing the size of the six switches in the generic gate multiplexer, the transistor cell design could be improved by a moderate resizing of the gate multiplexer. A viable compromise between the increase in parasitic capacitances and area and a reduced on-resistance of the switches of the gate multiplexer is expected to be found in the range of $W_P = 3$ and $W_N = 2\ldots3\,\mu\mathrm{m}$.

3.3.6 Layout of the Programmable Transistor Cell

Finally, the layout of one complete transistor cell shall be discussed. Layout and floorplan of one NMOS transistor cell are illustrated in Fig. 3.9. Since the two types of transistor cells merely differ in their respective programmable transistor, their layouts are almost identical in general and completely identical at their boundaries. N- and PMOS transistor cells can be lined up in X- and Y direction with a pitch of 198.2 and 201.15 µm, respectively.

Overview of the Floorplan. The programmable transistor is located in the center of the layout of the transistor cell. The selection circuitry necessary to select the desired M_S, that is, the NOR gate, the transmission gate T_1, and the NMOS transistor M_1 depicted in Fig. 3.4 are arranged around the field of the 20 selectable transistors M_S. The layout of the programmable transistor is routing limited



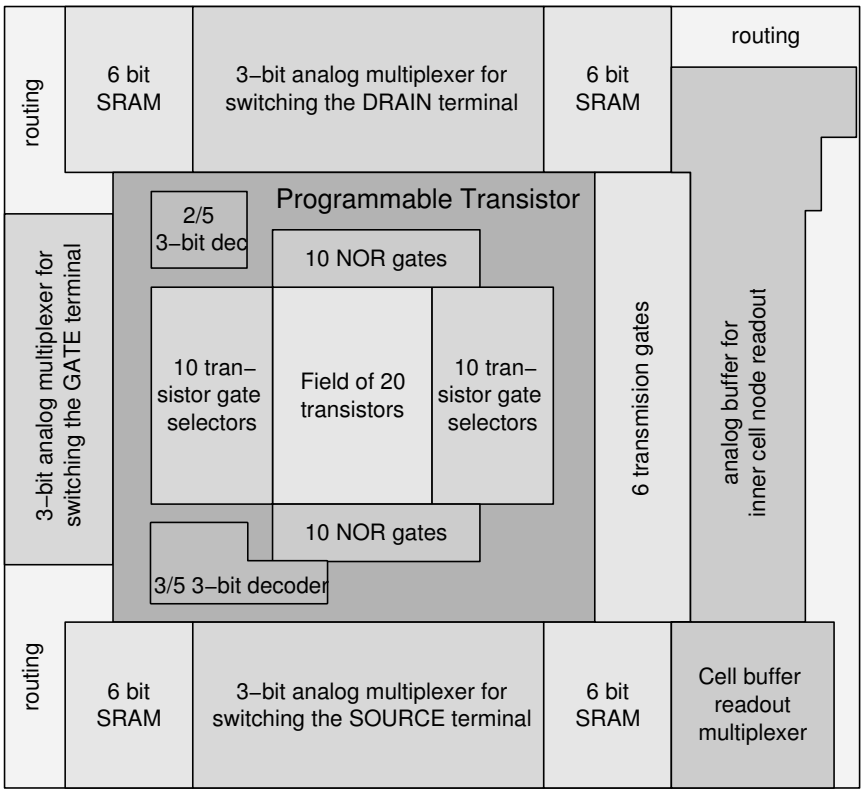


Figure 3.9: Layout (top) and floorplan (bottom) of one complete NMOS transistor cell.

in that some of its area is dedicated exclusively to host the necessary metal and polysilicon lines. The multiplexers for the generic transistor terminals TG, TD and TS are placed next to the upper, left and lower border of the programmable transistor, respectively. Note, that the gate multiplexer occupies only $2/3$ of the area of the source and drain multiplexers, due to its smaller transmission gates. The four blocks of SRAM are placed at the four corners of the necessary transistor cell ingredients described above. On the right hand side of the cell the buffer for probing inner-cell signals together with the necessary selection circuitry is added. The global signals, as e.g. the BIT and WORD lines, power and array power (vdd, gnd, vdda, gnda), and the selection lines for the cell buffer, V<0:1> and H<0:1> are routed along the four edges of the transistor cell.

Layout: Style and Concept. It is impossible to quantify the area efficiency of the proposed layout. However, a few comments on its style and compactness seem to be in place: First, it was indeed attempted to keep the occupied silicon area small. Yet, the result largely depends on the particular placing of the different components, which severely affects the necessary routing. The following arguments strongly depend on the given placement being reasonably area-efficient. Second, the layout carefully supports the electrical functionality in two regards: On one hand, extra-wide metal lines are used for the current conducting nodes, as e.g. the border terminals, the generic source and drain terminals, and all power supply connections to avoid self-destruction due to electromigration. On the other hand, substrate and n-well contacts as well as guard ring structures are generously spent to properly isolate the different subsystems from each other. Third, the whole design is routing limited in that there is little to no space left for additional routing in either of the three metal layers. This puts the excess area introduced by the inner-cell node probing buffer into perspective, because it does not extend the width of the transistor cell by the full 30 µm of its own width: Since the global signals must cross at each corner, the vertical signal routing could be distributed only to one additional metal 1 layer without the buffer, albeit would still occupy a considerable amount of the then free area. However, taking into account the three additional signals V<0:1> and COUT required by the cell buffer, it becomes clear, that the inner-cell probing does not come entirely for free. Finally, as the miniaturization of the transistor cell layout is limited by the necessary routing, more advanced fabrication technologies providing six or more metal layers and allowing for stacking of vias that connect the different metal layers is expected to foster more compact transistor cell layouts in second generation FPTA cells.

Layout Sizes and Configuration Overhead. In addition to the floorplan drawn to scale, the dimensions of the components of the transistor cell are summarized in Table 3.6. Apart from width X, length Y and area A of the respective components, Table 3.6 also calculates the area percentage of the entire transistor cell. To further quantify the configuration overhead inherent to the transistor cell, the area of each of the components is also calculated in units of the largest transistor that can be realized featuring $W = 15\,\mu\text{m}$ and $L = 8\,\mu\text{m}$. Accordingly, the entire cell exceeds the area occupied by this largest feasible transistor by a factor of 228 and is still 18 times bigger than the array of the 20 selectable transistors. In other words, the remaining 94% of silicon area are only used for the configuration.

The programmable transistor already makes up for roughly a third of the transistor cell. The 24 bits of SRAM use up a modest 13.4 % while the three analog multiplexers together with the routing switches demand almost another third of the available space. Although the inner-cell-probing facilities are listed with an area consumption of 15 %, the effective increase in cell area is probably considerably smaller as discussed above.

Device	X[µm]	Y[µm]	A[µm^2]	A in % of the whole cell	$A[A(M_{15/8})]$
Transistor cell	198.2	201.15	39,868	100	227.87
Largest Transistor (15/8)	10.80	16.20	175	0.44	1.0
Field of 20 transistors	42	54	2241	5.62	12.81
Programmable transistor	117	109	12,755	31.99	72.9
10 transistor selectors	27	58	1541	3.86	8.81
6 routing switches	25	102	2570	6.45	14.69
6-pack of SRAM bits	32	42	1335	3.35	7.63
24 SRAM bits	—	—	5340	13.4	30.52
3-bit TG multiplexer	28	86	2402	6.02	13.73
3-bit TD/TS multiplexer	88	42	3713	9.31	21.22
Cell buffer	30	140	4200	10.53	24.01
Cell buffer multiplexer	41	47	1921	4.82	10.98
Cell buffer unit	28–41	187	6121	15.35	34.99

Table 3.6: Absolute and relative sizes and area consumption of the different subcircuits constituting one complete NMOS transistor cell.

Protection against Self-Destruction. According to Chakraborty and Pinaki [Cha02], the two degradation processes that apply to the transistor cell level are electro- and stress migration. While the first phenomenon describes the deterioration of metal lines caused by excessively high current densities, the second one similarly affects the metalization layers but is the result of mismatches in thermal expansion coefficients and excessive thermal stress. While the latter hazard may be avoided by triggering a shutdown of the FPTA depending on the die temperature, the problem of electromigration is prevented by an extra-wide layout of the metal lines realizing the current conducting nodes.

For the given selectable transistor dimensions and the maximum saturation currents defined by the process technology [Aus97c], the maximum current through an N- and PMOS transistor, if short-circuited between the power supplies, amounts to 7.95 and 4.05 µA, respectively. However, in the transistor cell, the current has to traverse at least one switch to get from either of the power supply voltages to the drain or source terminal. Thus, the maximum current flowing through one transistor is limited to approximately 400 to 500 mA for an NMOS cell. In order to comply with the maximum dc-currents allowed by the manufacturer [Aus97c], the metal lines for the generic drain/source terminals TD and TS are laid out 4 µm wide. On one hand, the currents of a large number of other transistor cells may be collected at the cell border terminals. On the other hand, these currents are attenuated by all of the transmission gates they have to pass. Since an exact estimate of the worst case currents that have to be taken into account (with a realistic probability of occurrence) is difficult to obtain, the 4 µm used for TD and TS are doubled to allow for a sufficient safety margin.

3.4 SRAM for Configuration Storage

In principle, three types of memory could be used to store the configuration of the 256 transistor cells: EEPROM[10] based on floating gate technology, DRAM or SRAM (cf. Rhein and Freitag [Rhe92]). The target specifications formulated in section3.1.2, namely to allow for a fast reconfigu-

[10] Electrically Erasable and Programmable Read Only Memory

ration ($t_{\text{config}} \leq 100\,\mu\text{s}$) for a nearly unlimited number of times (at least hundreds of billions), rule out the use EEPROM [Rhe92]. The use of DRAM, on the other hand, requires a more elaborate control logic for the inevitable refresh, which also deteriorates the adjacent analog signals by crosstalk and substrate coupling effects. Yet its main advantage, the small area consumption, would be used up by the line drivers necessary to drive the configured decoders and switches. In contrast, SRAM meets all these constraints and is therefore considered the technology of choice, which also manifests itself in its widespread usage in FPGAs, whose requirements are similar to those of the FPTA.

3.4.1 SRAM Cell

The SRAM cell itself is designed as a six transistor (6-T) cell, which is discussed in a variety of text books on CMOS design, e.g. [Gei90a], [Pre01] or [Mil87a]. Its implementation including transistor dimensions is depicted in Fig. 3.10. The latch formed by the transistors M_1, M_2, M_5 and M_6 stores the information bit, while the pass transistors M_3 and M_4 allow to read out and overwrite the state of the latch. In the proposed FPTA chip, the latch is also used to drive the decoders and switches directly connected to its outputs OUT and $\overline{\text{OUT}}$.

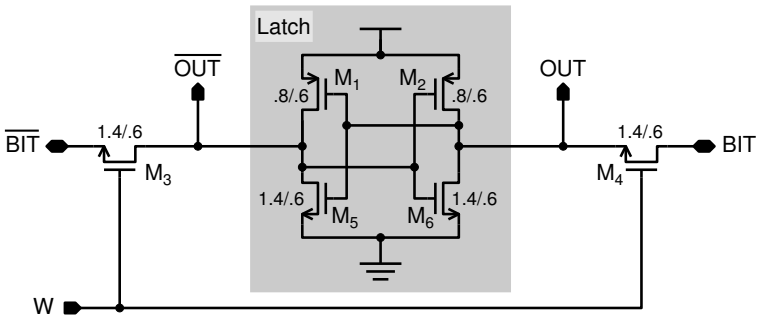


Figure 3.10: Six transistor SRAM cell. Channel dimensions are added in µm.

Given that the bit lines are precharged to a voltage close to vdd, the transistor dimensions can be determined following the arguments of Preston [Pre01]: The read operation requires the aspect ratio of the pass transistors to be sufficiently small compared with that of the NMOS transistors to avoid a read upset. The write operation, on the other hand, demands the aspect ratio of the pass transistors to be sufficiently large compared with that of the PMOS transistors of the latch for the SRAM cell to be writable. Although Preston [Pre01] reports some rules of thumb for the abovementioned ratios, the final sizing of the SRAM cell is also subject to minimizing the necessary die area and must be verified by elaborate simulations, since it largely depends on the actual process parameters and their variations. Compared to Preston's [Pre01] recommendations, the transistor dimensions reported here stress the ability to overwrite the content of the latch and trade off a bit of the safety margin for reading the cell against its compactness in terms of layout.

3.4.2 SRAM Architecture

The SRAM itself, that is, abstracted from its being embedded in the transistor cell array, is organized as an array of 96 x 64 bits. More precisely, the SRAM contains 96 horizontal bit lines and is controlled by 64 vertical word lines. The link between the arrangement of the 24 SRAM bits contained in each transistor cell as shown in Fig. 3.2 and the global view onto the SRAM itself is provided by Fig. 3.11. Accordingly, the SRAM of each transistor cell is controlled by six bit and four word lines.

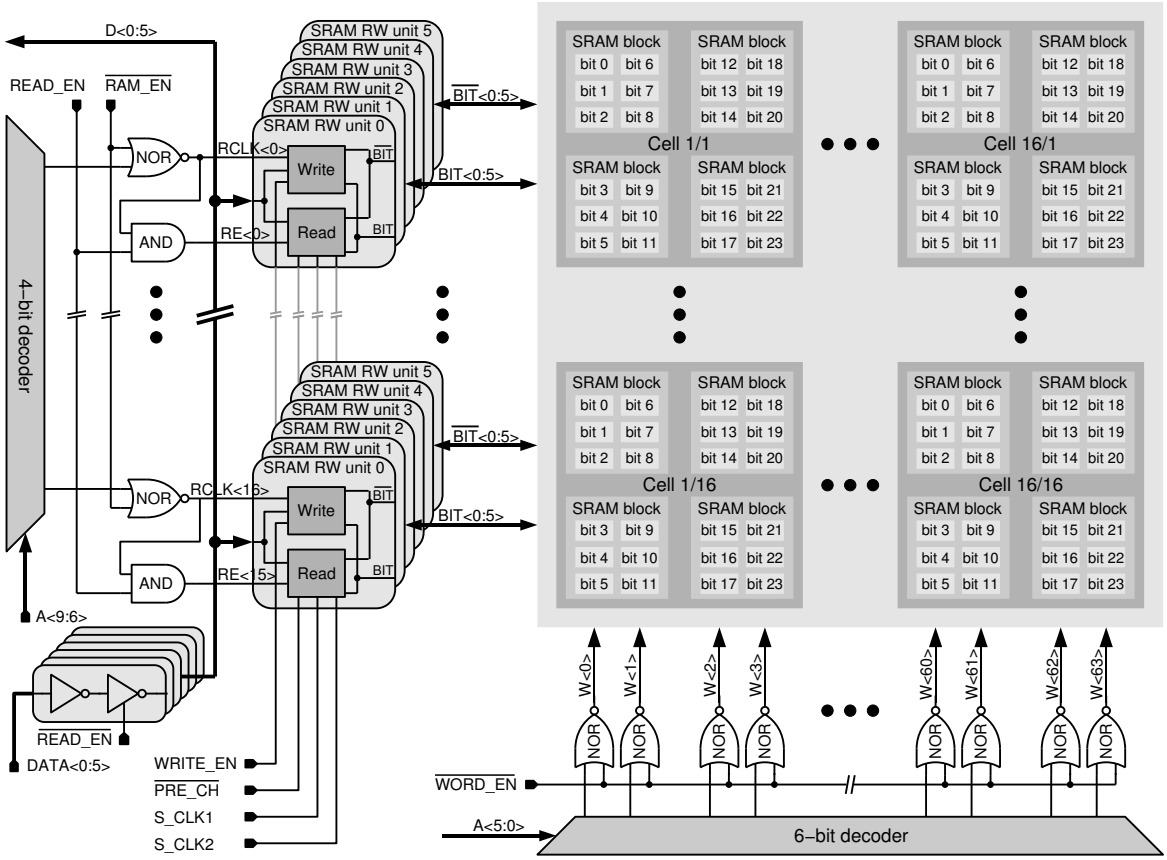


Figure 3.11: Structure of the distributed SRAM with read/write amplifiers.

The signals for the word lines W<0> to W<63> are generated by means of a 6-bit decoder, whose outputs are activated by the external $\overline{\text{WORD\_EN}}$ signal through the subsequent row of NOR gates. This enabling scheme is employed for all of the global decoders on the FPTA chip for the following reasons: First, it allows to deactivate all of the signals to be generated by the decoder (here that is the word line signals). Second, the enabling scheme avoids any glitches on the decoded signals due to changes in the address code and ensures that at most one of the decoded signals will be active at all times. Third, the scheme helps relaxing any timing constraints in that the address can be determined prior to the respective enable signal; the NOR gates can be designed such that they can drive the signals at hand fast.

Each of the 96 bit line pairs is controlled by its own read/write unit. The read/write units can store the information read from or to be written to one column of the SRAM allowing to read and write all 96 bits of one column in parallel. The read/write units for each row of transistor cells are aggregated to one larger unit that can be addressed via a 4-bit decoder. Its signals are activated by the $\overline{\text{RAM\_EN}}$ signal. The READ_EN decides upon the kind of operation. The limited number of digital signals available from the external electronic system severely constrains the possible width of the data bus DATA<0:5>. A width of six bit was chosen to comply with the logical structure of the SRAM prescribed by the organization of the transistor cell. In case of writing to the SRAM, the information for each column has to be divided into 16 packages of 6 bit. These have to be written to the read/write units in 16 subsequent cycles, prior to writing the information of one entire column into the RAM[11]

[11] Random Access Memory

itself. For the readout of the SRAM, the information of one column is stored in the 96 read/write units from which it has to be read out subsequently in packages of 6 bit, again using the data bus D<0:5>.

3.4.3 SRAM: IO-Circuitry

The implementation of the read/write units is illustrated in Fig. 3.12 for row $l = 16 \cdot m + i$. Please note, that the tristate buffer consisting of I_1 and I_2 and the respective data line DATA<i> are shared among all 96 read/write units. In addition, the AND gate of transistor cell row m, that is shared among all six read/write units of this transistor cell row, is included in Fig. 3.12 to simplify the explanation of the operation principle.

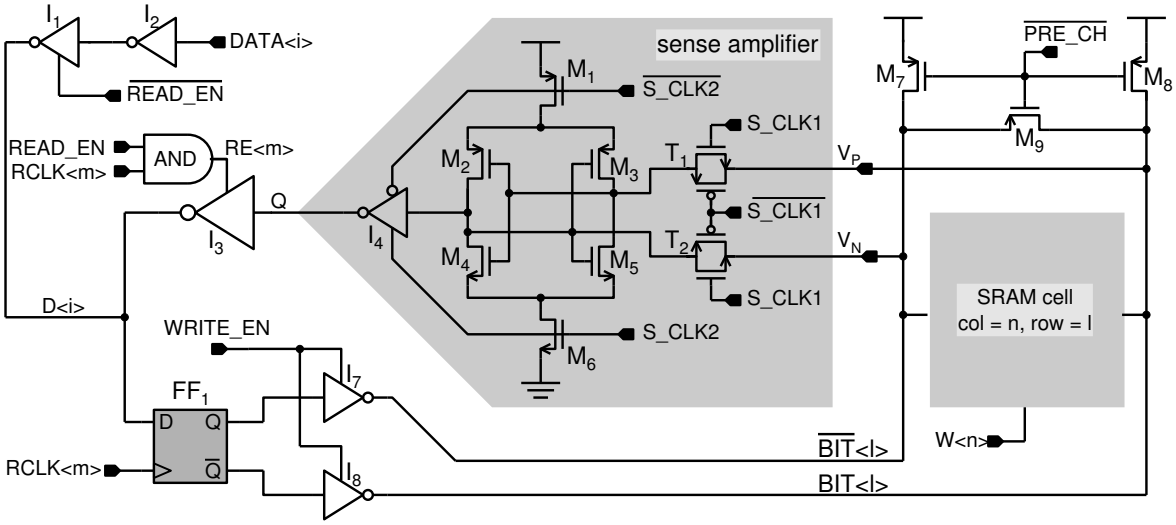


Figure 3.12: Write and read circuitry of one row together with the SRAM cell in row $16 \cdot m + i$ and column n. The tri-stating of DATA<i> is implemented only once, the generation of RE<m> only 16 times. Nevertheless, they are depicted in the upper left corner to allow for a complete understanding of the read and write access to one SRAM cell.

3.4.3.1 Write Operation

The bit to be written to the SRAM cell in row l and column n is stored in flip-flop FF_1, using the tri-state buffer (I_1 and I_2). The necessary RCLK<m> signal is generated with the 4-bit transistor cell column decoder only shown in Fig. 3.11. After the desired column has been selected by means of word line W<n>, it can be written to the cell by activating the global signal WRITE_EN. The required sequence for writing to the SRAM is summarized in the timing diagram shown in Fig. 3.13.

The left process writes the data for one column of SRAM into the 96 flip-flops FF_1 in 16 cycles, which consists of activating the correct RCLK signal by means of the 4-bit row decoder. Afterwards, all 96 bits are written to the RAM column selected by the column decoder. In order to configure the entire FPTA chip, the procedure depicted in Fig. 3.13 has to be repeated for 64 times. The timing diagram assumes all changes to be synchronous to the rising edge of the main CLK signal. In case of the presented timing scheme, the configuration of one row of the FPTA's SRAM requires exactly 32 clock cycles. Thus, the configuration of the entire FPTA chip takes $64 \cdot 32 = 2048$ clock cycles, which corresponds to 51.2 µs for a system clock frequency of 40 MHz. However, in the current implementation of the VHDL code controlling the RAM access, every operation is performed through the PCI bus. Therefore, in the system proposed in chapter 4, the time for the configuration rather

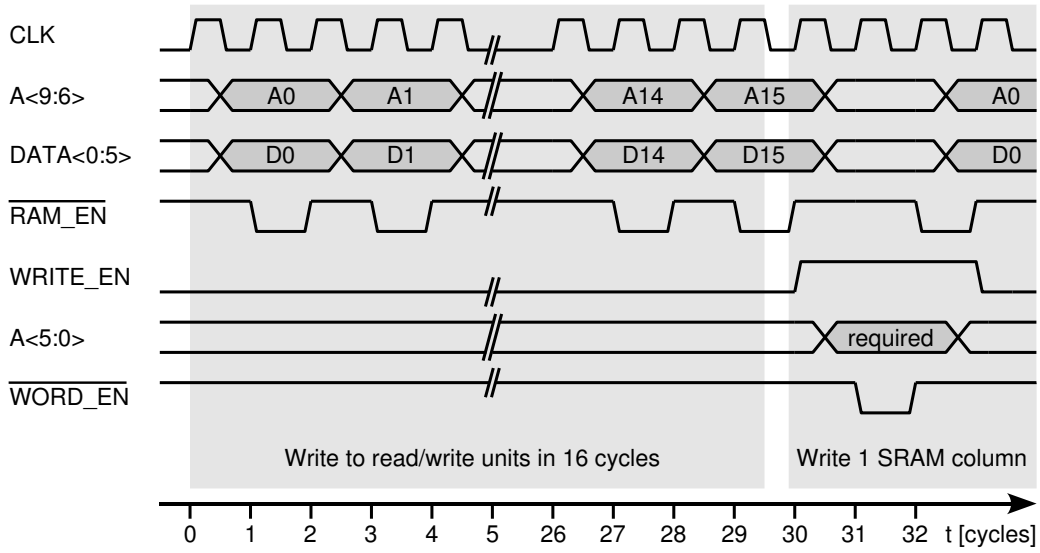


Figure 3.13: Timing diagram for writing to the SRAM cells.

amounts to values between 60 and 100 µs, that is, if the configuration process is not interrupted by the operating system.

3.4.3.2 Read Operation

For the readout of the FPTA's SRAM the content of one column is first sampled by the 96 read/write units before it can be read out over the tri-stated data bus D<0:5>. This sampling process starts with an equalization of the bit line potentials, which is achieved by means of transistors M_7 to M_9 depicted in Fig. 3.12. When the bit line equalization is finished, the desired column is activated by the according word line W<n>. As the bit line pairs are loaded with a considerable amount of parasitic capacitances, it takes rather long for the small transistors in the SRAM cell to charge these bit line pairs. Therefore, a sense amplifier is used to amplify the small potential difference on the bit lines and store the result until it is read out from the data bus D<0:5>. The exact timing is summarized in Fig. 3.14.

Please note, that the bit line equalization includes the gates of the sense amplifier, namely of the transistors M_2 to M_5; the sense amplifier has to be turned off during the equalization phase. During the readout of the sense amplifiers, the READ_EN signal has to be active. The readout then consists of a 16-fold repetition of providing the necessary transistor row address and activating the according RE<m> signal. In contrast to the 6-bit write accesses, the $\overline{\text{RAM\_EN}}$ signal is held low for two clock cycles such that the data can be read conveniently from the data bus D<0:5> by the external circuitry. In the conservative, strictly synchronous timing scheme illustrated in Fig. 3.14 the readout of one column of the FPTA's SRAM requires 53 clock cycles. Thus, the total time for a readout of the configuration of the entire FPTA amounts to $64 \cdot 53 = 3392$ clock cycles, which corresponds to 84.8 µs for a 40 MHz system clock. As the current VHDL implementation of the SRAM control may contain some further wait states and is also depending on the time slots available from the operating system, the time for the readout in the evolution system proposed in chapter 4 may be somewhat larger than these 84.8 µs, albeit still in the vicinity of 100 µs. However, in those cases in which the timing is critical for the successful readout operation, that is, the time for the pre-charge phase and the time between the activation of the $\overline{\text{WORD\_EN}}$ and the S_CLK2 signal, the VHDL code adheres to the timing diagram of Fig. 3.14.

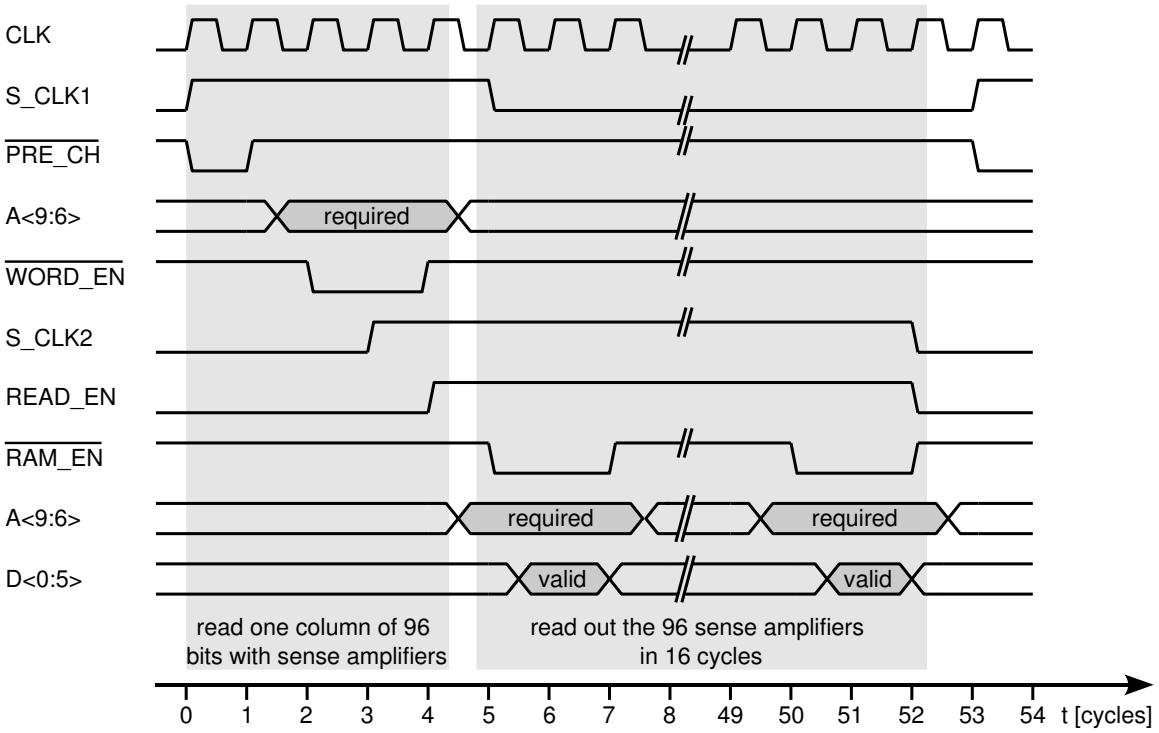


Figure 3.14: Timing diagram for the readout of one column of 96 SRAM cells.

3.4.3.3 Precharge and Sense Amplifier

Prior to the readout of an SRAM cell, the bit lines must be equalized to ensure the correct readout and avoid read upset. In the current design depicted in Fig. 3.12 the bit line pairs are precharged to vdd, as recommended e.g. by Geiger et al. ([Gei90a], section 9.10) and Preston [Pre01]. This choice accelerates the readout in that only the NMOS intrinsically stronger NMOS transistors of the SRAM cell are responsible for the voltage difference between the bit line pairs. Moreover, the precharge mechanism ensures that the bit lines are always restored to a well defined state before the next read operation.

In the proposed readout circuitry of Fig. 3.12 the sense amplifier serves two purposes: First, it accelerates the charging of the bit pairs. After the SRAM cell has provided a sufficient (in the order of 100 mV) voltage difference, the sense amplifier drives the bit lines to the full logic levels. Second, it stores the result of the read operation until it is readout from the FPTA chip. The first task requires a discrete-time comparator that is ideally very fast and possesses a low offset. Here, this comparator is realized as a dynamic latch, although combinations of differential pairs and latched structures are considered to be faster ([All02b], section 8.6, [Sas89], [Pre01]). Yet, the dynamic latch is advantageous in that it shortens the readout data path, since it also solves the second task of storing the result of the readout operation. The fact that the by far largest contribution to the time for the readout of one SRAM column stems from the successive readout of 6 bit packages further supports this choice. The minimum time necessary between the enabling of the word line W<n> and turning on the sense amplifier with the S_CLK2 signal depends on the resolution of the sense amplifier, the sizing of the RAM cell and the capacitive load of the bit line pairs. Special care has been taken in the layout of the sense amplifier to narrow the expected offset distribution. Nevertheless, according to [Sar91], the achievable resolution is also affected by differences in the capacitive loads of the bit line

pairs. Thus, the presented design could be further improved by adding a dummy inverter to balance the capacitive load introduced by I_4 as e.g. described in [Sil93].

3.4.4 SRAM: Concluding Remarks

3.4.4.1 Peculiarities of Embedded SRAM

The fact that the SRAM cells are scattered across the entire array of programmable transistor cells has the following effects on the RAM circuitry itself: First, the output nodes of the SRAM cell, OUT and $\overline{\text{OUT}}$, are capacitively loaded. The load varies between 7 and 160 fF. On one hand, these parasitic capacitances facilitate the readout, because they add an initial voltage difference to the bit line pairs. On the other hand, these capacitive loads must be charged through the pass transistors M_3 and M_4 in Fig. 3.10, which slightly increases the time necessary for re-writing the SRAM cells. However, these effects are not expected to severely affect the operation of the SRAM. Second, the spatial distribution of the SRAM cells causes the SRAM to appear larger to the according IO-circuitry than it actually is. A parasitic extraction of the metal line capacitances C_{ML} introduced by the bit and word lines yields values between 1.5 and 2.3 pF only in the transistor array itself. Taking further into account that another 700 to 800 µm of metal routing are necessary to cross the distance from the array boundaries to the word and bit line drivers (made up by the IO-cells), a comparison between the metal line capacitances and the parasitic capacitances caused by the pass transistors M_3 and M_4 of the SRAM cells depicted in Fig. 3.10 yields an effective SRAM size of at least $500 \times 400 = 200$ kbit.

3.4.4.2 Layout of the SRAM Read/Write Units

The layout of six read/write units serving 6 bit line pairs is displayed in Fig. 3.15. It occupies $336.3 \times 201.15\,\mu\text{m}^2$ of silicon area. Except for the AND gate at the top of the left block of circuitry, which corresponds to the AND gate in Fig. 3.12, the layout consists of 6 identical, albeit vertically mirrored, rows. The circuitry shown in the left half of the figure corresponds to the flip-flop FF_1 and the tri-state buffers I_3, I_7 and I_8 depicted in Fig. 3.12; it is composed of standard cells provided by the manufacturer [Aus97a]. The circuitry shown in the right block of Fig. 3.15 accounts for the precharge unit and the sense amplifier.

As was explained in section 3.4.3 all of the 96 bit line pairs are (dis-) charged at once during the read-out, write and precharge phases. During normal operation one bit line of each bit line pair will have to be completely charged or discharged. In the worst case of two inverted write accesses however, both bit lines have to flip their logic level. Due to the finite resistance and inductance of the bond wires, the required peak currents may not be available from the power supply. In order to avoid a break-down of the power supply voltage on the chip, the digital power supply must be buffered by blocking capacitors. Therefore, the circuitry of the read/write units is embedded in three blocking capacitors C_{block} providing a total of 64.18 pF. For all 16 transistor cell rows this adds up to 1.03 nF. If one neglects the external power supply and assumes that exactly one bit line of each bit line pair is recharged, any of the read, write, or precharge operations results in a drop of 1 V in the power supply voltage. Yet, given that there are additional blocking capacitors in each of the IO-cells (see section 3.5.5) and including the charge delivered by the external power supply, the chosen blocking capacitors leave a large safety margin. As they occupy approximately 40% of the shown layout, a reduction may be considered for saving silicon area in future implementations.

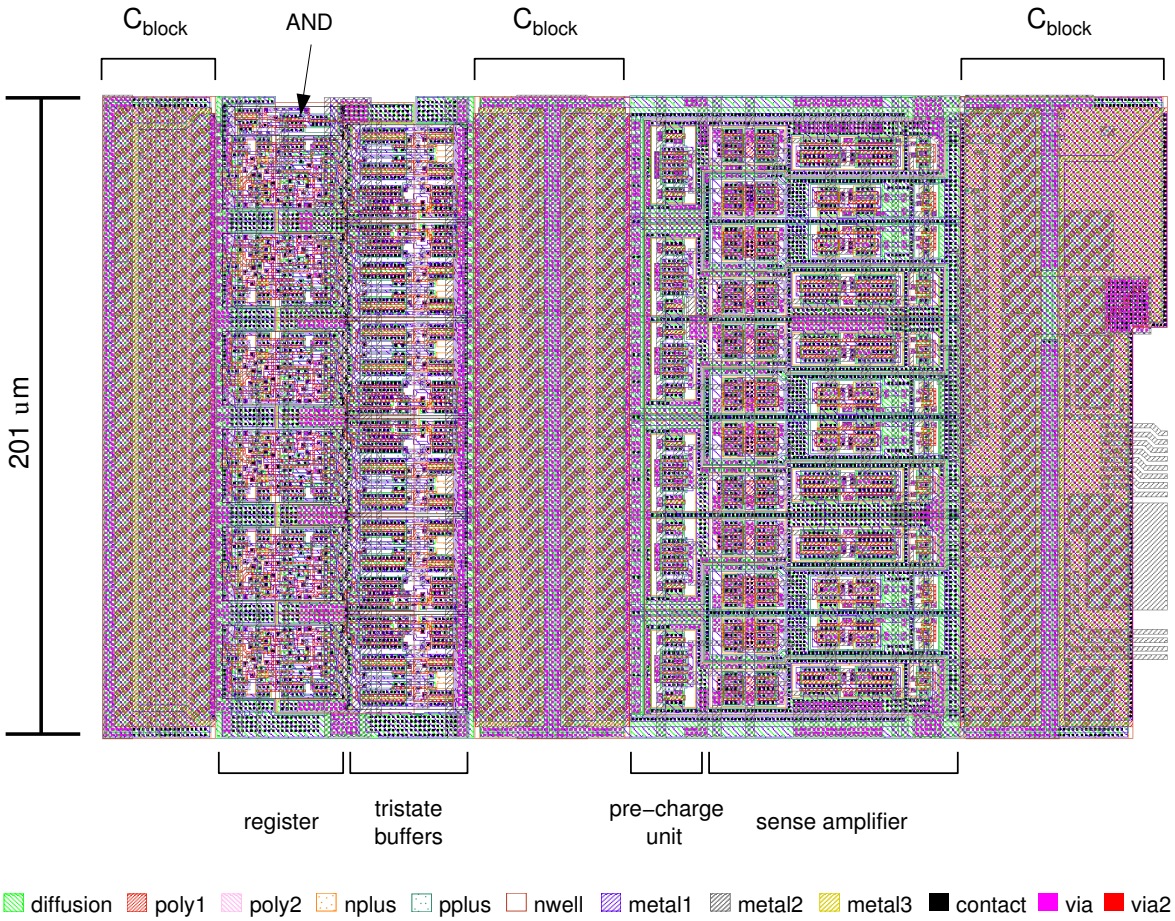


Figure 3.15: Layout of six SRAM read/write units that serve 6 bit line pairs. The pitch between two adjacent blocks amounts to 201.15 µm.

3.4.4.3 Timing Considerations

A full configuration of the FPTA as well as the readout thereof can be easily achieved within 100 µs according to section 3.4.3. This seems reasonable given an expected testing time of approximately 1 ms. To date, six dice from one MPW[12] run have been successfully operated at the abovementioned system clock rate of 40 MHz. The circuit simulations using the BSIM3.3 model suggest that the RAM can be operated at least up to clock rates of 100 MHz if synchronous timing schemes similar to those of Fig. 3.13 and 3.14 are used. Nevertheless, the proposed SRAM is by no means *verified* to do so: On one hand, this would require extensive system level simulations that take into account the most important worst case scenarios as well as Monte Carlo simulations of device variations. On the other hand, a final verification in hardware would have to report maximum bit error rates for a given maximum operation speed. Both approaches were foregone owing to the prototype character of the proposed FPTA chip. Besides, even though time critical operations did not stand higher clock rates, this flaw could be mitigated by inserting additional wait cycles.

Yet, if the configuration time must nevertheless be reduced, for instance, in case of a larger second generation FPTA implementation, this may be achieved by some of the following means; First, the RAM control could use a higher clock rate or take advantage of both clock edges as was explained in section 3.4.3. Second, if only a fraction of the PTA is used, a partial reconfiguration of either a

[12] Multi Project Wafer

subselection of columns or, if the unused cells are not to be configured in a well defined way, of a subselection of transistor cells can be applied. For future FPTA designs, however, a faster interface, most easily implemented by a wider data bus, would be desirable.

3.5 IO-Cells

The mixed signal test environment – described in chapter 4 – provides only one fast analog input and output channel, respectively. An integration of the necessary analog-to-digital and digital-to-analog conversion, on the other hand, is prone to errors and limited analog performance. Moreover, the effort its design would have required in terms of money and time is beyond the scope of the first FPTA prototype. Hence, a sample-and-hold based multiplexing system capable of generating complex spatio-temporal input patterns as well as of realizing complex sampling procedures is a must. The task is solved by means of an ensemble of *IO-cells* that can be either used to sample an input voltage and present its buffered version to the programmable transistor array or sample the output of the array and hold it until it is driven off-chip. As shown in Fig. 3.1, an IO-cell is dedicated to each of the 64 transistor cell edges of the PTA. Thus, any of the boundary cells of the PTA can be used as an in- or output. On one hand, this concept combines a maximum of flexibility with minimum analog signal paths (in terms of multiplexing switches and path lengths between the array border and the according buffer in- or output) and therefore maximum analog performance. On the other hand, the concept is expensive in terms of occupied silicon area. Nevertheless, future FPTA chips may benefit from experiences gathered from experiments that use the proposed system and thus may realize a more efficient signal management.

3.5.1 Functionality of the IO-cells

The IO-cells consist of a sample and hold stage, a set of switches to configure this S/H stage and some control circuitry by which the cell can be configured. Each S/H cell features two identical rail-to-rail amplifiers. While the description of the actual implementation is deferred to section 3.5.2, this section is confined to a functional view of the IO-cells. The different operation modes of the IO-cells are summarized in Table 3.7. In the current scope, only modes 0 to 4 and mode 8 are of interest together with the according entries of the last three columns. Apart from the in- and output configurations, the IO-cell can be completely turned off via mode 0 – the *passive* function: This opens all switches, in particular those three connected to the transistor cell node, whose capacitive load is thereby reduced to the minimum parasitic capacitances caused by these open transmission gates. Furthermore, both operational amplifiers are turned off in the passive mode, in order to minimize the thermal impact of the IO-cells.

3.5.1.1 Output Configurations

The two output configurations *direct out* and *buffered output* (mode 2 and 4) as well as the configuration for *direct I/O* (mode 1) are depicted in Fig. 3.16. In the recommended output configuration, namely in *buffered output* mode, switches S_3 and S_1 are open and the uppermost signal path is used: The voltage present at the respective cell node is buffered by OP_1, and subsequently sampled on C_{SH} via S_{sam}. The voltage hold by C_{SH} is then buffered by OP_2 until it is multiplexed to the analog output line ANA_OUT that is buffered by the output buffer described in section 3.7. Since switches S_1, S_3 and S_4 are meant to be configured at most once *before* each circuit test, they are referred to as static switches. In contrast, S_{sam} and S_{out} must change their state multiple times during a circuit test and are thus referred to as *dynamic* switches.

mode	DATA<3>	DATA<2>	DATA<1>	DATA<0>	function	OP1	OP2
0	0	0	0	0	passive	off	off
1	0	0	0	1	direct I/O	off	off
2	0	0	1	0	direct out	off	off
3	0	0	1	1	direct in	off	off
4	0	1	0	0	buffered output	on	on
8	1	0	0	0	buffered input	off	on
5	0	1	0	1	test/debug	on	on
6	0	1	1	0	test/debug	on	on
7	0	1	1	1	—	on	on
9	1	0	0	1	test/debug	off	on
10	1	0	1	0	test/debug	off	on
11	1	0	1	1	—	off	on
12–15	1	1	X	X	—/self-destruction	on	on

Table 3.7: The six different IO-modes. Only modes 0–4 and 8 are meant for regular use. IO modes 5,6 and 9,10 may be beneficial for testing debugging and characterizing the buffer circuits. IO modes 12–15 may result in short-circuiting the outputs of the input driver and OP1, which may be exploited for heating the chip, but may also allow for detrimental effects.

The sample and hold unit can be bypassed via switch S_3 in the *direct out* configuration (mode 2). Besides being a fall-back solution if the sample and hold unit should fail to work, it can be used for testing and debugging purposes, e.g. for measuring the offset distribution of the sample and hold stages. Moreover, in an evolution experiment, randomly choosing either IO-mode 2 or 4 prior to each circuit test can help establishing a selection pressure towards circuits capable of driving different capacitive loads. Finally, the configuration *direct I/O* (mode 1) connects the respective transistor cell node to the according *probepad*, which can be both, input or output. As the probepads can be connected via bond wires, this allows for a direct access to the PTA. Apart from testing and debugging the *direct I/O* configuration allows to connect two FPTA dice via bond wires between the respective probepads.

3.5.1.2 Input Configurations

The two output configurations *direct in* and *buffered input* (mode 3 and 8) are illustrated in Fig 3.17. In mode 3 the IO-cell is again used as a sample and hold unit (upper signal path is active, S_2 open). Yet, the input signal applied to the external analog input ANA_IN is assumed to be sufficiently strong

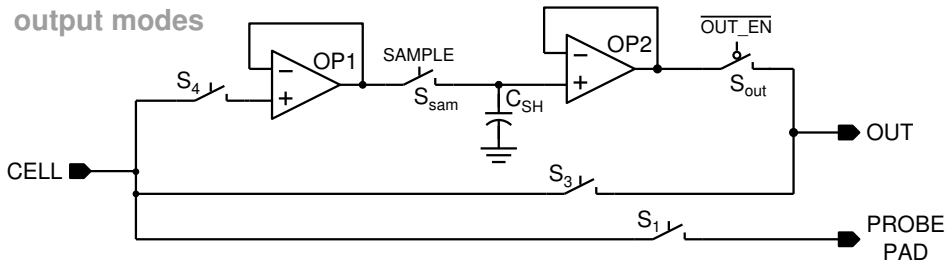


Figure 3.16: Simplified schematic illustrating the three different output modes. The connection to the probepad actually serves as both, in- and output.

to directly drive the hold capacity C_{SH}. As switch S_6 is configured *statically*, the cell node is always driven by OP_2, which follows the voltage sampled on C_{SH}. Here the possibility to update a set of input voltages all at once is foregone to simplify the signal path in particular and the IO-cell design in general.

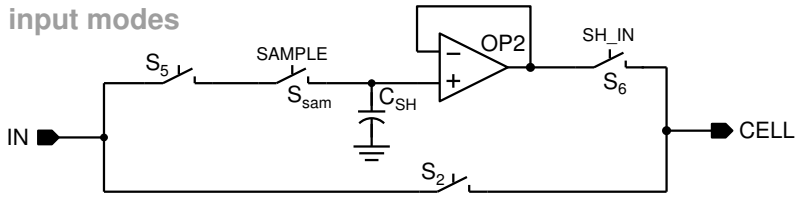


Figure 3.17: Simplified schematic depicting the two different input modes.

If only one analog input is required, it can be freed from the additional noise and distortion introduced by the sample and hold stage by using the *direct in* configuration, which shortcircuits the analog input port and the according cell node. If the programmable transistor array is used to route test signals between different IO-cells, all of the buffers contained in the IO-cells can, in principle, be characterized: Since a combination of IO-modes 8 and 2 allows to characterize OP_2, the behavior of OP_1 can be inferred from the behavior of the IO-cell in mode 1. This should at least allow for the measurement of the offset distribution of the integrated buffers.

3.5.2 Architecture of the IO-Cells

The implementation of the different IO-cell functionalities discussed above is shown in Fig. 3.18. The analog core of the cell realizing the sample and hold circuits as well as the switches connecting the analog signals are depicted in the upper part of the figure. The digital circuitry necessary to control the analog functionality is divided into four different blocks, which are displayed in the lower part of Fig. 3.18:

Lower left corner: Decoder An 8-bit decoder is used to configure the 64 different IO-cells. The last two bits of the address bus, A<1:0>, are used to determine three independent clock signals DFCLK<1:3>. Again, a separate active-low signal, $\overline{\text{SH\_EN}}$, is used to activate the outputs of the address decoder to avoid glitches.

Lower middle part: IO-mode The DFCLK<1> signal is used as a clock for flip-flops FF_1 to FF_4, which are programmed by the first four bits of the data bus, DATA<0:3>. The content of Flip-flops FF_1 and FF_2 determines whether any of the direct modes 1 to 3 are enabled. The buffered out- and input modes 4 and 8 are configured using FF_3 and FF_4, respectively. That is, their outputs control the state of switches S_4 to S_6 and provide the appropriate power-down signals for the operational amplifiers.

Lower right corner: Sample signal This block decides upon the generation of the SAMPLE signal that is used to steer S_{sam} and thereby define the sample and hold stages. The sample signal of each IO-cell can be either chosen from seven independent signals, namely SAM<0:3> and DATA<3:5>, or can be connected to gnd to turn off the sampling mechanism. The necessary information is stored in the three flip-flops FF_5 to FF_7, which are programmed by means of the DFCLK<2> clock in conjunction with DATA<0:2>. The manner in which the data bus signals are allocated avoids unwanted glitches on the SAMPLE line during the configuration phase.

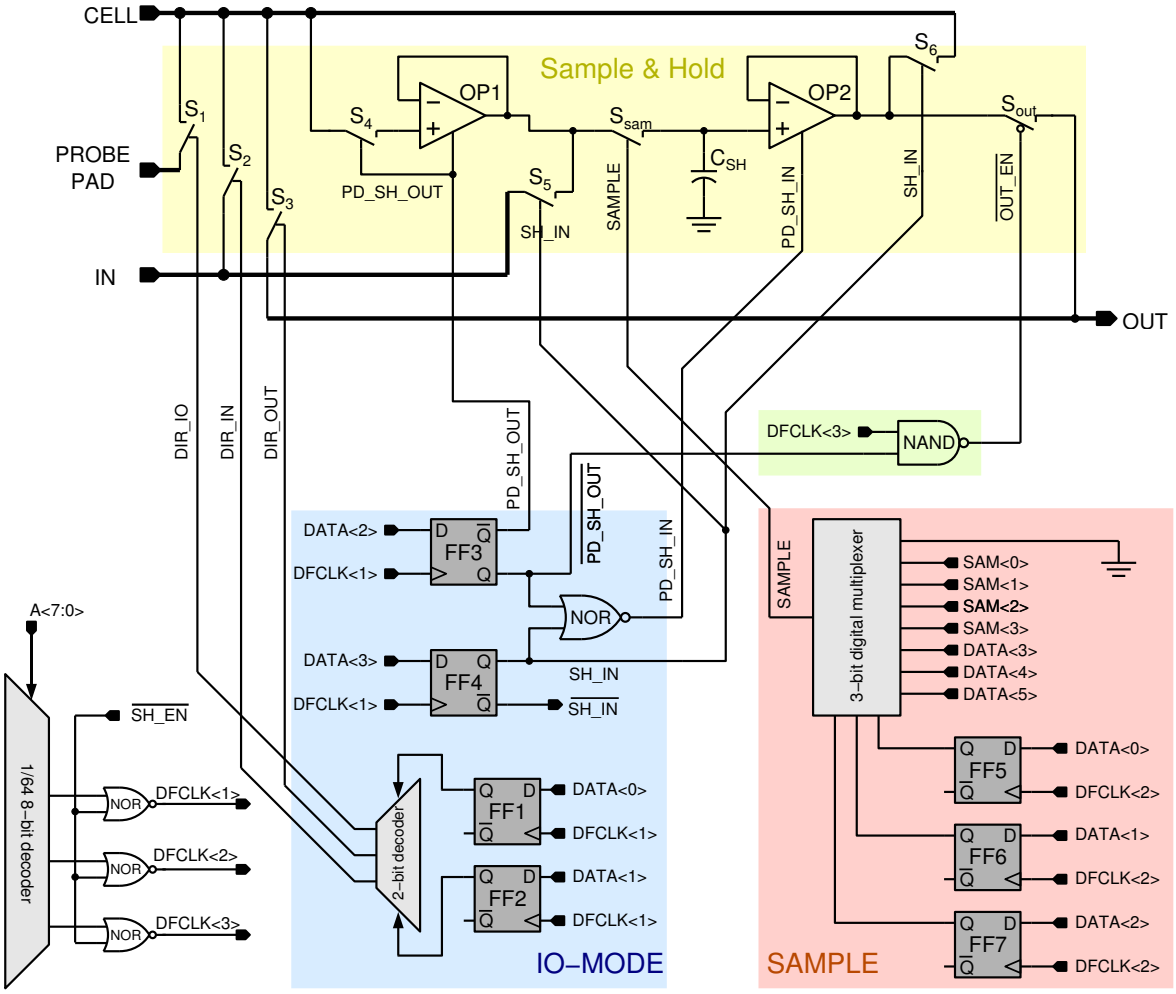


Figure 3.18: Simplified schematic of one entire IO-cell. All switches are implemented as transmission gates. Digital signals are usually depicted in one polarity only. The additional inverters necessary to generate the inverse signals are omitted as are any additional buffers for signal restoration.

Right middle part: Output enable In the *buffered output* configuration (mode 4), the node voltages sampled at the according cell must be read out from the output of OP_2. To allow the output of exactly one IO-cell to be multiplexed onto the global ANA_IN signal, switch S_{out} can be controlled by using the DFCLK<3> signal. The NAND gate ensures that S_{out} can only be activated if the IO-cell is indeed configured as a *buffered output*.

3.5.3 Configuration of the IO-cells

The IO-cells have to be configured in two steps: First, the IO-mode of the respective IO-cell addressed by A<7:2> must be set according to Table 3.7. This is achieved by using sub-address A<1:0> = 01 during the required activation period of $\overline{\text{SH\_EN}}$, as depicted in Fig. 3.19. Second, the source of the SAMPLE signal must be chosen for the respective cell. Table 3.8 lists the bit patterns necessary for the eight possible configurations. The second and fourth block shaded in gray in Fig. 3.19 exemplify the required timing of the signals involved. In particular, the sub-address A<1:0> must be set to 10 to enable the required DFCLK2 clock.

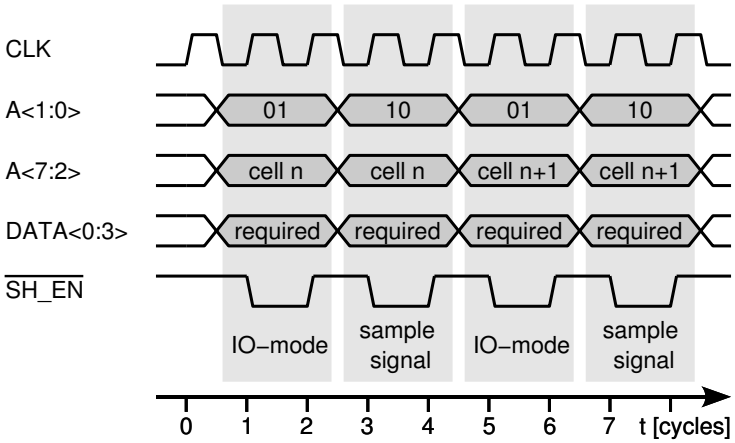


Figure 3.19: Timing diagram for the configuration of two IO-cells.

No.	DATA<2>	DATA<1>	DATA<0>	SAMPLE
0	0	0	0	GND
1	0	0	1	SAM<0>
2	0	1	0	SAM<1>
3	0	1	1	SAM<2>
4	1	0	0	SAM<3>
5	1	0	1	DATA<3>
6	1	1	0	DATA<4>
7	1	1	1	DATA<5>

Table 3.8: Configuration of the SAMPLE signal. In the second phase defining the SAMPLE signal, only data lines DATA<0:2> are required.

3.5.3.1 S/H Operation Mode 1

In a typical application, the IO-cells are meant to operate as *buffered in-* and *outputs*. If the number of independent input voltages does not exceed six or seven and all outputs can be sampled at the same time, the required SAMPLE signals can be generated from SAM<0:3> and DATA<3:5>. In this case, applying two input voltages to different cells can be achieved by a signal pattern similar to that illustrated in Fig. 3.20 (provided that the IO-cells have been correctly configured in the *buffered input* mode beforehand): Here, the sample signals SAM<0> and SAM<1> are used to subsequently sample the analog voltages V<0> and V<1> applied to the global ANA_IN terminal so that they are stored on C_{SH} and applied to the respective transistor cell node by OP_2.

The procedure necessary to read out three cell nodes at two separate times is described by the timing diagram of Fig. 3.21. As the sample lines SAM<0> and SAM<1> are already used up for the sampling of the input voltages V<0> and V<1>, SAM<2> is used to sample the outputs at cells *i* and *j* in the second clock period. The node voltage at the output of cell *k* is recorded during the fourth clock period by SAM<3>. The output voltages sampled in IO-cells *i*, *j* and *k* are multiplexed to the global analog output ANA_OUT by combining the $\overline{\text{SH\_EN}}$ signal and the respective cell address. In this case, the sub-address A<1:0> must be set to 11 for $\overline{\text{SH\_EN}}$ to be converted into the required DFCLK<3> clock.

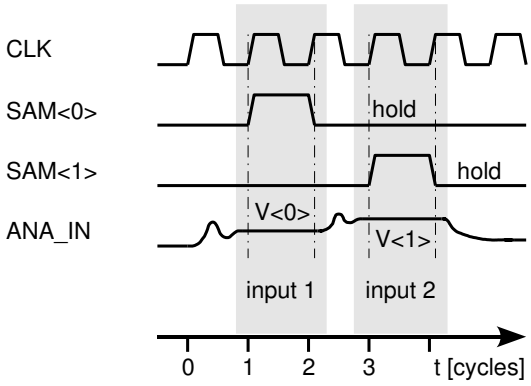


Figure 3.20: Timing diagram for writing two analog voltages to two IO-cells.

The abovementioned limitation to six or seven separate input signals depends on the design of the external electrical system: Provided that the external circuitry allows to hold the analog input signal at the desired level until the set of outputs is sampled, the signal used for sampling the last (seventh) set of input voltages can also be used for sampling the set of output signals. While the rising edge of the last sample signal determines the start of applying the last set of analog inputs, its falling edge defines the termination of the sample period of the set of outputs. However, if less than seven sample signals are used, it is recommended to use separate signals for in- and output sampling for the following reason: As explained in section 3.5.4, the falling edge of the SAMPLE signal will cause some charge injection onto C_{SH}, which will add a small offset to the desired input voltage. Omitting the falling edge during the sampling of the last input voltage will therefore result in a small mismatch between the last set of input voltages and those sampled before.

3.5.3.2 S/H Operation Mode 2

In the unusual case in which more than 7 independent sample signals are essential, as e.g. for 8-bit DACs, the IO-cells can be operated in another scheme that allows for an arbitrary number of independent sample actions. At first, in the configuration phase, the sample signal for all IO-cells

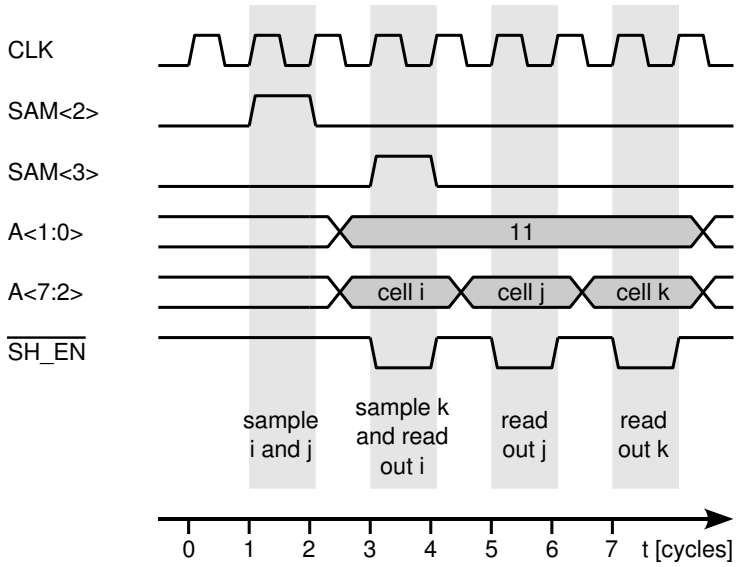


Figure 3.21: Timing diagram for the analog readout of an analog voltage by means of an IO-cell.

designated to work in either of the two buffered modes has to be set to gnd. During the actual circuit test, the different sample signals used for the in- and output of analog voltages according to the timing diagrams depicted in Fig. 3.20 and 3.21 have to be generated as depicted in Fig. 3.22:

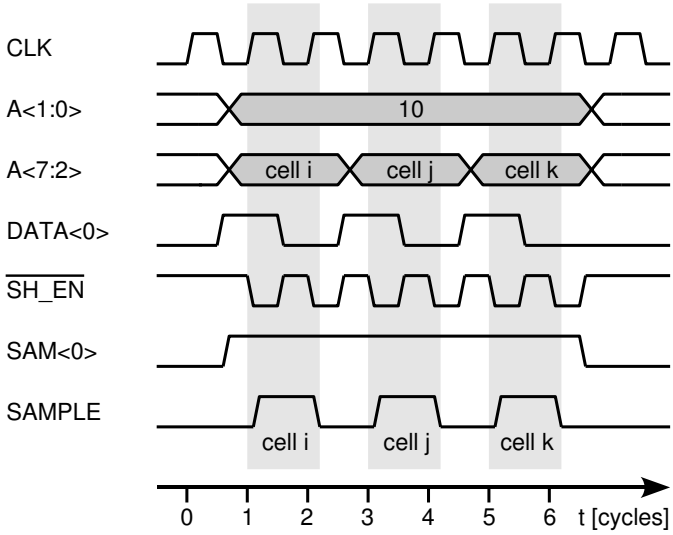


Figure 3.22: Timing diagram for the generation of an arbitrary number of SAMPLE signals.

During the sampling phase, one of the seven possible sample lines must be tied to logic high (here SAM<0> is chosen) and the sub-address A<1:0 must be set 10. A sample signal in any of the 64 IO-cells is then activated by switching the SAMPLE line from gnd to SAM<0> and deactivated again by switching it back to gnd, which is achieved by the combination of the cell's address A<7:2>, the SH_EN signal and the necessary sample line configuration encoded in DATA<0:2>. Although this scheme is more versatile than the one previously discussed, it is prone to a number of disadvantages: For one, the scheme is more complex and not yet provided by the VHDL module described in section 4.3.2. Second, the maximum sampling frequency is limited to half of the system clock, since the flip-flops FF_5 to FF_7 need to be set twice in separate clock cycles. Finally, an overlap of output sampling and readout as suggested in Fig. 3.21 is impossible, since DFCLK<2> and DFCLK<3> cannot be active at the same time.

3.5.4 Sample and Hold Units: The Analog Perspective

The analog specifications such as bandwidth, precision and distortion depend heavily on the implementation of the operational amplifiers and the switches used in the S/H stage, especially S_{sam}.

3.5.4.1 Rail-to-Rail Operational Amplifier

The requirements for OP_1 and OP_2 are almost identical: Apart from the usual targets, as e.g. high precision, low distortion, fast settling time and the like achieved with a minimum of quiescent current, both op amps must be able to drive between 5 and 30 pF in the unity-gain configuration, operate on the complete power supply range, and provide a large-signal bandwidth that ideally exceeds 5 MHz. In normal operation, that is when the analog output is driven by the output buffer, a resistive load can only be encountered in the *buffered input* mode, where it can be a part of the circuit residing in the PTA. Though undesired in this case, the ability to drive resistive loads is almost unavoidable when designing the op amp to drive large capacitive loads at high speed. Due to these most similar target

specifications, the same op amp is used for both buffers OP_1 and OP_2. Its design is detailed in section 3.7.

3.5.4.2 Switches

Most of the analog switches in the S/H block shown in the upper part of Fig. 3.18 are not switched during the normal operation thereof. They must merely be large enough not to affect the necessary analog bandwidth. Therefore, all of these switches are implemented as transmission gates with the same channel dimension as the one depicted in the center of Fig. 1.3.2.3. Within the S/H block shown of Fig. 3.18, the largest load is given by $C_{\mathrm{SH}} = 5\,\mathrm{pF}$. Thus, the bandwidth is limited most severely by the time constant of the on-resistances of S_5 and S_{sam} and the C_{SH}. According to (1.10) the settling time for the V_{SH} can be calculated by

$$T_{\mathrm{settle}} \;=\; R_{\mathrm{TG}}(C_{\mathrm{SH}} + C_{\mathrm{OP,in}})\ln(2)\mathrm{Res\ [bit]} \;=\; 2\cdot 350\,\Omega\cdot 5.46\,\mathrm{pF}\cdot 16\ln(2) = 42.2\,\mathrm{ns}\ , \qquad (3.2)$$

where an approximate input capacitance of OP_2, $C_{\mathrm{OP,in}} = 463\,\mathrm{fF}$ is added to C_{SH} and the maximum on-resistance is estimated from Fig. 3.5(a). The relatively small T_{settle} leaves room for sampling any kind of signals with a precision of 16 bit at 20 MHz and matches rather well with the smallest settling times achieved by the used op amps, which amount 60 to 80 ns (cf. section 3.7).

The second time constant of interest is described by the maximum hold time T_{hold}. It can be computed by extending (1.12) to all of the involved reverse-biased terminal diodes:

$$T_{\mathrm{hold}} \;=\; \frac{C_{\mathrm{SH}} + C_{\mathrm{OP,in}}}{I_{\mathrm{np}} - I_{\mathrm{pn}}}\cdot \mathrm{Vdd}\cdot 2^{-\mathrm{Res\ [bit]}} \;=\; \frac{5.46\,\mathrm{pF}}{54\,\mathrm{fF}}\cdot 5\,\mathrm{V}\cdot 2^{-16} \;=\; 7.7\,\mathrm{ms}\ . \qquad (3.3)$$

Depending on the desired precision, the stored voltage V_{SH} should be refreshed or read out after less than 5 to 100 ms. As typical circuit tests are not to exceed a few ms during which a series of hundreds of analog test vectors are applied, the hold time is more than sufficient.

Of the two *dynamically* used switches, namely S_{out} and S_{sam}, only the charge injected by the latter one causes an error voltage on a hold capacitor in the way described in section 1.3.2.3. To reduce the resulting error, the transmission gate is extended by a pair of dummy transistors as depicted in Fig. 3.23. The dummy transistors M_5 and M_6 are designed to be exactly half of the transistors M_3 and M_4 constituting the transmission gate itself. For an infinitely fast clock signal V_ϕ the amount of charge injected by M_5 and M_6 countervails that injected by the transmission gate itself. In this case, the uncorrected error voltages V_{err_0} induced by either M_3 or M_4 can be calculated by means of (1.17). Table 3.9 lists some values for the error voltages caused by M_3 and M_4 as well as for the sum

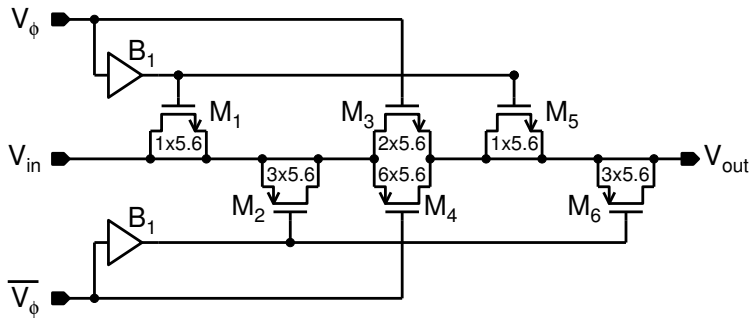


Figure 3.23: Implementation of S_{sam} as a transmission gate with dummy transistors to reduce charge injection. All transistors possess the minimum channel length $L = 0.6\,\mu\mathrm{m}$ and width $W = 5.6\,\mu\mathrm{m}$, albeit a different number of gates.

	$\tau_{\text{crit}} = \frac{5\,\text{V}}{\alpha_{\text{crit}}}[\text{ns}]$		$V_{\text{err}_0}[\text{mV}]$			$V_{\text{err}}[\text{mV}]$
Source	(1.15)		(1.17)			BSIM3.3
$V_S \cong V_{\text{SH}}[\text{mV}]$	M_3	M_4	M_3	M_4	sum(M_3, M_4)	sum$(M_3, \ldots, M_6)$
0	1.34	∞	−5.34	5.19	−0.15	0.45
1	2.94	∞	−4.17	5.19	1.03	0.21
2	9.48	53.44	−3.09	6.91	3.82	0.21
3	168.45	10.34	−2.05	9.09	7.04	0.21
4	∞	4.18	−1.73	11.32	9.59	0.21
5	∞	2.23	−1.73	13.58	11.85	0.86

Table 3.9: Data for the clock feedthrough analysis of the transmission gate depicted in Fig. 3.23. The last column reports the results of a typical mean simulation of a sample operation performed by the IO-cell.

thereof. Their respective contributions partially cancel each other. Yet, due to the different channel dimensions and process parameters, cancellation is far from complete.

In reality, however, the clock signal V_ϕ is bound to change in a finite time. As discussed in section 1.3.2.3, this situation can be described by (1.16) for one NMOS pass transistor without dummy cancellation. Here some of the injected charge can escape through the channels of the transmission gate during its turn-off. As this also holds for the dummy transistor pair, proper charge-injection cancellation can be achieved with two different timing schemes, as is reported by Eichenberger and Guggenbühl [Eic89]. Both schemes are discussed based on the nomenclature and definitions of section 1.3.2.3.

simultaneous switching The principle idea is to use *slow* clock signals which satisfy $\alpha \ll \alpha_{\text{crit}}$ (where α_{crit} is defined in (1.15)) and thus allow almost all of the charge injected during the on-phase of the switch to be equalized through the respective transistor channels. For a given signal voltage $V_S \cong V_{\text{SH}}$, the relative size of say M_3 and M_5 can be adjusted such, that for the actual delay between the two clock signals V_ϕ and V'_ϕ the two contributions of pass and dummy transistor cancel out. This method can be advantageous, if the signal levels are confined to a small range, as the *slow* switching minimizes the net charge left on node V_{SH}. However, as the error voltage induced on V_{SH} depends on V_S, it does not work very well for signals spread over the whole power supply range. Moreover, the required synchrony of V_ϕ and V'_ϕ is difficult to achieve.

delayed switching If the dummy clock signal V'_ϕ is delayed until the transmission gate ceases to conduct, the dummy transistors will deposit their full charge on node V_{SH} inducing the error voltage $-V_{\text{err},0}$ defined in (1.17). In the ideal case of a very *fast* clock signal V_ϕ satisfying $\alpha \gg \alpha_{\text{crit}}$, the same is true for the transistors of the transmission gate. Consequently, the two effects can be arranged to cancel by designing the dummy transistors to be half as wide as the pass transistors.

As the voltages to be sampled on C_{SH} can take on any value in the full power supply range, the delayed switching scheme is adopted in the proposed IO-cell implementation. Relatively strong buffers are used to drive the clock signals V_ϕ and V'_ϕ, thus ensuring steep slopes. The delay is achieved by using the buffers B_1 and B_2 shown in Fig. 3.23. Table 3.9 summarizes the critical switching times $\tau_{\text{crit}} = \frac{5\,\text{V}}{\alpha_{\text{crit}}}[\text{ns}]$ for six different signal levels. Typical mean simulations of the involved part of the IO-cell using realistic metal line capacitances yield switching times between 300 and 400 ps, which is at

least smaller than τ_{crit} by a factor of 3. The resulting error voltages are summarized in the last column of Table 3.9. According to these simulations of the typical mean case, the error voltage induced by the injected charge of the transmission gate is reduced by at least a factor 10, seems to be constant in the range between 1 and 4 V, and always stays below 1 mV. However, more elaborate simulations including back-annotation and Monte Carlo analyses would be necessary to verify the implemented clock feedthrough cancellation to work as well for realistic device mismatch and process variations, too. Precise testing is elaborate, but can, in principle, be achieved by using different operation modes, as has been hinted at in the preceeding subsections.

3.5.5 Layout of an IO-Cell

Layout and floorplan of the circuitry placed to the right of one row of the PTA array are shown in Fig. 3.24. Apart from the IO-cell itself, the layout contains part of the power routing, a signal bus with row decoders and some block capacitors. In principle, the same layout is placed adjacent to the programmable transistor array on all of its four boundaries. Therefore, the pitch of two neighboring layout blocks can be adapted to the pitch of the PTA cells in X- and Y-direction, namely between 201.15 µm and 198.2 µm.

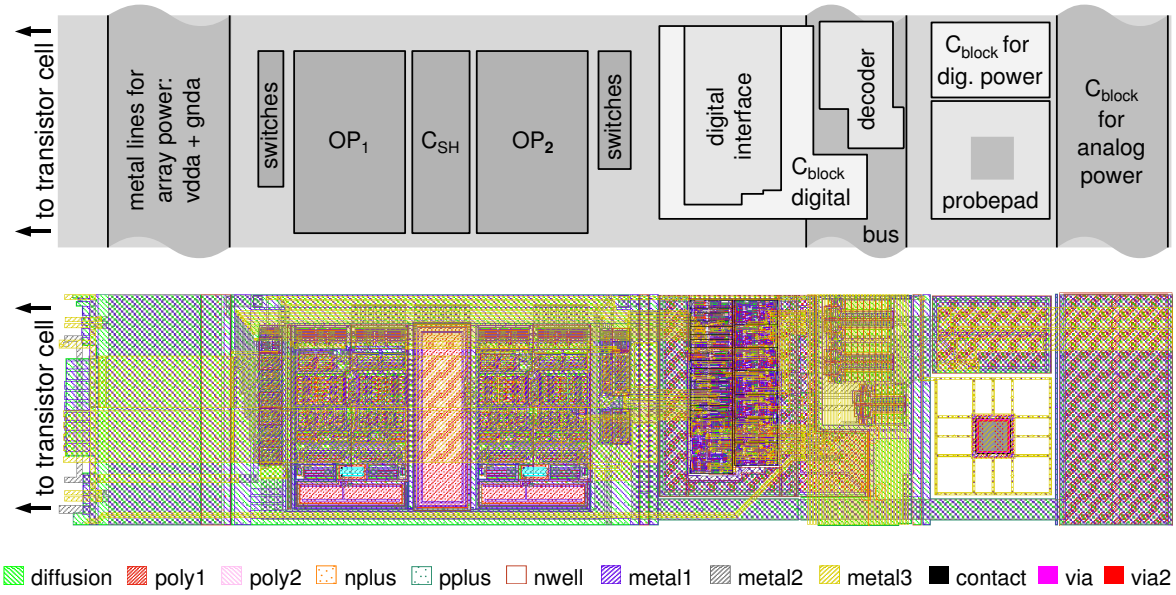


Figure 3.24: Floorplan (top) and layout (bottom) of one complete IO-cell, as used to the right of the transistor array. The displayed structures occupy an area of approximately 985×205 µm. The pitch between two adjacent IO-cells amounts to 201.15 µm.

The analog circuitry is placed closer to the PTA than the digital circuitry to avoid deterioration of the analog signals in the cell array and the S/H stage of the IO-cell. Read from left to right, the following blocks are realized in the layout shown in Fig. 3.24: The PTA is surrounded by 100 µm wide metal lines providing the power for the analog array. The substrate beneath is used as a guard ring to isolate the PTA substrate from the substrate beneath the outer circuitry. The actual sample and hold unit, whose schematic is depicted at the bottom of Fig. 3.18, is placed next to the power supply ring and followed by the digital interface of the IO-cell (the schematic of which is depicted at the bottom of Fig. 3.18). It is surrounded by block capacitors which are realized as PMOS transistor gates (cf. section 1.2.2). These block capacitors are used to stabilize the digital power supply against the current spikes caused by the fast buffers generating the SAMPLE signals. On one hand, this

ensures proper logic high and low levels at the gates of the sample and hold switches, and on the other hand, the power blocking is meant to reduce the substrate coupling of the fast digital signals. A signal bus containing most of the global analog and digital signals encircles the PTA. It is located between the digital block and the probepad usable for inter-chip bonding. The necessary row and column decoders are placed beneath this signal bus. The probepad occupies an area of approximately $100\,\mu m \times 100\,\mu m$ to facilitate bonding. However, the actual contact area is limited to $40\,\mu m \times 40\,\mu m$ to reduce its capacitance. The area next to the probepad is used for further blocking of the digital power. At the right hand side, the chip ends with part of the analog power supply ring, which is also blocked by PMOS gates. The blocking capacitances per IO-cell amount to 43.3 pF and 47.3 pF for the digital and analog power supply, respectively. For the complete chip, this corresponds to 2.77 nF and 3.03 nF.

3.6 Inner-Cell Signal Probing

Successfully evolved circuits are of most interest if they meet the desired specifications in an unexpected, unconventional way. It is precisely in this case, that new (sub-) circuits and circuit principles are found by the evolution system. However, such unconventional circuit solutions may also defy understanding. Though simulations can indeed mitigate the analysis of these circuits, they may not be precise enough to capture all parasitic effects the evolved circuit may exploit. Hence, the FPTA chip offers[13] the possibility to probe most of the inner nodes of the PTA and even attain a rough estimate of most of the involved currents therein.

3.6.1 Inner-Cell Probing Concept

Concretely, the inner-cell probing facilities allow to read out the voltage of either of the cell border terminals N or E, or of the generic transistor terminals TD or TS. The situation is illustrated in Fig. 3.25 for a cutout of 3×3 transistor cells. Those terminals that can be read out are shaded in white. As the east and west as well as the north and south terminals of adjacent cells are directly connected, the W and S terminals of cell (i,j) can be read out as the E and N terminals of cells (i-1,j) and (i,j+1), respectively. Moreover, since the gate terminals ideally do not sink or source any current, the voltage at the generic gate terminal TG can be inferred from its connecting node, that is from either of the known power supply voltages or from one of the border terminals N, W, S, E. Consequently, all node voltages within the programmable transistor array are, in principle, accessible by the inner-cell probing facilities.

Based on the available node voltages, the voltage drop across the routing and multiplexer switches can be used to estimate the involved currents flowing through the generic transistor terminals TD and TS as well as through the routing connections. Assuming a precision of 1.25 mV (corresponding to a resolution of 12-bit) for the measurement of the node voltages and using an average resistance of $300\,\Omega$ as suggested by Fig. 3.5(a) for the resistance of the involved transmission gates yields a current resolution of approximately $4\,\mu A$. Yet, even after an elaborate calibration of the cell buffers, an accuracy of $10\,\mu A$ is probably more realistic.

3.6.2 Implementation of the Inner-cell Probing

Overview. The inner-cell probing is based on the cell buffer circuits included in the programmable transistor cell (see section 3.3.1). Fig. 3.26 illustrates the involved circuitry included in the transistor

[13] Actually, at the time of writing, this feature has not been verified by a hardware test due to limited time and need.

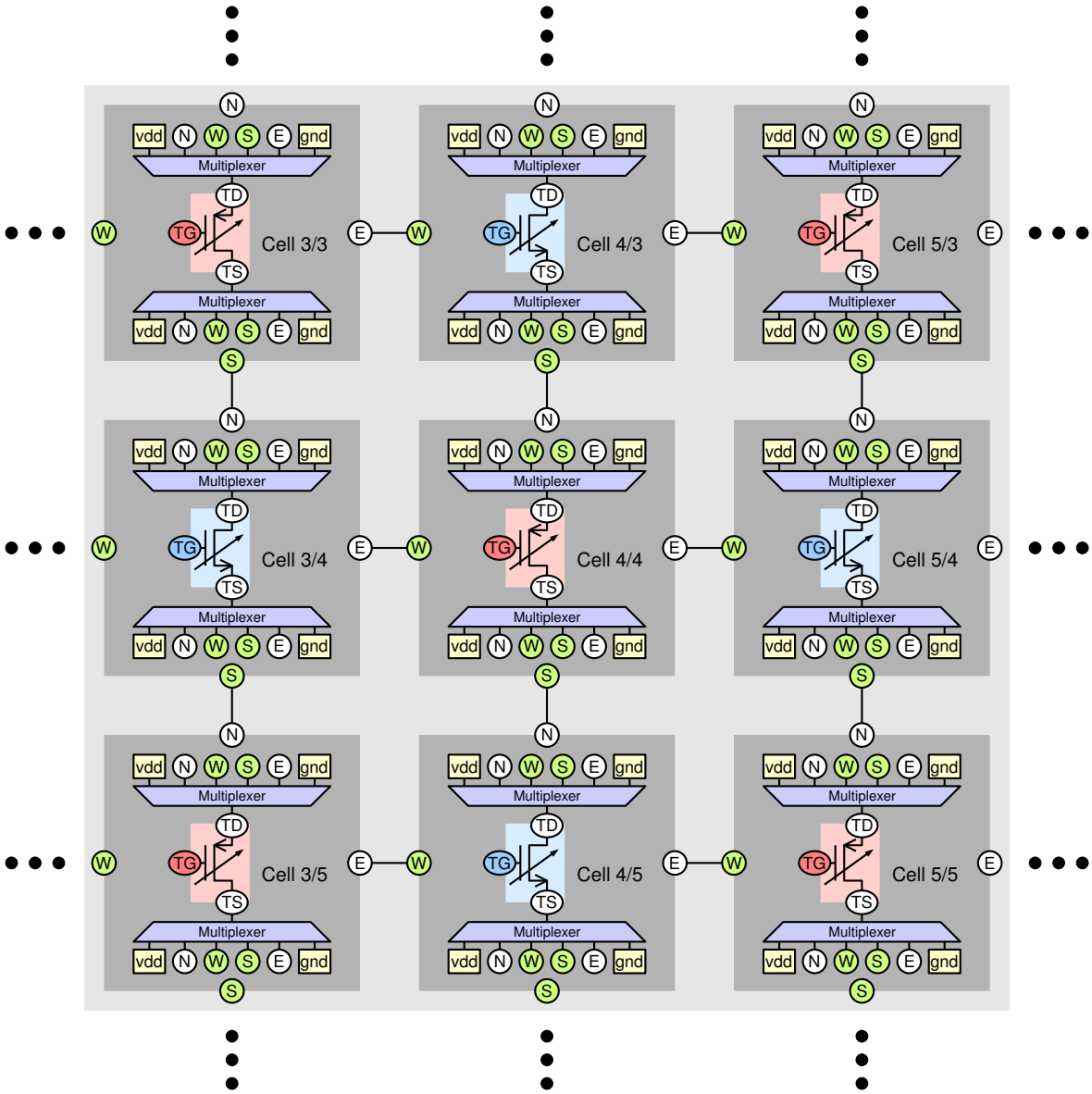


Figure 3.25: Cutout of 3×3 programmable transistor cells in a simplified schematic representation to elucidate the inner-cell probing principle. The whitely shaded circuit nodes N, E, TD and TS can be selected for analog readout.

cells themselves as well as the peripheral circuitry necessary for the row, column and terminal selection. The diagram depicts only the first and last row and column of the respective circuitry; 14 rows and columns are omitted. Two 5-bit decoders are used to produce two horizontal and two vertical select signals for each transistor cell; they are denoted as H<n> and V<m>, respectively. The chosen address is activated by a global $\overline{\text{PROBE\_EN}}$ signal, which is achieved through the NOR gates following the row and column decoders. If the $\overline{\text{PROBE\_EN}}$ signal goes low, exactly one horizonal and one vertical select signal are activated according to the address present on the address bus A<9:0.

The multiplexer included in each of the 256 transistor cells uses the two pairs of horizontal and vertical select signals to determine if and which of the four nodes N, E, TD and TS is to be connected to the cell buffer. The encoding of the five multiplexer states is summarized in Table 3.10. The

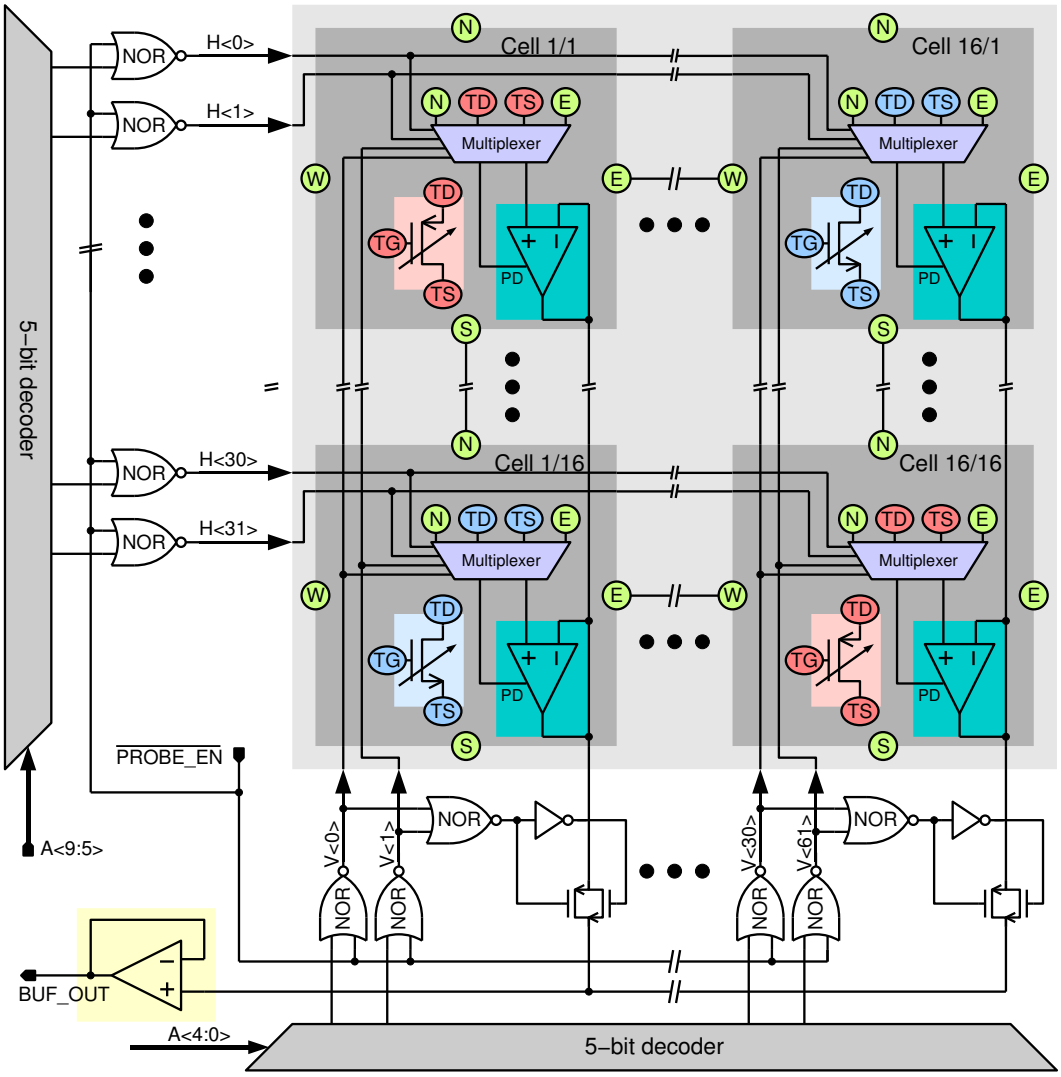


Figure 3.26: Architecture and control of the inner-cell probing facilities.

multiplexer also controls the state of the cell buffer: It is only powered up, if indeed one of the four cell nodes are to be buffered.

All buffer outputs of the same column share one node, which is feasible since only one of the 256 cell buffer in the whole array can be active at a time. This mitigates the need for an additional switch at the output of the cell buffer. If either one of the vertical select lines of one transistor cell column is activated, the buffer output of this column is connected to the global output buffer through a transmission gate.

Timing Diagram. The proposed architecture requires an input signal pattern similar to that one depicted in Fig. 3.27: After the desired transistor cell and terminal have been determined by the according address A<9:0>, the $\overline{\text{PROBE\_EN}}$ signal is used to activate the necessary select lines H<n> and V<m>. Now, the cell buffer, the global output buffer and any other off-chips circuits used for further processing have to settle to the probed node voltage, before the result can be quantized by an external ADC. The chosen buffer is switched off again, before the cycle can be repeated for the next terminal node to be probed.

V<0>	V<1>	H<0>	H<1>	$\overline{PD}$	COUT
1	0	1	0	0	S = SOUTH
1	0	0	1	0	TS = SOURCE
0	1	1	0	0	TD = DRAIN
0	1	0	1	0	E = EAST
X	X	0	0	1	Z
0	0	X	X	1	Z

Table 3.10: Encoding of the inner-cell signal multiplexing and the power down signal for the output buffer. Note, that at most one of each select line pairs V<0>,V<1> and H<0>,H<1> can be active, because each pair is generated by a multiplexer.

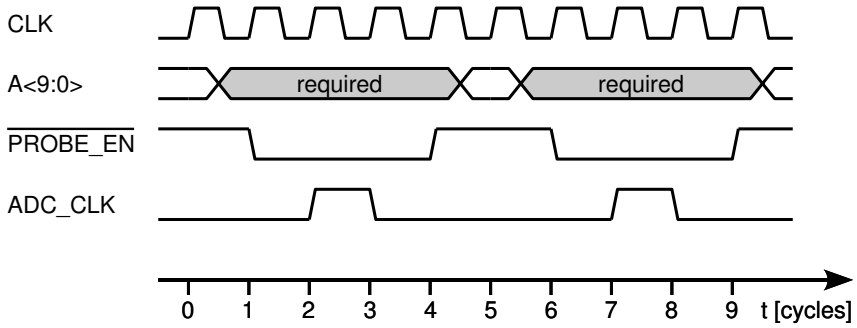


Figure 3.27: Timing diagram for the successive readout of two inner-cell nodes.

Inner-Cell Multiplexer. The cell buffer unit consisting of the cell buffer and the according multiplexer is implemented as shown in Fig. 3.28. The transmission gates that connect the selectable terminals N, E,TD and TS to the input of the cell buffer possess a channel width of 4.2 µm and 1.4 µm for the P- and NMOS transistor, respectively, which is more than sufficient with respect to the small capacitive load present at the input of the buffer. However, during the readout of either of the four node voltages, this capacitance is added to the selected node through the closed switch. The additional capacitance is calculated from the input capacitance of the cell buffer, the capacitance introduced by the necessary metal lines and the additional capacitance of the three open and the one closed transmission gate seen from the buffer input. It adds up to 266 to 290 fF. On one hand, this is of the same order as the capacitances inherent to the respective nodes to be probed and thus may very well affect the circuit behavior. On the other hand, compared to the sum of only those capacitances in the vicinity of the nodes to be probed, the influence of the additional capacitance introduced by the cell buffer node is relatively small and its effect should not deteriorate the circuit behavior too gravely in most situations.

Analog Considerations. The design of the rail-to-rail operational amplifier used for buffering the probed terminal voltages is discussed in section 3.7. Nevertheless, a few remarks concerning the analog intricacies inherent to the inner cell probing concept seem to be appropriate here: First, it is not until the $\overline{\text{PROBE\_EN}}$ changes to low that the selected cell buffer is powered up. According to transient circuit simulations, the time between the cell buffer being switched on and the final settling of the signal is dominated by the buffer's slew rate and settling time. However, the inner-cell probing concept might be enhanced by storing the information about the state of the amplifier within one of the two unused SRAM bits of the transistor cell.

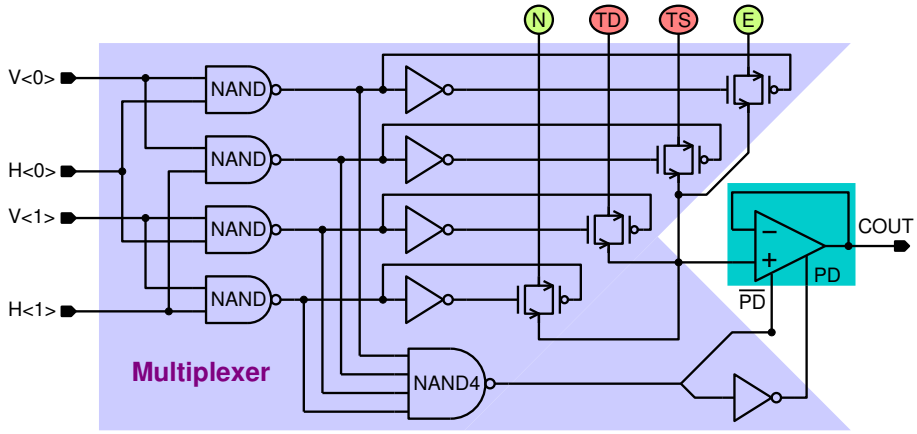


Figure 3.28: Inner-cell readout circuitry for one PMOS cell.

Second, the cell buffer has to drive a fairly large capacitive load, which was underestimated at its design-time. In fact, the cell buffer is optimized for driving capacitive loads in the order of 5 pF. Yet, a closer look at the involved capacities yields load values between 20 and 40 pF, of which approximately 10 pF can be attributed to the shared output node before the column-selecting transmission gate (in Fig. 3.25), while the rest is introduced by the global ANA_OUT line. 60% of the former contribution stem from the coupling to the input gates and Miller capacitors of the 15 deactivated buffers. They could be avoided by additional switches at the buffers' output. However, the more severe problem is caused by the large metal capacitance of the global ANA_OUT node, which encircles the whole PTA. In future, this problem could be mitigated by utilizing the buffers of the according row of IO-cells to buffer the output of the cell buffers. For the existing chip, this capacitive load will probably increase the settling time of the cell buffers by a factor of two, at least for input voltages between 1 and 4 V. According to simulations for typical mean process parameters, the cell buffers are still stable for capacitive loads of up to 40 pF, albeit exhibit a considerable decrease in phase margin to about 40°. While probably sufficient for the chips already fabricated, this phase margin would offer too little security for future submissions/designs.

Depending on the capacitive load, the buffer is simulated to settle to 0.1% of the input step within 100 to 200 ns in the range of 1 to 4 V and at least within 400 ns for voltages close to the power supply rails. Accordingly, sample rates between 2 and 5 V should be feasible. However, due to the limitation to one analog output, this rate may not be achievable if more than one cell node and or the voltage at one or more border cells must be read out.

As the amplification of the probed node voltages is performed by different cell buffers, precise measurements ,as e.g. necessary for estimating the node currents, will require an accurate calibration measurement of the offset distribution of the cell buffers. The respective offsets must then be subtracted from the measured data in some post-processing stage within the software.

3.7 Family of Rail-to-Rail Operational Amplifiers

The implementation of the FPTA chip described in the preceeding sections required three different operational amplifiers. All of them are connected as unity gain buffers and all of them serve a similar purpose, that is to buffer a signal for a subsequent processing stage. Since nothing particular is known about the voltage levels of the signal, all of the buffers must cover the full power supply range with their input and output compliance. Moreover, at least two of the three types of amplifiers must be

disconnectable and all of them shall consume relatively little current in order to limit the self-heating of the FPTA chip. However, while the cell buffer as a part of the transistor cell (depicted in Fig. 3.2) needs to be small to reduce area consumption and capacitive loading, the IO-cell buffer (Fig. 3.18) and the global output buffer (Fig. 3.1) do not suffer from this constraint and can thus be designed to be faster. As the output buffer needs to drive larger resistive and capacitive loads than the IO-cell buffer, different amplifier implementations deem necessary here, too. Owing to the similar requirements, all of the three amplifier designs are derived from the same type of topology, which is exemplarily discussed for the IO-cell buffer.

3.7.1 IO-cell Buffer

3.7.1.1 Design

The schematic of the IO-cell buffer is illustrated in Fig. 3.29. Its topology is inspired by the work of Milkovic [Mil85] referred to by Allen and Holberg [All02b] (section 7.1), Ferri et al. [Fer98], Loose [Loo99] (section 4.4.5 and 4.5.2), and Laker and Sansen [Lak94a] (section 6.7.2). The op amp consists of four main parts: An NMOS and a PMOS input stage to ensure proper operation over the full input voltage range, a class AB output stage to combine low output resistance with a modest quiescent current and a bias generator to bias the cascodes and current sink/source transistors M_{18} and M_{19}. The rail-to-rail nature of the design manifests itself in the fact that topologically in- and output stages are totally symmetric in their usage of P- and NMOS devices.

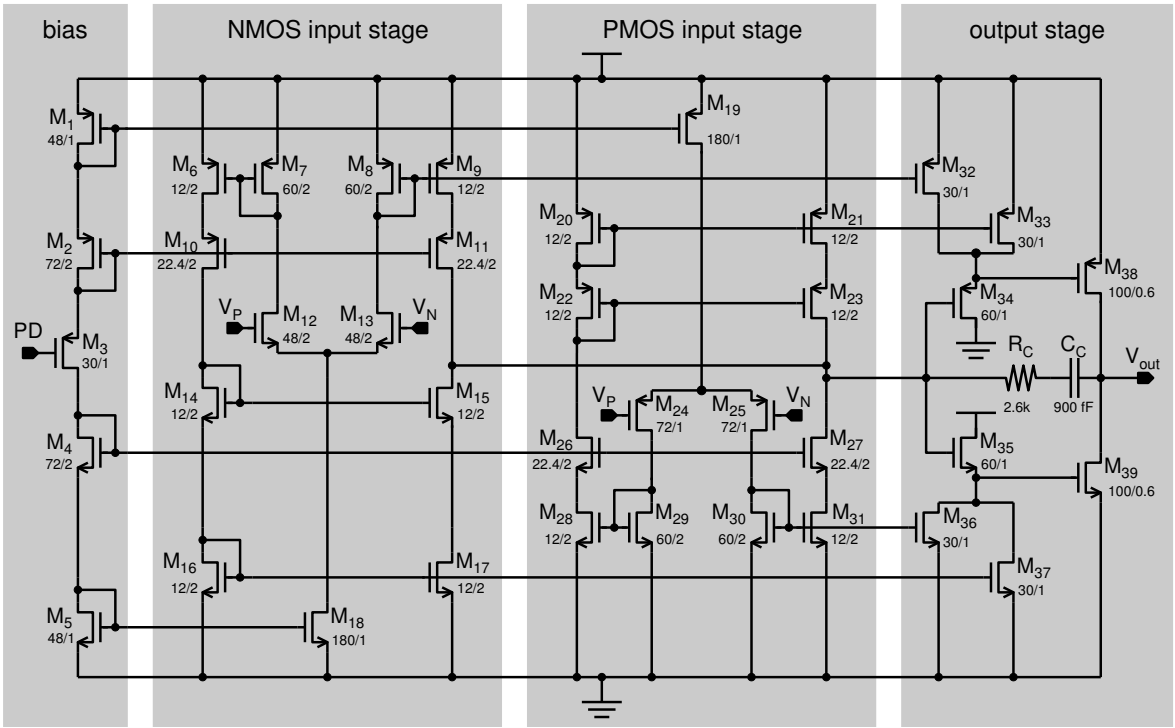


Figure 3.29: Schematic of the operational amplifier used as a buffer for the sample and hold units of the IO-cells. Some of the transistors controlling the power down are omitted for clarity. The transistor sizes are denoted in µm.

The input stages are implemented as cascode symmetrical CMOS OTAs[14] as discussed by Laker and Sansen [Lak94a], with the exception that one of the two cascodes is replaced by a second current mirror (transistors M_{14}, M_{15} and M_{22}, M_{23}). Some peculiarities of the implementation shall be exemplified by the NMOS input stage: The current mirrors consisting of M_6, M_7 and M_8, M_9 form a current amplifier with a current gain of $B \equiv W_7/W_6 = W_8/W_9 = 1/5$, which is considerably smaller than the values suggested by Laker and Sansen [Lak94a]. However, according to their discussion of the symmetrical OTA, this choice is motivated as follows: A small value of B is chosen to increase the PM[15], thus allowing for a smaller compensation capacitor C_C. The entailed decrease in GBP[16] and SR[17] is compensated by increasing the channel widths of the differential pair (M_{12} and M_{13}) and an increased bias current sunk by M_{18}. The P- and NMOS input pairs are sized to deliver the same transconductance g_m, which balances the contributions of both input stages, whose outputs are shorted as suggested by Ferri [Fer98] and Loose [Loo99].

The output stage consists of the push-pull amplifier formed by M_{38} and M_{39} and the source followers realized by M_{32} to M_{34} and M_{35} to M_{37}. The paramount task of the source followers is to split the gate voltages of the push-pull amplifier by two V_{GS} voltages to reduce their quiescent current. The push-pull amplifier is biased to work in the class AB operation mode to avoid crossover distortion. Each of the two source followers is actively biased by both input stages, which generalizes the concept of Milkovic [Mil85] to a complementary pair of input stages. The active biasing acts as a feedforward of the difference in the currents through the differential pairs and thus increases the gain of the output stage. The compensation capacitor C_C is inserted between the in- and output of the output stage to benefit from a Miller-like enlargement, which owes to the gain of the output stage. Except for the push-pull amplifier no minimum size transistors have been used to foster proper matching, which helps in reducing the offset and distortion of the fabricated amplifier circuits.

3.7.1.2 Layout

The layout of the rail-to-rail buffer is shown in Fig. 3.30. It occupies $161\,\mu m \times 101\,\mu m$ of silicon area. Signal routing is limited to the first two metal layers to allow for signal and power routing on the third metal layer. Device mismatch is reduced by using common centroid geometries (see e.g. [All02b], section 2.6) for the differential input pairs and current mirrors. Ground and n-well contacts are generously spent to reduce the influence of substrate noise and coupling.

3.7.2 Global Output Buffer

The global output buffer is a case of successful design-reuse in that it could be derived from the IO-cell buffer by resizing only the output stage. As will be shown in section 3.7.4, its performance does indeed closely resemble that of the IO-cell buffer. Yet, owing to the larger output stage, the global output buffer can drive larger capacitive and resistive loads, which is paid for by a higher quiescent current dominated by the push-pull amplifier. Schematic and layout view of the output buffer are presented at the top and bottom of Fig. 3.31, respectively.

Fortunately, as the global output buffer is not limited by severe geometrical restrictions, the layout thereof could reuse the layout of the IO-cell buffer to a large extent. The main difference is due to the increased push-pull amplifier and its according source followers, whose channel widths area increased to account for the increased gate capacities of the output transistors M_{38} and M_{39}. In addition, the

[14] Operational Transconductance Amplifiers
[15] Phase Margin
[16] Gain Bandwidth Product
[17] Slew Rate

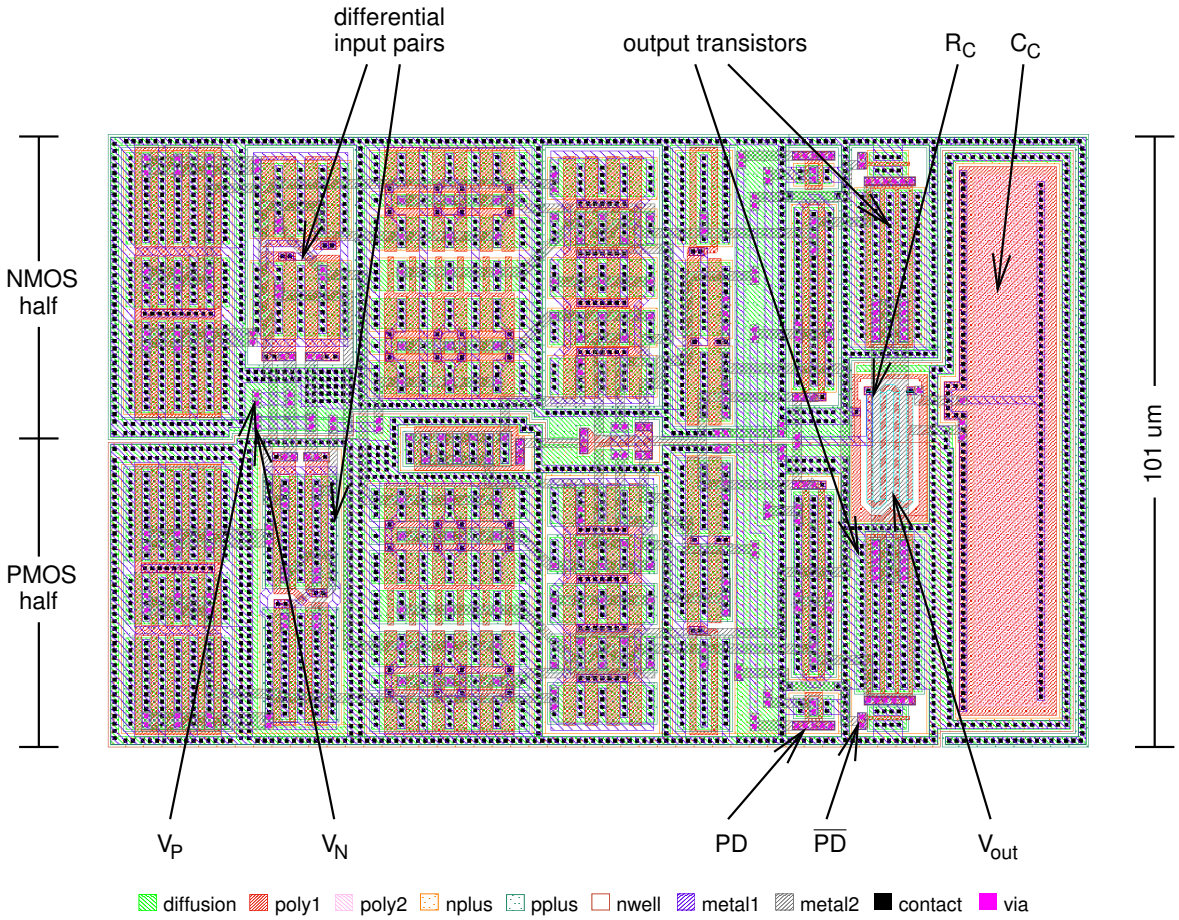


Figure 3.30: Layout of the rail-to-rail operational amplifier used in the sample and hold cells.

layout of the output buffer is surrounded by a wide power supply ring delivering the necessary currents in the range of 10 mA. The area beneath the power ring is again used for an ample supply of substrate contacts. The complete layout occupies a silicon area of 246.75 µm × 147.4 µm.

3.7.3 Cell Buffer

The circuit schematic of the cell buffer is depicted in Fig. 3.32. Its design is primarily driven by the necessity to fit it into the silicon area left in the IO-cell. Therefore, the goal was to implement a rail-to-rail buffer meeting the given area constraints whilst optimizing its electrical specifications. Consequently, the cell buffer performs worse in terms of gain, GBP and SR when compared to his *big brothers*. To reduce the layout size of the cell buffer, the cascodes in the input stages have been omitted, which reduces the transistor count by five. Moreover, the amplifier had to be completely resized, as can be seen from Fig. 3.32. However, the cell buffer design also utilizes the abovementioned current gain of $1/5$. The layout of the cell buffer measures approximately 30 µm × 140 µm and is included in the layout of the NMOS transistor cell presented in Fig. 3.9.

3.7.4 Summary of Simulated Performance

The three buffer implementations have been simulated for typical mean process parameters at a temperature of $T = 27\,^{\circ}\mathrm{C}$. The simulation results are summarized in Table 3.11. They are further detailed

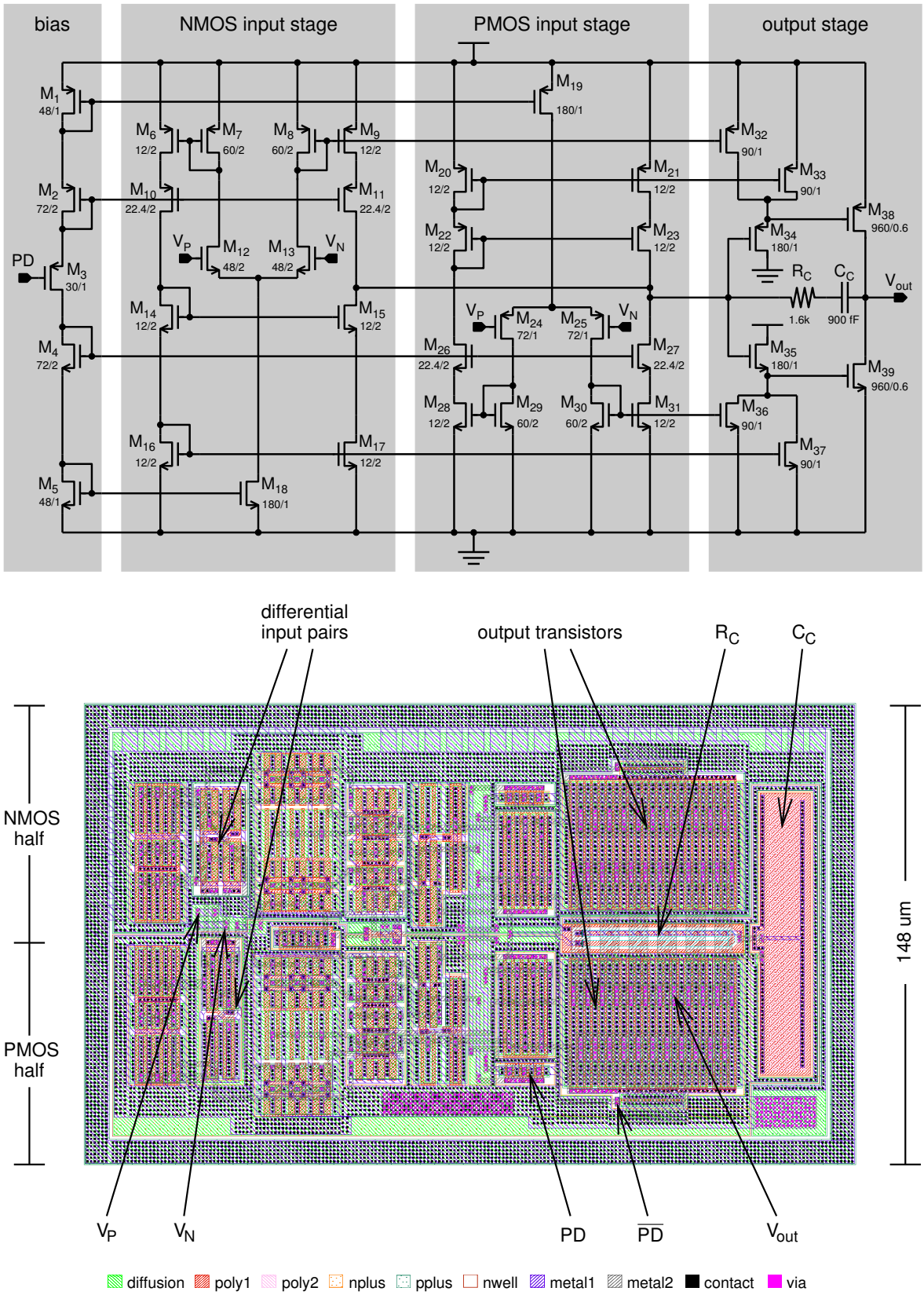


Figure 3.31: Top: schematic of the rail-to-rail operational amplifier used as the output buffer. Some of the transistors controlling the power down are omitted for clarity. The transistor dimensions are denoted in µm. **Bottom:** Layout of the output buffer.

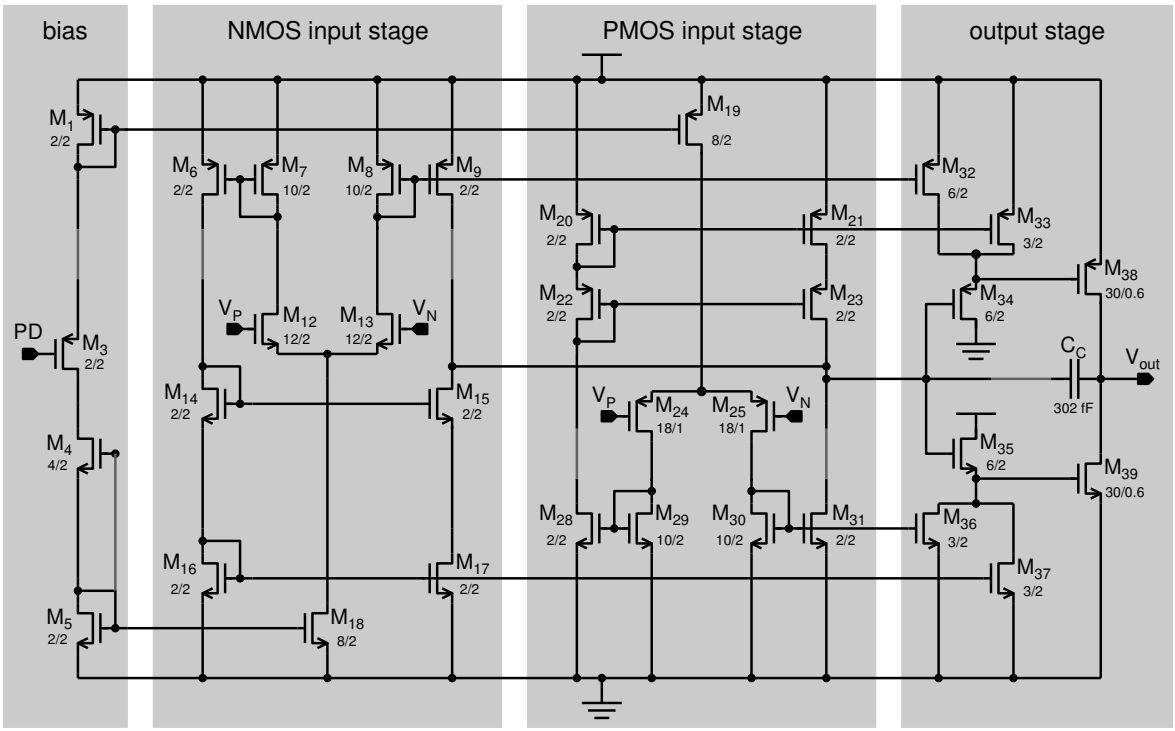


Figure 3.32: Schematic of the operational amplifier used as a buffer for probing of inner-cell nodes and currents. Some of the transistors controlling the power down are omitted for clarity. Transistor sized are denoted in µm. The wires replacing the devices omitted in comparison to the IO-cell buffer are colored in grey.

in a number of performance plots in appendix B. All results of Table 3.11 apply for an almost infinite load resistance of $R_{\text{load}} = 1\,\text{G}\Omega$, albeit a different set of capacitive loads: The IO-cell buffer was originally designed for capacitive loads of $C_{\text{load}} \approx 10\,\text{pF}$. The load capacitance of the ANA_OUT signal the buffer has to drive is estimated to a value between 10 and 30 pF. The global output buffer has been targeted at driving capacitive loads up to 100 pF, which is probably more than sufficient to drive the subsequent external circuitry. Finally, the cell buffer has been developed for driving capacitive loads of approximately 5 pF, whereas post-design considerations lead to an estimated capacitive load of 20 to 40 pF.

Table 3.11 accounts for the *best* situations in terms of the applicable in- and output voltage or frequency range as denoted in the according footnotes. For instance, the UGB[18] decreases for common mode voltages $V_{\text{CM}} \in [0\,\text{V}, 1\,\text{V}] \bigcup [4\,\text{V}, 1\,\text{V}]$ as can be observed from Fig. B.1(d), B.6(d) and B.10(d) in appendix B. First, the at V_{CM} around 1 and 4 V one of the two input transistor pairs ceases to conduct as its V_{GS} decreases below the threshold voltage V_{T}. The further drop for V_{CM} approaching either vdd or gnd closer than approximately 0.3 V is due to the decreasing transconductance g_m of the single active differential pair as it leaves the saturation region. This effect can be mitigated enlarging the W/L ratio of the according current mirror loads (M_7,M_8 and M_{29},M_{30}) relative to that one of the differential pair transistors. This is actually the second reason for choosing a small current gain B. The decrease of dynamic performance towards the power supply rails also manifests in the decreased SR and the increased settling times T_{settle} for common mode voltages in the vicinity of vdd and gnd.

Typically, the three buffers are operated in some kind of sample and hold fashion: The cell buffer has to drive the output line to the sensed input voltage subsequent to its start up for every measured voltage, the IO-buffer is indeed used in a sample and hold unit, and the output buffer has to drive

[18] Unity Gain Bandwidth

Buffer	**IO-cell**		**Output**	**Cell Buffer**		
$C_{\text{load}}[\text{pF}]$	10	30	100	5	20	40
$A_{\text{OL}}{}^{a}[\text{dB}]$	114	114	114	≥ 82	≥ 82	≥ 82
UGB^{b} [MHz]	≥ 55	≥ 38	≥ 52	≥ 27	≥ 27	≥ 27
PM^{c} [°]	≥ 71	≥ 51	≥ 67	≥ 69	≥ 42	≥ 31
$\text{SR}_{1\text{-}4}{}^{d,e}$ [$\frac{\text{V}}{\mu\text{s}}$]	87.9	85.5	100.2	39.3	36.1	29.0
$\text{SR}_{0\text{-}5}{}^{d,f}$ [$\frac{\text{V}}{\mu\text{s}}$]	12.6	14.5	12.2	7.4	9.3	6.4
CMRR^{g} [dB]	112	112	112	90	90	90
THD^{h} [dB]	≤ -85	≤ -85	≤ -85	≤ -80	≤ -80	≤ -81
$\text{T}_{\text{settle,1-4}}{}^{i,j}$ [ns]	51	61	51	107	140	270
$\text{T}_{\text{settle,0-5}}{}^{i,k}$ [ns]	391	388	403	432	428	417
$I_{\text{tot}}{}^{l}[\text{mA}]$	≤ 1.7	≤ 1.7	≤ 10.2	≤ 0.3	≤ 0.36	≤ 0.36

[a] Minimum for $V_{\text{CM}} \in [1\,\text{V}, 4\,\text{V}]$.
[b] Minimum for $V_{\text{CM}} \in [1.5\,\text{V}, 3.5\,\text{V}]$.
[c] Minimum over entire power supply range: $V_{\text{CM}} \in [0\,\text{V}, 5\,\text{V}]$.
[d] The slew rate is calculated between 10 and 90 % of the amplitude of the output step.
[e] Minimum for steps between 1 V and 4 V.
[f] Minimum for steps between 0 V and 5 V.
[g] Minimum for frequencies smaller 1 kHz.
[h] Maximum for input amplitudes $V_{\text{in}} \geq 0.5\,\text{V}$.
[i] T_{settle} is calculated for a precision of 0.1% of the output step.
[j] Maximum for steps between 1 V and 4 V.
[k] Maximum for steps between 0 V and 5 V.
[l] Maximum over entire power supply range: $V_{\text{CM}} \in [0\,\text{V}, 5\,\text{V}]$.

Table 3.11: Overview over the specifications of the three different rail-to-rail op amps for different capacitive loads.

the voltages that are multiplexed to its input from different IO-cells to the input of a – clocked – ADC. In short, the paramount dynamic specification is described by the time needed to settle to the output voltage at hand, which in case of the IO-cell buffer may be complicated by the fact that the input voltages may be randomly drawn from the entire power supply range, as for instance for the quasi-dc tests presented in chapters 5 and 6. Consequently, the three buffers are rather optimized for high SR and PM than for a high UGB to attain short settling times for large voltage steps and guarantee stability in the unity gain configuration. In case of global output and IO-cell buffer, the according settling times allow for sample rates of up to 20 MHz for $V_{\text{CM}} \in [1\,\text{V}, 4\,\text{V}]$ and 2.5 MHz if the full input voltage range has to be covered. The large signal bandwidth can be calculated by $f(A) = \frac{\text{SR}}{2\pi A}$ for an input amplitude A. In case of the global output and the IO-cell buffer, the large signal bandwidth thus amounts to approximately 15.9 MHz and 5.9 MHz for $A = 1\,\text{V}$ and $A = 2.4\,\text{V}$ when the according slew rates depicted in Fig. B.2 and B.7 are inserted. This in turn is in good accordance with the frequency spectra shown in Fig. B.3, B.4 and B.8

From the viewpoint of design automation, the simulation on which the results presented above are prohibitively elaborate and time-consuming. Nonetheless, the set of op amp specifications is

far from complete as the calculation of the PSRR[19] and a noise analysis are foregone. Moreover, the simulations have been restricted to one die temperature and did not account for the performance spread due to varying process parameters, nor did they extend to Monte Carlo techniques to capture offset distribution allowing for a more realistic value for the CMRR[20]. Accordingly, the performance evaluation within a design automation environment must be simplified to a reasonable amount of test cases that nonetheless samples the space of feasible input conditions densely enough to guarantee proper functionality.

3.8 Measurement of Die Temperature

Monolithic temperature sensors typically exploit a specific property of pn junctions inside of diodes or bipolar transistors to establish a voltage proportional to absolute temperature (PTAT) (cf. e.g. [Tim76], [Sza96], [Bro96], [Bak96], [Tut98]). Here, the concept is exemplified for the pnp transistor displayed on the left hand side of Fig. 3.33:

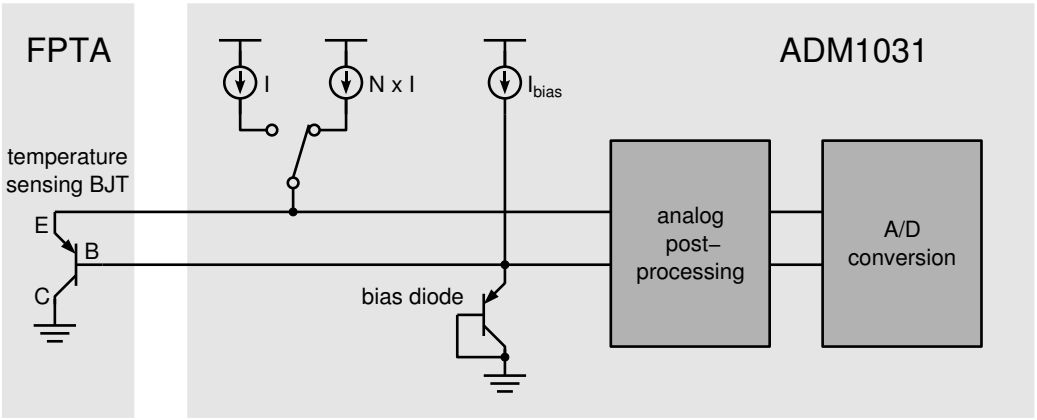


Figure 3.33: Schematic of the die temperature sensing concept using an ADM1031 device from Analog Devices, Inc.

The emitter current of a forward biased npn transistor can be described by

$$I_E = \frac{\beta - 1}{\beta} I_S = \frac{\beta - 1}{\beta} I_S \cdot e^{\frac{V_{\mathrm{EB}}}{kT/q}} \quad , \tag{3.4}$$

if the Early effect is neglected (see for example [Lak94a], section 2.1.4). Equ. (3.4) can be solved for the emitter base voltage V_{EB}, which depends on the absolute temperature T and the emitter current:

$$V_{\mathrm{EB}}(I_E, T) = \ln\left(\frac{\beta}{\beta - 1}\frac{I_E}{I_S(T)}\right) \cdot \frac{kT}{q} \quad . \tag{3.5}$$

For a pair of fixed emitter currents, $I_{E,1} = N \cdot I$ and $I_{E,2} = I$, the difference between the according base emitter voltages amounts to

$$\Delta V_{\mathrm{EB}}(N, T) \equiv V_{\mathrm{EB}}(N \cdot I, T) - V_{\mathrm{EB}}(I, T) = \ln(N)\frac{k}{q} \cdot T = \ln(N) \cdot 86.2\,\frac{\mu\mathrm{V}}{\mathrm{K}} \cdot T \quad , \tag{3.6}$$

which solely depends on the absolute temperature and the ratio of the two emitter currents. Hence, it is referred to as the desired PTAT voltage. Unfortunately, for the typically used current ratios of 8

[19] Power Supply Rejection Ratio
[20] Common Mode Rejection Ratio

[Tim76], [Tut98] or 10 [Sza96], the slope of ΔV_{EB} amounts only to a meager 200 $\frac{\mu V}{K}$. Accordingly, temperature measurements with an absolute accuracy around 1 K necessitate elaborate low-noise amplification as well as proper offset cancellation techniques. Viable candidates are, for example, auto-zeroed switched-capacitor circuits used by Tuthill [Tut98], or chopper-stabilized amplifiers proposed by Bakker and Huijsing [Bak96].

Though feasible, the design effort necessary to integrate either one of these concepts into the FPTA has been considered beyond the scope of the first prototype. As a compromise, four vertical pnp transistors are positioned next to the four corners of the PTA (as illustrated in Fig. 3.1 and 3.34), which have to be read out by an external device. The decision for vertical pnp transistors is based on the arguments of Bakker and Huijsing [Bak96] and Tuthill [Tut98], who point out that in CMOS technology, vertical pnp transistors are usually better suited for temperature sensing than their lateral counterparts. Since, in an n-well CMOS process, the vertical pnp transistors are implemented by using the substrate for the collector terminal, the collector terminal cannot be used, which decreases the number of possible readout chips.

Fig. 3.33 illustrates how the four pnp transistors on the FPTA could be used to determine its die temperature by means of the remote temperature sensors of the ADM1031 [Ana03b] offered by Analog Devices, Inc. : The sensing transistor is forward biased by the voltage drop across the bias diode. The base voltage is then compared to the emitter voltage at two different emitter currents. The difference between the two subsequent measurements of V_{EB} yields the desired ΔV_{EB}, which is converted to a digital expression for the measured temperature in °C in the ADM1031 chip.

3.9 Layout of the Complete Chip

Top Level Organization. From the annotated microphotograph depicted in Fig. 3.34, it can be seen that the global structure of the chip has already been captured by the schematic overview of Fig. 3.1. The main discrepancies are that the output buffer is located in the upper right corner instead of the upper left corner – that is in close vicinity to the analog output pads – and the fact that the bipolar transistors dedicated to the temperature measurement are smaller and placed closer to the PTA. The microphotograph also marks the locations of the probepads, that are placed as closely as possible to the chip boundaries, and points out the signal bus gathering the global digital signals use in and for the IO-cells as well as for the different sorts of address decoders adjacent to those four edges of the IO-cell blocks that point outward.

Pad Layout. All of the regular pads relevant to a normal operation of the chip are placed at its northern and western edges. The regular pads on its southern and eastern side are either power pads or connected to the base or emitter of the BJT[21] in the lower right corner. As the power pads on the northern and western edges of the chip should be sufficient for normal operation of the chip, this arrangement lends itself to directly connecting two or four FPTA chips together by means of the probepads. The parallel control of several FPTA chips with one set of digital signals is supported by a $\overline{\text{CHIP\_SEL}}$ signal, which must be low for accessing any of the enable signals controlling all the necessary controllers.

Power Supply Strategy. The FPTA chip features four different power supplies:

3.3 V **digital power (VDD33, GND33)** The tri-stated data bus DATA<0:5> runs on 3.3 V and therefore needs its own pair of power pads. All of the pads are located left to the SRAM control unit.

[21] Bipolar Junction Transistor

The extra voltage level of 3.3 V is imposed by the external circuitry of the PCI card Darkwing, which is used as the mixed-signal interface between chip and computer (see chapter 4).

5 V **digital power (DIGVDD, DIGGND)** The regular digital power supply is mainly used for the digital interface of the IO-cells, the SRAM control, digital pads and the decoders. It is separated from the analog power supply to increase analog signal integrity and reduce substrate noise. Moreover, the digital power supply is blocked by a total capacitance of 3.8 nF. The according power ring encircles the chip and uses the power lines in the pads whenever possible.

Analog power supply (VDDA, GNDA) The analog circuitry, that is, all of the circuitry in the PTA, except for the programmable transistor themselves and all of the analog circuits in the periphery, possesses its own analog power supply. The according power ring surrounds the chip at the outward pointing edge of the probepads, where the area beneath the according metal lines is used for capacitive blocking, which totals to 3.03 nF.

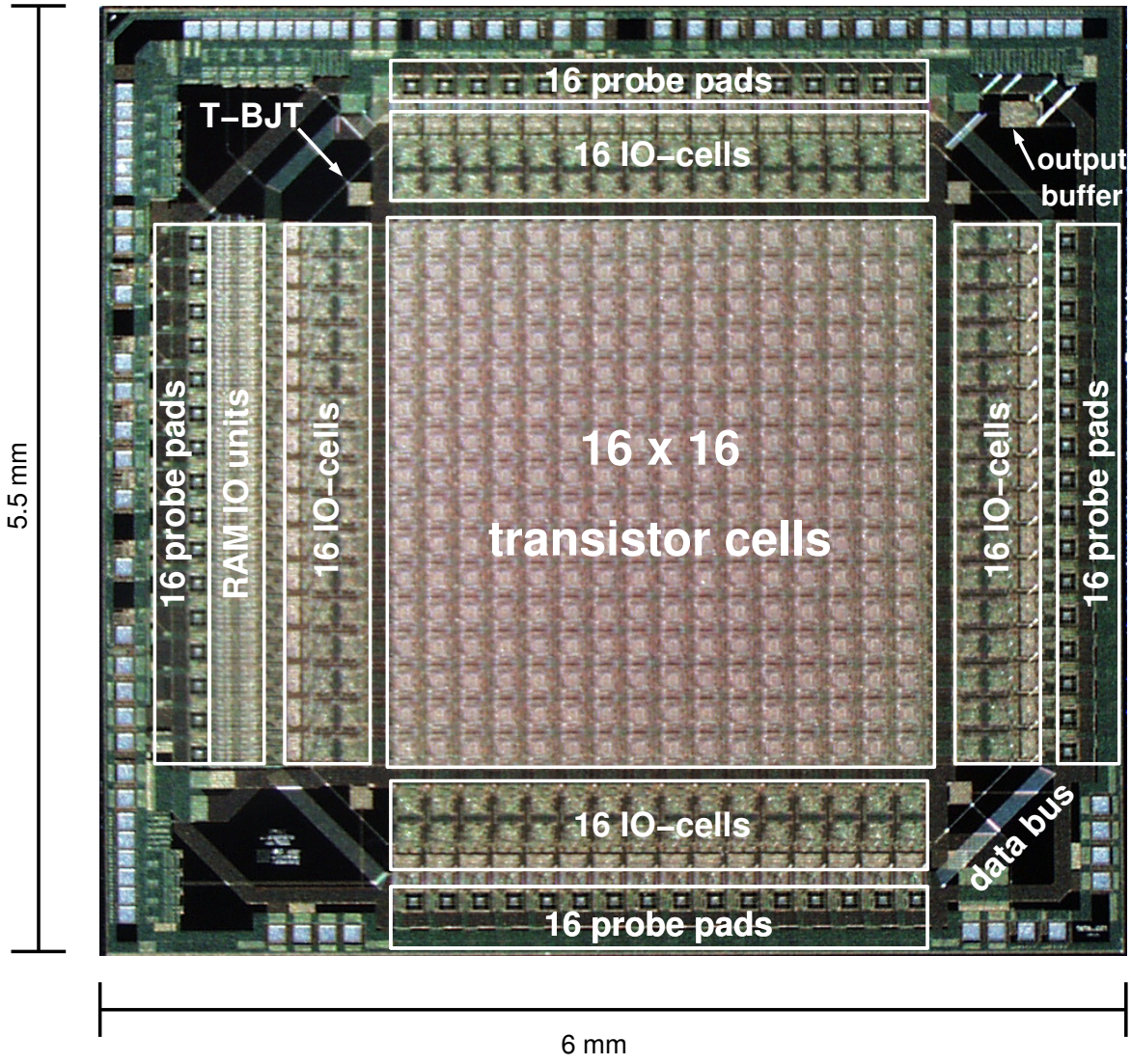


Figure 3.34: Microphotograph of the FPTA chip.

Transistor array power (ANAVDD, ANAGND) The gnd and vdd taps included in Fig. 3.2, from which the programmable transistors draw their power, possess their own power supply. The according power ring encloses the PTA. This choice allows to measure the power consumed by the candidate circuit at hand, to shut down its power supply in case of overheating, or to change the power supply voltages offered to the PTA. The latter possibility lends itself to the artificial evolution of low-voltage circuits.

Layout Sizes. The entire chip occupies $6 \times 5.5 = 33\,\mathrm{mm}^2$. However, only $3.2 \times 3.2\,\mathrm{mm}^2$, that is, a lean 30% are occupied by the transistor array itself. As the peripheral circuits grow only linear in size, this leaves room for larger PTA featuring 32×32 or even 64×64 cells. However, given the expected signal deterioration due to the parasitic resistors and capacitors, a simple enlargement of the PTA may not be the best alternative for the artificial evolution of more complex circuits.

3.10 Yield Analysis

A total of 30 FPTA chips was attained from the MPW run. Of these thirty, one chip was destroyed by the experimenter and 13 exhibited a short between the digital gnd pad and a regular signal (1) or a vdd pad of one of the analog power supply lines. Of these 13 shortcircuited chips, four were found to be defect after being bonded, while the remaining 9 were only tested on a wafer-prober. Thus, a destruction during the bond process can be eliminated as the main cause of failure. To date, a final explanation for the high failure rate has not been found, which leaves open the question whether the failure is caused by a weakness in the design or is the result of a fabrication problem. While five of the six bonded chips without shorted power supplies have been found to work, one possesses a column defect in the SRAM. This, however, may point to a yield rate that is conceivable for the given chip size and fabrication technology.

At least four of the five bonded chips have been extensively used throughout the last three years in various evolution experiments. The effective time they have been operated within an evolution loop is estimated to be between 0.5 and 1.5 years. During this time, none of these FPTA chip showed any sign of severe degradation.

3.11 Comparison of the Heidelberg and JPL FPTA Chips.

The work that most closely resembles the approach presented in this thesis is a research project on hardware evolution performed at the JPL by A. Stoica, R. Zebulum, D. Keymeulen et al. [JPL]. As has already been mentioned at the beginning of this chapter, the JPL project also focuses on artificial evolution of analog electronic circuits on a reconfigurable device referred to as an FPTA. Hence, this last section shall compare the two different approaches to FPTA-based intrinsic hardware evolution. However, it should be noted that the group, or members of the group, also published work on extrinsic hardware evolution. While part of this work is based on simulating a model of one of the group's FPTA chips, another part is devoted to experiments in which an unconstrained circuit representation is chosen.

FPTA Chips Developed at the JPL. To date, the efforts of Stoica's group have lead two three FPTA generations of increasing complexity. To distinguish the FPTA chip proposed in this thesis from the JPL FPTA chips, they will thus be referred to as FPTA0, FPTA1, and FPTA2, or FPTAx in short. The reconfigurable part of one of the elementary PTA cells used in the FPTA2 chip is depicted in Fig. 3.35. In case of the FPTA0 and FPTA1 chips, a programmable cell is similar to the part denoted as stage

1. While the FPTA0 consists of exactly one of these cells [Zeb00b], that is features eight transistors amenable to synthesizing new circuits connected by 24 switches, the FPTA1 chip hosts an array of these programmable cells [Sto00b], [Sto01d]. The FPTA2 however, features an array of 8×8 cells, whose core is given by the reconfigurable circuitry shown in Fig. 3.35 [Sto01c], [Sto02b], [Zeb03]. In addition, each cell also embodies some additional programmable non-transistor devices such as photo diodes, variable resistors and capacitors [Sto01c], [Zeb03]. Similar to the FPTA chip proposed here, the cells of the FPTA2 are also connected to the four nearest neighbours in North, West, North and East direction. Yet, unlike the FPTA proposed here, the FPTA2 cells possess in- and output terminals that suggest a predefined direction.

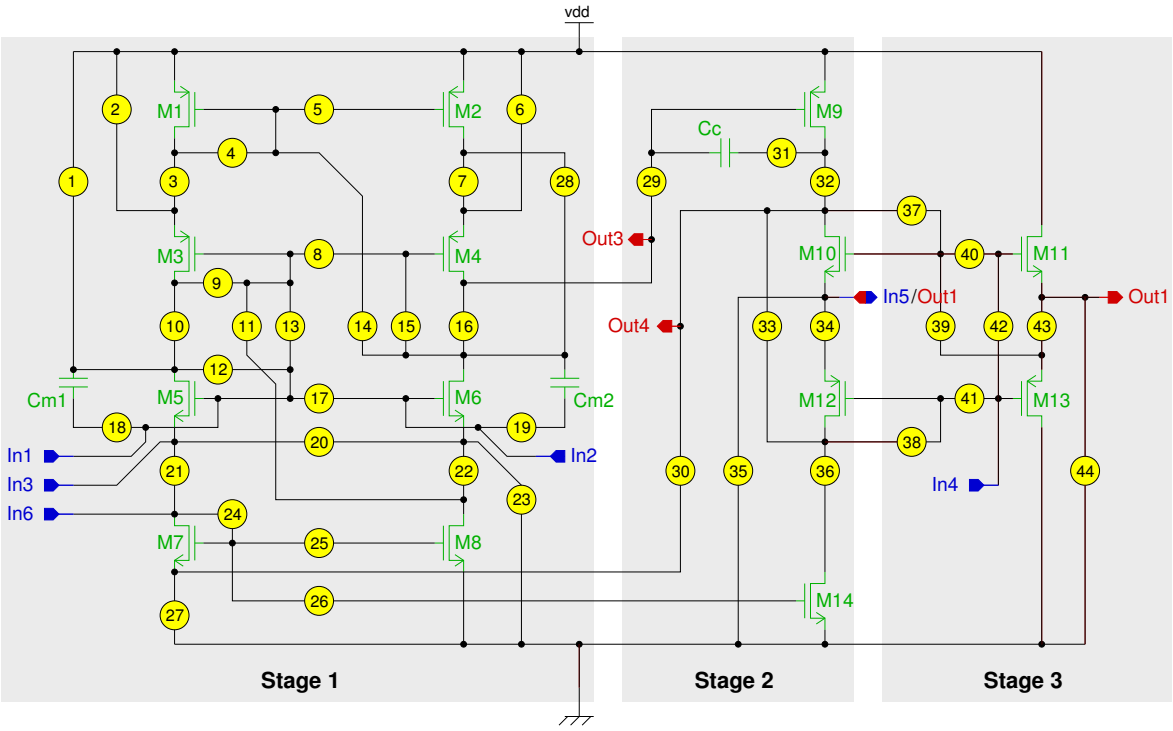


Figure 3.35: Schematic of the elementary transistor cell of the FPTA2 chip developed by the group of Adrian Stoica at the JPL. The encircled numbers denote switches. Figure was manually copied from [Sto01c] for better readability.

The reconfigurable part of the transistor cell illustrated in Fig. 3.35 consists of a total of 14 transistors used to synthesize the desired circuits. The configuration is achieved by a total of 44 switches. Besides, the cell embodies two 100 fF capacitors (C_{m1} and C_{m2} and the compensation capacitor $C_c = 5\,\mathrm{pF}$. The FPTA2 transistor cell has been demonstrated to be capable of hosting a variety of different building blocks and circuits including logic gates, common source amplifiers, and transconductance amplifiers [Zeb00b]. The latter structure can be extended to a two or three stage operational amplifier by means of the second an third stage.

Comparison of the Different FPTA Concepts. The proposed FPTA chip provides exactly one programmable transistor per cell, that is it provides the elementary unit of CMOS circuits, whereas the cells of the FPTAx chips possess some inner structure. On one hand, this implies that it is impossible to use any of the FPTAx cell's transistors in an unconstrained fashion. On the other hand, in case of the FPTA2, this increases the complexity of the circuits synthesizable on the chip. In principle, both types of FPTAs lend themselves to the evolution of analog or mixed signal circuits on the transistor

level as well as to circuit synthesis using predefined building blocks[22]. However, the FPTAx chips may induce a bias towards conventional circuit design due to their inner structure. While this may preclude the discovery of new circuits and concepts, it may also accelerate the hardware evolution process.

Another important difference between both FPTA concepts is that the analog substrate proposed here is restricted to CMOS transistors only, whereas the FPTAs developed by Stoica et al. feature several passive components, which can facilitate the synthesis process. In particular, capacitors are considered to be very helpful in implementing a desired frequency behavior. Yet, in the design of the proposed FPTA, such passive components have omitted intentionally to enforce the evolution of transistor-only circuits, which can be implementd in a larger variety of fabrication processes and are less expensive. Last, but not least, the FPTA chip presented and used in this thesis features programmable transistor dimensions, which yet offer another important aspect of circuit synthesis that is foregone in the FPTAx designs.

[22] For theFPTA chip proposed here this will be exemplified in chapter 8 for the FPTA

Chapter 4

Evolution System

USER, n. The word computer professionals use when they mean "idiot".

Claw Your Way to the Top
DAVE BARRY

The evolution system described in this chapter is both, one of the important results of the thesis described here as well as the research tool that allows for the hardware evolution experiments presented in chapters 5 to 8. The system can be divided into three parts: The FPTA chip serving as a an analog substrate for the test of candidate circuits, a mixed-signal test environment, and a software package that allows to implement the algorithms and define the experiments in terms of test patterns and target functionalities. While the implementation of the FPTA chip has already been detailed in the previous chapter, this chapter concentrates on a survey of the system as well as a description of those components that are most relevant to this thesis.

Hardware evolution of analog electronic circuits necessitates a computer program that simulates the artificial evolution process, whose general form has been described in Chapter 2. If it is to be intrinsic, it furthermore requires some sort of hardware that serves as a substrate to test the candidate solutions created by the evolutionary algorithm. The analog substrate, on which the experiments presented within this thesis are based, has been detailed in the previous section. However, the FPTA chip must be embedded in an appropriate mixed-signal test environment to be amenable to the evolution process. From the user's point of view it is even desired to integrate all necessary functionalities in a software package that provides the interface to the experimentalist who is interested in setting up a particular hardware evolution experiment. On one hand, the software has to encapsulate all of the low level functionalities. On the other hand, it must provide sufficient and convenient means to change the optimization algorithm as well as to formulate a large variety of test scenarios. The entirety of all

these components, that is, chip, external electronics, low level control thereof as well as the necessary software is referred to as the evolution system.

4.1 Overview of the Evolution System

The evolution system consists of three main parts: The software package DarkGAQT that is executed on a general purpose computer, the FPTA chip hosted by a dedicated carrier board, and an intermediate hardware level that interfaces the former two. The entire system as well as the signal flow between its main components is depicted in Fig. 4.1. During an evolution experiment, the evolutionary algorithm is executed on the computer as part of the DarkGAQT software. Thereby it generates a stream of candidate circuits that are passed to the PCI-interface card. The interface card is used to configure the FPTA chip and subsequently tests this individual. To accomplish this, the test data – stored in the local RAM on the interface card – is first converted into an analog test pattern that is applied to the FPTA. The circuit response of the individual under test is converted back into the digital domain before it is stored in the local memory. Eventually, the computer fetches the measured circuit response data from the local RAM and calculates the according fitness. After having tested all individuals of the current generation, the evolutionary algorithm proceeds by creating a new generation based on the fitness results recorded for the last generation.

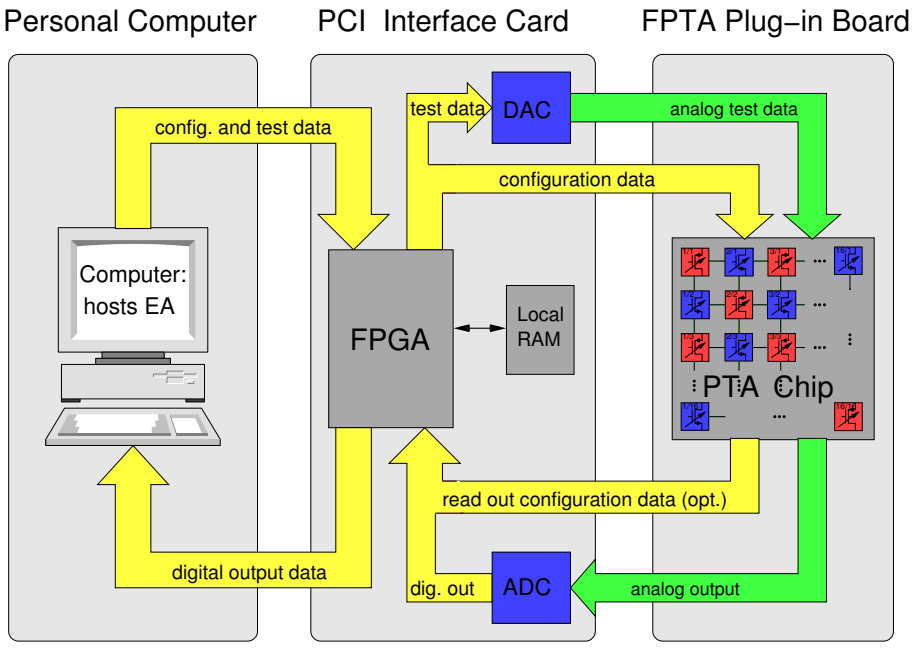


Figure 4.1: Overview of the evolution system.

Besides the execution of the evolutionary algorithm, the DarkGAQT software has to manage the test patterns, which are stored on the PCI interface card before the evolutionary loop is started. Moreover, it also has to ensure that all data of interest is stored and must provide the desired user interfaces. In the proposed project, the FPGA together with the local memory on the mixed-signal test card are primarily used to perform the circuit tests in real-time, independently of the interrupts triggered by the operation system running on the host computer. An early version of the evolution system is shown in Fig. 4.1. The experiments presented in chapter 5 are actually performed with a

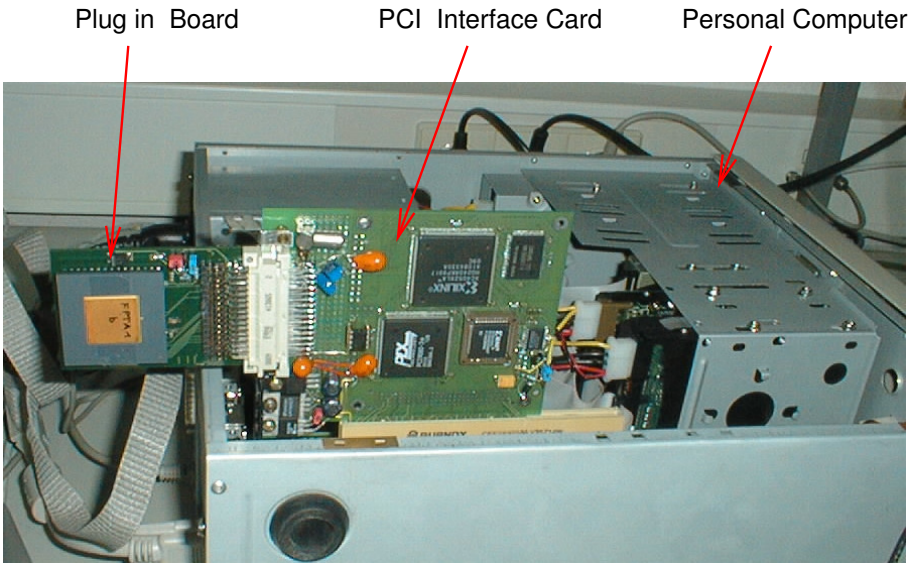


Figure 4.2: Photograph of an evolution system with the old PCI interface card.

similar setup. The remaining sections of this chapter will detail some aspects of functionality and control of the external electronics, and of the DarkGAQT software.

4.2 Mixed-Signal Test Environment

The mixed signal test environment serves two main purposes, namely the configuration of the FPTA and the test of the candidate circuits. As the hardware evolution approach heavily relies on the evaluation of many candidate circuits, a high testing rate is sought. Hence, re-configuration of the chip as well as the test procedure must be sufficiently fast. The desired timing for the re-configuration of the FPTA has already been stated in section 3.4.3. On the other hand, the evaluation of candidate circuits requires the synchronous generation of analog input voltages and control signals as well as the synchronous readout of the resulting outputs produced by the circuit under test. Thereby, a large signal bandwidth of up to 5 MHz must be covered according to the discussion of section 3.5.4. However, to allow for proper sampling of smaller signal variations, as e.g. in sinusoidal signals, the sampling rate must be considerably higher. In addition to the temporal requirements, it must be ensured that the fidelity and precision of the analog signals suffices the demands of the fitness evaluation at hand.

4.2.1 Electrical Test System: Background

The electrical test system outlined in Fig. 4.1 comprises three main components, namely a host computer, a PCI-based mixed signal test board, and a peripheral chip carrier board. The necessity for distinguishing between a mixed signal test board and the chip carrier board arises from practical rather than electrical reasons: The mixed signal test card serves as a research tool for three independent projects in the Electronic Vision(s) group. In addition to the project described here, the test board supports the characterization of high dynamic range sensors [Bre05] and the training of hardware neural network chips [Sch05] and [Hoh05].

Within the project described here, two different PCI-based mixed signal PCBs[1] have actually been used. The first one, which is a part of Fig. 4.2, was developed by Holger Blinzinger [Bli00] for a

[1] Printed Circuit Boards

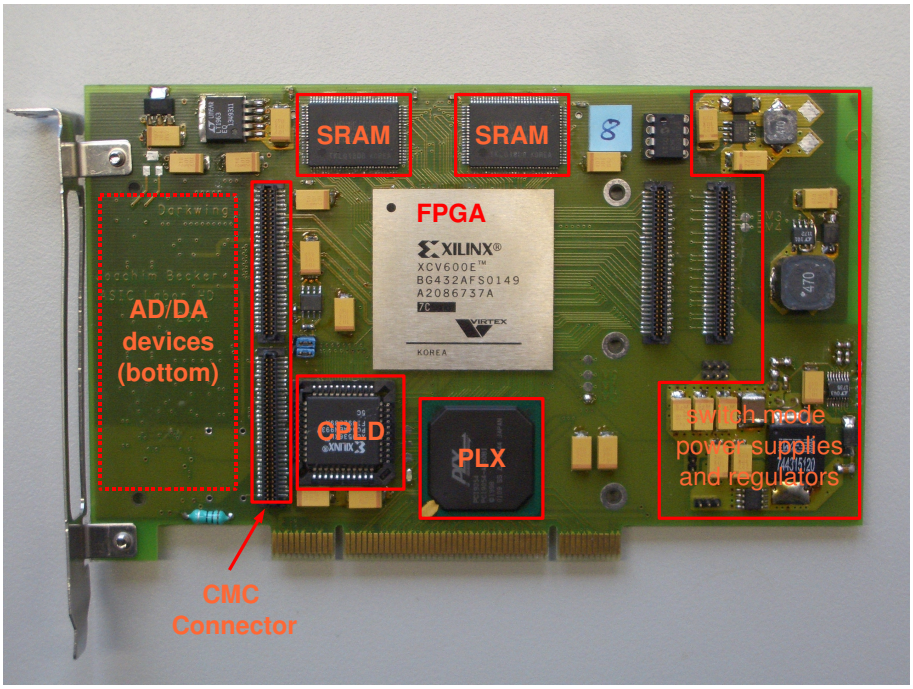


Figure 4.3: Annotated photograph of the top side of the Darkwing board. Photograph courtesy of Felix Schürmann.

different purpose and adapted to serve as a mixed signal test card by other members of the Electronic Vision(s) group. The second mixed signal test board developed by Joachim Becker [Bec01] includes the changes mentioned above and improves and extends the concept of the first one; its top side is depicted in Fig. 4.3. Since the latter test board — called Darkwing— has been used for all experiments except for those of chapter 5 and as it is currently the standard test board for the evolution systems described here, the further discussion concentrates on the Darkwing based test system.

Besides being a necessary prerequisite for this thesis, the available mixed signal test boards impose some constraints on the design of the chip carrier board — called Brightwing— and the FPTA chip itself. First, the connector between Brightwing and Darkwing is limited to 48 pins[2] , of which only 33 provide freely programmable digital signals. Second, the output voltage range of the DACs and the input voltage range of the ADC are limited so that neither DAC nor ADC can be directly used as an input or output, respectively. While the former constraint had to be considered in the FPTA design, the latter one necessitated further processing of the analog in- and output signals on the Brightwing board.

4.2.2 Test Environment: Digital Part

The electrical test system is summarized in Fig. 4.4. In case of the Darkwing board, only those components are considered that are relevant to this thesis. For a more thorough description the reader is referred to [Sch05] and [Bec01]. As far as the Brightwing board is concerned, only the current default setup that is usually used during an artificial evolution experiment is taken into account, where the connector strip at the bottom of Fig. 4.4 forms the only exception. While the operation of the

[2]Technically, the connection between Darkwing and Brightwing requires a further adapter board called Gosalyn. However, as the Gosalyn [Bec01] board merely translates the signals from a CMC (Common Mezzanine Card) connector to another connector format, the Gosalyn board is left out of consideration here.

digital components is summarized here, the analog signal path is discussed in the next section. The software controlling the programmable logic chip is dealt with in section 4.3. Most of the components of the Darkwing that are discussed below are annotated in the photograph of the board's top side depicted in Fig. 4.3. However, the following description of the digital components as well as the discussion of the analog signal path are essentially based on Fig. 4.4.

General Purpose Computer. The evolution system is hosted by an IBM compatible general purpose computer in the sense that the Darkwing board is inserted into one of its PCI slots. The computer architecture as well as the operation system are essentially a tradeoff between speed, effort necessary for the setup and maintenance thereof, and cost. In the course of the project, two types of processors have been in use, namely Athlon XP 1700+ and Intel Pentium IV 2.4 GHz processors. The host computers are operated under Linux, kernel 2.4.x. A more detailed account of the host computer system is available in [Sch05]. In principle, the general purpose computer, the Darkwing boards, and the software managing the communication between computer and hardware allow for the parallel operation of several Darkwing boards in one host computer. Although the parallelization of the candidate evaluation would result in a significant speedup, this feature is not yet taken advantage of due to a shortage in Darkwing boards.

FPGA and Local Memory. The main control instance of the Darkwing board is a Xilinx Virtex-E FPGA [Xil]. Within the Electronic Vision(s) group three different types that differ in the available number of logic blocks are used, namely XCV 300E, XCV 400E, and XCV 600E. However, for the evolution system described here, all of them have been successfully used. On one hand, the FPGA provides a relatively fast communication channel between the host computer and the FPTA chip. On the other hand, it controls the mixed signal circuitry dealing with the analog interface of the Brightwing board. In other words, the Darkwing board serves as a combination of a mixed signal pattern generator and a mixed signal scope, albeit not necessarily providing high frequency sampling. In order to ensure a synchronous operation, the Darkwing board provides some local memory. On one hand, this allows to store some information that remains unchanged during an evolution run, as e.g. the test pattern. On the other hand, the local memory serves as a buffer between host computer and FPGA in the sense that the host computer can transparently read and write to this local memory via DMA[3] burst accesses. Although the Darkwing board offers both, SDRAM[4] and SRAM, only those two Mbyte provided by the local SRAM are utilized as this proved to be sufficient and simpler to use due to the constant latency.

PCI Bus Interface. The FPGA is connected to the PCI bus via the PCI bridge PLX PCI 9054 [PLX00]. As the PCI bridge can be configured in different ways, the desired configuration is provided by a non-volatile memory, that is an EEPROM. The PCI bridge essentially implements the relatively complex PCI protocol [PCI95] and mediates between the local bus running at a user-specified frequency and the PCI bus running at 33 MHz. The PCI bus yields a maximum bandwidth of $33\,\text{MHz} \times 32\,\text{bit} = 132$ Mbytes/s. The PCI bus frequency results in a FPTA re-configuration time of approximately 100 µs as has already been explained in section 3.4.3. On one hand, this is not considered a severe limitation, on the other hand the configuration time might be further reduced by using the local memory on the Darkwing board to buffer the respective configuration information. The second operation that may be affected by the limited bandwidth of the PCI bus is the readout of the measurement results stored temporarily in the local SRAM. Thereby, each 32-bit word transports two measured voltages. The

[3] Direct Memory Access
[4] Synchronous Dynamic Random Access Memory

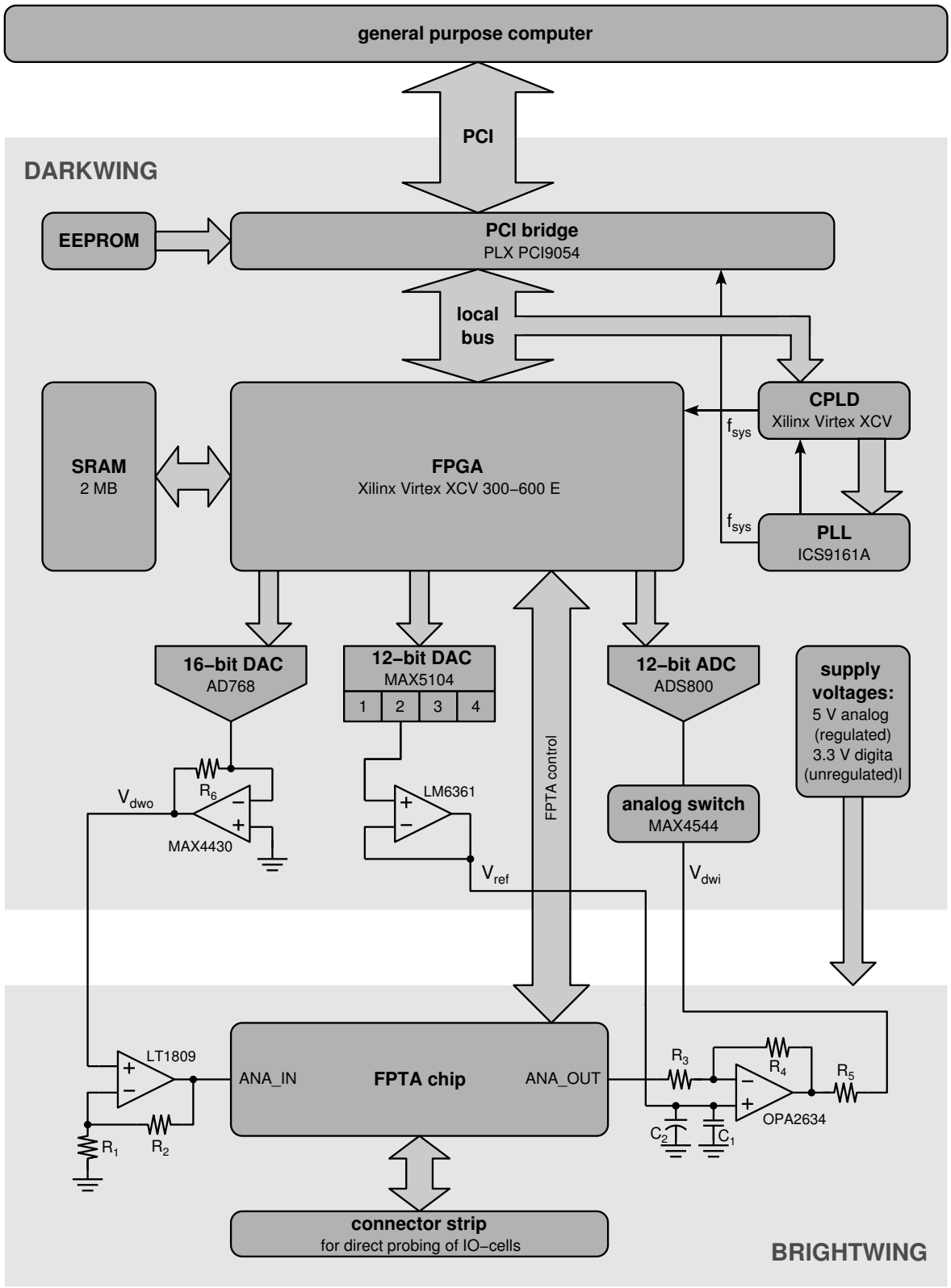


Figure 4.4: Block diagram of the electronic mixed-signal test environment. Its main components are: general purpose computer (top), Darkwing FPGAs interface card and Brightwing chip-hosting board.

maximum rate thus amounts to 66 MSPS[5], whereas a more realistic estimate on the basis of the practically achieved transfer rate [Sch05] results in 40 MSPS. Given that the measured data is read after the measurement process has terminated, this could prolong the measurement time for up to a factor of 1/2 in case of the maximum sample frequency of 20 MHz. However, in a typical experiment, the sampling rate will probably be in the order of 1 MHz, in which case the additional time for the readout of the measured data becomes negligible.

Configuration of the FPGA. Typically, the FPGA is to be configured at least once at the start of the respective software accessing the Darkwing board. The configuration is taken care of by the Xilinx CPLD XC9536XL chip [Xil04]. Owing to the PCI bridge, this can be accomplished via the PCI bus. The CPLD chip is also used to program the PLL[6] which derives the system clock from a reference crystal oscillator providing a frequency of 16 MHz. The system clock is utilized by the PCI bridge as well as by the FPGA. In the latter case, the clock amplitude must be reduced from 5 V to 3.3 V by the CPLD.

Power Supply. The Darkwing board provides several different power supply voltages that are derived from the supply voltages available from the PCI bus. In case of the Brightwing board, only the regulated analog 5 V supply and the unregulated digital 3.3 V supply are of interest. The latter 3.3 V supply is directly taken from the PCI bus and thus expected to be very noisy. However, this voltage is exclusively used for the supply of the tristated DATA<0:5> pads. Since the 3.3 V power is completely decoupled from the rest of the FPTA chip, the according noise contribution should be negligible.

4.2.3 Analog Signal Path

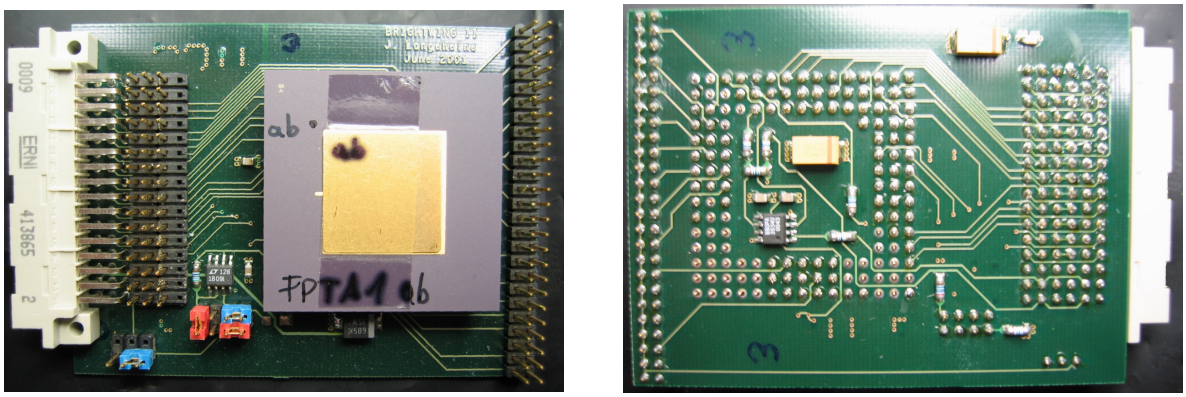
The analog signal path is depicted in the lower half of Fig. 4.4. It comprises the data converters and op amps belonging to the Darkwing board as well as the op amps and the FPTA chip located on the Brightwing board, which is depicted in Fig. 4.5. The analog loop starts with the digital-to-analog conversion performed in the AD768 and ends with the analog-to-digital conversion achieved by the ADS800. The following discussion adopts an FPTA centric perspective, that is, the analog path is divided into the external circuitry ending at the input of the FPTA chip and another branch that processes the signal available at the FPTA's analog output.

4.2.3.1 Analog Input

The generation of the input signals for the FPTA starts with a D/A conversion provided by the 16-bit DAC AD768 [Ana96]. Being a current steering DAC, the device is connected to source between 0 and 20 mA, which is converted into a voltage by means of R_6 and the MAX4430 [Max00] operational amplifier. With $R_6 = 200\,\Omega$, this results in an output swing of 0 to 4 V, which is already buffered. Since the input voltage required at the ANA_IN terminal of the FPTA chip must include both power supply rails, the output signal of the MAX4430 op amp must be further amplified on the Brightwing board. To achieve this, an LT1809 [Lin00b] op amp is used as a noninverting amplifier. If the output of the MAX4430 is denoted as V_{dwo}, the desired voltage at the input of the FPTA, $V_{\mathrm{ANA\_IN}}$ can be calculated by:

$$V_{\mathrm{ANA\_IN}} = V_{\mathrm{dwo}} \frac{R_1 + R_2}{R_1} = V_{\mathrm{dwo}} \frac{3\,\mathrm{k}\Omega + 2\,\mathrm{k}\Omega}{3\,\mathrm{k}\Omega} = \frac{5}{3} V_{\mathrm{dwo}} \ , \tag{4.1}$$

[5] Mega Samples Per Second
[6] Phase Locked Loop



(a) Brightwing top side

(b) Brightwing bottom side

Figure 4.5: Photographs of top (a) and bottom 4.5(b) of the chip carrier board Brightwing.

where $R_1 = 3\,\mathrm{k}\Omega$ and $R_2 = 2\,\mathrm{k}\Omega$ have been used. This particular choice reduces the feasible output range of the AD768 DAC by 25% and thus decreases its potential linearity more severely than necessary. However, the chosen gain was imposed by the inability of the LM6361 [Nat99] op amp that was used instead of the MAX4430 on the predecessor of the Darkwing board, because its output swing was limited to 3.3 V.

4.2.3.2 Analog Output

The analog output voltage of the FPTA again includes both of its power supply rails. However, the full scale input range of the 12-bit ADC ADS800 [Bur95] that is used for the analog-to-digital conversion stretches only from 0.25 to 4.25 V. Therefore, the output signal $V_{\mathrm{ANA\_OUT}}$ must not only be attenuated, but also be shifted to cover most of the available input range of the ADC. To accomplish this, an OPA2634 [Bur99] op amp is connected as an inverting amplifier as depicted in the lower right corner of Fig. 4.4. The necessary reference voltage V_{ref} that serves as a virtual ground is provided by the 12-bit DAC MAX5104 and buffered by the OPA2634 operational amplifier. The MAX5104 DAC provides four conversion channels, yet can only be used for slow varying signals due to its large conversion time of 12 µs.

For a given, fixed reference voltage V_{ref}, the output of the OPA2634 op amp can be calculated by

$$V_{\mathrm{dwi}} \;=\; V_{\mathrm{ref}} + (V_{\mathrm{ref}} - V_{\mathrm{ANA\_OUT}})\,\frac{R_4}{R_3} \;=\; \frac{3}{4}\,V_{\mathrm{ref}} + (V_{\mathrm{ref}} - V_{\mathrm{ANA\_OUT}}) \tag{4.2}$$

where $R_4 = 1.8\,\mathrm{k}\Omega$ and $R_3 = 2.4\,\mathrm{k}\Omega$ have been assumed. The gain is chosen as $3/4$ as to leave some security margin. Thus it is ensured that the output voltage V_{dwi} can always be mapped on the input range of the ADC independently of devices variations. The ideal reference voltage that places the output voltage range of V_{dwi} exactly in the middle of the ADC's full scale input range can be calculated to $V_{\mathrm{ref}} = 2.357\,\mathrm{V}$. In this case, the feasible interval of output voltages, $[0\,\mathrm{V}, 5\,\mathrm{V}]$, is mapped onto an interval of according V_{dwi} voltages that is given by $[4.125\,\mathrm{V}, 0.375\,\mathrm{V}]$. Eventually, prior to conversion, the transformed output voltage has to pass an MAX4544 analog switch, which might further decrease the signal fidelity.

In order to account for the limited precision of the resistor values as well as the reference voltage V_{ref} and potential offsets of the op amps, a three-step calibration procedure is mandatory. First, the lowest DAC input code that drives $V_{\mathrm{ANA\_IN}}$ to 5 V must be determined. Second, the references voltage V_{ref} needs to be adjusted to place V_{dwi} nicely in the middle of the ADC input range. Finally, the

conversion of the resulting ADC codes must be adapted such, that the voltages calculated from the ADC codes coincide with the real voltages present at the FPTA's output.

4.2.3.3 Analog Performance

All of the external operational amplifiers are chosen as to support the maximum sampling rates provided by the data converters, that is a minimum of 30 MSPS in case of the AD768 DAC and up to 40 MSPS for the ADS800 ADC; they settle to the desired output voltage in less than 37 ns[7] with a precision of at least 0.1%. The MAX4430 and the OPA2634 are operated such, that these values can be expected to hold over the entire range of feasible in- and output voltages of the FPTA. In contrast, the LT1809 op amp will suffer from similar performance losses as the op amps used in the FPTA chip itself (cf. section 3.7.4), because the output swing of the LT1809 also has to cover the device's own power supply range. Concretely, the op amp is expected to become slower and loose some of its open loop gain in the vicinity of the power supply rails. However, in the proposed evolution system this will probably not affect the overall performance too gravely, as the same effects apply to the op amps in the FPTA chip itself, whose settling times exceed those of the LT1809 by a factor of two.

Since all of the external devices in the analog path are faster than the analog in- and output amplifiers in the FPTA chip, the latter ones limit the maximum sample rate for uncorrelated large signals to 20 MHz, which has been set forth in section 3.7.4. In fact, the series of amplifiers will add a considerable delay to the analog signal. Without proper modeling or measurement of the exact timing conditions, a sample rate of 20 MHz will not be feasible for uncorrelated signals. On the other hand, a high sampling rate may be beneficial for sampling a relatively slowly varying signal like a sine wave of a frequency less than 1 MHz. Typically[8], the evolution system is synchronously operated with a clock frequency of $f_{\text{sys}} = 40\,\text{MHz}$. In accordance with the DAC specifications and the maximum sample rates dictated by the FPTA, the clock signals for the data converters on the Darkwing board are limited to 20 MHz. Firstly, the data converters are expected to exhibit better performance below their maximum sample frequencies. Secondly, this relaxes the timing constraints for the necessary digital control signals. Finally, it is still possible to increase the frequency of the system clock, provided that the logic design hosted by the FPTA can be executed at higher clock speeds.

The overall analog signal path is relatively long in that it involves three operational amplifiers to get the signal to the input of the programmable transistor array on the FPTA and four op amps to take it from the output cell of the programmable transistor array to the ADC. As none of these devices is ideal, they will all increase the noise and distortion of the analog signal. At least those two external op amps on the Brightwing board could be avoided as they are owing to the insufficient in- and output range of the mixed signal channels on the Darkwing board. Moreover, the signal range transformations reduce the used fraction of the dynamic range of the data converters and therefore further decrease the analog precision. Thus, future implementations may improve this situation by shifting the mixed signal devices to the successor of the Brightwing board. However, this may require the generation of further regulated analog voltages.

4.3 Hardware Control Software

As has already been mentioned above, it is the responsibility of the FPGA to control the digital and mixed signal peripheral devices on the Darkwing board. Furthermore, it serves as the digital interface

[7] The MAX4430 op amp is reported to settle within 37 ns to a precision of 0.015% [Max00]. Thus, it is believed to comply with the 33 ns required for a sample rate of 30 MSPS if the desired precision is reduced to 0.1%.

[8] The system clock frequency was set to 36 MHz for the experiments presented in chapter 6 and to 40 MHz for those described in chapters 7 and 8.

to the respective ASIC. Being a reconfigurable logic device, the FPGA can be programmed relatively easily to host a wide variety of digital control circuits. This flexibility is of great importance in the sense that it makes the Darkwing board amenable to being used for different projects of the Electronic Vision(s) group. Moreover, it allows for gradual improvements of the system at hand: Time-critical functionalities may first be realized on a software level, subsequently be transferred to the FPGA and may finally even make their way into the digital part of a mixed signal ASIC. In case of the hardware evolution system, it is conceivable to migrate computationally expensive data manipulations, as e.g. Fourier transforms from the host computer to the FPGA. Another option would be the integration of the evolutionary coprocessor developed and used in the related hardware neural network project [Sch03], [Hoh05], [Sch05].

The control circuitry residing in the FPGA is described in the high-level developing language VHDL [Des97]. In order to attain the configuration bitstring necessary to program the FPGA, the following steps have to be taken: First, the written VHDL code is verified by means of a simulation. Second, the code is translated into a netlist (here the FPGA Compiler II offered by Synopsys [Syn01] is used) of generic library devices. Third, the netlist is finally mapped onto the technology at hand, which is the Xilinx VirtexE device in this case, but could also be another suited FPGA or the standard cell library of an appropriate CMOS process (here, this is achieved by the Xilinx ISE [Xil03]). Provided that the place and route process succeeded in finding a feasible solution, the resulting delays inherent to the physical circuit have to be annotated back to the netlist. If the behavior obtained from the simulation of the back-annotated netlist is acceptable, the bitstring can be used to configure the FPGA.

In the context of design automation, it is interesting to note that the design procedure described above is greatly alleviated by the available synthesis tools. In comparison to the analog design process surveyed in section 1.5, the logic design is described on a much more abstract level, that is a programming language. The synthesis of the according netlist and the generation of the configuration bit string or physical layout are automatically performed by software tools. Besides the significant reduction in human design effort, this also facilitates the migration of existing logic modules to new technologies. However, it should be mentioned that, practically, the digital design process is not always as straightforward as suggested above. In addition, migration to a different technology may nevertheless require some changes in the VHDL code.

4.3.1 Overview

The programmable logic design currently in use in the hardware evolution system is sketched in Fig. 4.1 by means of a simplified block diagram. As the FPGA is to control the peripheral components of the Darkwing board, the structure of the logic design must somehow reflect the structure of the Darkwing board itself. Accordingly, the logic design provides interfaces to the PCI bus, the local memory and to the FPTA chip.

4.3.1.1 General Darkwing specific modules.

The three modules depicted in the upper part of Fig. 4.6, that is, the *PCI/IO*, *SRAM control*, and *main control*, are specific to the Darkwing board, but independent of the group-specific hardware for whose test the Darkwing board is actually used. In other words, they can similarly be used in other projects of the Electronic Vision(s) group.[9] The VHDL code describing these modules was written by Dr. Johannes Schemmel and Tillmann Schmitz.

[9]The main control must usually be adapted to integrate the project-specific modules at hand.

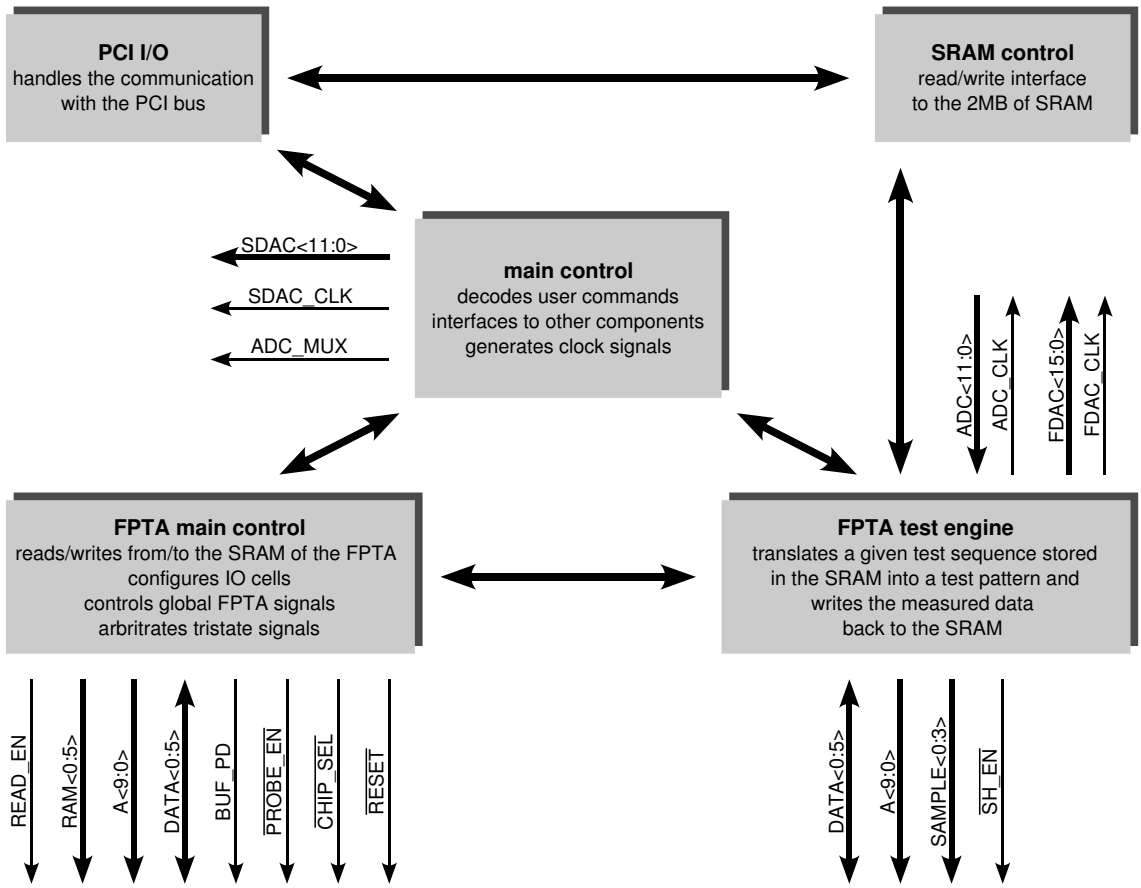


Figure 4.6: Simplified block diagram of the VHDL modules executed on the FPGA. The FPTA signals are explained in appendix A.5. Those signals controlling the FPTA's SRAM are gathered in RAM<0:5>. The signals relating to the mixed signal devices on the Darkwing board are denoted as follows: FDAC refers to the fast AD768 DAC, SDAC to the slow 12-bit DAC, and ADC_MUX to the analog switch multiplexing the two analog input channels to the ADC.

The *main control* module integrates and connects all peripheral modules and provides an interface between the host computer and the state-machines executed on the FPGA. In particular, the main control provides access to several control registers, which can be read out and written to by the host computer via the PCI bus. First, this allows to configure some features of the Darkwing board, as, for example, the analog multiplexer selecting the input to the ADS800 ADC, henceforth referred to as fast DAC or FDAC. Second, the *main control* registers provide a means to directly access the data converters and their clock enabling signals from the host computer. Third, the *main control* module implement a mechanism to trigger the start of user-specific processes on the FPGA and allow the host computer to acquire information about these processes. For instance, in case of the *FPTA test engine*, the test pattern sequence is started by setting a bin in the according register. While the *FPTA test engine* is running, the computer polls another register, which is changed after the data acquisition process has terminated. This signals to the software running on the host computer that the measured data can now be read out from the local memory.

The *PCI/IO* implements the interface to the PCI bridge and thus enables the communication with the host computer. It is the prerequisite for the abovementioned communication process between the logic design residing in the FPGA and the software executed on the host computer. The *SRAM control*

module manages the communication with the local memory on the Darkwing board. In the simplified illustration of Fig. 4.1, the *SRAM control* module handles both, the (direct) memory access of the host computer as well as the requests from within the FPGA.

4.3.1.2 FPTA-specific modules

The part of the logic design that controls the analog and digital signals exchanged with the FPTA is split into two modules: While the *FPTA main control* is liable for the configuration of the FPTA, the *FPTA test engine* generates the digital signals controlling the analog test of the candidate circuit at hand. As the latter module is vital to the experiments presented in chapters 6 to 8 in that it realizes a means for real-time testing, the next section is devoted to its functional description. More concretely, the *FPTA main control* allows to write to and read from the SRAM of the FPTA chip as well as to configure the IO-cells of the FPTA. Moreover, the *FPTA main control* sets the necessary global signals BUF_PD, $\overline{\text{CHIP\_SELECT}}$, and $\overline{\text{RESET}}$ and deactivates the inner cell probing mechanism.

To realize the read and write access to the FPTA's SRAM, the timing diagrams depicted in Fig. 3.13 and 3.14 are implemented by the respective VHDL code. Thereby, the packages of six bits that are written to or read from the respective flip-flops or latches of the FPTA are directly accessed through the PCI bus. The according state-machine is described in [Bec01]. Since, according to the discussion in section 3.4.3, the possible increase in the time needed for re-configuring the FPTA has been estimated to be of minor importance in most experiments, the simple communication scheme described above has not been altered yet. However, an improved version that exploits the full bandwidth of the PCI bus and possibly uses the local memory on the Darkwing board to buffer parts of the configuration bit string for the FPTA is desired in the long run.

The configuration of the IO-cells resembles that of the SRAM communication. It is implemented by writing to a dedicated register bank and transferring the resulting information into the IO-cells according to the timing diagram of Fig. 3.19. Hence, a full configuration of all 64 IO-cells requires 128 write accesses across the PCI bus. It should be noted, that the *FPTA main control* also provides the possibility of an analog test of candidate circuits configured into the FPTA. Yet, as this is also achieved by direct register accesses through the PCI bus, the resulting test patterns cannot be guaranteed to follow a predefined timing scheme. Although rather an advantage for the quasi-dc tests proposed in chapter 5, the asynchronous candidate tests are in general disadvantageous and are thus depreciated in the current version of the evolution system. However, they are indeed used for the experiments presented in chapter 5. Finally, the *FPTA main control* must arbitrate the signals that may be used by both FPTA related modules. Examples of which are the data and address bus of the FPTA, and the respective control and data lines of the data converters.

4.3.2 Analog Test Engine

As any real physical process is bound to happen in time, time is an important input parameter for the specification of electronic circuits. Thus, the hardware evolution system must provide a means to evaluate the temporal behavior of the evolving candidate circuits. To accomplish this, the *FPTA test engine* in conjunction with the according software interfaces allow the user to specify the relative temporal position of the different phases and actions occurring in the test pattern generation and data acquisition processes. Thereby, these relative temporal positions refer to the system clock f_{sys}, which is typically set to 40 MHz during evolution experiments and verification tests.

Generally speaking, the test of a candidate circuits must cover a variety of different specifications. This may involve different types of circuit analysis (cf. 1.4.2) as well as different test benches. On one hand, the test bench may be simply realized by the configuration of the IO-cells and the particular

Algorithm 4.1: Principal measurement procedure during an evolution experiment.

```
write test patterns to SRAM ;
write IO-cell configuration ;
for all candidate evaluations do
    download candidate circuit ;
    for all test modes tm do
        if number of test modes > 1 then
            download IO-cell configuration IOCONF[tm] ;
            if new test bench required then
                download new candidate circuit mixed with test bench ;
            end if
        end if
        run test sequence TEST[tm] ;
    end for
end for
```

test pattern. On the other hand, it may also utilize part of the PTA to provide further routing of in- and output signals or to load the output of the circuit under test[10]. Owing to the resulting need for different test setups, the circuit test is partitioned into different *test modes*, which can be edited via the DarkGAQT software. Each test mode allows for a separate IO-cell configuration, test sequence, and test bench configuration on the PTA.

From the viewpoint of circuit testing, the course of an evolutionary algorithm run can be summarized by Alg. 4.1: At first, the system has to be initialized by the following two steps: For one, the test sequences for all test modes are stored on the local memory of the Darkwing board. For the other, the IO-cell configuration of the (first) test mode is written to the FPTA chip. Within the evolutionary loop, the circuit test starts with the download of its respective configuration bit string to the FPTA. If only one test mode is used, the according test sequence is then executed by the FPGA. If more than one test mode is used, the IO-cell configuration must be reloaded prior to running the test sequence. Furthermore, the utilization of different test benches in the sense of partial PTA configurations also requires an appropriate reconfiguration of the FPTA before each test.

The test sequences of different test modes are stored in different regions of the SRAM. For each run of a test sequence, the host computer passes the according SRAM address to the test engine in the FPGA. The *FPTA test engine* subsequently executes the *measurement instructions* stored in a contiguous block of the local memory on the Darkwing board. The test sequence ends upon a special instruction code that indicates the termination condition. As detailed in chapters 5 and 6, some circuits evaluations necessitate the test patterns to be randomized. To accomplish this, the same test sequence is stored multiple times in the local SRAM, where each copy provides a different random order of the test cases. For each circuit test, the DarkGAQT software randomly chooses one viable random order and passes the respective memory address to the *FPTA test engine*. The number of random orders can be specified by the user with the only constraint that the resulting test sequence information must not exceed the size of the SRAM.

4.3.2.1 Test Sequence Elements

To facilitate the subsequent discussion of the test sequence, the element it is composed of shall be denoted as a test sequence element (TSE). Accordingly, a test sequence, which describes the tempo-

[10]This mechanism is only used in the experiments presented in section 8.5.

rally coherent process of test pattern application and data acquisition, is a sequence of TSEs, which are encoded as *measurement instructions*. A TSE defines the application of one input voltage to a set of PTA boundary cells, the process of sampling a different set of PTA boundary cells, the subsequent readout of one analog voltage stored in an IO-cell, and the settling time between the in- and output sampling.

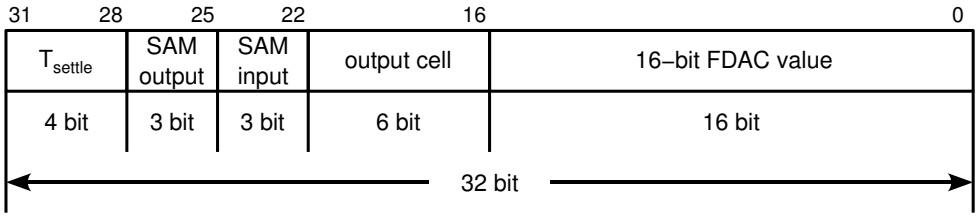


Figure 4.7: Encoding of an instruction for the mixed-mode test processor. Two 32-bit instructions are stored in one 64-bit word accessible from the SRAM.

One measurement instruction is encoded in 32 bits as shown in Fig. 4.7. In addition to the 16 bit wide FDAC value, which defines the analog input voltage, the instruction contains the information necessary to realize the desired sampling processes. Here, the IO-cells are operated in S/H operation mode 1, which is described in section 3.5.3.1. Accordingly, three bits are required to choose between the seven possible sources for the sample signals. While the bits 25 to 27 define the sample signal that shall be used sampling the analog input voltage, bits 22 to 24 determine which sample signal is activated for sampling the output(s) at the respective PTA boundary cell(s). The output cell, whose analog voltage is to be read out by the global buffer of the FPTA and is to be converted by the ADC, is specified by bits 16 to 21. Finally, T_{settle}, encoded by the last four bits 28 to 31, specifies the number of clock cycles the *FPTA test engine* is to wait between starting the input sample phase and terminating the output sample phase.

In the most general case, one TSE comprises the application of an input voltage and the sampling and readout of an output voltage. However, if multiple input and output voltages are to be applied and sampled for evaluating one test case, not all of the three actions are necessary or even allowed for all TSEs. Therefore, one measurement instruction can also encode different types of TSEs, which are summarized in Table 4.9.

First, if n input voltages shall be applied, the first $n-1$ TSEs are restricted to applying an input voltage, which is indicated by setting the output sample signal code to '000'. The n^{th} TSE then will do both, apply the last input and sample the first output voltage. Second, if m output voltages shall

	T_{settle}	SAM input	SAM output	A<out>	16-bit FDAC value
input + output	VVVV	VVV	VVV	VV VVVV	VVVV VVVV VVVV VVVV
input only	VVVV	VVV	000	XX XXXX	VVVV VVVV VVVV VVVV
output only	VVVV	000	VVV	VV VVVV	XXXX XXXX XXXX XXXX
output, no sample	VVVV	000	000	VV VVVV	1111 1111 1111 1111
sequence stop	1111	000	000	00 0000	1111 1111 1111 1111

Table 4.1: Instruction codes for the mixed mode test processor. While 'V' indicates that the value of the bit is evaluated, X' denotes a "don't care".

be read out, the TSEs $n+1$ to $n+m-1$ are restricted to sampling an output voltage and allowing for the subsequent conversion by the ADC. This type of TSE is encoded by a measurement instruction in which the input sample signal is set to zero. However, often it is not required or even desired to sample the outputs at different times wasting precious sample signals. Hence, in this case all sample signals would be sampled with the n^{th} TSE. The last $m-1$ TSEs are then restricted to reading out the output voltages sampled and hold by the respective IO-cells. The according instruction code, as well as the instruction code used to indicate the end of the test sequence are listed in Table 4.1. Note, that the codes for termination and for the readout of a sampled output voltage without sampling cannot be mixed up with the other three TSE types, yet are themselves defined ambiguously. Although a maximum settling time for reading out sampled output voltages is rarely used, this ambiguity must be remedied in future versions of the VHDL code. However, this can simply be achieved by choosing different FDAC codes.

4.3.2.2 Timing Diagram

The general measurement scenario featuring multiple in- and outputs is illustrated in the timing diagram of Fig. 4.8 for $n = 2$ in- and $m = 3$ outputs. Here, the FDAC values, the timing of the sample signals SAM<0:3>, and the address of the output cells A<7:2> are defined by the respective TSEs. Please note, that in case of the latter two, the information in the measurement instructions must comply with the chosen IO-cell configuration to attain the desired results. Analog to the timing diagram shown in Fig. 3.21, the outputs of cells i and j are sampled at once via SAM<2>. and cell k is sampled via SAM<3>. Thus all of the first four TSE types listed in Table 4.1 are applied here. The FDAC_CLK and ADC_CLK signals are generated automatically by the *FPTA test engine* based on the timing information included in the measurement instructions.

In principle, the settling time for each type of non-terminating TSE is well defined and can be used to adjust the timing, even if not all or none of the sample signals are actually used. In case of input-only TSEs, the settling time T_{settle} defines the number of clock cycles between two input sample phases $\Delta T_{\text{in}} = T_{\text{settle}} + 2$. In the according DarkGAQT user interface, this settling time can be configured once for each test mode and is denoted as `TSin`[11], which is the notation used in Fig. 4.8. The limitation to one global `TSin` per test mode should not be too severe, since the sequential application of input voltages is a restriction imposed by the external hardware and the structure of the analog IO of the FPTA chip. Hence, typically `TSin` is minimized under the constraint of a predefined analog precision. In case of an IO TSE, the settling time determines the number of clock cycles between the activation of the input sample signal and the deactivation of the output sample signal. Again, the exact number of cycles is calculated by $\Delta T_{\text{IO}} = T_{\text{settle}} + 2 = \texttt{TsettleIn} + 2$, where `TsettleIn` corresponds to the DarkGAQT notation. Finally, for output-only TSEs, the settling time between two successive output sample processes is given by $\Delta T_{\text{out}} = T_{\text{settle}} + 2 = \texttt{TsettleOut} + 2$. Thereby, the time between two readouts (and A/D conversion) of sampled output voltages is also defined — even if no output is sampled. The settling times `TsettleIn` and `TsettleOut` can be set individually for each output of each test case in the according DarkGAQT user interface.

The DarkGAQT software allows to adjust two further parameters of the *FPTA test engine*. First, the variable `FDAC2Sample` specifies the number of clock cycles between the activation of the FDAC_-CLK and the SAM<0> signal. For the remainder of this thesis, this value has been set to one, which corresponds to the situation depicted in Fig. 4.8. This allows the input voltage $V_{\text{ANA\_IN}}$ to partially adapt its new value, before it is sampled on the hold capacitor of the respective IO-cell. The second parameter `InputSampleSettleMax` determines the number of cycles for which the input sample

[11] Throughout this chapter, entities that are part of a software, as e.g. the DarkGAQT program, are printed in `typewriter` font.

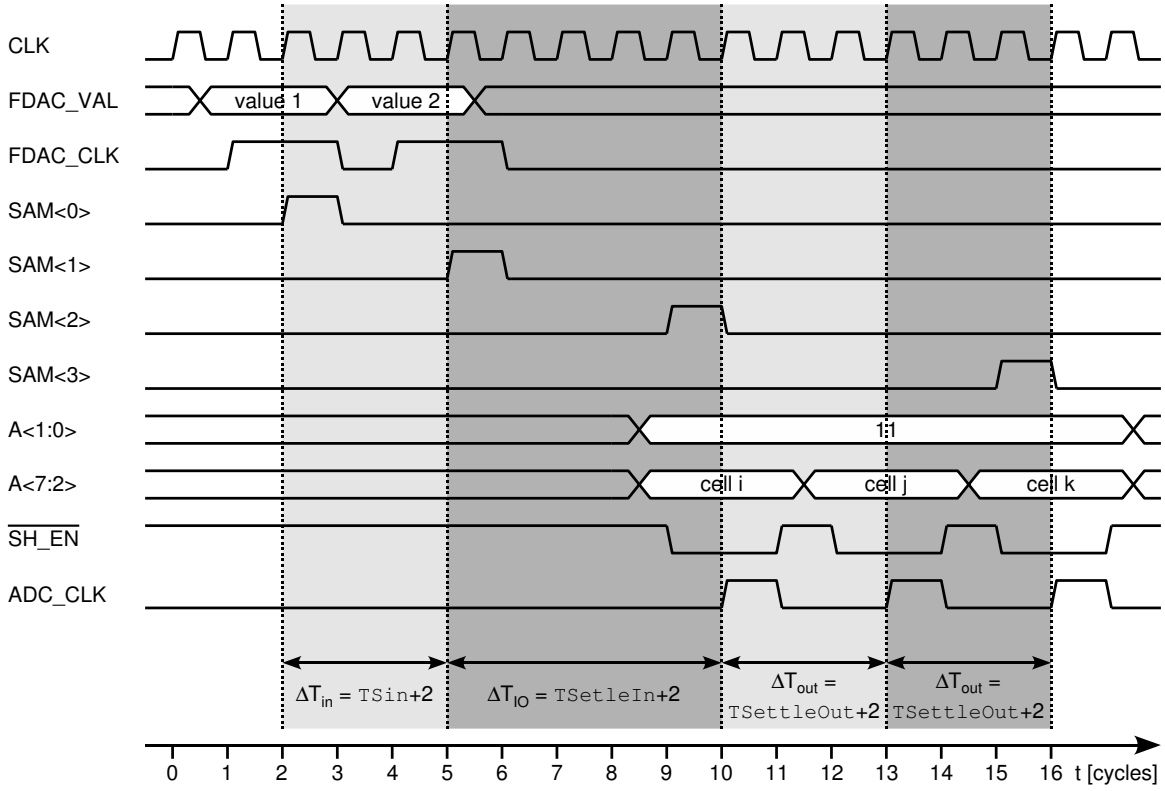


Figure 4.8: Timing diagram for a measurement comprising the application of two input and the readout of three output voltages.

signal can be maximally active in case of long settling times. It was experimentally verified that `InputSampleSettleMax` should be equal to or larger than four to avoid an impaired signal quality. Altogether, these parameters may be adjusted in order to optimize the analog precision of the hardware evolution system at hand.

4.3.2.3 FPTA Test Engine: Implementation

Due to the encoding of one TSE (depicted in Fig. 4.7), the settling time T_{settle} can only be chosen from 16 different values. In order to increase this rather small dynamic range, a global multiplier $1 \leq \texttt{TSMul} \leq 16$ (DarkGAQT notation) is introduced. Though sufficiently practical for most problems, this solution is disadvantageous for frequency sweeps, since they require an impractical large number of test cases for large dynamic ranges as e.g. used in the experiments of chapter 7. However, the particular encoding of the TSE meets concerns about the maximum rate with which the measurement

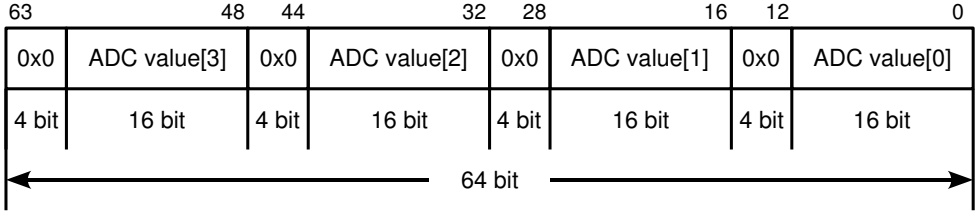


Figure 4.9: Structure of the SRAM words stored by the analog test engine.

Algorithm 4.2: Simplified and serialized procedure for generating the control signals for one contiguous test sequence. The aggregation of four ADC values into one SRAM word is omitted for brevity.

```
repeat
    WORD<63:0> ← SRAM;                                          // fetch new data from SRAM
    INSTR[0]<31:0> ← WORD<63:32> ;
    INSTR[1]<31:0> ← WORD<31:0> ;
    for i ← 0 to 1 do
        T_settle[i]<7:0>   ←  INSTR[i]<31:28> · TSMUL;          // parse instruction
        SAM_IN[i]<2:0>     ←  INSTR[i]<27:25>;
        SAM_OUT[i]<2:0>    ←  INSTR[i]<24:22>;
        CELL_OUT[i]<5:0>   ←  INSTR[i]<21:16>;
        FDAC_VAL[i]<15:0>  ←  INSTR[i]<15:0>;
        // execute one measurement instruction
        if SAM_IN[i]<2:0> ≠ '000' then
            set DAC to FDAC_VAL[i] ;
            apply resulting V_ANA_IN to IO-cells controlled by SAM_IN[i] ;
        end if
        wait for T_settle[i] cycles ;
        if SAM_OUT[i]<2:0> ≠ '000' then
            sample one or more PTA boundary signals via SAM_OUT[i] ;
        end if
        if (SAM_OUT[i]<2:0> ≠ '000') || ((SAM_OUT[i]<2:0> == '000') && (FDAC_VAL[i] ==
            '0xFFFF')) then
            read out IO-Cell CELL_OUT[i] and apply to V_ANA_OUT ;
            convert V_ANA_OUT to ADCVAL[i] ;
            store ADCVAL[i] to SRAM;
        end if
    end for
until (INSTR[1] == termination code) || (INSTR[0] == termination code)
```

instructions can be fetched from the local SRAM. As this problem only arises for maximum sample rates, dynamic range and therefore convenience was traded off against speed here. In this vein, two measurement instructions are stored in one 64 bit wide SRAM word to make best use of the bandwidth and capacity available from the SRAM. Likewise, four ADC values are gathered before they are written back to the SRAM as one word. The concrete encoding is illustrated in Fig. 4.9.

Altogether, the principle operation of the *FPTA test engine* can be described by Alg. 4.2. In short, the engine fetches a word from the local memory on the Darkwing board, parses the according two measurement instructions, realizes the according TSEs, aggregates the resulting ADC data, and subsequently writes it back to the local memory. Nevertheless, though the procedure is presented in a serialized form in Alg. 4.2, the actual VHDL implementation is split into several parallel state machines to decouple processes that feature different periodicities as, for instance, fetching new instruction from the SRAM, executing one TSE or writing back the ADC values. Moreover, to provide useful functionality for small sample rates requires some signals and processes must be pipelined. For example, the FDAC_CLK signal for TSE $n+1$ is already activated before the output sample signal of TSE n is deactivated. Finally, the maximum sample rate was chosen to be half of the system clock f_{sys} for the following reasons: First, the sampling rate of 20 MHz for a typical $f_{\text{sys}} = 40\,\text{MHz}$ are already close to the maximum sample frequency guaranteed by the FDAC. Second, the execution of

one measurement instruction per system clock would have unnecessarily increased the complexity of the VHDL design.

4.4 DarkGAQT Software Package

The DarkGAQT [12] software represents the front end of the hardware evolution system. It hosts the evolutionary algorithm, encapsulates the hardware access, manages the test patterns and evaluation criteria, and provides the according user interfaces. Some features desirable for such a software tool are flexibility and fast executions. Flexibility is sought for the design of the evolutionary algorithm and for the description of the experiments. The need for fast execution refers to the evolutionary loop itself. The hardware evolution approach intrinsically relies on a large number of candidate evaluations, which is supported by relative fast testing in hardware. In this vein, the implementation of the EA is not to form the bottleneck for the achieved evaluation rates. The above demands render an implementation in the programming language C++ [ISO98] an almost optimal choice, as it combines object orientation and fast execution. Furthermore, C++ is widely used and provides the necessary low-level function for hardware access. To speed up the evolutionary process, the evolutionary loop of DarkGAQT is partitioned into three separate threads.

In its current state, DarkGAQT is exclusively developed and used under Linux using the kernel version 2.4.x and the GCC 3.3 compiler [gcc]. However, in principle it should be straightforward to migrate the software to a *Windows* operation system and another compiler, because all of the used libraries are also available for *Windows*: First, graphical user interface, string manipulation, and XML[13] [Ray01] handling employ the QT library by Trolltech [Tro]. Second, the multithreading architecture of the evolutionary loop is based on Pthreads [Nic98], which are also available under *Windows*. Finally, the low level hardware access is mediated by the WinDriver 6.x product offered by Jungo, Inc. [Jun03], which was actually developed for *Windows* first. DarkGAQT is a conjoint work with S. Fölling, M. Trefzer and D. Brüderle: The multithreaded genetic algorithm framework and the visualization of candidate circuits was developed by S. Fölling [Föl00]. D. Brüderle implemented most of the building block representation used in chapter 8. M. Trefzer has been mainly liable for organizing, extending, refining and improving the algorithmic framework including for example the XML functionality. The low level hardware access class is derived from the tree of `TAccess` classes that were originally developed by J. Schemmel in [Sch99].

4.4.1 Overview

While the following subsection explains the architecture of the DarkGAQT software package, this section takes a more functional view on the program in that it explains its prevalent features. However, concrete implementation of the evolutionary algorithm as used for the experiments presented in chapters 5 to 8 is deferred to section 4.4.3.

Evolutionary Algorithm. Currently, the implementation of the evolutionary algorithm is based on a generational model that most closely resembles a genetic algorithm. Yet, as the DarkGAQT software allows for the implementation of new representations, it could also host a genetic programming approach. As has been pointed out in chapter 2, evolutionary algorithms are inherently modular, in that their main constituents are mostly independent of each other. The object-oriented nature of

[12] At first, the software was called GAQT, which is a combination of GA and QT, that is genetic algorithm and the QT [Tro] library used for graphical user interfaces. Later on, the prefix 'Dark' was added to indicate the software's customization to the Darkwing board.

[13] Extensible Markup Language

DarkGAQT supports this modularity in that it does not only include different selection methods, fitness functions and circuit representations already, but also provides mechanisms to incorporate new types thereof. DarkGAQT offers the four standard selection schemes presented in chapter 2, that is truncation, rank-based, fitness-proportional and tournament selection.

The main unit of evolution is the `Population`, which essentially holds an array of `Genotype`[14] objects, which encode the genome of the candidate circuits. The standard `Genotype` class is detailed in section 4.4.3. In order to implement different representations and variation operators, new classes `GenotypeXXX` have to be derived from `Genotype`. The derived classes must at least provide the generic methods `crossoverGenotype()`, `mutateGenotype()` and `updateRepresentation()`. The latter method is used to convert the information stored in the genotype object at hand into the generic format of the standard `Genotype` class, which can be processed by other components of the DarkGAQT program, e.g. for visualization of the corresponding FPTA configuration or for the download to the chip. Within this thesis, two genotypes, namely the standard `Genotype` class and the class `GenotypeBBlock`, which is discussed in chapter 8 are used. A third representation – based on the class `GenotypeTurtle` – has been presented in [Tre04].

Representation, selection scheme, and parameters of the evolutionary algorithms are configured through the `GaConfig` class, which reads the according information from an XML file on invocation of the DarkGAQT program. In the context of the DarkGAQT software, the fitness function is perceived rather as a part of the experiment than as a part of the evolutionary algorithm, because its main purpose is to describe the desired target functionality. Accordingly it is handled in the part of the program that describes the experimental setup.

Definition of the Experiment. The experimental setup is managed in the class `Experiment`, which comprises one or several `TestMode` objects. Fig. 4.10 shows a screenshot of the according graphical user interface. The `TestMode` class contains the entire information necessary for the generation of a test sequence executed by the *FPTA test engine* that has been explained in section 4.3.2. In particular, that are the test pattern itself, the IO-cell configuration, possibly a test bench gene possibly a number of random orders or the test pattern. In addition to the information about the actual hardware test, the `TestMode` class also determines how test pattern and acquired data are to be interpreted. First, the table holding the input data also stores the target values with which the candidate circuits are to respond to the input stimuli. Second, the `TestMode` defines the fitness function that is to be used. The fitness function is also liable for necessary conversions of the input data and/or the measured circuit response. Typical examples are the conversion of a binary pattern into an integer representation (employed in chapter 6, or a Fourier transform to convert the measured data into a frequency response, which is utilized in in chapter 7. On one hand, the combined application of the actual fitness criterion and a possibly necessary conversion of in- or output data allows to minimize the computational effort. On the other hand, one may object, that this interwoven implementation hinders efficient reuse, which may eventually lead to decoupling fitness evaluation and data conversion. Finally, the `TestMode` objects define how the output data shall be plotted in the according `ResultPlot` windows.

The test pattern data is organized in a table, in which each test case, that is one contiguous set of input voltages, settling times and target values, corresponds to one row. On one hand, the user interface allows to manipulate single entries or selected regions by hand. On the other hand, the `TestMode` class provides a large number of powerful functions to create particular input patterns, which are also accessible from the user interface. Hence, the implementation of the test modes and the according interface combine convenience with full control and flexibility. However, in some cases

[14] Actually, in the DarkGAQT source code, many classes are organized such that a concrete version dedicated to the current FPTA is derived from a more abstract class. The according class names all contain an 'FPTA'. For simplicity, this distinction as well as the additional 'FPTA' is omitted here.

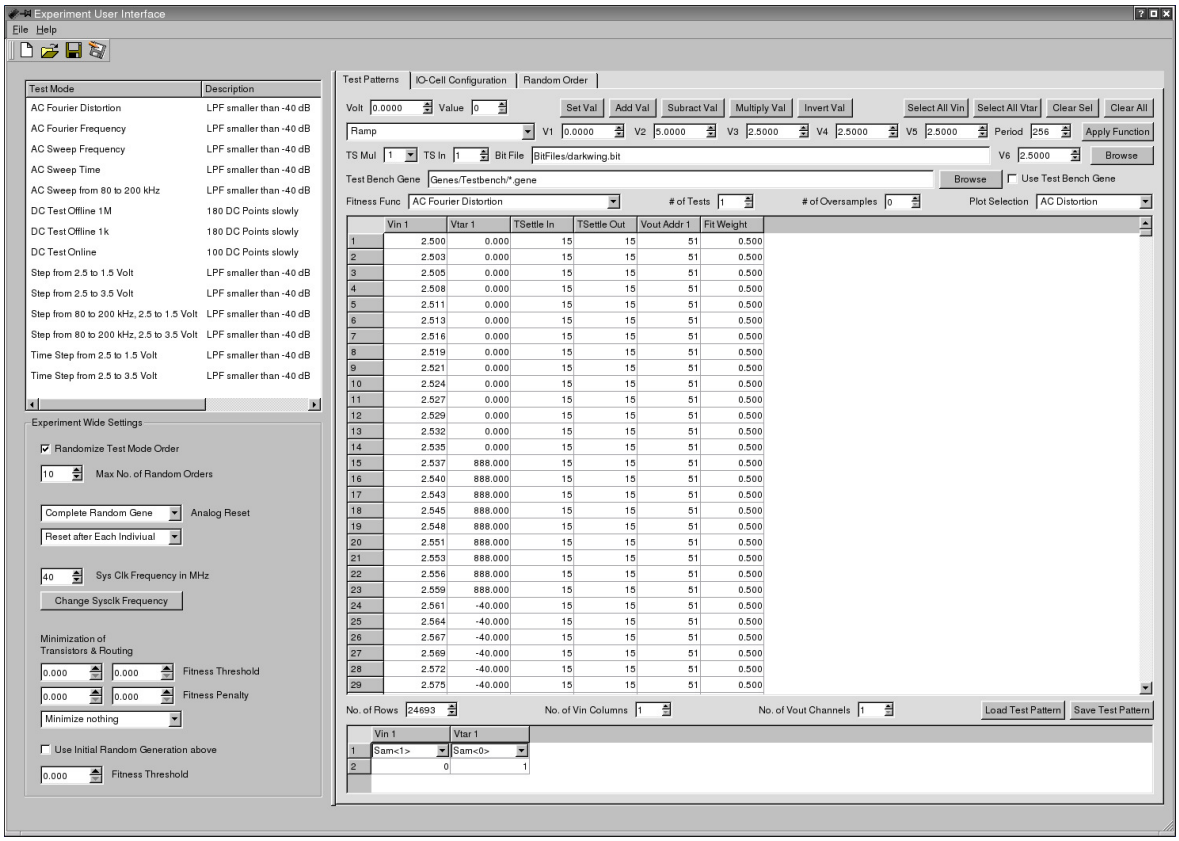


Figure 4.10: Screenshot of the graphical user interface to the `Experiment` class.

further encapsulation into more user-friendly interfaces may be desirable. The fitness functions are implemented as objects that are derived from the base class `FitnessBase`. If the desired function is not already covered by the set of implemented functions, a new class must thus be added to the source code.

The `Experiment` class itself organizes the `TestMode` objects. This includes some support in distributing the test sequences into the local memory. Moreover, the `Experiment` objects hold some general information necessary by all test modes. Viable examples are the system clock frequency f_{sys} or the randomization of the test mode order. Due to the current architecture of the DarkGAQT software, it is not advisable to change the experimental setup during a measurement or the course of the evolutionary algorithm. However, DarkGAQT ensures that the current settings of the `Experiment` object are used whenever the evolution process is (re-) started or a verification test is performed.

Visualization of Measured Output Data. In the design of a hardware evolution experiment targeted at a particular circuit functionality, it is often beneficial to receive some feedback from the evolutionary process. Therefore, the fitness of the best, worst and mean individual of each generation is plotted against the number of generations. Yet, even more interesting, and also more important for the user, is the evolution of the output behavior of the currently best individual: First, the according behavior allows for a quick assessment of the quality of the evolved solutions. Second, it allows the user to discover possible local minima that may be avoided by an appropriate alteration of the fitness criterion. Therefore, the DarkGAQT software provides a family of `ResultPlotTMXXX` classes that

plot the respective circuit response. For each analysis type[15] a specialized class is derived from the base class `ResultPlotTM`. All `ResultPlotTMXXX` objects are gathered in an `ResultPlotXP` object. `ResultPlotTM` itself is derived from a plotting widget of the QWT library.

Verification Tests. One of the key questions arising in hardware evolution is whether the performance of evolved designs is a) reproducible under similar and b) under different conditions. In the context of this thesis, different conditions may be another FPTA chip, another circuit temperature, another time scale or another substrate, as e.g. a simulation. At least the reproducibility under similar conditions and on a second FPTA chip is investigated for all the evolved circuits proposed in chapters 5 to 8. To allow for these investigations, the DarkGAQT software provides an interface that allows to load and test the evolved circuits outside of the evolution loop. The resulting fitness values as well as the measured output behavior can be stored in ASCII files. The latter functionality can utilize a special function of the `ResultPlotTM` class family, since the same arrangement of the measured data is needed in both applications.

Editors. In addition to the `Experiment` window several other editors are provided in the current DarkGAQT version: First, a circuit editor allows to visualize evolved circuits as well as to create new circuits by hand. This functionality is mandatory for all kinds of calibration measurements and system characterizations as well as for the generation of suited test benches. The circuit editor also allows for direct download of the current configuration to the FPTA and also allows to view the configuration currently residing on the FPTA. Second, further editors allow to manage libraries of building blocks and the *genetic access rights*, both of which are introduced and utilized in the experiments presented in chapter 8.

Log Files. The implementation of the evolutionary algorithm allows to write log files that record some of the information generated during the evolution run. Such information may be the best, mean or fitness values, the best individual of every 10^{th} generation or the mean hamming distance between the individuals of each generation. Which information is to be stored can be configured in the `GaConfig` file.

Hardware Abstraction Layer. The hardware access is encapsulated in the class `FPTAConfig`, which is derived from the tree of `TAccess` classes that were originally developed by J. Schemmel in [Sch99]. `FPTAConfig` provides access to both, the Darkwing board and the actual FPTA chip. For instance, `FPTAConfig` converts the standard `Genotype` data into the necessary configuration bitstring for the FPTA, converts the test sequences stored in the `TestMode` objects into the measurement instructions stored in the local SRAM on the Darkwing board, reads the measured output data from the local memory, and calculates the according voltages from the attained ADC data. `FPTAConfig` is also liable for matching the real voltages present on the FPTA chip and their virtual counterparts residing in the DarkGAQT program. To achieve this, `FPTAConfig` receives a configuration object of the type `CardManConfig` that can be edited via the `HWControl` window. For each Darkwing Brightwing combination, this user interface must be used to calibrate the analog voltage levels and their models in the computer. The resulting configuration is usually loaded in the startup phase of DarkGAQT. The information is then transferred to the `FPTAConfig` object, which sets the reference voltage V_{ref} (cf. Fig. 4.4) accordingly and uses the conversion factors provided by the `CardManConfig` object.

[15] Note, that there are more and different analysis types in the context of DarkGAQT that were presented in section 1.4.2. Here, naturally all measurements are bound to be transient analyses. Yet, they can be looked at and evaluated in different ways, e.g. as a frequency response attained from a step function.

4.4.2 Multithreading Architecture

The principal architecture of the DarkGAQT software is illustrated in Fig. 4.11. The user interfaces providing the actual front end are summarized in the box at the top end of the diagram. The low-level interface to the Darkwing board hosting the FPTA chip is indicated by the aforementioned `FPTAConfig` class. Most of the class objects and thus most of the functionality is linked via the central `GaMain` object, which grants access to the actual evolution engine depicted below the `GaMain` object. While the main control (`GaMain`) as well as all user interfaces share a single thread, the evolution engine is distributed over three additional threads. Here, multithreading is used to speed up the evolution process: If the entire application was run in one thread, the CPU could not be used during the actual hardware test, because it had to to wait for the procedure to end, before it gets in control again. Yet, if the part of the program performing the hardware test is executed in a separate thread, the processor is free to carry out other tasks like the evaluation of individuals that are already tested or the creation of new offspring.

The first thread hosts the population manager object `PopMan`. The population manager encapsulates the evolutionary part of the loop. That is, the `PopMan` sorts the old generation according to the attained fitness values and creates a new generation by means of the according variation operators and based on the chosen selection scheme. The new individuals created by the `PopMan` are subsequently passed to the `CardMan` running in another thread using pointers to `Genotype` objects.

The prevalent functionality of the `CardMan` is to organize the hardware test of the new candidate circuits, which entails their download, the application of the test pattern and the gathering of the resulting measurement results. Therefore it possesses an `FPTAConfig` object that provides the low-level interface to the Darkwing board. The measured results are collected in an object of the special container class `Result`. A pointer to these raw results is then passed to the `Evaluator` executed in yet another thread. Here, the primarily task is to calculate the fitness based on the measured data received from the `CardMan`. However, as explained above, this may necessitate a transformation of the raw data, as, for example, done in a Fourier transform.The resulting fitness value is stored in the `Result` object which is finally transmitted to the `PopMan` again. The `PopMan` can now assign the received `Result` to its respective `Genotype` by means of an identification tag that is a member of the `Genotype` class as well as of the `Result` class.

The three thread objects of the `PopMan`, `CardMan` and `Evaluator` classes are members of the `GaMain` class. They are linked by pipelines that are used to pass the object pointers of the respective transport objects from one thread to the next. In particular, these pointers are enqueued on the transmitting end of the pipeline and are taken out of the pipeline on the receiving end. This mechanism ensures that the evolutionary loop itself is inherently thread-safe, since each of the shared data objects can only be accessed by one thread at a time. Yet, the pipelining concept can also be very fast, because the data exchange is restricted to object pointers.

The `PopMan` class offers some means to control the evolutionary loop rendering it as the master thread. For instance, the `PopMan` provides the methods to start and stop the evolutionary loop, where the latter functionality is implemented such, that all individuals of the last generation are gathered prior to stopping the loop. Furthermore, the `PopMan` provides a mechanism to access some data of the current generation, which is needed for the visualization of the fitness curves as well as the plots of the currently best individual. Here, function calls from within the `PopMan` thread as well as requests from the main thread the `GaMain` object is running in can be targeted at the same data. Therefore, thread safety is ensured here via the mutual exclusion principle provided by the Pthreads library [Nic98].

In case the evolution engine is stopped, the `GaMain` object can directly access the three daughter threads constituting the evolutionary loop. First this is used to pass the necessary configuration files to these objects, namely a pointer to the `Experiment` object to both, `Evaluator` and `CardMan`, a

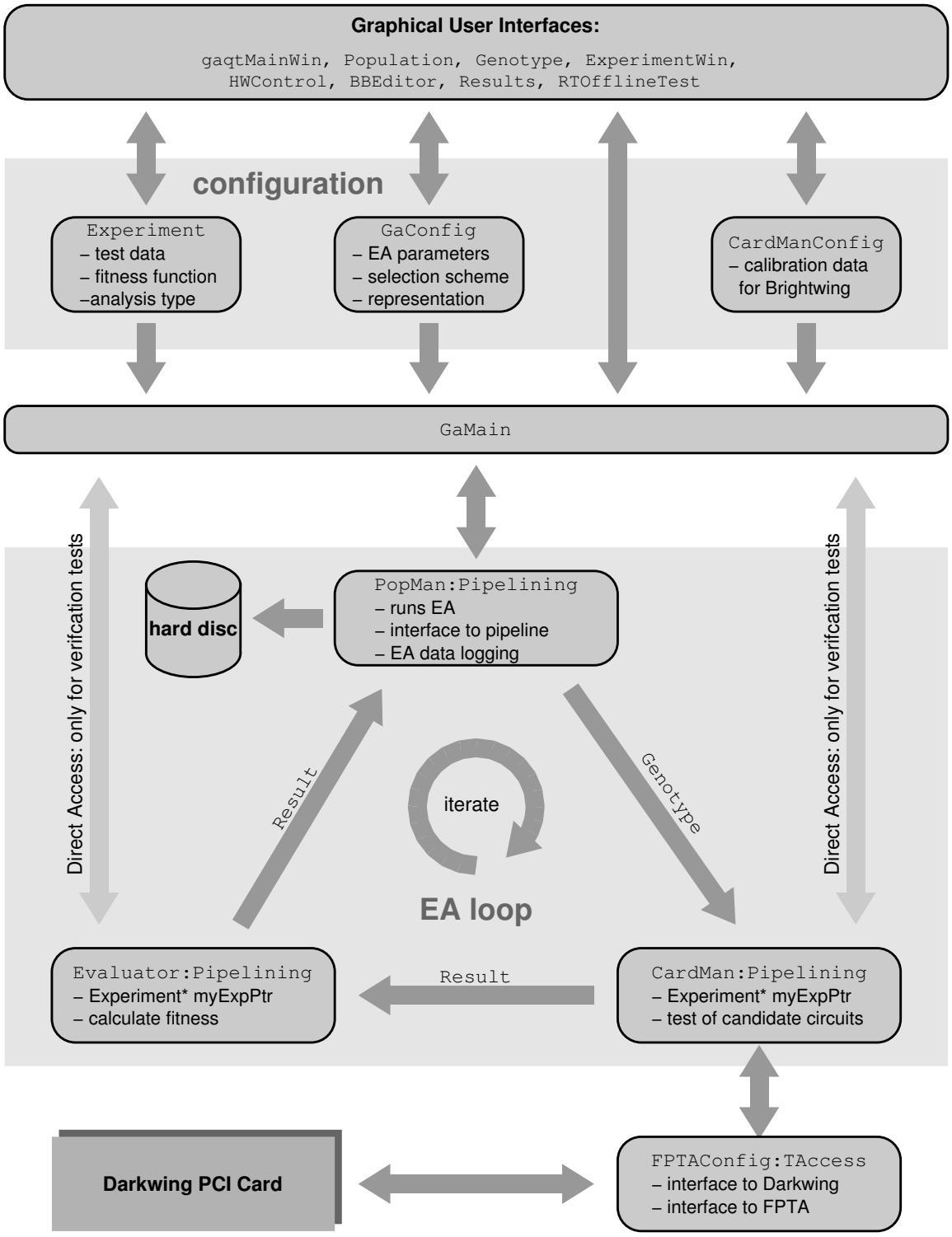


Figure 4.11: Simplified block diagram of the DarkGAQT software. All boxes whose first line is printed in typeface courier represent classes that exist in the actual source code, albeit sometimes with slightly different names. Exceptions are the Darkwing card in the lower left and the box at the top of the figure, which summarizes the main user interfaces found in the actual source code. Please note that the arrows in the evolutionary loop, `Genotype` and `Result` are denoting class objects used to exchange in information between `PopMan`, `CardMan` and `Evaluator`. Most classes contain a short summary of their functionality. In case of the `CardMan` and `Evaluator` classes it is also indicated that they possess a pointer to the `Experiment` class.

Algorithm 4.3: Evolutionary Algorithm implemented in DarkGAQT.

```
for g ← 1 to number of generations do
    Sort population in descending order          // best = lowest fitness value = first
    for i ← 1 to keepPart · populationSize do
        newGeneration[i] ← oldGeneration[i]
    end for
    for i ← keepPart · populationSize + 1 to populationSize do
        Select Genotype1
        Select Genotype2
        newGeneration[i] ← Crossover(Genotype1,Genotype2)
        newGeneration[i]->mutate();
        evaluate(newGeneration[i]);
    end for
end for
```

copy of the `CardManConfig` to the `CardMan` and a copy of the `GaConfig` to the `PopMan`. Second, this direct access allows to use the daughter threads for the verification tests.

4.4.3 Implementation of the Evolutionary Algorithm

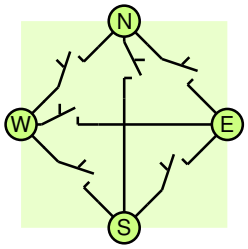
The basic genetic algorithm implemented in the DarkGAQT software is described by Alg. 4.3. At first, the received fitness results are used to sort the population. Note, that unlike in section 2.2.3 and 7.3.1.3 the population is sorted in descending order, that is the fittest individuals featuring the lowest fitness values attain the lowest rank. At first, the best `keepPart · populationSize` individuals are promoted to the next generation, which could be perceived as an extension of the elitism concept. Although greatly increasing the selection pressure, a `keepPart` resulting in a direct transfer of more than the best individual can be beneficial in the context of intrinsic hardware evolution, because it prevents good solutions from being replaced by worse candidate circuits that encounter more favorable conditions once.

Second the rest of the population is filled by means of crossover and mutation. To accomplish this, two genotypes are selected from the old generation by means of the chosen selection scheme. Next, the two individuals undergo a crossover operation with probability $p =$ crossover rate. Note, that the crossover operator generates one offspring only in the DarkGAQT implementation. If it is decided not to have crossover, the individual selected first, that is `Genotype1` is chosen. The resulting gene — independent of the occurrence of a crossover operation — is then mutated with probability $p =$ mutation rate . Finally, the individual is submitted to the `CardMan` thread for evaluation.

If truncation selection is used, the first individual, that is `Genotype1` is randomly chosen from the best `mutatePart · populationSize` individuals, whereas the second individual `Genotype2` is randomly drawn from the best `crossOverPart · populationSize` individuals.

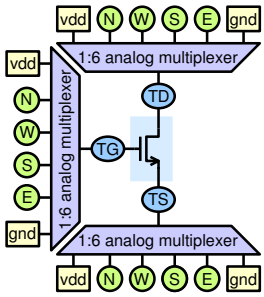
Representation. The standard (or plain) `genotype` used for all experiments presented in chapters 5 to 7 is a straightforward encapsulation of the information necessary to configure the FPTA. The representation of one transistor cell is depicted in Fig. 4.12; it is stored in a `CellGene` object. The switch states of the six routing switches are also encoded as single bits in the `CellGene`. The connections of the three generic transistor gates TD, TG and TS are modeled by enumerations featuring the same eight choices available on the chip itself. That is, both power supply voltages vdd and gnd, the four cardinal points N,W,S,E and two multiplexer codes that leave the respective gate unconnected.

Finally, width and length are also encoded as enumerations that again comprise exactly those values offered by a PTA cell. Yet, in contrast to the terminal connections, the transistor dimensions are ordinal attributes which can be exploited to implement a special mutation operator that applies only small variation of the actual value. Note that a transistor width $W = 0$ means that the programmable transistor is not connected at all. The whole `Genotype` is composed of an array of `CellGene` objects.



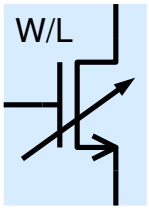
Routing Switches

Representation:
switch sw $\in \{0,1\}^6$
Mutation:
for $i \leftarrow 1$ **to** 3 **do**
flip bit sw[i] with $p = \mu_{\text{Routing}}$;
end for



Gate Multiplexing

Representation:
multiplexer mux $\in \{vdd, gnd, N, S, E, W, NC, NC\}^3$
Mutation:
for $i \leftarrow 1$ **to** 3 **do**
randomly choose new mux[i] with $p = \mu_{\text{TermConnect}}$;
end for



Transistor Dimensions

Representation:
Width $W \in \{0,1,2,\ldots,15\}$, Length $L \in \{0.6,1,2,4,8\}$
Mutation:
randomly choose W with $p = \mu_{\text{W/L}}$;
randomly choose L with $p = \mu_{\text{W/L}}$;

Figure 4.12: Genotypical representation and 'point mutations' of one transistor cell (chromosome). The Mutation rates are denoted as μ.

Mutation Operator. The mutation operator is applied to all attributes of all `CellGene` objects of the `Genotype` with probability $p =$ mutation rate. The particular mutation operations are also illustrated in Fig. 4.12. Essentially, the respective attribute values are randomly chosen from the set of all feasible choices provided by the respective part of the representation.

Crossover Operator. Crossover is applied at the level of transistor cells only, that is only integer transistor cells can be exchanged between crossover partners. The crossover operation is elucidated by Fig. 4.13. In contrast to most of the common crossover operations, the two genotypes are not perceived as linear objects, but rather as two-dimensional entities. Accordingly, the crossover starts

with randomly choosing a rectangle of transistors cells, which are denoted by squares in Fig. 4.13. Subsequently, the transistor cells bounded by the chosen rectangle of the first genotype are replaced by those of the crossover partner to yield the desired offspring. The maximum edge length of the randomly selectable rectangles can be prescribed by the parameter *crossover block size*. The two dimensional crossover is preferred over a linear alternative, as it reflects the intrinsic topological nature of electronic circuits hosted on the FPTA. It is thus expected to be less destructive and to yield better results.

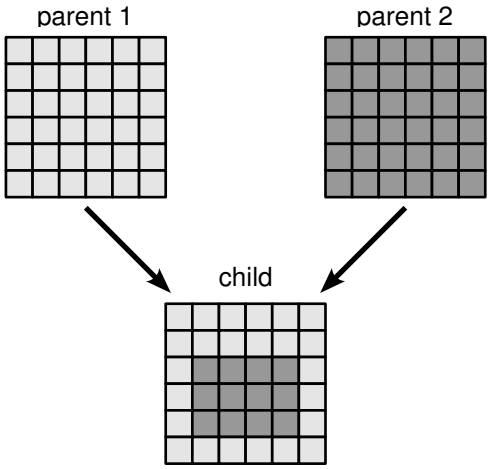


Figure 4.13: Two-dimensional crossover operation used in the DarkGAQT software exemplified for genomes restricted to 6×6 transistor cells.

Genetic Algorithm Parameters. The parameters defining the actual version of the genetic algorithm are summarized in Table 4.2. The particular parameters used for the experiments presented in chapters 5 to 8 will be summarized in similar tables, in which the explanation is replaced by the actual value.

GA Parameter	Meaning
population size	number of individuals in one population (or generation)
number of generations	defines for how many generations the algorithm shall be run
selection scheme	choose truncation, rank based, fitness proportional or tournament selection
reprod. fraction	also referred to as `keepPart`. Defines the number of best individuals deterministically transferred to the next generation
mut. fraction	fraction of the best individuals from which the first individual of a crossover operation is randomly chosen if truncation selection is utilized
mut. rate Term. Con.	mutation rate for mutating the connections of the three generic transistor terminals. Choices are both power supply voltages as well as N, W, S, E
mut. rate W,L	mutation rate for changing the transistor's width or length
mut. rate Routing	mutation rate for flipping a routing bit
crossover fraction	fraction of the best individuals from which a crossover partner is randomly chosen if truncation selection is utilized
crossover rate	probability for performing a crossover operation
, crossover block size	maximum edge length of the rectangles of transistor cells that are exchanged between two crossover partners

Table 4.2: Summary of the genetic algorithm parameters.

Part III

Experiments and Results

Chapter 5

Evolution of Quasi-DC Solutions

Der Worte sind genug gewechselt,
laßt mich nun endlich Daten sehen.

freely adapted from *Faust*
JOHANN WOLFGANG VON
GOETHE

The proposed evolution system is initially tested with experiments aiming at specific target DC behaviors. First, the test of the DC behavior allows for a relatively simple setup and can be easily stated in terms of the fitness function. Second, verifying the DC behavior is both, one of the most important modes of circuit analysis and, strictly speaking, impossible, since every measurement is bound to happen in time. The experiments can be divided into two case studies, the evolution of symmetrical logic gates and that of a Gaussian voltage to voltage transfer characteristic. While a series of experiments illuminates the influence of the number of transistor cells available to the EA, the former experiments reveal that evolution's success varies for different types of logic gates. Moreover, by means of a second series of experiments targeted at finding logic gate circuits, it is shown, that it is both, necessary and sufficient to randomize the test pattern in order to find the desired quasi-DC solutions. Finally, the behavior of all of the evolved circuits measured on two different dice is compared.

5.1 Introduction

5.1.1 Rationale

In order to test and study the evolution system described in chapters 3 and 4, two sorts of experiments have been carried out in which the goal has been to find circuits with a desired DC output characteristic. On one hand the restriction to the DC case simplifies the task as well as the analysis and is thus

considered a good method to test the system. On the other hand, in conventional electronics a stable DC operating point is necessary for most circuits to work properly, Therefore, one has to ensure that stable DC solutions can be found by the evolution engine.

In the literal sense, testing a DC solution would require to wait for an infinite time for the outputs to settle to its dc values for every single test case. Since, of course, this is as infeasible as unnecessary, the circuits under test could either be verified to exhibit the correct output behavior for the maximum time prescribed in the desired specifications or the order of the test data must be randomized. The former approach may be very time consuming precluding its application within the loop of an evolutionary algorithm. Thus, the latter approach is chosen and the evolved circuits discussed in this paper are referred to as *quasi* DC solutions. However, in order to *guarantee* the stability of the evolved DC circuits, an additional long term test would be mandatory. Although this is omitted for the experiments presented in this chapter, the digital-to-analog converters described in chapter 6, which are also evolved with randomized test patterns, are verified to work on a second, larger time scale. Thus, the randomization procedure is considered sufficient. The question whether a randomization of the test patterns is indeed necessary is addressed by a preliminary series of experiments. Here, the behavior of circuits evolved with randomized and non-randomized test patterns obtained from different tests is compared.

The artificial evolution of *quasi* DC solutions is presented on the basis of two test cases, namely the analog behavior of the six logic gates and a Gaussian voltage transfer characteristic (V-V curve). For one, logic gates are key elements to digital as well as mixed signal design; for the other, implementations of the logic gates known to work well exist within the given design space. Accordingly, they can be used as a benchmark for the performance of the evolved circuits. The design of a Gaussian function circuit, on the other hand, is less intuitive and more of an analog nature; thus it may be better suited for the proposed evolution system. Compared to other computational circuits as e.g. a cubic or a root circuit, a Gaussian function seems to be more of a challenge, because it is non-monotonic and possesses two inflection points.

5.1.2 Related Work

5.1.2.1 Logic Gates

Although quite a few hardware evolution experiments address the evolution of logic gates, they usually differ from the work proposed here, in that a transient analysis – which covers only one or few particular timings/transitions – is used to evaluate the candidate circuits. Table 5.1 summarizes some experiments dedicated to the artificial evolution of logic gates.

While Santini et al. [San01] use their PAMA device for the evolution of an XOR gate, the group of Adrian Stoica either uses one of their three FPTAs, the according software model or do not constrain the evolution process to any hardware at all. The experiments performed by Stoica et al. cited in Table 5.1 are not restricted to the evolution of the logic gates itself, but emphasize different aspects of the evolution process and or qualities inherent to the evolved circuits. For instance, in [Key00b] evolution is used to regain the functionality of an evolved XNOR gate after the application of different faults, [Sto01a] investigates whether Hardware evolution can help to find circuits working at extreme temperatures and [Sto02a] deals with NAND gates at low power supply voltages. On the other hand, the work documented in [Sto01b] and [Sto02b] addresses the reliability and portability of evolved circuits: In the former publication the candidate solutions are tested in Software as well as in Hardware – a technique the authors refer to as *mixtrinsic* Hardware evolution, because it comprises intrinsic and extrinsic tests. In the latter contribution the fitness is calculated from two transient analyses at different time scales. Zebulum et al. ([Zeb98c]) as well as Bennett et al. ([Ben99]) evolve logic gates from bipolar transistors and resistors in an unconstrained manner using SPICE simulations, albeit with

Group	Gate	Used Devices	**HW!** Mode	No. of Evaluations	Reference
Santini et al.	XOR	BJT, R	intrinsic	60.4 k	[San01]
Stoica et al.	XNOR	CMOS	intrinsic	12 k	[Key00b]
	AND	CMOS	mixtrinsic	900	[Sto01b]
	AND	CMOS	intrinsic	10 k	[Sto01a]
			unconstrained		
	NAND	CMOS	unconstrained	?	[Sto02a]
	NAND	CMOS	unconstrained	?	[Sto02b]
Zebulum et al.	AND, OR, XOR	BJT, R	unconstrained	2.5 k	[Zeb98c]
Bennett et al.	NAND	BJT, R	unconstrained	< 2.224 M	[Ben99]
Shibata	all 2-input gates	CMOS	unconstrained	1 M	[Shi01]

Table 5.1: Related work concerning the artificial evolution of logic gates.

different circuit representations: The former approach uses genetic algorithms, the latter one genetic programming. Finally, Shibata ([Shi01] evolves all logic gates featuring two inputs and one output within one evolution run: A special technique is used to decompose a total of four human made logic gates (NAND, NOT, NOR, XOR) provided to the design engine and re-synthesize the resulting components to form all of the desired gates. In general, the extrinsic approaches yield better circuits than their intrinsic counterparts. In fact, probably none of the intrinsically evolved logic gates cited here, would have been denoted as perfectly meeting the fitness criterion that will be set forth in the next section, even if only those test cases, for which results are documented in the respective publications would be used for this comparison. On the other hand, most of the extrinsically evolved gates referenced here, probably would match the part of our fitness criterion that is covered by the reported test data.

5.1.2.2 Gaussian Function Circuit

In analog circuit design Gaussian function circuits received attention as one of several so-called membership functions used for the fuzzyfication of crisp input signals in fuzzy logic controllers. Hence, they are primarily found in analog implementations of such fuzzy logic controllers: For instance, Lin et al. ([Lin98]) present a circuit that, if operated in weak inversion, exhibits a Gaussian I-V output characteristic, which is adjustable in amplitude, width and offset. Other examples comprise the work of Kettner et al. ([Ket93]), who presents a piecewise approximation of a Gaussian I-I curve and that of Shuwei et al. ([Shu96]), who reports a universal membership function circuit that – among others – can produce a Gaussian-like (triangular) I-I curve.

In the realm of hardware evolution, the problem of designing a Gaussian function circuit in CMOS technology has been approached by the groups of Adrian Stoica and John Koza as well as by Kevin Sheehan. While Koza et al. use genetic programming to evolve a Gaussian I-V characteristic in an unconstrained extrinsic approach reported in [Koz99f], the problem is considered for proof of concept experiments as well as for benchmarking tests in a bunch of publications by Stoica et al.. An extrinsic approach using a SPICE model of the group's field programmable transistor array (FPTA-0) chip is documented in [Sto99] (I-V characteristic) and an intrinsic one utilizing the actual chip itself is reported in [Sto00a] (V-V characteristic). Finally, [Zeb02] reports an unconstrained hardware evolution experiment targeted at a Gaussian I-V characteristic that is turned into a V-V characteristic by means of a 100 kΩ resistor. To the author's knowledge, only this last paper contains a quantitative analysis

of the resulting Gaussian IO characteristic. In fact, this latter IO-characteristic appears to resemble the target curves more closely than the characteristic resulting from the aforementioned intrinsic approach. In [She02] Sheehan reports the ex- and intrinsic evolution of Gaussian V-V characteristics by means of predefined circuits consisting of three amplifiers and additional two-terminal elements the author refers to as *generalized impedances* comprising resistors and diodes. It is the values and types of these generalized impedances that can be chosen for predefined locations in the proposed circuit template to generate the desired circuit behavior. As a tribute to the limited range of generalized impedance values, Sheehan[1], presumably chooses a fitness criterion that does not prescribe the scale and offset of the output voltage, which further facilitates the design task. The best intrinsically evolved circuits perform significantly worse than the best extrinsically evolved one, which itself is e.g. flawed by discontinuities in the output characteristic close to zero as well as an almost linear slope.

Although the referenced Gaussian function circuits differ in the physical in- and output quantities as well as in the parameters of the according output curves, the approaches for which a relative error is either reported or could be inferred [2] are compared in Table 5.2. Please note that the circuits

Group	Type of Characteristic	EHW Mode	No. of Evaluations	Mean Error [%]	Reference
Kettner et al.	I-I	—	—	≈ 2	[Ket93]
Stoica et al.	I-V (V-V)	unconstrained	1600	1.86	[Zeb02]
Koza et al.	I-V	unconstrained	23.04 M	≤ 1.2	[Koz99f]

Table 5.2: Related work: Design/Evolution of Gaussian function circuits.

produced by artificial evolution are not as thoroughly tested nor as well understood as can be assumed for hand-designed ones that are actually produced. Thus, it is conceivable that they are inferior to the circuit proposed in [Ket93] in terms of settling time, long term stability, portability to other process technologies or temperature variations. The relative mean error is calculated from the mean error per data point divided by the full scale of circuit's output. In case of the circuit proposed in [Ket93], the error is estimated from a plot of the deviation from the target function versus the input current. The mean error for the circuit evolved by Koza et al. is calculated from the circuit's fitness value by means of the reported fitness function, assuming that none of the errors exceeds a value of 5% and therefore represents an upper threshold.

5.2 Experimental Setup

5.2.1 Problem Definition

As mentioned above, the task is to find (quasi-)DC solutions for two classes of problems, namely for the six symmetric two-input gates and a Gaussian function circuit. More specifically, the task is to find circuits, whose outputs settle – after a finite period of time – at a target voltage that is completely defined by the set of input voltages (V-V characteristic). Thus, the problem can be described in terms of this particular target output characteristic, the test pattern used to sample the circuit's behavior and

[1] This actually holds for all of the abovementioned evolutionary approaches.

[2] Although Sheehan does specify different error values for the set of experiments presented in [She02], the exact definition of the underlying error measure can not be found within the entirety of 184 pages constituting his thesis, thus rendering the values useless for a comparison.

the time allowed for the circuit to settle. The fitness criterion maps the measured behavior of the circuit onto a single quality scale allowing to compare the performance of different circuits.

5.2.1.1 Target Function

Logic Gates. The general form of the target function for the experiments aimed at the artificial evolution of logic gates is a function of two input voltages V_{in1} and V_{in2}:

$$V_{\text{tar}} = V_{\text{tar}}(V_{\text{in1}}, V_{\text{in2}})\,. \tag{5.1}$$

The actual target voltage depends on the respective gate that shall be evolved. Table 5.3 contains the truth table of the six symmetric two-input gates considered here. The analog DC behavior of

Input A	Input B	NOR	NAND	AND	OR	XOR	XNOR
0	0	1	1	0	0	0	1
0	1	0	1	0	1	1	0
1	0	0	1	0	1	1	0
1	1	0	0	1	1	0	1

Table 5.3: Truth table for the 6 symmetric logic gates the evolution experiments are aimed at.

logic gates is typically defined by the two thresholds V_{IL} and V_{IH} specified in Table 5.4. For input voltages below V_{IL}, the input must be treated as a logic zero, for those above V_{IH} as a logic one. The exact behavior for input voltages in the transition region between these two thresholds is irrelevant for digital circuits; in fact it would constrain the designer (or artificial evolution in this case) more severely than necessary. The outputs of the evolved circuits on the other hand are required to identify the two possible logic values with the power supply rails, as summarized in Table 5.4: While a logic 0 translates to a target voltage of 0 V, the output is required to reach 5 V in case of a logic 1.

Input low: V_{IL}	Input high: V_{IH}	Output low: V_{OL}	Output high: V_{OH}
$V_{\text{in1,2}} \leq 2\,\text{V}$	$V_{\text{in1,2}} \geq 3\,\text{V}$	$V_{\text{out}} = 0\,\text{V}$	$V_{\text{out}} = 5\,\text{V}$

Table 5.4: Definition of the analog (threshold) voltages associated witht the logic in- and output values high and low.

In order to illustrate the definitions above, the target behavior of a NOR gate is depicted on the left hand side of Fig. 5.1. The target function is sampled at the input voltages defined by the test pattern described in section 5.2.1.3.

Gaussian Function Circuit. The target output voltage for the evolution of Gaussian function circuits is completely defined by one input voltage V_{in}:

$$V_{\text{tar}} = 1\,\text{V} + 3\,\text{V} \cdot \exp\left(\frac{V_{\text{in}} - 2.5\,\text{V}}{1\,\text{V}}\right)^2 \quad . \tag{5.2}$$

The graph of this function is plotted on the right hand side of Fig. 5.1. Please note, that the chosen target function is considerably wider than that used e.g. by [Zeb02] and [Koz99f]. Furthermore it differs from the target functions defined in the publications cited above in the offset of 1 V.

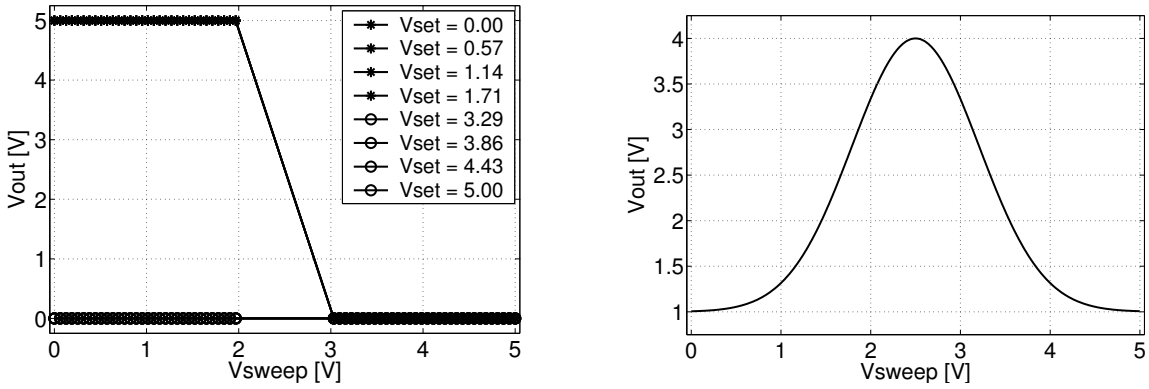


Figure 5.1: Left: Illustration of the output characteristic of an ideal NOR gate for the test pattern depicted in Fig. 5.2. **Right**: Visualization of the Gaussian target function defined by (5.2).

5.2.1.2 Fitness Function

The fitness function used throughout all experiments during the process of evolution is simply the sum of the squared errors,

$$\mathrm{SSE} = \sum_{i=1}^{512} (V_{\mathrm{tar}}(i) - V_{\mathrm{out}}(i))^2 \quad , \tag{5.3}$$

where a total of 512 input test points is used to evaluate the behavior of the candidate circuits throughout all experiments. In order to assign a physical meaning to the fitness measure, the fitness values obtained from (5.3) are converted to the root mean square error per data point in mV by:

$$F = \mathrm{RMSE} = \sqrt{\frac{\mathrm{SSE}}{512}} \cdot 1000 \quad . \tag{5.4}$$

5.2.1.3 Test Patterns

Logic Gates. Instead of testing the candidate circuits on a fine, equidistant 2-dimensional grid of input voltages, one input is held at either one of eight different voltages V_{set} while the other input is swept by means of 64 V_{settle} voltages. This allows to sample the circuit's output with a high resolution using only a moderate number of test points. Taking furthermore into account that the input region between 2 and 3 V needs not be sampled (cf. 5.2.1.1) results in the following test pattern: For each of 8 different set voltages V_{set}, evenly spaced between 0 and 1.714 V, and 3.286 and 5 V 64 sweep voltages for V_{sweep} are taken from 64 voltages evenly spaced between 0 and 1.968 V and between 3.032 V and 5 V, either chosen randomly or applied in ascending or descending order. The test pattern is shown on the right hand side of Fig. 5.2. However, for the illustration of the characteristic output curves of the evolved circuits 11 evenly spaced V_{set} and 250 $Vesettle$ voltages are used including the transition region between 2 and 3 V. This is depicted on the left hand side of Fig. 5.2.

Gaussian Function Circuits. The input test pattern for the Gaussian Function Circuits consists of 512 input voltages V_{in} that are randomly drawn from 512 different equally probable input voltages that are evenly spaced in the interval from 0 to 5 V (equaling the power supply range). In contrast to all other experiments in the remainder of this thesis, the test patterns used for the experiments of this chapter are generated for each candidate circuit individually and may contain the same input voltage more than once.

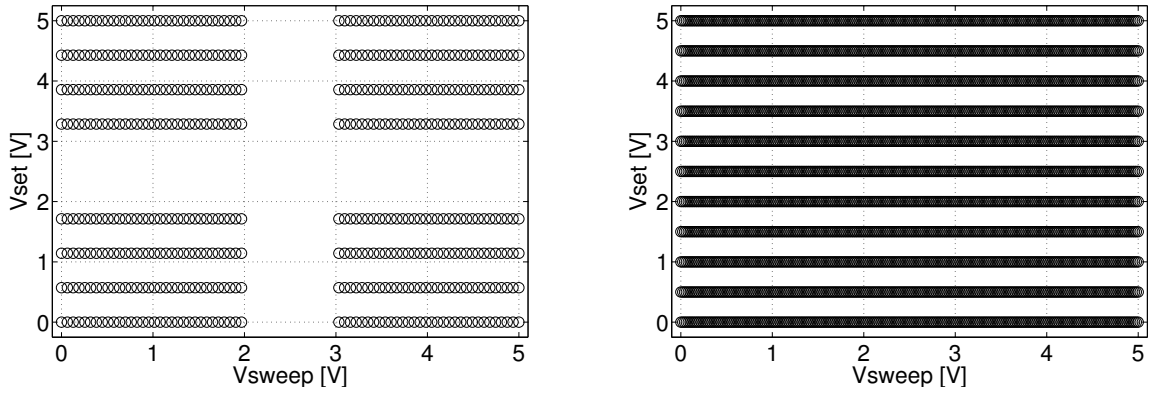


Figure 5.2: Test voltage pairs for the evolution of logic gates as used during evolution (left) and for the verification tests (right).

Timing. Throughout all experiments of this chapter, each input voltage is set by the code running on the host PC[3]. In contrast to the synchronous test pattern generation proposed in section 4.3.2, the exact timing of the voltage change can not be controlled. Moreover, the operating system may interrupt the measuring process for fairly large intervals. On one hand, this precludes strict timing constraints, on the other hand it does raise some selection pressure towards solutions working on different time scales. Yet, from the average test rate of 118 individuals per second one can infer that the output of the candidate circuits must settle at least within 16.6µs. In fact, the required settling time will be somewhat smaller (around 10µs) since the average test rate includes the time necessary to download the circuits under test as well as the time required by the software, i.e. the GA itself, display and storage of the results and the operating system.

5.2.2 Geometrical Setup

All experiments presented in this chapter are restricted to a fraction of the FPTA, namely a square of transistor cells located in the lower right corner of the chip. The size of this square of transistor cells represents a tradeoff: On one hand, a larger number of transistor cells increases the number of possible successful circuits and/or circuit implementations. On the other hand, a larger square of transistor cells available to the GA also increases the search space, which may prolong the time necessary for the algorithm to converge to a good solution.

Throughout all of the logic gates experiments, the active area of the FPTA is restricted to 5×5 transistor cells, as can be seen from the left hand side of Fig. 5.3, which also depicts the location of the in- and outputs. The geometrical setup for the Gaussian function circuits is illustrated on the right hand side of Fig. 5.3. Here, the edge length of the array available to the GA is varied from 4 to 11 in order to investigate the influence of the number of transistor cells on the results produced by the artificial evolution process. Despite the variation in array size the location of the in- and output is kept fix for all experiments.

5.2.3 Overview of the Experiments

The artificial evolution of quasi-DC solutions has been targeted at two different types of electronic circuits, namely symmetric two-input logic gates and Gaussian function circuits. The experiments focusing on the evolution of logic gates are divided into two case studies:

[3]Personal Computer

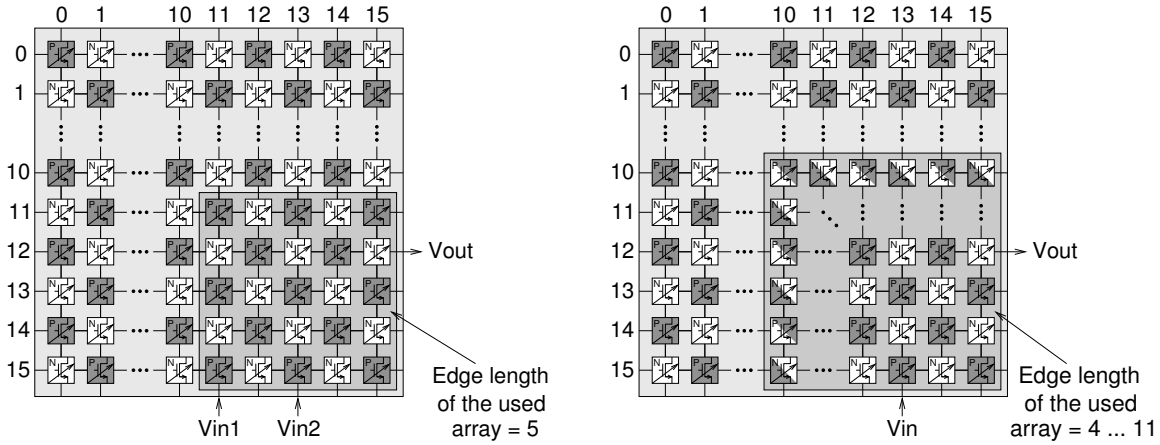


Figure 5.3: Geometrical setup used for the Evolution of: **Left:** Logic gates and **Right:** V-V Gaussian circuits.

5.2.3.1 Evolution of Logic Gates I

The main idea of this first case study is to investigate the influence of different degrees of test pattern randomization on the according evolution results. Therefore, three experiments each featuring 30 runs were carried out for the symmetric two-input gates NAND, NOR, AND, OR and XOR. The test paradigms for the three different experiments of case study I are summarized in Table 5.5: During

Experiment number	V_{in1}	V_{in2}
1	V_{set}	V_{sweep} : forward sweep
2	V_{set}	V_{sweep} : random values
3	V_{set} / V_{sweep} : random values	V_{sweep} / V_{set} : random values

Table 5.5: Test paradigms for the fitness evaluation for the evolution of the logic gates in case study I.

experiment 1, the 8 set voltages V_{set} are always applied to input V_{in1} – in ascending order – and V_{sweep} – applied to input V_{in2} – is swept through all of its possible 64 different values in ascending order for each V_{set}. Experiment 2 differs from experiment 1, in that here V_{sweep} is randomly chosen from the 64 possible values for 64 times. Finally, during experiment 3 the mapping of V_{set} and V_{settle} to V_{in1} and V_{in2} is determined randomly before each individual is tested.

5.2.3.2 Evolution of Logic Gates II

The second case study utilizes the setup of experiment 3 of case study I to increase the statistical importance of the data by performing a total of 100 runs per logic gate. Moreover, case study II also includes data for the XNOR gate, which completes the list of symmetric two-input gates.

5.2.3.3 Evolution of DC V-V Gaussian Circuits

For the evolution of Gaussian function circuits 8 experiments have been carried out. The experiments differ in the number of transistor cells available to the GA. Thereby, the edge length of the square of cells is varied from 4 to 11 (cf. Fig. 5.3). For each edge length a total of 10 runs has been carried out.

5.2.4 GA Parameters

A plain genetic algorithm employing truncation selection was used for all of the experiments presented in this chapter. The according parameters are gathered in Table 5.6. The two sets of parameters

GA Parameter	Logic Gates	Gaussian curve
population size	50	50
reproduction fraction	0.2	0.2
mutation fraction	0.4	0.4
crossover fraction	0.6	0.6
crossover rate	40%	40%
mutation rate	3%	3%
number of generations	5000	10000
crossover block size	3	3...6
edge length of used array	5	4...11

Table 5.6: Genetic algorithm parameters used throughout the experiments presented in this chapter.

for the logic gates and the Gaussian function circuit experiments are identical, except for the maximum edge length of the rectangles exchanged between two crossover partners and the number of generations per run that serves as the stop criterion. The maximum crossover block size is calculated to be the rounded up half of the edge length of the used array. A fairly high reproduction fraction is used to prevent the algorithm from loosing the currently best solutions due to the randomized nature of the test pattern or to noise inherent to the measurement.

5.2.5 Verification Tests

In order to qualify the evolved circuits, their fitness – calculated by (5.4) – can be established in different ways: First, the RMS[4] error obtained from the evaluation of the last generation of the evolution run is taken into account. Second, the best individual of each run is subsequently tested 100 times outside of the evolution loop and the according mean, minimum or maximum RMS errors may be used as the fitness criterion. These values will be referred to as the *last, mean, best* or *worst* fitness value throughout the remainder of this chapter.

Case study I is designed to investigate the influence of spatio-temporal order of the applied test patterns on the evolutionary success. This is achieved by applying a total of 13 different test methods to characterize the performance of the evolved gates; Table 5.7 gives an overview. While test mode 1 simply uses the fitness value achieved by the best individual of the last generation, methods 2 to 13 employ the mean of 100 verification tests. Methods 2,3 and 4 differ merely in the order in which the V_{sweep} values are applied, i.e. whether they are applied in ascending, descending or random order. Test methods 5 to 7 correspond to methods 2 to 4 except for the fact that the allocation of V_{set} and V_{sweep} to V_{in1} and V_{in2} is swapped. Finally, the test methods 2 to 7 are applied to evaluate the performance on a second die, which constitutes methods 8 to 13.

5.3 Results: Evolution of Logic Gates I

Since this first case study is merely used as a pre-study, the acquired data is presented in a highly aggregated manner. A more detailed analysis is deferred to section 5.4 describing the results of case

[4]Root Mean Square

Number	V_{in1}		V_{in2}		Chip
1	lowest error from last generation				
2	V_{set}		V_{sweep}	forward sweep	
3				backward sweep	
4				random values	1
5	V_{sweep}	forward sweep	V_{set}		
6		backward sweep			
7		random values			
8	V_{set}		V_{sweep}	forward sweep	
9				backward sweep	
10				random values	2
11	V_{sweep}	forward sweep	V_{set}		
12		backward sweep			
13		random values			

Table 5.7: Thirteen different test methods employed for the characterization of the evolved logic gates.

study II. The results for the three different experimental setups of case study I (cf. 5.5) are summarized in Fig. 5.4 for the five logic Gates NAND, NOR, AND, OR and XOR: The best-of-run solutions of all 30 runs of each experiment are evaluated for all 13 test methods given in Table 5.7. For each gate the mean RMS error averaged over all runs of one experiment is plotted against the test method.

Regarding Fig. 5.4, the following features are evident for the results obtained from all five logic gates: On average, the circuits evolved within experiment 1 yield significantly better results when tested with method 1,2 or 8 than for any other test method. The curves belonging to experiment three on the other hand exhibit the least amount of variation: The mean of the measured RMS errors is comparable for all test methods with the small exception that the results of the first test mode are always slightly better than those for the remaining 12. Moreover, averaged over all test methods but 1,2 and 8, the results of experiment 3 outperform those of the first two experiments for all gates but the AND gate, where they are approximately equal to those of experiment 1. Apart from the XOR gates, the circuits resulting from experiment 2 perform worse on average for tests $5-7$ and $11-13$ than for tests $2-4$ and $8-10$, that is for those tests with swapped inputs.

From the above observations the following can be concluded: First, the GA tends to exploit a fixed order in the input pattern, such that a reversed or randomized order will lead to significantly worse results. This undesired behavior can be remedied by randomizing the input test pattern, as can be seen from the small variation of the curves corresponding to the results of experiment 3. Second, the data recorded from experiment 2 indicates that a fixed spatial allocation of input signals may also be abused by the process of artificial evolution. Again, at least for *symmetric* gates this can be prevented by randomizing the mapping of the input signals. Since the evolved gates presented within this case study are bound to work under all of the tested conditions, it makes sense to compare the worst mean RMS errors obtained for the three different experimental settings. In this regard, the results of experiment 3 surpass those of the remaining two experiments, and thus must be considered the method of choice.

Finally, a comparison of the RMS errors measured on the two different dice may inspire two insights: First, the fact that the results obtained on both chips are so similar deems those results reproducible. Second, the evolved circuits can be said to perform almost equally well on both chips. A closer look at the results for test method 1 reveals that in most cases the according RMS error is somewhat smaller than for all other test methods. This could be due to a population that, at the

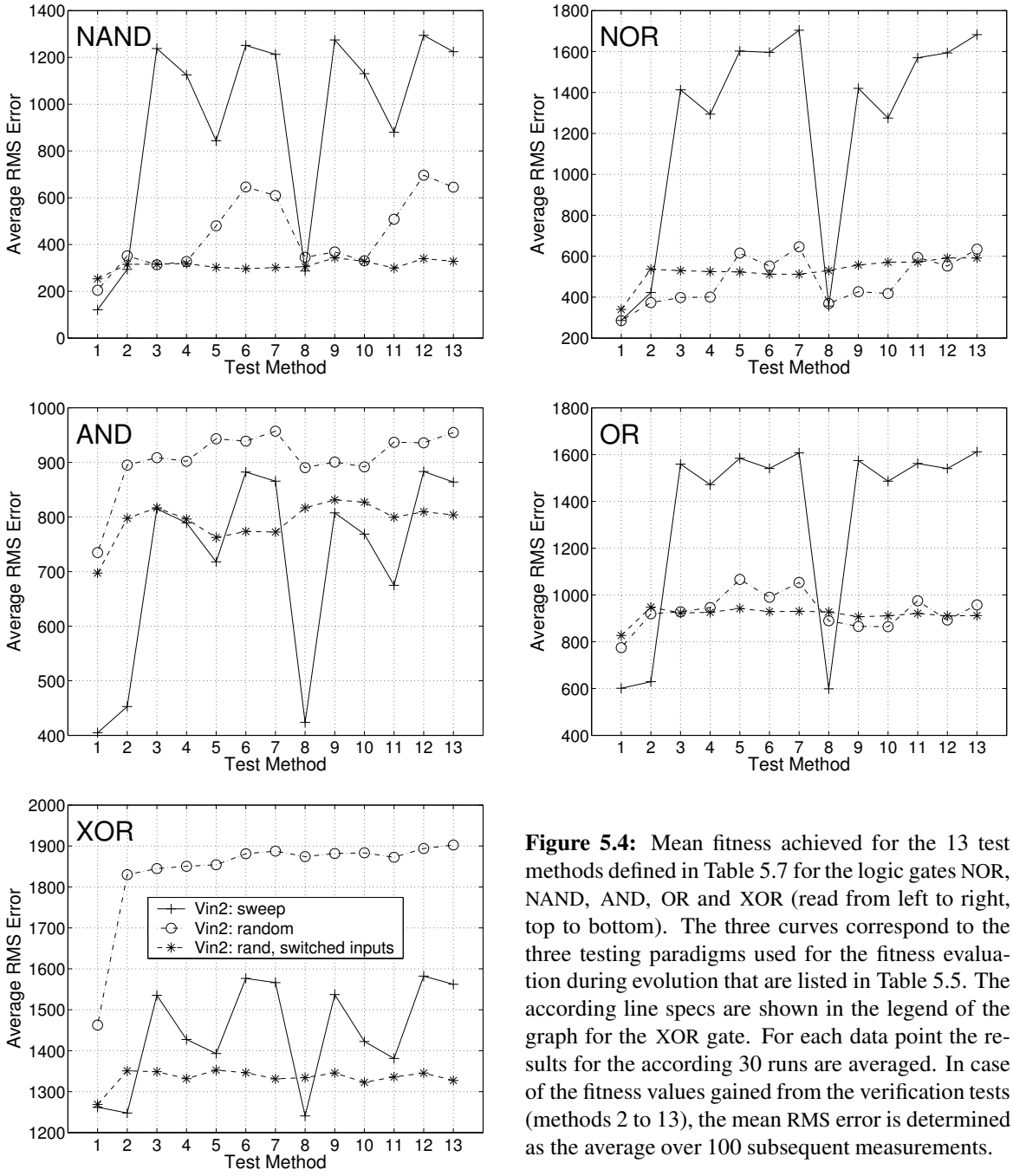


Figure 5.4: Mean fitness achieved for the 13 test methods defined in Table 5.7 for the logic gates NOR, NAND, AND, OR and XOR (read from left to right, top to bottom). The three curves correspond to the three testing paradigms used for the fitness evaluation during evolution that are listed in Table 5.5. The according line specs are shown in the legend of the graph for the XOR gate. For each data point the results for the according 30 runs are averaged. In case of the fitness values gained from the verification tests (methods 2 to 13), the mean RMS error is determined as the average over 100 subsequent measurements.

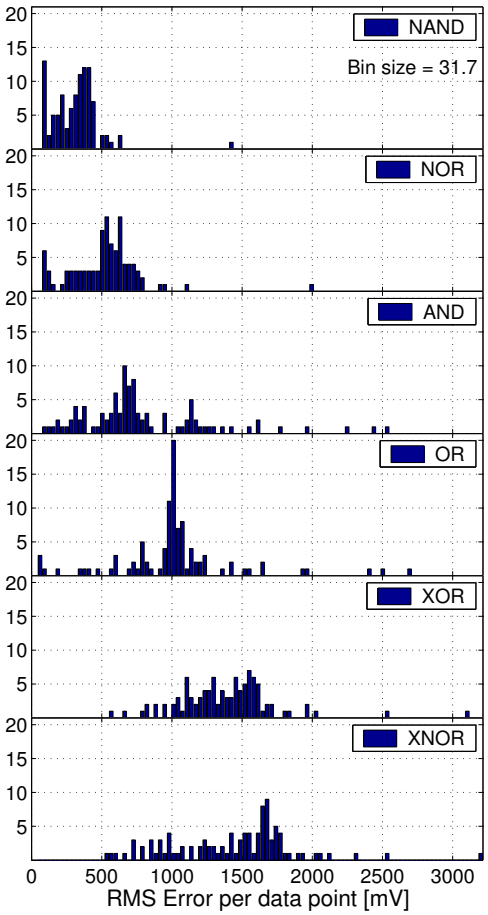
end of the evolution run, has converged to small variations of one genotype, which complies with the large fraction of individuals that are promoted to the next generation without change as well as the truncation selection scheme and the according portions allowed to be selected for crossover or mutation. Thus, although an individual is measured to be best, it may lack an effective genetical advantage, yet may have encountered the conditions that suited it best in terms of noise or test pattern order and timing.

5.4 Results: Evolution of Logic Gates II

To further study the evolution of all six logic two-input symmetric gates, the experimental setup of experiment 3 of the first case study described in Table 5.5 is used. A total of 100 runs was performed for each of the six gate types. With each run taking about 30 min to finish, the experiments kept running for an effective time of approximately 12.5 days.

5.4.1 Overview over the Results of all Runs

At first, an overview of the results shall be given by means of the fitness histograms shown on the left hand side of Fig. 5.5. For each of the six gates the frequency with which the best-of-run solution achieved a fitness value falling in one of 100 bins is plotted, where fitness refers to the RMS error defined in (5.4). This RMS error is calculated as the maximum over the 13 test methods described in Table 5.7. The bin size, imprinted in the histogram for the evolved NAND gates, amounts to 31.7 mV and is used for all of the six graphs.



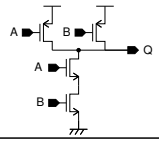
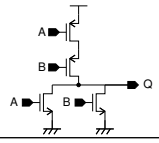
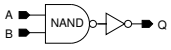
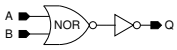
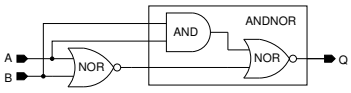
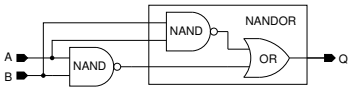
Gate	Implementation	Transistor Count
NAND		4
NOR		4
AND		6
OR		6
XOR		10
XNOR		10

Figure 5.5: Left: RMS deviation from the ideal response for 100 evolutions of the DC behavior of the six logic gates. The maximum error calculated from the 13 test methods defined in section 5.2.5 is taken as the RMS error. The same x-axis is used for all six histograms. **Right:** Typical CMOS implementations of the 6 logic gates NAND, NOR, AND, OR, XOR and XNOR.

The results suggest that the evolution experiments were quite successful in finding solutions for the NOR and the NAND gate behavior, but had more difficulties in finding good solutions for the AND

and OR problems. While for these four gates solutions with an error value less than 100 mV were found by the GA, this did not happen for the XOR and XNOR problems, where the lowest error was observed to be above 500 mV. Standard text book implementations of the six logic gates considered here are shown on the right hand side of Fig. 5.5. The complexity of these text book solutions increases from NOR and NAND to AND and OR to XOR and XNOR, indicated for example by the number of necessary transistors they are composed of. A comparison between this complexity and the evolution results displayed on the left hand side of Fig. 5.5 suggests that the difficulty in evolving the according gate functionality corresponds to the complexity of their typcial CMOS implementations.

As can be seen from the left hand side of Fig. 5.5, even the best of the evolved circuits exceed a minimum RMS error of 50 to 80 mV depending on the gate type. This must not necessarily be due – at least to full extend – to an imperfect behavior of the tested circuits: The readout of the circuit's output voltages described in section 4.3.1 was not perfectly calibrated neither during the evolution experiments nor at the time of the verification tests. In fact, the data acquired during the verification tests indicates an offset of 70 mV for the ground potential. Considering that depending on the gate type 25, 50 or 75% of the test data points ought to produce an output voltage of 0 V, this offset causes a minimum RMS error between 18 and 53 mV. Another error source inherent to the system is the limited precision of the measurement, which is caused e.g. by noise, distortion or the finite resolution of the ADC. Yet, this contribution is estimated to be in the order of a few mV and thereby considerably smaller.

The plot shown in Fig. 5.6 tries to estimate the number of evolution runs that produced *perfect* solutions to the posed problems. A circuit is considered to be *perfect* if the maximum RMS error obtained from the 13 test methods of section 5.2.5 is smaller than 100 mV. Because of the imperfections

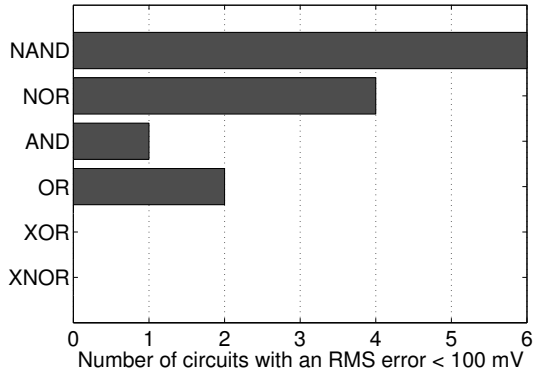


Figure 5.6: Number of evolved circuits that achieved an overall error smaller than 140 mV.

inherent to the measurement process discussed above, even the most ideal circuit will exhibit a finite fitness, such that the *perfect* ones must be determined by means of a threshold. The particular choice for the threshold is somewhat arbitrary. Therefore it was chosen such that the output characteristics of the circuits with a fitness below the threshold are verified to be close to the desired behavior. Since, from a designer's point of view, deviations from the target behavior may be crucial for input voltage pairs close to the power supply rails and negligible for those occurring in the vicinity of the transition region, the yield illustrated in Fig. 5.6 may underestimate the number of *perfect* solutions found.

The number of evolved gates that reveal a *perfect* output characteristic is rather small. As was already observed from the histograms on the left hand side of Fig. 5.5 the success rate decreases with the complexity of the text book solution for the respective target gate: It is in the order of 5% for the NAND and NOR problem, amounts to 1-2% for the AND and OR gates and to 0% for the most difficult of the symmetric gates, XOR and XNOR.

5.4.2 Output Characteristic of the Best Evolved Gates

In order to gain insight into the quality of the circuits rendered *perfect* in the last section, the output characteristic of the *best* solutions for the six logic gate problems are depicted on the left hand side of Fig. 5.7 (NAND, NOR, AND and OR) and Fig. 5.8 (XOR and XNOR). Here, *best* refers to those best-of-experiment solutions, which achieve the smallest value for the maximum RMS error obtained from the 13 test methods defined in section 5.2.5. The according fitness scores are stated in the title of each plot. The output characteristics are based on data acquired using the test pattern of 11 V_{set} voltages described in section 5.2.1.3 in conjunction with test method 2 from Table 5.7. While the measured output characteristics are depicted in the left column of Fig. 5.7 and Fig. 5.8, the right column of Fig. 5.7 and Fig. 5.8 illustrate the simulated output characteristic of the respective text book solution. The text book solutions, which are taken from the standard cell library of the CMOS fabrication process used for the FPTA chip [Aus97a], possess the same topology as those shown on the right hand side of Fig. 5.5; they are simulated for typical mean conditions at $T = 27\,^{\circ}\text{C}$.

While the output behaviors of the best evolved NAND, NOR, AND and OR gates shown in Fig. 5.7 match the desired specifications almost perfectly, the output curves of the best circuits found for the XOR and XNOR gates depicted in Fig. 5.8 fail to reach the power supply rails even for input voltage combinations close to the power supply rails. Yet, for a more relaxed set of specifications, e.g. $V_{\text{IL}} < 1\,\text{V}$, $V_{\text{IH}} > 4\,\text{V}$, $V_{\text{OL}} < 1\,\text{V}$ and $V_{\text{OH}} > 4\,\text{V}$ the best evolved XOR and XNOR gates would be solving the problem.

In order to further qualify the achieved gate characteristics, they are compared to the simulated DC behaviors of their standard cell counterparts shown on the right hand side of Fig. 5.7 and 5.8: In terms of the DC behavior considered here, the evolved NAND and NOR gates outperform the simulated text book solutions; they possess the higher gain and the transitions of their outputs are confined to a more narrow region. The output curves of the best evolved AND and OR gates on the other hand seem to be of similar quality as those of the simulated human-designed solutions. A comparison of both types, NAND and NOR versus AND and OR, suggests that the GA chose a two stage topology in both cases to match the desired specifications. For both classes, the NOR and NAND as well as the XOR and XNOR problem, the standard cell solutions would fail to perfectly satisfy the fitness requirements used throughout the evolution experiments. Since these circuits are successfully used as building blocks in all sorts of applications, it must be concluded that the used fitness criterion was too ambitious and therefore may have decreased the success rate of the evolution experiments.

Finally, in case or the XOR and XNOR problems, it should be noted that the number of transistor cells offered to the algorithm may have been to small: Although the geometrical setup described in section 5.2.2 provides 25 transistor cells, the author found it impossible to implement the text book solution shown on the right hand side of Fig. 5.5 within one hour of time. Thus a shortage in resources together with a fitness criterion too strict may partly explain the failure to find *perfect* solutions for the XOR/XNOR problems.

For the sake of fairness, it should be noted that the above comparison of evolved gates and their standard cell equivalents is restricted to their DC behavior and must be understood against the background of the particular fitness criterion used for the described evolution experiments. In addition to their DC behavior, the standard cell gates possess a bunch of beneficial qualities as for example a fast settling time or a low power and area consumption, which are not verified for the evolved circuits. However, as described in section 5.2.1.3, the evolved gates must at least settle in less than approximately 10 µs and are bound to use no more than 25 transistors.

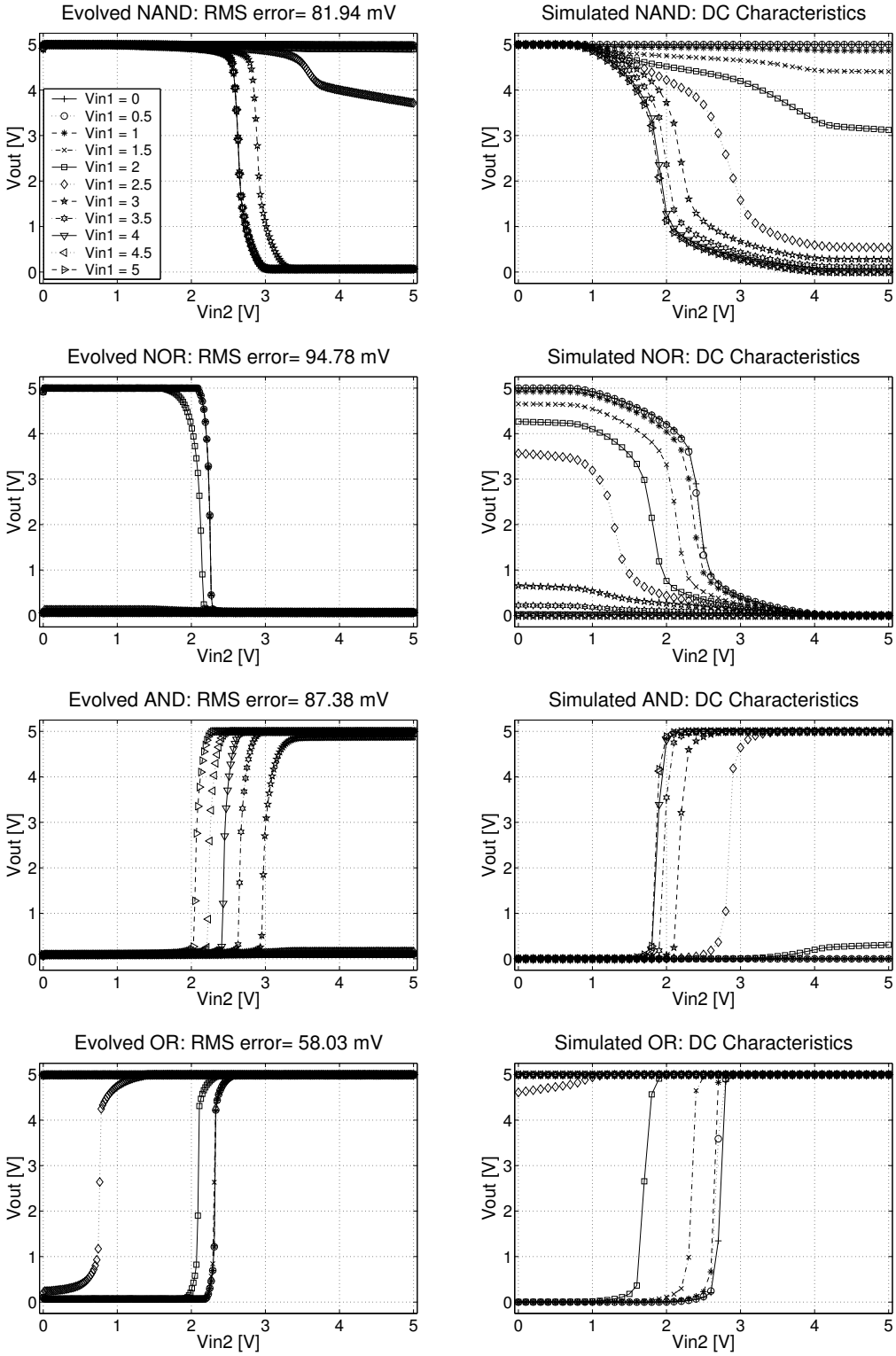


Figure 5.7: Left: Measured performance of the best evolved NAND, NOR, AND and OR gates. **Right:** Simulation results for the NAND, NOR, AND and OR gates depicted on the right hand side of Fig. 5.5. The legend is shown in the plot in the upper left corner.

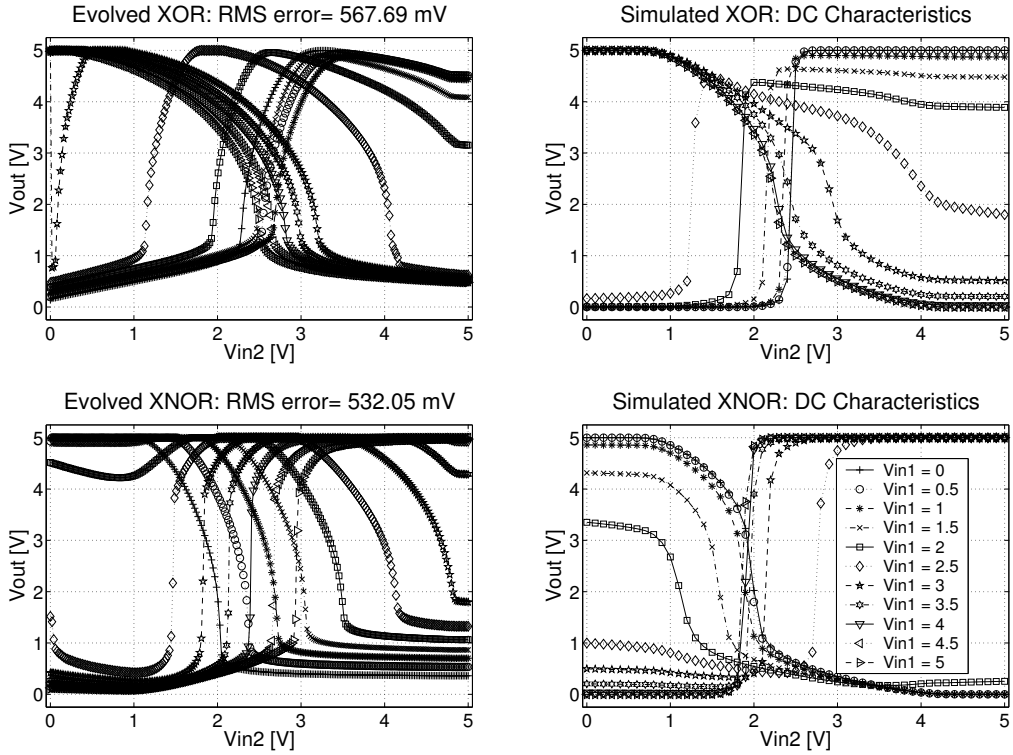


Figure 5.8: Left: Measured performance of the best evolved XOR, and XNOR gates. **Right:** Simulation results for the XOR and XNOR gates depicted on the right side of Fig. 5.5. The legend shown in the plot in the lower right corner is used for all 4 plots.

5.4.3 Performance Comparison for Different Tests

In order to verify that the behavior of the evolved gates is reproducible for multiple tests and similar for different chips, minimum and mean fitness of all 100 evolution runs is plotted against the test method for all six gates in Fig. 5.9. For each of the test methods 2 to 13 the best, mean and worst RMS error value obtained from 100 verification tests is plotted. The graphs are similar to those of Fig. 5.4; in fact, the curves denoted as the mean mean RMS error in the legend correspond to the curves plotted there. Since this second case study uses the same experimental setup as experiment 3 of case study I, the aforementioned mean mean curve can be compared with the according curve of Fig. 5.4 and is indeed found to be of similar shape and falling in the same range of RMS error values for all five logic gates considered in Fig. 5.4.

Similar to the results for experiment 3 of case study I, the mean as well as the minimum fitness calculated from the 100 runs is almost constant over all thirteen test methods. On one hand, this underlines the hypothesis that the randomization of the test data described in Table 5.5 forces the evolution process to find solutions that work for a variety of different test patterns. On the other hand, a comparison of the results for test methods 2 to 7 with those attained from methods 8 to 13 proves that the evolved circuits work similarly well on a second chip and thus can be assumed to possess a minimum of robustness against variations of the characteristics of the used devices. However, for the curves indicating the mean RMS error averaged over all 100 runs, the last fitness value, i.e. that for test method 1, is smaller than the fitness values for all other test methods. To some extend this can be explained with a highly converged population as discussed at the end of section 5.3. Following this argument, one actually has to compare the mean last fitness values with the *bes*t mean fitness achieved for the 100 verification tests of methods 4 and 7. The remaining small differences – at least

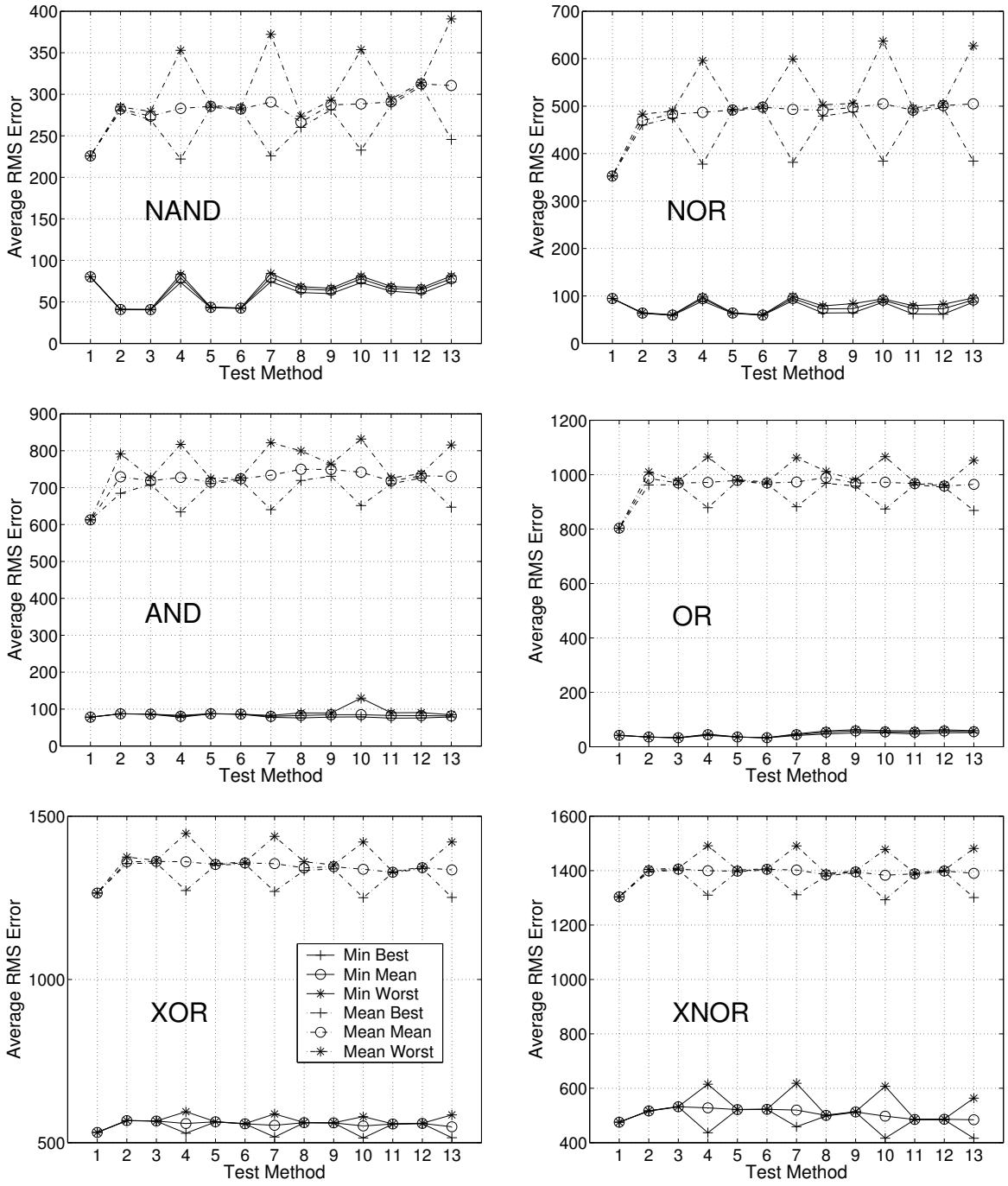


Figure 5.9: Minimum and mean fitness achieved for the 13 test methods defined in Table 5.7 for the logic gates NAND, NOR, AND, OR, XOR and XNOR (read from left to right, top to bottom). In case of the fitness values gained from the verification tests (methods 2 to 13), the minimum, mean and maximum RMS errors obtained within the 100 verification tests are plotted. The legend for all plots is shown in the graph belonging to the XOR gates (lower left).

observable in case of the NOR, AND and OR gates – may be due to circuits whose performance during evolution was not reproducible during the verification tests, because they strongly relied on the conditions present at the time of their evaluation. Examples for such conditions could be the temperature or the actual charge distribution on the chip.

The small spread of the curves for the worst, mean and best result achieved in 100 verification tests reveals that the performance of the evolved circuits is reproducible. Exceptions are the test methods for which V_{sweep} is randomized (methods 4,7,10,13). A considerable performance spread is observed for the means averaging over all 100 runs as well as for the performance of the best individuals found for the XOR and XNOR problems. Apparently, the RMS error obtained for imperfect solutions varies depending on the actual random order of the applied test pattern. However, this is not the case for the *perfect* solutions found for the NAND, NOR, AND and OR problem, because they respond correctly in *all* allowed test cases.

5.5 Results: Evolution of DC V-V Gaussian Circuits

The experiments dedicated to the artificial evolution of Gaussian function circuits are studied less rigorously – in terms of performed runs – than those dedicated to the synthesis of logic gates. Due to a lack of typical circuit implementations in the literature, it is hard to predict the number of transistors necessary to solve the problem. Accordingly, a series of 8 experiments has been carried out in which the number of transistor cells available to the algorithm is varied from 16 to 121; that is, the different sizes of the quadratic arrays are characterized by their edge length varying from 4 to 11. For each experiment 10 runs have been performed.

5.5.1 Overview over All Experiments

The results of all of the 80 runs are summarized in the 8 histograms shown in Fig. 5.10. The data used for these histograms is calculated according to (5.4) as the highest RMS error values obtained from 100 verification tests. The majority of the evolved Gaussian function circuits achieve RMS errors in the range of 50 to 350 mV per data point. Due to the small number of runs, significant differences in the results for different edge lengths of the array of transistor cells available to the GA are hardly recognizable. However, considering the mean and minimum RMS error for each available edge length plotted on the right hand side of Fig. 5.12, there seems to be a slight correlation between larger edge lengths and better results.

5.5.2 Output Characteristic of the Best Evolved Circuits

In order to illustrate the quality of the evolved circuits, the best solutions found for each array size are plotted in Fig. 5.11. Each graph contains the Gaussian target curve as well as the two transfer characteristics of the according circuit measured on chip 1 and chip 2.

While the measured curves plotted in Fig. 5.11 approximate the target function quite closely, they are not exactly of a Gaussian shape. In fact, the output characteristics of the evolved circuits rather resemble a piecewise nonlinear approximation of the Gaussian target function. For one, the synthesis of circuits whose output characteristic is exactly Gaussian is a difficult task. Even worse, the specification of absolute values for the width and amplitude of the target function in general as well as the actual values used in particular increase the difficulty of the task. For the other, the approximative nature of the evolved solution corresponds to the formulation of the fitness function which rewards close proximity to the target function rather than a Gaussian shape. All of the best-per-edge-length solutions work well on the second chip, too. From Fig. 5.11 it can be observed that

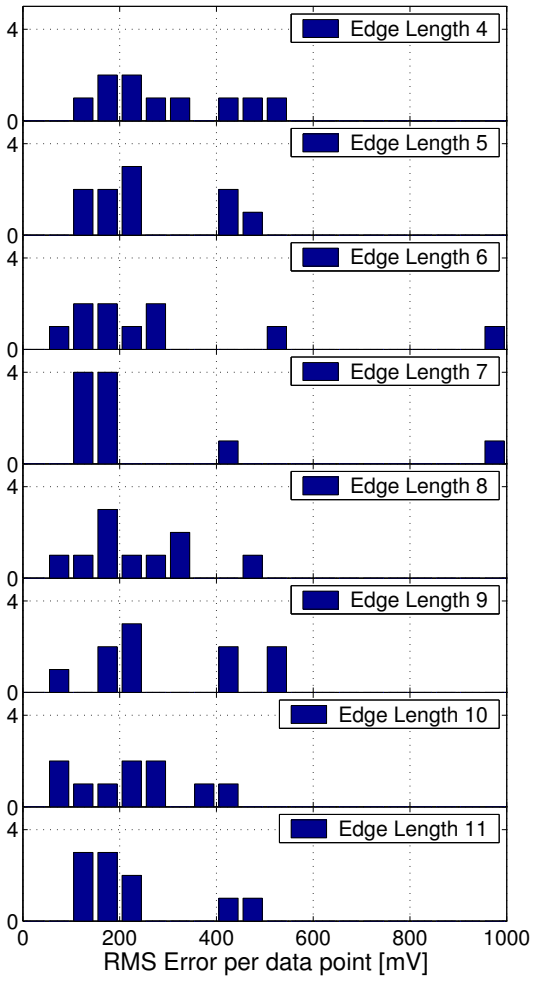


Figure 5.10: RMS deviation from the ideal response for the evolution of circuits exhibiting a Gaussian V-V characteristic. From top to bottom, the edge length of the square array of transistor cells is increased from 4 to 11. The RMS error is calculated as the maximum obtained from 100 verification tests.

the according output curves are slightly shifted towards higher voltages. Possible reasons for this behavior are different calibrations of the two evolution systems – as was already explained in section 5.4.1 and die-to-die variations between the used chips.

The RMS errors obtained for the best solutions obtained for each of the 8 array sizes available to the algorithm are listed in Table 5.8. From the maximum RMS error in mV the relative error in % is calculated by dividing the former one by the nominal output amplitude of the target curve. The according relative errors range from 2.5 to 4.9% and – on average – exceed those found in the related work summarized in Table 5.2 by a factor of 1.5 to 2. Since the work cited in Table 5.2 is

edge length	4	5	6	7	8	9	10	11
RMS error in mV	148.2	104.0	99.2	128.0	98.6	89.8	76.3	102.4
RMS error in %	4.94	3.47	3.31	4.27	3.29	2.99	2.54	3.41

Table 5.8: RMS error for the best-per-edge-length solutions. The maximum error value obtained from 100 verification tests is used. The RMS error in % is obtained by dividing the RMS error in mV by the nominal output amplitude of 3 V.

either a human design that was published in a journal ([Ket93]) or the result of extrinsic hardware evolution ([Zeb02], [Koz99f]), these results seem to be absolutely promising. Finally, one should

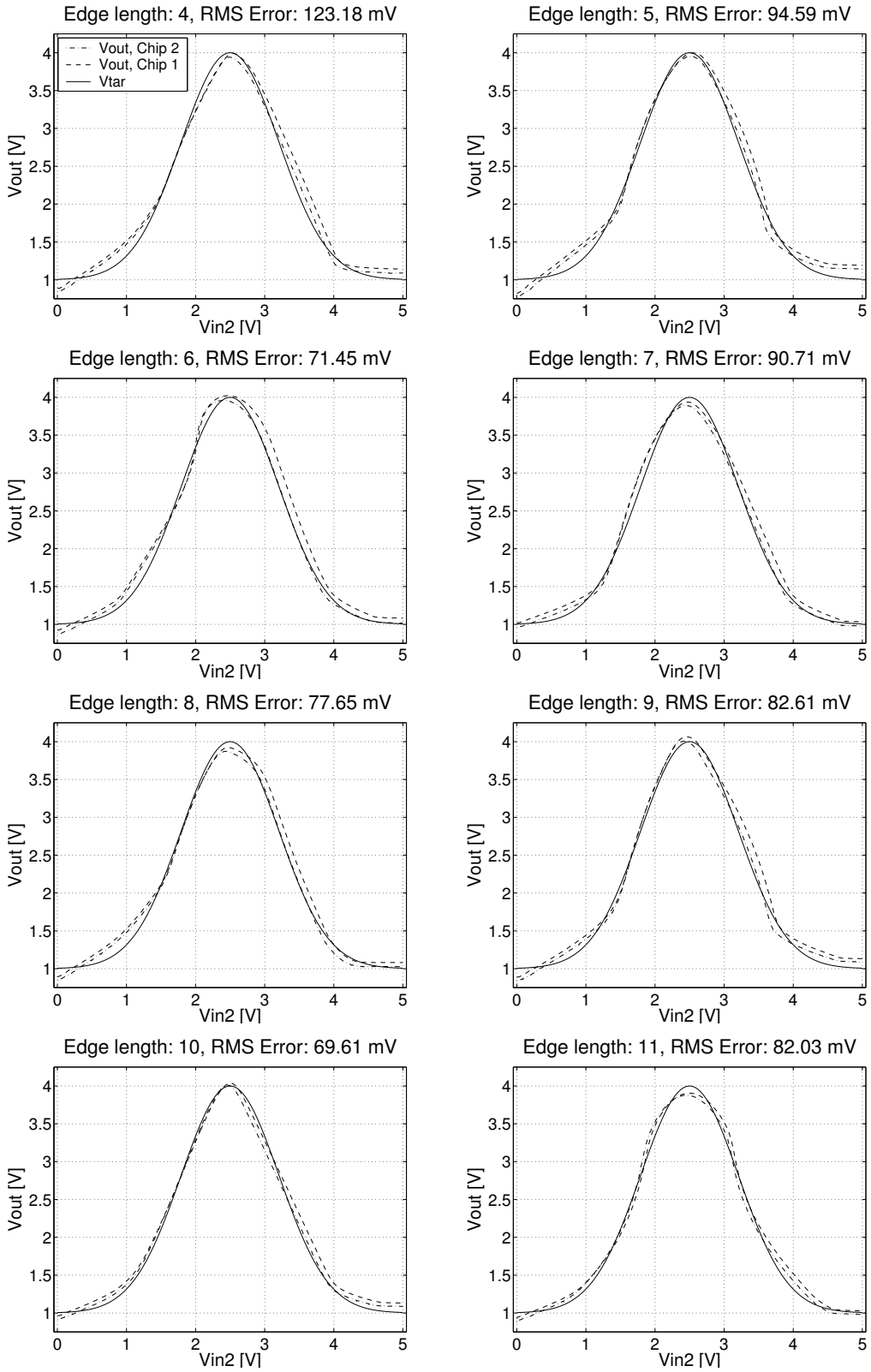


Figure 5.11: Output characteristic of the best Gaussian circuits for each available array size. The legend imprinted in the plot in the upper left corner is used throughout all eight graphs. The RMS error specified in the title is calculated from the last evaluation of the best-of-run solution of the respective run.

not overestimate this quantitative comparison, because the experimental setups differ considerably in terms of the specification of the target curve, the test pattern and the physical type of the in- and output signals utilized by the different groups.

5.5.3 Verification Measurements

In analogy to section 5.4.3 the reproducibility and transferability of the results attained for the evolved Gaussian function circuits shall be analyzed here. The results for the 100 verification tests are summarized on the left hand side of Fig. 5.12. Except for the minimum worst and mean last fitness for edge lengths 6 and 7 all curves are located in relatively close proximity. Thus, it can be stated, that the measured performance of the evolved circuits can be reproduced within the 100 tests. The remaining spread can be explained with the variation introduced by the different orders in which the input voltages were applied.

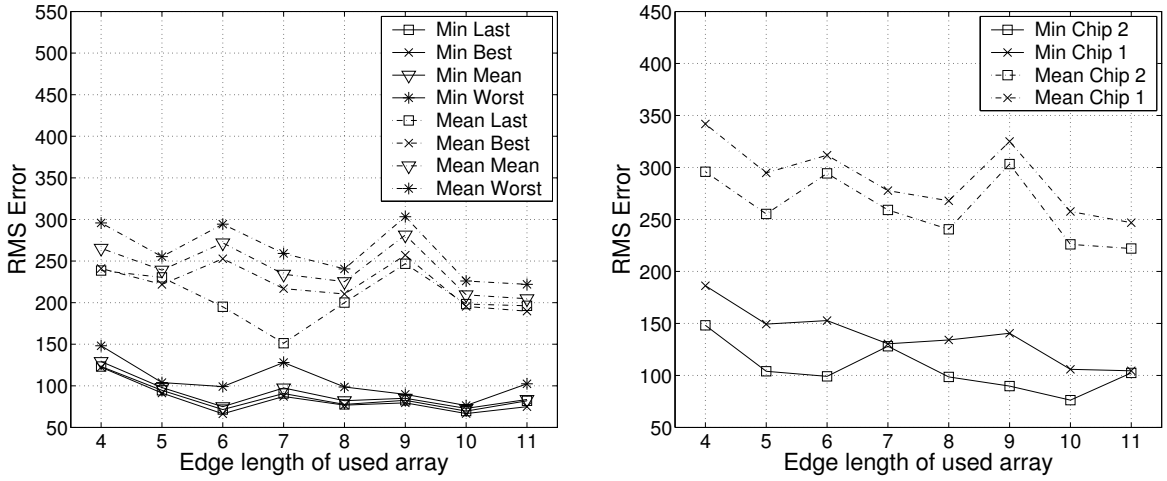


Figure 5.12: Left: Comparison of the minimum and mean of the last, best, mean and worst fitness values. While last refers to the last fitness value measured at the end of the evolution process, best, mean and worst denote the minimum, mean and maximum RMS error values obtained from the 100 verification measurements. Minimum and mean are then calculated from the ensemble of those according 10 values for each evolution run. **Right:** Comparison of the minimum and mean worst fitness measured on 2 chips.

The performance of the evolved circuits measured on a second chip, denoted as chip 2, is compared with that obtained from chip 1, the chip used throughout all of the evolution runs, in the graph on the right hand side of Fig. 5.12: It can be observed that the evolved circuit still work fairly well on the second chip. The fact, that the RMS error obtained on the second chip is up to 50 mV larger than its counterpart measured on the first chip may – at least partially – be due to differences in the calibration of the external analog circuitry of the two different evolution systems. This hypothesis is sustained by the systematic shift seen in the output characteristics discussed in section 5.5.2.

5.6 Discussion

Two different kinds of intrinsic hardware evolution experiments have been presented. The results show that the proposed evolution system is capable of finding quasi-DC solutions for simple analog circuit design tasks. While perfect solutions were found for the DC functionality of the four logic gates NAND, NOR, AND and OR, this was not the case for the XOR and XNOR functionality. On one hand, the failure of finding perfect solutions for the XOR and XNOR problems is due to

the higher difficulty immanent to the problem compared to the other four symmetrical two-input-gates considered here. On the other hand, an overly strict specification of the target function may have precluded the search process from finding solutions to the XOR/XNOR problem that would have been perfectly compliant with industry's standards for the DC output characteristic. It should be mentioned though, that some of the circuits found would satisfy a more relaxed specification of the output characteristic. Moreover, the XOR/XNOR circuits reported in the literature discussed in section 5.1.2.1 are not believed to perfectly satisfy the constraints posed to the candidate circuits throughout the experiments presented here. However, regarding the yield of perfect solutions for all six symmetric two-input gate problems, better algorithms will have to be found to face more realistic circuit synthesis problems.

By means of a series of experiments targeted at the evolution of logic gates, it was also shown that it is necessary to cover as many realistic test cases as possible in the test pattern used during the evolution process. A randomization of the test pattern was found to be necessary to prevent the GA from exploiting its spatio-temporal order. Finally, the successful evolution of Gaussian voltage characteristics demonstrates that it is feasible to find circuits approximating basic mathematical functions with the evolution system presented. Compared to the results reported in the literature the evolved Gaussian function circuits perform well, especially, if one takes the differences in the respective setup and target specification into account.

In general, the performance of the evolved solutions was found to be reproducible in 100 consecutive tests for the logic gates as well as for the Gaussian function circuits. Moreover, the circuits were found to work well on a second chip and thus are believed to be portable between different dice of the FPTA. Future work will have to answer the even more important question, whether the evolved circuits can be simulated by simple transistors (as opposed to programmable transistor cells) using the process parameters of the fabrication process the FPTA was produced with.

Chapter 6

Evolution of Digital-to-analog Converters

I never read; I only look at pictures.

ANDY WARHOL

Within this chapter the proposed system is used to find digital-to-analog converters (DACs) by hardware evolution for two reasons: First, DACs are a important building blocks in today's electronic systems. Second, the necessary setup is used to test the ability of the evolution system to handle problems that require multiple input signals. The target DACs are unipolar, possess a resolution of six bits and produce a voltage mode output. Different experiments investigate the influence of the geometrical setup as well as the dependency on the analog voltage levels used to code the input signals. While the evolved circuits achieve an effective resolution of up to five bits and are verified to work well at different time scales as well as on different dice, they lack the ability to abstract from the analog voltage levels of the digital input signals. It is experimentally verified that this can be remedied by inserting digital buffers at the circuits' inputs.

During the last decades, many signal processing tasks have been shifted from the analog to the digital domain. However, in order to interface electronic systems with the real world, digital signals have to be translated into physical signals, which usually requires a conversion into analog signals. Digital-to-analog Converters (DACs) thus have become key elements in many of today's electronic systems. As a matter of fact, they are used in a large variety of applications ranging from CD players to graphic cards, from wireless communication devices to automotive applications. Moreover, DACs are important components in industrial process control applications as well as in fly-by-wire systems used in modern airplanes. Accordingly, if evolvable hardware is ever to be useful for building up complex electronic systems, it will have to be able to interface to digital signals.

The DACs found by means of hardware evolution reported in the literature so far are restricted to three [Ben99] and four bits [Zeb01], [Zeb03]. The former experiments are based on simulations using a generic SPICE 3 model called from a genetic programming algorithm. It took approx. 4.5×10^7

evaluations to find the best-of-run solution, which uses bipolar transistors as well as resistors and capacitors. Since some of these possess values down to 1 Ω and up to 100 µF, a direct implementation of the circuit to one piece of silicon would be impractical.

The latter work by Zebulum et al. presents different divide and conquer approaches yielding 3 and 4-bit DACs. While the 4-bit DAC obtained in [Zeb01] possesses a current mode output and was tested using SPICE simulations, the circuit proposed in [Zeb03] was evolved using the field programmable transistor array (FPTA-2) chip described in [Sto02b] and produces an output voltage. In addition to facilitating artifical evolution by using a hierarchical approach, a total of four human designed operational amplifiers are included in the circuit. The work presented in this section focuses on designing unipolar digital-to-analog converters with a voltage mode output and a target resolution of six bits. All evolution runs were allowed to freely explore the used analog substrate, i.e. the FPTA, without any form of human guidance.

In order to be useful in real world applications a digital-to-analog converter must not rely on the exact voltage levels of its inputs. Hence, a number of experiments are devised to the problem of evolving circuits that are robust against those input voltage variations. Since this task turns out to be too difficult to be solved with the given setup, another series of experiments investigates if this obstacle can be remedied by human intervention, i.e. by inserting digital buffers at each of the six digital inputs.

6.1 Experimental Setup

6.1.1 Problem Definition

Digital-to-analog converters come in a large variety of architectures. First, they can be divided into serial and parallel DACs. The parallel architectures can be subdivided into voltage, charge and current scaling DACs as well as multistage implementations that may employ mixtures of the different scaling mechanisms. As can be seen from Fig. 6.1, a complete parallel DAC implementation comprises - besides the actual scaling network for the conversion - a digital interface, a reference voltage generator and an output stage.

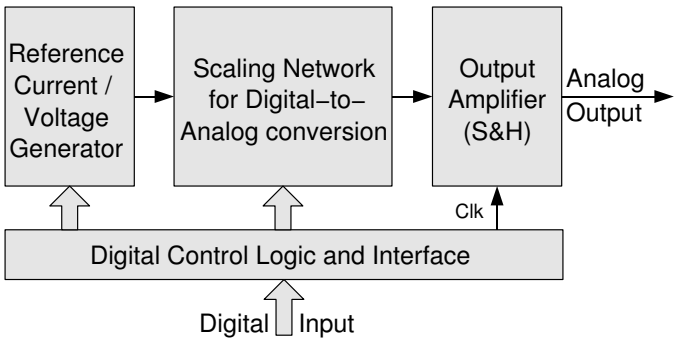


Figure 6.1: Implementation of a parallel DAC.

For a *multiplying* DAC, the output is given as the product of the digital input code and the input reference range (either in- or external). The output stage could be used to buffer the output of the scaling network, avoid glitches at the output during code transitions, transform the output to the desired quantity (i.e. voltage or current), or to change sign or scale of the output. The experiments presented within this chapter focus on the artificial evolution of a voltage scaling network in that, on one hand, no reference voltages were provided and, consequently, no multiplying abilities of the

candidate solutions were tested, and on the other hand, neither a load impedance was applied to the output, nor a glitch measurement was carried out. The evolved unipolar DACs were targeted to exhibit a resolution of six bits; no measures to encourage the evolution of multi-staged DACs were taken.

6.1.2 Overview of the Experiments

Altogether five series of eight experiments, each featuring 20 runs, were carried out, as summarized in Table 6.1. For the series FW1, FW4 and FWB4 the task was to map the digital words to analog voltages in an unsigned binary encoding, where the lowest word (all inputs low) corresponds to the lowest output and the highest word (all inputs high) to the highest output voltage (see Fig. 6.2). In the remaining two series INV1 and INV4 the encoding is inverted, that is, the output voltage should be at its maximum for the lowest input word and vice versa.

Series	Input Encoding	Number of Curves	Input Voltage Level V_I in V
FW1	Forward	1	0, 5
FW4	Forward	4	0, 5 ; 0.5, 4.5 ; 1, 4 ; 1.5, 3.5
INV1	Inverse	1	0, 5
INV4	Inverse	4	0, 5 ; 0.5, 4.5 ; 1, 4 ; 1.5, 3.5
FWB4	Forward Buffered	4	0, 5 ; 0.5, 4.5 ; 1, 4 ; 1.5, 3.5

Table 6.1: The five different experiment series.

Since in initial experiments the output of the evolved DAC circuits was found to strongly depend on the input voltage levels, series FW4 and INV4 were designed to evolve circuits that rely only on the digital information present at the inputs. This is achieved by testing each candidate circuit with four different input voltage levels as described in Table 6.1.

Assuming a dependency of the output on the input voltage level, it is interesting to investigate whether the task is more difficult for a reversed encoding scheme: The reverse encoding might bias artificial evolution to use inverters at the digital inputs, thereby gaining robustness against the input voltage variations. This should be observable by a comparison of series INV4 and FW4. Finally, in the experiments of series FWB4 digital buffers are inserted at the inputs of the circuit under test to restore the analog voltage level of the input signals (cf. Fig. 6.3). Thereby, evolution of DACs robust against input voltage level variations should be significantly facilitated.

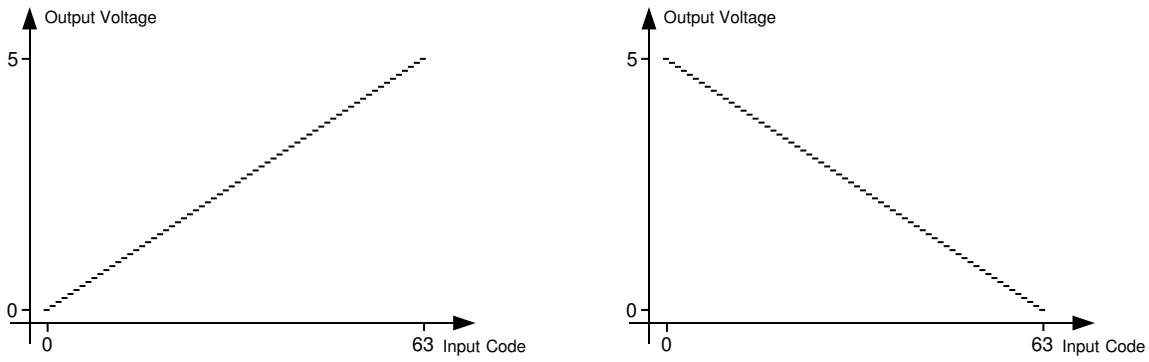


Figure 6.2: Ideal voltage mode DAC output: **Left:** Forward response **Right:** Inverse response.

For each series of experiments, three parameters of the setup are varied as shown in Table 6.2. First, the desired output voltage range is varied between the intervals 0 to 5 V and 1 to 4 V, where

Experiment Number	Output range	Used Cell Array Dimensions	Input Order
1	0...5 V	14×14	forward
2	1...4 V	14×14	forward
3	0...5 V	10×10	forward
4	1...4 V	10×10	forward
5	0...5 V	14×14	reverse
6	1...4 V	14×14	reverse
7	0...5 V	10×10	reverse
8	1...4 V	10×10	reverse

Table 6.2: Experiments carried out for each experiment series.

the former one corresponds to the power supply range of the programmable transistor array. Second, two differently sized areas were made available to the GA. The according locations used for inputs and output are depicted in Fig. 6.3. The upper row contains the geometrical setups for all series of experiments except for those of series FWB4, which is depicted in the lower row. The setups for experiments 1,2,5 and 6 are shown in the left column of Fig. 6.3, whereas those for experiments 3,4,7 and 8 are illustrated on the right hand side of the figure. Assuming the GA uses a resistive network to solve the DAC design problem, the task intuitively appears easier for a setup that places the more significant bits close to the circuit's output, because they are expected to influence it more directly. Accordingly, this *reversed* input order is used for experiments 5 to 8 to test the above hypothesis.

6.1.3 Fitness Function

The fitness function used throughout all experiments is simply the sum of the squared errors

$$\mathrm{SSE} = \sum_{j=0}^{63} (V_{\mathrm{out}}(j) - V_{\mathrm{tar}}(j))^2 \tag{6.1}$$

with regard to the target function

$$V_{\mathrm{tar}}(j) = \begin{cases} V_{\mathrm{low}} + (V_{\mathrm{high}} - V_{\mathrm{low}})\frac{j}{63} & \text{for} \quad \text{FW1, FW4, FWB4} \\ V_{\mathrm{high}} - (V_{\mathrm{high}} - V_{\mathrm{low}})\frac{j}{63} & \text{for} \quad \text{INV1, INV4} \end{cases} , \tag{6.2}$$

where the integer input code j is calculated from the inputs V_{I_i} by

$$j = \sum_{i=0}^{5} I_i \cdot 2^i \qquad \text{with} \qquad I_i = \begin{cases} 0 & \text{if} \quad V_{I_i} < 2.5\,\mathrm{V} \\ 1 & \text{if} \quad V_{I_i} > 2.5\,\mathrm{V} \end{cases} , \tag{6.3}$$

and V_{low} and V_{high} are the boundaries of the output ranges listed in Table 6.2. Accordingly, this sum of squared errors has to be minimized by means of the used algorithm. This choice of fitness function aggregates the different objectives high linearity, exact gain and minimal offset, but does not allow to control the weight of their contributions to the total fitness.

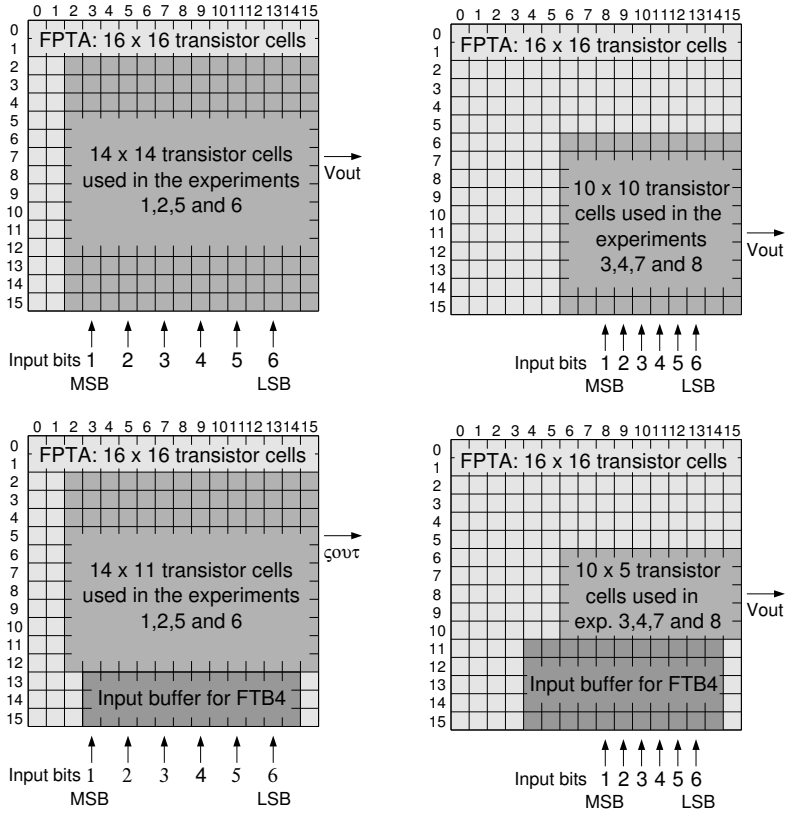


Figure 6.3: Geometrical setup for experiments 1,2,5 and 6 (upper left corner), 3,4,7 and 8 (upper right corner) for all series, except for FWB4: Experiments 1,2,5 and 6 (lower left corner) and 3,4,7 and 8 (lower right corner). For the experiments 5 to 8 the order of the input bits is reversed (6 to 1 from left to right instead of 1 to 6).

6.1.4 Test Patterns

For series FW1 and INV1 all of the 64 input codes are tested exactly once resulting in one output curve. In case of the other series (FW4, INV4 and FWB4) each input code was tested for all different input voltage levels yielding a total of four output curves. In order to prevent artificial evolution from abusing information from the timing/order of the test pattern, one out of ten different random orders is chosen randomly for each fitness test. In addition, this ensures that varying input code transitions are used for the fitness evaluations in the course of the evolution process.

Due to the fact that the FPTA has only one single analog input, the input voltages have to be written sequentially to the IO cells of the chip. During evolution the time between the application of two successive input voltages is 167 ns. The output voltage is sampled approximately 1.27 µs after the first input and 0.47 µs after the last input voltage is applied to the transistor array. Thus the sample frequency with which the different input codes are tested amounts to 750 kHz. The values of the test pattern timing are summarized in Table 6.3.

In order to test whether the evolved converter circuits are also working on a different time scale, verification tests were done at the sample rate of 750 kHz used during evolution as well as at 12.4 kHz, where the latter timing is referred to as *slow*. A complete run featuring 10,000 generations and a population size of 50 took between 15 and 20 minutes depending on the number of different input voltage levels.

Time Parameter	Normal Test	Slow Test
settling time for last input	0.47 µs	13.4 µs
settling time for first input	1.27 µs	80.6 µs
system clock f_{sys}	36 MHz	18 MHz
sample frequency high f_S	750 kHz	12.4 kHz
time per run: FW1, INV1	≈ 15 min	-
time per run: FW4, INV4, FWB4	≈ 20 min	-

Table 6.3: Time and Timing considerations for the DAC experiments.

6.1.5 GA Parameters

Throughout all 40 experiments a simple genetic algorithm was used in conjunction with truncation selection. As can be seen from Table 6.4, a large fraction of 20% was directly promoted to the next generation in order to prevent the algorithm from loosing an already good solution due to noise in the measuring process. The individuals taking part in crossover were chosen from the best 60% and

GA Parameter	Value
population size	50
number of generations	10000
selection scheme	truncation selection
reproduction fraction	0.2
mutation fraction	0.4
mutation rate Terminal Connection	3%
mutation rate Width, Length	2%
mutation rate Routing	3%
crossover fraction	0.6
crossover rate	30%
crossover block size	2

Table 6.4: Genetic algorithm parameters used for the presented DAC experiments.

the ones undergoing only mutations from the best 40% of the current generation, respectively. Since the crossover block size is limited to two, the genetic differences between two generations are fairly small and mutation was probably the driving force in the evolution process. For each experiment a total of 20 runs has been performed.

6.2 Results for Series FW1

The best genotypes of the last generation of all evolution runs are taken as the result of the experiment. After all runs had been finished, the phenotypical behavior of all these genotypes was verified by testing the according circuit response for 100 times using the same test patterns as during the evolution process. On one hand, the resulting data is compared to the fitness achieved during evolution, and on the other hand, it is used to calculate the derived quality measures offset, gain, differential and integral nonlinearity.

Since the sum of squared errors defined in (6.1), which is used for the fitness evaluation during evolution, is not an intuitive quality measure, it is converted to the root mean square error per data

point in lsb[1] by

$$F = \mathrm{RMSE} = \frac{\sqrt{\frac{\mathrm{SSE}}{N_{\mathrm{IC}}}}}{1\,\mathrm{lsb}} \qquad \text{with} \qquad N_{\mathrm{IC}} = \begin{cases} 64 & \text{for} \quad \text{FW1, INV1} \\ 256 & \text{for} \quad \text{FW4, INV4, FWB4} \end{cases} \text{, where} \tag{6.4}$$

$$1\,\mathrm{lsb} = \frac{V_{\mathrm{tar}}(63) - V_{\mathrm{tar}}(0)}{63} \tag{6.5}$$

corresponds to 79.4 mV for an output range of 0 to 5 V and 47.6 mV for one of 1 to 4 V, respectively. Equation (6.4) is used to calculate the fitness values throughout the remainder of this chapter.

6.2.1 Root Mean Square Error

The influence of the eight different experimental setups listed in Table 6.2 is studied exemplary for series FW1. In Fig. 6.4 the results of all experiments are plotted as eight histograms. For each run

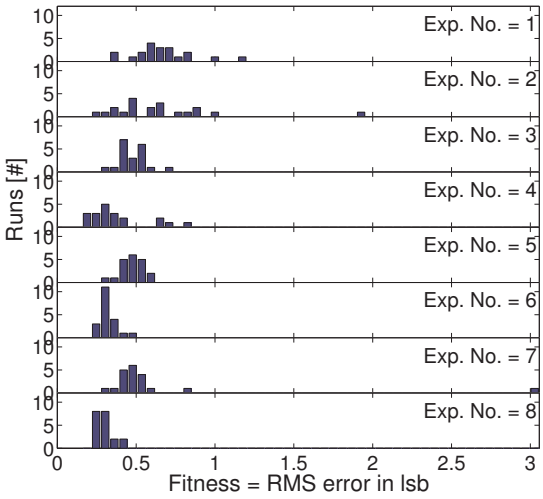


Figure 6.4: Fitness Histograms for all experiments for the experiment series FW1.

the worst fitness value out of 100 verification measurements is used for the plot. Apparently, the runs targeted at an output range of 1 to 4 V performed significantly better than their counterparts required to cover the full power supply range with their outputs. In contrast, neither the geometrical setup nor the size of the transistor array available to the EA influences the evolution results significantly, except for the combinations chosen for experiments 1 and 2: Evolving on the large array of 14×14 cells (see Fig. 6.3) together with having the less significant bits closer to the output edge yields worse results than all other combinations, independent of the output voltage range.

6.2.2 Nonlinearity, Offset and Gain Error

One of the most important measures to evaluate the quality of digital-to-analog converters are their INL[2] and DNL[3]. They are defined as

$$\mathrm{DNL}(j) = \frac{V_{\mathrm{out}}(j) - V_{\mathrm{out}}(j-1)}{V_{\mathrm{lsb}}} - 1 \quad \text{for} \quad j = 1,2,\ldots,63 \tag{6.6}$$

$$\mathrm{INL}(j) = \frac{V_{\mathrm{out}}(j) - (V_{\mathrm{out}}(0) + V_{\mathrm{lsb}} \cdot j)}{V_{\mathrm{lsb}}} \quad \text{for} \quad j = 0,1,\ldots,63 \quad \text{, with} \tag{6.7}$$

[1] lsb = least significant bit
[2] Integral NonLinearity
[3] Differential NonLinearity

$$V_{\text{lsb}} = \frac{V_{\text{out}}(63) - V_{\text{out}}(0)}{63} \quad . \tag{6.8}$$

Differential as well as integral nonlinearity are plotted in Fig. 6.5 for the *best* circuit of series FW1, where *best* refers to the lowest RMS error achieved. This circuit was found among the runs of experiment four. The INL and DNL values are averaged over 100 verification tests, and the error bars

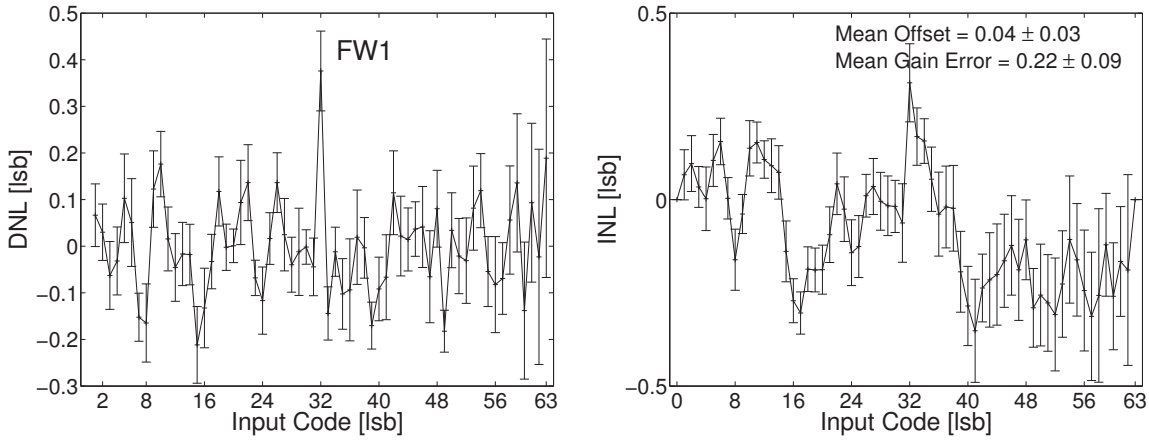


Figure 6.5: DNL (left) and INL (right) for the best evolved DAC of series FW1 (experiment 4).

indicate the according standard deviations. As can be seen from Fig. 6.5, both, INL as well as DNL amount to less than ± 0.5 lsb, error bars included. It is thus save to say, that the linearity of this DAC, on average, complies with the full target resolution of six bits.

However, the histograms of Fig. 6.6 illustrate that this does not hold for worst case conditions: For each of the 100 verification tests the absolute maximum DNL/INL value is determined. The maximum of the resulting 100 values is taken as the result for one run and appears in the according histogram. The bin size was set to 0.25 lsb for all X-Axes. While a considerable amount of evolved DACs manage to achieve maximum nonlinearities of less than 1 lsb for experiments 4,6 and 8, no single circuit was found to have a nonlinearity of less than 0.5 lsb. Using the definition of the DNL given in 6.6, it can be deduced that a DAC's output is bound to be monotonic if $|\text{DNL}| < 1$ is satisfied. Hence, the histograms in Fig. 6.6 indicate that for experiments 3 to 8 in the order of five to ten evolved DACs possess a monotonic output characteristic.

Since differential as well as integral nonlinearity, defined in (6.6) and (6.6), respectively, disregard offset OS and gain error GE inherent to the measured output characteristic, these criteria also have to be taken into account. They are calculated from the following equations:

$$\text{OS} = \frac{1}{\text{lsb}} \cdot (V_{\text{out}}(0) - V_{\text{tar}}(0)) \tag{6.9}$$

$$\text{GE} = \frac{1}{\text{lsb}} \cdot [(V_{\text{out}}(63) - V_{\text{out}}(0)) - (V_{\text{tar}}(63) - V_{\text{tar}}(0))] \quad . \tag{6.10}$$

It should be mentioned, that the four characteristics INL, DNL, offset and gain error and the fitness measure sum of squared errors (SSE) from (6.1) are not independent. Solving (6.7) for

$$V_{\text{out}}(j) = \text{INL}(j) \cdot V_{\text{lsb}} + V_{\text{out}}(0) + V_{\text{lsb}} \cdot j \quad . \tag{6.11}$$

and inserting the result into (6.6) yields:

$$\text{DNL}(j) = \text{INL}(j) - \text{INL}(j-1) \quad . \tag{6.12}$$

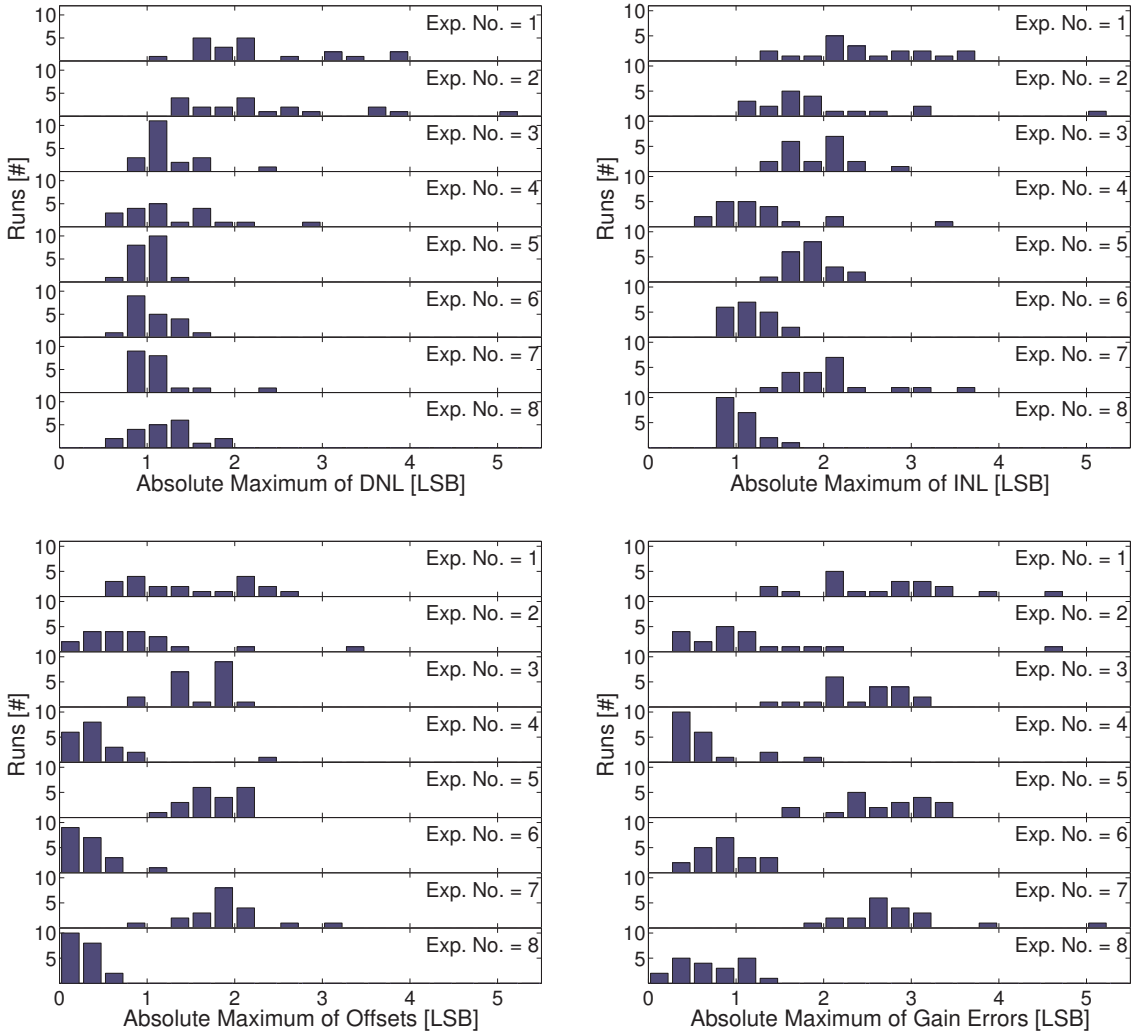


Figure 6.6: Histograms for all experiments in the category FW1 for the four different criteria: **Upper left:** INL, **upper right:** DNL, **lower left:** Offset and **lower right:** Gain error. For all four measures the absolute maximum of 100 verification tests is plotted. In case of the DNL and INL the maximum nonlinearity with respect to all input codes is used. The last bin contains all runs exceeding an error of 5 lsb for the respective property.

Similarly, the sum of squared errors SSE given in (6.1) can be expressed by means of INL, offset and gain error. Therefore (6.5) and (6.8) are used to rewrite (6.10)

$$V_{\mathrm{lsb}} = \mathrm{lsb} \cdot \left(\frac{\mathrm{GE}}{63} + 1 \right) \tag{6.13}$$

and (6.9) is solved for

$$V_{\mathrm{out}}(0) = \mathrm{lsb} \cdot \mathrm{OS} + V_{\mathrm{tar}}(0) \qquad . \tag{6.14}$$

Thus, inserting (6.13) and (6.14) into (6.11) yields

$$V_{\mathrm{out}}(j) = V_{\mathrm{tar}}(0) + \left[\mathrm{INL}(j) \left(\frac{\mathrm{GE}}{63} + 1 \right) + \mathrm{OS} \right] \cdot \mathrm{lsb} + \mathrm{lsb} \cdot \left(\frac{\mathrm{GE}}{63} + 1 \right) \cdot j \qquad , \tag{6.15}$$

Finally, equation 6.15 can be utilized to replace V_{out} in (6.1) to obtain the desired formula for the sum of squared errors:

$$\text{SSE} = \sum_{j=0}^{63} \left[(V_{\text{tar}}(0) + \text{INL}(j) + \left(\frac{\text{GE}}{63} + 1\right) + \text{OS}) \cdot \text{lsb} + \text{lsb} \cdot \left(\frac{\text{GE}}{63} + 1\right) \cdot j - V_{\text{tar}}(j) \right] \quad , \quad (6.16)$$

which depends only on the three derived quantities INL, GE, and OS and a few constants. Accordingly, minimizing the fitness function used during evolution described by (6.1) minimizes all of the four objectives used to characterize the evolved circuits, namely offset, gain error and both nonlinearities. However, the weighting of the different criteria is fixed in the evolution process and the values of these derived objectives can not be obtained from the fitness criterion.

In addition to INL and DNL, offset as well as gain error of the evolved DACs of series FW1 are depicted as histograms in Fig. 6.6. Again, the largest values obtained from 100 verification tests are used. In general, the results for the nonlinearities, offset and gain error are similar to those of the fitness histogram depicted in Fig. 6.4. The more detailed analysis, however, reveals that the worse results for DACs covering the full power supply range are due to their larger INL, offset and gain error and not to larger DNL values. The – on average – larger values for offset and gain error suggest, that the evolved circuits fail to reach the boundaries of the power supply range, which causes the output curves to deviate more severely from the ideal linear output characteristic as indicated by the INL histograms. The fact, that the results achieved for the larger output voltage range are inferior to those for the smaller range of 1 to 4 V may be either caused by the increased problem difficulty, or by deteriorations of the analog in-/output signal present in the vicinity of the power supply rails. The former argument would e.g. be plausible if the algorithm chose to create a resistive voltage scaling network to solve the conversion task that contains an additional biasing network connected to power and ground, in order to fine tune the output range.

As already mentioned in the discussion of Fig. 6.4, geometrical setup of the inputs affect the results most significantly in combination with the large array size used in experiments 1 and 2. However, Fig. 6.6 indicates that this is rather caused by larger INL (exp. 2) and DNL (exp. 1, 2) values. The combination of the enlarged search space and the counterintuitive geometrical input order seems to either cause the algorithm to get stuck in local optima or to decelerate its convergence.

6.3 Comparison of the Five Different Series of Experiments

The results presented in the previous section (6.2) look promising. Unfortunately, the obtained DAC circuits are found to strongly rely on the analog voltage level of their digital input signals. This section compares the results of different attempts to account for those dependencies to each other as well as to the results obtained for series FW1.

6.3.1 Root Mean Square Error

In order to compare the overall fitness given as the root mean square error (RMSE) of the five different series of experiments, the mean as well as the minimum RMSE is calculated for the best, mean, and worst fitness value obtained from 100 verification tests, respectively. Accordingly, if

$$F_{rt} \quad t = 1 \ldots 100 \, , \, r = 1 \ldots 20 \quad (6.17)$$

denotes the fitness of run r and test t, the minimum and mean fitness values referred to in Fig. 6.7 are calculated as indicated in Table 6.5. These fitness values are compared to the minimum and mean fitness achieved at the end of the evolution runs in Fig. 6.7. First, it can be observed that

Name	Mathematical Expression	Name	Mathematical Expression
Min. Last	$\min_{r=1...20}(F_{\text{last}r})$	Min. Mean	$\min_{r=1...20}(\text{mean}_{t=1...100}(F_{rt}))$
Mean Last	$\text{mean}_{r=1...20}(F_{\text{last}r})$	Mean Mean	$\text{mean}_{r=1...20}(\text{mean}_{t=1...100}(F_{rt}))$
Min. Best	$\min_{r=1...20}(\min_{t=1...100}(F_{rt}))$	Min. Worst	$\min_{r=1...20}(\max_{t=1...100}(F_{rt}))$
Mean Best	$\text{mean}_{r=1...20}(\min_{t=1...100}(F_{rt}))$	Mean Worst	$\text{mean}_{r=1...20}(\max_{t=1...100}(F_{rt}))$

Table 6.5: Mathematical meaning of the different minima and averages in Fig. 6.7.

minimum of the last, best, mean, and worst fitness values per experiment do not differ significantly. This also holds for the according averages, with four exceptions: In series FW1 the mean last fitness value differs considerably from the remaining best, mean, and worst ones for experiment 7. Similar deviations occur in series FTB4 for experiments 2,3 and 4. These differences between the mean

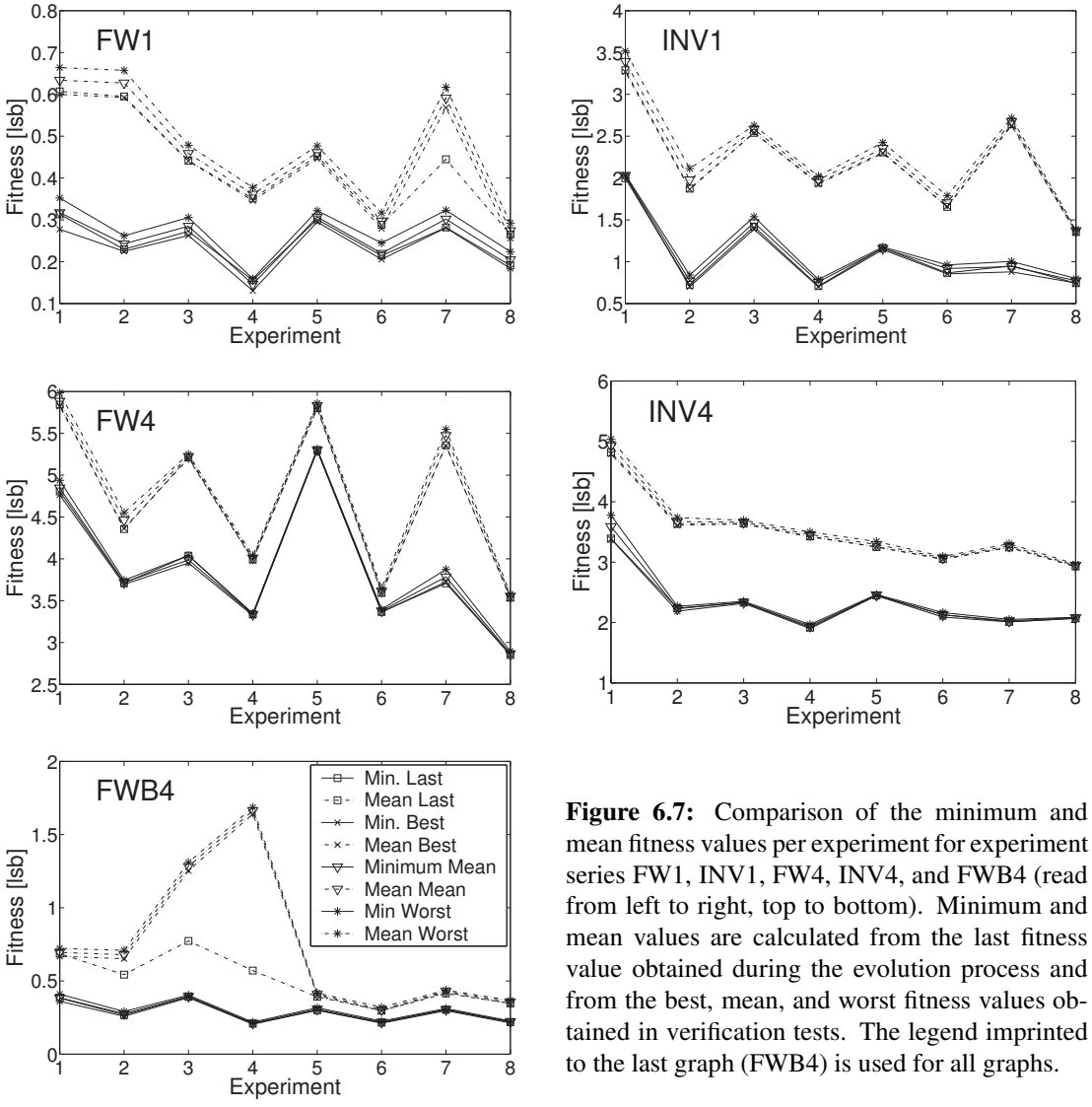


Figure 6.7: Comparison of the minimum and mean fitness values per experiment for experiment series FW1, INV1, FW4, INV4, and FWB4 (read from left to right, top to bottom). Minimum and mean values are calculated from the last fitness value obtained during the evolution process and from the best, mean, and worst fitness values obtained in verification tests. The legend imprinted to the last graph (FWB4) is used for all graphs.

last fitness measured at the end of each run and the according mean fitness values obtained from the verification tests can be understood by means of Fig. 6.8:

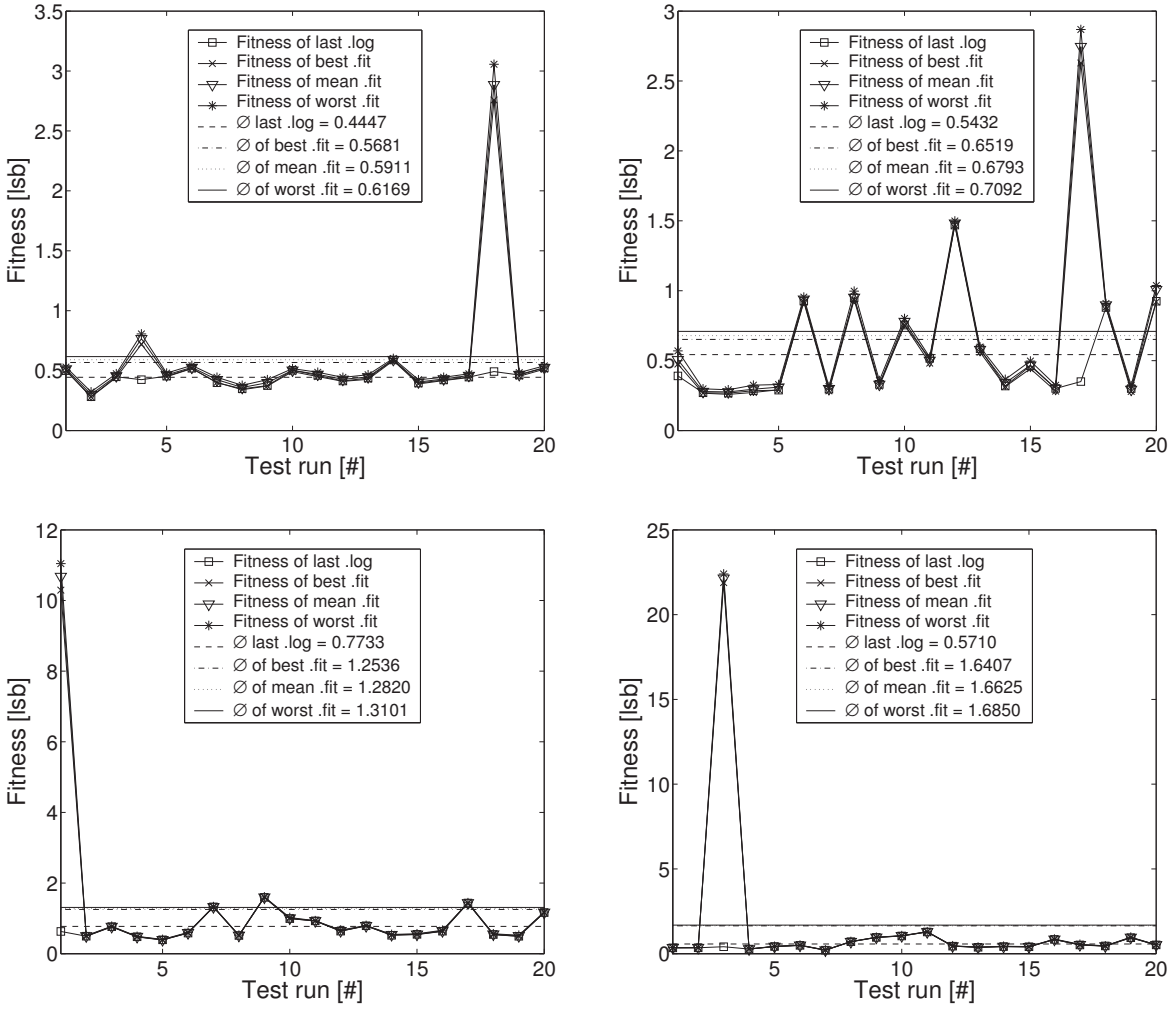


Figure 6.8: Comparison of the last fitness achieved during evolution with the best, mean, and worst fitness values measured in 100 verification tests: **Upper left**: series FW1, experiment 7, **upper right**: series FWB4, experiment 2, **lower left**: series FWB4, experiment 3, **lower right**: series FWB4, experiment 4, The best, mean, and worst are obtained from 100 measurements for each run.

The four graphs show the last, best, mean, and worst fitness for each run of the aforementioned experiments of series FW1 and FWB4 as well as the according averages with respect to all 20 runs. In all four cases, exactly one evolved circuit exhibits a performance during verification tests that is not anywhere near its fitness obtained at the end of the evolution run, while for all other runs the last fitness values are in close vicinity of their counterparts measured for verification. The malfunctioning of DACs tested outside the evolution loop may be due to a change of environmental conditions between evolution and verification test. For instance, the circuit may strongly depend on global conditions like temperature of supply voltage or may have exploited the charge distribution left on the FPTA by the previously tested candidate solution.

In summary, Fig. 6.7 and 6.8 prove that the performance of almost all of the evolved analog to digital converters is reproducible on the chip they are evolved on. The width of the remaining distribution of measured fitness can, for instance, be caused by the limited precision of the measurement

itself and/or the variation among the ten different randomly chosen random orders. The different random orders mentioned in the latter argument contain different input code transitions, which require different lower bounds to the slew rate inherent to the circuits under test.

In principle, the results shown in Fig. 6.7 agree with those discussed in section 6.2: The best as well as the average fitness achieved in experiments requiring the smaller output voltage range usually outperforms that of the according experiments asking for the larger output range. The dependency on geometrical input order and array size available to the EA is less unique. On average, the experiments requiring the smaller output range yield slightly better results for the reverse input order. Except for series FTB4 a reduced design space of 10×10 cells seems to improve the average fitness per experiment. Presumably, this does not hold for series FWB4, because the design space is reduced too severely by inserting the input buffers (cf. Fig. 6.3). However, the best runs are found for the settings of experiment four throughout all series except for series FW4.

The plots of Fig. 6.7 as well as the histograms of Fig. 6.4 and 6.9 show that it is significantly harder to find digital-to-analog converters that use an inverse encoding as required in series INV1 (cf. section 6.1.2). The results for the two series FW4 and INV4, in which the output characteristic is tested for four different input voltage levels, are even worse. This indicates that the EA strongly relies on the analog voltage level of the digital inputs instead of extracting the digital information included. As was expected, the circuits produced in series INV4 behave – on average – slightly better than their counterparts of series FW4. The necessary inversion of the input signals seems to be helpful in abstracting the digital information from the analog input signals. Moreover, the histograms for series INV1 and INV4 in Fig. 6.9 show that the final fitness values are spread out more broadly than those obtained for series FW1 and FW4 (Fig. 6.9 and 6.4). Either the used algorithm ends up in a larger variety of different local optima for this task, or an increase in the number of generations would considerably improve the obtained results. However, as can be inferred from the graphs for series INV1, the algorithm did never choose to place inverters at the inputs, because this would have resulted in circuits with fitness values similar to those of the runs in series FW1. It is worth noting though, that the gain of one stage inverters realizable with the FPTA's transistor cells is not sufficient to restore all four different input voltage levels to exactly 0 and 5 V. Hence, inverting the input signals once does not solve the problem entirely.

Finally, a comparison of the histograms for series FWB4 in Fig. 6.9 with that of series FW1 shown in Fig. 6.4 proves that the desired robustness against variations of the input voltage levels can be achieved by inserting buffers (two inverters in series) at the inputs of the prospective DAC circuits. Thereby, the total number of used transistor cells was almost preserved, as can be seen from Fig. 6.3. Thus, the resources available to the EA for implementing the digital-to-analog converter are reduced accordingly; in fact, for the setup using the smaller array size, they are actually halved. This and/or the harder timing constraints caused by the additional two gate delays of the input buffers may be responsible for the fact, that the circuits evolved in FWB4 perform slightly worse than those from series FW1.

6.3.2 Offset, Gain and Nonlinearities

As already explained in section 6.2.2, the quality of the evolved DACs can be further appraised by means of the measures INL, DNL offset and gain error defined in (6.6), (6.6)), (6.9) and (6.10), respectively. In case of the series FW4, INV4 and FWB4, all of the four derived measures are calculated separately for each of the four curves first; afterwards the maximum of all four curves is used for the remaining calculations. In order to survey the results of all runs, DNL and INL, as well as offset and gain error, are combined as described in Table 6.6. The resulting maxima of all experiments are shown in five scatter plots for the different series in Fig. 6.10.

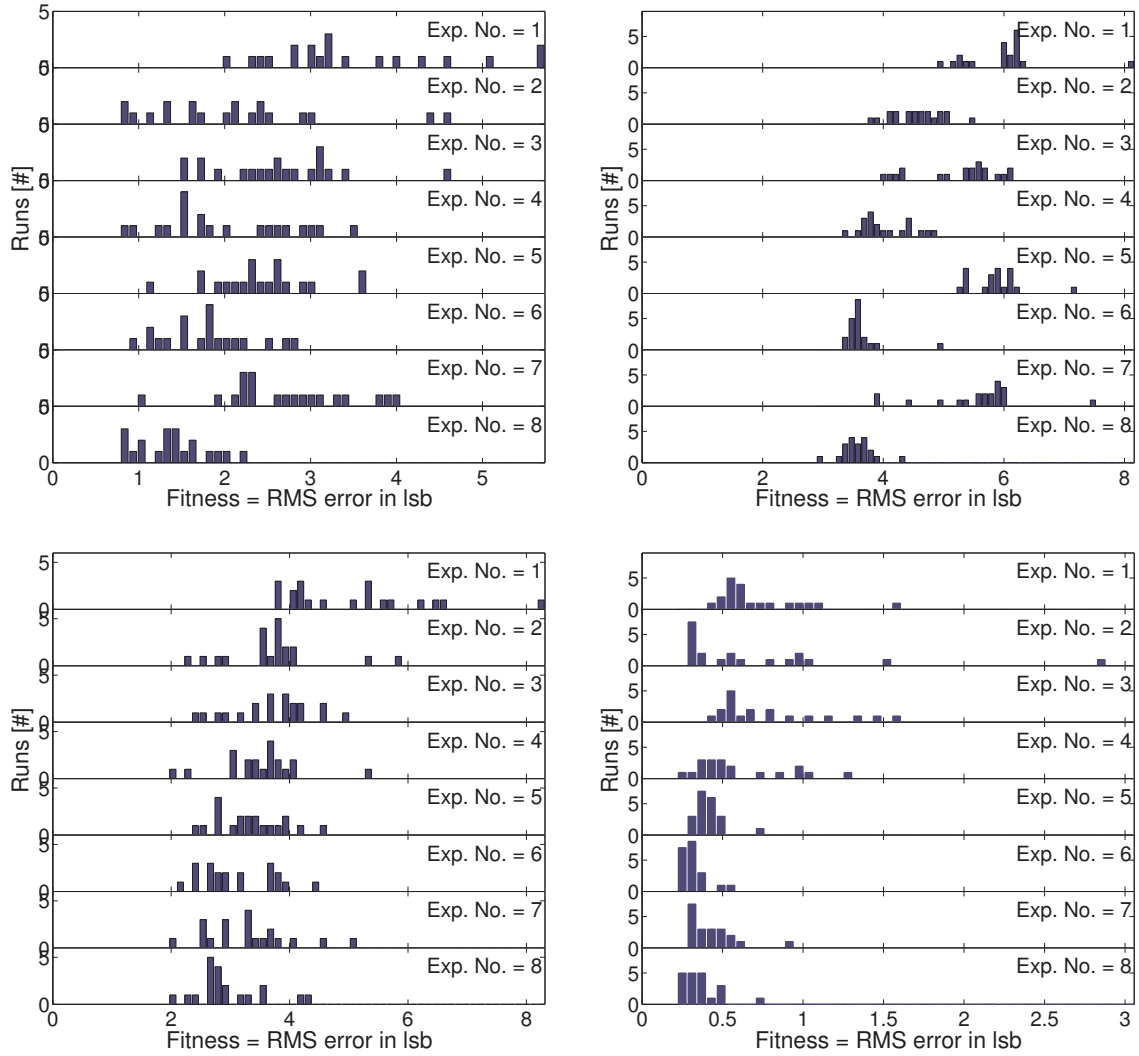


Figure 6.9: Fitness histograms for all experiments for the experiment series INV1, FW4, INV4 and FWB4 (read from top to bottom, left to right). For series FWB4, the histograms for experiments 2 and 3 neglect one outlying run each. The fitness value is taken as the worst of 100 verification tests.

It is worth while noting that neither do the chosen measures conflict nor is the mapping from the fitness defined in 6.4 to those measures bound to preserve the rank of the according solutions. For instance, the triangle in the lower left corner of the scatter plot for series INV1 possesses both, the lowest nonlinearity as well as the lowest maximum of gain error and offset. Nevertheless, as can be seen from Fig. 6.12, it is by no means the best DAC in terms of the fitness criterion used during evolution. This is due to the fact that the original fitness function defined in 6.1 sums up all INL errors without an offset correction, whereas the scatter plots use the maximum offset/gain error and nonlinearity. Thus, it may be beneficial to use another fitness function if the maximum gain error/offset/INL/DNL shall be minimized.

The plots for series FW1 and FWB4 illustrate that for both series a few runs ended up with nonlinearities and offsets/gain errors smaller than 1 lsb. Thus, the according analog to digital converters are verified to possess a true resolution of five bits under all tested conditions. Moreover, in case of series FWB4, the successful circuits are proven to be robust against variations of the analog input voltage level. The comparison of the graphs for both series reveals that most of the evolved solutions

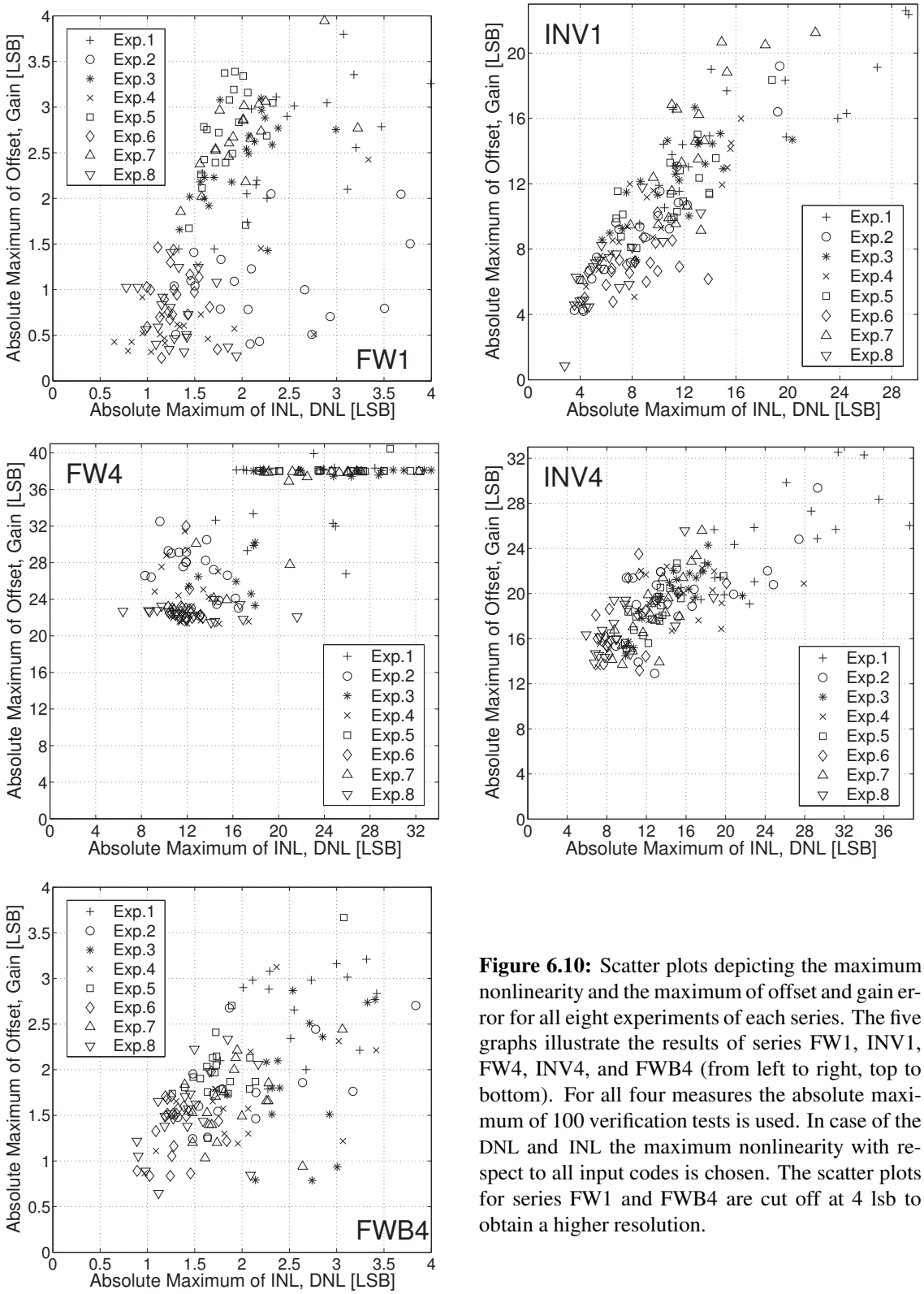


Figure 6.10: Scatter plots depicting the maximum nonlinearity and the maximum of offset and gain error for all eight experiments of each series. The five graphs illustrate the results of series FW1, INV1, FW4, INV4, and FWB4 (from left to right, top to bottom). For all four measures the absolute maximum of 100 verification tests is used. In case of the DNL and INL the maximum nonlinearity with respect to all input codes is chosen. The scatter plots for series FW1 and FWB4 are cut off at 4 lsb to obtain a higher resolution.

Name	Mathematical Expression
Max. of offset and gain error	$\max(\max_{t=1...100}(\mathrm{OS}_t), \max_{t=1...100}(\mathrm{GE}_t))$
Max. of DNL and INL	$\max(\max_{t=1...100}(\mathrm{DNL}_t), \max_{t=1...100}(\mathrm{INL}_t))$

Table 6.6: Mathematical meaning of the measures used in Fig. 6.10.

for series FWB4 are clustered around nonlinearity/offset,gain error value pairs of (1.4/1.6) lsb, while their counterparts of series FW1 are rather accumulated around two different points at (0.7/0.7) and (2/2.5) lsb nonlinearity/offset,gain error. Therefore, both series exhibit similar results on average, but series FW1 produced a larger number of good solutions.

While the distribution of the nonlinearity/offset,gain error values resemble blurred lines in case of series INV1 and INV4, they are more spread out for the series demanding a non-inverted input output mapping. In the latter case, the lower bounds of the ensemble of data points are formed by one line almost parallel to the nonlinearity axis and second line forming an acute angle with the first line. Therefore, the minimum nonlinearity depends on the maximum of offset and gain error but not vice versa. This is due to the fact that the fitness function 6.1, used for the artificial evolution process, penalizes a global offset more severely than narrow spikes causing larger nonlinearities. Hence, evolution falls short of providing the necessary selection pressure for removing large nonlinearities in the presence of large offsets. Moreover, a large fraction of the evolved DAC circuits of series FW4 exhibit a maximum offset/gain error of about 38 lsb featuring different maximum nonlinearities. Since the gain error is taken as the maximum gain error for all four output curves, this behavior can be explained by converters that do not suppress the influence of the input voltage levels at all: The input voltage range stretches from 2 V to 5 V. Thus, the gain differs by a factor of $\frac{3}{5} * 63\,\mathrm{lsb} = 37.8\,\mathrm{lsb}$. As can be seen from the scatter plot for INV4 in Fig. 6.10, evolution does not get stuck in this type of local minimum here. Again, Fig. 6.10 illustrates that the results for series INV4 outperform those for FW4. However, the influence of the input voltage level deteriorates offset and gain more severely than the nonlinearity of the evolved DACs for series FW4 as well as for INV4, which can be explained by a relatively smooth output characteristic, whose slope depends on the input voltages used. Therefore, this is another hint for the EA's inability to abstract the digital information from the analog input voltages.

6.3.3 Best per Series Results

To further illuminate the differences between the five different series of experiments, the output characteristic of the best of series DACs are plotted in Fig. 6.11. Each plot shows the mean voltage characteristic averaged over 100 consecutive measurements. For series FW1 and INV1 the error bars indicate the according standard deviation; this is omitted for the remaining three graphs for clarity, since they contain four curves each. The graphs contain information about the best, mean, and worst fitness value calculated from the 100 verification tests as well as the fitness achieved during evolution. The proximity of these four values proves the underlying circuits to be stable.

While the output characteristic of the best circuit of series FW1 looks almost perfect, the corresponding curve for the best DAC of series INV1 does not form a perfectly straight line. Moreover, both ends resemble the characteristic curve of an inverter. The situation is worse for series FW4 and INV4: The four characteristic curves exhibit large nonlinearities and differ significantly in offset and gain. Finally, the four curves shown for the best individual found for series FWB4 compare well with that belonging to FW1 *and* perfectly coincide.

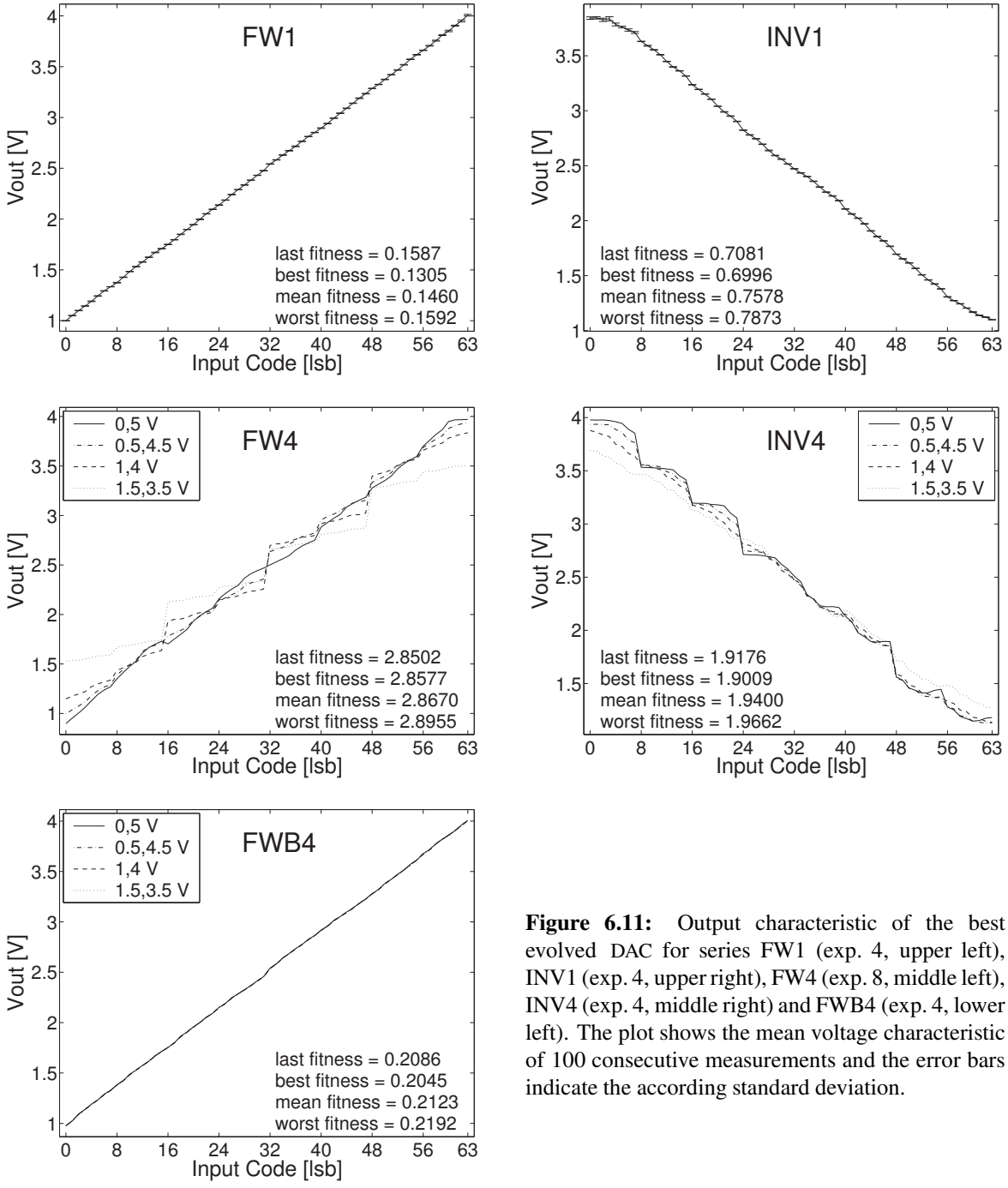


Figure 6.11: Output characteristic of the best evolved DAC for series FW1 (exp. 4, upper left), INV1 (exp. 4, upper right), FW4 (exp. 8, middle left), INV4 (exp. 4, middle right) and FWB4 (exp. 4, lower left). The plot shows the mean voltage characteristic of 100 consecutive measurements and the error bars indicate the according standard deviation.

Finally, the INL and DNL of the best evolved DACs are plotted in Fig. 6.12 for all series of experiments except for FW1 (shown in Fig. 6.5). Similar to Fig. 6.5, DNL and INL are calculated from the averages of 100 verification tests. For clarity, the errors bars denoting the according standard deviation are omitted for all series for which four different output curves are measured. Please note, that all INL curves are zero at both ends of the input codes by definition. Mean gain error as well as mean offset are shown in the inset of the INL graphs. In accordance with the histograms of Fig. 6.6 the mean gain error exceeds the mean offset for all best of series DACs. The counterparts of the

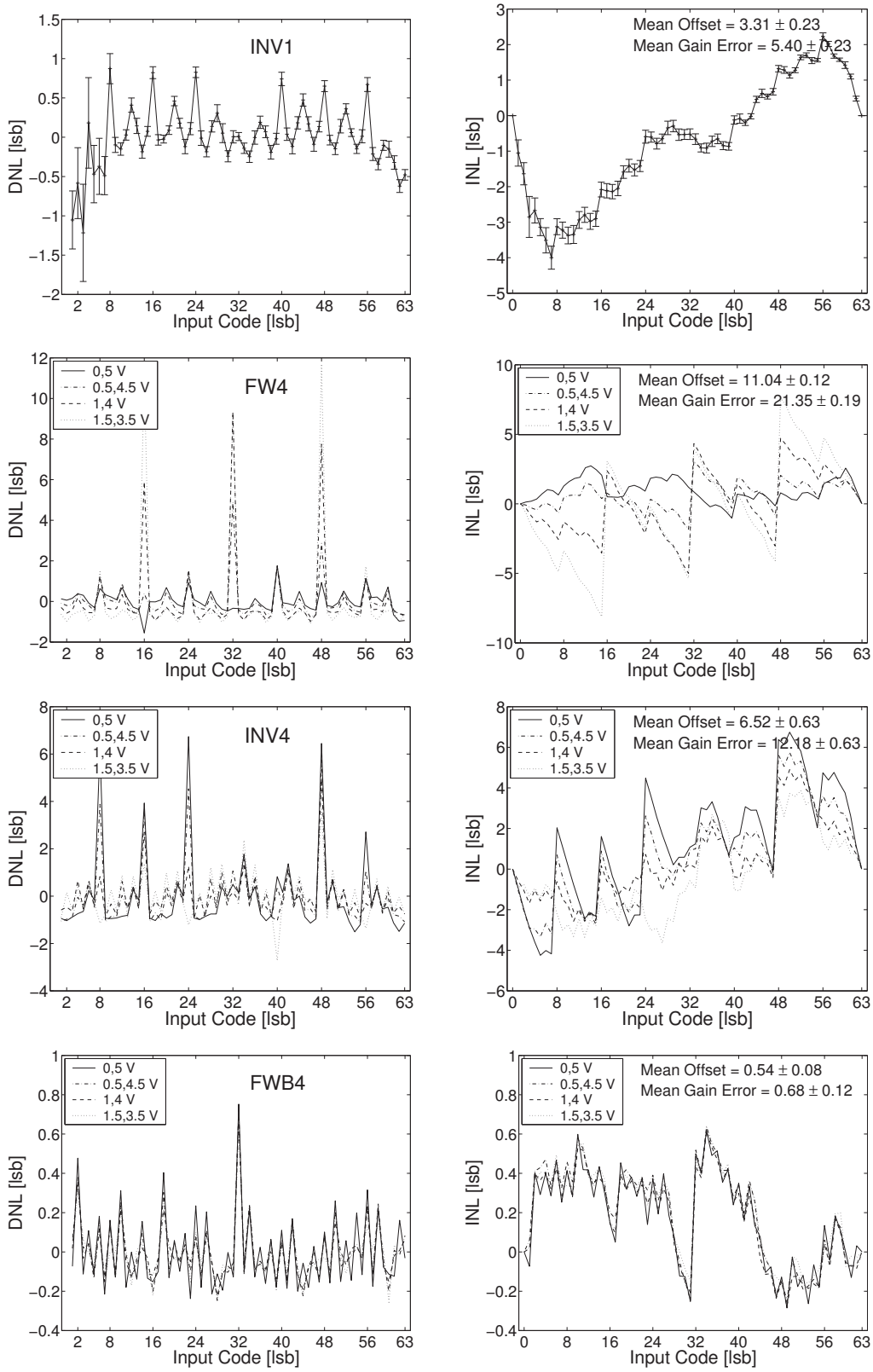


Figure 6.12: Left: DNL, **right:** INL for the best evolved DAC of series INV1 (exp. 4), FW4 (exp. 8), INV4 (exp. 4), and FWB4 (exp. 4) (from top to bottom). The data is averaged over 100 verification measurements, the error bars indicate the according standard deviations.

histograms of Fig. 6.6 for the remaining series – omitted here for brevity – show that this is a general trend.

The four curves in the graphs for series FWB4 lie almost perfectly on top of each other, which confirms the impression from Fig. 6.11. Although the absolute maxima of INL and DNL are slightly higher compared to those obtained for the best solutions of series FWB4, they are of similar quality. The INL of the best DAC found in series INV1 reflects the smoothness of the according output characteristic as well as its bending at both ends of the input code axis. In contrast to the four different nonlinearity curves of series FWB4, those for series FW4 and INV4 differ considerably depending on the input voltage level. For the best DAC of series FW1 smaller input voltage ranges cause larger nonlinearities and vice versa for series INV4. While the maximum nonlinearities of the best solutions of series FW1, INV1 and FWB4 are comparable to their offsets and gain errors, the gain error exceeds the maximum nonlinearities exhibited by the best circuits of series FW4 and INV4 by far, which coincides with the observations from Fig. 6.10.

6.4 Generalizability of the Results of Series FW1 and FWB4

In section 6.3 the evolved circuits were found to respond with reproducible output characteristics in 100 verification tests as well as for their last test during the evolution process. This section answers the question, whether the evolved circuits still work under different conditions, namely a different sampling rate, and on another FPTA chip. Since the DACs obtained for series INV1, FW4, INV4 badly fail to meet the desired specifications, the analysis is restricted to the circuits evolved in series FW1 and FWB4. For all four comparisons, mean and worst fitness, as well as DNL and INL, as well as offset and gain error are compared. More precisely, if one denotes differential and integral nonlinearity of run r, test t and input code j as

$$\mathrm{DNL}_{rtj} \quad j = 1\ldots63\,,\ t = 1\ldots100\,,\ r = 1\ldots20 \tag{6.18}$$

$$\mathrm{INL}_{rtj} \quad j = 1\ldots63\,,\ t = 1\ldots100\,,\ r = 1\ldots20 \tag{6.19}$$

and refers to offset and gain error of run r and test t as

$$\mathrm{OS}_{rt} \quad t = 1\ldots100\,,\ r = 1\ldots20 \tag{6.20}$$

$$\mathrm{GE}_{rt} \quad t = 1\ldots100\,,\ r = 1\ldots20 \tag{6.21}$$

and uses 6.17 for the definition of the fitness of run r and test t, then the used measures utilized for the comparisons are defined in Table 6.7

6.4.1 Verification at a Second Time Scale

As already explained in section 6.1.4, special precautions were taken to prevent the algorithm from abusing temporal correlations in the test pattern: For each fitness test, the input codes were applied in fixed random orders. Since the exploitation of temporal information was observed in prestudies for other experiments as well as in the work reported in [Zeb01], the functionality of the evolved digital-to-analog converters of series FW1 was nevertheless tested on a different time scale. Table 6.3 sums up the larger settling times and lower sample frequencies of the cross-check as well as those used for all other verification tests and during evolution; they are referred to as *slow* and *normal*, respectively. The settling times differ by a factor of 63 for the first and 28 for the last input.

The fitness values achieved under the two different timing conditions are plotted in Fig. 6.13 and Fig. 6.14 for series FW1 and FWB4 respectively: For each experiment of series FW1 (Fig. 6.13) and

Name	Mathematical Expression
Mean fitness	$\underset{r=1...20}{\text{mean}}\left(\underset{t=1...100}{\text{mean}}\left(F_{rt}\right)\right)$
Max. abs. DNL	$\underset{r=1...20}{\text{mean}}\left(\underset{t=1...100}{\text{mean}}\left(\underset{j=1...63}{\max}\left(\lvert \text{DNL}_{rtj} \rvert\right)\right)\right)$
Max. abs. INL	$\underset{r=1...20}{\text{mean}}\left(\underset{t=1...100}{\text{mean}}\left(\underset{j=1...63}{\max}\left(\lvert \text{INL}_{rtj} \rvert\right)\right)\right)$
Max. abs. offset	$\underset{r=1...20}{\text{mean}}\left(\underset{t=1...100}{\text{mean}}\left(\lvert \text{OS}_{rt} \rvert\right)\right)$
Max. abs. gain error	$\underset{r=1...20}{\text{mean}}\left(\underset{t=1...100}{\text{mean}}\left(\lvert \text{GE}_{rt} \rvert\right)\right)$

Table 6.7: Mathematical meaning of the fitness, nonlinearity, offset and gain error values used for the performance comparison under different time scales and on different chips.

FWB4 (Fig. 6.14) the averages defined in Table 6.7 are used to create five different graphs comparing the according results obtained at a sample rate of of 750 kHz – the same rate that was used during evolution – and for the sampling rate of 12.4 kHz used for cross-checking. For series FW1, the resulting curves do not differ significantly, neither for the derived measures nor for the mean fitness (less than a tenth of an lsb !). The same observation applies for the results of experiments 5 to 8 of series FWB4. Thus, the evolved converters of the aforementioned experiments can be said to work well on both time scales and can be expected to do so for the whole frequency range in between. However, in case of the experiments 1 to 4 of series FWB4, the results for both time scales do differ noticeably, in a way that sometimes the performance of the evolved DACs is even improved for the lower sample rate not used during evolution. First, if the evolved circuits were successfully prevented from abusing timing information from the test pattern, the conversion task should be simpler for a more relaxed timing. Second, experiments 2 to 4 contained one run each that suffers from a performance break-down between the end of the artificial evolution process and the verification tests, which may be partly remedied by the lower timing requirements.

In order to appreciate the timing constraints imposed on the evolved circuits, it is instructive to compare the time scale inherent to the FPTA with that of its fabrication process, namely a 0.6 µm CMOS technology. As shown in [Lan01], the gate delay of the programmable transistor cells is larger by a factor of 100 to 150 compared to that of equally dimensioned standard cells implemented in the same process. Thus, if the evolved DACs *could* be translated to this 0.6 µm CMOS process, the sampling rate achieved *would* amount to approximately 100 MHz. In fact, this is unlikely to be achieved, because most probably the GA extensively exploited the parasitic resistances present in the chip. Therefore, simulation of and insight in the evolved circuits are some of the most urgent research avenues to be followed in this project.

6.4.2 Performance on a Second Chip

An important issue in the field of hardware evolution is whether the evolved solutions can be generalized to work under realistic conditions, or if they are bound to the particularities of the very special substrate/model they are evolved on: While simulation based approaches may produce circuits that rely on the special models and parameters of the used simulator, circuits found on one particular die may rely on its exact electrical qualities and fail to work on another die. Therefore the performance of the circuits evolved in the two series FW1 and FWB4 was tested on a second chip. The results are plotted in Fig. 6.15 and 6.16 for series FW1 and FWB4, respectively. Again, the measures defined in Table 6.7 are used to compare the performance of the evolved circuits on both chips. On first sight,

the plots suggest that most circuits of both series work properly on the second die, too, but that their analog performance may be slightly degraded.

As can be seen from the plot on the lower left of Fig. 6.15, the mean fitness of the tested DACs is slightly increased when measured on chip 2. The effect is more distinct for the experiments requiring the smaller output voltage range of 1 to 4 V. Similarly, for these experiments offset and gain error are increased, when they are measured on the second chip. Surprisingly, the opposite is true for the experiments demanding the DACs' outputs to cover the full power supply range. While the integral nonlinearity is also slightly degraded when the designs are migrated from their native die to the second one, the average DNLs do not differ significantly, if all experiments are taken into account. The abovementioned observations can be explained if the tested DACs undergo a global shift of offset and gain when they are tested on the second FPTA chip. This shift may be traced back to two different origins: First, the used transistor cells on the two different dice may not perfectly match.

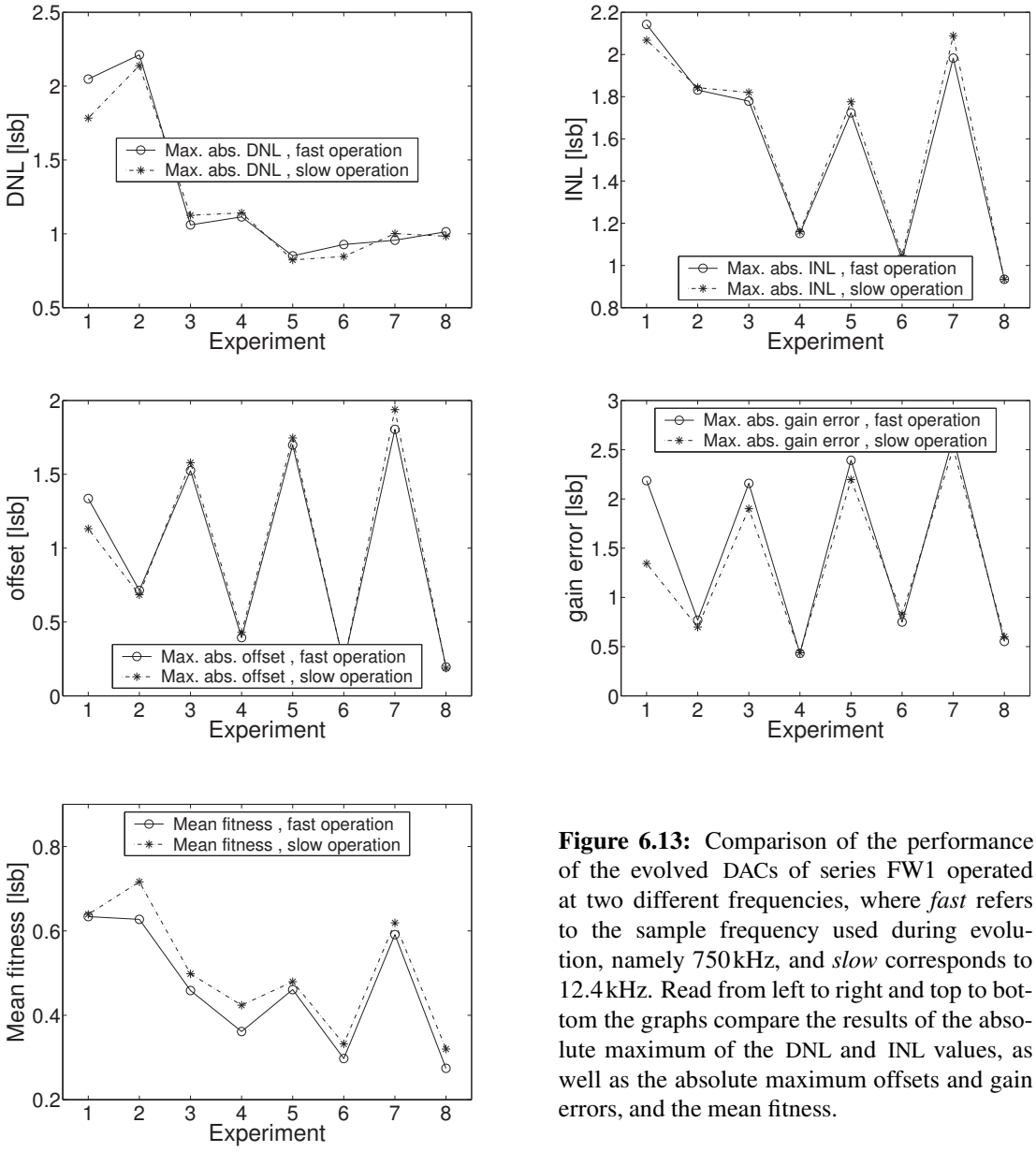


Figure 6.13: Comparison of the performance of the evolved DACs of series FW1 operated at two different frequencies, where *fast* refers to the sample frequency used during evolution, namely 750 kHz, and *slow* corresponds to 12.4 kHz. Read from left to right and top to bottom the graphs compare the results of the absolute maximum of the DNL and INL values, as well as the absolute maximum offsets and gain errors, and the mean fitness.

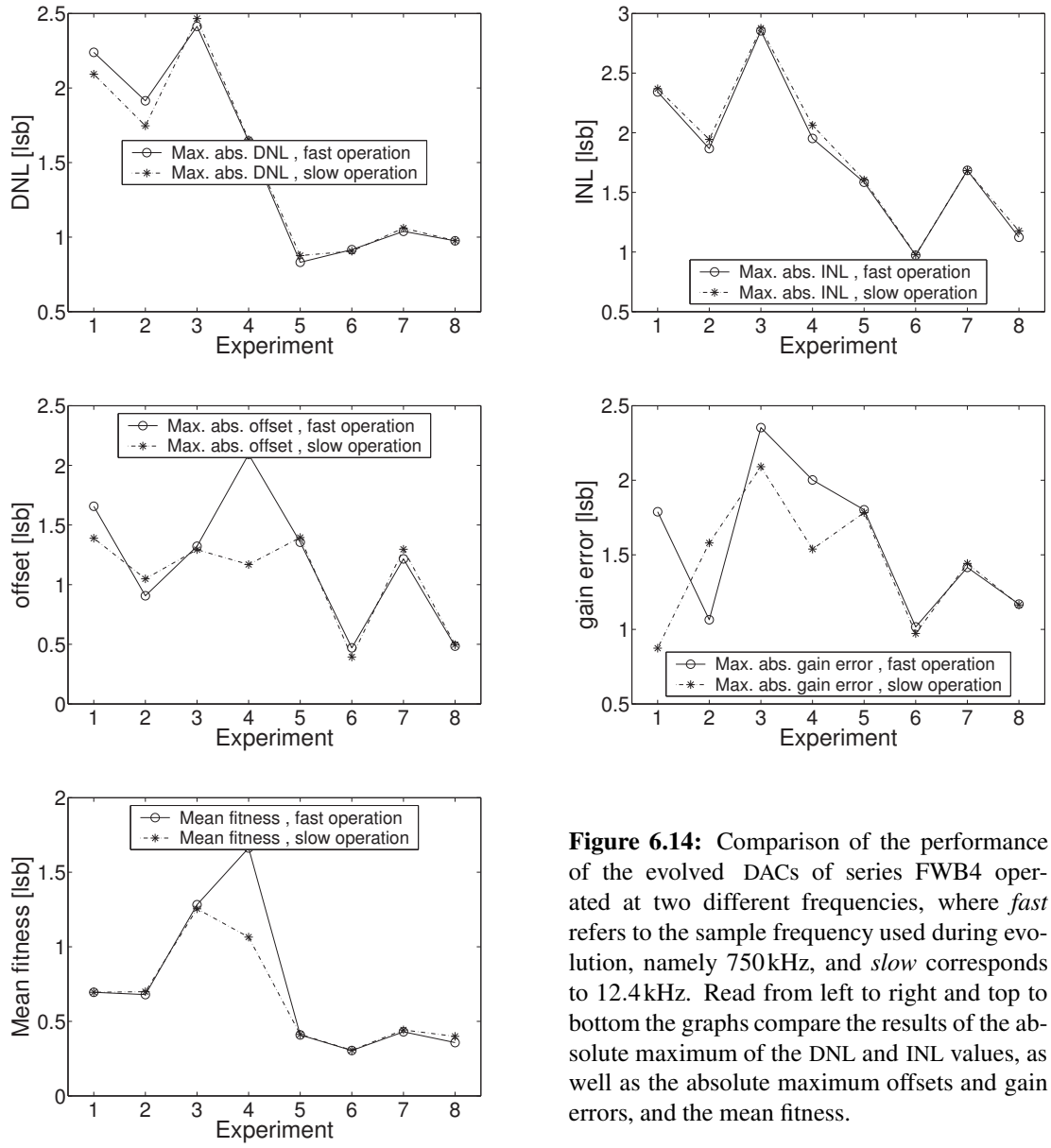


Figure 6.14: Comparison of the performance of the evolved DACs of series FWB4 operated at two different frequencies, where *fast* refers to the sample frequency used during evolution, namely 750 kHz, and *slow* corresponds to 12.4 kHz. Read from left to right and top to bottom the graphs compare the results of the absolute maximum of the DNL and INL values, as well as the absolute maximum offsets and gain errors, and the mean fitness.

This could either be due to die to die variations of global process parameters, as e.g. threshold voltage or transistor gain, or be caused by local variations of the transistor cell characteristics in the form of fixed pattern noise. Since the obtained measures used for the comparison represent averages over 20 different DAC circuits, the effect of the spread of transistor characteristics due the fixed pattern noise should cancel. This hypothesis is confirmed by the close vicinity of each of the two DNL and INL curves. Second, the global shift of offset and gain can be inherent to the system: As explained in sections 4.2.3 and 3.5, the analog input signals as well as the analog output signal are buffered several times after or respectively before their conversion, such that this analog path has to be calibrated. Accordingly, the shift in offset and gain may be due to imperfect calibrations in conjunction with different power supply voltages delivered by the PCI interface of the different computers used.

The mean performance of the DACs of series FWB4 tested on the two different FPTA chips is captured in Fig. 6.16: Apart from experiment 4, which contains the worst outlier (cf. section 6.3.1), the results resemble those obtained for the circuits of series FW1: The averaged fitness values are

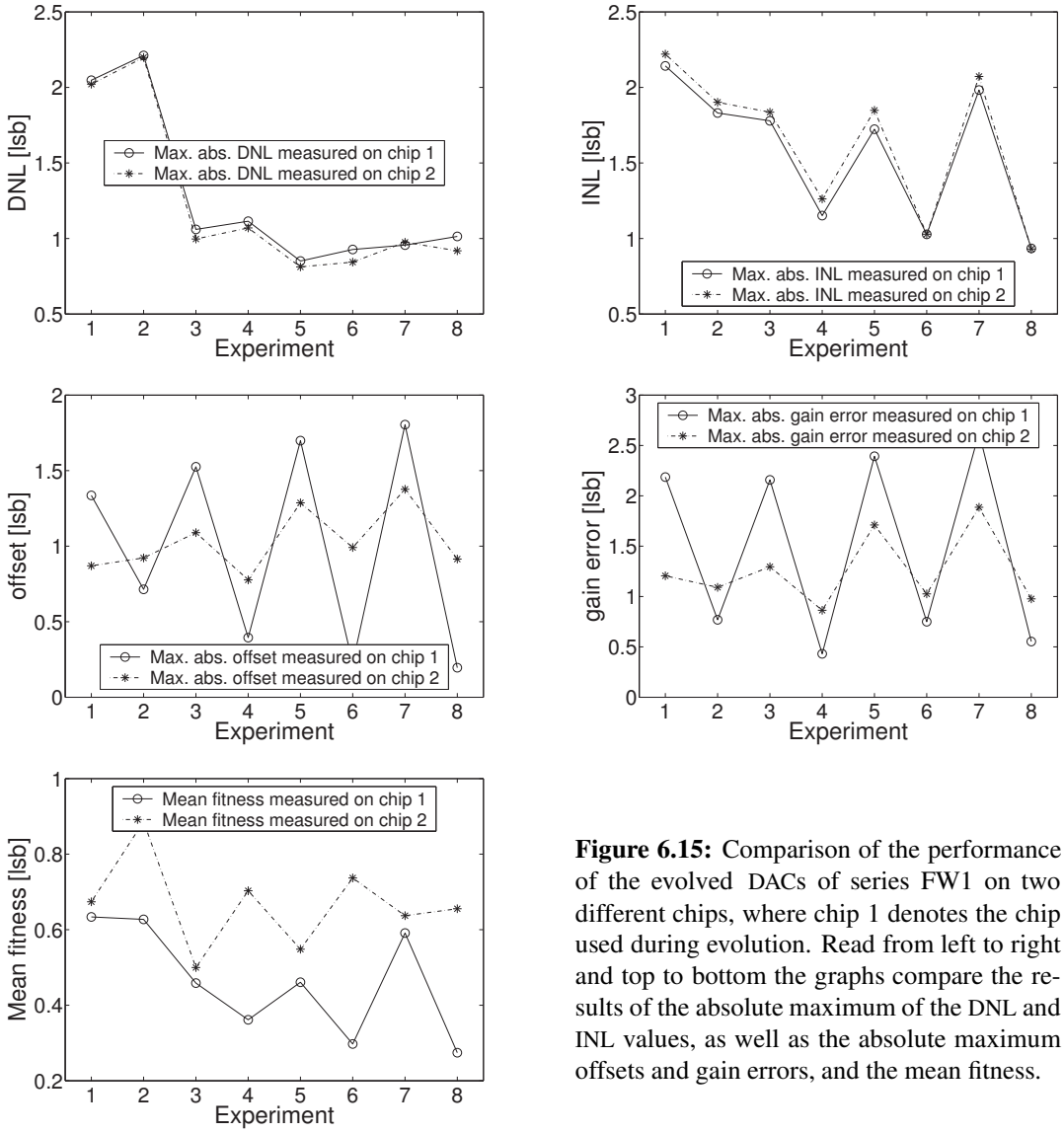


Figure 6.15: Comparison of the performance of the evolved DACs of series FW1 on two different chips, where chip 1 denotes the chip used during evolution. Read from left to right and top to bottom the graphs compare the results of the absolute maximum of the DNL and INL values, as well as the absolute maximum offsets and gain errors, and the mean fitness.

slightly worse, when the converters are tested on the second chip, INL and DNL values are almost identical for both tests. Yet, the differences of the measured offsets and gain errors are smaller than in series FW1. While the offsets for DACs with the larger output range are still smaller and vice versa, this is not true for the gain errors in series FWB4 any more. The fact that the global shift in offset and gain observed for the circuits of FW1 is hardly to be seen for their counterparts of series FWB4 is probably due to the buffers inserted at the inputs of the latter circuits. Thereby, the susceptibility to the nonidealities of the input signals is eliminated; the performance differences on the different dice are reduced to variations in the output voltage processing and the programmable transistors themselves.

6.5 Discussion

First, the proposed experiments demonstrate that the evolution system described in chapters 3 and 4 is capable of processing problems requiring multiple input signals. Second, the analysis of different series of experiments targeted at finding digital-to-analog converters with six bits resolution revealed

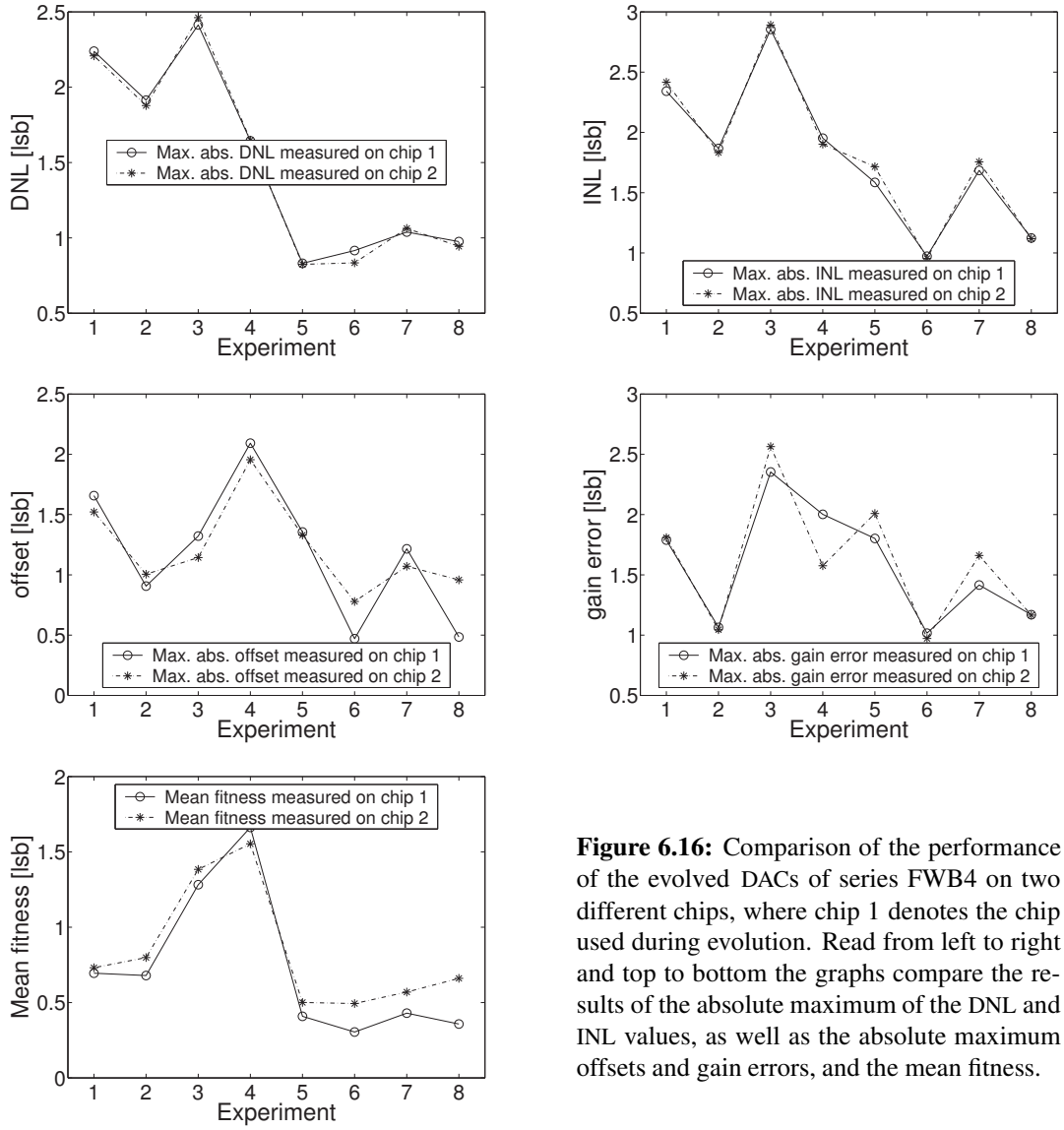


Figure 6.16: Comparison of the performance of the evolved DACs of series FWB4 on two different chips, where chip 1 denotes the chip used during evolution. Read from left to right and top to bottom the graphs compare the results of the absolute maximum of the DNL and INL values, as well as the absolute maximum offsets and gain errors, and the mean fitness.

the following: Choosing an output range of 1 to 4 V in conjunction with a suited geometrical setup allows to evolve DAC circuits with an effective resolution of five bits working at a sample rate of 750 kHz. This raises the question whether it is possible to increase the effective resolution by using more sophisticated fitness functions and optimized algorithms.

Further analysis yields that the evolved DACs fail to provide a digital interface, i.e. strongly rely on the analog voltage level of their inputs. It is demonstrated that this flaw can be remedied by inserting buffers at the circuit's inputs. Future experiments should thus provide a pair of reference voltages to the candidate solution, which define the output voltage range. On one hand, this may aid the EA in abstracting from the analog voltage of the input signals, on the other hand it supports the evolution of multiplying DACs. Moreover, the evolving DACs have not been exposed to a resistive load, which will have to be included to find circuits useful in real world applications. A randomly varied resistive load, however, will further constrain the design space to solutions that do not rely on the analog voltage level of the inputs. Since these additional constraints increase the problem difficulty, they may raise the need for more elaborate methodologies, as for example hierarchical approaches.

The average performance of the evolved circuits gracefully degrades when they are tested on a second chip. Moreover, they are believed to work well in a large range of sampling frequencies, because their performance was only slightly deteriorated, when tested at a much larger time scale. In order to get circuits working well on different dice, they could either be fine tuned to the specific electrical properties of the particular die, or be evolved to work equally well on different dice. The latter goal could be achieved by aggregating the fitness values achieved on different dice during the process of artificial evolution, as e.g. done in [Tho98b].

Chapter 7

Evolution of Filters

Time is an illusion perpetrated by
the manufacturers of space.

GRAFFITI, ANONYMOUS AND
UNKNOWN

In the previous two chapters time as a parameter played only a minor role in that compliant circuits had to reach the desired output voltage within the specified settling time. However, many types of analog circuits are characterized to large extents by their dynamic properties defined in the time and/or frequency domain. Prominent examples are the frequency response of operational amplifiers, which is usually captured by means of its Unity Gain Bandwidth (UGB) and Phase Margin (PM), or the frequency dependency of gain and phase of filter circuits. Accordingly, the purpose of this chapter is to study different means of evaluating the magnitude response in the frequency domain and its effect on the artificial evolution process. The experiments are targeted at the automatic synthesis of MOSFET-only linear filters using the FPTA-based evolution system. More concretely, the evolution experiments are targeted at lowpass filters (LPFs) characterized by different corner frequencies and widths of their respective transition regions as well as highpass filters (HPFs). In case of the latter ones, only the desired width of the transition region is varied.

7.1 Introduction

Linear frequency selective filters[1] are of utmost importance in any kind of signal processing task. Their ability to filter out the particular part of the frequency spectrum interesting to the subsequent

[1] In the remainder of this chapter, linear frequency selective filters will be simply referred to as filters – as is common practice throughout the literature.

processing stage and suppress the rest of the spectrum is widely used in telecommunication as well as data acquisition and control systems. The list of applications utilizing linear filters includes but is not limited to all sorts of parts in hifi systems, speech recognition, cordless and wireless telephony and industrial control systems.

During the last decades, many signal processing tasks were shifted from the continuous-time analog to the digital domain. Given unlimited resources and moderate requirements in terms of the signal bandwidth digital signal processing offers a higher degree of flexibility and precision and thus is often considered the better choice. Nevertheless, if die area and power consumption are an issue, analog-to-digital and digital-to-analog conversion are unnecessary and increase the design complexity and/or further impair the signal at hand, or if pre- or post-processing of the signal is a necessity, analog solutions are favorable or even the only possible choice. The importance of analog filters is reflected in recent publications covering a wide variety of applications and frequency domains: Abidi [Abi96] and Kuhn et al. [Kuh03] describe filter circuits for single chip radio transceivers that use spiral inductors. Baki et al. [Bak03] report a 7^{th} order LPF for hard-disc drive read channel recovery with a tunable corner frequency f_c of 5 to 70 MHz. While Willingham et al. [Wil93] proposed a monolithic anti-aliasing filter with 10-bit linearity at around 8 MHz for HDTV[2], ADSL[3] codecs utilizing analog filters were reported by Phelps et al. [Phe00b] and Siniscalchi et al. [Sin01]. Finally, Murakawa et al. ([Mur03]) describe an intermediate frequency bandpass filter (BPF) with a corner frequency of 455 kHz for cellular phones and Python et al. [Pyt01] report an anti-aliasing lowpass filter with a cutoff frequency of 45 kHz targeted at the DECT[4] standard.

Although "Filter design is one of the very few areas of engineering for which a complete design theory exists, starting from specification and ending with a circuit realization" [Sed91a], these theories neither necessarily lead to optimal filter designs nor are they easy to realize without profound expertise. In other words, the design of high performance analog filters is a difficult and tedious task, which could be facilitated considerably by proper design automation procedures. It is thus an intriguing idea to shortcut the elaborate conventional design process by directly synthesizing analog filter circuits on the transistor level by means of evolutionary techniques. In accordance with the two long-term goals that have been specified in the introduction to this thesis, an improved successor of the proposed FPTA-based evolution system may serve either in finding new, more efficient filter architectures or may be used as a versatile piece of hardware that can be automatically configured (e.g. via artificial evolution) to approximate the desired filter specifications for the task at hand. The final configuration may also take into account or even exploit the nonidealities of the actual die that is used as well as the electrical conditions the chip is to be used in, as for instance the output impedance of the incoming signal or the load impedance the filter circuit is to drive. In order to appreciate the difficulties inherent to such an endeavor as well as its potential impact, the two subsections of this introduction will on one hand briefly summarize the conventional methodology for analog filter design and on the other hand give a comprehensive overview of related work.

7.1.1 Conventional Design of Analog VLSI Filters in a Nutshell

7.1.1.1 LTI Systems

Formally, linear filters can be treated as LTI systems. Following the ideas presented in [Fli91a], an LTI system can be described mathematically by an operation that transforms an input signal $x(t)$ into an output signal $y(t)$ by means of

$$y(t) = \mathscr{T}\{x(t)\} \quad . \tag{7.1}$$

[2] High Definition Television
[3] Asymmetric Digital Subscriber Line
[4] Digital European Cordless Telephone

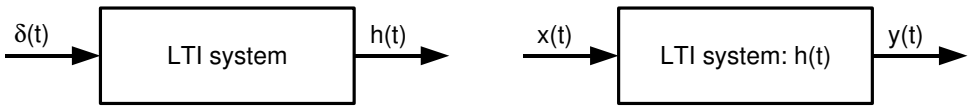


Figure 7.1: LTI system with (a) impulse and (b) arbitrary excitation.

This situation is also depicted in Fig. 7.1. While the linearity of this transformation allows one to write

$$\mathscr{T}\left\{\int_{-\infty}^{\infty} k(\tau)\,x(t,\tau)\,d\tau\right\} = \int_{-\infty}^{\infty} k(\tau)\,\mathscr{T}\{x(t,\tau)\}\,d\tau \ , \tag{7.2a}$$

for any suited real-valued function $k(\tau)$, the time-invariance is defined by the follwing equivalence:

$$y(t) = \mathscr{T}\{x(t)\} \quad \Leftrightarrow \quad y(t-\tau) = \mathscr{T}\{x(t-\tau)\} \quad \forall t,\tau \in \mathbb{R} \ . \tag{7.2b}$$

If one denotes the impulse response of the LTI system by

$$h(t) = \mathscr{T}\{\delta(t)\} \tag{7.3}$$

as is illustrated in Fig. 7.1(a), and expresses the input signal $x(t)$ in terms of Dirac's delta distribution

$$x(t) = \int_{-\infty}^{\infty} x(\tau)\,\delta(t-\tau)\,d\tau \ , \tag{7.4}$$

the following equation holds:

$$y(t) \ = \ \mathscr{T}\{x(t)\} \ \overset{\text{L}}{=} \ \int_{-\infty}^{\infty} x(\tau)\,\mathscr{T}\{\delta(t-\tau)\}\,d\tau \ \overset{\text{TI}}{=} \ \int_{-\infty}^{\infty} x(\tau)\,h(t-\tau)\,d\tau \ = \ x(t)*h(t) \ . \tag{7.5}$$

It is worth noting that once the impulse response $h(t)$ is known, the output of the LTI system can be calculated for any input signal $x(t)$.

Although (7.5) describes the behavior of the LTI system in the time domain, in which any physical measurement is bound to take place, it is more convenient and natural for the analysis of frequency selective filters to describe the system in terms of its frequency response. Adopting the notation of [Lak94c], the output can be written in terms of the imaginary angular frequency $j\omega$[5] by means of a Fourier transform:

$$Y(j\omega) \ = \ \mathscr{F}\{y(t)\} \ = \ \int_{-\infty}^{\infty} y(t)\,e^{-j\omega t}\,dt \ . \tag{7.6}$$

Exchanging $y(t)$ by the convolution expression on the right hand side of (7.5) yields

$$Y(j\omega) \ = \ \mathscr{F}\{y(t)\} \ = \ \mathscr{F}\{x(t)*h(t)\} \ = X(j\omega)H(j\omega) \ , \tag{7.7}$$

where the transfer function $H(j\omega)$ is defined as the Fourier transform of the impulse response $h(t)$:

$$H(j\omega) \ = \ \mathscr{F}\{h(t)\} \ = \ \int_{-\infty}^{\infty} h(t)\,e^{-j\omega t}\,dt \ . \tag{7.8}$$

[5]Following the conventions used in electrical engineering $j \equiv \sqrt{-1}$ is used instead of i.

Thus the transfer function can be calculated from the complex frequency response of the LTI system and the spectral input signal:

$$H(j\omega) = \frac{Y(j\omega)}{X(j\omega)} = M(\omega)\, e^{j\varphi(\omega)} \quad \text{with} \tag{7.9}$$

$$M(\omega) = |H(j\omega)| \quad \text{and} \quad \varphi(\omega) = \arctan\left(\frac{\Im[H(j\omega)]}{\Re[H(j\omega)]}\right) \ . \tag{7.10}$$

Typically, the magnitude response $M(\omega)$ is measured in dB:

$$G(\omega) = 20\log_{10} M(\omega)\,\text{dB} \ . \tag{7.11}$$

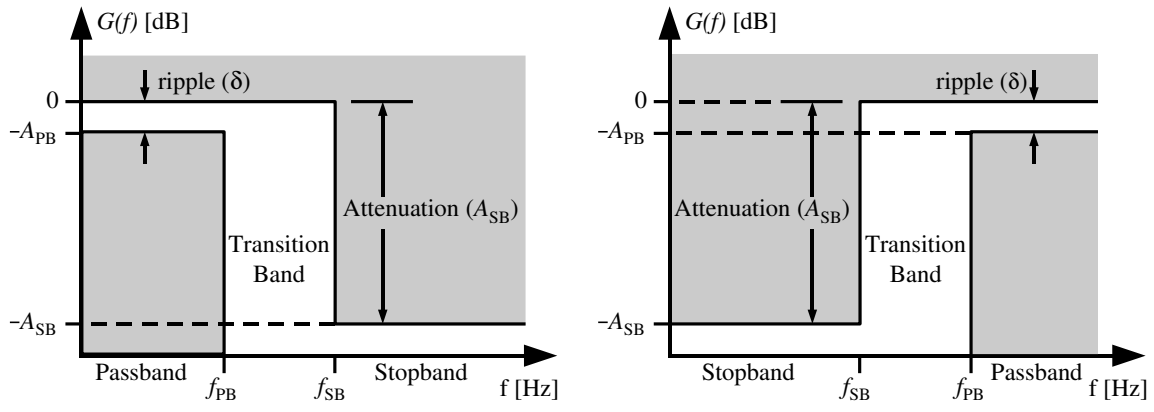
Although the work presented in this chapter is restricted to the automatic synthesis of *gain-shaping* filters, i.e. filters characterized by their magnitude response $G(\omega)$, the phase response $\varphi(\omega)$ or its derivative the group delay

$$\tau(\omega) = \frac{d\varphi(\omega)}{d\omega} \tag{7.12}$$

must be taken into account for many applications (see e.g. [Bak03]).

7.1.1.2 Abstract Filter Specification

The target specifications for the magnitude response of a filter can be formulated by a couple of constants, whose meaning is illustrated in Fig. 7.2 for two types of filters, a lowpass and a highpass filter (nomenclature and illustration follow those found in [Lak94c]). The frequency spectrum is divided into the pass-, transition and stopband by means of f_{PB} and f_{SB}, where the former one is identical to the corner frequency f_c alias $f_{\text{-3dB}}$ if the allowed passband ripple δ amounts to 3 dB. The magnitude response of the filter is only allowed in regions shaded in white, i.e. must not deviate from 0 dB[6] by more than δ in the passband and must be attenuated by at least A_{SB} dB in the stopband.



(a) lowpass filter (b) highpass filter

Figure 7.2: Gain $G(f)$ specifications for (a) lowpass and (b) highpass filters.

[6] Without loss of generality a gain $G(f) = 1$ is assumed for the desired filter.

7.1.1.3 The Transfer Function and its Approximations

The design of an analog filter starts with translating the abstract set of specifications described above into a suited transfer function $H(j\omega)$. For ease of mathematical treatment, the complex angular frequency $j\omega$ is thereby generalized to the complex number $s = \sigma + j\omega$. Throughout the literature (see e.g. [Gre86a], [Gei90b], [Mil87b], [Lak94d], [Fli91b], [Sed91b], [Mat]) the transfer function is assumed to be a rational function[7]

$$H(s) = \frac{\sum_{i=0}^{M} a_i s^i}{s^N + \sum_{i=0}^{N-1} b_i s^i} = \frac{a_M \prod_{i=1}^{M} (s - z_i)}{\prod_{i=1}^{N} (s - p_i)} \quad , \tag{7.13}$$

which is often restricted further by requiring $M \leq N$. In this latter case, the filter order is defined by N. Because of the fundamental theorem of algebra, the enumerator and denominator polynomials can also be expressed by means of their complex roots allowing to write $H(s)$ in terms of the rightmost expression of (7.13). Accordingly, $H(s)$ is fully characterized by the gain factor a_M and its zeros $\{z_1, \ldots, z_M\}$ and poles $\{p_1, \ldots, p_N\}$.

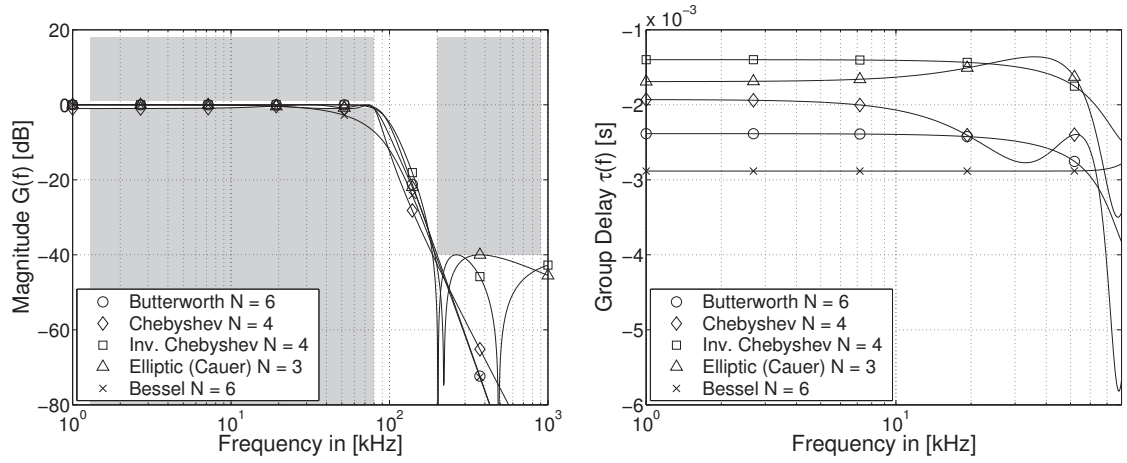
The designer now has to find a tradeoff between different possible design goals of the target filter. Those goals are e.g. attenuation slope steepness, pass- and stopband ripple, phase linearity and filter order. Since the latter one is strongly related to the resources needed to implement the filter, one usually tries to minimize the filter order. Unfortunately it is impossible to find transfer functions that satisfy/optimize all of the above requirements. Fortunately however, the designer can choose from a large variety of tradeoffs between the different design goals. These *filter approximations*, specified e.g. as rules to determine the necessary set of coefficients for the polynomials constituting $H(s)$, are compiled to tables found in the literature. Since the filter coefficients obtained from the literature usually have to be adapted to the corner frequency and filter type of the problem at hand, it is nowadays more convenient to access the *filter approximations* through software tools as for instance MATLAB [Mat]. The most common filter approximations are listed in Table 7.1. Their

Approximation	Passband	Attenuation slope in Trans. Region	Stopband	Group Delay in Passband
Butterworth	**maximally flat**	modest steepness	-20N dB/decade	varying
Chebyshev	equi-ripple	**steep**	**-20N dB/decade**	varying
Inv. Chebyshev	**flat**	**steep**	equi-ripple	varying
Cauer (Elliptic)	equi-ripple	**maximally steep**	equi-ripple	varying
Bessel	flat	gradual slope	-20N dB/decade	**maximally flat**

Table 7.1: The most common filter approximations. The criteria they are optimized for are printed in bold.

respective benefits are a flat magnitude response in the passband (Butterworth), a steep attenuation in the transition band (Chebyshev and Cauer) and a maximally flat group delay (Bessel). To further illustrate the differences of these different filter approximations, their magnitude response and group delay are depicted in Fig. 7.3 for the case of an LPF. The target specifications – $f_{\mathrm{PB}} = 80\,\mathrm{kHz}$,

[7] While Oppenheim et al. (section 4.11.1 in [Opp83]) demonstrate that all systems characterized by linear constant-coefficient differential equations possess a rational transfer function by means of a Fourier or Laplace transform, Fliege [Fli91b] claims that this comprises most systems of technical relevance and Gregorian and Temes [Gre86a] argue that as *RLC* networks are naturally described by such differential equations they in particular are included in the class of LTI systems described by rational transfer functions.



(a) magnitude response (b) gate delay filter

Figure 7.3: (a) magnitude response $G(f)$ and (b) gate delay for the five filter approximations enlisted in Table 7.1. The target specifications, which are indicated by the *forbidden* regions shaded in grey, are similar to those of experiment 5 in section 7.3. The frequency range of the gate delay depicted in (b) is restricted to the passband.

$f_{SB} = 200\,\text{kHz}$, $\delta = 1\,\text{dB}$, $A_{SB} = 40\,\text{dB}$ – are similar to those used for experiment 5 of the series of LPF evolution experiments presented in section 7.3. The allowed region for the magnitude response of Fig. 7.3(a) is colored in white The transfer functions of the Butterworth, Chebyshev and Cauer filter approximations were calculated from the abstract specifications mentioned above using MATLAB [Mat]. Thereby, the lowest possible filter order was used. For the calculation of the transfer function of the Bessel filter, the same order and corner frequency as for the Butterworth filter are used. Note that its magnitude response does not comply with the target specifications at the boundary between pass- and transition band. The differences in the attenuation slopes can be seen from the different filter orders required for the respective approximation characteristic. Due to the very gradual attenuation in the transition band exhibited by Bessel filters, it is often beneficial to use a cascade of gain and phase shaping filters to achieve the desired attenuation *and* phase linearity (cf. [Lak94c]).

7.1.1.4 VLSI Realizations

In order to realize the chosen transfer function as an analog circuit the designer must first decide upon a viable implementation scheme and second find a suitable filter topology. Laker and Sansen [Lak94c] identify three different implementation schemes, namely switched capacitors (SC), active *RC* and G_m-*C* circuits. Recently, a fourth type referred to as *Q*-enhanced *LC* filters has been developed to design analog monolithic RF[8] filters using planar spiral inductors (see e.g. [Kuh03]). Important criteria for the decision upon the best implementation scheme are the frequency band and corner frequencies of interest, the versatility of the realized filter in terms of tunability of corner frequencies, quality factors, filter type and approximation, the precision with which the target filter specifications are likely to be met, the sensitivity to device variations and the set of analog performance goals defined

[8] Radio Frequency

by THD[9], noise figures, DR[10], power consumption and dc offset. Some of the features, merits and disadvantages of SC[11], active RC and G_m-C filters are summarized below (cf. [Lak94e]):

Switched capacitor filters are best suited for general purpose applications in the lower frequency range, as their power consumption increases with frequency and the necessary op amps need a considerably higher UGB as the maximum bandwidth (BW) of the filter because of their oversampling nature. Since resistors and capacitors are composed of capacitors, the filter can be designed such that its transfer function is only dependent on ratios thereof, which can be designed to match well and should be insensitive to temperature variations. Thus, SC filters can precisely match the desired transfer function without tuning and are relatively insensitive to device-to-device variation. Moreover, they are not (significantly) impaired by parasitic impedances of the utilized switches and capacitors. However, the limited UGB of the employed op amps as well as clock feedthrough effects must be considered in the design of SC filters. Finally, because of their sampling, time-discretizing nature, SC filters are infeasible for some applications, as for instance for anti-aliasing or reconstruction filters.

Active RC filters are networks of op amps, capacitors and resistors. For commercially available filter chips that use external resistors to program the filter functionality and define the transfer function they may yield lower THD and noise figures than their SC counterparts, but are less conveniently and flexibly configured (as can be seen from a comparison of the data sheets for [Max96] and [Max02]). For monolithic CMOS implementations the main difficulty is to realize resistors in the range of $10\,\mathrm{k}\Omega$ to at least tens of $\mathrm{M}\Omega$ that are linear over a wide range of input signals. In order to deal with the poor matching of capacitors and resistors, it is desirable to retain a possibility of adjusting the resistor values. This can either be done by laser trimming of resistors laid out in an extra resistive layer (e.g. polysilicon), or by using MOST-Rs[12] as resistors that can be auto-tuned by means of a bias voltage. While the former implementations should achieve higher linearity, the latter ones can be stabilized against environmental changes as e.g. that of temperature. Compared to the SC scheme, active RC realizations are better suited to achieve a higher BW cost-effectively in terms of area and power consumption. Yet, they are more sensitive to component variations and parasitics inherent to the used resistors and capacitors and in case of using MOST-Rs are likely to exhibit higher nonlinearities.

***Gm-C* filters** are usually composed of OTAs and capacitors. Here, the problem of linearity and matching is shifted from the MOST-Rs to the transconductance amplifiers. Similar to the case of active RC implementations, the poor matching of the OTA's transconductances and the respective capacitor values requires auto-tuning schemes to achieve the desired transfer function. This can be achieved by adjusting the transconductance values of the OTAs by changing their bias currents. Although *Gm-C* filters are sensitive to parasitics of the transconductance amplifier as well as to component variations, they seem to be best suited if either high bandwidth, or low power consumption or both is required ([Bak03], [Pyt01], [Mur03]), especially in low voltage designs.

Circuit Topology. Once the implementation scheme is determined, the transfer function $H(s)$ must be translated into a suitable circuit topology. According to Laker and Sansen [Lak94c], [Lak94e] there are two alternatives, namely cascades of biquadratical and bilinear (for uneven filter orders)

[9] Total Harmonic Distortion
[10] Dynamic Range
[11] Switched Capacitor
[12] Resistor implemented by means of MOS Transistors

filter stages and active ladder topologies. A cascade of first and second order stages, if need be, further connected by additional feedback loops, can be used to realize any filter characteristic ([Lak94c]). As first and second order filter stages for all three of the abovementioned implementation schemes are discussed in quite a few text books (see e.g. [Lak94c], [Sed91a], [Mil87c], [Gre86b]), the design of cascaded topologies is considered to be easier and faster, albeit inferior to active ladder filters ([Lak94e]). Active ladder topologies are derived from passive, resistor terminated LC ladder filters by exchanging the inductors with active components. Typically, a passive LC ladder filter must be chosen from or designed with the aid of principles found in the literature first; subsequently, this LC ladder topology has to be translated into an active circuit, which is often achieved using an intermediate type of description referred to as SFG[13]. According to Laker and Sansen [Lak94e] active ladder filters are less sensitive to device variations compared with cascaded designs and therefore recommended for high performance implementations.

7.1.1.5 Overall Methodology

The overall methodology of active filter design may look similar to the following procedure:

Step 1 Define the filter design task in terms of an abstract magnitude and phase response and if applicable further design goals as e.g. quality factor and overshoot in the step response.

Step 2 Determine a suited transfer function in terms of filter order and approximation.

Step 3 Use global constraints, such as target frequency range, power and area consumption as well as analog performance goals such as noise, THD and DR to determine a suited implementation scheme.

Step 4 Choose a topology (e.g. cascade/ladder, type of ladder, etc.).

Step 5 Determine the necessary component values for passive components and derive the necessary specification for active components as e.g. UGB of op amps or transconductance (range) of OTAs.

Step 6 Specify necessary design support functions as for instance a clock generation subsystem and anti-aliasing and signal reconstruction filters in case of an active SC implementation or the auto-tuning system in case of the continuous time filter alternatives.

Step 7 Verify/optimize the overall system performance using abstract active building blocks.

Step 8 Design the low level active components.

Step 9 Verify/optimize system performance in a fully analog simulation including the analog performance goals.

Step 10 Design the layout of the system.

Step 11 Verify the filter performance including the parasitic capacitances extracted from the layout.

Step 12 Verify/optimize yield and analog precision, for instance by means of Monte-Carlo simulations.

[13] Signal Flow Graphs

In addition to this already impressive list of design tasks, it is most likely that design goals are not met in the first approach taken for some of the steps such that decisions taken earlier in the design procedure have to be changed requiring to perform some of the steps more than once. With respect to the complexity of the analog active filter design task it is an intriguing idea to directly exploit the richness of a given analog substrate in an evolutionary loop to sidestep the various levels of optimization necessary in conventional filter design. On the other hand, this complexity also renders the task difficult and the emergence of near optimal filters based on transistor-level artificial evolution rather unlikely.

7.1.2 Related Work

As frequency selective filters are amongst the most important analog circuits and as their design is quite tedious, the spectrum of work related to the automatic synthesis thereof as well as to devices programmable to host a variety of filtering and signal processing circuits is wide. The following comprehensive collection is not meant to be exhaustive, but tries to present an overview pointing out some of the achievements as well as shortcomings of the work reported to date with a twofold purpose: First, it provides the context for the attempts presented in the remainder of this chapter; second, it may be useful to guide future research in the field of automatic filter synthesis. Owing to the variety of scopes, attempts and techniques presented a quantitative comparison is forgone. The publications discussed here are (somewhat arbitrarily) organized in three main categories, namely more traditional contributions from the field of analog design automation, work related to the hardware evolution community using evolutionary techniques, and configurable frequency selective filter chips that can be used to realize different filter types and responses.

7.1.2.1 Design Automation Tools Supporting Analog Filter Design

Owing to the complexity of the task that requires design decisions and optimization at different levels of abstraction, three different approaches are summarized ranging from optimizing the transfer function to the synthesis of SPICE-simulatable netlists from an abstract set of specifications. Yet, none of the approaches can be considered as the ultimate solution to the filter design problem.

In [DV99] Damera-Venkata and Evans report a framework of Mathematica and MATLAB code that can be used to find near optimal transfer functions via SQP[14]. In their examples the authors start from well-known filter approximations as for instance Butterworth filters and find new transfer functions that trade off magnitude, phase and step response as well as the quality factor against each other according to the user's target specifications.

Doboli and Vermuri [Dob03] present a high-level synthesis method to create sized active RC filter networks from SFGs. More precisely, the output is given as a netlist of resistors, capacitors and op amps, together with the respective R and C values and a list of specifications that must be achieved by the op amps. Thereby the ac behavior as well as the total silicon area consumption are optimized. The authors claim that a combination of the proposed high-level synthesis and techniques for transforming abstract filter specifications into a the necessary SFGs as e.g. presented by [Ant95] and of tools for the automatic synthesis and layout of op amps resulted in a fully automated synthesis of analog filters. Nevertheless, the proposed method is limited to the design of active RC filters, forgoes a minimization of component/parasitic sensitivities, necessitates the design of suitable resistors and is also limited by the fact that not all filter structures can be described in terms of SFGs.

Ray et al. [Ray02] propose a more comprehensive tool (BECAS 1.0) for the synthesis of G_m-C filters: Taking abstract filter specifications, transfer functions or passive RLC prototypes as an input,

[14] Sequential Quadratic Programming

the tool synthesizes a netlist consisting of standardized OTAs and sized capacitors and computes the necessary bias currents for the OTAs. The synthesis is achieved by cascading elementary integrator stages that possess a low component sensitivity and minimize the THD of the resulting filter. The synthesis results are verified by means of SPICE simulations. The proposed tool offers the complete synthesis from abstract specifications to a fully specified circuit. Yet, the procedure is restricted to the synthesis of G_m-C filters, in particular to one type of G_m-C filter topologies and to the usage of one special type of transconductance amplifiers. Moreover, the tool does not provide the possibility to optimize for some of the second layer analog design goals as e.g. offset, noise, power and area consumption. In this vein the distance to the optimal solution is not clear and depends largely on the chosen topology and OTA circuit.

7.1.2.2 Hardware Evolution of Filters

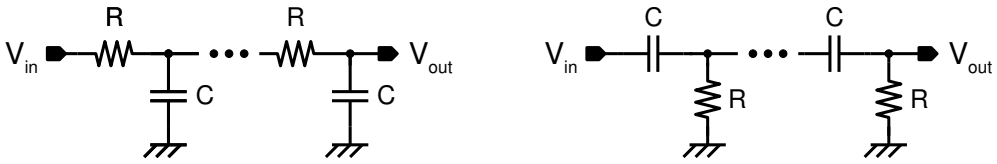
Passive (RLC) Filters. Although passive RLC networks are not exactly active filters, they are of interest in that they can be used for deriving active ladder topologies. Moreover, to date, the most impressive results in hardware evolution of both, topology and component sizes for analog filters are reported for RLC filters. This is due to the fact that the task of finding an RLC network to satisfy a given filter task in an ideal simulation is easier than synthesizing an active circuit for the same task for the following reasons: First, resistors, capacitors and inductors are ideally suited for frequency selective filtering applications in that they are (in the simulation model) perfectly linear and form exactly *the* small set of components to create any passive filters of. Second, their linearity implies that candidate circuits can be evaluated quickly and precisely by means of an ac-analysis and/or the computation of an analytical expression. Probably the first successful artificial evolutions of RLC filters are reported by Koza et al. (see e.g. [Koz96c], [Koz96d]) [Koz99b], using genetic programming. Further attempts based on GAs differing in their circuit representations are found in [Loh99] and [Zeb98b], whereas Grimbleby ([Gri99], [Gri00]) employs a GA to generate new circuit topologies in conjunction with a quasi-Newton optimization algorithm to determine the component values. Using this hybrid approach, Grimbleby was able to find a solution to an asymmetric BPF problem stated by Nielsen [Nie95] that is composed of less components than the solution reported by Koza et al. in [Koz99c], [Koz96c] and [Koz96d] and even requires one component less than the human design proposed by Nielsen.

Despite their success, all of the abovementioned approaches ignore the following problems inherent to the design of real discrete RLC filters: First, real RLC devices are restricted to a discrete set of values. Second, for the design of real RLC filters, the parasitic capacitances (R,L) and resistances (C,L) must also be taken into account. Finally, a realistic optimization must also include the device-dependent spread and the different prices for different passive components to find cost and performance optimal designs. The restriction to a discrete set of component values is taken into account by Zebulum et al. [Zeb00a] and Horrocks and Khalifa [Hor94]. In [Hor96] they extend this concept to include parasitics ascribed to the different types of components. Yet, in both papers Horrocks and Khalifa restrict the task to finding a set of suited component values for a given topology, which severely facilitates the task.

Extrinsic Hardware Evolution of Active Filters. Opposite to the evolution of passive RLC filters, none of the experiments targeted at the synthesis of topology and sizing of active filters that are presented below achieves a transfer characteristic of industrial strength. Nevertheless, the evolution of useful transfer characteristics for band amplifiers for AM radio receiver application are reported in [Zeb99], [Zeb00b] and [Key00a]. The first publication by Zebulum et al. differs from the latter ones from the group of Adrian Stoica in that the algorithm is allowed to use RLC components and BJTs and

in that a linear unconstrained circuit representation is chosen. Conversely, the experiments presented in [Zeb00b] and [Key00a] provide one to four cells of *their* FPTA-0 chip (cf.section 3.11) plus one or more capacitors and feedback paths to the algorithm, which thusly has to deal with a constrained genotype representation. Similar experiments aiming at the artificial evolution of BPFs are published in [Sto01d] and [Sto02a], again by Stoica et al.. All of the abovementioned experiments solely base their fitness evaluation on an ac-analysis.

Extrinsic Hardware Evolution of MOSFET-only Active Filters. The experiments published in [Bot03] and [Vie04] relate even more closely to those presented in the remainder of this chapter, in that they also restrict the set of possible components to CMOS transistors with in- and output resistors being the exception. In the realm of analog filter design , this adds the principal difficulty of realizing capacitors by CMOS transistors. As explained in section 1.2.2, a sufficiently large capacity can be realized between gate and channel of the transistor if the voltage difference between gate and channel keeps the transistor in strong inversion. This can be achieved by choosing a circuit topology that requires only grounded capacitors (see e.g. [Hua97] or [Pav00]) or by explicitly biasing the voltages across the MOS capacitor as for instance reported in [Lin00a]. Nevertheless, as the abovementioned solutions require complex implementation schemes (active-RC and G_m-C), the restriction to MOSFET-only circuits adds a severe complication to the task. For instance, the simplest way for an EA to implement a lowpass behavior should be to use the resistive and capacitive features of the available transistors to form an n^{th} order passive RC filter as depicted in Fig. 7.4(a). In case of the highpass



(a) Lowpass filter (b) Highpass filter

Figure 7.4: Schematic of an n^{th} order passive RC LPF (a) and HPF (b).

filter equivalent illustrated in Fig. 7.4(b), this cannot be realized as easily since the transistor's gate capacity cannot be used without proper biasing.

Botelho et al. [Bot03] report the artificial evolution of an LPF and an HPF with a corner frequency of 10 and 100 kHz and a rolloff of about -40 and -20 dB per decade, respectively. Vieira et al. [Vie04] present similar experiments focusing on the artificial evolution of LPFs with different passband gains of 0 and 40 dB respectively. A filter circuit perfectly matching the 40 dB of passband gain could only be evolved by including a macromodeled 2-terminal amplifier, which lead to an LPF with a rolloff of 20 dB per decade. Yet, LPFs with passband gains of 22 and 37 dB are reported to have been found without any predefined amplifier. Botelho et al. as well as Vieira et al. verify their LPF circuits to be linear in a range of almost 2 V by means of a dc-analysis. On one hand, the results are remarkable, as they require active amplification in case of the LPFs and to overcome the floating capacitor problem. On the other hand, all the evolved *active* filters cited within this section share one fundamental problem: In order to ensure their linearity over a finite range of input signals, a transient analysis is mandatory. An ac-analysis linearizes the given circuit at an operation point established by means of a dc-analysis, i.e. any nonlinearities are discarded and the input signal is assumed to be of infinitesimal amplitude. The additional dc-analysis used to ensure the linearity of the evolved LPFs published in [Bot03] and [Vie04] ignores any capacitive coupling and thus is likely not to hold for any realistic time scale within the passband.

Intrinsic Hardware Evolution of Active Filters. Most of the work published on intrinsic hardware evolution (HWE) of analog filters is restricted to parameter optimization. In their groundbreaking work Murakawa et al. ([Mur98] and [Mur03]) successfully replace the common laser trimming method typically used for calibrating high-performance IF[15] BPFs by a GA-based calibration procedure. Thereby, a total of 39 bias currents are fine-tuned with a precision of 3 bits to compensate the device variations inherent to the production process. The calibration procedure has to be done once for every produced die, takes a few seconds and uses an LSI[16]-tester. In comparison, the intrinsic HWE experiments reported by Sheehan et al. ([Flo00], [She02]) and Greenwood et al. [Gre04] are less ambitious; both groups use commercial FPAAs as their hardware platform to evolve a given filter behavior. Thereby, the role of the EA is restricted to controlling a few component values, like resistances (Sheehan) or two gain factors and a capacitance (Greenwood). More precisely, Sheehan et al. use a 2nd order Tow-Thomas architecture implemented on a TRAC chip (see [Zet99]) to find a BPF characterized by its transient response. Greenwood et al. on the other hand, evolve the parameterization for a compensator that aims at restoring the original frequency response of a third order LPF after an emulated degradation by a fault or aging process. Compensator as well as the actual filter are implemented on a ispPAC10 chip (see [Lat00]). The fitness is evaluated by taking the magnitude response at five different frequencies; the necessary test is supported by a signal generator and a spectrum analyzer connected to the host PC via a GPIB[17] interface and takes around 5 sec.

To date[18], experiments on the intrinsic evolution of analog filters including topology synthesis have only been published by Stoica et al.. The experiments reported in [Key04], [Sto04] and [Zeb04] aim at the intrinsic synthesis of LPFs and HPFs (the latter ones are only reported in [Zeb04]) on 10 cells of the group's FPTA-2 chip. Although not very carefully documented, the reader can conclude that the fitness evaluation is based on the deviation of the magnitude response from the target specification, where the magnitude response is measured at 1 and 10 kHz. All frequency components above 1 kHz or below 10 kHz are to be suppressed for the LPF and HPF task, respectively. The evolved filters are reported to possess a maximum gain of 3.6 dB and -3 dB and a rolloff of about ∓ 14 dB per decade for the LPF and HPF, respectively. It is interesting to note that the frequency responses in terms of rolloff and attenuation of the extrinsically evolved BPFs reported by Stoica's group look much more promising than those of the groups's intrinsically evolved low- and highpass filters.

7.1.2.3 FPAAs Dedicated to Analog Filter tasks

The related work section closes with a small compilation of chips that can be used as analog frequency-selective filters for a variety of specifications. This set, summarized in Table 7.2, can be divided into three groups: Off-the-shelf programmable general purpose filters, commercially available FPAAs for signal processing, and research prototypes of configurable filters and FPAAs published in the literature.

The first two examples from the group of off-the-shelf programmable filters are among those offered on the website of Maxim[19] [Max] that feature the largest corner frequencies. While the MAX274 [Max96], an active RC filter, needs to be configured by four external devices, the SC-based MAX262 [Max02] can be configured more conveniently by a microprocessor. As expected, the SC based filter is not only more conveniently programmed, but also more versatile in terms of realizable filter types and characteristics. However, the active RC filter on the other hand yields the better noise

[15] Intermediate Frequency
[16] Large-Scale Integration
[17] General Purpose Interface Bus
[18] and to the author's knowledge.
[19] Maxim Integrated Products, Inc. is one of the leading vendors of analog devices.

Group/ Part	Filter Types	Charac- teristic	Max. Order	Max. f_0[Hz]	Max. BW	Operation Principle	Remark	Reference
Maxim MAX274	LPF, BPF[a]	various[b]	8	100–150k	10 MHz	active RC	contin. time	[Max96]
Maxim MAX262	All	various[c]	4 per chip	1–140k[d]	1 MHz	Switched Capacitor	can be cascaded	[Max02]
Anadigm AN221E04[e]	All	various[f]	8 per chip	$\leq$ 2M[g]	2 MHz	Switched Capacitor	chips can be cascaded	[AND04], [Ana], [AN203]
Pankiewicz et al.	All[h]	various[h]	> 10	several k–M	20 MHz	G_m-C	40 CABs[i]	[Pan02]
Pavan et al.	LPF	Butter- worth	4	60 M – 350 M[j]	???[j]	G_m-C	cont.& time	[Pav00]

[a] BSF requires an additional external op amp.
[b] programmed by 16 external resistors.
[c] For each second order section, Q and f_0 can be selected with a resolution of 7 and 6 bits respectively.
[d] f_0 is mainly defined by the external clock; fine tuning is possible by choosing the f_{clk}/f_0 ratio.
[e] The ANx2xE0x series available from Anadigm Inc. is not limited to conventional filtering tasks, but can be used for a wide variety of analog signal processing applications.
[f] The software AnalogDesigner2 allows to configure e.g. Butterworth, Chebyshev, inverse Chebyshev, Elliptic and Bessel filters.
[g] The corner frequency f_0 can be varied from $f_{clk}/500 \ldots f_{clk}/10$, with $f_{clk} \leq 20$MHz.
[h] The actual limits and precision of approximations to the transfer function and filter types are not discussed in [Pan02]. However from the structure of the chip [i], a wide variety of filter types and approximations is conceivable.
[i] A CAB consists of 1 programmable OTA (5 bits), one programmable capacitor (5 bits) and 12 switches.
[j] The authors identify bandwidth with the f_{-3dB} of their LPF. They present measurements up to 500 MHz; Yet, any system tends to exhibit a lowpass-like behavior for frequencies exceeding its bandwidth. It is not clear, if the concept yielded the same bandwidths for other filter types as e.g. HPFs.

Table 7.2: A small selection of chips suited for filter applications.

and THD performance as well as a higher BW and is free of the artifacts inherent to discrete time architectures.

The Anadigm AN221E04 [AN203] is among the best (in terms of analog performance and variety of realizable circuits) commercially offered FPAAs dedicated to signal processing tasks. It is also based on SC technology and like the MAX262 can be cascaded to form higher order filters. In comparison to the MAX262 the AN221E04 seems to be more versatile, because apart from filtering it also lends itself to a variety of signal conditioning and processing tasks. Maxim and Anadigm offer software tools to support the designer in finding the best configuration for the filter task at hand. The software offered for the Anadigm AN22XE0X series [AND04] is probably most convenient and allows to configure the chip directly in the field.

Finally, Pankiewicz et al. [Pan02] and Pavan et al. [Pav00] propose an FPAA for filter applications and an example implementation for a widely programmable high-frequency filter concept. Both implementations are targeted at high bandwidths and thus are based on the G_m-C principle. The FPAA reported in [Pan02] can in principle be configured to host any filter type and approximation. It consists of 40 CABs[20], which contain a programmable OTA, capacitor and 12 switches. The authors claim that the BW of 20 MHz could be considerably increased by migrating the design to a state of the art submicron process technology. Another G_m-C-based FPAA concept featuring techniques and

[20] Configurable Analog Blocks

circuits similar to those proposed in [Pav00] is proposed by Becker et al. [Bec04]. The prospective FPAA is dedicated to continuous-time filter applications with BWs up to 200 MHz. Yet, this being a concept at the time of writing, a verification in terms of concrete simulations or measurements of a first prototype have to be awaited. The 4th order Butterworth LPF reported by Pavan et al. [Pav00] on the other hand offers even higher BWs and can be configured to corner frequencies between 60 and 350 MHz. Yet, filter type and characteristic are fixed.

In summary, field evolvable hardware chips for filter applications featuring BWs of around 1 MHz are not desperately needed, as commercially available FPAAs can be easily configured to a wide variety of filter types and approximations with a precision and analog performance that seems to be out of reach on medium time scales for FPTA-based solutions. For filter applications requiring a BW of hundreds of Mega-Hertz, on the other hand, dedicated architectures based on a suited set of building blocks seem to be unavoidable. The related work summarized above suggests that G_m-C based CABs are to date the best technology for field programmable/evolvable high-frequency filter chips. According to the detrimental parasitic effects introduced by analog switches it should be beneficial to minimize the number of switches in such an FPAA design. This may be achieved by realizing large parts of the configurability through programmable bias currents of generalized OTAs.

7.2 Evaluation of the Magnitude Response

One of the central issues of this chapter is to establish a method of evaluating the magnitude response for a candidate filter circuit. Throughout all *extrinsic* HWE experiments cited in the above section, the magnitude response has been realized by means of an ac-analysis, whose possible shortcomings have already been discussed above. In case of *intrinsic* HWE different test methods are used: While Sheehan [She02] directly extracts two parameters from the step response that sufficiently characterize his second order BPF, Greenwood et al. [Gre04] employ a sine wave generator together with a spectrum analyzer to measure the attenuation for five different frequencies. The former approach is not only inconvenient in terms of the statement of the problem, but also infeasible for unknown filter orders. The latter approach, that is taking the Fourier transform of the filter's response to a sinusoidal input signal to find the magnitude response for the according frequency and that is not using external devices connected via the prohibitively slow GPIB interface, bears some similarities to the third method discussed below. Stoica et al. [Zeb04] also attain the magnitude response by using sine waves of different frequency as an input; yet, they restrict their number to two and do not detail how the magnitude is calculated from the filter's output.

As it is not entirely clear which way of determining the magnitude response yielded the highest analog precision or was suited best to support the success of the artificial evolution process, a total of three different methods are proposed in the remainder of this section and their influence on the results achieved for evolving different LPFs is studied in section 7.3. While the first approach utilizes the Fourier transform of a step response, the second and third one excite the input with sine waves of different frequencies. The according output is then either integrated over an integer number of periods or Fourier transformed to determine the magnitude response.

7.2.1 M1: Transfer Function from the Step Response

Within the theoretical framework of LTI systems sketched in section 7.1.1.1, the transfer function $H(j\omega)$ can be calculated from the impulse response defined in (7.3) by means of the Fourier transform of (7.8). However, since delta pulses are difficult to produce in reality, one has to resort to the step response

$$g(t) = \mathcal{T}\{\theta(t)\} \tag{7.14}$$

to calculate the impulse response $h(t)$:

$$h(t) \;=\; \mathcal{T}\{\delta(t)\} \;=\; \mathcal{T}\left\{\frac{d}{dt}\theta(t)\right\} \;=\; \frac{d}{dt}\mathcal{T}\{\theta(t)\} \;=\; \frac{d}{dt}g(t) \;\; . \tag{7.15}$$

Here, $\theta(t)$ denotes the Heaviside step function, whose relation to the delta distribution can be described by:

$$\delta(t) \;=\; \frac{d}{dt}\theta(t) \;\; . \tag{7.16}$$

As both, the input stimulus as well as the measured output voltage are sampled for N times in equidistant time intervals of length T, one needs to find the discrete-time analogon to (7.15). For the transition from continuous time to discrete time systems, the following transformations for time t, frequency f, Heaviside function $\theta(t)$ and delta distribution $\delta(t)$ are used:

$$t \to t_n = nT \to n \quad \text{and} \quad f \to f_k = \frac{k}{NT} \;\text{ with }\; \omega \to \omega_k = 2\pi f_k \tag{7.17a}$$

$$\theta(t) \to \theta(n) = \begin{cases} 1 & \text{if } n \geq 0 \\ 0 & \text{if } n < 0 \end{cases} \tag{7.17b}$$

$$\delta(t-\tau) \to \delta(n-m) \equiv \delta_{nm} = \begin{cases} 1 & \text{if } n = m \\ 0 & \text{if } n \neq m \end{cases} \;\; . \tag{7.17c}$$

Thus, again following the arguments of Fliege [Fli91c], the discrete version of the step response $g(n)$ can be written as

$$g(n) \;=\; \theta(n) * h(n) \;=\; \sum_{k=-\infty}^{\infty} h(k)\,\theta(n-k) \;=\; \sum_{k=-\infty}^{n} h(k) \;\; , \tag{7.18}$$

which is derived from the discrete analogon to (7.5). In order to calculate the desired impulse response $h(n)$, we can rewrite (7.18) as

$$h(n) \;=\; g(n) - g(n-1) \;\; . \tag{7.19}$$

In order to obtain the desired (discrete) transfer function $H_D(f_k)$, the impulse response $h(n)$ must be transformed into frequency space. For the discrete and finite set of N output voltages $\{X_0,\ldots,X_{N-1}\}$ the Fourier transform of (7.8) thus has to be replaced by the DFT[21] defined in appendix D.1 (cf. [Fli91d]):

$$H_D(f_k) \equiv H(j\omega_k) \;=\; \mathcal{F}_D(h(n)) \;=\; \sum_{n=0}^{N-1} h(n)\, e^{-2\pi jk\frac{n}{N}} \quad \text{for} \quad k = 0,1,\ldots,N-1 \;\; . \tag{7.20}$$

The magnitude response in dB, $M_D(f_k)$, is then obtained from (7.20) by:

$$M_D(f_k) = |H_D(f_k)| \quad \text{for} \quad k = \begin{cases} 1,2,\ldots,\frac{N}{2} & N \text{ even} \\ 1,2,\ldots,\frac{N-1}{2} & N \text{ uneven} \end{cases} \tag{7.21a}$$

$$G_D(f_k) \;=\; 20\log_{10} M_D(f_k)\,\text{dB} \;\; . \tag{7.21b}$$

In accordance with the *Nyquist Theorem*, the magnitude response is sampled at the frequencies $f_k = 0, 1/(NT), \ldots, 1/(2T)$ (if N is even), where $M_D(f_0)$ denotes the dc component of the magnitude response.

[21] Discrete Fourier Transform

Implementation. Since pre-studies revealed that the magnitude response obtained from the step response is quite sensitive to high frequency noise, the software DarkGAQT allows to average $h(n)$ for M times before the Fourier transform is applied. Throughout all the experiments presented in the remainder of this chapter M was set to 3. In order to allow the circuit to settle to its dc output voltage, the step in the input voltage is applied approximately in the middle of the recorded sampling time and all input samples before the input step are overwritten by zeros. A total of $3N$ zeros are appended to the resulting vector of output voltages before it is transformed by means of (7.20), which is achieved by means of the FFTW[22] package [Fri03]. Here, this technique called *zero-padding* effectively increases the density of frequency samples f_k by a factor of 4 (see e.g. [But00] section 4.6 for an introduction to zero-padding). Only the first half of the resulting output vector of $2N$ frequency components f_k satisfying $f_k \leq f_{\text{Nyquist}} \equiv 2/T$ are used to visualize $G_D(f_k)$ and calculate the according fitness contribution. Thus, the number of input samples equals that of the number of outputs and the output signal is oversampled with a factor of 2 to suppress high-frequency noise and stay away from possible artefacts close to the Nyquist frequency f_{Nyquist}.

The fitness belonging to the magnitude response obtained from the step response is calculated by

$$F_{\text{step}} \;=\; P_{\text{OS}} \quad + \sum_{\substack{k=0 \\ f_k \notin \text{ transition band}}}^{N-1} \left(G_D(f_k) - G_{\text{tar}}(k)\right)^2 w(k) \quad \text{with} \quad w(k) = \frac{1}{k+1} \;\; . \tag{7.22}$$

The weight factors $w(k)$ are hard-coded and chosen to counterbalance the increasing density of sample points relative to the absolute frequency f_k. In order to force the prospective filter circuits to settle their outputs to the input voltage in the dc case, (7.22) contains an additional penalty term

$$P_{\text{OS}} \;=\; 10 \sum_{n=N_{\text{step}}-50}^{N_{\text{step}}-1} \left(V_{\text{out}}(n) - V_{\text{in}}(n)\right)^2 \tag{7.23}$$

that quantifies the offset between in- and output voltage for the last 50 time steps before the actual position of the input step, N_{step}.

7.2.2 M2: Magnitude Response from the Mean Signal Power

In the last section the square root of the power spectrum was used to evaluate the magnitude response. As will be shown below, this evaluation can also be approximated by calculating the ratio of the mean power of the in- and output signals for sine wave inputs of different frequencies f_k. Since the sine and cosine functions are eigenfunctions of LTI systems that possess a real-valued impulse response $h(t)$, the response of such an LTI system can be described by a change in amplitude and phase. Thus, for an input signal $x_k(t)$ with an offset $\overline{x_k} = \langle x_k(t) \rangle$, described by

$$x_k(t) \;=\; \overline{x_k} + A \sin(\omega_k t) \;\; , \tag{7.24}$$

the output of the LTI system can be written as

$$\mathscr{T}\{x_k(t)\} \;=\; y_k(t) \;=\; \overline{y_k} + M(\omega_k) A \sin(\omega_k t + \phi(\omega_k)) \;\; , \tag{7.25}$$

where $\overline{y_k}$ denotes the dc offset: $\overline{y_k} = \langle y_k(t) \rangle = \mathscr{T}\{\overline{x_k}\}$. For the proof within the theoretical framework developed so far, the reader is referred to appendix D.3, otherwise to chapter 3, pp. 99–103 in [Rup93].

[22] Fastest Fourier Transform of the West

Now, integrating the ac contribution of the in- and output over one period of time T_k and calculating the ratio thereof leaves one with:

$$\left[\frac{\int_0^{T_k}(y_k(t)-\overline{y_k})^2\,dt}{\int_0^{T_k}(x_k(t)-\overline{x_k})^2\,dt}\right]^{1/2} = \left[\frac{M^2(\omega_k)\,A^2\int_0^{T_k}\sin^2(\omega_k t+\phi(\omega_k))\,dt}{A^2\int_0^{T_k}\sin^2(\omega_k t)\,dt}\right]^{1/2} \equiv M_D(\omega_k)\ , \tag{7.26}$$

where the subscript D is added to distinguish the measured, *discrete* set of points from the magnitude itself as a property of the filter at hand. Being restricted to a finite set of N sampled data points, $M_D(\omega_k)$ can be approximated by a summation and substituting $t \to \frac{n}{N}T_k$:

$$M_D(f_k) = \left[\frac{M^2(\omega_k)\,A^2\sum_{n=0}^{N_k-1}\sin^2(\omega_k\frac{n}{N}T_k+\phi(\omega_k))}{A^2\sum_{n=0}^{N_k-1}\sin^2(\omega_k\frac{n}{N}T_k)}\right]^{1/2} = \left[\frac{\sum_{n=0}^{N_k-1}(y_k(n)-\overline{y_k})^2}{\sum_{n=0}^{N_k-1}(x_k(n)-\overline{x_k})^2}\right]^{1/2} . \tag{7.27}$$

In the first step, the following relation is used to cancel the sums in the nominator and denominator:

$$\sum_{n=0}^{N-1}\sin^2\left(2\pi\frac{n}{N}+\phi\right) = \frac{N}{2} \quad \text{for} \quad \{N\,|\,N\in\mathbb{N}\wedge N>2\}\ . \tag{7.28}$$

The derivation is given in section D.2 of the appendix.

Implementation. The data acquisition procedure used for this type of magnitude response evaluation is illustrated in Fig. 7.5. For each frequency $f_k, k \in \{1,\ldots,K\}$, $m = 2$ periods of a sine wave

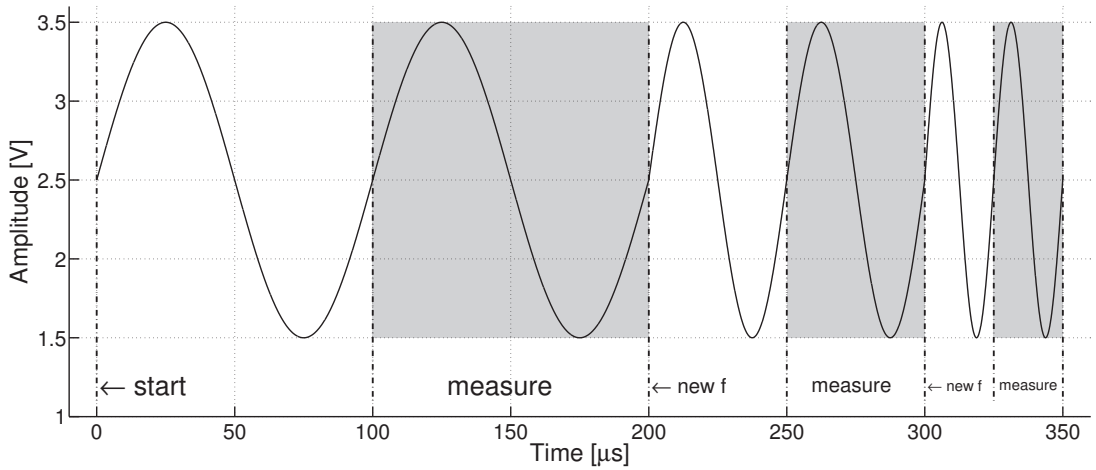


Figure 7.5: Principal idea of the stimulation for the ac-sweep based analyses: While the first period for each sine wave featuring a new frequency f is provided for the filter circuit under test to tune in, the output is only measured during the second period, shaded in gray.

are applied to the input. For the experiments presented in the remainder of this chapter, the sample frequencies f_k are logarithmically spaced spanning a range from 1 kHz to 1 MHz. For each decade 10 frequencies are tested resulting in a total of $K = 31$ different sine waves. The different sine waves are concatenated in ascending frequency order such that the resulting input signal is continuous at the frequency transition, albeit not differentiable. In order to allow the circuit under test to adapt to and

process the new input signal, the output is recorded only during the second half (i.e. the second period in this case) of the input pattern. The fitness according to the response to this logarithmic ac-sweep is again calculated from the sum of squared deviations from the target behavior:

$$F_{\text{acs}} = \sum_{\substack{k=0 \\ f_k \notin \text{transition band}}}^{K-1} \left(G_D(f_k) - G_{\text{tar}}(f_k)\right)^2 \ , \tag{7.29}$$

where $G_D(f_k)$ is again defined as the magnitude response $M_D(f_k)$ in dB.

7.2.3 M3: Magnitude Response from Fourier Analyzed Sinusoidal Stimuli

Another way of analyzing the filter response to the sinusoidal input stimulus $x_k(t)$ of (7.24) is to apply a Fourier transform and analyze the resulting frequency spectrum. With an additional phase offset ϑ_k introduced for generality, the Fourier transform of the input signal is calculated to

$$\begin{aligned} X_k(j\omega) \ &= \ \mathscr{F}\{x_k(t) - \overline{x_k}\} \ = \ A \int_{-\infty}^{\infty} \sin(\omega_k t + \vartheta_k) e^{-j\omega t}\, dt \\ &= \ \frac{\pi A}{j}\left[e^{j\vartheta_k}\,\delta(\omega_k - \omega) - e^{-j\vartheta_k}\,\delta(\omega_k + \omega)\right] \ . \end{aligned} \tag{7.30}$$

According to (7.7) the Fourier transform of the response of the LTI system can therefore be expressed for the complex frequencies $\pm j\omega_k$ by :

$$Y_k(\pm j\omega_k) \ = \ H(j\omega) X_k(\pm j\omega) \ = \ H(j\omega)\,\frac{\pi A}{j} e^{\pm j\vartheta_k}\,\delta(\omega_k \mp \omega) \ = \ \pm\frac{\pi A}{j} e^{\pm j\vartheta_k} H(\pm j\omega_k) \ . \tag{7.31}$$

Again, assuming the transfer function $h(t)$ to be real, one can use $H(-j\omega) = H^*(j\omega)$ (see appendix D.3) to find $Y_k(-j\omega) = Y_k^*(j\omega)$. Hence, taking the absolute value of both sides of (7.31) yields identical equations for both signs, namely

$$|H(j\omega_k)| \ = \ \frac{1}{\pi A}\,|Y_k(j\omega_k)| \ , \tag{7.32}$$

if one solves for $|H(j\omega_k)|$ to obtain the desired magnitude response.

Again, this theoretical result has to be transferred to the discrete world to be accessible by digital computation. The input signal $x_{k_0}(n)$ then is described by

$$x_{k_0}(n) - \overline{x_{k_0}} \ = \ A\sin(\omega_{k_0} t_n + \vartheta_{k_0}) \ = \ A\sin\left(2\pi\frac{n}{N_{k_0}} k_0 + \vartheta_{k_0}\right) \quad \text{with} \quad k_0, N_{k_0} \in \mathbb{N}\,,\ \frac{k_0}{N_{k_0}} \ll 1 \tag{7.33}$$

for $n = 1, \ldots, mN_{k_0}$ with $m \in \mathbb{N}$. That is, data is sampled for m periods of length N_{k_0} with a resolution $r_k = N_{k_0}/k_0$. Accordingly, the Fourier transform of (7.30) is exchanged by a discrete one such that the k^{th} Fourier component $X_{k_0}(k)$ can be written as

$$\begin{aligned} X_{k_0}(k) \ &= \ X_{k_0}(f_k) \ = \ \mathscr{F}\{x_{k_0}(n) - \overline{x_{k_0}}\} \ = \ A \sum_{n=0}^{mN_{k_0}-1} (x_{k_0}(n) - \overline{x_{k_0}})\, e^{-2\pi j k \frac{n}{N_{k_0}}} \\ &= \ \frac{A\, m N_{k_0}}{2j}\left(e^{j\vartheta_{k_0}}\,\delta(k - k_0) - e^{-j\vartheta_{k_0}}\,\delta(k + k_0)\right) \ , \end{aligned} \tag{7.34}$$

where the following relation is used:

$$\frac{1}{N_{k_0}} \sum_{n=0}^{N_{k_0}-1} e^{2\pi j n \frac{k}{N_{k_0}}} = \begin{cases} 1 & \text{if } k = 0, \pm N_{k_0}, \pm 2N_{k_0}, \dots \\ 0 & \text{otherwise} \end{cases} \\ = \delta(k) \quad \text{for} \quad 0 \leq k < N_{k_0} \ . \tag{7.35}$$

Since both k and $k_0 \in \{0, \dots, N_{k_0}\}$, the discrete analogon to (7.31) becomes:

$$Y_{k_0}(k) \ = \ \frac{A\,m\,N_{k_0}}{2j} H(k) e^{\pm j\vartheta_{k_0}}\, \delta(k \pm k_0) \ = \ \frac{A\,m\,N_{k_0}}{2j} H(\pm k_0) e^{\pm j\vartheta_{k_0}} \ , \tag{7.36}$$

which collapses to one single equation if the absolute value is taken on both sides for any real $h(t)$ (with (D.15) in appendix D.3). Solving for $|H(k_0)|$ finally yields:

$$M(k_0) \ = \ |H(k_0)| \ = \ \frac{2}{A\,m\,N_{k_0}} \, |Y_{k_0}(k_0)| \ . \tag{7.37}$$

As was already mentioned in section 7.1.1.1, the above relations only hold for LTI systems. In other words, the above calculation is only valid insofar as the filter circuit under test indeed is linear. As the components available on the FPTA are in general highly nonlinear, candidate circuits are likely to show some nonlinear behavior. While the first two methods for evaluating the magnitude response presented above fall short of quantifying this nonlinearity, the third method allows to do so by taking into account the frequency components $M(k)$, $k \neq k_0$, which would ideally be equal to zero for a perfectly linear circuit. Typically these harmonics are analyzed by means of the THD, which is defined as the RMS sum of all harmonics divided by the RMS amplitude of the fundamental frequency $M(k_0)$ (see [All02d], pp. 221):

$$\text{THD} \ = \ \frac{1}{|M(k_0)|} \sum_{l=2}^{N_{k_0}/2} \sqrt{M^2(l k_0)} \ , \tag{7.38}$$

where $M(k_0)$ represents the fundamental component of the sinusoidal input. Practically, only the first 4 to 9 harmonics are included in the THD of (7.38) (see e.g. [Max01], [Kes03], [Jou03]), which reflects the observation that harmonics of higher orders do not contribute significantly for systems, which closely resemble their linear ideal. A more thorough account of all recorded Fourier components except the dc one is given by the THD+N[23] (cf. e.g. [Kes03], [Car01]):

$$\text{THD+N} \ = \ \frac{1}{|M(k_0)|} \sum_{\substack{k=1 \\ k \neq k_0}}^{m N_{k_0}/2} \sqrt{M^2(k)} \ . \tag{7.39}$$

The distinction between THD and THD+N is based on the assumption that the noise is present in all frequency components, whereas the distortion effects are only exceeding the noise level by a significant amount for the first couple of harmonics of the input tone.

Implementation. The test pattern used for the Fourier-analyzed sinusoidal response resembles that one for the analysis of the mean signal power described in section 7.2.2: The input tone is applied for $2m$ periods, of which only the last m periods are used for the Fourier analysis. Yet, each frequency f_{k_0} is tested in a separate test mode. Moreover, the number of fundamental frequencies is reduced to six

[23] THD + Noise

(instead of 31), half of which are distributed logarithmically spaced in the pass- and in the stopband, respectively. The number of periods subject to the Fourier analysis is kept to a minimum satisfying the condition $mN_{k_0} \geq 100$. The constraint $k_0/N_{k_0} \gg 1$ for the number of samples per period in (7.33) is satisfied as $k_0/N_{k_0} \geq 20$, where '=' is reached only for $f_{k_0} = 1\,\text{MHz}$ and $k_0/N_{k_0} \geq 100$ is sought for all f_{k_0} for which the maximum sampling rate of 20 MHz suffices.

The resulting fitness criterion

$$F_{\text{acs+d}} = P_{\text{dist}} + \sum_{k_0=0}^{K_{k_0}-1} \big(G(k_0) - V_{\text{tar}}(k_0)\big)^2 w(k_0) \tag{7.40}$$

consists of two main parts, namely a penalty term P_{dist} that captures the amount of harmonic distortion and dc-offset and the sum of quadratic deviations from the desired magnitude response. Again, $w(k), k \in \{0, 1, \ldots, mN_{k_0}\}$ denote a set of weight factors. Apart from the way the magnitude responses are obtained and the number thereof, the second summand is identical to the fitness functions F_{step} of (7.22) and F_{acs} of (7.29). The penalty term is based on the expression for THD+N in (7.39), albeit, in its general form, includes the dc component of the Fourier transform:

$$P_{\text{dist}} \;=\; \sum_{k_0=0}^{K_{k_0}-1} \sum_{\substack{k=1 \\ k\neq k_0}}^{mN_{k_0}} \theta\big(G(k) - V_{\text{tar}}(k)\big)\,\big(G(k) - V_{\text{tar}}(k)\big)^2 w(k) \quad . \tag{7.41}$$

The penalty distortion term P_{dist} differs in two additional regards from the expression for THD+N in (7.39): First, opposite to the definition of THD+N, P_{dist} is based on the magnitude response in dB, $G(k_0)$. Second, only those contributions are added that exceed a predefined target distortion $V_{\text{tar}}(k)$, which is set to $-40\,\text{dB}$ for the experiments presented below, and third, only the deviation from this target distortion suppression is regarded, such that the $P_{\text{dist}} = 0$ if all but the fundamental frequency component remain below V_{tar}.

In order to quantify distortion and magnitude response separately by means of M3 in a unique fashion for filter circuits evolved with either of the three proposed methods M1 to M3, two additional test modes are derived from M3 and added to the verification tests. Both test modes are based on the same test pattern used for evaluating F_{acs}, which is described in section 7.2.2. The fitness contribution F_{acsf} capturing only the quality of the magnitude response is described by

$$F_{\text{acsf}} = \sum_{k_0=0}^{K_{k_0}-1} \big(G(k_0) - V_{\text{tar}}(k_0)\big)^2 w(k_0) \quad , \tag{7.42}$$

which possesses the same structure as $\text{F}_{\text{acs+d}}$ defined in (7.40), except for the missing penalty term P_{dist}. The values for the magnitude response in dB, $G(k_0)$, are again taken as the according frequency components of the Fourier transform of the input tone of frequency f_{k_0}. The expression for the additional distortion measure is related more closely to the THD+N defined in (7.39) and hence different from the penalty factor P_{dist} used in (7.40):

$$F_{\text{dist}} \;=\; \sum_{k_0=0}^{K_0-1} \theta\big(\text{THD+N}(k_0) - V_{\text{tar}}(k_0)\big)\,\text{THD+N}(k_0) \quad \text{with} \tag{7.43}$$

$$\text{THD+N}(k_0) \;=\; 10\log_{10}\left(\sum_{\substack{k=1 \\ k\neq k_0}}^{N_{k_0}/2} |H_{k_0}(k)|^2\right) \quad , \tag{7.44}$$

where only those terms contribute that exceed the target maximum THD+N that can be tolerated. In the according experiments presented in the remainder of this chapter, $V_{\text{tar}}(k_0)$ is set to $-40\,\text{dB}$.

7.2.4 Noise Floor

The evolution and test of filter circuits requires the correct determination of the magnitude response in both, the pass- and the stopband. The maximum stopband attenuation that can be measured for *real* circuits depends on the amount of noise and distortion introduced by the measurement system. However, as will be shown below, the noise floor varies for the three different methods of establishing the magnitude response. In order to simplify the analysis, the discussion is restricted to white noise $r(t)$ with a mean $\langle r(t)\rangle = 0$. After the analog to digital conversion, the noise is given as a discrete time series $r(n), n = 0,\ldots,N-1$. Thus its power density is approximated by

$$r_0{}^2 \equiv \left\langle r^2(n)\right\rangle \;=\; \frac{1}{N}\sum_{n=0}^{N-1}|r(n)|^2 \;\approx\; \frac{1}{T}\int_0^T |r(t)|^2\,dt \;=\; \left\langle r^2(t)\right\rangle \quad . \tag{7.45}$$

In order to compare the mean noise contribution to each of the frequency components in the recorded spectrum, Parseval's relation, defined in (D.2) in section D.1.1, can be used to calculate the mean noise power in frequency space:

$$R_0{}^2 \equiv \left\langle R^2(k)\right\rangle \;=\; \frac{1}{N}\sum_{k=0}^{N-1}|R(k)|^2 \;\overset{\text{Parseval}}{=}\; N\cdot\frac{1}{N}\sum_{n=0}^{N-1}|r(n)|^2 \;=\; N\left\langle r^2(n)\right\rangle \quad , \tag{7.46}$$

where $R(k)$ denotes the discrete Fourier Transform of $r(n)$ at frequency f_k. Due to the linearity of the Fourier transform, the frequency spectrum of the measured output signal $y(n) = s(n) + r(n)$ – the sum of noise $r(n)$ and ideal signal $s(n)$ – can be written as:

$$\mathscr{F}\{y(n) = s(n) + r(n)\} \;=\; S(k) + R(k) \quad \text{for} \quad n,k = 0,\ldots,N-1 \quad . \tag{7.47}$$

Therefore, the noise contributions encountered in the applications of methods M1 and M3 can be studied independently of the actual desired signal. Since $R(k)$ is independent of frequency, its expectation value defined in (7.45) can be used to describe $R(k)$ and due to its probabilistic nature, it must be used in (7.47).

For the quantitative analysis, the noise $r(n)$ is assumed to be uniformly distributed with a maximum amplitude of $\pm 1/2 \cdot V_{\text{LSB}}$ and $V_{\text{LSB}} = 5\,\text{V}/2^{n_{\text{eff}}}$, where the effective number of bits n_{eff} is bound to lie between the effective number of bits measured for the respective sample frequency $\text{ENOBS}(f_{\text{sample}})$[24]

and the resolution of the conversion device, that is $n_{\text{eff}} = 16$ for the noise generated at the input of the circuit r_{in} and $n_{\text{eff}} = 12$ for the quantization noise encountered in the final analog-to-digital conversion. According to Allen and Holberg ([All02b], chapter 10.1, pp. 616) the mean noise power for a uniform distribution is described by:

$$r_{\text{RMS}} \;=\; \sqrt{\overline{r^2}} \;=\; \sqrt{\left\langle r^2(t)\right\rangle} \;=\; \left[\frac{1}{T}\int_0^T |r(t)|^2\,dt\right]^{\frac{1}{2}} \;=\; \frac{V_{\text{LSB}}}{\sqrt{12}} \;=\; \frac{5\,\text{V}}{\sqrt{12}\,n_{\text{eff}}} \quad . \tag{7.48}$$

7.2.4.1 Noise for Method M1

The evaluation procedure of method M1 is illustrated in the block diagram of Fig. 7.6: The ideal input step $u(n) = \theta(n)$ is deteriorated by the input noise $r_{\text{in}}(n)$ generated in the digital-to-analog conversion and the analog input circuitry used for its further preparation. The step response of the filter circuit

[24] Between 7.5 and 8.8 ENOBS have been measured for sufficiently low frequencies depending on the system

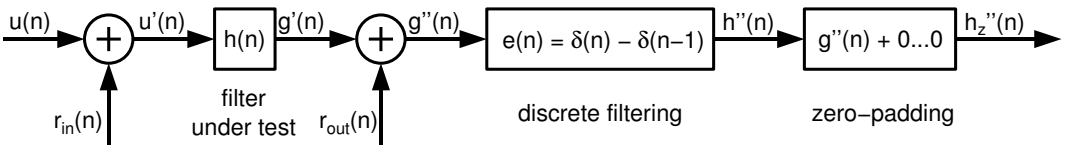


Figure 7.6: Data path for experiments using method M1 to establish the magnitude response.

under test is sampled and buffered by the analog output circuitry before it is finally converted back to the digital domain. The accompanying noise contribution are summarized by $r_{\text{out}}(n)$. This procedure is repeated three times and the resulting step responses $g''(n)$ are averaged, which is omitted in Fig. 7.6. The discretized signal is then subjected to the operation described by (7.19), which can be treated as a discrete linear filter with an impulse response

$$\mathscr{T}_E\{\delta(n)\} \;\equiv\; e(n) \;=\; \delta(n)-\delta(n-1) \;\; . \tag{7.49}$$

Finally, $3N$ zeroes are appended at the resulting impulse response $h''(n)$ before it is Fourier transformed to yield the transfer function $H_z''(k)$.

Mathematically, the above procedure can be described by the following equation:

$$y(n) \;\equiv\; h''(n) \;=\; e(n) * \big[r_{\text{out}}(n) + h(n) * \big(x(n) + r_{\text{in}}(n)\big)\big] \;\; , \tag{7.50}$$

where the averaging over three measurements and the zero-padding are omitted. To obtain the sought transfer function, the Fourier transform is applied to (7.50), . If the input is excited by a step function this results in:

$$\begin{aligned} H''(k) \;&=\; \mathscr{F}\{h''(n)\} \\ &=\; \mathscr{F}\{e(n) * r_{\text{out}}(n)\} + \underbrace{\mathscr{F}\{e(n) * h(n) * x(n)\}}_{H(k)\;\text{ for }\;x(n)=\theta(n)} + \mathscr{F}\{e(n) * h(n) * r_{\text{in}}(k)\} \\ &\overset{x(n)=\theta(n)}{=}\; H(k) + E(k)R_{\text{out}}(k) + E(k)H(k)R_{\text{in}}(k) \;\; . \end{aligned} \tag{7.51}$$

Compared to the output noise contribution $E(k)R_{\text{out}}(k)$, the contribution stemming from the noise affecting the input signal $E(k)H(k)R_{\text{in}}(k)$ is weighted with the according frequency component of the filter under test $H(k)$. Therefore, in the critical frequency regime of the stopband, in which the smallest frequency components of the transfer function have to be detected, the input noise is suppressed by the attenuation A_{SB}, such that its contribution to $H''(k)$ is negligible compared to that one caused by the noise encountered in sampling the circuit's output. Thus, (7.51) can be approximated by:

$$H''(k) \approx\; H(k) + E(k)R_{\text{out}}(k) \;\equiv\; H(k) + R''_{\text{out}}(k) \;\; . \tag{7.52}$$

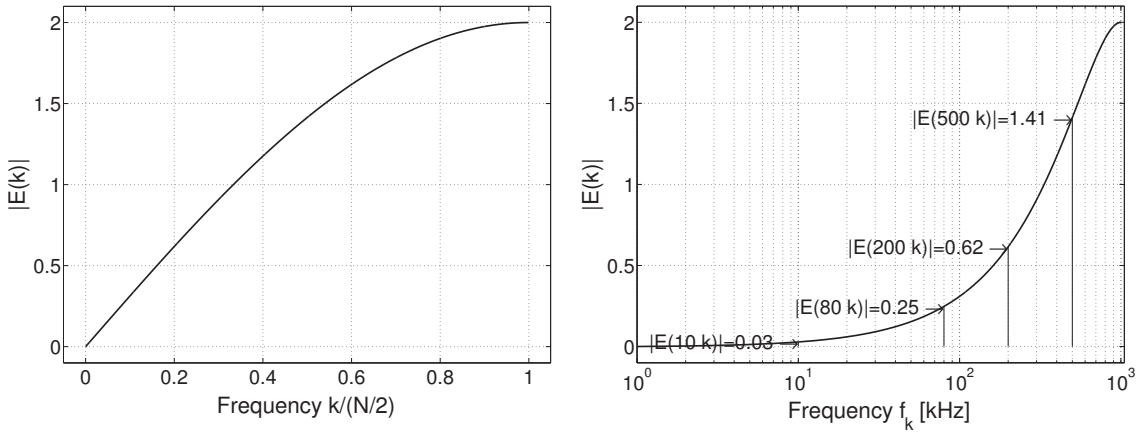
To further qualify the noise contribution $R''_{\text{out}}(k)$, the magnitude response of the digital post-processing operation $\mathscr{T}_E$ has to be calculated. Therefore, a DFT is applied to the impulse response $e(n)$ defined in (7.49):

$$\begin{aligned} E(k) \;&=\; \mathscr{F}\{e(n)\} \;=\; \mathscr{F}\{\delta(n)-\delta(n-1)\} \\ &=\; \sum_{n=0}^{N-1} \big(\delta(n)-\delta(n-1)\big) e^{-2\pi j \frac{kn}{N}} \\ &=\; 1 - e^{-2\pi j \frac{k}{N}} \;=\; e^{-2\pi j \frac{k}{2N}} \big(e^{2\pi j \frac{k}{2N}} - e^{-2\pi j \frac{k}{2N}}\big) \;=\; 2e^{-2\pi j \frac{k}{2N}} \sin\Big(2\pi \frac{k}{2N}\Big) \;\; , \end{aligned} \tag{7.53}$$

which yields the following formula for the magnitude response of the post-processing filter:

$$|E(k)| \;=\; 2\sin\!\left(2\pi\frac{k}{2N}\right) \;. \tag{7.54}$$

The magnitude response $|E(K)|$ is plotted twice in Fig. 7.7: For one, against a normalized frequency $k/(N/2)$ in Fig.7.7(a) and for the other against a logarithmic axis denoting the real frequencies f_k in Fig.7.7(b). The latter plot contains the values of $|E(K)|$ at four particularly interesting frequen-



(a) linear (b) logarithmic

Figure 7.7: Magnitude response $|E(k)|$ of the digital post-processing applied to the step response recorded for method M1. (a) depicts (7.54) on a linear normalized frequency axis and (b) plots $|E(k)|$ versus the corresponding real frequency f_k.

cies f_k, namely at 10 kHz and 80 kHz, which span the range of passband edges used for the different LPF experiments, at the stopband edge located at 200 kHz and at 500 kHz, the end of the frequency spectrum considered. The discrepancy between the plot stretching to 1 MHz and the maximum frequency used for the evaluation is due to the fourfold zero padding indicated in Fig. 7.6: The $N = 500$ samples are inflated to 2000 by appending 1500 zeros and Fourier transformed to yield $4N/2 = 1000$ frequency samples below the Nyquist frequency. These $H_z''(k)$ are finally truncated after the 500$^{\text{th}}$ sample. Accordingly, $|E(k)|$ does not exceed $\sqrt{2}$ for all frequencies considered and amounts to about 1 on average throughout the stopband. In the passband on the other hand the magnitude of the post-processing filter is bound to be less than 0.25.

Due to the highpass characteristic of $E(k)$, the output noise $R''_{\text{out}}(k)$ this is more harmful for the evaluation of the magnitude response of LPFs than for that of HPFs. In case of HPFs, the smallest frequency components of $H(k)$ occur at low frequencies, where the noise contribution of R_{out} is also suppressed by $E(k)$. For the experiments presented in section 7.5 the stopband edge is set to 10 kHz, such that $E(k) < 0.03$. For LPFs on the other hand $E(k)$ ranges from 0.62 to $\sqrt{2}$. In the latter case, the noise floor in the stopband can thus be estimated by:

$$\begin{aligned}\overline{R''_{\text{out}}} \;&=\; E(k)\,\frac{1}{\sqrt{3}}\,\overline{R_{\text{out}}} \;\le\; \sqrt{\frac{2}{3}}\,\sqrt{N\,\langle r^2(n)\rangle} \;=\; \sqrt{\frac{2}{3}}\,\sqrt{N\frac{V_{\text{LSB}}^2}{12}} \\ &=\; \sqrt{\frac{N}{18}}\,\frac{5}{2^{n_{\text{eff}}}} \;=\; \begin{cases} -43.83\,\text{dB} & \text{for} \quad n_{\text{eff}} = 12 \\ -19.75\,\text{dB} & \text{for} \quad n_{\text{eff}} = 8 \end{cases},\end{aligned} \tag{7.55}$$

where $N = 500$ is used, because the zero-padding does not affect the result except for the scaling of $E(k)$ already accounted for in the factor $\sqrt{2}$. The mean output noise power $\overline{R_{\text{out}}}$ is weighted by a

factor of $\sqrt{3}$, because the step response is averaged over three measurements. As small signals of the order of $\overline{R''_{\text{out}}}$ and below cannot be resolved, the stopband attenuation $A_{\text{SB}} = 40\,\text{dB}$ sought in the LPF experiments is on the edge of the theoretically achievable accuracy that is only limited by the finite resolution of the analog-to-digital conversion. For the stopband of the prospective HPF evolved in section 7.5, the mean noise contribution $\overline{R''_{\text{out}}}$ improves by at least $20 \cdot \log_{10}(0.03/\sqrt{2}) = -33.47\,\text{dB}$ such that $\overline{R''_{\text{out}}} \leq 53.22\,\text{dB}$ is achieved for an effective resolution of only $n_{\text{eff}} = 8$ bits.

7.2.4.2 Noise for Method M2.

The situation for evaluating the magnitude response according to either of the methods M2 and M3 is illustrated in Fig. 7.8; in both cases the input signal $x(n)$ is described by (7.24). For method M2 the

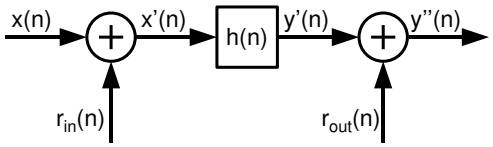


Figure 7.8: Data path for experiments using method M2 or M3 to establish the magnitude response.

magnitude response is calculated by (7.27), which can be simplified to

$$M_D''^2(f_{k_0}) \;=\; \left[\frac{\sum\limits_{n=0}^{N_{k_0}-1} \widetilde{y''_{k_0}}^2(n)}{\sum\limits_{n=0}^{N_{k_0}-1} (x_{k_0}(n)-\overline{x_{k_0}})^2}\right]^{1/2} \;=\; \sqrt{\frac{2}{NA^2}\sum_{n=0}^{N_{k_0}-1} \widetilde{y''_{k_0}}^2(n)}\;, \tag{7.56}$$

by means of (7.28). Here, the measured output voltages consist of the following contributions:

$$\begin{aligned}\widetilde{y''_{k_0}}(n) \;&=\; y''_{k_0}(n) - \overline{y''_{k_0}} \;=\; y_{k_0}(n) - \overline{y_{k_0}} + r_{k_0}(n) - \overline{r_{k_0}} \;=\; y_{k_0}(n) - \overline{y_{k_0}} + r_{k_0}(n) \\ &=\; h(n) * (x_{k_0}(n) - \overline{x_{k_0}}) + h(n) * r_{\text{in}}(n) + r_{\text{out}}(n) \;=\; \tilde{y}_{k_0}(n) + h * r_{\text{in}}(n) + r_{\text{out}}(n)\;.\end{aligned} \tag{7.57}$$

Inserting $\widetilde{y''_{k_0}}(n)$ from (7.57) into (7.56) yields:

$$\begin{aligned}M_D''^2(f_{k_0}) \;=\; &\frac{2}{NA^2}\sum_{n=0}^{N_{k_0}-1} \tilde{y}_{k_0}^2(n) + \frac{2}{NA^2}\left[\sum_{n=0}^{N_{k_0}-1}\left((h * r_{\text{in}}(n))^2 + r_{\text{out}}^2(n)\right)\right. \\ &\left. + \underbrace{\sum_{n=0}^{N_{k_0}-1} 2\tilde{y}_{k_0}^2(n)\; h * r_{\text{in}}(n)}_{\approx 0} + \underbrace{\sum_{n=0}^{N_{k_0}-1} 2\tilde{y}_{k_0}^2(n)\; r_{\text{out}}(n)}_{\approx 0} + \underbrace{\sum_{n=0}^{N_{k_0}-1} 2h * r_{\text{in}}(n)\; r_{\text{out}}(n)}_{\approx 0}\right]\end{aligned} \tag{7.58a}$$

If r_{out} and r_{in} is noise with a uniform or Gaussian distribution and a vanishing mean, the three sums of the second line will converge to zero for taking N_{k_0} to infinity. Though this does not hold for finite N_{k_0} in a strict sense, the summands are nevertheless assumed to be small enough to be neglected. (7.58a) can be simplified by using (7.27) for the first summand and Parseval's relation (D.2) for the second one:

$$M_D''^2(f_{k_0}) \;\approx M_D^2(f_{k_0}) + \frac{2}{N^2A^2}\sum_{k=0}^{N_{k_0}-1} |H(k)|^2\,|R_{\text{in}}(k)|^2 + \frac{2}{NA^2}\sum_{n=0}^{N_{k_0}-1} r_{\text{out}}^2(n)\;. \tag{7.58b}$$

Replacing $|R_{\text{in}}(k)|^2$ and $r^2_{\text{out}}(n)$ by their expectation values and again obeying Parseval's relation from (D.2) leads to

$$M_D''^2(f_{k_0}) \approx M_D^2(f_{k_0}) + \frac{2}{A^2}\left\langle r^2_{\text{in}}\right\rangle \frac{1}{N}\sum_{k=0}^{N_{k_0}-1}|H(k)|^2 + \frac{2}{A^2}\left\langle r^2_{\text{out}}\right\rangle \tag{7.58c}$$

$$= M_D^2(f_{k_0}) + \frac{2}{A^2}\left\langle r^2_{\text{in}}\right\rangle \left\langle |H(k)|^2\right\rangle + \frac{2}{A^2}\left\langle r^2_{\text{out}}\right\rangle \ . \tag{7.58d}$$

Due to the integrating nature of (7.27), which defines the components of the magnitude response, input noise $R_{\text{in}}(k)$ is not suppressed by the according component of the filter's transfer function $H(k)$ as found for method M1, but rather weighted by its mean energy density $\left\langle |H(k)|^2\right\rangle$. The latter one however can only exceed one, if the filter is to amplify the frequency components in the passband. For the proposed fitness criteria used for the LPF and HPF experiments described below, the filter's mean energy density $\left\langle |H(k)|^2\right\rangle$ is bound to be smaller than one. Since the mean input noise power $\left\langle r^2_{\text{in}}\right\rangle$ is expected to be of the same size[25] as the output noise power r^2_{out}, the total noise can be estimated as follows:

$$\begin{aligned}\overline{R_{k_0}} &= \sqrt{M_D''^2(f_{k_0}) - M_D^2(f_{k_0})} = \sqrt{\frac{2}{A^2}\left\langle r^2_{\text{in}}\right\rangle \left\langle |H(k)|^2\right\rangle + \frac{2}{A^2}\left\langle r^2_{\text{out}}\right\rangle} \\ &\leq 2\sqrt{\left\langle r^2_{\text{out}}\right\rangle} = 2\sqrt{\frac{V^2_{\text{LSB}}}{12}} = \frac{1}{\sqrt{3}}\frac{5}{2^{n_{\text{eff}}}} = \begin{cases} -38.96\,\text{dB} & \text{for} \quad n_{\text{eff}} = 8 \\ -63.04\,\text{dB} & \text{for} \quad n_{\text{eff}} = 12 \end{cases} .\end{aligned} \tag{7.59}$$

7.2.4.3 Noise for Method M3

Since Method M3 uses the same input stimulus for evaluating the magnitude response as method M2, the block diagram of Fig. 7.8 as well as (7.57) apply in this case, too. Therefore, to estimate the noise floor present in the magnitude response spectra obtained by method M3, one has to plug the Fourier transform of (7.57)

$$\widetilde{Y''_{k_0}}(k) = \tilde{Y}_{k_0}(k) + H(k)R_{\text{in}}(k) + R_{\text{out}}(k) \ , \tag{7.60}$$

into (7.37):

$$\widetilde{M''_{k_0}}(k) = \frac{2}{A\,m\,N_{k_0}}|\widetilde{Y''_{k_0}}(k)| = \frac{2}{A\,m\,N_{k_0}}\left|\tilde{Y}_{k_0}(k) + H(k)R_{\text{in}}(k) + R_{\text{out}}(k)\right| \ , \tag{7.61}$$

where (7.37) is evaluated for all $k = 1,\ldots,N_{k_0}$ and not only at k_0 to account for the noise in the harmonics of f_{k_0} used to evaluate the THD. Being interested in the noise floor only, the further analysis is restricted to the latter two summands in (7.61) constituting the total noise contribution $R''_{k_0}(k)$ in frequency space:

$$R''_{k_0}(k) = \frac{2}{A\,m\,N_{k_0}}\left|H(k)R_{\text{in}}(k) + R_{\text{out}}(k)\right| \ , \tag{7.62a}$$

where the total number of regarded samples amounts to m periods of N_{k_0} samples. The above expression can be evaluated by replacing $R_{\text{in}}(k) + R_{\text{out}}(k)$ by their respective RMS averages

$$\overline{R''_{k_0}(k)} = \frac{2}{A\,m\,N_{k_0}}\left|H(k)\sqrt{\left\langle R^2_{\text{in}}\right\rangle} + \sqrt{\left\langle R^2_{\text{out}}\right\rangle}\right| \ , \tag{7.62b}$$

[25] Actually, the quantization noise of the 12-bit ADC clearly exceeds that one of the 16-bit DAC; the analog circuitry used to sample and amplify the signal on the chip consists of similar circuits for the in- and output signal and the differences in the off-chip analog circuitry are not as severe as to justify a separate treatment of in- and output noise.

which are subsequently replaced by their counterparts in the time domain by means of Parseval's relation defined in (D.2):

$$= \frac{2}{A\,m\,N_{k_0}} \left| H(k) \sqrt{m N_{k_0} \left\langle r_{\text{in}}^2 \right\rangle} + \sqrt{m N_{k_0} \left\langle r_{\text{out}}^2 \right\rangle} \right| \tag{7.62c}$$

$$= \frac{2}{A\,\sqrt{m N_{k_0}}} \left| H(k) \sqrt{\left\langle r_{\text{in}}^2 \right\rangle} + \sqrt{\left\langle r_{\text{out}}^2 \right\rangle} \right| \quad . \tag{7.62d}$$

Similar to method M1, but unlike method M2, the input noise $\langle r_{\text{in}}^2 \rangle$ is suppressed with the magnitude of the according frequency component of the transfer function of the filter under test, $H(k)$. However, in the experiments presented in the remainder of this chapter $|H(k)|$ can be assumed to be smaller or equal to 1, and is thus approximated by 1, because (7.62d) is used to estimate the noise floor for all frequency components f_k, independent of the fundamental frequency f_{k_0} of the input tone. To quantify the total noise $\overline{R''_{k_0}(k)}$, the mean in- and output noise power are estimated by r_{RMS} defined in (7.48):

$$\begin{aligned} \overline{R''_{k_0}(k)} &\leq \frac{2}{A\,\sqrt{m N_{k_0}}} \sqrt{2} \frac{V_{\text{LSB}}}{\sqrt{12}} = \sqrt{\frac{2}{3}} \frac{1}{A\,\sqrt{m N_{k_0}}} \frac{5\,\text{V}}{2^{n_{\text{eff}}}} \\ &\leq \begin{cases} -48.96\,\text{dB} & \text{for} \quad n_{\text{eff}} = 8 \\ -73.04\,\text{dB} & \text{for} \quad n_{\text{eff}} = 12 \end{cases} , \end{aligned} \tag{7.63}$$

7.2.4.4 Comparison of the Different Noise Floors for Methods M1-M3

The results of the noise floor calculations are summarized in Table 7.3. First, a comparison of the

Meth.	Equ.	RMS noise power	$n_{\text{eff}} = 8$	$n_{\text{eff}} = 12$
M1	(7.55)	$\overline{R''_{\text{out}}} \approx E(k) \frac{1}{\sqrt{3}} \sqrt{N \langle r^2(n) \rangle}$	$-19.75\,\text{dB}$	$-43.83\,\text{dB}$
M2	(7.59)	$\overline{R_{k_0}} \approx \sqrt{\frac{2}{A^2} \left\langle r_{\text{in}}^2 \right\rangle \langle \lvert H(k) \rvert^2 \rangle + \frac{2}{A^2} \left\langle r_{\text{out}}^2 \right\rangle}$	$-38.96\,\text{dB}$	$-63.04\,\text{dB}$
M3	(7.62d)	$\overline{R''_{k_0}(k)} \approx \frac{2}{A\,\sqrt{m N_{k_0}}} \left\lvert H(k) \sqrt{\left\langle r_{\text{in}}^2 \right\rangle} + \sqrt{\left\langle r_{\text{out}}^2 \right\rangle} \right\rvert$	$-48.96\,\text{dB}$	$-73.04\,\text{dB}$

Table 7.3: Comparison of the noise floor equations and values achieved by the three different methods of establishing the magnitude response, M1–M3. The numerical values for different ENOBS can be calculated by adding 6.02 dB for each additional bit.

formulas for the RMS noise power reveals that the noise picked up in the procedure M1 increases with $\sqrt{N}$, that of M2 is independent and that of M3 decreases with $\sqrt{m N_{k_0}}$. Therefore, the noise floor can be lowered by decreasing the sample size for method M1 and increasing the number of sampled periods m in case of method M3. While the latter one can easily be realized by spending more time for the evaluation of the filter's magnitude response, the former decrease in sample size necessarily reduces the bandwidth of the measured frequency spectrum at the lower end. Considering that because of the magnitude response of $E(k)$, the noise floor of M1 is fairly good, the measurement could be divided into two parts to account for the higher frequencies with a smaller number of samples. Second, the noise floor for the situation at hand is considerably lower for methods M2 and M3 compared to that of M1, which is due to the abovementioned dependencies on the number of samples. In principle, this implies that – at least for LPFs – M3 is the method of choice to obtain a reliable magnitude response at

higher frequencies. However, the net-effect may be less significant as the numbers suggest: Methods M2 and M3 employ sine tones sampled at frequencies up to $f_{\text{sample}} = 20\,\text{MHz}$ for the components beyond 200 kHz. This certainly entails a decrease in the precision of the analog in- and output circuitry n_{eff}, which increases the according noise floor values. For instance, for a decrease of n_{eff} by 2 bits, the distance between the noise floor of M1 and M3 would decrease from about 29 dB to only 17 dB.

7.2.5 List of All Test Modes

While the third method M3 for evaluating the magnitude response of the circuit under test offers a possibility to quantify its nonlinearities in terms of its harmonic distortion, the methods M1 and M2 are merely *based* on the assumption that the evaluated filters are (sufficiently) linear and time invariant, they do not provide any means of measuring or even controlling linearity. Therefore, an additional test mode is added for the experiments based on method M1 and M2: A simple transient analysis of a sinusoidal input tone

$$V_{\text{in}}(n) \;=\; 2.5\,\text{V} + A\sin\!\left(2\pi f\,\frac{nT}{N}\right) \quad \text{with} \quad 0 \le n < N \tag{7.64}$$

with frequency $f = 1.67\,\text{kHz}$ and amplitude $A = 1.5\,\text{V}$ is used to test in how far the filter under test deteriorates the output signal compared with the (ideal) input signal in the passband. The test pattern is restricted to one period, sampled in $N = 100$ time steps, only, but a total of 20 time steps is used to allow the circuit to adapt to the mean input voltage level of 2.5 V. The amplitude A of the input tone is chosen to be larger than the excursion used for the ac-sweep and the step response (1.5 V instead of 1 V) to force the filter to be linear over a larger range of input voltages in the passband. The according fitness contribution is simply described by the sum of squared deviations

$$F_{\text{trans}} \;=\; \sum_{n=0}^{N_{\text{trans}}} \big(V_{\text{out}}(n) - V_{\text{tar}}(n)\big)^2 \;\;, \tag{7.65}$$

where $V_{\text{tar}}(n) = V_{\text{in}}(n)$ describes the target response of the filter in the passband, namely a gain of 0 dB and no phase shift.

The complete set of test modes used in the experiments presented below are summarized in Table 7.4. The 20 test modes can be divided into three classes by their purpose indicated by the according column. While only a fraction of them is used during the evolution, others are used to further specify the circuits in the verification test and enable comparisons between different experiments and experiment series. Although not all of the test modes labeled with *illustration* are used for the plots shown in the results sections, they are all included in the table for reasons explained in the next section.

In case of evaluating the magnitude response by means of a step response (method M1), two test modes are used to provide steps from 2.5 V down to 1.5 V as well from 2.5 V up to 3.5 V. On one hand, this increases the sampled input compliance, on the other hand it imposes further pressure on the linearity of the prospective circuits under evolution. In general, if not stated otherwise, the amplitude of all sinusoidal input tones is set to 1 V and the starting and mean dc voltage is always kept at 2.5 V.

Opposite to the abstract filter specification in section 7.1.1.2, the fitness criteria evaluating the magnitude response do not allow any passband ripple δ. However, for small values, say $\delta \le \pm 1\,\text{dB}$, the according error contributions are small and thus not expected to mislead the EA. The exact composition of fitness criteria and figures of merit from the ensemble of test modes as well as a further motivation of those not yet discussed are delayed to the description of the respective experimental setups.

No.	Description	Name	Equation	Purpose	Weight[a]	N_{total}
1	Distortion: Fourier-analyzed ac-sweep	F_{dist}	(7.43)	verification	0.5	24693
2	Magnitude response: Fourier analyzed ac-sweep	F_{acsf}	(7.42)	verification	0.5	24693
3	**M2**: Magnitude response: mean signal power, ac-sweep	F_{acs}	(7.29)	evolution	0.5	24693
4	Trans. resp. for ac sweep		–	illustration	–	24693
5	Magnitude response: mean signal power, criterion of experiment 5[b]	F_{acs}	(7.29)	verification	0.5	24693
6	Transient response, 625 kHz sine tone		–	illustration	–	201
7	Transient response, 822 Hz sine tone		–	illustration	–	201
8	Transient response, 1.67 kHz sine tone	F_{trans}	(7.65)	evolution	10	20+100
9	**M1**: Step response: 2.5 V → 1.5 V	$F_{\text{step}\downarrow}$	(7.22)	evolution	1	500
10	**M1**: Step response: 2.5 V → 3.5 V	$F_{\text{step}\uparrow}$	(7.22)	evolution	1	500
11	Step response: 2.5 V → 1.5 V criterion of exp. 5[b]	$F_{\text{step}\downarrow}$	(7.22)	verification	1	500
12	Step response: 2.5 V → 3.5 V criterion of exp. 5[b]	$F_{\text{step}\uparrow}$	(7.22)	verification	1	500
13	Trans. resp. for step: 2.5 V → 1.5 V		–	illustration	–	500
14	Trans. resp. for step: 2.5 V → 3.5 V		–	illustration	–	500
15-20	**M3**: Magn. resp. + distortion: Fourier analyzed ac-sweep, 6 different f_{k_0}	$F_{\text{acs+d}}$	(7.40)	evolution	1	201-331

[a] Weight refers to the global weight/scaling of the respective fitness contribution. Test modes 9-12 and 15-20 possess additional weights for their different frequency components as discussed above and below, respectively.
[a] The transition band is the narrowest for experiments of type exp. 5. It stretches from 80 to 200 kHz (cf. section 7.3.1).

Table 7.4: List of all test modes used during the artificial evolution and/or for verification tests.

7.2.6 Randomization for Time-Dependent Experiments

For quasi-dc hardware evolution experiments it has been demonstrated in 5.3 on page 141 that the unwanted exploitation of the temporal order of the test pattern can be avoided by randomizing the input data. This, of course, is impossible for time-dependent tasks. Nevertheless, the optimization algorithm may find fake solutions that rely on a particular state of the circuit – e.g. a special type of charge distribution – produced by a previously test or tested circuit that will not automatically emerge or be sustained by the desired operation of the circuit itself. To overcome this problem, two mechanism are incorporated in the DARKGAQT software. First, the order of executing the different test modes can be randomized; throughout the experiments of this chapter, ten random orders are used. Second, the analog substrate can be reset by different mechanisms selectable for the user. This *substrate reset* is achieved by downloading a special type of genotype either after each test mode or after each complete test of one individual. The type of *reset* is determined by means of the reset-genotype. The following *substrate reset* alternatives currently available are listed in Table 7.5.

The first six reset-types can be grouped into three pairs that only differ in the parts of the configurable transistor cells that are included: In the respective **last** transistor variant, the genotype described in the according column of Table 7.5 is downloaded only once using the transistor dimensions of the previous individual. If the **all** transistors option is selected, all transistors forming the configurable transistor cell are activated by downloading the gene five times for each different transistor length and setting the transistor width to the maximum of 15 µm. Since the evolving circuits shall be independent of the actual state of the substrate, i.e. are not to depend on any floating charges left on the PTA, the

No.	Name	Description of the reset genotype
1	**last** transistor only	all switches closed; all transistor terminals → *gnd*
2	**all** transistors	
3	**last** transistor + routing	all switches closed; NMOST: all terminals → *gnd*
4	**all** transistors + routing	PMOST: Gate/Drain → *gnd*, Source → South
5	**last** transistor only	all terminals of each transistor → *gnd or vdd*
6	**all** transistors only	randomly chosen
7	complete random gene	completely randomized genotype

Table 7.5: Seven different genotypes for a *substrate reset* of the FPTA.

download of a randomly generated genotype is used to approximate a random charge distribution on the chip. Therefore, in all experiments of this chapter as well as those presented in section 8.4 on page 257 the *substrate reset* number seven is applied after each complete test of an individual.

7.3 Evolving Low Pass Filters

The three different methods of measuring and evaluating the magnitude response of a candidate circuit are applied to the problem of evolving an analog linear lowpass filter. Thereby these methods can be compared to each other in their ability to quantify the circuit's frequency behavior as well as in their impact upon the success of the evolutionary process. The target filter specifications are given in terms of the abstract description presented in section 7.1.1.2. Therefore, Table 7.6 summarizes the

Parameter	f_{start}	f_{stop}	f_{PB}	f_{SB}	A_{PB}	δ	A_{SB}
Target Value	1 kHz	0.5 ... 1 MHz	10 ... 80 kHz	200 kHz	0 dB	0 dB	40 dB

Table 7.6: . Parameter set describing the LPF-task. These parameters are defined in Fig. 7.2(a).

parameters that are used to define the magnitude response in the sense of Fig. 7.2(a): The candidate circuits are tested in a frequency band ranging from 1 kHz to 500 kHz or 1 MHz for the step response based method M1 and the ac-sweep based methods M2,M3, respectively. The transition band starts at $10-80$ kHz depending on the experiment and ends at 200 kHz. Samples in the transition band are ignored in evaluating the magnitude response. As was already indicated in the description of the respective implementations of method M1 to M3, the experiments sought a passband gain of $A_{PB} = 0$ dB and a stopband attenuation of $A_{SB} = 40$ dB.

7.3.1 Experimental Setup

7.3.1.1 Overview of the Experiments

The difficulty of the LPF-evolution task is varied by means of the passband edge P_{PB}. Table 7.7 lists the resulting five experiments that solely differ therein. The increasing level of difficulty caused by the increase in the passband edge frequency is demonstrated by means of the minimal filter orders required to comply with the target magnitude response for different filter approximations. The minimal filter orders are obtained from MATLAB for an allowed passband ripple of 1 dB. Accordingly, the requirements of experiments 1 to 5 necessitate Butterworth filters of the orders 2 to 6. As the minimal order for all filter approximations is found to be ≥ 2, the problem is – at any rate – non-trivial.

Exp. No.	1	2	3	4	5
Transition band f_{PB}-f_{SB} [kHz]	10-200	20-200	40-200	57-200	80-200
Min. order: Butterworth	2	3	4	5	6
Min. order: Chebyshev I	2	2	3	4	4
Min. order: Chebyshev II	2	2	3	4	4
Min. order: Elliptical	2	2	3	3	3

Table 7.7: Transition band ranges for the five different experiments of each series listed in Table 7.8. The last four rows denote the minimal orders for the four different filter approximations Butterworth, Chebyshev I and II and Elliptical that are necessary to satisfy the specifications defined by each of the five experiments. The filter orders are based on a maximum passband ripple $\delta \leq 1\,\mathrm{dB}$.

The five experiments of Table 7.7 are carried out five times in five different *series* of experiments, which are detailed in Table 7.8. The paramount difference between the five series is given by the

No.	Gen. size	Geometry	Fitness criterion
1	10,000	I (Fig. 7.9(a))	$F_{\mathrm{step}\downarrow} + F_{\mathrm{step}\uparrow} + F_{\mathrm{trans}}$
2	50,000	II (Fig. 7.9(b))	$F_{\mathrm{step}\downarrow} + F_{\mathrm{step}\uparrow} + F_{\mathrm{trans}}$
3	10,000	I (Fig. 7.9(a))	$F_{\mathrm{acs}} + F_{\mathrm{trans}}$
4	10,000	I (Fig. 7.9(a))	$F_{\mathrm{acs+d}}:\ w(k) = 1\ \forall\, k$
5	10,000	I (Fig. 7.9(a))	$F_{\mathrm{acs+d}}:\ w(k) = \begin{cases} 0.1 & \text{if } k < k_0 \text{ and } f_{k_0} \in \mathrm{PB} \\ 0 & \text{if } k < k_0 \text{ and } f_{k_0} \in \mathrm{SB} \\ 1.0 & \text{if } k = k_0 \\ 0.01 & \text{if } k > k_0 \end{cases}$

Table 7.8: The five experiment series for the evolution of LPFs.

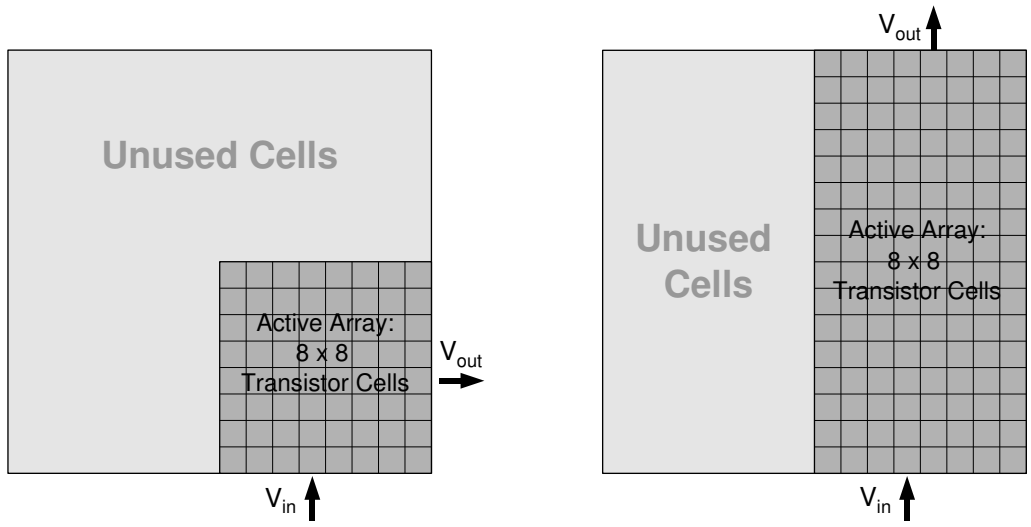
method of establishing the magnitude response. While series 1 and 2 rely on method M1, the analysis of a step response, series 3 is based on computing the magnitude response from the mean signal power, i.e. method M2. In both cases, the fitness is calculated as the sum of the magnitude and transient response(s). In series 4 and 5 on the other hand, the magnitude response is evaluated by means of a Fourier analysis of the circuit response to six sinusoidal input tones of different frequencies. Since the resulting frequency spectrum contains additional information on the dc component as well as on the harmonic distortions caused by the circuit under test, a transient test in the manner of TM 8 has been forgone.

Series 4 and 5 only differ in the weight factors that are used in (7.40) and (7.41) for the different frequency components obtained from the Fourier transform. Therefore, and in accordance with Table 7.8, the resultant spectrum is divided into three types of components: Those f_k that are below the fundamental frequency f_{k_0} defined by the input tone, the fundamental frequency itself, and all components exceeding the f_{k_0}. In series 4, all $w(k)$ are set to 1, which greatly overemphasizes the dc and distortion terms at the expense of neglecting the desired magnitude response itself. Thus, a more appropriate set of weight factors $w(k)$, which accounts for the different component numbers and importance of the three frequency domains, has been employed in series 5; it is detailed in respective entry in Table 7.8. As a further improvement, the Fourier components below the fundamental frequency f_{k_0} are only considered for those input tones taken from the passband. This constraint is imposed by the actual realization of the test procedure: Each of the six input tones is encapsulated in an own test mode. As the order of test mode application is randomized and the substrate is reset

between subsequent candidate circuits (cf. section 7.2.6), the time granted to the potential *lowpass* filters to settle to their desired dc output (2.5 V) is not sufficient for the shorter measurement times inherent to the test for higher fundamental frequencies, especially those from the stopband.

7.3.1.2 Geometrical setup

The two types of geometrical setups that are used for the five series of experiments summarized in Table 7.8 are depicted in Fig. 7.9. They differ in the fraction of the array of programmable transistor



(a) Geometry I (b) Geometry II

Figure 7.9: Geometrical setup for the evolution analog filters: Geometry I (a) allots a quarter and geometry II (b) half of the PTA to the EA.

cells granted to the EA and in the direction of the signal flow. The larger active array provided by geometry II is only used for series 2, which differs from series 1 solely in the resources allowed to the EA: By allowing the EA to use twice the number of transistor cells in conjunction with running it for five times the number of generations alloted to the experiments of series 1, the influence of available resources can be estimated from a comparison of both series.

7.3.1.3 Genetic Algorithm

Throughout all series of experiments, the same type of genetic algorithm has been used that has already been used for the experiments presented in Chapter 5 and 6, the only difference being the selection scheme: Here, a linear rank based selection scheme is employed instead of the truncation selection favored in the preceeding chapters. Using (2.2a) for $s = 2$, the probability for an individual with rank r to be selected for a mutation or crossover operation is given by

$$p_r = \frac{2r}{\mu(\mu-1)} \quad \text{for} \quad 0 \leq r < \mu \ , \tag{7.66}$$

where $\mu - 1$ denotes the size of the population and higher ranks are assigned to individuals with higher fitness. Accordingly, the worst individual of each generation will be precluded from any participation in the creation of the next generation.

The parameters customizing the GA are summarized in Table 7.9. While the second and third

column refer to the experiments targeted at LPF circuits, those employed for the evolution of HPFs presented in section 7.5 are listed in the remaining columns. Differences are only found in the number of generations, the number of runs, and the fraction of best individuals directly promoted to the next generation.

7.3.1.4 Analysis and Verification Tests

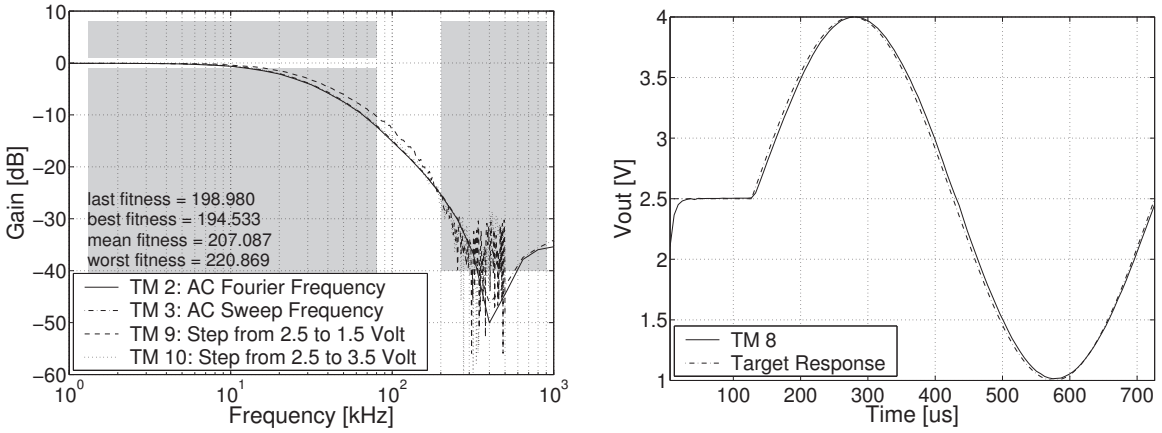
For each of the five experiments of each of the five different series, 30 runs have been carried out. The fitness results have been re-evaluated outside the evolution loop using the complete set of 14 (20) test modes listed in Table 7.4. Thereby, each evolved circuit was tested 100 times. In order to visualize the actual circuit response to the different test stimuli, the according results are recorded for the first, 50^{th} and last of the test. If not denoted otherwise, the results obtained during the first test are used for plotting these circuit responses. The candidate circuits are characterized by the fitness measures defined in Table 7.4 as well as by partial summations thereof, as e.g. the fitness criteria used during the evolution process listed in 7.8. The definitions of Table 6.5 on page 165 are applied to these different figures of merit to define the min last, min mean and min worst fitness values, where in this case the run index r takes on the values 1 to 30 and the index of tests t runs again from $t = 1, \ldots, 100$.

7.3.2 Illustration and Discussion of the Different Test Modes

To further explain the different types of test modes used to test the evolved circuits in general and to compare the different methods of establishing the magnitude response in particular, the output behavior for all non-redundant test modes of Table 7.4 are plotted in Fig. 7.10 and 7.11 for the min best circuit of experiment 5 of series 1. The different magnitude responses obtained from test modes 2,3,9 and 10 are depicted in Fig. 7.10(a). In this plot, as well as in all of the plots depicting a filter circuit's magnitude response that follow until the end of this chapter, the target specifications are visualized by gray-shading the regions the magnitude response is to avoid. This, in principle, resembles the presentation chosen for Fig. 7.3, yet illustrates the passband by $G_{\text{tar}}(f) = 0 \pm 1\,\text{dB}$ instead of the interval $[G_{\text{tar}}(f) - \delta, G_{\text{tar}}(f)]$ used in Fig. 7.3.

GA Parameter	LPF	LPF, var. f	HPF: Exp. 1-2	HPF: Exp. 3	HPF: Exp. 4
population size	50	50	50	50	50
# of generations	**10,000**	**50,000**	**100,000**	**100,000**	**100,000**
# of runs	**30**	**10**	**20**	**20**	**20**
selection scheme	rank-based	rank-based	rank-based	rank-based	rank-based
reprod. fraction	**0.1**	**0.1**	**0.04**	**0.2**	**0.1**
mut. fraction	0.9	0.9	0.9	0.9	0.9
mut. rate Term. Con.	1%	1%	1%	1%	1%
mut. rate W,L	2%	2%	2%	2%	2%
mut. rate Routing	1%	1%	1%	1%	1%
X-over fraction	0.9	0.9	0.9	0.9	0.9
X-over rate	1%	1%	1%	1%	1%
X-over block size	4	4	4	4	4

Table 7.9: Genetic algorithm parameters used for the evolution of analog filters. Only the lines printed in bold contain parameters varied in the course of experiments.



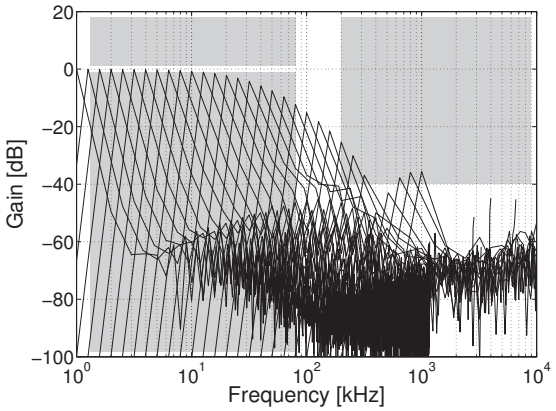
(a) TM 2,3,9,10 (b) TM 8

Figure 7.10: Output behavior of the *best* filter circuit of series 1, experiment 5 for all non-redundant test modes characterizing the frequency response and the transient analysis used during the evolution process. Here, *best* refers to the fitness criterion used during evolution listed in Table 7.8, that is the sum of the error contributions of test modes 8-10.

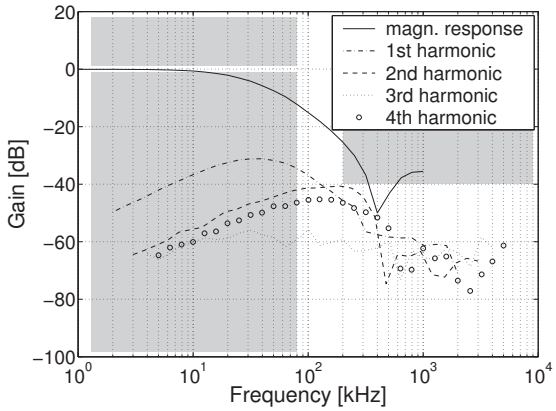
Firstly, it can be observed that the four magnitude responses deviate only little from each other. Thus it can be inferred that the three different ways of establishing the magnitude response of the filter circuits are consistent. Secondly, the step-based magnitude responses obtained from test modes 9 and 10 exhibit a noise level of up to $-30\,\mathrm{dB}$, which may prevent the GA from achieving a higher stopband attenuation. In accordance with the results of the noise analysis presented in section 7.2.4.4, the magnitude response curves attained from test modes 2 and 3 corresponding to methods M3 and M2, respectively, exhibit a considerably smaller noise level. A comparison of the input signals fed into the Fourier transform for methods M1 and M3 depicted in Fig. 7.11(f) and Fig. 7.11(c), respectively, impressively illustrates that this is due to the different SNRs[26]: While the derivative of the step response shown in Fig. 7.11(e) obtained from the post-processing operation $\mathscr{T}_E$ defined in (7.49) reveals a clearly visible noise contribution, the response to the ac-sweep of Fig. 7.11(c) is practically free of noise on the plotted scale. Thirdly, the plots of Fig. 7.10 show that the best circuit evolved for series 1, experiment 5 does work properly as a LPF, albeit with an imperfect frequency response: On one hand, the evolved filter does not meet the desired target attenuation for all of the tested frequencies in the stopband. On the other hand, it cuts corners in the vicinity of the transition band, because it fails to achieve the necessary steepness in the rolloff.

The set of non-redundant test modes not used during the evolution of series 1–3 is visualized for the best circuit of experiment 5 of series 1 in Fig. 7.11. Test mode 1 captures the nonlinearity of the transfer function of the filter under test according to (7.43) and (7.44). The measured magnitude responses for all of the 31 fundamental frequencies f_{k_0} are depicted in Fig. 7.11(a), where the dc component of the respective Fourier transform is arbitrarily set to $-100\,\mathrm{dB}$ for better readability. The resulting magnitude response as well as the first four harmonics are also shown in Fig. 7.11(b): While the first harmonic exceeds the threshold of $-40\,\mathrm{dB}$, the harmonics 2 and 4 get close to $-40\,\mathrm{dB}$ within the transition band. It is left to the third harmonic to rest almost flatly at $-60\,\mathrm{dB}$, which may indicate the underlying noise floor. According to section 7.2.4.4, this would suggest a relative precision of approximately 10 ENOBS.

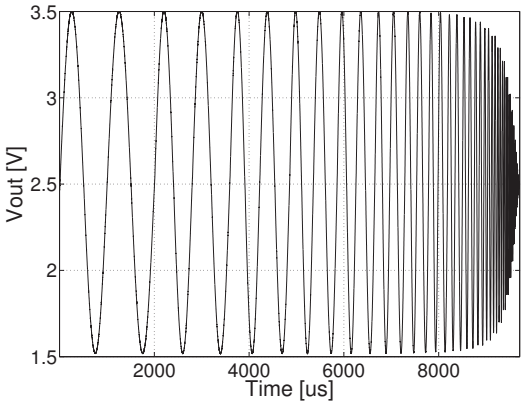
[26] Signal to Noise Ratios



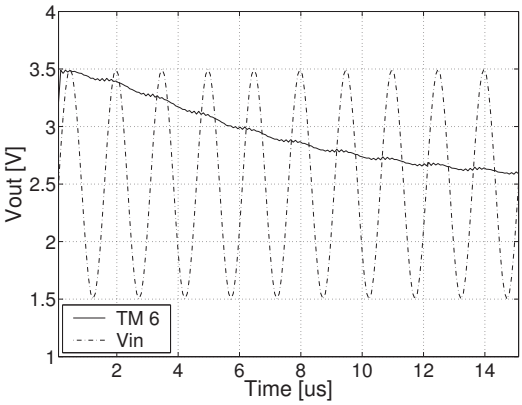
(a) TM 1



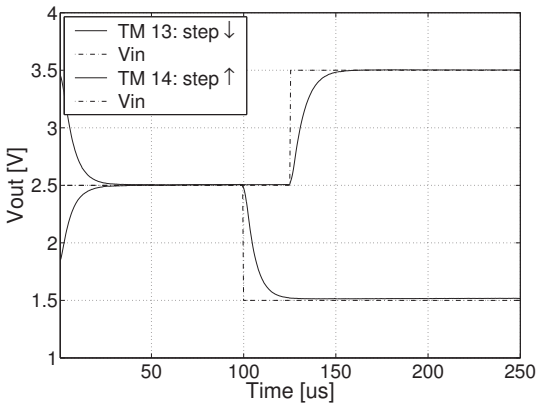
(b) TM 1: fundamental and first four harmonics



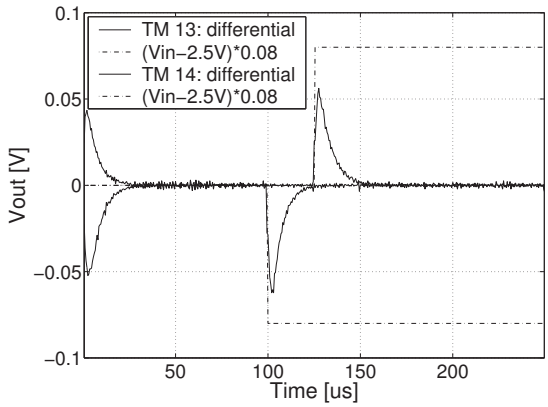
(c) TM 4



(d) TM 6



(e) TM 13,14



(f) TM 13,14 differential

Figure 7.11: Output behavior of the *best* filter circuit of series 1, experiment 5 for the non-redundant test modes not contained in Fig. 7.10. Again, *best* refers to the sum of the error contributions of test modes 8-10. In (a), the dc components of all magnitude responses are set to $-100\,\mathrm{dB}$ for better readability.

Fig. 7.11(c) and Fig. 7.11(d) visualize the transient response of the evolved filter for the full frequency sweep used for test modes 1–5 and for a sinusoidal input tone of 1 MHz, respectively. In case of the sweep, the expected output behavior can be observed: Signals of lower frequency pass the filter, whereas those of higher frequency are attenuated, while the output is kept at a dc-level of 2.5 V. In case of the 1 MHz sine wave, the output of the filter does correctly suppress the input signal and slowly converges to the desired dc-level.

The transient responses to similar input steps as those used for test modes 9 and 10 are illustrated in Fig. 7.11(e). The steps are applied after 100 and 125 µs for the down- and upward step, respectively to allow the circuit under test to settle to the desired dc-level beforehand. Taking into account the different time scales, the exponential-like changes of the output resembles that one observed in Fig. 7.11(d). In order to demonstrate the lower SNR inherent to the procedure M1 of establishing the magnitude response compared to method M3, the discrete derivative according to $\mathscr{T}_E$ from (7.49) is applied to the step responses shown in Fig. 7.11(e) to obtain Fig. 7.11(f).

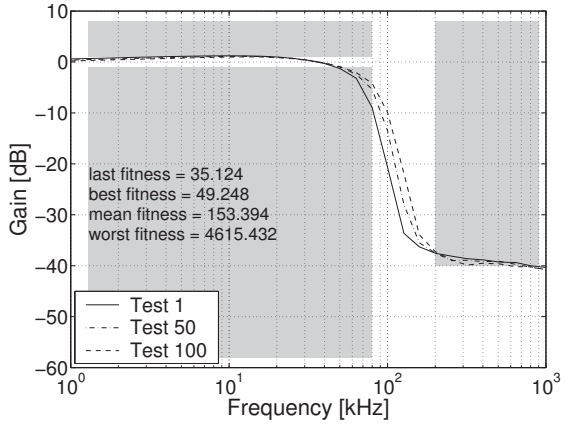
7.3.3 Reproducibility

In the preceeding chapters 5 and 6 it was pointed out that the performance of the evolved circuits could be reproduced within the 100 verification tests for the majority of evolved circuits. As can be seen from Fig. 7.12(d), this is not necessarily the case for the evolved LPFs. Fig. 7.12(a) and Fig. 7.12(b) compare the output characteristics for the first, 50^{th} and last of the 100 verification tests for the min best circuit obtained from experiment 5 of series 3. Apparently, the according min worst circuit, whose output characteristic is depicted in Fig. 7.12(c) and Fig. 7.12(d) is a different one, because the worst fitness of the min best circuit amounts to more than a hundred times the best fitness value, as indicated by the imprints of Fig. 7.12(a) and Fig. 7.12(c); the worst fitness of 4615 corresponds to almost completely failing the desired filter task, as the average fitness of a random circuit amounts to approximately 9540. Yet, this discrepancy can not be explained by the three different output characteristics plotted, since both the according magnitude and transient responses differ only little from each other (and despite the offset in the transient response would rather attain fairly low fitness values).

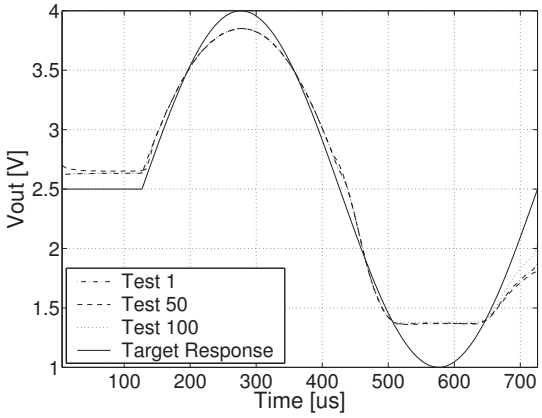
Finally, it has to be pointed out, that the min best circuit of experiment 5, series 3 reliably manages to solve the filtering task defined by experiment 3, that is to achieve a transition from $\geq -1\,\text{dB}$ to $\leq -40\,\text{dB}$ within the frequency range of $40 - 200\,\text{kHz}$. According to Table 7.7, this requires at least a 3^{rd} order filter for the classical filter approximations and even a 4^{th} order filter in case of the Butterworth approximation. The only drawback of the evolved circuit is given by its degree and range of linearity: From Fig. 7.12(d) the filter can be expected to be almost linear only in the range of $2 - 4\,\text{V}$.

In order to obtain a more comprehensive account of the stability of the evolved LPF circuits, the distribution of the percentage deviation of the last from the best fitness (a), the worst from last fitness (b) and the worst from the best fitness (c), (c) are presented by means of histograms in Fig. 7.13. Therefore, the relative deviations of all experiments of one series are aggregated to form one histogram per series that comprises a total of 150 runs. Please note, that for the remainder of this chapter, all histograms are organized such that their rightmost bin gathers all runs exceeding the value of the second highest bin. Fig. 7.13(a) to 7.13(c) use the *native* fitness criterion of the respective series, that is, the fitness criterion used during the evolution process.

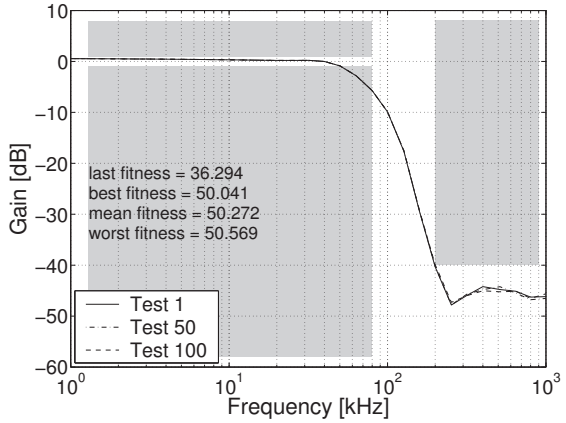
The deviation of the last from the best fitness values basically complies with the expectation in that most of the runs perform better than at the end of the evolution process at least once, if tested 100 times. Only a small fraction degraded between the time of evolution and that of the verification tests, of which most runs deviate only by less the tens of percent. Interestingly, the distribution of fitness



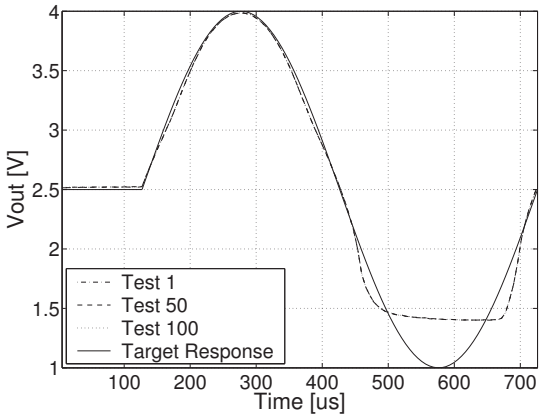
(a) TM 2 for the min best



(b) TM 8 for the min best



(c) TM 2 for the min worst

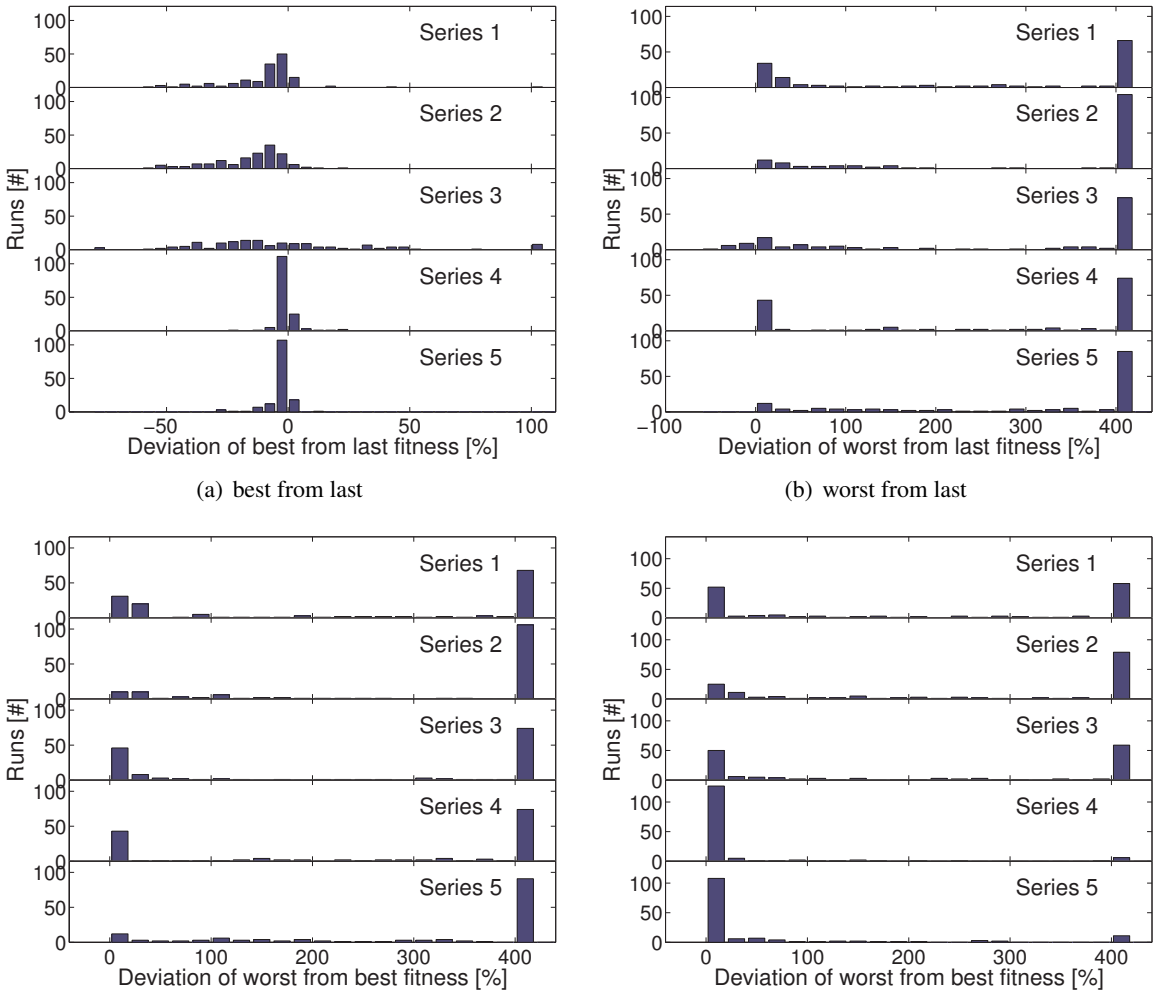


(d) TM 8 for the min worst

Figure 7.12: Comparison of the frequency (a), (c) and transient (b), (d) response for three different measurements, namely the first, 50th and last of 100 verification tests. (a) and (b) depict the behavior of the circuit with the minimum error among the best results for all runs; for (c) and (d) those runs possessing the minimum error in a comparison of the worst results of the 100 verification tests are chosen.

deviations are wider for series 1-3 than for series 4 and 5. A large fraction between appoximately 40 and 70% of the evolved circuits attain a worst fitness exceeding the last or best one by more than a factor 5, i.e. sort of fail to reliably reproduce their desired behavior. For the runs of series 4 and 5 this situations dramatically changes, if the best and worst fitness are calculated as the sum of test modes 1-3 and 9-10 [27] , which includes all types of measures for the magnitude response plus the evaluation of the distortion by test mode 1: Here, for more than 70% of the evolved circuits the difference between the best and the worst fitness measured within the 100 verification tests amounts to less than 20%. First, this suggests that the filter circuits evolved by means of method M3 work more reliable. Second, the different behaviors between the native fitness and that used in Fig. 7.13(d) must be explained: As was observed from Fig. 7.11(d), the dc level of the output of the circuit under test and therefore of at least some of its inner nodes may be changed between two subsequent test modes by the analog reset

[27] A transient analysis with a sine wave stimulus taken from the passband as e.g. test mode 7 or 8 could not be included, because some of the evolved filters obtained from series 4 and 5 invert the polarity of the input signal at the output; a transient response to penalize this was forgone in the design of the experiment and for the evolution of the magnitude response only, an additional phase shift of 180° does not matter, but increases the feasible design space.



(a) best from last

(b) worst from last

(c) worst from best

(d) worst from best for test mode 1-3, 9, 10

Figure 7.13: Histograms for the deviations of the best from the last fitness values (a), the worst from the last (b) and the worst from the best (c) based on the native fitness criterion use for the respective series of experiments. In contrast, (d) illustrates the fitness deviations of the worst fitness values from the last ones expressed by the fitness criterion of series 1.

and/or a drift effect. For fundamental frequencies f_{k_0} taken from the stopband, it takes more than 1 cycle of the sinusoidal input to recover from this excitation due to the intrinsic lowpass behavior of the filter under test. Thus, the circuit may not be able to reach a proper operation point necessary to perform the desired task correctly and hence may be misevaluated. Since the above scenario exactly describes the situation created for the evaluation of the magnitude response according to method M3, i.e. by test modes 15-20 in Table 7.4, the EA may be deluded by bad evaluation results, and verification tests may fail to determine the correct fitness of the circuit under test. This problem could be avoided most thoroughly by using a sweep similar to test mode 2 instead of test modes 15-20, because a sweep starting at low frequencies f_{k_0} would allow the circuit to settle to its operation point within one cycle of the lowest frequency chosen from the passband.

Despite all efforts to prevent the EA from abusing the initial state of the FPTA substrate prior to the evaluation measurements, namely the substrate reset and the randomization of the test mode order (see section 7.2.6), a large number of evolved filters are found to be unstable. However, the effect is

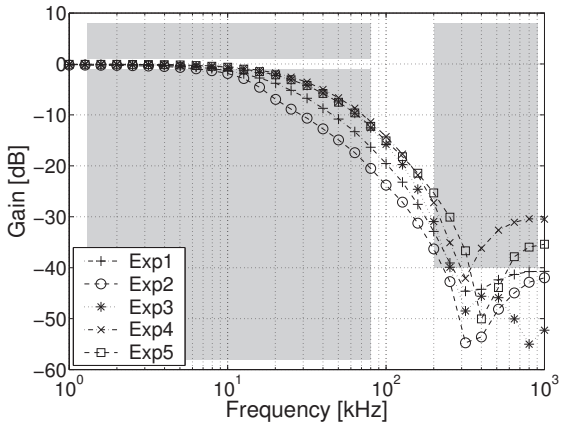
more severe for series 1–3 than for series 4 and 5. Different arguments lend themselves to a possible, albeit not finally conclusive, explanation: First, Fig. 7.13 does not contain any information about the absolute fitness values achieved by the respective runs. The differences between the distribution of deviations of series 1–3 and 4–5 therefore may be due to the fact that the circuits evolved in series 4 and 5 are inferior to those of series 1–3. Accordingly, the better circuits of series 1–3 possess a higher potential to fail. In fact, as will be discussed in section 7.3.5, this is indeed the case and may explain at least some of these differences. Second, the circuits evolved in series 2, which differs from series 1 solely by the increased number of generations alloted to the EA, seem to be even more likely to unstable behavior. This phenomenon may be explained by an increase in premature convergence, where large parts of the population are genetically similar and phenotypically identical, which implies that the dominating genotype is evaluated many times within one generation. Thus, a failure in coping with some initial states of the substrate is masked by the large number of trials, of which at least some are allowed to start with more favorable conditions. The fairly large number of 5 individuals that are promoted into the next generation (cf. Table 7.9) accelerates and supports this decrease in selection pressure towards stable filter circuits. In this vein, the higher likeliness of circuits evolved within series 4 and 5 to be forced to a detrimental operation point in between two consecutive test modes may increase the selection pressure towards being more independent thereof.

7.3.4 Comparison of Different Experiments

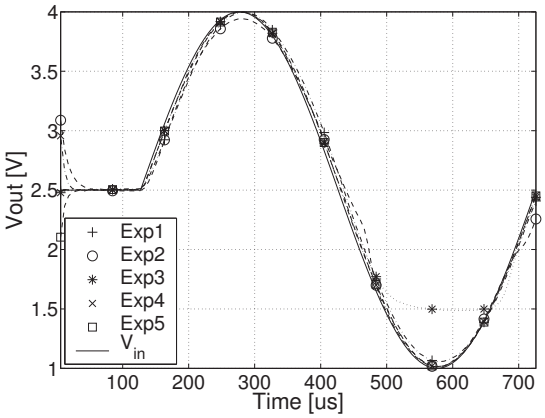
Within the previous two sections Fig. 7.10(a) and Fig. 7.12(d) indicated that even the best filters evolved in series 1 and 3 did not meet the specification of experiment 5. This raises the question, to which extent the actual specification of the higher passband edge influences the resulting performance of the evolved circuits. In order to attain a *qualitative* impression thereof, the output behavior of the best-of-experiment solution is compared for series 1,3 and 5 in Fig. 7.14 (the according plots for series 2 and 4 are omitted for brevity; the actual selection covers experiments carried out for all three types of the evaluation methods M1 to M3.). The best circuits are determined with regard to the min worst native fitness attained within the 100 verification tests. The plots on the left hand side of Fig. 7.14 present the magnitude responses of the best-of-experiment circuits evaluated according to test mode 1, that is by means of the Fourier analyzed ac-sweep. The plots on the right hand side of Fig. 7.14 illustrate the transient response to a sinusoidal input tone with a frequency taken from the passband. In case of series 1 and 3 (plot (b) and (d)) test mode 8 is utilized; for the transient response of the according circuits of series 5 test mode 7 is preferred, since the circuits evolved within series 4 and 5 were only required to be linear in the input voltage range of $1.5 - 3.5\,\mathrm{V}$.

First, all of the 15 output characteristics plotted in Fig. 7.14 are indeed viable LPFs, which are almost linear over at least a large fraction of the desired range of $1 - 4\,\mathrm{V}$. Second, throughout all three series, the magnitude responses obtained for the best solutions of the respective experiments differ only little, especially for experiments 3–5. In greater detail, experiments 1–3 of series 1 comply most closely with the expected behavior, in that their magnitude response starts to drop at higher frequencies and with a steeper rolloff for the higher passband edges located at higher frequencies. A similar trend can be seen for series 3. In case of series 5, the magnitude responses look even more alike and seem to be merely shifted downward for wider transition bands. In conclusion, the best-of-experiment solutions of series 1,3 and 5 are phenotypically very similar. Yet, the more ambitious specifications seem to be more likely to yield better results in terms of a steep rolloff.

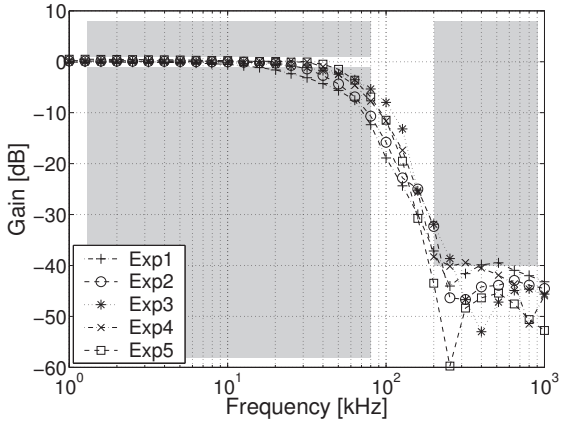
Quantitative Comparison of Different Experiments. For a more quantitative analysis of the evolution results for the different task difficulties, the data of all runs is summarized in the histograms depicted in Fig. 7.15 for series 1 and in Fig. 7.16 for series 2, 3 and 5. Again, series 4 is omitted for



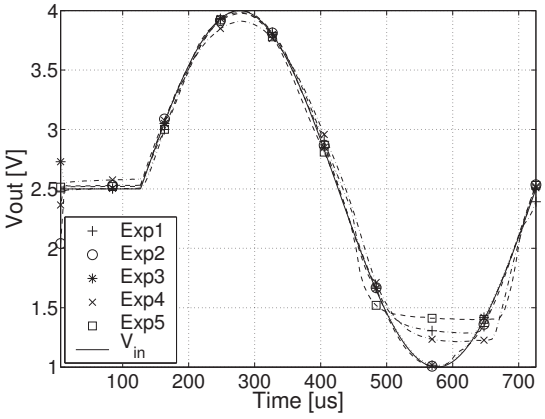
(a) Series 1: TM 2



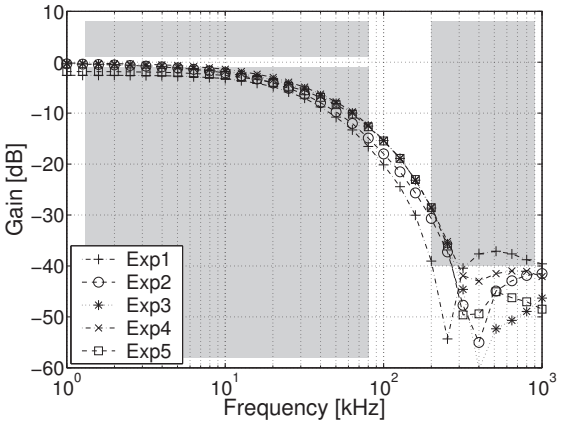
(b) Series 1: TM 8



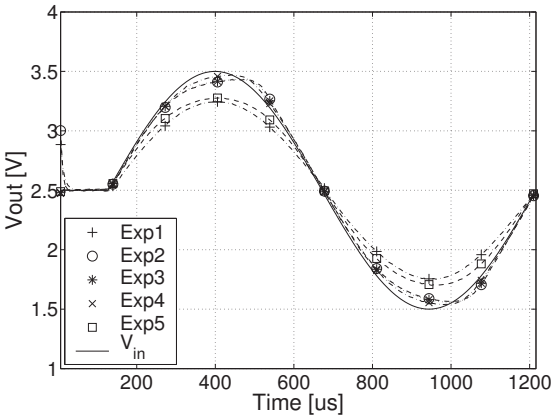
(c) Series 3: TM 2



(d) Series 3: TM 8



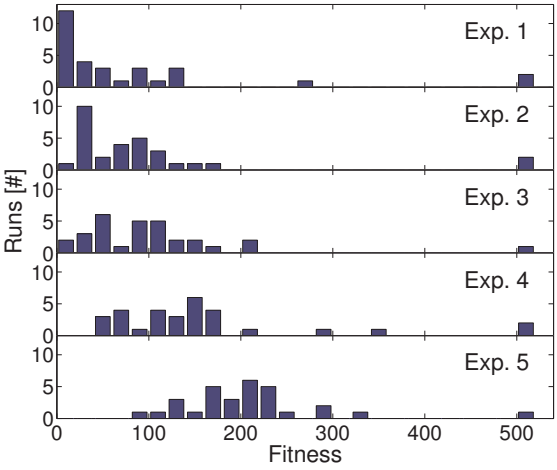
(e) Series 5: TM 2



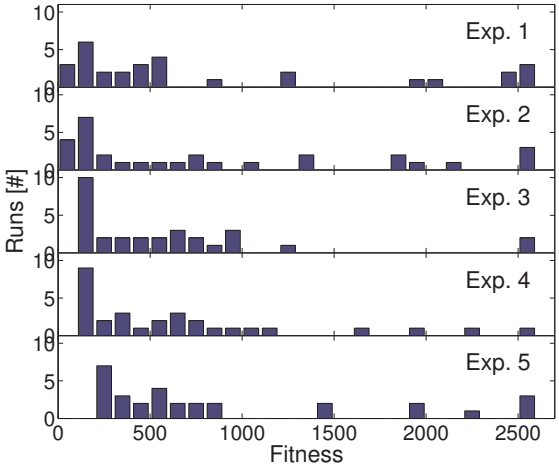
(f) Series 5: TM 7

Figure 7.14: Comparison of the output behavior of the *best* filter circuits of all five experiments of series 1,3,5 (from top to bottom). Here, *best* refers to the fitness criterion used during evolution listed in Table 7.8, that is the sum of the error contributions of test modes 8-10. (a) shows the magnitude response determined according to method M3 and (b) the transient response to a sine wave according to test mode 8.

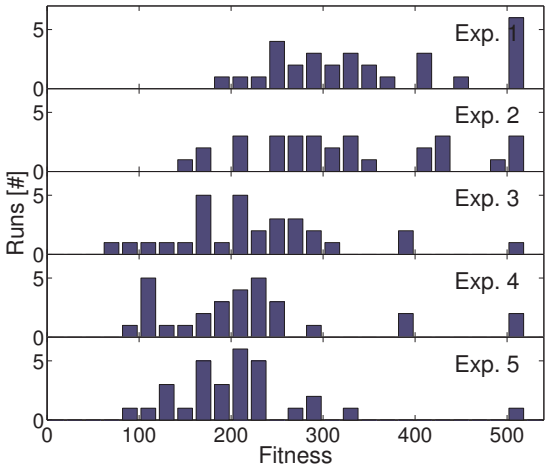
brevity, since its evaluation method is similar to that of series 5 and the according evolution results are inferior to those of series 5, as will be discussed in section 7.3.5. In Fig. 7.15 the histograms on the



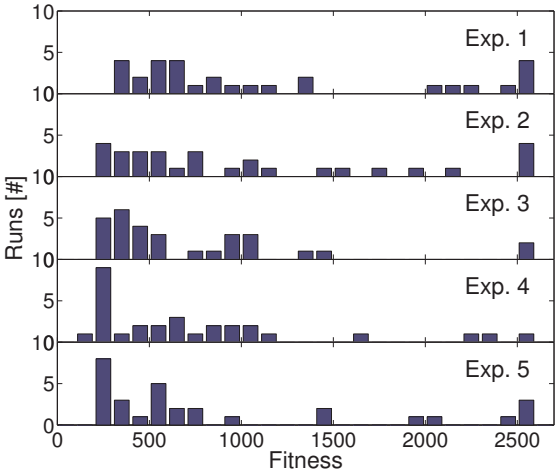
(a) best fitness for the native criterion



(b) worst fitness for the native criterion



(c) best fitness for the criterion of experiment 5



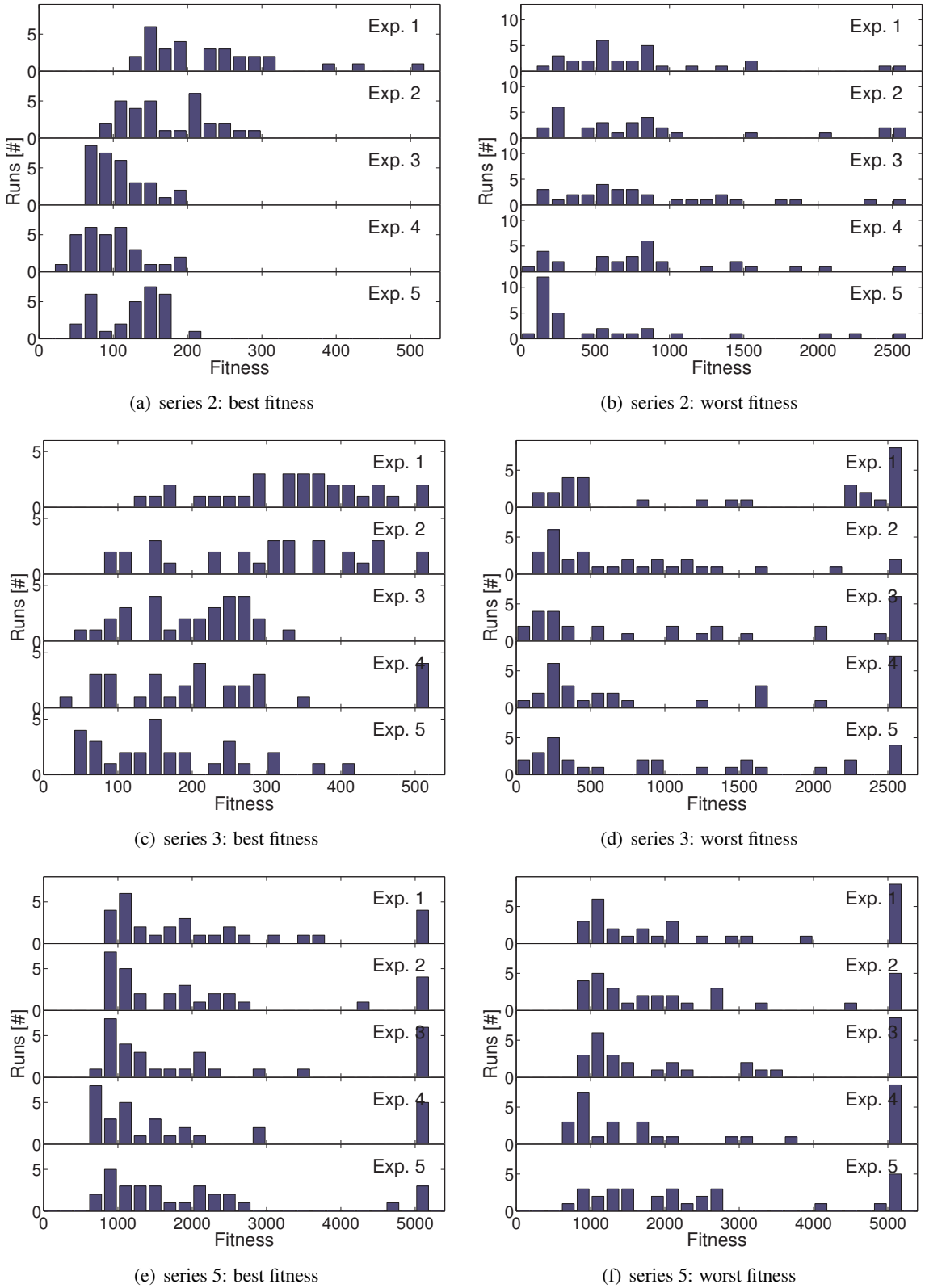
(d) worst fitness for the criterion of experiment 5

Figure 7.15: Fitness histograms for all experiments of series 1. While the histograms (a) and (b) are based on the fitness criterion used in the course of the experiments, the fitness criterion describing experiment 5 is applied for the generation of those shown in (c) and (d). (a) and (c) display the best fitness value attained in 100 verification tests, whereas (b) and (d) account for the respective worst values.

left hand side contain the results for the best and those on the right hand side those of the worst fitness obtained from the 100 verification tests. While Fig. 7.15(a),(b) use the fitness criterion native to each of the experiments, the data plotted in Fig. 7.15(c),(d) uses the fitness measure native to experiment 5.

Besides the awaited discrepancy between the best and worst fitness values, the histograms in Fig. 7.15(a),(b) reveal that better results are obtained for the simpler tasks. On the contrary, evaluated by the standards of experiment 5, the circuits evolved with the more ambitious fitness criterion tend to perform better, as can be observed from Fig. 7.15(c),(d). Yet, the effect is stronger for experiments 1–3 than for a comparison of experiments 3–5.

Each row of Fig. 7.16 corresponds to the second row of Fig. 7.15. While Fig. 7.16(a)–(d),



(a) series 2: best fitness

(b) series 2: worst fitness

(c) series 3: best fitness

(d) series 3: worst fitness

(e) series 5: best fitness

(f) series 5: worst fitness

Figure 7.16: Fitness histograms for all experiments of series 2,3,5 (from top to bottom). The fitness criterion describing experiment 5, that is the sum of TM 8,11,12 for series 2, and that of TM 5,8 for series 3 are applied for the generation of those shown in (a) to (d). (e) and (f) are obtained from the sum of the test modes 1,5,11,12, which are all compatible with the experiment 5 specification.

which contain the according histograms for series 2 and 3, are based on the native fitness criterion of experiment 5 used during the evolution process, Fig. 7.16(e)–(f) employ the sum of test modes 1,5,11 and 12 to compare the results achieved for the five different experiments, where the latter test modes also capture the problem definition described by experiment 5. The native criterion is discarded because of its unnecessary low reliability identified in section 7.3.3.

For series 2 and 3, the results are similar to those obtained for series 1. The only exception being that for series 2 the best fitness values are attained for experiment 4 rather than for experiment 5. Here, the increased number of generation allotted to the EA seemed to be helpful to evolve better circuits for the task defined experiment 4, but did not help to find perfect solution satisfying the problem described by experiment 5. Unlike for all other series, for series 5, there is only a slight difference between the results obtained for experiments 1–3 and those obtained for experiment 4 and 5. In conclusion, however, the hypothesis that tighter constraints for the allowed magnitude response create better results in terms of the criterion of experiment 5 does hold up to a certain level of difficulty. In case of series 1, that is given by the specifications of experiment 3, for series 2 this is extended to those of experiment 4.

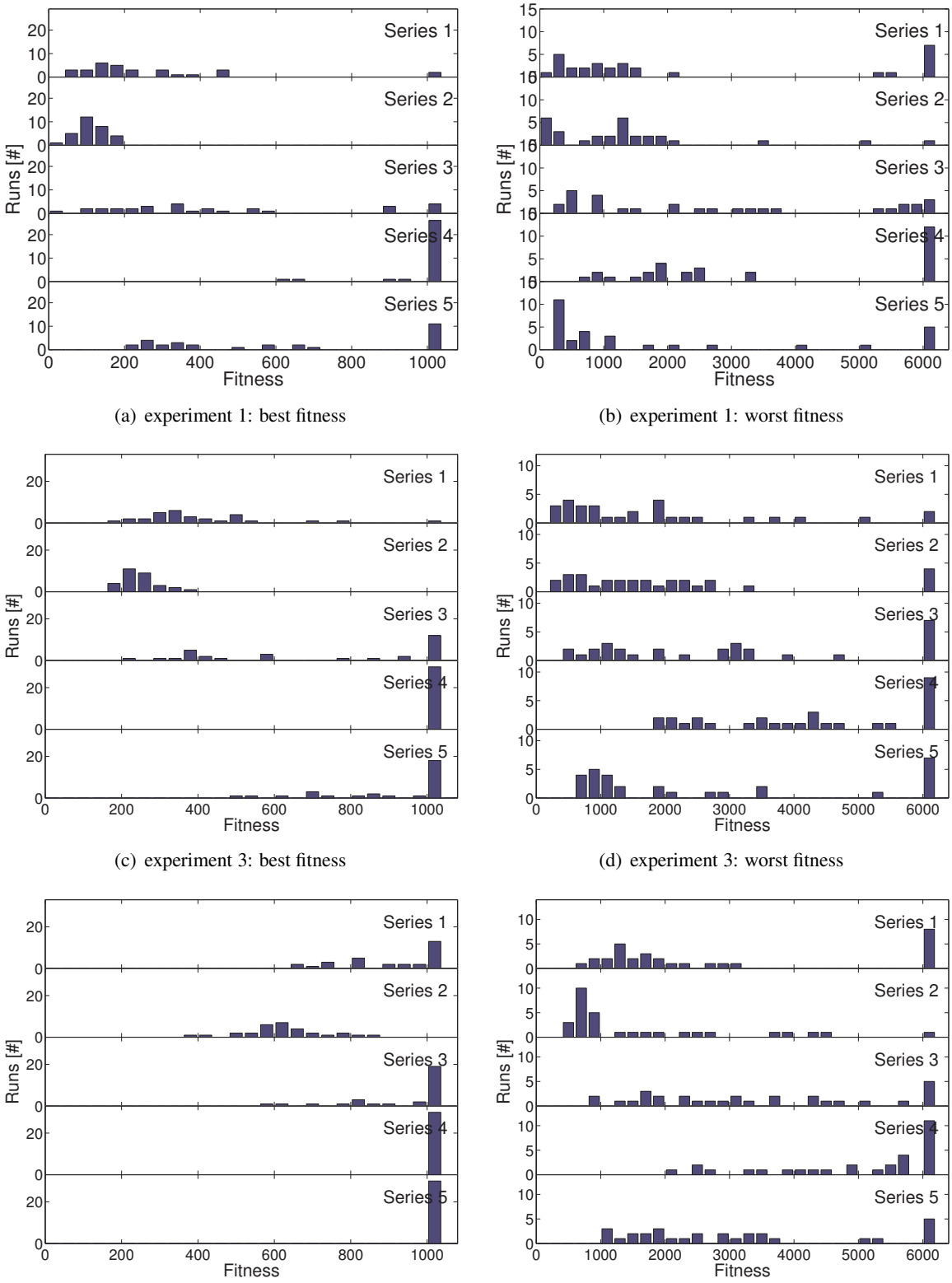
7.3.5 Comparison of Different Series of Experiments

To gain some insight into the influence of the evaluation method used to establish the magnitude response of the circuit under test, the data discussed above is analyzed a second time to allow for a comparison of the results of different series for the problem definition of each experiment. First, this comparison shall be done by means of the histograms of Fig. 7.17, which are similar to those of Fig. 7.16, yet with exchanged roles of series and experiments. According to section 7.3.3, the fitness for series 4 and 5 cannot be evaluated using any of the transient analyses of test mode 7 or 8, such that none of the native fitness criteria of either series 1,2 or 3 can be used for a fair comparison. Hence, the quality of the evolved circuits is again, as for Fig. 7.13(d) determined by the sum of test modes 1-3, 9 and 10. The plots on the left hand side of Fig. 7.17 depict the according histograms for the best and those on the right hand side those for the worst fitness values gained from this sum for the task definition of experiment 1, 3 and 5.

With regard to the best fitness, series 2 yielded – on average – the best results, followed by series 1 and 3 throughout all of the considered experiments. As expected from the preceeding sections, series 5 yields the second worst and series 4 the worst results if best fitness is concerned. The according histograms for the worst fitness confirm the general impression generated by their best fitness counterparts; yet, the differences between the different series prove to be smaller. The dominance of series 1 ,for instance, is reduced to the task described by experiment 5, and the results of series 1,3 and 5 move closer together, so that for experiment 1 the circuits evolved by series 5 seem to outperform those of series 1 and 3 on average. However, series 4 can still easily be distinguished as yielding the worst results.

Relation of Magnitude and Linearity. The histograms presented in Fig. 7.17 aggregate the quality of the magnitude response and the linearity of each circuit into one criterion, which overemphasizes the meaning of the magnitude response. The scatter plots shown in Fig. 7.18, however, relate these two qualities to each other.

Therefore, the fitness attained from test mode 1 – quantifying the distortion caused by the respective circuit – is plotted against that of test mode 2 – accounting for the magnitude response in one scatter plot per experiment. Each of the five plots of Fig. 7.18 gathers the data of all series for one experiment; yet, only those runs are considered whose penalty for the magnitude response does not exceed a value of 1250. The five plots correspond to experiments 1 to 5 when read from left to right,



(a) experiment 1: best fitness

(b) experiment 1: worst fitness

(c) experiment 3: best fitness

(d) experiment 3: worst fitness

(e) experiment 5: best fitness

(f) experiment 5: worst fitness

Figure 7.17: Comparison of the best (a),(c),(e) and worst (b),(d),(f) fitness values attained for the target specifications of experiment 1 (a),(b), experiment 3 (c),(d) and experiment 5 (e),(f) in all 5 different series. To allow for the comparison, the fitness is evaluated as the sum of the test modes 1-3, 9 and 10.

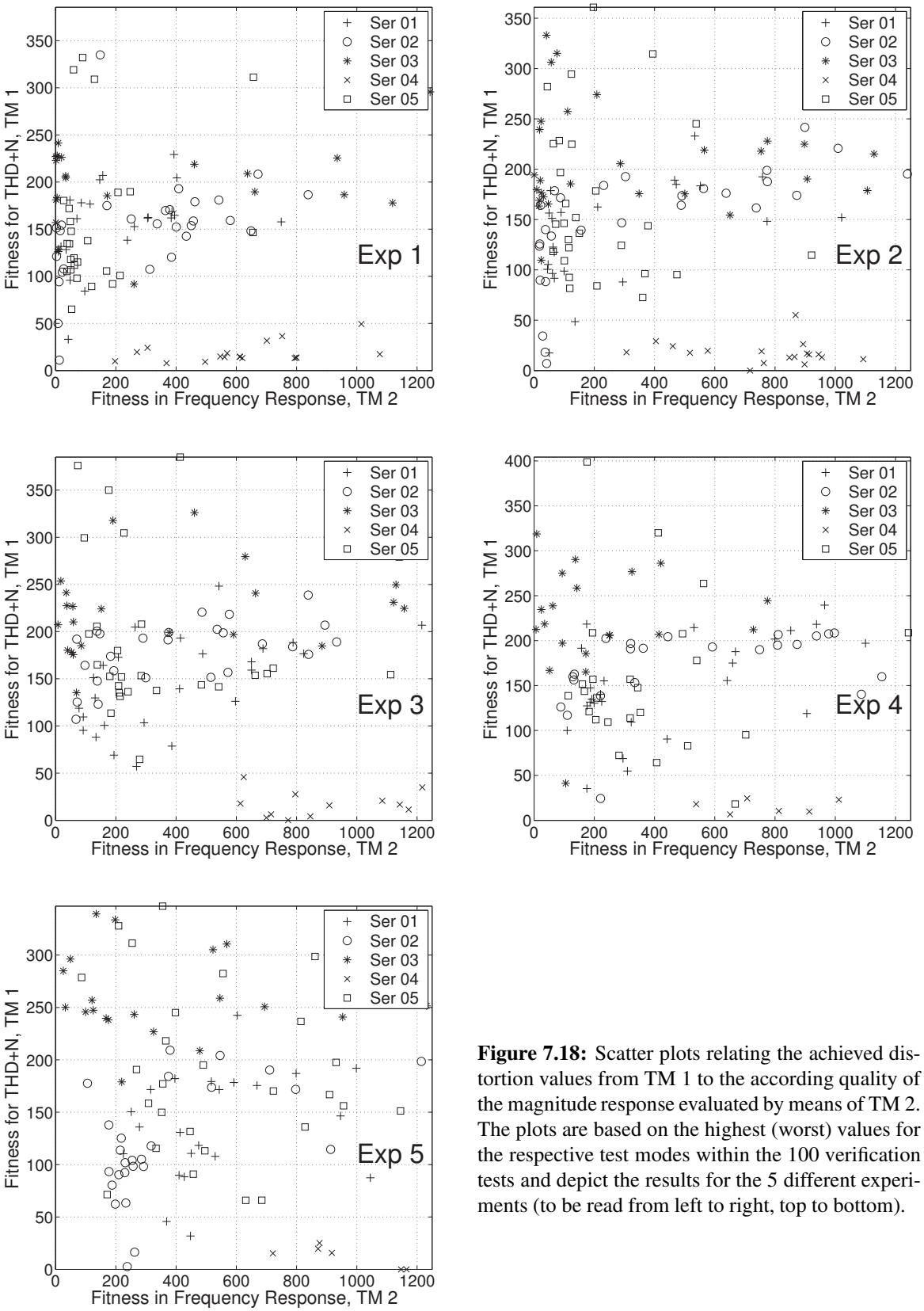


Figure 7.18: Scatter plots relating the achieved distortion values from TM 1 to the according quality of the magnitude response evaluated by means of TM 2. The plots are based on the highest (worst) values for the respective test modes within the 100 verification tests and depict the results for the 5 different experiments (to be read from left to right, top to bottom).

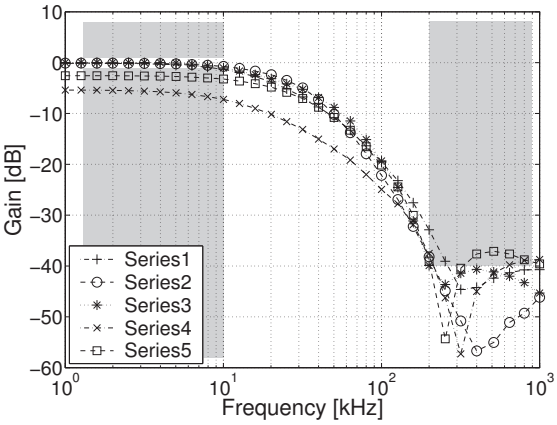
top to bottom. The actual fitness values ascribed to the two test modes are taken as the worst of 100 verification tests for each test mode.

The two opposite poles of the five distributions visualized in the five scatter plots of Fig. 7.18 are represented by those of the runs evolved within series 3 and 4. All of the runs plotted for series 4 exhibit very low, if not the lowest distortion values, albeit large penalties for their bad magnitude response. As many of them are not contained in the scatter plots, they are bound to possess an even worse magnitude response. All but one of the runs plotted for series 3, on the other hand, possess relatively high distortion values, where the run on the Pareto front of the plot of experiment 4 represents the exception. Yet, a large fraction of the run evolved in series 3 exhibits a very good magnitude response. In comparison, the runs of series 1,2 and 5 are scattered over the five plots more evenly. In most cases, the runs that are closest to the origin of the respective plot, stem from either of these three series, with series 1 and 2 being more successful than series 5.

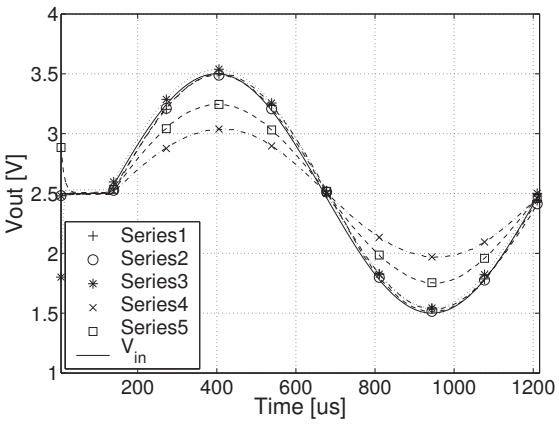
The fact, that the circuits evolved in series 4 are spread out more evenly than those of series 5 can be explained by the different weights assigned to the different frequency components obtained from test modes 15-20, which are stated in Table 7.8. In case of series one, all components, including those below the fundamental frequency f_{k_0} and all measured harmonics thereof, are weighted equal. Thus, the meaning of the non-fundamental frequency components is gravely overemphasized. In contrast, the weighting scheme chosen for series 5 seems to trade off of magnitude response and distortion in a more suitable fashion. Together with the fitness boost expected from a more reliable sweep-type measurement, the possibility of a user-defined trade-off between distortion and magnitude response, makes evaluation method M3 an ideal candidate for future experiments. Since both, series 1,2 and series 3 evaluate the transient response in addition to the different ways of establishing the magnitude response, it is rather surprising that their distributions differ so strongly. On the other hand, it is by all means conceivable that any nonlinearity in the voltage range covered by the two step responses will cause a detectable change in the frequency spectrum of the magnitude response. This would explain the difference in the distributions generated by methods M1 and M2 in terms of a selection pressure towards linear circuits in case of method M1.

Output Behavior of the Best of Series Solutions. Finally, the output characteristics of the five different series shall be compared exemplary for the best runs of experiments 1,3 and 5. Therefore, the according output responses for test mode 2 and test mode 7 are depicted on the left and right hand side of Fig. 7.19, respectively. The best circuits are determined by their min worst native fitness for series 1–3 and by the min worst result for the sum of test modes 1 and 2 in case of series 4 and 5.

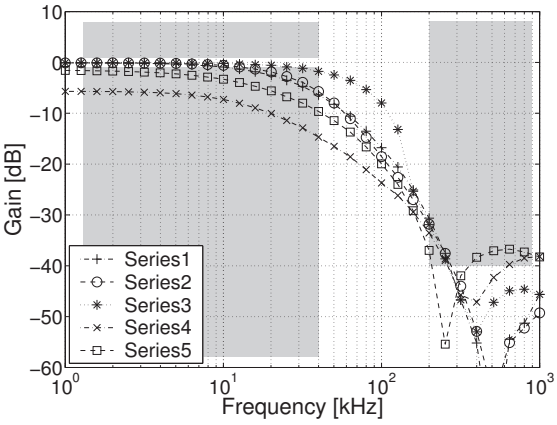
The best-looking magnitude response are obtained from series 3, whose best runs outperform those of all other series for the more difficult tasks described by experiments 3 and 5. Yet, in case of experiment 5, this is paid for by severe nonlinearities in the transient response for signals below 2 V (the output response of the best filter of series 3, experiment 5 was already detailed in section 7.3.3). Except for the latter circuit, all of the presented best-of-experiment solutions exhibit an almost perfectly linear transient response. Generally speaking, the best runs of series 1 and 2 perform second best in terms of the magnitude response, with a slight advantage for series 2, manifesting in experiments 1 and 5). The best runs of series 4 and 5 are the least successful in complying with the desired magnitude response specifications: First, they fail to reach the required passband gain by approximately 6 and 3 dB for series 4 and 5, respectively. Second, they show the least steep rolloff within the transition band in comparison with the best runs of the other 3 series. Nevertheless, a significant advantage in linearity cannot be observed from their transient responses, either.



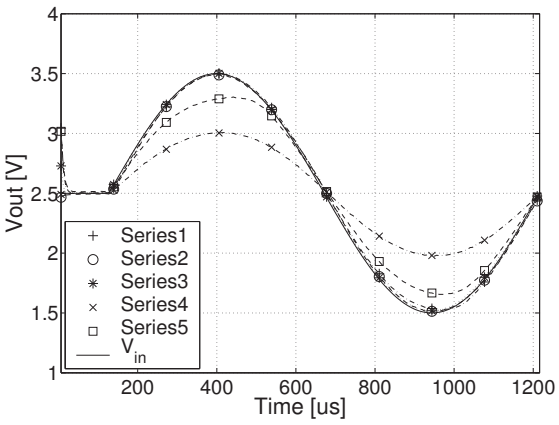
(a) experiment 1: magnitude response



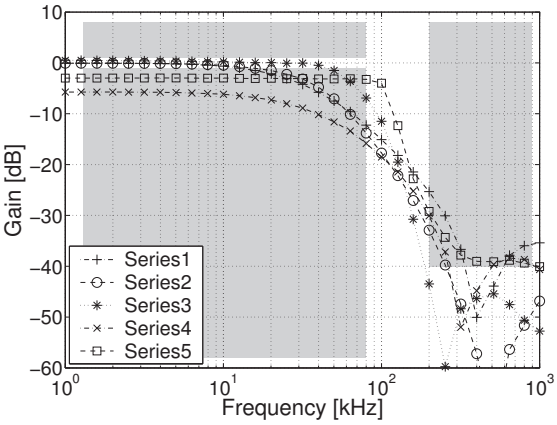
(b) experiment 1: transient behavior at 822 Hz



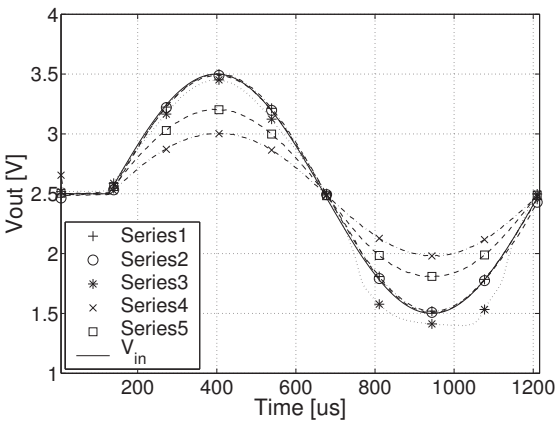
(c) experiment 3: magnitude response



(d) experiment 3: transient behavior at 822 Hz



(e) experiment 5: magnitude response

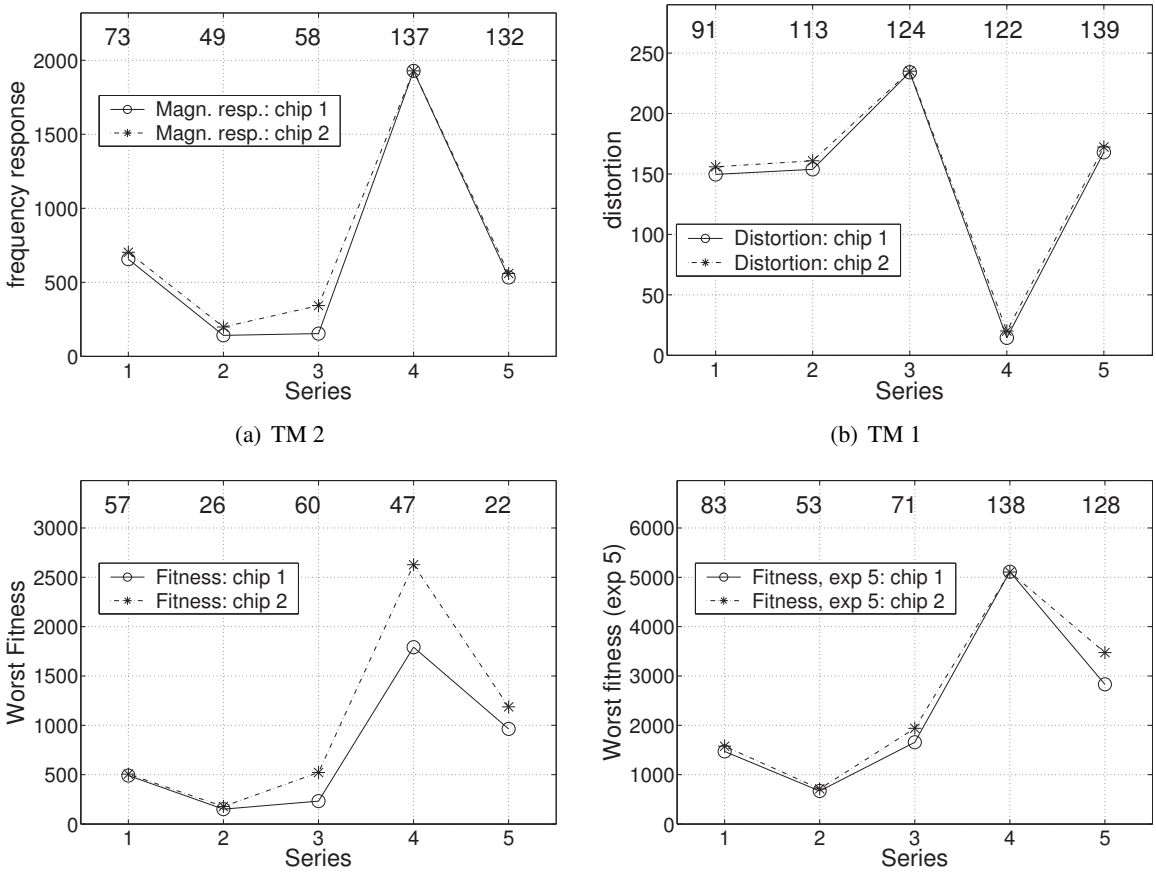


(f) experiment 5: transient behavior at 822 Hz

Figure 7.19: Left: Magnitude response (TM 2) and **right**: Transient response to a sinusoidal input stimulus (TM 7) for the specifications of the experiments 1,3 and 5 (top to bottom). Each plot illustrates the behavior of the best-of-series filter circuits. In case of series 1–3 best refers to the min worst native fitness obtained from 100 verification tests. For series 4 and 5 the minimum worst result for the sum of test modes TM 1,2 are used.

7.3.6 Migration to a Second Chip

The comparison of different methods of evolving LPFs concludes with a comparison of the test results achieved on a second chip with those obtained from the chip used during the evolutionary circuit synthesis itself. The latter one will henceforth be referred to as chip 1, while the former one is denoted as chip 2. The accomplish this, the average performance of the evolved circuit on both chips is compared for four different figures of merit, namely the magnitude response (test mode 2), the distortion penalty (test mode 1), the native fitness criterion, and the sum of test modes 1,5,11 and 12; the according results are depicted in Fig. 7.20(a) to (d), respectively. For each performance measure, the worst value obtained from the 100 verification tests is used. Since the fitness evaluation of



(a) TM 2

(b) TM 1

(c) native fitness criterion

(d) TM 1+5+11+12 (measure of experiment 5)

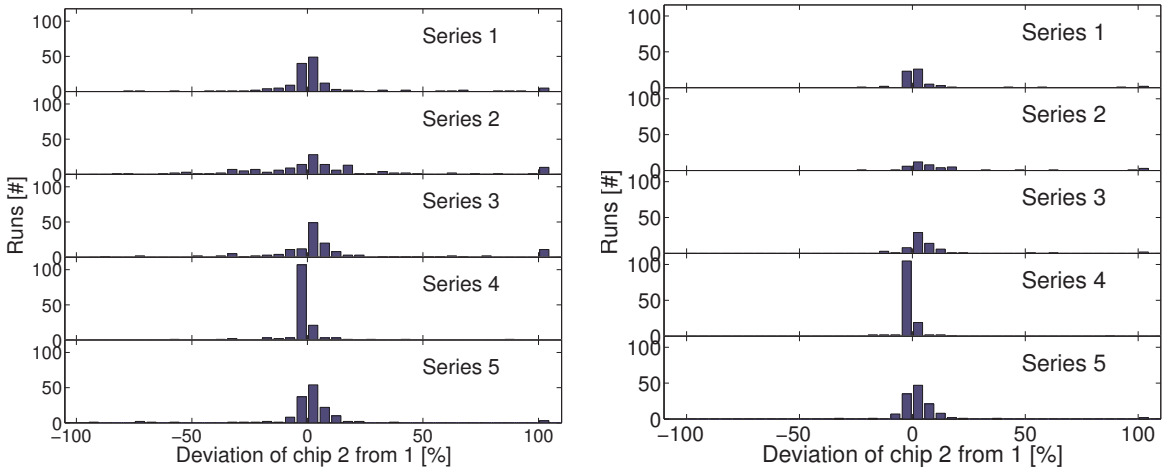
Figure 7.20: Comparison of the performance on two different FPTA chips, where chip 1 denotes the die used during the evolution process. For each series, the results for the frequency response (a), distortion (b), native fitness (c) and fitness in terms of the specifications of experiment 5 (d) are averaged over all runs of all experiments whose worst fitness deviates less than 100% from its best fitness. The number of circuits taken into account is denoted at the top of each plot for the respective experiment.

many circuit was found to be unreliable in section 7.3.3, the respective performance measures are averaged over all those runs of all experiments of one series, for which the deviation of the worst and best fitness values did not exceed 100% on chip 1. The number of runs considered for each series depends on the actual figure of merit; they are imprinted at the top of each of the four plots. While the last fitness criterion used for Fig. 7.20(d) applies the problem definition inherent to experiment 5 to all of the evolved circuits, the other three measures are unfair in that they overemphasize the fitness

contribution of the experiments describing the more difficult tasks, because these on average obtain higher error values.

First, the averages depicted in Fig. 7.20 confirm two previous observations: The performance differences described in the above section reveal themselves in the presented averages, too. For instance, it is again observed that series 4 excels by a low distortion, but exhibits a prohibitively bad magnitude response on average. The second observation relates to the number of circuits whose worst fitness deviates by less than 100% from its best fitness: Except for the native fitness criterion the reliability inherent to this number is significantly higher for series 4 and 5 than for the other series. Second, it is important to note that the performance differences between the two chips are in general fairly small and in particular much smaller than the variations encountered within the 100 verification test on chip 1. Third, the deviations between the performance on chip 1 and 2 are most significant for series 3, whose circuits seem to be somewhat less robust against migration to a second chip. The large differences observed in Fig. 7.20(c) for series 4 may be solely due to the unreliable means of fitness evaluation chosen here. The difference for series 5 observed in Fig. 7.20(d) on the other hand must be taken more seriously and probably stems from a performance degradation in test modes 11 and or 12.

The histograms in Fig. 7.21 display the distribution of percentage performance differences yielded by the test on the different dice. They are based on the sum of test modes 1,2,3,9 and 10, where the



(a) all runs

(b) difference between best and worst fitness < 100%

Figure 7.21: Percentage performance difference between the fitness values achieved on a second chip to those measured on the chip 1 used for the artificial evolution thereof. The fitness values are taken as the worst values measured in 100 verification tests for the sum of test modes 1,2,3,9 and 10, that is all types of evaluations of the magnitude response plus the distortion penalty from test mode 1. While (a) considers all 150 runs of each series, (b) rejects all those runs, for which the worst and best fitness values differ by more than 100%.

worst sum out of 100 verification tests is used. From the different histograms illustrating either all runs, or only those with a limited deviation between worst and best fitness values depicted in Fig. 7.21(a) and Fig. 7.21(b), respectively, it can be concluded that most of the circuits of series 1–3 that seem to fail on chip 2 in fact behave unreliable on chip 1, too. Moreover, the vast majority of circuits attains similar fitness values on both chips, with only a small fraction of outliers, especially in case of series 4 and 5.

7.4 Evolving LPFs on Different Frequency Scales

The experiments presented in the previous section are confined to the same frequency range. The FPTA chip, as probably any other piece of (electronic) hardware, features a set of time scales inherent to it. For instance, any resistor-capacitor pair defines such an intrinsic time scale. Therefore, this section tries to shed light on the question, in how far the desired corner frequency influences the success of the FPTA based evolution experiments targeted at the synthesis of LPFs.

7.4.1 Experimental Setup

In principle, exactly the same experimental setup that was used for the experiments of series 1 described in the previous section is used for the experiments described below. In particular, during the evolution process the fitness is evaluated by $F_{step\downarrow} + F_{step\uparrow} + F_{trans}$, as described in the second line of Table 7.7, while the full set of test modes 1–14 of Table 7.4 is applied in the verification tests. The difference to the experiments of the previous section are the smaller number of runs used for each experiment (10 instead of 30) and the actual task described by the different experiments. Instead of varying the distance of upper pass- and lower stopband frequency f_{PB} and f_{SB}, the five experiments listed in Table 7.10 are used to vary the overall frequency scale. Thereby, the range of the transition

Experiment	f_{start}	f_{stop}	f_{PB}	f_{SB}
1	3.33 kHz	1.67 MHz	133.33 kHz	666.67 kHz
2	1 kHz	500 kHz	40 kHz	200 kHz
3	526.32 Hz	263.16 Hz	21.05 kHz	105.26 kHz
4	270.27 Hz	135.14 kHz	10.81 kHz	54.05 kHz
5	136.99 Hz	68.5 kHz	5.48 kHz	27.40 kHz

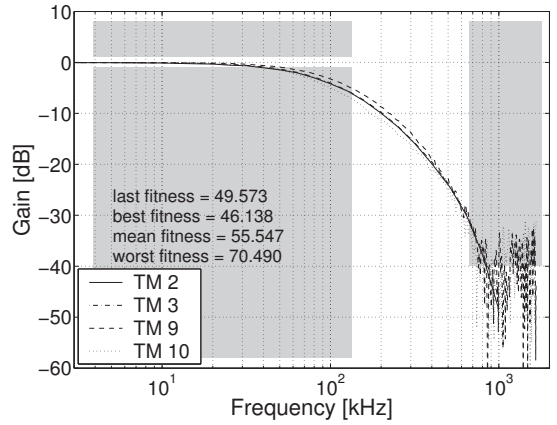
Table 7.10: Five experiment series for the evolution of LPFs.

band limited by the upper passband frequency f_{PB} and the lower stopband frequency f_{SB} is scaled such, that the ratio of f_{SB} and f_{PB} remains constant, that is equal to 5, which corresponds to the task difficulty defined by experiment 3 of the previous section. In fact, experiment 2 of Table 7.10 is identical to experiment 3 of series 1 in the previous section and consequently uses the same set of evolved circuits.

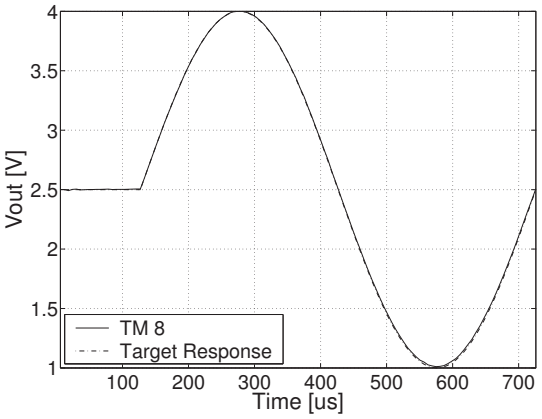
7.4.2 Output Behavior for the Best-Of-Experiment Circuits

The output behavior of the best-of-experiment solutions are presented in Fig. 7.22 and Fig. 7.23 for experiments 1–3 and 4–5, respectively. Here, best refers again to the min worst values achieved for the native fitness criterion. Analog to Fig. 7.10, the plots on the left hand side depict the magnitude responses determined by test modes 2,3,9 and 10 and the plots on the right hand side show the transient response according to test mode 8. As this test mode uses a sinusoidal input with a frequency of 1.67 kHz irrespective of the frequency scale of the target corner frequency determined by the experiment, the distance of this input tone to the upper passband edge f_{PB} varies for different experiments. Although always located within the passband, an input stimulus closer to the passband edge will cause a higher penalty in the transient analysis for imperfect, but viable LPFs, which may hinder the EA in finding good filter circuits.

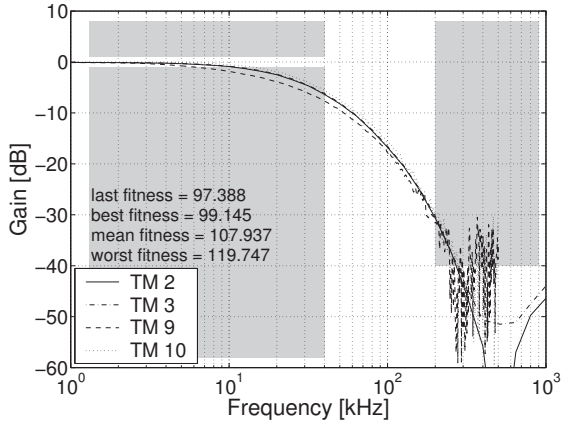
The best solutions obtained from experiments 1–3 look like such viable, albeit imperfect, LPFs, similar to those presented in section 7.3: They show good linearity for at least an input voltage range



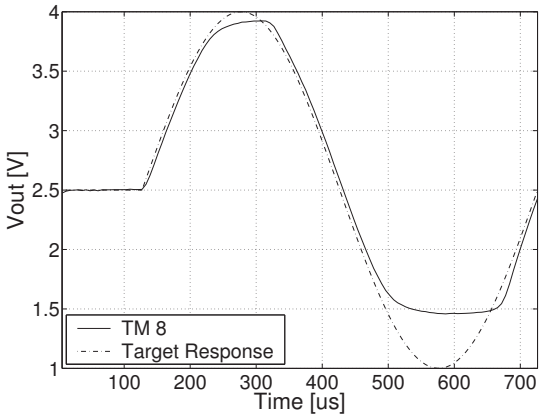
(a) experiment 1: TM 2,3,9,10



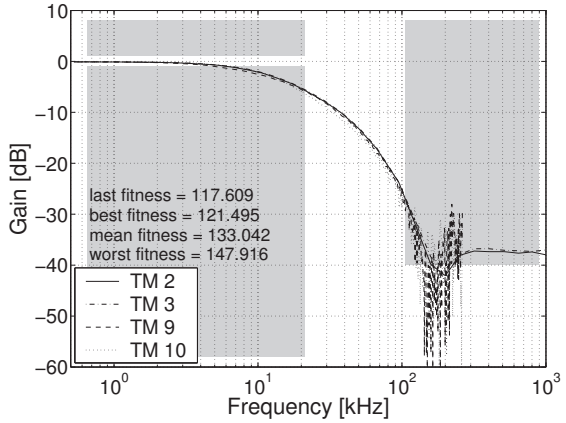
(b) experiment 1: TM 8



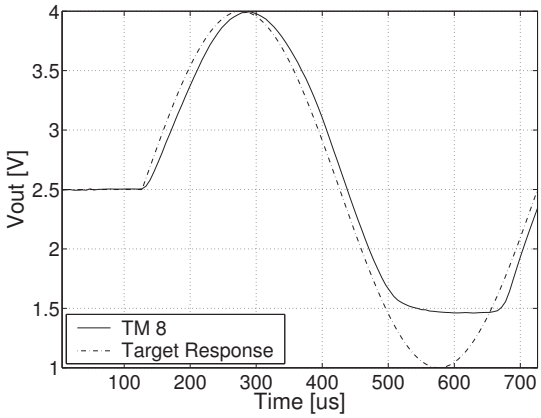
(c) experiment 2: TM 2,3,9,10



(d) experiment 2: TM 8

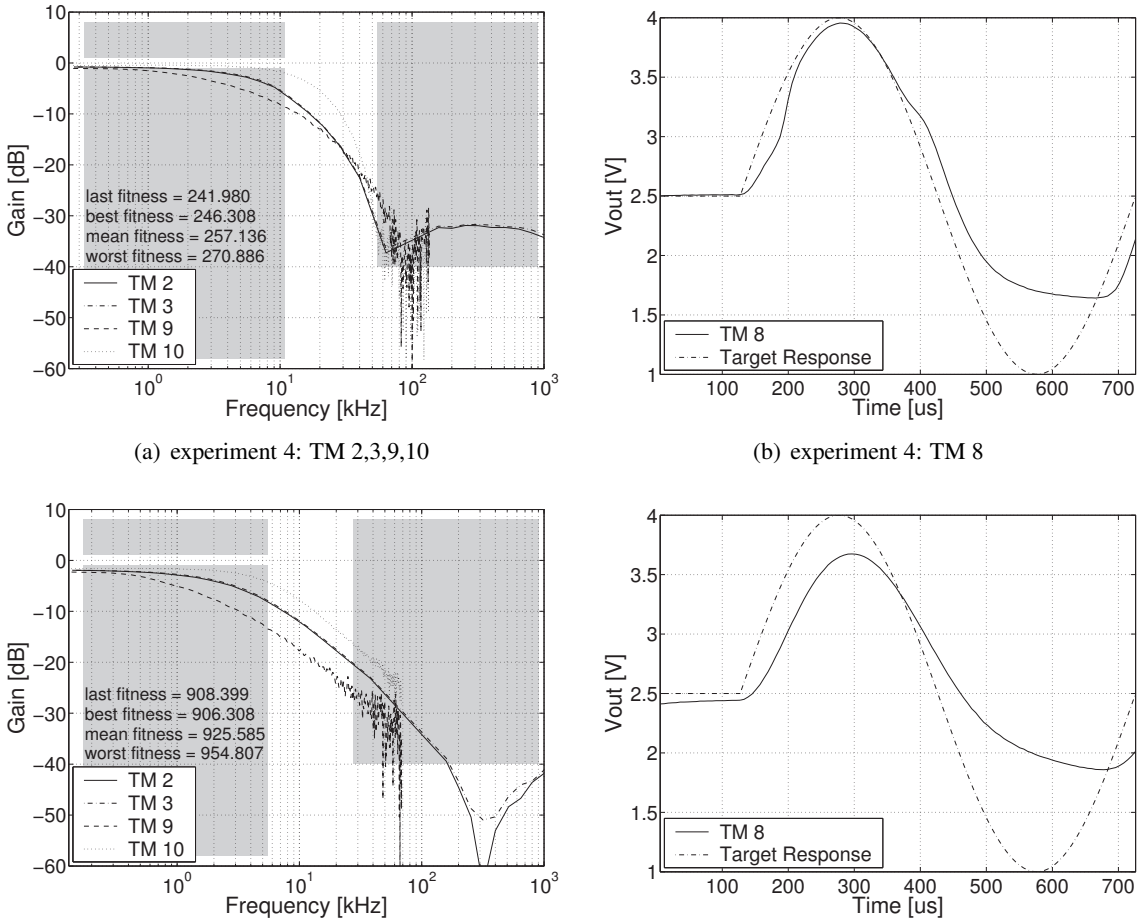


(e) experiment 3: TM 2,3,9,10



(f) experiment 3: TM 8

Figure 7.22: Comparison of the output behavior of the *best* filter circuits for experiments 1-3 (from top to bottom). Here, *best* refers to the fitness criterion used during evolution listed in Table 7.8, that is the sum of the error contributions of test modes 8-10. The plots on the left hand side illustrate the frequency response in terms of the test modes 2,3,9 and 10. The right hand side depicts the transient responses to a sine wave according to test mode 8.



(a) experiment 4: TM 2,3,9,10

(b) experiment 4: TM 8

(c) experiment 5: TM 2,3,9,10

(d) experiment 5: TM 8

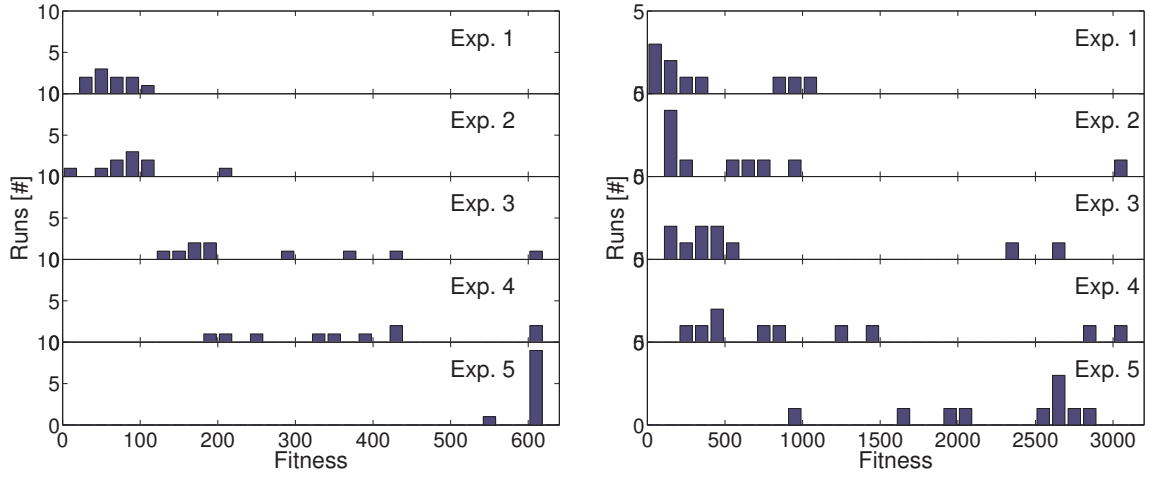
Figure 7.23: Comparison of the output behavior of the *best* filter circuits for experiments 4-5 (from top to bottom); refer to Fig. 7.22 for details.

of 1.5 to 3.5 V and a magnitude response that, in principle, complies with the desired specifications, yet cuts off some of the forbidden regions in the pass- and stopband. However, from the imprinted numbers for their best and worst fitness scores as well as from the plots themselves, the best-of-experiment solution can be observed to decrease in quality with a decreasing corner frequency.

The best solutions attained in experiments 4 and 5 suit this scheme well by exhibiting an even worse output behavior: while the magnitude response of the best circuit of experiment 4 fails to achieve the desired attenuation of $A_{\mathrm{SB}} = 40\,\mathrm{dB}$ even worse than its counterpart of experiment 3, the best filter obtained from series 5 shows a less steep rolloff rendering it a simple first order filter. Both filter circuits reveal severe nonlinearities, which cause their respective magnitude responses from the up and down step responses to fall apart.

7.4.3 Statistical Analysis

The above observations must be backed up with an overview over the distribution of fitness scores obtained for the different experiments. This is done by means of the histograms in Fig. 7.24(a) and 7.24(b), which are based on the best and worst min worst native fitness, respectively. Both of these fitness comparisons indeed confirm the decrease in performance with decreasing corner frequencies



(a) best fitness (b) worst fitness

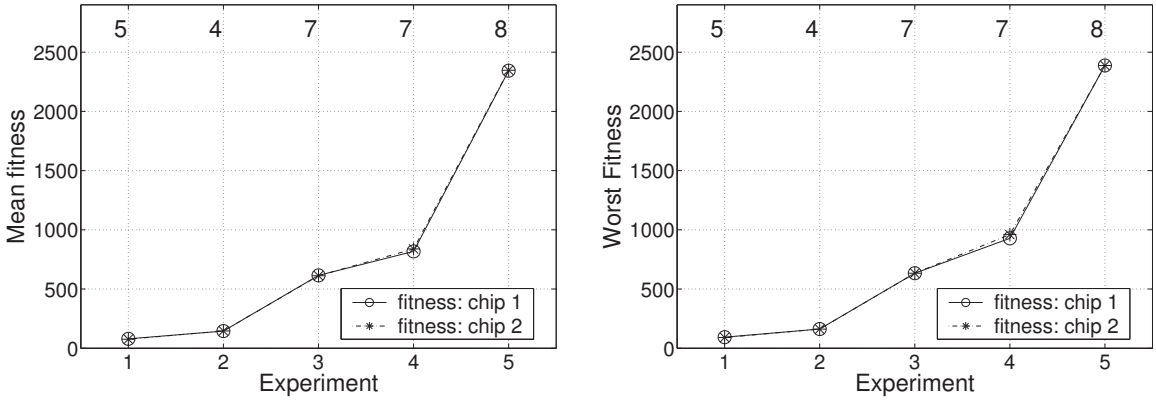
Figure 7.24: Histograms for the best (a) and worst (b) native fitness obtained for experiments 1–5.

that was already seen for the best-of-experiment solutions. The effect is more pronounced for the comparison of the best fitness values than for those of the worst ones. However, in both cases the average fitness scores increase monotonically with the experiment number, and the difference between the results of experiment 5 and experiments 1–4 is most significant. As expected from the results of section 7.3.3, there is an enormous discrepancy between the distribution of worst and best fitness scores.

Most probably, the correlation of evolution results and frequency scale must be mainly ascribed to the time scales inherent to the FPTA substrate. Given that the typical capacities available on the transistor array amount to something like 100 fF, a corner frequency of $f_{-3\,\mathrm{dB}} \approx 10\,\mathrm{kHz}$ necessitated by the target criterion of experiment 5 leads to a resistance in the order of $R = (2\pi \cdot f_{-3\,\mathrm{dB}} \cdot C)^{-1} = 160\,\mathrm{M\Omega}$ to realize a simple RC lowpass filter stage with the desired corner frequency. Although the possibility to split the capacitor as well as the resistor into a bunch of some ten devices brings the necessary resistance value down to about 1 MΩ, such resistors are still hard to implement with only CMOS transistors, when their linearity is required over a macroscopic voltage range. However, in case of series 4 and 5 the relatively high frequency used for the transient analysis may have further impeded the successful synthesis of good LPFs. Finally, it should be noted that from the results presented in Chapters 5 and 6 any influence of the bandwidth of the analog measurement system can be ruled out to play a role in the evolved circuits.

7.4.4 Migration to a Second Chip

The ability of the evolved circuits to work on a second FPTA chip is verified by means of 7.25(a) and 7.25(b) which are based on the mean and worst scores obtained for the native fitness criterion. Analog to Fig. 7.20(c), the performance of all runs of one experiment is averaged over all those circuits whose best and worst fitness deviate by less than 100%. First, the monotonic increase of fitness scores with increasing experiment number or rather target corner frequency is indeed observed for the according averages, too. Second, the performance on both chips is – on average – almost identical for the *reliable* circuits evolved.



(a) mean fitness (b) worst fitness

Figure 7.25: Comparison of the performance attained on two different FPTA chips, where chip 1 denotes the die used during the evolution process.

7.5 High Pass Filter

While the preceeding two sections focused on the evolution of LPFs, this section presents four experiments dedicated to the evolution of HPFs. Since other types of filters targeted at certain magnitude responses can in principle be thought of as combination of these two archetypes, this section therefore completes this chapter.

7.5.1 Experimental Setup

In principle, the experimental setup chosen for the HPF experiments closely follows that one used for the LPF experiments of series 1 presented in section 7.3. The particular choices as well as all difference to LPF settings are described below.

7.5.1.1 Task Description

As was already hinted in section 7.1.2.2, the evolution of LPFs is expected to be more difficult than that of LPFs. Hence, a relatively easy task whose difficulty corresponds to that described by experiment 2 of section 7.3 is adapted to define the target specifications by exchanging the roles of pass- and stopband. Again, the filter task is described by the abstract set of parameters defined in Fig. 7.2(b). The concrete values for those parameters are summarized in Table 7.11, the analogon to Table 7.6. On one hand, the parameters are chosen such, that the task is not too difficult. On the other hand, any

Parameter	f_{start}	f_{stop}	f_{SB}	f_{PB}	A_{PB}	δ	A_{SB}
Target Value	1 kHz	0.5…500 kHz	10 kHz	100 kHz	0 dB	0 dB	40 dB

Table 7.11: . Parameter set describing the HPF-task. These parameters are defined in Fig. 7.2(b).

of the traditional filter approximations that is to perfectly meet these filter specifications must be at least of second (Chebyshev I,II or Cauer filter) or even third (Butterworth filter) order (cf. Table 7.7).

7.5.1.2 Fitness Measures

All of the experiments presented below rely on establishing the magnitude response by means of a Fourier transform of a step response as described by method M1 in section 7.2.1. For one, this method proved to be helpful in generating circuits that achieve a good compromise between a good magnitude response and high linearity. For the other, their main disadvantage, namely the high noise floor encountered for high frequencies shall not be detrimental to the proper evaluation of HPFs, since they possess a large amplitude in the frequency range in question. Therefore, the SNR remains uncritical throughout the whole frequency range of interest (cf. section 7.2.4.1).

Overview of the Experiments. The differences characterizing the four different experiments presented in this section are summarized in Table 7.12. Apart from the differences in the GA-parameter

Experiment	reproduction fraction	Fitness Criterion
1	0.04	$F_{2.5\to1.5} + F_{2.5\to3.5}$
2	0.04	$F_{2\to3}$
3	0.2	$F_{2\to3}$
4	0.1	$F_{2\to3} + F_{3\to2}$

Table 7.12: Four experiments for the evolution of HPFs.

reproduction fraction, the fitness criterion used to evaluate the prospective HPF circuits during the evolution process represents the more interesting variation. First, it is important to note, that there is no need for a transient analysis at low frequencies, as they would fall into the stopband. A transient analysis at high frequencies located in the passband, on the other hand, could be advantageous to give the evolutionary loop some feedback about the linearity of the hopefully undistorted signals in the passband. Yet, such a transient response is hard to evaluate, since the phase shift between the input sine wave and the output of the candidate circuit are neither known in advance nor helpful to prescribe. Second, the fitness measures used for all of the experiments are all based on the step responses described by test mode 9 and 10 of Table 7.4. In fact, while $F_{2.5\to1.5}$ and $F_{2.5\to3.5}$ are identical to $F_{\text{step}\downarrow}$ and $F_{\text{step}\uparrow}$, $F_{3\to2}$ and $F_{2\to3}$ merely differ from those by a shifted input voltage of $\pm0.5\,\text{V}$, respectively.

The particular composition of step responses is motivated as follows: The sum of test modes used for experiment 1 is basically identical to that one used for series 1 and 2 of section 7.3 freed of the transient analysis. Experiments 2 and 3 try to answer the question, whether it is sufficient, beneficial or detrimental for the quality of the evolved circuits to use only one type of step, which happens to point upward here. With regard to the difficulty of the task, this implicit reduction in evaluation time would suit the synthesis of HPF well. Finally, the fitness criterion used in experiment 4 simplifies the task defined in experiment 1 by reducing the in- and output range the candidate circuits have to comply with.

List of all Test Modes. The complete list of test modes used in the verification tests is presented in Table 7.13. It is the analogon to Table 7.4 on page 208 used for the experiments on LPFs. Apart from the slightly changed target parameters contained in Table 7.11, the set of HPF test modes differs from its HPF counterpart in the following aspects: The analoga to the former test modes 5 and 8 are discarded. The new versions of test modes 9 and 10 take on a similar role to the former test modes 11 and 12 in that they account for the shifted step responses $F_{3\to2}$ and $F_{2\to3}$ used in experiments 2–4. There are no correspondents to the former modes 15–20.

No.	Description	Name	Equation	Purpose	Weight[a]	N_{total}
1	Distortion: Fourier-analyzed ac-sweep	F_{dist}	(7.43)	verification	0.5	24693
2	Magnitude response: Fourier analyzed ac-sweep	F_{acsf}	(7.42)	verification	0.5	24693
3	**M2**: Magnitude response: mean signal power, ac-sweep	F_{acs}	(7.29)	evolution	0.5	24693
4	Trans. resp. for ac sweep		–	illustration	–	24693
5	Transient response, 625 kHz sine tone		–	illustration	–	201
6	Transient response, 822 Hz sine tone		–	illustration	–	201
7	**M1**: Step response: 2.5 V → 1.5 V	$F_{2.5\rightarrow1.5}$	(7.22)	exp. 1	1	500
8	**M1**: Step response: 2.5 V → 3.5 V	$F_{2.5\rightarrow3.5}$	(7.22)	exp. 1	1	500
9	**M1**: Step response: 3 V → 2 V	$F_{3\rightarrow2}$	(7.22)	exp. 4	1	500
10	**M1**: Step response: 2 V → 3 V	$F_{2\rightarrow3}$	(7.22)	exp. 2-4	1	500
11	Trans. resp. for step: 2.5 V → 1.5 V		–	illustration	–	500
12	Trans. resp. for step: 2.5 V → 3.5 V		–	illustration	–	500

[a]Weight refers to the global weight/scaling of the respective fitness contribution. Test modes 9-12 and 15-20 possess additional weights for their different frequency components as discussed above and below, respectively.
[a]The transition band is the narrowest for experiments of type exp. 5. It stretches from 80 to 200 kHz (cf. section 7.3.1).

Table 7.13: List of all test modes used during the artificial evolution and/or for the verification tests of HPFs.

7.5.1.3 GA and Other Parameters

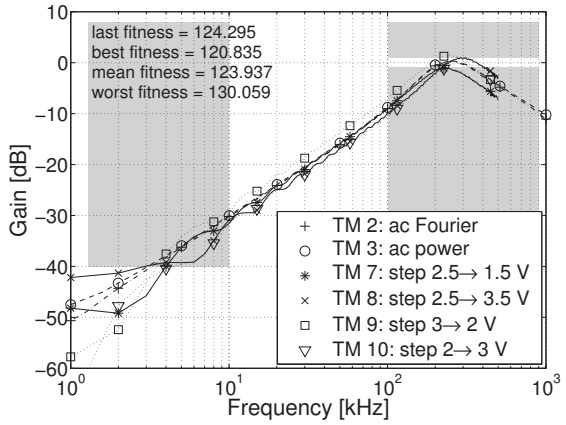
The genetic algorithm parameters have already been stated in Table 7.9. Apart from the different reproduction fractions applied in the four different experiments, the main difference is given by the increased number of generations. The tenfold increase compared to most of the LPF experiments shall account for the increased difficulty of the task at hand. As this implies a significant increase in the necessary computation time, the number of runs per experiments is reduced to 20. Finally, the smaller geometry I depicted in Fig. 7.9(a) is utilized throughout all four experiments.

7.5.2 Output Behavior for the Best-Of-Experiment Circuits

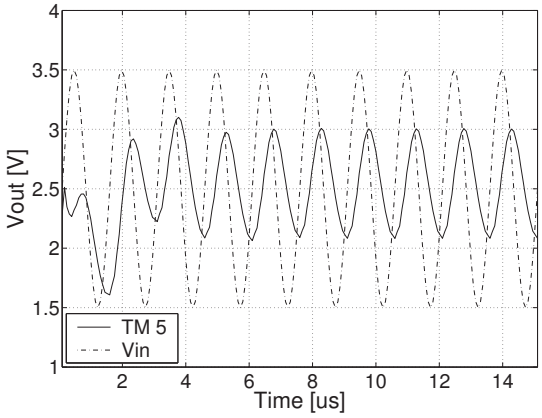
To get an impression of the quality of the evolved HPF filters, the output behavior of the best circuit of each experiment is plotted in Fig. 7.26 and Fig. 7.26 for experiments 1–3 and experiment 4 respectively. Here best refers to the min worst native fitness achieved within the 100 verification tests. While the magnitude responses of the respective filter determined by test modes 2,3 and 7–10 are depicted on the left hand side of Fig. 7.26, the plots on the right hand side present the circuits' output for a sinusoidal input stimulus of frequency 625 kHz as described by test mode 5.

In general, all of the best magnitude responses that are determined by test modes also used during the evolution meet the desired specifications fairly well, although they usually fall short of perfectly keeping out of the forbidden regions of the pass- and stopband as well as of realizing a flat response of approximately 0 dB attenuation in the stopband. The respective transient responses are fairly linear; yet, they are attenuated in comparison with the input signal. Please note, that the output signal of highpass filters in general has to undergo some transient period and experiences a phase shift with respect to the phase of the input signal.

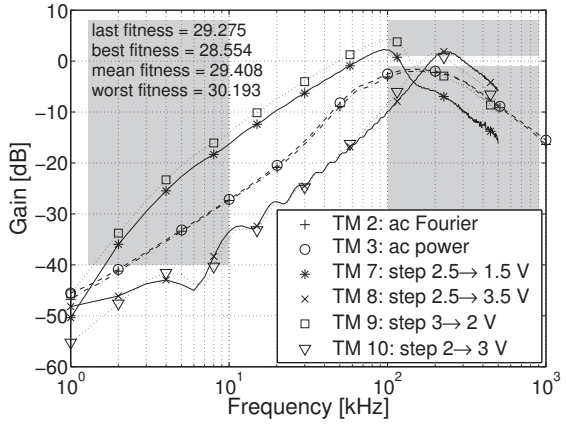
A comparison of the magnitude responses of the best circuits of the four different experiments reveals the following: First, from the results of experiments 2 and 3, it can be observed that it is not sufficient to evaluate the magnitude response for only one input step direction. Second, a comparison of the consistence of the magnitude responses determined by different test modes for the best circuit



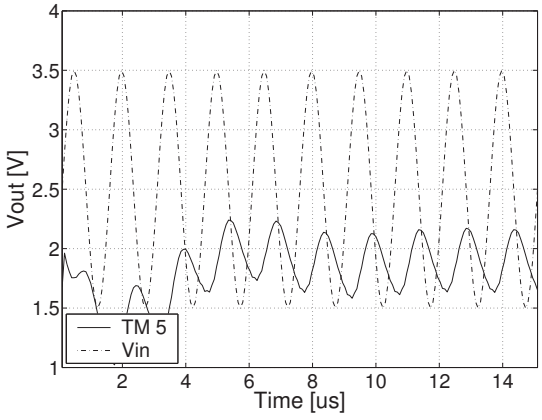
(a) experiment 1: TM 2,3,7–10



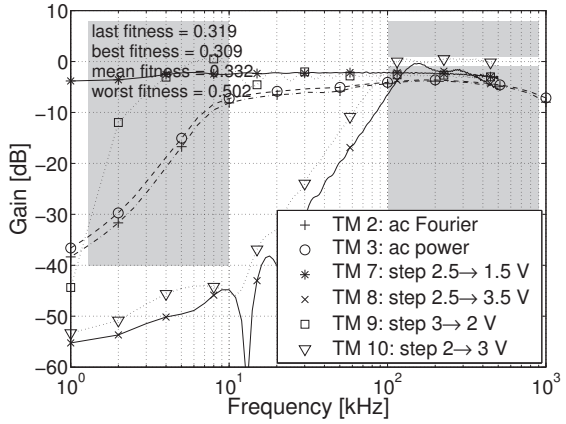
(b) experiment 1: TM 5



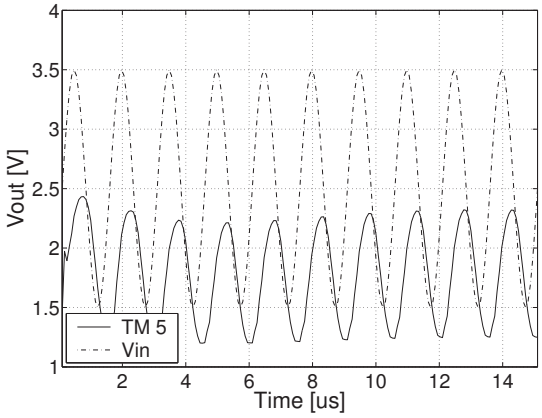
(c) experiment 2: TM 2,3,7–10



(d) experiment 2: TM 5

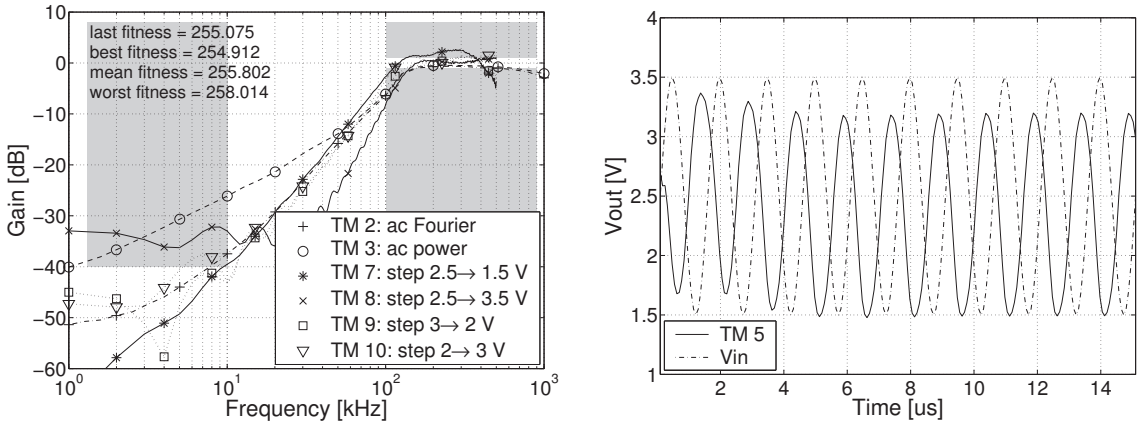


(e) experiment 3: TM 2,3,7–10



(f) experiment 3: TM 5

Figure 7.26: Comparison of the best output behavior of the *best* HPF filter circuits for experiment 1-3. Here, *best* refers to the fitness criterion used during evolution as listed in Table 7.12. The plots on the left hand side illustrate the frequency response in terms of the test modes 2,3 and 7–10. The right hand side depicts the transient responses to a sine wave at $f = 625\,\text{kHz}$ according to test mode 5.



(a) experiment 4: TM 2,3,7–10

(b) experiment 4: TM 5

Figure 7.27: Continuation of Fig. 7.26 for best circuit of experiment 4. The same conditions as in Fig. 7.26 apply.

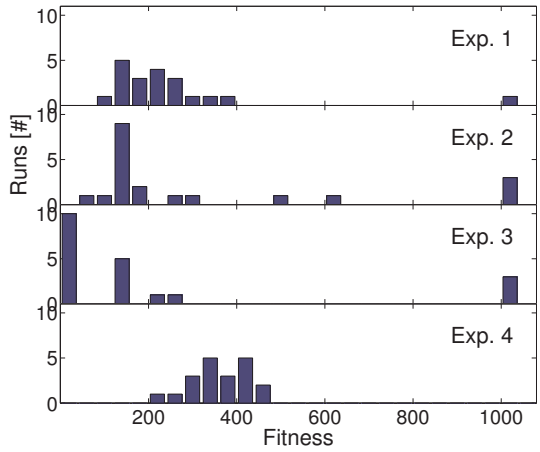
of experiments 1 and 4 indicated that the reduced input voltage range is not a sufficient fitness criterion either if the evolved filters are to work on the larger input voltage range. However, the deterioration caused for the full fitness criterion of experiment 1 is less severe as for the single step criterion. Conversely, the EA could take advantage of the simpler fitness criterion in case of experiments 3 and 4, as the best overall magnitude responses are obtained for their native test modes.

Although the magnitude responses obtained for test modes 9 and 10 depicted in Fig. 7.27(a) almost perfectly match the target specifications, the fitness attained by the best circuit of experiment 4 exceed the fitness values achieved by all other best-of-experiment solutions. This is due to a flaw in the fitness criterion of experiment 4: According to (7.23) the step-response based fitness evaluations contain a dc penalty for the last 50 samples before the step position. Therefore, the two test modes $F_{3\to2}$ and $F_{2\to3}$ require successful filter circuits to adapt to the input level of either 3 or 2 V. But on the other hand, by definition, an ideal HPF possesses a dc gain of zero. Consequently, the best a HPF under evolution can do is to force its output to a dc level of 2.5 V, which results in an offset penalty of $2 \cdot 10 \cdot 50 \cdot 0.5^2 = 250$. Therefore, the evaluation by means of experiment 4's fitness criterion carries a fitness offset of at least 250. However, since the selection is performed in a rank based scheme and all other fitness criteria also force the circuits under test to a prescribed dc output level, this does not discriminate the evolution process itself.

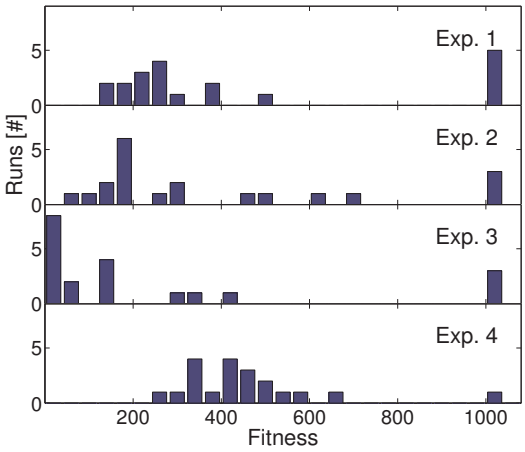
7.5.3 Statistical Analysis

The histograms of Fig. 7.28 display the performance of all runs of all experiments. Fitness values are taken from the native fitness (Fig. 7.28(a) and (b)), the fitness criterion of experiment 1 (Fig. 7.28(c) and (d)) and the criterion used for experiment 4 (Fig. 7.28(e) and (f)) for the best and the worst fitness, respectively. In order to allow for a fair comparison of the results attained for experiments 2 and 3, which use only one single step response, with those of experiments 1 and 4, the fitness values of the former ones are doubled before they entered Fig. 7.28(a) and (b).

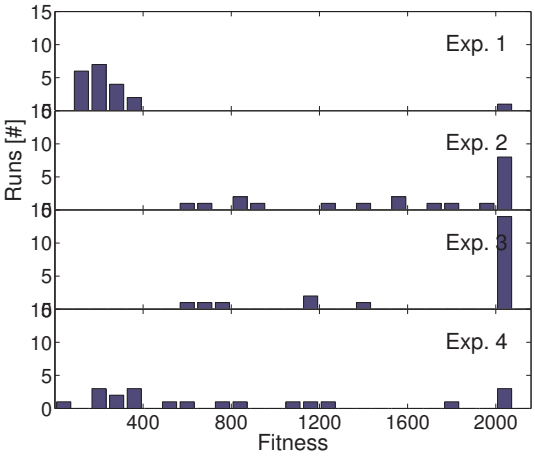
Firstly, the histograms for the native fitness criterion reveal that most of the runs are confined to a fairly narrow region, i.e. perform similarly well, while the rest clearly fails to come close to the target specifications. Secondly, the deviations between the best and worst fitness results seem to be less dramatic than for the LPFs. Thirdly, Fig. 7.28(a) and (b) indicate that the easier task described by the single step response $F_{2\to3}$ lead to better fitness scores. Finally, a comparison between the results



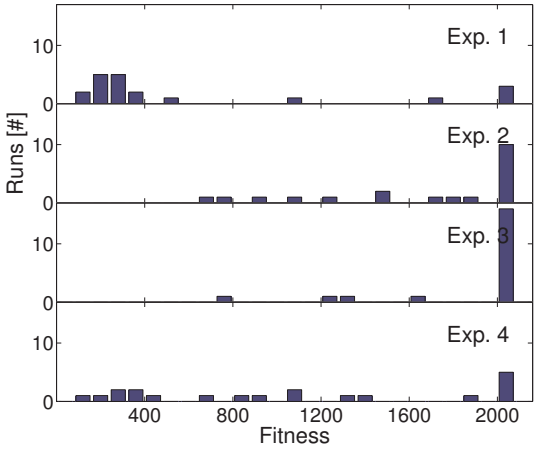
(a) best native fitness



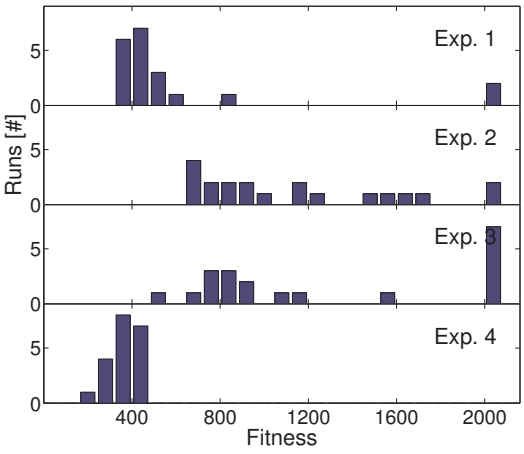
(b) worst native fitness



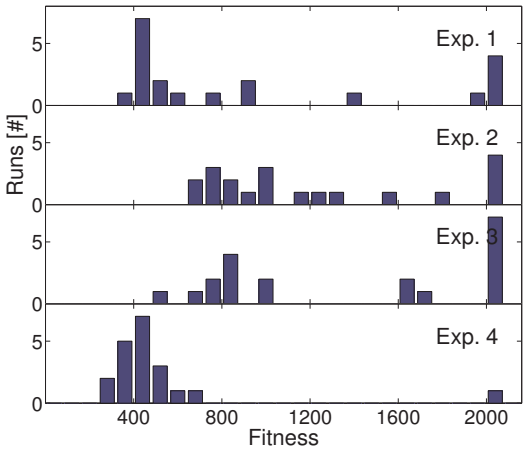
(c) best fitness, criterion of experiment 1



(d) worst fitness, criterion of experiment 1



(e) best fitness, criterion of experiment 4



(f) worst fitness, criterion of experiment 4

Figure 7.28: Histograms comparing the results for the different experiments. The plots on the left hand side account for the best and those on the right hand side for the worst fitness values attained within the 100 verification tests. From top to bottom, the native fitness criterion, that for experiment 1 and that for experiment 4 are used. The histograms for experiments 2 and 3 of Fig. (a) and (a) are doubled to allow a fair comparison with the respective histograms for experiments 1 and 4.

for experiments 2 and 3 suggests that a higher reproduction fraction is beneficial to the success of the EA for the given problem formulation.

In accordance with the observations for the best-of-experiment circuits, the histograms for the fitness criteria of experiment 1 and 4 prove the circuits evolved in experiments 2 and 3 incapable of fulfilling the more general tasks involving two step responses in opposite directions. Here, the increase in the covered input voltage range makes things even worse. The circuits evolved in experiment 4 outperform those of experiment 1 when compared in their native fitness criterion, but most of them falls short of reproducing their good results for the criterion of experiment 1. Thus, once again, the presumably easier task allows for the evolution of better circuits that in general to not generalize to the more complex problem.

The statistical analysis of the ensemble of HPF experiments shall be concluded with a scatter plot relating the quality of magnitude response and linearity of the evolved circuits with each other. The graph in Fig. 7.29 is equivalent to the plot for experiment 2 in Fig. 7.18. A comparison between

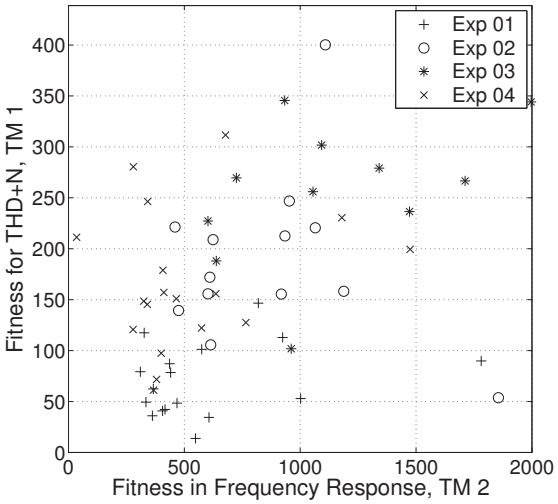
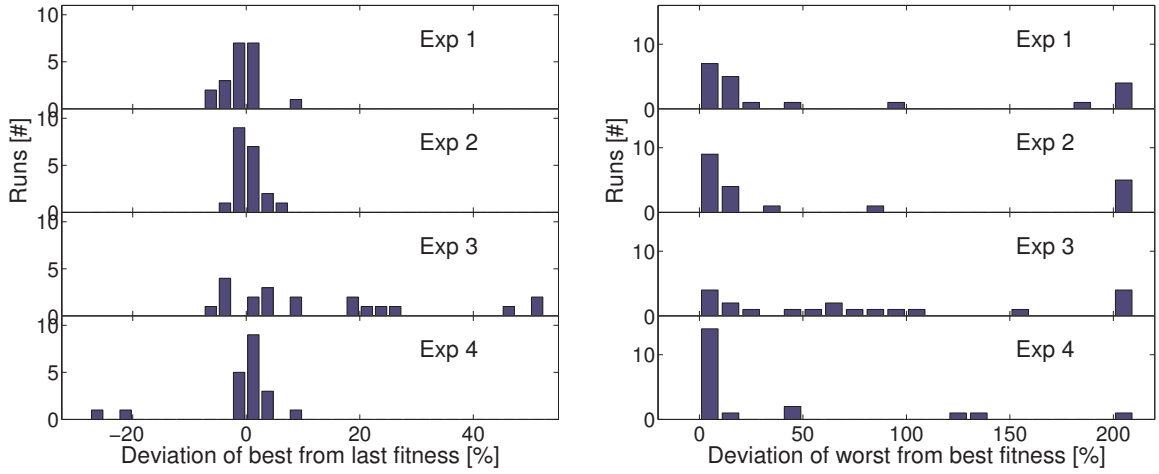


Figure 7.29: Scatter plot relating the achieved distortion values from TM 1 with the according quality of the magnitude response evaluated by means of TM 2. The plots are based on the highest (worst) values for the respective test modes attained within the 100 verification tests.

these two graphs reveals that the best of the evolved HPF possess a rather small distortion penalty in comparison with the LPFs, but perform worse in terms of the magnitude response. As expected, the best tradeoffs between distortion and magnitude response are achieved by the circuits evolved with the combination of test modes 7 and 8, whereas those using only test mode 10 perform relatively bad. However, it is interesting to note, that the better circuits evolved in experiment 4 perform almost as well in terms of the magnitude response of those evolved in experiment 1, but exhibit a considerably higher amount of distortion.

7.5.4 Reproducibility

The reliability of the evolved HPFs is analyzed by means of the histograms depicted in Fig. 7.30(a) and Fig. 7.30(b), which are the equivalents of Fig. 7.13(a) and Fig. 7.13(c). In comparison, the distribution of deviations between the best and the last fitness is more confined for the evolved HPFs compared with their LPF counterparts from series 1 and 2, at least for experiments 1,2 and 4. Furthermore, the fraction of HPFs whose worst fitness deviates by less than 100% from their best fitness is considerably larger than the respective fraction obtained for the evolved LPFs presented in section 7.3. A comparison of the deviation for the circuits of experiments 2 and 3 reveals that the larger reproduction fraction tends to cause the evolved circuits to perform less reproducible. This confirms the hypothesis stated in section 7.3.3 that an increased number of phenotypically (and here also genotypically) identical individuals per generation reduces the selection pressure towards reliable circuits.



(a) best from last (b) worst from best

Figure 7.30: Histograms for the deviations of the best from the last fitness values (a) and the deviation of the worst from the best fitness values (b).

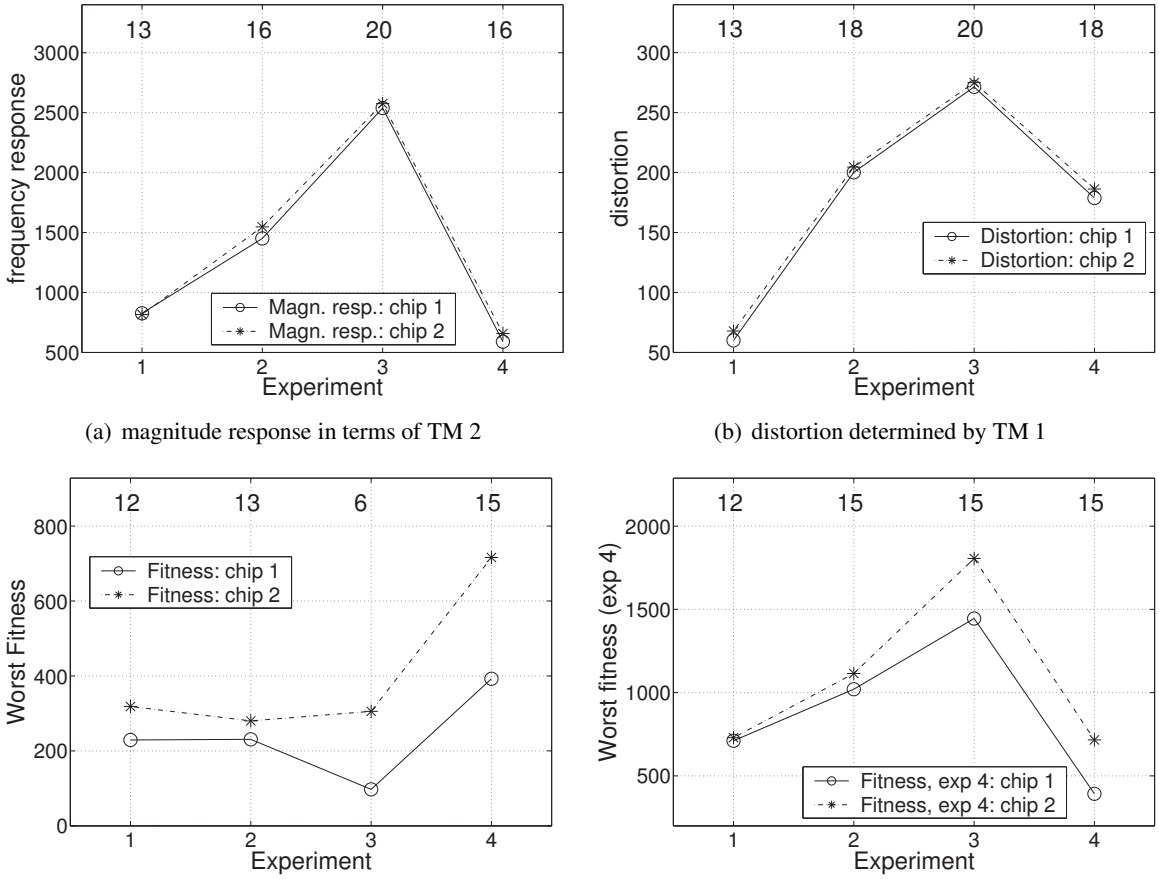
7.5.5 Migration to a Second Chip

Analog to section 7.3.6 the ability to work on a second FPTA chip is tested for different figures of merit. The average performance on both chips are depicted in Fig. 7.31, the equivalent to Fig. 7.20. Here, only those runs are considered whose deviation between best and worst fitness does not exceed 25% instead of 100%, because they were found to perform more reliably in the preceeding section. While the performance losses in terms of test modes 1 and 2 are negligible, they are significant in terms of the native fitness criterion as well as for that of experiment 4, which are both based on a step response rather than a Fourier analyzed ac-sweep. In the latter cases, the performance differences on both chips are larger than for their LPF counterparts. Opposite to the evolved LPFs, the average performance differences between two chips are of similar magnitude as those between the best and worst fitness scores obtained within the 100 verification tests. One possible reason for the dependency of the performance loss experienced on a second chip on the actual figure of merit would be a difference in the dc output voltage level resulting in an increased penalty P_{OS} described by (7.23), since this only affects the step-based fitness criteria.

7.6 Summary and Discussion

In the course of this chapter, different types of experiments have been presented that are aimed at the intrinsic evolution of transistor level low- and highpass filters.

Thereby, the experiments targeted at the automatic synthesis of LPFs addressed the comparison of three different methods of evaluating the magnitude response of the circuit under test as well as the dependency of the evolutionary success on the target corner frequency. The experiments aimed at the evolution of HPFs, which turned out to be more difficult to synthesize, focused on possible reductions and simplifications of the fitness measure. In the section below, the most important findings are summarized. In a second step, the results are compared to related work found in the literature, which eventually leads to an outlook to possible future work.



(a) magnitude response in terms of TM 2

(b) distortion determined by TM 1

(c) native fitness criterion

(d) TM 9+10 (measure of experiment 4)

Figure 7.31: Comparison of the performance on two different FPTA chips, where chip 1 denotes the die used during the evolution process. For each experiment, the results for the frequency response (a), distortion (b), native fitness (c) and fitness in terms of the specifications of experiment 4 (d) are averaged over those runs of each experiment, for which the percentage difference of best and worst fitness did not exceed 25% within the 100 verification tests. . The fitness values attained for experiment 2 and 3 in (c) are doubled to allow for a fair comparison with the remaining two experiments.

7.6.1 Lessons Learned

Noise The three proposed methods evaluate the magnitude response of a candidate circuit either by taking the Fourier transform of a step response (M1), calculating the mean signal power of the response to a set of sinusoidal input tones (M2), or by taking the Fourier transform of the response to such a set of sinusoidal input tones of different frequencies (M3). For the same analog accuracy of the IO system, they differ considerably in their inherent noise floor that defines the maximum possible attenuation that can be reliably measured. The noise floor associated with M3 exceeds that of M2 by 19 dB and that of M1 even by 29 dB for the settings used in the presented experiments. Though the latter difference is likely to be reduced at high frequencies, it is possible to lower the noise floor even further for method M3 by increasing the number of cycles measured for each of the input tones. Since the noise floor inherent to method M1 increases for higher frequencies, this method is better suited for testing HPFs. In the proposed experiments and for an effective number of bits equal to 12 – the nominal resolution

of the used ADC – the noise levels amount to $-44\,\mathrm{dB}$, $-63\,\mathrm{dB}$ and $-73\,\mathrm{dB}$ for methods M1, M2 and M3, respectively.

Consistency of M1–M3 The magnitude responses obtained from the three different methods M1–M3 are found to be consistent, as long as the circuits under test are sufficiently linear. A significant distortion of the filter's output manifests itself in deterioration of the step-response based magnitude responses, which can be exploited to guide the EA towards more linear solutions.

Evolutionary Success for M1–M3 In the presented experiments, method M2, which bases the evaluation of the magnitude response on the mean signal power of the response to a set of sine waves, led to the best magnitude responses, but in general also to higher levels of nonlinearity. A better compromise between these two goals was usually obtained from the experiments using the step-response based evaluation of method M1. Here, fivefold increase of the number of generations only led to a very modest performance improvement. Finally, the concrete implementation of method M3, the Fourier analyzed sine response, is found to produce unreliable fitness scores and can therefore be expected to perform better for a more consistent setup. Method M3 stands out in that it offers the possibility to choose the tradeoff between linearity and magnitude response by means of a small set of weighting factors. Together with its low noise floor, this suggests M3 to be the method of choice.

Task Difficulty The difficulty of the LPF task has been varied by choosing five different ratios of stop- and passband frequencies. For the first three ratios of 20,10 and 5, the more difficult task on average led to better filter circuits. The further decrease to ratios of 3.5 and 2.5 did not improve the set of evolved filters significantly. Thus, the latter tasks are inferred to be too difficult for the given experimental setup, including the EA.

Corner Frequency The success in automatically synthesizing LPFs on the proposed evolution system decreases with decreasing target corner frequencies. For the two lowest stopband frequencies f_{SB} of 54 and 27 kHz the chosen setup seems to be relatively impractical.

HPFs versus LPFs In general, the evolution of HPFs turned out to be more difficult. In spite of a tenfold increase in the number of generations used per EA run, the evolved HPFs do not match the performance attained by their LPF counterparts. For instance, for the analogon to the LPF task, the best filter circuit achieves a rolloff of only slightly more than the 20 dB encountered for a first order filter.

HPFs: Reduction of Fitness Criteria The HPFs have been evolved for three variations of the step-response based fitness criterion. Thereby, the difficulty of the task was decreased either by a decrease in the required in- and output compliance or by reducing the evaluation to only one upward step. The task simplification did indeed help to evolve circuits that perform better at this respective task. However, the according circuits fail in solving the original, more difficult task.

Reliability Despite the introduction of an analog substrate reset and a randomization of the test mode order, a large fraction of the evolved LPFs prove unable to reproduce their best output performance throughout a series of 100 subsequent verification tests. This flaw is much more severe for the circuits attained by means of methods M1 and M2, where more than 40% exhibit a deviation between the best and worst fitness scores of more than 100%, than for those evolved using method M3. In the latter case, only about 10–15% of the circuits vary by more than 100% in their fitness score. In fact, the spread of fitness scores for method M1 and M2 significantly exceeds that observed for the evolved circuits presented in the previous two chapters. Although

the experiments targeted at the synthesis of HPFs also rely on the step-response based evaluation of the magnitude response, the fraction of circuits that fail in at least one of the verification tests amounts to approximately 20%, which is significantly lower than for their HPF counterparts. The presented experiments suggest that a larger number of phenotypically identical individuals present in one generation, as e.g. caused by a higher reproduction fraction or a premature convergence, decreases the selection pressure towards reliable filter circuits and thus results in a higher fraction of unreliable filters.

Migration to a Second Chip While the evolved LPFs experience only a very small performance loss when operated on a different FPTA chip, the performance of the evolved HPFs decreases more severely on this second chip. In the latter case, the performance variations within the 100 verification tests and those between the two dice are of comparable magnitude.

7.6.2 Comparison with Related Work

Automated Filter Synthesis. The results presented are not in the least anywhere close to the performance reported for tools from the automatic design community or the artificial evolution/synthesis of passive *RLC* filters. This does not come as a surprise, as both problems are less difficult than the one tackled here. While the former task is usually reduced to parameterizing a given well suited, albeit universal, filter architecture and does not include transistor level circuit optimization or synthesis, the latter ones restricts its allowed devices to a set ideally suited for filter applications and moreover can rely on ac-analyses or even analytical calculations, because the according transfer functions are assumed to be perfectly linear.

The best magnitude response presented for the LPFs belongs to experiment 5 of series 3 and is depicted in Fig. 7.12(c). According to Table 7.7, the according filter operation is of order 3 or 4. Yet, the circuit is only linear in a restricted input voltage range of $2-3.5\,\mathrm{V}$. Apart from this particular circuit, there are quite a few LPF filter circuits which are supposed to be at least of second or third order and are almost linear of the full input range of $1.5-3.5\,\mathrm{V}$. Examples are found for series 3, experiment 2 and 3 in Fig. 7.19(c), or for series 1 and 2 in Fig 7.14(a). While the best HPF found for a restricted input range of $2-3\,\mathrm{V}$ exhibits a second to third order characteristic, its counterpart for the larger input compliance of $1.5-3.5\,\mathrm{V}$ achieved only a first order rolloff and fails to provide a flat response throughout the passband.

Although these results are far from relevant to any possible application and clearly lack an industrial strength, they compare remarkably well to those reported for similar types of problems: Most of the intrinsic HWE experiments targeted at automatic filter design are restricted to tuning a few passive component values of an otherwise predefined topology, as e.g. described in [She02], [Flo00], [Gre04]. Even though the approach proposed by Zebulum et al. [Zeb99] does include topology selection to find a second order SC LPF, it does only generate the switching network and thus cannot be said to work on the transistor level. Nevertheless, the LPF reported in [Zeb99] provides a rolloff of less than $-20\,\mathrm{dB}$ per decade.

To the author's knowledge, the results that have been published on *transistor level* intrinsic HWE of analog frequency selective filters to date are restricted to the group of Adrian Stoica. In [Sto04], [Key04], and [Zeb04] Stoica et al. report the evolution of an LPF and HPF with a rolloff of $\pm 14\,\mathrm{dB}$ per decade and a corner frequency $f_{\text{-3 dB}}$ of 1 and 10 kHz, respectively, which is verified over a frequency range of little more than a decade. In their experiments, the EA was allowed to use 10 of their FPTA-2 cells, which include linear capacitors.

In contrast to these indeed moderate results, in [Key00a], [Sto01d] and [Sto02a] Stoica et al. report the successful *extrinsic* evolution of several BPFs with passbands located between 1 and 25 kHz whose rolloffs are of the order of $\pm 40\,\mathrm{dB}$. Thereby, one or two of their FPTA-0 cells together with

some additional linear capacitors are used. This strongly suggests that the *intrinsic* approach is far more difficult than the *extrinsic* one. A possible explanation may on one hand point to capacitor values that are prohibitively large for a VLSI[28] integration ([Sto01d]) and on the other hand may object against restricting the fitness evaluation to an ac-analysis in the simulation based experiments. As an ac-analysis linearizes all devices by assuming infinitesimally small signals, circuits that are solely tested by such an ac-sweep are likely to fail for any realistic conditions.

Finally, Botelho et al. [Bot03] and Vieira et al. [Vie04] propose experiments on the *extrinsic transistor-level* HWE of low- and highpass filters using only MOS transistors. Their respective filters are also targeted at corner frequencies between 10 and 100 kHz and achieve between 20 and -40 dB rolloff together with linear output voltage range of almost 2 V. On one hand, both publications report between 20.000 and 50.000 necessary fitness evaluations, which is significantly less than were used in the experiments presented in this chapter. On the other hand, in [Bot03] the EA can use transistor lengths and widths between 0.6 and 600 µm, which greatly facilitates the task. Moreover, the fitness evaluation is restricted to ac- and dc-analyses. Hence, without a transient analysis, it remains questionable, whether the proposed filters would work as real circuits.

Configurable Devices for Filtering Applications. The presented filter experiments reveal that the feasible range of corner frequencies as well as the available bandwidth for evolved filters is similar to or rather a subrange of that of today's commercial FPAAs, as e.g. offered by Anadigm (cf. [AN203] or [Ana03a]). In terms of feasible filter orders, range of corner frequencies, precision, possible attenuation or variety of realizable filter types and approximations, such chips will be hard to top with an evolutionary approach. As such FPAAs are supported by software that lets the user conveniently configure the desired filters from abstract specifications, there seems to be no need for evolved filters operating in the frequency regime below 1 MHz.

Conversely, even most recent publications (see for instance [Pan02], [Bec04] or [Pav00]) on configurable filters allowing for corner frequencies beyond 100 MHz have not arrived at such full flexibility, ease of programming or such low noise and distortion levels as the abovementioned low frequency FPAAs. Hence, to allow for a higher bandwidth, higher corner frequencies, and to facilitate the synthesis task, future HWE experiments targeted at filter applications should resort to a different type of hardware substrate, which is based on recent techniques used for filter implementations. From the author's point of view, a chip based on G_m-C cells that are programmed mainly by varying the necessary bias currents and to a lesser extend by programmable capacitors should be most promising. In the best case, the configuration can be shifted from digital switch information to quasi-continuous time constants, which does not only smooth the fitness landscape but also reduces the parasitic effects introduced by the configuration switches.

[28] Very Large-Scale Integration

Chapter 8

Evolution Using Human Made Building Blocks

We must either find a way or make one.

HANNIBAL

While the previous result chapters focused on different types of target behaviors and thereby different types of test modes, the following three case studies introduce a new methodology to facilitate the analog circuit synthesis based on the proposed FPTA evolution system: Instead of restricting the automated circuit design to using only plain transistors, a library of simple user defined building blocks is provided to the genetic algorithm. While the first two case studies employ a library consisting of the simple logic gates NOR, NAND, NOT and a buffer to evolve the more complex gates XOR and XNOR as well as a tone discriminating circuit, a larger library of more analog circuit primitives is used for the third case study that is aims at the synthesis of comparator circuits.

The results presented in the previous three chapters, especially in Chapter 5, indicate that a simple GA implementation fails to produce human competitive solutions with an acceptable yield rate – even for relatively small problems. Basically, there are two ways to overcome this problem: By improving the search algorithm in general so that it is better suited for the class of problems to be solved, and/or by including human domain knowledge in the algorithm. The latter approach will reduce the search space and preclude solutions that are too different from those created by humans. In other words, introducing human design knowledge sacrifices some of artificial evolution's ability to explore the design space beyond conventional design rules and methodologies.

Nevertheless, a trend away from the credo of unconstrained evolution towards search strategies guided by human design principles can be observed in recent publications in the field of evolvable hardware: The advantages of an unconstrained artificial evolution process over conventional digital design were first proposed and demonstrated by Adrian Thompson ([Tho98a]), [Tho97], [Tho99],

[Tho00]) during the late 90s. This approach was adopted by a number of researchers, whose work is accounted for e.g. in [Lay01], [Mil97], [Hue99], [Hol00]. Although recently Julian Miller extended the idea of unconstrained evolution insofar as to claim that it should be applied to other physical media that are better suited for this purpose than the typical primitives realized in today's silicon technology [Mil02], a growing number of publications regard the allowance of human domain knowledge as necessary or at least beneficial for the progress of artificial hardware evolution. Prominent examples in the field of analog synthesis are documented in [Koz04c] and [Zeb03], [Zeb01].

Parallel to this trend, a series of experiments employing human designed building blocks have been performed with the proposed evolution system. Thereby, three different problems, namely the synthesis of XOR/XNOR gates, of tone discriminators and that of comparators have been tackled using two different types of building block libraries. Similar material to that concerning the former two topics has already been published in [Lan04].

8.1 Methodology

8.1.1 Rationale

The usage of human designed building blocks (BBs) can be viewed in three different ways: First, utilizing higher level building blocks facilitates the task at hand by narrowing down the design space. Instead of having to reinvent the wheel in every single run, evolution is free to take advantage of circuit primitives that have been proven successful numerous times in human designs. In this vein, the success rate of the automatic design process and the quality of the evolved solutions as well as its convergence speed should be considerably enhanced.

Second, the results presented within the previous three chapters as well as those found in the literature render the success of a direct flat artificial evolution of more complex systems improbable; evolutionary algorithms in general and hardware evolution in particular scale badly with the size of the system to be synthesized. Moreover, regarding current search techniques together with the amount of parallelization and circuit evaluation times feasible with today's technologies, one would not expect to evolve a CPU from scratch. Accordingly, the automatic design of complex systems either requires the usage of high level building blocks or techniques to automatically divide the problem into sub-problems. The proposed approach can thus be viewed as a first step towards the evolution of more complex systems by means of problem decomposition. It should be noted though, that the automatic decomposition itself represents the more difficult problem.

Third, the concept of human-designed building blocks allows to incorporate some a priori problem specific knowledge into the search process. For one, the building blocks themselves usually are chosen to be practical for the design task to be accomplished, while the other senseless or redundant circuit primitives of this complexity are discarded. For the other, the approach taken here suggests a signal flow from left to right and a current flow from top to bottom to the automated design process, where the latter direction is actually more important for case study III than for cases I and II. Since this complies with the conventions for illustrating circuit schematics, the generated circuits are hoped to be easier to understand than those evolved with the plain transistor level genotype described in section 4.4.3. Finally, taking into account some conventional design principles may bias artificial evolution to create circuits that are more robust against variations in environmental conditions like temperature, the exact die, or its position thereon compared to circuits found in a completely unconstrained search.

8.1.2 Related Work

This work was inspired by the cell-structured current-path based synthesis described in chapters 3 and 4 of [Shi01]. In the current-path based synthesis Shibata uses a representation consisting of several sequences of meta structures in which a current flows from the positive power supply voltage vdd to ground or vss. The meta structures are either current sources, transistors or transistor pairs. Crossover is restricted to the exchange of integer current-path forming sequences. Beside the usual mutation operator acting on the properties of each meta structure another mutation operator adds/removes meta structures to/from one of the current paths. This concept is transferred to a cell-based approach which utilizes a stack of 4 transistors connected in series between the power supply voltages augmented with a number of switches to configure the cell. Finally, in [Shi02] the cell-based representation is applied to an array of 9 such cells to intrinsically evolve 3 different computational circuits. The proposed building block concept adopts the idea of meta structures and extends it to allow for user-defined building block libraries and geometries. The topological configuration of Shibata's meta structures are realized by choosing different BBs from the library. Thereby, the user can design building block libraries which strongly suggest the use of distinct directions of signal and current flow to the evolutionary algorithm or constrain the design to be strictly symmetrical. With regard to the crossover operation, existing current paths are preserved by preserving all BBs.

In principle, the FPTA1 and FPTA2 chips designed by Stoica et al. (cf. e.g. [Sto01c], section 3.11) are arrays of configurable cell structures. In this vein, all experiments that utilize more than one of these FPTA cells for the description of their genotype are also closely related to the work proposed here. In fact, in [Zeb01] and [Zeb03] both, human designed as well as evolved building blocks are used as substructures to compose a 4-bit DAC.

In a broader sense, the use of analog building blocks in intrinsic hardware evolution experiments is reported in a variety of publications: First, high level building blocks, i.e. typically operational amplifiers, are frequently combined with passive circuitry to evolve signal processing circuits like filters. The restricted variety of different building blocks may be due to the limited resources offered by the typical off-the-shelf FPAA devices available. While the work found in [Gre03], [Gre04], [Her04] and [She02] focuses on the optimization of a few size parameters of the passive devices in an (almost) fixed topology, the genotypical representation described in [Ozs98] consists of a mixture of higher level functions, topology describing information and sizing parameters and thus includes the evolution of new topologies.

In the world of digital substrates the reported publications on intrinsic hardware evolution range from gate level synthesis of analog circuits ([Hue99], [Tho97] and [Tho00]) to the synthesis of digital signal processing systems by means of parameterized functional building blocks (e.g. [Sek03], [Zha04]). It should be noted though, that the variety of logic functions available in the FPGA cells used as building blocks in the former examples is still richer than that used in the case studies I and II described below. Moreover, the analog behavior of the gates available to the experiments presented in [Hue99], [Tho97] and [Tho00] is probably much closer to that of the simple gates used in experiments documented below.

Finally, the concept of building blocks also plays a role in the following publications based in the field of analog design automation. In [Sri02] the authors report the usage of a small library consisting of two different gain stages in a GP-based evolution of an analog amplifier. Graeb et al. ([Gra01]) on the other hand, facilitate three types of parameter optimizations for a given topology of an analog circuit by their sizing rule method. Thereby the circuit is divided into human defined building blocks ranging from 1 to 4 transistors. The channel dimensions of the transistors forming the identified building blocks are then calculated according to a set of sizing rules derived from human design knowledge. The relation to the work presented here is twofold: For one, some of the sizing

rules established by Graeb et al., as e.g. to choose the identical channel length for certain transistor pairs can also be implemented in the current evolution framework, while the inclusion of others may be realized fairly easily. For the other, the circuit primitives that make up the building block library employed in case study III are possess similar if not identical functionality in many cases.

8.1.3 Building Block Concept

The general idea of the building block concept is to allow artificial evolution to synthesize circuits from user defined building blocks. The implementation shall be based on the cell based genotype representation described in section 4.4.3, which is closely related to the FPTA's structure itself. The user defined building blocks are stored in a library and inserted into the actual genotype. The principal issues that have to be dealt with are where to place and how to connect the building blocks as well as how to determine their channel dimensions.

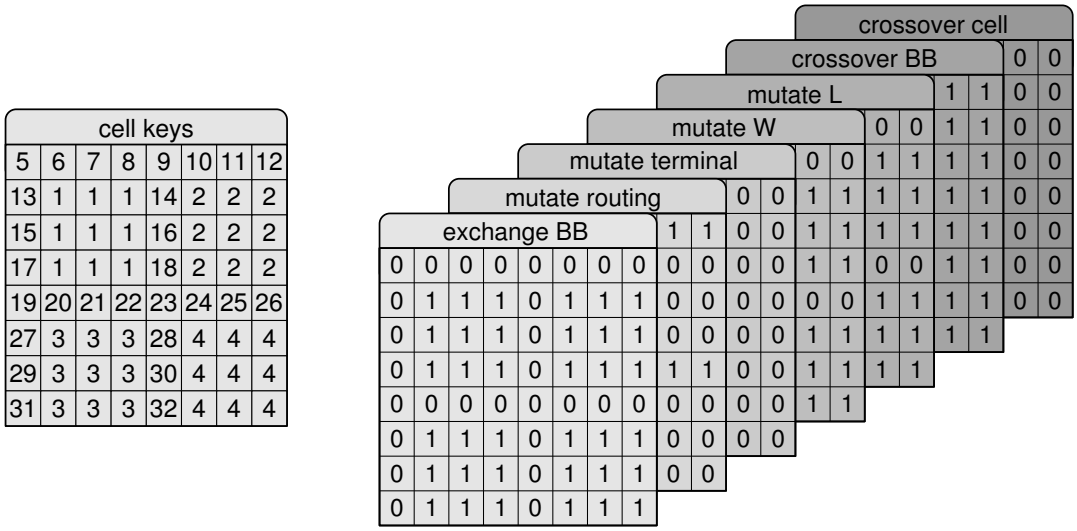
For the sake of simplicity, the locations where building blocks are placed are kept fix. The connections necessary between the building blocks can in principle be realized in two different ways: Either the ensemble of building blocks must be designed such, that all possible BB combinations can emerge from the synthesis process, or the BBs must be embedded in a layer of extra cells that can host the necessary connections and are themselves subject to the evolutionary process. The former approach gives rise to two objections: First, the need to include all necessary routing capabilities as BBs enlarges the BB library unnecessarily. Second, depending on the size of the building blocks the routing blocks may either consume unreasonably large areas or, if the routing capability is mixed with functionality, will complicate and enlarge the building blocks themselves. Hence, the availability of routing cells is considered desirable.

The channel dimensions of the transistors of the BBs can be determined with different degrees of flexibility. The extremes would be either to let the user define the sizing of the building blocks and keep the according values fixed throughout the evolutionary process, or to let the transistor channel geometries be evolved totally freely by the EA. The sizing rules described in [Gra01] constitute a compromise between flexibility and the consideration of prior design knowledge. So far, the implementation of the building block idea is only capable of fixing either the width or length of all transistors in all building blocks. Finally, while place and route as well as the sizing of the building blocks relate to the mutation operators, the crossover operation is required to preserve the building blocks.

8.1.4 Implementation

The concept sketched above is implemented by introducing further attributes to each of the 256 FPTA transistor cell equivalents constituting the genotype: To account for the placement and routing of the building blocks, a numeric *cell key* is assigned to each cell; cells that possess the same key are treated as one building block. The upper left quarter of the cell keys used for case studies I and II are visualized in Fig. 8.1(a). The cutout shows 4 quadratic building blocks of edge length 3 that are labeled with the cell keys 1 to 4 and 28 single cells labeled with the cell keys 5 to 32.

A set of eight attributes referred to as *genetic access rights* defines which genetic operators are allowed to access the respective cell. Again, the upper left quarter of the configuration for case studies I and II is used as an example to illustrate the *genetic access rights*; they are depicted in Fig. 8.1(b) and further explained in Table 8.1. For each of the single cells the genetic operations allowed to the EA are restricted to mutations of the routing bits and the building block based crossover explained below. Cells belonging to one of the 4 building block sites on the other hand can be altered by mutations of the transistor's width and length, the building block based crossover operator and the exchange



(a) cell keys (b) genetic access rights

Figure 8.1: Genetic access rights for a subarray of 8×8 cells: (a) shows the cell keys used to identify cells belonging to the same BB. (b) illustrates the genetic access rights for different variation operators. A '1' signifies that the genetic operator can access the according cell.

building block mutation also explained below. In this way, four building blocks that can be sized freely by the EA are embedded within 28 routing cells that can be used by the EA to connect the building blocks with each other and to the in- and outputs.

Please note that the same arrays of *cell keys* and *genetic access rights* are applied to all individuals of one generation. In the current state, this building block configuration has to be defined by the user and remains fixed for the whole evolution run. Nevertheless, it is conceivable to hand part of the control over to the EA itself, e.g. the placement of building blocks and routing cells, which could be used to minimize the number of routing cells.

The building block structure of the genotype requires new mutation and crossover operators: The new mutation operator exchanges an existing building block with another one randomly chosen from the building block library. On insertion, a new building block possesses the channel dimensions defined in the building block library. It is only in subsequent generations that these values can be changed by the according mutation operators – unless the building block is not exchanged again. This type of sequence guarantees that the newly inserted building blocks possess the functionality intended

abbreviation	description
exchange BB	choose a BB from the library to exchange the existing BB with
mutate routing	flip one of the routing bits of the cell
mutate terminal	choose a new terminal for one of the transistor terminals of the cell
mutate W	choose a new channel width for the transistor of the cell
mutate L	choose a new channel length for the transistor of the cell
crossover BB	use the crossover operator that preserves BBs
crossover cell	use the cell based crossover

Table 8.1: The genetic access rights.

by their designer until their respective genotype is evaluated for the first time. Further optimization can be achieved in subsequent generations.

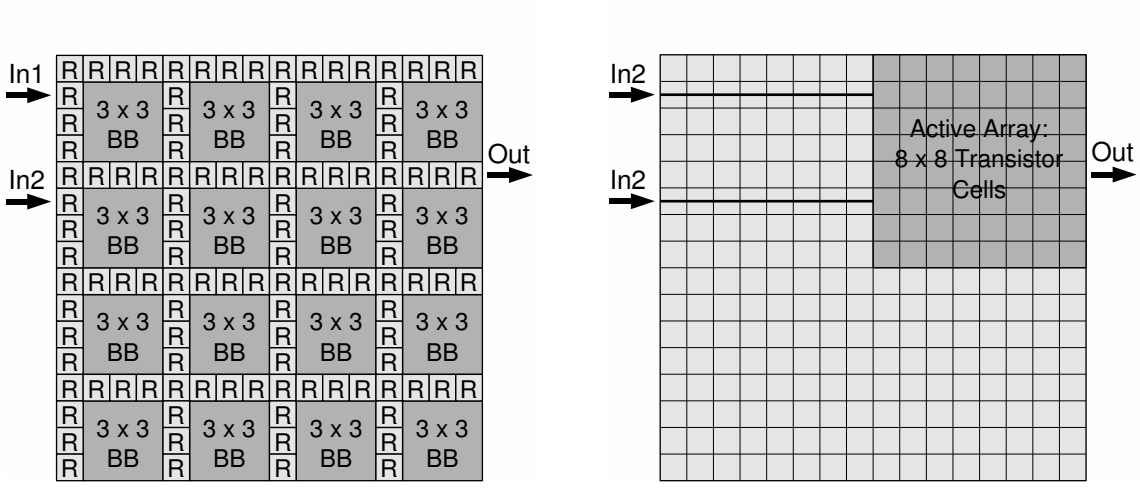
The natural extension of the crossover operator defined for the plain genotype (cf. 4.4.3) would be to extend the rectangular array destined to be exchanged in the crossover operation such, that all building blocks touched by the original rectangle are completely included. Unfortunately, the current implementation of the BB crossover is restricted to exchanging complete building blocks or single cells depending on the cell key configuration for the respective cell. However, since mutation is believed to be much more important for all the experiments presented within this thesis, this should not affect the success of the following evolution experiments too severely. The initialization of the first population is similar to the procedure for the plain genotype (see 4.4.3), in that all initial values are set randomly. Concretely, this means that all building block sites are filled with BBs randomly chosen from the BB library and that the routing of all single cells is determined randomly, too. However, the channel dimensions of all transistors remain unchanged from their values specified in the BB library.

8.2 Experimental Setup for Case Studies I and II

Case study I and II share the same building block library as well as the same genetic algorithm including most of its parameters. Furthermore, their geometric setups are almost identical. In contrast, the respective parts of the experimental setup applied in case study III differ substantially from those used in case studies I and II. Hence, these similar aspects of the setup for case studies I and II are summarized in this section.

8.2.1 Geometrical Setup for Case Studies I,II

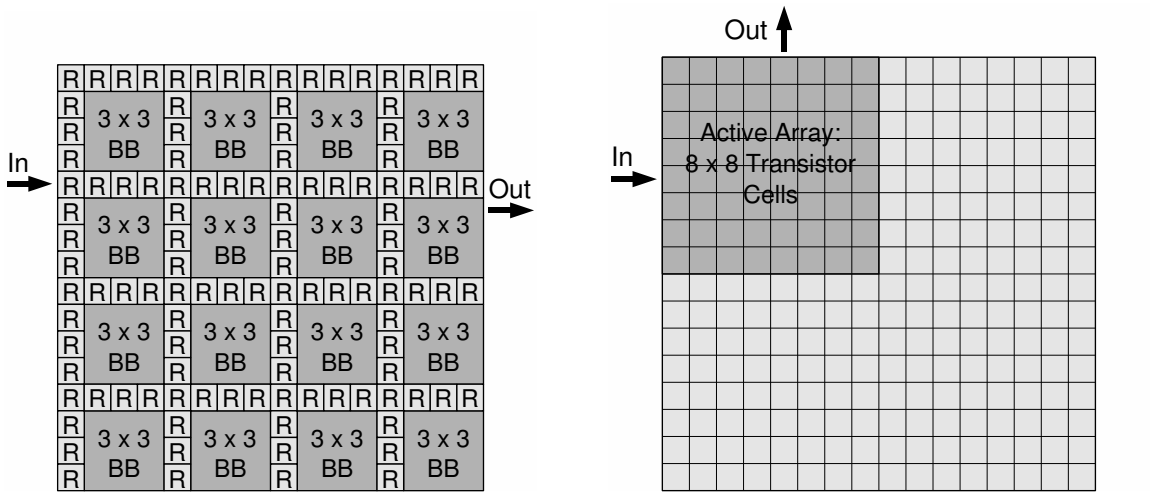
For all three case studies the results obtained by using the above building block concept are cross-checked against similar experiments based on the plain cell genotype. The main difference between those two types of experiments is the geometrical setup of building blocks and cells on the FPTA chip. The geometrical setups for case studies I and II are depicted in Fig. 8.2 and Fig. 8.3, respectively.



(a) gates, building blocks

(b) gates, plain genotype

Figure 8.2: Geometrical setup for case study I: (a) is used to host the building block experiments. R denotes routing cells. (b) illustrates the setup for experiments utilizing the plain genotype.



(a) tone discriminator, building blocks (b) tone discriminator, plain genotype

Figure 8.3: Geometrical setup for case study II: (a) is used to host the building block experiments. R denotes routing cells. (b) illustrates the setup for experiments utilizing the plain genotype.

For both case studies the complete area of the chip is used by 16 building blocks that can be connected via 112 routing cells as shown in Fig. 8.2(a) and Fig. 8.3(a). The building blocks are squares of an edge length of 3 transistor cells. In accordance with the explanations in section 8.1.3 and Fig. 8.1, all cells denoted with an R are restricted to their routing capabilities and all 16 building block sites allow the GA to exchange BBs from the library as well as to change the channel geometry of the transistors therein. Since the inputs are located at the left hand side of the chip and the outputs on the right hand side, the macroscopic signal flow is forced to be horizontally from left to right.

The geometrical setup for the reference experiments based on the plain genotype is illustrated in Fig. 8.2(b) and Fig. 8.3(b) for case study I and II, respectively. The active chip area that can be used by the GA is limited to a square of 8×8 cells to constrain the design space to a size more comparable to that of the building block experiments but still large enough to allow for the emergence of good solutions. While the overall signal flow for the reference experiments for case study I is still directed from left to right, the output for the reference experiments of case study II is positioned on the upper side of the chip for the sake of simplicity. Although this represents a slight inconsistency, experiences gathered in other experiments suggest that this should not affect the results too badly.

8.2.2 Building Block Library for Case Studies I,II

Digital gates have proven of great value in countless designs, which to some extent is due to their high level of abstraction from the actual device behavior as well as to their good encapsulation. That is, digital gates possess few well-defined in- and outputs. In particular, CMOS implementations of digital gates operate only in the voltage regime so that one does not have to worry about current flows yet. These advantageous properties render digital gates as ideal candidates for a first proof of principle test of the proposed building block concept. With this in mind, a small library of 4 of the most simple digital gates, namely an inverter, a buffer, a NOR and a NAND gate, has been composed. Fig. 8.4 illustrates the according circuit schematics as well as their building block implementations by means of the FPTA's transistor cells.

Due to the limited routing capabilities available on the FPTA, the NOR and NAND gates are impossible to implement using only 2×2 transistor cells. Hence, a square of 3×3 cells is allotted

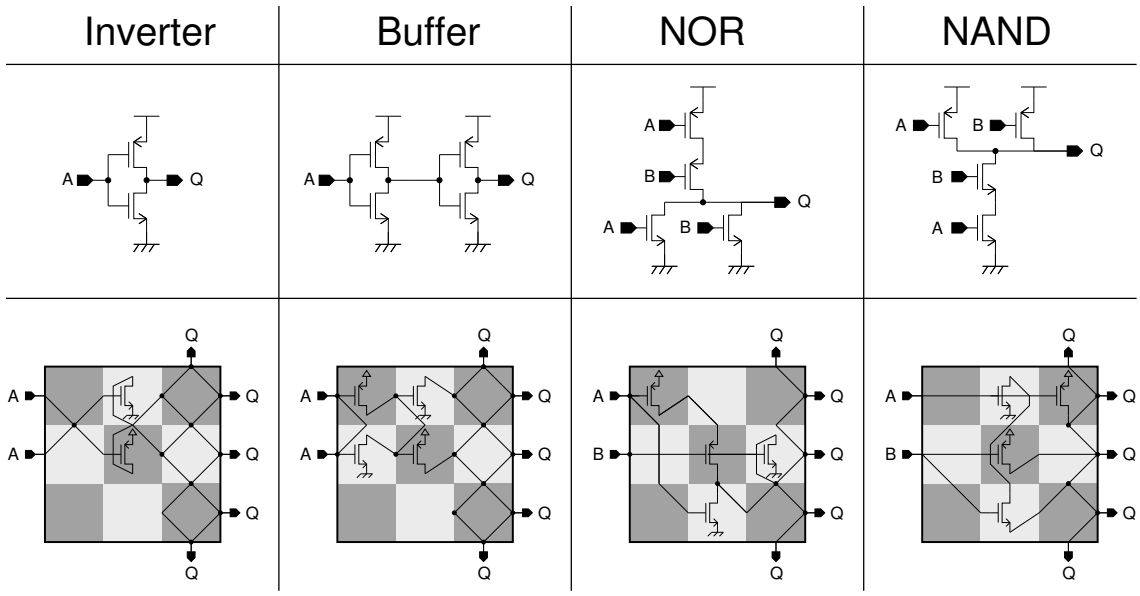


Figure 8.4: Building block library used for case studies I and II. The second row shows the schematics of the used circuits and the third one displays their implementation as a block of 3×3 transistor cells. PMOS transistor cells are shaded in darker, NMOS transistors in lighter gray .

to all of the building blocks. According to the global signal flow defined by the geometrical setup shown in Fig. 8.2(a) and Fig. 8.3(a), the two inputs A and B are located at the left hand side of each building block and the output Q at its respective right hand side. In case of inverter and buffer, the two inputs are short-circuited and the output Q is available at 5 cell borders for all gates to facilitate the evolution of the necessary connections between the building blocks. Please note that apart from these in- and outputs, no further signals are accessible at the boundary of the building blocks to restrict the EA to using only their pure functionality. The channel dimensions of all transistors are set to medium values of $W = 4\,\mu m$ and $L = 2\,\mu m$ for all building blocks. This choice leaves some head room for optimization, especially for the evolution of tone discriminators in case study II. For the evolution of X(N)OR gates in case study I, a shorter gate length and a more balanced switching point for the NOR would probably be a better starting point.

8.2.3 Evolutionary Algorithm for Case Studies I,II

For case studies I and II the same straightforward genetic algorithm is used as in the previous three chapters. Analog to chapters 5 and 6 truncation selection is employed. The GA parameters, which are summarized in Table 8.2, are similar to those applied in 6 (compare Table 5.6) except for a decrease in the reproduction fraction and an increase in the crossover and mutation fraction; this should cause a slight drop in selection pressure. The number of mutation rates has grown by one to account for the new building block exchange operator. The small crossover block size of 3×3 (for a building block) and 1 (for a single cell) transistor cell stems from the unfinished crossover operator.

8.3 Case Study I: XOR/XNOR Gate

The building block concept and its implementation are first tested by synthesizing the two most complex symmetric 2-input gates XOR and XNOR. On one hand, theses types of gates proved to be difficult to evolve with the plain cell genotype as is described in section 5.4. On the other hand, the

GA Parameter	X(N)OR: BB	X(N)OR: Cell	TD: BB	TD:Cell
population size	50	50	50	50
reprod. fraction	0.1	0.1	0.1	0.1
mutation fraction	0.8	0.8	0.8	0.8
crossover fraction	0.8	0.8	0.8	0.8
crossover rate	0.2	0.2	0.2	0.2
mut. rate routing	3%	3%	3%	3%
mut. rate W/L	3%	3%	3%	3%
mut. rate term. con.	–	3%	–	3%
mut. rate BB	2%	–	2%	–
no. of used blocks	16	–	16	–
no. of used cells	112	64	112	64
crossover block size	3/1	6	3/1	6
no. of generations	5000	5000	10000	10000

Table 8.2: Genetic algorithm parameters used for case studies I and II.

synthesis of XOR/XNOR gates should be significantly alleviated by the support of a building block library of simpler logic gates. In other words, this first case study is rather a test whether the proposed methodology *can* be beneficial than a proof that it *will* facilitate the synthesis of more interesting circuits.

Altogether 4 experiments featuring 30 runs each are presented in this first case study. For each of the 2 gate types 2 experiments are performed, one based on the proposed building block concept and the other employing the plain cell genotype. The results presented in section 5.4 are not taken into consideration as reference experiments, because they were attained with a different experimental setup: Apart from GA parameters and test pattern the external analog circuitry as well as the timing of the analog stimuli have been changed. Nevertheless, both of these plain cell experiments are believed to produce similar results.

8.3.1 Problem definition

Target Behavior. The dc output characteristic at which the artificial evolution of this case study is targeted is identical to that defined in section 5.2.1: The truth table for the XOR and XNOR gate are specified in Table 5.3. As described in Table 5.4, the inputs are considered a logic zero, if $V_{\text{in1,2}} \leq 2\,\text{V}$ and a logic one, if $V_{\text{in1,2}} \geq 3\,\text{V}$, whereas the outputs are required to match the power supply rails or $0\,\text{V}$ and $5\,\text{V}$ for a logic zero and one, respectively.

Fitness Function. The only difference to the fitness functions defined in section 5.2.1 refers to the number of test points, which is reduced from 512 to 256. Therefore the fitness function used during evolution – again taken to be the sum of squared errors (SSE) – is given by:

$$\text{SSE} = \sum_{i=0}^{256} (V_{\text{tar}}(i) - V_{\text{out}}(i))^2 \quad , \tag{8.1}$$

whereas the root mean square error per data point in mV is used for the presentation of the results:

$$F = \text{RMSE} = \sqrt{\frac{\text{SSE}}{256}} \cdot 1000 \quad . \tag{8.2}$$

Test Pattern. The test pattern shown in Fig. 8.5 is identical to that shown on the left hand side of Fig. 5.2, except for four differences: First, the range covered by the input voltages V_{sweep} and V_{set}

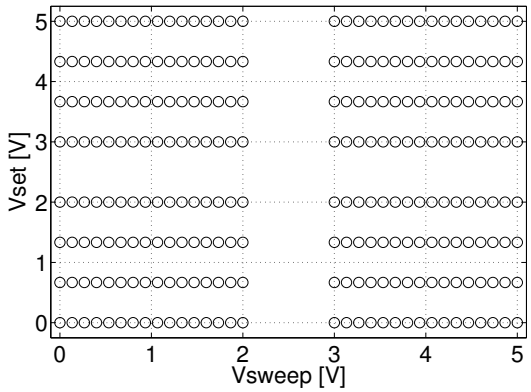


Figure 8.5: Pattern to test the candidate X(N)OR circuits during evolution and in the verification tests.

stretches from 0 to 2 V and 3 to 5 V. Second, the number of different V_{sweep} voltages is reduced from 64 to 32 for each of the 8 V_{set} values. Third, the voltages are applied by choosing randomly from 10 random orders instead of choosing each voltage randomly; in particular, no input voltage is applied twice for one test of one candidate circuit. Finally, the settling time – the time between the application of the second input voltage and the readout of the output voltage – is fixed to exactly 4.825 µs instead of being subject to variations caused by the operating system. The total sampling time thus amounts to 5.7 µs instead of 10 to 15 µs on average.

8.3.2 Results

The results are analyzed in the same manner as in the previous chapters: The fitness is calculated as the root mean square error per data point in mV by means of (8.2). The RMS error is determined for 100 verification tests; if not denoted otherwise, the highest of these error values is used for the analysis.

8.3.2.1 Comparison of the Results of the Different Experiments

The results for all runs of all four experiments are compared in the histograms shown in Fig. 8.6. The difference in the success of the artificial evolution experiments that do and do not use building blocks is enormous: While the experiments based on the plain cell genotype fail to produce any circuit that comes close to satisfying the target specifications, the experiments based on the building block concept excel with yield rates of 50 and 90% for the XNOR and XOR, respectively (where a solution with an average RMS error of less than 50 mV is considered perfect).

The distribution of fitness values attained for the plain cell experiments looks similar to that shown on the left hand side of Fig. 5.5. Actually, the results seem to be slightly worse than those presented in chapter 5. Apart from the smaller number of samples, this may be due to the differences in the test patterns and the timing schemes as well as to the different GA parameters. The fairly large difference in the yield rates achieved for the XOR and XNOR gate in case of the building block base experiments are believed to be of statistical rather than systematic nature. While similar experiments using another set of GA parameters produced yield rates of 80 and 83% for the XNOR and XOR gate respectively, a series of experiments that required the gates to settle in 3.23 instead of 4.825 µs resulted in a yield of 73% for the XNOR and only 50% for the XOR.

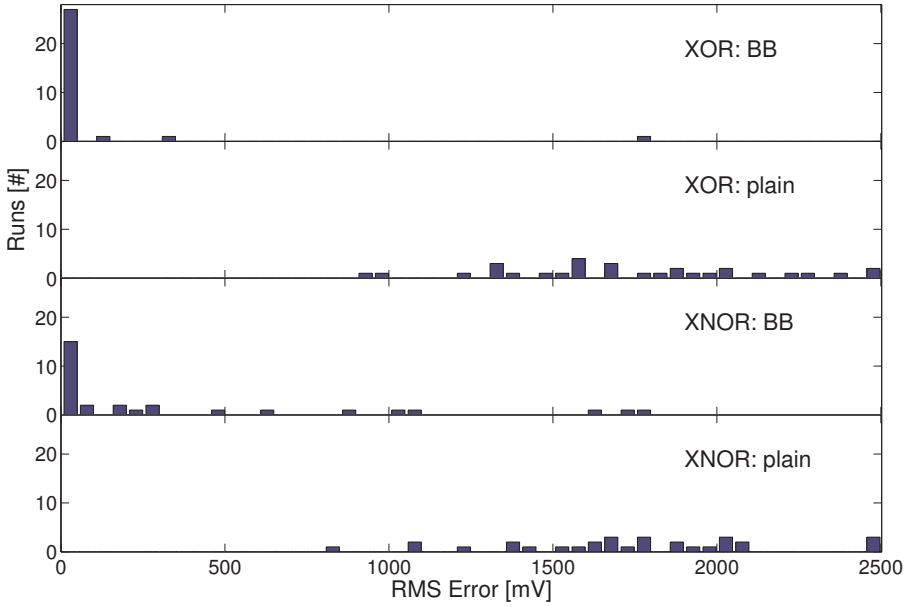


Figure 8.6: Worst fitness obtained from 100 verification tests for all 4 experiments. The bin size amounts to 50 mV.

8.3.2.2 Voltage Characteristics of the Evolved Logic Gates

For each experiment the output characteristic of the best evolved circuit is shown in Fig. 8.7. Both, the XOR as well as the XNOR circuit synthesized by means of the building block methodology exhibit a perfect quasi-DC response. The solutions found without building blocks classify most of the input voltages correctly, but fail to reach the power supply rails for some of the four input cases. That is, for a relaxed specification, in which the input voltages are guaranteed to be outside the interval between 1 and 4 V and in which $V_{\mathrm{OL}} \leq 1\,\mathrm{V}$ and $V_{\mathrm{OL}} \geq 3\,\mathrm{V}$ would be sufficient, these solutions would work as the XOR and XNOR gates. As was already mentioned in section 5.4, the target specification is overly ambitious in that even the XOR's and XNOR's standard cell implementations of the FPTA's fabrication process do not satisfy it (for simulation results refer to 5.8). The fact that circuits that fulfill these strict requirements can be synthesized by the proposed building block concept underlines its potential to enhance the evolution of analog circuits.

8.3.2.3 Test on a Second Chip

In order to test to what extent the evolved circuits rely on the exact conditions present on the very die they are evolved on, the verification tests are repeated on a second FPTA chip for all best-of-run solutions. Table 8.3 summarizes the mean RMS errors averaged over all runs as well as the percentage deviation of the errors measured on chip 2 from those measured on chip 1 (the chip used during the evolution experiments).

The mean RMS errors for the experiments abstaining from the building block concept are practically identical on both chips. In contrast, the circuits synthesized from building blocks perform 30 and 59% worse for the XOR and XNOR respectively. Since the absolute errors are considerably smaller in these cases, expecially for the XOR experiment, the question arises whether this performance degradation is due to offset and gain differences between the two evolution systems, or if the performance of single circuits breaks down on the second chip. In order to answer this question, the correlation

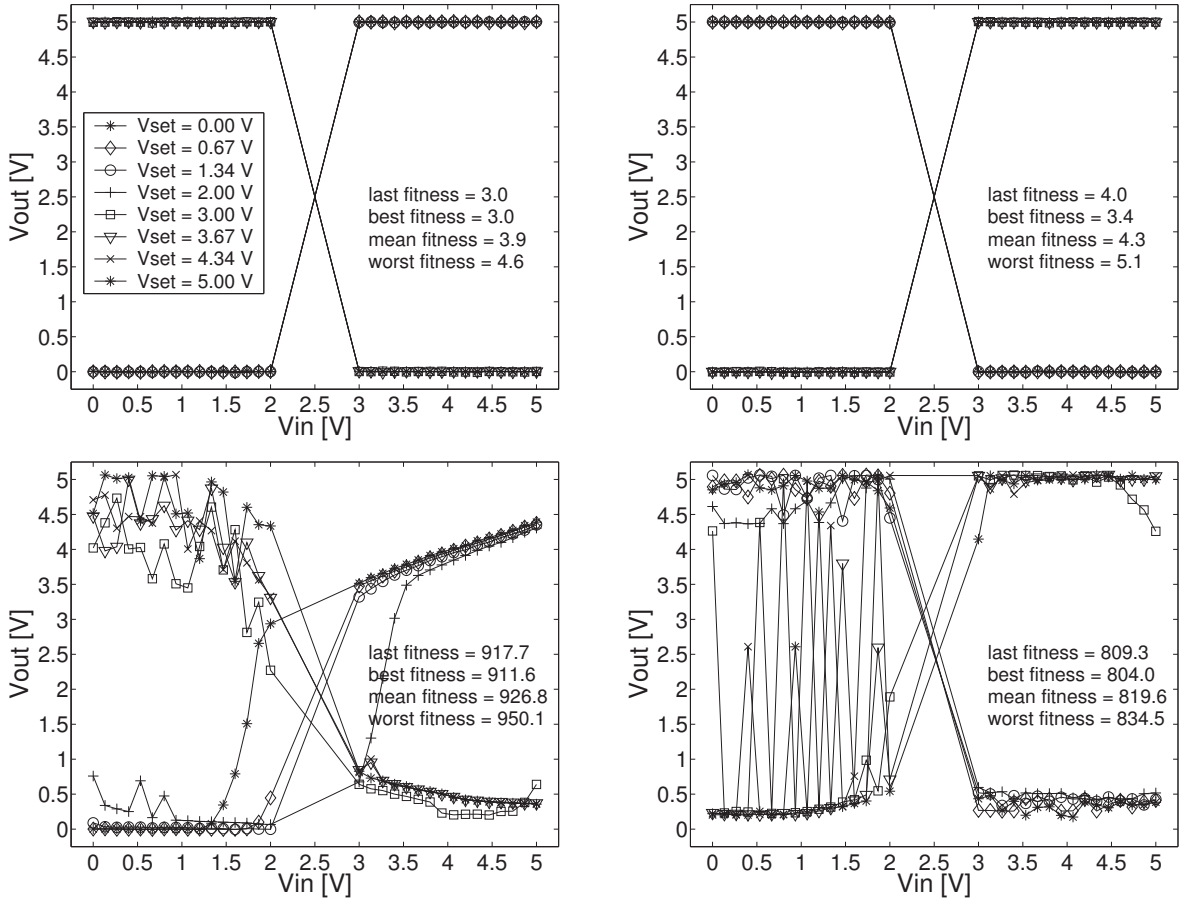


Figure 8.7: Performance of the best-of-series runs, where best refers to the worst value obtained from 100 verification tests evaluated in terms of the fitness criterion used during the evolution process.**Upper left:** XOR evolved using building blocks. **Lower left:** XOR evolved using plain Genotype. **Upper right:** XNOR evolved using building blocks. **Lower right:** XNOR evolved using plain Genotype. The legend imprinted in the graph in the top left corner is used for all four plots.

coefficient R defined in appendix C is given in the last row of Table 8.3. The fact that R is close to one for the XOR evolved with building blocks indicates that the performance of all these circuits is evenly degraded on the second chip. For the building block based XNORs R equals 0.7; hence the degree of malfunctioning on the second chip is believed to be distributed more inhomogeneously.

In conclusion, the circuits incorporating building blocks seem to be less robust than their counterparts evolved from scratch. This is a surprising result, because logic gates are designed to abstract from the physical nonidealities of the underlying substrate. However, since the absolute performance

Experiment	XOR: BB	XOR: Cell	XNOR: BB	XNOR: Cell
F1: RMS error chip 1	101.36	1740.42	366.81	1741.89
F2: RMS error chip 2	131.57	1745.30	581.36	1734.63
Deviation [%]	29.81	0.28	58.49	-0.42
R(F1,F2): Correlation coefficient	0.989	0.998	0.700	0.960

Table 8.3: Comparison of the worst RMS errors obtained in 100 verification tests on two different dice.

of the building block based circuits exceeds that of the circuits evolved without building blocks, the former ones possess a higher chance of failing than the latter ones, which thus puts the above statement into perspective.

8.4 Case Study II: Tone Discrimination

In this second case study the synthesis of circuits that distinguish between square waves of different frequency, so-called tone discriminators, is sought. Several reasons lead to the consideration of this kind of problem: First, the tone discrimination task is meant as a reminiscence to the work of Adrian Thompson [Tho97], [Tho00]), who introduced and solved the problem by means of intrinsic hardware evolution on an FPGA. Second, test pattern, fitness function and analysis of the problem are naturally situated in the time domain. Thus, this case study also serves as a demonstrator for the ability of the proposed evolution system to perform transient analyses, as was already advertised in section 4.3.2. Third, in the absence of an external clock, the problem is inherently analog; this naturally raises the question whether artificial evolution will benefit from using digital building blocks. Finally, the problem of discriminating square waves of different frequencies suits hardware evolution well: On one hand, the problem definition in terms of test patterns and fitness function is relatively simple; on the other hand, the design of an analog tone discriminator is a nontrivial (and uncommon) task.

Apart from Thompson who sought the discrimination of 1 and 10 kHz tones, the problem was recently tackled by Harding et al. ([Har04a]) using a completely different hardware: A liquid crystal display is connected to 64 lines that can be driven or read out by 8 channels, to which they are connected via multiplexers. One channel each is used for the ground potential, the input stimulus and the output monitoring. The remaining 5 channels are set to a static voltage between -10 and 10 V. The way the 8 channels are multiplexed to the 64 LCD[1] terminals as well as the 5 analog voltages are controlled by the GA. The authors claim to reach a useful tone discrimination after as little as 6,000 circuit evaluations.

8.4.1 Problem Definition

8.4.1.1 Test Pattern

In contrast to the original experiment, the frequencies to be distinguished were shifted to 40 and 200 kHz in order to decrease the time necessary for one fitness evaluation. As can be seen from Fig. 8.8 the input is stimulated with a sequence of 200 and 40 kHz tones. Both frequencies are applied twice in an interchanging order. The bursts last 100 µs each resulting in 20 periods per burst for the higher and 4 periods for the lower frequency. The output is sampled with a frequency of 2 MHz, resulting in 800 test points equally distributed in the total testing time of 400 µs. As a measure to prevent successful candidate solutions from exploiting the charge distribution left from the test of its predecessor, a randomly created gene was written to the chip before the next candidate solution was downloaded and tested. This procedure was introduced in section 7.2.6 on page 208 as a *substrate reset*[2].

[1] Liquid Crystal Display

[2] Actually, for the BB-based experiments (7–11) the substrate reset does not make much of a difference: The building blocks themselves are well defined without any floating components. The routing cells, on the other hand, do not provide any active components anyway. Thus, there should not be any special charge distribution left over from the previously tested circuit. Moreover, due to a software flaw, the substrate reset was only allowed to download changes to the width and length of the BBs's transistors and the routing itself. In fact, it was verified experimentally that the effects of this kind of substrate reset are negligible

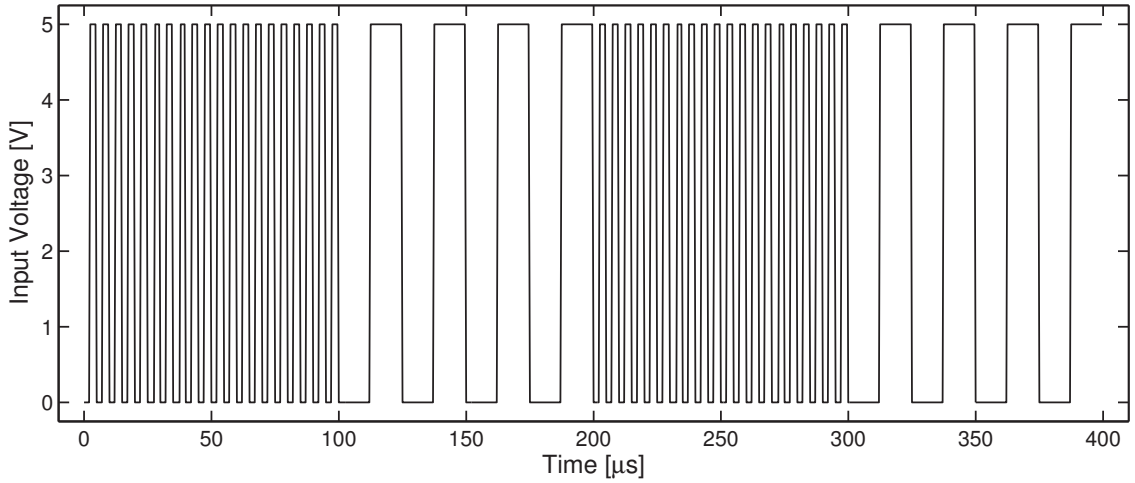


Figure 8.8: Pattern to test the candidate tone discriminators during evolution and in the verification tests.

8.4.1.2 Fitness Function

During the evolution process the fitness is evaluated by

$$\text{Fitness} = \sum_{i=1}^{800} (V_{\text{tar}}(i) - V_{\text{out}}(i))^2 + w \sum_{i=2}^{800} (V_{\text{out}}(i) - V_{\text{out}}(i-1))^2 \quad , \tag{8.3}$$

with the target voltage defined as

$$V_{\text{tar}} = \begin{cases} 0,5 & \text{for} \quad f = 200\,\text{kHz} \\ 5,0 & \text{for} \quad f = 40\,\text{kHz} \end{cases} \quad . \tag{8.4}$$

The actual $V_{\text{tar}}(f)$ is chosen to minimize the fitness value; thereby the GA is relieved of the constraint of finding a solution with a prescribed output polarity. While the left term of (8.3) yields the sum of squared deviations from the target voltage (8.4), the right sum penalizes unwanted glitches and oscillations of the output. The weighting factor w was varied between 1 and 10 to investigate its influence on evolution's success.

However, for the analysis of the results the fitness is again calculated as the root mean square error per data point given in mV:

$$\text{RMS Error [mV]} = \sqrt{\frac{\sum_{i=1}^{800} (V_{\text{tar}}(i) - V_{\text{out}}(i))^2}{800}} \cdot 1000 \quad . \tag{8.5}$$

8.4.1.3 Overview of the Experiments

All in all 11 experiments were carried out that are grouped into 2 series of experiments in Table 8.4. Thereby two different parameters are varied. First, the experiments 1 to 6 belonging to series 1 are based on the plain cell genotype, whereas evolution was allowed to use building blocks in the remaining experiments of series 2. Second, the weighting factor w for the second term in the fitness function (8.3) that is meant to penalize unwanted glitches and oscillations is varied from 0 to 10 in series 1 and varied from 1 to 10 for series 2 as stated in Table 8.4. One may object that the variation of the penalty factor w is slightly off-topic; nonetheless, its inclusion lends more statistical weight to the presented results, as the experiments are otherwise identical and its effect rather subtle.

Series No.	Experiment No.	BBs used	Penalty Factor	# of Experiments
1	1 – 6	no	0,1,2,3,5,10	6
2	7 – 11	yes	1,2,3,5,10	5

Table 8.4: Summary of the 2 series of tone discriminator experiments.

8.4.2 Results

8.4.2.1 Comparison of the Different Experiments

Fig. 8.9 displays the histograms for all 11 experiments. For each run the RMS error is calculated to be the maximum error of 100 verification tests using 8.5. None of the experiments produced a perfect solution to the discrimination task; the best-of-series solutions typically achieve RMS errors around 750 mV regardless of the used methodology. Nevertheless, the experiments relying on the building block concept (7–11) produce circuits of this medium quality with RMS errors ranging between 600 and 1200 mV at a much higher rate than those based on the plain cell genotype (1–6). In contrast, at least a third of the runs of the non-BB experiments end up with error values around 2500 mV. As an RMS error value of 2500 mV per data point is easily achieved by clamping the output to 2.5 V independent of the input signal, this accumulation is likely to be caused by a local minimum. While

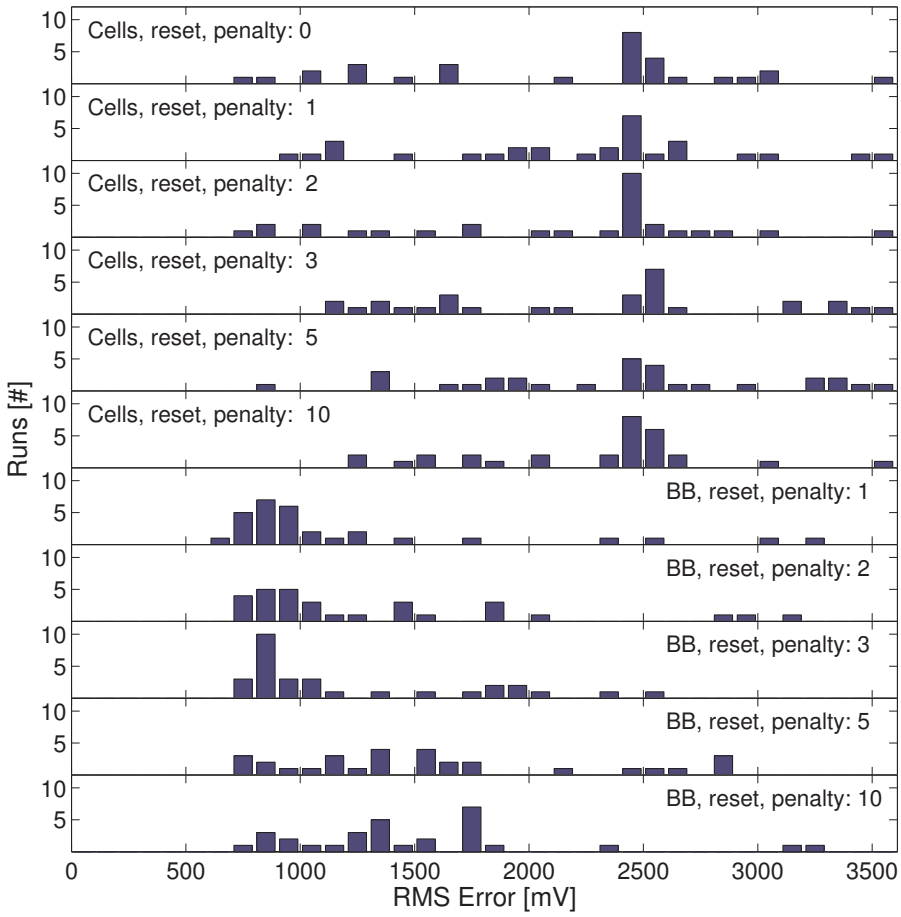


Figure 8.9: Worst fitness obtained from 100 verification tests for all 11 experiments. Bin size = 100 mV

many of the non-BB runs get trapped in this local optimum, this is hardly ever the case for the building block based experiments. In this sense, the proposed building block concept again leads to a significant performance increase.

Since the local optimum at 2500 mV can be found so easily, it can be inferred that the performance of the evolved circuits exhibiting error values beyond 2500 mV is severely degraded in at least one of the verification tests, i.e. they are unstable. In fact the RMS error of only one out of the 480 best-of-run solutions exceeds 2500 mV for the error values recorded at the end of the evolution process. Although this performance break-down can be observed BB-based as well as for non-BB experiments, it seems to occur more frequently for the non-BB experiments.

Throughout both series of experiments, a larger weighting factor w for the glitch and oscillation penalty term in (8.3) tends to slightly decrease the achieved RMS errors. This can be explained as follows: Unfortunately, the penalty addend does not only penalize *unwanted* glitches and oscillations but also the *desired* transitions between the logic high and low states discriminating the different frequencies. Thus, the fitness criterion of (8.3) may sometimes lead to counterproductive selections. More generally speaking, the additional penalty term changes the fitness landscape such, that the evolutionary search process is rather complicated than facilitated.

8.4.2.2 Best-of-Series Solutions

In order to convey an impression of the quality of the attained tone discriminators the actual output behavior of the best-of-series solutions is investigated in two regards: First, the circuit response to the test stimulus used during evolution is shown on the left hand side of Fig. 8.10. A second test mode is used to analyze whether the best-of-series solutions are able to solve the actual question underlying the target description of (8.4), i.e. to distinguish between *low* and *high* frequency tones. Therefore, the successful circuits are tested with square waves of different frequencies that range from 1 kHz to 1 MHz. The test frequencies are logarithmically spaced with a density of 10 samples per decade. Each frequency is applied for 4 periods to comply with the number of periods used for the 40 kHz tone test pattern during evolution. During the first two cycles of the square wave the circuit is allowed to adapt to the new frequency; its response is then integrated over the last two cycles. The sampling frequency is varied to minimize the size of the test pattern while satisfying the constraint that for each period of a square wave at least 100 sampling points are used. Since the sampling frequency is limited to 20 MHz this constraint cannot be met above 200 kHz. The response to this frequency sweep is displayed on the right hand side of Fig. 8.10 for the best-of-series solutions.

As can be seen from the left hand side of Fig. 8.10 both best-of-series solutions do clearly discriminate the two different input frequencies. However, the circuit responses reveal the following imperfections: For the higher frequency of 200 kHz, the output is close to the desired 5 V, but both circuits fail to completely suppress the feedthrough of the input signal. In case of the lower test frequency, the glitches impair the output signals that are otherwise close to the gnd potential even more gravely, especially for the best of the non-BB series solution. In fact, the evolution experiments produced solutions that outperform the circuits shown in terms of the minimum distance of the output voltages for the different frequencies; their respective output signals just happened to be farther away from the power supply rails *on average* and thus attained a higher RMS error value. Accordingly, the chosen fitness criterion seems to be a suboptimal formulation of what the circuits actually have been expected to do. It should be noted though, that in principle the presented solution should easily be extended to perfectly solve the posed discrimination task by adding an inverter or buffer to the output whose switching point is adjusted to the gap between the output voltages for the respective input frequencies.

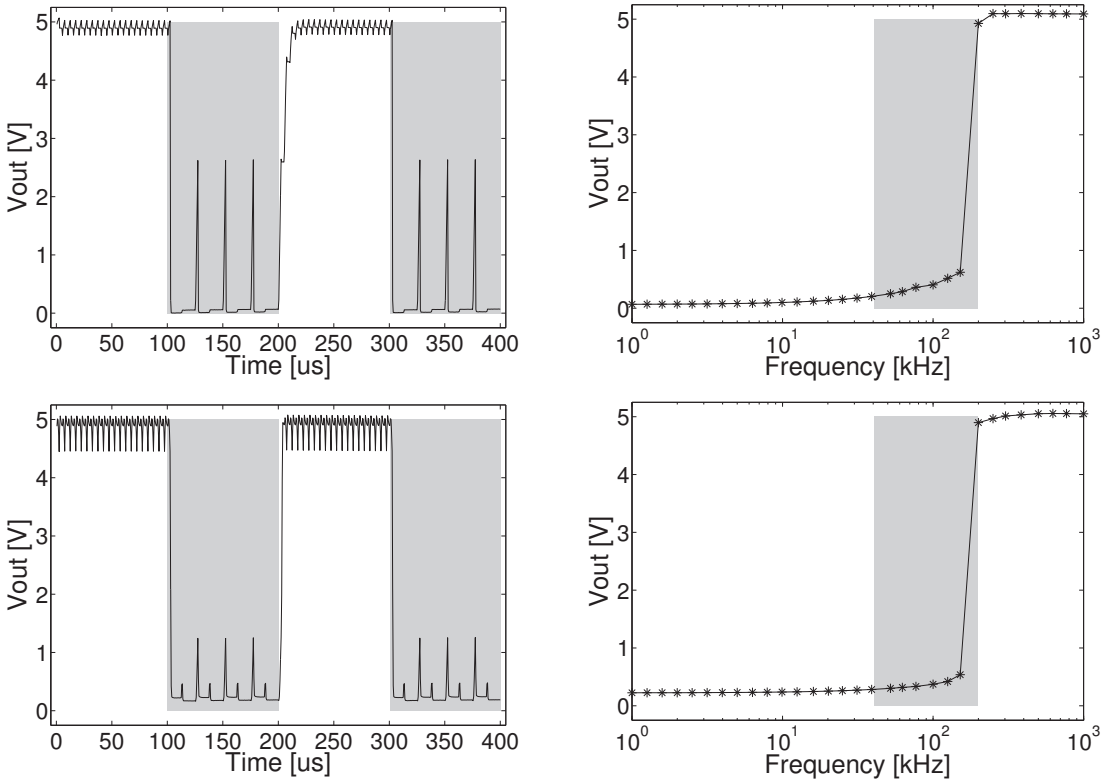


Figure 8.10: Performance of the best-of-series runs, where best refers to the worst value obtained from 100 verification tests evaluated in terms of the fitness criterion used during the evolution process. From top to bottom, the best results for series 1 to 2 are plotted. **Left:** Output characteristic used for the fitness tests during evolution. Areas shaded in gray denote input frequencies of 40 kHz. **Right:** Output characteristic for the frequency sweep test. The areas shaded in gray mark the intermediate frequency interval from 40 to 200 kHz in which a transition is expected to occur.

The integration of the output voltages over two periods of the square wave is more closely related to the used fitness criterion defined in (8.4); the glitches seen in the graphs on the left hand side of Fig. 8.10 are averaged out in the resulting curves for the frequency sweep displayed on the right hand side of the figure and the resulting curves thus look more encouraging. Both best-of-series solutions generalize well to the tested frequency range of 1 kHz to 1 MHz. It is interesting to note, that the transition from low to high is a) sharper than necessary and b) occurs closer to the higher of the two test frequencies, a behavior that is not observed for all of the proper circuits evolved, but for their majority. The latter observation may be explained qualitatively by the fact that the tone discrimination involves some sort of lowpass filtering, which is easier to achieve for higher frequencies in the respective frequency band, as was already shown in section 7.4.

8.4.2.3 Verification Tests

Finally, stability as well as portability of the attained tone discriminators shall be analyzed to account for their expected reliability and robustness. The stability of the evolved circuits is again studied on the basis of the spread among the RMS errors taken from the 100 verification measurements. In principle, analogous definitions of equation (6.17) and Table 6.5 used in section 6.2.1 are applied to the data recorded for the tone discriminators[3]. Since the *last* value obtained at the end of the evolution

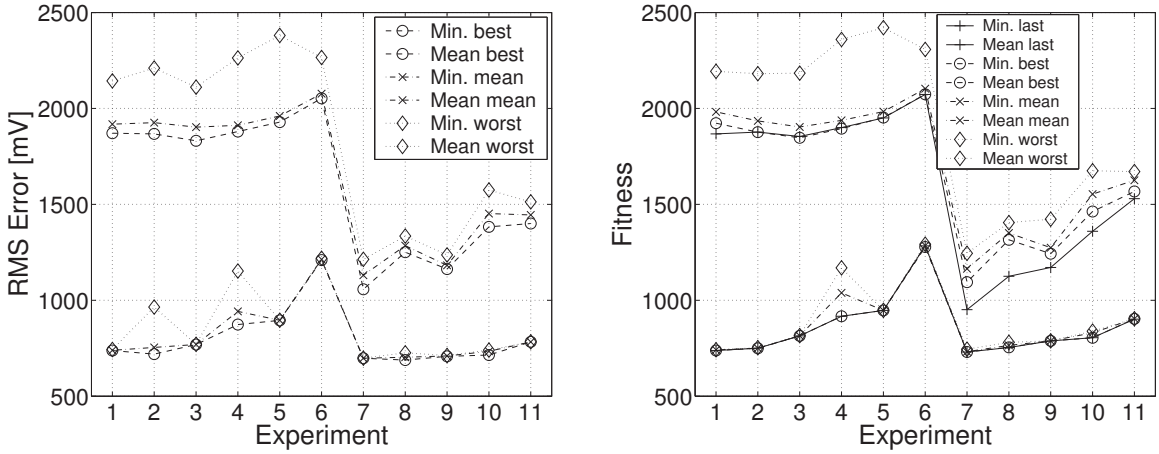
[3]The only difference being that r here takes on values from 1 . . . 30 instead of from 1 . . . 20 as in the original definition.

of each run includes the penalty term as defined in (8.3), the results are once presented as the RMS error defined in (8.5) forgoing the unavailable *last* values and once in terms of the full fitness criterion used during evolution, thus including these *last* values. In order to allow for a comparison with the plain RMS error values, the fitness values obtained from (8.3) are converted to the root mean square fitness per data point in mV:

$$F = \mathrm{RMSF} = \sqrt{\frac{\mathrm{Fitness}}{800}} \cdot 1000 \quad . \tag{8.6}$$

Although the resulting RMS fitness values can easily be compared to the plain RMS error values used beforehand, they lack a physical meaning because they nonlinearily incorporate the additional glitch penalty addend.

While Fig. 8.11(a) illustrates the minimum and mean best, mean and worst RMS errors obtained from the verification tests according to (8.5), Fig. 8.11(b) displays those computed by means of (8.6) plus the two additional curves for the minimum and mean last values obtained at the end of the respective evolution runs. Although throughout all experiments a considerable amount of spread



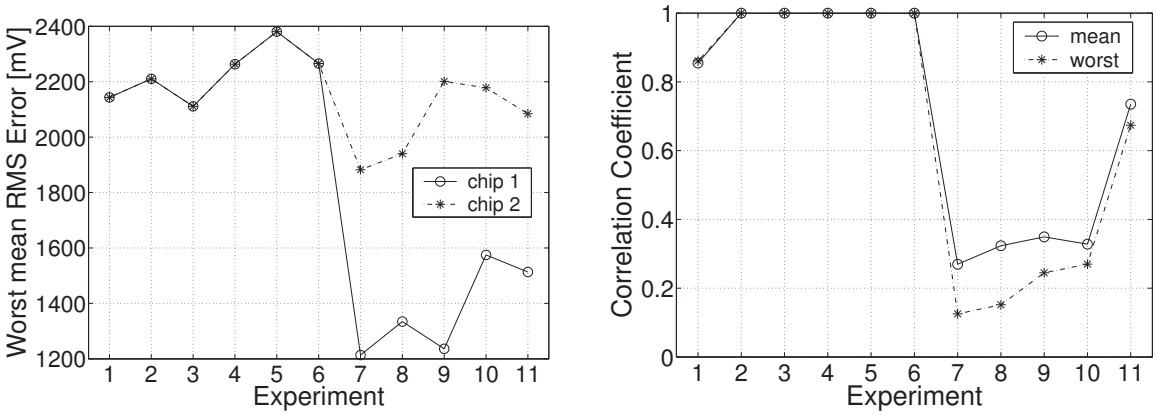
(a) fitness without oscillation penalty

(b) fitness including oscillation penalty

Figure 8.11: Comparison of the minimum and mean last, best, mean and worst fitness values. While last refers to the last fitness value measured at the end of the evolution process, best, mean and worst denote the minimum, mean and maximum RMS error values obtained from the 100 verification measurements. Minimum and mean are then calculated from the ensemble of those according 10 values for each evolution run. For (a) the fitness was reduced to the RMS error, for (b) the fitness criterion used during evolution was utilized.

between the last/best and the mean and worst error and fitness values can be observed, the better circuits can be expected to maintain a satisfactory functionality within the 100 verification tests. In particular, all of the best-per-experiment solutions except for those of experiments 2 and 4 exhibit a virtually constant performance in all of the 100 verification tests. On average, the circuits of series 2 – the BB-based experiments – manage to reproduce their performance more reliably than those of series 1 (experiments 1–6). This may be caused by the higher chances of encountering the FPTA in an unfavorable condition introduced by the substrate reset. On the other hand, in case of the BB-based experiments 6–11, the last fitness value is considerably smaller than the best mean fitness value averaged over all runs of the respective experiment, an absolute novelty within the results presented so far. Is this a harbinger of lacking robustness? Finally, the graphs shown in Fig. 8.11 confirm a) that the fraction of the glitch and oscillation penalty addend in the fitness criterion of (8.5) is relatively small and b) that it nonetheless is detrimental to the success of the evolution experiments.

All the evolved circuits are tested on a second FPTA chip to investigate to which extent they rely on the actual conditions present on the very die they were optimized for. The results are presented in terms of the worst mean RMS error per experiment calculated from (8.3). Fig. 8.12(a) compares the according RMS error values obtained on the two different dice. In order to account for the distribution of the performance differences, Fig. 8.12(b) shows the correlation coefficient $R(\text{RMSE1}, \text{RMSE2})$, where RMSE1 and RMSE2 denote the worst mean RMS errors obtained on chip 1 and 2, respectively and R is defined in appendix C.



(a) RMS error on two chips

(b) correlation coefficient

Figure 8.12: Comparison of the RMS errors measured on two chips: In (a) the RMS error measured on chip 1 (chip used for evolution) is compared to that one measured on a second chip (chip 2) . (b) accounts for the correlation between the two sets of fitness values obtained for all runs of each experiment.

The average performance of the circuits of all non-BB experiments (1–6) is virtually identical. In contrast, the average performance of the BB-based circuits (experiments(7–11) is seriously degraded when they are tested on a different die. The performance loss is so large, that the BB-based circuits are – on average – not significantly better on the second FPTA chip than the non-BB solutions. The magnitude of this performance degradation is the more remarkable as it is unrivaled throughout all experiments previously described in this thesis. As expected, the linear correlation between the RMS errors achieved on the different chips is low for the BB-based experiments (7–11), that is, not all of the evolved circuits suffer the same relative increase in their respective RMS errors.

Apparently, the usage of the library of four digital gates on one hand boosts the yield of satisfactorily working circuits compared to evolution based on the plain cell genotype. On the other hand, this performance gain is lost when the circuits are tested on another chip. Naturally, the question arises why the building block based solutions are so much more susceptible to the particularities of the substrate they are evolved on than the non-BB solutions. The building block library consists of logic gates that, from an analog designer's point of view, possess fairly high gains. In absence of an external clock, however, artificial evolution is more or less forced to find an analog circuit solution to the problem. Since the resulting analog circuits are supposed to be composed of the high gain stages offered by the building block library, they are likely to metastable with regard to the actual parameters of the utilized transistors.

Similar difficulties in finding a robust solution composed of logic gates have been reported by Thompson in [Tho97] (circuits susceptible to variations of temperature) and [Tho98b] (an attempt to evolve robust circuits that work on different dice of different foundries at different temperatures in different packages and locations in the lab (e.g. in- and outside a computer).). The task has never been shown to be perfectly solved. Instead, in [Tho00] Thompson had to introduce a 6 MHz clock as

an additional input in order to find a circuit working rather in a digital fashion to robustly solve the problem under a wide variety of environmental conditions.

8.4.2.4 Reconsider and Repent

A comparison with the two[4] different attempts to automatically synthesize tone discriminators reported by Thompson and Harding et al. reveals the following: First, both attempts use substrates that are completely different from the substrate used in the proposed case study. Moreover they also differ in the considered frequency range as well as the formulation of the fitness criterion. Hence, a quantitative comparison is virtually impossible.

Second, the tone discrimination device reported by Harding in [Har04a] exhibits a similar quality in that it clearly distinguishes the different frequencies but only produces an output voltage that is neither free of feedthrough nor spans the full power supply range (0.6 Vout of 10 V). However, this solution is achieved after only 6000 circuit evaluations. Two possible explanations apply: On one hand, the LCD used in conjunction with the five analog input voltages may be extremely well suited to distinguish frequencies in the considered range. On the other hand, the Harding's fitness function rewarding every sample that is correctly responded to with a voltage below or above a threshold voltage of 0.1 V may have been a more clever choice than the one defined in (8.5). However, the approach of Harding and Miller proposed in [Har04a], [Har04b] and [Mil02] to date has two drawbacks: For one, it requires a large amount of external circuitry to provide the analog voltage, namely 5 16-bit DACs. For the other, it cannot rely on a matured technology that would allow the integration of an LCD together with the additional circuitry necessary on one single chip.

Third, Thompson used a similar number of circuit evaluations in his original experiment reported e.g. in [Tho97] or [Tho98a], but obtained a circuit that is reported to perfectly discriminate the two frequencies. Here too, a variety of differences between Thompson's evolution system and that one used here may be responsible for the different quality of the respective results: On one hand, the FPGA hosting Thompson's candidate circuits provides a larger amount of resources in at least two regards: The logic cells contain a richer set of logic functions compared to the small set of building blocks used in this case study. Moreover, in Thompson's experiment the GA controlled an array of 100 logic cells exceeding the 16 BB sites provided here. On the other hand, instead of measuring the deviation of a target function, Thompson used a fitness function that rewarded the differences between the output voltages for the different frequencies. This choice of fitness function supposedly helps to prevent the GA from being trapped in a local optimum.

In conclusion, a more sophisticated design of the fitness function can be hoped to alleviate the evolution of tone discriminators based on the plain cell genotype, which seem to be intrinsically less susceptible to variations of the particularities of the respective die they are bred on.

Noch'n Gedicht

HEINZ ERHARDT

8.5 Case Study III: Comparators

Comparators. Although not as versatile as operational amplifiers, comparators are of great importance in analog circuit design. A comparator is typically used whenever a decision must be made

[4]The author is aware of the publication of Raichman, Segev and Ben-Jacob that appeared in Physica A 326 (2003) pages 265–285. However, as the reported experiments are mainly a remake of Thompson's research, they are intentionally left out of consideration here.

based upon an analog signal level. This situation could e.g. arise in the readout electronics of high energy physics experiments, where a signal passing an analog threshold voltage can be used to trigger further readout circuitry ([Bau03]) or in high dynamic range image sensors, where the reset of individual pixels depends on the charge already accumulated by the photo current to be measured ([Bre04]). Comparators form the basis of any analog-to-digital conversion; in fact they can themselves be considered as a 1-bit analog-to-digital converter. A special type of comparators known as sense amplifiers are used to restore the digital information contained in a bit of e.g. random access memory in the read or refresh process.
According to [All02e] three basic types of comparators can be distinguished:

a) Open-loop comparators.

b) comparators with hysteresis.

c) discrete-time (regenerative) comparators.

The former type can be considered the step brother of an operational amplifier in that it lacks the necessity of frequency compensation but, on the contrary, is rather targeted at larger bandwidths and therefore small propagation delays. The latter two types encourage the application of positive feedback. In case of the regenerative comparators, this requires a reset after each comparison. The designs of high speed comparators usually mix uncompensated op amp input stages of type a) with latch type output stages found in the designs of regenerative comparators.

The experiments discussed within this section aim at the synthesis of open-loop comparators, which is fostered by the following properties of the experimental setup:

- The absence of an external clock signal makes the evolution of discrete time comparators improbable.

- The fitness criterion penalizes any hysteretic behavior

- In case of the BB based experiments, the artificial evolution of regenerative comparators is not explicitely supported in that no latch type building blocks are provided in the BB library.

Open-loop comparators were intentionally chosen as a test case for intrinsic hardware evolution experiments because of their close relation to operational amplifiers, which are frequently used to evaluate parameter optimization tools within the analog design automation community. Although the particular requirements with regard to the dynamic characteristics of op amps and comparators are somewhat different, their desired output behavior has a lot in common, e.g. static behavior, high gain, low input voltage offset and a high slew rate. In other words, from an evolution experiment's point of view, the requirements that have to be satisfied by a comparator form a subset of those imposed on an operational amplifier. Hence, the ability to automatically synthesize comparator circuits is a necessary condition for the successful synthesis of operational amplifiers. On the other hand, forgoing tests concerning linearity and ac behavior of the candidate comparator circuits significantly reduces the time needed for the test of each individual. In conjunction with the reduced problem difficulty compared to the operational amplifier task, the time necessary for an artificial evolution is drastically decreased and thorough studies of the subject matter become feasible. Consequently, the lessons learned from the comparator experiments can subsequently be applied in the design of experiments targeted at the evolution of operational amplifiers.

Related Work. To the author's knowledge, the only experiments on transistor level artificial evolution of comparators reported in the literature to date are those described by Trefzer, Langeheine et al. in [Tre04]; these experiments are carried out with the same type of evolution system proposed within this thesis. From the definition of open-loop comparators stated above, however, it can be concluded that *operational* amplifiers can be used as comparators and hence should be considered in this section. To account for the recent progress achieved in this subject matter as well as for its difficulty, Table 8.5 summarizes some of the most recent attempts to the automatic synthesis – in the sense of topology design and sizing – of *amplifiers* in general. Apart from the op amp synthesis tool

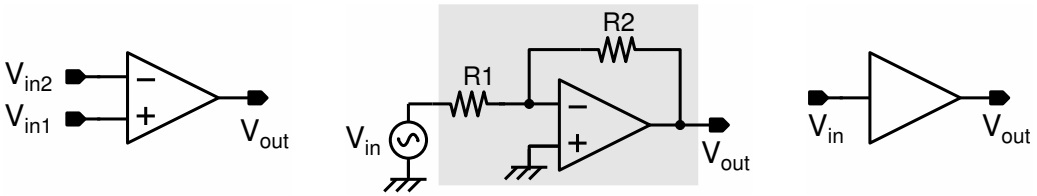
Group	type	analyses	design objectives	Comment	Used Devices	No. of Eval.
Kruiskamp [Kru96]	op amp	equation-based (HSPICE)	PM, UGB, GBP IC, OC, R_{out}	34 predefined topologies	CMOS, 2.4 µm	O(10 k)
Koza et al. [Koz99d] [Koz96b]	amp	dc $-10,5,\ldots,10$ mV (ac, transient)	A_{OL} V_{OS} linearity	test bench: 8.13(b)	R,C, BJT diode	19 M
Koza et al. [Koz99e] [Koz96a]	amp	dc $-10,5,\ldots,10$ µV (ac, transient)	A_{OL} V_{OS} linearity	test bench: 8.13(b)	R,C, BJT diode	70 M
Lohn et al. [Loh99]	amp	dc $-10,5,\ldots,10$ µV (ac, transient)	A_{OL} V_{OS} linearity	test bench: 8.13(b)	R,C, BJT	6 M
Zebulum et al. [Zeb00a]	amp	dc $-10,5,\ldots,10$ mV (dc in HW, failed)	A_{OL} V_{OS} linearity	test bench: 8.13(c)	R,BJT	8 k
Zebulum et al. [Zeb98a]	amp	dc $-10,5,\ldots,10$ mV (dc in HW, O. K.)	$\max(\lvert VS\rvert)$	test bench: 8.13(c)	R, BJT	8 k
Sripramong and Toumazou [Sri02]	amp	dc ac	A_{OL}, V_{OS}, GBP, PM, PD, VS, linearity	test bench: 8.13(c)	R,C,T in CMOS	5… 50 k
Koza et al. [Koz04b]	diff amp	dc ac transient	A_{OL},V_{OS}, PM, GBP, VS, linearity, I_{tot}, I_{bias}	5 test benches: 2 × ac, 2 × dc, 1 × transient	R,C,BJT, diode, current source	–
Shibata [Shi01]	op amp	dc: V_+,V_- $\in[-0.5,0.5]$ ac	A_{OL}, PM, I_{tot}, dc functionality	target process: 0.35 µm CMOS	transistors, current sources	30 k

Table 8.5: Related work concerning the artificial evolution of amplifiers. In the third column, the analysis modes in brackets denote additional tests applied to the evolved circuits reported. HW here indicates a test in hardware. The unusual abbreviations in the first cell of the design objectives column IC and OC refer to In- and Output compliance of the amplifier under test. In the same column, the output Voltage Swing is abbreviated to VS.

DARWIN reported by Kruiskamp [Kru96], which approximates each candidates' characteristics by a set of closed equations, all other approaches use circuit simulators as e.g. SPICE (see e.g. [Qua94]) for fitness evaluation. While the DARWIN synthesis tool first reduces the search space to an approximation of the feasible subspace by using polytopes and subsequently uses a GA to navigate through the resulting subspace, all the other referenced synthesis tools use EAs as the main optimization engine. In the latter group, prior domain or task dependent knowledge is included in the circuit representation

used for the genotype. In case of [Sri02] this is accompanied by a new technique referred to as *current flow analysis* to enhance artificial evolution and guide the search towards more human-like circuits. Strictly speaking, the approach described by Kruiskamp is not exactly an unconstrained search for op amp topologies, because it is restricted to a total of 34 feasible combinations of four input, two intermediate and 3 output stages. The abovementioned search-space reduction uses the target specifications and process parameters at hand to restrict the choice of transistor dimensions for each of the nine different stages. The small set of available topologies ensures that some design goals are inevitably met, as for instance the absence of an input offset, if one assumes the transistors to be ideal.

Although the experiments reported by Koza, Lohn and Zebulum in [Koz99d],[Koz96b],[Koz99d], [Koz96a], [Loh99] and [Zeb00a], respectively, differ in algorithm, representation and details of the target specification they all employ a similar test setup. During evolution a dc sweep small in voltage range and number of test points is used to infer the dc open loop gain A_{OL}, the input offset voltage V_{OS} and the linearity of the candidate circuits. A schematic of the test bench is depicted in Fig. 8.13(b), where the circuit under test is marked by the op amp symbol. In effect, this setup probes the candidate



(a) Op amp with differential inputs

(b) Op amp: inverting configuration

(c) Amplifier: single ended

Figure 8.13: Different amplifiers: (a) shows the symbol of an op amp. (b) displays the inverting configuration, which amplifies signals with respect to the gnd potential. This configuration is used as a test bench in some of the referenced publications on the automatic design of op amps. (c) summarizes the grey shaded part of (b) in a symbol: an amplifier with one single in- and output terminal, not an op amp anymore.

circuit's ability to invert and amplify an input signal with respect to the gnd potential (bipolar power supplies are used in the reported test setups) as illustrated in Fig. 8.13(c). Moreover, the feedback included in the test bench of Fig. 8.13(b) ensures that the input voltages stay most closely around the ground potential for any circuit exhibiting the desired amplifying behavior. Thus, the task defined by this setup is considerably easier, since the evolving circuits are freed from the symmetry constraint inherent to any amplifier with differential inputs as well as from the necessity to work at a large range of input voltages. The authors of [Koz99d],[Koz96b],[Koz99d], [Koz96a],[Loh99] do not analyze, neither during evolution nor in any verification test, the PM of the prospective amplifiers, which renders them useless for applications requiring a smaller voltage gain, which is another property of *operational* amplifiers that makes them so useful in analog design. The experiments described by Zebulum et al. in [Zeb00a] and [Zeb98a] basically suffer from the same shortcomings as those described above and mainly differ in that they use the test bench depicted in Fig. 8.13(c) and employ a different fitness criterion. The former publications address another problematic issue of insufficient testing: The best hardware realization of the evolved circuit fails to meet the objective that it was supposed to fulfill according to simulation.

The more recent publications by Sripramong, Koza and Shibata found in [Sri02], [Koz04b] and [Shi01], respectively, are more promising in that they combine refined techniques with a larger set of test objectives compared to the previously discussed experiments. Yet, they still fall short of generating operational amplifiers that would satisfy industrial standards: In [Sri02] Sripramong and

Toumazou also use the one-input test bench shown in Fig. 8.13(c), i.e. synthesize an inverting amplifier. Furthermore, their fitness criterion omits a transient analysis that would analyze the slew rate of the amplifier. Although Koza et al. in [Koz04b] include a transient analysis in their fitness evaluation and ensure that successful amplifiers do possess truly differential inputs, they restrict their evaluations to one operation point defined by input voltages that are virtually set to zero. The ac analysis is bound to use a small signal analysis; the transient analysis used by Koza et al. is restricted to small signals: the according input voltages allow for a maximum voltage across the inputs of $20\,\mu V$ translating to an output voltage of $2\,mV$ when the target A_{OL} is met. Because of this lacking large signal analysis and test of input compliance, the resulting amplifiers are denoted as *differential*, but not *operational* amplifiers. Finally, in [Koz04b] Shibata successfully evolved op amps that satisfy the required large signal dc behavior and achieve reasonably good values for UGB, PM and the total current consumption I_{tot}. Even though Shibata omits a transient analysis and disregards many typical op amp specifications, he includes the most preeminent features making a circuit an op amp, that is those that according to the author's experience a human designer is most concerned about at first: While the dc test, which resembles the test pattern proposed below (Fig. 8.21), in conjunction with the A_{OL} and UGB ensure the principal functionality of the emerging op amp, the PM guarantees its stability under all feasible negative feedback conditions.

In summary, apart from the DARWIN synthesis tool the operational amplifiers found by Shibata are those the author trusts most to be useful as real amplifiers in general and henceforth as comparators in particular. However, none of the referenced publications considers things like load dependency, susceptibility to variations in the design or process parameters or yield rate. On the other hand, none of these publications reports the usage of more advanced techniques for multi-objective optimization.

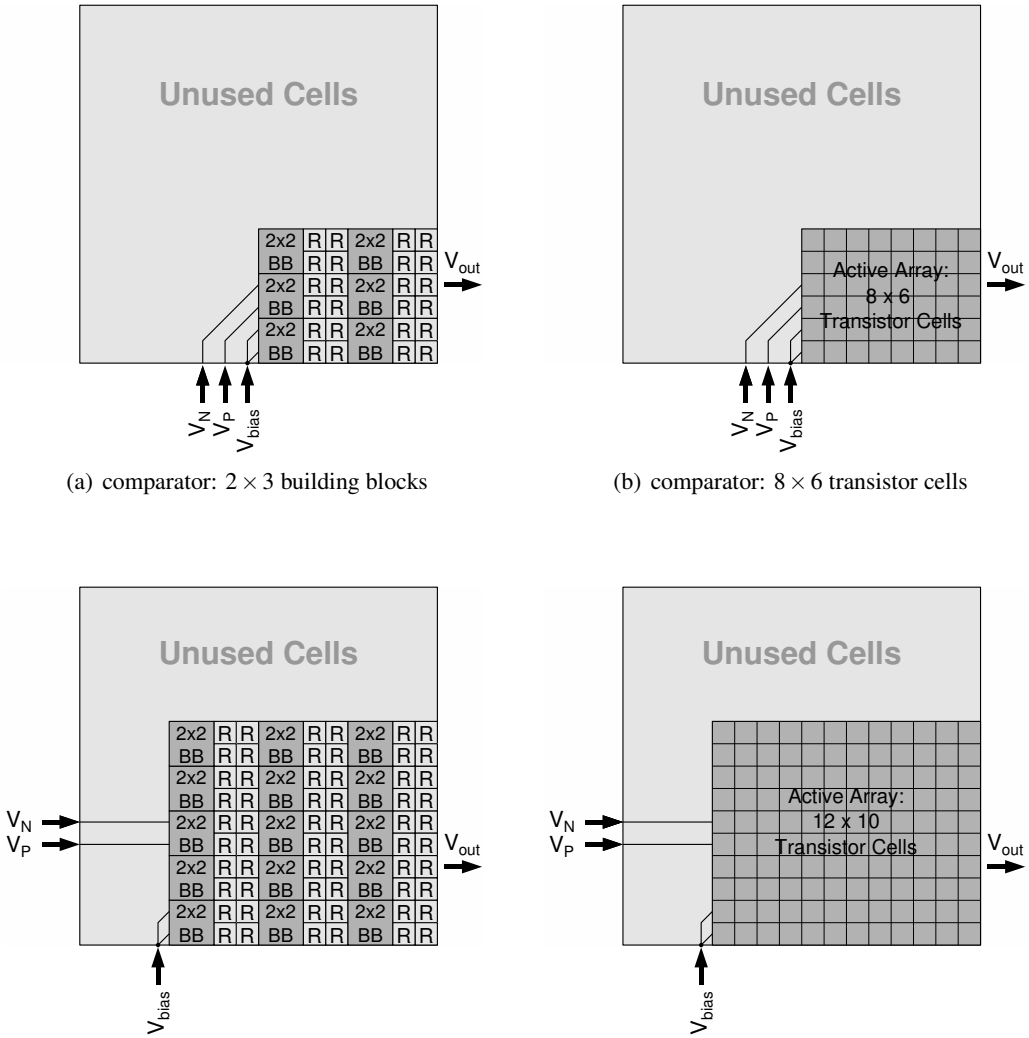
The best evolved circuit reported in [Shi01] achieves a dc gain of 72 dB, possesses a UGB of $15\,MHz$, and a PM of 49° it was found after only $30k$ circuit evaluations. Key elements to this success are supposedly:

- the current-path based representation already sketched in section 8.1.2
- the inclusion of current sources into the meta element library.
- the small input compliance range described by $V_{+,-} \in [-0.5, 0.5]$
- the further restriction of the design space to 2 to 4 current paths featuring 2 or 3 meta elements.
- fixed power supply connections at the top and bottom ends of each current path for vdd and vss respectively
- the much simpler routing as compared to cell based approaches on reconfigurable devices: just pick the correct circuit node in the net list at random.

8.5.1 Geometrical Setup

The geometrical setup for this third case study is depicted in Fig. 8.14. Since two different array sizes are used in different experiments and the results shall be compared to experiments achieved without using BBs a total of four different setups is required. Analog to the experiments of case studies I and II, the signal flow is supposed to be directed from left to right. Unlike the above scenario of case studies I and II, a current is likely to pass the BB borders. In accordance with the conventions used for the illustration of circuit schematics, this current is supposed to flow through the BBs from top to bottom.

In order to accommodate subcircuits of typically 2 transistors, the size of the BBs was set to 2×2 transistor cells. The building blocks are stacked directly on top of each other in order to minimize any



(a) comparator: 2×3 building blocks

(b) comparator: 8×6 transistor cells

(c) comparator: 3×5 building blocks

(d) comparator: 12×10 transistor cells

Figure 8.14: Geometrical setup for case study III: (a) and (c) are used to host the building block experiments. (b) and (d) illustrate the setup for experiments utilizing the plain genotype. R denotes the routing cells.

waste of area and possible operation speed. Thus, the necessary routing resources must be provided by the BBs themselves. However, to allow for more complex routing scenarios, the building block library discussed below contains several blocks that are exclusively devised to routing tasks. Adjacent columns of BBs are glued together by two columns of routing cells to allow for a flexible routing between these columns. The reason for the choice of two columns is twofold: First, in symmetric designs it is likely that two signals have to be passed in parallel from one BB column to the next. Second, the underlying checkerboard pattern of the PTA limits the translation invariance to a step size of two cells. Hence, either all BBs must be placed on a grid spaced two transistor cells apart or must be adequately transformed. Although the latter procedure should usually be feasible it takes up precious computing time and adds another potential source of errors.

8.5.2 Building Block Library

8.5.2.1 Contents of the Building Block Library

The entire building block library consists of a selection of subcircuits frequently encountered in analog CMOS designs that are believed to be useful in the design of a comparator or other analog transistor level circuits. In order to provide sufficient routing resources, in many cases several implementations are included for the same subcircuit. The resulting library of 124 building blocks is symmetric with respect to the MOS flavors PMOS and NMOS. Due to its size its illustration is distributed over four figures: Current mirrors, level shifters and the like are shown in Fig. 8.15, output stages in Fig. 8.17 and a variety of blocks featuring 22 transistor pairs, two bias generators and ten routing blocks in Fig. 8.17. The largest variety of implementations is created for blocks containing a single transistor only. The implementations for the single NMOS transistor are depicted in Fig. 8.18; those for the single PMOS transistors are equivalent to their NMOS counterparts and therfore not shown.

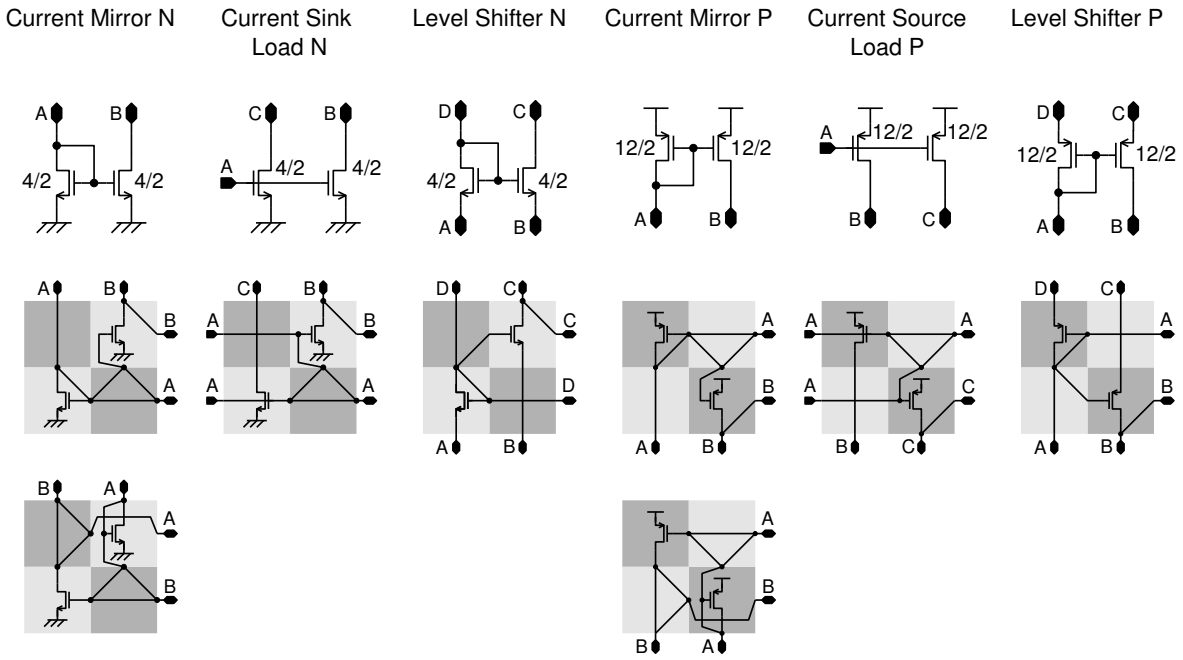


Figure 8.15: Building Block Library: Current mirrors and the like.

8.5.2.2 Structure of the BBL Figures

Fig. 8.15 to 8.18 on pages 270–273 are organized as follows: The first two rows contain name and schematic of the respective subcircuit. The rows below depict the implementation as blocks of 2×2 transistor cells, where PMOS cells are shaded in darker gray than the NMOS ones. The terminal names in the schematic match those in their transistor cell implementations. The only exception are the routing cells that are presented in the two rightmost columns of Fig. 8.17, where they are located below all entries belonging to the two bias generators. In Fig. 8.15 and 8.18 the abbreviation power supply (PS) is used to indicate that these subcircuits possess a connection to either one of the power supplies – to vdd for PMOS and to gnd for NMOS transistors. The numbers next to the transistors in the schematic views denote the initial gate width and length used on insertion of the respective building block.

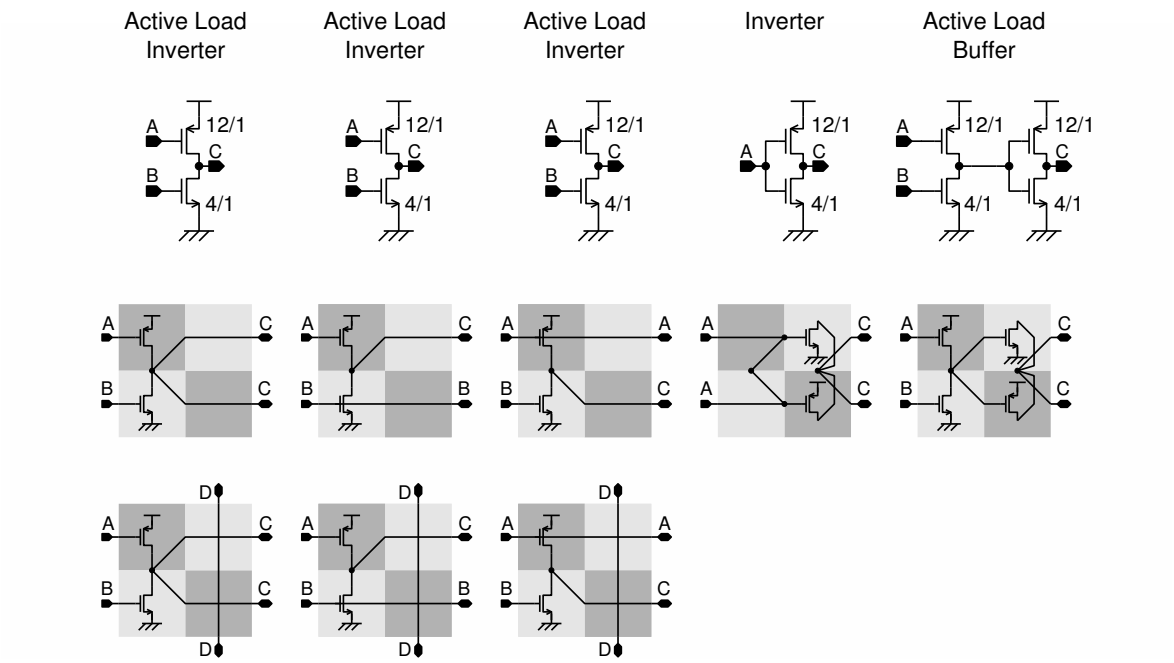


Figure 8.16: Building Block Library: Output stages.

8.5.2.3 Critique of Pure Building Block Library

In [Gra01] Graeb et al. identify seven useful 2-transistor combinations, which are referred to as current mirror, level shifter, current mirror load, differential pair, flip flop and two voltage references. Of these seven subcircuits, the current mirror and the level shifter are directly implemented in the building block library. Since the prospective comparators are supposed to work continously in time without hysteresis, the meta-stable flip-flop is discarded. The two voltage references and the current mirror load are combinations of two transistors in series. Except for the output stages and bias generators, the building blocks included in the proposed library do not contain stacked transistors, but are to be stacked to series connections by the GA. Nevertheless, the inclusion of these three stacked subcircuits may be a beneficial option for future experiments, because it makes it easier to provide better current mirrors or the necessary biasing for cascode elements in the circuit. It should be noted, however that principally it is possible to realize all four forgone transistor pairs by means of the existing building blocks and some additional routing.

Finally, no direct implementation of the differential pair is provided in the BB library to limit the number of necessary BBs. Since differential pairs are used in virtually any input stage of an op amp, this deliberate non-consideration seems rather perfidious. On the other hand, in the vast majority of cases, differential pairs are used in series with a current sink/source to form a differential stage; this in turn can be realized by stacking a transistor pair and one of the single transistor building blocks, which shorten the two source/drain terminals of the transistor pair, on top of each other.

The building block library at hand contains two additional types of 2-transistor subcircuits not considered by Graeb et al., namely the current sink/source load, which e.g. can be used as a passive part of a current mirror bank (which actually is considered by Graeb et al. as a higher level BB) and a selection of transistor pairs. The current sink/source load are not implemented in a floating version, a flaw that must be compensated by using a transistor pair with shortened inputs. Since the transistor pairs are completely uncorrelated, they are no real subcircuits themselves, which is why they are disregarded by Graeb et al.. Within the building block library, however, they are believed to be

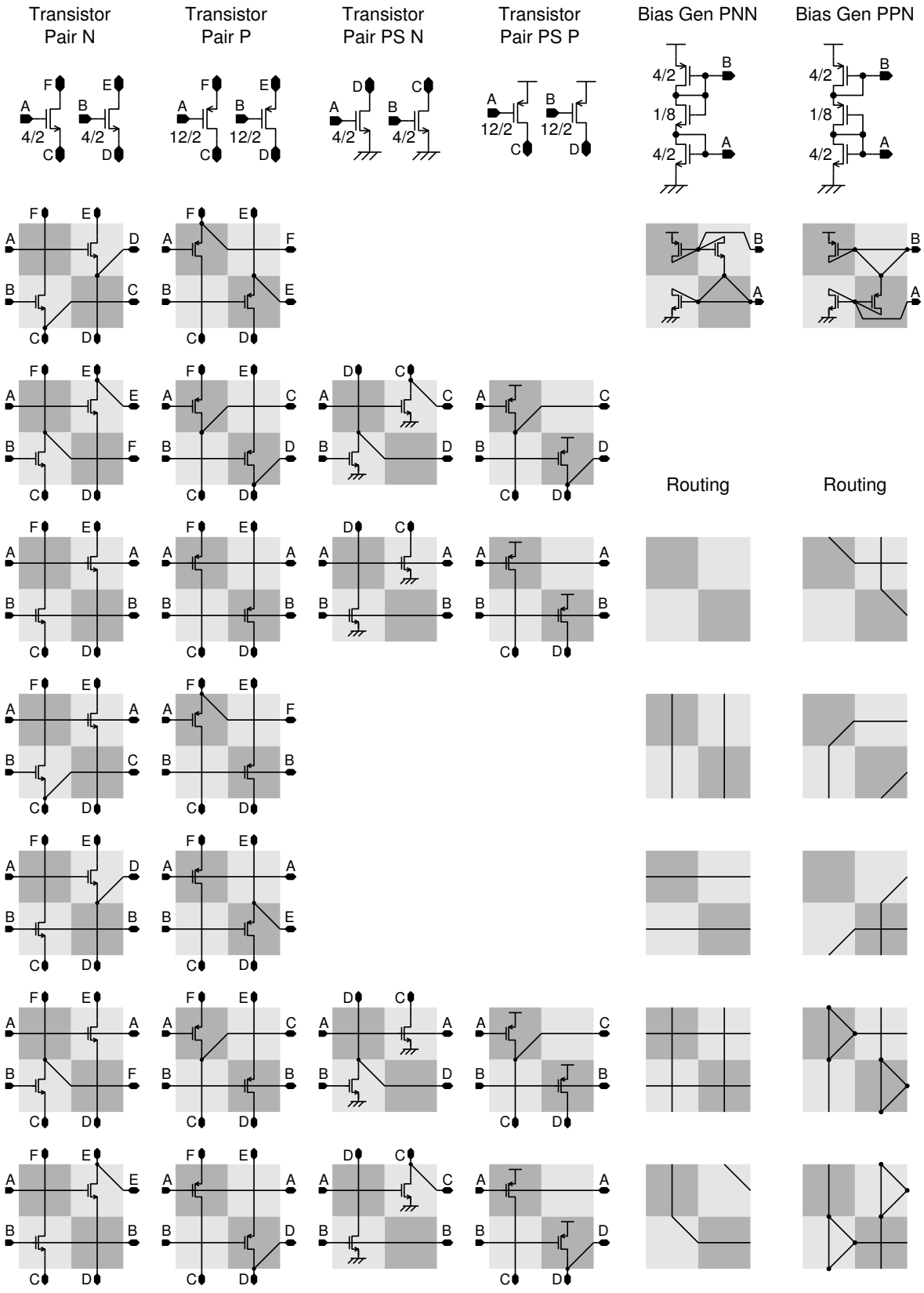


Figure 8.17: Building Block Library: Transistor pairs, bias generators and routing blocks.

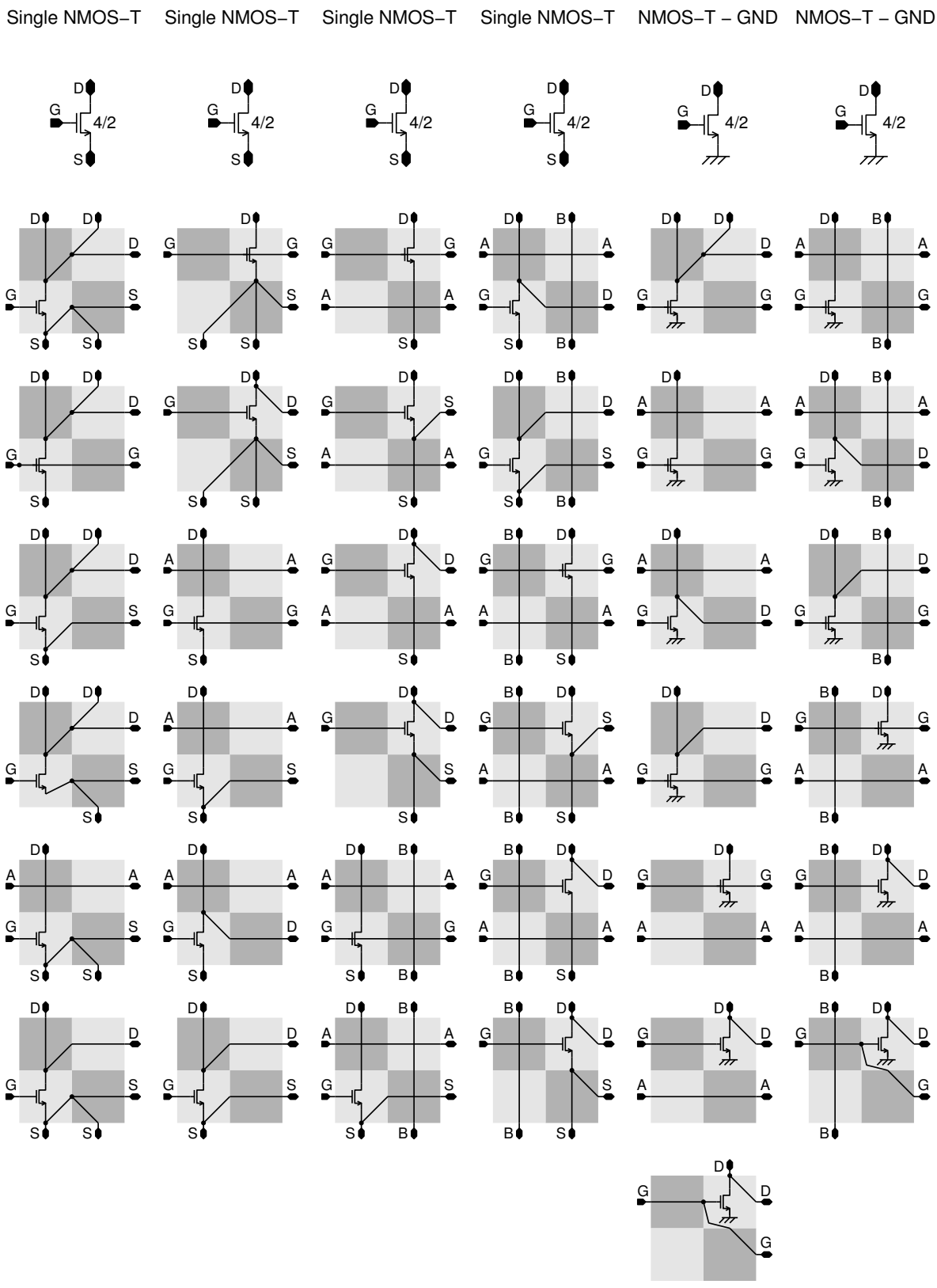


Figure 8.18: Building Block Library: Single NMOS transistors.

beneficial, because they *suggest* the formation of useful subcircuits, as e.g. the abovementioned composition of floating current sinks/loads or differential pairs. Moreover, even without being connected to each other, they may be an important part of a differential twofold current-path.

In accordance with Graeb et al., the most simple building block is represented by a single transistor. Although the library at hand contains many implementations thereof, the size of the feasible design space is considerably reduced, for instance because the current flow is restricted to the north-south direction and the signal flow strongly biased into the east west direction. In contrast to Graeb et al., no larger – second level – subcircuits, as for example 4 transistor or cascaded current mirrors or banks of level shifters or current mirrors, are employed. These substructures are not realizable with 2×2 transistor cells and their inclusion therefore would break the symmetry, homogeneity and simplicity of the chosen methodology.

So far, only subcircuits restricted to the same type of MOS transistors have been considered. In addition to these, the proposed library contains subcircuits that mix P- and NMOS transistors. For one, these are the bias generators illustrated in Fig. 8.17, that can be used to create static voltages that can e.g. bias current sink/source loads or cascodes. For the other, Fig. 8.16 depicts three different output stages, namely active load circuits, inverters and a combination thereof called active load buffer. The latter one can be transformed into a real buffer by shortening its inputs in the routing cell columns left to it. The active loads are often used as output stages in simple op amps like the Miller operational amplifier.

8.5.2.4 Critique of Practical Building Block Library

The selection of layouts for each subcircuit tries to capture all useful routing options that are feasible. Thereby the necessities arising from the geometrical structure of the prospective circuits (cf. Fig. 8.14) shall be taken into account:

1. The current is flowing vertically within each building block layout. For all building blocks containing connections to either of the power supply voltages, it is necessarily directed from top to bottom. A signal flow directed from left to right is sustained by connecting the input terminals – which happen to be gate terminals only – of the respective subcircuit to the western side of the block. In order to insert a current from a prior stage located to the west of the block at hand, it must either be inserted at the top of the building block, or an appropriate routing block must be used.

2. The only means to connect the evolving circuits to the power supply voltages is through the building blocks themselves. Thus, whenever applicable, the proposed subcircuits are realized in two versions, a floating one, and another one connected to either of the power supply voltages (vdd in case of P- and gnd in case of NMOS transistors). The location of the power supply connections complies with the desired top-to-bottom current flow as was explained above.

3. The principle of directly stacking the BBs on top of each other (as shown in Fig. 8.14) requires the implementation of all necessary *vertical* routing within the building blocks. For instance, the single NMOS-transistor layouts must contain a version that merges the two northern terminals of the block before feeding it in the actual transistor in order to allow for the composition of differential NMOS stages from transistor pair BBs and single NMOS-transistors. Moreover, it may prove successful to allow vertical routing to pass the actual subcircuit implementation where applicable (i.e. in active loads and single transistors). Yet, this may in other situations create rather detrimental effects. Hence, wherever applicable, two layout versions are provided, one with and one without the extra vertical connection.

4. Due to the direct stacking of BBs all signals must be routed *horizontally* through the building blocks. Since building blocks of different columns are connected via two columns of routing cells, the terminals to the east- and western side of the building blocks are usually fully connected. In the layouts of the active loads as well as the single transistors additional east-west connections are provided whenever possible.

5. Even within the constraints imposed by the routing capabilities of the transistor cells, the described building block library is far from being complete:

   (a) For some circuits it could be beneficial also to provide versions where the position of the outputs on the eastern side is swapped, because it is impossible to cross two signal running in horizontal direction within two rows. Since this is not possible for all circuits, this possibility was discarded for symmetry reasons, which is not to severe a limitation, as signals can very well be crossed in 90° turns.

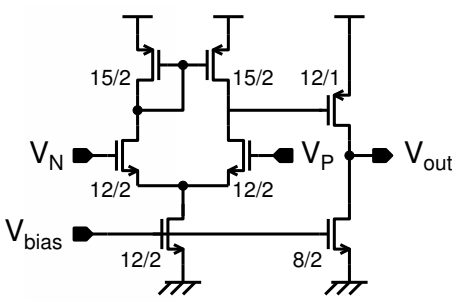
   (b) In some cases, certain choices of output signal combinations have been discarded because they are not in compliance with the experience and conventions present in traditional human designs. For instance, it seems to make little sense to propagate the source terminal of one and the drain terminal of the other transistor of a transistor pair to the BB's eastern output side.

   (c) Some combination may just have been believed to be of little value, easily synthesized otherwise or have been forgotten.

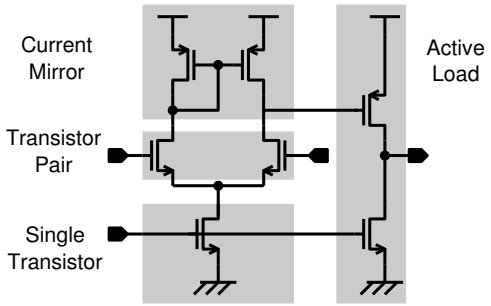
   Nevertheless, even though probably suboptimal, the library is believed to be well suited for the task at hand and a reasonable choice for other transistor level synthesis experiments.

On insertion into a genotype under evolution, the transistor gates of the building blocks possess the initial values denoted in their schematic views in Fig. 8.15 to 8.18 on pages 270–273. Except for the output stages and the bias generators, the transistor lengths of all BBs are set to $2\,\mu m$, which is the median of the five available lengths and a reasonable value to start with, since it presents a good compromise between high transconductance and a low channel length modulation parameter λ. In case of the output stages the tradeoff is shifted in favor of a higher transconductance parameter to a channel length of $1\,\mu m$. The dimensioning of the bias generators simply aims at providing useful bias voltages. Again with the exception of the bias generators, the gate widths of all PMOS transistors are initially set to 12 and those of the NMOS transistors to $4\,\mu m$. The factor three by which the PMOS transistors are wider than their NMOS counterparts is used to compensate for the difference in their respective transconductance parameters (see [Aus97c]). Satisfying this constraint, the transistor initial widths are rather on the larger end of the scale to provide sufficient transconductance to the prospective subcircuits. Please note that according to these initial transistor dimension all current mirrors process a current gain of (approximately) one.

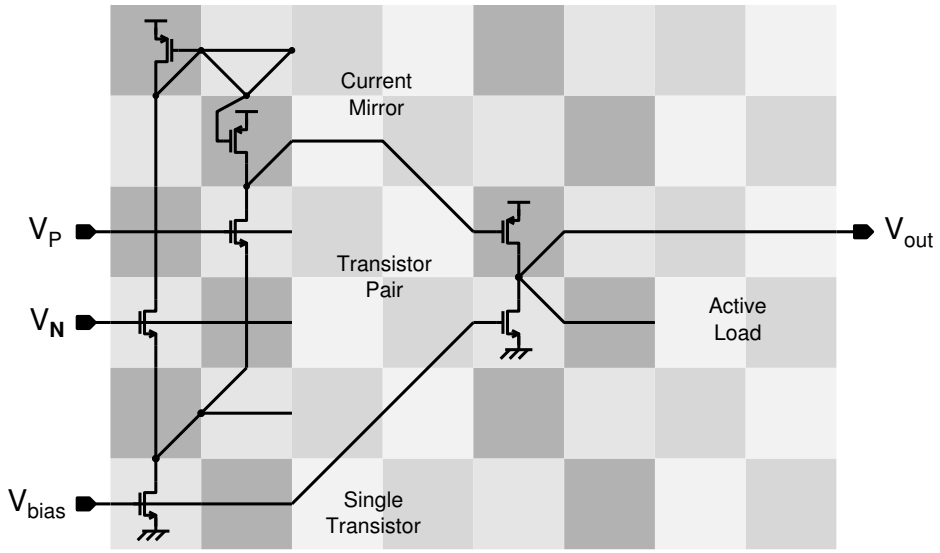
A Hand-Designed Comparator as an Illustrative Example. In order to illuminate the way the proposed building block methodology can be used to compose transistor level circuits, a simple 2-stage comparator composed of building blocks provided by the described BB library is fitted into the geometrical arrangement depicted in Fig. 8.14(a). While the circuit schematic including transistor dimensions and terminals is shown in Fig. 8.19(a), the building blocks the comparator can be composed of are identified in Fig. 8.19(b). Fig. 8.19(c) illustrates the implementation in terms of building blocks and additional routing that is compatible with the BB-based genotype. Please note that the transistor dimensions annotated in Fig. 8.14(a) differ slightly from the initial values defined for the BBs above.



(a) schematic view



(b) schematic view divided into BBs



(c) comparator composed of BBs

Figure 8.19: Hand-made 2-stage open loop comparator: (a) shows the schematic view with terminal definitions and the used transistor dimensions. (b) identifies the different BBs by shading them in gray. (c) shows how the comparator is assembled using BBs from the proposed library.

Comparison to the Current-Path based synthesis proposed by Shibata. As stated above, the proposed building block methodology was inspired by Shibata's work on current-path based synthesis described in [Shi01]. Analog to the current-path based synthesis, the chosen combination of BBs and the geometrical structure of the genotype suggests a current flow from top to bottom through columns of single or pairs of transistors. Nonetheless, a closer look also reveals a bunch of subtle features included in the current-path based synthesis that may result in a considerable performance enhancement:

1. The proposed meta-elements can take on the functionality of ideal current sources. Though a straight-forward task for an engineer, the effort of realizing such a current source can be immense for an EA, especially with a cell based genotype representation (see below).

2. The connections to the power supply are bound to occur at the correct ends of the current path. To achieve this using the BB concept, the algorithm must try and err until an appropriate BB is chosen from the library.

3. A twofold current path is automatically merged into a single one, where it is connected to a 1-device element. In comparison with spending precious time waiting for the right BB to come, this renders the composition of a differential stage child's play.

4. The gate(s) of one meta-element are connected to exactly one node chosen from the entire circuit. In case of a two transistor meta-element these nodes are selected from one meta-element; in case the target meta-element possesses two transistors as well, the analog nodes in the different current paths are chosen, in case of a one-device meta-element, the gates are shortcircuited. Thus, the connectivity of the circuit is easily changed and bound to a restricted subset of reasonable routes. In contrast, the emergence of the correct signal routing is much more tedious for any cell-based genotype in that it usually involves a large number of routing switch manipulations. Moreover, in areas of unconstrained routing a lot of unwanted effects as e.g. the shortcircuiting of signals can happen. In case of the BB concept, a change in routing may also require the exchange of one or rearrangement of more than one BB, where especially the latter situation is very unlikely to happen. These difficulties are typical for cell based genotype representations in which the logic functionality and geometric layout of the circuit are inextricably mixed.

5. In its current implementation, the crossover operator is restricted to the exchange of BBs. In the particular setup proposed in this third case study, a crossover that preserves the current-flow oriented column structure of the genotype in a similar way as the crossover used by Shibata may prove successful.

The above arguments indicate that the current-path based synthesis proposed by Shibata may be better suited fo finding comparator-like circuits than the proposed building block concept. A direct adaptation of this method to the proposed evolution system however, necessitates the separation of the logical circuit from its realization on the FPTA chip. This in turn requires an efficient way of mapping the logical circuit onto the chip, which itself is a nontrivial task that may prove to be computationally expensive. Yet, the philosophy of intrinsic hardware evolution adopted for the scope of this thesis suggests to use a rather simple genotype that adds little computation time to the processing of an individual such that the relatively fast fitness evaluation achieved in hardware can be fully exploited to run the experiments for a larger number of generations. On the other hand, the proposed building block concept is more flexible and versatile in that the geometrical structure of the genotype as well as the building block library can be defined by the user.

Concluding Remarks. Aside from the problem of which subcircuits to implement in which layout versions discussed above, another problem arises in automatically composing circuits by inserting BBs of the respective library: How to distribute the probabilities for inserting a particular BB. In the current implementation, for each BB exactly one instance is a member of the BB library and the probability of being chosen for insertion is uniformly distributed among all BBs. This may very well be a suboptimal choice, because subcircuits for which more layout alternatives are implemented, as e.g. single transistors, are chosen more frequently than others. To improve the situation one could either adapt the number of copies of each building block present in the library, or allow a partial reconfiguration of the routing in the building blocks themselves.

In [Gra01] Graeb et al. use the concept of BBs to facilitate the sizing of the circuit at hand by means of *sizing rules* applicable to the different BBs. Many of their *sizing rules* require a precise knowledge of the operation region of the respective transistor. Though at least all of the node voltages can be made available in the FPTA chip (see section 3.6), this would gravely increase the testing time for each candidate circuit. Thus, the application of such – more elaborative – *sizing rules* would naturally

lend itself to *ex-* or *mixtrinsic* hardware evolution. Nonetheless, in the realm of *intrinsic* hardware evolution the application of simple *sizing rules* relating the feasible subset of transistor dimensions to the respective building block may be beneficial. In its current state, the used software restricts this method to fixing the transistor's gate length and/or width to its initial value depending on the BB's location on the genotype. A desirable extension thereof would allow to define independent sizing rules for each BB separately, such that width and/or length as well as the aspect ration W/L could be fixed for all transistors of one BB.

The current implementation of the BB concept is hampered by the fact that each BB is dimensioned according to the initial values specified in the BB library. Since the BB insertion operation is applied in the same step as the other mutation operators, the transistor dimensions have little time to evolve before they become meaningless due to a new BB insertion. Thus, it may pay off to split the topology altering variations and the parameter optimization. Viable techniques for splitting the population into subpopulations have been sketched in 2.5.1. In this context, the hierarchical fair competition scheme proposed by Hu et al. [Hu02a], [Hu03a], [Hu03b] seems very promising, as it may lend itself to distinguishing between levels which are limited to breeding new topologies and others that focus on parameter optimization.

8.5.3 Experimental Setup

Similar to the previous chapters and case studies, the problem is specified in terms of a target behavior, a fitness function and a test pattern that defines where the behavior of the candidate circuit is measured. The synthesis algorithm used for this case study is refined in three regards pointed out below and otherwise specified in terms of the used GA parameters.

8.5.3.1 Fitness Function

A comparator possesses two analog inputs and one output whose voltage is usually interpreted by further digital circuitry. Hence, the desired functionality of a comparator is usually described by

$$\begin{array}{lll} V_{\text{tar}} \geq V_{\text{OH}} & \text{if} & V_{\text{P}} > V_{\text{N}} \\ V_{\text{tar}} \leq V_{\text{OL}} & \text{if} & V_{\text{P}} < V_{\text{N}} \end{array} \quad , \tag{8.7}$$

where V_{OH} and V_{OL} denote the higher and lower threshold for the output voltage, respectively. The thresholds must be chosen such, that the subsequent digital circuitry is bound to interpret the output correctly under all possible conditions. According to [All02e] the thresholds V_{OH} and V_{OL} are usually taken as 70% and 30% of the power supply voltage, which results in $V_{\text{OH}} = 3.5\,\text{V}$ and $V_{\text{OL}} = 1.5\,\text{V}$ for the FPTA's fabrication technology. For the fitness criterion used during evolution however, the thresholds are set to 4.5 V and 1.5 V, respectively. For one, solutions satisfying this criterion do exist, as will be shown in section 8.5.5; for the other, defining V_{OH} and V_{OL} closer to the power supply rails increases robustness and usefulness of the successfully synthesized comparators. Unlike all other experiments of this thesis, the fitness criterion used for the evolution process is not based on the sum of squared errors (SSE), but employs a penalty scheme described by the following equation:

$$\text{penalty}_i = \begin{cases} 0 & \text{if} & & & |V_{\text{tar}} - V_{\text{out}}| & < & 0.5\,\text{V} \\ 1 & \text{if} & 0.5 & \leq & |V_{\text{tar}} - V_{\text{out}}| & < & 1\,\text{V} \\ 2 & \text{if} & 1 & \leq & |V_{\text{tar}} - V_{\text{out}}| & < & 2\,\text{V} \\ 3 & \text{if} & 2 & \leq & |V_{\text{tar}} - V_{\text{out}}| & < & 3\,\text{V} \\ 4 & \text{if} & 3 & \leq & |V_{\text{tar}} - V_{\text{out}}| & < & 4\,\text{V} \\ 5 & \text{if} & 4 & \leq & |V_{\text{tar}} - V_{\text{out}}| & < & 4.9\,\text{V} \\ 6 & \text{if} & & & |V_{\text{tar}} - V_{\text{out}}| & \geq & 4.9\,\text{V} \end{cases} \quad \text{for} \quad i = 1 \ldots 462 \ , \tag{8.8}$$

which is visualized in Fig. 8.20. The fitness is then defined by the sum of penalties for all 462 input voltage pairs:

$$\text{Fitness} = \sum_{i=1}^{462} \text{penalty}_i \ . \tag{8.9}$$

A similar penalty scheme was shown to outperform the SSE fitness function, when applied to the comparator problem using a similar setup and the same type of evolution system by Trefzer in [Tre03]. A possible explanation may be given in terms of the fitness landscape. Compared with the SSE based fitness evaluation the penalty scheme has a looser grip on the evolving genotypes, in that a larger variety of circuits happens to attain the same fitness value. Hence, a larger variety of changes in the genotype is reachable for a given circuit without becoming extinct due to a decrease in fitness. With this larger variety of equally rated starting points in reach, an improving genetic alteration is more likely to occur. The importance of such *neutral networks* is e.g. discussed in [Har97] and [Shi00]. Since, on the other hand, the artificial evolution process must be guided by the fitness criterion, the granularity with which a performance decrease is measured is bound to be a tradeoff whose optimum is likely to vary depending on the task at hand. Yet another aspect of the proposed fitness scheme is its ability to suppress some of the noise inherent to the – randomized – measurement, which may also prove beneficial for a hardware-in-the-loop system.

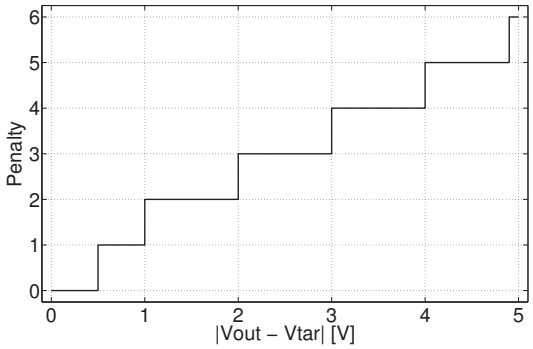


Figure 8.20: Penalty values used for the fitness function.

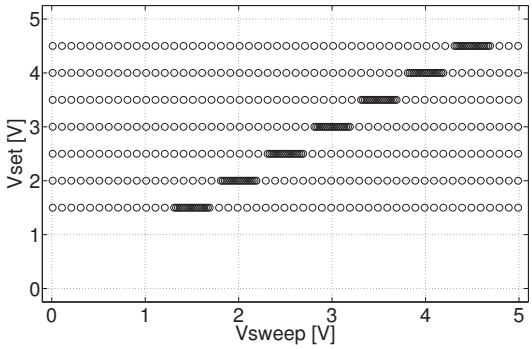


Figure 8.21: Test Pattern used during evolution.

8.5.3.2 Test Pattern

The candidate comparator circuits are tested in quasi-dc mode, similar to the test of the XOR/XNOR circuits of section 8.3. Here, the seven V_{set} voltages are chosen to be $1.5, 2.0, \ldots 4.5\,\text{V}$. For each V_{set} the other input V_{sweep} is swept through 66 voltages. In a small interval of 380 mV around the respective V_{set} the test points are spaced 20 mV apart. The rest of the power supply range is covered in a coarser resolution of 100 mV. Thereby, the region around the target trip points is sampled with a relatively high resolution, whereas the remaining test points ensure the correct large signal behavior while maintaining a moderate test length. The test pattern is visualized in Fig. 8.21. Analog to the test patterns used in chapter 6 and section 8.3, the order in which the input voltage pairs are applied to the circuit's input are chosen randomly from ten random orders for each test to sustain the quasi-dc character of the test mode (the importance of which was discussed in chapter 5).

In order to account for the totally antisymmetric nature (with regard to the polarity of the input voltage difference) of ideal comparators, the test pattern described above is applied for two different mappings of V_{set}, V_{sweep} to V_{P}, V_{N}. For the first test mode, the positive input of the prospective comparators V_{P} is stimulated by V_{sweep}; for the second test mode, the role of V_{P} is switched to the set voltage V_{set}.

In addition to the two input voltages V_P and V_N a third voltage V_{bias} is offered to the candidate circuits as shown in Fig. 8.14. This bias voltage is kept constant at 1.6 V and may be used by the comparator circuits e.g. for biasing one or more NMOS current sinks, which would be required for the 2-stage comparator design shown in Fig. 8.19. For each input voltage pair, the inputs are set in the order V_{bias}, V_P and V_N. The output is sampled 1.075 µs after the second and 0.825 µs after the first input voltage is set. The settling time is conservatively estimated to be the former of those two times. Hence, a successful comparator can be said to settle in less than 1.075 µs. Since this time comprises the comparator's propagation delay as well as the time to slew its output, the SR must[5] amount to at least $3.72\,\frac{\mathrm{V}}{\mu\mathrm{s}}$. With regard to the intrinsic speed of the FPTA that is decreased by the combination of parasitic capacitances and resistors, this represents a remarkably ambitious timing constraint. In summary, the times necessary for the test of one input voltage pair add up to 1.35 µs resulting in a sampling rate of 741 kHz.

8.5.3.3 Verification Tests

Analog to the experiments of the previous chapters and case studies, all of the evolved circuits are tested for 100 times outside of the evolution loop. Thereby, the comparators are also tested in two additional test modes featuring a finer but evenly spaced resolution of 5 mV. Since the same seven set voltages V_{set} voltages are used, each of this test modes samples a total of $7 \cdot 1001 = 7007$ input voltage pairs. The reason for these additional test modes is twofold: First, their high resolution allows to measure offset and gain of the evolved comparators with high accuracy. Second, the fact that they are evenly spaced can be exploited to derive the root mean square deviation from the target behavior per data point in mV for each or the two test modes:

$$\mathrm{RMSE}_{\mathrm{tm}}[\mathrm{mV}] = \sqrt{\frac{\sum_{i=1}^{7007}(V_{\mathrm{tar}}(i) - V_{\mathrm{out}}(i))^2}{7007}} \cdot 1000 \ , \tag{8.10}$$

where the target output voltages V_{OL} and V_{OH} where chosen as 0 V and 5 V, respectively. The final root mean square error is then calculated as the mean of the RMSE for both test modes:

$$\mathrm{RMSE}[\mathrm{mV}] = \frac{\mathrm{RMSE}_{\mathrm{tm1}} + \mathrm{RMSE}_{\mathrm{tm2}}}{2} \ . \tag{8.11}$$

Although this measure is unfair in that the target output voltages differ from the output voltage thresholds V_{OL} and V_{OH} used in the original fitness criterion, it is the natural way of providing a quality measure for the evolved circuits that allows for a comparison with other results.

8.5.3.4 Genetic Algorithm

In principle, the same GA as for case studies I and II is employed for the automated synthesis of comparators. However, besides being run with slightly different parameters as summarized in Table 8.6, the algorithm is altered in three regards:

1. The linear rank based selection scheme proposed in section 7.3.1.3 on page 211 is employed instead of the truncation selection favored in the preceeding case studies.

[5] Actually, this condition only occurs if two subsequent input voltage pairs force the output to change its polarity. For each test pattern this is guaranteed to happen at least once, but due to the randomization of the input order it is likely to occur much more frequently.

2. The four mutation rates listed in Table 8.6 are controlled in a deterministic adaptive manner (following the terminology of [Eib03b]), namely by being multiplied by a fitness dependent factor called multiplication rate multiplier (MRM) defined as:

$$\mathrm{MRM} = \begin{cases} 3 \cdot (\log(\mathrm{Fitness}) - \log(70)) + 1 & \text{if} \quad \mathrm{Fitness} > 70 \\ 1 & \text{if} \quad \mathrm{Fitness} \leq 70 \end{cases} . \tag{8.12}$$

The fitness dependency of MRM is illustrated in Fig. 8.22. As long as the fitness attained by the best individual of the last generation exceeds a threshold – set to 70 here – the mutation rate is divided by three for a fitness drop of one decade. This mutation rate adaptation is motivated by the supposed course of a successful evolution run: During the first phase, the EA is to *explore* the search space at hand; in the second phase, the algorithm is to *exploit* the raw solution present in the population in the sense of fine-tuning. Hence, for large fitness values indicating the *exploration* phase, a larger number of mutations per individual is desired to sample a larger portion of the design space, while for smaller fitness values a smaller number of alterations per individual is more likely to preserve the successful part of the genome. Due to the discrete nature of the genotype, however, too small a mutation rate prevents mutation from happening at all; hence the threshold of 70 that avoids a further decrease of the mutation rate.

3. A fraction of the population referred to as *mutants* receives a special genetic treatment, which was introduced by Trefzer in [Tre03]. In the current case study, a fraction of 20% of the population is chosen as a *mutants*, which are mutated with a mutation rate that is five times higher than the nominal one (the rate itself, excluding the adaptively defined MRM). Moreover, *mutants* undergo the crossover operation twice: Once they may be recombined with an individual taken from the whole population and once with an individual chosen from the *mutants* only. As can be seen from Fig. 8.22, the mutation rates for *mutants* and the rest of the population are almost the same at the beginning of an evolution run. Later on, however, the small fraction of *mutants* keeps exploring the design space, while the rest of the population is designated to be fine-tuned. Thus, the *mutants* can help to overcome local optima.

GA Parameter	Comp: BB	Comp: Cell
population size	20	20
reprod. fraction	0.1	0.1
crossover rate	0.4	0.4
mut. rate routing	MRM ·1%	MRM ·1%
mut. rate W/L	MRM ·1%	MRM ·1%
mut. rate term. con.	–	MRM ·1%
mut. rate BB	MRM ·1%	–
mutants fraction	0.2	0.2
MRM for mutants	5	5
no. of used blocks	6/15	–
no. of used cells	24/60	48/120
crossover block size	2/1	6
no. of generations	10,000	10,000

Table 8.6: Genetic algorithm parameters used for case studies I and II.

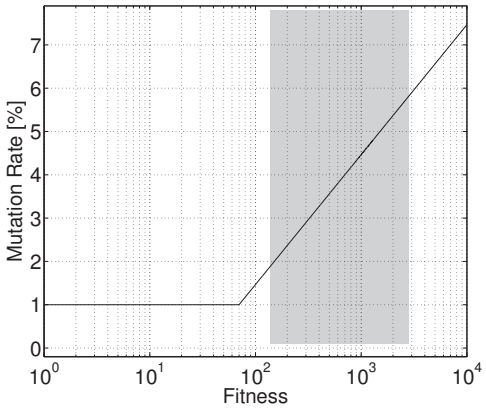


Figure 8.22: Fitness dependency of the mutation rates. The fitness regime encountered in the described experiments is shaded in gray.

The most important difference with regard to the GA parameters is the small population size of only 20. This number is inspired by an empirical analysis of the performance dependence on mutation

rate and population size done by Hohmann et al. published in [Hoh02b]. Hohmann concluded, that in training a hardware neural network to solve the 4-bit parity problem by means of an EA, a population size between 15 and 20 would be ideal. Though different in task and hardware, the system used by Hohmann et al. and that one proposed here are both believed to mainly mainly on mutation as the dominant variation operator rather than on recombination. Hence, a number of 20 individuals seemed a good choice.

8.5.4 Overview of the Experiments

This third case of BB-based evolution of comparators is studied by means of 8 different experiments whose distinguishing features are summarized in Table 8.7. These are the type of and restrictions im-

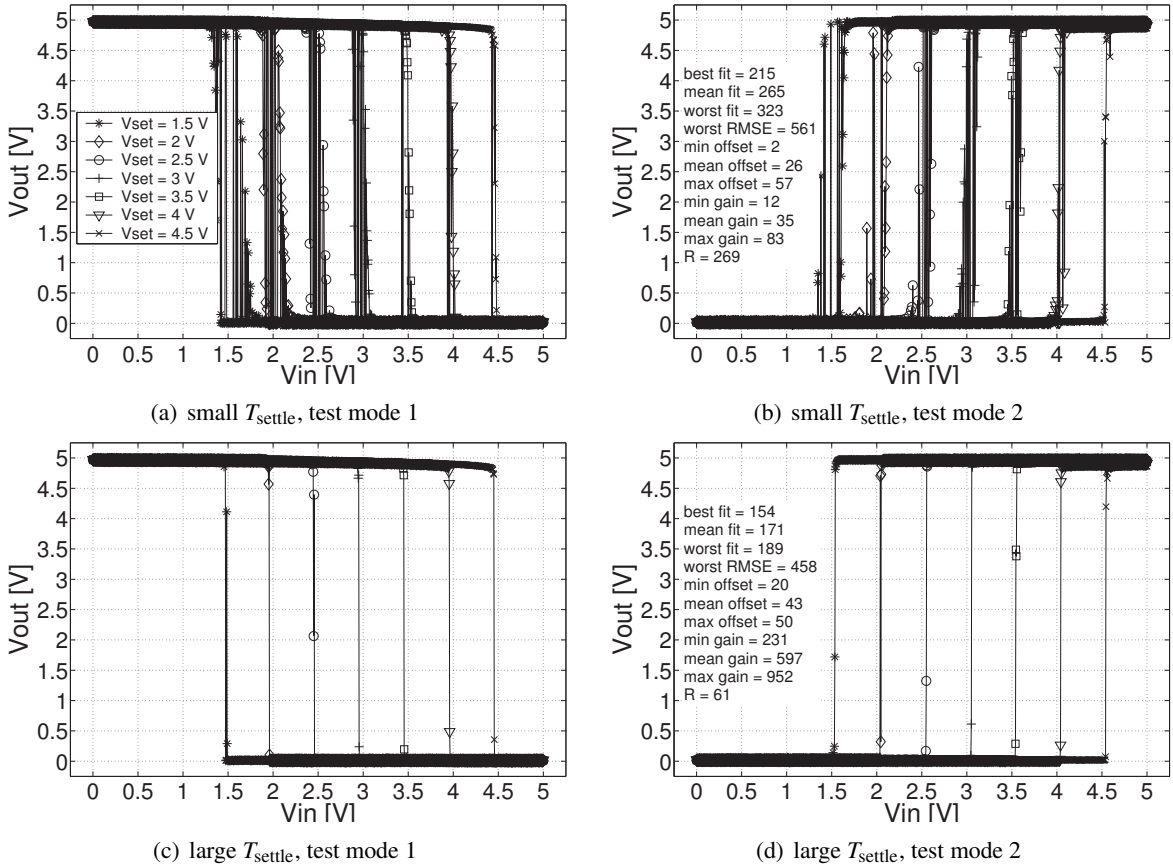
Experiment	Genotype	W	L	# of BBs ($X \times Y$)	Geometry in Figure
1	plain	free	free	2×3	8.14(b)
2	BB	fixed	fixed	2×3	8.14(a)
3	BB	free	fixed	2×3	8.14(a)
4	BB	free	free	2×3	8.14(a)
5	plain	free	free	3×5	8.14(d)
6	BB	fixed	fixed	3×5	8.14(c)
7	BB	free	fixed	3×5	8.14(c)
8	BB	free	free	3×5	8.14(c)

Table 8.7: Overview of the 8 different experiments.

posed on the genotype as well as the size of the array available to the artificial evolution process. The latter one distinguishes experiments 1 to 4 from experiments 5 to 8, which are identical otherwise. The two different array sizes refer to the four different geometries displayed in Fig. 8.14. While experiments 1 and 5 utilize the plain genotype, the remaining six experiments rely on the building block library described above. They merely differ in the freedom of (re-) sizing the gate dimensions of the transistors the BBs are composed of. While experiments 2 and 6 guarantee the genuine functionality of BBs containing more than one transistor, the freedom to change the respective gate dimensions granted to the algorithms in the experiments 3,4 and 5,6 may lead to more unconventional usage of the subcircuit topologies offered by the BB library. On the other hand, the GA can optimize the circuit's performance by tweaking the transistor dimensions. The compromise of fixing the transistor lengths corresponds to the unrealistic hope that the algorithm may be able to define current mirrors with different gains, but sticks to identical aspect ratios where necessary, e.g. in case of differential pairs. Since the smaller geometrical setup of experiments 1 to 4 is considered almost minimal for a useful comparator design (compare the hand-designed comparator of Fig. 8.19(c)), the second group of experiments is meant to investigate whether a larger number of BB sites would be beneficial. The larger geometry could ideally be used to construct three current conducting stages featuring up to 5 transistors or transistor pairs stacked onto each other, allowing for fully cascaded stages.

8.5.5 Test of the Hand-Designed Comparator

In order to set the scene for the results achieved for the different experiments, the 2-stage open loop comparator depicted in Fig. 8.19 has been subjected to the same verification test procedure as the evolved circuits. The resulting output characteristics are shown in Fig. 8.23. The tests were carried out using two different timing schemes: While Fig. 8.23(a) and 8.23(b) show the results for the test



(a) small T_{settle}, test mode 1

(b) small T_{settle}, test mode 2

(c) large T_{settle}, test mode 1

(d) large T_{settle}, test mode 2

Figure 8.23: Output characteristic of the hand designed comparator. The legend imprinted in Fig. (a) is used throughout all four plots. The plots on the right hand side contain some information about the measured fitness, RMS error, offset and gain.

pattern timing discussed in section 8.5.3.2, the settling time between the sampling clock for the first of the two input voltages (V_{N}) and that one sampling the output is raised from 1.075 µs to 5.275 µs for the test pattern underlying the results presented in Fig. 8.23(c) and 8.23(d). In the latter case, the desired output behavior of a comparator can be observed. In the former case this is not true for the region close to the trip point, which is due to the small settling time: For large input voltage differences, the circuit is still fast enough to achieve the desired output voltage transition if necessary; for smaller input voltage differences, the circuit is not fast enough anymore.

The plots (b) and (d) of Fig. 8.23 contain some information about the fitness and RMS error defined in (8.9) and (8.11) as well as the achieved gain and offset. The definitions for the latter ones as well as for the overall offset R are given in section 8.5.6.3. As can be seen from those values as well as from the plots themselves, the hand-designed comparator is not perfect even for the slower timing, but rather exhibits a mean offset of about 45 mV. This surprisingly high value may either be caused by the transistor cell implementation on the PTA itself or by errors introduced by external circuitry. The former source of errors is likely to contribute significantly stronger; they could either be caused by device to device variations between the used transistor cells, or by differences in the source potential of the differential pair that receives the input signals V_{N} and V_{P}. The latter condition can be caused by a different number of switches connecting the source terminals to a common potential, because all switches create a voltage drop according to their on-resistance.

It is instructive to contemplate the large RMS error per data point in mV, which even for the slower timing amounts to almost 500, although the circuit works fairly well. Let us assume a mean offset per curve of 50 mV to investigate its impact on the RMSE. Since the resolution of test points is set to 5 mV and the curve belonging to one V_{set} contains 1001 test points, the resulting error can be computed by

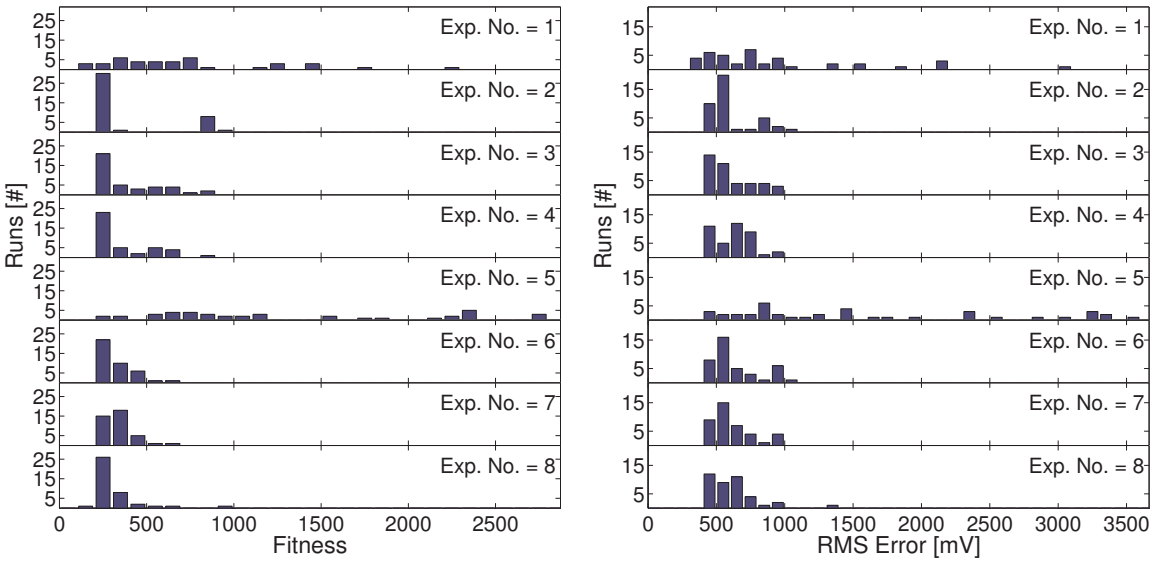
$$\text{RMSE}\,[\,\text{mV}] = \sqrt{\frac{10\cdot(5\,\text{V}-0\,\text{V})^2}{1001}}\cdot 1000 \approx 500\,\text{mV}\ . \tag{8.13}$$

Accordingly, the large RMSE of the hand-designed comparator is mainly due to its offset, as could be expected from the plots (c) and (d) in Fig. 8.23 that render its output behavior almost flawless otherwise.

8.5.6 Evolution Results

8.5.6.1 Comparison of the Different Experiments: Histograms

The results of all 40 runs of all 8 experiments are summarized in the respective histograms of Fig. 8.24. In order to allow for their comparison with results attained in experiments using a different fitness



(a) fitness criterion used during evolution

(b) RMS error [mV]

Figure 8.24: (a) worst fitness and (b) worst RMS error obtained from 100 verification tests for all 8 experiments. The bin size is set to 100 mV in all histograms.

criterion, the results are plotted in terms of the fitness criterion of (8.9) used during the evolution process as well as in terms of the RMS error per input voltage pair in mV, which is defined in (8.11) and (8.10).

In both histogram stacks, the non-BB experiments possess a very different signature compared with the BB-based experiments: While most of the runs of the latter ones are confined to fitness values between 300 and 500 (RMS errors between 400 and 1000 mV), those of the former ones are distributed much more uniformly along the plotted fitness (RMS error) range. In fact, none of the BB-based experiments produced a circuit with a fitness worse than 1000. On the other hand, the best solutions of experiment 5 fall into the same fitness bin as those of experiments 2-4 and 6,7; a fitness of less than 200 is achieved by more runs of experiment 1 than for all BB-based experiments together.

In summary, the experiments based on the building block concept generate good solutions (in all BB-based experiments more than half of the 40 runs finish with a fitness of less than 300) much more frequently and reliably than the experiments employing the simple genotype; however, the best runs of both experiments are of comparable fitness.

With regard to experiments 1 and 5, the larger geometry of 10 transistor cells turns out to be unfavorable for the evolution of comparators with the given GA and its parameters. Supposedly, the search space is simply too big for the number of circuit evaluations granted to the algorithm. In comparison with the results concerning the evolution of comparators using the *basic* GA reported in [Tre04], the RMS errors achieved in the runs of experiment 1 are of similar quality, albeit slightly inferior. Apart from a variety of small differences between the two experiments – including, but not limited to fitness function, test pattern, settling time, electrical test modes, parsimony pressure, number of transistor cells available to the algorithm – Trefzer et al. used five times as many circuit evaluations per run. Accordingly, the evolution runs of experiment 1 possibly could have gained from a larger number of circuit evaluations, too. Hence, this indicates that the GA parameters summarized in Table 8.6 are not a particular bad choice for evolving comparators with the transistor cell based genotype.

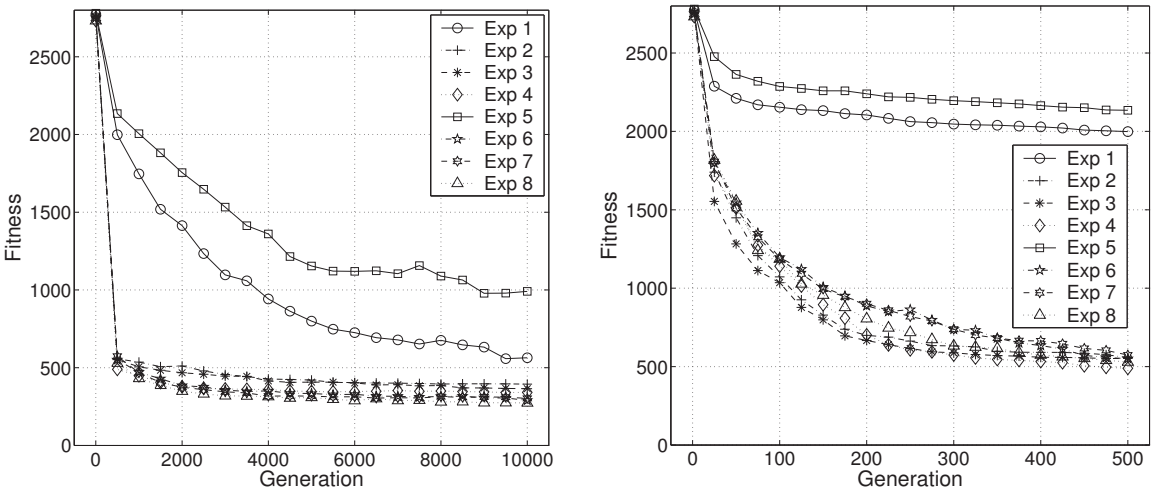
The differences between the six experiments employing the BB methodology are almost negligible. If at all, the confinement of the fitness values slightly increases with ascending experiment number, that is for higher degrees of freedom in sizing the BBs's transistors and a larger number of BB sites; the effect turns out even more subtle for the RMSE values.

The best results of the presented experiments compare well with the hand-designed comparator discussed in section 8.5.5, even if one compares their performance with that of the hand-designed comparator measured for the larger settling time: In terms of fitness values, most of the best-of-experiment runs are slightly worse, but in terms of RMS error they usually fall into the same fitness bin, and can thus be considered of higher value, because they are tested using the smaller settling time.

8.5.6.2 Comparison of the Different Experiments: Convergence

The convergence speed with which the best-of-generation solutions improve is investigated by means of Fig. 8.25. In (a), the mean fitness averaged over all 40 runs of each experiment is plotted for multiples of 500 generations; (b) presents a close-up into the first 500 generations on the basis of the average fitness taken every 25 generations. Please note that fitness here, unlike all other utilizations of this term within this section, refers to the fitness values measured during the evolution process.

First, it can be observed that the average fitness values obtained for generation 10,000 – at the end of each evolution run – are consistent with the histograms shown in Fig. 8.24. Second, the convergence speed for all of the BB-based experiments exceeds that of the experiments restricted to pure transistor cells by far. In fact, the average fitness has already dropped to approximately 130% of its final value for all BB-based experiments, whereas the plain genotype experiments – at least on average – are not anywhere near the synthesis of useful comparators. In case of the BB-based experiments, evolution seems to switch from the *exploration* to the *exploitation* phase after approximately 500 generations. A closer look reveals that it is beyond generation 500, where the experiments allotted a larger design space in terms of the number of BBs, routing and transistor sizing overtake those restricted to a smaller design space. In summary, the runs relying on the building block concept converge faster and more reliably than non-BB runs. Moreover, their fitness curves are surprisingly smooth and almost monotonically decreasing.



(a) generations $0 \ldots 10,000$ (b) generations $0 \ldots 500$

Figure 8.25: Mean fitness averaged over all 40 runs of each of the 8 experiments for selected generations. (a) displays averages for the full length of the runs, (b) zooms into the first 500 generations.

8.5.6.3 Comparison of the Different Experiments: Gain and Offset

Calculation of Gain and Offset. Comparators that, in principle, accomplish the desired functionality can be characterized by their offset and gain as well as their dynamic properties, as for instance, their propagation delay and slew rate. Due to the nature of the fitness test, all of the evolved comparators exhibiting the principal functionality must satisfy the timing constraints inherent to the test pattern set forth in section 8.5.3.2. Hence, the evolved circuits are further characterized by their gain and offset, whose computation is sketched in Alg. 8.1. The procedure is applied to the output characteristics measured in the verification tests described in section 8.5.3.3. Gain and offset are first calculated for all 14 *curves*, that is for the output characteristics of the comparator measured when V_{set} is fixed to one of seven values for both test modes. From these 14 gain and offset values the statistical parameters minimum, mean and maximum gain are calculated for each comparator.

The computation of the offset and gain of each curve features the following useful properties: First, for a principally correct output behavior, the offset is approximated well, whereas the gain may be underestimated for higher values. For instance, comparator curves exceeding the maximum detectable gain, which amounts to approximately $800 \ldots 1000$, may be measured such, that one of the V_{sweep} values falls into the transition region. As a result, the gain would be computed to a value between 400 and 500. Second, comparator curves that exhibit oscillatory behavior in the vicinity of the transition region are assigned a reasonable estimation of offset and gain; the larger the region of oscillations, the smaller their gain[6]. The most likely source for such oscillations is the inability of the comparator to decide upon the correct output voltage within the allowed settling time. Finally, curves that do not possess a transition between the two threshold voltages V_{OH} and V_{OL} at all or whose overall transition possesses the wrong polarity receive a gain of zero and an offset that is large enough to clearly distinguish the corresponding circuit by means of the according mean or maximum offset value. Since those circuits fail to produce the desired output functionality, their mean offset and gain values are not properly defined. Therefore, they are discarded from the following analyses that involve the computation of gain and offset. It should be noted, though, that comparators that perform

[6]Usually one is rather interested in the resolution $[\text{V}] = \frac{V_{\text{OH}} - V_{\text{OL}}}{gain}$ than the gain of a comparator. In this vein, the definition of gain is well chosen.

Algorithm 8.1: Computation of offset and gain

for all test modes **do** *// All 2 test modes (tm)*
  for all V_{set} **do** *// All 7 curves*
    if test mode == 1 **then** *// transition from high to low*
      $V_{\text{sweep}_l} \leftarrow \max\{V_{\text{sweep}} | V_{\text{out}}(V_{\text{in}}) > V_{\text{OH}} \,\forall\, V_{\text{in}} \leq V_{\text{sweep}}\}$;
      $V_{\text{sweep}_h} \leftarrow \min\{V_{\text{sweep}} | V_{\text{out}}(V_{\text{in}}) < V_{\text{OL}} \,\forall\, V_{\text{in}} \geq V_{\text{sweep}}\}$;
    else *// transition from low to high*
      $V_{\text{sweep}_l} \leftarrow \min\{V_{\text{sweep}} | V_{\text{out}}(V_{\text{in}}) < V_{\text{OL}} \,\forall\, V_{\text{in}} \leq V_{\text{sweep}}\}$;
      $V_{\text{sweep}_h} \leftarrow \max\{V_{\text{sweep}} | V_{\text{out}}(V_{\text{in}}) > V_{\text{OH}} \,\forall\, V_{\text{in}} \geq V_{\text{sweep}}\}$;
    end if
    if $\exists\, V_{\text{sweep}_h}, V_{\text{sweep}_l}$ **&&** $V_{\text{sweep}_h} > V_{\text{sweep}_l}$ **then**
      $\text{offset} \leftarrow \left| V_{\text{set}} - \dfrac{V_{\text{sweep}_h} + V_{\text{sweep}_l}}{2} \right|$;
      $\text{gain} \leftarrow \dfrac{V_{\text{out}}(V_{\text{sweep}_h}) - V_{\text{out}}(V_{\text{sweep}_l})}{V_{\text{sweep}_h} - V_{\text{sweep}_l}}$;
    else
      offset $\leftarrow 70\,\text{V}$;
      gain $\leftarrow 0$;
    end if
  end for
end for
// Calculate statistical information

min offset $\leftarrow \min_{\text{tm},V_{\text{set}}}(\text{offset})$;	max gain $\leftarrow \max_{\text{tm},V_{\text{set}}}(\text{gain})$;	*// best case*
mean offset $\leftarrow \text{mean}_{\text{tm},V_{\text{set}}}(\text{offset})$;	mean gain $\leftarrow \text{mean}_{\text{tm},V_{\text{set}}}(\text{gain})$;	*// mean condition*
max offset $\leftarrow \max_{\text{tm},V_{\text{set}}}(\text{offset})$;	min gain $\leftarrow \min_{\text{tm},V_{\text{set}}}(\text{gain})$;	*// worst case*

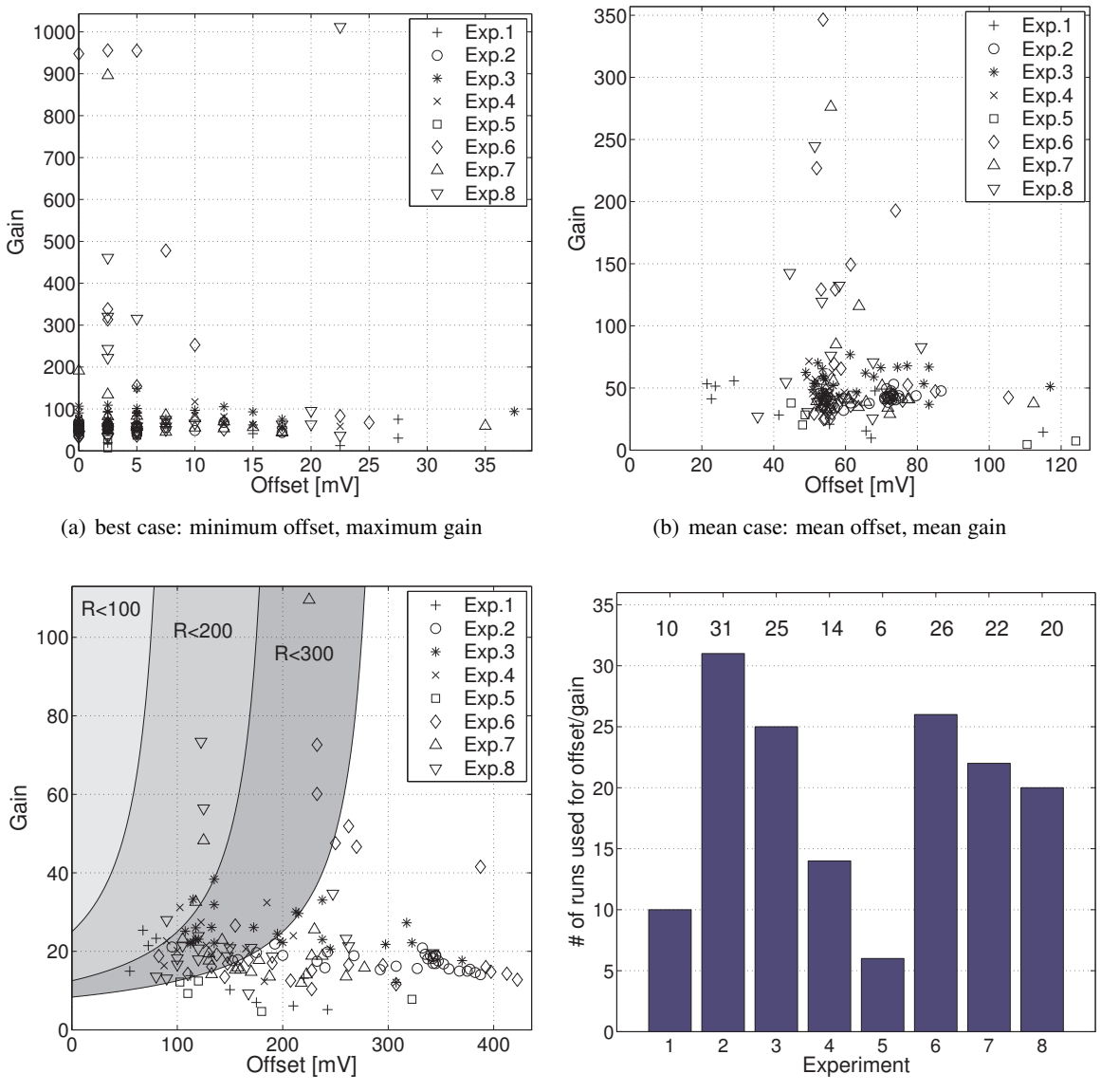
well for five or six of the V_{set} voltages while failing to produce a transition for the remaining one or two curves are evolved quite frequently; in fact, the best circuit of experiment 8 exhibits this behavior, as can be seen from Fig. 8.28.

Results for Gain and Offset. The number of circuits that are found to accomplish the comparator functionality in principle for all V_{set} and both test modes are presented in Fig. 8.26(d). In accordance with the fitness results summarized in Fig. 8.24, the BB-based experiments produced more *feasible* comparators than experiments 1 and 5 that employed the plain transistor cell genotype. The second trend recognizable from Fig. 8.26(d) is rather surprising: Additional freedom in resizing the BBs seems to reduce the yield of *feasible* solutions. Fig. 8.26(a)-(c) show offset and gain of the *feasible* comparators accounted for in (d). Thereby, three cases based on the statistical information extracted from the offsets and gains belonging to the 14 tested curves are distinguished: While (a) illustrates the best case, i.e. minimum offset and maximum gain recorded for the 14 V_{set}, (b) accounts for the mean offset and gain values and (c) captures the worst case, i.e. the maximum offset and the minimum gain. In order to increase the information in the scatter plots of Fig. 8.26(d), a few points exceeding the offset scale of the respective graphs are omitted; all of them possess a relatively very low gain. In particular, these are 2 runs of experiment 5 in (a), 1 run of experiment 5 in (b) and 1 run for each of the experiments 3,5,6,7 in (d).

As can be seen from Fig. 8.26(a), the vast majority of evolved circuits possesses a maximum gain of 100 or less. Maximum gains exceeding 100 exclusively emerge from experiments employing the BB concept, gains exceeding a threshold of 200 are all ascribed to runs using 3×5 BBs. The latter

statement also holds true for the mean gain values exceeding 100 and the minimum gains exceeding a value of 40. The smallest mean and minimum offsets, on the other hand, are attained by a group of 4 runs obtained from experiment 1. Thus, the best runs of experiment 1 seem to excel by their low offset, whereas those of experiments 6 to 8 are characterized by their relatively large gain; this perception is sustained by the examination of the best-of-experiment circuits in section 8.5.6.4. Both, offset and gain values exhibit a fairly large spread in terms of their minimum, mean and maximum values, i.e. many comparators seem to work fairly well only for a subset of the chosen V_{set} values and/or test modes.

From an application engineer's point of view, a circuit is best characterized by the minimum performance that can be guaranteed for the desired operation region. In this case, this would be the



(a) best case: minimum offset, maximum gain

(b) mean case: mean offset, mean gain

(c) worst case: max offset, min gain

(d) # of experiments considered for the offset/gain analysis

Figure 8.26: (a)-(c) Scatter plots of offset and gain for all the best-of-run solutions of all 8 experiments that exhibit the principle comparing functionality for all V_{set} . (d) accounts for the number of runs per experiment considered in (a)-(c).

overall resolution with which two signals can be guaranteed to be correctly compared for all V_{set} of both test modes. Thereby, the overall resolution can be defined as

$$R[\text{mV}] = \max_{\text{tm},V_{\text{set}}}(\text{Offset}) + \frac{1}{2} \cdot \frac{5000\,\text{mV}}{\min\limits_{\text{tm},V_{\text{set}}}(\text{gain})} \quad , \tag{8.14}$$

where the second addend accounts for half of the interval in which the comparator does not produce a unique decision due to its finite gain. Since a value outside the interval $[V_{\text{OL}}, V_{\text{OH}}]$ is considered sufficient for a correct decision, the maximum value of 5000 mV ensures that the second addend of R can only be overestimated. For input voltage differences larger than this overall resolution, the circuit is bound to produce the correct output voltage exceeding the respective thresholds V_{OL} and V_{OH}. The according regions in the offset-gain space satisfying an overall resolution of $R < 100$, $100 \leq R < 200$ and $200 \leq R < 300\,\text{mV}$ are highlighted in different shades of gray in Fig. 8.26(c). While only 10 runs exhibit an overall resolution of $R < 200\,\text{mV}$, a larger fraction of the 154 evolved comparators accomplishes at least a resolution of less then 300 mV. To be useful in a real world application however, an overall resolution of at least less than 20 mV would be needed; high-precision applications may even demand resolutions below 1 mV.

8.5.6.4 Best-of-Experiment Output Characteristics

The output behavior of the best comparators evolved for each of the 8 experiments is visualized in Fig. 8.27 (experiments 1-4) and 8.28 (experiments 5-8). Here *best* refers to the worst fitness attained within the 100 verification tests for the criterion used during evolution described by (8.9). The plotted data is taken from the first of the 100 verification tests. The output behavior measured for test mode 1 is depicted on the left hand side and that for test mode 2 on the right hand side of Fig. 8.27 and 8.28, respectively. Analog to the corresponding plots for the hand-designed comparator in Fig. 8.23(b) and (d), the graphs for test mode 2 contain additional information about fitness and RMS error values as well as for offset, gain and the overall offset R.

All of the output characteristics presented in Fig. 8.27 and 8.28 approximate the desired comparator functionality relatively close, at least for most of the V_{set} voltages. Moreover, the corresponding best-of-experiment comparators respond symmetrically to the two different input patterns of the two test modes. While the largest offsets for the comparators of experiments 1 and 5 (composed of simple transistor cells) are due to the deterioration of the curves corresponding to the outmost set voltages V_{set}, the maximum offset of the comparators evolved in experiments 2-7 (assembled from BBs) is dominated by the offset obtained for the curves with $V_{\text{set}} = 1.5\,\text{V}$ and in experiment 8 by the failure to produce and output transition at all for $V_{\text{set}} = 4.5\,\text{V}$. Yet, the affected curves of experiments 2-7 are not deteriorated in shape compared with the rest of the curves. Unlike the hand-designed comparator, the comparators synthesized in experiments 2-7 do not exhibit a unique offset for all curves (or at least all in a presumably feasible connected operation region), but rather significant offsets for one or two of the inner set voltages, most often for $V_{\text{set}} = 4\,\text{V}$. The only – unsatisfactory – explanation is that those circuits are *unconventional* in the sense of being very different from human designs.

As could have been expected from the analysis of offset and gain in section 8.5.6.3, the gain achieved by the circuits synthesized on the basis of the 3×5 BBs geometry significantly exceeds that achieved by all of the other presented comparators: While the according curves of the latter ones all resemble the output characteristic of an inverter, those of the former ones exhibit a steep and narrow transition between the low and high output voltages. Yet, these high gain values achieved for the best comparators of experiments 6 to 8 are countervailed by maximum offsets that are at least twice as high as those of the other best-of-experiment solutions.

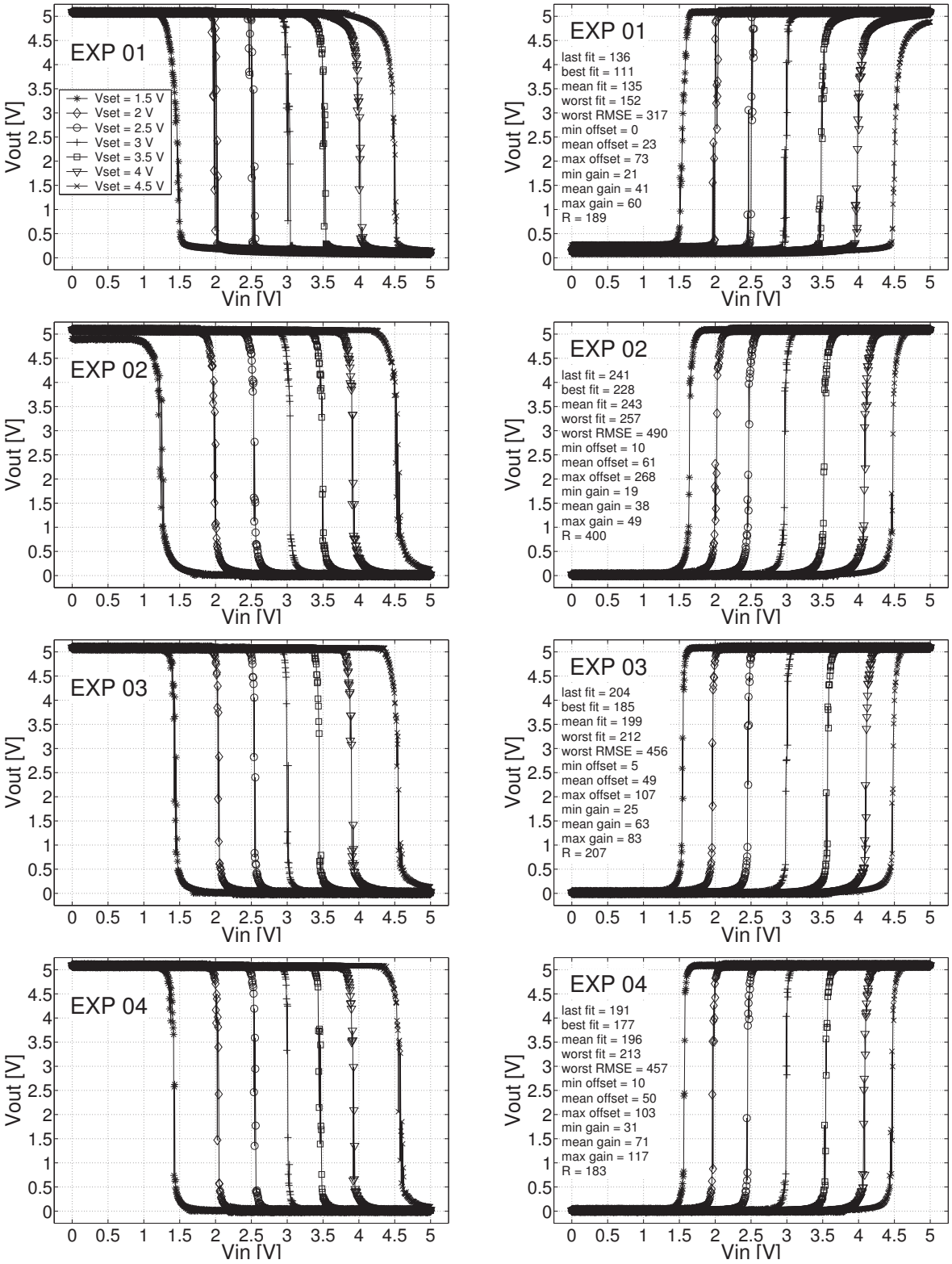


Figure 8.27: Output characteristic for best-of-experiment solutions for experiments 1...4. The data is obtained from the first of 100 verification tests, where *best* here refers to the worst fitness value obtained in 100 verification tests evaluated by means of (8.9), the fitness criterion used during evolution. **Left**: Output behavior for test mode 1. **Right**: Output behavior for test mode 2. Read from top to bottom the plots correspond to experiments 1 to 4.

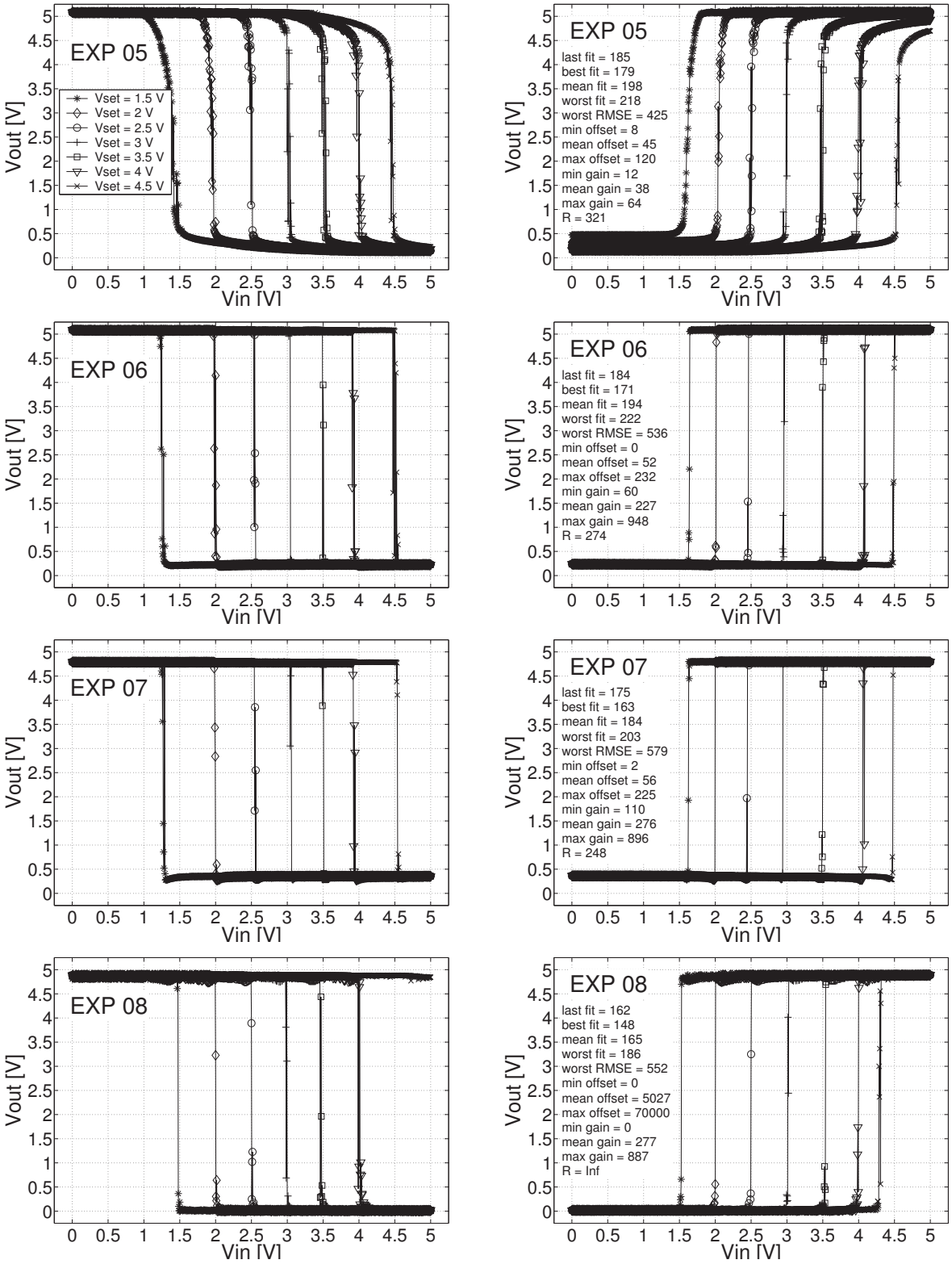
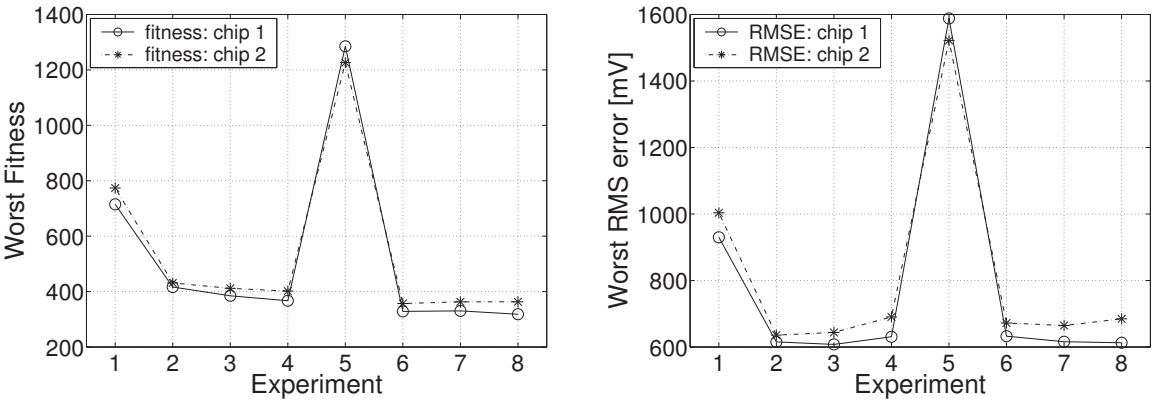


Figure 8.28: Output characteristic for best-of-experiment solutions for experiments 5 . . . 8. The data is obtained from the first of 100 verification tests, where *best* here refers to the worst fitness value obtained in 100 verification tests evaluated by means of (8.9), the fitness criterion used during evolution. **Left**: Output behavior for test mode 1. **Right**: Output behavior for test mode 2. Read from top to bottom the plots correspond to experiments 5 to 8.

In summary, all of the presented best-of-experiment solutions are most severely flawed at one or both ends of the range of set voltages. Yet, while for the BB-based circuits the according output curves mainly suffer from a displacement of the transition region, i.e. an offset, those of the comparators synthesized from plain transistor cells are mainly deteriorated in shape, i.e. suffer from a loss of gain. Finally, the relatively small spread in the last, minimum, mean and maximum fitness reported for the circuits presented in Fig. 8.27 and 8.28 proves their performance to be reproducible and thus renders them robust against at least the smallest of environmental changes.

8.5.6.5 Test on a Second Chip

Analog to all previous case studies, the verification test procedure is repeated on a second evolution system. The values obtained for the worst fitness according to (8.9) as well as for the worst RMS error described by (8.11) are averaged over all 40 runs of each experiment. The resulting means for both chips are compared to each other in Fig. 8.29. The average performance of the runs of all



(a) fitness criterion used during evolution

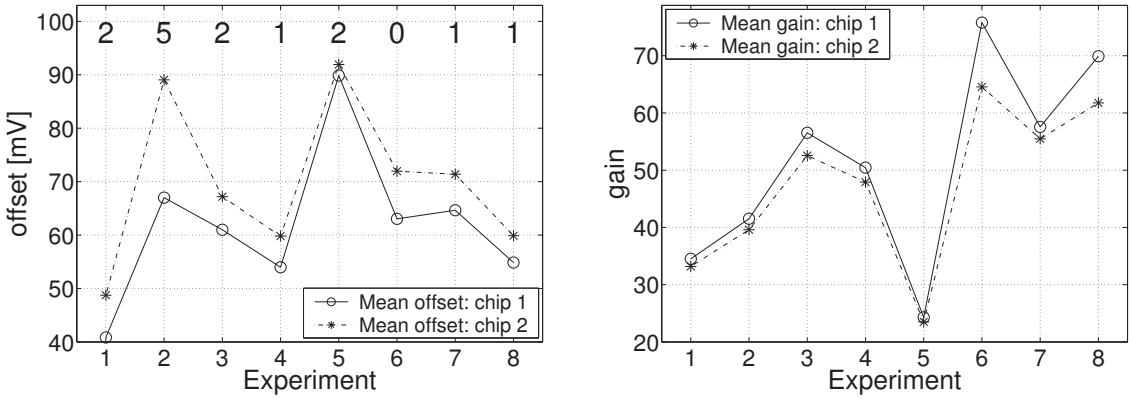
(b) RMS error [mV]

Figure 8.29: Comparison of the performance achieved on chip 1 and 2 in terms of (a) the fitness criterion used during the evolution process described by (8.9) and (b) of the RMS error according to (8.11).

experiments but experiment 5 is slightly decreased. The effect is more significant for experiments 6 to 8, i.e. the experiments employing the 3×5 BB geometry. On average, the BB-based experiments using the larger geometry exhibit a slightly better fitness and an almost equal RMS error, albeit clearly outperform the results gained for the evolution experiments restricted to pure transistor cells.

The verification test data obtained on the two different chips is further analyzed with regard to offset and gain: From the first output characteristic measured in the 100 verification tests the mean gain and offset with regard to all V_{set} and test modes are calculated as described in section 8.26. The according gain and offset values are averaged over all runs that achieve a syntactically correct comparison (*feasible* comparators) for all 14 curves on *both* chips. The number of circuits that proved *feasible* on the chip used during evolution but *infeasible* when tested on the second chip is annotated at the top of graph (a) in Fig. 8.30 for each experiment. The according averages for the comparator's gain are depicted in Fig. 8.30(b).

The offset and gain differences between the different chips look slightly larger than those for fitness and RMS error. Moreover, they vary stronger between the 8 experiments; yet there is no direct correlation for larger differences in mean offset and mean gain. The percentage of circuits achieving syntactically correct comparisons for all V_{set} on chip 1, but not on chip 2 is below 10% for all experiments but 1,2 and 5. Nevertheless, on average approximately 50% of the circuits evolved



(a) offset (b) gain

Figure 8.30: Comparison of the performance achieved on chip 1 and 2 in terms of (a) offset and (b) gain of the evolved comparators.

using the BB concept achieve such syntactically correct comparisons on both chips. In summary, the interesting circuits that approximate the desired comparator behavior closely for most of the 14 V_{set} and test mode combinations are believed to do so on both chips and probably even on other dice of the FPTA.

8.5.7 Summary of the Comparator Results

Results for a Larger Settling Time. Another series of experiments has been carried out which is identical to the one presented here (compare Table 8.7) in all parameters but the timing scheme used for the application of the test pattern. This timing scheme is the same as that one used for the second test of the hand-designed comparator whose results are given in Fig. 8.23(c) and (d). The preeminent difference between the two timing schemes is the almost 5 times larger settling time between the application of the first input voltage and the sampling of the output, that amounts to 5.275 µs instead of the 1.075 µs used for the experiments presented so far. The results are presented in appendix E based on the same type of plots used in the previous section and shall be briefly summarized below.

Generally, the results obtained from the new series of experiments using the larger settling time are indeed very similar to the experiments presented before; yet, a few differences apply: The experiments restricted to the plain genotype produce even less circuits that exhibit a relatively good performance, hence, the gap between the non-BB-based experiments and the BB-based ones is even larger. In contrast, some of the runs of the BB-based experiments are found to be superior to the best-of-experiment solutions found for the smaller settling time. The highlights among these outperformers include worst fitness values of 54 and 79 as well as four cases of an overall resolution $R < 100\,\text{mV}$; the two best of which achieve an R of 48 and 75. In general, the same picture for the distribution of gain and offset emerges for the experiments of both settling times. Nevertheless, the distribution of the comparators' gains is shifted and stretched to considerably higher values. Finally, the results for the migration to a second die mainly differs in that the increase in offset is more evenly distributed, so that it looks like a systematic effect.

Comparison with the Hand-Designed Comparator. In terms of the overall resolution R the best solutions evolved for both settling times can compete with the hand-designed comparator: In case of the larger settling time, 47 out of 320 evolved circuits accomplish an overall resolution $R < 269\,\text{mV}$, that is 14.7% of the evolved circuits outperform the hand-designed comparator. Though the dynamic

behavior exhibited by the hand-designed comparator renders it useless for applications in which a comparison must be reached within the smaller of the two settling times, the comparator's failure in satisfying this timing constraint qualifies the chosen design problem as nontrivial. In this vein, the performance of the automatically synthesized circuits *is* remarkable. In case of the smaller settling time, only the best of the 320 automatically synthesized comparators achieves a smaller overall resolution than the 61 mV measured for the hand-designed one. Another three evolved comparators accomplish an $R < 100\,\mathrm{mV}$ and thus can be considered of almost equal quality. In this sense, one could look at the automatically synthesized circuits as being human-competitive, albeit with a modest yield rate. Yet, though being briefly optimized for speed, the hand-designed comparators do not necessarily establish a lower bound for the overall resolution achievable with the chosen topology, let alone for the set of topologies that can be implemented on the chosen array of transistor cells: At least the offset of the hand-designed circuit should be easily compensated by some small adjustments in the aspect ratios of the input stage.

In addition, there is also a much more fundamental difference between the hand-made comparator and the evolved ones: The two stage comparator of Fig. 8.19(a) is primarily a concept for a comparator that can be adapted to the circuit technology at hand. For substrates that provide transistors approximating the underlying ideal behavior sufficiently close, a suitable dimensioning of the circuit will yield predictably good results. Besides the general suspicion against evolved designs, two observations concerning the evolved circuits decrease the confidence to find concepts of similar quality: First, the evolved circuits suffer from a loss of performance when migrated to a second FPTA. An increase in offset alone could probably be fixed easily by some adjustments in some of the transistor sizes, but the decrease in gain is likely to resist any simple repair mechanism. Second, many of the best-of-experiment circuits depicted in Fig. 8.27-8.28 and Fig. E.4-E.4, exhibit offsets depending strongly on the different V_{set} values. Hence, they are not believed to be amenable to simple offset minimization techniques feasible for the hand-designed comparator, as those are restricted to compensating *global* offsets.

Future Work. As has already been mentioned above, the high offset of the hand-designed comparator as well as its consequential mediocre performance in terms of fitness and overall resolution R defined in (8.14) are surprising and disenchanting. Thus, it deems necessary to investigate the actual causes of those offsets as well as to characterize the precision that can be achieved with the current evolution system in terms of offset, deterioration and noise. On one hand, the subsequent insights can then be used to improve the existing system – e.g. by a better calibration of the external analog test circuitry – and to define maximum specifications feasible for circuits implemented or evolved on the FPTA. On the other hand, these insights may lead to modifications of the external test system as well as to a redesign of the FPTA chip itself.

The final comment addresses the fitness criterion. In the course of presenting the evolution results, the overall resolution R was identified as a viable means to capture the quality of potential comparator circuits. By definition (see (8.14)) the overall resolution emphasizes the *worst case* performance the circuit is guaranteed to satisfy. On the contrary, the fitness criterion used during evolution (cf. (8.9)) encourages the development of circuits that perform well *on average*. Though a fitness of zero ensures the minimal overall resolution that can be enforced with the applied test pattern, low but finite fitness values may very well lead to circuits failing to compare voltages from a subinterval of the input voltage range. Hence, the fitness criterion does not exactly describe the desired behavior. On the other hand, the last argument implies that the overall resolution R can only be established as a criterion for candidate circuits that already achieve a syntactically correct comparison over the full range of set voltages V_{set}. Hence, it may be a good idea to design a hierarchical fitness criterion that prefers circuits for which R can be calculated to those where this is not possible, and otherwise decides

in favor of the minimum R or the minimum penalty-based fitness of (8.9), whichever is applicable. Further sophistication of the testing procedure may include an adaptive (fitness dependent) schedule for the required thresholds V_{OL} and V_{OH}, the number and range of the V_{set} voltages and the resolution of the input voltages V_{sweep}.

8.6 Discussion

Two different types of building block libraries were used to tackle three different tasks: A small library of the most simple logic gates boosted the yield of perfect XOR/XNOR gates from 0 to over 50%. The same library, although unlikely to be a particular good choice for this task, substantially increased the yield of medium quality tone discriminators. Finally, the evolution of comparator circuits was greatly enhanced in terms of convergence speed and yield of good solutions by means of a more elaborate BB library of 124 subcircuits composed of 1 to 3 transistors. Coarse-grained, these results can be summarized as follows: The application of the proposed building block concept can significantly accelerate the evolution process, which must be attributed at least partially to the inherent reduction of the feasible search space. However, a quantum leap for the best circuits evolved can only be achieved if the BB library is well suited to the problem and the problem is sufficiently facilitated by the usage of the BB library. Moreover, the usage of a presumably inappropriate ensemble of BBs may affect the transferability to other FPTA chips, as was observed in case of the tone discrimination circuits.

Although the analog BB library deployed in the third case study was meant to bias evolution towards more human-like designs, the attained comparator circuits are found to behave differently from a manually implemented bread-and-butter comparator. A possible reason for the limited success of the BB concept may be the large number of building blocks implementing different routing variants of essentially the same functionality. The different degrees of freedom in resizing the current BBs of the genotype had little effect on the success of the automatic synthesis. This may be due to the fact that a topological change of a building block can only be achieved by reinserting a new one into the according BB site, thus overriding any previously evolved transistor dimensions.

Both of the above problems can be addressed by smoothing the mutation operator: The first of two approaches to be sketched here transfers the *genetic access rights* from the genotype-template to the implementations of the BBs. Thereby, the user could define a restricted set of operations allowed for a particular BB, which thus may truly represent one subcircuit in all necessary routing variants. Apart from the exchange of whole BBs, a set of different mutation operators can be used to change certain properties of the respective BB as e.g. the way it is connected to the rest of the circuit, if it possesses a connection to either of the power supplies or even change the polarity of its transistors. The price for this advanced BB concept is an increased difficulty and maybe even a loss in versatility as it may require to hard-code the building blocks. The second approach would be to split the BB exchange operator into one that can exchange the BBs on a global scope and another one that is restricted to replacing the current BB with one of the same *family* of BBs, which would probably comprise all BBs that provide different routing variants of the same electrical subcircuit. The latter operator may even be enhanced by allotting different probabilities to the prospective exchange candidates depending on their distance to the original BB measured in the number of one point mutations to morph one into the other. Last, but not least, an advanced crossover operator is definitely necessary to complete the proposed BB concept. In case of the geometry chosen in case study III, the obvious choice would be to restrict the crossover operator to the exchange of parts of or even entire columns of BBs. Thereby, the column structure and hence also the current path structure of the genotype would be preserved.

Finally, the current-path based synthesis proposed by Shibata in [Shi01] has been shown to be most promising in comparison with the concepts proposed by other researchers in the field. It may thus be rewarding to grapple with the inevitable difficulties of automatically mapping a schematic circuit to the FPTA chip and aim at a direct implementation of this current-path based synthesis method.

8.7 Disclaimer and Acknowledgment

First it should be noted, that material and experiments similar to those presented within case studies I and II are published in [Lan04]. Since the experimental setup closely resembles that used in [Lan04] the results are similar. Yet, the experimental data presented here is completely disjunct from that published in [Lan04] and all of the presented experiments were run by the author. Nevertheless, the author is indebted to his coworkers whose work entered into this section: The *genetic access rights* concept was developed and implemented by Martin Trefzer, who also helped in implementing the necessary functionalities of the BB concept, especially to adapt it to the *genetic access rights*. The implementation of the BB concept including the BB library editor was primarily worked on by Daniel Brüderle. The beginnings of his student's project are summarized in [Brü02]. Finally, the author adopted the following of the algorithmic sophistications described in section 8.5.3.1 and 8.5.3.4 of case study III: The concept of *mutants* as well as the penalty based fitness function of Equ. 8.9 were introduced by Martin Trefzer. The choice of GA parameters was also inspired by previous work done by Martin Trefzer. A similar fitness function as well as some of the abovementioned GA parameters are published in [Tre04]. The idea for the implementation of the fitness-dependent mutation rate together with a slightly different implementation was contributed by Miguel Garvie.

Conclusion

A Journey of a Thousand Miles
begins with a Single Step.

LAO TZU

In this thesis a novel approach to the automatic synthesis of analog circuits on the transistor level has been presented. An evolutionary algorithm is employed as an invention machine that creates new candidate circuits based on the information gathered from the performance of their predecessor. The evaluation of new candidate circuits uses a dedicated analog substrate, that is, an array of programmable transistors implemented as CMOS VLSI chip. It is the combination of a hardware-in-the-loop approach and the particular implementation of the analog substrate realized as a field programmable transistor array (FPTA) that renders the proposed approach (almost[7]) unique.

The project underlying this thesis produced two types of results. First, the approach sketched above has been realized as a hardware evolutions system, which consists of the software front-end DarkGAQT, a library of different experimental setups stored as configuration files, a mixed signal test environment realized as a PCI board[8], and the FPTA chip itself. Besides the three versions of this research tool that are in use at the University of Heidelberg, one version of this research tool has been made available to another research group at the University of Sussex, where it has been successfully used in different hardware evolution experiments [Gar04]. Second, the hardware evolution system has been applied to a variety of different circuit design problems yielding a vast amount of results. The suite of different experiments can be used as a benchmark for new algorithmic solutions whose results can be compared with those results that have been presented here. Both, the hardware evolution system as well as the experimental results will be summarized and discussed below:

Hardware Evolution System

Summary. The evolution system combines flexibility in the design of the evolutionary algorithm with the capability to test the evolving candidate circuits on the FPTA chip while allowing for fast circuit evaluation[9]. This is achieved by the following system architecture: An IBM compatible general purpose computer is used to accommodate the evolutionary algorithm itself, which is written in the C++ programming language. The candidate circuits are tested on the field programmable transistor array (FPTA) chip and the measurement results are evaluated by the host computer. The hardware

[7] As has been discussed in chapter 3, a very similar concept emerged independently in the research group of A. Stoica at the JPL.

[8] The PCI-based mixed signal test board Darkwing has actually been developed by other members of the Electronic Vision(s) group. Yet, the system has been adapted to the hardware evolution system in terms of an additional daughter board and further hardware control software within the scope of this thesis.

[9] The time needed for processing one individual is usually dominated by the hardware test itself and thus is largely problem specific.

test of candidate circuits is supported by a mixed signal test environment centered around the FPGA-based PCI board Darkwing [Bec01]. In particular, this test environment is used to enable real-time measurements, which are the prerequisite or evolving circuits featuring a desired temporal behavior.

The FPTA chip itself consists of an array of 16×16 transistor cells. Each cell can be either used as a P- or as an NMOS transistor whose channel dimensions can be chosen from 75 different W/L combinations. P- and NMOS cells are arranged in a checkerboard pattern. All of the boundary cells of the PTA are accessible for analog signal I/O through 64 configurable S/H cells. These provide ample means to generate complex analog test patterns from one single analog input as well as to sample any desired number of outputs that can subsequently be read out from the analog output of the chip. The PTA configuration comprises a total of $256 \times 22 = 5632$ coding bits and is stored in embedded SRAM cells to allow for fast unlimited reconfiguration. The chip was fabricated in a 0.6µm double poly triple metal CMOS process.

Discussion. Most of the experiments on intrinsic hardware evolution rely either on commercially available FPAAs [She02], [Gre04], [Zeb99] or on custom-made platforms that are restricted to connecting discrete devices via an analog switch matrix [San01]. Although usually providing good analog performance, FPAAs are usually to coarse-grained in that they allow only a limited number of configurations of higher level CABs and thus preclude the synthesis of transistor level circuits. On the other hand, hardware evolution platforms realized on the board level can, in principle, be designed to host any, not necessarily homogeneous, set of devices. Yet, on one hand, the implementation of large complex substrates is tedious. On the other hand, board level evolution platforms cannot be used to directly evolve monolithic CMOS circuits, which currently is the technology of choice for most applications. In contrast, the proposed FPTA lends itself to intrinsic hardware evolution experiments targeted at the transistor level synthesis of analog circuits. This outstanding feature is shared with a family of FPTA chips developed by the group of A. Stoica at the JPL (cf. section 3.11). In contrast to these FPTAs featuring cells that provide between 8 and 12 transistors, the cells of the chip presented here do not contain an inner structure that may bias artificial evolution towards conventional designs. Moreover, only the FPTA proposed here allows to choose different gate dimensions for each transistor. In this vein, the FPTA proposed in this thesis models the substrate available in the analog design process more closely.

In contrast to most of the reported intrinsic hardware evolution platforms, the evolution system proposed here combines high test rates with the ability to perform complex circuit test. While the utilization of dedicated test equipment like signal generators or spectrum analyzers is typically hard to integrate into a fast evolutionary loop due to a lack of a fast communication protocol ([Gre04]), the utilization of custom-made hardware may offer only limited analog precision or bandwidth. For instance, analog IO of the SABLES test environment used by Stoica et al. was limited to 100 kHz in 2002 [Fer02]. In summary, to the author's knowledge, the test sequences utilized in the hardware evolution experiments presented within this thesis are probably the most comprehensive ones reported from the evolvable hardware community. However, the current implementations of the hardware test system allows for processing over 550 individuals per second for simple circuit tests (cf. section 6.1.4).

Inherent Limitations. So far, the advantages of circuit evaluations performed directly in hardware in comparison with circuit simulations have been emphasized in this thesis. Nevertheless, the hardware also entails some disadvantages: First, physical measurements usually cannot provide the same precision as simulations. In case of the proposed hardware evolution system, the nominal analog precision is limited to 12 bit by the ADC used for converting the measured voltages. However, preliminary system tests revealed that the effective number of bits of the complete analog signal path

described in section 4.2.3 amounts only to 8.5 bits for moderate frequencies, for rail-to-rail signals[10]. This limited precision significantly aggravates — and in some cases will even impede — the correct evaluation of circuits featuring a high dynamic range, as e.g. high performance filters. Thus, for some classes of circuits, analog hardware evolution seems to be infeasible or at least impractical, as the effort to provide the necessary analog precision would be unjustifiable.

Second, programmable transistor cells are not transistors. As has been detailed in section 3.3.5, the configuration circuitry accounting for the 'programmable' in FPTA introduces parasitic resistors and capacitors. While the parasitic resistors deteriorate the dc behavior of the transistor cells compared to the behavior of genuine transistors, the parasitic capacitances change its dynamic behavior. Accordingly, the transistor 'model' used is wrong, where the deviations get worse for larger currents and frequencies. This raises the utterly important question if it is indeed possible to migrate the evolved circuits to their genuine fabrication processes. Yet, the answer to this question is beyond the scope of this thesis. Nevertheless, the parasitic effects have been discussed in greater detail in [Lan01]. An important result of which has been the observation that in some sense[11] the programmable transistor cells provided by the FPTA are approximately a factor of 100 slower than genuine transistors of the same geometry fabricated in the same fabrication process.

Probably the most severe implication of the finite on-resistance of the configuration switches on circuit performance is that it may effect matched pairs in the following way: Consider a matched transistor pair as ,for example, in a current mirror or differential pair. In order to attain equal currents through both transistors, the respective gate source voltages V_{GS} must be the same. If, however, the source terminals of both transistors are connected to a common potential via different number of switches, the currents of both transistors will differ. Thus the finite on-resistance of the configuration switches may hinder the evolutionary algorithm in finding solutions that employ the concept of matched transistor pairs that is of great importance for most analog circuits designed by humans. On the other hand, the evolutionary algorithm may also make use of the switches to compensate the device mismatch inherent to the programmable transistors. The situation could be mitigated by providing matched pairs to the evolution process, either as predefined building blocks encoded in software, or by direct integration into a second generation FPTA.

Experiments and Results

Within this thesis, a variety of different analog circuit design problems has been tackled. They can be classified by the prevailing type of circuit behavior that was sought. The design tasks can then be described as circuits whose analog dc behavior is prescribed as a function of the input voltage(s) (logic gates, Gaussian function circuit), circuits whose output must settle to a value determined by the input signal(s) within a given time (DACs, comparator), circuits featuring a prescribed transient behavior (tone discriminator), and circuits that are to meet a certain frequency behavior (LPF, HPF).

For most of the experiments, a GA-like simple algorithm has been used in conjunction with either truncation or rank-based selection. However, for the synthesis of XOR/ XNOR gates, tone discriminators, and comparators, two different user-defined building block libraries have been used. In general, the GA parameters have slightly been adapted to the problem based on pre-studies, but are far from being thoroughly optimized, where the parameters and fitness function are chosen somewhat more carefully in case of the comparator experiments. Except for the comparators, the primary fitness criterion was usually evaluated as the root means square deviation of the respective target behavior. For each type of experiment — describing a particular experimental setup — between 10 and 100

[10] However, in a restricted voltage range between e.g. 1 and 4 V the *relative* precision is believed to be higher.
[11] Actually for ring oscillators composed of inverters with specific channel dimensions.

evolution runs have been carried out to obtain insight into the statistical distribution of the resulting fitness values. The results accomplished for different experiments are usually compared for the best solutions evolved and with respect to the fitness distribution of the ensemble of evolution runs. For all experiments, the best circuit of each run was subsequently tested for 100 times outside of the evolutionary loop to verify that its performance is reproducible. Typically, the results are reported for the worst performance values attained within these 100 verification tests to account for the worst case scenario.

Summary of the Evolution Results

Logic gates For the possibly over-ambitious dc target specifications only few (5%) perfect solutions have been found for the NOR and NAND and even fewer (2%) for the AND and OR gates. None of the XOR and XNOR gates could match the target specifications; however, the XOR and XNOR implementations typically used in standard cell libraries would not have either. The difficulties encountered in synthesizing the different types of two-input symmetric logic gates reflect the complexity of their conventional CMOS implementations.

Gaussian function circuit Typically, the evolved solutions approximate the target output curve quite well. Yet, they do not exactly approximate its shape and are usually not differentiable at some input voltages. The best solutions achieve a relative mean deviation between 2.5 and 5%, which is slightly larger than the values reported for two similar but extrinsic evolution experiments as well as for one human design.

6-bit digital-to-analog converters Generally speaking, the results are surprisingly good in that most of the evolved DACs achieve a linearity of at least 4 bits while the better and best solutions exhibit a nonlinearity between 0.5 and 1 bit. The results are the more remarkable as the settling time between the last input voltage and the sampling of the output voltage amounted to less than 0.5 µs, especially if one takes into account that the substrate provided by the FPTA is probably a factor of 100 slower than its fabrication process. However, it has been shown that these solutions rely on the analog voltage levels of the digital input signals. Yet, it has also been shown that this dependence can be avoided by adding digital buffers at the input, which is supposedly a simpler task than the design of the DAC itself.

Frequency selective filters The artificial evolution of low- and highpass filters has been used to establish and test three different methods of evaluating the frequency response of the circuit under test. Here, measuring the step response and applying a Fourier transform turned out to yield the best results. On the other hand, applying sinusoidal input signals of different frequencies in conjunction with a Fourier transform applied to the according output signals is believed to be the most prospective method when applied more carefully. First, this method has been shown to possess the lowest noise floor of the three methods considered. Second, it stands out in that it offers the possibility to choose the tradeoff between linearity and magnitude response by means of a small set of weighting factors. The analysis of the evolution results revealed the following: First, LPFs exhibiting a second or third order rolloff behavior whilst adding a relatively small amount of distortion to the signal. Yet, the evolution results tend to get worse for lower corner frequencies. In contrast, the artifical evolution of HPFs proved to be more difficult than that of LPFs, in that, on average, results tended to be worse despite using a ten times larger number of generations. Probably, this has to be attributed to the lack of passive capacitors in the PTA.

Evolution based on a digital BB library A building block library consisting of the four simple logic gates has been employed to evolve XOR/XNOR gates as well as tone discriminators. While

the utilization of BBs boosted the yield of perfect XOR/XNOR gates from 0 to over 50%, it greatly increased the probability to find viable tone discriminators. While most of the better tone discriminators indeed provide significantly different output voltages for different input frequencies, none of them manages to provide the desired smooth glitch-free voltage levels at the output.

Evolution of comparators using an analog BB library The utilization of an analog building block library of 124 analog BBs comprising between 0 and 3 transistors has been demonstrated to a) greatly accelerate the evolution of comparator circuits and b) massively enhance the yield of good comparators. Moreover, the best comparators evolved from the BB library usually exhibit a larger gain than their counterparts evolved with the standard genotype representation presented in section 4.4.3. In terms of the achieved fitness results as well as for the guaranteed resolution, the best of the evolved comparators are competitive with a human-designed two stage comparator. Yet, the human-designed comparator suffers from an unusually high offset and has not been particularly well optimized for the task. The offset of this reference design may be due to different source resistances of matching transistor pairs, which has been explained above.

Reliability and Migration. Typically, the results obtained from the best-of-run solutions at the end of the evolution run and those attained in the verification tests coincide well for those design problems targeted at a particular dc behavior. This has been ensured by using randomized test patterns to avoid the temporal or ordinal structure of the test pattern to be exploited. Although with slightly less rigor, the above observation also applies to the tone discriminators which are characterized by their transient behavior. However, in case of the filter circuits evaluated by the deviation of the desired frequency behavior, large variations between the fitness values obtained from the 100 verification tests are observed. As frequency selective filters implemented on the FPTA substrate are bound to utilize capacitors, the evolutionary algorithm may produce circuits that exploit the given charge distribution on the PTA. If this is the case, the circuit will only work satisfactorily if the charge distribution indeed suits it. In order to prevent the evolution process from using such floating nodes, an analog substrate reset was introduced. For the proposed filter experiments, the FPTA was configured with a random gene, before the download of the respective candidate circuits. Yet, this mechanism did not entirely solve the problem, which may be partially due to the fact that the substrate reset was not applied between different test modes.

One of the long-term goals of this project is to evolve new circuits or circuit principles that can be migrated from the FPTA to other analog substrates, as e.g. new fabrication processes. Though far beyond the scope of this thesis, a first step in this direction has been taken in that all of the evolved circuits have been subjected to verification tests on a second hardware evolution system. In the vast majority of cases, the performance measured on the second FPTA matched that one obtained on the first chip, on which the circuits have been evolved, very closely. Yet, of all experiments, it is those circuits that are composed of the digital building blocks that experience a significant performance loss when used on the second chip. Hence, it must be concluded that simple logic gates, which guarantee robustness by large noise margins in digital designs, are not very well suited to evolve analog circuits with. This is in good accordance with the results from the tone discriminator experiments published by A. Thompson [Tho99].

Discussion. A comparison between hardware evolution results reported from different hardware evolution experiments is difficult and must not be overestimated, since the experiments usually differ in many regards, as e.g. the analog substrate, the chosen EA representation, the evaluation criterion,

the number of candidate evaluations spent, and the measurement/analysis mode. However, a few remarks seem to be adequate here:

With regard to most intrinsic hardware evolution experiments published in the literature, especially those that include topology synthesis on the transistor level, the evolution results presented in this thesis are highly competitive. Here, the results obtained from the logic gates seem to have yielded the poorest performance. Yet, this may at least partially be due to the (over-) ambitious specification of the target behavior. The second exception to the general competitiveness stated above concerns the tone discriminator experiments. In fact, the results suggest that artificial evolution using logic gates yields better results than experiments based on plain transistors. On one hand, this may explain, why the BB-based tone discriminators cannot match the performance reported by A. Thompson [Tho98b], as the latter ones had access to a larger number and variety of logic gates. On the other hand, all of the BB block experiments confirm that it is easier to evolve circuits from more complex structures than from transistors themselves. Yet, most often it is not that equally good solutions based on transistors could not be found; they are just less probably to emerge.

While the evolved DACs that are reported from in- and extrinsic evolution experiments to date are limited to three [Ben99] and four [Zeb03] bits of resolution, many of the other evolved circuits presented here are outperformed by circuits evolved extrinsically. This may be due to several different reasons: First, many of the extrinsically evolved circuits are not tested by a transient analysis. Yet, as dc and ac analyses each exclude some part of circuit's electrical properties, a transient analysis seems to be mandatory to warrant that the evolved circuits would indeed work in reality. In contrast, the evolved circuits presented here have been proven to work at least on two different FPTA chips. Second, a closer look on the results reported by A. Stoica et al. reveals the following: Typically, the performance of their evolved circuits is better for extrinsic experiments than for intrinsic ones. Moreover, they are usually best, if circuit representation has been used that is not constrained to the architecture of their FPTA. The advantages of unconstrained circuit representations are twofold. For one, they are not limited by any routing constraints. For the other, EAs that utilize representations directly reflecting the structure of a reconfigurable device must actually accomplish the electrical circuit design and the layout, that is the placing and routing onto the device, in one single step. In case of unconstrained representations, the second step is not considered at all. Hence, intrinsic hardware evolution bound to a concrete substrate is the more complex task.

Outlook for the Heidelberg FPTA Project

Despite the wealth of experiments and results presented within this thesis, the Heidelberg FPTA project still offers numerous possibilities to extend and improve the existing evolution system and a vast number of new research avenues. Some of them are sketched below:

Algorithm Design. First, as has been hinted in chapter 2, analog circuit design problems are in fact multiobjective optimization problems. Therefore, it is highly desirable to include appropriate extensions in the DarkGAQT software that allow for multiobjective optimization, which in fact has already been accomplished by M. Trefzer. In conjunction with the capability to evaluate the frequency behavior of candidate circuits that has been proposed in this thesis, this has led to a first approach to the automatic synthesis of operational amplifier circuits [Tre05].

Second, the previous discussion on unconstrained and constrained representations as well as the inability to evolve human-competitive circuits observed for almost all of the design problems tackled within this thesis reveal that better algorithms are desperately needed. Here, algorithms that allow for developmental growth, as, for instance, genetic programming, may turn out to enhance the evolu-

tionary process. In this vein, decoupling the electrical circuit design from the process of mapping the candidate circuit onto the chip is considered to be rewarding, albeit difficult to implement. Moreover, it may turn out to be necessary to include prior design knowledge into the algorithm, that is, in case of an evolutionary algorithm, into the representation.

Analysis of Evolved Circuits. One of the main goals of this project has been to use the FPTA as a search engine for new circuit designs that can be used in the design of other VLSI circuits. Yet, as has been pointed out above, this ability to migrate circuits to a fabrication process is challenged by the parasitics entailed in the configuration circuitry. In order to support the migration process, an export filter that translates the evolved FPTA configurations into SPICE-readable netlists has been added to the DarkGAQT software by M. Trefzer. First results on simulating evolved comparator circuits are promising [Tre04]. However, further work will be necessary to allow for a deeper understanding of the differences between the circuit behavior obtained from hardware measurements and the according simulation results. Essentially, it will be crucial to design a suited model of the programmable transistor cell that captures its prevalent parasitics and yet can be simulated with small computational effort. Eventually, it may deem necessary to evaluate the candidate solutions in software and in hardware. This concept is referred to as *mixtrinsic* evolution and has been proposed by Stoica et al. [Sto01b].

Mixed Signal Test Environment. The mixed system test environment can be improved in many regards. First, it would be desirable to increase the analog precision to allow evolving circuits featuring a high dynamic range. Second, complex experimental setups involving multiple test modes, test benches and analog substrate resets are not supported by dedicated VHDL modules. In such situations, moving some of these tasks into the FPGA can considerably increase the evaluation speed. Although the evaluation of candidate circuits can be faster in hardware than in software, the hardware test and possibly the whole evolution system will have to be parallelized to achieve competitive speed compared to multiprocessor computers or Linux clusters. A framework for the parallel training of analog neural network chips has been developed in the Electronic Vision(s) group [Fie04] and may also lend itself to hardware evolution experiments when appropriately adapted.

Second Generation FPTA. Although the proposed FPTA chip has been successfully used for a large variety of experiments, this thesis has also pointed out some of its limitations. A design of a second generation FPTA should therefore realize the following improvements: The problems caused by finite resistances connected to the source terminals of matched transistor pairs should be mitigated by providing matched pairs in some or all of the transistor cells. Another approach may be to directly connect several transistors of different cells to one common metal line. Moreover, an increase in routing capabilities seems also beneficial. Yet, the advantage of additional routing must be traded off against increased parasitic devices. As many analog circuits are characterized by an I-V characteristic, a future FPTA should provide analog input circuitry that is capable of current and voltage I/O. Finally, a wider data bus or another type of faster protocol for the transfer of the configuration data as well as a means for partial reconfigurations can speed up the reconfiguration, especially if the chip requires a larger configuration string.

Final Remarks

Originally, the Heidelberg FPTA project set out to pursue two different goals: First, to use intrinsic hardware evolution to search for new circuit topologies that can be used in other VLSI designs. Second, to provide a device that can be used as an FPAA that can be adapted to a variety of tasks

and environmental conditions by means of artificial evolution. To serve both of these purposes, the FPTA chip has been designed as a fine grained analog array with distinct programmable transistors which were meant to dominate its electrical behavior so that the emerging circuits can be understood. However, the results presented within this thesis render the proposed chip to be incapable of fully achieving either of these goals. On one hand, the parasitic devices entailed in the configuration circuitry are likely to prevent the successful migration of evolved circuits to new technologies. On the other hand, these parasitic effects decrease the analog bandwidth of the FPTA and probably deteriorate the fidelity of the analog signals. In addition to this drawback, the fine-grained character of the chip hinders the efficient evolution of human competitive circuits.

Therefore, it is believed that future research will have to pursue the aforementioned goals with different approaches. To tackle the synthesis of analog transistor level circuits a simulation based approach combined with sufficient computational power provided e.g. by a large Linux cluster will probably offer closer device modeling and more independence of one particular target technology. Moreover, a software based approach can easily be adapted to solve problems not related to analog circuit design. Field evolvable hardware devices whose electrical characteristics in terms of noise, distortion and bandwidth shall be competitive with commercially available FPAAs, will have to be based on higher level building blocks. Viable alternatives may be operational amplifiers, transconductance amplifiers, G_m-C cells, or current conveyors. As topology design persists to be much more difficult than parameter optimization, the ideal analog cell would be one whose behavior can be controlled by a quasi-continuous parameters as, for example, bias current and capacitor values in a G_m-C cell.

Appendix A

FPTA Chip: Configuration Details and Pad and Signal Description

This appendix collects some definitions and data not of interest to the scientific treatment of the FPTA in chapter 3, which nonetheless may be of interest to the reader in general and to a potential FPTA user in particular. The main ingredients of this appendix are: The assignment of the configuration bits needed for programming the FPTA, pad and bonding diagrams and description of the different analog and different signals of the FPTA and their location on chip and on the Brightwing connector.

A.1 Configuration Bit Assignment

The bit patterns necessary for configuring one programmable transistor cell are summarized in Table A.1. The information is arranged in subtables for the different aspects of the cell that can be configured. Each subtable contains all relevant bits on the left hand side and the resulting property on the right hand side. Thereby NC abbreviates 'Not Connected'. The bits are enumerated in the order that lends itself best to their being programming from outside of the chip. The bit enumeration is depicted in Fig. 3.2 and 3.11. The overall structure of the SRAM and the programming thereof is explained in section 3.4.

The awkward bit orders are not meant as a special nuisance to the user, but simply offered the easiest, most area efficient way of implementing/layouting the respective part of the transistor cell circuitry. Note, that for the analog multiplexers as well as for the W and L selection at least one bit pattern must result in a 'not connected' to allow the transistor to be turned off and remain unused. The bits 16 and 17 are not used. They are included in the chip for symmetry reasons and simplicity of implementation. First, the gain in silicon area would not have justified the more in design effort (WORD and BIT lines are necessary anyway). Second, it is common practice to add dummy structures to the layout in order to preserve symmetry.

Bit 0	Bit 1	Bit 2	GATE →
0	0	0	EAST
0	0	1	SOUTH
0	1	0	ANAVDD
0	1	1	WEST
1	0	0	NORTH
1	0	1	ANAGND
1	1	0	NC
1	1	1	NC

(a) Gate connection

Bit 6	Bit 7	Bit 8	DRAIN →
0	0	0	EAST
0	0	1	ANAVDD
0	1	0	NORTH
0	1	1	NC
1	0	0	SOUTH
1	0	1	ANAGND
1	1	0	WEST
1	1	1	NC

(b) Drain connection

Bit 9	Bit 10	Bit 11	Source →
0	0	0	EAST
0	0	1	SOUTH
0	1	0	NORTH
0	1	1	WEST
1	0	0	ANAVDD
1	0	1	ANAGND
1	1	0	NC
1	1	1	NC

(c) Source connection.

Bit 9	Bit 10	Bit 11	Transistor Length
0	0	0	$L_0 = 0.6\,\mu m$
0	0	1	$L_2 = 2\,\mu m$
0	1	0	$L_4 = 8\,\mu m$
0	1	1	NC
1	0	0	$L_1 = 1\,\mu m$
1	0	1	$L_3 = 4\,\mu m$
1	1	0	NC
1	1	1	NC

(d) Transistor length

Bit 15	Bit 21	Bit 22	Bit 23	Transistor Width
0	0	0	0	NC
1	0	0	0	$W_3 = 8\,\mu m$
0	1	0	0	$W_1 = 2\,\mu m$
0	0	1	0	$W_2 = 4\,\mu m$
0	0	0	1	$W_0 = 1\,\mu m$

(e) Transistor Width

Bit 12	Bit 13	Bit 14	Bit 18	Bit 19	Bit 20	Routing
1	0	0	0	0	0	NORTH ⟷ EAST
0	1	0	0	0	0	NORTH ⟷ WEST
0	0	1	0	0	0	EAST ⟷ WEST
0	0	0	1	0	0	SOUTH ⟷ WEST
0	0	0	0	1	0	SOUTH ⟷ EAST
0	0	0	0	0	1	NORTH ⟷ SOUTH

(f) Routing

Table A.1: Assignment of the 24 SRAM bits of one transistor cell: (a) gate connection, (b) drain connection, (c) source connection, (d) transistor length L, (e) transistor width W and, (f) routing bits. In case of (e) and (f) any bit combination is valid resulting in 15 different transistor widths $W = 1, 2, \ldots, 15$.

A.2 Bond Pads of the FPTA

Fig. A.1 illustrates the location of the bond pads of the FPTA. The numbers next to the bond pad names are referred to as pad numbers in this appendix. They are enumerated counter-clockwise starting at the upper left corner of the chip excluding the probepads. Note that some of the power pads possess the same name, albeit different pad numbers. The propepads are enumerated in the same counter-clockwise fashion as the main pads. However, only those that can be easily bonded, that is probepads 16 to 47 on the chip's southern and western edges are highlighted.

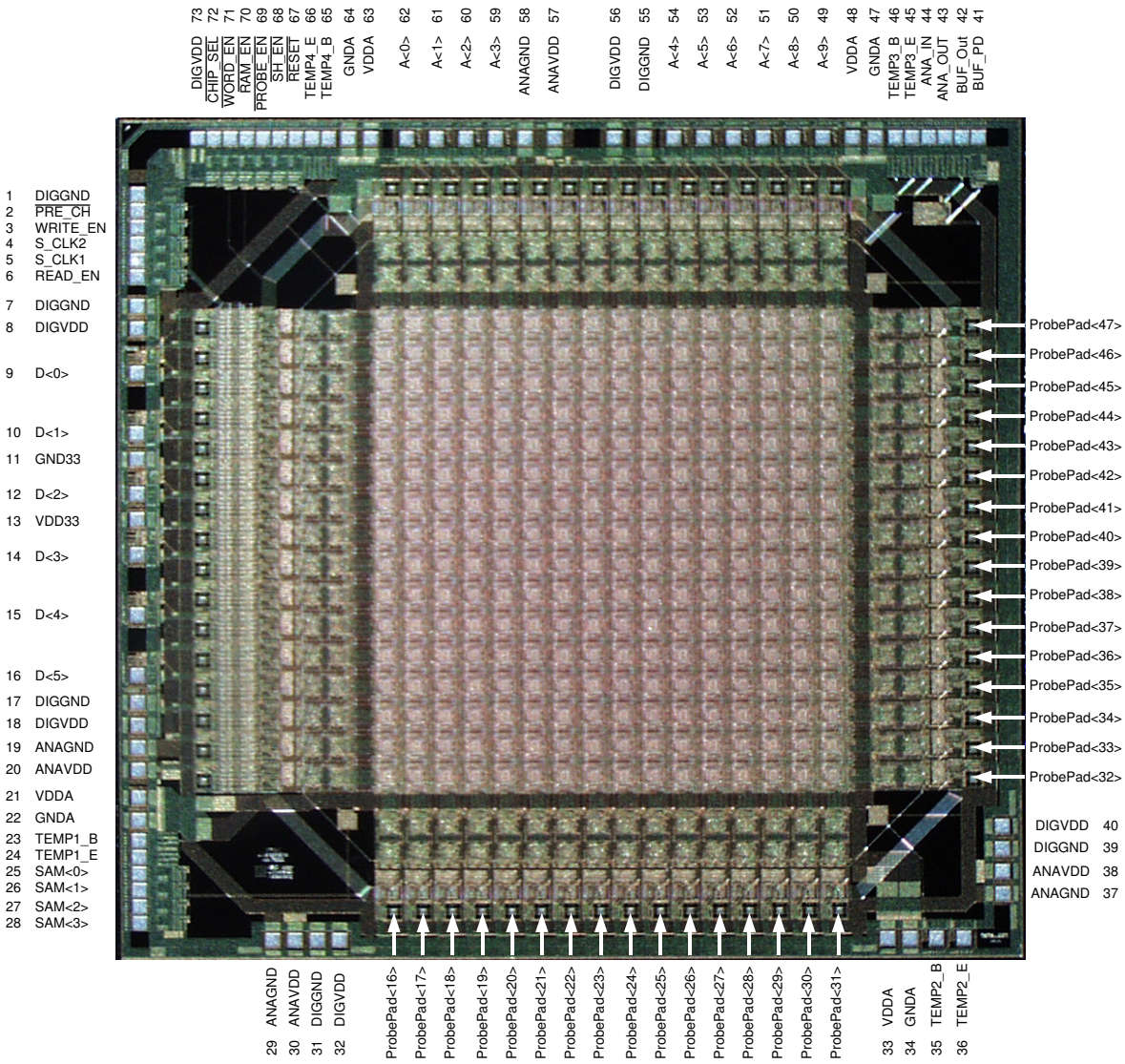


Figure A.1: Chip photo with pad names.

A.3 Bonding Diagram

The bonding diagram used for the chips currently in use is shown in Fig. A.2. A PGA144 is used as the chip carrier. Although not all of the 144 pins of the carrier are in use, it must be that large due to the large numbers of bond pads on the northern and western side of the FPTA, which are arranged in this way to allow for the direct connection of up to four FPTA chips via the probepads on its southern and eastern side. The pad/pin numbers of the PGA144 chip carrier are enumerated counter-clockwise starting from the upper left corner, too; they are included in Fig. A.2. The assignment pin numbers of the chip carrier and the pad numbers of the FPTA is contained in Table A.5.

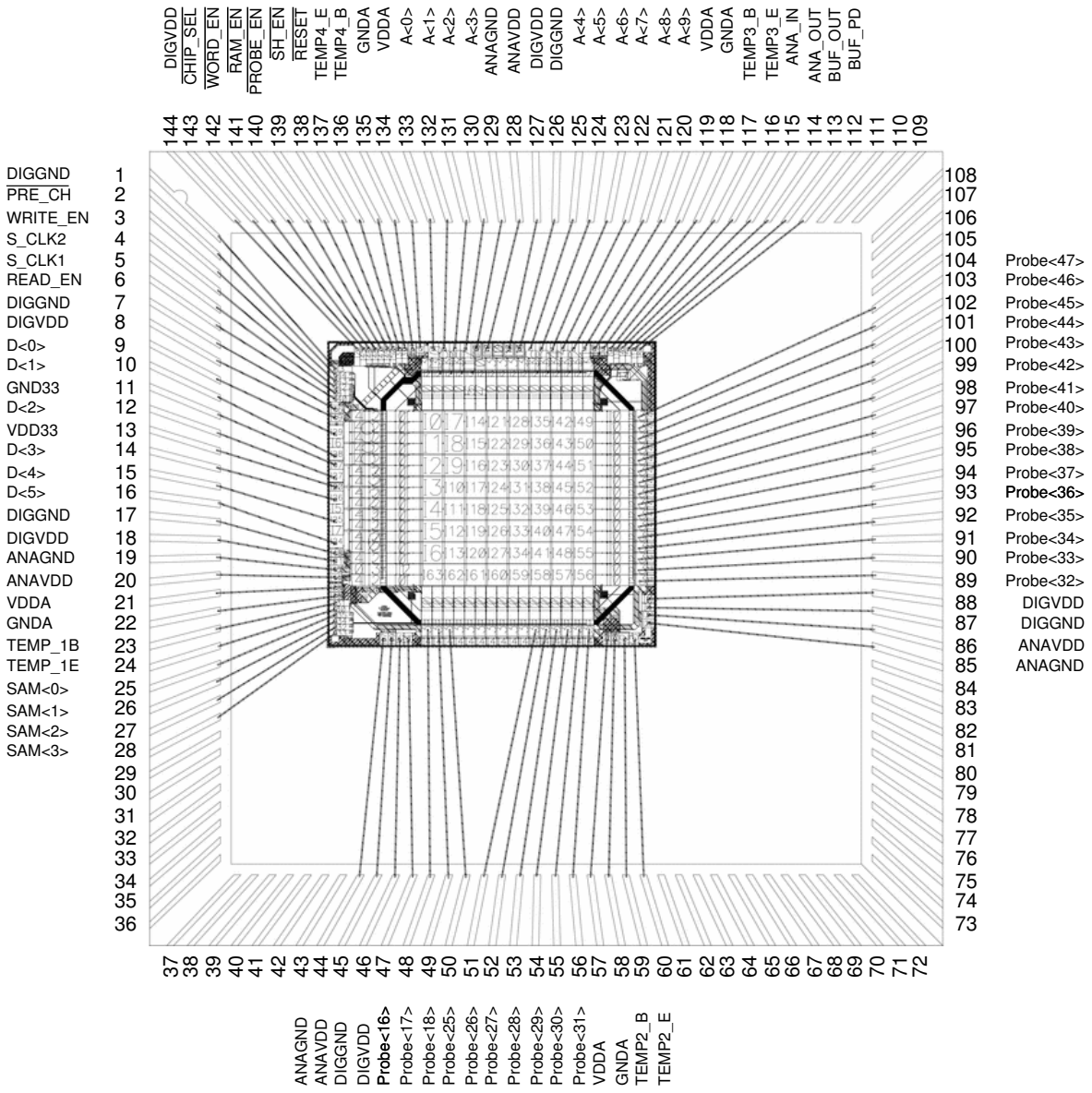


Figure A.2: Bonding diagram for the FPTA chip. All of the probepads that are bonded to in the diagram can in principle be accessed through the Brightwing board by means of jumper settings.

It should be noted, that the probepads have to be treated with utmost care: First, mechanically, during the bonding process itself, because of their small size and vicinity to the rest of the IO-cell,

block capacitors and the power ring. Second, electrically, since they lack any ESD[1] protection. Not all of the bonded FPTA chips have their probepads bonded (in the way suggested by Fig. A.2, least of them *all* of them.

A.4 Chip Carrier Pins

Fig. A.3 displays the assignment of the pin names of the PGA144 carrier to its actual pins accessible on a PCB. The pins are enumerated for the chip carrier seen from the top. Note that the orientation of the carrier is defined by pin 0 in the lower left corner. The unused pins are shaded in a dark gray, while the pins connected to the main pads of the FPTA are shaded in a lighter gray and those connected to probepads are shaded in white. This carrier pin bond pad assignment is based on the bonding diagram in Fig. A.2.

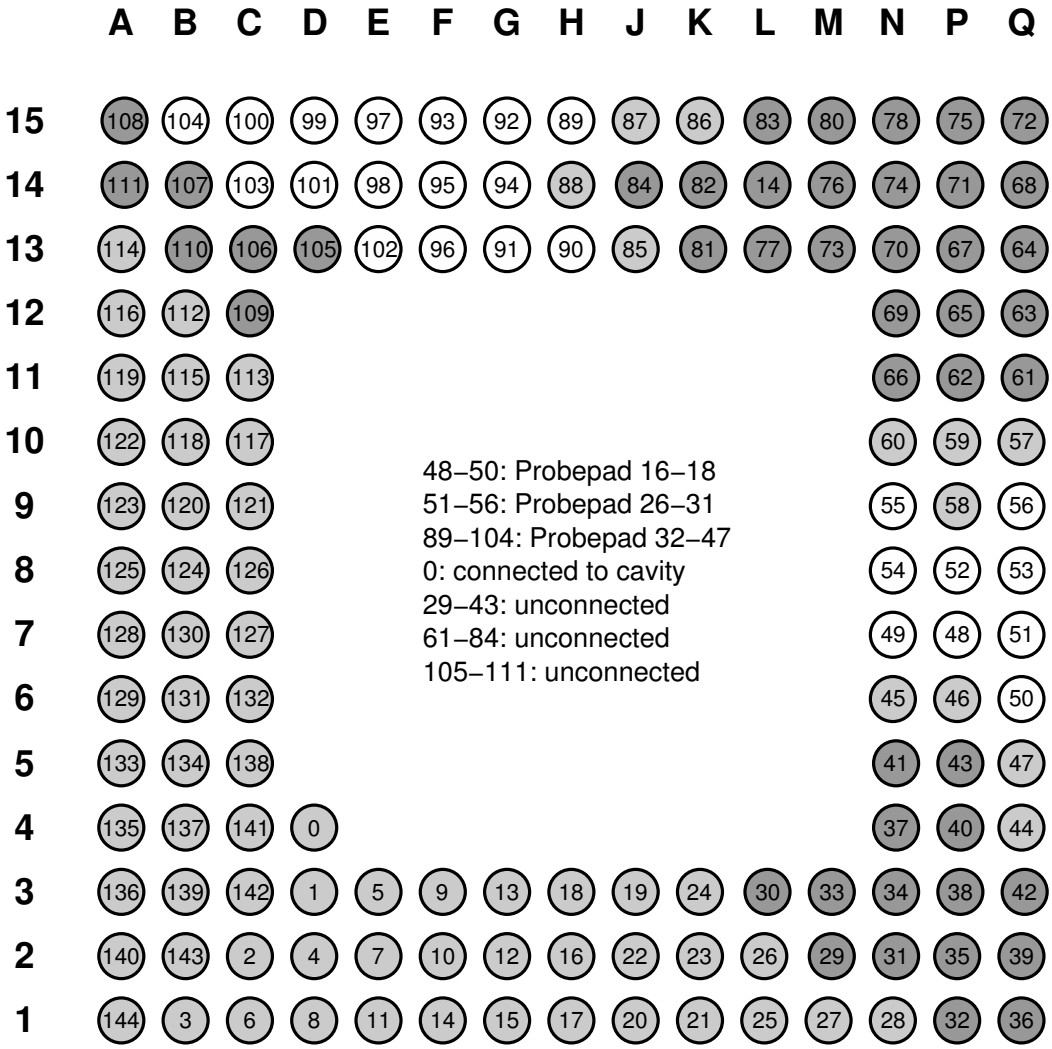


Figure A.3: The correlation between pad numbers and pins of the PGA144 chip carrier. The graphic shows the carrier seen from the top.

[1] Electro-Static Discharge

A.5 Signal Description

Tables A.2, A.3 and A.4 contain a short description of all FPTA signals. The pin names refer to the signal names used within this thesis. The CAD names denote the signal names in the top level of the FPTA's schematic used during the design phase. Note, that the data bus D<0:5> runs on 3.3 V and possesses its own 3.3 V power supply.

pad no.	pin name	CAD name	signal type	description
1, 7, 17, 31, 39, 55	DIG_GND	DigGnd	power	gnd connection for all digital circuitry, that is the digital interface of the IO-cells and the SRAM control.
2	$\overline{\text{PRE\_CH}}$	_PreCh	digital input	for SRAM readout: Pulls the bit lines towards vdd and equalizes them before the bit lines are connected to the respective sense amplifier.
3	WRITE_EN	WE	digital input	for SRAM configuration: Upon WRITE_EN the tri-state buffers driving the bit line pairs are activated. This is used during an SRAM write operation to push the information stored in the according flip-flops the selected SRAM column.
4	S_CLK2	clk2	digital input	for SRAM readout: Clock signal that clocks the power supply of the 96 sense amplifier sensing the bit-line pairs on. The result is latched as long as the S_CLK2 is active.
5	S_CLK1	clk1	digital input	for SRAM readout: Clock signal upon which the bit line pairs are connected to the input gates of the sense amplifier. To be activated after the chosen word line is activated and after before the S_CLK2 signal.
6	READ_EN	ReadEn	digital input	for SRAM readout: Controls the arbitration of the data bus D<0:5>. Within an SRAM read operation READ_EN is activated to read out the information latched in the sense amplifiers. All other operations use D<0:5> as digital input signals. Hence READ_EN must be set to low apart from the SRAM readout.
8, 18, 32, 40, 56, 73	DIG_VDD	DigVdd	power	power supply for all digital circuitry, that is the digital interface of the IO-cells and the SRAM control. Needs 5 V.

Table A.2: Description of the I/O pads (first part).

pad no.	pin name	CAD name	signal type	description
9–10, 12, 14–16	D<0:5>	D<0:5>	digital I/O	tristated data bus: Used as digital output during the readout of the SRAM and as a digital input for SRAM write, IO-cell configuration and for the generation of the SAMPLE signal controling the sample and hold operation of the IO-cells. Externally , that is on the bond pad, D<0:5> only possess a voltage swing of 3.3 V. D<0:5> may sometimes be referred to as DATA<0:5>
11	GND33	Gnd33	power	gnd for the 3.3 V power supply for the data bus D<0:5>.
13	VDD33	Vdd33	power	vdd for the 3.3 V power supply for the data bus D<0:5>.
19, 29, 37, 58	ANA_GND	AnaGnd	power	gnd for the analog power that supplies the all of the analog circuitry except for the programmable transistors themselves. This is the substrate gnd.
20, 30, 38, 57	ANA_VDD	AnaVdd	power	vdd for the analog power.
21, 33, 48, 63	VDDA	vdda	power	power supply for the programmable transistors only Corresponds to the vdd terminal in Fig. 3.2 and 3.25
22, 34, 47, 64	GNDA	gnda	power	power supply for the programmable transistors only Corresponds to the gnd terminal in Fig. 3.2 and 3.25
23	TEMP1_B	Temp1_B	analog I/O	base terminal of the vertical pnp transistor dedicated to temperature sensing. Lower left corner.
24	TEMP1_E	Temp1_E	analog I/O	emitter terminal of the vertical pnp transistor dedicated to temperature sensing. Lower left corner.
25–28	SAM<0:3>	Sam<0:3>	digital input	independent signals that can control the SAMPLE signal in one or more IO-cells that are accordingly configured. D<3:5> can be used in the same way.
35	TEMP2_B	Temp2_B	analog I/O	base terminal of the vertical pnp transistor dedicated to temperature sensing. Lower right corner.
36	TEMP2_E	Temp2_E	analog I/O	emitter terminal of the vertical pnp transistor dedicated to temperature sensing. Lower right corner.
41	BUF_PD	Buf_PD	digital input	power down signal to turn off the global output buffer.
42	BUF_OUT	Buf_Out	analog output	global analog output signal buffered by the global output buffer to drive up to 100 pF and resistive loads down to 1 kΩ and below.

Table A.3: Description of the I/O pads (second part).

pad no.	pin name	CAD name	signal type	description
43	ANA_OUT	ANA_Out	analog output	global analog output before the global output buffer. Maybe already buffered by an IO-cell buffer.
44	ANA_IN	ANA_In	analog input	The one and only analog input that must be multiplexed by the IO-cells. Capacitively loaded by ten(s) of pF. Resistive load depends on application (as digital input for the PTA.
45	TEMP3_E	Temp3_E	analog I/O	emitter terminal of the vertical pnp transistor dedicated to temperature sensing. Upper right corner.
46	TEMP3_B	Temp3_B	analog I/O	base terminal of the vertical pnp transistor dedicated to temperature sensing. Upper right corner.
49–54, 59–62, 65	A<9:0>	A<9:0>	digial input	adress bus used for all sorts of configuration and readout.
65	TEMP4_B	Temp4_B	analog I/O	base terminal of the vertical pnp transistor dedicated to temperature sensing. Upper left corner.
66	TEMP4_E	Temp4_E	analog I/O	emitter terminal of the vertical pnp transistor dedicated to temperature sensing. Upper left corner.
67	$\overline{\text{RESET}}$	_Reset	digital input	global, active-low reset signal. All of the resetable logic included in the IO-cells and the SRAM control can be reset by this signal.
68	$\overline{\text{SH\_EN}}$	_SH_En	digital input	enable signal to activate the chosen address for communicating with the IO-cells.
69	$\overline{\text{PROBE\_EN}}$	_AProbe_En	digital input	enable signal for activating the chosen row, column and terminal in the PTA that is to be probed.
71	$\overline{\text{RAM\_EN}}$	_AR_En	digital input	enable signal to activate the selected PTA row during SRAM write operations.
71	$\overline{\text{WORD\_EN}}$	_AW_En	digital input	enable signal to activate the decoded word line.
72	$\overline{\text{CHIP\_SEL}}$	_Chip_Sel	digital input	chip select signal that allows to control a group of FPTA chips with the same signal. The enable signals control the action to be taken, the $\overline{\text{CHIP\_SEL}}$ decides any of these actions are relevant to the chip.

Table A.4: Description of the I/O pads (third part).

A.6 Signal – Bond Pad – Carrier Pin Assignment

Table A.5 and A.6 relate the FPTA signals to the pad number of the chip, the according pin number of the chip carrier and the respective pin of the Brightwing connector, that is, the connector of the add-on board hosting the chip and some additional circuitry, which is plugged onto the Darkwing PCI card.

pad no.	pin name	CAD name	pin no. for PGA144	Brightwing connector
1, 7, 17, 31, 39, 55	DIG_GND	DigGnd	1, 7, 17, 46, 87, 126	A1,A16,C1,C16
2	$\overline{\text{PRE\_CH}}$	_PreCh	2	A5
3	WRITE_EN	WE	3	C9
4	S_CLK2	clk2	4	C7
5	S_CLK1	clk1	5	B7
6	READ_EN	ReadEn	6	C8
8, 18, 32, 40, 56, 73	DIG_VDD	DigVdd	8, 18, 47, 88, 127, 144	B16
9	D<0>	D<0>	9	C4
10	D<1>	D<1>	10	C6
11	GND33	Gnd33	11	A1,A16,C1,C16
12	D<2>	D<2>	12	C5
13	VDD33	Vdd33	13	B15
14	D<3>	D<3>	14	B4
15	D<4>	D<4>	15	A4
16	D<5>	D<5>	16	B5
19, 29, 37, 58	ANA_GND	AnaGnd	19, 44, 85, 129	A1,A16,C1,C16
20, 30, 38, 57	ANA_VDD	AnaVdd	20, 45, 86, 128	B16
21, 33, 48, 63	VDDA	vdda	21, 57, 119, 134	B16
22, 34, 47, 64	GNDA	gnda	22, 58, 118, 135	A1,A16,C1,C16
23	TEMP1_B	Temp1_B	23	–
24	TEMP1_E	Temp1_E	24	–
25	SAM<0>	Sam<0>	25	B8
26	SAM<1>	Sam<1>	26	B6
27	SAM<2>	Sam<2>	27	A7
28	SAM<3>	Sam<3>	28	A6
35	TEMP2_B	Temp2_B	59	–
36	TEMP2_E	Temp2_E	60	–

Table A.5: Description of the I/O pads (first part).

pad no.	pin name	CAD name	pin no. for PGA144	Brightwing connector
41	BUF_PD	Buf_PD	112	B14
42	BUF_OUT	Buf_Out	113	B1
43	ANA_OUT	ANA_Out	114	B2
44	ANA_IN	ANA_In	115	C2
45	TEMP3_E	Temp3_E	116	–
46	TEMP3_B	Temp3_B	117	–
49	A<9>	A<9>	120	B13
50	A<8>	A<8>	121	C13
51	A<7>	A<7>	122	A14
52	A<6>	A<6>	123	A13
53	A<5>	A<5>	124	C12
54	A<4>	A<4>	125	B12
59	A<3>	A<3>	130	A12
60	A<2>	A<2>	131	B11
61	A<1>	A<1>	132	C11
62	A<0>	A<0>	133	C10
65	TEMP4_B	Temp4_B	136	–
66	TEMP4_E	Temp4_E	137	–
67	$\overline{\text{RESET}}$	_Reset	138	A15
68	$\overline{\text{SH\_EN}}$	_SH_En	139	A9
69	$\overline{\text{PROBE\_EN}}$	_AProbe_En	140	A10
71	$\overline{\text{RAM\_EN}}$	_AR_En	141	B10
71	$\overline{\text{WORD\_EN}}$	_AW_En	142	A8
72	$\overline{\text{CHIP\_SEL}}$	_Chip_Sel	143	B9

Table A.6: Description of the I/O pads (second part).

Appendix B

Rail-to-Rail Operational Amplifier: Simulation Results

This appendix contains a more detailed selection of simulation results attained for the three different rail-to-rail amplifier implementations utilized in the FPTA design than is presented in section 3.7.4 on page 95. All of the simulations are based on a circuit temperature of $T = 27\,^{\circ}\mathrm{C}$ and typcial mean process parameters. The different circuit specifications, the type of analysis as well as the op-amp configuration imposed by the respective testbench are summarized in Table B.1. Neither is the selec-

Property	Configuration	Analysis	Description/Comment
V_{OS}	buffer	DC	offset voltage
I_{tot}, I_{M38}	buffer	DC	quiescent current
A_{OL}	open loop	AC at 1 Hz	dc open loop gain
UGB, GBP	open loop	AC	
magn. response	open loop	AC	
phase response	open loop	AC	
PM	open loop	AC	
SR	buffer	transient	for 10 to 90 % of output step
			$V_{\mathrm{start}}, V_{\mathrm{stop}} \in \{0\,\mathrm{V}, 0.5\,\mathrm{V}, 1\,\mathrm{V}, 2.5\,\mathrm{V}, 4\,\mathrm{V}, 4.5\,\mathrm{V}, 5\,\mathrm{V}\}$
CMRR	buffer + V_{CM}	AC	
THD	buffer	transient	sinusoidal input signal, output attained from Fourier transformation
T_{settle}	buffer	transient	precision relative to output step: 0.1 or 1%.

Table B.1: Description of the I/O pads (first part).

tion of specifications complete, nor is their simulation sufficient. In particular V_{OS}, CMRR and THD necessitate Monte-Carlo simulations including device mismatch. Nonetheless, the presented simulation results do characterize the three amplifiers well enough to allow for some performance estimates of the analog IO signal path on the FPTA chip.

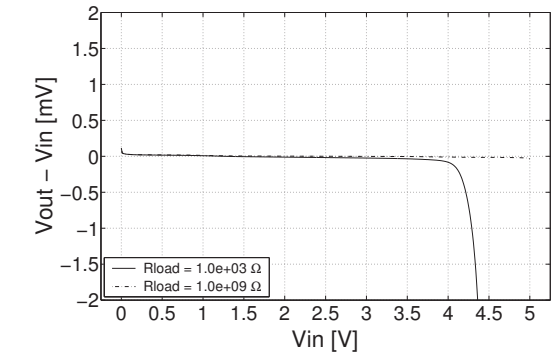
Output Loads. Simulation results are presented for different resistive and capacitive loads. The different capacitive load are motivated in section 3.7.4. For each of the three amplifier designs two different load resistors are used: One almost infinite load resistance $R_{\mathrm{load}} = 1\,\mathrm{G\Omega}$ and one small load resistance that amounts to $10\,\mathrm{k\Omega}$, $1\,\mathrm{k\Omega}$ and $100\,\Omega$ for the output, IO-cell and cell buffer, respectively.

The small load resistors for the IO-cell and the global output buffer turned out to be too small, as can be seen by the dc offset curves of Fig. B.1(a) and B.6(a). It should be noted though, that in the normal operation of the evolution system only the global output buffer needs to drive a resistive load, which is considerably larger than the $100\,\Omega$ tested here. However, the performance for the minimum feasible load resistors should be found between the two extreme situations simulated.

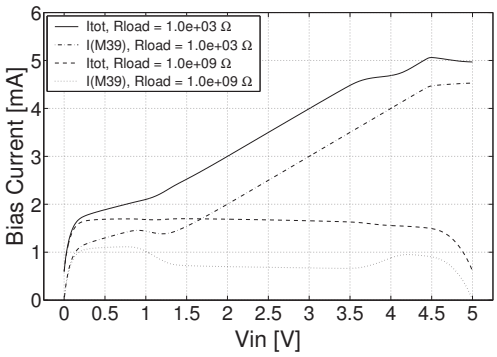
Overview. The simulation results of the three different rail-to-rail op-amps are presented in the same order as used in section 3.7.4, that is, IO-cell buffer, global output buffer and cell buffer. In principle, the same collection of plots account for the performance of the three amplifiers. However, the plots depicting the PM, the frequency spectra and the settling time tables are adapted to the most important load conditions.

Special Representations. The slew rate and the settling times are extracted from transient analyses featuring positive and negative voltage steps between the following seven voltages: 0 V, 0.5 V, 1 V, 2.5 V, 4 V and 5 V, where the first and the last value are identical to gnd and vdd. Voltage steps therefore start at a voltage V_{start} and end at a target voltage V_{stop}. As the different input values are naturally described by a two-dimensional matrix or plane, a suited representation is difficult to obtain. In case of the slew rate, the data of each V_{stop} is gathered in one curve plotted against V_{start}. as can e.g. be seen from Fig. B.2(a). The slew rate curves for rising and falling steps are plotted into different graphs. The slew rate for $V_{\text{start}} = V_{\text{stop}}$ is set to 0. Accordingly, the zero crossing of each curve marks its stop voltage V_{stop}. In contrast, the settling time is illustrated by means of 3-dimensional bar plots to elucidate the dependence on V_{start} and V_{stop}. As the settling times are found to be amongst the most important figures of merit and given that the 3-dimensional bar plots are too imprecise to read off concrete numbers, some of the settling time data is also compiled into Tables. Again, in case of the 3-d graphs, settling times for $V_{\text{start}} = V_{\text{stop}}$ are set to zero.

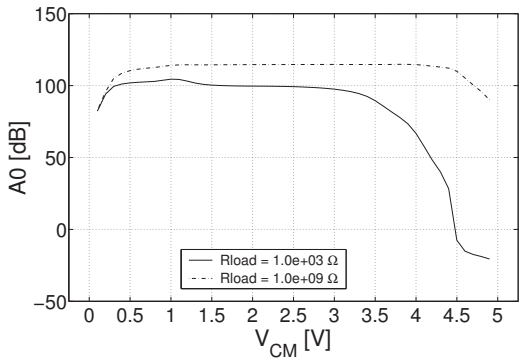
B.1 Simulation of the Rail-to-Rail IO-cell Buffer



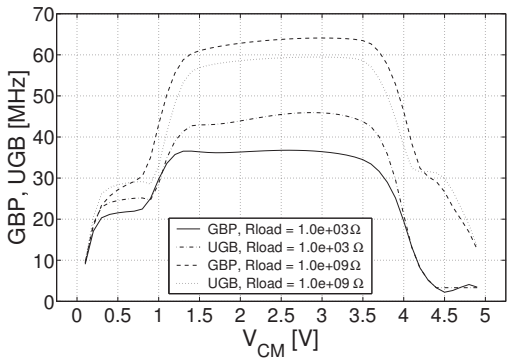
(a) DC offset in unity gain buffer configuration



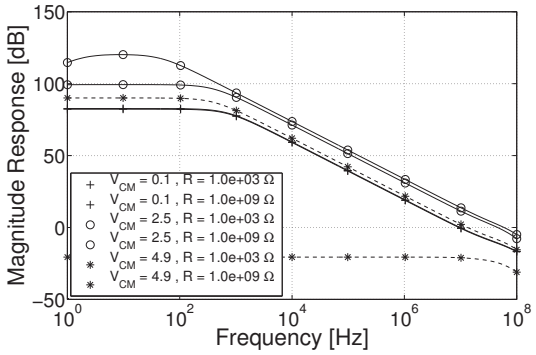
(b) DC currents in unity gain configuration



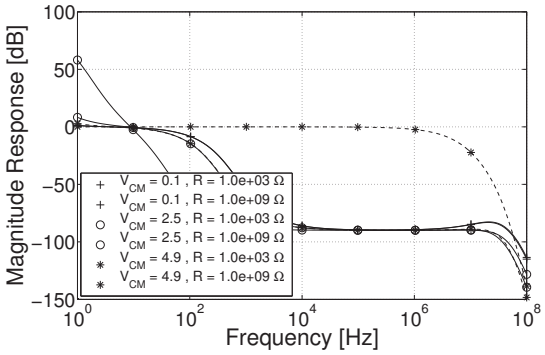
(c) A_{OL} in DC case



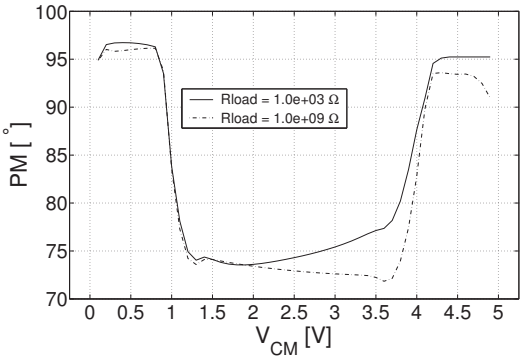
(d) UGB and GBP



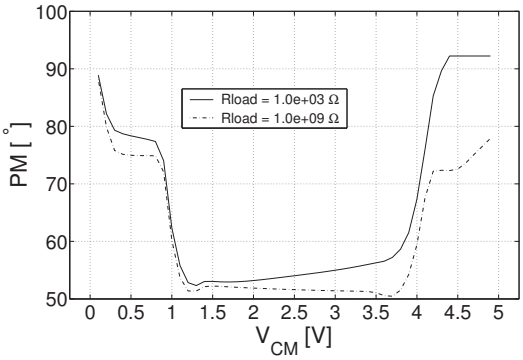
(e) magnitude response



(f) phase response

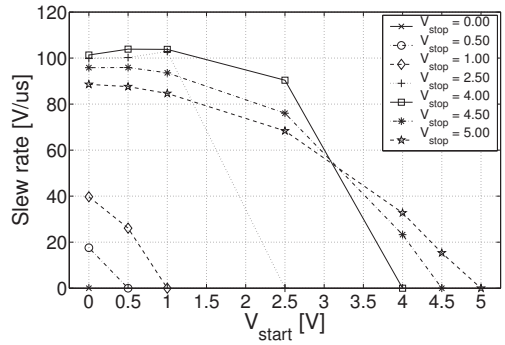


(g) PM, $C_{\mathrm{load}} = 10\,\mathrm{pF}$

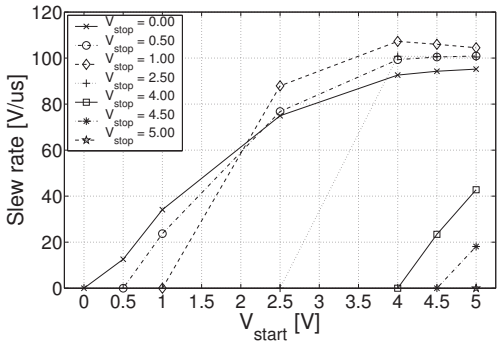


(h) PM, $C_{\mathrm{load}} = 30\,\mathrm{pF}$

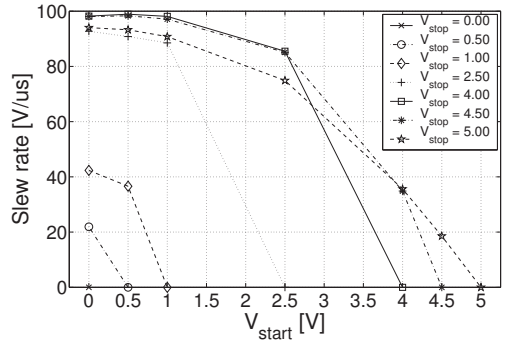
Figure B.1: DC and AC simulation results for the rail-to-rail operational amplifier used twice in the IO-cells.



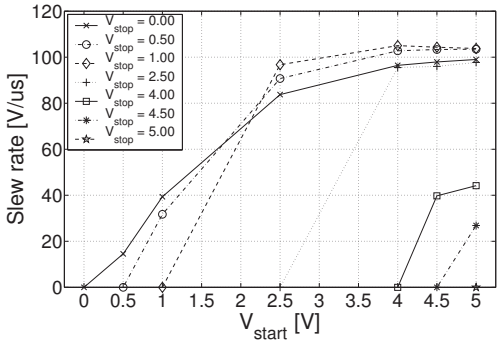
(a) SR rising edge, $C_{\text{load}} = 10\,\text{pF}$



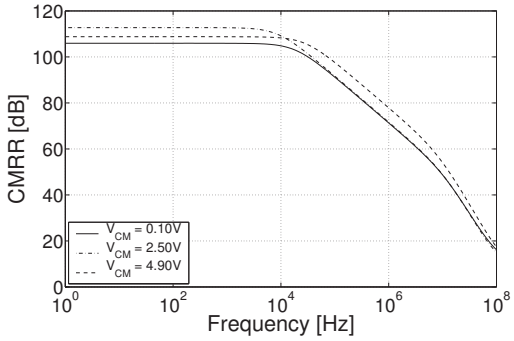
(b) SR falling edge, $C_{\text{load}} = 10\,\text{pF}$



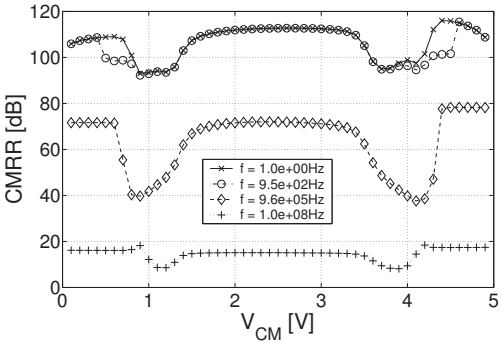
(c) SR rising edge, $C_{\text{load}} = 30\,\text{pF}$



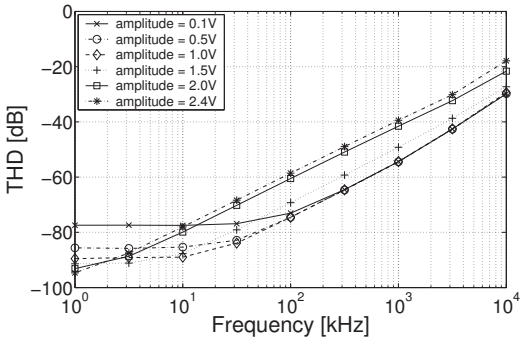
(d) SR falling edge, $C_{\text{load}} = 30\,\text{pF}$



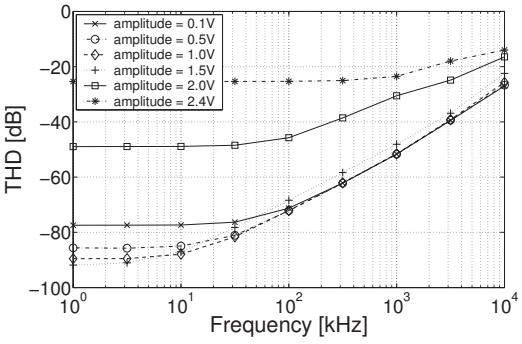
(e) CMRR vs frequency



(f) CMRR vs common mode voltage

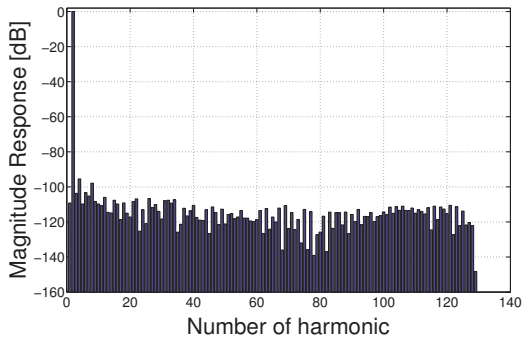


(g) THD, $R_{\text{load}} = 1\,\text{G}\Omega$

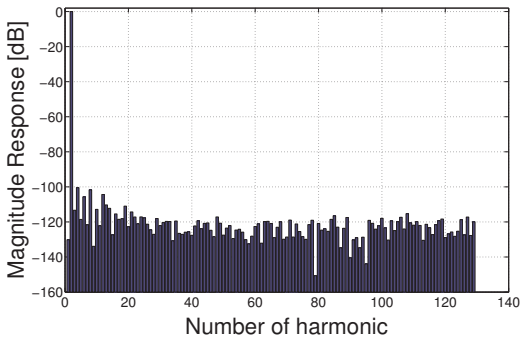


(h) THD, $R_{\text{load}} = 1\,\text{k}\Omega$

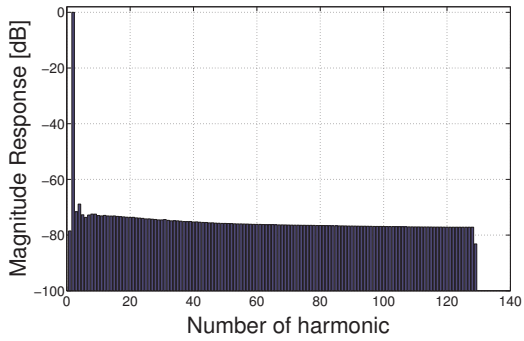
Figure B.2: CMRR ((a) and (b)), THD ((c) and (d)) and SR for the rising ((e) and (g) and the falling edge ((f) and (h) for the op-amp used twice in the IO-cells. In (a) to (d) the stop voltage of each curve can be identified by its zero-crossing.



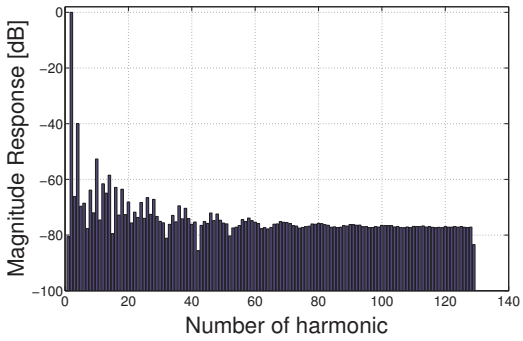
(a) amplitude $= 1.0\,\mathrm{V},\ f = 1\,\mathrm{kHz}$



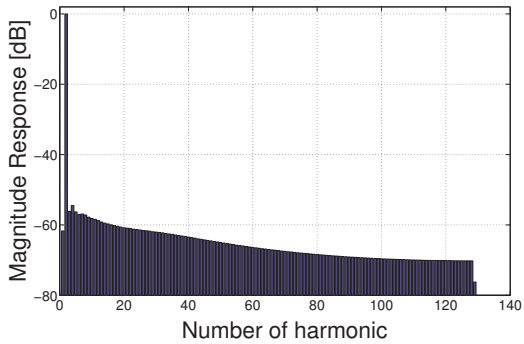
(b) amplitude $= 2.4\,\mathrm{V},\ f = 1\,\mathrm{kHz}$



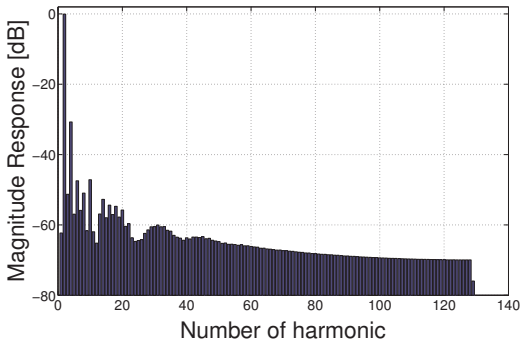
(c) amplitude $= 1.0\,\mathrm{V},\ f = 1\,\mathrm{MHz}$



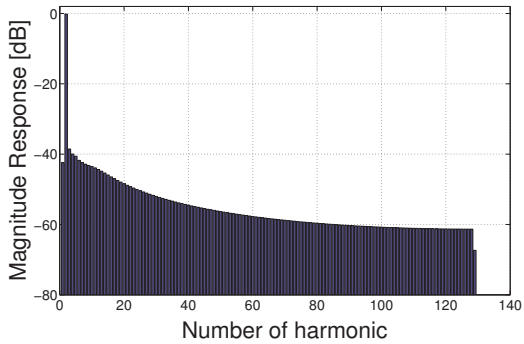
(d) amplitude $= 2.4\,\mathrm{V},\ f = 1\,\mathrm{MHz}$



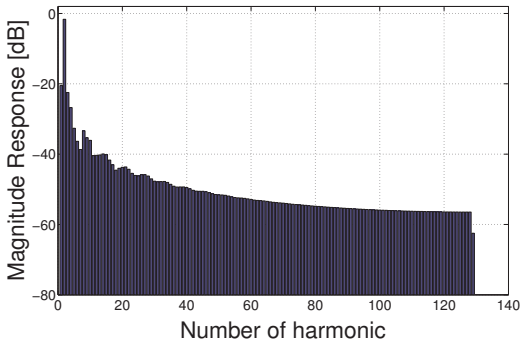
(e) amplitude $= 1.0\,\mathrm{V},\ f = 3.2\,\mathrm{MHz}$



(f) amplitude $= 2.4\,\mathrm{V},\ f = 3.2\,\mathrm{MHz}$

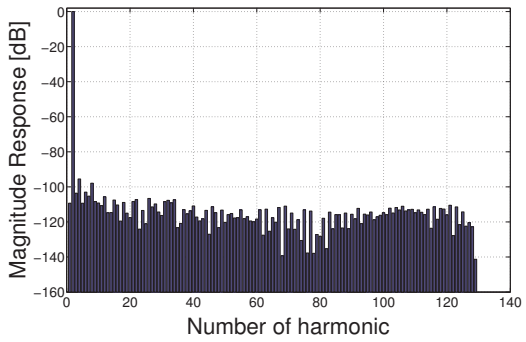


(g) amplitude $= 1.0\,\mathrm{V},\ f = 10\,\mathrm{MHz}$

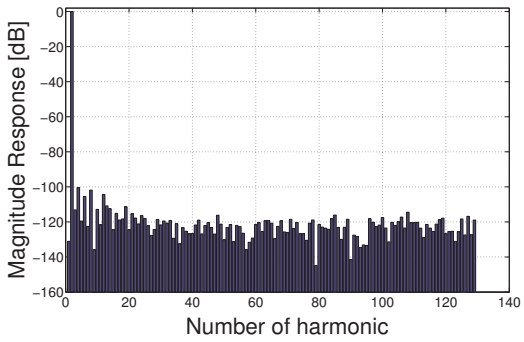


(h) amplitude $= 2.4\,\mathrm{V},\ f = 10\,\mathrm{MHz}$

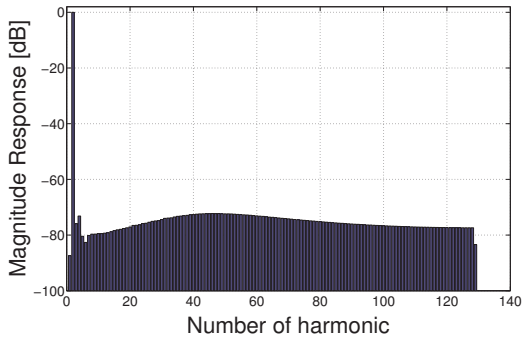
Figure B.3: Frequency spectra of the op amp used in the IO-cells for different input amplitudes and fundamental frequencies f. For each plot, the fundamental, the dc component and the first 126 harmonics are displayed. The op-amp was configured as a unity gain buffer. $R_{\mathrm{load}} = 1\,\mathrm{G\Omega}$, $C_{\mathrm{load}} = 10\,\mathrm{pF}$.



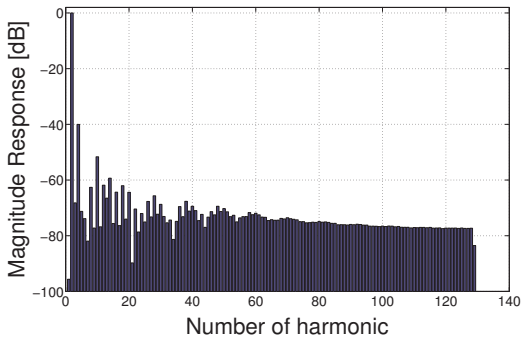
(a) amplitude $= 1.0\,\mathrm{V}$, $f = 1\,\mathrm{kHz}$



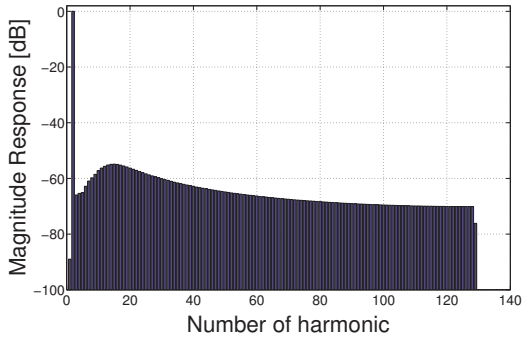
(b) amplitude $= 2.4\,\mathrm{V}$, $f = 1\,\mathrm{kHz}$



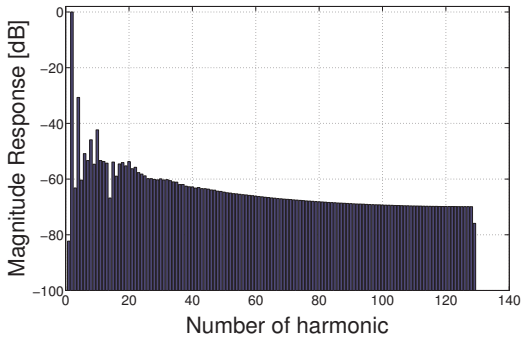
(c) amplitude $= 1.0\,\mathrm{V}$, $f = 1\,\mathrm{MHz}$



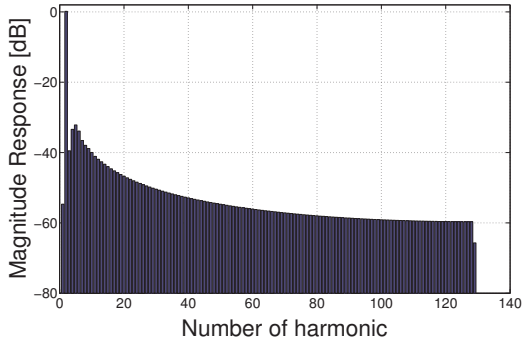
(d) amplitude $= 2.4\,\mathrm{V}$, $f = 1\,\mathrm{MHz}$



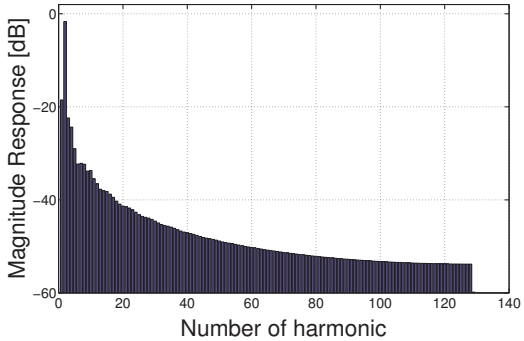
(e) amplitude $= 1.0\,\mathrm{V}$, $f = 3.2\,\mathrm{MHz}$



(f) amplitude $= 2.4\,\mathrm{V}$, $f = 3.2\,\mathrm{MHz}$

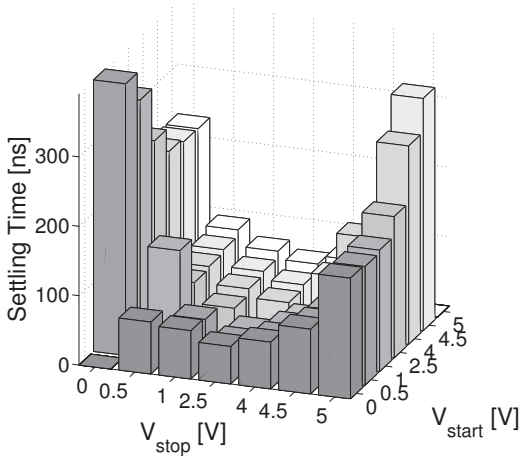


(g) amplitude $= 1.0\,\mathrm{V}$, $f = 10\,\mathrm{MHz}$

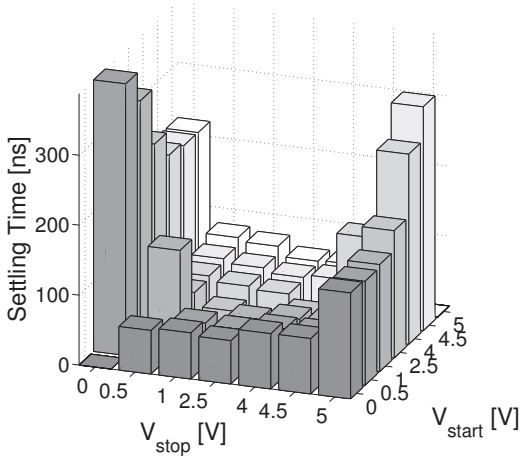


(h) amplitude $= 2.4\,\mathrm{V}$, $f = 10\,\mathrm{MHz}$

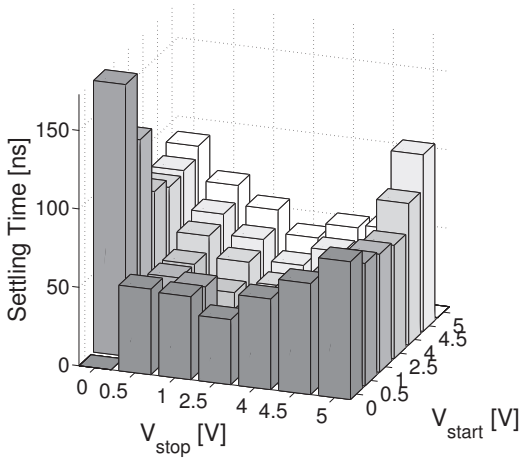
Figure B.4: Frequency spectra of the op amp used in the IO-cells for different input amplitudes and fundamental frequencies f. For each plot, the fundamental, the dc component and the first 126 harmonics are displayed. The op-amp was configured as a unity gain buffer. $R_{\mathrm{load}} = 1\,\mathrm{G\Omega}$, $C_{\mathrm{load}} = 30\,\mathrm{pF}$.



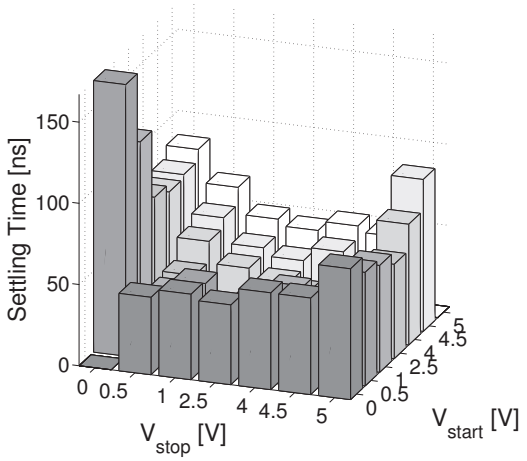
(a) settling time, precision = 0.1%, $C_{\text{load}} = 10\,\text{pF}$



(b) settling time, precision = 0.1%, $C_{\text{load}} = 30\,\text{pF}$



(c) settling time, precision = 1%, $C_{\text{load}} = 10\,\text{pF}$



(d) settling time, precision = 1%, $C_{\text{load}} = 30\,\text{pF}$

Figure B.5: Settling time for the op-amp used in the IO-cells for different precisions (top/bottom) and different load capacitors (left/right).

V_{stop} \ V_{start}	0.0	0.5	1.0	2.5	4.0	4.5	5.0
0.0	–	391	348	270	235	229	228
0.5	75	–	139	73	76	81	90
1.0	70	66	–	43	51	57	66
2.5	54	42	37	–	39	44	55
4.0	67	56	51	42	–	63	67
4.5	93	84	80	80	147	–	74
5.0	174	171	175	203	286	334	–

Table B.2: Settling time in ns for the op-amp used for in the IO-cells: Precision $= 0.1\%$, $R_{\text{load}} = 1\,\text{G}\Omega$, $C_{\text{load}} = 10\,\text{pF}$.

V_{stop} \ V_{start}	0.0	0.5	1.0	2.5	4.0	4.5	5.0
0.0	–	388	344	263	228	222	221
0.5	62	–	138	58	62	67	78
1.0	66	60	–	38	55	61	73
2.5	62	58	53	–	51	55	58
4.0	79	67	61	49	–	59	60
4.5	79	68	63	67	146	–	61
5.0	151	148	153	182	272	320	–

Table B.3: Settling time in ns for the op-amp used for in the IO-cells: Precision $= 0.1\%$, $R_{load} = 1\,G\Omega$, $C_{load} = 30\,pF$.

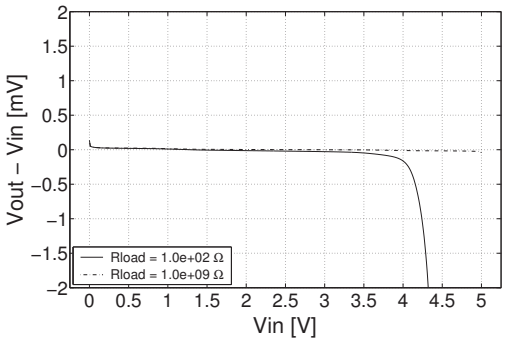
V_{stop} \ V_{start}	0.0	0.5	1.0	2.5	4.0	4.5	5.0
0.0	–	394	353	279	248	255	257
0.5	82	–	140	82	90	108	112
1.0	77	72	–	53	65	83	87
2.5	72	60	54	–	46	64	67
4.0	93	81	75	62	–	91	97
4.5	207	197	194	189	207	–	132
5.0	114	102	103	93	128	50	–

Table B.4: Settling time in ns for the op-amp used for in the IO-cells: Precision $= 0.1\%$, $R_{load} = 1\,k\Omega$, $C_{load} = 10\,pF$.

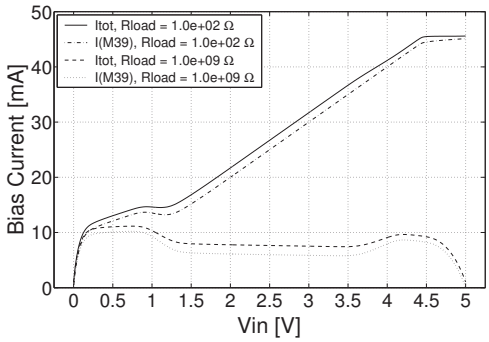
V_{stop} \ V_{start}	0.0	0.5	1.0	2.5	4.0	4.5	5.0
0.0	–	392	350	273	242	248	250
0.5	70	–	139	69	78	96	100
1.0	70	66	–	44	65	86	90
2.5	72	59	54	–	67	78	82
4.0	89	78	73	57	–	86	92
4.5	198	189	185	181	202	–	131
5.0	113	102	102	92	129	52	–

Table B.5: Settling time in ns for the op-amp used for in the IO-cells: Precision $= 0.1\%$, $R_{load} = 1\,k\Omega$, $C_{load} = 30\,pF$.

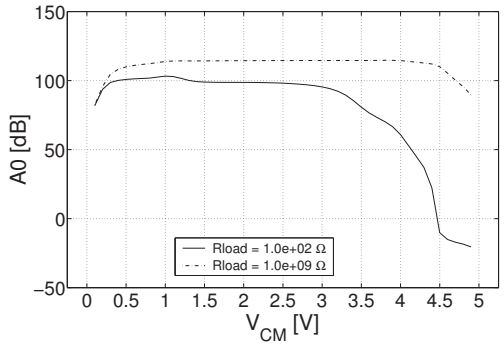
B.2 Simulation of the Rail-to-Rail Output Buffer



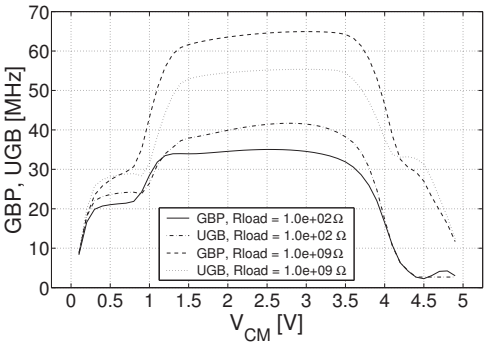
(a) DC offset in unity gain buffer configuration



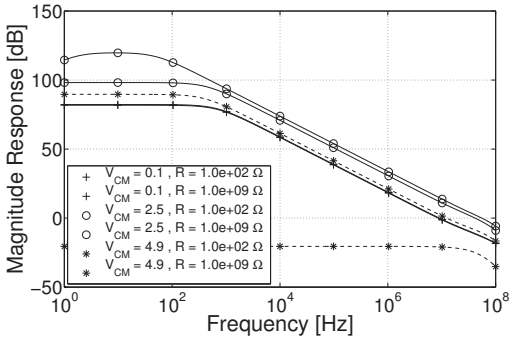
(b) DC currents in unity gain configuration



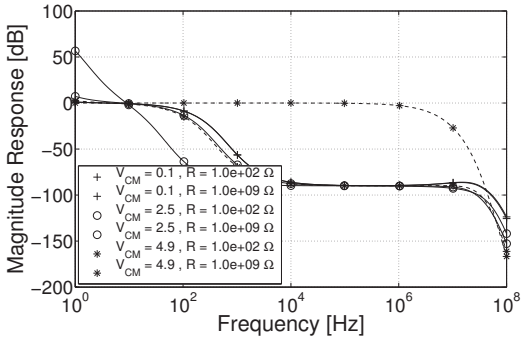
(c) A_{OL} in DC case



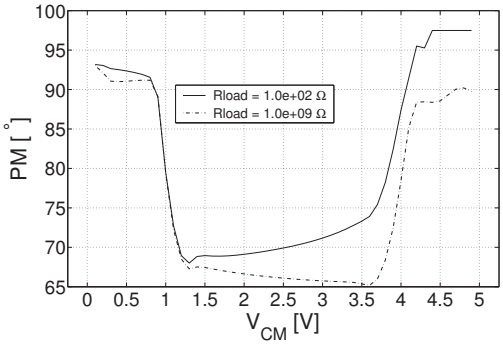
(d) UGB and GBP



(e) magnitude response

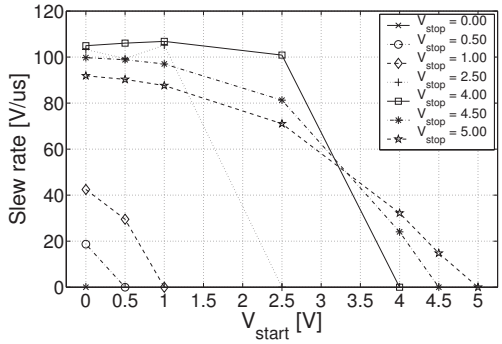


(f) phase response

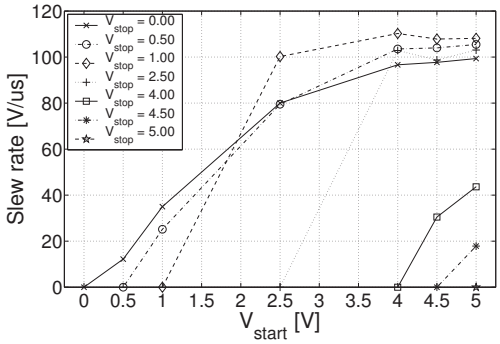


(g) PM

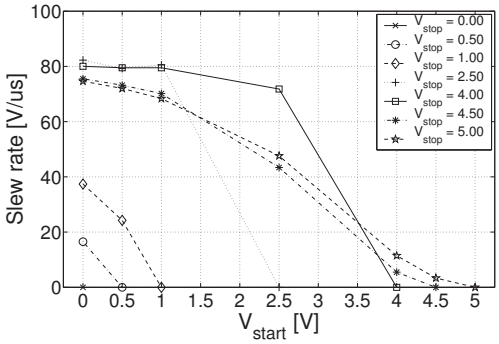
Figure B.6: DC and AC simulation results for the rail-to-rail operational amplifier used as the global output buffer.



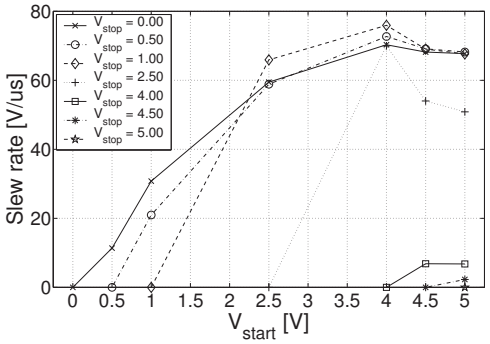
(a) SR rising edge, $R_{\text{load}} = 1\,\text{G}\Omega$



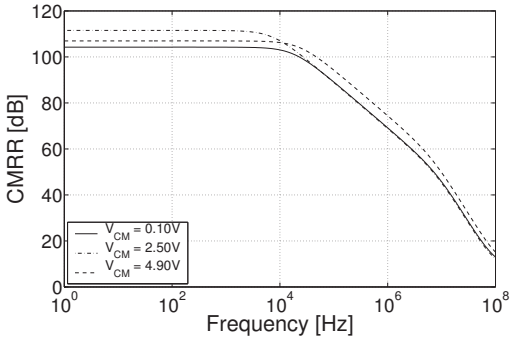
(b) SR falling edge, $R_{\text{load}} = 1\,\text{G}\Omega$



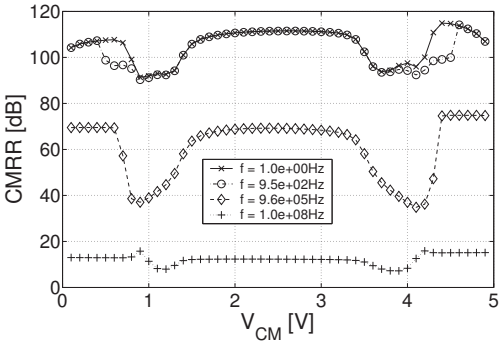
(c) SR rising edge, $R_{\text{load}} = 1\,\text{k}\Omega$



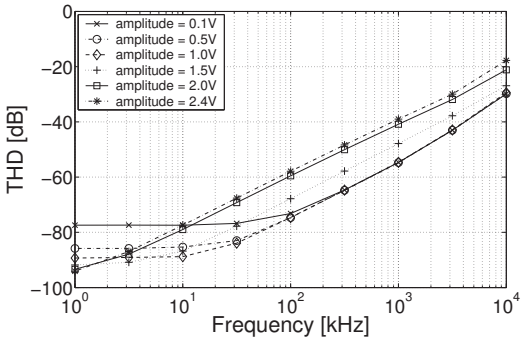
(d) SR falling edge, $R_{\text{load}} = 1\,\text{k}\Omega$



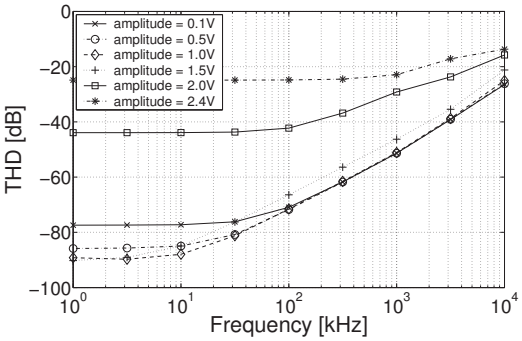
(e) CMRR vs frequency



(f) CMRR vs common mode voltage

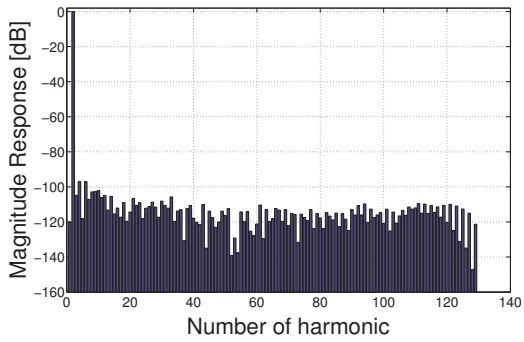


(g) THD, $R_{\text{load}} = 1\,\text{G}\Omega$

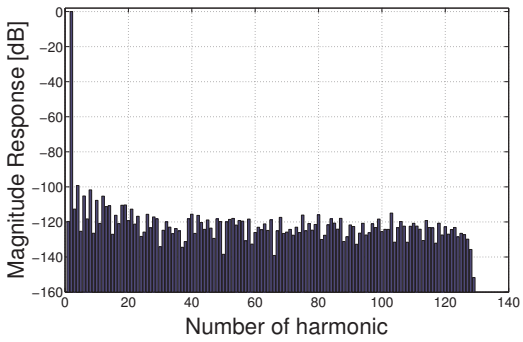


(h) THD, $R_{\text{load}} = 1\,\Omega$

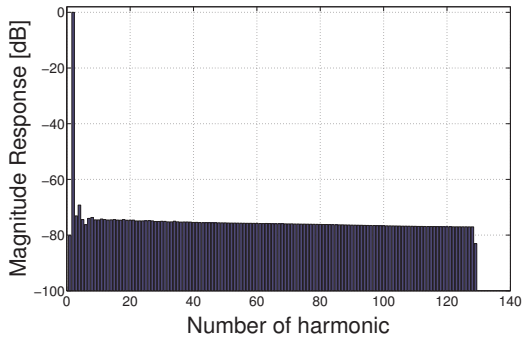
Figure B.7: CMRR ((a) and (b)), THD ((c) and (d)) and SR for the rising ((e) and (g) and the falling edge ((f) and (h) for the global output buffer. In (a) to (d) the stop voltage of each curve can be identified by its zero-crossing.



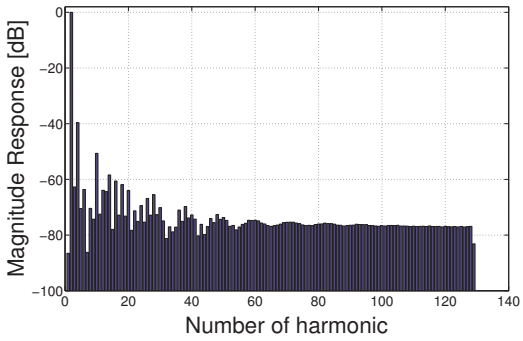
(a) amplitude $= 1.0\,\mathrm{V},\ f = 1\,\mathrm{kHz}$



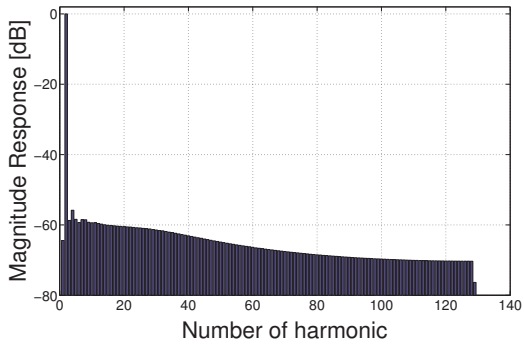
(b) amplitude $= 2.4\,\mathrm{V},\ f = 1\,\mathrm{kHz}$



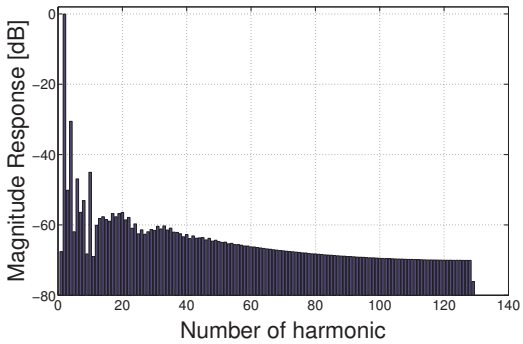
(c) amplitude $= 1.0\,\mathrm{V},\ f = 1\,\mathrm{MHz}$



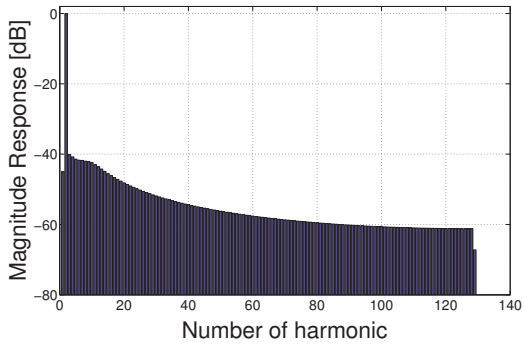
(d) amplitude $= 2.4\,\mathrm{V},\ f = 1\,\mathrm{MHz}$



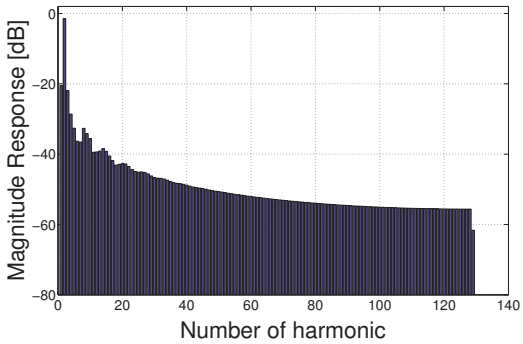
(e) amplitude $= 1.0\,\mathrm{V},\ f = 3.2\,\mathrm{MHz}$



(f) amplitude $= 2.4\,\mathrm{V},\ f = 3.2\,\mathrm{MHz}$

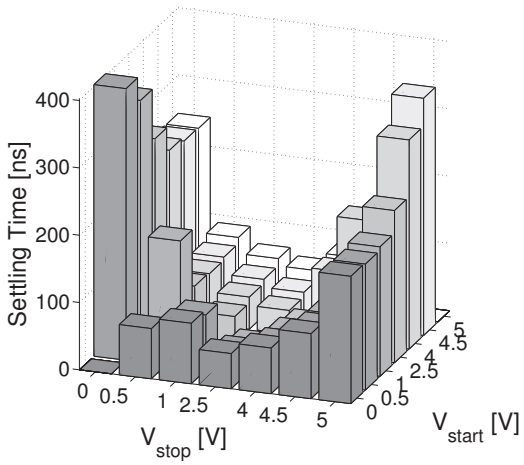


(g) amplitude $= 1.0\,\mathrm{V},\ f = 10\,\mathrm{MHz}$

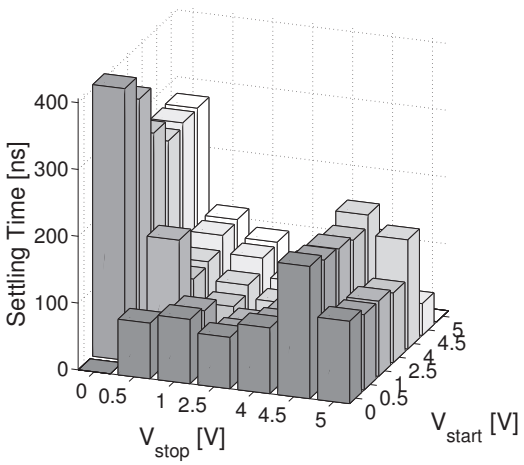


(h) amplitude $= 2.4\,\mathrm{V},\ f = 10\,\mathrm{MHz}$

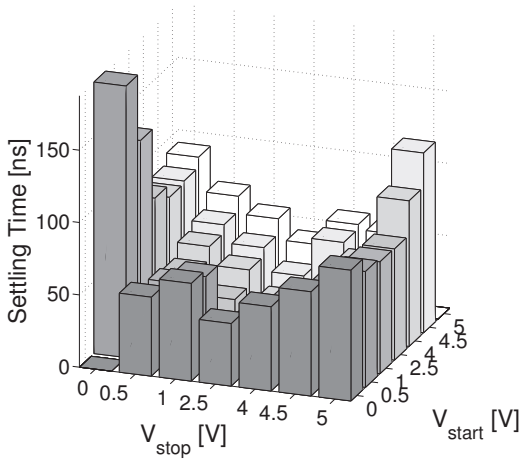
Figure B.8: Frequency spectra of the main output buffer for different input amplitudes and fundamental frequencies f. For each plot, the fundamental, the dc component and the first 126 harmonics are displayed. The op-amp was configured as a unity gain buffer. $R_{\mathrm{load}} = 1\,\mathrm{G\Omega}$.



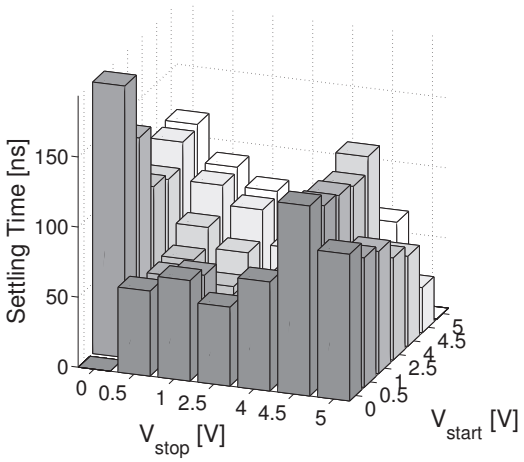
(a) settling time, precision = 0.1%, $R_{\text{load}} = 1\,\text{G}\Omega$



(b) settling time, precision = 0.1%, $R_{\text{load}} = 1\,\text{k}\Omega$



(c) settling time, precision = 1%, $R_{\text{load}} = 1\,\text{G}\Omega$



(d) settling time, precision = 1%, $R_{\text{load}} = 1\,\text{k}\Omega$

Figure B.9: Settling time for the main output buffer for different precisions (top/bottom) and different load resistors (left/right).

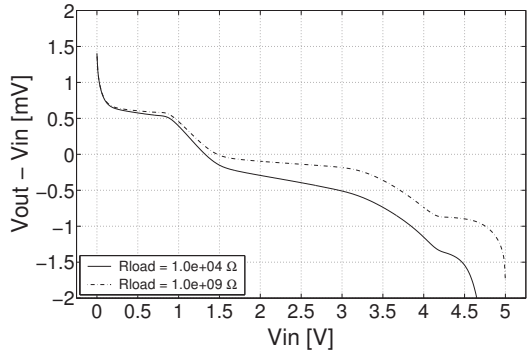
V_{stop} \ V_{start}	0.0	0.5	1.0	2.5	4.0	4.5	5.0
0.0	–	403	366	288	251	244	243
0.5	75	–	164	77	75	79	88
1.0	90	82	–	39	48	54	64
2.5	52	37	26	–	35	42	53
4.0	68	56	51	43	–	79	80
4.5	96	87	84	95	183	–	75
5.0	189	187	192	226	310	352	–

Table B.6: Settling time in ns for the output buffer: Precision $= 0.1\%$, $R_{load} = 1\,\mathrm{G}\Omega$, $C_{load} = 100\,\mathrm{pF}$.

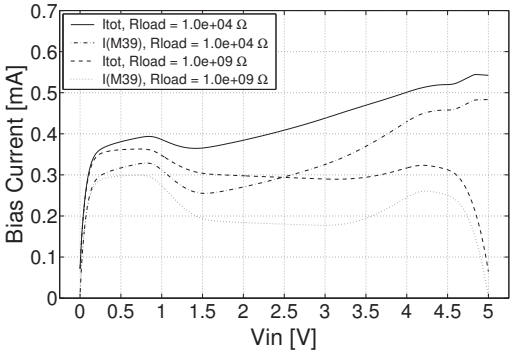
V_{stop} \ V_{start}	0.0	0.5	1.0	2.5	4.0	4.5	5.0
0.0	–	407	370	299	267	274	275
0.5	83	–	167	88	93	113	116
1.0	96	89	–	54	66	86	89
2.5	77	64	57	–	46	57	61
4.0	98	86	80	66	–	103	109
4.5	198	187	183	176	193	–	109
5.0	123	113	112	104	163	47	–

Table B.7: Settling time in ns for the output buffer: Precision $= 0.1\%$, $R_{load} = 100\,\Omega$, $C_{load} = 100\,\mathrm{pF}$.

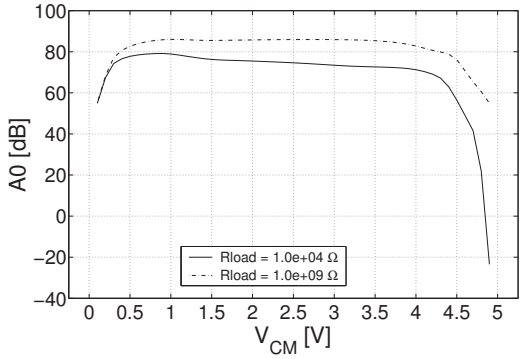
B.3 Simulation of the Rail-to-Rail Cell Buffer



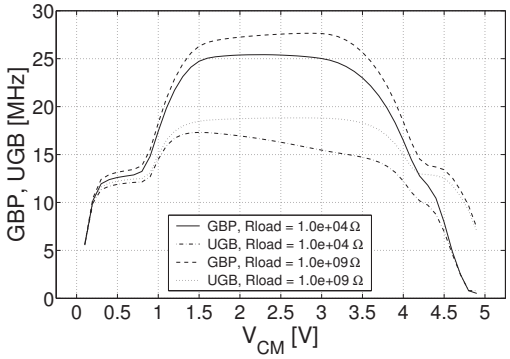
(a) DC offset in unity gain buffer configuration



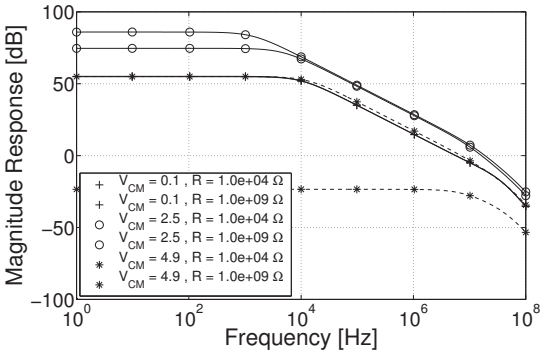
(b) DC currents in unity gain configuration



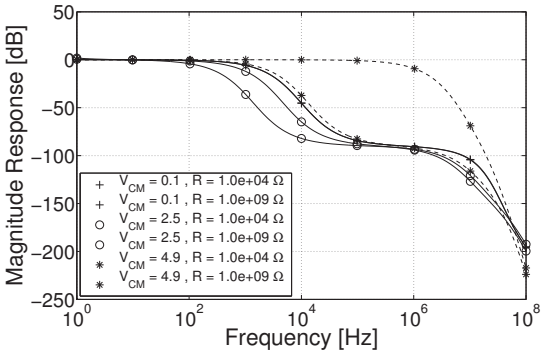
(c) A_{OL} in DC case



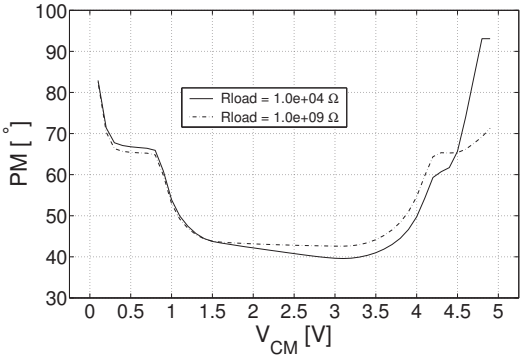
(d) UGB and GBP



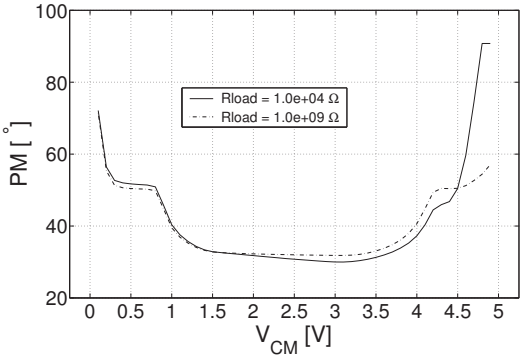
(e) magnitude response



(f) phase response

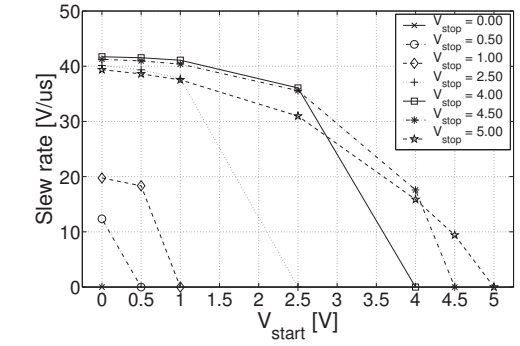


(g) PM, $C_{\mathrm{load}} = 20\,\mathrm{pF}$

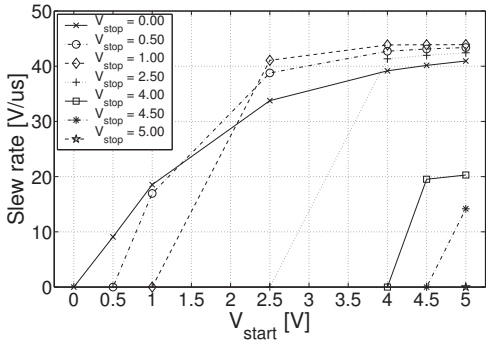


(h) PM, $C_{\mathrm{load}} = 40\,\mathrm{pF}$

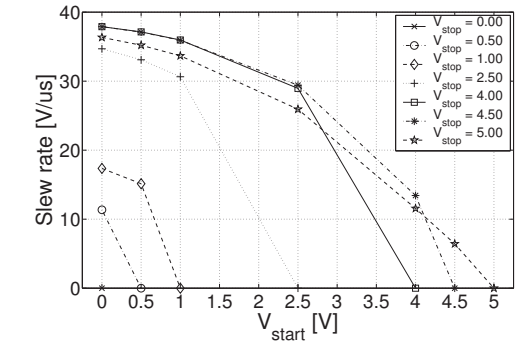
Figure B.10: DC and AC simulation results for the rail-to-rail operational amplifier used for the inner-cell probing.



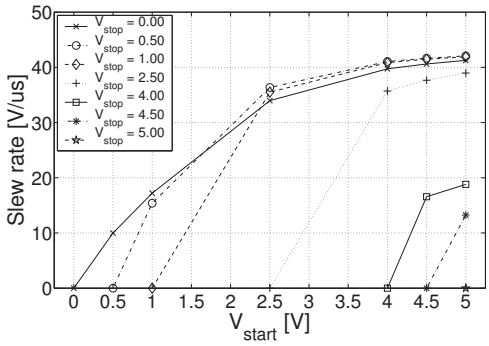
(a) SR rising edge, $C_{\text{load}} = 20\,\text{pF}$



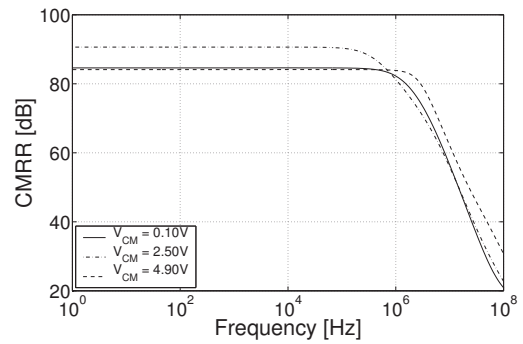
(b) SR falling edge, $C_{\text{load}} = 20\,\text{pF}$



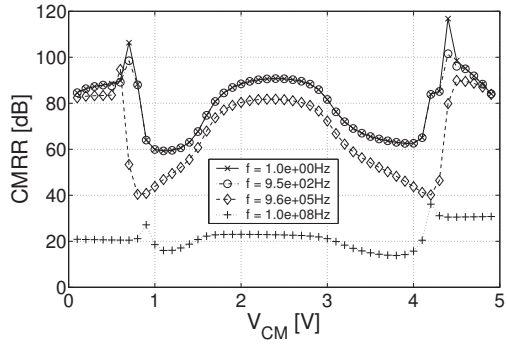
(c) SR rising edge, $C_{\text{load}} = 40\,\text{pF}$



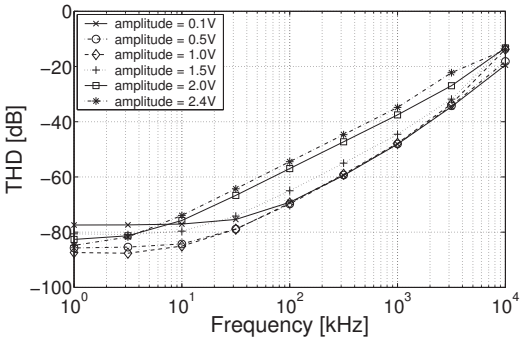
(d) SR falling edge, $C_{\text{load}} = 40\,\text{pF}$



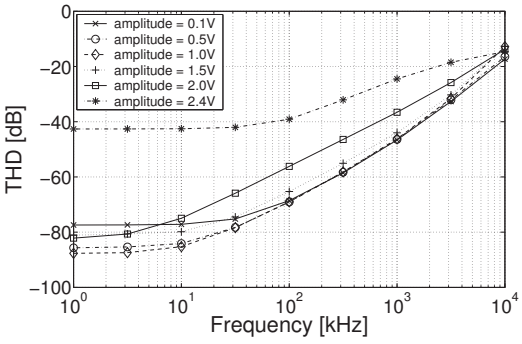
(e) CMRR vs frequency



(f) CMRR vs common mode voltage

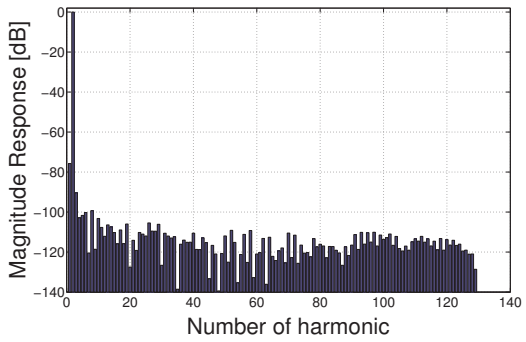


(g) THD, $R_{\text{load}} = 1\,\text{G}\Omega$

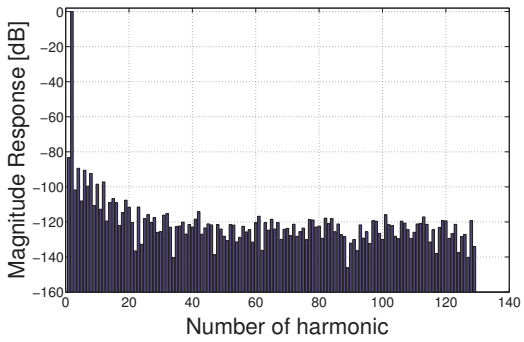


(h) THD, $R_{\text{load}} = 1\,\text{k}\Omega$

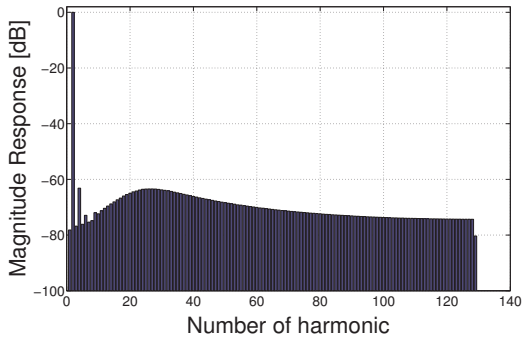
Figure B.11: CMRR ((a) and (b)), THD ((c) and (d)) and SR for the rising ((e) and (g) and the falling edge ((f) and (h) for the cell buffer. In (a) to (d) the stop voltage of each curve can be identified by its zero-crossing.



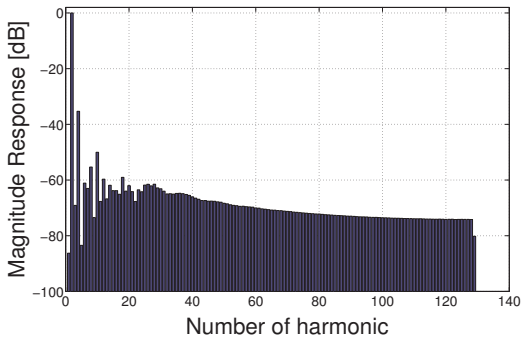
(a) amplitude $= 1.0\,\mathrm{V}$, $f = 1\,\mathrm{kHz}$



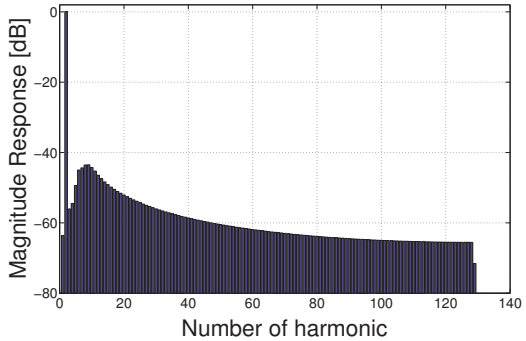
(b) amplitude $= 2.4\,\mathrm{V}$, $f = 1\,\mathrm{kHz}$



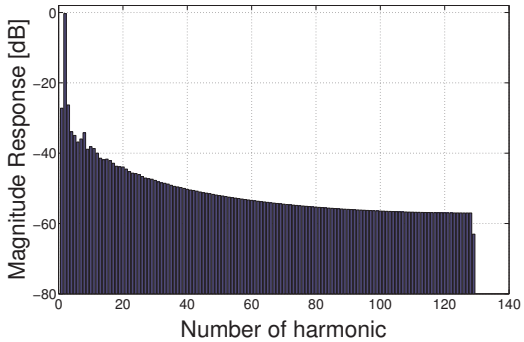
(c) amplitude $= 1.0\,\mathrm{V}$, $f = 1\,\mathrm{MHz}$



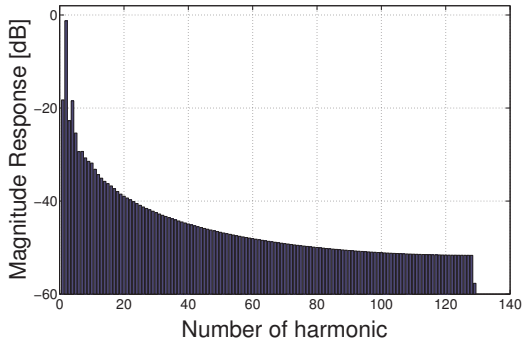
(d) amplitude $= 2.4\,\mathrm{V}$, $f = 1\,\mathrm{MHz}$



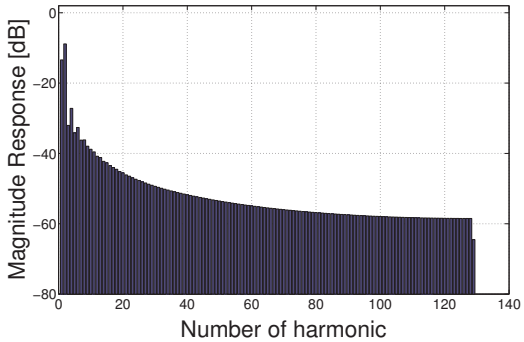
(e) amplitude $= 1.0\,\mathrm{V}$, $f = 3.2\,\mathrm{MHz}$



(f) amplitude $= 2.4\,\mathrm{V}$, $f = 3.2\,\mathrm{MHz}$

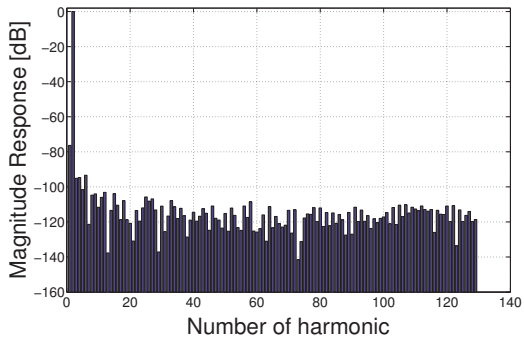


(g) amplitude $= 1.0\,\mathrm{V}$, $f = 10\,\mathrm{MHz}$

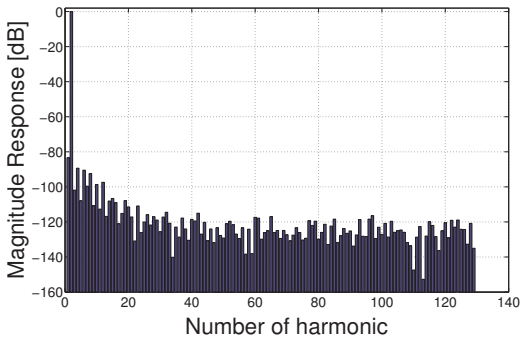


(h) amplitude $= 2.4\,\mathrm{V}$, $f = 10\,\mathrm{MHz}$

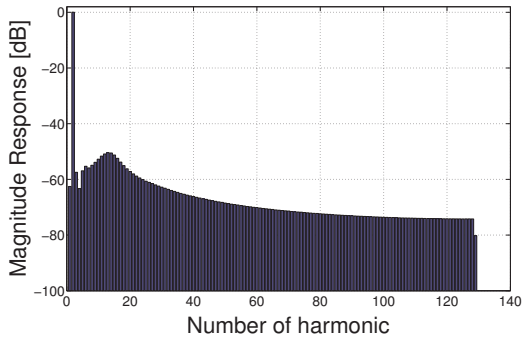
Figure B.12: Frequency spectra of the cell buffer for different input amplitudes and fundamental frequencies f. For each plot, the fundamental, the dc component and the first 126 harmonics are displayed. The op-amp was configured as a unity gain buffer. $R_{\mathrm{load}} = 1\,\mathrm{G\Omega}$, $C_{\mathrm{load}} = 20\,\mathrm{pF}$.



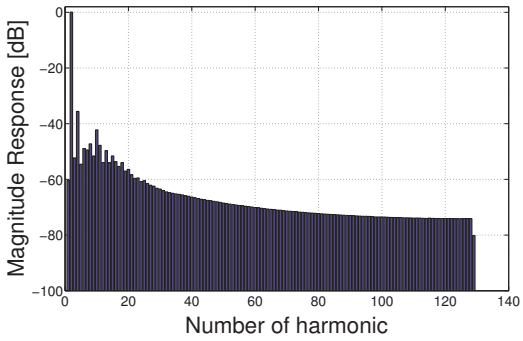
(a) amplitude $= 1.0\,\mathrm{V},\ f = 1\,\mathrm{kHz}$



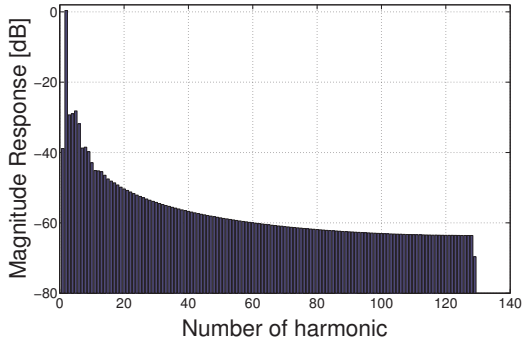
(b) amplitude $= 2.4\,\mathrm{V},\ f = 1\,\mathrm{kHz}$



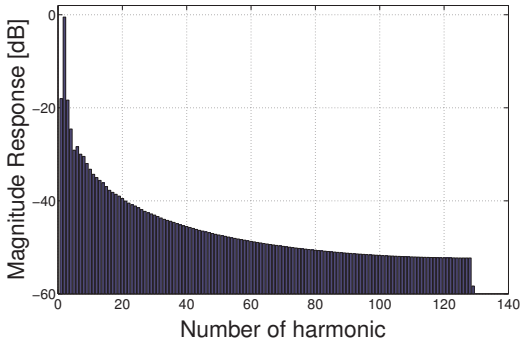
(c) amplitude $= 1.0\,\mathrm{V},\ f = 1\,\mathrm{MHz}$



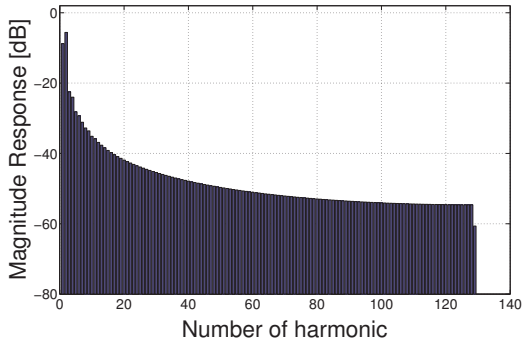
(d) amplitude $= 2.4\,\mathrm{V},\ f = 1\,\mathrm{MHz}$



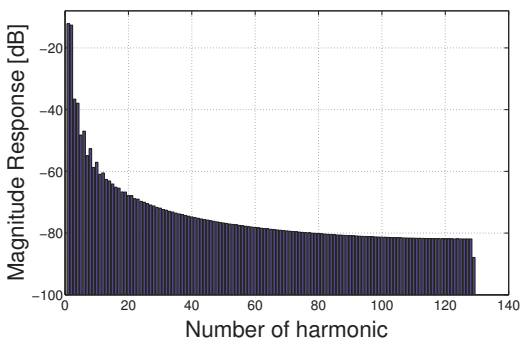
(e) amplitude $= 1.0\,\mathrm{V},\ f = 3.2\,\mathrm{MHz}$



(f) amplitude $= 2.4\,\mathrm{V},\ f = 3.2\,\mathrm{MHz}$

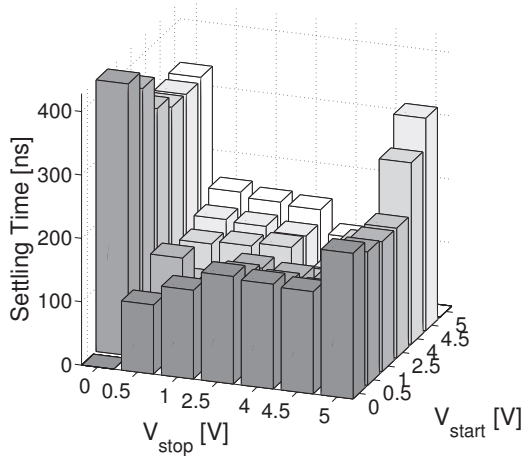


(g) amplitude $= 1.0\,\mathrm{V},\ f = 10\,\mathrm{MHz}$

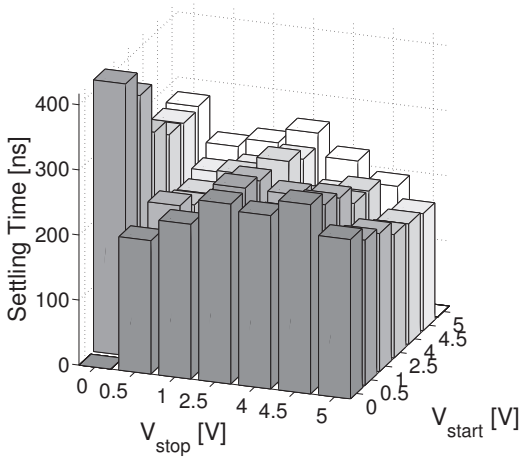


(h) amplitude $= 2.4\,\mathrm{V},\ f = 10\,\mathrm{MHz}$

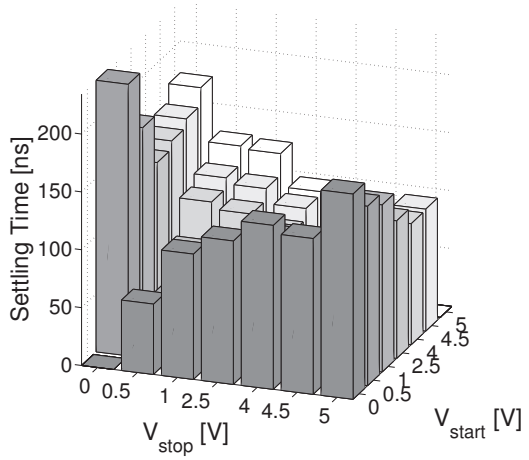
Figure B.13: Frequency spectra of the cell buffer for different input amplitudes and fundamental frequencies f. For each plot, the fundamental, the dc component and the first 126 harmonics are displayed. The op-amp was configured as a unity gain buffer. $R_{\mathrm{load}} = 1\,\mathrm{G\Omega}$, $C_{\mathrm{load}} = 40\,\mathrm{pF}$.



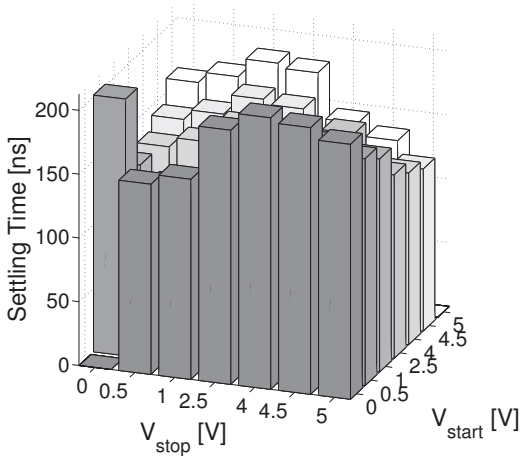
(a) settling time, precision = 0.1%, $C_{\text{load}} = 20\,\text{pF}$



(b) settling time, precision = 0.1%, $C_{\text{load}} = 40\,\text{pF}$



(c) settling time, precision = 1%, $C_{\text{load}} = 20\,\text{pF}$



(d) settling time, precision = 1%, $C_{\text{load}} = 40\,\text{pF}$

Figure B.14: Settling time for cell buffer for different precisions (top/bottom) and different load capacitors (left/right).

V_{stop} \ V_{start}	0.0	0.5	1.0	2.5	4.0	4.5	5.0
0.0	–	428	400	346	327	326	331
0.5	109	–	140	82	119	136	158
1.0	140	100	–	96	124	133	150
2.5	170	149	140	–	129	126	146
4.0	165	144	133	106	–	79	108
4.5	161	139	128	99	150	–	91
5.0	230	210	205	204	286	335	–

Table B.8: Settling time in ns for the cell buffer: Precision $= 0.1\%$, $R_{\text{load}} = 1\,\text{G}\Omega$, $C_{\text{load}} = 20\,\text{pF}$.

V_{stop} \ V_{start}	0.0	0.5	1.0	2.5	4.0	4.5	5.0
0.0	–	417	378	301	276	273	277
0.5	205	–	217	197	193	204	224
1.0	238	196	–	208	230	218	237
2.5	276	280	270	–	258	241	260
4.0	266	243	256	230	–	194	224
4.5	291	270	259	230	228	–	193
5.0	245	223	212	189	185	180	–

Table B.9: Settling time in ns for the cell buffer: Precision $= 0.1\%$, $R_{load} = 1\,G\Omega$, $C_{load} = 40\,pF$.

V_{stop} \ V_{start}	0.0	0.5	1.0	2.5	4.0	4.5	5.0
0.0	–	429	402	351	333	333	364
0.5	109	–	141	89	126	144	199
1.0	139	100	–	99	113	126	179
2.5	189	168	157	–	152	145	198
4.0	200	176	164	161	–	143	234
4.5	215	192	180	148	181	–	261
5.0	284	262	250	221	201	185	–

Table B.10: Settling time in ns for the cell buffer: Precision $= 0.1\%$, $R_{load} = 10\,k\Omega$, $C_{load} = 20\,pF$.

V_{stop} \ V_{start}	0.0	0.5	1.0	2.5	4.0	4.5	5.0
0.0	–	419	383	313	290	289	322
0.5	206	–	175	195	198	211	264
1.0	240	199	–	211	211	224	278
2.5	304	312	303	–	294	273	326
4.0	348	326	315	321	–	299	351
4.5	391	370	361	338	313	–	403
5.0	271	248	238	212	207	194	–

Table B.11: Settling time in ns for the cell buffer: Precision $= 0.1\%$, $R_{load} = 10\,k\Omega$, $C_{load} = 40\,pF$.

Appendix C

Correlation Coefficient

The *empirical* correlation coefficient R measures the linear correlation of a 2-dimensional sample (X,Y) with

$$X = \{X_1, X_2, \ldots, X_N\} \quad \text{and} \quad Y = \{X_1, X_2, \ldots, X_N\} \quad . \tag{C.1}$$

In order to define R, we denote the *empirical* mean for X and Y by

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \quad \text{and} \quad \bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i \quad . \tag{C.2}$$

The *empirical* squares of the according variances S_x and S_y can then be defined as

$$S_x^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2 \quad \text{and} \quad S_y^2 = \frac{1}{N-1}\sum_{i=1}^{N}(y_i - \bar{y})^2 \quad , \tag{C.3}$$

whereas the covariance of both sets is calculated as

$$S_{xy} = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y}) \quad . \tag{C.4}$$

The correlation coefficient R(x,y) can now be written as

$$R(x,y) = \frac{S_{xy}}{S_x \cdot S_y} \quad . \tag{C.5}$$

The correlation coefficient R(X,Y) has the following useful properties ([Bos97], [Moo97], [Sta00]):

a) $-1 \leq R(X,Y) \leq 1$

b) $|R| = 1 \quad \Rightarrow \quad y_i = ax_i + b \quad \text{for} \quad i = 1,2,\ldots,N \quad \text{and} \quad R = \begin{cases} 1 & \Rightarrow \quad a > 0 \\ -1 & \Rightarrow \quad a < 0 \end{cases}$

c) $y_i = ax_i + b \quad \text{for} \quad i = 1,2,\ldots,N \quad \Rightarrow \quad |R| = 1 \quad \text{with} \quad R = \begin{cases} 1 & \text{if} \quad a > 0 \\ -1 & \text{if} \quad a < 0 \end{cases}$

d) The smaller the correlation coefficient R, the less significant is the linear relation between X and Y. However, there are innumerate ways that lead to a small $|R|$ or even $|R| = 0$. Some important examples are:

- X and Y are statistically independent
- Y is a nonlinear function of X
- One or more outliers severely reduce an R value that would be close to or equal 1 without these outliers

Appendix D

Signals and Systems Analysis

This appendix gathers a few definitions, proofs and auxiliary calculations related to the pieces of signal and system analysis utilized in Chapter 7. The different topics are presented in their order of appearance within this chapter.

D.1 Discrete Fourier Transform

According to Fliege ([Fli91d], pp. 228) and in accordance with the definition in the utilized FFT[1] software [Fri03], the DFT can be written as:

$$\text{DFT:} \quad F_D(k) = \sum_{n=0}^{N-1} f_D(n)\, e^{-2\pi j \frac{nk}{N}} \tag{D.1a}$$

$$\text{IDFT:} \quad f_D(n) = \frac{1}{N} \sum_{k=0}^{N-1} F_D(k)\, e^{2\pi j \frac{nk}{N}} \quad , \tag{D.1b}$$

where IDFT[2] denotes the inverse DFT.

D.1.1 Parseval's Relation for the DFT

Given the signal x(n) and its Fourier coefficients X(k), both sampled at N points, Parseval's relation states that the total energy inherent to the signal can be expressed by its Fourier coefficients by the following equation:

$$\sum_{n=0}^{N-1} |x(n)|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |X(k)|^2 \quad . \tag{D.2}$$

The proof follows that one given in [Fli91d] (pp. 225–226), which derives the relation for discrete Fourier series, which places the normalization factor $1/N$ in the transform and not as in the DFT in the inverse transform: The left hand side of (D.2) can be rewritten by means of (D.1b), which after

[1] Fast Fourier Transform
[2] Inverse Discrete Fourier Transform

some reordering leads to the desired result:

$$\begin{aligned}\sum_{n=0}^{N-1} x(n)x^*(n) &\overset{\text{IDFT}}{=} \sum_{n=0}^{N-1} x(n)\,\frac{1}{N}\sum_{k=0}^{N-1} X^*(k)\,e^{-2\pi j\frac{nk}{N}} \\ &= \frac{1}{N}\sum_{k=0}^{N-1} X^*(k)\,\underbrace{\sum_{n=0}^{N-1} x(n)\,e^{-2\pi j\frac{nk}{N}}}_{X(k)} \\ &= \frac{1}{N}\sum_{k=0}^{N-1} |X(k)|^2 \ .\end{aligned} \tag{D.3}$$

D.2 Around the Unit Circle on a Shoestring

In section 7.2.2 the following relation was utilized as (7.28):

$$\sum_{n=0}^{N-1} \sin^2\left(2\pi\frac{n}{N}+\varphi\right) = \frac{N}{2} \quad \text{for} \quad N \geq 3 \ , \tag{D.4}$$

which we will derive below: First, rewriting (D.2) in terms of complex exponential functions yields:

$$\begin{aligned}\sum_{n=0}^{N-1} \sin^2\left(2\pi\frac{n}{N}+\varphi\right) &= \sum_{n=0}^{N-1}\left(\frac{1}{2j}\left(e^{2\pi j\frac{n}{N}+j\varphi} - e^{-2\pi j\frac{n}{N}-j\varphi}\right)\right)^2 \\ &= \frac{N}{2} - \frac{1}{4}\left(e^{2j\varphi}\sum_{n=0}^{N-1} e^{2\pi j\frac{2n}{N}} + e^{-2j\varphi}\sum_{n=0}^{N-1} e^{-2\pi j\frac{2n}{N}}\right) \ .\end{aligned} \tag{D.5}$$

This expression can be simplified by rewriting the second summation term. We replace n by $\tilde{n} = N-n$, and obtain:

$$\begin{aligned}\sum_{n=0}^{N-1} e^{-2\pi j\frac{2n}{N}} &= \sum_{\tilde{n}=1}^{N} e^{-2\pi j\frac{2(N-\tilde{n})}{N}} = \sum_{\tilde{n}=1}^{N} e^{-4\pi j} e^{2\pi j\frac{2\tilde{n}}{N}} \\ &= \sum_{\tilde{n}=0}^{N-1} e^{2\pi j\frac{2\tilde{n}}{N}} + e^{2\pi j\cdot 2} - e^{2\pi j\cdot 0} = \sum_{n=0}^{N-1} e^{2\pi j\frac{2n}{N}} \ .\end{aligned} \tag{D.6}$$

Thus, (D.5) simplifies to

$$\sum_{n=0}^{N-1} \sin^2\left(2\pi\frac{n}{N}+\varphi\right) = \frac{N}{2} - \frac{1}{2}\cos(2\varphi)\sum_{n=0}^{N-1} e^{2\pi j\frac{2n}{N}} \ . \tag{D.7}$$

Finally, we need to show that the sum on the right hand side of (D.5) vanishes for $N > 2$: To achieve this, we rewrite this sum as a geometric series and calculate the according result:

$$\begin{aligned}\sum_{n=0}^{N-1} e^{2\pi j\frac{2n}{N}} &= \sum_{n=0}^{N-1}\left(e^{\frac{4\pi j}{N}}\right)^n = \frac{1-\left(e^{\frac{4\pi j}{N}}\right)^N}{1-e^{\frac{4\pi j}{N}}} \\ &= \frac{1-1}{1-e^{\frac{4\pi j}{N}}} = 0 \quad \text{for} \quad N \geq 3 \ .\end{aligned} \tag{D.8}$$

Thus, (D.7) becomes (D.4), which completes the derivation.

D.3 LTI System Response to Sinusoidal Inputs

Within this section, we want to prove the generalized form of (7.25) in 7.2.2 on page 196: For a harmonic input signal of the form

$$x(t) = A\sin(\omega t + \phi) \ , \tag{D.9}$$

the output of an LTI system that possesses a real-valued impulse response $h(t)$ with according transfer function $H(j\omega)$ is given as:

$$y(t) = A\,|H(j\omega)|\,\sin(\omega t + \phi + \varphi(\omega)) \ . \tag{D.10}$$

According to (7.5) the output $y(t)$ can be written as:

$$\begin{aligned} y(t) = h(t) * x(t) &= \int_{-\infty}^{\infty} h(\tau)\,x(t-\tau)\,d\tau \\ &= \frac{A}{2j}\int_{-\infty}^{\infty} h(\tau)\left(e^{j\omega(t-\tau)+\phi} - e^{-j\omega(t-\tau)-\phi}\right)d\tau \\ &= \frac{A}{2j}\Bigg[\underbrace{\int_{-\infty}^{\infty} h(\tau)\,e^{-j\omega\tau}\,d\tau}_{H(j\omega)}\ e^{j\omega t+\phi} - \underbrace{\int_{-\infty}^{\infty} h(\tau)\,e^{j\omega\tau}\,d\tau}_{H(-j\omega)}\ e^{-(j\omega t+\phi)}\Bigg] \ . \end{aligned} \tag{D.11}$$

If we had looked at eigenfunctions of the form $x(t) = \exp(j\omega t + \phi)$, the proof would be complete, since we would have obtained only one expression resembling that one on the left of the last line of (D.11). For the real-valued harmonic function of (D.9) things are a bit more complicated, in that we have to rewrite $H(j\omega)$ as

$$H(j\omega) = |H(j\omega)|e^{j\varphi(\omega)} = M(\omega)e^{j\varphi(\omega)} \ , \tag{D.12}$$

and we have to assume that

$$H(-j\omega) = M(\omega)e^{-j\varphi(\omega)} \ , \tag{D.13}$$

which we will relate to the condition of $h(t)$ being real-valued later. With (D.13) (D.11) becomes:

$$\begin{aligned} y(t) &= \frac{A}{2j}\left[M(\omega)e^{j\omega t+\phi+\varphi(\omega)} - M(\omega)e^{-(j\omega t+\phi+\varphi(\omega))}\right] \\ &= A\,M(\omega)\,\sin(\omega t + \phi + \varphi(\omega)) \ . \end{aligned} \tag{D.14}$$

Now we still need to justify (D.13). Therefore we start with showing that a real valued $h(t)$, implies

$$h^*(t) = h(t) \ \rightarrow \ H^*(j\omega) = H(-j\omega) \ . \tag{D.15}$$

Since $h(t)$ and $H(j\omega)$ are related by the Fourier transform of (7.8), we can write

$$\begin{aligned} H^*(j\omega) &= \left[\int_{-\infty}^{\infty} h(t)\,e^{-j\omega t}\,dt\right]^* \\ &= \int_{-\infty}^{\infty} h^*(t)\,e^{j\omega t}\,dt \overset{h\ \text{real}}{=} H(-j\omega) \ , \end{aligned} \tag{D.16}$$

yielding the desired result. According to (D.12) we can now write:

$$H(-j\omega) = H^*(j\omega) = |H^*(j\omega)| \left[e^{j\varphi(\omega)} \right]^* = |H^*(j\omega)| e^{-j\varphi(\omega)} \ , \tag{D.17}$$

which proves (D.13).

Final Remarks. In section 7.1.1.3 on page 185 it was argued that most of the technically important systems can be described by rational transfer functions. Therefore, it should be noted that it can be shown that the according impulse response of such LTI systems must be real-valued. To do so, nominator and denominator of the transfer function are split into an even and an odd polynomial of $j\omega$:

$$\begin{aligned} H(j\omega) &= \frac{N_e(j\omega) + N_o(j\omega)}{D_e(j\omega) + D_o(j\omega)} \\ &= \frac{N_e(j\omega)D_e(j\omega) - N_o(j\omega)D_o(j\omega)}{D_e^2(j\omega) - D_o^2(j\omega)} + \frac{N_o(j\omega)D_e(j\omega) - N_e(j\omega)D_o(j\omega)}{D_e^2(j\omega) - D_o^2(j\omega)} \ . \end{aligned} \tag{D.18}$$

While the first term is an even, the second is an odd function of $j\omega$. In conclusion, the even term is real and the second one is pure imaginary. Altogether, this leads to the desired result of $H^*(j\omega) = H(-j\omega)$. Now first, this implies that harmonic functions are also eigenfunctions to LTI systems that possess a rational transfer function. Furthermore, the reverse direction of the statement contained in (D.16) can be shown:

$$\begin{aligned} h^*(t) &= \left[\int_{-\infty}^{\infty} H(j\omega)\, e^{j\omega t}\, d\omega \right]^* \\ &\overset{H^*(j\omega)=H(-j\omega)}{=} \int_{-\infty}^{\infty} H(-j\omega)\, e^{-j\omega t}\, d\omega \\ &\overset{\text{substitute } \omega \text{ by } -\omega'}{=} \int_{-\infty}^{\infty} H(j\omega')\, e^{j\omega' t}\, d\omega' h(t) \ , \end{aligned} \tag{D.19}$$

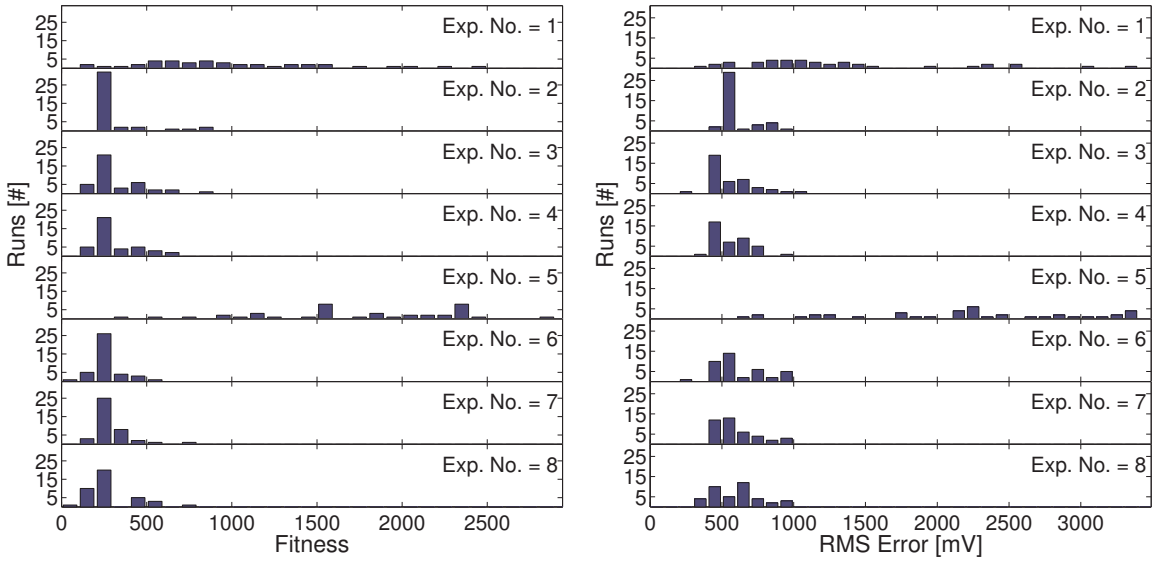
such that indeed all LTI systems possessing a rational transfer function must possess a real impulse response.

Appendix E

Additional Series of Comparator Experiments: Large Settling Time

The same series of 8 experiments whose results were presented in case study III (section 8.5) of Chapter 8 are carried out in the slower timing setting, which was used for the test of the hand-designed comparator in Fig. 8.23(c) and (d) in section 8.5.5. Thereby, the settling time between the clock enable signal for the second input and that one triggering the sampling of the output is raised from 1.075 µs to 5.275 µs. The results are presented below using the same types of plots as are used in section 8.5.

E.1 Comparison of the Different Experiments: Histograms

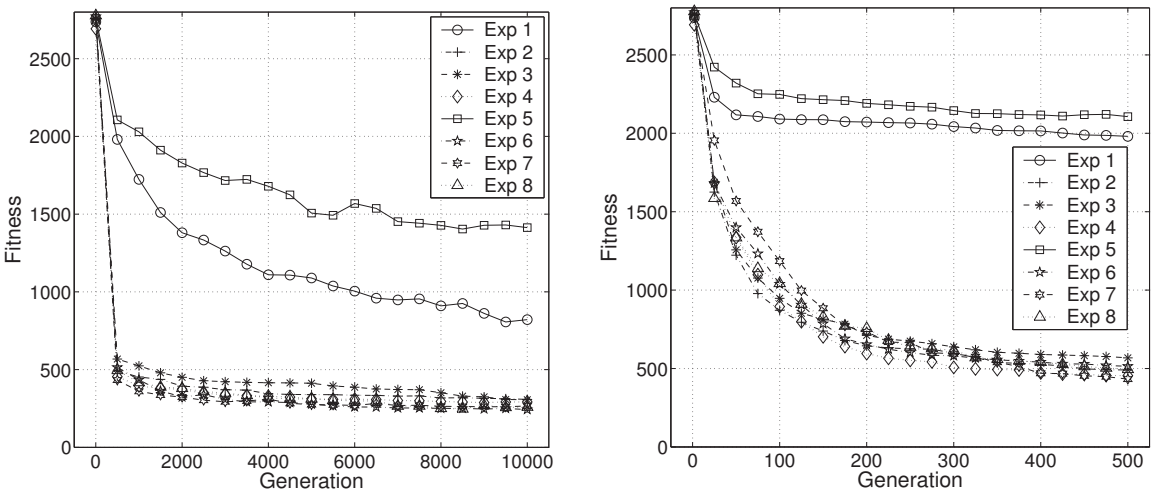


(a) fitness criterion used during evolution

(b) RMS error [mV]

Figure E.1: (a) worst fitness and (b) worst RMS error obtained from 100 verification tests for all 8 experiments. The bin size is set to 100 mV in all histograms. Analogon to Fig. 8.24.

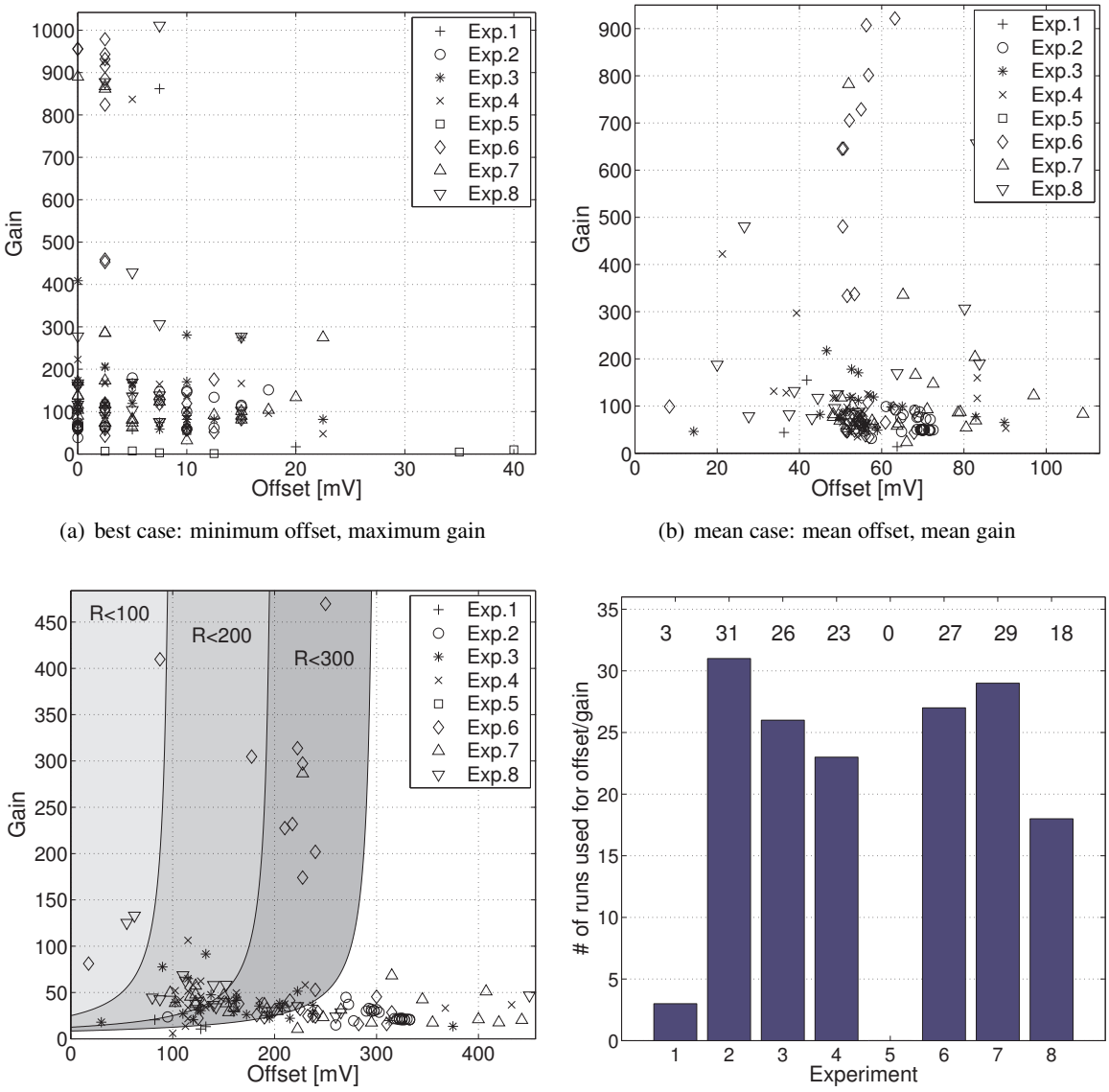
E.2 Comparison of the Different Experiments: Convergence



(a) generations $0 \ldots 10,000$ (b) generations $0 \ldots 500$

Figure E.2: Mean fitness averaged over all 40 runs of each of the 8 experiments for selected generations. Analogon to Fig. 8.25. (a) displays averages for the full length of the runs, (b) zooms into the first 500 generations.

E.3 Comparison of the Different Experiments: Gain and Offset



(a) best case: minimum offset, maximum gain

(b) mean case: mean offset, mean gain

(c) worst case: max offset, min gain

(d) # of experiments considered for the offset/gain analysis

Figure E.3: (a)-(c) Scatter plots of offset and gain for all the best-of-run solutions of all 8 experiments that exhibit the principle comparating functionality for all V_{set} . (d) accounts for the number of runs per experiment considered in (a)-(c). Analogon to Fig. 8.26.

E.4 Best-of-Experiment Output Characteristics

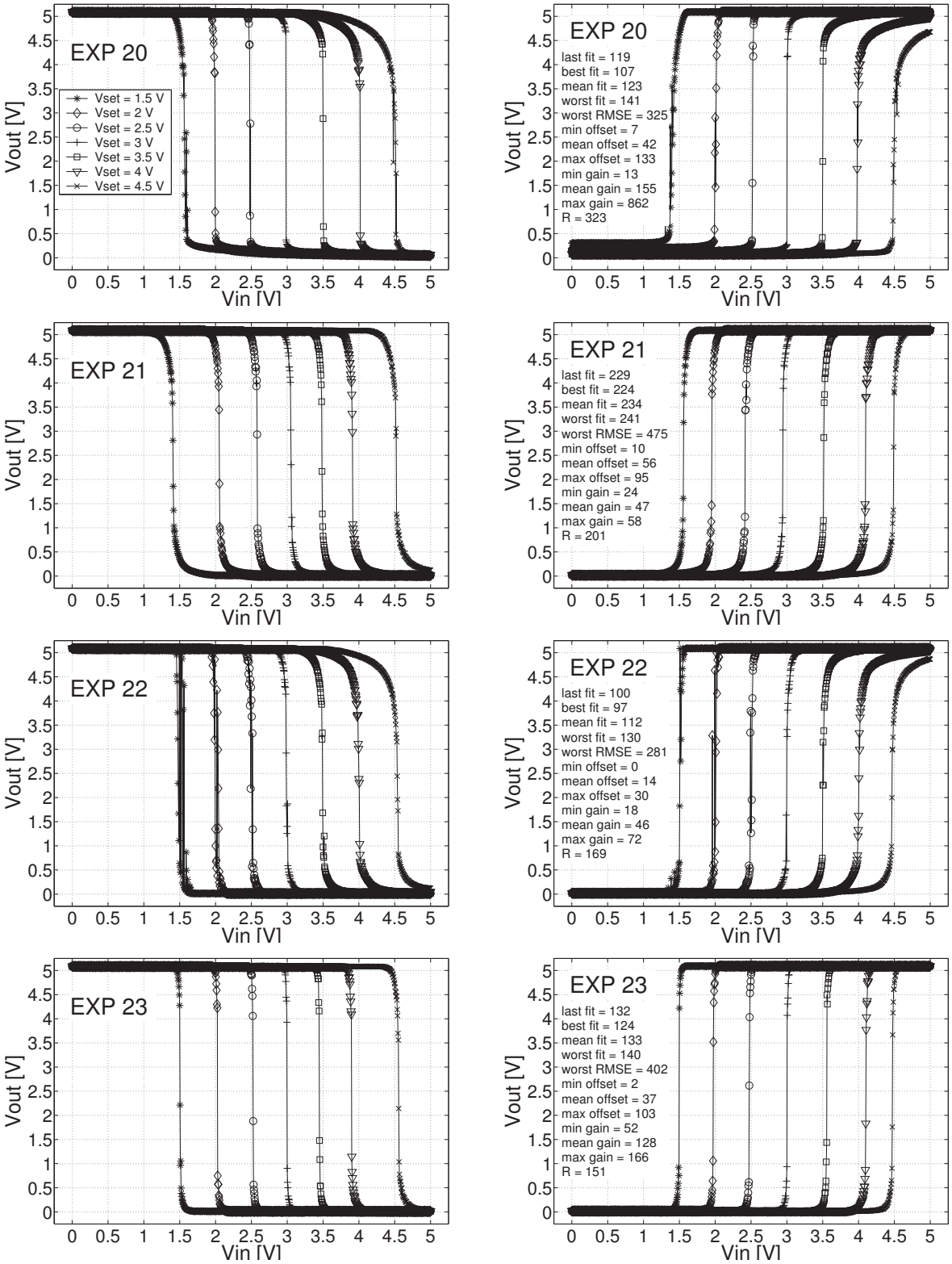


Figure E.4: Output characteristic for best-of-experiment solutions for experiments 1 . . . 4. The data is obtained from the first of 100 verification tests, where *best* here refers to the worst fitness value obtained in 100 verification tests evaluated by means of (8.9), the fitnesscriterion used during evolution. **Left**: Output behavior for test mode 1. **Right**: Output behavior for test mode 2. Read from top to bottom the plots correspond to experiments 1 to 4. Analogon to Fig. 8.27.

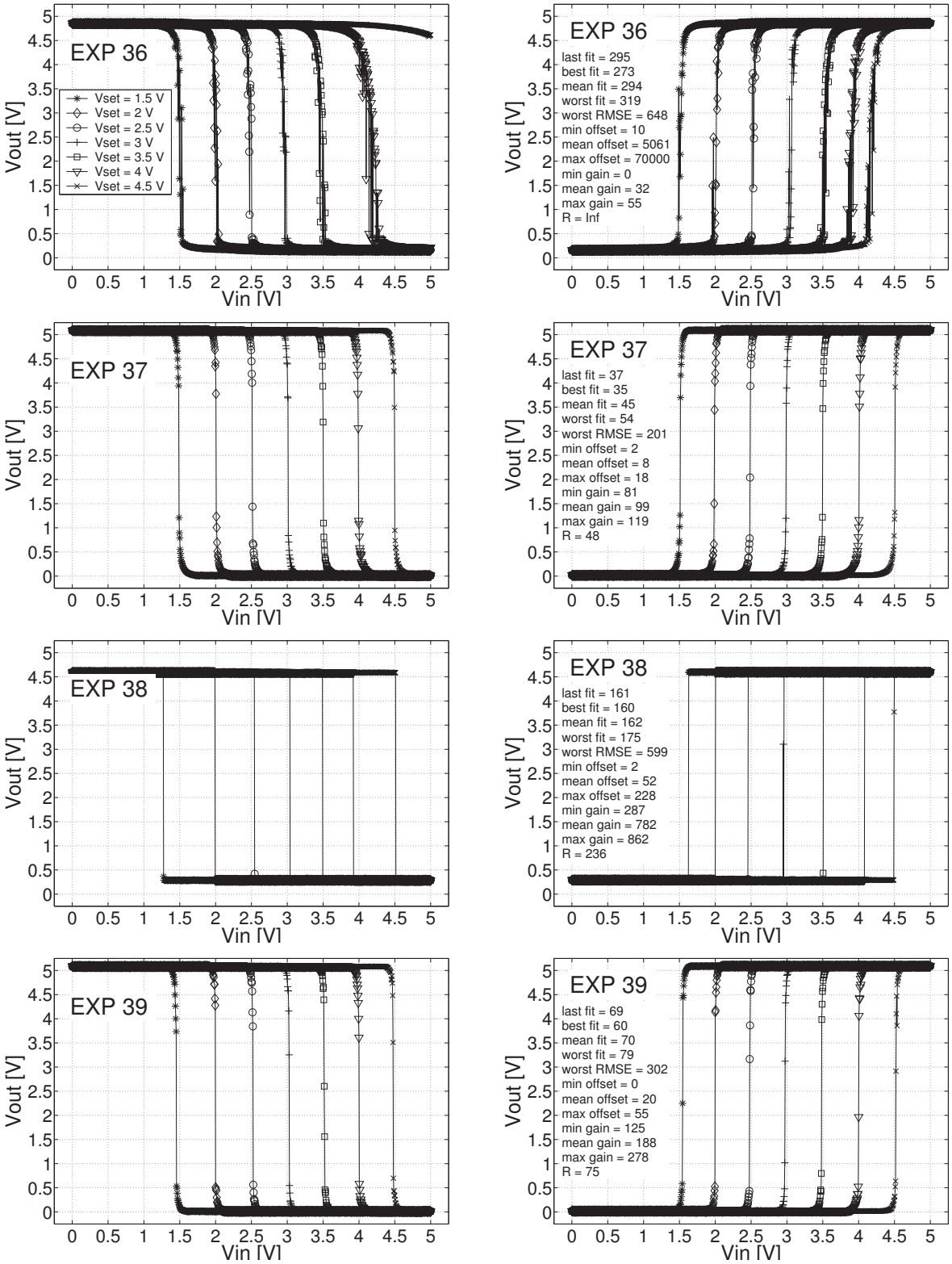
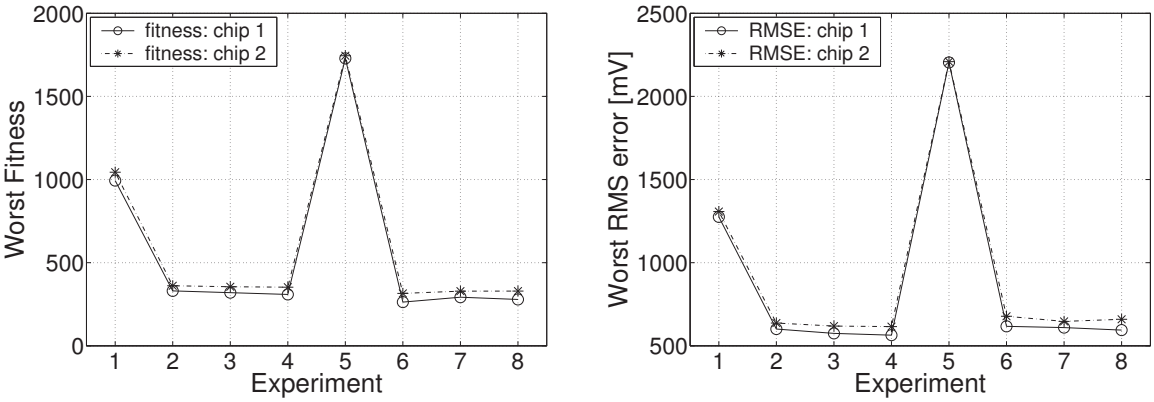


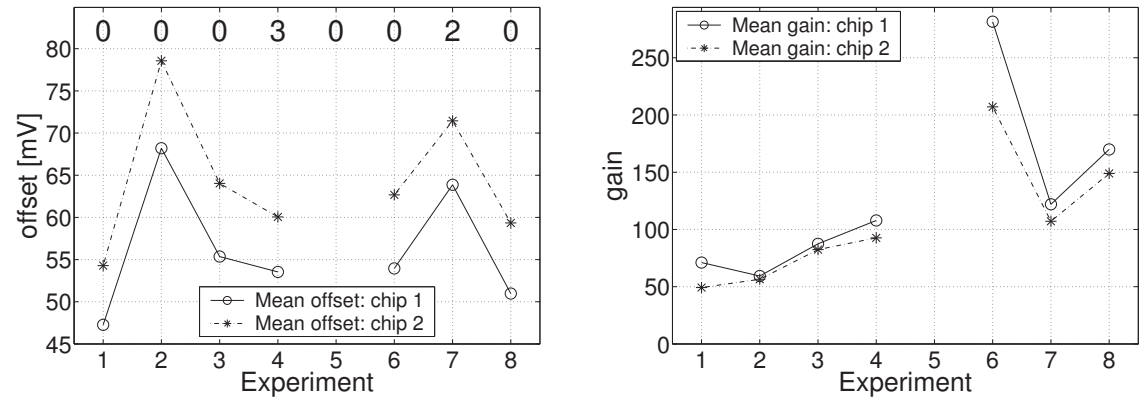
Figure E.5: Output characteristic for best-of-experiment solutions for experiments 5 . . . 8. The data is obtained from the first of 100 verification tests, where *best* here refers to the worst fitness value obtained in 100 verification tests evaluated by means of (8.9), the fitnesscriterion used during evolution. **Left**: Output behavior for test mode 1. **Right**: Output behavior for test mode 2. Read from top to bottom the plots correspond to experiments 5 to 8. Analogon to Fig. 8.28

E.5 Test on a Second Chip



(a) fitness criterion used during evolution (b) RMS error [mV]

Figure E.6: Comparison of the performance achieved on chip 1 and 2 in terms of (a) the fitness criterion used during the evolution process described by (8.9) and (b) of the RMS error according to (8.11). Analogon to Fig. 8.29



(a) offset (b) gain

Figure E.7: Comparison of the performance achieved on chip 1 and 2 in terms of (a) offset and (b) gain of the evolved comparators. Analogon to Fig. 8.30

Appendix F

List of Acronyms

ADC	Analog-to-Digital Converter
ADSL	Asymmetric Digital Subscriber Line
AHDL	Analog Hardware Description Language
ASIC	Application Specific Integrated Circuit
BB	building block
BJT	Bipolar Junction Transistor
BPF	bandpass filter
BSF	bandstop filter
BW	bandwidth
BiCMOS	Bipolar Complementary Metal-Oxide Semiconductor
CAB	Configurable Analog Block
CAD	Computer-Aided Design
CMOS	Complementary Metal-Oxide Semiconductor
CMRR	Common Mode Rejection Ratio
DAC	Digital-to-Analog Converter
DMA	Direct Memory Access
DECT	Digital European Cordless Telephone
DFT	Discrete Fourier Transform
DNA	DesoxyriboNucleic Acid
DNL	Differential NonLinearity
DR	Dynamic Range

DRAM Dynamic Random Access Memory

DSL Digital Subscriber Line

EA evolutionary algorithm

EC evolutionary computation

EEPROM Electrically Erasable and Programmable Read Only Memory

ENOBS Effective Number Of BitS

ES evolution strategies

ESD Electro-Static Discharge

FFT Fast Fourier Transform

FFTW Fastest Fourier Transform of the West

FPAA Field Programmable Analog Array

FPGA Field Programmable Gate Array

FPTA field programmable transistor array

FPTA-0 field programmable transistor array

FPTA-2 field programmable transistor array

GA genetic algorithm

GBP Gain Bandwidth Product

GP genetic programming

GPIB General Purpose Interface Bus

HWE hardware evolution

HDTV High Definition Television

HPF highpass filter

IDFT Inverse Discrete Fourier Transform

IF Intermediate Frequency

INL Integral NonLinearity

JPL Jet Propulsion Laboratory

LCD Liquid Crystal Display

LPF lowpass filter

LSI Large-Scale Integration

LTI	Linear Time-Invariant
MOS	Metal-Oxide Semiconductor
MOSFET	Metal-Oxide Semiconductor Field Effect Transistor
MOST	Metal-Oxide Semiconductor Transistor
MOST-R	Resistor implemented by means of MOS Transistor
MPW	Multi Project Wafer
MSPS	Mega Samples Per Second
NMOS	N-channel Metal-Oxide Semiconductor
OTA	Operational Transconductance Amplifier
PC	Personal Computer
PCB	Printed Circuit Board
PCI	Peripheral Component Interconnect
PD	power dissipation
PLL	Phase Locked Loop
PM	Phase Margin
PMOS	P-channel Metal-Oxide Semiconductor
PS	power supply
PSRR	Power Supply Rejection Ratio
PTA	programmable transistor array
RAM	Random Access Memory
RF	Radio Frequency
RMS	Root Mean Square
RMSE	root mean square error
RNA	RiboNucleic Acid
SC	Switched Capacitor
SDRAM	Synchronous Dynamic Random Access Memory
SFG	Signal Flow Graphs
SNR	Signal to Noise Ratio
SQP	Sequential Quadratic Programming

SR Slew Rate

SRAM Static Random Access Memory

SSE sum of squared errors

THD Total Harmonic Distortion

THD+N THD + Noise

TSE test sequence element

UGB Unity Gain Bandwidth

VHDL Verilog Hardware Description Language

VLSI Very Large-Scale Integration

VS voltage swing

XML Extensible Markup Language

Bibliography

[Abi96] Asad A. Abidi. CMOS-only rf and baseband circuits for a monolithic 900 mhz wireless transceiver. In *Proceedings of the 1996 Bipolar Circuits & Technology Meeting*, pages 35–42, Oct 1996.

[All02a] Phillip E. Allen and Douglas R. Holberg. *CMOS Analog Circuit Design*, chapter 2, pages 18–71. Oxford University Press, Inc. 198 Madison Avenue, New York, NY, USA, 2 edition, 2002. ISBN: 0-19-511644-5.

[All02b] Phillip E. Allen and Douglas R. Holberg. *CMOS Analog Circuit Design*. Oxford University Press, Inc. 198 Madison Avenue, New York, NY, USA, 2 edition, 2002. ISBN: 0-19-511644-5.

[All02c] Phillip E. Allen and Douglas R. Holberg. *CMOS Analog Circuit Design*, chapter 4.1, pages 113–124. Oxford University Press, Inc. 198 Madison Avenue, New York, NY, USA, 2 edition, 2002. ISBN: 0-19-511644-5.

[All02d] Phillip E. Allen and Douglas R. Holberg. *CMOS Analog Circuit Design*, chapter 5, pages 167–242. Oxford University Press, Inc. 198 Madison Avenue, New York, NY, USA, 2 edition, 2002. ISBN: 0-19-511644-5.

[All02e] Phillip E. Allen and Douglas R. Holberg. *CMOS Analog Circuit Design*, chapter 8, pages 439–491. Oxford University Press, Inc. 198 Madison Avenue, New York, NY, USA, 2 edition, 2002. ISBN: 0-19-511644-5.

[Alp03] Güner Alpaydın, Sina Balkır, and Günhan Dündar. An evolutionary approach to automatic synthesis of high-performance analog integrated circuits. *IEEE Transactions on Evolutionary Computation*, 7(3):240–252, June 2003.

[AN203] An221e04 datasheet – dynamically reconfigurable fpaa with enhanced i/o, 2003.

[Ana] Anadigm, inc. – website. http://www.anadigm.com.

[Ana96] Analog Devices, Inc. One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, USA. *AD768 Data Sheet: 16-Bit, 30 MSPS D/A Converter*, 1996.

[Ana03a] Anadigm, Inc. *AN121E04, AN221E04 Field Programmable Array – User Manual*, 2003.

[Ana03b] Analog Devices, Inc. One Technology Way, P. O./ Box 9106, Norwood, MA, USA. *Intelligent Temperature Monitor and Dual PWM Fan Controller*, rev. a edition, 2003. http://www.analog.com.

[AND04] Anadigmdesigner2 – user manual, 2004.

[Ant95] B. A. A. Antao and A. J. Brodersen. Archgen: Automated synthesis of analog systems. *IEEE Transaction on Very Large Scale Integration (VLSI) Systems*, 3(2):231–244, Jun 1995.

[Aus97a] Austria Mikro Systeme International AG, Unterpremstätten, Austria. *0.6 Micron Standard Cell Databook*, 1997.

[Aus97b] Austria Mikro Systeme International AG, Unterpremstätten, Austria. 0.6μm *CMOS Design Rules*, document 9931025 revision b edition, 1997.

[Aus97c] Austria Mikro Systeme International AG, Unterpremstätten, Austria. 0.6μm *CMOS Joint Group Process Parameters*, document 9933011 revision a edition, 1997.

[Aus97d] Austria Mikro Systeme International AG, Unterpremstätten, Austria. *Hit-Kit 3.20*, 1997. HitKit contains switches for analog extract, although no hint on this is given anywhere (not in the HitKit www documentation, neither in the openbook, nor in the Hit-Kit menus of CDS DFII.

[Bäc97] Thomas Bäck, Ulrich Hammel, and Hans-Paul Schwefel. Evolutionary computation: Comments on the history and current state. *IEEE Trans. on Evolutionary Computation*, 1(1):3–17, April 1997.

[Bäc04] Thomas Bäck. An overview of evolution strategies. Genetic and Evolutionary Computation Conference (GECCO 2004): Tutorial Program, jun 2004.

[Bak96] Anton Bakker and Johan H. Huijsing. Micropower CMOS temperature sensor with digital output. *IEEE Journal of Solid-State Circuits*, 31(7):933–937, July 1996.

[Bak03] Rola A. Baki and Mourad N. El-Gamal. A low-power 5-70-MHz seventh-order log-domain filter with programmable boost, group delay, and gain for hard disk drive applications. *IEEE Journal of Solid-State Circuits*, 38(2):205–215, Feb 2003.

[Bau03] Daniel Baumeister. *Development and Characterisation of a Radiation Hard Readout Chip for the LHCb-Experiment*. PhD thesis, Ruperto-Carola University of Heidelberg, Jan 2003.

[Bec01] Joachim Philipp Becker. Ein FPGA-basiertes Testsystem für gemischt analog/digitale ASICs. Diploma thesis (german), University of Heidelberg, HD-KIP-01-11, 2001.

[Bec04] Joachim Becker and Yiannos Manoli. A continuous-time field programmable analog array (fpaa) consisting of digitally reconfigurable g_M-cells. In *Proc.. of the 2004 Int.ernational Symposium on Circuits and Systems (ISCAS)*, pages 1092–1095, Vancouver, Canada, May 2004. IEEE Press.

[Ben99] Forrest H. Bennett III, John R. Koza, Martin A. Keane, Jessen Yu, William Mydlowec, and Oscar Stiffelman. Evolution by means of genetic programming of analog circuits that perform digital functions. In W. Banzhaf et al., editors, *Proc. of the Genetic and Evolutionary Computation Conference*, pages 1477–1483, Orlando, Florida, USA, July 1999. Morgan Kaufmann.

[Ben03] Peter J. Bentley. Evolving fractal proteins. In Andy M. Tyrell, Pauline C. Haddow, and Jim Torresen, editors, *Proc. 5th Int. Conf. on Evolvable Systems: From Biology to Hardware, ICES 2003, LNCS 2606*, LNCS 2606, pages 81–92, Trondheim, Norway, March 2003. Springer Verlag.

[Bli00] Holger Blinzinger. Aufbau eines kompakten bildverarbeitungsrechners für ein taktiles blindenhilfssystem. Diploma thesis (german), University of Heidelberg, March 2000.

[Bos97] Karl Bosch. *Elementare Einführung in die angewandte Statistik*, pages 128–138. Friedr. Vieweg & Sohn Verlagsgesellschaft mbH, Braunschweig/Wiesbaden, 5^{th} edition, 1997.

[Bot03] João P. B. Botelho, Leonardo B. Sá, Pedro F. Vieira, and A.C̃. Mesquita. An experriment on nonlinear synthesis using evolutionary techniques based only on CMOS transistors. In *Proc. of the 2003 NASA/DoD Conference on Evolvable Hardware*, pages 50–58, Chicago IL, USA, July 2003. IEEE Press. ISBN: 0-7695-1977-6.

[Bre04] Andreas Breidenassel, Karlheinz Meier, and Johannes Schemmel. A flexible scheme for adaptive integration time control. In *Proc. of IEEE Sensors 2004*, Vienna, Austria, Oct 2004.

[Bre05] Andreas Breidenassel. A high dynamic range CMOS image sensor witht adaptive integration time control. *PhD thesis*, University of Heidelberg, submitted, 2005.

[Bro96] A.P̃aul Brokaw. A temperature sensor with single resistor set-point programming. *IEEE Journal of Solid-State Circuits*, 31(12):1908–1915, December 1996.

[Brü02] Daniel Brüderle. Entwurf und implementierung von building blocks zum einsatz in genetischen algorithmen auf einem FPTA. Internal report for a student project, Oct 2002. KIP No: HD-KIP 02-25.

[Bug03] G. Bugeda, J.-A. Désidéri, J. Périaux, M. Schoenauer, and G. Winter, editors. *Evolutionary Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems (EUROGEN 2003)*, Barcelona, Spain, September 2003. International Center for Numerical Methods in Engineering (CIMNE). Published on CD: ISBN: 84-95999-33-1.

[Bur95] Burr-Brown Corporation, 6730 S. Tucson Blvd., Tucson, AZ 85706, USA. *ADS 800 Data Sheet: 12-Bit, 40MHz Sampling Analog-to-Digital Converter*, 1995.

[Bur99] Burr-Brown Corporation, 6730 S. Tucson Blvd., Tucson, AZ 85706, USA. *OPA2634 Data Sheet: Dual, Wideband, Single-Supply Operational Amplifier*, 1999.

[But00] Tilman Butz. *Fouriertransformation für Fußgänger*. B. G. Teubner, Stuttgart, Germany, 2 edition, 2000. ISBN: 3-519-06140-6.

[Cada] Cadence Design Systems, San Jose, CA, USA. *Cadence Online Library Openbook: Diva Reference*, 4.43 edition, 10. Chapter 14-9: Diva extract for setting the extract switches (e.g. capall). switches are part of extract rules provided by AMS (HitKit ?).

[Cadb] Cadence Design Systems, San Jose, CA, USA. *Cadence Online Library Openbook: spectreS reference manual*, 4.43 edition, 10. Description of the spectreS simulator.

[Cad04a] Cadence Design Systems, Inc. San Jose, CA, USA. *Cadence Virtuoso Neocell Analog Physical Synthesis Datasheet*, 06 2004.

[Cad04b] Cadence Design Systems, Inc. San Jose, CA, USA. *Cadence Virtuoso Neocircuit Datasheet*, 2004.

[Car01] Bruce Carter, Patrick Rowland, Jim Karki, and Perry Miller. *Amplifiers and Bits: An Introduction to Selecting Amplifiers for Data Converters*. Texas Instruments, Inc. P. O. Box 655303, Dallas, TE, USA, rev. b edition, Dec 2001. http://www.ti.com.

[CC02] C. A. Coello Coello, D. A. Van Veldhuizen, and G. B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic/Plenum Publishers, New York, 2002.

[Cha02] Kanad Chakraborty and Pinaki Mazum. *Fault-Tolerance and Reliability Techniques for High-Density Random-Access Memories*, chapter 1. Prentice Hall Modern Semiconductor Design Series. Prentice Hall PTR, 800 East 96th Street, Indianapolis, Indiana, USA, 1 edition, Jun 2002. ISBN: 0130084654.

[Czi92] G. Czihak, H. Langer, and H. Ziegler, editors. *Biologie*. Springer Verlag, Berlin, Heidelberg, New York, 5 edition, 1992. ISBN: 3-540-55528-5.

[Dar59] Charles R. Darwin. *On the Origin of Species*. John Murray, London, 1859.

[Deb04] Kalyanmoy Deb et al., editors. *Genetic and Evolutionary Computation Conference (GECCO 2004)*, Seattle, WA, USA, Jun 2004. Springer-Verlag, LNCS 3102. ISBN 3-540-22344-4.

[Des97] Design Automation Standards Committee of the IEEE Computer Society, New York. *VHDL Language Reference Manual, IEEE Std. 1076.1*, 1997.

[Dev03] Ontogenetic Development and Fault Tolerance in the POEtic Tissue. Tempesti, gianluca and roggen, daniel and sanchez, eduardo and thoma, yann and canham, richard and tyrell, andy. In Andy M. Tyrell, Pauline C. Haddow, and Jim Torresen, editors, *Proc. 5th Int. Conf. on Evolvable Systems: From Biology to Hardware, ICES 2003, LNCS 2606*, LNCS 2606, pages 141–152, Trondheim, Norway, March 2003. Springer Verlag.

[dJ04] Kenneth de Jong. Evolutionary computation: A unified approach. Genetic and Evolutionary Computation Conference (GECCO 2004): Tutorial Program, jun 2004.

[dMH01] Maria del Mar Hershenson, Stephen P. Boyd, and Thomas H. Lee. Optimal design of a CMOS op-amp via geometric programming. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 20(1):1–21, January 2001.

[dMH02a] Maria del Mar Hershenson. Design of pipeline analog-to-digital converters via geometric programming. In *Proc. of the 2002 Int. Conf. on Computer-Aided Design*, pages 317–324, San Jose, CA, USA, November 2002. IEEE Press.

[dMH02b] Maria del Mar Hershenson, Dave Colleran, Arash Hassibi, and Navraj Nandra. Synthesizable full custom mixed-signal IP. In *9'th IEEE/DATC Electonic Design Processes Workshop, http://www.eda.org/edps/edp02/program.html*, Monterey, CA, USA, April 2002.

[Dob03] Alex Doboli and Ranga Vermuri. Exploration-based high-level synthesis of lineaer analog systems operating at low/medium frequencies. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 22(11):1556–1568, Nov 2003.

[DV99] Niranjan Damera-Venkata and Brian L. Evans. An automated framework for multicriteria optimization of analog filter designs. *IEEE Transaction on Circuits and Systems – II: Analog and Digital Signal Processing*, 46(8):981–990, Aug 1999.

[Edw01] R.T̃imothy Edwards and C. J. Kim. Breaking the resistivity barrier. In *Proc. of the Third NASA/DOD Workshop on Evolvable Hardware*, pages 167–171, Long Beach, CA, USA, July 2001. IEEE Press.

[Eib03a] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Springer Verlag, Berlin, Heidelberg, New York, 2003.

[Eib03b] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*, chapter 8, pages 129–151. Springer-Verlag, Berlin Heidelberg New York, 2003. ISBN: 3-540-40184-9.

[Eic89] Christoph Eichenberger and Walter Guggenbühl. Dummy transistor compensation of analog MOS switches. *IEEE Journal of Solid-State Circuits*, 24(4):1143–1146, August 1989.

[Fer98] Guiseppe Ferri, Willy Sansen, and Vincenzo Peluso. A low-voltage fully differential constant-gm rail-to-rail CMOS operational amplifier. *Analog Integrated Circuits and Signal Processing*, 16:5–15, April 1998.

[Fer02] M.Ĩ. Ferguson, R. Zebulum, D. Keymeulen, and A. Stoica. An evolvable hardware platform based on DSP and FPTA. In Erick Cantú-Paz, editor, *Late Breaking Papers of the Genetic and Evolutionary Computation Conference GECCO 2002*, pages 145–152. Morgan Kaufmann Publishers, July 2002.

[Fie04] J. Fieres, A. Grübl, S. Philipp, K. Meier, J. Schemmel, and F. Schürmann. A platform for parallel operation of VLSI neural networks. In *Proc. of the 2004 Brain Inspired Cognitive Systems Conference (BICS2004)*, University of Stirling, Scotland, UK, 2004.

[Fli91a] Norbert Fliege. *Systemtheorie*, chapter 1, pages 1–31. B. G. Teubner, Stuttgart, Germany, 1 edition, 1991. ISBN: 3-519-06140-6.

[Fli91b] Norbert Fliege. *Systemtheorie*, chapter 3, page 90. B. G. Teubner, Stuttgart, Germany, 1 edition, 1991. ISBN: 3-519-06140-6.

[Fli91c] Norbert Fliege. *Systemtheorie*. B. G. Teubner, Stuttgart, Germany, 1 edition, 1991. ISBN: 3-519-06140-6.

[Fli91d] Norbert Fliege. *Systemtheorie*, chapter 5, pages 165–229. B. G. Teubner, Stuttgart, Germany, 1 edition, 1991. ISBN: 3-519-06140-6.

[Flo00] S. Flockton and K. Sheehan. Behavior of a building block for intrinsic evolution of analogue signal shaping and filtering circuits. In Jason Lohn, Adrian Stoica, and Didier Keymeulen, editors, *The Second NASA/DoD Workshop on Evolvable Hardware*, pages 117–124, Palo Alto, California, 13-15 July 2000. Jet Propulsion Laboratory, California Institute of Technology, 1730 Massachusetts Avenue, N.W., Washington, DC, 20036-1992, USA.

[Fog94] David B. Fogel. An introduction to simulated evolutionary optimization. *IEEE Transactions on Neural Networks*, 5(1):3–14, January 1994.

[Fog98] D. B. Fogel, editor. *Evolutionary Computation: the Fossil Record*. IEEE Press, Piscataway, NJ, 1998.

[Föl00] Simon Fölling. The GAQT handbook. Internal Report, University of Heidelberg, Institut für Hochenergiephysik, Electronic Vision(s) group, 2000.

[Fri03] Matteo Frigo and Steven G. Johnson. *FFTW User's Manual, Version 2.1.4*. Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA, USA, Mar 2003.

[Gar04] Miguel Garvie. *Reliable Electronics through Artificial Evolution*. PhD thesis, University of Sussex, December 2004.

[gcc] The GNU Compiler Collection. Free Software Foundation, Inc., 59 Temple Place, Boston, MA, USA, `http://gcc.gnu.org/`.

[Gei90a] Randall L. Geiger, Phillip E. Allen, and Noel R. Strader. *VLSI Design Techniques for Analog and Digital Circuits*. McGraw-Hill series in electrical engineering. McGraw-Hill, Inc. 1 edition, 1990. ISBN: 0-07-100728-8.

[Gei90b] Randall L. Geiger, Phillip E. Allen, and Noel R. Strader. *VLSI Design Techniques for Analog and Digital Circuits*, chapter 8, page 675. McGraw-Hill series in electrical engineering. McGraw-Hill, Inc. 1 edition, 1990. ISBN: 0-07-100728-8.

[Gie00] Georges G. E. Gielen and Rob A. Rutenbar. Computer-aided design of analog and mixed-signal integrated circuits. *Proceedings of the IEEE*, 88(12):1825–1852, Dec 2000.

[Gol89] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley Longman Inc., 1989.

[Gor02] T. G. W. Gordon and Peter J. Bentley. On evolvable hardware. In S. Ovaska and L. Sztandera, editors, *Soft Computing in Industrial Electronics*. Physica-Verlag, Heidelberg, Germany, 2002.

[Got94] Werner Gottschalk. *Allgemeine Genetik*. Georg Thieme Verlag, Stuttgart, Germany, 4 edition, 1994. ISBN: 3-13-508904-5.

[Gra01] H. Graeb, S. Zizala, J. Eckmueller, and K. Antreich. The sizing rules method for analog integrated circuit design. In *Proc. of the IEEE/ACM Int. Conf. on Computer Aided Design*, pages 343–349, San Jose, CA, USA, November 2001. IEEE Press. Published on CD, ISBN: 0-7803-7249-2.

[Gre86a] Roubik Gregorian and Gabor C. Temes. *Analog MOS Integrated Circuits for Signal Processing*, chapter 2, page 26. Wiley Series on Filters. John Wiley & Sons, Inc. 605 Third Avenue, New York, NY, USA, 1 edition, 1986. ISBN: 0-471-62569-8.

[Gre86b] Roubik Gregorian and Gabor C. Temes. *Analog MOS Integrated Circuits for Signal Processing*, chapter 5, pages 265–410. Wiley Series on Filters. John Wiley & Sons, Inc. 605 Third Avenue, New York, NY, USA, 1 edition, 1986. ISBN: 0-471-62569-8.

[Gre03] Garrison W. Greenwood, Edward Ramsden, and Saima Ahmed. An empirical comparison of evolutionary algorithms for evolvable hardware with minimum time-to-reconfigure requirements. In *Proc. of the 2003 NASA/DoD Conference on Evolvable Hardware*, pages 59–66, Chicago IL, USA, July 2003. IEEE Press. ISBN: 0-7695-1977-6.

[Gre04] Garrison W. Greenwood, David Hunter, and Edward Ramsden. Fault recovery in linear systems via intrinsic evolution. In *Proc. of the 2004 NASA/DoD Conference on Evolvable Hardware*, pages 115–122, Seattle, WA, USA, June 2004. IEEE Press. ISBN: 0-7695-2145-2.

[Gri99] James B. Grimbleby. Hybrid genetic algorithms for analogue network synthesis. In *Proc. of the IEEE Int. Congress on Evolutionary Computation (CEC 99), Vol 3*, pages 1781–1787, Washington, Jul 1999.

[Gri00] James B. Grimbleby. Automatic analogue circuit synthesis using genetic algorithms. *IEE Proceedings: Circuits, Devices and Systems*, 147(6):319–323, 2000.

[Had01] Pauline C. Haddow and Gunnar Tufte. Bridging the genotype-phenotype mapping for digital FPGAs. In *Proc. of the Third NASA/DOD Workshop on Evolvable Hardware*, pages 109–115, Long Beach, CA, USA, July 2001. IEEE Press.

[Har97] Inman Harvey and Adrian Thompson. Through the labyrinth evolution finds a way: A silicon ridge. In Tetsuya Higuchi, Masaya Iwata, and L. Weixin, editors, *Proc. 1st Int. Conf. on Evolvable Systems (ICES'96)*, volume 1259 of *LNCS*, pages 406–422. Springer-Verlag, 1997.

[Har04a] Simon Harding and Julian Francis Miller. Evolution in materio: A tone discriminator in liquid crystal. In *Proc. of the Congress on Evolutionary Computation, Vol. 2*, pages 1800–1807, Portland, OR, USA, June 2004. IEEE Press.

[Har04b] Simon Harding and Julian Francis Miller. Evolution in materio: Initial experiments with liquid crystal. In *Proc. of the Fourth NASA/DOD Workshop on Evolvable Hardware*, pages 298–305, Seattle, WA, USA, June 2004. IEEE Press.

[Hen02] Eckhard Henning, Ralf Sommer, and Lisa Charlack. An automated approach for sizing complex analog circuits in a simulation-based flow. Conference and Exhibition on Design Automation & Test in Europe (DATE 2002), March 2002.

[Her04] James Hereford and Charles Pruitt. Robust sensor systems using evolvable hardware. In *Proc. of the 2004 NASA/DoD Conference on Evolvable Hardware*, pages 161–168, Seattle, WA, USA, June 2004. IEEE Press. ISBN: 0-7695-2145-2.

[Hes] Jürgen Hesser. Evolutionäre Algorithmen, lecture notes, University of Mannheim, winter term 2002.

[Hoh02a] S. Hohmann, J. Schemmel, F. Schürmann, and K. Meier. Exploring the parameter space of a genetic algorithm for training an analog neural network. In W.B. et. al. Langdon, editor, *Proceedings of the Genetic and Evolutionary Computation Conference GECCO 2002*, pages 375–382. Morgan Kaufmann Publishers, July 2002.

[Hoh02b] Steffen Hohmann, Johannes Schemmel, Felix Schürmann, and Karlheinz Meier. Exploring the parameter space of a genetic algorithm for training and analog neural network. In Kalyanmoy Langdon et. al. editor, *Proc. of the Genetic and Evolutionary Computation Conference (GECCO 2002)*, pages 375–382, New York City, NY, USA, Jul 2002. Morgan Kaufmann Publishers. ISBN 1-55860-878-8.

[Hoh05] Steffen Hohmann. Stepwise evolutionary training strategies for hardware neural networks. *PhD thesis*, University of Heidelberg, submitted, 2005.

[Hol00] G. Hollingworth, S. Smith, and A. Tyrrell. The safe intrinsic evolution of virtex devices. In *Proc. of the Second NASA/DOD Workshop on Evolvable Hardware*, pages 195–202, Palo Alto, CA, USA, July 2000. IEEE Press.

[Hor94] D. H. Horrocks and Y. m. A. Khalifa. Genetically derived filters using preferred value components. In *Proc. of the IEE Colloq. on Linear Analogue Circuits and Systems*, pages 4/1–4/5, Oxford, UK, 1994.

[Hor96] D. H. Horrocks and Y. m. A. Khalifa. Genetic algorithm design of electronic analogue circuits including parasitic effects. In *Proc. of the First Online Workshop on Soft Computing (WSC1), Special Session on Evolutionary Electronics*, pages 274–278, Aug 1996.

[Hro04] J. Hromkovič. *Algorithmics for Hard Problems*. Springer Verlag, Berlin, Heidelberg, New York, 2004.

[Hu02a] Jian Jun Hu and Erik D. Goodman. The hierarchical fair competition (HFC) model for parallel evolutionary algorithms. In *Proc. of the IEEE World Congress on Evolutionary Computation, CEC 2002*, Honolulu, Hawai, May 2002. IEEE Press.

[Hu02b] Jianjun Hu, Erik D. Goodman, Kisung Seo, and Min Pei. Adaptive hierarchical fair competition (AHFC) model for parallel evolutionary algorithms. In W.B. et. al. Langdon, editor, *Proceedings of the Genetic and Evolutionary Computation Conference GECCO 2002*, pages 772–779. Morgan Kaufmann Publishers, July 2002.

[Hu03a] Jianjun Hu, Erik D. Goodman, and Kisung Seo. Continous hierarchical fair competition model for sustainable innovation in genetic programming. In *Genetic Programming Theory and Practice*, chapter 6, pages 81–98. Kluwer, 2003.

[Hu03b] Jianjun Hu, Erik D. Goodman, Kisung Seo, Zhun Fan, and Ronald C. Rosenberg. HFC: a continuing EA framework for scalable evolutionary synthesis. In H Lipson, E.K. Antonsson, and J.R. Koza, editors, *Proceedings of the 2003 AAAI Spring Symposium - Computational Synthesis: From Basic Building Blocks to High Level Functionality*, pages 106–113, Stanford CA, USA, March 2003. AAAI Symposium. AAAI Press ISBN 1-57735-179-7.

[Hua97] Qiuting Huang. A mosfet-only continuous-time bandpass filter. *IEEE Journal of Solid-State Circuits*, 32(2):147–158, Feb 1997.

[Hue99] Lorenz Huelsbergen, Edward Rietman, and Robert Slous. Evolving oscillators in silico. *IEEE Trans. on Evolutionary Computation*, 3:197–204, September 1999.

[ISO98] ISO/IEC 14882. *Programming Language C++*, July 1998.

[Jou03] Stéphane Jouin. *APPLICATION NOTE – TDA8769HW –: 12-bit High-Speed A/D Converter Demonstration Board, AN/10261-1*. Philips Semiconductor, Inc. Caen, France, Sep 2003. http://www.philips.com.

[JPL] JPL Evolvable Systems Web Page: http://ehw.jpl.nasa.gov/.

[Jun03] Jungo Ltd., Netanya. *Windriver 6 User's Manual*, 2003.

[Kes03] Walt Kester and James Bryant. *Mixed-Signal and DSP Design Techniques, Section 2: Sampled Data Systems*. Analog Devices, Inc. One Technology Way, P. O./ Box 9106, Norwood, MA, USA, Feb 2003. http://www.analog.com.

[Ket93] Thorsten Kettner, Christian Heite, and Klaus Schumacher. Analog CMOS realization of fuzzy logic membership functions. *IEEE Journal of Solid-State Circuits*, 28(7):857–861, July 1993.

[Key00a] Didier Keymeulen, Gerhard Klimeck, Ricardo Salem Zebulum, Adrian Stoica, and Carlos Salazar-Lazaro. Ehwpack: A parallel software/hardware environment for evolvable hardware. In Whitley Darrell, editor, *Proc. of the Genetics and Evolutionary Computation Conference (GECCO-2000)*, pages 538–539, Las Vegas, Nevada, USA, July 2000. Morgan Kaufmann.

[Key00b] Didier Keymeulen, Adrian Stoica, Ricardo Salem Zebulum, and Vu Duong. Results on the fitness and population based fault tolerant approaches using a reconfigurable electronic device. In *Proc. of the Congress on Evolutionary Computation*, pages 537–540, San Diego, CA, USA, July 2000. IEEE Press.

[Key04] Didier Keymeulen, Ricardo Salem Zebulum, Vu Duong, Xin Guo, Ian Ferguson, and Adrian Stoica. High temperature experiments for circuit self-recovery. In Kalyanmoy Deb et. al. editor, *Proc. of the Genetic and Evolutionary Computation Conference (GECCO 2004), Part I*, pages 792–803, Seattle, WA, USA, Jun 2004. Springer-Verlag, LNCS 3102. ISBN 3-540-22344-4.

[Kim03] Joonwon Kim, Wnjiang Shen, Laurent Latorre, and C. J. Kim. Mercury droplet microswitch for re-configurable circuit interconnect. In *Proc. of the 12th Int. Conf. Solid-State Sensors and Actuators and Microsystems (Transducers'03)*, pages 464–467, Boston, MA, USA, June 2003.

[Koz90] John R. Koza. Genetic programming: a paradigm for genetically breeding populations of computer programs to solve problems. Technical report, Computer Science Department, Stanford University, Margaret Jacks Hall, Stanford, CA 94305, June 1990.

[Koz96a] John R. Koza, David Andre, Forrest H. Bennett III, and Martin A. Keane. Design of a 96 decibel operational amplifier and other problems for which a computer program evolved by genetic programming is competitive with human performance. In Mitsuo Gen and Weixuan Zu, editors, *Proc. of l996 Japan-China Joint International Workshop on Information Systems*, pages 30–49, Ashikaga, Japan, Jul 1996.

[Koz96b] John R. Koza, David Andre, Forrest H. Bennett III, and Martin A. Keane. Evolution of a low-distortion, low-bias 60 decibel op amp with good frequency generalization using genetic programming. In J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, editors, *Genetic Programming 1996: Proc. 1st Annual Conf. (GP96)*, pages 94–100, Stanford, CA, USA, Jul 1996. Cambridge, MA: MIT Press.

[Koz96c] John R. Koza, Forrest H Bennett III, Andre David, and Martin A. Keane. Automated wywiwyg design of both the topology and component values of electrical circuits using genetic programming. In John R. Koza, David E. Goldberg, David B. Fogel, and Rick L. Riolo, editors, *Proceedings of the First Annual Conference on Genetic Programming (GP 96)*, pages 123–131, Stanford, CA, USA, Jul 1996. Cambridge, MA: MIT Press.

[Koz96d] John R. Koza, Forrest H Bennett III, Andre David, and Martin A. Keane. Four problems for which a computer program evolved by genetic programming is competitive with human performance. In *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation*, pages 1–10, 1996.

[Koz99a] John R Koza, Forrest H Bennett III, David Andre, and Martin A. Keane. *Genetic Programming III*. Morgan Kaufmann Publishers, Inc., 1999.

[Koz99b] John R Koza, Forrest H Bennett III, David Andre, and Martin A. Keane. *Genetic Programming III*, chapter 25–37, pages 391–684. Morgan Kaufmann Publishers, Inc., 1999.

[Koz99c] John R Koza, Forrest H Bennett III, David Andre, and Martin A. Keane. *Genetic Programming III*, chapter 31, pages 593–612. Morgan Kaufmann Publishers, Inc., 1999.

[Koz99d] John R Koza, Forrest H Bennett III, David Andre, and Martin A. Keane. *Genetic Programming III*, chapter 44, pages 771–785. Morgan Kaufmann Publishers, Inc., 1999.

[Koz99e] John R Koza, Forrest H Bennett III, David Andre, and Martin A. Keane. *Genetic Programming III*, chapter 45, pages 787–802. Morgan Kaufmann Publishers, Inc., 1999.

[Koz99f] John R. Koza, Forrest H Bennett III, David Andre, and Martin A. Keane. *Synthesis of a MOSFET Circuit*, chapter 51, pages 861–874. Morgan Kaufmann Publishers, Inc. San Francisco, CA, USA, Orlando, Florida, USA, 1999. ISBN 1-55860-543-6.

[Koz04a] John R. Koza. Introduction to genetic programming. Genetic and Evolutionary Computation Conference (GECCO 2004): Tutorial Program, jun 2004.

[Koz04b] John R Koza, Lee W. Jones, Martin A. Keane, Matthew J. Streeter, and Sameer H. Al-Sakran. *Toward automated design of industrial-strength analog circuits by means of genetic programming*, volume Genetic Programming Theory and Practice II, chapter 8, pages 121–142. Kluwer Academic Publishers, Boston, MA, USA, 2004.

[Koz04c] John R. Koza, Martin A. Keane, and Matthew J. Streeter. Routine high-return human-competitive evolvable hardware. In *Proc. of the 2004 NASA/DoD Conference on Evolvable Hardware*, pages 3–17, Seattle, WA, USA, June 2004. IEEE Press. ISBN: 0-7695-2145-2.

[Kru96] Marinus W. Kruiskamp. *Analog Design Automation Using Genetic Algorithms and Polytopes*. PhD thesis, Technische Universiteit Eindhoven, Department of Electrical Engineering, Sep 1996. ISBN: 90-386-0150-6.

[Kuh03] William B. Kuhn, Dan Nobbe, Dylan Kelly, and Aaron W. Orsborn. Dynamic range performance of on-chip RF bandpass filters. *IEEE Transaction on Circuits and Systems – II*, 50(10):685–694, Oct 2003.

[Lak94a] Kenneth R. Laker and Willy M. C. Sansen. *Design of Analog Integrated Circuits and Systems*. McGraw-Hill, Inc. 1 edition, 1994. ISBN: 0-07-113458-1.

[Lak94b] Kenneth R. Laker and Willy M. C. Sansen. *Design of Analog Integrated Circuits and Systems*, chapter 7, pages 1–91. McGraw-Hill, Inc. 1 edition, 1994. ISBN: 0-07-113458-1.

[Lak94c] Kenneth R. Laker and Willy M. C. Sansen. *Design of Analog Integrated Circuits and Systems*, chapter 7, pages 648–757. McGraw-Hill, Inc. 1 edition, 1994. ISBN: 0-07-113458-1.

[Lak94d] Kenneth R. Laker and Willy M. C. Sansen. *Design of Analog Integrated Circuits and Systems*, chapter 7, page 654. McGraw-Hill, Inc. 1 edition, 1994. ISBN: 0-07-113458-1.

[Lak94e] Kenneth R. Laker and Willy M. C. Sansen. *Design of Analog Integrated Circuits and Systems*, chapter 8, pages 758–887. McGraw-Hill, Inc. 1 edition, 1994. ISBN: 0-07-113458-1.

[Lan98] Jörg Langeheine. Entwurf und test eines optisch konfigurierbaren widerstandsnetzwerkes. Diploma Thesis HD-IHEP98-10, Ruperto-Carola University of Heidelberg, Kirchhoff-Institute for Physics, INF 227, 69120 Heidelberg, Germany, Oct 1998.

[Lan01] Jörg Langeheine, Joachim Becker, Simon Fölling, Karlheinz Meier, and Johannes Schemmel. Initial studies of a new VLSI field programmable transistor array. In Yong Liu, Tanaka Kiyoshi, Iwata Masaya, Tetsuya Higuchi, and Moritoshi Yasunaga, editors, *Proc. 4th Int. Conf. on Evolvable Systems: From Biology to Hardware*, pages 62–73, Tokio, Japan, October 2001. Springer Verlag.

[Lan02] W.B. Langdon et al., editors. *Genetic and Evolutionary Computation Conference (GECCO 2002)*, New York City, NY, USA, Jul 2002. Morgan Kaufmann Publishers. ISBN 1-55860-878-8.

[Lan03] Jörg Langeheine, Karlheinz Meier, and Johannes Schemmel. Intrinsic evolution of analog electronic circuits using a CMOS FPTA chip. In G. Bugeda, J.-A. Désidéri, J. Périaux, M. Schoenauer, and G. Winter, editors, *Proc. of the 5th Conf. on Evolutionary Methods for Design, Optimization and Control (EUROGEN 2003)*, pages 87–88, Barcelona, Spain, September 2003. IEEE Press. Published on CD: ISBN: 84-95999-33-1.

[Lan04] Jörg Langeheine, Martin Trefzer, Daniel Brüderle, Karlheinz Meier, and Johannes Schemmel. On the evolution of analog electronic circuits using building blocks on a CMOS FPTA. In Kalyanmoy Deb et. al. editor, *Proc. of the Genetic and Evolutionary Computation Conference (GECCO 2004), Part I*, pages 1316–1327, Seattle, WA, USA, Jun 2004. Springer-Verlag, LNCS 3102. ISBN 3-540-22344-4.

[Lat00] Lattice Semiconductor Corp. 5555 Northeast Moore Ct. Hillsboro, OR, USA. *Specifications ispPAC 10 – In-System Programmable Analog Circuit*, Sep 2000.

[Lat01] Lattice Semiconductor Corp. 5555 Northeast Moore Ct. Hillsboro, OR, USA. *Specifications ispPAC – In-System Programmable Analog Circuit*, Oct 2001.

[Lay01] Paul Layzell. *Hardware Evolution: On the Nature of Artificially Evolved Electronic Circuits*. PhD thesis, University of Sussex, May 2001. http://www.cogs.susx.ac.uk/users/adrianth/lazwebpag/web/Publications/THESIS/ssxdphil.pdf.

[Lia01] Jianming Liang, Trent McConaghy, Alexandre Kochlan, Tuan Pham, and Glen Hertz. Intelligent systems for analog cirucit design automation: A survey. 2001. http:://www.analogsynthesis.com.

[Lin98] Shang-Yi Lin, Ren-Jiun Huang, and Tzi-Dar Chiueh. A tunable gaussian/square function computation circuit for analog neural networks. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 45(3):441–446, March 1998.

[Lin00a] Saska Lindfors, Kari Halonen, and Mohammed Ismail. A 2.7-v elliptical mosfet-only g_mc-ota filter. *IEEE Transaction on Circuits and Systems – II*, 47(2):89–95, Feb 2000.

[Lin00b] Linear Technology Corporation, 1630 McCarthy Blvd., Milpitas, CA 95035-7417, USA. *LT1809/1810 Data Sheet Single/Dual 180MHz,*350 $\frac{V}{\mu s}$ *Rail-to-Rail Input and Output Low Distortion Op Amps*, 2000.

[Loh99] Jason Lohn and Silvano P. Colombano. A circuit representation technique for automated circuit design. *IEEE Trans. on Evolutionary Computation*, 3:205–219, September 1999.

[Loo99] Markus Loose. *A Self-Calibrating CMOS Image Sensor with Logarithmic Response*. PhD thesis, Ruperto-Carola University of Heidelberg, Oct 1999.

[Mat] The MathWorks Inc. 3 Apple Hill Drive, Natick, MA, USA, http://www.mathworks.com. *MATLAB: Signal Processing Toolbox 6.0, User's Guide*, release 13 edition.

[Max] Maxim Integrated Products, Inc. – Website. http://www.maxim-ic.com.

[Max96] Maxim Integrated Products, Inc. 120 San Gabriel Drive. *MAX274/MAX275 Datasheet*, 3 edition, Oct 1996.

[Max00] Maxim Integrated Products, 120 San Gabriel Drive, Sunnyvale, CA, USA. *MAX4430-MAX4433 Data Sheet:Dual-Supply, 180MHz, 16-Bit Accurate, Ultra-Low Distortion Op Amps*, July 2000.

[Max01] Maxim Integrated Products, Inc. San Gabriel Drive, Sunnyvale, CA, USA. *Application Note 728: Defining and Testing Dynamic Parameters in High-Speed ADCs, Part 1*, Feb 2001. http://www.maxim-ic.com.

[Max02] Maxim Integrated Products, Inc. 120 San Gabriel Drive. *MAX260/MAX261/MAX262 Datasheet*, 2 edition, Jul 2002.

[Mea91] Carver A. Mead. *Analog VLSI and Neural Systems*. Addison Wesley, 1991.

[Mic99] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programms*. Springer Verlag, Berlin, Heidelberg, New York, 1999.

[Mil85] Miran Milkovic. Current gain high-frequency CMOS operational amplifiers. *IEEE Journal of Solid-State Circuits*, 20(4):845–851, August 1985.

[Mil87a] Jacob Millman and Arvin Grabel. *Microelectronics*. McGraw-Hill Series in Electrical Engineering. McGraw-Hill, Inc. 2 edition, 1987. ISBN: 0-07-100596-X.

[Mil87b] Jacob Millman and Arvin Grabel. *Microelectronics*, chapter 16, page 733. McGraw-Hill Series in Electrical Engineering. McGraw-Hill, Inc. 2 edition, 1987. ISBN: 0-07-100596-X.

[Mil87c] Jacob Millman and Arvin Grabel. *Microelectronics*, chapter 16, pages 706–769. McGraw-Hill Series in Electrical Engineering. McGraw-Hill, Inc. 2 edition, 1987. ISBN: 0-07-100596-X.

[Mil97] Julian Francis Miller, Peter Thomson, and Terence Fogarty. Designing electronic circuits using evolutionary algorithms. arithmetic circuits: A case study. In D. Quagliarella, J. Périaux, C. Poloni, and G. Winter, editors, *Genetic Algorithms and Evolution Strategy in Engineering and Computer Science*, chapter 6, pages 105–131. John Wiley and Sons, Chichester, Nov 1997.

[Mil02] Julian Francis Miller and Keith Downing. Evolution in materio: Looking beyond the silicon box. In *Proc. of the Fourth NASA/DOD Workshop on Evolvable Hardware*, pages 167–176, Alexandria, VA, USA, July 2002. IEEE Press.

[Moo97] David S. Moore. *The Basic Practice of Statistics*, pages 111–117. W. H. Freeman and Company, USA, 1997. ISBN: 0-7167-2628-9.

[Mur98] Masahiro Murakawa, Shuji Yoshizawa, Toshio Adachi, Shiro Suzuki, Kaoru Takasuka, Masaya Iwata, and Tetsuya Higuchi. Analogue EHW chip for intermediate frequency filters. In Moshe Sipper, Daniel Mange, and Andrés Pérez-Uribe, editors, *Proc. 2nd Int. Conf. on Evolvable Systems: From Biology to Hardware*, pages 134–143, Lausanne, Switzerland, September 1998. Springer-Verlag.

[Mur03] Masahiro Murakawa, Toshio Adachi, Yoshihiro Niino, Yuji Kasai, Eiichi Takahashi, Kaoru Takasuka, and Tetsuya Higuchi. An ai-calibrated if filter: A yield enhancement mehthod with area and power dissipation reductions. *IEEE Journal of Solid-State Circuits*, 38(3):495–502, March 2003.

[Nat99] National Semiconductor Corporation, , USA. *LM6161/LM6261/LM6361/High Speed Operational Amplifier*, May 1999.

[Nat04] 931?945 (2004). | Article | Nature 431, 931 945 (2004). International Human Genome Sequencing Consortium Nature 431. Finishing the euchromatic sequence of the human genome. *Nature*, (431):931–945, October 2004.

[Nic98] Bradford Nichols, Dick Buttlar, and Jaqueline Proulx Farrell. *Pthreads Programming*. O'Reilly & Associates, Inc., 101 Morris Street, Sebastopol, CA, February 1998.

[Nie95] Ivan Riis Nielsen. A c-t filter compiler – from specification to layout. *Analog Integrated Circuits and Signal Processing*, 7:21–31, 1995.

[Opp83] Alan V. Oppenheim, Alan S. Willsky, and Ian T. Young. *Signals and Systems*. Prentice-Hall Signal Processing Series. Prentice-Hall International, In. London, GB, 1 edition, 1983. ISBN: 0-13-811175-8.

[Ozs98] Ian Ozsvald. Short-circuiting the design process: Evolutionary algorithms for circuit design using reconfigurable analogue hardware. Master's thesis, University of Sussex, Sep 1998. URL: http://www.cogs.susx.ac.uk/users/adrianth/IAN_OZSVALD/ian.html.

[Pan02] Bogdan Pankiewicz, Marek Wojcikowski, Stanislaw Szczepanski, and Yichuang Sun. A field programmable analog array for CMOS continuous-time ota-c filter applications. *IEEE Journal of Solid-State Circuits*, 37(2):125–135, Feb 2002.

[Pav00] Shanthi Pavan, Yannis P. Tsividis, and Krishnaswamy Nagaraj. Widely programmable high-frequency continuous-time filters in digital CMOS technology. *IEEE Journal of Solid-State Circuits*, 35(4):503–511, Apr 2000.

[PCI95] PCI Special Interest Group. *PCI Local Bus Specification, Revision 2.1*, Dec 1995.

[Phe00a] Rodney Phelps, Michael J. Krasnicki, Rob A. Rutenbar, L. Richard Carley, and James R. Hellums. Anaconda: Simulation-based synthesis of analog circuits via stochastic pattern search. *IEEE Transactions on Compuater-Aided Design of Integrated Circuits and Systems*, 19(6):703–717, June 2000.

[Phe00b] Rodney Phelps, Michael J. Krasnicki, Rob A. Rutenbar, L. Richard Carley, and James R. Hellums. A case study of synthesis for industrial-scale analog IP: Redesign of the equalizer/filter frontend for an ADSL CODEC. In *Proc. ACM/IEEE Design Automation Conference*, pages 1–6, Los Angeles, CA, USA, June 2000.

[PLX00] PLX Technology, Inc., Sunnyvale. *PLX 9054 Data Book*, version 2.1 edition, January 2000.

[Pre01] Ronald P. Preston. *Register Files and Caches*, chapter 14, pages 285–308. IEEE Press, 445 Hoes Lane, P. O. Box 1331 Piscataway, NJ, USA, 1 edition, 2001. ISBN:0-7803-6001-X.

[Pyt01] Dominique Python and Christian C. Enz. A micropower class-AB CMOS log-domain filter for DECT applications. *IEEE Journal of Solid-State Circuits*, 36(7):1067–1075, Jul 2001.

[Qua94] T. Quarles, A. R. Newton, D. O. Pederson, and Sangiovanni-Vincentelli. *SPICE 3 Version 3F5 User's Manual*. Dept. of Elec. Eng. Comput. Sci. Univ. of California, Berkeley, CA, USA, 1994.

[Ray01] Eric T. Ray. *Learning XML*. O'Reilly & Associates, Inc., 101 Morris Street, Sebastopol, CA, 2001.

[Ray02] Baidya Nath Ray, P.P̃al Chaudhuri, and P. K. Nandi. Efficient synthesis of ota network for linear analog functions. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 21(5):517–533, May 2002.

[Rhe92] Dietrich Rhein and Heinz Freitag. *Mikroelektronische Speicher*. Springer-Verlag/Wien, 1 edition, 1992. ISBN: 3-540-56853-0.

[Rud97] G. Rudolph. *Convergence Properties of Evolutionary Algorithms*. Verlag Dr. Kovač, Hamburg, 1997.

[Rup93] Werner Rupprecht. *Signale und Übertragungssysteme*. Springer-Lehrbuch. Springer-Verlag, 1 edition, 1993. ISBN: 3-540-56853-0.

[San01] Christina Costa Santini, Ricardo Salem Zebulum, Marco Aurélio C. Pacheco, Marley Maria R. Vellasco, and Moisés H. Szwarcman. Pama – programmable analog multiplexer array. In *Proc. of the Third NASA/DOD Workshop on Evolvable Hardware*, pages 36–43, Long Beach, CA, USA, July 2001. IEEE Press.

[Sar91] Rahul Sarpeshkar, John L. Wyatt, Jr. C. Lu, Nicky, and Porter D. Gerber. Mismatch sensitivity of a simultaneously latched CMOS sense amplifier. *IEEE Journal of Solid-State Circuits*, 26(10):1413–1422, Oct 1991. Mismatch of Dynamic latch used as sense amplifier for.

[Sas89] Katsuro Sasaki, Ishibashi Koichiro, Toshiaki Yamanaka, Naotaka Hashimoto, Takashi Nishida, Shimohigashi Katsuhiro, Shoji Hanamura, and Shigeru Honjo. A 9-ns 1-Mbit CMOS SRAM. *IEEE Journal of Solid-State Circuits*, 24(5):1219–1225, Oct 1989. 9 ns access time for SRAM sensing: 3-staged Sense-Amp: 1st stage = PMOS Cross-Coupled Amplifier, 0.5 um process Sense delay = 4.5 ns.

[Sch99] Johannes Schemmel. *An Integrated Analog Network for Image Processing*. PhD thesis, Ruperto-Carola University of Heidelberg, Oct 1999.

[Sch03] T. Schmitz, S. Hohmann, K. Meier, J. Schemmel, and F. Schürmann. Speeding up Hardware Evolution: A Coprocessor for Evolutionary Algorithms. In Andy M. Tyrrell, Pauline C. Haddow, and Jim Torresen, editors, *Proceedings of the 5th International Conference on Evolvable Systems ICES 2003*, pages 274–285. Springer Verlag, 2003.

[Sch05] Felix Schürmann. Exploring liquid computing in a hardware adaptation: Construction and operation of a neural network experiment. *PhD thesis*, University of Heidelberg, submitted, 2005.

[Sed91a] Adel S. Sedra and Kenneth C. Smith. *Microelectronic Circuits*, chapter 11, pages 762–840. Oxford University Press, 198 Madison Avenue, New York, New York 10016, 3 edition, 1991. ISBN: 0-19-510369-6.

[Sed91b] Adel S. Sedra and Kenneth C. Smith. *Microelectronic Circuits*, chapter 11, page 767. Oxford University Press, 198 Madison Avenue, New York, New York 10016, 3 edition, 1991. ISBN: 0-19-510369-6.

[Sek03] Lukáš Sekanina and Richard Růžička. Easily testable image operators: The class of circuits where evolution beats engineers. In *Proc. of the 2003 NASA/DoD Conference on Evolvable Hardware*, pages 135–144, Chicago IL, USA, July 2003. IEEE Press. ISBN: 0-7695-1977-6.

[Sha02] Akshat H Shah, Stephane Dugalleix, and Francois Lemery. Technology migration of a high performance CMOS amplifier using an automated fron-to-back analog design flow. Conference and Exhibition on Design Automation & Test in Europe (DATE 2002), March 2002.

[She84] Bing J. Sheu and Chenming Hu. Switch-induced error voltage on a switched capacitor. *IEEE Journal of Solid-State Circuits*, 19(4):525–519, August 1984.

[She02] Kevin Michael Sheehan. *Evolving Analogue Electronic Signal Processing Circuit Behaviour in Hardware*. PhD thesis, Royal Holloway University of London, 2002. Graduate Department of Physics.

[Shi00] R. Shipman, M. Shackleton, and I. Harvey. The use of neutral genotype-phenotype mappings for improved evolutionary search. *BT Technology Journal*, 18(4):103–111, October 2000. ISSN 1358-3948.

[Shi01] Hajime Shibata. *Computer-Aided Design of Analog Circuits Based on Genetic Algorithm*. PhD thesis, Tokyo Institute of Technology, 2001.

[Shi02] Hajime Shibata, Soji Mori, and Nobuo Fujii. Automated design of analog circuits using cell-based structure. In *Proc. of the Fourth NASA/DOD Workshop on Evolvable Hardware*, pages 85–92, Alexandria, VA, USA, July 2002. IEEE Press.

[Shu96] Guo Shuwei, Liliane Peters, and Hartmut Surmann. Design and application of an analog fuzzy logic controller. *IEEE Transactions on Fuzzy Systems*, 4(5):429–438, November 1996.

[Sil93] L. Silburt, Allan, Richard S. Philips, Randall G. F. Gibson, Steven W. Wood, Armin G. Bluschke, James S. Fujimoto, Stephen P. Kornachuk, Benoit Nadeau-Dostie, Rajesh K. Verma, and Peter M.J̃. Diedrich. A 180-MHz 0.8-μm BiCMOS modular memory family of DRAM and multiport SRAM. *IEEE Journal of Solid-State Circuits*, 28(3):222–232, Mar 1993. 1. Generic family of D- and SRAM -> a schema/blueprint ? 2. Self Timing of RAM access by tracking time-critical signals -> asynchronous operation 3. Use dynamic latch as Sense-Amp (only NMOS power switch) 4. PreCharge to Vdd 5. Dummy Inverter as sense amp load to balance the capacitive load.

[Sin01] Patrick P. Siniscalchi, Jeanne K. Pitz, Richard K. Hester, Stewart M. DeSoto, Minsheng Wang, Sucheendran Sridharan, Robert L. Halbach, Donald Richardson, William Bright, Maher M. Sarraj, James R. Hellums, Christopher L. Betty, and Glenn Westphal. A CMOS ADSL codec for central office applications. *IEEE Journal of Solid-State Circuits*, 36(3):356–365, Mar 2001.

[Sri02] Thanwa Sripramong and Christofer Toumazou. The invention of CMOS amplifiers using genetic programming and current-flow analysis. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 21(11):1237–1252, November 2002.

[Sta00] Werner A. Stahel. *Statistische Datenanalyse*, pages 36–42. Friedr. Vieweg & Sohn Verlagsgesellschaft mbH, Braunschweig/Wiesbaden, 3^{rd} edition, 2000.

[Sto96] Adrian Stoica. Reconfigurable transistor array for evolvable hardware. In *Caltech/JPL Novel Technology Report*, July 1996.

[Sto98] Adrian Stoica. Evolutionary technique for automated synthesis of electronic circuits. In *Caltech/JPL Novel Technology Report*, July 1998.

[Sto99] Adrian Stoica, Gerhard Klimeck, Carlos Salazar-Lazaro, Didier Keymeulen, and Anil Thakoor. Evolutionary design of electronic devices and circuits. In *Proc. of the Congress on Evolutionary Computation*, pages 1271–1278, Washington DC, USA, July 1999. IEEE Press.

[Sto00a] Adrian Stoica, Didier Keymeulen, Vu Duong, and Carlos Salazar-Lazaro. Automatic synthesis and fault-tolerant experiments on an evolvable hardware platform. In *Proc. of the IEEE Aerospace Conference*, Big Sky, MT, USA, March 2000. 0-7803-5846-5/00.

[Sto00b] Adrian Stoica, Didier Keymeulen, Ricardo Salem Zebulum, Anil Thakoor, T. Daud, Gerhard Klimeck, Yili Jin, R. Tawel, and Vu Duong. Evolution of analog circuits on field programmable transistor arrays. In *Proc. of the Second NASA/DOD Workshop on Evolvable Hardware*, pages 99–108, Palo Alto, CA, USA, July 2000. IEEE Press.

[Sto01a] Adrian Stoica, Didier Keymeulen, and Ricardo Salem Zebulum. Evolvable hardware solutions for extreme temperature electronics. In *Proc. of the Third NASA/DOD Workshop on Evolvable Hardware*, pages 93–97, Long Beach, CA, USA, July 2001. IEEE Press.

[Sto01b] Adrian Stoica, Ricardo Salem Zebulum, and Didier Keymeulen. Mixtrinsic evolution. In *Proc. ICES 2000, LNCS 1801*, pages 208–217, Edinburgh, UK, April 2001. Springer Verlag.

[Sto01c] Adrian Stoica, Ricardo Salem Zebulum, and Didier Keymeulen. Progress and challenges in building evolvable devices. In *Proc. of the Third NASA/DOD Workshop on Evolvable Hardware*, pages 33–35, Long Beach, CA, USA, July 2001. IEEE Press.

[Sto01d] Adrian Stoica, Ricardo Salem Zebulum, Didier Keymeulen, Raoul Tawel, Taher Daud, and Anil Thakoor. Reconfigurable vlsi architectures for evolvable hardware: From experimental field programmable transistor arrays to evolution-oriented chips. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 9(1):227–232, February 2001.

[Sto01e] Volker Storch, Ulrich Welsch, and Michael Wink. *Evolutionsbiologie*. Springer Verlag, Berlin, Heidelberg, New York, 2001.

[Sto02a] Adrian Stoica, Didier Keymeulen, Ricardo Salem Zebulum, Ian Ferguson, and Xin Guo. On two new trends in evolvable hardware: Employment of hdl-based structuring, and design of multi-fuctional circuits. In *Proc. of the Fourth NASA/DOD Workshop on Evolvable Hardware*, pages 56–59, Alexandria,Virginia, USA, July 2002. IEEE Press.

[Sto02b] Adrian Stoica, Ricardo Salem Zebulum, Ian Ferguson, Didier Keymeulen, and Vu Duong. Evolving circuits in seconds: Experiments with a stand-alone board level evolvable system. In *Proc. of the Fourth NASA/DOD Workshop on Evolvable Hardware*, pages 67–74, Alexandria,Virginia, USA, July 2002. IEEE Press.

[Sto04] Adrian Stoica, Didier Keymeulen, Tughrul Arslan, Vu Duong, Ricardo Zebulum, Ian Ferguson, and Xin Guo. Circuit self-recovery experiments in extreme environments. In *Proc. of the 2004 NASA/DoD Conference on Evolvable Hardware*, pages 142–145, Seattle, WA, USA, June 2004. IEEE Press. ISBN: 0-7695-2145-2.

[Syn01] Synopsys, Inc., Mountain View. *FPGA Compiler II*, 2001.08-fc3.6 edition, June 2001.

[Syn05] Synopsys, Inc., Mountain View. *Synopsys Circuit Explorer Datasheet*, June 2005.

[Sza96] Kenneth S. Szajda, Charles G. Sodini, and H.F̃rederick Bowman. A low noise, high resolution silicon temperature sensor. *IEEE Journal of Solid-State Circuits*, 31(9):1308–1313, September 1996.

[Tho97] A. Thompson. An evolved circuit, intrinsic in silicon, entwined with physics. In Tetsuya Higuchi, Masaya Iwata, and L. Weixin, editors, *Proc. 1st Int. Conf. on Evolvable Systems (ICES'96)*, volume 1259 of *LNCS*, pages 390–405. Springer-Verlag, 1997.

[Tho98a] Adrian Thompson. *Hardware Evolution: Automatic design of electronic circuits in reconfigurable hardware by artificial evolution*. Distinguished dissertation series. Springer-Verlag, 1998. ISBN: 3-540-76253-1.

[Tho98b] Adrian Thompson. On the automatic design of robust electronics through artificial evolution. In *Proc. 2nd Int. Conf. on Evolvable Systems: From Biology to Hardware, ICES 1998, LNCS 1478*, pages 13–24, Lausanne, Switzerland, September 1998. Springer Verlag.

[Tho99] Adrian Thompson, Paul Layzell, and Ricardo Salem Zebulum. Explorations in design space: Unconventional electronics design through artificial evolution. *IEEE Trans. on Evolutionary Computation*, 3:167–196, September 1999.

[Tho00] A. Thompson and P. Layzell. Evolution of robustness in an electronics design. In J. Miller, A. Thompson, P. Thomson, and T. Fogarty, editors, *Proc. 3rd Int. Conf. on Evolvable Systems (ICES2000): From biology to hardware*, volume 1801 of *LNCS*, pages 218–228. Springer-Verlag, 2000.

[Tim76] Michael P. Timko. A two-terminal IC temperature transducer. *IEEE Journal of Solid-State Circuits*, 11(6):784–788, December 1976.

[Tou02] Chris Toumazou. Intuitive analog circuit design. In Chris Toumazou, George Moschytz, and Barrie Gilbert, editors, *Trade-Offs in Analog Circuit Design*, chapter 1, pages 1–6. Kluwer Academic Publishers, P.O. Box 17, 3300 AA Dordrecht , The Netherlands, 2002. ISBN: 1-4020-7037-3.

[Tre03] Martin Trefzer. Internal status report for the electronic vision(s) group meeting. Oral Presentation, April 2003.

[Tre04] Martin Trefzer, Jörg Langeheine, Johannes Schemmel, and Karlheinz Meier. New genetic operators to facilitate understanding of evolved transistor circuits. In *Proc. of the 2004 NASA/DoD Conference on Evolvable Hardware*, pages 217–224, Seattle, WA, USA, June 2004. IEEE Press. ISBN: 0-7695-2145-2.

[Tre05] Martin Trefzer, Jörg Langeheine, Karlheinz Meier, and Johannes Schemmel. Operational amplifiers: An example for multi-objective optimization on an analog evolvable hardware platform. In *Submitted to: the 6th International Conference on Evolvable Systems ICES 2005*, Barcelona, Spain, September 2005. Springer Verlag.

[Tro] Trolltech AS. The Qt application development framework. Waldemar Thranes gate, 98, NO-0175 Oslo, Norway, `http://www.trolltech.com/products/qt/`.

[Tut98] Mike Tuthill. A switched-current, switched-capacitor temperature sensor in 0.6-μm CMOS. *IEEE Journal of Solid-State Circuits*, 33(7):1117–1122, July 1998.

[Vie04] Pedro F. Vieira, Leonardo B. Sá, and João P. B. Botelho. Evolutionary synthesis of analog circuits using only mos transistors. In *Proc. of the Fourth NASA/DOD Workshop on Evolvable Hardware*, pages 38–45, Seattle, WA, USA, June 2004. IEEE Press.

[Weg] Ingo Wegener. Spezialvorlesung Evolutionäre Algorithmen, lecture notes, University of Dortmund, summer term 2002.

[Wei02] Karsten Weicker. *Evolutionäre Algorithmen*. Leitf"aden der Informatik. B. G. Teubner, Stuttgart, Germany, 2002. ISBN: 3-519-00362-7.

[Wil93] Scott D. Willingham, Kenneth W. Martin, and A. Ganesan. A BiCMOS low-distortion 8-mhz low-pass filter. *IEEE Journal of Solid-State Circuits*, 28(12):1234–1245, Dec 1993.

[Wol97] David H. Wolpert and William G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, April 1997.

[Xil] Xilinx, Inc., 2100 Logic Drive, San Jose, CA 95124-3400, USA. *Virtex E Datasheet*.

[Xil03] Xilinx, Inc., San Jose. *ISE Quick Start Tutorial 6.x*, 2003.

[Xil04] Xilinx Inc., San Jose. *Xilinx XC9536XL High Performance CPLD*, September 2004.

[Zeb98a] Ricardo Salem Zebulum, Marco Aurélio Pacheco, and Marley Vellasco. Analog circuits evolution in extrinisic and intrinsic modes. In *Proc. ICES 1998, LNCS 1478*, pages 154–165, Lausanne, Switzerland, September 1998. Springer Verlag.

[Zeb98b] Ricardo Salem Zebulum, Marco Aurélio Pacheco, and Marley Vellasco. Comparison of different evolutionary methodologies applied to electronic filter design. In *Proc. of the IEEE International World Congress on Computational Intelligence, ICEC?98*, pages 434–439, Alaska, USA, May 1998. IEEE Press.

[Zeb98c] R.S. Zebulum, M. Vellasco, and M.A. Pacheco. Evolutionary design of logic gates. In *Proc. of the Workshop in Evolutionary Design, AID98*, pages 12–17, Lisbon, July 1998.

[Zeb99] Ricardo Salem Zebulum, Marco Aurélio Pacheco Pacheco, and Marley Vellasco. Artificial evolution of active filters: A case study. In *Proc. of the First NASA/DOD Workshop on Evolvable Hardware*, pages 66–75, Pasadena, CA, USA, July 1999. IEEE Press.

[Zeb00a] Ricardo Salem Zebulum, Marco Aurélio Pacheco Pacheco, and Marley Vellasco. Variable length representation in evolutionary electronics. *Evolutionary Computation Journal*, 8(1):93–120, 2000.

[Zeb00b] Ricardo Salem Zebulum, Adrian Stoica, and Didier Keymeulen. A flexible model of a CMOS field programmable transistor array targeted for hardware evolution. In J. Miller, A. Thompson, P. Thomson, and T. C. Fogarty, editors, *Proc. of the Third Int. Conference on Evolvable Systems: From Biology to Hardware (ICES2000), LNCS 1801*, pages 274–283, Edinburgh, UK, April 2000. Springer. ISBN 3-540-67338-5 LNCS 1801.

[Zeb01] Ricardo Salem Zebulum, Adrian Stoica, and Didier Keymeulen. Experiments on the evolution of digital to analog converters. In *Proc. of the IEEE Aerospace Conference*, Montana, USA, March 2001. ISBN: 0-78-3-6600-X (Published in CD).

[Zeb02] Ricardo Salem Zebulum, Adrian Stoica, Didier Keymeulen, Ian Ferguson, Vu Duong, and Taher Daud. Current mode circuits: Increeasing the chances of evolution to find a way. In *Proc. of the IEEE Aerospace Conference*, Big Sky, Montana, USA, March 2002.

[Zeb03] Ricardo Salem Zebulum, Didier Keymeulen, Vu Duong, Guo Xin, M.I. Ferguson, and Adrian Stoica. Experimental results in evolutionary fault-recovery for field programmable analog devices. In Jason Lohn, Ricardo Zebulum, James Steincamp, Didier Keymeulen, and Michael I. Stoica, Adrian an Ferguson, editors, *Proc. of the Fifth NASA/DOD Workshop on Evolvable Hardware*, pages 182–186, Chicago, IL, USA, July 2003. IEEE Press. ISBN 0-7695-1977-6.

[Zeb04] Ricardo S. Zebulum, Didier Keymeulen, Vu Duong, Xin Guo, Ian Feruson, and Adrian Stoica. High temperature experiments using programmable transistor array. In *Proc. of the IEEE Aerospace Conference*, Big Sky, MO, USA, March 2004.

[Zet99] Zetex Semiconductors, Fields New Road, Lansdowne Road and Zetex Technology Park. *Totally Re-Configurable Analog Circuit – TRAC*, Mar 1999.

[Zha04] Yang Zhang, Stephen L. Smith, and Andy M. Tyrrell. Digital circuit design using intrinsic evolvable hardware. In *Proc. of the 2004 NASA/DoD Conference on Evolvable Hardware*, pages 55–62, Seattle, WA, USA, June 2004. IEEE Press. ISBN: 0-7695-2145-2.

Danksagung

> Vision without action is a daydream. Action without vision is a nightmare.
>
> JAPANESE PROVERB

Während der Arbeit an dieser Dissertation gab es immer Menschen, die mir die notwendige fachliche und persönliche Unterstützung zukommen ließen. Ihnen möchte ich an dieser Stelle ganz herzlich danken:

- Herrn Prof. Dr. K. Meier, der mir überhaupt erst die Möglichkeit eröffnet hat, eine Arbeit mit so ungewissem Ausgang in Angriff nehmen zu dürfen,
- Herrn Prof. Dr. Norbert Herrmann, der freundlicherweise die Aufgabe des Zweitgutachters übernommen hat,
- Herrn Dr. Johannes Schemmel, der die Idee zum Projekt geliefert hat, es seitdem inhaltlich mitbetreut hat, und für die vielen Ratschlaege und Diskussionen Software, Hardware und anderen Fragen, sowie für die Bereitstellung der C++ Routingen zur Hardware Ansteuerung der PCI-Karten,
- Herrn Joachim Becker, der während seiner Diplomarbeit die PCI-Karte Darkwing entwickelt und den VHDL code für die elementare Ansteuerung der ersten Version der PCI-Karte, des "Fädelboards", geschrieben hat,
- Herrn Simon Fölling, der das Grundgeruest der DarkGAQT Software erstellt hat,
- Herrn Daniel Brüderle und Herrn Martin Trefzer, die große Teile der Implementierung des Building-Block Konzeptes übernommen haben,
- Herrn Martin Trefzer für die Mitwirkung bei der Weiterentwicklung der DarkGAQT Software, der Mitarbeit im "Tagesgeschäft" des Projektes, die Weiterfürung des Projektes, die angenehme Zusammenarbeit sowie die gute Arbeitsatmosphäre in unserem Büro,
- Herrn Felix Schürmann, für seine umfassende Hilfe in allen Linux-Lagen,
- Herrn Tillmann Schmitz für das SRAM interface und die Einführung in und die Bereitstellung der Skripte zur skriptgesteuerten Synthese des Darkwing VHDL codes,
- Herrn Stefan Philipp für die Pflege der Windows Server,
- Herrn Michael Keller und Herrn Markus Dorn, die die für ihre Betreuung der CAD Software,

- Herrn Ralf Achenbach, für seine Hilfe und Anleitung beim/zum Bonden und die Hilfestellungen bei der Benutzung des Testlabors,
- Herrn Johannes Schemmel, Herrn Steffen Hohmann, Herrn Thorsten Maucher, Herrn Andreas Breidenassel, Herrn Felix Schürmann, Herrn Simon Fölling und Herrn Martin Trefzer, da die mit ihnen geführten Fachdiskussionen wesentlich zum Gelingen dieser Arbeit beigetragen haben,
- Allen Mitarbeitern der Arbeitsgruppe Electronic Vision(s) für die gute Arbeitsatmosphäre und die große Hilfsbereitschaft,
- Allen Mitarbeitern des Kirchhoff-Instituts für Physik und denen des ehemaligen Instituts für Hochenergiephysik, die sowohl für die nötige Infrastruktur als auch das gute Betriebsklima gesorgt haben,
- Frau Silke Vogel, Frau Bettina Riegler, Herrn Moritz Diehl, Herrn Martin Trefzer und Herrn Andreas Breidenassel für die kritische Durchsicht von Teilen des Manuskriptes,
- Herrn Thorsten Schneider für seinen Beistand in biologischen Fragen,
- Allen Freunden, die mich während meiner Promotion unterstützt haben,
- Meinen Eltern, deren Hilfe vielleicht nicht so konkret, aber durch ihre bedingungslose Unterstützung meiner Ausbildung um so wichtiger war.