Microelectronics Journal ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Microelectronics Journal

journal homepage: www.elsevier.com/locate/mejo

Towards a framework for designing applications onto hybrid nano/CMOS fabrics

Catherine Dezan [a,*], Ciprian Teodorov [a], Loïc Lagadec [a], Michael Leuchtenburg [b], Teng Wang [b], Pritish Narayanan [b], Andras Moritz [b]

[a] *LAB-STICC UMR 3192, Université de Bretagne Occidentale, France*
[b] *Electrical and Computing Engineering Department, University of Massachusetts at Amherst, USA*

A R T I C L E I N F O

Article history:
Received 27 February 2008
Accepted 31 July 2008

Keywords:
CAD tool
Nanotechnology
Fault tolerance

A B S T R A C T

The design of CAD tools for nanofabrics involves new challenges not encountered with conventional design flow used for CMOS technology. In this paper, we propose to define a new framework able to help the designer to map an application on a wide range of emerging nanofabrics. Our proposal is based on a variety of models that capture as well as isolate the differences between these fabrics. This tool supports the design flow starting from behavioral description up to final layout. It integrates fault-tolerant techniques and fabric-related density transformations with more conventional design automation techniques. After an overview of common requirements, physical models, and associated techniques, a case study in the context of NASIC fabrics is used to illustrate some of the concepts.

1. Introduction

As an alternative to CMOS based designs, novel nanofabrics are being proposed based on a combination of lithographic processes and bottom-up self-assembly based manufacturing. These fabrics include NanoPLA [1,2], CMOL [3], FPNI [4], and NASIC [5]—to name a few. They are based on a variety of devices such as FETs, spin-based devices, diodes, and molecular switches. Furthermore, all these architectures include some support in CMOS: some like FPNI would move the entire logic into CMOS, others, like NASIC, would only provide the control circuitry in CMOS. The rationale for this varies but includes targeted application areas as well as manufacturability issues.

Other differences include fault handling: e.g., some proposals would use reconfigurable approaches, while others like NASICs would rely on built-in techniques based on redundancy, voting, error correction, and/or unique fabric structures. The architectures proposed range from general purpose processors to programmable logic arrays similar to FPGAs, and to more specialized devices such as cellular arrays and cellular neural networks.

In order to implement an application on a nanofabric, specific tools are already proposed by the respective research groups [1,5,6] as CAD tools are necessary to be able to design and evaluate the capabilities of larger-scale systems. As the underlying technologies are still evolving according to advances in devices, manufacturing, and fabric structures, CAD tools for nanofabrics should be made, ideally generic enough to integrate added features or to enable new paradigms as well as comparison between various approaches.

This paper proposes a prototyping CAD tool that considers an explicit specification of the underlying nanofabric. It extends the classical design flow—shown in the Fig. 1—for designing an application from behavioral specifications (e.g., in VHDL, Verilog, or SystemC) onto physical designs. It is based on a range of transformations applied at different levels of description/abstraction of the application/problem that is mapped.

The new design flow proposed incorporates a variety of models associated with the nanofabric to allow optimizations to occur on generic data structures. Through a computational model, an architectural model, a technological model, and a fault model key aspects of a particular fabric can be captured and abstracted. The proposed models interact with the behavioral and the physical tools to produce an abstract layout for the design—starting from a high-level description. Parts that are mapped to nanoscale are separated from parts that use conventional CMOS technology.

Nanoscale fabrics under consideration have the following features:

- The use of self-assembly based manufacturing techniques, e.g., nanopatterning, fluidic alignment, DNA-based self-assembly,

* Corresponding author. Tel.: +33 2 98 01 69 73.
E-mail addresses: catherine.dezan@univ-brest.fr (C. Dezan), ciprian.teodorov@univ-brest.fr (C. Teodorov), loic.lagadec@univ-brest.fr (L. Lagadec), mleuchte@ecs.umass.edu (M. Leuchtenburg), twang@ecs.umass.edu (T. Wang), pnarayanan@ecs.umass.edu (P. Narayanan), andras@ecs.umass.edu (A. Moritz).

0026-2692/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.mejo.2008.07.072

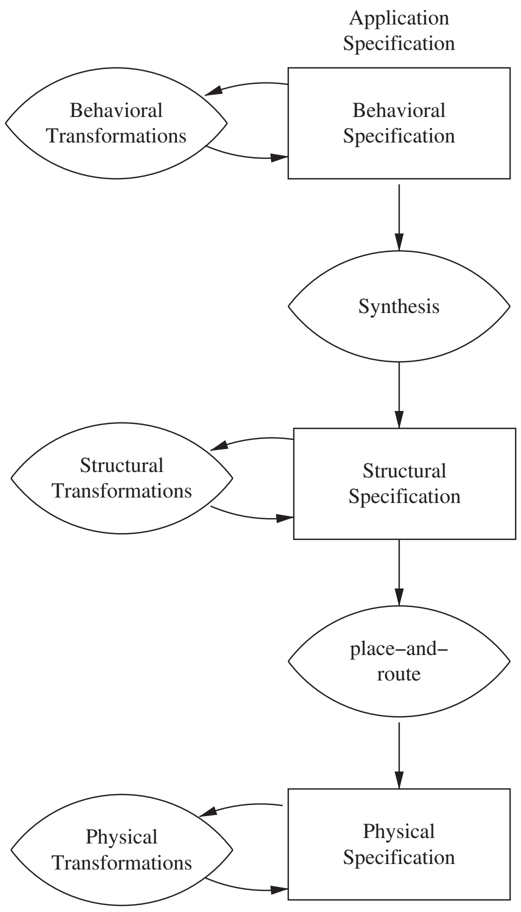


Fig. 1. Classical design flow of a conventional CAD tool.

and di-block co-polymers, in conjunction with conventional lithography: this is driving their structure to be quasi-regular such as based on 2-D crossbars.

- Nanoscale fabrics could be hybrid nano/CMOS structures as opposed to just nanoscale.
- Nanoscale fabrics are expected to have high defect rates, e.g., in the range of 10+%; thus, defect tolerance techniques need to be incorporated and taken into account in the design of any new CAD tool. In comparison, conventional CMOS designs in 90 nm technology have only 0.4 defects per cm^2.

The need to build a new framework able to support large applications based on emerging fabrics is apparent. While not all of the physical constraints are finalized, investigations have began and significant progress is made on all areas. We can expect that the development of a nano CAD framework can reduce the design gap between nanoscale designs and CMOS counterparts. As known, the classical tools are able to map millions of transistors large designs into CMOS technology.

In summary, this paper makes the following key contributions: (1) we propose to develop a new framework able to manage hybrid CMOS/nano architectures based on model specifications; and (2) the classic design-flow is extended to interact with these new models based on new and adapted tools/algorithms. Our broader objective is to develop a framework that could be used by research groups in this field and that could help them in their investigation of new materials, devices, and architectures evaluating implications at the system level.

The paper is organized as follows. Section 2 presents the proposed models used by this CAD framework to capture the characteristics of the nanofabrics. Section 3 gives an overview of the general organization of the proposed CAD framework. The subsequent sections discuss the main components of the new design flow. The last section shows the feasibility of the approach by taking an example application dedicated to NASIC nanofabric.

2. Models for fabric specification

The prototyping tool presented here is henceforth referred to as NanoMadeo. It is based on four meta-models. These meta-models provide abstractions of the nanofabrics concerning their computation paradigm (*computational meta-model*), their structural organization (*structural meta-model*), technological constraints (*technological meta-model*), and their fault-tolerance ability (*fault meta-model*).

Through these, the designer is able to capture different aspects of the target fabric. These models interact with behavioral transformations, structural transformations, and physical tools, needed to design and to implement an application onto the nanofabric support. These interactions are mediated by the meta-models. The aspects that need to be captured by the meta-models are detailed in the following subsections. The general flow of this tool is presented in Section 3.

2.1. Computational meta-model

CAD tools for emerging nanofabrics must be able to handle both traditional CMOS and nanoscale technologies. The distribution of functionality between the two depends on the nanoscale capabilities, manufacturability constraints, the trade-offs between area, performance, and the reliability of the underlying nanoscale technology.

For instance, the nanoscale parts of the system can be used solely for computation in order to gain orders of magnitude improvements in density and performance compared to CMOS technology [7]. Nanoscale technology could also be utilized for interconnect-only to speed-up communication in the fabric by reducing load capacitances and allowing a denser interconnect structure. A prototyping tool must handle both of these cases and in-between in order to be useful.

The computational meta-model also needs to be able to model different roles for the nano and/or CMOS segments related to both computation and interconnect. This partitioning is a new requirement of hybrid, i.e., nano/CMOS, fabrics that is not present in conventional tools.

2.2. Architectural meta-model

The architectural meta-model provides support for describing the building blocks of a target nanofabric. These building blocks correspond to nanoscale or CMOS components necessary to build the architecture on the fabric. These can be classified into basic devices, pre-composed blocks and wires as shown in Fig. 2. The architectural meta-model allows the specification of the types of tiles specific to each fabric: nanoBlocks for NanoPLA [1], tiles of basic cells for CMOL [3], hypercells for FPNI [4], and tiles for NASIC [5]. The topological organization of the nano and microcomponents, including their hierarchical structure in tiles, can also be captured by a model based on this meta-model.

2.3. Technological meta-model

The technological meta-model permits to model the physical constraints of the fabric based on the underlying technology. This

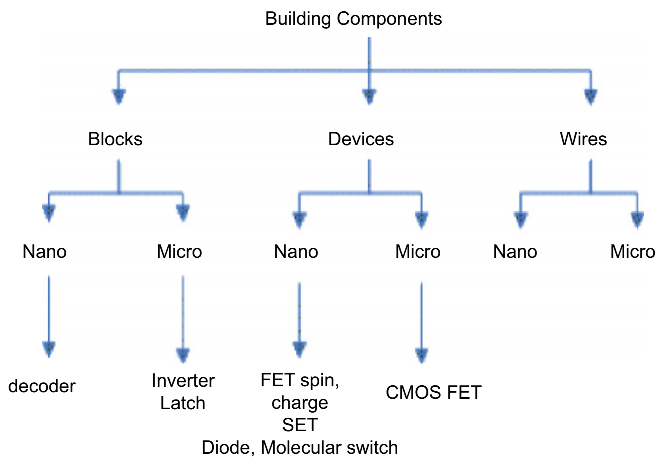


Fig. 2. Building blocks used in architectural model.

information is useful for place and route routines. Nanoscale systems often do not allow arbitrary routing and placement, complicating their design as compared to CMOS designs. For example, a placement constraint related to the fabric might be the doping constraints in the NASIC fabric that limits each type of transistor to one dimension: horizontal or vertical, for easing manufacturing.

Constraints on routing are particularly important in the case of reconfigurable fabrics where connections are limited to certain routes. The costs for these routes also give guidance to the routing procedures for choosing the best possible routes.

Another constraint may be the defects present in a particular chip in the case of a reconfigurable fabric. These present additional routing and placement constraints in configuring around the defects [7].

2.4. *Fault meta-model*

As nanoscale computational fabrics are commonly based on bottom-up manufacturing processes, accounting for the reliability implications is crucial. The fault meta-model proposed allows modeling of different type of faults, as well as their distribution and their probability on the target fabric.

Fault types include: permanent defects such as stuck-on or off transistors or broken nanowires/microwires; transient faults due to internal noise, particle impact, or electromagnetic interference; and process variation, including doping, channel length, wire thickness, and others. A detailed treatment of various fault models is discussed in [5]. For each possible fault in a given technology, the rate and distribution (uniform/clustered) is included in the fault model.

3. Design flow

The general flow of the framework for nanofabrics enlarges the classical flow introduced in Fig. 1 by adding an explicit specification of the fabric. The fabric specification is expressed through four models based on the meta-models presented in the previous section. These models interact with the transformations applied at one specific level of description (behavioral, structural, or physical) and interact with processes that are applied between different levels of description. The modified flow is illustrated in Figs. 3 and 4, respectively.

The computational model interacts with behavioral transformations and the synthesis process, by giving information about the nano and micro role and about the computational organization of the nanostructure (regular, two-level logic based, and hybrid logic [23,24]).

The architectural model interacts with the structural transformations and the place-and-route process by using some information about the building components and their topological organization.

The technological model introduces constraints in the place-and-route process like doping constraints, routing priorities on reconfigurable fabrics, defect map constraints in order to route around the defects. This technological model is taken into account for physical transformations (like physical replications at a floor-planning level).

The fault model of the fabric is essentially used for estimating the yield of the circuit based on its architectural description. The yield projection gives an overview of the circuit's fault tolerance and capability to be still correct in terms of its outputs even if some faults are introduced in the architectural description of the application (like transistor turned to stuck-off, stuck-on, or broken nanowire, etc.) related to a certain distribution of defects.

The yield of the circuit can give some insight about the efficiency of the fault-tolerance techniques that are defined as transformations (behavioral and/or structural transformations). The choice of fault-tolerance transformations has different impact on the yield. If the predicted yield is not satisfactory, it is possible to reiterate, applying different kinds of fault-tolerant transformations to different portions of the application. These iterations may continue until an acceptable level of yield is projected. More details on fault-tolerance related transformations are given in Sections 4.2 and 5.3.

4. Application specification and behavioral transformations

The behavioral description of an application is written in an object-oriented language (ST80) that is similar to the traditional description used in Madeo [8]. The compilation procedure produces directed acyclic graphs (DAGs), which are the intermediate representation (IR) used by NanoMadeo.

Some classical behavioral transformations can be applied on this (IR) in order to perform optimizations, e.g., dead code removal, constant propagation, and node fusion.

Some specific transformations related to fabric are also introduced at behavioral level and can be classified in two categories: (a) transformations for the fabric itself and (b) transformations required for supporting fault tolerance.

4.1. *Transformations for hybrid nano/CMOS fabrics*

These transformations include:

- nano/CMOS pre-partitioning,
- synchronization mechanism.

The nano/CMOS pre-partitioning transformations assume a first role-distribution between the nano part and the CMOS part. These transformations take into account the computational model that defines the computation and interconnect related tasks for CMOS or nanoscale parts. This assignment information is essentially captured through flags that will be considered all through the design process.

Transformations for introducing synchronization mechanisms depend on the way the sequential logic and the control signals are expressed at the fabric level, related to the computational model. These can be translated in subsequent steps by explicit registers or

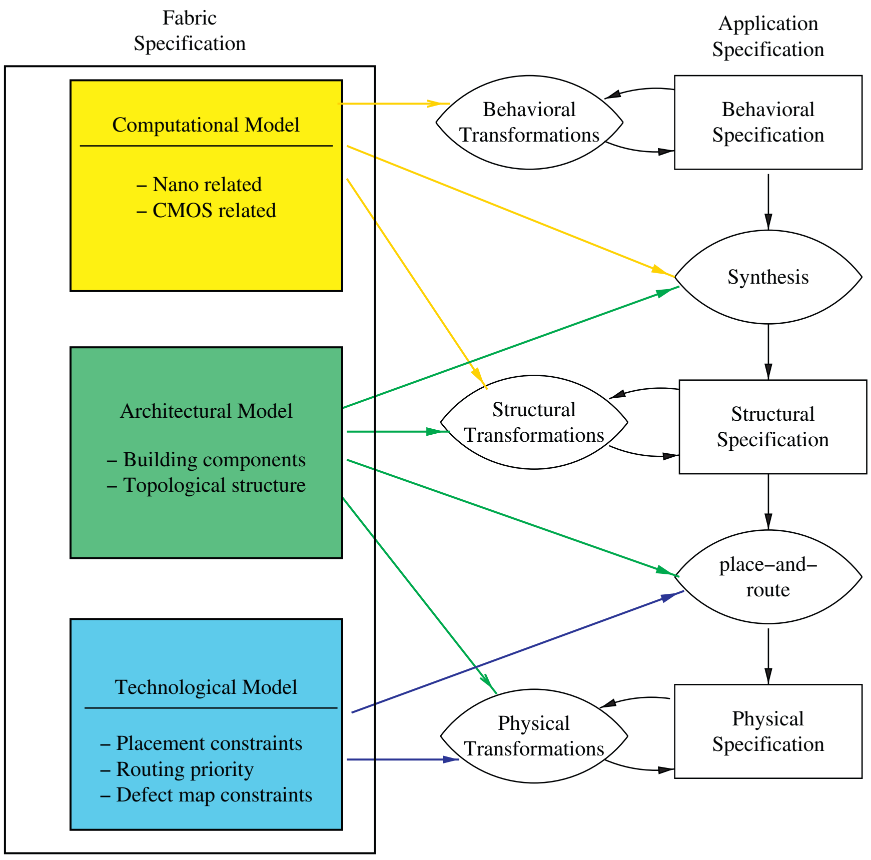


Fig. 3. Design flow with NanoMadeo and model interactions.

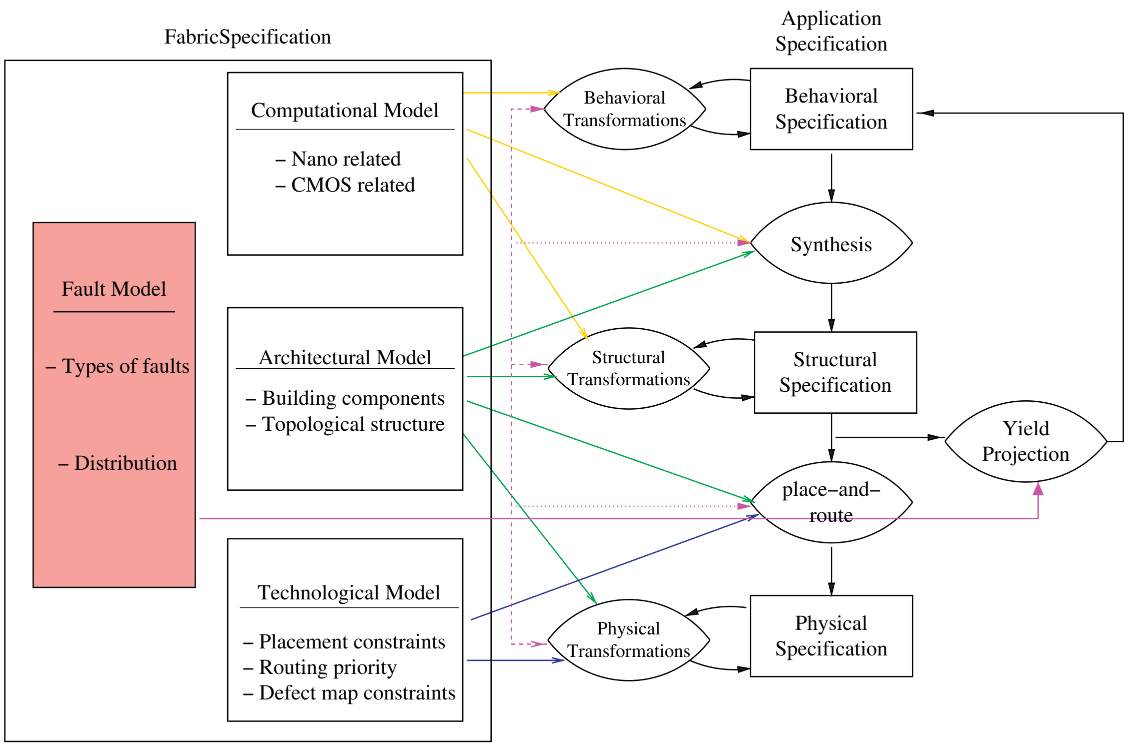


Fig. 4. Design flow with NanoMadeo and model interactions. Fault model integration into the fabric specification.

by a dynamic logic activated by specific control signals. Dynamic style designs are for example applied in NASICs.

4.2. Transformations for fault tolerance

Fault-tolerance techniques may be applied at this level. They correspond to three kinds of transformations:

- transformation introducing voters, like TMR, in order to duplicate portion of codes and to vote between copies,
- transformation introducing different data encoding based on redundant codes: RNS codes and ECCs. These codes are expressed at this level like data types,
- transformation changing the physical support: the computation can be realized by the CMOS part to be more reliable.

Fault-tolerance transformations at behavioral level interact with the computational model and the fault model. The computational model guides the synthesis of the additional parts (like voters for TMR techniques), to be mapped on a nanostructure or a CMOS part. The fault model guides the designer on the relevance of the fault-tolerance transformations after being evaluated using yield projection. For instance, it would not be worth using an ECC with a large Hamming distance, that would yield too many redundant bits without the yield increasing proportionally.

5. Synthesis, structural transformations and yield projection

5.1. Synthesis

The resulting logic is then synthesized in the appropriate type of logic (PLA, LUT, and multi-level logic) as defined by the architectural model. Standard external tools such as SIS [9] can be used for this process. This is done on block-wise basis; each high-level code operation is compiled hierarchically into a single block.

Different levels of operator decomposition can be applied, allowing the complexity of each block to be traded off against the number of blocks. The synthesis process may interact with the fault model of the fabric if probabilistic synthesis [10] is used.

Once the initial structural representation of the application has been generated through synthesis, transformations at the structural level are applied.

In the following subsections we focus on structural transformations applied for both fabric and for fault tolerance.

5.2. Structural transformations

The synthesis is based only on the type of logic and may not take into account all structural requirements (for instance, the signal restoration required by the NanoPLA fabric). At this point, logic is synthesized for the combinatorial parts such as decoders. Sequential parts such as registers or dynamic logic controls are defined around the synthesized segments as required in the architecture.

Specific architectural components may be introduced extracted from a library, corresponding to some architectural investigations and whose necessity could be demonstrated through a validation tool (simulation tool).

These structural transformations for a fabric may introduce:

- restoration circuitry,
- stochastic decoder, needed for the interface between the micro and the nano parts,
- CMOS/nano repartitioning.

These components are needed for the interface and are defined in the architectural model. The CMOS/nano repartitioning enables possible migration from nano to CMOS parts according to performance and/or reliability requirements.

5.3. Structural fault-tolerance related transformations

Fault-tolerance transformations can be also applied at the structural level. Examples include:

- Structural redundancy techniques at fine grain level, like N-way redundancy to provide additional copies of input/output signals and of some intermediate signals (the copies of minterms in a case of PLA structures) [5].
- Modular redundancy techniques at coarse grain level, based on TMR and similar techniques—specific structures are selected and voter circuits are provided to implement TMR or similar schemes, but at this level, a more detailed architecture of this kind of circuitry is provided.

These transformations corresponding to the fault-tolerance techniques applied at the structural description of the application are, essentially, redundancy based on fine grained or coarse grained (like TMR) copies. Voters here could be using the CMOS/nano repartitioning transformation for improving yield—also depending on manufacturing requirements.

5.4. Yield projection

The structural representation of the circuit plus the fault distribution given by the fault model can be used to make yield projections. This is performed by an external yield simulator. A yield simulator for PLA-based structures proposed in [11] could be used for different kinds of 2D nanofabrics. Yield estimation can also be done using Monte Carlo simulation [7] or the FTSim developed by the NASIC group [5].

The yield simulator depends on the fault model of the fabric in terms of the types of faults considered and the distribution of these faults. It gives feedback on the efficiency of the fault-tolerance techniques applied at different levels of the design flow.

6. Physical design

Nanofabrics are generally organized into tiles, hypertiles or nanoblocks that correspond to clusters of PLAs, basic cells, or hypercells. The partitioning techniques used to define such blocks are based on clustering heuristics for PLA packing, as in PLAmap [12], T-VPACK [13], or the Singh algorithm [14] (Table 1).

The parameters for clustering are the number of elementary cells or P-terms of the PLA and the number of inputs and outputs associated with the cluster. The placement problem consists of placing each basic cell inside a cluster, once the clusters are defined. This is achieved using generic optimization heuristics like simulated annealing, using, e.g., congestion costs in the case of reconfigurable fabrics.

Routing procedures for nanofabrics can use adaptive maze router algorithms like Pathfinder [15] from VPR [13], or they can be more specific to the fabric using, for example, custom adaptation of shortest Steiner tree problems [16] or other VLSI algorithms [17].

For reconfigurable fabrics, a defect map provides extra constraints for placement and routing to configure around the defects previously detected.

Table 1
Main features of some hybrid nanofabrics related to models

Fabric models		NanoPLA	CMOL	FPNI	NASIC
Computational	Nano	Computation, interconnect	Computation, interconnect	Interconnect only	Computation, interconnect
	CMOS	Limited to I/O	Specific computation (inv)	Computation	Control
Architectural	Devices	FET, diode	Molecular switch, SET	CMOS	NW-FET
	Structure	2D-grid	3D-grid	3D-grid	2D-grid
Technological (placement, routing)		Connection restricted	Connection restricted	Connection restricted	Doping constraints
Fault		Permanent	Permanent	Permanent	Transient, permanent

Table 2
CAD tools used for different fabrics

	NanoPLA [1]	CMOL [6]	FPNI [4]
Partitioning	PLAMAP [12]	T-VPACK [13]	Singh's greedy algorithm (specific cost) [14]
Placement	Simulated annealing (VPR-like)	Simulated annealing (VPR-like) modified congestion function	Simulated annealing (VPR-like)
Routing	NPR—custom tool (based on Pathfinder [15])	Custom tool (based on RSA heuristic) [16]	Maze router (Pathfinder-like) with several iterations

Table 2 gives an overview of the different algorithms applied in physical layout tools for the NanoPLA, CMOL, and FPNI fabrics. Physical tools for nanofabrics use two kinds of algorithms or heuristics: adaptive generic algorithms or custom procedures. Adaptive generic algorithms include general purpose optimization heuristics like simulated annealing or genetic algorithms and algorithms for FPGAs like Pathfinder and PLA clustering as the ones implemented in Madeo [18] or the VPR tool.

On NanoMadeo, the place and route step is done using generic algorithms capable to take into account the nanofabric constraints extracted from the architectural and technological models. Constraints like the tile dimension, the position of each specific component, and the available hierarchical levels can be extracted from the architectural model in order to be used by the placement algorithm. The technological model provides fine-grained constraints such as the doping constraints for the NASIC fabric.

The placement is done using the simulated annealing heuristic on a generic representation of the placement problem (similar to TCG-S [19]).

After the extraction of routing constraints from the architectural and technological models, the circuit can be routed using a generic maze router like the PathFinder.

The optimization goal for the place and route tool can be inferred from the architectural and technological models. For example, in a fabric using the nanowires for signal routing the goal will be wire-length minimization knowing that the length of the nanowires is limited due to fabrication constraints.

7. Illustration of the framework: NASIC case study

Table 1 presents some particularities of four emerging nanoscale fabrics and which model can capture them best. For additional clarification of the NanoMadeo framework, the rest of this section is structured as a case study on NASIC fabric architecture.

7.1. NASIC fabric description

NASIC [5,20–24] is a hybrid system based around tiles of nanowires and FETs with CMOS providing support and some control circuitry. Recent versions of NASICs also explore CNTs and Spin FETs. For the purpose of this paper we assume NASICs with NW FETs, such core-shell based ones or crossed-NW ones.

The tiles are made up of crossed nanowires with FETs at the intersections, forming cascaded PLA-like structures. In each tile (or supertile), there are possible several planes of transistors—one with the channels running horizontally and one with the channels running vertically typically. Thus, each tile implements any logic function using two-level logic such as AND–OR, NAND–NAND, or NOR–NOR. Recent versions of NASICs also use the so-called hybrid logics [23] that would extend this functionality allowing mixing logic gates in the same logic stage.

In the NASIC fabric, each nanotile is surrounded by microwires which provide power and control signals. The control signals implement typically various styles of a dynamic control schemes. The use of dynamic logic puts a synchronization constraint on the synthesis of applications onto NASICs, which NanoMadeo must manage. CMOS also provides support for modular redundancy schemes, encoding/decoding of inputs and outputs for the entire system (not between tiles), and control signal generation.

7.2. Modeling the NASIC fabric

To model the NASIC fabric, in the context of NanoMadeo, the particularities of the fabric have been identified and four NASIC fabric models produced according to the meta-models described in Section 2.

The distribution of the role between CMOS and nano layer is driven by the computational model of the NASIC fabric and explicit two main points: one is the computation organization of the nanogrid in two level logic in order to be mapped later into PLA structures (information related to synthesis), the other is related to the control aimed to be mapped to CMOS level.

The architectural model of the NASIC fabric points the building components used (e.g. FET, nanowire, and microwire) and explicit structural organization into tiles based on topological rules on building components. Building components may be assembled in a predefined way (for instance, for the nano and microwires, if the number of nanotiles are supposed to be fixed) or in an adaptable way related to the application (placement of FETs on the PLA

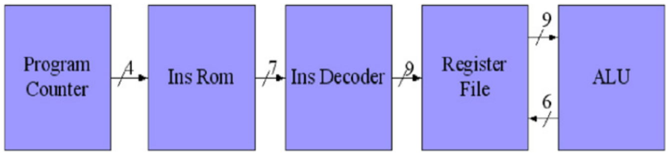


Fig. 5. WISP-0 block diagram.

structure). In the latter case, these topological rules are active during the place-and-route phase.

The technological model provides the physical constraint of the NASIC fabric essentially due to the doping constraints of the two types of transistors that can be used (N-FET, P-FET). These constraints introduce some additional complexity in the placement routines inside one tile. The mapping onto different tiles adds I/O constraints between tiles.

The fault model takes into account two types of faults: permanent and transient faults. Distribution and rate of these types of faults have an impact on the reliability of the implementation. The fault-tolerant transformations have different capabilities in masking errors that can be evaluated using a yield simulator [5].

7.3. WISP-0 application design for NASIC

WISP-0 [21] is a stream processor, built on NASIC, that implements a streaming processor architecture with 5-stage pipeline: fetch, decode, register file (RF), execute, and write back. It is a multi-tile design with five nanotiles (Fig. 5). A key feature is that intermediate values during execution are often stored on the nanowires directly without explicit latches using a three-phase dynamic control. Other key aspects relate to its fault-masking strategy, density optimizations, and control schemes.

The specification of WISP-0 with NanoMadeo is functional, including the followings:

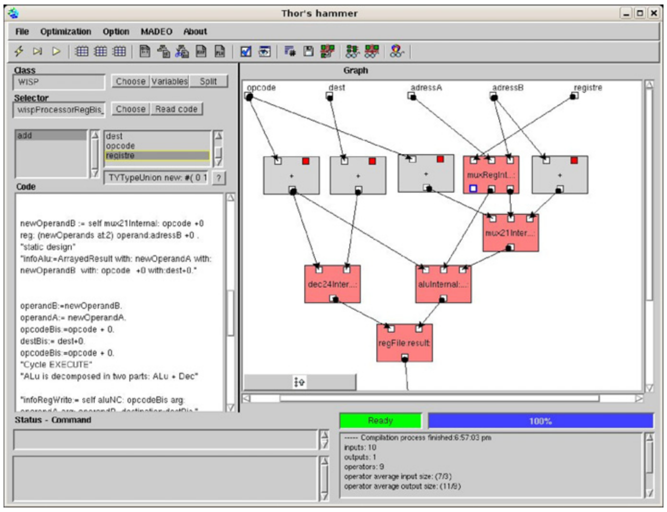
- a synchronization primitive,
- a reflexive operator to define feed-back,
- specific types to introduce redundancy.

In Fig. 6, we give an example of a DAG produced by the compiler from source code defining the ALU and RF stages of WISP-0. In this DAG, nodes correspond to function calls or operators that will be synthesized into logic PLA blocks equivalent with NASIC logic.

This description implicitly describes some data synchronization, but this information could be made more explicit and could be managed through specific behavioral transformations. No explicit partitioning between nano/CMOS is done at this point because every logic functionality is aimed to be implemented at nanoscale in the case of NASICs. Nevertheless, if some fault-tolerant techniques are added in CMOS, this information needs to be explicit (for instance, the generation of CMOS TMR voters by transformation for fault tolerance). Other transformations for fault tolerance can be applied by injecting specific data types for the inputs. These types represent future data encodings for the input data; for instance, in the case of WISP-0, BCH codes (as ECC) are used to introduce built-in redundancy.

We show, in Fig. 7, the effect of ECC fault-tolerance transformations on the LUT[1] specification of the instruction memory of Wisp-0 processor. The ECC transformations correspond to BCH codes applied to the memory addresses and instruction codes.

[1] LUT—look up table.



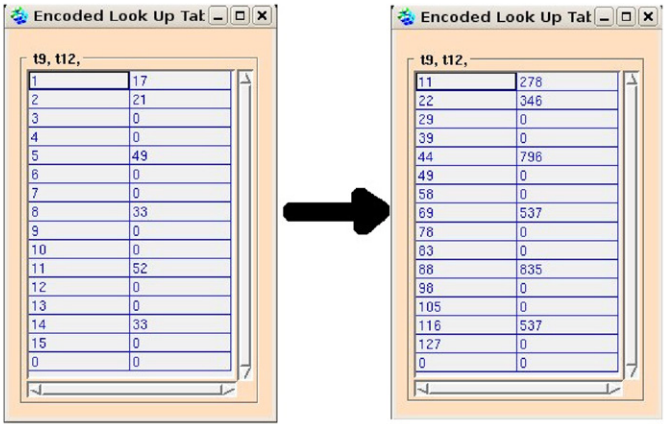
```
newOperandB := self mux21Internal: opcode +0
reg: (newOperands at:2) operand:adressB +0 .
"static design"
"infoAlu:=ArrayedResult with: newOperandA with:
newOperandB  with: opcode  +0 with:dest+0."


operandB:=newOperandB.
operandA:= newOperandA.
opcodeBis:=opcode + 0.
destBis:= dest+0.
opcodeBis:=opcode + 0.
"Cycle EXECUTE"
"ALu is decomposed in two parts: ALu + Dec"

"infoRegWrite:= self aluNC: opcodeBis arg:
```

```
----- Compilation process finished:6:57:03 pm
inputs: 10
outputs: 1
operators: 9
operator average input size: (7/3)
operator average output size: (11/9)
```

Fig. 6. DAG from NanoMadeo representing ALU and RF.



1	17
2	21
3	0
4	0
5	49
6	0
7	0
8	33
9	0
10	0
11	52
12	0
13	0
14	33
15	0
0	0

11	278
22	346
29	0
39	0
44	796
49	0
58	0
69	537
78	0
83	0
88	835
98	0
105	0
116	537
127	0
0	0

Fig. 7. LUT of instruction memory before and after BCH encoding.

7.4. Structural transformations and yield projection for WISP-0

In the case of a NASIC fabric (without hybrid logic), NanoMadeo utilizes the external synthesis tool SIS to perform the two-level logic synthesis of PLAs associated with each node of the DAG. Assembly of synthesized portions is addressed by NanoMadeo to define the complete logical structure of the WISP-0 application. WISP-0 may use some structural fault-tolerance techniques such as TMR and N-way redundancy of signals. The result of synthesis is then transformed here to implement these techniques, when they are in use.

We have developed a yield simulator to evaluate fault-tolerance techniques in NASICs. The simulator generates random defect maps for designs based on a defect model and runs logic simulations on them, testing with many different possible sets of input. By measuring what proportion of the generated defect maps result in correct output when simulated, the yield can be estimated. NanoMadeo can automatically call the yield simulator to evaluate defect and fault-tolerance techniques. One example of output after several runs of the yield simulator, using different fault rates and different fault-tolerance techniques (TMR, ECC, and N-way) is shown in Fig. 8. This graph provides information on the efficiency of the fault-tolerant techniques related to the fault rate and the types of permanent defects (for instance, if the fault rate is above 6%, the yield is better with ECC techniques considering 10% stuck-off and 90% stuck-on).

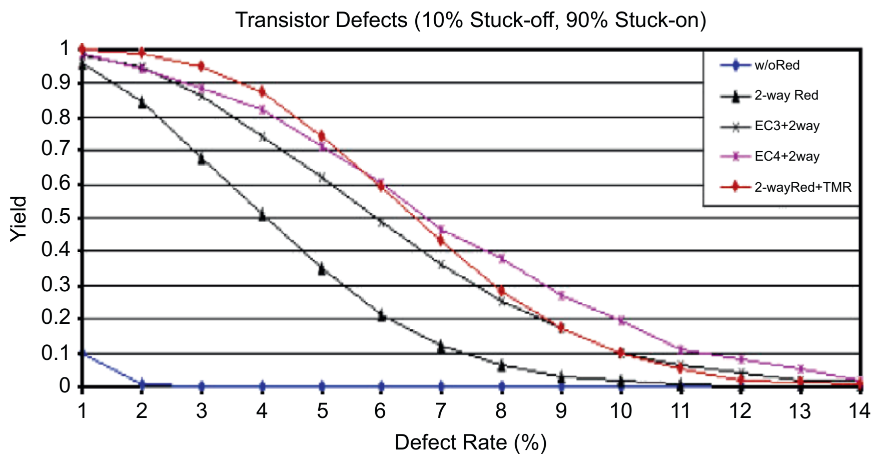


Fig. 8. Graph of yield simulator output for WISP-0.

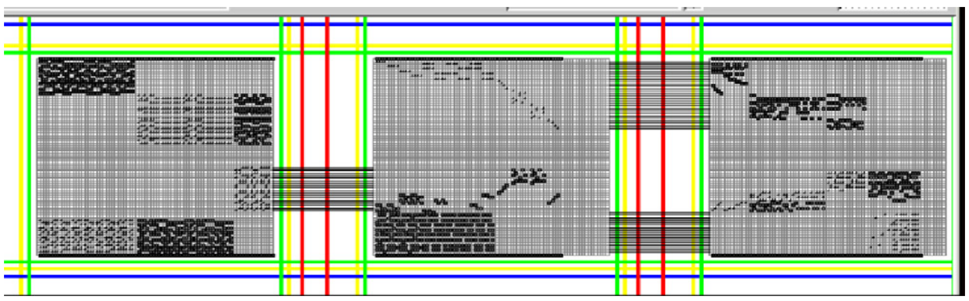


Fig. 9. WISP-0 layout from NanoMadeo.

7.5. *Wisp0 layout*

Based on the architectural and technological model of NASIC fabric, an abstract layout can be produced taking into account the layout constraints inside one tile. In Fig. 9, we present the abstract layout of WISP-0 onto a nanogrid of three tiles, partially integrating some fault-tolerant techniques. A more efficient place-and-route algorithm without constraints on the size of the tile is under study.

8. Conclusion

In order to handle next generation hybrid nano architectures, CAD tools will have to evolve. Highly heterogeneous multi-part fabrics introduce new challenges which must be met efficiently. In this paper, we have shown that the proposed tool, NanoMadeo, can handle many of these challenges and can be used productively for work on NASIC designs. Its design will make it easy to adapt it for other hybrid nanoscale architectures.

References

[1] A. DeHon, Nanowire-based programmable architectures, JETC 1 (2) (2005) 109–162.
[2] A. DeHon, Design of programmable interconnect for sublithographic programmable logic arrays, in: FPGA '05: Proceedings of the 2005 ACM/SIGDA 13th International Symposium on Field-programmable Gate Arrays, ACM, New York, NY, USA, 2005, pp. 127–137.
[3] D.B. Strukov, K.K. Likharev, Cmol fpga: a reconfigurable architecture for hybrid digital circuits with two-terminal nanodevices, Nanotechnology 16 (6) (2005) 888–900 ⟨http://stacks.iop.org/0957-4484/16/888⟩.
[4] G.S. Snider, R.S. Williams, Nano/CMOS architectures using a field-programmable nanowire interconnect, Nanotechnology 18 (3) (2007) 035204 11pp. ⟨http://stacks.iop.org/0957-4484/18/035204⟩.
[5] C. Moritz, T. Wang, P. Narayanan, M. Leuchtenburg, Y. Guo, C. Dezan, M. Bennaser, Fault-tolerant nanoscale processors on semiconductor nanowire grids, Circuits Syst. I: Regular Papers, IEEE Trans. Circuits Syst. I: Fundam. Theory Appl. 54 (11) (2007) 2422–2437.
[6] D.B. Strukov, K.Likharev, A reconfigurable architecture for hybrid cmos/nanodevice circuits, in: FPGA '06: Proceedings of the 2006 ACM/SIGDA 14th International Symposium on Field-programmable Gate Arrays, 2006, pp. 131–140.
[7] C. He, M. Jacome, Defect-aware high-level synthesis targeted at reconfigurable nanofabrics, IEEE Trans. Comput. Aided Des. Integrated Circuits Syst. 26 (5) (May 2007) 817–833.
[8] E. Fabiani, L. Lagadec, B. Pottier, A.Poungou, S. Yazdani, Abstract execution mechanisms in a synthesis framework, in: Workshop on Non-Silicon Computations (NSC3), June 2004.
[9] E.M. Sentovich, K.J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P.R. Stephan, R.K. Brayton, A. Sangiovanni-Vincentelli, SIS: a system for sequential circuit synthesis, Technical report, 1992 ⟨http://citeseer.ist.psu.edu/sentovich92sis.html⟩.
[10] R. Bahar, J. Mundy, J. Chen, A probability-based design methodology for nanoscale computation, in: Proceedings of the International Conference on Computer-Aided Design San Jose, CA, November 2003, pp. 480–486.
[11] F. Angiolini, M. Jamaa, D. Atienza, L. Benini, G. De Micheli, Improving the fault tolerance of nanometric PLA designs, Design, Automation and Test in Europe Conference and Exhibition, 16–20 April 2007, pp. 1–6.
[12] D. Chen, J. Cong, M. Ercegovac, Z. Huang, Performance-driven mapping for CPLD architectures, IEEE Trans. Comput. Aided Des. Integrated Circuits Syst. 22 (10) (2003) 1424–1431.
[13] V. Betz, J. Rose, A. Marquardt (Eds.), Architecture and CAD for Deep-Submicron FPGAs, Kluwer Academic Publishers, Norwell, MA, USA, 1999.
[14] A. Singh, G. Parthasarathy, M. Marek-Sadowska, Efficient circuit clustering for area and power reduction in fpgas, ACM Trans. Design Autom. Electron. Syst. 7 (4) (2002) 643–663.
[15] L. McMurchie, C. Ebeling, Pathfinder: a negotiation-based performance-driven router for FPGAs, Field-Programmable Gate Arrays, 1995. FPGA '95. in: Proceedings of the Third International ACM Symposium, 1995, pp. 111–117.
[16] S.K. Rao, P. Sadayappan, F.K. Hwang, P.W. Shor, The rectilinear steiner arborescence problem, Algorithmica 7 (2 & 3) (1992) 277–288.
[17] S.H. Gerez, Algorithms for VLSI Design Automation, Wiley, New York, NY, USA, 1999.
[18] L. Lagadec, Abstraction, modélisation et outils de cao pour les circuits intégrés reconfigurables, Ph.D. Thesis, Université de Rennes1, Rennes, France, 2000.

[19] J.-M. Lin, Y.-W. Chang, TCG-S: orthogonal coupling of p*-admissible representations for general floorplans, in: 39th Design Automation Conference (DAC'02), 2002, p. 842.
[20] C.A. Moritz, T. Wang, Latching on the wire and pipelining in nanoscale designs, in: 3rd Workshop on Non-Silicon Computation (NSC-3), ISCA'04, Germany, 2004.
[21] T. Wang, M. Ben-Naser, Y. Guo, C.A. Moritz, Wire-streaming processors on 2-d nanowire fabrics, NSTI (Nano Science and Technology Institute) Nanotech'05, California, 2005.
[22] T. Wang, M. Ben-Naser, Y. Guo, C.A. Moritz, Self-healing wire-streaming processors on 2-d semiconductor nanowire fabrics, NSTI (Nano Science and Technology Institute) Nanotech'06, Boston, MA, 2006.
[23] T. Wang, P. Narayanan, C.A. Moritz, Combining 2-level logic families in grid-based nanoscale fabrics, in: IEEE International Symposium Nanoscale Architectures, 2007, NANOSARCH 2007, 21–22 October, 2007, pp. 101–108.
[24] T. Wang, P. Narayanan, C.A. Moritz, Hybrid Logic and its Density and Fault Tolerance Implications in Nanoscale Fabrics, accepted by IEEE Transactions on Nanotechnology.