Test and Debug Solutions for 3D-Stacked Integrated Circuits

by

Sergej Deutsch

Department of Electrical and Computer Engineering
Duke University

Date: ______________________

Approved:

____
Krishnendu Chakrabarty, Supervisor

____
Pankaj Agarwal

____
Rolf Drechsler

____
Nan Jokerst

____
Hisham Massoud

____
James Morizio

Dissertation submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in the Department of Electrical and Computer Engineering
in the Graduate School of Duke University
2015

Abstract

Test and Debug Solutions for 3D-Stacked Integrated Circuits

by

Sergej Deutsch

Department of Electrical and Computer Engineering
Duke University

Date: ___________________________
Approved:

____
Krishnendu Chakrabarty, Supervisor

____
Pankaj Agarwal

____
Rolf Drechsler

____
Nan Jokerst

____
Hisham Massoud

____
James Morizio

An abstract of a dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Electrical and Computer Engineering in the Graduate School of Duke University
2015

Abstract

Three-dimensional (3D) stacking using through-silicon vias (TSVs) promises higher integration levels in a single package, keeping pace with Moore's law. TSVs are small copper or tungsten vias that go vertically through the substrate of a die and provide vertical interconnects to a die stacked on top. TSV-based interconnects have benefits in terms of performance, interconnect density, and power efficiency.

Testing has been identified as a showstopper for volume manufacturing of 3D-stacked integrated circuits (3D ICs). A number of challenges associated with 3D test need to be addressed before 3D ICs can become economically viable. This dissertation provides solutions to new challenges related to 3D test content, test access, diagnosis and debug.

Test content specific to 3D ICs targets defect that occur during TSV manufacturing and stacking process. One example is the effect of thermo-mechanical stress due to TSV fabrication process on the surrounding logic gates. In this dissertation, we analyze these effects and their consequences for delay testing. We provide quantitative results showing that the use of TSV-stress oblivious circuit models for test generation leads to considerable reduction in delay-test quality. We propose a test flow that uses TSV-stress aware circuit models to improve test quality.

Another example of 3D-specific test challenge is the testability of TSVs. In this dissertation, we focus on TSV test prior to die bonding, as access to TSVs is limited at this stage. We propose a non-invasive method for pre-bond TSV test that does not require TSV probing. The method uses ring oscillators and duty-cycle detectors in order to detect variations in propagation delay of gates connected to a single-sided TSV. Based on the measured variations, we can diagnose the TSV and predict the size of resistive-open and leakage faults using a regression model based on artificial neural networks. In addition, we exploit different voltage levels to increase the robustness of the test method.

In order to efficiently deliver test content to structures under test in a 3D stack, 3D design-for-test (DfT) architectures are needed. In this dissertation, we discuss existing 3D-DfT architectures and their optimization. We propose an optimization approach that takes uncertainties in input parameters into account and provides a solution that is efficient in the presence of input-parameter variations and minimizes test time, therefore reducing test cost.

Post-silicon debug is a major challenge due to continuously increasing design complexity. Traditional debug methods using signal tracing suffer from the limited capacity of on-chip trace buffers that only allow for signal observation during a short time window. This dissertation proposes a low-cost debug architecture for massive signal tracing in 3D-stacked ICs with wide-I/O DRAM dies. The key idea is to use available on-chip DRAM for trace-data storage, which results in a significant increase of the observation window compared to traditional methods that use trace buffers. In addition, the proposed on-chip debug circuitry can identify erroneous segments of observed data by using compact signatures that are stored in the DRAM *a priori*. Only failing intervals are off-loaded from a temporary trace buffer into DRAM, allowing for a more efficient use of the memory, resulting in a larger observation window.

In summary, this dissertation provides solutions to several challenges related to 3D test and debug that need to be addressed before volume manufacturing of 3D ICs can be viable.

Contents

List of Tables

List of Figures

Acknowledgements

I would like to acknowledge the financial support received from the National Science Foundation and the Semiconductor Research Corporation. I thank my PhD advisor, Krishnendu Chakrabarty, for his mentorship. I also thank Sun Kyu Lim and Shreepad Panth for the 3D design benchmarks that they contributed for use in research. I acknowledge Erik Jan Marinnisen for his collaboration. I also acknowledge the contributions of the past and present students at Duke University, including Mukesh Agrawal, Qing Duan, Hongxia Fang, Zipeng Li, Yan Luo, Brandon Noia, Ran Wang, and Fangming Ye. Finally, I would like to thank the members of my committee for for their time and effort in reading this dissertation and attending my defense.

1

Introduction

Three-dimensional IC stacking using through-silicon vias (TSVs) is a relatively new technology that has a number of advantages over conventional stacking methodologies [1]. TSVs are vertical copper or tungsten conducting nails passing through a thinned die. Typical TSV dimensions are 5 μm diameter and 50 μm height. The actual connection to the next die can be a direct copper-to-copper bond of the TSV onto a small landing pad, but today is often implemented by means of a CuSn micro-bump, of which typical dimensions are 25 μm diameter at 40 μm pitch [2]. As they form direct vertical interconnects between stacked dies, TSVs allow for a much larger number and higher density of interconnects than conventional wire-bonds. Due to their geometry, TSVs have relatively low capacitance and inductance, and thus enable high bandwidth and low power consumption [3].

Despite the numerous benefits offered by 3D integration, test challenges for 3D-stacked integrated circuits (ICs) must be addressed before volume manufacturing and defect screening can be feasible [4, 5, 6]. These challenges include the following.

- **Test content**

As any semiconductor product, a 3D IC should be tested for defects that lead to errors in the functional operation of the IC. These defects can be divided into two categories: (a) defects that are specific to 3D ICs, and (b) defects that occur both in traditional (2D) ICs and 3D ICs. The category (a) includes defects in TSVs and micro-bumps, as well as defects caused by mechanical stress during TSV and micro-bump manufacturing. New test techniques need to be developed to screen for these defects. The category (b) includes defects in the internal die logic. Traditional test methods, such as scan test, have been successfully used to test for these defects. However, test costs tend to multiply with the increasing complexity of 2D and 3D ICs; therefore, more efficient test solutions are required to keep test costs low.

- **Test access**

  Test access is difficult both in pre-bond and post-bond test of 3D ICs. In pre-bond test, probing on micro-bumps and TSVs is constrained due to their small dimensions. Therefore, special techniques are required to test TSV-based connections prior to die bonding. In post-bond test, only one die in a 3D stack has external connections - other dies require test access through this die, necessitating a 3D design-for-test (DfT) architecture that allows for test access to all components in the stack. In addition, 3D DfT architectures must be optimized for on-chip area requirements and test time, which both have a major impact on test costs.

- **Diagnosis and Debug**

  Silicon debug requires a relatively large engineering effort, accounting for a significant portion of the total time-to-market of the silicon product and this portion has been projected to grow [7, 8]. In order to keep pace with advances in system-level integration, including 3D stacking, traditional methodologies for bug localization need to be improved.

This introduction provides a motivation for this dissertation, which addresses the challenges of 3D test presented above.

In Chapter 2, we address the issue of thermo-mechanical stress due to TSV fabrication. TSV stress changes the timing profile of the digital logic surrounding TSVs, which has an impact on delay-fault testing. We analyse this effect and show quantitatively that test quality is significantly reduced if the test patterns are generated with TSV stress-oblivious circuit models. The detrimental impact of TSV stress on pattern effectiveness and test quality can be overcome by using stress-aware circuit models for test generation.

Chapter 3 presents a technique for contactless pre-bond TSV test and diagnosis using ring oscillators and duty-cycle detectors. TSVs are used as capacitive loads of their driving gates that are configured in a ring oscillator. By measuring the oscillation period and the duty cycle of the signal generated by the ring oscillators, we can detect resistive open and leakage faults. A regression model based on artificial neural networks can predict the fault type and the fault size using the oscillation period and the duty cycle measured at multiple voltage levels as input parameters. The accuracy of the regression model is evaluated through simulation using realistic models for a 45nm CMOS technology.

Chapter 4 addresses the challenge of robust optimization of 3D test-access architectures. Traditional optimization frameworks suffer from the drawback that they ignore potential uncertainties in input parameters. In realistic scenarios, however, the input parameter values used in the design phase may differ from the actual values that are known only after the design phase. Examples of such parameters include test power and configuration of the die test-access mechanism. We propose a robust-optimization framework that takes uncertainties in input parameters into account and provides robust solutions. We evaluate the proposed framework and show that robust solutions are superior to single-point solutions in terms of average test time when there are uncertainties in the values of input parameters.

Finally, Chapter 5 presents a method for massive signal tracing using on-chip DRAM for in-system silicon debug. Traditional debug techniques using signal tracing suffer from the limited capacity of on-chip trace buffers. We propose a low-cost debug architecture for signal tracing that exploits large amounts of fast on-chip memories available in wide-I/O 3D ICs. The key idea of the proposed architecture is to store the trace data into functional memory, thereby significantly increasing the signal-observation window compared to traditional methods that use dedicated trace buffers. We evaluate the proposed method and show that the observation window can be increased by orders of magnitude compared to prior work at comparable hardware cost.

This introduction will continue with a general overview of 3D integration using TSVs and prior work in 3D test.

1.1 Introduction to 3D Integration Using TSVs

3D stacking by means of through-silicon vias (TSVs) is a relatively new technology that has a number of advantages over conventional stacking methodologies. TSVs are usually copper or tungsten nails going through the silicon substrate, electrically connecting the circuitry of the stacked dies [9]. Dies are bonded together by using copper or copper-tin micro-bumps, which are part of the inter-die interconnects. TSV-based interconnects allow a much higher density than traditional wire-bonds, which can only be placed along the perimeter of the die. This technology has the following benefits:

- The option of combining dies from different process technologies, i.e., optimized for digital logic, analog logic, or DRAM.

- High-bandwidth, low-power, high-density inter-die connections
- Increased compound yield, as systems can be partitioned into smaller dies, increasing the yield of individual dies.
- Smaller footprint

Figure 1.1 shows a packaged 3D stack consisting of two dies. In *Die1*, TSVs (in orange) have been processed and the silicon wafer has been thinned down in order to expose the TSVs on the back side. *Die2* has been manufactured using conventional (2D) technology without TSVs. Together with micro-bumps, TSVs form electrical connections between the active circuitry (in black) of both dies.

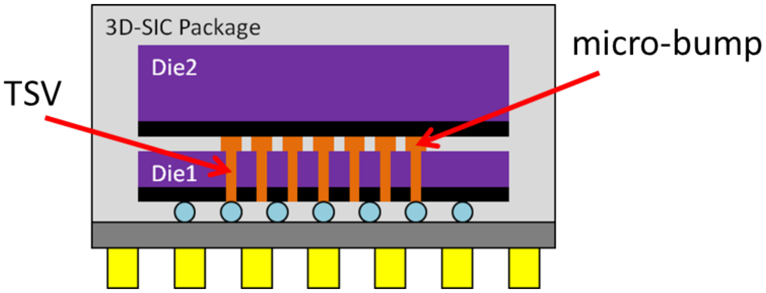
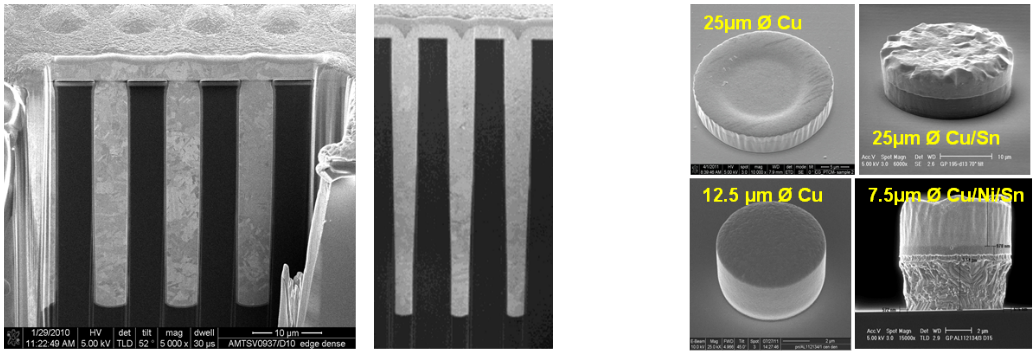


FIGURE 1.1: Packaged 3D IC on a board.

Cross-sections of two silicon wafers containing TSVs with different aspect ratios are depicted in Figure 1.2(a). Figure 1.2(b) shows micro-bumps of different forms and sizes. Table 1.3 shows the TSV and micro-bump dimensions that are used in the current technology at IMEC [10].

Figure 1.4 shows a typical TSV manufacturing flow. In the first step, holes are etched into the silicon substrate. Then an insulator is deposited and the holes are filled with copper. Chemical Mechanical Polishing removes the excess layer of copper. In order to expose the TSVs on the back-side of the silicon substrate, it needs to be thinned down, which is the next step.



(a)

(b)

FIGURE 1.2: SEM photographs of (a) TSVs and (b) micro-bumps (source IMEC).

Dimension	**TSV**	**micro-bump**
Diameter	5.2 μm	7.5 – 25 μm
Height	40 μm	5 – 10 μm
Minimum pitch	10 μm	40 μm

FIGURE 1.3: Typical TSV and micro-bump dimensions.

Wafer thinning and bonding processes are shown in Figure 1.5. The silicon substrate is temporarily bonded with the front-side to a carrier wafer and the back-side is thinned down. The exposed TSVs on the back-side are bonded to the front-side of the bottom wafer by means of micro-bumps. Finally, the carrier wafer is removed.

TSV can be processed at different stages of the CMOS manufacturing flow. There are three common approaches [11]:

- *Via-first.* In this case, TSVs are processed before the transistors (front end of line, FOEL).
- *Via-middle.* In this case, TSVs are fabricated after FOEL but prior to metal layers (back end of line, BOEL).
- *Via-last.* In this case, TSVs are processed after BOEL.

Typically, via-first and via-middle TSVs are smaller and denser than via-last TSVs. A major drawback of the via-first approach is that the TSVs are exposed to high

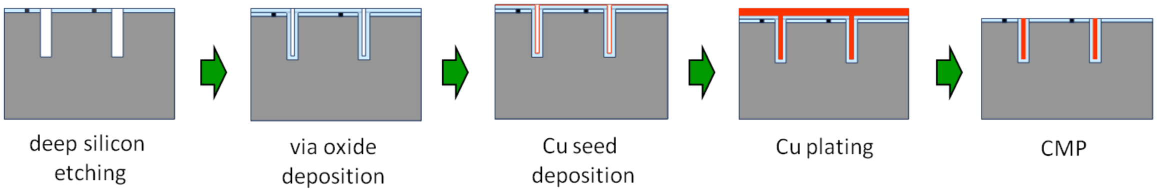


FIGURE 1.4: TSV fabrication steps in IMEC's process.

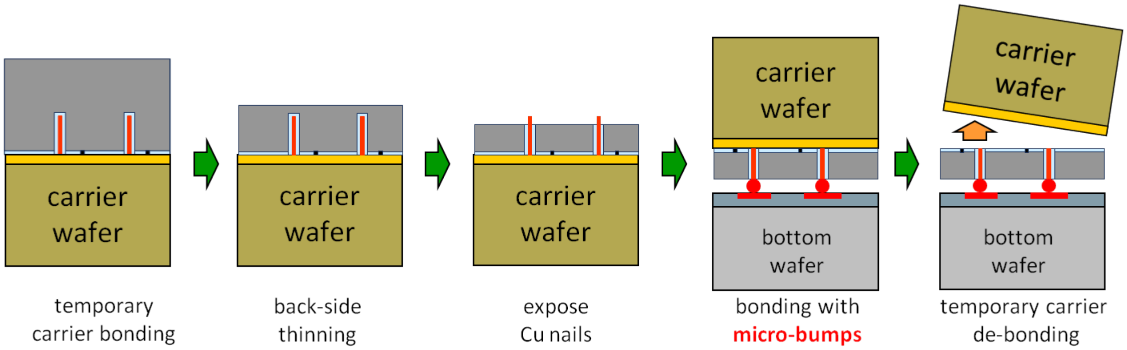


FIGURE 1.5: Wafer thinning and stacking steps in imec process.

temperatures during annealing, which is done at FOEL. Therefore, copper cannot be used as TSV filling material in the via-first approach due to material diffusion as well as thermo-mechanical stress, which results from different thermal expansion coefficients of copper and the silicon substrate.

Dies can be stacked in three different ways: Wafer-to-Wafer (W2W), Die-to-Wafer (D2W), and Die-to-Die (D2D). W2W approach has the advantage that it does not require pick-and-place operations for single dies and hance can be done faster than D2W and D2D. However, in the W2W approach, it is difficult to match Known Good Die of the wafers being stacked; therefore, the compound yield of W2W scheme is typically lower than those of D2W and D2D.

A number of 3D IC architectures have been considered for volume manufacturing.

- Interposer-based 3D ICs, called "2.5D ICs", in which multiple active dies are placed next to each other on top of a passive silicon interposer base and interconnected through it. The active dies are bonded to the interposer by means of micro-bumps. An example of a 2.5D ICs is depicted in Figure 1.6(a)), in which the interposer contains TSVs that provide connections to the external package pins. 2.5D ICs are attractive for high-performance compute and communication applications, as they offer high-bandwidth interconnect between the various active dies and good cooling opportunities [12, 13].

- "True" 3D ICs, with multiple dies stacked in a single tower, as shown in Figure 1.6(b). This type of 3D ICs is attractive for applications, where cooling is not an issue but the form factor needs to be optimized, for instance, for hand-held mobile devices.
- Combination of 2.5D ICs and 3D ICs, where multiple towers of active dies are mounted next to each other on a silicon interposer, as illustrated in Figure 1.6(c).

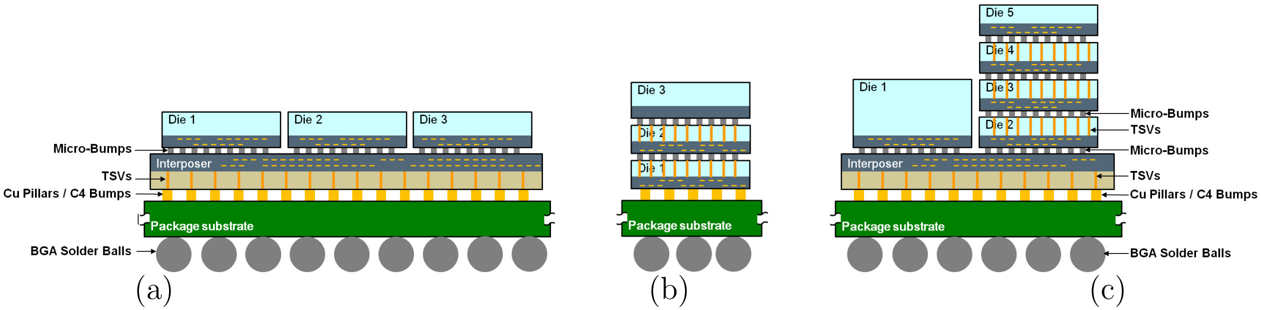


FIGURE 1.6: Various vertical die stack architectures: (a) 2.5D IC, (b) 3D IC, and (c) 5.5D IC [14].

The main advantages of 3D stacking is the possibility of integration heterogeneous dies and high-bandwidth interconnects. It is therefore no surprise that memory-on-logic stacks are among the first 3D IC applications appearing on the market [15, 16]. Recently, JEDEC, a microelectronics industry association, has released a new standard for Wide-I/O DRAMs for stacking with TSVs (JESD-229) [17]. This standard defines the mechanical and electrical logic-memory interface, allowing for 512-bit data signals. The standard also specifies boundary-scan-like test structures for DRAM

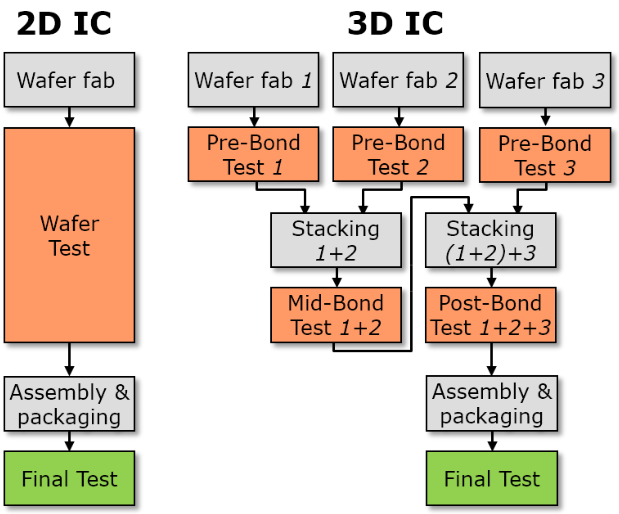


FIGURE 1.7: 2D (a) and 3D (b) test flows [18].

dies, enabling interconnect test for TSV-based connections between the logic and the memory dies [14].

1.2 3D Test Flow

Test flows for 3D ICs are distinctly different from a conventional test flow. Figure 1.7 shows a comparison between the two flows. A conventional (2D) IC is typically tested twice: (a) after wafer fabrication (wafer-level test) and after packaging (final test). A wafer-level test ensures that most of the defective dies are screened out before packaging in order to save cost, and a final test ensures the outgoing product quality.

A 3D-IC test flow has several potential test insertions [18].

- **Pre-bond test**

  All dies should be tested prior to bonding to ensure that only known good dies (KGD) are stacked. This test targets the internal logic as well as TSVs, which are processed before die bonding.

- **Post-bond test**

  Partial and full stacks can be tested for defects that may have been induced during the stacking process, which include defects in the TSV-based inter-die connections, as defects in the internal die areas surrounding TSVs.

- **Final test**

  As in traditional ICs, the entire 3D IC is thoroughly re-tested after packaging in order to ensure the outgoing product quality.

1.3 3D DfT Architectures and Optimization

In a typical 3D stack, only one die holds all external connections and the other dies communicate with the outside world through that die. Therefore, in order to provide test data to all dies in the stack, a 3D DfT architecture is required that can transport test data from external test pins through the stack to the die under test and back.

Marinissen et al. have proposed a 3D test architecture based on die-level wrappers for logic-on-logic 3D stacks [19], which has been extended to support memory-on-logic stacks [14]. Figure 1.8 shows a schematic view of the architecture. The die-level wrappers are based on embedded-core wrappers in IEEE Std 1500 [20] and allow for modular test of die-internal structures (INTEST mode), as well as inter-die connections (EXTEST mode). The architecture provides a serial test-access mechanism (TAM) and a parallel TAM for each die. The serial TAM is connected to an IEEE 1149.1 interface in Die 1 [21], and the parallel TAM is multiplexed onto functional I/Os in Die 1. Each die wrapper contains a wrapper instruction register (WIR) that configures the wrapper in one of three test modes: (a) INTEST, in order to test the die logic using internal scan chains, (b) EXTEST, in order to test the TSV-based connections using the wrapper boundary register, and (c) BYPASS, in order to bypass the die.

Dies in a 3D stack may contain embedded cores that need to be tested as separate entities. Typically, the width of the parallel TAM is not sufficient to provide test access to all cores of the stack simultaneously; therefore, only certain cores can be

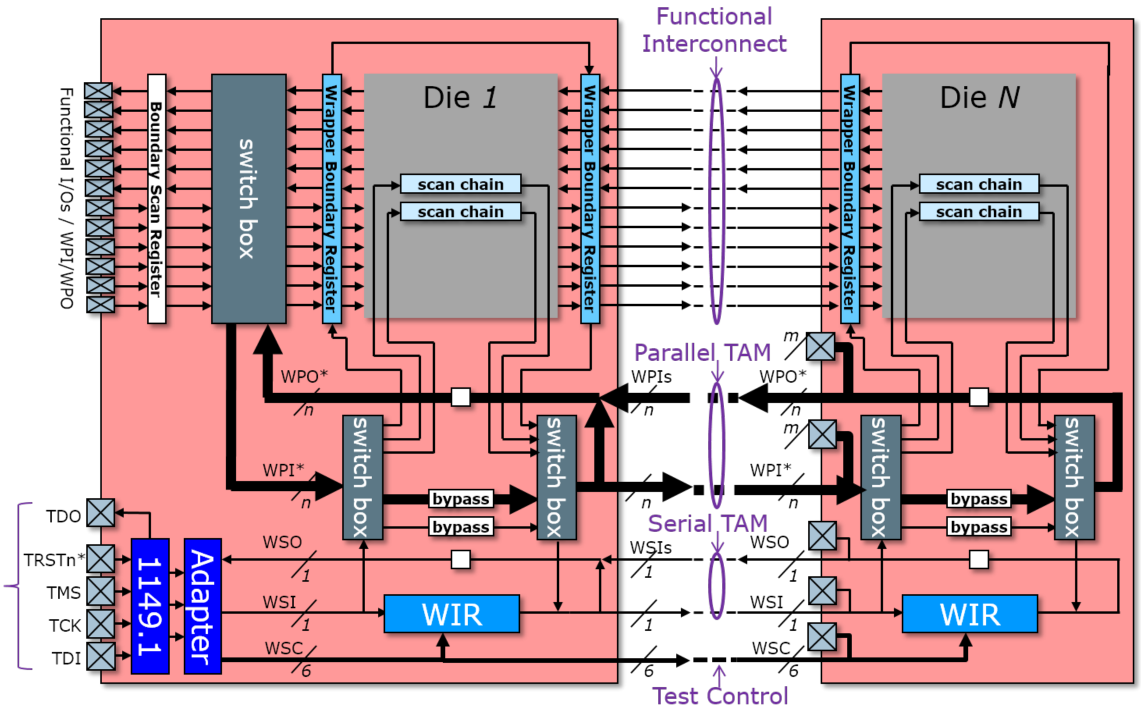


Figure 1.8: Schematic view of the logic-on-logic 3D DfT architecture [14].

tested in parallel, depending on the design of the TAMs. In order to optimize the total test time of a 3D IC, optimization of the test architecture and test scheduling is necessary.

Optimization of 2D system-on-chip (SoC) test architectures has been explored well in the past [22, 23, 24, 25, 26]. A general problem of TAM optimization can be formulated as follows. Given the total TAM width , as well as the parameters for each core in the SoC, including scan-chain information and the test-pattern information, determine an optimal assignment of cores to the global TAM such that the test time is minimum.

Iyengar et al. [27] have proposed a framework for co-optimization of TAM design and wrappers. This work assumes a *test-bus* model for the TAMs, where cores can be tested independently from each other; however, cores sharing a TAM are tested sequentially. The paper presents an efficient algorithm to construct wrappers that reduce the core-testing time, as well as mathematical models for TAM optimization that use the calculated core-testing times as input parameters. The TAM optimization for small SoCs is solved using integer linear programming (ILP). For large SoCs, the authors have proposed an efficient heuristic based on rectangle stacking [28]. Each core can be represented as a rectangle, where the width is the TAM width of the core, and the length is the test time, as shown in Figure 1.9(a). The objective is to pack the rectangles without overlapping in a bin, the width of which equals the maximum TAM width, such that the length of the bin, i.e., the SoC testing time, is minimized. Once an optimal packing is obtained, it can be mapped back to the corresponding core-TAM assignment, as shown in Figure 1.9(b).

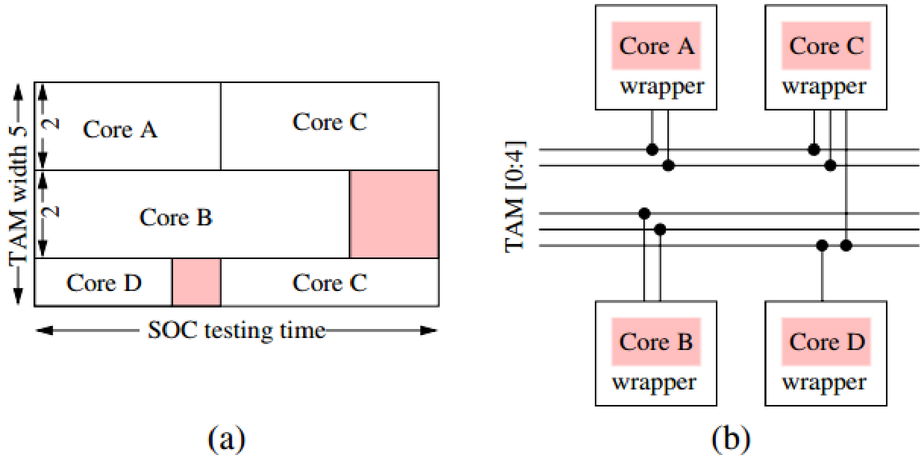


FIGURE 1.9: An illustration of (a) rectangle packing and (b) the corresponding TAM design [28].

The authors of [29] have addressed the issue of robustness of test architectures against variations in input parameters and proposed to solve the problem of robust optimization by using a daisy-chain architecture, matching the width of the core-level TAM to that of the system-level TAM, and testing one core at a time. The requirement of reconfigurable core wrappers is, however, difficult to meet in practice because of complex scan-chain routing. In addition, some cores do not benefit from extra TAM width, wasting the bandwidth that could be used for testing other cores in parallel.

Recent publications have been focusing on optimization of test architectures for 3D ICs [30, 31, 32]. Noia et al. [32] have proposed a framework for test-architecture optimization for 3D stacked ICs with *hard*, *soft*, and *firm* dies. In a *hard* die, the test architecture is fixed and known in advance. *Firm* dies allow for serial/parallel conversion in order to adjust the TAM width. In the case of *soft* dies, the test architecture is to be designed, which offers extra degrees of freedom. In addition, two different 3D test architectures are considered: (a) serial and (b) parallel test architectures, as shown in Figure 1.10. In a serial architecture, all dies share TAM wires and hence only one die can be tested at a time. In a parallel test architecture, certain dies have a dedicated TAM and can be tested independently of other dies, for instance, Die 1 in Figure 1.10(b). This can reduce the total test time but typically requires a wider stack-level TAM. The authors formulate a mathematical model of the problem, using the number of available test pins and the number of dedicated test TSVs as constraints, and solve the problem using ILP.

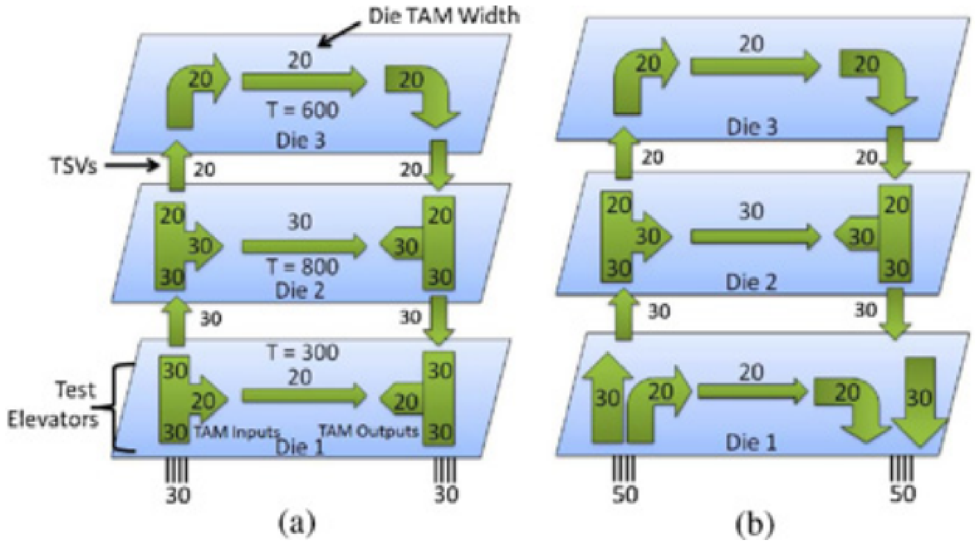


FIGURE 1.10: An example of a 3D test architecture with (a) serial TAM and (b) parallel test architecture [32].

1.4 Defects due to TSV Manufacturing

As in any microelectronics, the manufacturing process of 3D ICs is defect-prone and hence 3D ICs need to be tested to ensure the outgoing product quality. Since this manufacturing process includes steps that are also used for conventional (2D) chips, the same defects may occur in both 2D and 3D products. A defect is a physical imperfection in the processed wafer, for instance an open in interconnects, a short between interconnects, missing transistor, incorrect doping level.

Due to TSV manufacturing and bonding steps, 3D ICs can potentially have a number of new defect types [6]. TSV-related defects include:

- Voids in TSVs
- Pinch-off of the TSV
- Oxide defects, e.g. pinholes
- Thermo-mechanical stress induced defects
- Voids and cracks in micro-bumps

Voids in TSVs, as shown in Figure 1.11(a), can result from Cu electroplating or insufficient wetting of the vias in the plating solution [33]. Another defect due to plating is *pinch-off* of the TSV, as depicted in Figure 1.11(b). These defects can increase the TSV resistance or even create an open circuit.

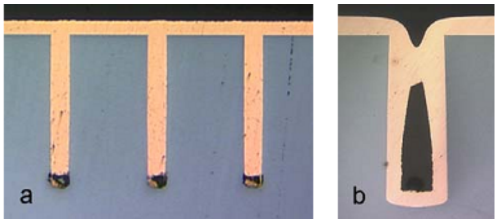
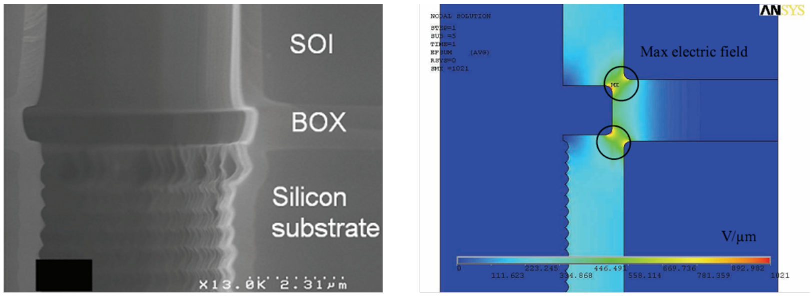


FIGURE 1.11: TSV plating defects: (a) voids, (b) pinch-off [33].

Oxide defects, such as pinholes, may occur along the TSV wall and create shorts between the TSV and the substrate [34].

Breakdown voltage reduction of the thermal oxide is caused by strong buried oxide (BOX) undercuts, an example of which is depicted in Figure 1.12(a) [35]. BOX undercuts result from TSV etching, post-etch cleaning, and insulator layer deposition. Figure 1.12(b) highlights the effect of a strong undercut: the electric field at the corner of the silicon-on-insulator is greatly increased. This degrades the breakdown voltage, which must be sufficiently high for high voltage applications (in the 200 V range) [35].



(a) (b)

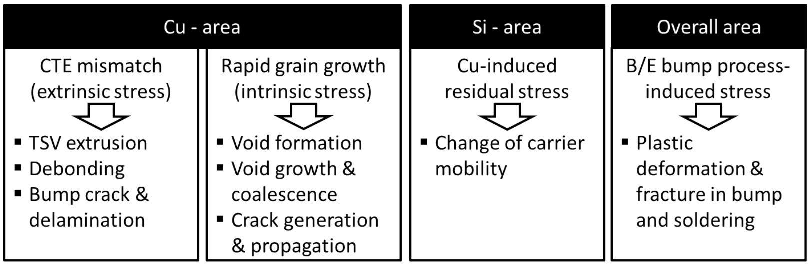
FIGURE 1.12: (a) BOX undercut, (b) Simulated electric field [35].

TSV reliability can be degraded by stress resulting from thermal expansion mismatch between Cu, Si, and SiO_2. Resulting failures can be categorized in terms of the stress origin: (a) Cu-area, (b) Si-area, and (c) overall area, as shown in Figure 1.13 [36].

(a) Although the strains in Cu are not high enough to cause failures in perfect TSV structures, the combination of these strains and manufacturing imperfections in TSVs might lead to failures, for instance *interfacial delamination*, *micro-bump cracks*, and *cracks in TSVs* [36, 37, 38, 39].

(b) Local stress in Si can alter the carrier mobility of a MOS device, depending on its location relative to the TSVs [40]. This effect causes *timing variations* of up to ±10% for individual cells in the TSV proximity, which may increase the critical path delay. This affect the test quality for small-delay defect (SDD) test [41], as the SDD test relies on an accurate timing model of the circuit. Recently, a study has shown that the SDD-test quality can be significantly decreased if stress due to TSVs is not taken into account during test generation

(c) Global stress results in die warpage which, in turn, might lead to cracks in the micro-bumps, as shown in Figure 1.14 [36].



Cu - area

CTE mismatch (extrinsic stress)

- TSV extrusion
- Debonding
- Bump crack & delamination

Rapid grain growth (intrinsic stress)

- Void formation
- Void growth & coalescence
- Crack generation & propagation

Si - area

Cu-induced residual stress

- Change of carrier mobility

Overall area

B/E bump process-induced stress

- Plastic deformation & fracture in bump and soldering

FIGURE 1.13: TSV stress types and their induced failures [36].

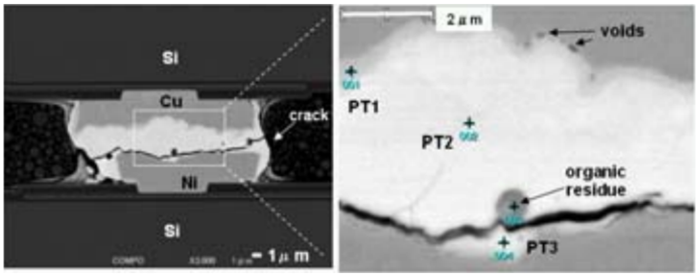


FIGURE 1.14: Crack in a micro-bump due to mechanical stress [36].

An important issue in 3D ICs is electromigration in micro-bumps, which are part of the TSV-based interconnects. Studies of micro-bump reliability report that *voids* and *cracks* can occur in the joints due to electromigration [42]. For example, Figure 1.15 shows such defects in a chip sample with an IMC–Sn–IMC (Intermetallic Compound) joint structure after 0.13A current stressing [42]. The resulting voids and cracks increased the resistance of this micro-bump by more than 20%.

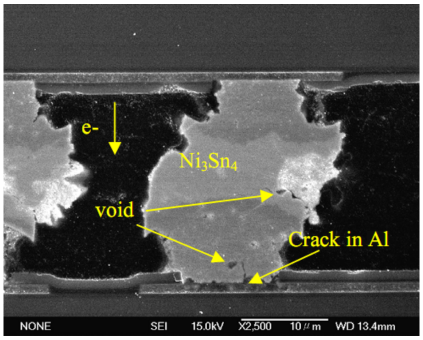


FIGURE 1.15: Micro-bump failure after current stress test [42].

Many of the listed defects affect the reliability of 3D ICs, for instance, by creating shorts or increasing the interconnect resistance. The phenomena causing these defects have been studied and the underlying physics has been explained in the literature [43, 44]. The studies also propose new design and manufacturing techniques to avoid these defects, for instance

- Layout optimization for reduced mechanical stress [45],
- Layout optimization with respect to carrier mobility variations near TSVs [40],
- Void-free filling techniques [46].

The efficiency of these techniques has been experimentally verified but volume production data is still awaited. It is unlikely that the industry is going to reveal any significant data about their manufacturing processes or yield, hence testing remains relevant.

1.5 Pre-Bond TSV Test

Pre-bond TSV testing remains one of the major challenges in a 3D test flow due to the limited access to TSVs [47]. In the following paragraphs, we review several methods for pre-bond TSV test that have been proposed in the literature.

Noia and Chakrabarty proposed a method for TSV testing in which multiple TSVs are mechanically contacted by the same probe needle to measure TSV capacitance and resistance [48, 49]. This approach allows for the testing of multiple TSVs simultaneously at the expense of measurement resolution, significantly reducing the test time. Simulation results have demonstrated high measurement accuracy, even in the presence of process variations and probe contact variations. Despite its benefits, this method has several drawbacks. First, it places extra burden on the test equipment, such as custom and active probe cards. Second, the method requires multiple contacts on the back side of the thinned wafer, which can be difficult in practice. Finally, mechanical force on TSV tips and micro-bumps can result in damage to TSVs, leading to their degraded performance or even failure.

Chen et al. have proposed a methodology for detecting capacitive TSV faults [50]. In this method, a TSV is charged to a certain voltage level and then its charge is shared among a number of TSVs. After charge sharing, a sense amplifier measures the voltage of TSV, from which the TSV capacitance can be deduced. This method also allows for the detection of leakage faults. A major drawback of this approach is its susceptibility to process variations. In addition, this method requires analog structures on the die that are not part of typical standard cell libraries and need to be manually designed and optimized.

Pak et al. have developed a technique for TSV connectivity test using ring oscillators [51]. The key idea of this approach is to create on-chip oscillators from inverters and to use TSVs as capacitive loads, which are connected through MOSFET switches to the oscillators. The number of toggles of the oscillating signals is counted twice in a fixed time frame: once, when the switch of the corresponding TSV under test is “on”, and once, when the switch is “off”. If the difference in the count is zero, the TSV is considered as disconnected from the metal layers. However, the work in [51] does not show whether this approach can be used to detect weak resistive-open faults and weak leakage faults in TSVs.

Huang et al. have developed a solution for detecting resistive open faults and leakage faults in TSVs using ring oscillators [52]. Their method is similar to the so-called "input sensitivity analysis", which was originally designed for post-bond TSV diagnosis using ring oscillators [53]. The disadvantage of the method in [52] is that it requires modification of functional TSV I/O cells that are carefully designed for performance and robustness.

Another method proposed by Huang et al. uses phase-locked loops (PLLs) for TSV leakage binning [54]. The key idea of this method is to measure the discharge time of a "floating" TSV. According to the simulation results, the method provides a high resolution even for very weak leakage below 1 μA. However, it needs to be shown that the method is robust against random-process variations. In addition, this method cannot detect resistive-open faults and requires modification of the functional I/O circuitry driving the TSV.

Hao and McCluskey proposed very-low-voltage testing to detect resistive shorts and hot-carrier-induced degradation [55]. They have shown that the effects of these defects are increased at voltage levels lower than V_{DD}; this finding has been supported by analytical models and SPICE simulations [56, 57]. This prior work motivated us to consider using multiple voltage levels for pre-bond TSV test. Since multiple-voltage testing neither imposes limits on the test equipment nor requires extra on-chip DfT structures, it can be applied in practice without introducing additional hardware costs [58, 59]. Since pre-bond TSV test does not require long test sequences (scan data), test time does not grow significantly if multiple voltages are used.

1.6 Post-Silicon Debug Using Signal Tracing

Despite steady progress in pre-silicon verification methods and tools [60, 61, 62], first silicon is rarely bug-free. Silicon debug requires a relatively large engineering effort, accounting for a significant portion of the total time-to-market of the silicon product and this portion has been projected to grow [7, 8]. Several design-for-debug solutions have been proposed in the past to provide observability of a circuit's internal signals [63, 64, 7, 65]. Most of these methods are based either on *scan dumps* or *signal tracing*.

A scan dump is a snapshot of the circuit's state at a particular point of time [66, 67, 68]. It is obtained by freezing the clocks during execution of a functional test and scanning out the content of the scan flip-flops. Analysis of the obtained values can help to localize errors in the design. A major difficulty of this method is the implementation of a cycle-accurate, deterministic clock freeze in the module under test. In addition, scan dump is destructive, which requires a reset of the circuit and rerunning the test program in order to obtain another snapshot.

Signal tracing is a commonly-used technique for post-silicon debug. The main idea of this method is to localize bugs in digital logic by observing internal nodes of the circuit during test-program execution. The tapped signals can be captured either on-chip or off-chip [64]. Figure 1.16 shows a traditional architecture for signal tracing. Busses of internal signals from different modules feed through a multiplexer tree into an on-chip trace buffer. The multiplexer select a module, the signals of which will be captured into the trace buffer at-speed during a functional test. After the capturing of data is complete, the content of the trace buffer is transferred to an external debug equipment through a debug interface, for instance, an IEEE 1149.1 [21] interface. The major limitations of this approach are (a) the requirement of extra chip pins, and (b) low bandwidth of the external equipment, which limit the amount of signals that

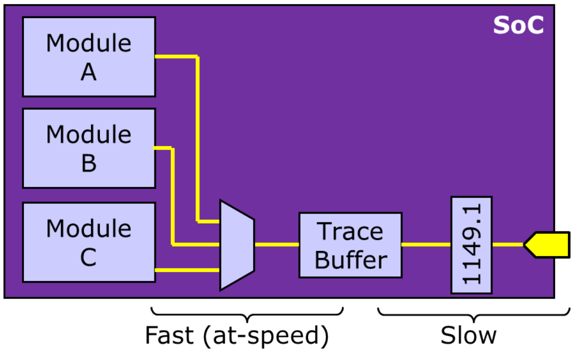


FIGURE 1.16: Traditional signal-tracing architecture.

can be sampled at-speed. Alternatively, the traced data can be temporarily stored in on-chip trace buffers that are read after program execution. Due to the limited size of the trace buffers, only a few signals can be observed over a relatively short period of time, which is typically a fraction of the total run time of a test program.

In order to overcome these limitations, a number of innovative methods have been proposed in the past. In [69], the authors reconstruct the values of more signals than actually traced, virtually increasing the number of signals observed. The authors of [70] propose a method for selection of signals for tracing that are most susceptible to possible errors. In [71], several signal-selection methods have been proposed as well as a metric described to quantify the observability of each technique. Another type of enhancement of signal tracing incorporates compression before the storage of trace data into a trace buffer[72, 73]. These methods can be effectively combined with other methods that increase the observation window.

The authors of [74] propose an iterative debug scheme. In the first iteration, time intervals in which erroneous data is captured are identified using lossy compression. In the following iterations, the method zooms into these intervals for a better resolution. The disadvantage of this method is the requirement to run multiple iterative debug sessions with intermediate post-processing steps.

In [75], the authors use a three-pass debug methodology in order to expand the observation window. In the first pass, the error rate is calculated using parity bits. In the second pass, suspect clock cycles are determined. In the third pass, the erroneous data is captured during suspect clock cycles. The price for an expanded observation window is extra debug passes and the need for post-processing of captured data in between these passes.

In [76], a generic debug infrastructure is proposed to gather trace information from different modules through a "Trace Memory Controller" and redirect it to different targets, for instance, a dedicated debug port, SRAM, or system memory. The solution described in [76] does not cover trace-data compression or selective trace-data storage and leaves these implementation details up to the user.

2

TSV Stress-Aware ATPG

Despite the numerous benefits offered by 3D integration, test challenges for 3D ICs must be addressed before volume manufacturing and defect screening can be feasible [4, 5]. One of the serious problems confronting 3D integration is that of thermo-mechanical stress due to TSV processing. The thermal expansion coefficient of copper, a common TSV fill material, is significantly higher than that of silicon: $17 \times 10^{-6}/K$ versus $3 \times 10^{-6}/K$ [77]. Due to this mismatch, TSVs are likely to cause residual stress in the silicon during fabrication and thermal cycling. One of the effects of thermal stress is mobility variation in MOS devices in the proximity of TSVs. These variations lead to a change in the timing profile of the circuit [40, 45], which affects delay-fault testing.

Recent work on 3D IC testing has targeted solutions to overcome problems related to test access in 3D ICs and TSV testing. We focus here on post-bond delay-fault testing of internal die logic in 3D ICs, a problem that has received much less attention in the literature. In this chapter, we study the impact of timing variations due to TSV stress on the quality of test patterns generated to screen small-delay defects (SDDs). In particular, we focus on the following problems: (i) How severe is the impact of

TSV-induced stress on the effectiveness of patterns for SDDs and test escapes? (ii) To what extent can test escapes be reduced by including analytical TSV stress models as a preprocessing step in the ATPG flow? (iii) What is the impact of TSV stress-aware ATPG on pattern count and how does the process yield affect test escapes due to TSV-induced stress?

We assume that SDD testing is done after stacking, such that the clock tree for functional operation is available for at-speed capture cycles. We show that the use of TSV stress-oblivious circuit models results in a significantly increased escape rate of faulty chips. The level of this increase depends on the yield of the fabrication process; we conclude that accurate modeling of TSV stress is more important for processes with lower yields.

The impact of TSV stress on pattern effectiveness is quantified using the statistical delay quality level (SDQL) metric [78]. This is a key metric in our approach, since the SDQL of a chip correlates with the expected test escape rate due to small-delay defects. We also show that the test escape can be reduced considerably by incorporating TSV stress in cell timing libraries and using these libraries with a commercial timing-aware ATPG tool. Therefore, any detrimental impact of TSV stress on pattern effectiveness and test quality can be overcome by using stress-aware models for test generation. We also show that TSV stress-aware testing leads to negligible increase, if any, in pattern count.

The remainder of this chapter is organized as follows. In Section 2.1, we give an overview of small-delay testing, and mobility variations due to TSV stress. Section 2.2 describes our methodology to create TSV stress-aware test patterns using conventional ATPG tools. In Section 2.3, we present simulation results obtained with 3D logic-on-logic benchmarks. Finally, Section 2.4 concludes this chapter.

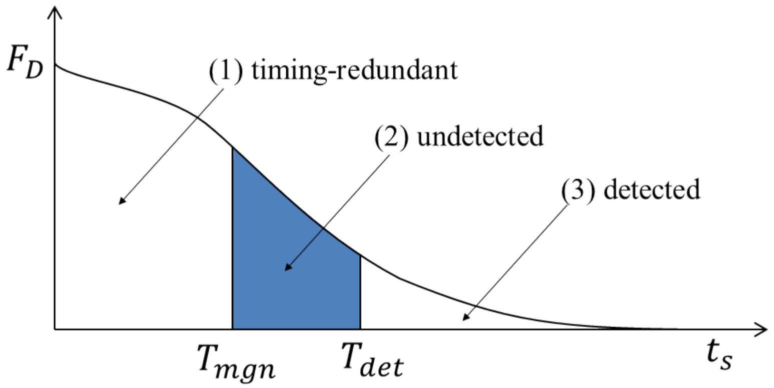


FIGURE 2.1: Delay distribution $F_D(t_s)$.

2.1 Related Prior Work

2.1.1 SDD Testing and SDQL

Due to continuous miniaturization, integrated circuits have become more susceptible to process variations and resistive defects. As a result, SDDs have become more prevalent [41]. Despite the fact that their size might be small compared to the clock period, SDDs can cause errors if the length of a path affected by them exceeds the clock period. Therefore, effective and low-cost screening for SDDs is important to ensure product quality. Several methods have been proposed in the literature to test for SDDs [79, 41, 80].

To quantify the effect of TSV stress on the quality of a delay-fault test, we use the statistical delay quality level (SDQL) proposed in [78]. With this metric, we can quantitatively estimate the test escape rate due to delay defects and evaluate the increase in test quality if TSV stress-aware circuit models are used for ATPG.

SDQL computation is based on the assumption that delay defects follow a probability distribution $F_D(t_s)$, where t_s is the size of the defect, and that the defects are equally distributed over all sites. This distribution is dependent on the manufacturing process and can be obtained by analyzing manufacturing data [81].

Figure 2.1 shows an example of a delay-defect distribution function. For each delay fault, this function is divided intro three regions by T_{mgn} and T_{det}, which are defined as follows. The time margin T_{mgn} for a fault is the slack on the longest of the paths that can propagate this fault:

$$T_{mgn} = T_{ck,f} - \max_i(T_i), \tag{2.1}$$

where $T_{ck,f}$ is the functional clock period and T_i are lengths of the sensitizable paths. A fault can only be detected if its size T_{det} exceeds the slack of the path sensitized by a particular test:

$$T_{det} = T_{ck,t} - T_{sens}, \tag{2.2}$$

where $T_{ck,t}$ is the test clock period and T_{sens} is the length of the sensitized path. As Figure 2.1 shows, delay faults can be put into three categories dependent on their size t_s: (1) timing-redundant, (2) undetected, and (3) detected. The area below the curve in the undetected region represents the probability of the fault being undetected and escaping the test. The summation of these probabilities for all faults is called SDQL [78]:

$$SDQL = \sum_{k}^{2N} \int_{T_{mgn}}^{T_{det}} F_D(t_s)\, \mathrm{d}t_s, \tag{2.3}$$

We use the SDQL metric to show that the use of TSV stress-oblivious circuit models may lead to a significantly increased escape rate for 3D ICs.

2.1.2 Mobility Variation due to TSV Stress

Due to a mismatch in thermal expansion coefficients of copper and silicon, TSVs cause thermo-mechanical stress in the surrounding silicon. This stress affects not only the mechanical device reliability but also material properties such as carrier mobility [38], which results in timing variations of the devices. Recent studies have reported up to ±10% variations for individual cells [40].

Since a correct timing model of the circuit is crucial for delay testing, we need an efficient methodology to take TSV stress into consideration in the ATPG flow.

In the literature, we can find a simple closed form formula for the thermo-mechanical stress caused by a TSV, known as the Lamé stress solution [40]. However, this model is 2D in its nature, capturing only the information in the x and y directions on the wafer surface, and it fails to capture the true 3D nature of the TSV stress field near the wafer surface. Since there is no simple formula for the 3D stress field available in the literature, we can apply the methodology outlined in [45] for full chip analysis. The main idea of this methodology is to perform a finite-element analysis (FEA) for a single TSV and use linear superposition to estimate the total stress σ_{rr} due to multiple TSVs.

The resulting σ_{rr} serves as an input to compute the carrier mobility variation, which can be expressed as a function of σ_{rr} and the device channel orientation θ with respect to the TSV [40]:

$$\frac{\Delta\mu}{\mu}(\theta) = -\Pi \times \sigma_{rr} \times \alpha(\theta), \tag{2.4}$$

where $\alpha(\theta)$ is the orientation factor as a function of the angle θ between the channel orientation and the center of the TSV, and Π is the piezo-resistive coefficient at $\theta = 0$. Π can be extracted using the methodology described in [82].

The estimated carrier mobility change can be used to update the timing information of the devices around TSVs for a more accurate circuit model.

2.2 Methodology

Our approach consists of two major parts:

1. Generation of a TSV stress-aware circuit model;

2. Test pattern generation and simulation.

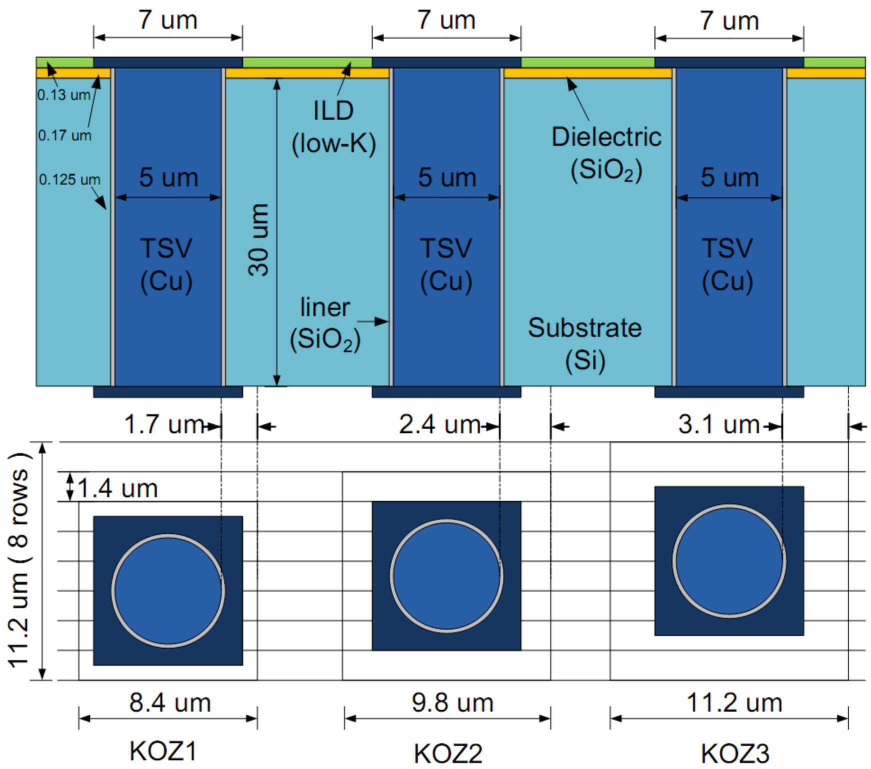


FIGURE 2.2: The TSV structure and three different KOZ sizes used in our studies.

The rest of this section describes the two steps in detail.

2.2.1 TSV Stress-Aware Model Generation

The TSV structure considered in our simulations is shown in Figure 2.2. The TSV diameter, height, landing pad size, and liner thickness are assumed to be 5 μm, 30 μm, 7 μm, and 125 nm, respectively. The TSV is assumed to be made of copper, and the liner of SiO_2. The material properties used in our simulations are: CTE (ppm/K) for Cu = 17, Si = 2.3, SiO_2 = 0.5; Youngs modulus (GPa) for Cu = 110, Si = 130, SiO_2 = 71. We also consider three different keep-out-zone (KOZ) sizes of 1.7 μm, 2.4 μm and 3.1 μm as shown. This corresponds to 6, 7 and 8 standard cell rows in the Nangate 45 nm technology library. A larger KOZ will mean less impact of TSV stress on the gates. However, increasing the KOZ will affect other design metrics such as area and wirelength.

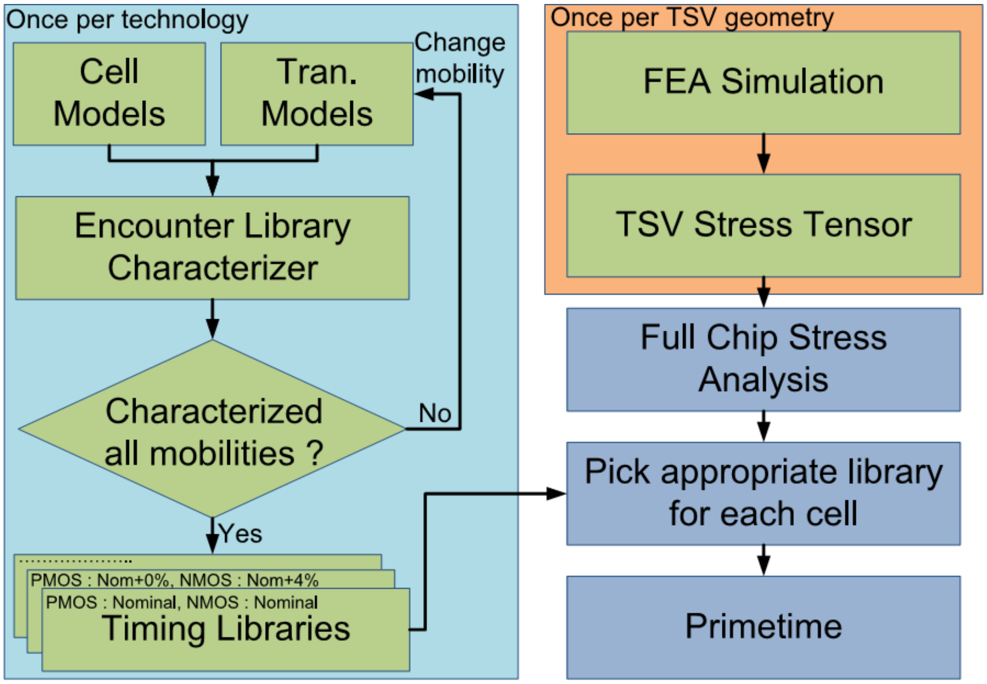

Once per technology
Cell Models
Tran. Models
Change mobility
Encounter Library Characterizer
Characterized all mobilities ?
No
Yes
....
PMOS : Nom+0%, NMOS : Nom+4%
PMOS : Nominal, NMOS : Nominal
Timing Libraries
Once per TSV geometry
FEA Simulation
TSV Stress Tensor
Full Chip Stress Analysis
Pick appropriate library for each cell
Primetime

FIGURE 2.3: The design flow used to obtain TSV-stress-aware timing

The overall design flow used to obtain stress aware timing is shown in Figure 2.3. We first start with creating a timing library with different mobility values. We start with the nominal PMOS and NMOS mobility, and characterize all the cells in increments of 4%. The next step is stress calculation, and this is performed as outlined in [45]. We first perform FEA simulation of the stress generated by a single TSV using the FEA software ABAQUS [83]. At any given point in the chip, the stress can be represented by its nine-component stress-tensor as follows:

$$\sigma = \sigma_{ij} = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{bmatrix}$$

The first index i indicates that the stress acts on a plane normal to the i axis, and the second index j denotes the direction in which the stress acts. In cylindrical coordinates, the three indices $1, 2$, and 3 represent r, θ, and z respectively. Since the stress of a single TSV is radially symmetric, we only need to store the stress tensors along one arbitrary radial line. The steps outlined so far only need to be performed once and the results can be used for any design.

For any given design, we need to perform full-chip stress analysis. For computation of stress due to multiple TSVs, we use linear superposition. The stress at each point in the chip is simply the vector sum of the stress caused by all TSVs at that point. A vector sum of the stress components is performed by transforming the cylindrical stress tensor into its Cartesian form, adding the components individually along the x, y and z axis, and then transforming it back to cylindrical coordinates.

With the stress tensor at each point in the design, the corresponding change in mobility of electrons (NMOS) and holes (PMOS) can be computed using the measured piezoresistive coefficients given in [82], assuming (100) silicon. With this approach, we obtain the change in mobility due to TSV stress for each cell in the design.

The appropriate timing library can then be picked from the pre-characterized set of timing libraries. All the libraries, netlists, and parasitics are fed into Synopsys Primetime to get TSV-stress-aware timing results.

2.2.2 TSV-Stress-Aware ATPG

Delay-fault ATPG relies on correct circuit timing information, which is used to generate the path profile in order to target the longest sensitizable paths for each fault. In 3D ICs, timing variations in devices due to TSV stress might occur, resulting in a change of the timing profile of the circuit. The slacks for paths that include the the affected devices can increase or decrease dependent on the device type (PMOS or NMOS) and on the location relative to the TSVs, changing the longest sensitizable path for certain delay faults. If these changes are not taken into account, the ATPG tool might propagate faults through paths that are shorter than the actual longest path. This will invariably result in a lower test quality, since delays of a particular size will not be detected.

To evaluate the impact of the TSV-induced stress on the test quality, we have developed a tool flow using conventional timing analysis and ATPG tools: Synopsys PrimeTime and TetraMax, respectively. Figure 2.4 gives an overview of the flow. As input, we use the original (non-stress-aware, NSA) models and the modified (stress-aware, SA) models. First, we perform timing analysis with PrimeTime to extract the slack data. Next, we generate two delay test pattern sets with TetraMax: one using the NSA and the other using SA models. Finally, we perform fault simulation and compute SDQL with TetraMax using the following combinations.

1. The NSA pattern set on the NSA model. The results of this simulation show the test quality that is expected if TSV stress is not present.

2. The NSA pattern set on the SA model. The simulated SDQL numbers indicate the actual test quality of the NSA pattern set.

3. The SA pattern set on the SA model. The simulated SDQL numbers indicate the actual test quality of the SA pattern set, i.e., under realistic conditions of TSV stress.

2.3 Case Study

2.3.1 Test Vehicles

We use three benchmarks taken from the open cores benchmark suite [84]. They are synthesized and scan-inserted using the Nangate open cell library, at the 45 nm node. Table 2.1 gives an overview of the design data, including gate, scan flip-flop and TSV count. We partition the netlist and create three different stacks for each core: two-die, three-die, and four-die stacks. For each die, we use a 3D force-directed placer to place the gates [85]. This placer places TSVs in a regular fashion, and assigns nets to TSVs using a 3D Minimum Spanning Tree approach.

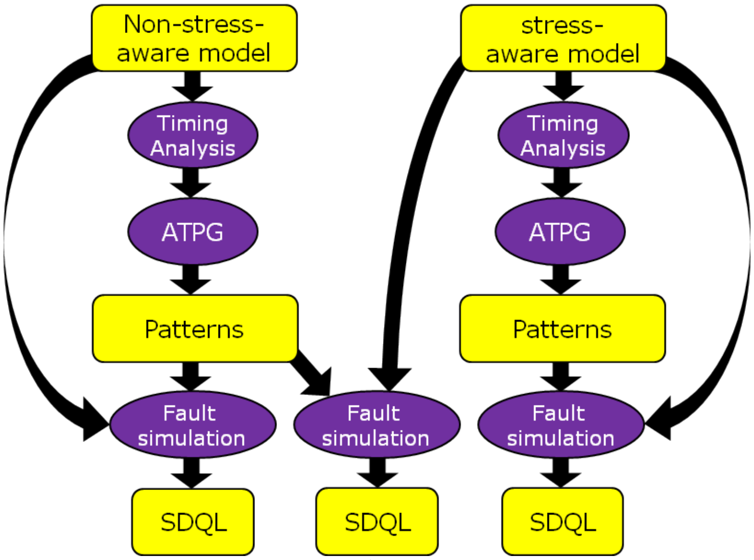


FIGURE 2.4: ATPG tool flow.

Table 2.1: Design statistics for the benchmarks.

Benchmark	# Gates	# Scan FFs	3D Impl.	# TSVs
des_perf	26,251	1,984	2-die	419
			3-die	884
			4-die	1322
cf_rca_16	156,624	20,480	2-die	589
			3-die	676
			4-die	953
cf_fft_256_8	299,273	75,723	2-die	2193
			3-die	4717
			4-die	5843

Once we have the placement result, we route each die separately in Cadence Encounter. Assuming the TSV resistance and capacitance to be 50 fF and 50 mΩ, respectively [86, 87], we carry out timing analysis in Synopsys Primetime. We treat dies as modules, and TSVs as top level interconnections. Primetime is also used to get die-level timing constraints, and these are used to perform timing optimization in Cadence Encounter. We also obtain stress aware timing from the methodology outlined in Section 2.2.1. Some sample layouts of des_perf are shown in Figure 2.5, along with the obtained PMOS mobility maps.

FEA simulation and library characterization took a relatively long time: eight hours and 180 hours, respectively. However, these steps need to be done just once per technology and the results can be re-used for all designs mapped to this technology. The actual mobility computation only took several minutes per layout.

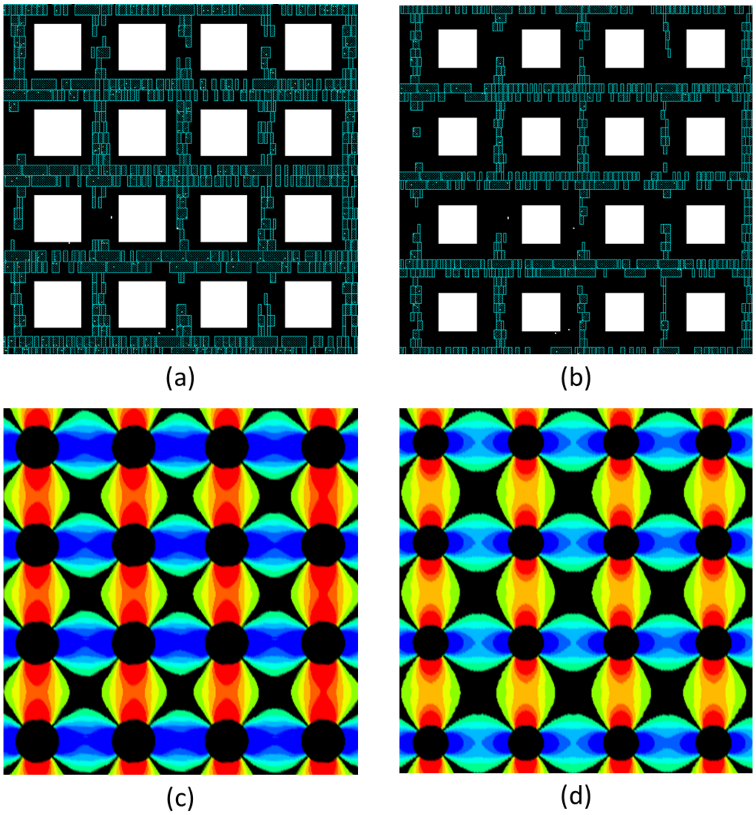


FIGURE 2.5: (a,b) Layout screenshots for KOZ1 and KOZ3 of benchmark des_perf. White rectangles indicate TSVs, and blue indicates standard cells. (c,d) The corresponding PMOS mobility maps. Black indicates TSVs. Green, red and blue represent nominal, positive, and negative mobility change respectively.

2.3.2 ATPG Results

We applied the flow depicted in Figure 2.4 to different benchmarks. In this simulation, we used the distribution function that the authors of [78] obtained by analysing experimental data presented in [81]. We assume that delay defects greater than 10 ns are screened out by stuck-at tests, therefore the probability of delay defects with

$s > 10$ ns is zero. The function was normalized such that the area under the curve is unity:

$$\int_0^\infty F_D(t_s)\,\mathrm{d}t_s = 1$$

The resulting probability distribution function is:

$$F_d(t_s) = \begin{cases} 1.97e^{-2.1t_s} + 0.005 & \text{if } 0 \leqslant t_s \leqslant 10 \\ 0 & \text{else} \end{cases}$$

Table 2.2 shows the test escape rate of the chips expressed in defective parts per million (DPPM). The DPPM is calculated as SDQL normalized by the number of delay-faults N_f:

$$DPPM = \frac{1}{N_f} SDQL$$

The results indicate that the actual escape rate of the NSA patterns (Row 2) is significantly higher than the estimated one (Row 1). This implies that neglecting TSV-stress in the ATPG flow by using a NSA model leads to decreased test quality.

The results presented in Table 2.2 also show that the test escape rate of the optimized patterns (Row 3) almost equals the one initially targeted (Row 1). Therefore, if TSV stress is taken into account during ATPG, it has no significant impact on the test quality. This is also true for small keep-out zones: regardless of the KOZ size, the test quality can reach the level of a circuit without the timing variation effects caused by TSV stress.

Table 2.3 shows the number of test patterns generated for different combinations. We observe that there is no significant variation in pattern count between the NSA and SA patterns for the same implementation. This implies that a stress-aware ATPG flow has negligible impact on pattern count compared to a conventional ATPG flow. In addition, we observe that the size of the KOZ has no impact on pattern count. In addition to the simulation described above, we applied the flow to the benchmarks

Table 2.2: Defective parts per million for various designs.

Design		Patterns - Model	KOZ1	KOZ2	KOZ3
des_perf	2-die	NSA patterns - NSA model	24.8	25.0	27.2
		NSA patterns - SA model	177.1	180.0	190.7
		SA patterns - SA model	23.9	23.7	28.1
	3-die	NSA patterns - NSA model	35.8	38.1	43.9
		NSA patterns - SA model	101.1	110.3	120.7
		SA patterns - SA model	34.1	38.7	42.8
	4-die	NSA patterns - NSA model	41.5	46.1	47.7
		NSA patterns - SA model	183.1	203.1	219.3
		SA patterns - SA model	40.8	44.6	47.7
cf_rca_16	2-die	NSA patterns - NSA model	4.05	4.26	4.38
		NSA patterns - SA model	5.70	6.00	6.16
		SA patterns - SA model	4.17	4.41	4.52
	3-die	NSA patterns - NSA model	5.00	5.19	5.20
		NSA patterns - SA model	7.81	8.00	8.01
		SA patterns - SA model	4.83	4.90	4.81
	4-die	NSA patterns - NSA model	4.70	5.20	4.27
		NSA patterns - SA model	7.55	8.22	6.87
		SA patterns - SA model	4.63	5.31	3.95
cf_fft_256_8	2-die	NSA patterns - NSA model	608.82	578.80	590.85
		NSA patterns - SA model	935.97	921.75	934.07
		SA patterns - SA model	617.55	578.41	591.05
	3-die	NSA patterns - NSA model	597.53	591.61	608.31
		NSA patterns - SA model	907.90	901.32	941.54
		SA patterns - SA model	604.73	592.37	609.87
	4-die	NSA patterns - NSA model	741.06	761.206	733.96
		NSA patterns - SA model	1072.54	1126.64	1099.41
		SA patterns - SA model	761.27	775.97	739.12

using the simplified distribution function $F_d(t_s) = \lambda e^{-\lambda t_s}$ with different values of λ. Small values of λ correspond to a flatter function $F(s)$, which implies that larger delays are more likely to occur. Therefore, smaller λ should lead to higher test-quality degradation of non-optimal (non-stress-aware) patterns, which can be expressed as the DPPM ratio r_D between the DPPM values of the NSA and SA test patterns:

$$r_D = \frac{P_{escape}(\text{NSA})}{P_{escape}(\text{SA})} = \frac{SDQL(\text{NSA})}{SDQL(\text{SA})} \tag{2.5}$$

This conclusion is consistent with the simulated values of the DPPM ratio r_D depicted

Table 2.3: Pattern count for various designs.

Design		Patterns	KOZ1	KOZ2	KOZ3
des_perf	2-die	NSA patterns	5761	5672	5722
		SA patterns	5632	5666	5789
	3-die	NSA patterns	4382	4400	4468
		SA patterns	4373	4356	4417
	4-die	NSA patterns	3114	3166	3199
		SA patterns	3057	3080	3191
cf_rca_16	2-die	NSA patterns	9884	9697	9631
		SA patterns	10070	10007	9955
	3-die	NSA patterns	7770	7873	7818
		SA patterns	7865	7840	7912
	4-die	NSA patterns	5857	5935	6014
		SA patterns	6034	5927	6007
cf_fft_256_8	2-die	NSA patterns	21405	23270	23456
		SA patterns	21354	23326	23533
	3-die	NSA patterns	18495	17697	19927
		SA patterns	17993	17706	20174
	4-die	NSA patterns	20935	21031	21151
		SA patterns	21074	21223	21211

in Figure 2.6: r_D is decreases with λ. This implies that less mature the fabrication process (i.e., lower the yield), the more improvement in test quality that can be achieved using. Larger values of r_D indicate that higher test escapes (relative to what we get with TSV stress-aware patterns) will result if TSV stress is not considered for test generation. However, r_D also depends on the design. The TSV density in des_perf is relatively high compared to that in cf_rca_16, therefore the relative number of the devices affected by TSV stress is higher in des_perf. This results in a higher sensitivity of DPPM to accurate timing modeling.

The CPU time for the ATPG flow strongly depends on the design size. For a layout of des_perf, cf_rca_16, and cf_fft_256_8, the CPU time was one minute, 30 minutes, and four hours, respectively.

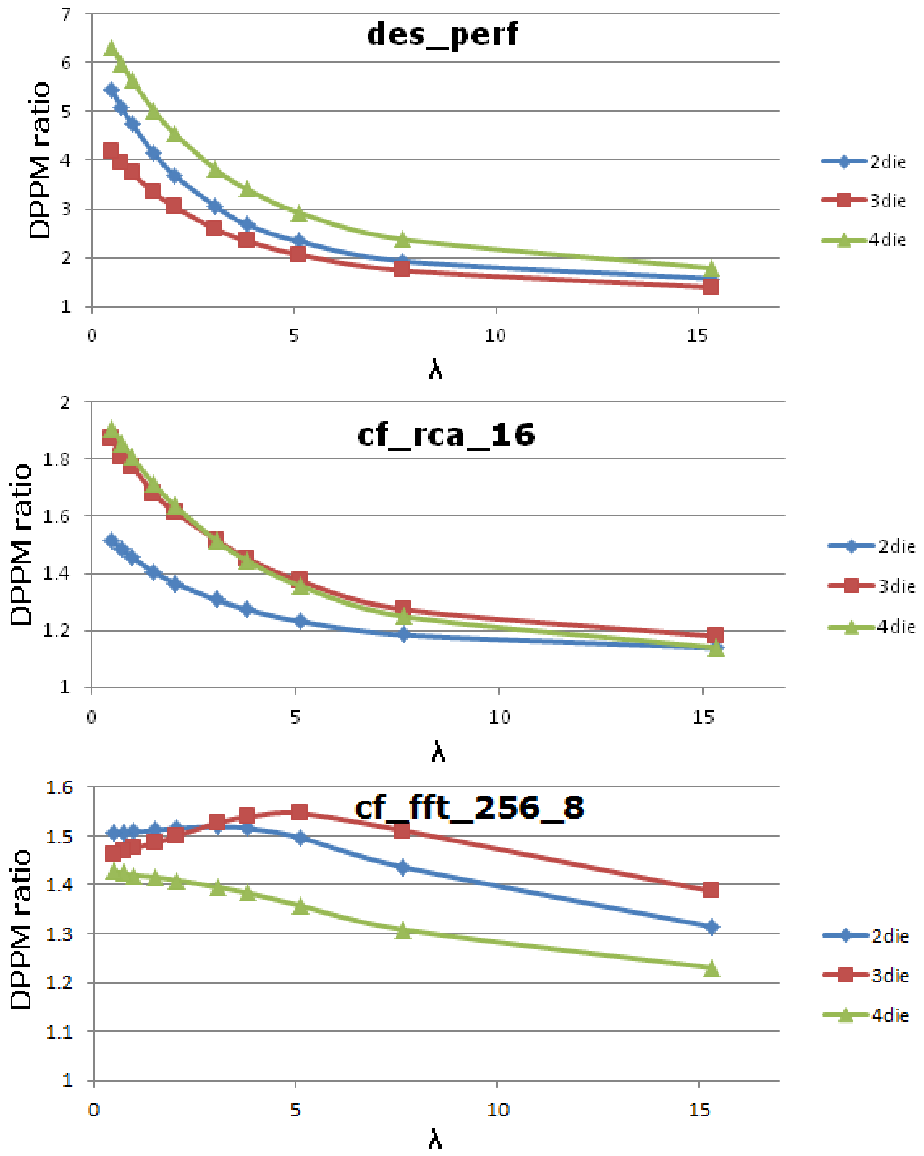


FIGURE 2.6: DPPM ratio as function of λ.

2.4 Conclusion

We evaluated the impact of TSV stress on the quality of SDD testing. We used an ATPG flow based on conventional ATPG tools to compare the test escape rate for the following cases: (1) anticipated test escape rate for TSV-stress unaware tests, (2) the actual test escape rate for TSV-stress unaware tests, and (3) the test escape rate for TSV-stress aware tests. Based on our results, we make the following conclusions.

- Neglecting TSV stress results in a significantly higher test escape rate compared to that obtained using a TSV-stress unaware ATPG flow.

- Using a TSV-stress aware ATPG flow will improves test quality and bring the escape rate back to the levels achieved using ATPG on circuits that are not affected by TSV stress. This is true for all KOZ sizes.

- Smaller KOZs are not an issue: even though the circuitry is stronger affected by TSV stress when using small KOZs, the ATPG flow with stress-aware models will still create high-quality tests. There is no noticeable impact on the pattern count.

- The degradation of the test quality if TSV stress is neglected is sensitive to the fabrication process quality. Therefore, the poorer the yield of the process, the more important is an accurate modeling of TSV stress in order to optimize the test quality.

3

Contactless Pre-Bond TSV Test and Diagnosis Using Ring Oscillators and Multiple Voltage Levels

One of the challenges associated with 3D test is new defects due to the TSV manufacturing process; such defects include voids and pinholes. Voids, as shown in Figure 3.1 are formed due to insufficient filling [46]. A pinhole is an oxide defect that creates a short between the TSV and the substrate [34]. Many of these defects arise prior to the bonding process. Therefore, they can be targeted during pre-bond testing, increasing the probability of getting a known good die (KGD) prior to bonding and therefore increasing the product yield. It has been widely acknowledged that the lack of KGD can be a serious yield limiter for 3D stacking [4, 50, 88].

However, pre-bond testing of TSVs is difficult because of test access limitations. Prior to wafer thinning, TSVs are buried in the silicon substrate and are only indirectly accessible through the circuitry connected to the TSVs. Even when the back side of the TSVs is exposed after wafer thinning, probing on those is challenging because of strict requirements on the probing equipment. Recent studies report success in mechanical probing at array pitches of 40 μm [2]; however, such probing solutions

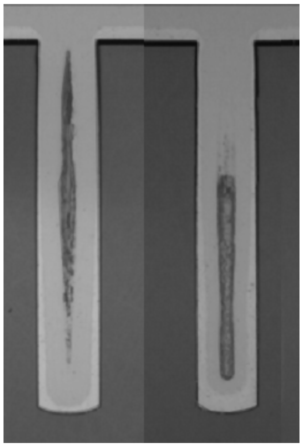


FIGURE 3.1: Voids in 10 μm x 60 μm TSVs [46].

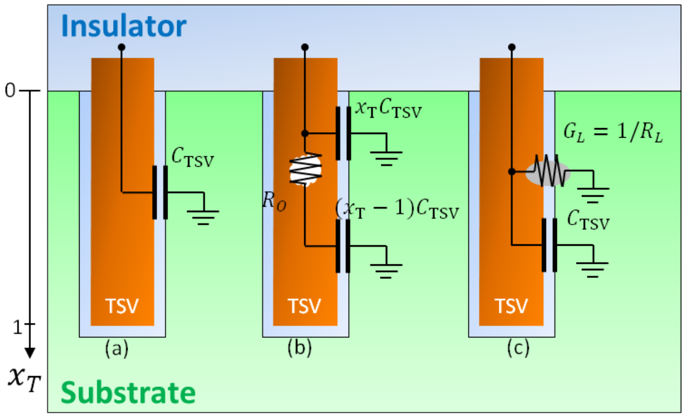


FIGURE 3.2: TSV models: (a) fault-free, (b) micro-void, (c) pinhole.

are still being researched and it remains to be seen how easily they can be used in practice. Therefore, probe-less solutions for pre-bond TSV test should also be investigated as an alternative to solutions that rely on probing.

In this dissertation, we propose a method for contactless pre-bond TSV test using ring oscillators and duty-cycle detectors. We can diagnose resistive-open and leakage faults by measuring the oscillation period and the duty cycle of signals generated by ring oscillators connected to TSVs. The proposed method offers the following benefits.

- We provide a solution to detect resistive-open and leakage faults in TSVs early during manufacturing testing, therefore improving the product yield, reducing long-life failures, and decreasing the need for field testing.

- We propose a regression model based on artificial neural networks (ANNs) to infer the fault size by measuring the oscillation period and the duty cycle of the signals generated by on-chip oscillators at different voltage levels.

- The regression model is able to diagnose weak leakage and weak resistive-open faults that can get aggravated over time, shortening the lifetime of the 3D IC.

- The proposed TSV test method is contactless, i.e., no TSV probing with external equipment is necessary.
- The test cost of the proposed method is low since we do not require TSV probing and both the DfT area overhead and the test time are negligible.

The remainder of this chapter is organized as follows. In Section 3.3, we describe our TSV fault models and the proposed circuitry for pre-bond TSV test using ring oscillators and duty-cycle detectors. Section 3.2 describes the regression model used for fault classification and fault size analysis. Section 3.3 presents simulation results. Finally, Section 3.4 concludes this chapter.

3.1 Pre-Bond TSV Test Method

This section describes (a) the electrical model of a fault-free and faulty TSV, (b) ring oscillators, (c) the duty-cycle detector, and (d) the overall DfT architecture required to implement the proposed test method.

3.1.1 TSV Fault Model

Many TSV defects, such as micro-voids or shorts to substrate, manifest themselves as changes in electrical parameters of the TSVs. In order to detect these defects, we create electrical fault models for them and create electrical tests that can measure variations in the models, i.e., the fault size. The fault size can be used as a metric for the effect of a fault on the performance of a TSV.

We target two types of TSV faults: resistive opens and leakage faults. Several TSV defects can be modeled by these faults. For instance, micro-voids increase the TSV resistance at the defect location and thus can be modeled as a resistive-open fault. Pinholes create a conduction path from a TSV to the substrate, resulting in a leakage fault.

Since a TSV is a passive structure resembling a wire, it can be modeled using a combination of lumped R, C, and L elements. The inductance of a TSV is relatively small and has no significant effect below a few GHz signal frequency [89]; therefore we limit our TSV models to RC circuits. In the following, we describe simple electrical TSV models for the three cases: fault-free, resistive-open fault, and leakage.

In the fault-free case, we model a TSV as an RC segment, where $R = R_{\mathrm{TSV}}$ is the TSV resistance and $C = C_{\mathrm{TSV}}$ is the capacitance between the TSV and the substrate. The values of the parameters R_{TSV} and C_{TSV} depend on the TSV technology; recent studies report the following values [89]: $R_{\mathrm{TSV}} = 0.1\ \Omega$, $C_{\mathrm{TSV}} = 59$ fF. Since R_{TSV} is significantly smaller than the output resistance of a typical driving gate, it can be neglected. The resulting model is a capacitor between the TSV and substrate, as shown in Figure 3.2(a). We verified this simplification by using HSPICE to simulate and compare charge curves of (1) multiple RC segments with combined resistance $C_{\mathrm{TSV}} = 0.1\ \Omega$ and combined capacitance $C_{\mathrm{TSV}} = 59$ fF, and (2) a single capacitor $C_{\mathrm{TSV}} = 59$ fF, where both loads are driven by a 4X buffer. The resulting curves show no measurable difference, which justifies the treatment of a fault-free TSV as a lumped capacitor.

Figure 3.2(b) shows a micro-void in the TSV at an arbitrarily chosen location x_{T} and the corresponding electrical model. This defect divides the TSV into two segments. The "top" segment ($[0, x_{\mathrm{T}}]$) is the part of the TSV until the location of the defect and can be approximated as a capacitor with the scaled-down capacitance $x_{\mathrm{T}} C_{\mathrm{TSV}}$. The "bottom" segment ($[x_{\mathrm{T}}, 1]$) includes the rest of the TSV capacitance $(1 - x_{\mathrm{T}})C\mathrm{TSV}$ and the increased resistance of the open R_{O} that depends on the size of the void. The parameter R_{O} can vary from a few Ω in case of a micro-void to infinity in the case of a full open in the TSV.

Figure 3.2(c) shows a leakage fault, e.g., due to a pinhole defect, which creates

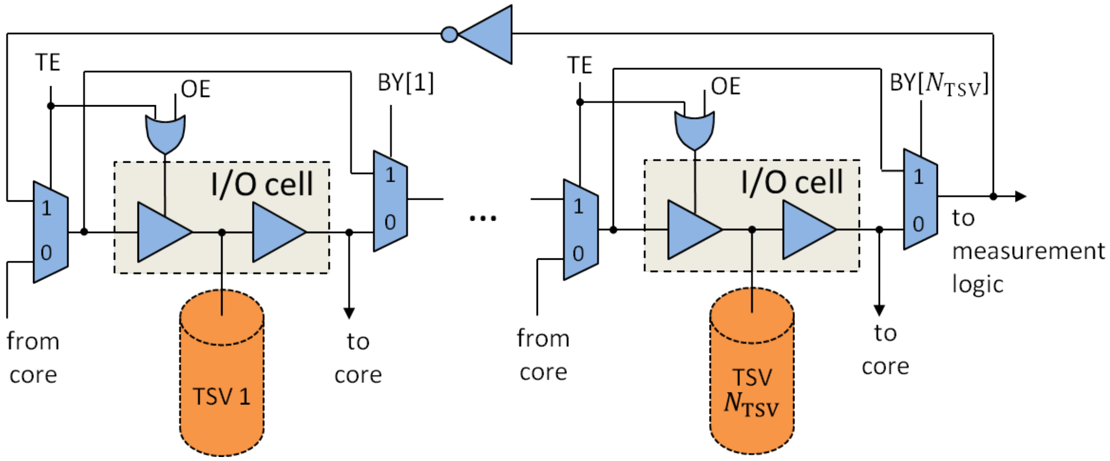


FIGURE 3.3: Ring oscillator with N TSVs.

a conduction path from the TSV to the substrate. The leakage is modeled by the resistor R_{L}, which is in parallel to the TSV capacitance. We use the conductance of this resistor $G_{\mathrm{L}} = 1/R_{\mathrm{L}}$, as the fault-free case can be simply expressed as $G_{\mathrm{L}} = 0$ and $R_{\mathrm{O}} = 0$. The value of G_{L} might increase over time, since leakage faults tend to deteriorate. Such faults are a serious concern for lifetime chip reliability.

3.1.2 Ring Oscillators with TSVs

In order to detect variations of R_{O} and G_{L}, we perform a parametric test using ROs. Deviations in these parameters due to faults lead to variations in the propagation delay of the net connected to the TSV. These variations can be measured by ring oscillators (ROs). An RO is a feedback loop containing an even number of inverters. Due to inversion of the signal in the loop, it keeps oscillating with a frequency that depends on the delay of the elements in the loop.

Figure 3.3 shows the configuration of an RO with TSVs as loads. The RO is comprised of multiple non-inverting I/O cells and an inverter. The oscillation period and the duty cycle of the generated signal are captured by on-chip DfT hardware based on binary counters and time-to-digital converters.

Each I/O segment includes a TSV and a bidirectional I/O cell connected to the front side of the TSV. We assume that the I/O cells are part of the functional circuitry, which is common in industrial designs. This aspect of our design contrasts with [52], where custom I/O cells are required even though these I/O cells are likely to be chosen based on functional requirements. TSVs can also be enhanced with tri-state drivers so that they can be driven appropriately.

The number of TSVs in a group (N_{TSV}) can be selected based on the desired oscillation frequency. In the extreme case, if $N_{\text{TSV}} = 1$, the RO contains only a couple of gates, which results in a relatively short oscillation delay (or high frequency). Such an oscillation frequency might be too high to drive the on-chip measurement logic. By appending extra segments, we increase the delay and thus reduce the oscillation frequency, relaxing the speed requirement on the measurement circuitry. In addition, all TSVs in the same group can share the same counter without extra decode logic, since all of them are in the oscillator loop. This reduces wiring and the amount of DfT logic.

The signal TE (test enable) controls the multiplexers selecting between the functional outputs coming from the internal logic and the oscillator loop. If TE = 0, the multiplexers 1 select the functional outputs coming from the internal logic, enabling a functional mode. If TE = 1, the circuit is configured in an oscillator loop and the functional driver output enable signal OE is overridden, activating a TSV test mode. The signals BY[1]. . . BY[N_{TSV}] (Bypass) control the multiplexers that include or exclude a TSV from the oscillator loop. OE (output enable) controls the tri-state drivers of the I/O cells. In functional mode, this signal is set by the internal logic. In test mode, OE is set to 1 to enable the drivers.

The oscillation period and the duty cycle of the generated signal are sensitive to resistive-open and leakage faults in different ways. First, we show by simulation how these faults affect the rise and fall times of an I/O cell with a TSV. We applied a rising and a falling voltage V_{in} to the input of an I/O cell and measured the output voltage ("to core") for (a) the fault-free case, (b) resistive-open fault at $x_{\text{T}} = 0.5$, and (c) leakage fault. Figure 3.4 shows the simulated waveforms of a rising edge of V_{in}. We can observe that the rise time decreases in the presence of a resistive-open defect and increases in the presence of a leakage fault. Figure 3.5 shows the simulated waveforms of a falling edge of V_{in}. We can observe that the fall time decreases in both cases.

The oscillation period is the sum of the rise times and the fall times of all gates comprising the ring oscillator. In the case of a leakage fault, the rise time of the TSV driver *increases* by Δt_{r}, as part of the current supplied by the TSV driver is leaking to the substrate instead of charging the TSV capacitance. However, the fall time of the TSV driver *decreases* by Δt_{f}, as the leakage contributes to discharging the TSV capacitance faster. For strong leakage faults, $\Delta t_{\text{r}} \gg \Delta t_{\text{f}}$; therefore, the overall oscillation period increases significantly and its deviation from a reference value can be measured. However, for weak leakage faults, $\Delta t_{\text{r}} \approx \Delta t_{\text{f}}$, hence the oscillation period does not deviate much from a reference value, making weak leakage faults difficult to detect by just measuring the oscillation period. This motivates us to measure the duty cycle of the oscillating signal and use these measurements as additional information in order to improve fault diagnosis.

Propagation delays in digital gates are affected by random-process variations [90]. Since the oscillation period is a function of gate propagation delays, it is sensitive to these variations. This has a negative impact on fault-diagnosis accuracy and can lead to aliasing. In order to reduce the noise effect introduced by random-process variations, we measure the oscillation period twice. T_{osc} is measured with the TSV under test included in the oscillation loop ($\text{BY}[i] = 0$), and $T_{\text{osc,b}}$ is measured with the TSV under test bypassed ($\text{BY}[i] = 1$). By subtracting $T_{\text{osc,b}}$ from T_{osc}, we can

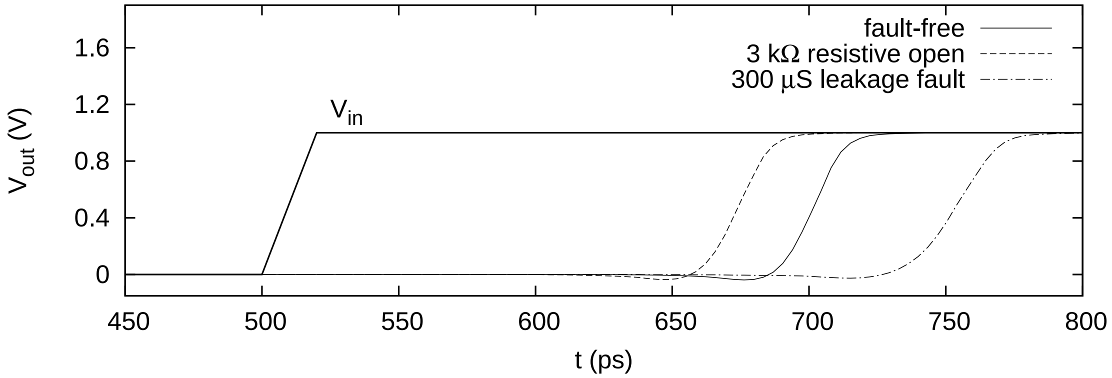


FIGURE 3.4: I/O-cell rise time for the fault-free case and 3 kΩ resistive-open and 300 μS leakage faults.

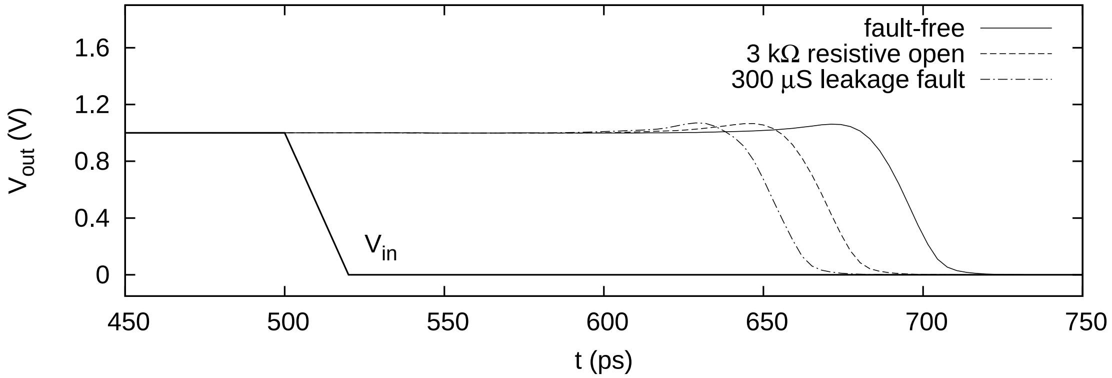


FIGURE 3.5: I/O-cell fall time for the fault-free case and 3 kΩ resistive-open and 300 μS leakage faults.

eliminate the common part, i.e., the propagation delay of the common circuitry part of both configurations.

3.1.3 Duty-Cycle Detectors

Our previously proposed method for pre-bond TSV test in [91] uses only one feature for fault detection and diagnosis—the oscillation period of the signal generated by on-chip ring oscillators. As shown above, this may not be sufficient to accurately diagnose weak leakage faults. In order to improve the diagnosis resolution, we propose to use duty-cycle detectors.

Even though the effects of weak leakage on rise and fall times of the TSV driver virtually cancel each other for the oscillation period, they strongly affect the duty cycle of the oscillating signal. In the following, we define the duty cycle D of an oscillating signal as the ratio of the on-time T_{on} of one cycle to the oscillation period T_{osc}:

$$D = \frac{T_{\text{on}}}{T_{\text{osc}}} = \frac{T_{\text{on}}}{T_{\text{on}} + T_{\text{off}}}, \tag{3.1}$$

where T_{osc} is the oscillation period, T_{on} is the on-time,defined as the time, for which the oscillating signal is above $V_{\text{dd}}/2$ during one cycle, and T_{off} is the off-time, defined as the time, for which the oscillating signal is below $V_{\text{dd}}/2$ during one cycle. The opposite effects of weak leakage faults on rise and fall times of the TSV driver create a difference between T_{on} and T_{off}, which can be effectively detected by measuring the duty cycle. This is the main motivation to use duty-cycle detectors in addition to frequency detectors, in order to be able to detect weak leakage faults more accurately.

Figure 3.6 shows an example of an on-chip duty-cycle detector that can be used for pre-bond TSV test. This is a time-to-digital converter based on the circuit proposed in [92]. The key idea of this detector is to integrate a constant current during (a) T_{on} and (b) T_{off} until a certain threshold voltage is reached and compare the integration times of (a) and (b). The following text describes the functionality of the duty-cycle detector in detail.

The oscillating signal CLK_OSC propagates through the XOR-gate, which inverts the signal if MODE = 1. The n-MOSFET Q1 opens during the on-time of CLK_OSC if MODE = 0, or during the off-time of CLK_OSC if MODE = 1. The gates of n-MOSFETs Q2 and Q3 are biased to $V_{\text{dd}}/2$, hence they are always “on”. When Q1 is closed, all of the current of Q3 is drawn from Q2. If Q1 is open, part of the current of Q3 is supplied by Q1. In reset mode (RST_n = 0), the current of Q1 is supplied by the p-MOSFET Q4. As Q4 is on, the input of the buffer is pulled to V_{dd}, and the capacitor C1 is discharged. When the circuit comes out of reset (RST_n = 1), Q1 starts drawing current through C1, but only when its gate voltage is high. C1 slowly charges and the voltage of the buffer's input starts decreasing. At some point, this voltage drops below a threshold, such that the output of the buffer drops from high to low. The flip-flop latches the value of the buffer at positive edges of a high-frequency reference clock CLK_REF_n. The output signal CNT_EN is therefore asserted after the circuit comes out of the reset and de-asserted when C1 has been charged enough to a threshold value to flip the output of the buffer to low.

Let us denote the time window during which CNT_EN is asserted as T_{en}. If C1 charges to the threshold value relatively slowly, i.e., it takes much longer than the clock period of the reference clock, T_{en} is proportional to the on-time of Q1, and hence proportional to T_{on} when MODE = 0, and proportional to T_{off} when MODE = 1. The time T_{en} is measured by using CNT_EN as the enable signal for a binary counter driven by the reference clock CLK_REF_n = ~CLK_REF. The count of this binary counter is read out twice. The first value, n_{on} is the count after a measurement with MODE = 0. The second value, n_{off} is the count after a measurement with MODE = 1. The duty cycle D of the input clock CLK_OSC can be simply calculated as

$$D = \frac{n_{\text{on}}}{n_{\text{on}} + n_{\text{off}}}. \tag{3.2}$$

Note that the measured D is independent of circuit parameters, hence careful calibration of the circuit is not required. We verified this property of the design through comprehensive HSPICE simulations.

A major difference of this duty-cycle detection method from that proposed in [92] is that we use the same circuitry for sequential measurement of n_{on} and n_{off} by inverting the input signal with an XOR gate. Therefore, the effect of random-process variations is virtually cancelled out. The method proposed in [92] uses two copies of the detector, one of which has an inverter in front of Q1 to invert the signal. This allows for measurement of n_{on} and n_{off} in parallel; however, local mismatches between

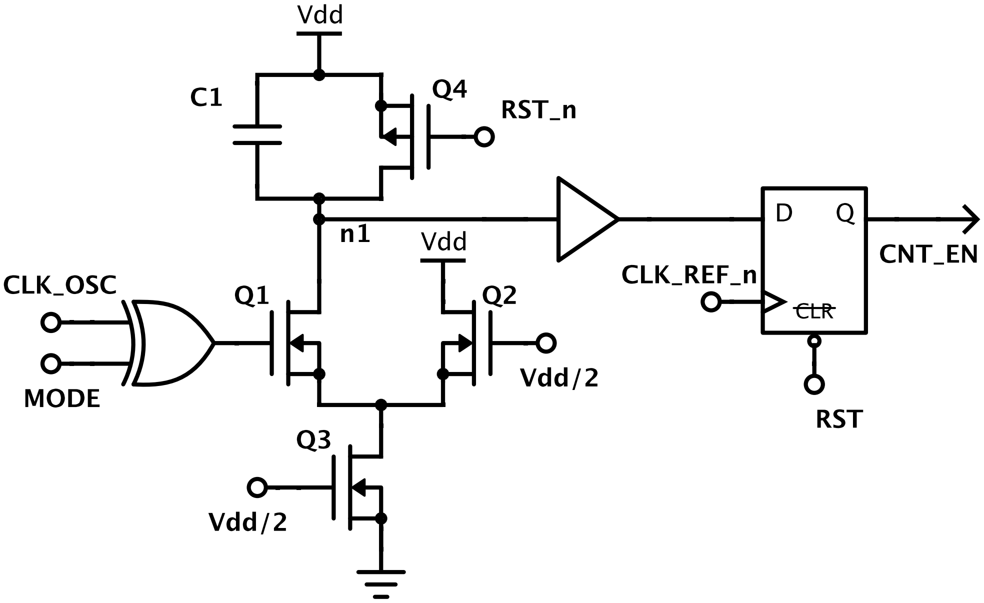


Figure 3.6: Schematic of the duty-cycle detector.

the two circuits decrease the accuracy of the detector. We verified through Monte-Carlo HSPICE simulations that the measurement error can exceed 15% in the case of two independent circuits due to random-process variations, which is unacceptable for the TSV diagnosis method we propose. Therefore, we use the same circuit to measure both n_{on} and n_{off} in a sequential manner, but with a much higher accuracy.

3.1.4 DfT Infrastructure for On-Chip Measurement

Next, we describe the DfT structures that are used to perform on-chip measurement of the RO oscillation period. Figure 3.7 presents an overview of the design. It consists of a control-logic block (TEST CTRL) that is connected to the IEEE 1149.1 (JTAG) TAP controller [21]. We assume that a TAP controller is already in place for other test and debug purposes, hence we can reuse it to control the TSV-test DfT circuitry from the test equipment. The only requirement on the TAP controller is two extra instructions to control the TSV test procedure, which is explained below. TEST CTRL is a logic block that is shared between multiple groups of ROs and that generates signals to control the ROs, the counters, the clocks, and the data flow.

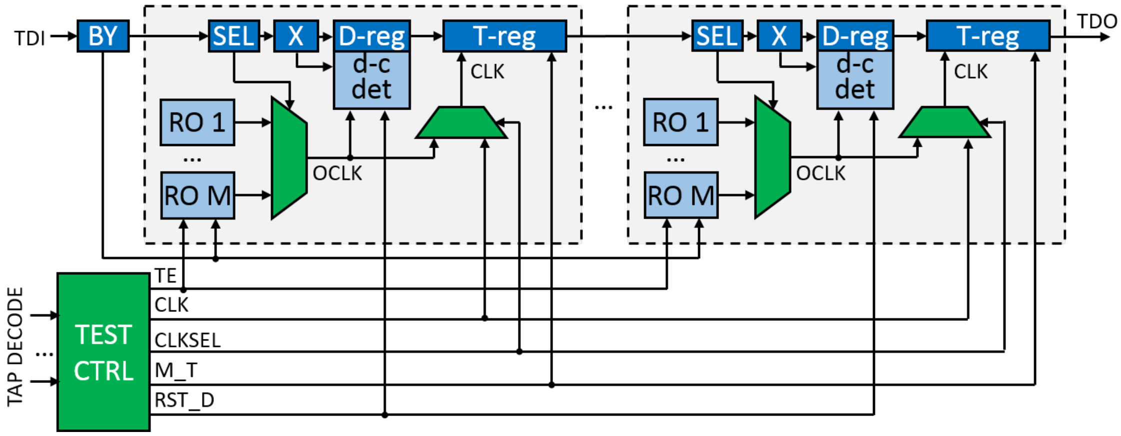


FIGURE 3.7: Pre-bond TSV test DfT infrastructure.

Each RO group contains M_{RO} ROs, a binary counter, an RO-select register (SEL), a register X to select the mode of the duty-cycle detector, and multiplexers to forward the clock signals to the counter. A global bypass-select register (BY) holds the values that are used to select what TSVs are included in the ring oscillator.

Table 3.1 shows the truth table that specifies the logic of TEST CTRL and defines the following modes of operation. In functional mode, TE = 0, which selects functional I/Os and disables the ROs. For enabling TSV test mode, two JTAG user instructions are added to the instruction set: INSTR1 and INSTR2. Once INSTR1 is loaded, i.e., the corresponding decode signal is asserted, TEST CTRL configures the binary counters into regular shift registers (M_T = 0) and concatenates BY, Xs, SELs, D-reg, and T-reg to a single scan chain from TDI to TDO. In this mode, we go to the ShiftDR state of the TAM FSM, which enables the shift clock (CLK = TCK) in order to shift in new bypass register values and initialize the counters D-reg and T-reg, as well as shift out the content of the counters from a previous measurement.

The second instruction, INSTR2, signals TEST CTRL to reconfigure the counters (M_T = 1). When we enter ShiftDR, the clock signals from the oscillators are fed to the counters. The objective of this mode is to count the pulses from the ROs over a defined time frame t, such that we can calculate the oscillation period as t/c, where

Table 3.1: Truth table for TEST CTRL output specification.

OP Mode	Input Signals			Output Signals				
	INSTR1	INSTR2	ShiftDR	TE	CLK	CLKSEL	M_T	RST_D
Functional	0	0	0	0	0	1	0	1
Shift conf.	1	0	0	1	0	1	0	1
Shifting	1	0	1	1	TCK	1	0	1
Setup	0	1	0	1	0	1	1	1
Measure	0	1	1	1	0	0	1	0

c is the pulse count. We control t by the number of cycles the TAM FSM stays in ShiftDR, i.e., the ShiftDR decode signal is asserted. Even though we have limited granularity to define the measurement time frame due to the slow test clock TCK, it is sufficient to know only the length of the time frame in order to achieve an accurate measurement of the RO frequency. The state of the counters is shifted out after the measurement by loading I1 again. In each iteration, we can select one RO in each group and one or multiple TSVs in this RO by loading BY and SEL registers.

Instruction INSTR2 is also used to measure the duty cycle. Once INSTR2 is loaded and the TAP FSM is in ShiftDR, the duty-cycle detector ("d-c det") will come out of reset (RST_D = 0) and perform the measurement. The register X drives the MODE signal of the detector and hence switches between measuring n_{on} and n_{off}. The measured count, n_{on} if $X = 0$, or n_{off} if $X = 1$, is stored into the register D-reg, from where it can be shifted out through TDO.

3.2 Regression Model Based on Artificial Neural Networks

In our previous approach presented in [91], we used a regression model for determining the likelihood f of the fault size R based on the measured ΔT, where ΔT is defined as the difference in the oscillation period when a TSV is bypassed.

This likelihood $f(R|\Delta T)$ is calculating by applying the Bayes' Theorem for continuous variables :

$$f(R|\Delta T) = \frac{f(\Delta T|R)f(R)}{\int f(\Delta T|R')f(R')\,\mathrm{d}R'}, \tag{3.3}$$

where $f(\Delta T|R)$ is the probability of measuring a certain ΔT given a fixed R, and $f(R)$ is the defect distribution, which depends on the manufacturing process and which can be obtained by analyzing manufacturing data. Note that R represents the fault size of both resistive-open faults R_{O} and leakage faults R_{L}.

The model based on Equation (3.3) has several disadvantages. First, it only uses ΔT as input variable. As described in Section III.C, ΔT is insensitive to very weak leakage faults. In addition to the oscillation period, we can use the duty cycle D as a second input variable, in order to increase the diagnosis resolution for the case of weak leakage faults. However, it is difficult to create an analytical model that uses both input types.

Second, the regression model based on Equation (3.3) uses the parameter ΔT measured at a fixed voltage level, and it is difficult to create a single model that that uses a set of inputs ΔT measured at different voltage levels.

Third, the model in [91] cannot effectively determine whether a fault is a leakage fault, resistive-open fault, or a combination of both.

In this section, we describe how to create a regression model for detecting faults in a TSV and inferring the fault size. The regression model is based on artificial neural networks (ANNs) [93]. ANNs are machine-learning algorithms inspired by biological neural networks. ANNs consist of layers of neurons and interconnections between these layers. Figure 3.8 shows an example of a three-layer ANN. Each neuron i in the input layer corresponds to an input variable α_i and forwards its value to the next layer. Neurons in other layers receive a weighted sum $\xi = \sum_i w_i \alpha_i$ of the neuron values from the previous layer, where w_i is the weight of input α_i. The sum is evaluated

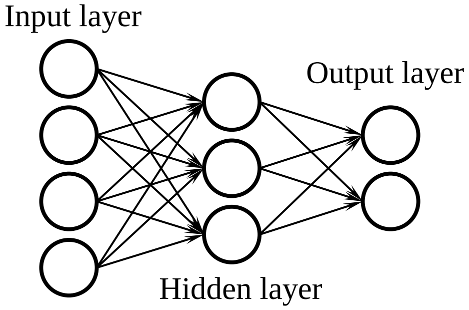


FIGURE 3.8: Example of a generic three-layer ANN.

using a given transfer function F_t, and the function value $F_t(\xi)$ is provided to the next layer as input. Examples of transfer functions include pure linear function $F_t(\xi) = \xi$ and sigmoid function $F_t(\xi) = \frac{1}{1+\exp(-\xi)}$. The values of the neurons in the output layer serve as outputs of the ANN. Between the mandatory input and output layers, there can be one or more hidden layers, which extend the ability of ANN to solve non-linear classification and fitting problems. The network can be trained using a sample set with known input and output value pairs. There exist a number of learning algorithms for training ANNs, for instance, the feedforward error-backpropagation learning algorithm. The key idea of this algorithm is to initialize the weight w of each neuron and iteratively adjust the weights to minimize an error function, which quantifies the deviation of the predicted output values to the actual ("target") output values. Once an ANN is trained, it can predict output values to previously unseen input combinations.

The motivation for using ANNs for fault diagnosis is the following. The previously proposed regression model in [91] is based on only one input variable, ΔT, which is measured at a fixed voltage. It is simple to find a one-dimensional function that fits the measured data points with little error. However, in order to increase the accuracy of fault diagnosis, we significantly increase the number of input variables, which makes it infeasible to manually find an empirical regression model. ANNs are considered universal tools to efficiently model such complex systems with a large number of input variables [93]. Therefore, we use ANNs to build a regression model that can accurately diagnose a TSV based on measured data.

Figure 3.9 shows the architecture of the regression model. Let K_{V} be the number of different voltage levels, at which the oscillation period and the duty cycle are measured. As inputs for the regression model, we provide K_{V} sets of four variables:

- T_{osc} is the oscillation period when the TSV under test is included in RO (BY = 0)
- $T_{\mathrm{osc,b}}$ is the oscillation period when the TSV under test is bypassed (BY = 1)
- D is the duty cycle when the TSV under test is included in RO (BY = 0)
- D_{b} is the duty cycle when the TSV under test is bypassed (BY = 1)

Note that we explicitly provide T_{osc} and $T_{\mathrm{osc,b}}$ as inputs to the regression model, as only providing their difference may result in loss of information that is important for determining the fault size.

All $4K_{\mathrm{V}}$ inputs feed into three ANNs. The first network, Class-net, is a classification network with three binary outputs that determine whether the TSV under test has a leakage fault (class_leak), resistive-open fault (class_open), or both (class_dual). This network is trained on a large set of simulation data that covers a range of single faults, as well as pairs of faults.

The other two networks, G_{L}-net and R_{O}-net, are fitting networks that output the value of the conductance of a leakage fault and the value of the resistive-open fault, respectively. G_{L}-net is trained on a training set obtained by simulations, in which R_{O} is set to zero and only G_{L} takes a range of inputs. This ensures that G_{L}-net is "specialized" for the case of a leakage fault. Similarly, R_{O}-net is trained on a data set with a fixed $G_{\mathrm{L}} = 0$ and variable R_{O}, in order to specialize this network for the case of resistive-open faults.

When Class-net outputs class_leak = 1, we use the output value of G_{L}-net as a prediction of the leakage-fault size and ignore the value output by R_{O}-net. When Class-net outputs class_open = 1, we use the output value of R_{O}-net as a prediction of the resistive-open fault size and ignore the value output by G_{L}-net.

In the case of class_dual = 1, we can determine that the TSV under test has both leakage and resistive-open faults. However, due to masking effects, it is difficult to predict the size of the faults, therefore, we ignore the values G_{L} and R_{O}. Even though the model will not provide a reliable analysis of the fault size in this case, it will still classify the TSV as being faulty.

We next present a hypothetical example for a TSV with a short to substrate that is equivalent to 100 μS leakage conductance. Suppose the ANNs are created for $K_{\mathrm{V}} = 3$, i.e., the ANNs take three sets of $\{T_{\mathrm{osc}}, T_{\mathrm{osc,b}}, D, D_{\mathrm{b}}\}$ as input variables, for instance, at $V_{\mathrm{dd}} = 0.85$ V, $V_{\mathrm{dd}} = 1.0$ V, and $V_{\mathrm{dd}} = 1.25$ V. The ANNs have been already trained using existing sample data obtained through simulation or actual manufacturing data. Suppose we obtain the following measurement data for a particular TSV under test:

- $\{3.46$ ns, 2.76 ns, 0.49, $0.46\}$ at $V_{\mathrm{dd}} = 0.85$ V
- $\{1.75$ ns, 1.41 ns, 0.48, $0.47\}$ at $V_{\mathrm{dd}} = 1.0$ V
- $\{1.32$ ns, 1.06 ns, 0.48, $0.47\}$ at $V_{\mathrm{dd}} = 1.25$ V

First, we input this data into Class-net and it outputs $\{1, 0, 0\}$, which indicates that the fault type of the TSV is leakage. Then, we input the measurement data into G-net and obtain 105. Therefore, we predict that the TSV has a leakage fault of size $G_{\mathrm{L}} = 105$ μS, which is close to the actual value of the leakage conductance.

Note that we do not explicitly define a class for a fault-free TSV. In practice, every TSV has a small leakage to ground and a small resistance of the filling material, i.e., both G_{L} and R_{O} are non-zero even for TSVs that are considered fault-free. Therefore,

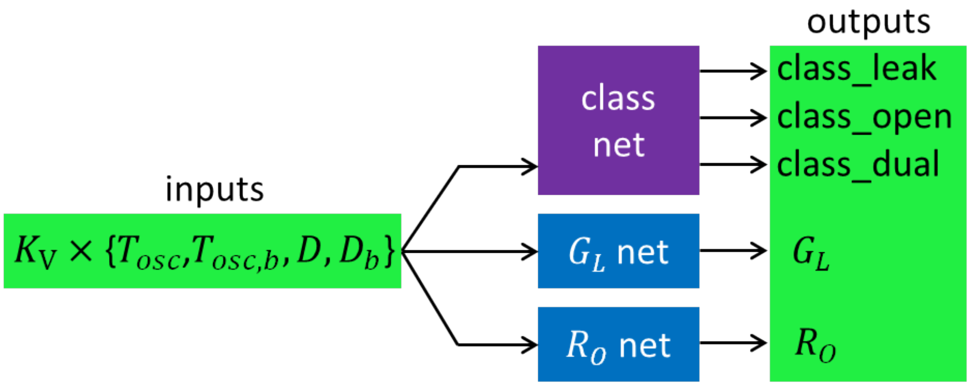


FIGURE 3.9: Regression model based on ANNs.

the class of fault-free TSVs is a special case of class_leak or class_open, in which G_L and R_O are below certain thresholds, which the user defines to distinguish fault-free TSVs from the faulty ones.

3.3 Simulation Results

We verified our approach through HSPICE simulations, for which we used the TSV models described in Chapter and the 45 nm Predictive Technology Model (PTM) low-power CMOS models [94]. As TSV drivers, we use BUF_X4 buffers from the Nangate 45nm Open Cell Library [95]. For other gates, X1 versions are used. These gate strengths are representative, as reported in recent literature [89].

In the following, we demonstrate the effect of resistive-open and leakage faults on the oscillation period using simulation. We show that resistive-open faults lead to a decreased oscillation period. Leakage faults, in contrast, result in an increased oscillation period. This is the reason for using the oscillation period as a distinguishing feature for fault diagnosis.

3.3.1 Resistive-Open Faults

First, we simulate a resistive-open fault in one of the TSVs at the location $x = 0.5$ as depicted in Figure 3.2(b). We create an HSPICE circuit model of the ring oscillator shown in Figure 3.3 with $N_{\text{TSV}} = 5$ TSVs and sweep R_O from 0(no fault) to 3 kΩ (strong resistive open) at the nominal voltage $V_{DD} = 1.1$ V. With this model, we perform transient analysis and record the oscillation period of the ring oscillator T_{osc}. In the first run, TSV1 is enabled (BY[1] = 0) and all other TSVs are bypassed (BY[$2 \ldots N_{\text{TSV}}$] = 1). In the second run, we disable all TSVs. Subsequently, we subtract the oscillation period of the second run T_2 from that of the first run T_1 for each value of R_O:

$$\Delta T = T_1 - T_2$$

During actual test, the values T_1 and T_2 will be measured by the on-chip DfT and the results sent to the test equipment and post-processed there.

The above subtraction step removes the propagation delay of the path through I/O cells $2 \ldots N_{\text{TSV}}$ and the inverter. The remainder is virtually the propagation delay due to the I/O cell and TSV1, which is under test. This approach greatly reduces the effect of delay variations in gates and interconnects due to random process variations.

The results of this simulation are shown in Figure 3.10. As expected, an increase in the resistance R_O leads to a reduction of the oscillation period. This indicates that we can detect resistive opens of a sufficient size by measuring the oscillation period. For instance, ΔT of a resistive defect of size 1 kΩ at $x_{\text{T}} = 0.5$ is reduced by 10% compared to the fault-free case that can be identified.

The location x of the defect plays a critical role for fault detection. The more the fault is moved to the top of the TSV, where it is connected to the driver, the more easily we can detect it. A void at the bottom of the TSV is not detectable. However, this is problem for all pre-bond TSV test methods mentioned in Section 1.5.

In a realistic (3D) IC, propagation delays of gates vary significantly because of random process variations. This can potentially have a detrimental effect on the resolution of our test method, since we rely on relatively constant delays in our DfT

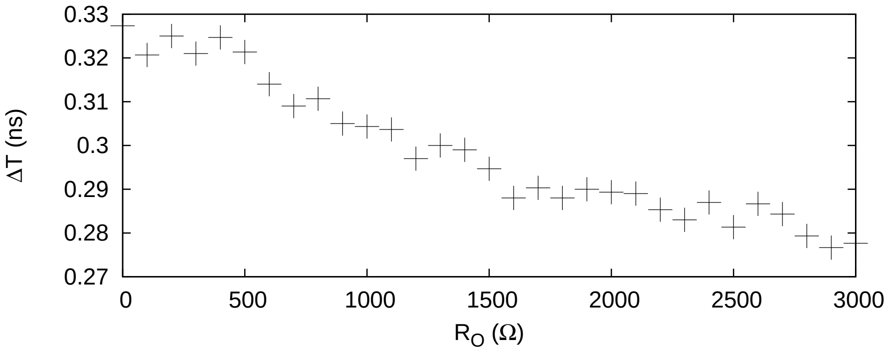


FIGURE 3.10: ΔT as a function of R_O at location $x_\mathrm{T} = 0.5$ and at 1.1 V supply voltage.

circuitry to be able to detect variations of the delay on the net connected to the TSV. Therefore, we need to verify whether our method is robust to random process variations, which leads us to the next set of simulations using HSPICE.

To see the effect of process variations and the effect of applying different supply voltages, we run a number of Monte Carlo (MC) simulations using the model described above extended by the following process-variation model: $3\sigma_{V_{th}} = 50$ mV, $3\sigma_{L_{eff}} = 10\%$, where $\sigma_{V_{th}}$ is the variation in threshold voltage and $\sigma_{L_{eff}}$ is the variation in gate length. These data are consistent with the those reported by industry for recent technology nodes [90].

Figure 3.11 shows the results of MC simulation for a fault-free TSV and a TSV with a resistive open defect of size 1 kΩ. We varied the supply voltage and analyzed the spread in the fault-free and faulty cases. We observe that at lower supply voltage levels, a part of data points from both fault-free and faulty cases overlap and thus become indistinguishable. If we increase the supply voltage, this overlap reduces to a minimum until we see no aliasing. From this, we conclude the following.

- Even in presence of process variations, our TSV test method allows for detection of resistive-open defects that have a sufficiently large size and are located in the upper part of the TSV. The test resolution depends on the process variation:

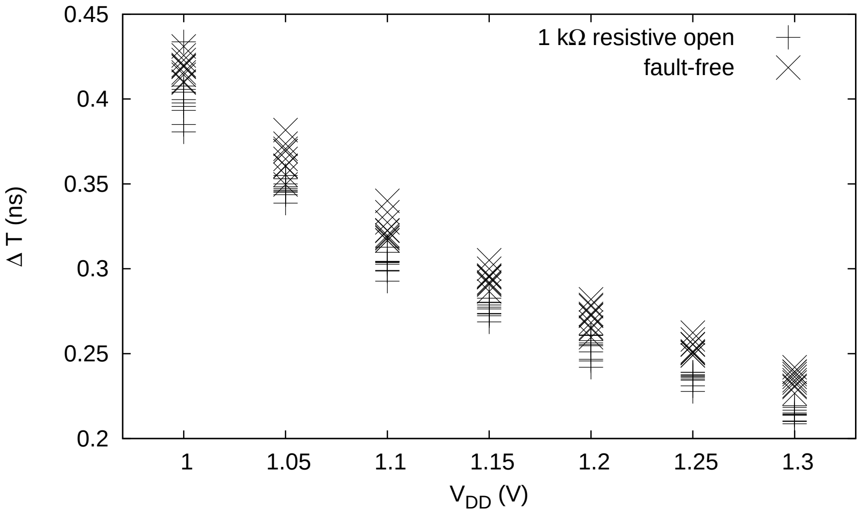


FIGURE 3.11: ΔT as a function of supply voltage in (a) fault-free case and (b) in case of 1 kΩ resistive open at $x_{\mathrm{T}} = 0.5$.

the more variation, the harder it becomes to distinguish small resistive opens from the fault-free case.

- Higher supply voltage results in a better resolution: aliasing is reduced, allowing for detection of smaller resistive-open faults.

3.3.2 Leakage Faults

Leakage faults exhibit a different behavior than resistive open faults. To show this, we used the same simulation approach as described above ($N_{\mathrm{TSV}} = 5$). Figure 3.12 shows the dependence of ΔT on the leakage R_L for different voltage levels. First, we observe that leakage faults increase the oscillation period, which makes them distinguishable from the fault-free case as well as resistive open faults. This is also consistent with our expectation. Second, strong leakage faults below a certain threshold, $R_L \approx 1$ kΩ, prevent the circuit from oscillating. In other words, the TSV exhibits stuck-at-0 behavior. This threshold depends on the supply voltage: it drops as we increase the voltage. The third observation is that in the regions slightly above each threshold, ΔT

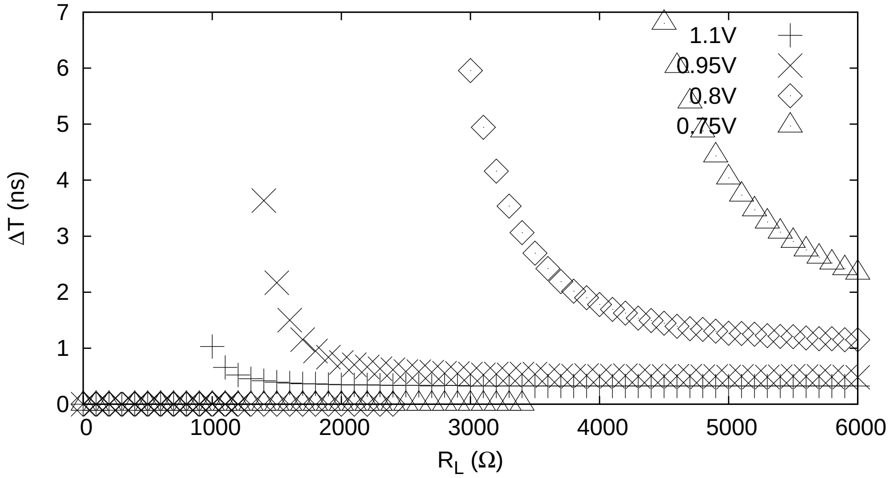


FIGURE 3.12: ΔT as a function of R_L at different voltage levels.

is extremely sensitive to small variation in leakage. This indicates that we can easily detect leakage of a wide range if we test at different voltage levels. Strong leakage (R_L low) will show up at higher V_{DD}. Weak leakage will become detectable at lower V_{DD}. This observation is consistent with the results of prior work on very-low-voltage tests for bridging faults [56, 57].

Next, we perform MC simulations to demonstrate the effect of random-process variations on the oscillation period in the presence of a leakage fault. Figure 3.13 shows the results of MC simulations for a 3 kΩ leakage fault (equivalent to $G_{\mathrm{L}} = 333.3$ μS) and the fault-free case at different voltage levels. We see that in the sensitive region right above the threshold (≈ 0.75 V), the data points for the two cases do not overlap. As we increase V_{DD}, the gap between them becomes smaller such that we cannot distinguish between the faulty and fault-free cases.

The simulation results confirm that resistive-open and leakage faults significantly change the oscillation period T_{osc}. Since resistive-open faults reduce T_{osc} and leakage faults increase T_{osc}, we can use T_{osc} as a distinguishing feature for fault diagnosis. The results also show that TSVs should be tested at multiple voltage levels in order

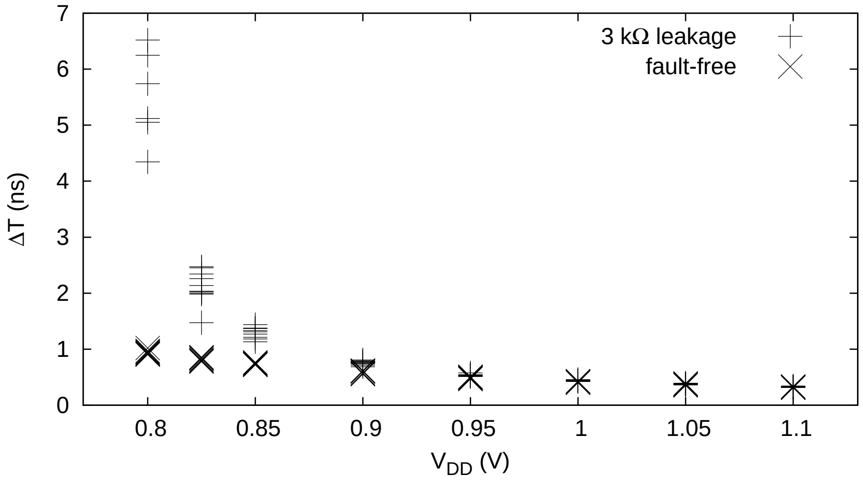


FIGURE 3.13: ΔT as a function of supply voltage in (a) fault-free case, and (b) in case of 3 kΩ (333.3 μS) leakage fault.

to increase fault detectability.

In the following, we present simulation results demonstrating the accuracy of the duty-cycle detector, as well as results evaluating the regression model.

3.3.3 Duty-Cycle Detector

We have verified the functionality and the accuracy of the proposed duty-cycle detector in the presence of random-process variations by HSPICE simulations. First, we created an HSPICE model of the detector using ditital gates from the the Nangate 45nm Open Cell Library [95] and MOSFETs with the following parameters.

- $L_{Q1} = L_{Q2} = L_{Q3} = L_{Q4} = 50$ nm, where L_{Qx} is the gate length of the MOSFET Qx
- $W_{Q1} = 0.5$ μm, $W_{Q1} = 2$ μm, and $W_{Q3} = W_{Q4} = 0.2$ μm, where W_{Qx} is the gate width of the MOSFET Qx
- $C1 = 2$ pF
- $f_{\text{CLK\_REF}} = 1$ GHz

The clock input of the duty-cycle detector was driven by a ring oscillator with a fault-inserted TSV. The fault size covered the range $G_{\mathrm{L}} = 0 \ldots 450\ \mu\mathrm{S}$ in order to produce a clock signal with duty cycle $D = 0.25 \ldots 0.5$. The actual duty cycle of the oscillating signal was determined through probing the signal and measuring T_{on} and T_{off}, whereas the measured duty cycle was determined by reading the binary counter values. We ran a number of Monte-Carlo simulations and calculated the maximum absolute error between the measured and the actual duty cycle. This error was 0.7% at $V_{\mathrm{dd}} = 1.1$ V, and 2.0% at $V_{\mathrm{dd}} = 0.85$ V, which shows high robustness of the detector against random-process variations.

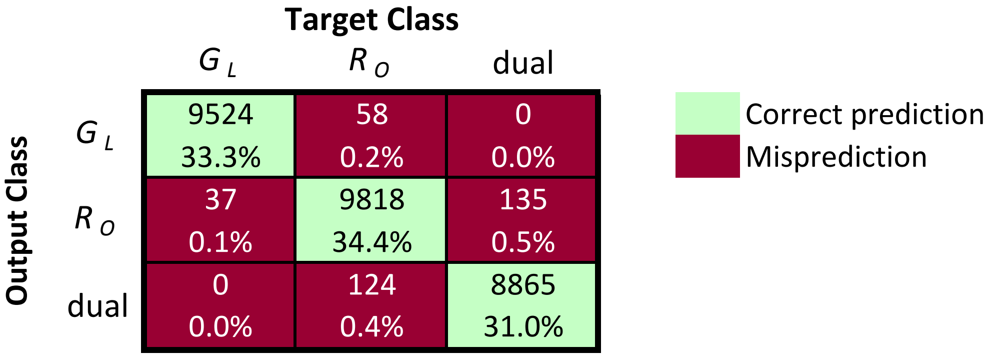
3.3.4 Regression Model for Inferring Fault Size

In order to verify the proposed regression model based on ANNs, we have generated two independent large sets of training and test data with the sample size over $10,000$. Each Monte-Carlo instance was simulated at $K_{\mathrm{T}} = 8$ different voltage levels ($0.85 \ldots 1.2$ V with 50 mV steps) in bypass and non-bypass mode, and the corresponding measurements T_{osc}, $T_{\mathrm{osc,b}}$, D, and D_{b} were recorded. The leakage fault size was varied from $G_{\mathrm{L}} = 0$ (fault-free) to 450 μS (very strong leakage). The resistive-open fault size was varied from $R_{\mathrm{O}} = 0$ (fault-free) to 5000 Ω (very strong resistive open).

First, we created Class-net in MATLAB and trained it using the training-data set. As the performance evaluation metric, we used the mean squared error (MSE), which is defined as follows:

$$\mathrm{MSE} = \frac{1}{N_{\mathrm{sam}}} \sum_{i=1}^{N_{\mathrm{sam}}} (y_{p,i} - y_{t,i})^2, \qquad (3.4)$$

where N_{sam} is the number of samples, $y_{p,i}$ and $y_{t,i}$ are the predicted and the target values of sample i, respectively. The training function for the network was selected following the guidelines in [96].



Output Class \ Target Class	G_L	R_O	dual
G_L	9524 33.3%	58 0.2%	0 0.0%
R_O	37 0.1%	9818 34.4%	135 0.5%
dual	0 0.0%	124 0.4%	8865 31.0%

🟩 Correct prediction
🟥 Misprediction

FIGURE 3.14: Confusion matrix for class net.

We considered different settings for the ANN architecture and achieved the best performance with the following settings:

- One output layer with three neurons (equals the number of output signals);
- One hidden layer with ten neurons;
- Scaled Conjugate Gradient (trainscg) as training method.

Then, Class-net was evaluated using the test-data set. Figure 3.14 shows the confusion matrix from that evaluation. The entry (i,j) shows the number of samples of type j classified as i by the network, and their percentage. For instance, the entry (2,1) shows 37 (or 0.1%), which means that 0.1% of all test points were classified as R_{O}, although the actual fault is G_{L}. According to this matrix, the majority of the samples were classified correctly (green cells), and only a small percentage of the samples were mispredicted (red cells), which shows the high accuracy of the classification network.

The network G_{L}-net was trained using a subset of the generated samples with $R_{\mathrm{O}} = 0$ (leakage faults only). We used the following settings:

- One output layer with one neurons (equals the number of output signals);
- One hidden layer with 10-20 neurons (depending on the seed);
- Levenberg-Marquardt (trainlm) as training method.

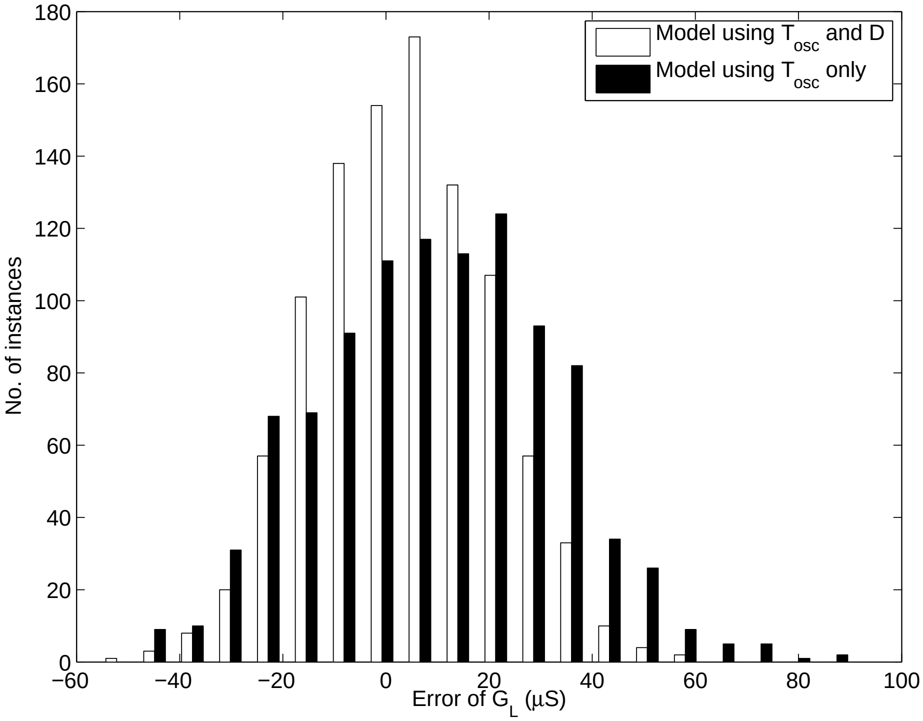


FIGURE 3.15: Error histograms of G_{L}-net and G_{L}-net_r at $G_{\mathrm{L}} = 100\ \mu$S.

In order to show the improvement in accuracy that we achieve by adding the duty cycle to the regression model as extra information, we performed the following simulation. We created a "reduced" regression model G_{L}-net_r with the same settings as G_{L}-net, with the only difference that it uses T_{osc} and $T_{\mathrm{osc,b}}$ pairs as input data, with $2K_{\mathrm{T}} = 16$ inputs in total. Figure 3.15 shows the error histograms of G_{L}-net and G_{L}-net_r evaluated on 1000 samples with $G_{\mathrm{L}} = 100\ \mu$S and $R_{\mathrm{O}} = 0$. It can be observed that G_{L}-net_r makes significantly better predictions, as the amount of samples predicted with an error of less than 20 μS is much larger than that predicted by the reduced model.

Figure 3.16 shows a comparison between G_{L}-net and G_{L}-net_r in terms of MSE for a range of leakage faults. As expected, the model using duty cycle as input variable provides significantly better predictions for weak leakage faults with $G_{\mathrm{L}} \leqslant 150\ \mu$S, which will help to reduce test escapes as well as over-testing.

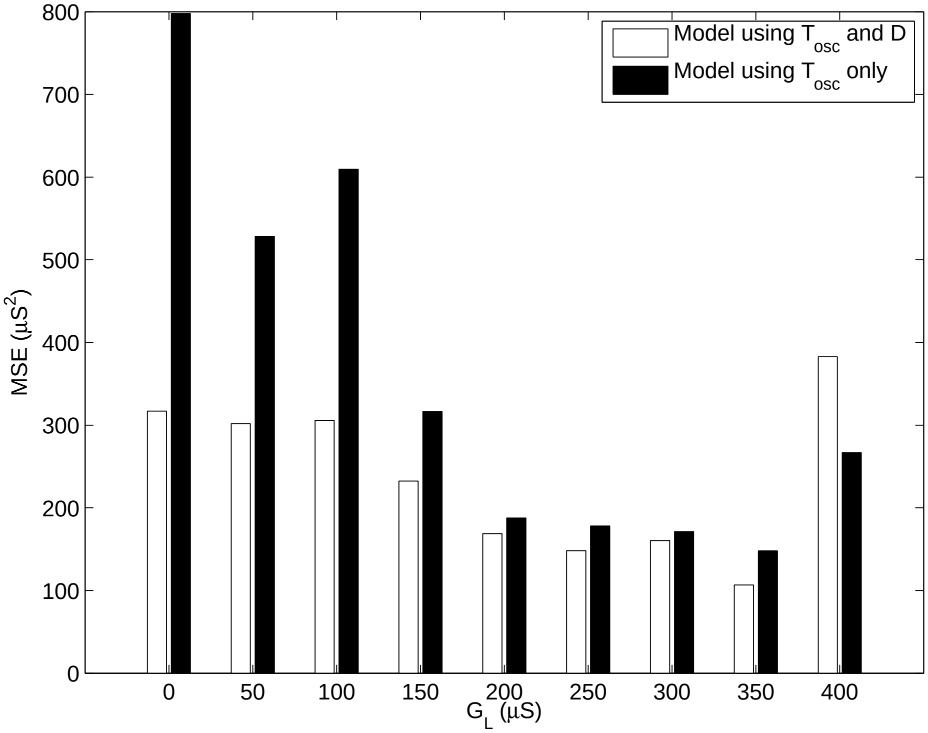


FIGURE 3.16: MSE of G_{L}-net and G_{L}-net_r for different values of G_{L}.

The network R_{O}-net was generated using a similar network architecture as G_{L}-net. It was trained using a subset of the generated samples with $G_{\mathrm{L}} = 0$ (resistive-open faults only). Since resistive-open faults affect the rise and fall times of the TSV drivers in the same way, the duty cycle is unsensitive to this type of faults. We demonstrate it by performing a similar simulation as described above, in which a "reduced" network R_{O}-net_r is generated using only T_{osc} and $T_{\mathrm{osc,b}}$ pairs as input data. As a result of the comparison, we observe that the performance of both networks is approximately the same. This indicates that the duty cycle is redundant information for determining R_{L}. Nevertheless, the duty cycle is essential for classification of faults and detecting weak leakage faults.

We also perform a simulation to show the impact on the diagnosis accuracy if T_{osc} and D are only measured at one voltage level. We created regression models that only use $\{T_{\mathrm{osc}}, T_{\mathrm{osc,b}}, D, D_{\mathrm{b}}\}$ at a specific voltage level instead of multiple levels and trained

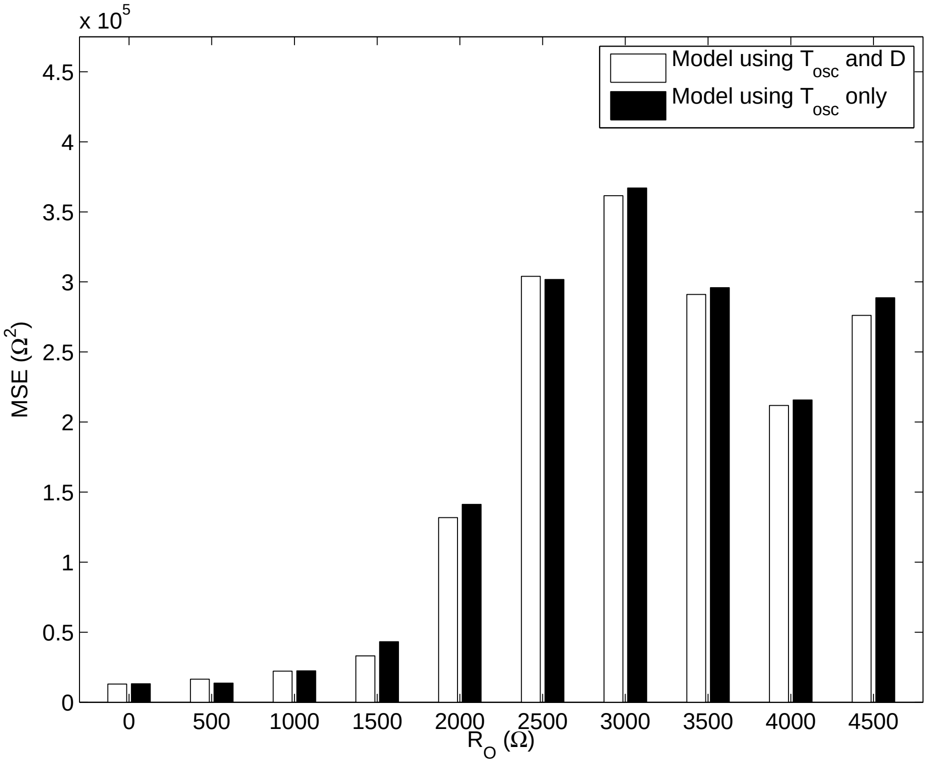


FIGURE 3.17: MSE of R_{O}-net and R_{O}-net_r for different values of R_{O}.

it using similar settings as for the full models G_{L}-net and R_{O}-net. Figure 3.18 shows a comparison of G_{L}-net to the model that uses the four input variables measured at $V_{\mathrm{dd}} = 0.85\mathrm{V}$, as leakage faults are better detectable at voltage levels below nominal V_{dd}. As we can observe, a reduction of the test to one voltage level would result in a significant loss of diagnosis accuracy.

Similarly, we created a reduced version of R_{O}-net that only takes the four input variables $\{T_{\mathrm{osc}},\ T_{\mathrm{osc,b}},\ D,\ D_{\mathrm{b}}\}$ at $V_{\mathrm{dd}} = 1.25\mathrm{V}$, since resistive-open faults are better detectable at higher voltage levels. Figure 3.19 shows a comparison between the reduced model and R_{O}-net. We can observe that the diagnosis accuracy drops significantly, especially for weak resistive-open faults $R_{\mathrm{O}} \leqslant 2500\ \Omega$. These results support our premise that a regression model using multiple voltage levels is more effective than that using a single voltage level.

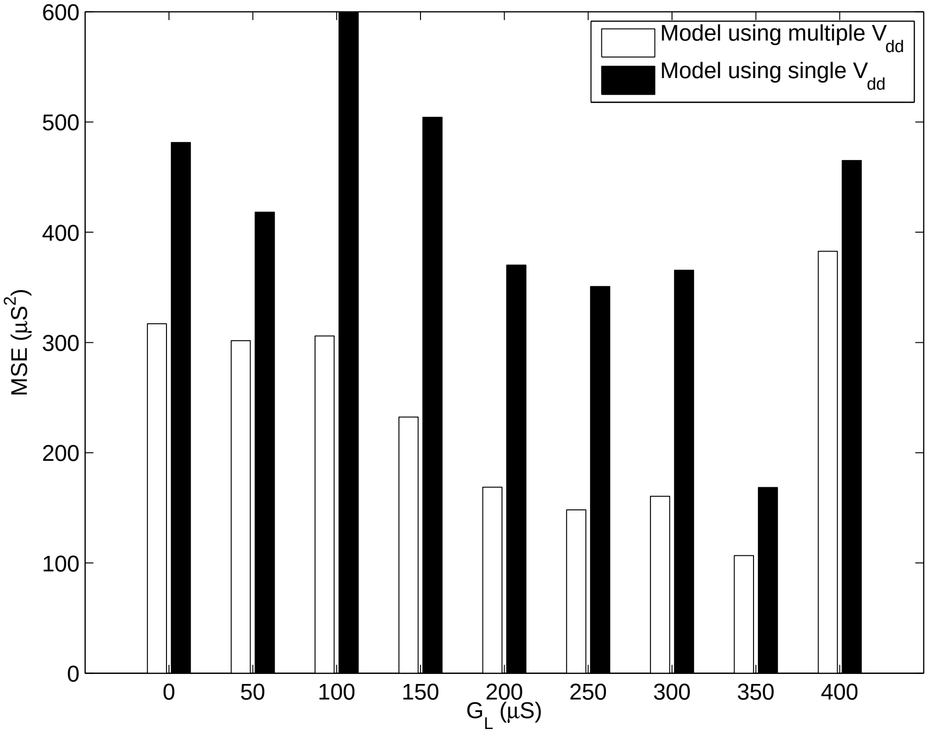


FIGURE 3.18: MSE of G_{L}-net using multiple vs. single V_{dd}.

3.3.5 DfT Area Cost and Test Time Estimations

We created Verilog RTL models of the DfT circuitry for TSV test shown in Figure 3.7, synthesized it and mapped it to the Nangate 45nm Open Cell Library [95] using Cadence RTL Compiler. In this example, we used four RO groups containing four TSVs each, which required two-bit wide SEL registers and a four-bit wide BY register. Since the duty-cycle detector is an analog circuitry with digital inputs and outputs, it was modeled as a black box for the purpose of digital simulation. To allow for precise measurement of the RO frequency and duty cycle, we used a twelve-bit binary counter and a ten-bit binary counter, respectively. The combined standard-cell area of TEST CTRL and BY is 35 $\mu\mathrm{m}^2$; the standard-cell area of each RO group is 208 $\mu\mathrm{m}^2$, excluding duty-cycle detectors and functional I/O drivers and receivers. The area of each duty-cycle detector is dominated by the integrating capacitance C1, which is implemented as PMOS capacitance with an approximate area of 60 $\mu\mathrm{m}^2$. Therefore,

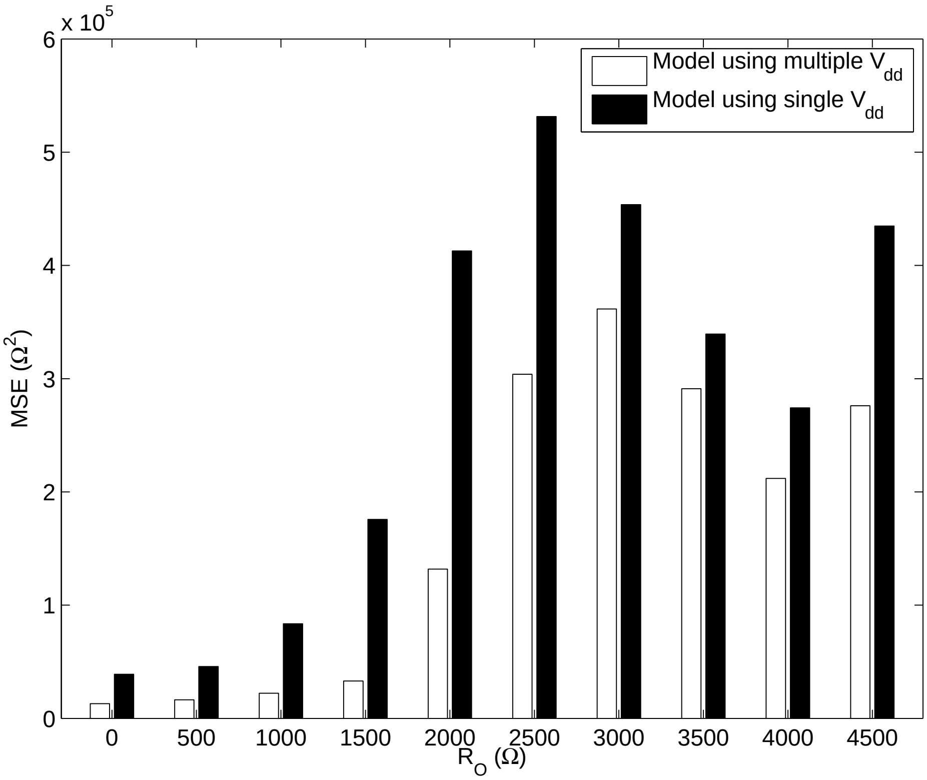


FIGURE 3.19: MSE of R_{O}-net using multiple vs. single V_{dd}.

if the functional design incorporates 1000 TSVs (64 TSV groups of 4x4 TSVs), to total DfT area for TSV test sums to 0.02 mm^2, which is negligible.

We estimated the time needed to test a die with 1000 TSVs that are placed in groups of 16 TSVs as described in the example above:

- Less than 15 TCK cycles to load a JTAG instruction (dependent on the length of the IR)
- Less than 1670 TCK cycles to load/unload the counters, BY, X, and SEL
- 7 TCK cycles to transition to/from ShiftDR for the measurement plus the actual measurement time 1...5 μs for RO frequency and 1 μs for duty cycle

Each TSV needs to be tested twice to measure T_{osc} and $T_{\mathrm{osc,b}}$. In order to measure D and D_{b}, each TSV needs to be tested four times (twice with X = 0 and twice with X = 1). Two of the measurements of D and D_b can be overlapped with the measurements T_osc and $T_\mathrm{osc,b}$. However, the remaining two measurements of D and D_b need to be performed separately. Assuming a TCK with the frequency f_tck, the time of one test cycle including testing all 16 TSVs in each group at eight different voltage levels, can be estimated as

$$8 \times 2 \times (\frac{1}{f_\mathrm{tck}} \times (15 + 1670 + 7) + 1 \ldots 5\ \mu\mathrm{s}) +$$
$$8 \times 2 \times (\frac{1}{f_\mathrm{tck}} \times (15 + 1670 + 7) + 1\ \mu\mathrm{s}).$$

Testing this die for resistive-open and leakage defects in TSVs would require approximately 1.5 ms test time at 20 MHz TCK or 0.65 ms at 50 MHz TCK, which is a relatively low test-time overhead.

3.4 Conclusions

We have introduced a method for pre-bond TSV test using ring oscillators and duty-cycle detectors. We have shown that we can detect leakage and resistive-open faults early during manufacturing testing, thereby increasing the product yield. We have provided a regression model using artificial neural networks for diagnosis of the defect based on the obtained measurement data. The new regression model significantly improves the accuracy of the fault diagnosis for weak leakage faults that are important to detect, as they might become critical during “aging” of the product. In addition, the proposed regression model can effectively classify the type of TSV fault, even if both leakage and resistive-open faults are present.

The proposed DfT structures have a minimal area overhead and use an existing 1149.1 test interface, hence extra test pins are not required. The total TSV test time for an example design with 1000 TSVs is in the range of 1 ms, which is negligible.

The proposed pre-bond TSV method does not require TSV probing which is associated with increased cost for the probe equipment and possible damage to the TSV. Most of the DfT circuitry uses standard digital cells, and only the design of the duty-cycle detector requires custom layout. However, this circuitry is relatively small and it does not require careful calibration, as the detector is insensitive to variations of design parameters, including random-process variations.

We exploit different levels of supply voltage to increase the diagnosis resolution. The results show that the test for resistive open defects is more robust at higher voltage levels. Leakage faults, in contrast, are better detected lower voltages. Therefore, we use measurements taken at different voltage levels as input variables for the regression model in order to increase the diagnosis resolution for different fault types and different fault sizes in the presence of random process variations.

4

Uncertainty-Aware Robust Optimization of Test-Access Architectures

An undesirable outcome of the potential of 3D stacks to integrate a number of large SoCs is that the complexity and cost of test are increased. Therefore, 3D test requires careful attention in order to optimize test cost. Recent work on 3D test strategies have addressed this issue and presented several methods for test architecture optimization and test scheduling. These methods are based on exact optimization techniques such as integer linear programming (ILP) and heuristics such as rectangle packing [30, 31, 32].

A drawback of the above methods is that they consider known (constant) values for input parameters, which may differ from the actual values. This can lead to non-optimal decisions made at the design stage, increasing the test time. Variations in input parameters can be attributed to several reasons.

- At the design stage, some input parameters such as power consumption and test pattern count, are not known exactly and hence we need to rely on estimates, which can be inaccurate.

- In a 3D scenario, a die can be used as "off-the-shelf" component in various 3D stack designs with different constraints on power consumption and available bandwidth for test data. A die that is optimized for a particular 3D stack can result in non-optimal test times in other stacks.

In Section 4.1, we list examples of uncertainties in input parameters for 3D test architecture optimization and test scheduling. Neglecting these uncertainties and assuming a single point in the input-parameter space results in solutions that are not optimized for scenarios when input parameters change, leading to increased cost. Therefore, despite the large body of work on SOC test scheduling [22, 97, 23], the test scheduling and test-architecture problems in realistic scenarios for core-based 3D ICs need more scrutiny.

To deal with the issue of uncertainties in input parameters in optimization, a method called "robust optimization" has been proposed in the past [98]. This approach takes variations in input parameters into account and This approach takes variations in input parameters into account and attempts to find a solution with minimum deviation from the optimum in case the parameters change. In [98], robust optimization is used for linear programming (LP); however, the same approach can also be applied to problems modeled by integer linear programming (ILP). Not surprisingly, we found that the complexity of robust ILP models for test-architecture design grows extremely fast with the number of cores and the number of parameter variations, such that solving these problems exactly becomes intractable. Therefore, we apply an efficient heuristic method based on simulated annealing.

The main contributions of this chapter are as follows.

- We formulate the problem of robust Test Access Mechanism (TAM) optimization and test scheduling for 3D ICs in the presence of variations in input parameters. A robust solution may be inferior to a solution optimized for certain values of the input parameters in case their estimates were accurate; however, a robust solution performs better than non-robust solutions by staying closer to the optimum in the presence of variations.

- We develop a mathematical model for robust optimization in the presence of variations in input parameters, such as power, core test time, and available bitwidth, for the test architecture

- For dies in a 3D stack that have a large number of embedded cores, we propose an efficient heuristic for robust optimization. The heuristic makes use of the simulated annealing algorithm [99] to quickly explore the search space and find reasonably good solutions.

- We show that, under the likely scenario of uncertainties in parameter values, robust solutions are closer to the optimal single-point solutions compared to non-robust solutions. This "closeness" is defined for each scenario as the relative time increase with respect to the optimal single-point solution. A formal definition is presented in Section 4.3.A. Robust solutions also result in lower expectation of test time. This is an important metric, as scenarios may occur with different probabilities, and the optimization algorithm should attach a higher weight to scenarios that are more likely to occur.

The remainder of this chapter is organized as follows. Section 4.1 gives examples of uncertain parameters in optimization of 3D test architecture and test scheduling. Section 4.2 presents an overview of related prior work on robust optimization and simulated annealing. In Section 4.3, we formulate an ILP model for robust optimization of 3D test architecture and present our heuristic method for robust optimization for systems containing a large number of cores. In Section 4.4, we provide simulation results for various designs using ITC'02 test benchmarks [100]. Finally, Section 4.5

concludes the chapter.

4.1 Uncertain Parameters in Optimization of 3D Test Architecture and Test Scheduling

This section presents examples of uncertainty in parameters, including the test architecture, test power, test time, and compatibility of tests.

A possible scenario in a 3D setting is the reuse of a die as a hard-IP component in stacks with different test architectures. In that case, the test-test access mechanism of a die should be reconfigurable in order to match the test interfaces between all dies in a stack. An additional benefit of reconfigurable TAMs is to make maximum use of the channels provided by the test equipment. In both cases, the width of the TAM becomes an uncertain parameter at the design stage, since it is not known *a priori* in which configuration the die will be tested. Figure 4.1 highlights an example of such a reconfigurable architecture with a variable number of test inputs. The figure shows the partitioning of cores at die level into four groups and the routing within groups as well as between groups. In this example, there are three possible configurations with different TAM bandwidths.

- An n-bit interface. In this case, only TI1 and TO1 are used as test inputs and test outputs, respectively, such that all groups are concatenated. Two cores can only be tested in parallel if they are assigned to different TAM wires, irrespective of their group assignment. For example, $c1$ and $c12$ cannot be tested in parallel in this configuration because they share wire 3.

- A $2n$-bit interface. TI1-TI2 and TO1-TO2 are used as test inputs and test outputs, respectively. Now, Group 1 and Group 2 are on separate test I/Os than Group 3 and Group 4. Therefore, any core of Group 1 or Group 2 can be tested in parallel with all cores of Group 3 and Group 4 irrespective of the

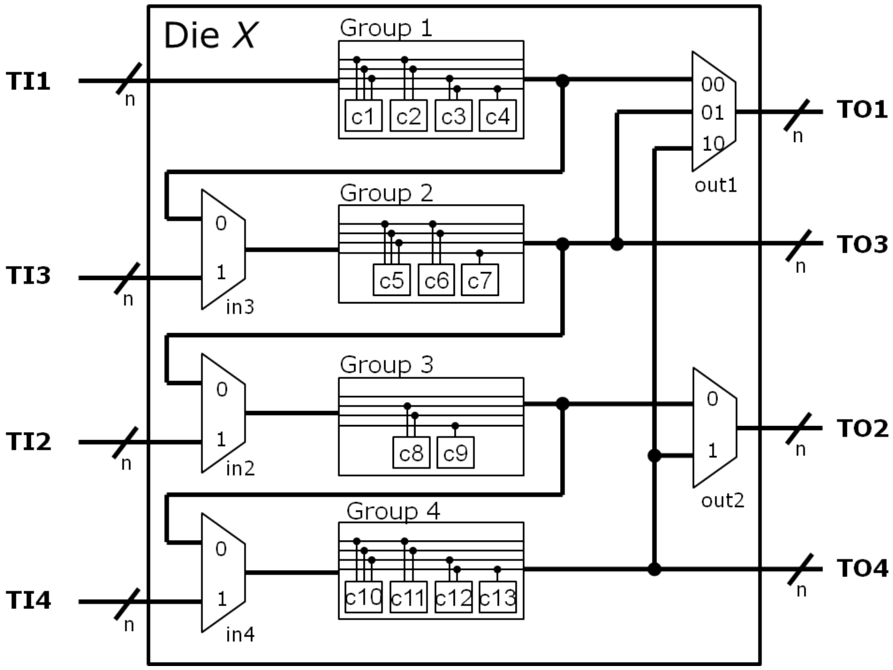


FIGURE 4.1: Die-level architecture with reconfigurable multiple-width TAM.

TAM assignment within the group, for instance $c1$ and $c12$.

- A $4n$-bit interface. In this configuration, all test inputs and test outputs are used, such that all groups can be tested independently from each other. However, the TAM-assignment within each group still limits the concurrent test of cores in the same group.

With this flexible 3D test architecture, the die can be integrated in different stacks that use n, $2n$, or $4n$ test inputs. Hence the die-level TAM width in test-architecture optimization for the stack becomes a distributed input parameter at the design stage.

This multiple-width die-level architecture is compatible with the proposed IEEE P1838 standard-under-development [101]. Even though the proposed standard was originally conceived to be used for a fixed-width parallel TAM, we can introduce additional parallel modes that use a subset of the available test data signals. In the example shown in Figure 4.1, we can match the full $4n$-bit wide test-data interface to the wrapper-level WPI. The instruction set of the wrapper instruction register (WIR) can be extended by two extra instructions to enable $2n$-bit and $1n$-bit configurations. With this extension, the die can be integrated in stacks with different stack-level WPI width, which is determined by the amount of package-level pins available for testing or by the number of available channels of the test equipment. In case only a subset of the test I/Os is used, the unused inputs will be tied to ground and the unused outputs will be floating.

Another example of uncertainty in input parameters is power consumption during test. As 3D stacking offers integration of multiple dies in a single package, the problem of high power consumption (and resulting heat dissipation) becomes even more critical than in 2D SoCs. Recent papers have identified this problem and several methods for enhanced cooling in 3D ICs have been proposed [102, 103, 104]. Nevertheless, power consumption during 3D-IC test must be limited to prevent over-heating and over-testing. This limit may not be known before the 3D-IC has been manufactured; in that case, we must rely on estimates of core power consumption obtained through simulations. These estimates can be inaccurate, e.g., due to process variations, and the actual power limit can be different from the value assumed at the design stage. In this case, tests may need to be rescheduled to satisfy the power constraint and optimize the test time. As the test architecture cannot be changed anymore, the new schedule may result in high test times. Moreover, the same die may be integrated in different stacks, each with a different package with its unique thermal properties and hence different power limits.

Variations in test time can result due to adaptive testing [105], which can be done in several ways.

- Additional types of tests might be added and because of limited test time budget, some scan tests that were considered in test architecture might have to be dropped. If some types of tests are dropped, additional scan patterns can be included due to the available test time budget.

- Sometimes test patterns might have to be dropped due to the need for truncation, which is typically motivated by limited test vector storage depth on the tester or test time limitations. This scenario can arise if tests are developed in a distributed manner and they are combined later for manufacturing test.

As the test content is dynamically adjusted during test application, the test time will change and hence should be considered as an uncertain parameter at the design stage [105]. A test architecture designed assuming nominal values for input parameters may not benefit from the updated test times, potentially limiting the advantages of adaptive testing.

4.2 Related Prior Work

Reconfigurable core wrapper approaches have been presented in [106, 107]. The proposed core wrappers allow for an adjustable core-level TAM. The scan-chains are partitioned into groups and interconnected such that the total test time is optimal for every configuration. We extend this idea to 3D wrappers and propose a reconfigurable TAM at die-level, as shown in Figure 4.1. The remainder of this section presents prior work on robust optimization for LP problems and simulated annealing.

4.2.1 Overview of Robust Optimization

The main idea of the robust-optimization method proposed in [98] is to consider a number of scenarios in each of which variable input parameters take certain values and to find a solution that stays near optimum in all scenarios. The robust optimization explicitly takes into account input-parameter variations, as a range for each input parameter is specified. The robust solution might not be the absolute optimum for a given point, but is optimized to not be too far off from that optimum as long as the input parameters are in the anticipated, pre-specified variation range. Even though a robust solution sacrifices optimality in some scenarios, it performs better on average than a single-point solution.

The following summarizes the method presented in [98]. It distinguishes between *design* and *control* variables:

- $x \in \mathbf{R}^{n_1}$, which denotes the vector of size n_1 containing the *design* variables. Their optimal values do not depend on the uncertain parameters and they cannot be adjusted once the values of the uncertain parameters are known. In our work, design parameters describe the test architecture, which is fixed at the design stage.

- $y \in \mathbf{R}^{n_2}$, which denotes the vector of size n_2 containing the *control* variables. These parameters can be adjusted once the values of the uncertain parameters are known. In our work, control parameters represent the test schedule as it can be changed even after the design has been fixed.

With design and control variables, the LP model is formulated as:

Objective:

$$\text{Minimize} \quad c^T x + d^T y$$

Subject to:

$$Ax = b,$$

$$Bx + Cy = e,$$

$$x, y \geqslant 0,$$

where A and b are constant parameters and d, B, C, and e can be uncertain. In presence of parameter variations, a set of scenarios $\Omega = \{1, 2, 3, \ldots, S\}$ is introduced. Each scenario $s \in \Omega$ occurs with the probability p_s ($\sum_{s \in \Omega} p_s = 1$), where the uncertain parameters take the values $\{d_s,\ B_s,\ C_s,\ e_s\}$. A robust solution of this LP is defined as a solution that does not move much from the optimum if a scenario $s \in \Omega$ occurs, i.e. the deviation of the cost function is minimized for the range of scenarios.

Using the set of control variables for each scenario $\{y_1, y_2, \ldots, y_s\}$ and the set of error vectors $\{z_1, z_2, \ldots, z_s\}$ that measure the allowed infeasibility, the mathematical model for robust optimization is formulated as follows:

Objective:

$$\text{Minimize} \quad \sigma(x, y_1, \ldots, y_s) + \omega\rho(z_1, \ldots, z_s)$$

Subject to:

$$\begin{aligned} & Ax = b, \\ & B_s x + C_s y_s + z_s = e_s, \qquad \forall\ s \in \Omega, \\ & x,\ y_s \geqslant 0, \qquad \forall\ s \in \Omega, \end{aligned} \tag{4.1}$$

where the function to minimize $\sigma(\cdot)$ can be either the expectation of the cost function $\sigma(\cdot) = \sum_{s\in\Omega} p_s(c^T x + d_s^T y_s)$ or the maximum value of the cost function for worst-case analysis $\sigma(\cdot) = \max_{s\in\Omega}(c^T x + d_s^T y_s)$. The function $\rho(z_1, \ldots, z_s)$ represents a penalty function to limit violations of the control constraints by assigning a weight ω. In this work, we set this function to zero.

As an example of robust optimization, let us consider a simple minimization problem $F = \min\{c^T x \mid Ax \geqslant b,\ x \geqslant 0\}$, where $c^T = [1\ 1]$, $b^T = [1\ 1]$, and A can take the values $A_{s_1} = \begin{bmatrix} 0.7 & 0.5 \\ -1.2 & 1.3 \end{bmatrix}$ in Scenario s_1 or $A_{s_2} = \begin{bmatrix} 0.65 & 0.7 \\ -1 & 1 \end{bmatrix}$ in Scenario s_2 with equal probabilities. Suppose that x_1 is a design variable, i.e. it is the same for both s_1 and s_2, and x_2 is a control variable that can be adjusted in each scenario. Table 4.1 shows the solutions of the LP problem (a) for $s1$, (b) for $s2$, and (c) taking both s_1 and s_2 into account. The conventional (non-robust) solutions are optimal for the corresponding scenarios; however, they show relatively large increase of the cost F if the

Table 4.1: Solutions of the LP problem for (a) $s1$, (b) $s2$, and (c) taking both $s1$ and $s2$ into account.

	x_1	$x_2(s_1)$	$x_2(s_2)$	$F(s_1)$	$F(s_2)$	F_{exp}
a) s_1-optimized	0.53	1.26	1.53	1.79	2.06	**1.92**
b) s_2-optimized	0	2	1.43	2	1.43	**1.74**
c) robust	0.22	1.69	1.22	1.91	1.44	**1.67**

scenario and hence the input parameters change. In contrast, in case of the robust solution, we sacrifice optimality for both scenarios, but win in terms of the expectation of the cost, if $s1$ and $s2$ occur with equal probabilities. We have proposed a method for robust optimization of test architecture and test scheduling in 2D SoCs [108]. We formulated an ILP model for the robust optimization problem, which is suitable for small SoCs and a small number of scenarios. For larger SoCs, a simple randomized divide & conquer heuristic was proposed. However, this approach does not consider a reconfigurable TAM, which is a likely scenario in 3D ICs. Therefore, while [108] is an important first step towards robust optimization for SoCs, it is not adequate for the additional uncertainty in 3D designs of reconfigurable die-level TAMs that interface to other dies.

4.2.2 Simulated Annealing

Due to the high complexity of the test scheduling problem for SoCs and 3D ICs with a large number of cores, efficient heuristics are required in order to find an acceptable solution within a reasonable time. We propose a heuristic using the simulating annealing algorithm [99]. In the past, a number of simulated annealing-based techniques for test scheduling as well as for 3D test architecture optimization has been introduced [109, 110, 111]. We exploit the concept of simulated annealing in order to develop a heuristic for the robust optimization problem. The main idea of simulated annealing is to iteratively explore the search space by perturbing the solution at each step and accepting an inferior solution with the probability

$$Pr = \exp\left(\frac{-\Delta\text{Cost}}{\text{Temp}}\right),$$

where ΔCost is the difference in the cost function and Temp is a parameter decreasing at each iteration step. The parameter Temp is analogous to the current temperature in a real annealing process. At each iteration step, the solution is perturbed and Temp decreased. In the beginning, when the Temp is high, a new inferior solution is likely to be accepted in order to prevent falling into a local optimum before a number of areas in the solution space have been explored. With decreasing Temp, this probability decays and eventually, we stay in one area and find a local optimum before the Temp reaches a specified threshold.

4.3 Robust Optimization of 3D Test Architecture

The method proposed in [98] targets robust optimization problems that can be modeled as LP and hence solved in polynomial time. However, many practical optimization problems are $\mathcal{NP}$-hard and cannot be solved in polynomial time using LP [112]. Examples of such problems include test-architecture optimization and test scheduling.

In recent work [113], we have formulated a mathematical model for robust optimization of 3D test architecture and test scheduling, and proposed a heuristic based on simulated annealing in order to solve the robust optimization problem for realistic 3D ICs. However, a limitation of this approach is that it is focused on minimizing the expectation of total test time, such that a robust solution performs best only on average, and there are no guarantees of the quality of the solutions. A disadvantage of [113] is that it disregards the fact that the robust solution may be “far” away from a point-optimal solution in some scenarios. For instance, the algorithm in [113] could generate a solution that is close to the point-optimal solutions for most cases apart from a few specific scenarios, such that the expectation of test time is relatively low. However, in the case when one of these specific scenarios occurs, the resulting test time would be significantly longer than that of a point-optimal solution. We consider a solution with a higher expectation of test time but low deviations from point-optimal solutions to be more practical, as this solution will not result in overly long test times.

Moreover, the framework in [113] uses an ILP-based heuristic that requires long CPU times to find reasonably good robust solutions for large SoCs with tens or hundreds of cores. This approach does not scale since the run time for an SOC with 22 cores is as high as 75 hours. Therefore, a more efficient heuristic is required to handle such SoCs.

In this work, we extend the previously proposed mathematical model for the robust optimization problem with the objective to keep the robust solution "close" to point-optimal solutions in every scenario. We demonstrate our framework using three uncertain parameters: (a) core-test power, (b) TAM configuration, and (c) core-test times. The set of uncertain parameters can be extended depending on the application. For instance, such an additional parameter could be the compatibility of two tests. If two cores access a shared resource during test other than the TAM wires, additional constraints need to be added; the sharing of test resources might be difficult to predict, hence it can be viewed as an additional source of uncertainty. As obtaining an exact solution for this robust-optimization problem for 3D-test architectures is intractable even for small dies, we propose an efficient heuristic based on simulated annealing. This heuristic solves the problem iteratively by finding TAM architecture candidates and evaluating them for each scenario using a scheduling algorithm.

In addition, we add core-test times as additional uncertain variables. This improves the quality of the robust solutions in case of significant deviations in core-test times, as an architecture optimized just for one set of test times may be far away from the optimal solution for another set. Adding new uncertain variables leads to an increased number of scenarios, and hence an increased complexity of the problem. However, the runtime of the proposed heuristic scales linearly with the number of scenarios, and the increase in CPU time required to solve the robust optimization problem is acceptable. Note that we are comparing one-time only compute times with test times that are recurring for every IC tested. Therefore, the larger the volume of the ICs tested, the smaller will be the relative Non-Recurring Engineering cost of compute time per IC. The new heuristic approach is also scalable for large designs, providing solutions for designs with 22 cores within 15 minutes, and the quality of the solutions thus obtained is comparable with that obtained using the previous ILP-based heuristic method. The proposed heuristic is described in detail in Section 4.3.B.

4.3.1 Mathematical Model for Robust Co-Optimization of Test Architecture and Test Scheduling

Traditional optimization methods neglect potential variations of input parameters and assume constant values instead. However, in practice, some of the input parameters may change. For instance, test power limits may vary for pre-bond, post-bond, and final test after packaging. In addition, a die can be manufactured to be stacked in different 3D ICs that have different requirements on power consumption. The originally obtained test schedule may not be optimal for all cases or it can even violate power constraints. However, the *design* variables related to the test architecture cannot be adjusted *a posteriori* for optimal scheduling. Therefore, for each scenario, the test architecture becomes a constraint for test scheduling.

Another uncertain parameter that we take into account is the width of the die-level TAM. As highlighted in Section 4.1, we can introduce a reconfigurable parallel

TAM in order to enable extra bandwidth for test data in case the stack-level test architecture allows it. To model that, we introduce a vector of integer variables g_i that holds the group number for each Core i and an integer variable C_s that encodes the configuration of the TAM in a particular scenario. In our examples, we limit the number of groups to four, hence $g_i \in \{1, 2, 3, 4\}$ and the number of configurations to three, hence $C_s \in \{1, 2, 3\}$. Since g_i defines the partitioning of the cores and therefore the die-level test architecture, this is a *design* variable. In contrast, the value of C_s depends on the scenario, i.e., on the stack in which the die will be integrated. Therefore, C_s is a *control* variable and can be adjusted after the die has been manufactured.

Let us consider a single die that is going to be stacked in a 3D IC with a daisy-chain stack-level test architecture, i.e., the stack is tested die-by-die in a modular fashion such that each die gets the full test bandwidth when it is tested. Let us denote the width of the group-level test access mechanism (TAM) n, such that the total die-level TAM width is n, $2n$, or $4n$ dependent on the selected configuration c_s as shown in Fig. 4.1. The die under test has N_c cores that need to be tested. Each core i has the TAM width W_i, test time T_i, and peak power during test P_i. Cores that share a die-level TAM bit cannot be tested in parallel. In addition, the total power during test must not exceed P_{max} at any time in order to prevent voltage droop and high heat dissipation, resulting in potential yield loss, over-testing, or even permanent damage of the 3D IC. Power limit is the only constraint we use here as example; however, we can easily extend our framework with extra constraints, such as shared test resources that make test of certain groups of cores incompatible [114]. Based on the above constraints, we need to perform three assignments. First, as all cores are initially unassigned to groups, we need to assign each core to a group. Second, as the TAM width of each core is typically smaller than the group-level TAM, therefore we

Objective:

$$\text{Minimize} \quad F = T_{\text{exp}}(1 + G\epsilon_{\text{max}}) \tag{4.2}$$

Subject to:

$$t_{i,s} + T_{i,s} \leqslant t_{\text{tot},s} \quad \forall i, s \tag{4.3}$$

$$T_{\text{exp}} = \sum_{s} t_{\text{tot},s} p_s \tag{4.4}$$

$$t_{\text{tot},s} = (1 + \epsilon_s) T_{\text{opt},s} \quad \forall s \tag{4.5}$$

$$\epsilon_{\text{max}} \geqslant \epsilon_s \quad \forall s \tag{4.6}$$

$$w_i + (W_i - 1) \leqslant n \quad \forall i \tag{4.7}$$

$$q_{i,j} \iff (w_i \geqslant w_j + W_j \text{ OR } w_j \geqslant w_i + W_i) \quad \forall i, j \tag{4.8}$$

$$q_{i,i} = 1 \quad \forall i \tag{4.9}$$

$$\text{part}_{i,s} = g_i \mod 2^{(C_s - 1)} \quad \forall s \tag{4.10}$$

$$v_{i,j,s} \iff \text{part}_{i,s} \neq \text{part}_{j,s} \quad \forall i, j, s \tag{4.11}$$

$$r_{i,j,s} \iff t_{j,s} \leqslant t_{i,s} < t_{j,s} + T_j \quad \forall i, j, s \tag{4.12}$$

$$r_{i,j,s} \leqslant q_{i,j} + v_{i,j,s} \quad \forall i, j, s \tag{4.13}$$

$$\sum_{j=1}^{N} P_j r_{i,j,s} \leqslant P_{\text{max},s} \quad \forall i, s \tag{4.14}$$

FIGURE 4.2: Mathematical programming model for non-robust test architecture optimization.

need to find an assignment of core's TAMs to specific wires of the group-level TAM. These two assignments define the architecture. Third, once the architecture is fixed, we schedule the tests such that the test time in each scenario remains "close" to the point-optimal solution.

Figure 4.2 shows our mathematical programming model for robust co-optimization of test architecture and test scheduling. In the following, we describe the details of the model.

We consider a number of scenarios, where each scenario s represents a combination uncertain parameters, i,e., a point in the input parameter space. In this work, we consider three uncertain parameters:

- Maximum-allowed test power $P_{\text{max},s}$

- Core test time $T_{i,s}$
- TAM configuration C_s

The variable C_s encodes the TAM configuration such that $C_s = 1$, $C_s = 2$, and $C_s = 3$ correspond to $4n$-bit, $2n$-bit, and $1n$-bit configuration, respectively.

Let the array of integer variables $t_{i,s}$ denote the start times of core tests i for a scenario s. The test of core i will therefore finish at $t_{i,s} + T_{i,s}$. The total test time in a particular scenario is the maximum of all finish times $t_{\text{tot},s} = \max_i\{t_{i,s} + T_{i,s}\}$. Line (4.3) represents this information.

The expectation of total test time T_{exp} is calculated based on the total test time for each scenario and the probability with which this scenario occurs, as shown in Line (4.4). For each scenario, we calculate the relative distance $\epsilon_{s,c}$ from a given point-optimal solution as $\epsilon_s = (t_{\text{tot},s} - T_{\text{opt},s})/T_{\text{opt},s}$, which is formulated as a constraint in Line (4.5). The objective of the programming model is to co-optimize T_{exp} and $\epsilon_{\max}$. We use the cost function $F = T_{\text{exp}}(1 + G_w\epsilon_{\max})$ as shown in Line (4.2), where G_w is a weight that shifts the focus between minimizing T_{exp} and minimizing $\epsilon_{\max}$. For instance, if $G_w = 0$, $\epsilon_{\max}$ is ignored, whereas with larger values of G_w, solutions with low $\epsilon_{\max}$ are preferred over solutions with low T_{exp}.

Next, the integer variables w_i, $1 \leqslant i \leqslant N$ indicate the lowest group TAM wire to which core i is connected. The last bit of the core TAM is hence connected to TAM wire $w_i + (W_i - 1)$. Line (4.7) adds the constraint the limit on maximum available TAM width.

In order to restrict parallel testing of two cores that “overlap” in die-level TAM, we introduce a matrix of binary variables q, such that

$$q_{i,j} = \begin{cases} 1 & \text{if Core } i \text{ and Core } j \text{ do not share a TAM wire} \\ 0 & \text{else} \end{cases} \tag{4.15}$$

The variables $q_{i,j}$ and w_i are therefore constrained as shown in Line (4.8) and Line (4.9).

The operator " $\iff$ " means "implication", i.e., the variable is 1 if and only if the condition is met. In addition, each Core i belongs to a group g_i. Depending on the TAM configuration, groups are divided into partitions, such that each partition connects to an independent die-level TAM bus. For $1n$-bit configuration, all groups are concatenated and hence all cores belong to one partition. For $2n$-bit configuration, there are two independent partitions. For a $4n$-bit configuration, each group belongs to an separate partition and hence connects to an independent die-level TAM. This is represented in Line (4.10). We introduce a new matrix of binary variable v, such that

$$v_{i,j,s} = \begin{cases} 1 & \text{if Core } i \text{ and Core } j \text{ do not share} \\ & \text{the same partition in configuration } c \\ 0 & \text{else} \end{cases}$$

The variables $v_{i,j,s}$ are constrained as shown in Line (4.11).

The start times $t_{i,s}$ are constrained through (a) the TAM assignment, (b) grouping and (c) maximum-allowed test power. The constraints due to TAM assignment and grouping are modeled as follows. We introduce a matrix of binary variables r, such that $r_{i,j,s}$ is 1 if the test of Core i starts while test of Core j is in progress in a scenario $\{s, c\}$. This is modeled in Line (4.12). In case that $t_{i,s} = t_{j,s} + T_j$, then $r = 0$, which implies that the test of Core i starts right after the test of Core j and therefore the tests do not overlap. Tests of Core i and Core j can only overlap if the cores do not share the same group TAM wire ($q_{i,j} = 1$) or belong to a different partition ($v_{i,j,s} = 1$). This constraint is represented in Line (4.13).

The power constraint is modeled as follows. For each core, we check at the starting time of each core whether the total power of the cores being tested at this time exceeds $P_{max,s}$. This constraint is captured in Line (4.14).

In total, we have the following *design* variables that cannot be adjusted after die manufacturing: w, q, and g. These variables are independent of the current scenario.

Other variables, i.e., t, t_{tot}, $part$, v, and r, are *control* variables and can be adjusted to optimize ϵ dependent on the current scenario.

Solving this problem provides an optimal core TAM assignment (w_i), grouping (g_i) and a set of optimal test schedules ($t_{i,s}$) for each scenario s.

The constraints (4.8), (4.10), and (4.11) are non-linear but can be linearized using standard linearization techniques in order to convert the model to ILP. To linearize Constraint (4.8), we first convert it to

$$q_{i,j} \iff k_{i,j} \text{ OR } m_{i,j}, \tag{4.16}$$

where $k_{i,j}$ and $m_{i,j}$ are auxiliary binary variables that imply $w_i \geqslant w_j + W_j$ and $w_j \leqslant w_i + W_i$, respectively. These implications are of the form "$z_{\text{bin}} \iff x_{\text{int}} \geqslant y_{\text{int}}$", where z_{bin} is binary, and x_{int} and y_{int} are integers. This type of a non-linear constraint can be linearized using the following technique:

$$x_{\text{int}} \geqslant y_{\text{int}} - U(1 - z_{\text{bin}}) \tag{4.17}$$

$$x_{\text{int}} \leqslant y_{\text{int}} - 1 + U z_{\text{bin}}, \tag{4.18}$$

where U is a sufficiently large constant number.

Expression (4.16) is then linearized using $a_{i,j}$ and $b_{i,j}$ as

$$q_{i,j} \geqslant k_{i,j} \tag{4.19}$$

$$q_{i,j} \geqslant m_{i,j} \tag{4.20}$$

$$q_{i,j} \leqslant k_{i,j} + m_{i,j}. \tag{4.21}$$

Constraint 4.10 is linearized by introducing auxiliary integer variables $d_{i,s}$ and adding the following constraints:

$$g_i = d_{i,s} 2^{C_s - 1} + \text{part}_{i,s} \tag{4.22}$$

$$\text{part}_{i,s} \leqslant 2^{C_s - 1} - 1. \tag{4.23}$$

Constraint (4.11) can be expressed as

$$v_{i,j,s} \iff \text{part}_{i,s} \geqslant \text{part}_{j,s} + 1 \text{ OR } \text{part}_{i,s} + 1 \leqslant \text{part}_{j,s}, \tag{4.24}$$

which is easily linearized using the same technique as for Constraint (4.8).

The optimal point-optimal solutions are obtained using the model described above. For each scenario, we set s and c to the corresponding values, and set $T_{opt,s}$ to a constant "dummy value", e.g., $T_{opt,s} = 1$. Since only one scenario is active, the solver will effectively optimize the total test time for this scenario $t_{tot,s}$. The obtained values are used as $T_{opt,s}$ during the robust optimization over all scenarios.

In our simulations, a commercial ILP solver was not able to provide an optimal solution even for a 7-core die within 24 hours. Therefore, we focused in this work on developing an efficient heuristic to handle dies with more than 20 cores.

4.3.2 Heuristic Method for Robust Optimization Based on Simulated Annealing

Due to the high complexity of the robust optimization problem for dies with even a small number of cores, we need alternative methods to handle realistic designs. In this work, we propose a heuristic method based on simulated annealing (SA).

Figure 4.3 provides on overview of our SA algorithm. The following text describes the main idea of the algorithm. Later, we will focus on the detailed description of the important blocks. In the beginning, we set the initial temperature, and find an initial grouping of cores and a TAM assignment of each core to the bus of n wires. Next, we randomly perturb the TAM assignment by rearranging some of the cores. The new test architecture is evaluated by finding an near-optimal test schedule for each scenario s, and the cost function F based on ϵ and T_{exp} over all scenarios is kept track of as F_{new}. If the cost of the new solution is not higher than that of the current solution, the new test architecture is accepted. Otherwise, it is accepted with the probability

$$Pr = \exp\left(\frac{F_{\text{cur}} - F_{\text{new}}}{\text{Temp}}\right). \tag{4.25}$$

If the temperature is above a specified threshold Temp_{min}, the temperature is de-

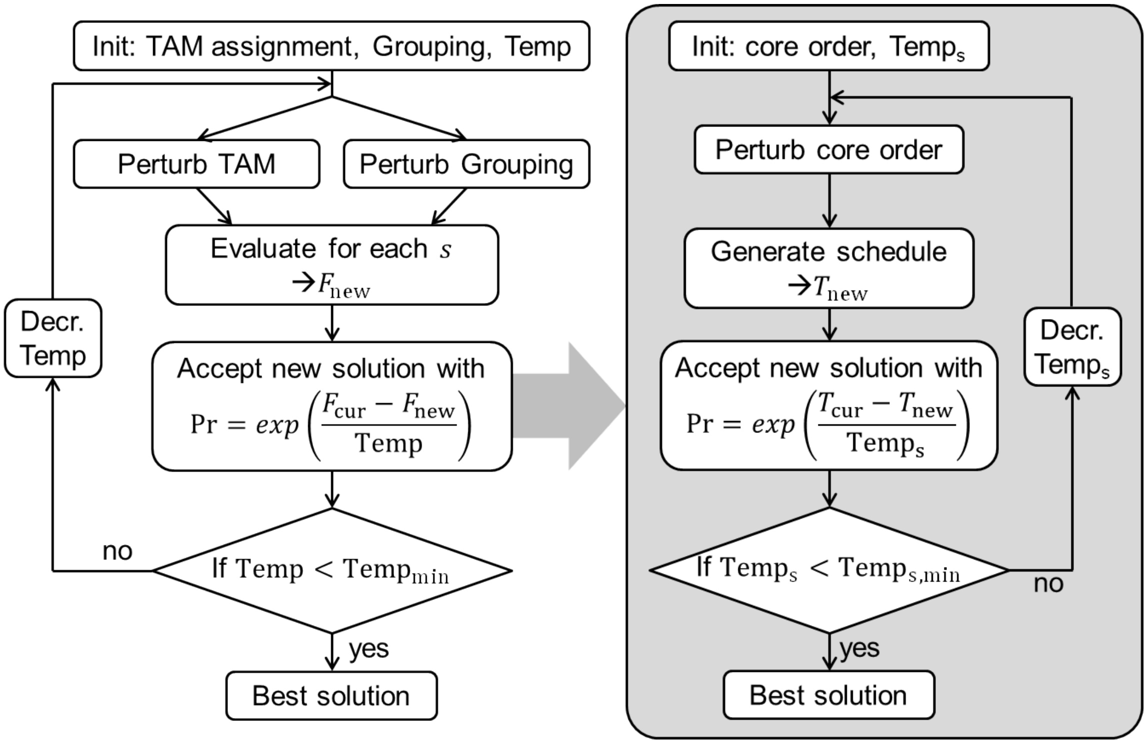

Init: TAM assignment, Grouping, Temp
Perturb TAM
Perturb Grouping
Evaluate for each s → F_{new}
Accept new solution with $\Pr = exp\left(\frac{F_{\text{cur}} - F_{\text{new}}}{\text{Temp}}\right)$
If $\text{Temp} < \text{Temp}_{\text{min}}$
no
yes
Decr. Temp
Best solution
Init: core order, Temp_s
Perturb core order
Generate schedule → T_{new}
Accept new solution with $\Pr = exp\left(\frac{T_{\text{cur}} - T_{\text{new}}}{\text{Temp}_s}\right)$
If $\text{Temp}_s < \text{Temp}_{s,\text{min}}$
no
yes
Decr. Temp_s
Best solution

FIGURE 4.3: SA-based Robust Optimization Flow.

creased by a specified factor and the algorithm goes back to the perturbation phase. This time, the grouping of the cores is slightly perturbed and the new test architecture is evaluated again. Perturbation of TAM and the grouping alternate between each iteration. Once the temperature has fallen under Temp_{min}, the algorithm stops and the best solution so far is provided as the result. In the following, we describe each step in detail.

The die-level TAM assignment is defined by the vector $w = [w_1, w_2, \ldots, w_{N_c}]$, as described in the mathematical model in Figure 4.2. The number of all feasible variations of w can be extremely large and hence completely random TAM perturbations would require an impractically large number of iterations. In our algorithm, we try to improve our solution by using the following properties of w.

First, there is a number of TAM assignments that correspond to the same overlap matrix q, which is defined in Equation (4.15). Since only q directly limits which cores can be tested in parallel (Equation (4.13)), all these TAM assignments will result in the same optimal schedule and hence the same minimum test time. Therefore, it is sufficient to perturb the matrix q and to find which one gives the best schedule. The actual solution is then any TAM assignment that maps to this q.

Second, assume a TAM assignment in which Core i and Core j overlap in TAM and therefore cannot be tested in parallel ($q_{i,j} = q_{j,i} = 0$). Let us change $q_{i,j} = q_{j,i}$ to 1, which means Core i and Core j are moved away from each other such that they do not overlap in TAM anymore. All test schedules that were feasible for the previous TAM assignment will remain feasible. Potentially, we can improve the test time by testing Core i and Core j in parallel. Therefore, it is enough to consider the new matrix as a candidate and discard the old (inferior) matrix. We exploit this property to find an initial TAM candidate and to perturb it. First, we start with an identity matrix (all cores share TAM wires). This is the most inferior solution as it only allows for serial testing. However, it is guaranteed that this TAM assignment is feasible because we require that $W_i \leqslant n$ for all i. Next, we try to improve the candidate by setting some randomly chosen entries to 1. By doing this, we force certain cores to be assigned to distinct TAM wires and therefore increase the minimum required group-level TAM width for the new q. Once this width exceeds the given maximum TAM width n, q becomes infeasible, hence we need to undo the change and try to change another entries of q to 1. After we have tried setting all entries to 1, we stop and receive a TAM candidate that is evaluated in the next step.

The checking of q is done by mapping it to the maximum-weight clique problem and solving it using an existing algorithm [115]. The mapping is done as follows. Each Core i is represented by a vertex v_i with the weight W_i. An edge between two vertices v_i and v_j exists if $q_{i,j} = 1$. The objective of the algorithm is to find a clique (a set of vertices in which there is an edge between every pair of two vertices) with the maximum total weight. In our original problem this clique is a set of cores that can be tested simultaneously and "span" the maximum width. If the weight of the maximum-weight clique exceeds n, the given TAM assignment is infeasible, as it requires more TAM wires than available. If the maximum weight does not exceed n, q is feasible and we can find a corresponding TAM assignment. The maximum-weight clique problem is $\mathcal{NP}$-hard; however, we can solve it within microseconds for dies with $N_c < 100$. We use this number, as SoCs currently available on the market integrate less than 100 IP blocks [116]. The time required to solve the maximum-weight clique problem is many orders of magnitude shorter than the CPU time required for evaluation of a candidate over the entire range of scenarios, hence the CPU time for generating a TAM candidate is negligible, even though we call the check function $O(N_c^2)$ times during each perturbation.

Perturbation of a given q is done as follows. We "reset" a number of elements back to 0 and try to randomly fill the matrix with 1 until no zero-element can be set to 1. If the number of elements that is reset is large enough, we will receive a different TAM candidate with high probability. In our implementation, we randomly choose a row and a column and reset their elements. However, our framework can be easily adjusted to perform a different type of perturbation.

The grouping of cores is stored in the vector $g = [g_1, g_2, \ldots, g_{N_c}]$. The initial grouping is assigned randomly. In order to perturb g, we perform either of the following operations with equal probability: (a) we swap two randomly chosen cores from different groups, (b) we move a randomly chosen core to a different group. During SA iterations, we keep track of the best solution in terms of the expectation of the test time. Once the temperature falls below a threshold, the best combination of a TAM assignment and grouping is provided as a result.

In the evaluation phase, we calculate the expectation of the total test time and maximum epsilon for the given TAM assignment and grouping. Since all design variables are fixed, the solutions (schedules) for each scenario s are not "coupled" between each other. This allows us to partition the evaluation problem in S sub-problems, where each sub-problem is a non-robust scheduling problem with the corresponding maximum-allowed test power $P_{\max}$, test core times T_i, the given TAM assignment q, and grouping g. In the previously proposed framework in [113], we used an ILP-based heuristic to solve this scheduling problem. However, the ILP-based approach is inefficient for large SoCs as it requires 75 hours to solve the robust optimization problem for a 22-core SoC from our benchmark set. We propose a new heuristic based on simulated annealing for evaluation of TAM architectures. As simulations show, the new algorithm finds reasonably good solutions orders of magnitude faster compared to the previous ILP-based algorithm.

The right-hand side of Figure 4.3 shows a flow diagram of the new evaluation algorithm. The key idea of this algorithm is to start with an initial schedule and then iteratively perturb the order of tests at every iteration. A schedule is constructed using the following greedy approach. We go through the list of all cores one by one in a particular order and assign the start of each core test to the earliest time possible, without violating the given constraints. These constraints include $P_{\max}$, q, and g. For example, consider a list of four cores $\{C_3, C_2, C_4, C_1\}$. The algorithm assigns the start time of the first core in the list, C_3, to $t = 0$. This is always a feasible step as no other test has been scheduled yet. Next, the algorithm assigns C_2 to $t = 0$ and checks for feasibility. If no constraint is violated, the algorithm proceeds to the next core. Otherwise, C_2 is rescheduled to the next potential start point, which coincides with the finish time of C_3. This procedure repeats until all four tests are scheduled. This algorithm loops through all N_c cores in the list and the last core is tried to be assigned N_c times in the worst case. Therefore, the algorithm has the run-time complexity $O(N_c^2)$ and requires low CPU time even for large SoCs. As the order of cores defines the constructed schedule, we can explore the solution space by perturbing this order. In our approach, we start with an initial (random) core-test order and move a randomly selected core to another position in the list.

At each iteration, a new schedule with the total test time T_{new} less or equal the total test time T_{cur} of a current schedule is always accepted; otherwise, it is accepted with the probability

$$Pr = \exp\left(\frac{T_{\text{cur}} - T_{\text{new}}}{\text{Temp}_{\text{s}}}\right), \tag{4.26}$$

where Temp_{s} is a variable ("temperature") that is decreased every iteration. The algorithm terminates after Temp_{s} reaches a given threshold, and the best solution is kept.

4.4 Simulation Results

We present simulation results to evaluate the proposed heuristic method for robust optimization. The framework is implemented in C++.

First, we demonstrate the effect of robust optimization using a simple example. Next, we show simulation results obtained with publicly available benchmarks.

Consider a die containing 5 cores with equal TAM widths $W_i = 2$ and the test times {6,5,4,2,6} in some arbitrarily chosen units. The die-level TAM can be reconfigured in 4-bit (n) and 8-bit (n) modes. Due to the power constraints, only three cores can be tested at a time. Figure 4.4 and Figure 4.5 show two test architectures (core TAM assignment and core partitioning) with the corresponding optimal schedules for both configurations, resulting in test times of {12,11} and {14,9}, respectively. If the die will be used in both configurations with equal probabilities, the expectation of the test time is 11.5, regardless of the architecture. However, if one scenario is more likely than the other, one architecture will perform better on average than the other.

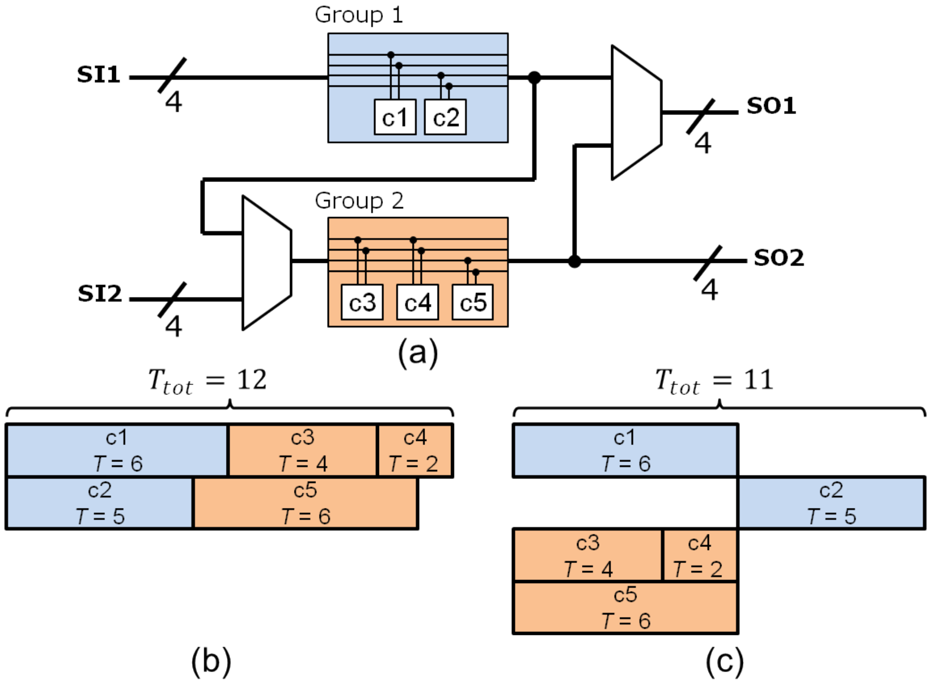


FIGURE 4.4: Test Architecture 1: (a) TAM assignment and partitioning, (b) schedule for 4-bit configuration, (c) schedule for 8-bit configuration.

For instance, a non-robust algorithm optimizing just for a 4-bit configuration would chose Test Architecture 1 (total test time 12 vs. 14). However, in the case of an 8-bit configuration, this solution would be further away from an optimal solution than Test Architecture 2. If that case is more likely, Test Architecture 2 would be preferred by a robust-optimization algorithm, as it considers the probability with which different scenarios are likely to occur.

Since there are no publicly available 3D-IC benchmarks, we applied our methods to SoCs from the ITC'02 SoC Test Benchmark set [100] and two industrial SoCs that we use as dies in a 3D stack in our simulations. Table 4.2 summarizes the relevant design data of the benchmarks, h953, d695, p93791, p22810, including the core TAM width W, the core test time T in units of 1000 clock cycles, and the percentage of the scan flip-flop count in relation to the total scan flip-flop count. Cores without scan chains were excluded from the set; therefore, the dies have 7, 8, 12, and 22 cores to

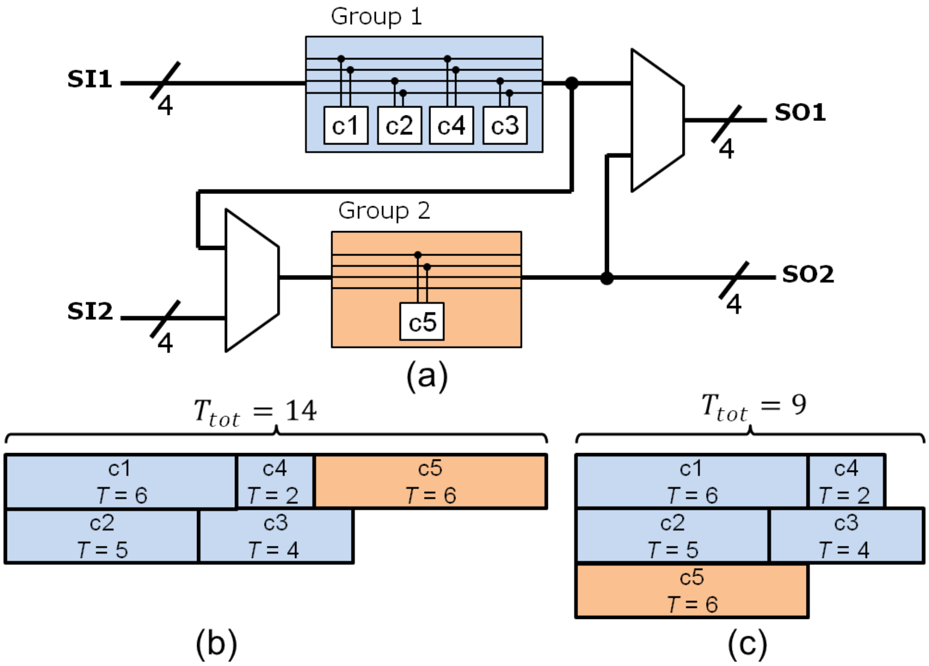


FIGURE 4.5: Test Architecture 2: (a) TAM assignment and partitioning, (b) schedule for 4-bit configuration, (c) schedule for 8-bit configuration.

be connected to the system TAM, respectively. In addition, we used the combination of the SoCs h953 and d695 as the fifth die. In order to test the performance of the framework, we created an SOC with 100 cores, comprising 16 large identical cores with 32-bit TAM and relatively high test power and long test time, 32 medium identical cores with 16-bit TAM and medium test power and test time, and 52 small cores with randomly chosen parameters. We approximated the test time of each core by multiplying the length the longest scan chain with the test pattern count.

We assume that the peak power during test depends linearly on the amount of logic that is switching, which is proportional to the amount of scan flip-flops. Therefore, we set the constraint on power during test by limiting the fraction of active scan flip-flops. For instance, if we set $P_{max} = 0.5$ for h953 then Core 1 cannot be tested in parallel with Core 7 because the power would exceed the limit and the schedule would become infeasible. For SOC-A and SOC-B, P_i were estimated using simulations. Due

Table 4.2: Design parameters of some of the benchmark SoCs used in simulations.

	h953			**d695**			**p93791**			**p22810**		
core	W	T	% FF	W	T	% FF	W	T	% FF	W	T	% FF
1	4	118	24.2	16	94	10.0	46	113	26.5	10	1021	5
2	2	3	14.0	4	56	3.3	44	75	8.3	29	432	9
3	2	1	1.4	32	50	22.5	46	68	7.6	24	214	9
4	4	1	1.8	4	44	2.8	46	62	3.4	4	38	1
5	4	13	10.4	32	35	25.5	46	42	10.6	8	83	2
6	4	33	15.8	16	31	8.3	46	42	10.6	11	76	3
7	8	57	32.4	1	24	0.5	46	40	8.5	4	1	1
8				32	6	27.0	46	36	4.7	3	79	1
9							35	32	7.3	6	30	1
10							43	32	7.1	1	9	1
11							44	21	4.8	4	22	1
12							11	15	0.6	5	438	1
13										3	25	1
14										4	95	1
15										10	848	4
16										3	45	1
17										7	46	4
18										5	27	1
19										18	389	9
20										31	724	46
21										1	1	1
22										5	26	1

to inaccuracies of these simulations, these data will be subject to variations that can be handled through robust optimization. In addition, P_{max} may vary significantly in case a die is integrated in different 3D stacks with different thermal properties. In practice, each of these parameters can vary independently from each other, which increases the complexity of the robust optimization problem. In our simulations, we consider P_{max}, T_i, and *conf* as uncertain parameters. Without loss of generality, we assume three discrete points for P_{max}, such that the nominal value occurs with the probability of 0.6 and the other two points $P_{max} \pm \Delta P$ with the probability of 0.2 each. Furthermore, we assume three different sets of T_i: (1) a set with nominal T_i, (2) a set, in which some of the core test times are reduced by 10%, and (3) a set, in which some of the core test times are reduced by 20%. Each set can occur with equal probability. The third input parameter that is subject to variations in our framework is the configuration of the TAM. We limit the possible configurations to n-bit, $2n$-bit, and $4n$-bit configurations, as shown in Figure 4.1, and assign equal probabilities to them. As all three input parameters can take three different values independently from each other, the total number of scenarios we consider is $S = 3^3 = 27$. In contrast to the exact ILP model, the complexity of our heuristic algorithm grows only linearly with the number of scenarios, as we solve the scheduling problem for each scenario independently. Therefore, we can easily extend our framework to handle more uncertain parameters while maintaining reasonable CPU-runtimes.

We limited the number of SA iterations to 250 in order to experiment with the setting and collect the results in a reasonable time. In addition, we ran every simulation ten times in parallel and picked the best result. In a real application, the number of SA iterations can be increased in order to improve the quality of the solution. The CPU time in our simulations varied from a few seconds for small benchmarks to a few hours for the 100-core benchmark. This is several orders of magnitude faster compared to the ILP-based heuristic used in [113], but the results are comparable in quality to the computationally expensive method from [113].

In order to demonstrate the advantage of robust solutions, we performed the following simulation. For each SoC, we obtained an optimal solution assuming a single point in the input parameter space (the nominal value of P_{max} and T_i, and the $1n$ configuration). For the robust solutions, we considered 27 scenarios for variable P_{max}, T_i, and *conf*. The non-robust solutions were evaluated for the same scenarios as the robust solutions in order to compare their performance in the presence of uncertainties in the input parameters. Table 4.3 shows the input parameters $W_{m,n}$ (TAM width for the case of $1n$-bit configuration), $P_{m,n}$ (nominal value of P_{max}), and a comparison of non-robust and robust solutions in terms of T_{exp} and $\epsilon_{\max}$. Note that

Table 4.3: Input parameters and resulting test times for the SoC benchmarks.

Benchmark	$W_{m,n}$	$P_{m,n}$	T_{exp}			$\epsilon_{\max}$		
			non-rob	rob	impr	non-rob	rob	Δ
h953	10	60						
d695	32	50	132	114	13.64%	0.40	0.00	40%
p93791	100	45	165	150	9.09%	0.51	0.09	42%
d695_h953	32	60	172	141	18.02%	0.54	0.10	44%
p22810	50	60	1453	1323	8.95%	0.56	0.12	44%
100-core	64	50	753	699	7.17%	0.22	0.06	16%

the difference in $\epsilon_{\max}$ for the 100-core benchmark is only 16%. Due to the combination of the input parameters for this benchmark, the non-robust algorithm was able to find a non-robust solution that was close to the optimum ($\epsilon_{\max} = 0.22$), hence the robust algorithm had little room to improve the solution.

As the results indicate, test architectures that were optimized using the proposed robust optimization method lead to significantly lower $\epsilon_{\max}$, i.e., the robust solutions stay "closer" to point-optimal solutions compared to non-robust solutions. In addition, we observe a measurable improvement of the expectation of test time for most SoCs.

Figures 4.6–4.8 show a detailed evaluation of non-robust and robust solutions for all 27 scenarios. We observed that both robust and non-robust solutions deviate from point-optimal solutions in several scenarios. However, for some scenarios, the test time of the non-robust solution is significantly higher than that of the robust solution, for instance, in the case of $2n$-bit TAM configuration and $P_{max} = 120\%$. This is less prominent in the case of $1n$-bit and $4n$-bit TAM configurations. The reason for that is the following. The non-robust algorithm targets a $1n$-bit TAM scenario, hence the solutions perform well for that case. For $4n$-bit TAM scenarios, the test schedule is less constrained due to a larger TAM (i.e., extra options to schedule cores from different groups in parallel, which may not be possible in other cases), and the optimization algorithm is therefore more likely to "pack" the schedule tightly

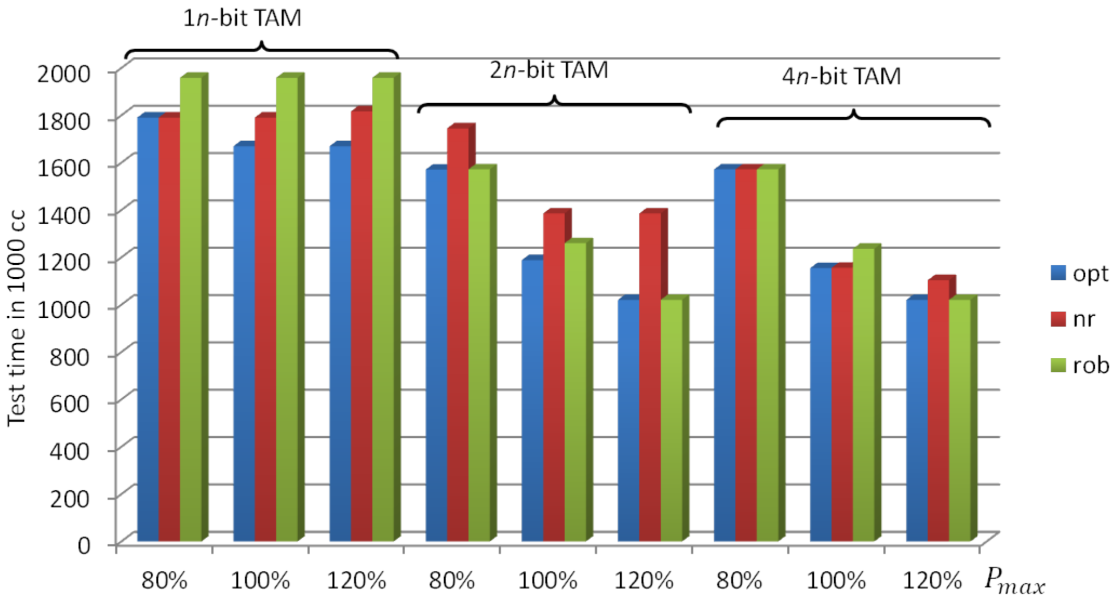


FIGURE 4.6: Evaluation of the robust and non-robust solutions for p22810 in $1n$-bit, $2n$-bit, and $4n$-bit configuration for different P_{max} for nominal core test-time values T_i.

up to the maximum-allowed power limit. Therefore, non-robust solutions tend to be closer to the optimum in $4n$-bit TAM scenarios.

For $1n$-bit scenarios in Figure 4.8, the optimal, non-robust, and robust solutions seem to be similar. This is an example of the case when different types of solutions are pareto-optimal, which can occur depending on input parameters.

We observed a similar behavior for other benchmarks, and we conclude that non-robust solutions can lead to prohibitively high test times.

4.5 Conclusion

We have presented a method for robust optimization of 3D test architecture and test scheduling in the presence of input parameter variations. We have formulated a mathematical model for the robust optimization problem using ILP. As the ILP model does not scale well with the problem size, we propose a heuristic based on the widely used simulated-annealing technique that optimizes the TAM assignment

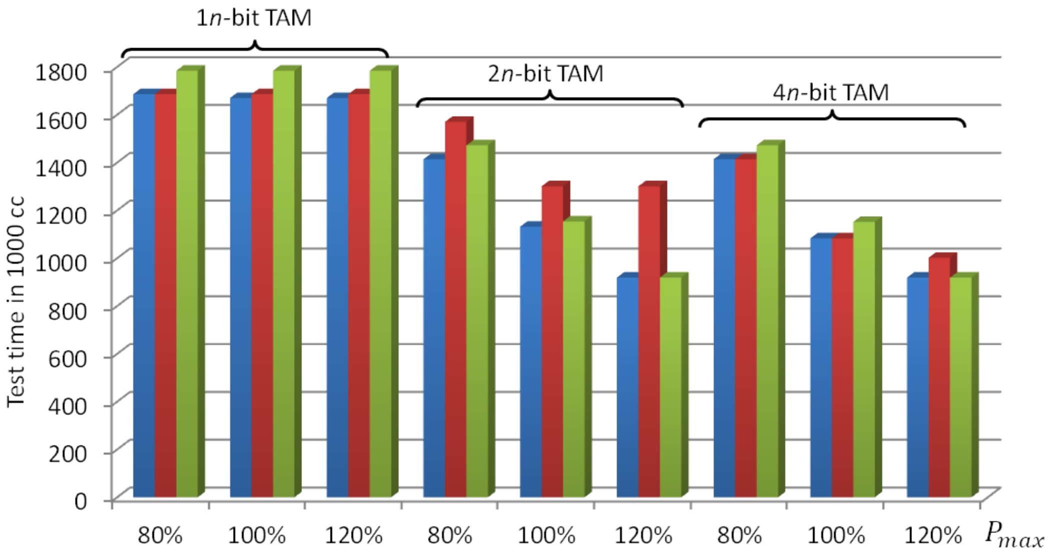


FIGURE 4.7: Evaluation of the robust and non-robust solutions for p22810 in $1n$-bit, $2n$-bit, and $4n$-bit configuration for different P_{max} for some core test-time values T_i reduced by 10%.

of the cores by perturbation and evaluation of the solution for the given scenarios of uncertain parameter values. Results show that neglecting variations in input parameters will result in inefficient single-point solutions with increased test time. The proposed method scales well with the complexity of the die and with the number of scenarios. The designed framework can easily be extended with additional constraints and uncertain parameters.

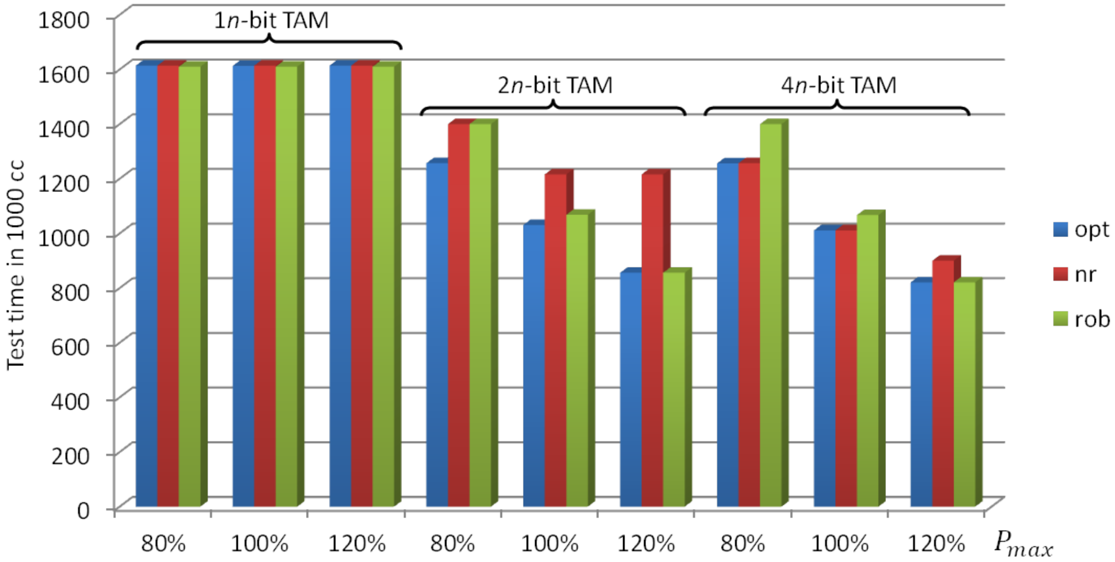


FIGURE 4.8: Evaluation of the robust and non-robust solutions for p22810 in $1n$-bit, $2n$-bit, and $4n$-bit configuration for different P_{max} for some core test-time values T_i reduced by 20%.

5

Massive Signal Tracing Using On-Chip DRAM for Silicon Debug

A commonly used post-silicon verification method is based on real-time signal observation using on-chip trace buffers [7]. However, a major problem in traditional signal tracing is the limited storage available for the trace data; hence, only a small number of internal nodes can be observed over a short period of time. Several methods have been proposed in the past to increase the observation window by incorporating on-chip compression and selective capture of debug data [75, 74]. Even though these methods increase the observation windows to thousands of clock cycles, this is still a small fraction of the duration of functional tests that runs over multiple seconds, i.e., billions of clock cycles [117]. As the signals can only be traced over a limited time frame, many debug sessions are typically required to localize the error in the design, resulting in labor-intensive debug sessions.

One of the limiters in trace-buffer-based debug is the relatively small size of the trace buffers, which is necessitated by the need to reduce area overhead. In order to overcome this limitation, we can use functional on-chip memories for debug purposes.

Such memories are expected to be available in future semiconductor products. For instance, 3D-stacked integrated circuits (ICs) have the potential to integrate DRAM dies with high bandwidth. Emerging standards for fast DRAMs, such as JEDEC Wide I/O 2 [118] and Hybrid Memory Cube (HMC) [119, 120], are aimed at 3D-IC architectures and promise memory bandwidth of 51.2 GB/s and 320 GB/s, respectively [121].

We propose a post-silicon debug framework for 3D ICs with on-chip DRAM. The main contributions of this chapter are the following:

- We propose a new design-for-debug (DfD) architecture that exploits fast DRAM as trace-data storage, which allows for massive signal tracing and the extension of the observation window by orders of magnitude compared to previously proposed methods, while requiring negligible silicon area overhead.

- The proposed method does not require multiple iterations or extra hardware to process the captured data in between debug sessions. The debug data is captured at-speed during the entire execution of a test program. The only overhead is the transfer time of the expected signatures to the DRAM and the transfer time of the stored debug data from the DRAM back to the debug equipment.

- The proposed technique can handle partially non-deterministic tests by capturing deterministic and hence predictable segments of the test using transition-matching circuitry.

- The proposed method can be applied both in-field during platform debug and system operation, and on automatic test equipment (ATE) during functional test execution. The captured data of the failing units can also be stored for offline analysis.

The remainder of this chapter is organized as follows. Section 5.1 describes our debug methodology using on-chip DRAM. In Section 5.2, we present simulation results. Finally, Section 5.3 concludes this chapter.

5.1 Proposed Debug Framework

Despite the improvements described in Section 1.6, debug using trace buffers still remains a time-consuming procedure. We propose a new debug solution for ICs with on-chip DRAM that overcomes the limitation arising from the small size of trace buffers. The main idea of our method is to transfer the captured data from small trace buffers into a much larger on-chip DRAM, thereby extending the observation window by orders of magnitude compared to traditional methods. We use multiple-input signature registers (MISRs) to generate independent signatures for each time interval in order to identify failing intervals and store them. Hence, we can skip error-free intervals and increase the observation window for the same amount of available memory. In addition, a trigger module can be implemented in order to "snoop" on the observed bus and issue a trigger when a specified transaction occurs on the bus. This trigger will initiate the tracing procedure, skipping irrelevant and potentially non-deterministic parts of the test. In this section, we first provide a brief overview of the proposed debug flow and the debug architecture and finally, discuss details of the implementation.

Fig. 5.1 shows an overview of the proposed debug flow. In Step 1, prior to the actual debug session, "golden" signatures are generated by simulation of the test program using a behavioral model of the circuit. Alternatively, the signatures can be calculated using emulations on a FPGA prototyping board. This approach is especially useful for large SoCs, simulations of which may be impractical. As multiple sets of internal signals can be tapped for tracing, a separate set of golden signatures is needed for each signal set. However, all signature sets can be generated in one sim-

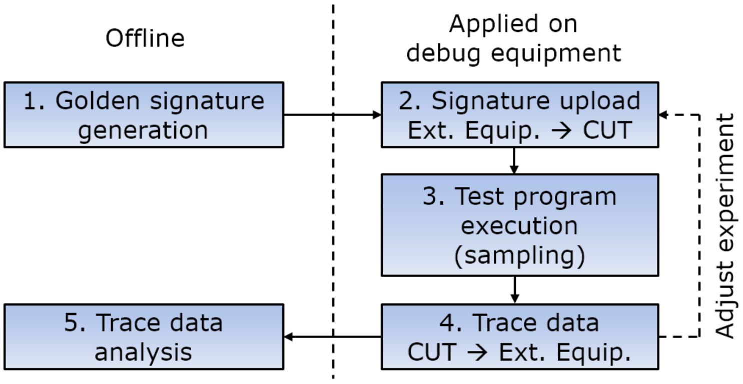

Offline
Applied on debug equipment
1. Golden signature generation
2. Signature upload Ext. Equip. → CUT
3. Test program execution (sampling)
4. Trace data CUT → Ext. Equip.
5. Trace data analysis
Adjust experiment

FIGURE 5.1: Proposed debug flow.

Table 5.1

Symbol	Meaning
L_{t}	Number of observed signals
M_{t}	Buffer depth and interval length in cycles
K_{t}	MISR length
N_{t}	Length of the observation window in cycles
S_{t}	Timestamp length
W_{mem}	Memory-interface width

ulation run. Steps 2-4 are applied on the debug equipment. In Step 2, a set of golden signatures for the selected internal bus to trace are uploaded to the circuit under test (CUT), for instance, through an IEEE 1149.1 (JTAG) interface [21] or through shared I/O chip pins. In Step 3, the test program is executed, the selected signals are traced, and the trace data is stored into the DRAM. In Step 4, the stored data is downloaded to the external equipment and stored for analysis. Next, another debug session can be executed with a different setup, for instance, (a) with different trace signals, (b) different test program, or (c) different temperature and voltage settings. Finally, in Step 5, the obtained trace data is analyzed, which can be performed offline on a work station, freeing the test equipment for another experiments.

Fig. 5.2 provides an overview of the proposed architecture, including the data flow between modules. In this work, we use the notation as shown in Table 5.1, which is similar to that used in [74] for ease of comparison.

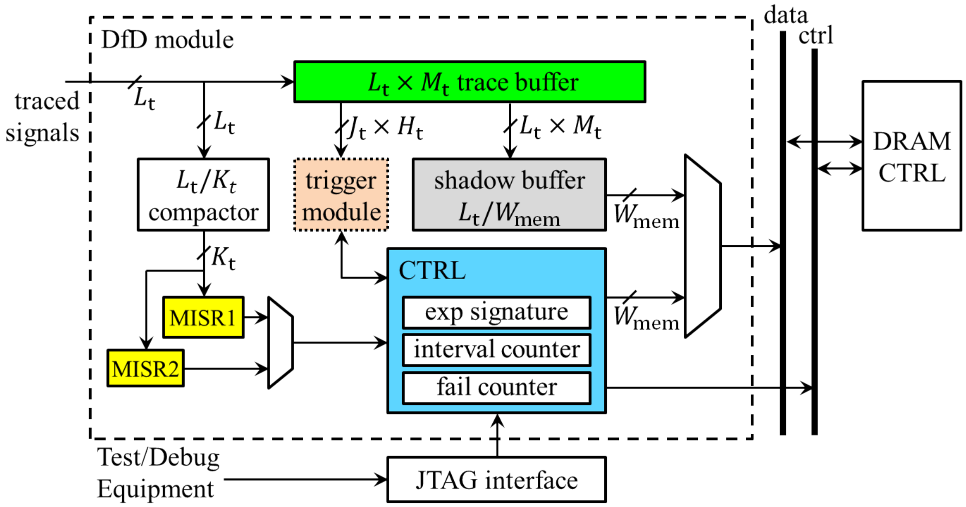


FIGURE 5.2: Overview of the proposed design-for-debug architecture.

We capture L_t signals at the functional clock frequency and shift in the sampled values into an $L_t \times M_t$ trace buffer, where L_t is the width and M_t is the depth of the buffer. This buffer serves as a temporary storage of trace data captured in one time interval of the length M_t. At the same time, we feed the observed signals through an L_t-to-K_t compactor into an L_t-bit multiple-input signature register (MISR) to identify failing intervals and skip the storage of error-free intervals. The compactor is optional and its purpose is to match the input width of the MISR to the width of the observed bus; it can be implemented as an XOR tree. The MISR calculates a K_t-bit long signature for each interval of M_t cycles that is used to identify failing intervals by comparing it with expected signatures that are stored in DRAM. Once a signature miscompare is detected, the captured data from the current interval is uploaded to the shadow buffer, from where it is shifted into the DRAM, together with the number of the current interval as time stamp. The trigger module is optional and it can be used to start capturing trace signals upon given transactions occurring on the bus. This module constantly tries to match the observed signals with a pre-loaded pattern and when a match occurs, a trigger signal is asserted, which triggers the starting point of signature generation. The control logic starts and ends the debug session and provides control signals to other DfD blocks and DRAM.

The following subsections discuss the MISR, the buffers, the trigger module, the control block, as well as the interface with DRAM in detail.

5.1.1 Multiple-Input Signature Register (MISR)

MISR-based lossy compression has been used in [74] to identify failing intervals, as well as in [75] to identify failing clock cycles. However, our approach differs from prior work in that we store previously calculated signatures in the DRAM and compare them with the actual signatures *at run-time*. This has the following advantages: (a) we do not require external hardware for signature comparison and identification of failing intervals; (b) we only require one debug iteration for a set of L_t selected signals, whereas previously proposed methods require multiple iterations and intermediate processing steps. Therefore, our method is more efficient and it can be applied during functional test on ATE to observe a set of any L_t internal signals while introducing little overhead due to data upload and download. The data acquired in this manner can be processed offline for the identification of systemic defects.

In order to ensure continuous generation of signatures at-speed, we employ two alternating MISRs. While one MISR is generating a signature for the current interval, the other MISR is on halt, holding the signature from the previous interval. This signature is compared with the expected signature, followed by a reset of the halted MISR prior to the start of the next interval, where the MISRs change their roles.

The expected MISR signatures are pre-calculated by simulating the behavioral model of the circuit. This calculation requires that the observed nodes always produce the same trace data. This is the case if the internal memory elements of the circuit are initialized and deterministic input data is applied. However, some designs do not provision for the initialization of all memory elements, hence, the observed trace data may exhibit some level of non-determinism. To cope with that, we propose two solutions. In the first solution, we identify intervals containing non-deterministic data (“unknown” or “X”) using simulation and choose a random expected signature (e.g, all “0s”) for these intervals. During debug, it is likely that the actual signature for these intervals will be different from the randomly chosen ones and the trace data will be stored. The probability of aliasing is in the order of $1/2^{K_\mathrm{t}}$, where K_t is the length of the MISR. This probability is negligibly low for sufficiently large values of K_t, for instance, $K_\mathrm{t} = 32$. The captured intervals of trace data containing non-deterministic bits can be used for statistical analysis or they can be ignored. An alternative solution is to provide a masking bit for each interval to the control circuitry in order to skip unimportant intervals. This approach introduces some extra hardware overhead but reduces the amount of data to store. Both techniques can also be combined in case a few intervals containing non-deterministic bits need to be observed and the rest is of no importance.

5.1.2 Trigger Module

The objective of the trigger module is to keep the tracing process on hold until the observed bus reaches a certain state. This approach has several benefits. First, in a functional test of a complex system, it may take a large and non-deterministic number of clock cycles until the module under test reaches a deterministic state. Without this technique, the debug equipment needs to set the starting point for tracing, which is difficult in practice because of non-determinism.

Another benefit is that the trigger module can filter out specific events during the debug session and record erroneous behaviour during those events, while ignoring the rest of the session. For instance, a common scenario in debug is to observe a data bus and verify that all data packages transferred through the bus match the expected values. Using the proposed trigger mechanism, the debug module can filter out these

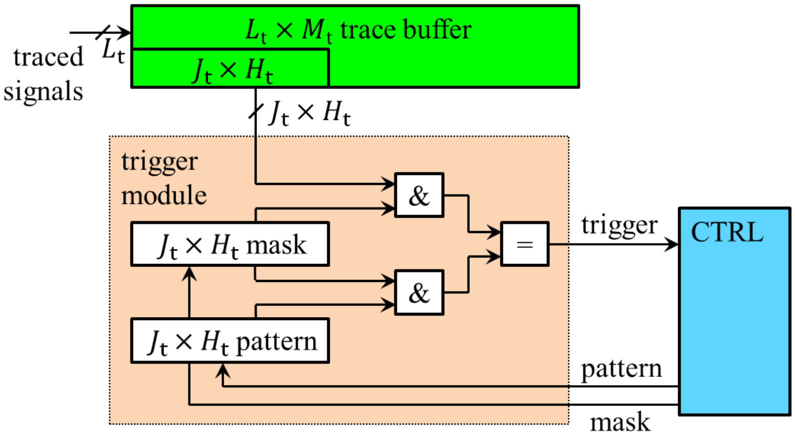


FIGURE 5.3: Architectural view of the trigger module.

data packages, e.g., by snooping on the data-valid signal, generate signatures of these data, and match them with the golden signatures. Since the test defines the content of the data packages and this information is easily available from a high-level description of the test, no computationally expensive, clock-accurate simulations are required to generate MISR signatures for a given test.

Figure 5.3 shows an architectural view of the trigger module. In order to snoop on the traced signals, the trigger module reads a subset of the trace buffer. This subset is a signal array of size $J_t \times H_t$, where J_t is the number of the observed signals that should be used for triggering ($J_t < L_t$), and H_t is the maximum length of the matching pattern in cycles ($H_t < M_t$). This signal array effectively holds the "history" of the bus of the last H_t cycles. In a simple implementation, when only rise or fall conditions of a single signal are used for triggering, these parameters can be set to $J_t = 1$ and $H_t = 2$, which will hold the state of two consecutive cycles. If this array matches "10", it indicates that a rising edge occurred in the last clock cycle. A pattern "01" corresponds to a falling edge. In a more complex implementation, J_t can be larger than 1 in order to match a combination of several signals, as well as H_t can be larger than 2 for matching transitions that span over multiple clock cycles.

The $J_{\mathrm{t}} \times H_{\mathrm{t}}$ signal array is compared to a given pattern, which is held in a register of the same size. For the case if certain bits of the observed signals need to be masked when generating the trigger signal, another register is used that holds the mask. A bit holding 1 in this mask indicates that the corresponding bit in the observed signal array is matched with the pattern; otherwise, the corresponding bit in the signal array is ignored, i.e., it will match any value of the corresponding bit in the pattern. Both the observed signal array and the pattern are passed through bitwise AND gates, such that don't-care bits are set to 0, while care bits are preserved. The outputs of the AND gates are compared using a comparator. If all care bits of the observed signal array and the pattern match, the trigger signal is asserted and sent to CTRL, indicated that a transition of interest occurred and that the DfD circuitry should start the actual tracing process.

Figure 5.4 shows a configuration example for matching transitions occurring on a 4-bit bus. In this example, the trigger needs to be asserted after a transition 10 → 01 occurs on the first two bits of the bus (bus[1:0]), followed by a transition 00 → 11 on the last two bits of the bus (bus[3:2]). These transitions are shown in Figure 5.4 as the content of the 4×3 trace buffer segment, where "x" represents a don't-care value. In order to match this event, the pattern register is loaded with the corresponding values, where don't care bits are filled with random values, for instance, 0s. The mask is loaded such that all care-bit locations are loaded with 1s, and the rest is filled with 0s.

Both the pattern and the mask registers are loaded with pre-defined values by CTRL, which gets these data from DRAM, in a similar fashion as golden MISR signatures are downloaded from DRAM for comparison. Both registers are loaded at the time when the test starts. In a more complex implementation, the registers can be dynamically updated during a debug session in order to match different transitions.

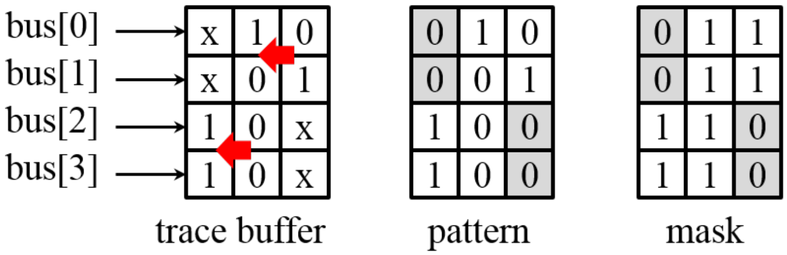


FIGURE 5.4: Setup example for a transition recognition.

5.1.3 Trace Buffer and Shadow Buffer

The goal of the proposed architecture is to identify intervals containing erroneous trace data and store them in DRAM. As the DRAM may operate at a different (lower) frequency compared to the data-sampling frequency, we use an $L_\mathrm{t} \times M_\mathrm{t}$ trace buffer to temporarily store the trace data.

The sampled trace data is shifted into the trace buffer. Every M_t cycles, before a new interval begins, the content of the trace buffer is copied into the shadow buffer, which has the same capacity as the trace buffer. The purpose of the shadow buffer is to hold an interval of trace data I_i until it is transferred to the DRAM, while the next interval I_{i+1} is sampled into the trace buffer, such that no trace data is lost. In addition, the shadow buffer serves as an adapter to the memory interface, which can operate at a frequency different from the sampling frequency and have a data bit-width W_mem different from L_t. In Fig. 5.2, we assume that W_mem is a multiple of L_t, i.e., $W_\mathrm{mem} = k_\mathrm{t} L_\mathrm{t}$ such that we can partition the shadow register into k_t segments that simultaneously shift their content into the DRAM. For instance, if the write data channel is 128 bits wide and the number of observed signals is 32, the shadow register is split into four segments, each of which feeds into the 128-bit-wide data channel. This allows for sampling frequencies much higher than the memory bus operating frequency (up to 4x).

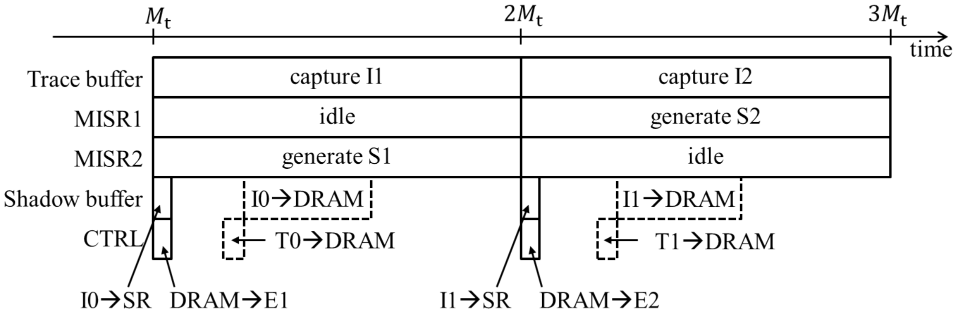


FIGURE 5.5: Operations of DfD modules during signal tracing.

5.1.4 Control Logic

The control logic block (CTRL) provides control signals to the other DfD blocks and schedules their operations as shown in Fig. 5.5. The observation window is divided into intervals of size M_{t} each. For each interval I_i, the active MISR calculates a corresponding signature S_i. Every M_{t} cycles, CTRL sends a signal to the trace buffer to upload I_i into the shadow register (SR). Before the capturing of I_i is complete, the expected signature E_i is loaded from the DRAM into an extra register in CTRL. At the time when the capturing of I_i is complete, i.e., at cycle $n_{\mathrm{s}}M_{\mathrm{t}}$, $n_{\mathrm{s}} = 1, 2, ...$, the MISRs switch roles, and the generated signature is compared with a golden signature. After that, the inactive MISR is reset, while the other MISR is generating a signature for the next interval.

If a miscompare of signatures occurs, CTRL first writes the current time stamp T_i into DRAM for interval identification. The word T_i is simply the value of an interval counter that is incremented every M cycles. Next, CTRL initiates shifting of the erroneous interval from SR into the DRAM. This operation must be completed before the next DRAM reading operation, i.e., loading the expected signature. This ensures that DRAM write and read operations do not overlap. CTRL also generates the address signals for the DRAM in order to (a) access the expected signatures for each interval, and (b) store failed intervals. This is implemented as two counters that are multiplexed into the address lines of the DRAM. These counters are initialized depending on where the memory block containing the expected signatures and the block containing trace data should be placed in the memory space.

In order to schedule the operations as described above, CTRL employs a finite-state machine (FSM) that generates control signals for other blocks in the DfD module. Fig. 5.6 shows the state diagram of this FSM. After deasserting the reset signal rst, the FSM moves into State “read signature” in order to fetch the expected signature for the next interval. If a trigger module is employed, the FSM remains in State “reset” until both the signal rst is deasserted and the trigger is issued. In this case, the DfD module will start capturing data only after a desired transition was observed. In addition, the shadow register is updated and signatures are compared in this state. In case of a miscompare, the FMS moves to “write time” and then to “write dada” in order to store the time stamp and the trace data, respectively. After that, the FSM waits in State “idle” until a new interval starts, which is determined by a cycle counter cycle_cnt. This procedure repeats until the interval counter has reached a specified value or until rst is asserted. At the end of the debug session, the counter pointing to the current address of the trace data in DRAM can be read by the external equipment in order to know what part of DRAM contains the trace data, which also implies the number of failed intervals.

Fig. 5.7 shows an example of a timing diagram for control signals generated by CTRL for another DfD blocks. Trace data is sampled by the MISRs and the trace buffer on positive edges of clk. MISR_sel alternates every interval and selects between the MISRs, such that MISR1 is active during even intervals, and MISR2 is active during odd intervals. MISR1_rst and MISR2_rst reset the corresponding MISRs at the end of their inactive interval. After the last value of an interval is sampled, SR_update is asserted on the negative edge of clk in order to update the content

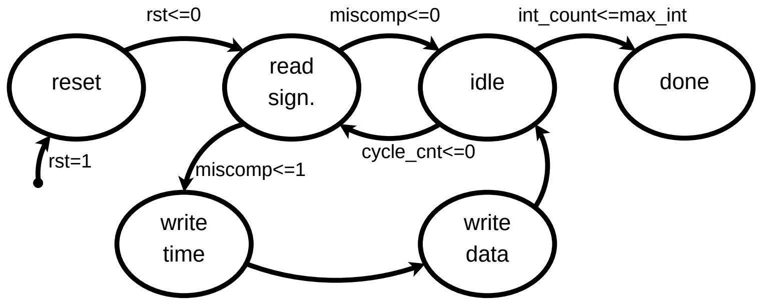


FIGURE 5.6: State diagram for the control block CTRL.

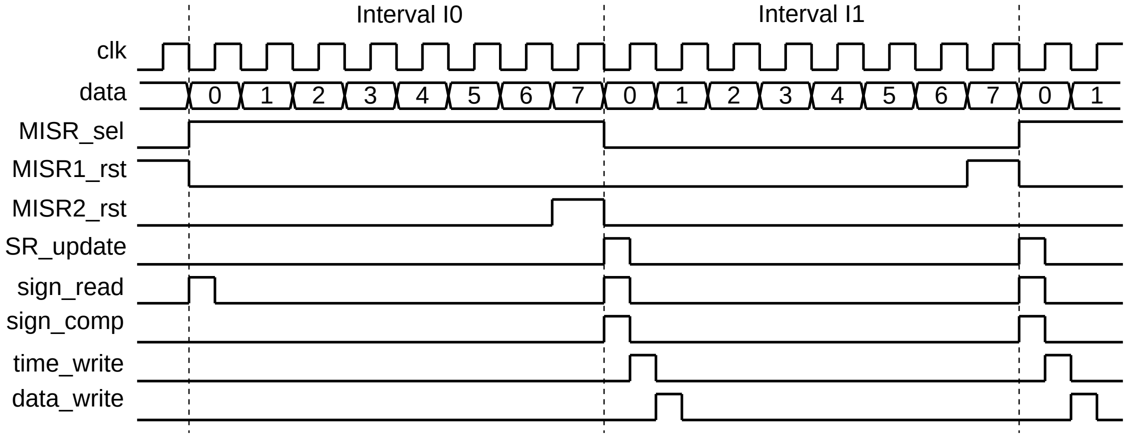


FIGURE 5.7: Timing diagram for control signals.

of the shadow register from the trace buffer. At the beginning of each interval, sign_read is asserted for reading the expected signature for the current interval from the DRAM. The signal sign_comp initiates the comparison of the generated signature with the expected one; therefore it is asserted. If a signature miscompare is detected, time_write and data_write are asserted after each other to initiate the writing in the DRAM of the timestamp and the content of the shadow register, respectively.

5.1.5 Interface with DRAM: Challenges, Limitations, and Solutions

Challenges associated with the proposed method are related to the requirement to communicate with DRAM during functional test. First, the test program may also be using DRAM, limiting the storage available for trace data. In this case, we assume that the DRAM can be partitioned such that one partition is dedicated for trace-data storage. Second, enough bandwidth must be available to shift the trace data in the DRAM at-speed. This is a reasonable assumption for functional tests that resemble real applications, as they are meant to be run in parallel with other applications, hence not consuming all available memory resources. Third, DRAM access operations from the DfD module need to be scheduled such that they do not interfere with the test program. Therefore, the interface with the DRAM requires special attention.

The data flow to and from the DRAM is typically managed by a memory controller. A realistic assumption is that processor cores and other embedded cores that are on the same die, have shared access to the memory controller through a memory bus. An example of such bus is AMBA AXI that has a wide range of applications [122]. In this scenario, we can treat the DfD module as a core and connect it to the bus in order to get access to the DRAM. The CTRL block needs to be enhanced with an interface to the bus, and this interface clearly depends on the bus architecture. The available bandwidth through this interface will be a limiting factor for the number of observed signals. However, emerging standards for wide-I/O DRAM promise high bandwidths, even a fraction of which is sufficient for the proposed method. For instance, 10% of the JEDEC Wide I/O 2 maximum bandwidth (51.2 GB/s) is sufficient to trace 32 signals at 1 GHz frequency (4 GB/s).

The proposed approach can be implemented using the infrastructure described in [76]. Note that [76] describes only a generic debug architecture and it does not consider selective storing of trace data using MISR signatures.

Another challenge related to the interface with the DRAM is potentially long memory-access latencies, especially if multiple modules connected to the bus request memory access at the same time. For instance, if the read-access latency exceeds one interval (M_t clock cycles), the expected signature will not be available for the comparison. Write-access latency is less critical, as memory controllers typically employ buffers for data to be written in the DRAM. However, certain test programs may cause high data-traffic peaks, in which case the DfD module will not be able to store a faulty interval within M_t cycles. To address the latency problem, we can select a sufficiently large M_t based on the memory-interface specification. As reported in [123], the average access latency of 3D-stacked DRAM using TSVs is 25 ns. Including the overhead due to the data bus and the memory controller, the latencies for loading expected signatures and storing trace data is considerably lower than 64 clock cycles at 1 GHz frequency; therefore we select the minimum depth of the trace buffer $M_\mathrm{t,min} = 64$ in this work. However, $M_\mathrm{t,min}$ needs to be adjusted depending on the specification of the memory controller and the memory workload of the test program. Test programs causing excessive memory access operations may increase the maximum memory latency for the DfD module; therefore M_t will need to be increased accordingly.

5.1.6 Analysis of Compression Effectiveness

By storing only erroneous intervals, we effectively achieve compression of trace data. We can estimate the compression ratio based on the frequency of errors in the stream of trace data, using the total amount of observed trace data as a baseline. The data that we store includes the actual trace data, the expected signatures, and the time stamps.

First, recall that we denote the length of the observation window as N_t (in clock cycles). The total amount of bits observed is therefore $L_\mathrm{t}N_\mathrm{t}$, which we use as a baseline. The total number of intervals can be expressed as $N_\mathrm{t}/M_\mathrm{t}$. Each interval signature contains K_t bits, hence the amount of bits required to store all signatures is $\frac{N_\mathrm{t}K_\mathrm{t}}{M_\mathrm{t}}$. Under the commonly made assumption that errors in the data stream are independent and erroneous words on the observed bus occur with the probability p, we can calculate the probability that an interval of length M_t bits contains at least one erroneous word as

$$P = 1 - (1 - p)^{M_\mathrm{t}}.$$

We define a random variable X_e as the number of erroneous intervals in the observed data stream. The expectation of X_e can be expressed as

$$E[X_\mathrm{e}] = P\frac{N_\mathrm{t}}{M_\mathrm{t}},$$

which we use to calculate the expected number of bits to be stored as

$$E[X_\mathrm{e}](S_\mathrm{t} + L_\mathrm{t}M_\mathrm{t}) = \frac{PN_\mathrm{t}}{M_\mathrm{t}}(S_\mathrm{t} + L_\mathrm{t}M_\mathrm{t}),$$

where S_t is the size of the time stamp that is added to each interval for identification. Now we can calculate the compression ratio using the total number of bits in the stream as a baseline:

$$r = \frac{\text{number of bits in stream}}{\text{stored data}} = \frac{L_\mathrm{t}N_\mathrm{t}}{\frac{PN_\mathrm{t}}{M_\mathrm{t}}(S_\mathrm{t} + L_\mathrm{t}M_\mathrm{t}) + \frac{K_\mathrm{t}N_\mathrm{t}}{M_\mathrm{t}}} \tag{5.1}$$

$$= \frac{L_\mathrm{t}M_\mathrm{t}}{(1 - (1 - p)^{M_\mathrm{t}})(S_\mathrm{t} + L_\mathrm{t}M_\mathrm{t}) + K_\mathrm{t}} \tag{5.2}$$

Fig. 5.8 shows the compression ratio r as function of error probability p for the case $L_\mathrm{t} = K_\mathrm{t} = 32$, $S_\mathrm{t} = 32$, and different values of M_t. For practical reasons, all implementation-related input parameters are powers of two. For low error probabilities, only a few intervals are erroneous and need to be stored, hence the stored data is dominated by the expected signatures, allowing for high compression. In the case of a high error probability, nearly all intervals are erroneous and need to be stored, in addition to the expected signatures, hence no compression is achieved. This observation is consistent with Fig. 5.8, as $M_\mathrm{t} = 128$ provides the highest compression for small values of p, and $M_\mathrm{t} = 1024$ provides the highest compression for large values of p.

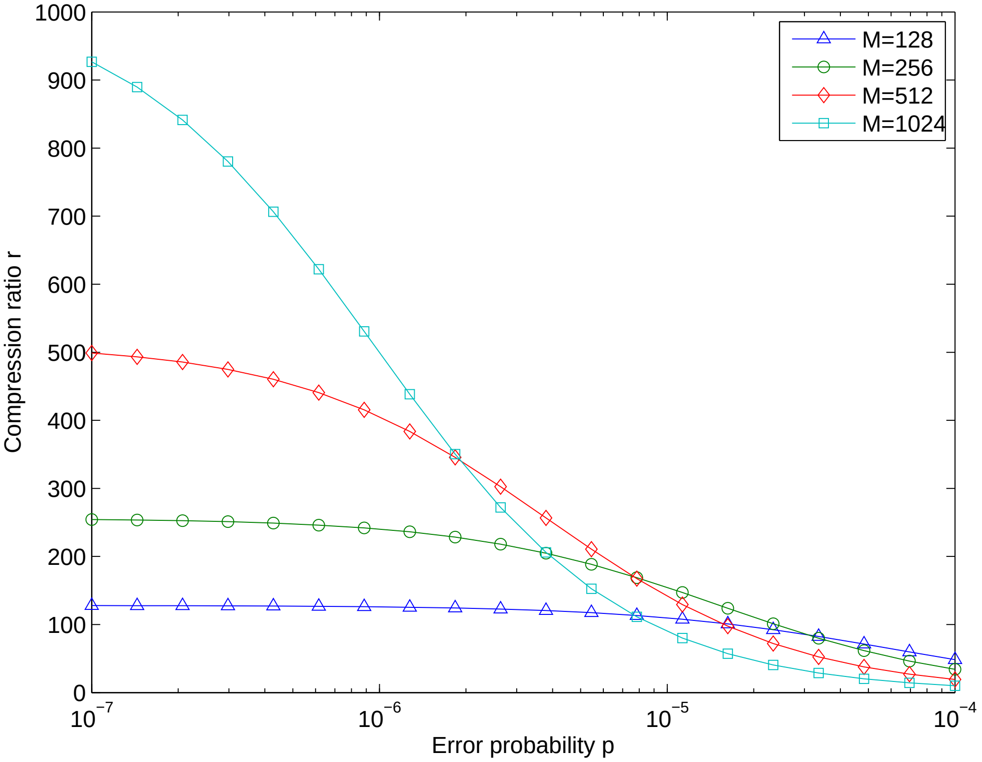


FIGURE 5.8: Compression ratio r as function of error probability p for $L_\mathrm{t} = K_\mathrm{t} = 32$, $S_\mathrm{t} = 32$, and different values of M_t in the case of a uniform distribution.

Similarly, we can obtain the compression ratio for the model using burst errors of a fixed size B. We assume that the depth of the trace buffers is larger than B, hence an error burst either occurs within one interval of size M_t or overlaps two intervals. In worst case, each burst overlaps two intervals and any two neighboring burst are sufficiently far apart such that no two bursts occur within the same interval. Therefore, the maximum number of erroneous intervals is twice the number of bursts, the expectation of which is $\lceil pN_\mathrm{t}/B \rceil$. As we store the trace data for each interval including the time stamp of size S_t, the amount of data that needs to be downloaded from DRAM is $2\lceil pN_\mathrm{t}/B \rceil \times (L_\mathrm{t}M_\mathrm{t} + S)$. In addition, expected MISR signatures of total size KN/M need to be uploaded prior to the debug session. Therefore, the

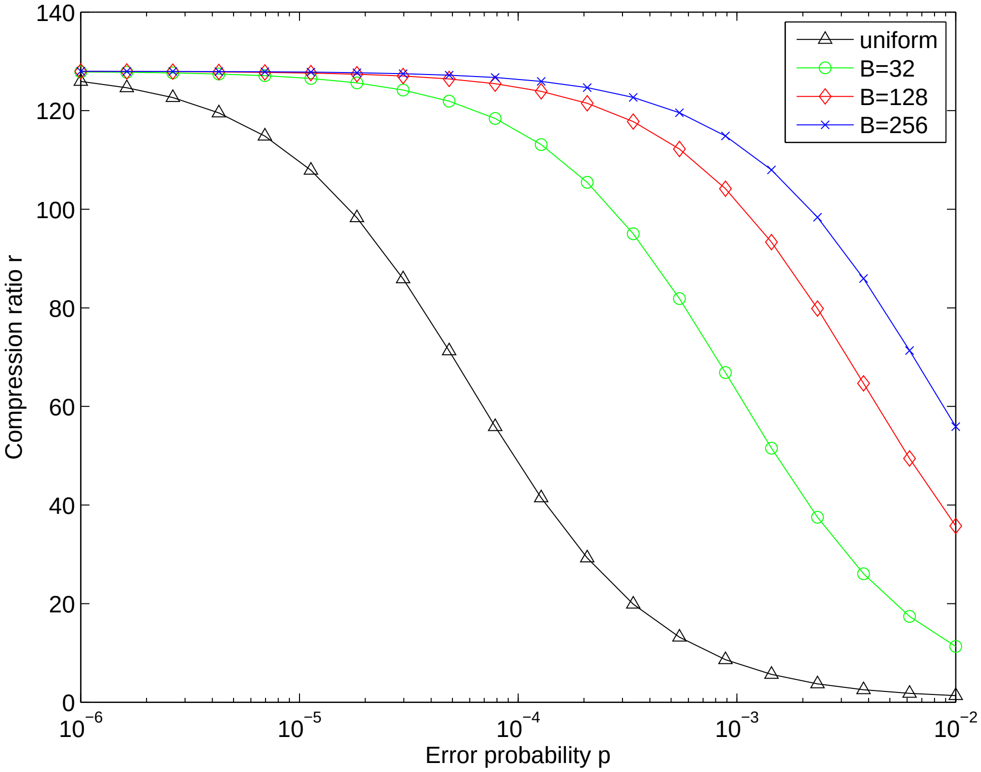


FIGURE 5.9: Compression ratio r as function of error probability p for $L_t = K_t = 32$, $S_t = 32$, and $M_t = 128$ in the case of error bursts of different lengths.

resulting worst-case compression ratio can be calculated as

$$r = \frac{\text{number of bits in stream}}{\text{stored data}} \tag{5.3}$$

$$= \frac{L_t N_t}{2\left\lceil \frac{pN_t}{B} \right\rceil (L_t M_t + S_t) + \frac{K_t N_t}{M_t}} \tag{5.4}$$

Fig. 5.9 shows the compression ratio in the case of error bursts, where "uniform" corresponds to uniformly distributed errors. As erroneous words are clustered in this case, we can achieve high compression even for relatively large values of p.

Given a given p, large buffers are more likely to capture at least one erroneous bit and hence more trace data need to be stored compared to smaller buffers. However, small buffers result in more intervals and hence more expected signatures that need to be uploaded to DRAM. Therefore, we can optimize the depth of the trace buffers

M_t for better compression depending on the assumed error probability p.

For sufficiently low error probabilities, i.e., $p \ll 1$, $(1-p)^{M_\mathrm{t}} \approx (1 - pM_\mathrm{t})$, hence we can simplify Equation (5.2) to

$$r \approx \frac{M_\mathrm{t}}{pM_\mathrm{t}(S_\mathrm{t} + L_\mathrm{t}M_\mathrm{t}) + K_\mathrm{t}}.$$

By setting the derivative $\partial r/\partial M_\mathrm{t}$ to zero, we can find that $M_\mathrm{t} = \sqrt{\frac{K_\mathrm{t}}{pL_\mathrm{t}}}$ provides maximum compression for the case when other parameters are given. However, M_t needs to satisfy the following constraints for practical reasons:

- M_t should be a power of two in order to simplify the implementation.
- M_t should have an upper limit due to silicon-area overhead. A reasonable number for this limit in case of $L_\mathrm{t} = 32$ is $M_\mathrm{t,max} = 512$, such that the combined memory size of the trace buffer and shadow register is 4 KB, which is used as the maximum buffer size in [75].
- M_t should have a lower limit; otherwise the interval length will be shorter than the memory latency and the DfD module will not be able to fetch expected responses and store trace data on time. As explained above, we use $M_\mathrm{t,min} = 64$ in this work. However, this number needs to be adjusted depending on the specification of the memory controller and the memory workload of the test program.

Due to these constraints, optimal values of M_t can be represented as a descending staircase function of p as shown in Fig. 5.10, where $M_\mathrm{t,u}$ is the unconstrained diagram for optimal M_t. The jump points that can be calculated by equating $r(M_\mathrm{t}) = r(2M_\mathrm{t})$. Solving this equation for p provides the jump points $p = \frac{K_\mathrm{t}}{2L_\mathrm{t}M_\mathrm{t}^2}$ where $M_\mathrm{t} = \{64, 128, 256\}$. The observation that optimal values of M_t are decreasing

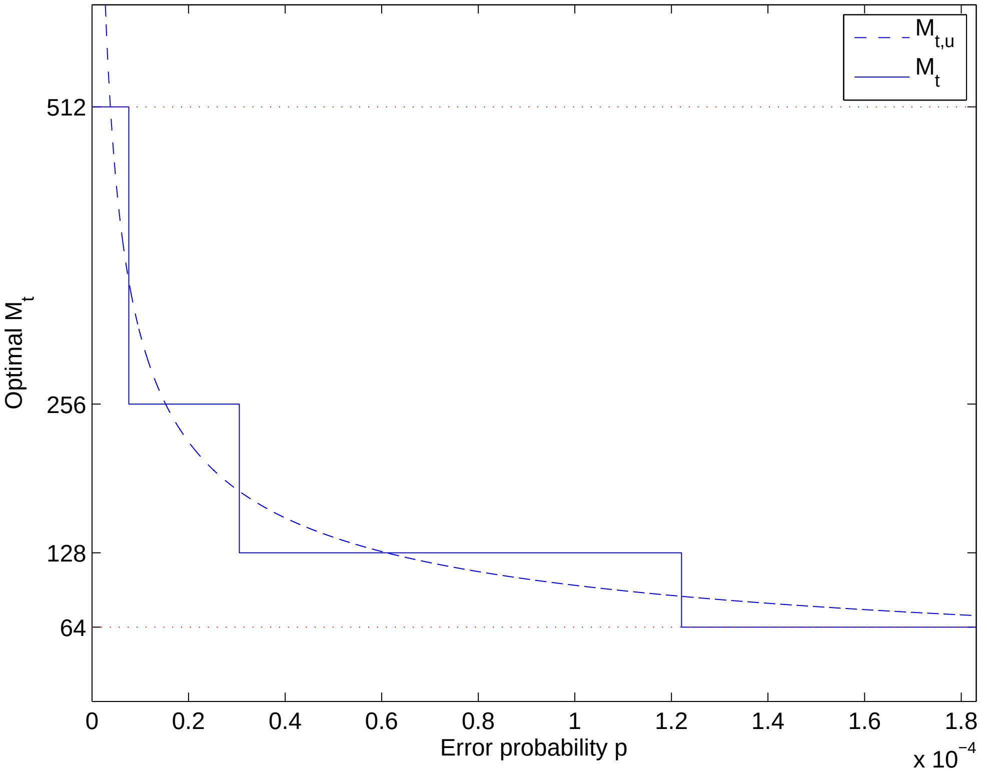


FIGURE 5.10: Optimal trace depth M_t as a function of error probability p for the case $L_\mathrm{t} = K_\mathrm{t} = 32$.

with the increasing values of p, is consistent with Fig. 5.8. Using this diagram, we can select the depth of the trace buffers in order to maximize data compression, reducing the memory requirements and the communication time with the ATE.

5.2 Simulation Results

5.2.1 DfD Implementation and Simulation

We created a Verilog RTL model of the proposed architecture shown in Fig. 5.2 and synthesized it using a 45 nm CMOS library [95]. For the case $L_\mathrm{t} = K_\mathrm{t} = 32$, $S_\mathrm{t} = 32$, $J_\mathrm{t} = 8$, $H_\mathrm{t} = 4$, and $M_\mathrm{t} = 128$, the synthesized circuit contained 8357 sequential and 8783 combinational gates. The standard-cell area of the design was 0.062 mm^2, which is negligible for realistic chips. For instance, on a 25 mm^2 die, the DfD circuitry would only occupy 0.25% of the die area.

In the first simulation, we verified the functionality of the design through simulation by using randomly generated data as an input stream. First, we calculated the expected signatures by applying the input stream to the MISR and recorded its value every M_t cycles. In the second run, we simulated the full circuitry, where the DRAM was modeled as a Verilog module and initialized with the pre-calculated signatures. As expected, all intervals passed the comparison and no trace data was stored into the DRAM. In the third run, we injected errors by altering some of the bits in the input stream. As expected, the circuitry was able to identify erroneous intervals and store them into the DRAM.

Next, we experimented with a Verilog model of a JPEG encoder [84]. We instantiated an instance of the encoder and an instance of the DfD module with the parameters $L_\mathrm{t} = K_\mathrm{t} = 32$, $S_\mathrm{t} = 32$, and $M_\mathrm{t} = 64$, and connected the decoder to an internal data bus in the encoder. First, we simulated the encoding of a picture in JPEG format (11,432 clock cycles) using the fault-free model, and recorded 177 golden MISR signatures, starting from clock cycle 125 in order to avoid uninitialized states of the bus. Then, we injected bugs in the RTL code of a FIFO unit within the encoder module and re-ran the simulation with the enabled DfD module. Depending on the location of the bug, we could observe a different behavior of the circuit. In some cases, a bug caused errors in all 177 intervals, resulting in storage of the all traced data. In practice, however, this kind of bugs is relatively simple to localize, as the errors already start to occur within the first few hundred clock cycles. In other cases, a bug manifested itself after 5,127 error-free clock cycles, i.e., after 78 error-free intervals, corrupting all following intervals. As expected, error-free intervals were excluded from storage in the DRAM. Therefore, only 56% of the intervals had to be stored for analysis. This type of a bug is difficult to localize using conventional tracing methods, as many iterations are required to identify the time window when errors start to occur. The proposed method allows for rapid error localization using only one iteration.

Finally, we simulated a Verilog model of a hyper-pipelined microprocessor OpenRISC 1200 HP [84]. As in the previous simulation, an instance of the DfD module with the parameters $L_\mathrm{t} = K_\mathrm{t} = 32$, $S_\mathrm{t} = 32$, and $M_\mathrm{t} = 64$ was connected to an address bus within the CPU. We executed a random instruction sequence over 20,000 clock cycles and recorded 313 golden signatures. Next, we injected bugs in the control flow of the program-counter generation unit. Similar to the previous simulation, some bugs corrupted all intervals. However, a certain bug resulted in only six erroneous intervals that were scattered over the entire observation window of 313 intervals. Therefore, only 2% of the intervals had to be stored in the DRAM for analysis. Another bug in the datapath of the ALU caused several short error bursts, resulting in only four erroneous intervals (1.3%) that occurred at the end of the observation window. In practice, this kind of bug requires an extensive effort for localization, as the observed data is erroneous for only a few clock cycles during the entire observation window. Using the proposed method, we can efficiently identify and store the erroneous intervals in only one iteration.

5.2.2 Comparison with Method Proposed in [75]

We next compared our framework with the method based on selective capture proposed in [75]. The first four columns of Table 5.2 repeat the data published in [75]: the design used, the trace buffer size, the error probability, and the length of the expanded observation window. We use these data to (a) compare the debug-session time, and (b) estimate the required DRAM capacity in order to achieve the same observation window using our framework.

First, we detail the calculation of the estimated debug-session time. This time is the sum of the *on-chip sampling time* and the *communication time*, where the

Table 5.2: Comparison of debug-session time with prior work [75].

Method in [75]						Proposed method				
	Buffer Size (Bytes)	Error Prob[1] $p\times100\%$	Window size (cycles)	Debug time α=2 (1 K cycles)	Debug time α=10 (1 K cycles)	Debug time Uniform, α=2 (1 K cycles)	Debug time Uniform, α=10 (1 K cycles)	Debug time Burst, α=2 (1 K cycles)	Debug time Burst, α=10 (1 K cycles)	DRAM required (KB)
ARM	512	0.016	19456	91.1	222.2	51.8	181.2	47.2	158.3	2.0
	512	0.051	12032	68.9	199.9	49.2	198.0	32.4	113.8	2.3
	512	0.097	8576	58.5	189.6	50.8	219.7	25.5	93.1	2.6
	512	0.513	3072	42.0	173.1	62.7	301.0	14.5	60.0	3.6
	512	1.387	1792	38.1	169.2	73.5	360.2	20.2	94.0	4.4
	4096	0.016	132096	658.4	1707.0	351.7	1230.0	280.8	875.8	13.4
	4096	0.051	61440	446.5	1495.0	251.4	1011.2	139.5	451.8	11.6
	4096	0.097	39936	382.0	1430.5	236.5	1023.0	104.8	364.4	12.0
	4096	0.513	17408	314.4	1362.9	355.0	1705.5	84.7	354.0	20.6
	4096	1.387	10240	292.9	1341.4	419.8	2058.2	95.4	435.8	25.0
NoC	512	0.037	4864.0	47.4	178.4	17.1	66.1	18.0	70.8	0.7
	512	0.072	3392	42.9	174.0	16.8	70.4	15.1	62.0	0.8
	1024	0.037	9088	92.8	354.9	32.0	123.6	26.5	96.1	1.4
	1024	0.072	7552	88.2	350.3	37.4	156.7	23.4	86.9	1.8
	2048	0.037	16384	180.2	704.5	57.7	222.8	41.1	139.9	2.5
	2048	0.072	13568	171.8	696.1	67.2	281.6	35.5	123.0	3.3
	4096	0.037	34816	366.6	1415.2	122.5	473.4	78.0	250.5	5.4
	4096	0.072	24064	334.3	1382.9	119.1	499.4	64.8	227.6	5.8

[1] Referred to as “error rate” in [75]

computation time between iterations is neglected [74]. The method in [75] requires three iterations, hence the on-chip sampling time can be estimated as $3 \times N/f_{CUD}$, where N is the length of the expanded window in cycles and f_{CUD} is the operating frequency of the circuit under test (sampling frequency). After the first iteration, parity data needs to be downloaded from the trace buffer to the external equipment. After the second iteration, generated signatures need to be downloaded and the tag data uploaded into the trace buffer. Finally, the actual trace data is downloaded. In total, the trace buffer needs to be accessed four times; therefore, the communication time can be estimated as $4 \times T \times \alpha/f_{CUD}$, where T is the size of the trace buffer and α is the ratio of f_{CUD} to the frequency of the external equipment, which is usually slower than f_{CUD}. The total debug time expressed in functional clock cycles is therefore

$$3N_{\mathrm{t}} + 4\alpha T.$$

We calculated this time for the method proposed in [75] for two values of α, 2 and

10, as shown in Table 5.2.

For comparison, we show in Table 5.2 the estimated debug time using the method proposed in this work for the same input parameters. We consider two cases: uniform error distribution and burst errors, which is used in [74]. Our method requires only one iteration, hence the sampling time is N_{t}/f_{CUD}. The communication time in case of a *uniform* defect distribution can be estimated using the compression ratio r calculated in Equation (5.2) as $N_{\mathrm{t}}L_{\mathrm{t}}/r \times \alpha/f_{CUD}$, where $N_{\mathrm{t}}L_{\mathrm{t}}$ is the uncompressed observed data in bits. Therefore, the total debug time expressed in functional clock cycles is

$$N_{\mathrm{t}}(1 + \alpha L_{\mathrm{t}}/r).$$

The depth of the trace buffers in our framework was set to $M_{\mathrm{t}} = 64$, which was optimal for most given error probabilities. Note that we only use the trace buffer as a temporary storage before transferring the data to the DRAM, hence the trace buffers can be small even for large observation windows. For *burst errors*, we estimated the communication time as follows. As described in Section 5.1.6, the amount of data that needs to be downloaded from DRAM is $2\lceil pN_{\mathrm{t}}/B\rceil \times (L_{\mathrm{t}}M_{\mathrm{t}} + S_{\mathrm{t}})$, where B is the burst length. In addition, expected MISR signatures of total size $K_{\mathrm{t}}N_{\mathrm{t}}/M_{\mathrm{t}}$ need to be uploaded prior to the debug session. The resulting debug time expressed in functional clock cycles totals

$$N_{\mathrm{t}} + \alpha\left(2\left\lceil\frac{pN_{\mathrm{t}}}{B}\right\rceil(L_{\mathrm{t}}M_{\mathrm{t}} + S_{\mathrm{t}}) + \frac{K_{\mathrm{t}}N_{\mathrm{t}}}{M_{\mathrm{t}}}\right).$$

We calculated these values using $M_{\mathrm{t}} = B_{\mathrm{t}} = 64$ for $\alpha = 2$ and $\alpha = 10$, as shown in Table 5.2. In addition, we estimated the amount of DRAM that must be available in order to achieve the same length of the observation as in prior work [75], shown in the last column of Table 5.2.

The results presented in Table 5.2 show that the debug time of our method is shorter than that of method in [75] for low error probabilities in case of uniform distribution. In case of burst errors, the debug time of the proposed method is even shorter since erroneous data words are condensed, resulting in less erroneous intervals that need to be stored. In order to achieve the same observation windows as in prior work, the proposed method requires only a few KB of available DRAM. In realistic circuits, such as in IC with eDRAM or 3D-ICs with wide-I/O DRAM, the amount of available on-chip-like memory will be larger by several orders of magnitude, hence we can extend the observation window by the same factor, resulting in much more efficient debug sessions.

In addition, we can compare the proposed method with the framework in [75] in terms of robustness. Both methods require deterministic execution of the test program in order to calculate "golden" signatures. However, even if the trace data observed in a fault-free circuit is deterministic, the values observed in a faulty circuit can be non-deterministic. For instance, an electrical bug can cause metastability issues on timing-critical paths, such that the affected flip-flops capture erroneous values only with a probability less than one. In cases of such electrical bugs, the tag data generated during one iteration in method proposed in [75] may not correspond to the actual erroneous cycles in the next iteration. Hence the data obtained by using this method may not represent the "full picture". In our framework, it is guaranteed that all erroneous cycles are captured unless an aliasing occurs, which is unlikely for sufficiently large K. Even if erroneous cycles are different for each test-program run, we can obtain a "full picture" containing all erroneous data for each run independently of previous runs.

5.2.3 Comparison with Method Proposed in [74]

In [74], the authors calculate the ratio of the debug time T_{prop} compared to the debug time using sequential debug sessions T_{seq}. In Table 5.3, we compare the results

Table 5.3: Comparison of debug-session time with prior work [74].

Error Prob[1] p × 100%	T_{seq}/T_{prop} for different burst length X								DRAM (MB)
	$B = 64$		$B = 128$		$B = 256$		$B = 512$		
	[74]	Prop.	[74]	Prop.	[74]	Prop.	[74]	Prop.	
0.78	13.2	191.2	21.9	229.9	32.8	255.8	48.9	271.0	2.0
1.55	7.2	143.6	12.1	191.7	18.6	230.2	26.4	255.8	3.0
2.3	5.0	114.9	8.4	164.2	13.2	209.0	18.4	242.1	4.0
3.08	4.0	96.0	6.6	144.0	10.3	191.9	14.6	230.2	5.0
3.83	3.3	82.6	5.4	128.4	8.4	177.6	12.0	219.8	6.0
4.58	2.8	72.5	4.6	115.8	7.2	165.1	10.3	210.0	7.0

[1] Referred to as “error rate” in [74]

obtaining using the proposed method with those presented in [74]. We calculate T_{seq}/T_{prop} for the proposed method as follows:

$$\frac{T_{seq}}{T_{prop}} = \frac{(1 + N_\mathrm{t}/M_\mathrm{t}) \times N_\mathrm{t}/2 + \alpha N_\mathrm{t} L_\mathrm{t}}{N_\mathrm{t} + \alpha \left(2 \left\lceil \frac{pN_\mathrm{t}}{B} \right\rceil (L_\mathrm{t} M_\mathrm{t} + S_\mathrm{t}) + \frac{K_\mathrm{t} N_\mathrm{t}}{M_\mathrm{t}} \right)}$$

The results show that the debug time using the framework proposed in this work is 5–20 times faster compared to that in [74] by using less than 7 MB of DRAM. Note that this memory is part of functional DRAM, hence it does not introduce any extra overhead on the chip.

5.3 Conclusion

We propose a framework for massive signal tracing in ICs that integrate fast DRAM, such as 2D-ICs with embedded DRAM or 3D-stacked ICs with wide-I/O DRAM dies. By using a relatively large on-chip DRAM for trace-data storage, we can increase the observation window by orders of magnitude compared to traditional methods that use trace buffers. Erroneous intervals are selectively captured into the DRAM by generating MISR signatures of the trace data and comparing them with the expected signatures that are pre-calculated offline and uploaded to the DRAM before a debug session. In addition, a triggering mechanism is employed to start capturing trace data only after a defined transition occurs on the observed signal bus. This technique can eliminate issues related to non-determinism in tests in realistic systems by focusing on deterministic and hence predictable segments of a test. As a side effect, this technique can also simplify the generation of golden signatures. The proposed method allows for capturing erroneous intervals in one iteration without intermediate processing steps. This allows for signal tracing on ATE during execution of functional tests in volume manufacturing, providing extra information for diagnosis of systemic defects. The area overhead of the proposed design-for-debug circuitry is comparable with that of the existing methods and is negligible for realistic designs.

6

Conclusions

This dissertation has covered several research topics related to testing, design-for-test, and debug for 3D-stacked ICs. These topics include test-content generation, test access, and diagnosis and debug.

Chapter 2 presented a solution for TSV-stress aware test-pattern generation. Conventional circuit models used for ATPG disregard the effects of TSV-induced stress on the timing profile of the circuit, which is a crucial information for generating tests targeting small-delay defects. Using TSV-stress oblivious circuit models for these tests results in degraded test quality. We verified this by evaluating conventional, non-stress aware test patters on stress-aware circuit models. The results show that neglecting TSV stress leads to a higher test escape rate compared to that obtained using TSV-stress aware models. Employing the proposed TSV-stress aware flow reduces the test escape rate to the expected levels, without a noticeable impact on the test-pattern count.

Chapter 3 proposed a method for pre-bond TSV test. Detecting defects in TSVs at the pre-bond stage is challenging due to their limited. The proposed solution employs ring oscillators and duty-cycle detectors to target these defects, which can be modeled as resistive-open faults and leakage faults. The TSVs under test are used as capacitive loads of the ring oscillators. Defects in TSVs lead to variations of TSV capacitance and resistance and therefore affect the oscillation period and the duty cycle of the produced oscillating signal. By measuring the oscillation period and the duty cycle, the proposed on-chip design-for-test circuitry is able to detect these faults. The measured values serve as inputs to a regression model created based on artificial neural networks. This model distinguishes between resistive-open and leakage faults and estimates their size. In order to increase diagnosis accuracy, the oscillation period and the duty cycle are measured at multiple voltage levels. Simulation results showed that this method is able to accurately diagnose TSV faults even in the presence of process variations.

Chapter 4 presented a framework for robust optimization of test-access architectures for 3D ICs. In contrast to traditional, non-robust optimization methods, the proposed approach takes uncertainties in input parameter values into account. Examples of such parameters include maximum allowed test power and the bit-width of the die test-access mechanism. Robust solutions are optimized for a range of uncertain input parameter values. Simulation results showed that non-robust solutions may be far away from point-optimal solutions when the input parameter values change, whereas robust solutions remain close to point-optimal solutions for a variety of scenarios.

In Chapter 5, we proposed a method for on-chip signal tracing for 3D-ICs that integrate large amounts of DRAM. The proposed debug architecture detects erroneous behavior of the observed signals using pre-calculated signatures and stores the signal values into functional DRAM, from which it can be transferred to external debug equipment. Compared to traditional signal tracing methods that use dedicated on-chip trace buffers, the proposed method allows for debug sessions that are orders of magnitude longer, significantly simplifying the debug effort.

6.1 Future Research Directions

3D IC testing offers several opportunities for research in order to improve current solutions and bring 3D IC closer to volume manufacturing. One such topic is TSV test. This dissertation mainly focused on pre-bond TSV test. However, testing TSV-based connections after bonding is required to ensure that no new defects were introduced during the bonding process, for instance, misaligned micro-bumps. Further research can be conducted to extend the proposed solution for pre-bond TSV test and create a unified solution for both pre-bond and post-bond TSV test.

The realm of robust optimization of test-access architectures for 3D ICs can be further explored in future research. This dissertation provided an initial framework for robust optimization, which targeted test-time optimization at die level, i.e., under the assumption that each die in the stack is tested in isolation from the other dies. This framework can be extended to cover possible scenarios, in which modules from different dies in the stack are tested in parallel. Another improvement of the proposed method could be considering the temperature constraints during test. The power dissipated during test may lead to overheating problems. The framework proposed in this dissertation used a limit on maximum test power to prevent issues related to both voltage droop and overheating. However, this model can be improved to account for dynamic behavior of heat dissipation in a 3D stack, which depends on many factors, including the property of the packaging, physical location of modules under test within the stack, and the duration of the test. A possible direction for future research is to incorporate such models in the proposed framework.

Bibliography

[1] P. Garrou, C. Bower, and P. Ramm, Eds., *Handbook of 3D Integration – Technology and Applications of 3D Integrated Circuits*. Weinheim, Germany: Wiley-VCH, Aug. 2008.

[2] E. J. Marinissen, B. De Wachter, K. Smith, J. Kiesewetter, M. Taouil, and S. Hamdioui, "Direct Probing on Large-Array Fine-Pitch Micro-Bumps of a Wide-I/O Logic-Memory Interface," in *Proceedings IEEE International Test Conference (ITC)*, Oct. 2014, pp. 1–10.

[3] G. V. der Plas *et al.*, "Design Issues and Considerations for Low-Cost 3-D TSV IC Technology," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 1, pp. 293–307, January 2011.

[4] H.-H. S. Lee and K. Chakrabarty, "Test Challenges for 3D Integrated Circuits," *IEEE Design & Test of Computers*, vol. 26, no. 5, pp. 26–35, 2009.

[5] E. J. Marinissen and Y. Zorian, "Testing 3D Chips Containing Through-Silicon Vias," in *Proceedings IEEE International Test Conference (ITC)*, Nov. 2009, pp. 1–11.

[6] K. Chakrabarty, S. Deutsch, H. Thapliyal, and F. Ye, "TSV Defects and TSV-induced Circuit Failures: The Third dimension in Test and Design-for-Test," in *Reliability Physics Symposium (IRPS), 2012 IEEE International*, 2012, pp. 5F–1.

[7] M. Abramovici, P. Bradley, K. Dwarakanath, P. Levin, G. Memmi, and D. Miller, "A Reconfigurable Design-for-Debug Infrastructure for SoCs," in *Proceedings ACM/IEEE Design Automation Conference (DAC)*, 2006, pp. 7–12.

[8] S. Mitra, S. A. Seshia, and N. Nicolici, "Post-Silicon Validation Opportunities, Challenges and Recent Advances," in *Proceedings ACM/IEEE Design Automation Conference (DAC)*, 2010, pp. 12–17.

[9] W. R. Davis *et al.*, "Demystifying 3D ICs: the Pros and Cons of Going Vertical," *IEEE Design & Test of Computers*, vol. 22, no. 6, pp. 498–510, 2005.

[10] A. Mercha *et al.*, "Impact of thinning and through silicon via proximity on high-k/metal gate first CMOS performance," in *VLSI Technology (VLSIT), 2010 Symposium on*, 2010, pp. 109–110.

[11] J. Van Olmen *et al.*, "3D Stacked IC Demonstration Using a Through Silicon Via First Approach," in *Electron Devices Meeting, 2008. IEDM 2008. IEEE International*, 2008, pp. 1–4.

[12] H. Jones, *Technical Viability of Stacked Silicon Interconnect Technology.* Xilinx, Oct. 2010, white Paper, see http://www.xilinx.com/publications/technology/stacked-silicon-interconnect-technology-ibs-research.pdf.

[13] K. Saban, *Xilinx Stacked Silicon Interconnect Technology Delivers Breakthrough FPGA Capacity, Bandwidth, and Power Efficiency.* Xilinx, Oct. 2010, white Paper, see http://www.xilinx.com/support/documentation/white_papers/wp380 _Stacked_Silicon_Interconnect_Technology.pdf.

[14] S. Deutsch *et al.*, "DfT architecture and ATPG for Interconnect tests of JEDEC Wide-I/O memory-on-logic die stacks," in *Proceedings IEEE International Test Conference (ITC)*, 2012, pp. 1–10.

[15] R. Goering, "Three Die Stack – A Big Step "Up" for 3D-ICs with TSVs," in *Cadence Community Blogs*, Dec. 2011, (See http://www.cadence.com/Community/blogs/ii/archive/2011/12/13/ three-die-stack-a-big-step-up-for-3d-ics-with-tsvs.aspx).

[16] "ST-Ericsson and CEA-Leti's WIOMING Prototype Shows How To Combine Wide IO Memory and Logic SoC for Future 3D Multi-Processor Architectures," *Yole Développement 3D Packaging Newsletter*, no. 22, pp. 16–18, Feb. 2012, http://www.i-micronews.com/upload%5Cnewsletter%5C3DPackaging_Feb2012_ iMN.pdf.

[17] J. S. S. T. Association, *Wide IO DRAM.* JEDEC, Dec. 2011.

[18] E. J. Marinissen and Y. Zorian, "Testing 3D Chips Containing Through-Silicon Vias," in *Proceedings IEEE International Test Conference (ITC)*, Nov. 2009, paper ET1.1.

[19] E. Marinissen, C. Chi, J. Verbree, and M. Konijnenburg, “3D DfT Architecture for Pre-bond and Post-bond Testing,” in *Proceedings IEEE International Conference on 3D System Integration (3DIC)*, 2010, pp. 1–8.

[20] I. C. Society, *IEEE Std 1500$^{\text{TM}}$-2005, IEEE Standard Testability Method for Embedded Core-based Integrated Circuits.* New York, NY, USA: IEEE, Aug. 2005.

[21] “IEEE Standard Test Access Port and Boundary Scan Architecture,” *IEEE Std 1149.1-2001*, 2001.

[22] E. Marinissen, S. Goel, and M. Lousberg, “Wrapper Design for Embedded Core Test,” in *Proceedings IEEE International Test Conference (ITC)*, 2000, pp. 911–920.

[23] V. Iyengar, K. Chakrabarty, and E. Marinissen, “Test Wrapper and Test Access Mechanism Co-optimization for System-on-Chip,” in *Proceedings IEEE International Test Conference (ITC)*, 2001, pp. 1023–1032.

[24] S. K. Goel and E. J. Marinissen, “SOC Test Architecture Design for Efficient Utilization of Test Bandwidth,” *ACM Transactions on Design Automation of Electronic Systems*, vol. 8, no. 4, pp. 399–429, Oct. 2003.

[25] E. Larsson, K. Arvidsson, H. Fujiwara, and Z. Peng, “Efficient Test Solutions for Core-Based Designs,” *IEEE Transactions on Computer-Aided Design*, vol. 23, no. 5, pp. 758–775, 2004.

[26] G. Giles, J. Wang, A. Sehgal, K. Balakrishnan, and J. Wingfield, “Test Access Mechanism for Multiple Identical Cores,” in *Proceedings IEEE International Test Conference (ITC)*, 2009, pp. 1–10.

[27] V. Iyengar, K. Chakrabarty, and E. J. Marinissen, “Test Wrapper and Test Access Mechanism Co-optimization for System-on-Chip,” *Journal of Electronic Testing*, vol. 18, no. 2, pp. 213–230, 2002.

[28] ——, “On Using Rectangle Packing for SOC Wrapper/TAM Co-optimization,” in *Proceedings IEEE VLSI Test Symposium (VTS)*, 2002, pp. 253–258.

[29] T. Waayers, R. Morren, and R. Grandi, “Definition of a Robust Modular SOC Test Architecture; Resurrection of the Single TAM Daisy-Chain,” in *Proceedings IEEE International Test Conference (ITC)*, 2005, pp. 10–pp.

[30] L. Jiang *et al.*, “Layout-Driven Test-Architecture Design and Optimization for 3D SoCs under Pre-Bond Test-Pin-Count Constraint,” in *ICCAD*, Nov. 2009, pp. 191–196.

[31] B. Noia *et al.*, “Test-Architecture Optimization for TSV-Based 3D Stacked ICs,” in *Proceedings IEEE European Test Symposium (ETS)*, 2010, pp. 24–29.

[32] ——, “Test-Architecture Optimization and Test Scheduling for TSV-Based 3-D Stacked ICs,” *IEEE Transactions on Computer-Aided Design*, vol. 30, no. 11, pp. 1705 –1718, Nov. 2011.

[33] D. Malta *et al.*, “Integrated Process for Defect-Dree Dopper Plating and Chemical-Mechanical Polishing of Through-Silicon Vias for 3D Interconnects,” in *Electronic Components and Technology Conference (ECTC)*, Jun. 2010, pp. 1769–1775.

[34] M. Tsai *et al.*, “Through Silicon Via (TSV) Defect/Pinhole Self Test Circuit for 3D-IC,” in *Proceedings IEEE International Conference on 3D System Integration (3DIC)*, Sep. 2009, pp. 1–8.

[35] C. Laviron *et al.*, “Via First Approach Optimisation for Through Silicon Via Applications,” in *Electronic Components and Technology Conference (ECTC)*, May 2009, pp. 14–19.

[36] G. Lee *et al.*, “Mechanical characterization of residual stress around tsv through instrumented indentation algorithm,” in *Proceedings IEEE International Conference on 3D System Integration (3DIC)*, 2012.

[37] C. Selvanayagam, J. Lau, X. Zhang, S. Seah, K. Vaidyanathan, and T. Chai, “Nonlinear Thermal Stress/Strain Analyses of Copper Filled TSV (Through Silicon Via) and Their Flip-Chip Microbumps,” *Advanced Packaging, IEEE Transactions on*, vol. 32, no. 4, pp. 720–728, Nov. 2009.

[38] K. H. Lu *et al.*, “Thermomechanical Reliability of Through-Silicon Vias in 3D Interconnects,” in *Proceedings IEEE International Reliability Physics Symposium (IRPS)*, Apr. 2011, pp. 3D.1.1–3D.1.7.

[39] S.-K. Ryu *et al.*, “Impact of Near-Surface Thermal Stresses on Interfacial Reliability of Through-Silicon Vias for 3-D Interconnects,” *Device and Materials Reliability, IEEE Transactions on*, vol. 11, no. 1, pp. 35–43, Mar. 2011.

[40] J.-S. Yang *et al.*, “TSV Stress Aware Timing Analysis with Applications to 3D-IC Layout Optimization,” in *Proceedings ACM/IEEE Design Automation Conference (DAC)*, Jun. 2010, pp. 803–806.

[41] N. Ahmed, M. Tehranipoor, and V. Jayaram, "Timing-Based Delay Test for Screening Small Delay Defects," in *Proceedings ACM/IEEE Design Automation Conference (DAC)*, 2006, pp. 320–325.

[42] Y.-M. Lin *et al.*, "Electromigration in Ni/Sn Intermetallic Micro Bump Joint for 3D IC Chip Stacking," in *Electronic Components and Technology Conference (ECTC)*, Jun. 2011, pp. 351–357.

[43] N. Ranganathan, K. Prasad, N. Balasubramanian, and K. Pey, "A Study of Thermo-Mechanical Stress and Its Impact on Through-Silicon Vias," *Journal of Micromechanics and Microengineering*, vol. 18, no. 7, 2008.

[44] K. H. Lu, X. Zhang, S.-K. Ryu, J. Im, R. Huang, and P. S. Ho, "Thermo-Mechanical Reliability of 3-D ICs Containing Through Silicon Vias," in *Electronic Components and Technology Conference (ECTC)*, 2009, pp. 630–634.

[45] M. Jung *et al.*, "TSV Stress-Aware Full-Chip Mechanical Reliability Analysis and Optimization for 3D IC," in *Proceedings ACM/IEEE Design Automation Conference (DAC)*, Jun. 2011, pp. 188–193.

[46] D. Rohde *et al.*, "Filling TSV of Different Dimension Using Galvanic Copper Deposition," in *IMPACT*, Oct. 2011, pp. 355–358.

[47] S. Deutsch and K. Chakrabarty, "Non-Invasive Pre-bond TSV Test Using Ring Oscillators and Multiple Voltage Levels," in *Proceedings Design, Automation, and Test in Europe (DATE) Conference*, 2013, pp. 1065–1070.

[48] B. Noia and K. Chakrabarty, "Pre-bond probing of TSVs in 3D stacked ICs," in *Proceedings IEEE International Test Conference (ITC)*, Sep. 2011.

[49] ——, "Identification of Defective TSVs in Pre-Bond Testing of 3D ICs," in *Proceedings IEEE Asian Test Symposium (ATS)*, Nov. 2011, pp. 187–194.

[50] P.-Y. Chen, C.-W. Wu, and D.-M. Kwai, "On-Chip Testing of Blind and Open-Sleeve TSVs for 3D IC before Bonding," in *Proceedings IEEE VLSI Test Symposium (VTS)*, Apr. 2010, pp. 263–268.

[51] J. S. Pak *et al.*, "Optimized Inverter Design of Ring Oscillator Based Wafer-Level TSV Connectivity Test (RO-TSV-CT)," in *Proceedings IEEE Electrical Design of Advanced Packaging and Systems Symposium (EDAPS)*, 2012, pp. 239–242.

[52] L.-R. Huang *et al.*, "Oscillation-Based Pre-Bond TSV Test," *IEEE Transactions on Computer-Aided Design*, vol. 32, no. 9, pp. 1440–1444, Sep. 2013.

[53] J.-W. You *et al.*, "Performance Characterization of TSV in 3D IC via Sensitivity Analysis," in *Proceedings IEEE Asian Test Symposium (ATS)*, Dec. 2010, pp. 389–394.

[54] S.-Y. Huang and L. Huang, "PLL-Assisted Timing Circuit for Accurate TSV Leakage Binning," *IEEE Design & Test of Computers*, vol. 31, no. 4, pp. 36–42, 2014.

[55] H. Hao and E. McCluskey, "Very-Low-Voltage Testing for Weak CMOS Logic ICs," in *Proceedings IEEE International Test Conference (ITC)*, Oct. 1993, pp. 275–284.

[56] Y. Liao and D. Walker, "Fault Coverage Analysis for Physically-based CMOS Bridging Faults at Different Power Supply Voltages," in *Proceedings IEEE International Test Conference (ITC)*, Oct. 1996, pp. 767–775.

[57] M. Renovell, P. Huc, and Y. Bertrand, "Bridging Fault Coverage Improvement by Power Supply Control," in *Proceedings IEEE VLSI Test Symposium (VTS)*, Apr. 1996, pp. 338–343.

[58] P. Engelke *et al.*, "The Pros and Cons of Very-Low-Voltage Testing: an Analysis Based on Resistive Bridging Faults," in *Proceedings IEEE VLSI Test Symposium (VTS)*, Apr. 2004, pp. 171–178.

[59] D. Arumi *et al.*, "Diagnosis of Bridging Defects Based on Current Signatures at Low Power Supply Voltages," in *Proceedings IEEE VLSI Test Symposium (VTS)*, May 2007, pp. 145–150.

[60] B. Wile, J. C. Goss, and W. Roesner, *Comprehensive Functional Verification: The Complete Industry Cycle.* Morgan Kaufmann, 2005.

[61] P. G. Maropoulos and D. Ceglarek, "Design Verification and Validation in Product Lifecycle," *CIRP Annals-Manufacturing Technology*, vol. 59, no. 2, pp. 740–759, 2010.

[62] G. Martin, B. Bailey, and A. Piziali, *ESL Design and Verification: a Prescription for Electronic System Level Methodology.* Morgan Kaufmann, 2010.

[63] M. E. Levitt, S. Nori, S. Narayanan, G. Grewal, L. Youngs, A. Jones, G. Billus, and S. Paramanandam, "Testability, Debuggability, and Manufacturability Features of the UltraSPARC-I Microprocessor," in *Proceedings IEEE International Test Conference (ITC)*, 1995, pp. 157–166.

[64] B. Vermeulen and S. K. Goel, "Design for Debug: Catching Design Errors in Digital Chips," *IEEE Design & Test of Computers*, vol. 19, no. 3, pp. 37–45, 2002.

[65] Q. Xu and X. Liu, "On Signal Tracing in Post-Silicon Validation," in *Proceedings of the 2010 Asia and South Pacific Design Automation Conference*, 2010, pp. 262–267.

[66] J. Katz, "A Case-Study in the Use of Scan in MicroSPARC Testing and Debug," in *Proceedings IEEE International Test Conference (ITC)*, 1994, pp. 456–460.

[67] S. K. Goel and B. Vermeulen, "Hierarchical Data Invalidation Analysis for Scan-Based Debug on Multiple-Clock System Chips," in *Proceedings IEEE International Test Conference (ITC)*, 2002, pp. 1103–1110.

[68] P. Dahlgren, P. Dickinson, and I. Parulkar, "Latch Divergency in Microprocessor Failure Analysis," in *Proceedings IEEE International Test Conference (ITC)*, vol. 3, 2003, pp. 755–763.

[69] Y.-C. Hsu, F. Tsai, W. Jong, and Y.-T. Chang, "Visibility Enhancement for Silicon Debug," in *Proceedings ACM/IEEE Design Automation Conference (DAC)*, 2006, pp. 13–18.

[70] J.-S. Yang and N. A. Touba, "Efficient Trace Signal Selection for Silicon Debug by Error Transmission Analysis," *IEEE Transactions on Computer-Aided Design*, vol. 31, no. 3, pp. 442–446, 2012.

[71] E. Hung and S. Wilton, "Scalable Signal Selection for Post-Silicon Debug," *IEEE Transactions on VLSI Systems*, vol. 21, no. 6, pp. 1103–1115, June 2013.

[72] K. Basu and P. Mishra, "Efficient Trace Data Compression Using Dtatically Delected Dictionary," in *Proceedings IEEE VLSI Test Symposium (VTS)*, 2011, pp. 14–19.

[73] F. Yuan, X. Liu, and Q. Xu, "X-Tracer: a Reconfigurable X-Tolerant Trace Compressor for Silicon Debug," in *Proceedings ACM/IEEE Design Automation Conference (DAC)*, 2012, pp. 555–560.

[74] E. Anis Daoud and N. Nicolici, "On Using Lossy Compression for Repeatable Experiments During Silicon Debug," *IEEE Transactions on Computers*, vol. 60, no. 7, pp. 937–950, 2011.

[75] J.-S. Yang and N. Touba, "Improved Trace Buffer Observation via Selective Data Capture Using 2-D Compaction for Post-Silicon Debug," *IEEE Transactions on VLSI Systems*, vol. 21, no. 2, pp. 320–328, Feb. 2013.

[76] R. Mijat, "Better Trace for Better Software," 2010, www.arm.com.

[77] T. Dao *et al.*, "Through Silicon Via Stress Characterization," in *Proceedings International Conference on IC Design and Technology (ICICDT)*, May 2009, pp. 39–41.

[78] Y. Sato *et al.*, "Invisible Delay Quality SDQM Model Lights Up What Could Not Be Seen," in *Proceedings IEEE International Test Conference (ITC)*, Nov. 2005.

[79] X. Lin *et al.*, "Timing-Aware ATPG for High Quality At-speed Testing of Small Delay Defects," in *Proceedings IEEE Asian Test Symposium (ATS)*, Nov. 2006, pp. 139–146.

[80] M. Yilmaz, K. Chakrabarty, and M. Tehranipoor, "Test-Pattern Selection for Screening Small-Delay Defects in Very-Deep Submicrometer Integrated Circuits," *IEEE Transactions on Computer-Aided Design*, vol. 29, no. 5, pp. 760–773, May 2010.

[81] S. Mitra *et al.*, "Delay Defect Screening Using Process Monitor Structures," in *Proceedings IEEE VLSI Test Symposium (VTS)*, Apr. 2004, pp. 43–48.

[82] R. C. Jaeger *et al.*, "CMOS Stress Sensors on (100) Silicon," *Journal of Computer and System Sciences*, pp. 85–95, Jan. 2000.

[83] Http://www.simulia.com.

[84] http://opencores.org.

[85] D. H. Kim, K. Athikulwongse, and S. K. Lim, "A Study of Through-Silicon-Via Impact on the 3D Stacked IC Layout," in *Proceedings International Conference on Computer-Aided Design (ICCAD)*, Nov. 2009, pp. 674–680.

[86] G. Katti *et al.*, "Electrical Modeling and Characterization of Through Silicon via for Three-Dimensional ICs," *IEEE Transactions on Electron Devices*, vol. 57, no. 1, pp. 256–262, Jan. 2010.

[87] P. Enquist *et al.*, "Low Cost of Ownership Scalable Copper Direct Bond Interconnect 3D IC technology for Three Dimensional Integrated Circuit Applications," in *Proceedings IEEE International Conference on 3D System Integration (3DIC)*, Sep. 2009, pp. 1–6.

[88] G. Maier, "IBM," May 2012, private Correspondence.

[89] Y.-J. Lee, I. Hong, and S. K. Lim, "Slew-Aware Buffer Insertion for Through-Silicon-Via-Based 3D ICs," in *Proceedings IEEE Custom Integrated Circuits Conference (CICC)*, 2012, invited Paper.

[90] K. Bernstein, "High-performance CMOS variability in the 65-nm regime and beyond," *IBM Journal of Research and Development*, vol. 50, no. 4.5, pp. 433–449, Jul. 2006.

[91] S. Deutsch and K. Chakrabarty, "Contactless Pre-Bond TSV Test and Diagnosis Using Ring Oscillators and Multiple Voltage Levels," *IEEE Transactions on Computer-Aided Design*, vol. 33, no. 5, May 2014.

[92] L. Ravezzi, "Duty-Cycle Detector Based on Time-to-Digital Conversion," *Electronics Letters*, 2013.

[93] I. Basheer and M. Hajmeer, "Artificial Neural Networks: Fundamentals, Computing, Design, and Application," *Journal of Microbiological Methods*, vol. 43, no. 1, pp. 3–31, 2000.

[94] 45nm Predictive Technology Model, http://ptm.asu.edu/.

[95] Nangate 45nm Open Cell Library, http://www.nangate.com/openlibrary.

[96] MathWorks, http://www.mathworks.com/help/nnet/ug/choose-a-multilayer-neural-network-training-function.html.

[97] E. Larsson and Z. Peng, "An Integrated Framework for the Design and Optimization of SOC Test Solutions," *Journal of Electronic Testing: Theory and Applications*, vol. 18, no. 4, pp. 385–400, 2002.

[98] J. M. Mulvey, R. J. Vanderbei, and S. A. Zenios, "Robust Optimization of Large-Scale Systems," *Operations Research*, vol. 43, no. 2, pp. 264–281, 1995.

[99] P. van Laarhoven and E. Aarts, *Simulated Annealing: Theory and Applications.* Springer, 1987, vol. 37.

[100] E. J. Marinissen, V. Iyengar, and K. Chakrabarty, "A Set of Benchmarks for Modular Testing of SoCs," in *Proceedings IEEE International Test Conference (ITC)*, Oct. 2002, pp. 519–528.

[101] I. D.-T. P. W. Group, http://grouper.ieee.org/groups/3Dtest/.

[102] J. Lau and T. Yue, "Thermal Management of 3D IC Integration with TSV (Through Silicon Via)," in *Electronic Components and Technology Conference (ECTC)*, 2009, pp. 635–640.

[103] Y. Joshi, A. Fedorov, Y. Lee, and S. Lim, "Thermal Characterization of Interlayer Microfluidic Cooling of Three-Dimensional Integrated Circuits with Nonuniform Heat Flux," *Journal of Heat Transfer*, vol. 132, 2010.

[104] A. Dembla, Y. Zhang, and M. Bakir, "Fine Pitch TSV Integration in Silicon Micropin-Fin Heat Sinks for 3D ICs," in *Proceedings IEEE International Interconnect Technology Conference (IITC)*, 2012, pp. 1–3.

[105] P. Maxwell, "Adaptive Testing: Dealing with Process Variability," *IEEE Design & Test of Computers*, vol. 28, no. 6, pp. 41–49, 2011.

[106] S. Koranne, "A Novel Reconfigurable Wrapper for Testing of Embedded Core-Based SOCs and Its Associated Scheduling Algorithm," *Journal of Electronic Testing*, pp. 415–434, 2002.

[107] E. Larsson and Z. Peng, "A Reconfigurable Power-Conscious Core Wrapper and Its Application to SOC Test Scheduling," in *Proceedings IEEE International Test Conference (ITC)*, 2003, pp. 1135–1144.

[108] S. Deutsch and K. Chakrabarty, "Robust Optimization of Test-Architecture Designs for Core-Based SoCs," in *Proceedings IEEE European Test Symposium (ETS)*, 2013.

[109] W. Zou *et al.*, "SOC Test Scheduling Using Simulated Annealing," in *Proceedings IEEE VLSI Symposium on Circuits (VLSI)*, 2003, pp. 325–330.

[110] L. Jiang, L. Huang, and Q. Xu, "Test Architecture Design and Optimization for Three-Dimensional SoCs," in *Proceedings Design, Automation, and Test in Europe (DATE) Conference*, 2009, pp. 220–225.

[111] C. Hsu *et al.*, "3D IC Test Scheduling Using Simulated Annealing," in *VLSI Design, Automation, and Test (VLSI-DAT), 2012 International Symposium on*, 2012.

[112] M. Garey and D. Johnson, "Computers and Intractability: a Guide to the Theory of NP-Completeness," *WH Freeman & Co., San Francisco*, 1979.

[113] S. Deutsch, K. Chakrabarty, and E. J. Marinissen, "Uncertainty-Aware Robust Optimization of Test-Access Architectures for 3D Stacked ICs," in *Proceedings IEEE International Test Conference (ITC)*, 2013, pp. 1–10.

[114] K. Chakrabarty, "Test Scheduling for Core-Based Systems Using Mixed-Integer Linear Programming," *IEEE Transactions on Computer-Aided Design*, pp. 1163–1174, 2000.

[115] P. Östergård, "A New Algorithm for the Maximum-Weight Clique Problem," *Nordic Journal of Computing*, vol. 8, no. 4, pp. 424–436, 2001.

[116] http://arstechnica.com/gadgets/2013/09/intels-atom-cpus-finally-get-serious-with-the-new-bay-trail-architecture, Sep. 2013, (last accessed in Feb. 2015).

[117] http://electronicdesign.com/eda/finding-bug-soc-haystack, Aug. 2013, (last accessed in Mar. 2014).

[118] JEDEC, www.jedec.org.

[119] Hybrid Memory Cube Consortium, www.hybridmemorycube.org.

[120] J. Jeddeloh and B. Keeth, "Hybrid Memory Cube New DRAM Architecture Increases Density and Performance," in *VLSI Technology (VLSIT), 2012 Symposium on*, 2012, pp. 87–88.

[121] Cadence Design Systems, www.cadence.com.

[122] AMBA, www.arm.com/products/system-ip/amba/index.php.

[123] T. Kgil, A. Saidi, N. Binkert, S. Reinhardt, K. Flautner, and T. Mudge, "PicoServer: Using 3D Stacking Technology to Build Energy Efficient Servers," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 4, no. 4, p. 16, 2008.

Biography

Sergej Deutsch was born in Melnica, Russia, on the 28th of August, 1984. He received his Diplom Degree in Electrical Engineering from the University of Technology Braunschweig (TU-BS), Germany, in 2011, and his Ph.D. in Electrical and Computer Engineering from Duke University in 2015.

As a TU-BS student, Sergej was a recipient of a scholarship from The German National Academic Foundation. He received best-paper award at the IEEE Asian Test Symposium in 2012. In September 2013, Sergej received best-in-session award at SRC TECHCON'13.

Sergej has a number of conference and journal publications. He presented his work at multiple conferences around the world, including the IEEE International Test Conference, the IEEE Design, Automation & Test in Europe, the IEEE European Test Symposium, and the IEEE Asia and South Pacific Design Automation Conference. Sergej's publications include:

1. E.J. Marinissen, M. Konijnenburg, S. Deutsch, V. Chickermane, B. Keller, S. Mukherjee, and S. K. Goel, "Automated Design-for-Test for 2.5D- and 3D-SICs", *Chip Scale Review Magazine*, Oct 2011.

2. S. Deutsch, V. Chickermane, B. L. Keller, S. Mukherjee, M. H. Konijnenburg, E. J. Marinissen, and S. K. Goel, "Automation of 3D-DfT Insertion", in *Proceedings IEEE Asian Test Symposium (ATS)*, Nov 2011.

3. K. Chakrabarty, S. Deutsch, H. Thapliyal and F. Ye, "TSV Defects and TSV-

Induced Circuit Failures: The Third Dimension in Test and Design-for-Test" (invited paper), in *Proceedings IEEE International Reliability Physics Symposium (IRPS)*, Apr 2012.

4. E.J. Marinissen, S. Deutsch, B. Keller, V. Chickermane, and S. Mukherjee, N. Sood, "Interconnect Test for Wide-I/O Memory-on-Logic Stacks", *Future Fab Int.*, Jul 2012.

5. S. Deutsch, B. Keller, V. Chickermane, S. Mukherjee, N. Sood, S. K. Goel, J. Chen, A. Mehta, F. Lee, and E. J. Marinissen, "DFT Architecture and ATPG for Interconnect Test of JEDEC Wide-IO Memory-on-Logic Die Stacks", in *Proceedings IEEE International Test Conference (ITC)*, Nov 2012.

6. S. Deutsch, K. Chakrabarty, S. Panth, and S. K. Lim, "TSV Stress-Aware ATPG for 3D Stacked ICs", in *Proceedings IEEE Asian Test Symposium (ATS)*, Nov 2012.

7. S. Deutsch, K. Chakrabarty, "Non-Invasive Pre-Bond TSV Test Using Ring Oscillators and Multiple Voltage Levels", in *Proceedings IEEE Design, Automation & Test in Europe (DATE)*, Mar 2013.

8. S. Deutsch, K. Chakrabarty, "Robust Optimization of Test-Architecture Designs for Core-Based SoCs", in *Proceedings IEEE European Test Symposium (ETS*, May 2013.

9. S. Deutsch, K. Chakrabarty, "Uncertainty-Aware Robust Optimization of Test-Access Architectures for 3D Stacked ICs", in *Proceedings IEEE International Test Conference (ITC)*, Sep 2013.

10. S. Deutsch and K. Chakrabarty, "Contactless Pre-bond TSV Test and Diagnosis Using Ring Oscillators and Multiple Voltage Levels", *IEEE Transactions on Computer-Aided Design*, May 2014.

11. S. Deutsch and K. Chakrabarty, “Massive signal tracing using on-chip DRAM for in-system silicon debug”, in *Proceedings IEEE International Test Conference (ITC)*, Oct 2014.

12. S. Deutsch and K. Chakrabarty, “Software-Based Test and Diagnosis of SoCs Using Embedded and Wide-I/O DRAM”, in *Proceedings IEEE Asia and South Pacific Design Automation Conference (ASPDAC)*, Jan 2015.

13. S. Deutsch, K. Chakrabarty, and E.J. Marinissen, “Robust Optimization of Test-Access Architectures under Realistic Scenarios”. accepted for publication in *IEEE Transactions on Computer-Aided Design*, 2015.