System-On-Chip Testing

A Dissertation

submitted to the designated
by the General Assembly of Special Composition
of the Department of Computer Science and Engineering
Examination Committee

by

Panagiotis Georgiou

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

University of Ioannina
October 2019

Advisory Committee:

- **Chrysovalantis Kavousianos**, Professor, Department of Computer Science & Engineering, University of Ioannina
- **Yiorgos Tsiatouhas**, Professor, Department of Computer Science & Engineering, University of Ioannina
- **Krishnendu Chakrabarty**, Professor, Department of Electrical and Computer Engineering, Duke University, USA

Examining Committee:

- **Chrysovalantis Kavousianos**, Professor, Department of Computer Science & Engineering, University of Ioannina
- **Yiorgos Tsiatouhas**, Professor, Department of Computer Science & Engineering, University of Ioannina
- **Krishnendu Chakrabarty**, Professor, Department of Electrical and Computer Engineering, Duke University, USA
- **Aristides Efthymiou**, Assistant Professor, Department of Computer Science & Engineering, University of Ioannina
- **Vasileios Tenentes**, Assistant Professor, Department of Computer Science & Engineering, University of Ioannina
- **Matteo Sonza Reorda**, Professor, Department of Control and Computer Engineering, Politecnico di Torino, Italy
- **Giorgos Dimitrakopoulos**, Assistant Professor, Department of Electrical and Computer Engineering, Democritus University of Thrace

Dedication

Στους γονείς μου.

Acknowledgements

First of all, I would like to sincerely thank my supervisor, Prof. C. Kavousianos, for his guidance throughout this study and also for creating collaboration opportunities with valuable research institutions.

Special thanks to Prof. K. Chakrabarty for providing useful feedback on my research and assisting me many times during my study.

I would also like to thank Prof. G. Tsiatouhas and Prof. K. Chakrabarty for their support as members of the advisory committee.

I would also like to express my sincere gratitude to the CAD group leader, Prof. M. Sonza Reorda, who has generously spared his valuable time to give me support with his expertise, during my visit at Politecnico di Torino.

I would also like to thank all of the CAD group members Ricardo, Emanuele, Andrea and Sara for their support, during my visit at Politecnico di Torino.

Many thanks to my colleagues F. Vartziotis, C. Papaioannou, I. Theodosopoulos and G. Sfikas for their support and the good times we had together.

Table of Contents

List of Figures

List of Tables

List of Algorithms

Abstract

Panagiotis Georgiou, Ph.D., Department of Computer Science and Engineering, University of Ioannina, Greece, October 2019.
System-On-Chip Testing.
Advisor: Chrysovalantis Kavousianos, Professor.

We already live in the era of Internet of Things. The common devices we use daily are connected together and are getting ”smarter” rapidly. In every device belonging in IoT, there is an SoC. In order to satisfy the continuous increased requirements of the new era, SoCs are constantly evolving.

3D-ICs is a promising solution to satisfy the demands of the new era and seem to secure the continuation of Moore's Law for the near future. 3D-ICs achieve higher packing density and higher performance than 2D-ICs and reduce the cost of wiring and power consumption. Recently, the semiconductor companies released products based on 3D-ICs.

This research focuses in the development of new TAM architectures and test-scheduling methods for 3D-SoCs, which exploit the high speed offered by TSVs, while power and thermal constraints are met. We introduce a new TAM architecture for 3D SoCs, which minimizes the test-time, the number of TSVs, and TAM lines used for transferring test-data to the cores. The test schedule is calculated by a very effective TDM method, and a highly efficient optimization method based on rectangle-packing and simulated-annealing. Experiments have shown that as much as 9.6× better test time can be achieved using the proposed method, especially under strict power and thermal constraints.

The previous method is compatible only with bus-based TAMs, which require long interconnection wires and many buffers at each die of the stack, therefore they fail to fully exploit the high frequencies of the global channels. In order to overcome

the limitations of the previous method, we propose a new TDM-based 3D TAM architecture, which uses daisy-chains and offers higher test-time benefits and significantly lower interconnection overhead.

This research also focuses in the improvement of the defect screening of processor-based devices. The continually increasing demands of the market for higher computational performance at lower cost and power consumption drive processor vendors to develop new microprocessor generations, which introduce new challenges on processor-based device testing. The need to test the processor-based devices at the normal mode of operation, impose the complementary use of non-intrusive test methods, such as SBST.

Most SBST techniques often target only the stuck-at fault model, which is inadequate for detecting many defects. SBST methods also require extensive human intervention and long development times. Moreover, they involve the CPU-intensive process of fault-simulating multi-million gate designs for multi-million clock cycles using multiple fault models and specialized functional (non-scan) simulators.

We introduce the first fault-independent SBST method, which offers short test-program generation time under strict test-application-time and test-program-size constraints. The test-programs are evaluated by means of a novel and very effective SBST-oriented probabilistic metric, which considers both the architectural model and the synthesized gate-level netlist of the DUT. The proposed metric, which is based on output deviations, can be calculated very quickly as it omits the time-consuming functional fault-simulation, and it can be applied to any SBST-based method.

Εκτεταμενη Περιληψη

Παναγιώτης Γεωργίου, Δ.Δ., Τμήμα Μηχανικών Η/Υ και Πληροφορικής, Πανεπιστήμιο Ιωαννίνων, Οκτώβριος 2019.
Έλεγχος Ορθής Λειτουργίας Ολοκληρωμένων Συστημάτων.
Επιβλέπων: Χρυσοβαλάντης Καβουσιανός, Καθηγητής.

Διανύουμε ήδη την εποχή του ''Ίντερνετ των Πραγμάτων''. Οι κοινές συσκευές που χρησιμοποιούμε καθημερινά, συνδέονται μεταξύ τους και γίνονται ''εξυπνότερες'' με ραγδαίους ρυθμούς. Σε κάθε τέτοια συσκευή βρίσκεται ένα Σύστημα σε Ολοκληρωμένο (Systems-On-Chip ή SoC). Το SoC εξελίσσεται συνεχώς, για να ικανοποιηθούν οι συνεχώς αυξανόμενες απαιτήσεις της νέας εποχής.

Τα τρι-διάστατα ολοκληρωμένα κυκλώματα (three-dimensional integrated circuits - 3D-ICs) είναι μια υποσχόμενη λύση για να ικανοποιήσουν τις απαιτήσεις τις νέας εποχής και φαίνεται να εξασφαλίζουν τη συνέχιση του Νόμου του Moore στο άμεσο μέλλον. Τα 3D-ICs πετυχαίνουν υψηλότερη πυκνότητα πυλών και καλύτερη απόδοση από τα συμβατικά SoC και μειώνουν το κόστος διασύνδεσης και κατανάλωσης. Πρόσφατα, οι κατασκευαστικές εταιρείες ολοκληρωμένων συστημάτων κυκλοφόρησαν προϊόντα βασισμένα σε 3D-ICs.

Η έρευνα αυτή εστιάζει στην ανάπτυξη νέων αρχιτεκτονικών μηχανισμού πρόσβασης ελέγχου (Test Access Mechanisms - TAMs) και νέων μεθόδων χρονοπρογραμματισμού ελέγχου ορθής λειτουργίας για 3D-SoCs, οι οποίες εκμεταλλεύονται την υψηλή ταχύτητα που προσφέρουν οι ειδικές κάθετες διασυνδέσεις μέσω-πυριτίου (Through Silicon Vias - TSVs), ενώ η κατανάλωση ισχύος και η θερμότητα πρέπει να διατηρηθούν κάτω από ορισμένα επίπεδα. Εισάγουμε μία νέα αρχιτεκτονική TAM για 3D SoCs, η οποία ελαχιστοποιεί το χρόνο ελέγχου ορθής λειτουργίας, το πλήθος των TSVs και τις γραμμές της αρχιτεκτονικής TAM που χρησιμοποιούνται για να μεταφερθούν τα δεδομένα ελέγχου. Ο χρονοπρογραμματισμός του ελέγχου ορθής λειτουργίας υπολογίζεται από μία αποδοτική μέθοδο χρονικής πολυπλεξίας

και μία πολύ αποδοτική μέθοδο βελτιστοποίησης που βασίζεται στους αλγορίθμους rectangle-packing και simulated-annealing. Πειραματικά αποτελέσματα δείχνουν έως και 9.6 φορές εξοικονόμηση στο χρόνο ελέγχου με την προτεινόμενη μέθοδο, ειδικά κάτω από αυστηρά όρια για την κατανάλωση ισχύος και τη θερμότητα.

Η προηγούμενη μέθοδος είναι συμβατή μόνο με TAMs που βασίζονται σε αρτηρίες (buses), οι οποίες απαιτούν διασυνδέσεις μεγάλου μήκους και πολλά buffers σε κάθε επίπεδο του 3D-IC, επομένως δεν καταφέρνουν να εκμεταλλευτούν πλήρως τις υψηλές συχνότητες των TSVs. Προτείνουμε μία νέα αρχιτεκτονική TAM βασισμένη στη χρονική πολυπλεξία, που χρησιμοποιεί σειριακές αλυσίδες (daisy-chains) για να ξεπεράσουμε τους περιορισμούς της προηγούμενης μεθόδου. Η μέθοδος αυτή προσφέρει μεγαλύτερα κέρδη όσον αφορά το χρόνο ελέγχου ορθής λειτουργίας και το κόστος διασύνδεσης.

Η έρευνα αυτή εστιάζει στη βελτίωση ανίχνευσης σφαλμάτων συσκευών βασιζόμενων σε επεξεργαστή. Οι ολοένα αυξανόμενες απαιτήσεις της αγοράς για υψηλότερη υπολογιστική απόδοση σε μικρότερο κόστος και χαμηλότερη κατανάλωση ισχύος, οδηγεί τους κατασκευαστές στην ανάπτυξη νέων μικροεπεξεργαστών, που εισάγουν νέες προκλήσεις στον έλεγχο συσκευών βασιζόμενων σε επεξεργαστή. Η ανάγκη ελέγχου των συσκευών αυτών κατά τη διάρκεια της κανονικής τους λειτουργίας, επιβάλλουν τη συμπληρωματική χρήση μεθόδων ελέγχου που δεν επηρεάζουν τη λειτουργία, όπως ο «αυτοέλεγχος βασισμένος σε λογισμικό» (Software-Based Self-Test - SBST).

Οι περισσότερες τεχνικές SBST στοχεύουν μόνο το μοντέλο σφαλμάτων stuck-at, που δεν αρκεί για την ανίχνευση πολλών σφαλμάτων. Επίσης, οι τεχνικές SBST απαιτούν εκτενή ανθρώπινη ενασχόληση με μεγάλους χρόνους ανάπτυξης των προγραμμάτων ελέγχου. Επιπλέον, περιλαμβάνουν την κοστοβόρα, από άποψη υπολογιστική ισχύος, εξομοίωση σφαλμάτων SoCs με εκατομμύρια πύλες για εκατομμύρια κύκλους ρολογιού, χρησιμοποιώντας πολλαπλά μοντέλα σφαλμάτων και εξειδικευμένους λειτουργικούς εξομοιωτές.

Εισάγουμε την πρώτη μέθοδο που δεν μεροληπτεί υπέρ κάποιου συγκεκριμένου μοντέλου σφαλμάτων. Η μέθοδος αυτή προσφέρει σύντομο χρόνο δημιουργίας προγραμμάτων ελέγχου, υπό αυστηρό περιορισμό στο χρόνο ελέγχου ορθής λειτουργίας και στο μέγεθος των προγραμμάτων ελέγχου. Τα προγράμματα ελέγχου αξιολογούνται από μία νέα αποδοτική πιθανοτική μέθοδο SBST, εκμεταλλευόμενη

την αρχιτεκτονική του επεξεργαστή, καθώς και τη netlist του επεξεργαστή σε επίπεδο πυλών που έχει προκύψει από σύνθεση. Η προτεινόμενη μετρική που βασίζεται στα output deviations είναι πολύ γρήγορη καθώς δεν απαιτεί τη χρονοβόρα διαδικασία της εξομοίωσης σφαλμάτων και μπορεί να εφαρμοστεί σε οποιαδήποτε μέθοδο που βασίζεται στην τεχνική SBST.

Chapter 1

Introduction

1.1 System On Chip

The vision of Internet of Things (IoT) promises billions of devices equipped with processing, memory and Internet connection abilities. Nowadays, this vision is being realized, as many common objects that people use are getting ”smarter”. IoT applies on various aspects of life, like housekeeping, transportation, medical health, agriculture, energy management etc. In Figure 1.1 IoT is presented as a Network of Networks.

The heart of every device belonging in IoT is a System on Chip (SoC). The viability of IoT depends on simple SoCs that need to be inexpensive and able to operate under strict performance, power and area constraints.

An SoC is more of a system, rather than a chip, as it integrates all components into a single chip. An SoC may consist of processor cores (including DSPs, microprocessors,

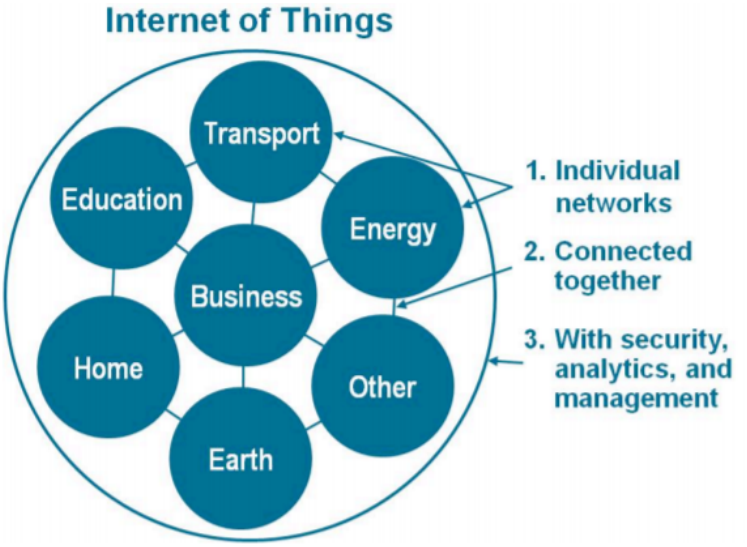


Figure 1.1: IoT as a network of networks [1].

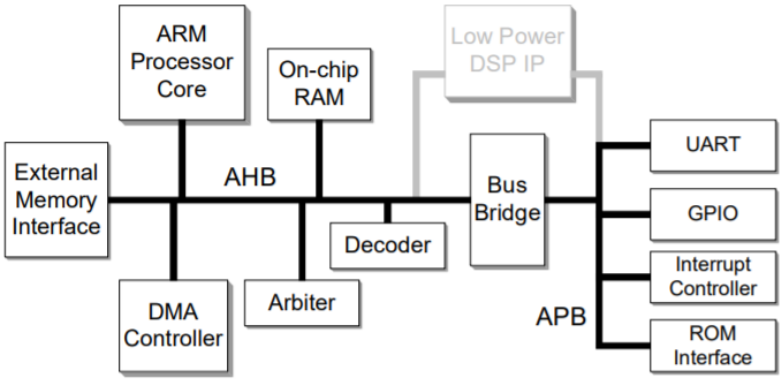


Figure 1.2: ARM based SoC architecture [2].

micro-controllers), on-chip interconnection, Intellectual Property (IP) cores (on-chip memory, peripherals, power management etc.), application specific hardware modules, analog circuits, ASICs logics, firmware, software. In Figure 1.2 an Advanced-RISC-Machines-(ARM)-based-SoC architecture is presented.

The complexity of SoC design is high, because of the communications between its different components. In SoC design, the density integration is high, in order to include many different components in small size. As the number of different components increases, there is the need to maintain the power low. Also, verification is necessary at different levels. All these aspects of SoC design are taken into account accompanied with the high time-to-market pressure.

The power consumption of a device is composed by the dynamic power con-

sumption P_D created by the switching activity and the static power consumption P_L, created by the leakage currents. Therefore, total power consumption is:

$$P_{total} = P_D + P_L \tag{1.1}$$

In traditional systems, the dynamic power consumption is the most important component of power consumption, but lately, the static power has become significant due to the deep sub-micron process technology. The heat produced during the operation of a circuit is proportional to the dissipated power. Therefore, there is a relationship between die temperature and power dissipation, which can be formulated by the laws of thermodynamics as follows [19]:

$$T_{die} = T_{air} + \theta \times P_D, \tag{1.2}$$

where T_{die} is the die temperature, T_{air} is the surrounding air temperature, θ is the package thermal impedance, and P_D is the average power dissipated by the circuit. It is clear that if the dissipated power is increased, the circuit temperature will be high. If the temperature becomes too high even for a short period, it may cause irreversible structural degradations, called hot spots, or affect circuit's performance and ageing.

1.2 VLSI Testing

1.2.1 Why Testing?

The stage of testing very-large-scale integration (VLSI) circuits is one of the most important part of their production line. The scale of integrated circuits (ICs) doubles every 18 months according to the famous Moore's law. The constant decrease of the transistor dimensions allows the packing of a huge amount of transistors in a single IC as well as increased operating frequencies. Given these, it is clear that the probability of a defect in the IC increases and, therefore, the semiconductor industry dedicates a lot of effort in order to avoid faulty chips.

There are two main causes of digital circuit malfunction, manufacturing defects and soft errors. In order to produce an electronic system, we must produce ICs, then assemble the ICs into printed circuit boards (PCBs), and finally use the PCBs to assemble the system. It is necessary to test components at every stage of the manufacturing process. Manufacturing defects are physical defects that may happen

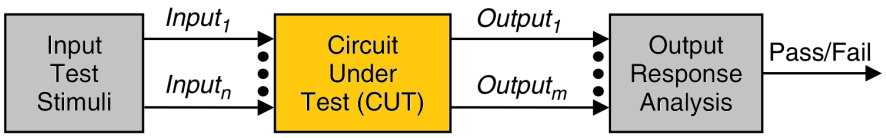


Figure 1.3: Basic testing approach [3].

at any step of the manufacturing process and may result in static or timing faults. The general rule of ten, declares that the cost of detecting a faulty device increases by an order of magnitude as we move through each stage of manufacturing [3]. The production of faulty ICs is inevitable, therefore testing is useful to improve production yield by analyzing the cause of defects when faults are encountered. Environmental conditions, such as α - particle radiation, may cause a fault-free circuit to malfunction during operation [20]. This situation is defined as a transient fault, resulting in a soft error. Unfortunately, transient faults cannot be detected in the manufacturing process, as they are not repeatable.

1.2.2 The Basics

A typical test procedure consists of applying a set of test stimuli to the inputs of the circuit under test (CUT), while analyzing the output responses, as it is illustrated in Figure 1.3 [3]. The circuits that produce the expected output responses for all input stimuli pass successfully the test and are considered fault-free, otherwise are considered faulty.

In Figure 1.4, the VLSI development process is illustrated. The first step is to specify the design specifications for the VLSI circuit. Afterwards, the synthesis of the circuit that satisfies the specifications and the design verification follows. The verification stage is necessary to ensure that the synthesized design will perform the required functions that are described in the specifications. If a design error is found, then the design is modified and the verification is repeated.

After successful verification, the VLSI design goes to fabrication. The testing performed during the manufacturing process tests the fabricated ICs on the wafer in order to determine which devices are defective. A defect is a flaw or physical imperfection that may lead to a fault.

All the chips that pass wafer-level testing are packaged. The packaged devices are once again tested in order to detect any devices that have been damaged during the

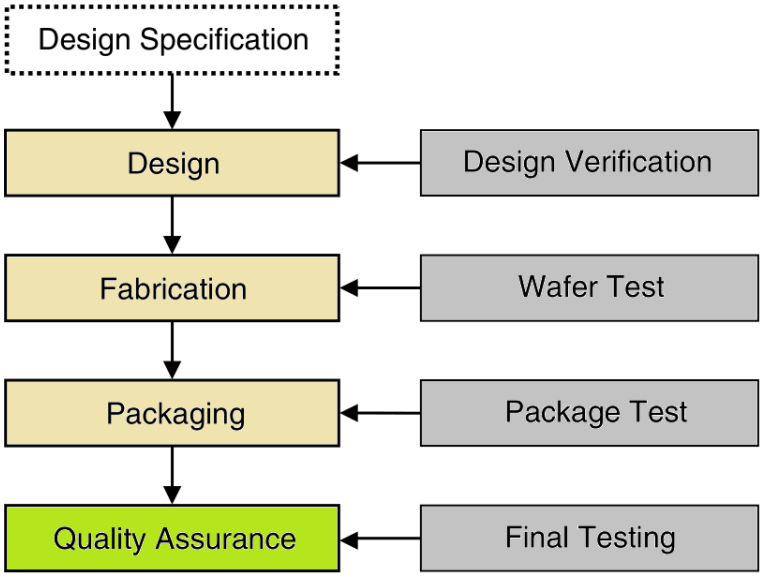


Figure 1.4: VLSI development process.

packaging process or put into defective packages.

Finally, another test procedure follows including the measurements of various parameters like input/output, timing, specifications, voltage and current. In this way, the final quality is assured before going to market.

Additionally, burn-in or stress testing is often performed where chips are subjected to high temperatures and supply voltage. The purpose of burn-in testing is to accelerate the effect of defects that could lead to failures in the early stages of operation of the IC.

1.2.3 The Standards

Some percentage of the manufactured ICs is expected to be faulty due to manufacturing defects. The yield of a manufacturing process is defined as the percentage of acceptable parts among all parts that are fabricated (# stands for Number):

$$Yield = \frac{\#\ of\ acceptable\ parts}{Total\ \#\ of\ parts\ fabricated} \tag{1.3}$$

There are two types of yield loss: catastrophic and parametric. Catastrophic yield loss is due to random defects, and parametric yield loss is due to process variations. When ICs are tested, the following two undesirable situations may occur:

- A faulty chip may pass the test.

- A good chip may fail the test and appear as faulty.

These two outcomes are often due to a poorly designed test or the lack of design for testability (DFT).

The ratio of field-rejected parts to all parts passing quality assurance testing is referred to as reject rate or defect level:

$$Reject\ rate = \frac{\#\ of\ faulty\ parts\ passing\ final\ test}{Total\ \#\ of\ parts\ passing\ final\ test} \tag{1.4}$$

It is clear that the reject rate provides an indication of the overall quality of the VLSI testing process [21]. Generally speaking, a reject rate of 500 parts per million (PPM) chips may be considered to be acceptable, while 100PPM or lower represents high quality.

As it is already mentioned, to test a circuit, a set of input patterns is applied to the CUT, and its responses are compared to the known good responses of a fault-free circuit. Each input pattern is called a test vector. We can use exhaustive approach in order to completely test a circuit, by applying all possible input patterns for testing. If a circuit passes exhaustive testing, we might assume that the circuit does not contain functional faults, regardless of its internal structure. Furthermore, this example of applying all possible input test patterns also illustrates the basic idea of functional testing, where every entry in the truth table for the combinational logic circuit is tested to determine whether it produces the correct response. However, exhaustive and functional testing, are not practical for large circuits. In addition, they lack of a quantitative measure of the defects that will be detected by the set of functional test vectors.

Structural testing is an efficient approach, where specific test patterns are selected based on the structural information of the circuit and a set of fault models. Using this approach, time is saved and test efficiency is improved. The total number of test patterns is decreased, because the test vectors target specific faults that would result from defects in the manufactured circuit. Structural testing cannot guarantee detection of all possible manufacturing defects, as the test vectors are generated based on specific fault models. However, the use of fault models provides a quantitative measure of the fault-detection capabilities of a given set of test vectors for a targeted fault model. This measure is called fault coverage and is defined as:

$$Fault\ coverage = \frac{\#\ of\ detected\ faults}{Total\ \#\ of\ faults} \tag{1.5}$$

It may be impossible to obtain 100% fault coverage because of the existence of undetectable faults. An undetectable fault means there is no test to distinguish the fault-free circuit from a faulty circuit containing that fault. As a result, the fault coverage can be modified and expressed as the fault detection efficiency, also referred to as the effective fault coverage, which is defined as:

$$Fault\ detection\ efficiency = \frac{\#\ of\ detected\ faults}{Total\ \#\ of\ faults - \#\ of\ undetectable\ faults} \tag{1.6}$$

Fault coverage is linked to the yield and the defect level by the following expression [22]:

$$Reject\ rate = 1 - yield^{(1-fault\ coverage)} \tag{1.7}$$

Overall, the goal of test generation is to find an efficient set of test vectors that will succeed the maximum fault coverage. The later is evaluated using fault simulation, which in turn requires fault models to emulate behavior of defects. As a result, fault models are needed for fault simulation as well as for test generation.

1.2.4 Fault Models

In general, a good fault model should satisfy two criteria [23]:

- It should accurately reflect the behavior of defects.
- It should be computationally efficient in terms of the time required for fault simulation and test pattern generation.

For a given fault model there will be k different types of faults that can occur at each potential fault site ($k = 2$ for most fault models). A given circuit contains n possible fault sites, depending on the fault model. Assuming that there can be only one fault in the circuit, then the total number of possible single faults, referred to as the single-fault model or single-fault assumption, is given by:

$$\#\ of\ single\ faults = k \times n \tag{1.8}$$

In reality, of course, multiple faults may occur in the circuit. The total number of possible combinations of multiple faults, referred to as the multiple-fault model, is given by:

$$\#\ of\ multiple\ faults = (k+1)^n - 1 \tag{1.9}$$

While the multiple-fault model is more accurate than the single-fault assumption, the number of possible faults becomes impractically large. In addition, it has been shown that high fault coverage obtained under the single-fault assumption will result in high fault coverage for the multiple-fault model; therefore, the single-fault assumption is typically used for test generation and evaluation. However, it should be noted that no single fault model accurately reflects the behavior of all possible defects that can occur. As a result, a combination of different fault models is often used in the generation and evaluation of test vectors and testing approaches developed for VLSI devices.

In the single-fault model, equivalent faults may happen, when two or more faults result in identical faulty behavior for all possible input patterns. These faults can be represented by any single fault from the set of equivalent faults. Therefore, we may consider much less single faults than $k \times n$ for test generation. This reduction of the entire set of single faults by removing equivalent faults is referred to as fault collapsing. Fault collapsing reduces both test generation and fault simulation times.

A stuck-at fault affects the state of logic signals on lines in a logic circuit. It transforms the correct value on the faulty signal line to appear to be stuck at a constant logic value, either a logic 0 or a logic 1, referred to as stuck-at-0 (SAO) or stuck-at-1 (SAl), respectively. Consider the example in Figure 1.5 [3]. There are 18 (2×9) possible faulty circuits under the single-fault assumption. Table 1.1 [3] gives the truth table for the fault-free circuit and the faulty circuits for all possible single stuck-at faults. It should be noted that, rather than a direct short to a logic 0 or logic 1 value, the stuck-at fault is emulated by disconnection of the source for the signal and connection to a constant logic 0 or 1 value. The truth table entries where the faulty circuit produces an output response different from that of the fault-free circuit are highlighted in gray. As a result, the input values for the highlighted truth table entries represent valid test vectors to detect the associated stuck-at faults. With the exception of line d SAl, line e SAO, and line f SAl, all other faults can be detected with two or more test vectors; therefore, test vectors 011 and 100 must be included in

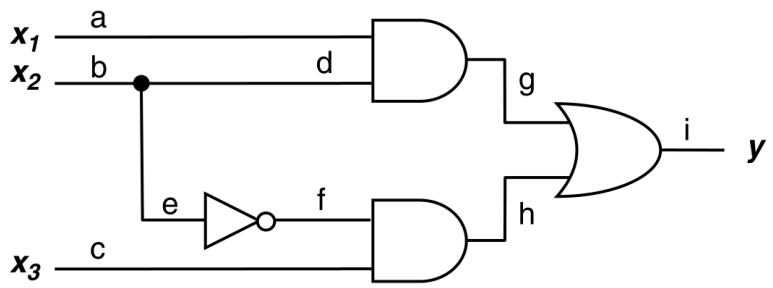


Figure 1.5: Example circuit [3].

any set of test vectors that will obtain 100% fault coverage for this circuit. These two test vectors detect a total of ten faults, and the remaining eight faults can be detected with test vectors 001 and 110. Therefore, this set of four test vectors obtains 100% single stuck-at fault coverage for this circuit.

Four sets of equivalent faults can be observed in Table 1.1. One fault from each set can be used to represent all of the equivalent faults in that set. Because there is a total of ten unique faulty responses to the complete set of input test patterns, then ten faults constitute the set of collapsed faults for the circuit. Fault collapsing typically reduces number of stuck-at faults by 50% - 60%.

Defects in VLSI devices can include opens and shorts in the wires that interconnect the transistors in a circuit. Opens in wires interconnecting transistors to form gates behave like transistor stuck-open faults. Opens in wires interconnecting gates to form circuit behave like stuck-at faults. Therefore, a set of test vectors that provide high stuck-at fault coverage and high transistor fault coverage will also detect open faults. It should be noted that resistive opens do not behave the same as a transistor or stuck-at fault but instead affect the propagation delay of the signal path. A short between two wires is known as a bridging fault. There are various bridging fault models as shown in Figure 1.6 [3].

In the first bridging fault model proposed, the logic value of the shorted nets was modeled as a logical AND or OR of the logic values on the shorted wires. This model is referred to as the wired-AND/wired-OR bridging fault model. The wired-AND/wired-OR bridging fault model was originally developed for bipolar VLSI and does not accurately reflect the behavior of bridging faults typically found in CMOS devices. Therefore, the dominant bridging fault model was proposed for CMOS VLSI where one driver is assumed to dominate the logic value on the two shorted nets. The dominant bridging fault model does not accurately reflect the behavior of a

Table 1.1: Truth table for fault-free and faulty circuits of Figure 1.5 [3].

$x_1x_2x_3$	000	001	010	011	100	101	110	**111**
y	0	1	0	0	0	1	1	1
a SA0	0	1	0	0	0	1	0	**0**
a SA1	0	1	1	1	0	1	1	**1**
b SA0	0	1	0	1	0	1	0	**1**
b SA1	0	0	0	0	1	1	1	**1**
c SA0	0	0	0	0	0	0	1	**1**
c SA1	1	1	0	0	1	1	1	**1**
d SA0	0	1	0	0	0	1	0	**0**
d SA1	0	1	0	0	1	1	1	**1**
e SA0	0	1	0	1	0	1	1	**1**
e SA1	0	0	0	0	0	0	1	**1**
f SA0	0	0	0	0	0	0	1	**1**
f SA1	0	1	0	1	0	1	1	**1**
g SA0	0	1	0	0	0	1	0	**0**
g SA1	1	1	1	1	1	1	1	**1**
h SA0	0	0	0	0	0	0	1	**1**
h SA1	1	1	1	1	1	1	1	**1**
i SA0	0	0	0	0	0	0	0	**0**
i SA1	1	1	1	1	1	1	1	**1**

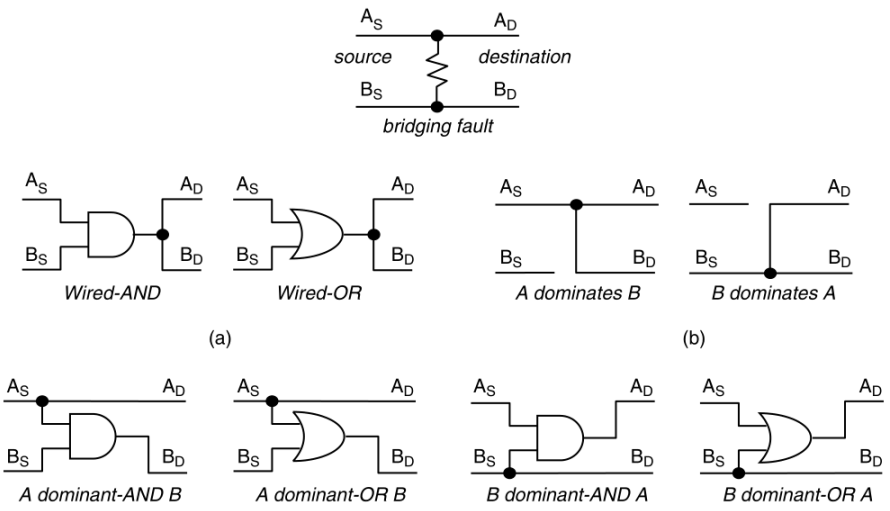


Figure 1.6: Bridging fault models [3].

resistive short in some cases. Thus, a new bridging fault model has been proposed, referred to as the dominant-AND/dominant-OR bridging fault. In this case, one driver dominates the logic value of the shorted nets but only for a given logic value. While there are four fault types to evaluate for this fault model, as opposed to only two for the dominant and wired-AND/wired-OR models, a set of test vectors that detect all four dominant-AND/dominant-OR bridging faults will also detect all dominant and wired-AND/wired-OR bridging faults at that fault site.

At the switch level, a transistor can be stuck-open or stuck-short, also referred to as stuck-off or stuck-on, respectively. Consider the two-input CMOS NOR gate shown in Figure 1.7 [3] and suppose that transistor N_2 is stuck-open. When the input vector $AB = 01$ is applied, output Z should be a logic 0, but the stuck-open fault causes Z to be isolated from ground Vss. Because transistors P_2 and N_1 are not conducting at this time, Z keeps its previous state, either a logic 0 or 1. In order to detect this fault, an ordered sequence of two test vectors $AB = 00 \rightarrow 01$ is required. For the fault-free circuit, the input 00 produces $Z = 1$ and 01 produces $Z = 0$.

But, for the faulty circuit, while the test vector 00 produces $Z = 1$, the subsequent test vector 01 will retain $Z = 1$. Thus, a stuck-open fault requires a sequence of two vectors for detection rather than a single test vector for a stuck-at fault. Stuck-short faults, on the other hand, will produce a conducting path between V_{DD} and V_{SS} . For example, if transistor N_2 is stuck-short, there will be a conducting path between V_{DD} and V_{SS} for the test vector 00. This creates a voltage divider at the output node Z

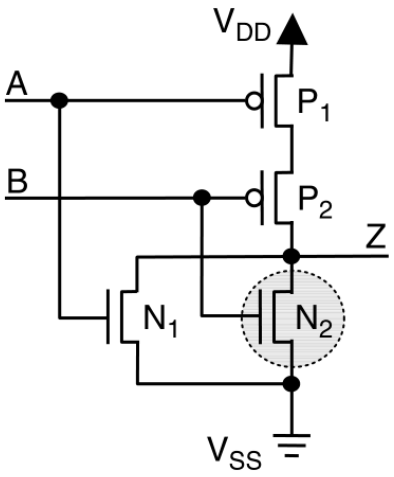


Figure 1.7: 2-input CMOS NOR gate [3].

Table 1.2: Truth table for fault-free and faulty circuits of Figure 1.7 [3].

AB	00	01	10	**11**
Z	1	0	0	**0**
N_1 stuck-open	1	0	Last Z	**0**
N_1 stuck-short	I_{DDQ}	0	0	**0**
N_2 stuck-open	1	Last Z	0	**0**
N_2 stuck-short	I_{DDQ}	0	0	**0**
P_1 stuck-open	Last Z	0	0	**0**
P_1 stuck-short	1	0	I_{DDQ}	**0**
P_2 stuck-open	Last Z	0	0	**0**
P_2 stuck-short	**1**	**I_{DDQ}**	**0**	**0**

where the logic level voltage will be a function of the resistances of the conducting transistors. This voltage may or may not be interpreted as an incorrect logic level to the output node Z. However, stuck-short transistor faults may be detected by monitoring the power supply current during steady state, referred to as I_{DDQ}. This technique of monitoring the steady-state power supply current to detect transistor stuck-short faults is referred to as I_{DDQ} testing.

The fault equivalence associated with the transistors can be seen in table in Table 1.2 [3], which gives the behavior of the fault-free circuit and each of the 8 possible faulty circuits under the single-fault model. Note that table entries labeled "last Z" indicate that the output node will retain its previous value and would require a two-

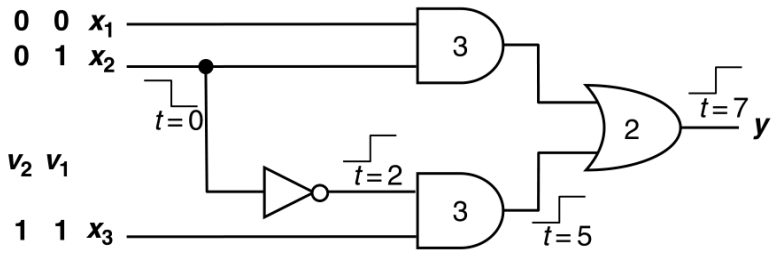


Figure 1.8: Path-delay fault test [3].

test vector sequence for detection. Similarly, entries labeled I_{DDQ} indicate steady-state power supply current monitoring. Because both N_1 and N_2 stuck-short faults as well as P_1 and P_2 stuck-open faults can be tested by the same test set, the collapsed fault count is 6. Fault collapsing typically reduces number of transistor faults by 25% to 35%.

Fault-free operation of a logic circuit requires not only correct logic function but also propagation of a signal along a given path in specific time limit. In case of excessive delay, a delay fault should be considered. There are different delay fault models. In the gate-delay and the transition-delay fault models, a delay fault occurs when the time interval taken for a transition from the gate input to its output exceeds its specified range. The other model is the path-delay fault model, which considers the cumulative propagation delay along a signal path through the CUT. The path delay comprises from the sum of all gate delays along the path. Thus, the path-delay fault model seems to be more practical for testing than the gate-delay fault or the transition fault model. A critical problem encountered when dealing with path-delay faults is the large number of possible paths in practical circuits. As with transistor stuck-open faults, delay faults require an ordered pair of test vectors to sensitize a path through the logic circuit and to create a transition along that path in order to measure the path delay.

For example, consider the circuit in Figure 1.8 [3]. The two test vectors, V_1 and V_2 , shown in the figure are used to test the path delay from input x_2 . Assuming the transition between the two test vectors occurs at time $t = 0$, the resulting transition propagates to the output y, through the circuit with the fault-free delays at time $t = 7$. A delay fault along this path would create a transition at some later time, $t > 7$. Note that such a measurement could require a high-speed, high-precision test machine. With decreasing feature sizes and increasing signal speeds, the problem of modeling

gate delays becomes more difficult. When the deep submicron technologies are used, the portion of delay contributed by gates reduces while the delay due to interconnect becomes dominant. In addition, if the operating frequencies also increase with scaling, then the on-chip inductances can play a role in determining the interconnect delay for long wide wires, such as those in clock trees and buses.

The use of nanometer technologies increases cross-coupling capacitance and inductance between interconnects, leading to severe crosstalk effects that may result in improper functioning of a chip. Crosstalk effects can be separated into two categories:

- Crosstalk glitch is a pulse that is provoked by coupling effects among interconnect lines.
- Crosstalk delay is a signal delay that is provoked by the same coupling effects among interconnect lines, but it may be produced even if line drivers are balanced but have large loads.

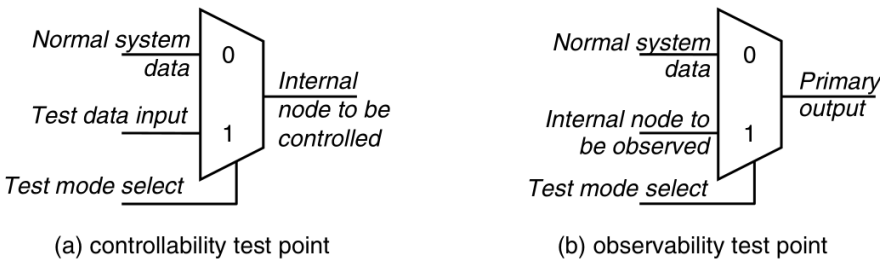
These fault models target high density RAMs. A pattern sensitivity fault means that the content of a memory cell is affected by contents of neighboring cells. A coupling fault results when a transition in one cell causes the content of another cell to change. It is necessary when testing memories to add tests for pattern sensitivity and coupling faults in addition to other fault models.

The improvement of fault coverage for structural tests can improve the yield, but there are still practical limits that do not allow a process completely without defects. Even if we could test against all known fault models, there would be still defects. The set of these remaining defects are called non-modeled defects.

1.2.5 Design for Testability

Automatic test equipment (ATE) is a computer-controlled equipment used in Wafer level, Package level and PCBs. Test patterns are applied to the CUT and the output responses are compared to stored responses for the fault-free circuit.

Automatic Test Pattern Generation (ATPG) refers to algorithms able to generate a sequence of test vectors for a given circuit based on specific fault models. A common approach, adopted by many ATPG tools, is to start from a random set of test patterns. Fault simulation then determines how many of the potential faults are detected. With the fault simulation results used as guidance, additional vectors are generated



(a) controllability test point (b) observability test point

Figure 1.9: Ad hoc DFT test points using multiplexers [3].

for hard-to-detect faults to obtain the desired or reasonable fault coverage. Fault simulation time can be proved significant, due to large number of faults to emulate. However, it can be reduced by implementing parallel, deductive, and concurrent fault simulation.

When testing is considered early in the design flow, then the resulting design is called ”Design for Testability (DFT)”. It helps to approach those parts of a digital circuit that will be most difficult to test and assists in test pattern generation for fault detection. DFT improves the controllability and/or observability of internal nodes of a chip or PCB. DFT techniques fall into one of the following three categories:

- Ad hoc DFT techniques.
- Scan design techniques.
- Built-in self-test (BIST).

The goal of Ad hoc DFT techniques is to target only those portions of the circuit that would be difficult to test and to add circuitry to improve the controllability and/or observability. Ad hoc techniques typically use test point insertion to access internal nodes directly. An example of a test point is a multiplexer inserted to control or observe an internal node, as illustrated in Figure 1.9 [3].

In a flip-flop-based scan design, testability is improved by adding extra logic to each flip-flop in the circuit to form a shift register, or scan chain, as illustrated in Figure 1.10 [3]. During the scan mode, the scan chain is used to scan in a test vector to be applied to the combinational logic. During one clock cycle in the system mode of operation, the test vector is applied to the combinational logic and the output responses are clocked into the flip-flops. The scan chain is then used in the scan mode to shift out the combinational logic output response to the test vector, while shifting in the next test vector to be applied.

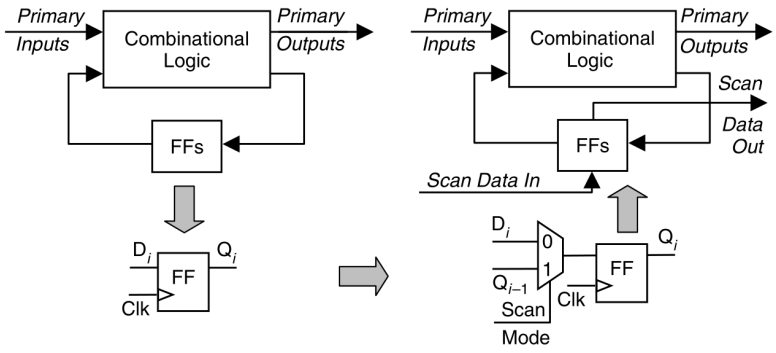


Figure 1.10: Transforming a sequential circuit for scan design [3].

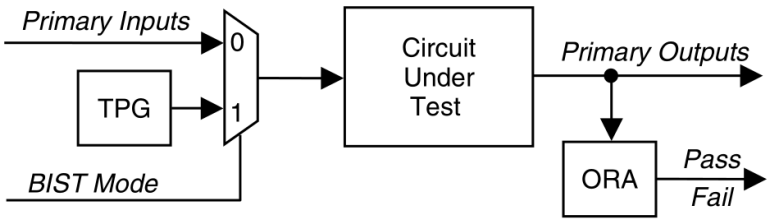


Figure 1.11: Basic BIST architecture [3].

Built-in self-test integrates a test-pattern generator (TPG) and an output response analyzer (ORA) in the VLSI device to perform testing internal to the IC, as shown in Figure 1.11 [3]. Since the test circuitry has been incorporated into the CUT, BIST can be used at all levels of testing, from wafer through system-level testing.

1.3 SoC Testing

1.3.1 The Basics

As the technology improves and the transistors are becoming constantly smaller, many new nanotechnologies and circuit design techniques are developed and adopted, which introduce new test challenges. These challenges are composed of a full spectrum of test technology trends crucial for nanometer designs, such as:

- Developing new DFT.
- Developing Design for Manufacturability (DFM) methods for digital and analog circuits, Microelectromechanical Systems (MEMS) and sensors.
- Developing the means to reduce manufacturing test costs.

- Enhance device reliability and yield.

- Developing techniques to facilitate defect and failure analysis.

The design of SoCs permits to apply modular testing on them. Test time is reduced and the power consumption is controlled, as the cores of the SoCs are only activated during their test. The test designer has the ability to plan the test order of the cores, in order to meet the test time and power constraints. Each embedded core in an SoC must be transformed to a testable unit, by using a core test wrapper. The wrapper isolates the core so that it can act as a stand-alone test unit and defines the interface between the core and the TAM, so that test access can be efficient.

The SoC industry mainly uses the core test wrapper described by IEEE 1500 Standard for Embedded Core Test (SECT) [24]. IEEE 1500 Standard is similar to 1149.1 [25], but it also provides parallel access capability for a core. As a result, test time for an SoC can be improved. Additionally, in contrast to 1149.1, where control signals are mainly generated by a finite state machine that is controlled by a single input, the 1500 Standard allows the direct application of the control signals directly to a core, thus providing more test flexibility.

The testable units (wrapped cores) in an SoC do not have direct access to SoC pins. In order to transport test data (test vectors) and test responses, to and from embedded cores, TAMs are necessary (Figure 1.14). Typically, the TAM resources are shared among different cores, in order to reduce the DFT structures cost. TAM resources sharing may lead to test conflicts, which are resolved by using TAM optimization and test scheduling techniques. The scan chains at each wrapper core are formed into wrapper chains, and given the test vectors, each wrapper core is associated with a test time. The objective of test scheduling is to maintain the overall test time as low as possible.

1.3.2 Test Wrapper

Cores can be deeply or hierarchically embedded in the SoC requiring a TAM for each core [4]. The diversity of the cores, provided by different vendors, results to different types of tests and test requirements. In the case of intellectual property (IP) cores, there may be little, if any, detailed information about the internal structure of the core. As a result, boundary scan alone does not provide a complete solution to the

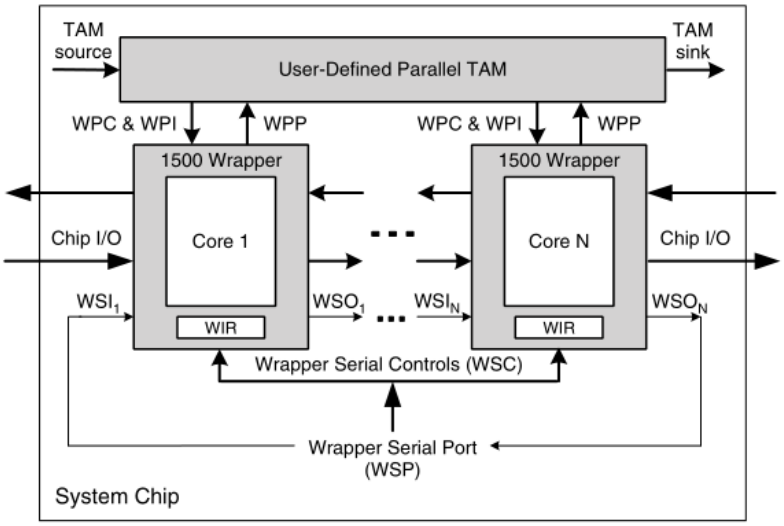


Figure 1.12: Overall architecture of a system per the IEEE 1500 standard [4].

problem of testing the cores in an SoC. Therefore, the IEEE 1500 standard [24] was introduced to address the problems associated with testing SoCs.

The architecture of an SoC with N cores, each one wrapped by an IEEE 1500 wrapper, is shown in Figure 1.12, and the structure of a 1500 wrapped core is presented in Figure 1.13. The wrapper serial port (WSP) is a set of I/O terminals of the wrapper for serial operations, which consists of the wrapper serial input (WSI), the wrapper serial output (WSO), and several wrapper serial control (WSC) terminals. Each wrapper has a wrapper instruction register (WIR) to store the instruction to be executed in the corresponding core, which also controls operations in the wrapper including accessing the wrapper boundary register (WBR), the wrapper bypass register (WBY), or other user-defined function registers. The WBR consists of wrapper boundary cells (WBCs) that can be as simple as a single storage element (flip-flop for observation only) or a complex cell with multiple storage elements on its shift path. The WSP supports the serial test mode (TM) similar to that in the boundary-scan architecture, but without using a Test Access Port (TAP) controller. In addition to the serial TM, the IEEE 1500 standard also provides an optional parallel TM with a user-defined, parallel TAM. Each core can have its own Wrapper Parallel Input (WPI), Wrapper Parallel Output (WPO), and Wrapper Parallel Control (WPC) signals.

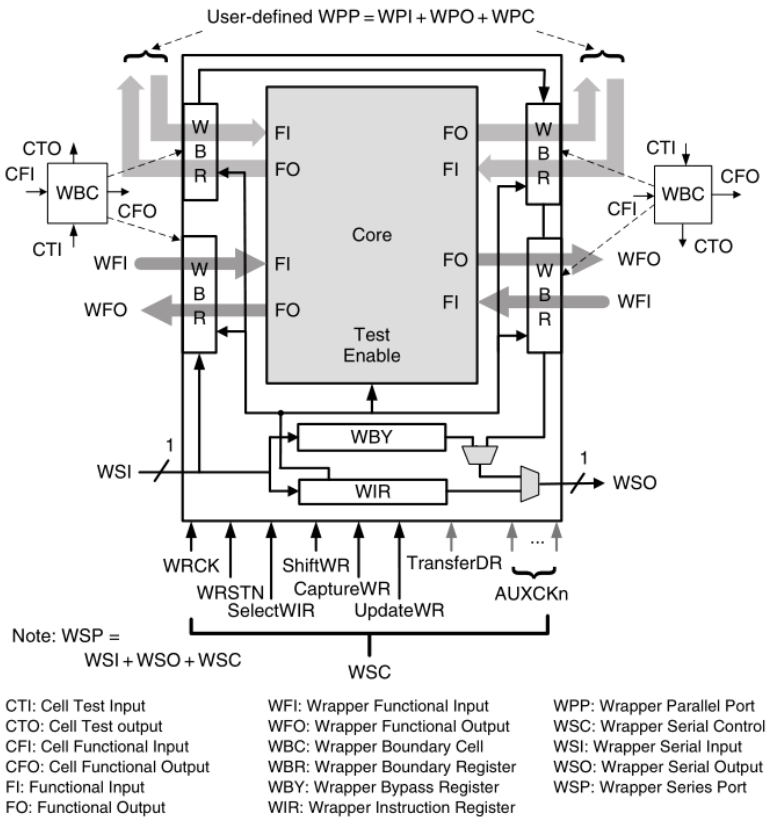


Figure 1.13: A core with the IEEE 1500 wrapper [4].

1.3.3 Test Access Mechanism

The testable units (wrapped cores) in an SoC do not have direct access to SoC pins. In order to transport test data (test vectors) and test responses, to and from embedded cores, TAMs are necessary. In Figure 1.14 are presented the basic types of TAMs. In the multiplexing and daisy-chain architectures, all cores get access to the total available TAM width, while in the distribution architecture, the total available TAM width is distributed over the cores. In the multiplexing architecture, only one core wrapper can be accessed at a time, resulting to only serial schedules. In the multiplexing architecture it is also hard to test the circuitry and wiring in between cores, as the interconnect test requires simultaneous access to multiple wrappers. On the other hand, daisy-chain and distribution TAMs allow both serial and parallel test schedules, and they also support interconnect testing.

The test bus architecture [26] (Figure 1.15) is a combination of the multiplexing and distribution architectures. A single test bus is the same as what is described by the multiplexing architecture, where cores connected to the same test bus can only be

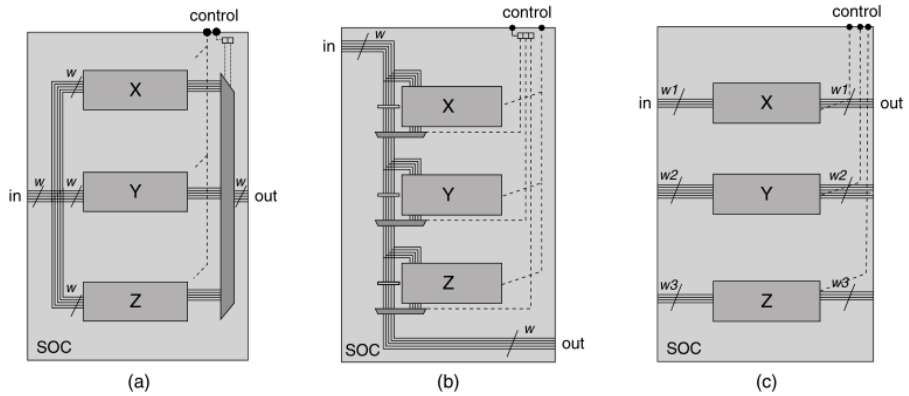


Figure 1.14: The (a) multiplexing, (b) daisy-chain, and (c) distribution architectures [5].

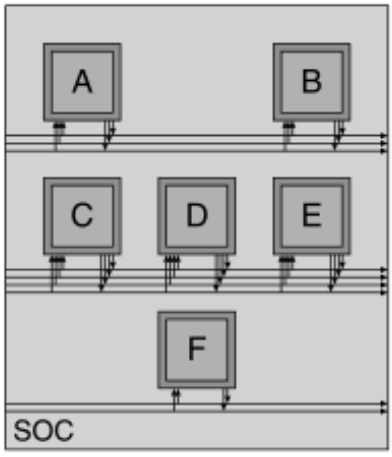


Figure 1.15: Test bus architecture [4].

tested sequentially. The test bus architecture allows for multiple test buses on one SoC that operate independently, as in the distribution architecture. Cores connected to the same test bus suffer from the same drawback as in the multiplexing architecture (their wrappers cannot be accessed simultaneously) making core-external testing difficult or impossible.

1.4 Software-Based Self-Testing

1.4.1 The Basics

It is clear that SoC has become the only solution for modern chip manufacturing as it meets the demands of the market for devices with rich functionalities, high porta-

bility, and low power consumption. As the trend of limited accessibility to individual components, increased operating frequencies, and shrunken feature sizes continues, testing will face a whole new set of challenges [4].

The difficulties in generating functional test patterns to reduce a chip's defect level and test cost contribute to the increasing cost of testing. DFT and BIST are two approaches which target to reduce the test cost. BIST empowers the IC by enabling at-speed test signals to be analyzed on-chip using an embedded hardware tester. BIST achieves greater testing accuracy and eliminates the need for high-speed external testers.

Existing BIST techniques are based on structural BIST. Although the most common scan-based BIST techniques [27, 28, 29, 3] achieve high test quality, the required circuitry to realize the embedded hardware tester bring along nontrivial area, performance, and design time overheads. Structural BIST also suffers from the problem of elevated test power consumption as test patterns are less correlated spatially or temporally than functional patterns, resulting in higher switching activity. Existing structural BIST suffers from various complex timing issues related to multiple clock domains, multiple frequencies, and test clock skews that must be resolved for timing related testing to be effective.

Software-Based Self-Testing (SBST) is a promising test solution. In SBST, memory blocks that facilitate SBST are tested first. Then, processors, DSPs and other on-chip programmable components are self-tested. Finally, these programmable components are configured as an embedded software tester to test on-chip global interconnects and other nonprogrammable components [4]. SBST is sometimes referred to as functional self-testing, instruction-based self-testing, or processor-based self-testing.

The SBST concept is presented in Figure 1.16 [4] using a bus-based SoC. The central processing unit (CPU) accesses the system memory via a shared bus, and all IP cores are connected to the system bus via a virtual component interface (VCI) [30]. The VCI simply acts as the standard communication interface between the core and the shared bus. Each core is surrounded by a test wrapper, in order to support the self-test methodology, which contains the required test support logic to control scan chain shifting as well as buffers to store scan data and support at-speed testing.

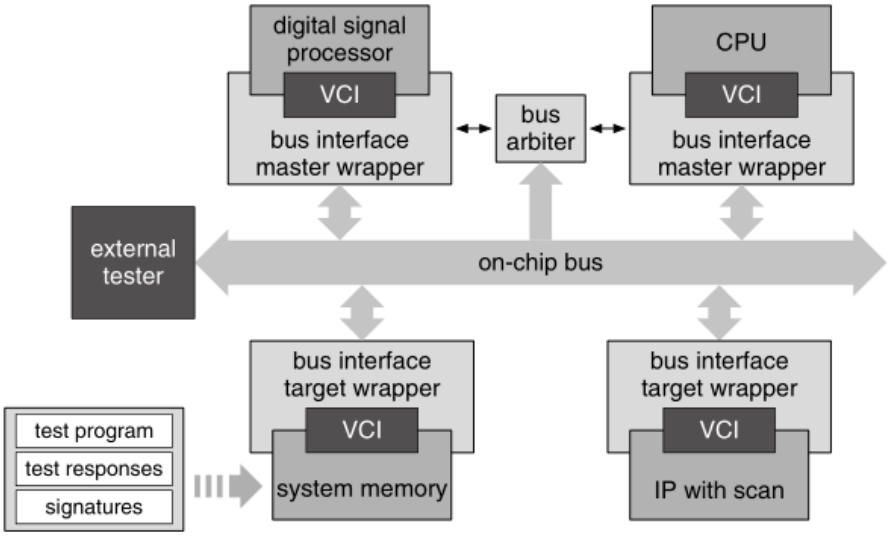


Figure 1.16: A software-based self-testable SoC [4].

1.4.2 SBST Flow

The SBST flow consists of the following steps:

1. **Memory self-testing**: The memory block (system or processor cache memory) that stores the test programs, test responses, and signatures is tested and repaired if necessary.

2. **Processor self-testing**: During processor self-testing, the external tester first loads the memory with the test program and the signatures. Then, the processor tests itself by executing the test program, aiming at the fault models of interest. The test program responses are written to the memory and later compared to the stored signatures to make the pass/fail decision.

3. **Global interconnect testing**: Predetermined patterns that activate the defects of interest are transmitted between pairs of cores among which data or address transmission exists. The responses captured in the destination cores are then compared to the stored signatures.

4. **Testing nonprogrammable cores**: The embedded processor controls the test pattern generation, test data transportation, and response analysis programs. For analog/mixed-signal cores, processor and DSP cores may be employed to perform the required pre- and postprocessing DSP procedures.

1.4.3 SBST vs BIST

SBST handles testing as a system application, while structural BIST places the system in nonfunctional test mode. SBST has the following advantages [4]:

1. The need for DFT circuitry is minimized. Structural BIST needs wrapper cells and necessary logic to control or observe the system's external input/output (I/O) ports that are not connected to low-cost testers. On the other hand, additional DFT techniques can be employed if SBST does not achieve the desired fault coverage.

2. The performance requirement for the external tester is reduced as all high-speed transactions occur on-chip and the tester's main responsibility is to upload the test program to the system memory and download test responses after self-testing.

3. Performing test pattern application and response capture on-chip achieves greater accuracy than that obtainable with a tester.

4. Because the system is operated in functional mode while executing the test programs, there is no excessive test power consumption and potential overkill problems as in structural BIST.

One major concern of SBST is its fault detection efficiency. If a fault can be detected only by test patterns that cannot be realized by any instruction sequence, then the fault is redundant and there is no need to test for this type of fault during manufacturing test, even though we may still want to detect and locate these faults during silicon debug and diagnosis for manufacturing process improvement. For an IP core that is not self-testable, structural BIST may be used to reach the desired level of fault coverage. In this case, using the processor as the TPG and ORA gives the flexibility of combining multiple test strategies to achieve the desired fault coverage. This is achieved by just altering the corresponding programs or parameters without any hardware modification. In [31], the authors discuss mixed-mode pattern generation for random and deterministic patterns using embedded processors. After identifying the best pattern generation scheme the test program is synthesized accordingly without the need to alter any BIST hardware.

1.5 3D-SoCs

Three-Dimensional Integrated Circuits (3D-ICs) is a popular research field for the hardware and testing community. 3D-ICs overcome the limitations of traditional 2D-ICs and seem to secure the continuation of Moore's Law for the near future. 3D-ICs prevail over traditional 2D integration techniques on various aspects [32]:

1. Reduction on global interconnect.
2. Higher packing density and smaller footprint due to the addition of a third dimension to the conventional two-dimensional layout.
3. Higher performance due to reduced average interconnect length.
4. Lower interconnect power consumption due to the reduction in total wiring length.
5. Support for realization of mixed-technology chips.

However, we have to address a variety of design, manufacturing, packaging, and testing issues before cost-effective, high-volume production can be achieved.

1.5.1 Manufacturing

There are two broad categories of 3D-IC manufacturing techniques, monolithic and die stacking (Figure 1.17). For the monolithic manufacturing process using epitaxy, multiple device layers are grown on the same wafer in a serial manner. Once a layer of devices and their associated interconnect are completed, an isolation inter-level dielectric layer can be deposited and polished to allow another layer of devices and interconnect to continue to grow vertically. To electrically connect devices across separate processed layers, 3D vias are etched through the isolation layer, and metal fillings are deposited. The same process is repeated to fabricate a 3D-IC consisting of multiple layers of devices.

The other 3D integration technique is to stack individual 2D die layers vertically, minimizing the impact of altering existing manufacturing technology and equipment. With 3D die stacking, the candidate dies to be integrated onto the same package can be designed and manufactured separately, just as they are with a regular, existing 2D planar process, with additional manufacturing processes of substrate thinning and

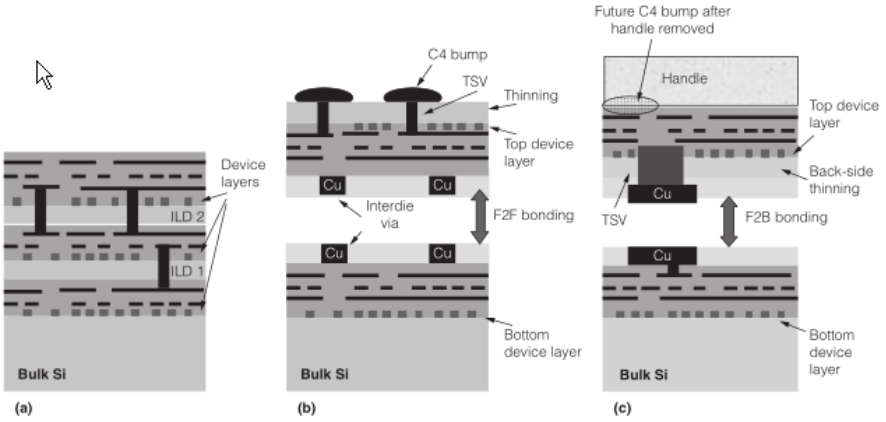


Figure 1.17: 3D-IC fabrication methods: (a) monolithic, (b) face-to-face, (c) face-to-back [6].

through-silicon via (TSV) filling, if needed. Note that some times TSVs are referred as elevators in the bibliography. Then they are bonded together by precise alignment of inter-die vias and the application of thermo-compression. In general, die stacking presents three integration alternatives: wafer to wafer, die on wafer, and die on die, each with their respective pros and cons from a cost or yield perspective [6]. Die stacking seems to have more advantages than monolithic technique and therefore we will focus on this area.

A number of methods have been proposed for the interconnection of the stacked dies, including wire bonding, microbump, contactless approaches, and TSVs [33]. Wire bonding makes connections between the board and stack or between dies themselves, though wires can only be on the periphery of the stack. Wire bonding thus suffers from low density, a limit on the number of connections that can be made, and the need for bonding pads across all metal layers due to the mechanical stresses of the external wires. Microbumps are small balls of solder or other metals on the surface of the die that are used to connect dies together. They have both higher density and lower mechanical stress than wire bonding. Microbumps do not, however, reduce parasitic capacitances because of the need to route signals to the periphery of the stack to reach destinations within it. Contactless approaches include both capacitive and inductive coupling methods. Though resulting in fewer processing steps, other manufacturing difficulties and insufficient densities limit these methods. TSVs hold the most promise as they have the greatest interconnect density, even though they require more manufacturing steps [32]. In Figure 1.18 a 3D-IC with silicon interposer

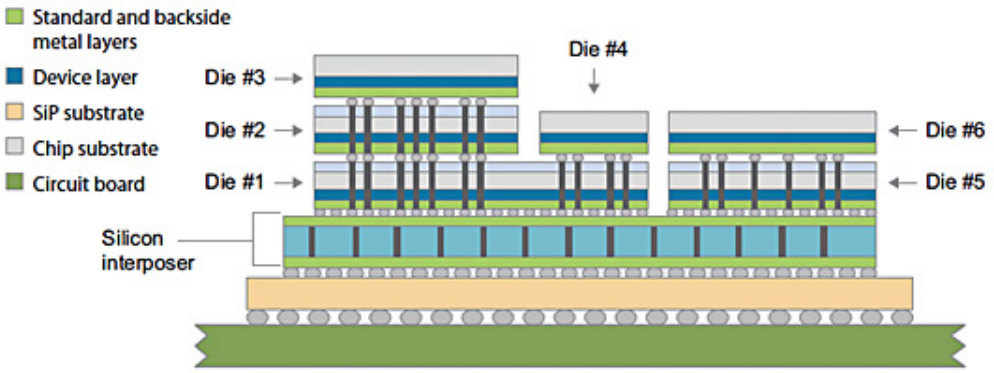


Figure 1.18: 3D-IC with silicon interposer layer, TSVs and 6 dies.

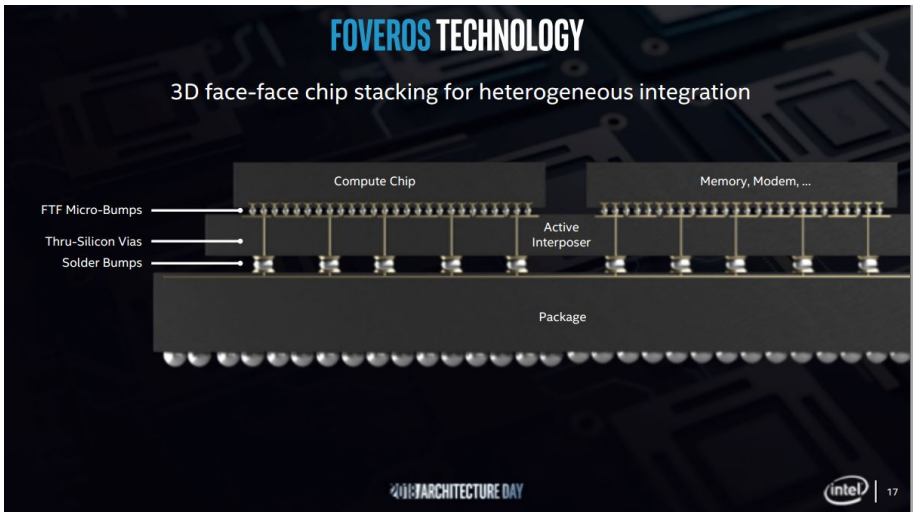


Figure 1.19: Foveros packaging.

layer, TSVs and 6 dies is presented, where the Inter-die communication is realized by TSVs.

In 2019, Intel has introduced Foveros, an advanced 3D face-to-face die stacking packaging process technology, which uses TSVs. Foveros packaging is designed to incorporate two or more chiplets assembled together. It comprises a base logic die on top of which sit additional active components such as another logic die, memory, FPGA, or even analog/RF (Figure 1.19). Intel announced Lakefield SoC (Figure 1.20) in January 2019 at CES, which is built using Intel's Foveros. Lakefield combines a powerful Sunny Cove core with four, lower-power Atom processor cores. Its design aimed at efficiency and physical space saving, allowing Lakefield to reside in smaller devices [34].

Despite various advantages, TSV-based 3D-ICs are subject to severe thermo-mecha-

Figure 1.20: Lakefield.

nical reliability hazards due to the coefficients of thermal expansion mismatch between TSV and silicon substrate. Under normal operations, the 3D-IC experiences heavy thermal cycles and thus induces large thermo-mechanical stress. Consequently, the stress may initiate micro cracks from the interface between a TSV and its dielectric liner, and further propagates them on the silicon substrate. To eliminate critical cracks, the simplest solution would be to make the keep-out-zone (KOZ) of TSVs large enough to cover the maximal possible length any crack might grow. However, the length a crack can grow from an originating TSV depends heavily on the locations of the nearby TSVs due to the superposition of thermo-mechanical stress, and its value can be large or even difficult to calculate when complicated TSV placement structures present. On the other hand, as the probability of crack initiation is low, it would be a waste of chip area to set the KOZ to the extreme value for every TSV. Alternatively, it is more economical to make the TSV KOZ reasonably small and filter out those chips with critical cracks during testing [35].

1.5.2 3D Testing

The usual testing sequence for 3D-ICs is pre-bond, mid-bond and post-bond testing. In the first stage, we have to test the individual 2D die layers, before stacking them together. Pre-bond testing can be performed either before and/or after wafer thinning. Afterwards, we start stacking dies on top of each other and we test the partial stack that is created. It is not necessary to test after every added die, but we can choose the stages we want to test, depending on our goal on test cost. In the end, after the stack is completed, the last part of testing follows, post-bond testing, to find out if any defects were caused during the stacking process. External test access is achieved

through probing, typically at wafer level. Once the stack is packaged, a final packaged test can be performed. External test access for the final test is typically via a test socket through the package pins.

A pre-bond test should primarily focus on defects in the die-internal (CMOS, DRAM, etc.) circuitry, and only optionally be extended for TSV defects. If performed on an original (still thick) wafer, the pre-bond test has the benefit of conventional wafer handling, but possible wafer-thinning defects are not covered. Moreover, testing for TSV defects on not-yet-thinned wafers is imperfect, as one side of the TSV is still buried in thick substrate. Therefore, there is only single-sided test access and this requires dedicated DFT and test methods. On the other hand, pre-bond testing on thinned-down wafers, which allows to cover defects induced by wafer thinning and actually test through TSVs, requires probe access on thinned wafers, which brings about a whole new set of challenges [36].

Mid- and post-bond tests should primarily focus on testing the newly formed TSV-based interconnects and only re-test that die-internal circuitry if stacking operations are likely to damage it. The final test is the last safety net before the packaged die stack is shipped to the customer. Hence, the test infrastructure should be such that all components and interconnects of the die stack can be (re-)tested if required. There is not a unique test flow, as testing should provide sufficient product quality at affordable costs for the application market addressed. The general desire is to detect defects as early as possible, in order to prevent faulty components to move forward in the production flow. However, early tests have the drawback that passing components might get damaged afterwards due to downstream (thinning, stacking, packaging) operations and hence require re-testing. The manufacturing costs and yields are in complex ways related with test flows and associated costs, and hence test cost modeling is required to determine the best test flow for a given product.

1.6 Thesis Organization

Chapter 2 presents the state of art in 3D SoC testing and SBST, two topics that are closely related to this research work. Various aspects of 3D SoC testing are presented, such as pre-bond, post-bond, TSVs testing etc., by describing existing works in detail. Concerning the Software-Based Self-Testing test scheduling, the basic principles are

analyzed initially, and existing methodologies are described.

Chapter 3 presents thoroughly and in depth the research work. It starts with the research directions and then, the main research objectives are defined. Afterwards, each proposed method is presented in detail, accompanied by explanatory examples.

Chapter 4 presents the tools and the work-flows that have been developed during this research work. This framework provided with test environments that enabled the efficient and reliable deployment and execution of experiments upon the methods discussed in Chapter 3.

Chapter 5 includes a set of carefully selected, evaluation-based experimental procedures that prove the innovation and the added value of the proposed methods.

Finally, Chapter 6 provides conclusions and directions for future work. The contributions outlined in Chapters 3, 4 and 5 resulted in original work published, which is itemized in the end of the dissertation.

Chapter 2

Background

2.1 3D SoCs Testing

2.1.1 Pre-Bond Testing

Pre-bond testing is necessary in order to identify the Known-Good-Dies (KGD). If a defected die was stacked and the defect was detected after the bonding, the whole stack would be useless. There is a large number of TSVs in a 3D-IC, one end of them is buried at the thick substrate and they have small pitch and density, making their pre-bond testing difficult. There are various TSV defect types and five of them can be detected at post-bond testing from errors in alignment, bonding or stress, while the rest should be explored prior to bonding [37]. BIST methods may be used for pre-bond testing as well as TSV probing. The partitioning of a circuit design among multiple layers of a 3D stack makes pre-bond testing difficult. There is the need to deal with partial logic and structures without complete functionality by innovations in DFT layout.

There are several key hardware components necessary to realize structural test of individual die layers pre-bond. These include a general test architecture, specialized scan registers, the necessary support nets (power, ground, and clocks), and an in-

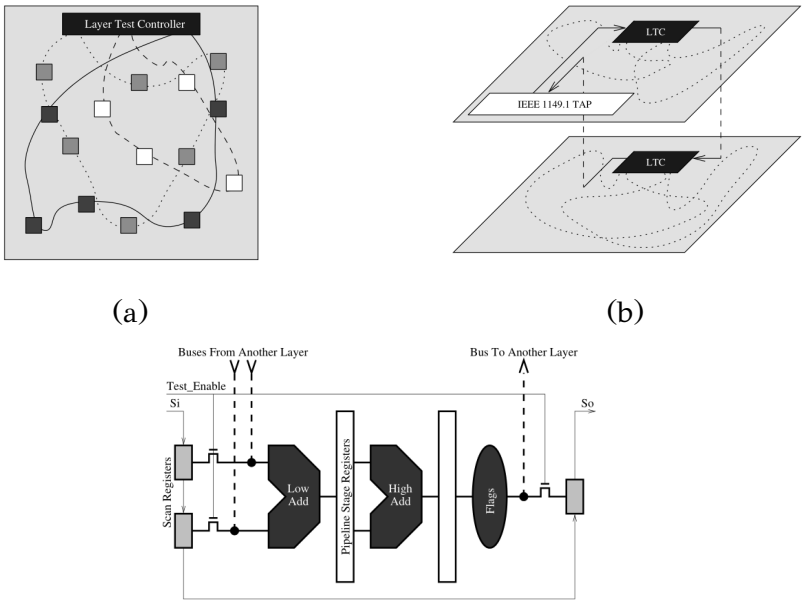


Figure 2.1: DFT technique for enabling pre-bond testability for 3D die stacked microprocessors [7].

terface to the outer world. In [7], Lewis & Lee propose and evaluate an applicable methodology to enable pre-bond testability for 3D die-stacked microprocessors for the first time, considering each layer as an isolated test island (Figure 2.1). TSVs cannot be tested by a scan chain, especially before wafer thinning and back metal patterning. In [38], Chen, Wu & Kwai proposed a method utilizing the parasitic capacitance of the TSV in order to detect the faulty TSVs with little area overhead from the testing circuit. Each TSV is treated as a DRAM cell and tested before bonding using sense amplification. In [39], the same team extended the previous proposal, developing two testing schemes while taking the effects of process variations into account. Panth & Lim in [40] were the first to explore the impact of scan chain TSVs on wirelength, power and voltage drop of 3D-ICs. They also explored the design options and requirements for power delivery during pre-bond testing, concluding that an increase at the number of scan TSVs up to a certain point, reduces both wirelength and power consumption.

In 3D technology, test pads occupy a much larger area when compared to TSVs and hence we can only fabricate a limited number of test pads for pre-bond testing. Therefore, it is necessary to take the pre-bond test-pin-count constraint into consideration during test planning. In [41], Jiang et al. design test architectures for pre-bond

tests and post-bond test separately so that this constraint can be satisfied in pre-bond tests. They propose optimization methods that allow us to share routing resources between pre-bond tests and post-bond test as much as possible and they show how to optimize test architectures to further reduce test access mechanism (TAM) routing cost with little impact on testing time. In [42], Li & Xiang propose a DFT scheme in order to reduce the number of wrapper cells needed for providing testability at the TSVs and the corresponding test data volume during pre-bond testing by reusing the Primary/Pseudo-Primary Inputs and Primary/Pseudo-Primary Outputs for different ends of the TSVs, respectively. In [43], Kumar et al. propose a DFT method for pre-bond testing of 3D-ICs using a hypergraph [44] based netlist partitioning scheme, achieving a significant reduction in the hardware required for scan-island based pre-bond testing. In [45], Agrawal & Chakrabarty showed that the general problem of minimizing the wrapper-cell count is NP-hard and they have shown how timing and layout information can be incorporated in the graph model to address the problem of increased capacitive load and delay on critical paths due to multiple flop reuse.

In [37], Noia & Chakrabarty presented a new technique for pre-bond TSV testing that is compatible with current probe technology and leverages the on-die scan architecture that is used for post-bond testing. It utilizes many single probe needle tips, each to make contact with multiple TSVs, shorting them together to form a single network, allowing the concurrent testing of many TSVs to reduce overall test time. Based on [37], the same authors presented an efficient algorithm, in [46], for designing parallel TSV test sessions such that test time is reduced and a given number of faulty TSVs within the TSV network can be uniquely identified under parallel test, resulting in the highest test cost reduction. In [47] Noia et al. extend the work on pre-bond TSV testing through probing [37], [46] by utilizing scan chains that can be reconfigured for full-scan pre-bond die logic test to allow scan-in and scan-out through TSVs.

2.1.2 Post-Bond Testing

New test architectures and optimization methods are required for post-bond testing. The stack may need to be tested multiple times during the bonding process to ensure that defects were not introduced to the stack during manufacturing steps between pre-bond testing and final test. When a test architecture already exists on each die

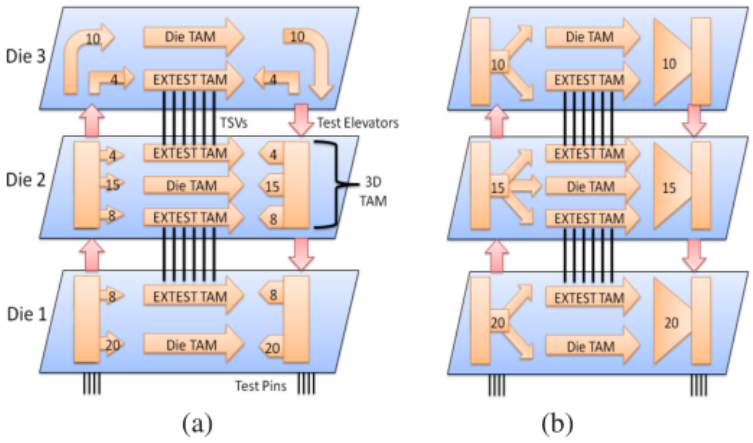


Figure 2.2: Example of a test architecture in a 3D-SIC including die-external tests [8].

of a 3D-IC, then we call them hard dies. Otherwise, we call them soft dies. IEEE 1500 wrapper is called thin wrapper, while a fat wrapper allows for core-external test (EXTEST) and core-internal test (INTEST) to run in parallel.

In [48], Jiang et al. proposed a TAM wire length minimization technique based on simulated annealing, allowing both pre-bond and post-bond tests. In [49], the authors propose a wrapper for entire die as opposed to only the embedded cores on a die, providing a uniform test interface for each die, supporting pre-bond die testing, post-bond stack testing and final packaged-product testing. Their architecture, based on IEEE 1500, reuses commonly encountered design-for-test structures within the various dies as much as possible. Noia, Chakrabarty & Marinissen, in [8], presented generalized optimization methods to minimize test time for a 3D-IC with hard dies, either for the final stack test or for any number of multiple test insertions during bonding. They presented optimization techniques that consider both die-internal and die-external (Figure 2.2) test for both fat and thin die wrappers, taking into account constraints on both test bandwidth and the number TSVs per die. The same people presented in [50], [51] and [52], including hard, soft and firm (a test architecture already exists for the die, but serial/parallel conversion hardware may be added to the die in order to reduce test-pin and TSV use and achieve better test resource allocation for stack testing) dies. In these works Integer Linear Programming (ILP) models were presented for test architecture optimization.

Working on 3D-IC with passive silicon interposer, Chi et al. presented in [53] a modular post-bond test strategy that enables testing of dies, interposer, and the micro-

bump interconnects in between them. A special constraint in this problem setting is that the passive silicon interposer does not contain active circuitry and therefore for DFT, they are restricted to adding interconnects only, as sparsely as possible, in order to keep the cost of the interposer low. In [54], Lin et al.presented a parametric delay test method. First, they approximate the propagation delay across a TSV with a resistive open fault and then make a test decision either based on a test threshold or through outlier analysis. The key of the method is a novel technique called variable output thresholding (VOT), implemented with an Ring Oscillator (RO) architecture consisting of two TSVs and a number of logic gates. In [55], Rajski & Tyszer introduce a new scan-based test infrastructure for TSVs and the corresponding post-bond test generation techniques capable of detecting and accurately identifying variety of single and multiple faults for TSVs in 3D stacked ICs. The proposed DFT environment is easy to automate and standardize, and offers quickly achievable complete fault coverage for a comprehensive list of failures even when using pseudo-random test patterns.

2.1.3 TSVs

In a 3D-IC the dies must be interconnected to one another. A number of methods have been proposed for this interconnection, including wire bonding, microbump, contactless approaches, and TSVs [33]. TSVs hold the most promise as they have the greatest interconnect density, even though they require more manufacturing steps [32]. TSVs are vertical metal interconnects that can be integrated into a substrate during manufacturing. In all TSV manufacturing approaches, the TSVs are initially buried in the substrate and need to be exposed, through a process called thinning, in which the substrate is ground away until the TSVs are exposed in a subsequent step. This step results in dies that are much thinner than conventional 2D substrates, and they are thus fragile and are commonly attached to carrier wafers during 3D integration. In order to be attached to other dies in a 3D stack, a die must go through alignment and bonding. During alignment, the dies are carefully placed such that their TSVs make direct connections to one another. The processes of alignment and bonding continue until all thinned dies are integrated into the 3D-IC [56].

Early in 2008, Loi et al. described the design of a defect-tolerant TSV-based multi-bit vertical link which enables significant yield improvement with respect to random

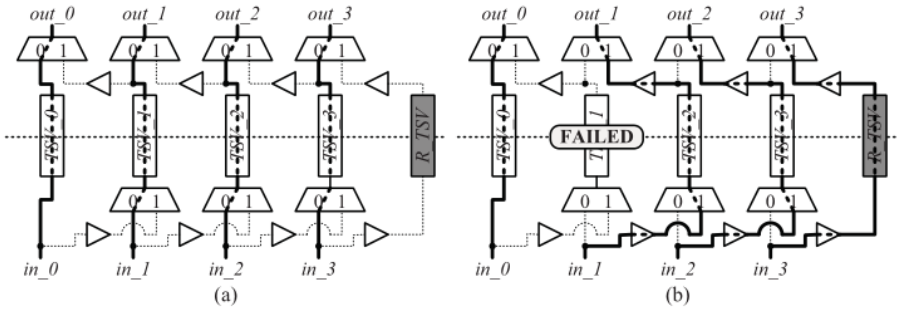


Figure 2.3: TSV recovery mechanism [9].

(complete or partial) open defects at an extremely low cost [57]. Their technique relies on redundancy and has minimal impact on the overall integrated system design and production test flows. In [58] Tsai et al. presented four self-test circuits capable of detecting pinholes in TSVs. In [9] Hsie et al. proposed a redundant TSV architecture with reasonable cost for ASICs, which can successfully recover most of the failed chips and increase the yield to 99.99% based on probabilistic models (Figure 2.3). In [59], another technique based on TSV redundancy was proposed in order to improve the yield of 3D-ICs. Zhao, Khursheed and Al-Hashimi partitioned regular and redundant TSVs into groups, where each group can have multiple spare TSVs and multiplexers are used to reroute signals through good TSV path, if the group has a defective TSV. They also model clustering defects (defects tend to cluster together to some extent rather than randomly distributed) in order to analyze their effect on yield.

Deutsch, Chakrabarty, Panth and Lim, in [60] focused on post-bond delay-fault testing of internal die logic in 3D-ICs and studied the impact of timing variations due to TSV stress on the quality of test patterns generated to screen small-delay defects. The impact of TSV stress on pattern effectiveness is quantified using the statistical delay quality level (SDQL) metric. One of the conclusions of [60] was that smaller KOZs are not an issue, as the ATPG flow with stress-aware models will still create high-quality tests, even though the circuitry is more affected in that case. Huang et al. in [61] proposed an on-chip measurement technique about small delay testing. Their technique, Variable Output Thresholding (VOT) is implemented on an RO architecture consisting of two TSVs and some logic gates. Metzler et al. proposed in [62] exploited mathematical models to express path delays as a function of physical and electrical factors to devise a relationship between delay variation and defect size. This metric computes a probability of detection for resistive open TSVs and allows us to sort defect sizes as detectable or not. Recently, Kuo et al. presented a novel test technique

for TSV-induced small delay faults for 3D-ICs, considering two effects of a defective TSV on nearby logic gates, mechanical stress and pinhole leakage.

In [63], Cheng et al. reduced testing TSVs as much as possible while keep testing time minimal, by converting the 3D wrapper optimization problem to its corresponding 2D one and adopt existing research efforts to tackle it. They proposed novel algorithms to determine how these wrapper chain components are connected to minimize the amount of TSVs. Kannan, Kim and Ahn presented in [64] an equivalent electrical circuit model for TSV in 3D-ICs, identified different defects that can affect their functional electrical performance and developed fault model for each one of them. They developed the multi-tone dither test technique to detect these faults in TSVs by integrating the fault models with the electrical model of TSV before measuring the peak-to-average ratio. Pasca et al. proposed in [65] an interconnect yield improvement solution based on Configurable fault-tolerant Serial Links (CSL). CSLs do not rely only on spares to achieve high interconnect yield, but on serialization and signal remapping on fault-free TSVs. Rashidzadeh presented three coupling techniques (inductive, radiative and capacitive) that can be utilized for contactless TSV probing [66]. He implemented and used a microscale probe as a contactless probe for TSV, supporting the small pitch and the high density requirements for TSV probing.

The TSV leakage test problem decides if there is excessive leakage current from the TSV to the substrate, degrading the TSV performance and maybe causing reliability issues. The leakage fault (most references in literature call it open or short fault) is often parametric and its detection involves a Leakage Test Threshold (LTT), defined as the leakage current value beyond which a TSV will be declared as faulty [67]. Open defects can be classified into hard open defects and soft open ones. In case of a hard open defect, an interconnect is divided into two parts completely and they are not connected each other. In case of a soft open defect, the parts are connected each other in part electrically [68].

Lin et al. presented a CAF-WAS (Charge-and-Float, Wait-and-Sample) technique, adapting to a wide range of die-to-die connections [67]. Their technique has the ability to perform leakage binning, by which one can approximate the leakage currents of the TSVs when the direct measurement by external ATEs may not be possible due to the too small size of the TSV. Huang et al. in [69] presented a self-timed timing control scheme in order to alleviate the timing skew problem, caused by imbalanced routing wire lengths extending the work presented in [67]. Resistive-

open defects cause unintended signal propagation delays, while open defects with high resistance lead to a floating end for a TSV, both of them affect chip functionality [70]. Therefore, Ye and Chakrabarty [70] used the Elmore delay model to model TSVs and analyzed the delay for TSVs, with or without defects. They presented a new method for detecting open and short defects using technique estimating the additional delay introduced due a resistive open defect as well as due to re-routing based on spare TSVs. To recover from open defects and increase the yield for TSV stacks, they presented an optimization method based ILP that allocates spares to functional TSVs.

Chi et al. in [71] proposed a novel DFT architecture, which focused on 3D-IC interconnects supporting testing, diagnosis, and repair. Their method can detect open and short defects and identify the faulty interconnects by applying proper test patterns and can repair faulty interconnects caused by open defects. Hashizume et al. proposed the electrical test method and a DFT method in [68]. They also examined feasibility of their electrical tests by Spice simulation and some experiments for a PCB circuit made of our prototyping IC that had been designed by the DFT method. Shih et al. presented a defect simulation and test generation flow to detect path delay faults induced by defective power TSV in [72].

We have already mentioned Ring Oscillators (ROs) in Subsection 2.1.2. The connection of two TSVs with some peripheral circuit to form an oscillation ring is a technique frequently used in TSV testing. You et al. in [73] proposed a method that can characterize the propagation delay across each TSV by enhancing existing RO based test concept with a new technique called sensitivity analysis. You et al. presented an improved method of [73] in [74]. Pai et al. proposed a unified RO-based horizontal/vertical interconnects test methodology for 3D-ICs to achieve interconnect reliability and yield with targets of interconnect faults under stuck-at and open fault models [75]. Deutsch and Chakrabarty proposed a method for non-invasive pre-bond TSV test using ROs and multiple voltage levels, without the need of TSV probing [76]. With their method, it is possible to detect resistive opens and leakage faults by measuring variations in the delay of nets connected to TSVs.

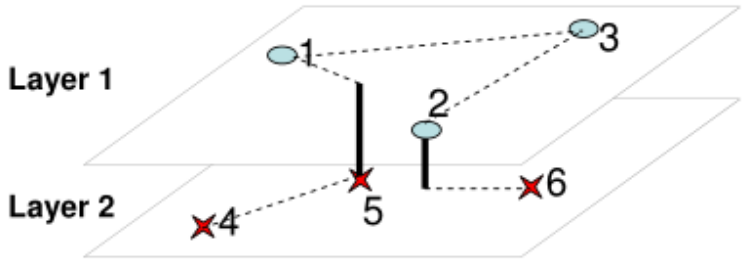


Figure 2.4: VIA3D approach [10].

2.1.4 TAM Architectures

In order to test every IC, we have to develop efficient TAMs. The same deal stands, also, for the 3D-ICs. The most common techniques are the wrapper cells (commonly based on IEEE Std 1500) and the scan chains, used in the 2D-IC technology, adapted to the 3D-IC conditions. These TAMs create area overhead, so there is the need to keep them as small as possible. Keep in mind that wrapper chains begin and end on the lowest layer, as all of the chip pins are at lowest layer.

Beginning from the work in [10], where Wu, Falkenstern and Xie first approached and investigated 3D scan chain design (Figure 2.4), many techniques were proposed on this direction. Lewis and Lee, in [77], investigated test strategies for circuit-partitioned 3D designs in which a functional unit can be partitioned into incomplete circuits across different die layers. They presented standard scan registers that can be integrated into the layer scan chains, allowing the ATE to directly test the circuit or initialize the registers for BIST. Noia, Chakrabarty and Xie studied the test-wrapper design for 3D SoC systems with true 3D cores [78]. The goal of wrapper optimization is to minimize the test time for each core and, therefore, they developed two faster heuristic approaches to produce near-optimal solutions.

In [79] proposed a test integration methodology for 3D-ICs with two test interfaces (compatible with the IEEE 1149.1) for supporting the pre-bond, known-good stack, and post-bond tests of 3D-ICs. Marinissen et al. proposed a 3D DFT architecture (based on a die-level test wrapper that should be included by the various die makers in the designs of the respective dies that together make up the stack) that services the test needs of die maker(s), stack maker, and stack user alike [11] (Figure 2.5). Their architecture supports pre-bond, post-bond and board-level interconnect testing, enabling a modular test approach. Chi et al. working on the same area, presented a

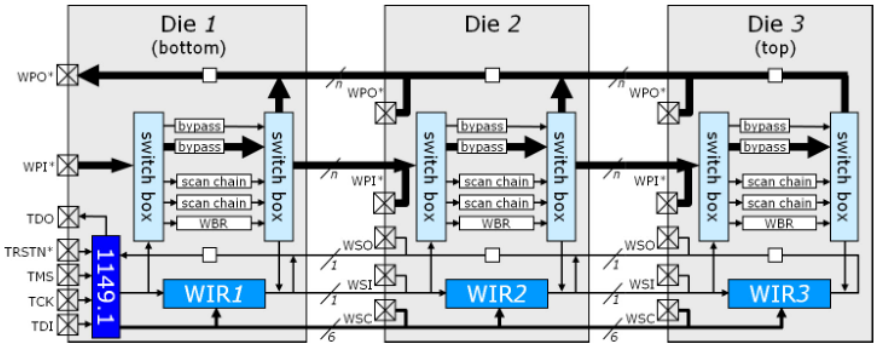


Figure 2.5: 3D-SIC DfT architecture for dies based on IEEE 1500 [11].

DFT architecture, using die-level wrappers, but for 3D-ICs with multiple towers [53]. We say that a 3D-IC consists of a single tower, when each layer contains exactly one die, otherwise it consists of multiple towers. Papameletis et al. also considered 3D-ICs with multiple towers, when they extended a DFT architecture and implemented in Cadence EDA tools [80].

Lewis et al. have made another step forward presenting a methodology for designing 3D test wrappers for embedded 3D IP cores [81]. They used the Best Fit Decreasing and Kernighan-Lin Partitioning heuristics to design flexible test wrappers that can adjust to varying test modes, achieving lower total test time for the circuit under test. Liao et al. proposed a flow for fast scan-chain ordering (formulated into a traveling salesman problem (TSP)) taking into account TSV constraints [82]. Noia and Chakrabarty move in a more algorithmic fashion using retiming techniques to recover the latency added by boundary scan cell bypass paths in a 3D-IC [83].

2.1.5 TAM Optimization

As it is already clear from the previous sections, TAMs are an important part of the testing process of the 3D-ICs. TAMs and test wrappers (a layer of DFT logic that connects a TAM to a core for the purpose of testing), like IEEE 1500 Standard Wrapper, are frequently used as a modular testing approach on 3D-ICs. A 3D-IC test architecture must be able to support testing of individual dies as well as testing of partial and complete stacks. Furthermore, test architecture optimization must not only minimize the test time, but also minimize the number of dedicated test TSVs used to route the 3D TAM, as each TSV has area costs associated with it and is a potential source of defects. It is important to note that all the chip pins on a 3D-IC

are at the lowest die and taking into account that the number of TSVs is limited, the test access to the higher layers should be dealt carefully. In [84] Wu et al. presented a modular testing approach, based on a combination of integer linear programming, LP-relaxation, and randomized rounding, for minimizing the test time for 3D core-based SoCs under constraints on the number of TSVs and the TAM width. Lo et al, in [85], proposed a test architecture for pre-bond and post-bond tests in core-based 3D-ICs, optimizing control signals and TAM in test pins and TSVs.

Deutsch & Chakrabarty, in [86], formulated a robust optimization problem for SoC testing for the first time. The main idea of robust optimization is to find a solution that may not be optimal for the nominal values of the input parameters but that remains close to the optimum in the presence of parameter variations. Previous works, like [41, 50, 51], based on exact optimization techniques such as ILP and heuristics (such as rectangle packing) consider known (constant) values for input parameters, which may differ from the actual values, leading to non-optimal decisions made at the design stage, increasing the test time. In [12], Deutsch, Chakrabarty and Marinissen formulated the problem of robust TAM optimization and test scheduling for 3D-ICs in the presence of variations in input parameters and developed an ILP model in the presence of variations in input parameters for the test architecture, such as power and available bit-width (Figure 2.6). They also proposed an efficient heuristic for robust optimization for dies in a 3D stack that have a large number of embedded cores, using the simulated annealing algorithm.

2.1.6 Self Test & Repair

BIST is present for years at conventional 2D-ICs, but the extend of these techniques to 3D-ICs is not trivial. BIST doesn't need external intervention and reuses the available area. In 3D-ICs, however, there are many possible test insertions, so there is a need for a distributed BIST framework to work at every step of the stacking process. Such a framework was presented by Agrawal, Chakrabarty and Eklow in [87]. Their framework is distributed, reusable and reconfigurable and its components interact with each other through handshaking protocols.

Before that, various works were published for self test and repair, usually focused on a specific part of the testing process. Cho et al. first presented an on-line test and self-repair structure to characterize TSVs at pre-bond testing in [88]. Their structure

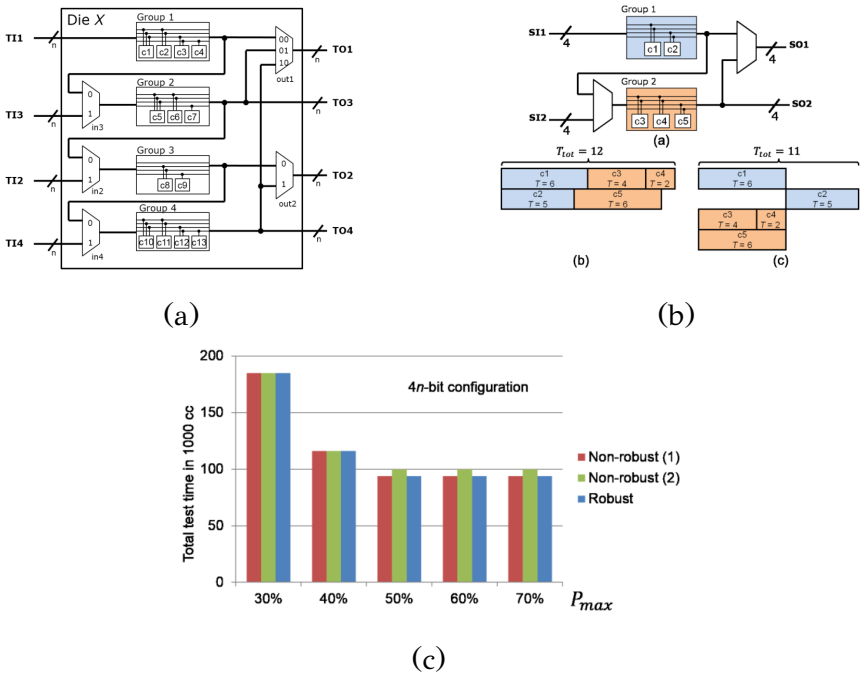


Figure 2.6: A method for robust optimization of 3D test architecture and test scheduling in the presence of input parameter variations [12].

reconfigures itself to compensate TSV defects and improves the fidelity of the passing signals. Huang et al. also focused on TSV testing with a BIST scheme at post-bond stage [13] (Figure 2.7) and in [89] Huang and Li added a TSV redundancy scheme with spare TSV row to repair the defective TSVs. Pasca, Angel and Benadbenbi proposed in [90] a TSV Interconnect BIST (IBIST) technique, based on K^{th}-Aggressor Fault (KAF) model, to deal with open, short and delay faults due to crosstalk.

SenGupta, Ingelsson and Larsson proposed a power constrained test scheduling for TSV based 3D-ICs for the first time in [91]. They proposed two test scheduling approaches (Partial Overlapping and ReScheduling) that minimize test application time while taking power constraints and the need to route JTAG and Test Data Registers into account. Hou and Li proposed a built-in self-repair (BISR) allocation scheme for RAMs in the SoC die of 3D-ICs under the constraints of the pre-bond and post-bond test sequences and the distance between the BISR circuit and served RAMs [92]. Jiang et al. presented the first in-field TSV repair framework for 3D-IC lifetime reliability enhancement [93]. They didn't focus only on faulty TSV replacement, but they find the set of signal-TSV pairs that satisfy the timing requirement of every signal. Yang, Chou and Li proposed a repair scheme based on a typical test access architecture, by reusing registers of wrapper cells [94].

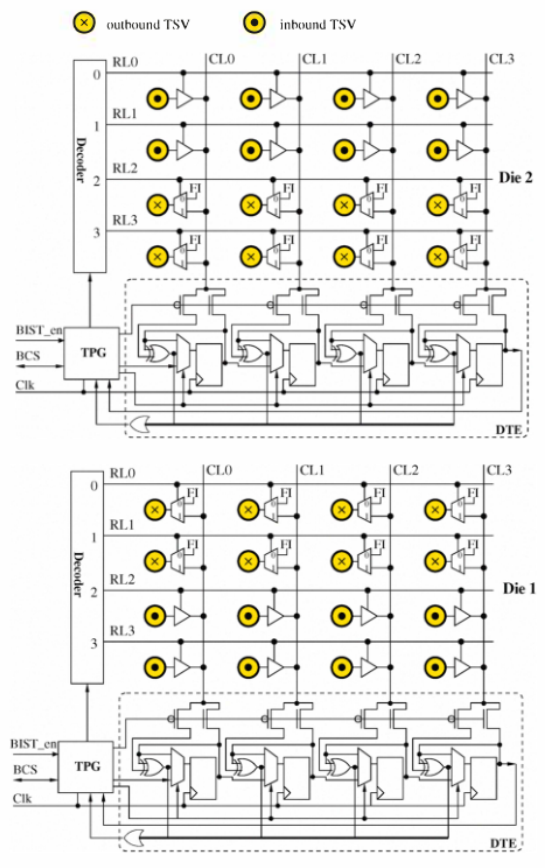


Figure 2.7: Proposed test architecture for a 4×4 TSV array [13].

2.1.7 Clock Tree

Early works in 3D-IC testing assumed that proper clock signals for both the prebond and post-bond operations had already existed. Every die needs a complete 2-D clock network for the pre-bond test and the final 3D-IC needs a complete 3D low power clock network for the post-bond test and normal operation after stacking. The temperature has important impact on clock trees in 3D-ICs, the different temperature zones may lead to significant clock skew. In 2-D circuits, symmetric interconnect structures, such as H- and X-trees, are widely utilized to distribute the clock signal across a circuit [95]. These symmetric structures allow the clock signal to arrive at the leaves of the tree simultaneously, achieving synchronous data processing. This kind of symmetry is difficult to achieve in 3D-ICs.

Mondal et al. in [96] attempted for the first time to reduce the clock skew for 3D-ICs, by proposing a new circuit technique that dynamically adjusts the driving strengths of the clock buffers according to the spatial and temporal temperature profile of the chip. Their technique uses only modified clock buffers that reduces the

design complexity and does not assume any fixed temperature profile. An important contribution to relevant area is the work of Minz, Zhao and Lim, who presented the Balanced Skew Theorem in [97]. The theorem provides a theoretical background on the efficient construction of a buffered 3D clock tree that perfectly balances the skew values under two distinct non-uniform thermal profiles. Zhao et al. in presented a pre-bond testable clock routing for the first time in [98]. They also proposed a new circuit element called TSV-buffer, which supports zero-skew pre-bond testing for the clock trees that use multi-TSVs and introduced redundant tree, which supports the pre-bond testing of dies. Buttrick and Kundu used a Delay Lock Loop (DLL) in testing while allowing for separate scan and test clock frequencies in [99]. Since a DLL cannot change operating frequencies or be turned on and off during testing, the proposed work is essential to the testing of incomplete clock networks in 3D-ICs using DLLs.

Arunachalam and Burleson proposed a global clock distribution network consisting of tree driven grids [100]. The clock generators and the buffers that drive the clock grids form the clock layer, which drives the clock grids in the other logic layers using TSVs. Pavlidis, Savidis and Friedman evaluated three topologies (H-trees, local meshes and global rings) to globally distribute a clock signal in 3D-ICs [101]. Kim and Kim in [102] formulated zero skew clock tree embedding problem in 3D-ICs and proposed the zero skew clock tree embedding algorithm ZCTE-3D, which is shown that optimally allocates TSVs for a given tree topology. They also proposed an complete flow of clock tree synthesis by proposing a new 3D aware topology generation algorithm MMM-3D and integrating it into ZCTE-3D. The same people proposed a new tree topology generation technique MMM-3D-cap to minimize potentially 'unstable' TSV-buffers, and a novel circuit element SCCTG to completely remove the pre-bond test control signal in [103]. They also proposed three key algorithms for synthesizing clock trees in TSV-based 3D-ICs: NN-3D, which is a cost-effective (in terms of TSVs, wirelength, and clock power) 3D clock tree topology generation algorithm, DLE-3D, which is a TSV-optimal layer embedding algorithm of tree nodes and DME-3D, which is a 3D clock tree routing algorithm with buffer insertion [104]. Kim and Kim improved their previous works in [105].

Lung et al. presented in [14] a fault-tolerant 3D clock network which utilizes the existing 2D redundant clock trees for pre-bond testing (Figure 2.8). A TSV fault-tolerant unit (TFU) is proposed to automatically reroute the clock signal using the

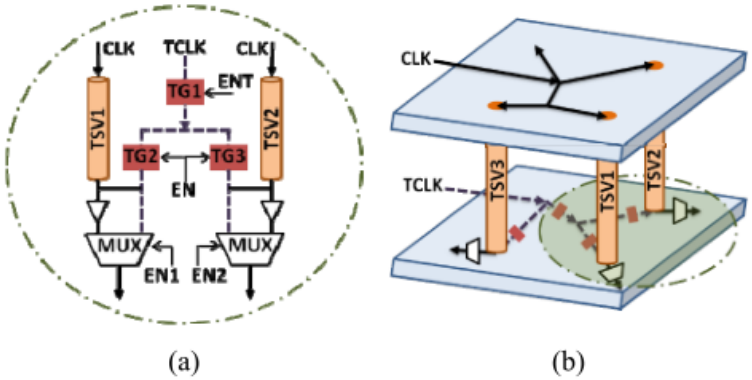


Figure 2.8: (a) The proposed TSV fault-tolerant unit (TFU). (b) The fault-tolerant 3D clock network using TFUs [14].

2D redundant tree in the presence of TSV failure. This work is the first that considers the fault tolerance of a 3D clock network. Zhao et al. in [106] extended their work [98] for both pre-bond and post-bond testing of 3D-ICs. They have shown that by allocating the clock source in a middle die in the stack, the pre-bond testable clock tree will use fewer TSVs, while they saved power and wirelength. They also analyzed the impact of the parasitic TSV capacitance on pre-bond testable clock trees in terms of wirelength, buffer count, and clock power. Zhao, Minz and Lim in [107] explored design optimization techniques for reliable low-power and low-slew 3D clock network design and they studied the impact of the TSV count and the TSV capacitance on clock power trends. They also developed a low-power 3D clock tree synthesis algorithm called 3D-MMM-ext. Park and Kim formulated the TSV pairing problem into a minimum-cost maximum matching problem in a graph and proposed an efficient solution while satisfying all constraints [108]. They designed a new circuit, called Slew-Controlled TSV Fault-tolerant Unit (SC-TFU) and implemented it for each TSV pair by taking into account both the slew problem and delay control problem caused by the increased load capacitance. The insertion of SC-TFUs causes clock skew variation and therefore they proposed a global skew timing technique to refine it.

2.1.8 Thermal

The thermal issue is always a concern for the conventional chips, but it is even worse in 3D-ICs, mainly because the upper layers have a less direct heat dissipation path [109]. Overheating during manufacturing test may lead to permanent chip damage.

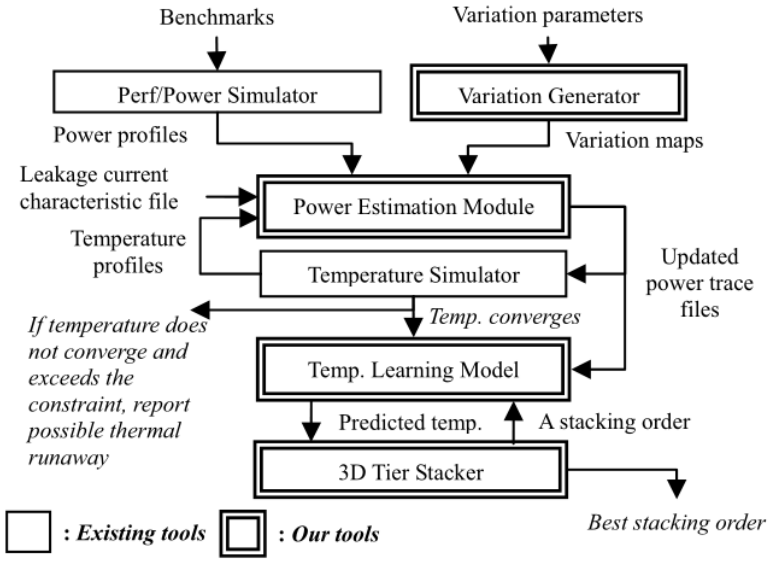


Figure 2.9: Overall flow of [15].

Therefore, it is essential to take thermal issues into account during 3D test architecture design and optimization, in order to avoid the yield loss problem. Thermal imaging is an established technique for fault diagnosis in regular integrated circuits.

Juan, Garg and Marculescu proposed in [15] a methodology to perform statistical thermal evaluation for 3D-ICs and an accurate learning-based regression model to predict the maximum temperature in the steady state (Figure 2.9). It may also used as a design exploration environment in order to improve thermal yield. Jiang et al. proposed a novel thermal-aware test scheduling algorithm to reduce the hotspot temperature of the SoC during post-bond test, while they designed different test architectures for pre-bond and post-bond tests in order to tackle the pre-bond test-pin-count constraint problem. In [110], Xiang, Shen and Deng proposed a novel thermal-driven scheme to reduce high temperature hotspots created by the test process. They used a scan tree architecture, minimizing the connection overhead and effectively reducing test application time. A test vector ordering scheme is developed in order not to worsen the hotspots and a new test application scheme applies the ordered test set reducing the test frequency, while it doesn't increase the test application cost. Dev, Woods and Reda proposed in [111] a new paradigm for pre-bond TSV testing and characterization using thermal imaging to test the electrical connectivity of TSVs. They analyzed the thermal signatures from a device, by establishing a reference thermal map and classified the status of TSVs (functional or defective) based on the thermal signatures from the device under test and the thermal reference map.

2.1.9 Memories

3D integration has been envisioned as a solution for future micro-architecture design to mitigate the interconnect crisis and the ”memory wall” problem. Approaches of memory stacking on top of core layers do not have the design complexity problem as demonstrated by the fine-granularity design approaches, which require re-designing all processor components for wire length reduction [112]. There are three integration technologies used in 3D-IC fabrication including 3D RAM: wafer-to-wafer (W2W), die-to-wafer (D2W), and die-to-die (D2D). Wafers with the same size of dies are directly bonded together in the W2W integration method. The W2W offers the highest throughput, the thinnest wafers, and high TSV density. However, the W2W can incur a low yield because the bad die can be stacked to another good die, which results in a bad 3D-IC. On the other hand, the D2W and D2D integration methods allow the use of different wafer and die seizes because the dies are bonded after dicing. The D2W and D2D can improve the yield of 3D-ICs due to its high flexibility that the dies are diced and tested in advance and only the good dies are stacked. However, these methods require a more complex manufacturing process that the dies must be aligned and integrated each other. This results in low throughput and low TSV density [113]. In general, a large memory is normally equipped with redundancy, which is used to repair (replace) defective parts of the memory and improve the yield.

Back in 2005, Tsai et al. explored the architectural design of cache memories using 3D structures [114]. They examined possible partitioning techniques for caches designed using 3D structures and presented a delay and energy model to explore different options for partitioning a cache across different device layers. Jiang et al. conducted extensive simulations to study the faulty behavior of TSV open defects and map them to functional fault models of the memory circuits, which serves as the first step to tackle the test and repair problem for 3D DRAMs [115]. Jiang, Ye and Xu proposed to conduct redundancy sharing across neighboring dies for yield enhancement of 3D-stacked memory circuits. They developed a repair strategy that enables redundancy sharing for any given pair and presented novel solutions that selectively match memory dies together for yield maximization [116]. Taouil and Hamdioui, in [117], investigated layer redundancy as a mean for compound yield improvement for 3D W2W stacked memories. Kang et al., in [113] proposed a new die selection and matching method with two stages, along with a new redundancy analysis (RA) algo-

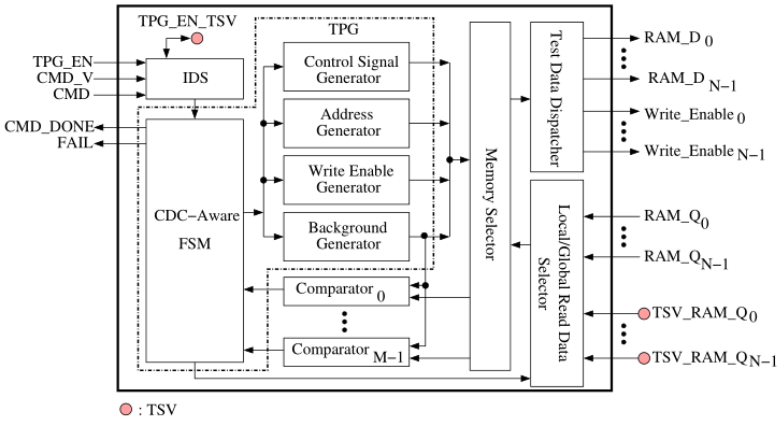


Figure 2.10: Block diagram of proposed BIST [16].

rithm, that holds other repair solutions for two stages of the proposed die selection and matching method. The proposed die selection and matching method consists of two stages.

Yu et al. have proposed a programmable BIST scheme for the 3D RAMs in [16]. The BIST scheme supports the selection of RAMs in 3D RAM for test in addition to the selection of test algorithms and enables that the BIST circuit in one die can evaluate the read data using the comparator in another one. The proposed BIST also has the feature of high programmability to support thermal management during the test. In [118], Lin, Lee and Wu proposed a tool, Raisin-C to help memory designers find suitable redundancy architecture and to repair the 2D and 3D memories efficiently. Taouil and Marinissen proposed a post-bond Memory Based Interconnect Test methodology cable to test interconnects between memory and logic dies by performing read and write operations from the logic die (CPU) to the memory dies [119]. They also provided a classification of interconnect defects, compiled them into fault models, and discussed the test pattern generation for these faults and used the proposed methodology to implement them.

2.1.10 Cost Modeling

Test cost has been the most serious concern in the adoption of 3D integration. The choice of test flow, i.e., what tests are used and when they are applied during 3D integration affects test cost. 3D stacking involves many possible test insertions. Due to multiple yield and test cost parameters corresponding to different dies and tests, such as for pre-bond, post-bond, and partial stack, an exponentially large number of

test flows must be evaluated. Therefore, analysis methods and tools are required for test-cost optimization and automated test-flow selection [17]. There are several works on various aspects of test-cost modeling and optimization for 3D-ICs, which attempt to hand-pick a test flow or select a test flow based on explicit enumeration of a few candidate test flows. Agrawal and Chakrabarty have presented the most complete, for now, cost model for various stages of the process in [17].

Earlier, in [120], Chen et al. proposed a testing cost analysis methodology with a break-down of testing cost for different integration strategies. With their model proved that the design choices could be different after testing cost is taken into account. The same year, Taouil et al. investigated the impact of several 3D test flows on the total cost in die-to-wafer stacking. Their framework is based on a combination applied at pre-bond wafer test, intermediate stack test, pre-package test and post-package test [121]. In [122], Hamdioui and Tamouil overviewed wafer matching and layer redundancy and showed how these schemes can significantly improve the overall yield. They also presented an analysis of different test flows and showed how different test flows can result into different cost and that the cheapest test flow does not necessarily results in lower overall 3D-SIC cost.

Chen, Huang and Li, in [123], focused on pre-pond testing and they proposed a test cost optimization technique for that part of the 3D-IC testing. Their technique attempts to minimize the number of required pads (additional test pads increase area cost) of each die in a wafer and takes the advantage of test parallelism to minimize the overall test time of the wafer. Chou et al. moved in a different direction in [124], as they focused on interposer-based 3D-ICs and proposed cost models for both the die-to-wafer and die-to-die stacking, which include manufacturing cost and test cost. They also proposed a tool based on proposed cost models for the cost analysis of test flows, that are likely to be used in the industry, they investigated impacts of different test flows on the total cost and analyzed the critical points of adopting the test operations.

The most recent work in this area is the model of Agrawal and Chakrabarty, which is generic and flexible and may be adapted for wafer-to-wafer, die-to-wafer and die-to-die stacking [17]. They explored test cost optimization for 3D-ICs by taking into account various test costs at every step of the stacking process (Figure 2.11), they defined a variety of parameters affecting test cost and presented a heuristic procedure, guided by a matrix-partitioning problem.

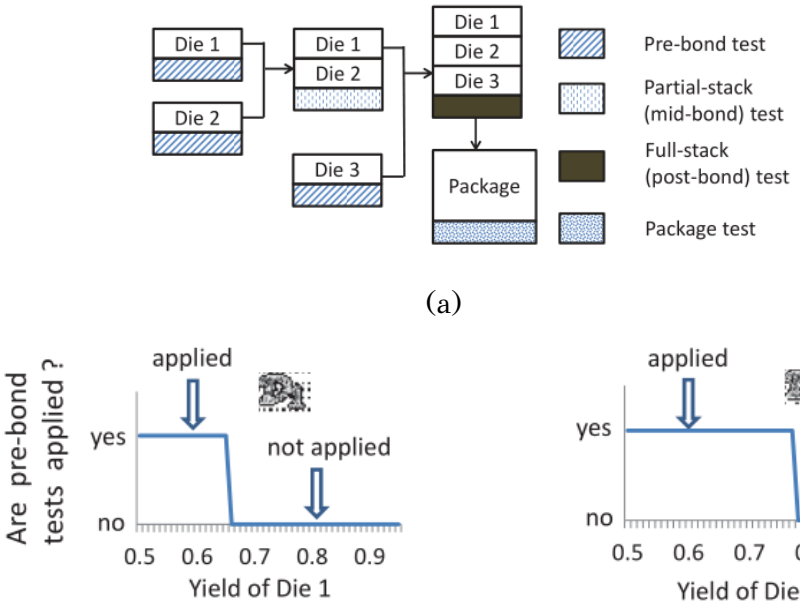


Figure 2.11: A generic and flexible cost model to account for various test costs incurred during 3D integration [17].

2.2 Software-Based Self-Testing

2.2.1 Processor Functional Fault Self-Testing

The processors are extremely hard to test because of their complexity and the limited accessibility of their internal logic. The situation is even worse for the test engineers who apply SBST as the processors' internal design details are usually unavailable and difficult to understand. Processor functional-level fault models and test generation methods have been proposed in [125] and [126] to resolve these problems. These techniques can provide a testing solution for general-purpose processors, because there is only need for knowledge of the processor's instruction set and its functions. In [125] fault models and test generation methods were proposed for functional faults associated with register decoding, instruction decoding & control, data storage, data transfer and data manipulation. In the following paragraphs we are going to describe the aforementioned methods of [125].

The processor is modeled at RTL by a system graph (S-graph), based on its instruction set and the functions it performs. Each register which can be explicitly modified by an instruction is represented by a node in the S-graph. The S-graph also contains two additional nodes, IN and OUT, which represent the main memory and

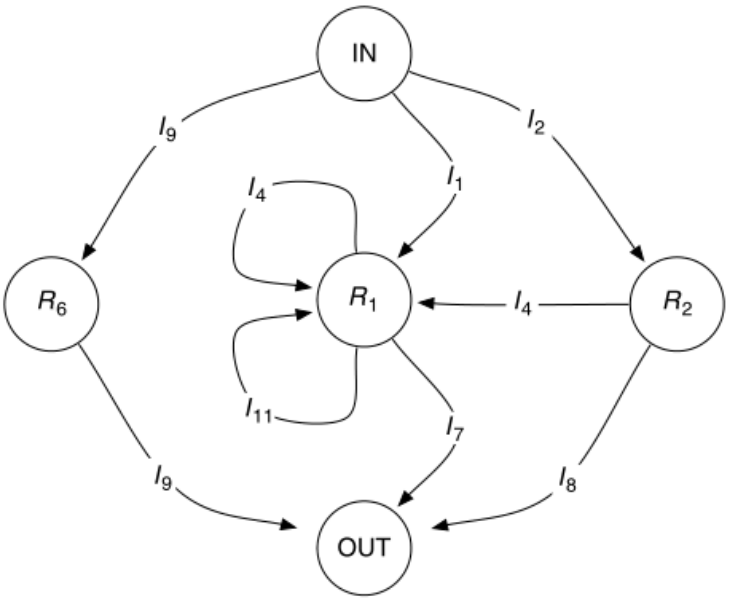


Figure 2.12: An example S-graph [4].

I/O devices. The nodes of the S-graph are connected by directed edges. If data flow occurs from node A to node B during the execution of any instruction, then a labeled directed edge exists from A to B. In Figure 2.12 [4] an example is presented. For convenience, only a subset of instructions and those registers in the processor that are directly involved in carrying out these instructions are presented in Table 2.1.

There are three classes of instructions in Table 2.1 as in [127]. Instructions I_4 and I_{11} belong to manipulation class (M), instruction I_9 belongs to branch class (B) and the rest belong to transfer class (T). Note that multiple edges may be associated with one instruction. However, an instruction is allowed to have multiple destination registers only if it involves a data transfer between the main memory or an I/O device and if it registers this transfer during its execution. We define the destination registers of an instruction as the set of registers that are changed by that instruction during its execution. Also, it is assumed that any register can be written or read using a sequence of either class T or B instructions.

After the construction of S-graph, integer labels are assigned to the nodes by the node labeling algorithm in Algorithm 2.1. The node label indicates the shortest distance of a node to the OUT node in the S-graph. On a processor, the node label corresponds to the minimum number of class T or B instructions that need to be executed in order to read the contents of the specific register. The nodes of the example of Figure 2.12 are labeled as follows: the OUT node is labeled 0, while R_1,

Table 2.1: Summary of the registers & instructions in Figure 2.12.

R_1	Accumulator (ACC)
R_2	General Purpose Register
R_6	Program Counter (PC)
I_1	Load R_1 from the main memory using immediate addressing. (T)
I_2	Load R_2 from the main memory using immediate addressing. (T)
I_4	Add the contents of R_1 and R_2 and store the results in R_1. (M)
I_7	Store R_1 into the main memory using implied addressing. (T)
I_8	Store R_2 into the main memory using implied addressing. (T)
I_9	Jump instruction. (B)
I_{11}	Left shift R_1 by one bit. (M)

Algorithm 2.1 The node labeling algorithm.

1: assign label 0 to the OUT node;
2: $K \leftarrow 0$;
3: **while** there exist unlabeled nodes **do**
4: assign $K + 1$ to unlabeled nodes whose contents can be transferred to any register(s) labeled K by executing a single class T or B instruction;
5: $K \leftarrow K + 1$;
6: **end while**

R_2, and R_6 are labeled 1. Node and edge labels are used during the test generation process.

Fault models of processor functional-level are developed at a higher level of abstraction independent of the details of their implementation. The functional-level fault models are the following:

- **Register decoding fault model.** Under the register decoding fault, the decoded address of the register(s) is incorrect. Consequently, the wrong register(s) may be accessed, or no register is accessed at all. In this case, the outcome retrieved is technology dependent.

- **Instruction decoding and control fault model.** The processor may execute a wrong instruction, execute some other instructions (including the original one), or execute no instruction at all.

- **Data storage fault model.** Single stuck-at faults may occur in any number of cells in any number of registers.
- **Data transfer fault model**. A line in the data transfer path may be stuck at 1 or 0. Also, two lines in the data transfer path may be coupled.
- **Data manipulation fault model.** No specific fault models are proposed for data manipulation units. Instead, it is assumed that the required test patterns can be created according to the implementation.
- **Processor functional-level fault model.** The processor may possess any number of faults that belong to the same fault model.

The goal of the test generation targeting the **register decoding faults** is to validate that the register mapping function ($f_D : R \rightarrow R$, where R is the set of all registers) is correct. The test generation flow is presented in Algorithm 2.2. Initially, the FIFO queue Q is initialized with the set of all registers such that registers with smaller labels are in the front of Q. Set A contains the processed registers, consisting of the first elements of Q. Afterwards, test generation is performed in two phases (write and read), one register at a time. In the write phase, all the registers in set A are written with ONE (all ones), and the first register in Q, denoted by R_{next}, is written with ZERO (all zeros). The required write operation for each register is the shortest sequence of class T or class B instructions required to write the target register. In the read phase, registers in set A are read in the order of ascending labels. Then, the content of R_{next} is read out. Similarly, the required read operation for each register denotes the shortest sequence of class T or class B instructions required to read the targeted register.

The test generation algorithm in Algorithm 2.2 assures that all the registers have disjoint image sets under the register mapping function, thus achieving its one-to-one correspondence. The generated test program is capable of finding any detectable fault in the fault model for the register decoding function. One example of a possible undetectable fault is the concurrent occurrence of $f_D(R_i) = R_j$ and $f_D(R_j) = R_i$.

For the detection of **instruction decoding and control faults**, let I_j be the instruction to be executed. To simplify the test generation, we assume that the labels of class M instructions are no greater than 2 and that all class B instructions have the label 1. Only class T instructions can have labels greater than 2. The faults in

Algorithm 2.2 Test generation for register decoding faults.

1: $Q \leftarrow$ sort($\mathbb{R}$);
2: $R_{next} \leftarrow$ dequeue(Q);
3: $A \leftarrow \{R_{next}\}$;
4: **while** $Q \neq \emptyset$ **do**
5: **for all** $R_i \in$ A **do**
6: append write (R_i, ONE) to test program;
7: **end for**
8: $R_{next} \leftarrow$ dequeue(Q);
9: append write (R_{next}, ZERO) to test program;
10: $Q' \leftarrow$ sort(A);
11: **while** $Q \neq \emptyset$ **do**
12: append read (dequeue(Q')) to test program;
13: **end while**
14: append read (R_{next}) to test program;
15: $A \leftarrow A \cup R_{next}$
16: **end while**
17: repeat steps 1-16 with complementary data;

which no instruction is executed at all is denoted by $f(I_j/\emptyset)$. The faults in which the wrong instruction I_k is executed instead of the instruction I_j is denoted by $f(I_j/I_k)$. The faults in which some other instruction I_k is also executed along with I_j are each denoted by is denoted by $f(I_j/I_j + I_k)$.

Algorithm 2.3 shows the order of tests appliance. The order of the faults detection is highly important in order to ensure fault coverage. Tests are applied so that the knowledge gained from testing instructions of lower labels is utilized for testing instructions with higher labels.

Let I_j with $label(I_j) = 2$. In order to detect the $f(I_j/\emptyset)$ fault, O_1 is written to destination register R_d of instruction I_j using a class T or class B instruction. Then, proper operand(s) are written to I_j's source registers, such that when I_j is executed, it produces O_2 ($O_2 \neq O_1$) in R_d. Afterwards, I_j is executed and R_d is read with the expected output is O_2. We define the set of source registers for an instruction as the set of registers that provide the operands for that instruction during its execution.

In order to detect the $f(I_j/I_k)$ faults consider that $label(I_j) = label(I_k) = K \geq 3$

Algorithm 2.3 The order of test generation for instruction decoding and control function.

1: $K \leftarrow 1$;
2: **for** $K = 1$ **to** K_{max} **do**
3: apply tests to detect $f(I_j/\emptyset)$, $f(I_j/I_k)$ and $f(I_j/I_j + I_k)$, where $label(I_j) = label(I_k) = K$;
4: apply tests to detect $f(I_j/I_j + I_k)$, where $1 \leq label(I_j) \leq K, label(I_k) = K + 1$, and $K < K_{max}$;
5: apply tests to detect, where $K + 1 \leq label(I_j) \leq K, label(I_k) = K$;
6: **end for**

and that instructions I_j and I_k have the same destination register. Then, instructions I_j and I_k are class T instructions, and they have only one destination register each. Initially, O_1 and O_2 ($O_1 \neq O_2$) are written to the source registers of instructions I_j and I_k, respectively. Then, I_j is executed, and R_d is read for K times with the expected output being O_1. In the end, I_k is executed and R_d is read with the expected output being O_2. If O_2 is really stored in the source register of I_k at the beginning of the procedure, $f(I_j/I_k)$ will be detected during the first execution of I_j read of R_d. However, because of the faults involved in instructions used to write O_2 in I_k's source register, O_1 may have been stored in the source register of I_k. In this case, $f(I_j/I_k)$ will not be detected. In the worst case scenario, instruction I_j will be executed K times to achieve the detection of $f(I_j/I_k)$.

In order to detect $f(I_j/I_j + I_k)$ faults, an example is when:

$$1 \leq label(I_j) \leq K,$$
$$label(I_k) = K + 1,$$
$$K \geq 2.$$

I_k is a class T instruction, and the destination registers R_j and R_k of I_j and I_k, respectively, are different. When the label of I_k's source register is less than K, different operands O_1 and O_2 are first written to I_k's source and destination registers, respectively. Then, I_k's source register is read, and the expected output is O_1. Finally, I_j is executed and I_k's destination register is read, and the expected output is O_2.

For the detection of **data storage and storage function faults**, there are different test generation procedures depending on the class which the involved instruction belongs.

For instructions that belong to class T, let's assume a sequence of instructions $I_{j1}, I_{j1}, \ldots, I_{jk}$. Their associated edges form a directed path in S-graph from node IN to node OUT. All paths of this kind should be tested. In the specific path, initially I_{j1} is executed with operand O_1. Afterwards, the remaining instructions are executed and the expected output is O_1. If the data transfer path width is 8, then the test procedure is repeated for the following O_1 configurations: 11111111, 11110000, 11001100, 10101010.

For instructions that belong to class M, we will use instruction I_4 of Table 2.1 as an example. See Figure 2.12 for the involved edges of I_4. For I_4, the paths from ALU to R_1 and from R_1, R_2 to ALU should be tested. For the first path, we use I_1 and I_2 to load R_1 and R_2 with O_1 and all zeros. Afterwards, I_4 is executed, followed by instruction I_7. The result is read and stored in R_1. If the processor is an 8-bit processor, to fully test the path, the procedure is repeated for the same following O_1 configurations of the previous paragraph (complemented and uncomplemented). For the paths from R_1 to ALU, the register is loaded with O_1, and R_2 with all zeros. Then, instructions I_4 and I_7 are executed and the expected output is O_1. The procedure is repeated for the following O_1 configurations: 00000001, 00000010, $\ldots$, 10000000. The testing of the path from R_2 to ALU is similar and not repeated here.

For instructions that belong to class B, we take as example the instruction I_9. If the address bus width is 8, instruction I_9 should be executed with the following jump addresses (in both complemented and uncomplemented forms): 11111111, 11110000, 11001100, 10101010.

No specific fault model is proposed for the **data manipulation functions**. Instead, given the test patterns for a data manipulation unit (ALU, shifter, etc.), the desired operands and the results can be delivered to its input(s) and the output(s), respectively, using class T instructions.

2.2.2 Processor Structural Fault Self-Testing

Processor Structural Fault Self-Testing techniques target structural faults, including stuck-at and delay faults. These techniques are composed of two phases, the test preparation phase and the self-testing phase.

In the **test preparation phase**, instruction sequences that deliver structural test patterns to the inputs of the processor component under test and transport the output responses to observable outputs are generated. A challenge for this test generation

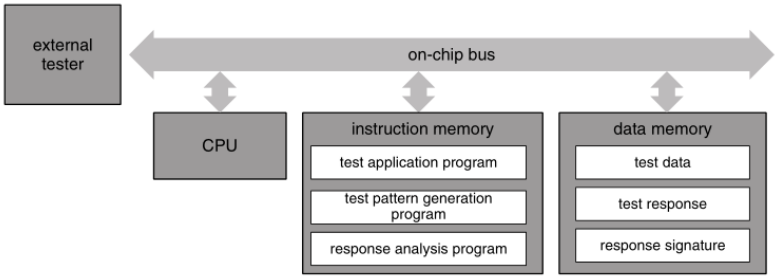


Figure 2.13: Processor self-testing setup [4].

is the I/O constraint imposed by instructions. The input constraints define the input space of the component allowed or realizable by processor instructions. A fault is undetectable, if none of its test patterns belongs in the input space. The output constraints define the subset of component outputs observable by instructions. A fault is undetected at the chip level if its resulting errors fail to propagate to any observable outputs. When the test generation doesn't take into account these constraints, it may produce test patterns that cannot be created by the processor's instructions.

There are methods ([128, 129, 130, 131, 132]), where the extracted component I/O constraints are expressed in the form of Boolean expressions or HDL descriptions and are fed to ATPG for constrained component test generation. Afterwards, the test program synthesis procedure maps the constrained test patterns to processor instructions. Additionally to the test application instruction sequence, test supporting instruction sequences may be added to set up the required processor state and to transport the test responses to main memory.

In Figure 2.13 [4] the processor **self-testing** setup is presented. The test program and its responses are stored in on-chip or cache memory. Therefore, the memory has to be tested with standard techniques as memory BIST and repaired if necessary to keep being functional. Then, an external tester loads the test program and data to the on-chip memory. The processor is set up to execute the loaded test program and then the test signatures are downloaded to the external tester for the test decision or diagnosis.

2.2.3 Global Interconnect Testing

In modern SoC designs, a device must be able to perform core-to-core communications across long interconnects. The gate delay is constantly decreasing and as a

result the interconnects must have also high performance in to order to achieve high overall performance. As the cross-coupling capacitance and mutual inductance are increased, the signals on neighboring wires may interfere with each other. This results at excessive delay or even loss of signal integrity. Many techniques have been proposed to reduce crosstalk, but there are limited design margins and unpredictable process variations. Therefore, crosstalk must also be addressed during manufacturing test.

Testing for crosstalk effects should be performed at the rated speed of the CUT, because of their impact on circuit timing. However, this may result to unacceptable cost for high-speed testers. In [133], a BIST technique has been proposed in which an SoC tests its own interconnects for crosstalk defects with the use of on-chip hardware pattern generators and error detectors. However, the amount of relative area overhead may be unacceptable for small systems. Because this method is a structural BIST technique, utilizing this technique may cause overtesting and yield loss, because not all test patterns generated are valid when the system is operated in normal mode.

Using the processor itself in order to execute self-test programs for its interconnections is a viable solution for SoCs with embedded processors, as most of the system-level interconnects are accessible to the embedded processor cores. During the execution of these test programs, test vector pairs can be applied to the appropriate bus in normal functional mode of the system. If there are crosstalk-induced glitches or delay effects, the 2^{nd} vector of the pair becomes distorted at the receiver end of the bus. This distortion can be stored by the processor in memory as a test response. This can be unloaded by an external tester and used for off-chip analysis.

Chapter 3

Research work

3.1 Research Directions

3.1.1 3D SoCs Testing

This research targets the reduction in test application time of 3D SoCs by exploiting the high speed offered by TSVs. In Section 1.5, we have already described how 3D-ICs prevail over traditional 2D integration techniques on various aspects. Therefore, the effective testing of 3D SoCs under strict time and power constraints with the minimum interconnection overhead is of high importance.

In Subsection 1.5.1 we explained why TSVs are the most promising choice for 3D-ICs. In the same subsection, we also described their disadvantages, which must be taken into account while designing a 3D SoC. Therefore, only a limited number of TSVs are used in any 3D design, and very few of them are available for test purposes.

The low number of test TSVs creates a serious bottleneck for the TAM of the 3D-ICs, which delivers high volume of test-data using a small number of horizontal and vertical interconnects. In addition, the scan-chains of the cores support low shift frequencies because they are not optimized for timing. Therefore, the highest rate at which test-data can be transferred through the TAM is very low. On top of that, the lower thermal conductivities of inter-tier and inter-metal dielectrics used in 3D-ICs block the heat generated inside the stacks from reaching the heat sink [134]. As a result, the scan shift frequencies are often further reduced to avoid violating power and thermal limitations of 3D-ICs. Since the test-time for a 3D-IC is dominated by the time needed for transporting test-data to various layers of the stack, the limited number of TSVs adversely affects the test-time of 3D-ICs.

3.1.2 Processor-Based Devices Testing

This research also targets the improvement of the defect screening of processor-based devices. The deep sub-micron semiconductor technologies combined with advanced architectural innovations have significantly improved the performance of SoCs. The continually increasing demands of the market for higher computational performance at lower cost and power consumption drive processor vendors to develop new microprocessor generations.

These technology advancements, however, introduce new challenges on processor-based device testing. Especially when used in safety-critical applications, SoCs require advanced testing techniques for screening defective devices. However, the strict design constraints and the need to test the target devices at the normal mode of operation impose the use of non-intrusive test methods [135]. In addition, any test applied in-the-field should not compromise the internal state of the Device Under Test (DUT) [136]. Therefore, design-for-testability solutions are complemented with functional solutions, such as SBST for processor-based ICs and SoCs.

SBST concept was presented in Figure 1.16. SBST is completely autonomous, as it does not require the assistance of any ATE. Moreover, it is applied exactly at the operating conditions, while at the same time it does not excite any redundant faults avoiding thus over-testing the core under test. Finally, SBST facilitates the periodic monitoring in-the-field with limited intrusiveness with respect to the normal-mission operation [137, 138], and it does not compromise the internal state of the circuit [136].

Despite their benefits, SBST methods suffer from several drawbacks. At first they often target only the stuck-at fault model, which is inadequate for detecting many defects. In addition, most SBST techniques are not systematic, therefore they require extensive human intervention and long development times. Moreover, they involve the CPU-intensive process of fault-simulating multi-million gate designs for multi-million clock cycles using multiple fault models and specialized functional (non-scan) simulators. Besides these deficiencies, the shrinking process technologies, the physical limits of photo-lithographic processes and new materials introduce new defects that are not always accurately modeled even by the most commonly used fault models [139]. Therefore, fast, low-cost and highly effective SBST-based techniques are required to improve the defect screening of processor-based devices.

The organization of the chapter is as follows. Section 3.2 lists the main objectives of this research work. Section 3.3 describes in detail a new TAM architecture and an effective TDM-based test scheduling technique for 3D SoCs. Section 3.4 presents a 3D test architecture which overcomes the limitations of the previous method. Finally, Section 3.5 presents the first fault-independent SBST method for processor-based SoCs.

3.2 Research Objectives

Based on the research directions presented in Section 3.1, the following research objectives were targeted in for this work:

- Introduce TAM improvements for 3D-ICs in order to:
  - Reduce the test time.
  - Minimize the interconnection overhead.
  - Satisfy power and thermal constraints.
- Increase the defect coverage of processors, which are hard to test because of:
  - Their complexity.
  - The limited accessibility of their internal logic.
  - The lack of test structures due to performance constraints.

- Avoid time-consuming fault-simulations using multiple fault-models for testing of processors.

3.3 Testing 3D-SoCs Using 2-D Time-Division Multiplexing

3.3.1 Background & Motivation

The 3D scan-chain design problem was investigated for the first time in [10], and various wrapper designs were proposed in [78, 81, 140, 141]. In [79], a test methodology for pre-bond, known-good stack, and post-bond tests of 3D-ICs was proposed. In [82], the authors presented a flow for fast scan-chain ordering under TSV constraints, while in [85, 11] test architectures for pre-bond and post-bond tests were proposed. In [83] re-timing techniques were proposed and in [84] a modular testing approach was presented for 3D-SoCs. In [142] a unicast-based multicast approach for testing 3D Networks-on-Chips (NoCs) was presented. Various TAM optimization methods were proposed in [48, 50, 51, 52, 143] and 3D-ICs with multiple towers were considered in [80]. In [8], generalized test-time optimization methods were presented, and a modular post-bond test strategy was presented in [144].

TSVs can transfer test-data in the GHz range, and they have been exploited in 3D NoCs to create fast vertical communication interfaces between different dies in the stack [145, 146, 147]. However, the test data are transferred using a small number of TAM lines and test TSVs, while the shift frequencies are very low due to scan-chain constraints (usually a few hundred MHz). This large gap between the maximum TSV frequency and the shift frequencies prevents test engineers from exploiting the high speed offered by TSVs to minimize test-time in 3D-ICs.

In order to bridge this gap we propose a 3D TAM that uses a small number of TSVs and TAM lines to transfer big volumes of test-data to the various dies at a high rate. This is achieved by the means of global test channels that start at the bottom die and end at the topmost die. Global channels consist of TSVs in the passive layers of the dies, and metal vias and buffers in the active layers of the dies. Test-data is time-multiplexed at each global channel at the bottom die, and it is transferred to the dies of the stack at the frequency supported by the TSVs. At each die the test-data is time-demultiplexed and distributed at every core using the slow shift-frequency

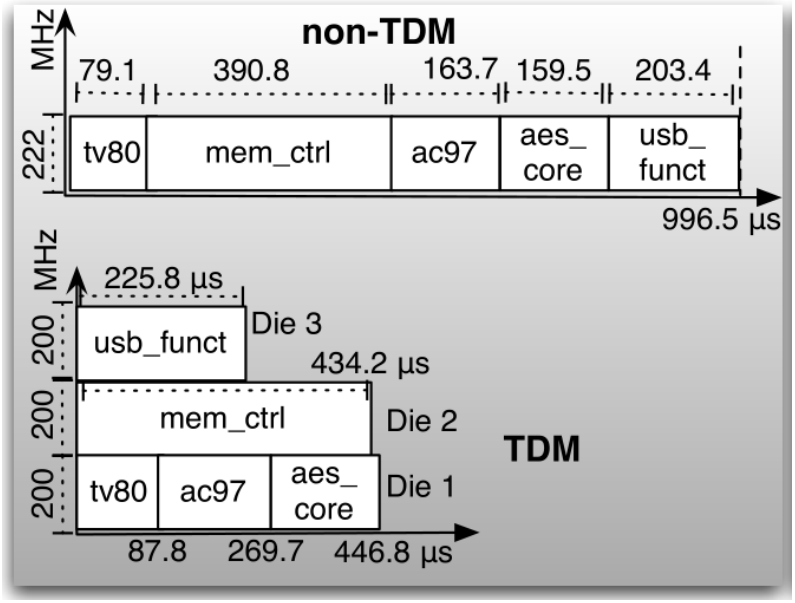


Figure 3.1: An example test-schedule with 8 TSVs.

permitted by the wrapper chain of the core and the TAM of the die.

The proposed TDM approach is applied in two dimensions, vertical and horizontal, and it gives a new perspective for testing 3D SoCs. In the vertical dimension, the test-data is time-multiplexed in a round-robin fashion and it is transferred through the TSVs; at the 1st clock cycle, the test-data of the 1st die is transferred; at the 2nd cycle, the test-data of the 2nd die is transferred, etc. In the horizontal dimension, the test-data for different cores is time-multiplexed at the specific clock cycles that the global channel transfers test-data for the die, and it is transferred horizontally to reach the cores of that die. The vertical TDM depends on the frequency supported by TSVs, while the horizontal TDM depends on the shift-frequency supported by the scan-chains of each core.

Example 1. Consider an example 3D-IC with 3 dies and 5 IWLS [148] cores: *tv80*, *ac97*, *aes_core* (die-1), *mem_ctrl* (die-2) and *usb_funct* (die-3). In Tables 5.3, 5.4 (see Section 5.1.2) we report the total average power consumption P and the test-time T for different shift frequencies SF and different number of wrapper chains (WC) $WC = 8, 16, 32$. The first shift-frequency reported for each core is the maximum shift-frequency supported by the scan and wrapper chains of the core. All 5 cores are connected to a single TAM structure consisting of three 8-bit input and three 8-bit output buses, one pair at each die, which are vertically connected using 8+8 TSVs. With any conventional (non-TDM) method and without any power constraints, test-

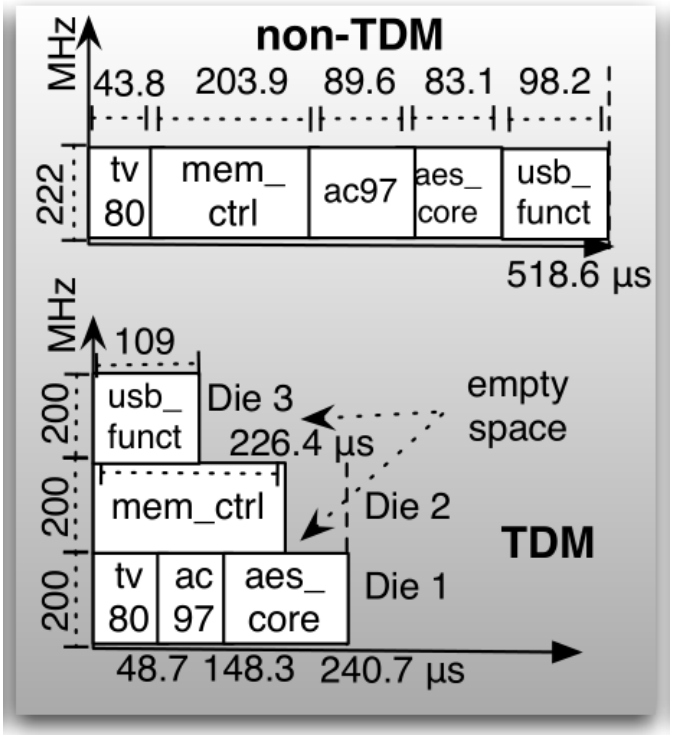


Figure 3.2: An example test-schedule with 16 TSVs.

data will be transferred at 222 MHz, which is the highest shift-frequency supported by every core. The total test-time is equal to almost 1 ms, as it is shown in Figure 3.1 (each block corresponds to a single test for one core with height proportional to the shift-frequency used and length proportional to the time taken by the corresponding test). Let us now assume that the TSVs can transfer test-data at 600 MHz. With the two-dimensional TDM technique, the test-data is vertically multiplexed at 600 MHz (each die receives test-data at 200 MHz). The test-data of the cores at each die is horizontally multiplexed by further dividing the frequency of 200 MHz. Therefore, the core *mem_ctrl* is tested in parallel with the other cores and the total time drops to 0.45 ms (see Figure 3.1). The results are similar when the number of TSVs increases to 16, as it is shown in Figure 3.2. Note that TDM reduces by a factor of 2x the number of TSVs and the routing resources (number of TAM lines) compared to the non-TDM case for the same test time, or alternatively it reduces by a factor of 2x the test time compared to the non-TDM case for the same number of TSVs and similar routing resources. ■

TDM permits a flexible control of the shift-frequency in order to maintain parallelism under power constraints. For example, when the average power constraint for testing each die is 6.8 mW, *aes_core* and *usb_funct* violate the power constraint for any

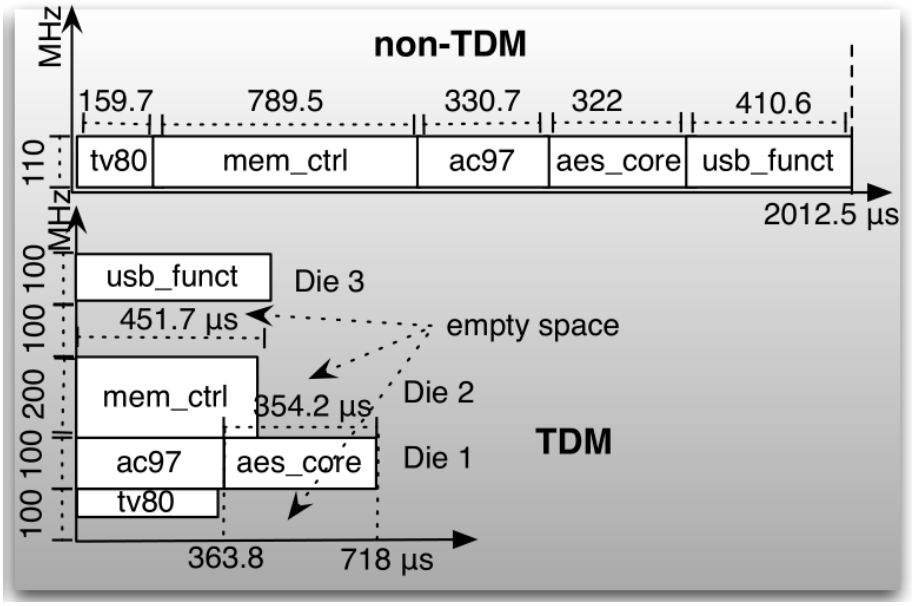


Figure 3.3: An example test-schedule with 8 TSVs and power constraints.

shift-frequency above 110 MHz. Therefore, with the traditional non-TDM approach, the maximum shift-frequency drops to 110 MHz, and the corresponding test-time is equal to 2 ms as shown in Figure 3.3. However, by using the two-dimensional TDM, the shift-frequency for each core can be set independently of the other cores. Therefore, *mem_ctrl* is tested at 200 MHz, *usb_funct*, *aes_core*, *ac97* are tested at 100 MHz and *tv80* is tested at 50 MHz without violating power constraints. As a result, the test-time is much lower than the Non-TDM case (0.72 ms), while there is available space to schedule additional tests.

3.3.2 Two-Dimensional Time-Division Multiplexing

The proposed TAM architecture is shown in Figure 3.4. Let N_{Dies} be the number of dies in the stack. The ATE channels are partitioned into groups, and each group loads one parallel-to-serial register of length L at the bottom die of the stack with frequency F_{ATE}. This register transmits serially the test-data through the Global TAM line using *Global Test Clock* (GTC). W parallel global TAM lines form a global channel of width W, which distributes the test-data at the local TAMs. *GTC* is a periodic signal with frequency $F_{GTC} = F_{ATE} \times L$, $L \geq 1$, which is divided into N_{Dies} local test-clocks LTC_1, LTC_2, ..., one for each layer (vertical TDM). The frequency of LTC, F_{LTC}, is lower or at most equal to the highest frequency supported by the TAM of the layer. F_{LTC} is divided at the local TAM at each layer (horizontal TDM) in order to shift

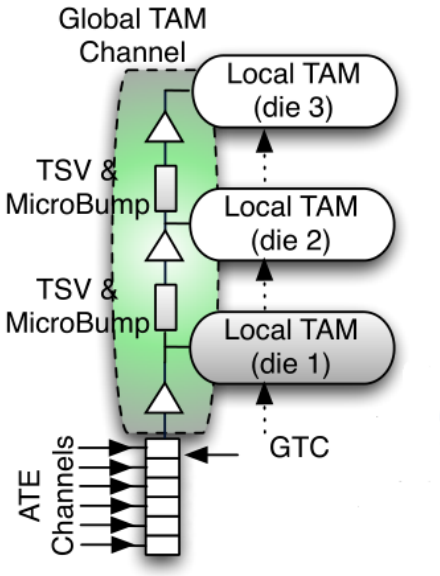


Figure 3.4: Global TAM circuitry of TDM architecture.

test-data into each core at the frequencies supported by the scan-chains under the power constraints of the core.

Example 2. Figure 3.5 shows one instance of a local TAM for the first tier of the 3D SoC used in Example 1. Each layer is assigned one cyclical shift-register of length N_{Dies}, which divides F_{GTC} by the number N_{Dies}. Specifically, the pattern "001" rotates inside the register and it drives LTC_1 (die-1), LTC_2 (die-2) and LTC_3 (die-3) with one active edge every N_{Dies} active edges of GTC. Then, each core is assigned one cyclical shift-register with length equal to 2^N, which divides F_{LTC} by a value equal to $2^0, 2^1, 2^2, \ldots 2^N$. The scan shift-frequency for every core is set by loading appropriate non-overlapping patterns into the registers. All shift-registers of dies 1, 2, 3 are clocked using LTC_1, LTC_2, LTC_3 respectively (with frequency F_{LTC}) and provide clock signals with frequencies equal to $F_{LTC}, F_{LTC}/2$ and $F_{LTC}/4$. At every cycle of *LTC*, W test bits are available at the common global and local bus. Since the patterns loaded into the registers are non-overlapping, only one layer is active at each GTC cycle and only one core loads the test-data from the bus at each LTC cycle. As shown in Figure 3.6, $F_{LTC} = F_{GTC}/3$ and the shift-frequency for Cores A, B and C is set equal to $F_{LTC}/4$ ($F_{GTC}/12$) with patterns 0001 and 0100, and $F_{LTC}/2$ ($F_{GTC}/6$) with pattern 1010, respectively. ■

Figure 3.7a shows the structure of a global channel carrying one test-bit in the 3D

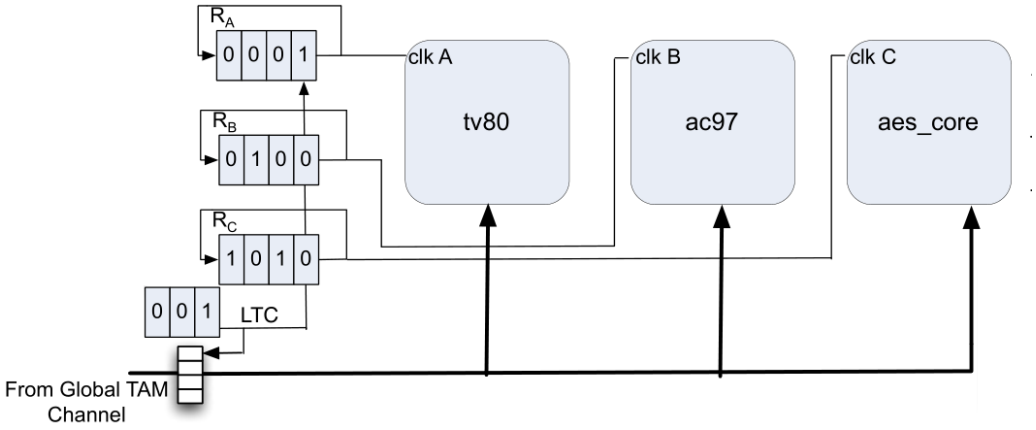


Figure 3.5: Local TAM structure of TDM architecture.

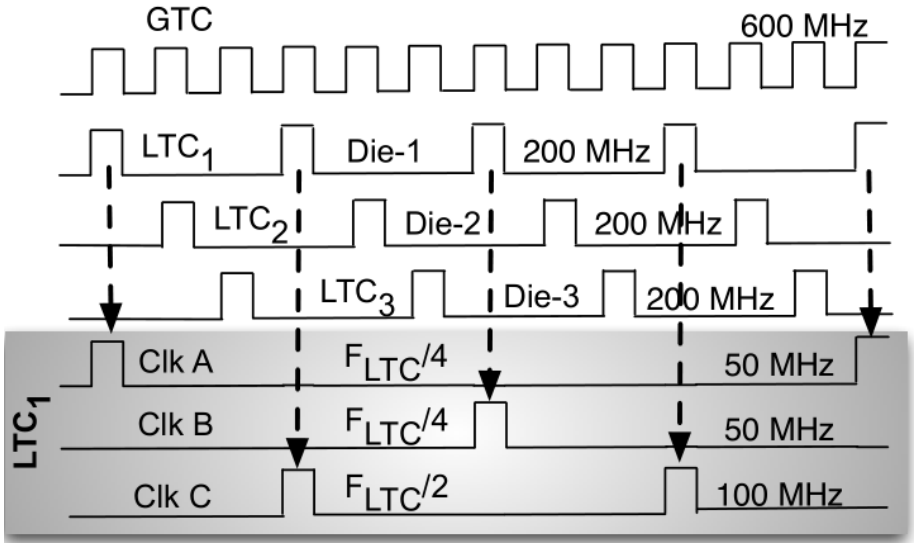


Figure 3.6: Clock frequency division of TDM architecture.

stack of Examples 1-2. All dies face downwards and they are connected face-to-back (other stacking orientations can be used as well). The global channel begins from the first metal layer of the bottom die, and it goes through a TSV and two micro-bumps to reach the top metal layer of the middle die. Then, through successive metal vias it reaches the first metal layer and the transistor layer of this die, where it is connected to the local TAM structure. Then through the next TSV and micro-bumps it is connected to the top metal layer of the third die. At each die, the signal of the global channel is transmitted both upwards and to the local TAM using buffers. The global channel ends at the first metal layer of the top-most die, where it is connected to the local TAM structure. The reverse direction is used for global channels carrying test responses out of the IC.

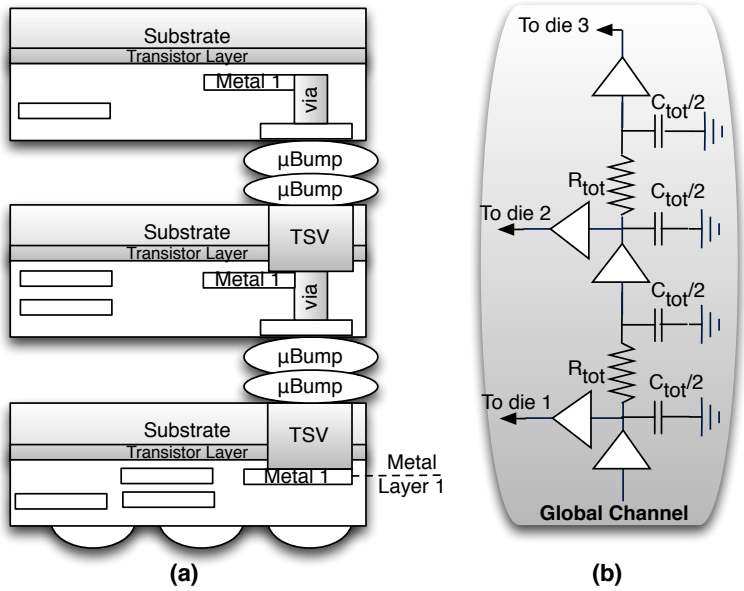


Figure 3.7: Global channel: (a) structure, (b) electrical model.

F_{GTC} can be as high as 3.45 GHz for stacks with 3 dies, 2.70 GHz for stacks with 4 dies and 2.27 GHz for stacks with 5 dies. This was determined through electrical simulations using the RC model and RC parameters reported in [149, 150] for TSVs of diameter 5 μm, and micro-bumps of 10 μm width as shown in Figure 3.7b. The digital gates are implemented in 45 nm technology and buffers with 6x driving strength are used. $R_{tot} = 136$ mΩ, $C_{tot} = 60$ fF represent the total resistance and capacitance of the TSV and the micro-bumps. The inductance of the TSV and the micro-bump can be ignored because the global channels operate in the low-GHz frequency range and very short die-to-die interconnects are considered (50 μm) [151]. The maximum shift-frequency for the IWLS cores was found using timing simulation to be in the range of [222, 400] MHz (see Tables 5.3, 5.4).

The proposed TAM architecture is modular and it can be used for testing both partial and complete stacks. In the particular case of partial stacks, each global test channel ends at the die that is at the top of the stack. For pre-bond testing the circuit at the bottom die that multiplexes test-data from the ATE must be replicated at each die of the stack. This circuit is very small, and it can be easily bypassed during post-bond testing. Finally, for 3D-ICs with their clock network split across different tiers the redundant pre-bond clock tree is used [106] for connecting each core with the clock generated by the TDM scheme. If the pre-bond clock tree is not available then the test clock inputs stemming from the other tiers must be bypassed and driven by

Table 3.1: Test times for Example 3.

SF	A	B	C	D
F	250	150	200	180
F/2	500	300	400	360
F/4	1000	600	800	720

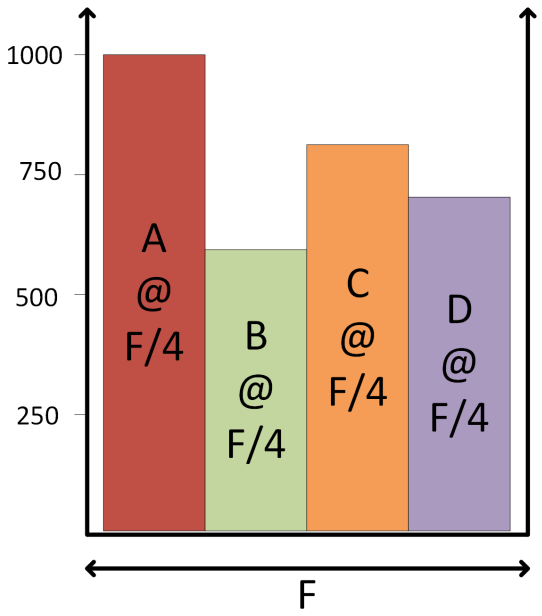


Figure 3.8: 1st test schedule for Example 3.

the TDM clock signal.

3.3.3 Test Scheduling Method

The proposed test-scheduling method is based on rectangle-packing (RP) enhanced with simulated annealing (SA). For every core C_i and every shift-frequency F_k we consider one candidate test $\tau_{C_iF_k}$, which is modeled as a rectangle $R_{C_iF_k}$ with width F_k and height $T_{C_iF_k}$ ($T_{C_iF_k}$ is the test-time corresponding to shift frequency F_k). Geometrically, the set of candidate tests for each core correspond to a set of rectangles with equal areas, and widths below a certain limit that corresponds to the maximum allowed shift-frequency in the scan-chains of the core. Every global channel is assigned one virtual bin for every local bus at each die. The width of each bin is set equal to F_{LTC} to ensure that rectangles with aggregate frequency higher than the frequency-capacity of the bus cannot be placed in parallel in the bin. The overall

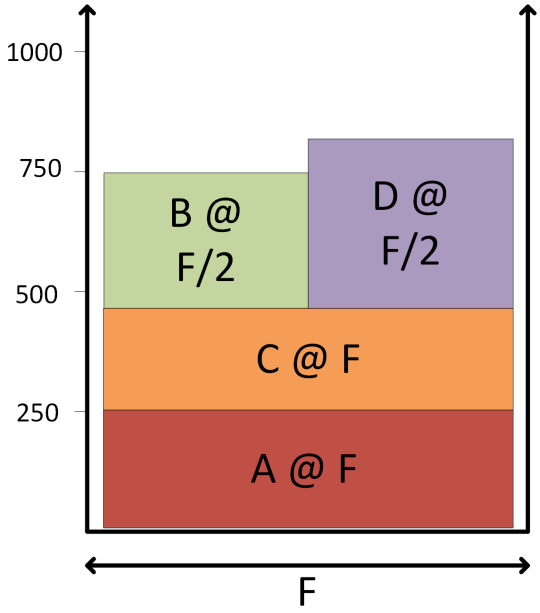


Figure 3.9: 2nd test schedule for Example 3.

occupied height in a virtual bin represents the test-time for the corresponding global channel, therefore the maximum of the derived test times, *TST*, per TAM bus is the SoC test-time. The selection of the shift frequency of each core affects the total test time.

Example 3. Table 3.1 presents the test times of cores A, B, C and D for Shift Frequency (SF) F, F/2 and F/4. If every core is tested with shift frequency F/4 the total test time is 1000 (Figure 3.8). In Figure 3.9 we present a different test schedule, where each core using a different shift. The total test time of the 2nd schedule is 810, achieving 19% reduction on 1st schedule. ■

The objective of RP is to pack a predetermined set of rectangles (one for each core) into the virtual bins, such that the occupied height in each bin is minimum. The packing of the rectangles is based on the Bottom-Left rule (BL) [152], where each rectangle is placed at the bottommost and leftmost position in the bin. Each rectangle is assigned a score for each possible placement of that rectangle in the virtual bin; the lower the y-coordinate of the top side of the rectangle at each position, the higher is the score for this particular placement. The rectangle with the highest score is placed in the bin and this procedure is repeated for the remaining rectangles.

The RP approach is very effective in minimizing the test-time for any given set of rectangles. However, different shift frequencies correspond to differently shaped

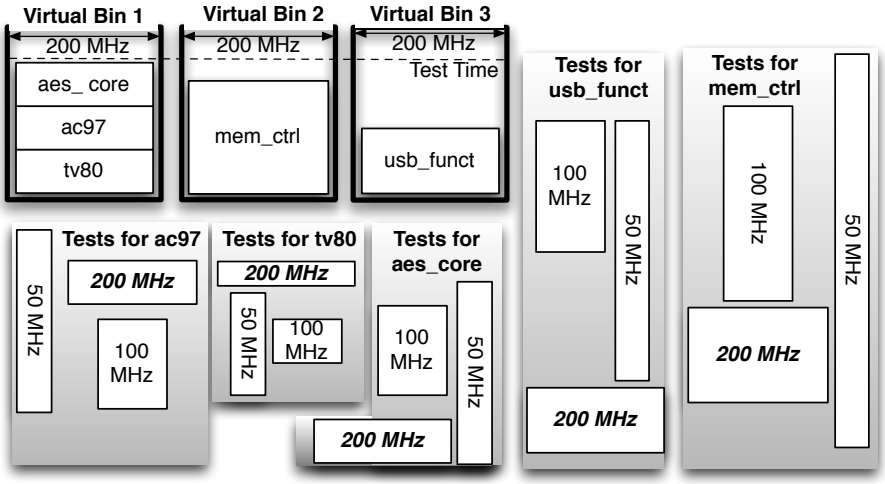


Figure 3.10: An example test-schedule.

rectangles for every test and they greatly affect the packing of the rectangles inside the bins. Unfortunately, this aspect is not addressed by RP. One very effective approach for identifying the best shift-frequency for each core is simulated annealing [153]. At first, one set S_{C_i} with all candidate tests $\tau_{C_iF_k}$ is formed per core C_i by varying F_k according to the frequencies provided by TDM. From each set S_{C_i} one test is selected in order to form a state for the SA approach. The initial state s_{init} includes from each set S_{C_i} the test with the maximum rated scan-frequency. The SA algorithm proceeds with the generation of a neighboring state s_1 for s_{init}, then a neighboring state s_2 for s_1, etc. Each such state corresponds to a different combination of shift frequencies for the various cores, and thus it better exploits the potential of RP to minimize test-time.

To create a neighboring state s_{p+1} for state s_p, r randomly selected candidate tests from state s_p are substituted with new differentiated candidate tests taken from the correspondent sets S_{C_i}. The SA algorithm is driven by the "*acceptance probability function, AP*", which is derived from the Maxwell-Boltzmann distribution [154] and determines if a state s will be accepted or not. AP depends on the test-time $TST(s_p)$ of the previous state s_p, the test-time $TST(s_{p+1})$ of the next state s_{p+1}, and the process temperature TH_P. We note that TH_P is not related to any thermal activity of the IC and it determines the likelihood of accepting states that are inferior to the previous states [153]. TH_P is calculated by the expression $TH_P = TH_P^{Init} \cdot CR$, where TH_P^{Init} is the initial temperature and CR is the cooling rate ($CR < 1$) used to reach a final temperature TH_P^{Final}, ($TH_P^{Init} > TH_P^{Final}$). A new solution is always accepted if it is better than the previous one. The AP value of a solution that is worse than the

Table 3.2: Power consumption for Example 5.

SF	A	B	C	D
F	400	260	380	320
F/2	200	130	190	160
F/4	100	65	85	80

previous one decreases a) with time (cooling process), and b) with the "distance" between the new (worse) solution and the old one. When TH_P reaches the value of TH_P^{Final} the SA algorithm ends and the final state is considered as the one that enables the RP algorithm to provide the lowest possible *TST*. Similar to [155, 156] we set $TH_P^{Init} = 400$, $CR = 0,999$, $r = 3$ and $TH_{Final} = 0,001$, because they provide consistently good results in short computational times.

Example 4. Figure 3.10 shows the representation of the test-schedule shown in Figure 3.1. Three virtual bins are used, one for each die, and the width of each bin corresponds to shift-frequency equal to 200 MHz. The left bin corresponds to die-1, the middle bin corresponds to die-2 and the right bin corresponds to die-3. The rectangles below the virtual bins represent the set of tests for each core for 200 MHz, 100 MHz and 50 MHz. Note that the test for any core at 200 MHz has the double (quadruple) width and the half (quarter) height of the test for the same core at 100 (50) MHz. The SA algorithm selects a combination of rectangles that best fit into the bins, and the RP algorithm places these rectangles inside each bin as shown in Figure 3.10. The height of the topmost rectangle among all virtual bins corresponds to the test-time of the 3D SoC. ■

When the dies of the stack are very unbalanced in terms of test data, the test-time can be minimized by dividing the TDM frequency among the dies in a non-uniform way. For example, when the volume of test data for the top die in Figures 3.4, 3.5, 3.6 is much larger than the volume of test data for the middle and bottom dies, then the 600 MHz of the TDM frequency can be split as follows: 300 MHz to the top die, and 150 MHz to each one of the middle and bottom dies. Such an approach increases the utilization of the bandwidth of the global TAM. Note that we use one local TAM at each die in the stack because it has been shown in [155, 156] that the test time is minimized when the TAM bus is connected to large numbers of cores.

The test operation of 3D-ICs is limited by power and thermal constraints to avoid

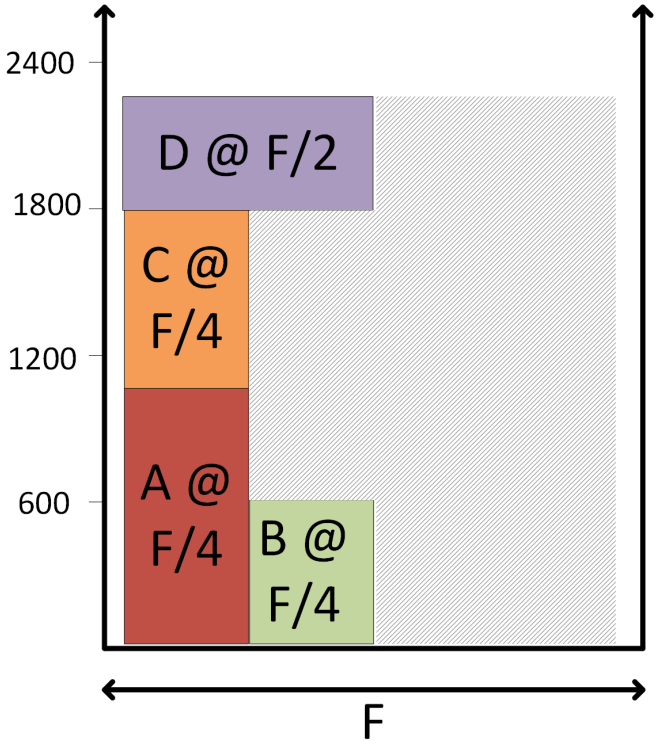


Figure 3.11: Test schedule for Example 5 under power constraints.

IR drop and high temperatures [157]. To this end we propose a power- and thermal-aware variation of the test-scheduling approach. At first, before we place rectangle $R_{C_iF_k}$ in the bin we ensure that it does not increase the average power consumption beyond the specified bounds. Let $ST_{C_iF_k}$ be the potential start time of test $\tau_{C_iF_k}$ and let $T_{C_iF_k}$ be the length of $\tau_{C_iF_k}$. Then, the average power consumption of every test $\tau_{C_jF_l}$ that is already scheduled during the period $[ST_{C_iF_k}, ST_{C_iF_k} + T_{C_iF_k}]$ is added to the average power consumption of $\tau_{C_iF_k}$ at the execution interval $[ST_{C_iF_k}, ST_{C_iF_k} + T_{C_iF_k}]$ of $\tau_{C_jF_l}$. If the tests scheduled during this interval violate the power consumption limit, the period $[ST_{C_iF_k}, ST_{C_iF_k} + T_{C_iF_k}]$ is rejected for test $\tau_{C_iF_k}$.

Example 5. We use the same cores of Example 3 and we present in Table 3.2 the power consumption of cores A, B, C and D for Shift Frequency (SF) F, F/2 and F/4. If the power consumption is constrained in 170mW for the die, then an acceptable test schedule is presented in Figure 3.11. In this test schedule, the total test time is 2160, which is $2.6\times$ the test time of the test schedule presented n Figure 3.9. The shaded area represents the frequency that is not exploited at all. ■

Furthermore we exploit the capability of the shift-frequency to control the thermal activity of each core. Specifically, by reducing the shift frequency the switching

activity at the internal nodes and the scan-chains (which is independent of the shift-frequency) is spread over a long period of time and the thermal activity of the core reduces. Even though the testing process is prolonged for those cores, TDM exploits the frequency released on the TAM to increase the shift-frequency and reduce the test-time of other cores. Additional test-time reductions can be achieved by the means of low-power DFT techniques, which reduce the power dissipated at each core during testing and increase thus the number of cores tested in parallel under predetermined power and thermal constraints.

Initially, the scheduler runs with power constraints and the generated schedule is evaluated using a thermal simulator. Thermal constraints are introduced for cores that violate thermal limits, by removing all the candidate tests corresponding to the highest (remaining) shift-frequency for these cores. Then the scheduler is being invoked again using the reduced set of tests for the thermally constrained cores, and the same process is repeated. For example, if core ac97 (see Figure 3.10) is thermally constrained then the test for ac97 at 200 MHz is removed from the set of candidate tests for this core. If it violates the thermal constraint again then the test at 100 MHz is also removed. This process terminates when the test-schedule generated does not violate the thermal constraint of any core.

Even though this heuristic is very efficient it does not guarantee optimality. Consequently there are cases where the proposed heuristic will not be able to minimize the test time. In such cases the test time can be further reduced by considering also the shift frequency of the cores at the vicinity of the hot cores. In that way the power dissipation and the thermal activity of the nearby cores will drop, and the temperature of the hot cores will further reduce. At the same time, the TDM scheme will exploit the available bandwidth in order to schedule more tests in parallel and reduce the overall test time.

3.4 K^3 TAM Optimization for Testing 3D-SoCs Using Non-Regular Time-Division-Multiplexing

3.4.1 Introduction

In this work, daisy-chains are utilized to efficiently connect the various cores to the global channels, instead of using the long, expensive and slow bus interconnections. In order to overcome the inherent incompatibility of TDM and daisy-chains, a novel TAM architecture is proposed to deliver the benefits of TDM to the cores of each layer, and a fully automated design process is proposed to minimize the total interconnection overhead of the TAM structure and the time required for testing each layer. The proposed $\mathbf{K}^3$ optimization process is based on two very effective heuristics, the **K**ruskal algorithm [158] and the Complete **K**armarkar-**K**arp heuristic [159], and it supports non-regular division of the frequency among the layers of the stacks. The proposed design-automation process eliminates the need for complex test-scheduling algorithms, and it is suitable for tight time-to-market constraints.

The organization of the rest of the section is as follows. Section 3.4.2 presents background material and motivation. Section 3.4.3 presents the 3D TAM architecture, while the Sections 3.4.4, 3.4.5 present the TAM optimization. In Section 5.1.3 an evaluation of the proposed method is presented.

3.4.2 Background & Motivation

TSVs serve as fast vertical communication interfaces between different dies in 3D stacks [145, 146, 147]. 2D-TDM [18] uses global test channels that extend from the bottom to the topmost die by using TSVs in the passive layers and metal vias-buffers in the active layers of the dies. Global channels support frequencies in the range of 2.27 GHz for stacks with 5 dies, up to 3.45 GHz for stacks with 3 dies [18]. Test-data are time-multiplexed and they are transferred to the dies in a round-robin fashion, using the frequency supported by the TSVs i.e. the 1^{st}, 2^{nd}, etc die receives the test data at the 1^{st}, 2^{nd}, etc clock cycle respectively. At each die the test-data are time-demultiplexed and distributed to the cores using the slow shift-frequency of the scan-chains.

Unfortunately, time-division-multiplexing requires a bus-based TAM infrastructure to support the direct connections between a) the input global channels and the

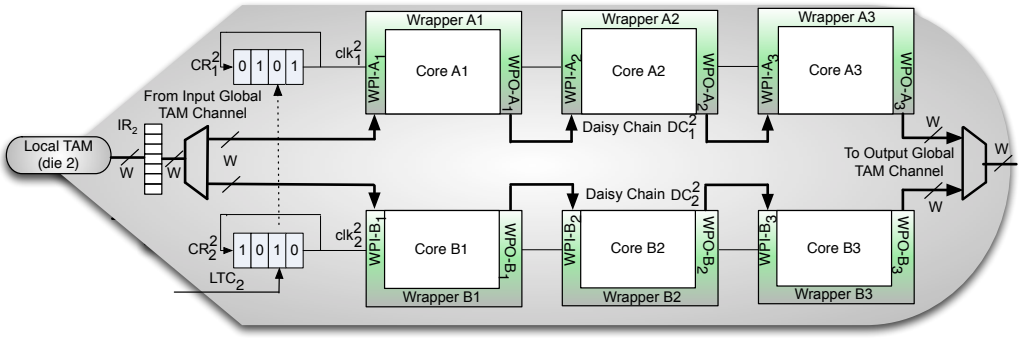


Figure 3.12: Local TAM structure of 3D TAM architecture.

WPIs, and b) the output global channels and the WPOs of the cores. Such long interconnections impose high routing overhead and excessive delays over the long metal lines and they require many buffers in order to travel through the entire die. Moreover, as shown in [18] global channels are more effective when they are time-shared by large numbers of cores (higher bandwidth-utilization is achieved), which further increases the routing overhead and the interconnection delays of the buses. As a result, the shift frequencies at the local TAMs are much lower than the frequencies supported by the global channels.

The proposed TAM architecture uses short and fast daisy-chains instead of the long and slow buses. The test-data are time-multiplexed at the global channels, and they are transferred with high frequencies to the dies, where they are time-demultiplexed and distributed to the daisy-chains. Every daisy-chain uses a divided version of the test clock to shift the test-data. At every time instance one core is selected at every daisy-chain to operate in test mode. At the same time instance the rest of the cores operate in bypass mode to transfer test-data from the input global channel to the selected core under test, as well as the responses of this core to the output global channel, where they are time-multiplexed with test responses from other layers. The time for testing the stack is equal to the maximum test-time among the layers, which is usually the test time of the most heavily-loaded layer. Therefore, the frequency of the global channels is divided non-regularly among the different layers (vertical direction) and the different daisy-chains (horizontal direction), in order to minimize the overall test-time.

3.4.3 3D TAM Architecture

The global TAM circuitry of the proposed TAM architecture is shown in Figure 3.4 for a 3D-SoC consisting of three dies ($N_{dies} = 3$). At the lower level the test-data enter the stack and they are transferred through a small number of TSVs to the various dies using high frequency. At every level of the stack the test-data are demultiplexed and they are shifted into the die using a division of the global-channel frequency. Then, they are distributed to multiple daisy chains using a second level of demultiplexing, and they are shifted into the cores by further dividing the shift frequency. During the shift/capture operations the daisy chains at every die are independently controlled using separate test clock and shift enable signals.

The ATE channels load one parallel-to-serial register at the bottom die of the stack with frequency F_{ATE}, which transmits serially the test-data through the Global TAM line synchronously with *Global Test Clock*, GTC (note that when $F_{ATE} \geq F_{GTC}$ the parallel-to-serial register can be omitted and each ATE channel drives directly one global TAM line). W parallel global TAM lines form a global channel of width W, which distributes the test-data to the local TAMs using the local test-clocks LTC_l generated for layers $l = 1, 2, \ldots, N_{dies}$. The local test clocks are divided versions of GTC generated by the global circular shift registers $GCR_1, GCR_2, \ldots, GCR_{N_{dies}}$. These registers circulate non-overlapping bit patterns that permit each clock cycle of GTC to be applied at a single die each time.

At every cycle of GTC W test bits are transmitted over the global channel, and they are distributed among the interface-registers IR by using the local test clocks. These W test bits are stored into one selected interface register IR_l upon the active clock edge of LTC_l. Let ND_l be the number of the daisy-chains $DC_1^l, DC_2^l, \ldots, DC_{ND_l}^l$ at layer l. The test-data stored into IR_l are distributed among these daisy-chains by further dividing the clock LTC_l into clock signals $clk_1^l, clk_2^l, \ldots, clk_{ND_l}^l$ (clk_j^l is used to shift the test-data from IR_l into DC_j^l). Similar to the division of GTC, the division of LTC_l is achieved by using circular shift registers with non-overlapping bit patterns, which enable LTC_l to be applied on one daisy-chain at a time. For example, in Figure 3.12 the test-data are alternatively shifted into DC_1^2, DC_2^2 by the means of two non-overlapping clocks clk_1^2 and clk_2^2 generated using circular registers CR_1^2 and CR_2^2 (similar register are used at every layer). Both of these registers shift non-overlapping patterns with the frequency of LTC_2, and generate two periodic signals with half the

frequency of LTC_2. Each logic value equal to '1' that reaches the rightmost cell of every circular register CR_j^l enables the test-data stored at IR_l to be shifted into DC_j^l.

DC_j^l connects a number of cores at layer l as follows: the WPI of the first core is driven by the local demultiplexing mechanism of the die, the WPO of the last core drives the output multiplexing mechanism of the die, and the WPIs (WPOs) of every intermediate core are connected to the WPOs (WPIs) of their predecessors (successors) cores. At every DC_j^l one core is tested at each time instance, and the test-data are shifted into this core by configuring the wrappers of the rest of the cores (at the same chain) in *bypass* mode according to the IEEE 1500 standard [24] (both the parallel and serial ports of the wrappers are connected in daisy chains in order to support compatibility with the IEEE 1500 standard). The test-time $T(DC_j^l)$ for shifting all test-data into DC_j^l is equal to the aggregate test-time of all the cores connected on chain DC_j^l. The design objectives of the TAM are the following:

- GTC must be divided among the die-layers and the daisy-chains in a non-regular manner, which depends on the test load and the constraints of each layer/daisy-chain.
- Multiple daisy-chains must be used at each layer in order to exploit the high TDM frequency of the global channels.
- The aggregate test-times of the cores at all daisy chains must be as balanced as possible.
- The interconnections of the daisy-chains among the cores must be as short as possible.

Since these objectives are conflicting, we propose an optimization approach to resolve them and optimize the TAM structure.

3.4.4 Layer-oriented 3D TAM Optimization

Let NC_l be the number of cores $C_1^l, C_2^l, \ldots, C_{NC_l}^l$ at layer l, and their test-times $TC(C_1^l), TC(C_2^l), \ldots, TC(C_{NC_l}^l)$ for a specific shift-frequency. The time $TD(DC_j^l)$ for testing the cores of DC_j^l is equal to the sum of their test-times. Since all daisy-chains of layer l shift test-data in parallel (in a time-multiplexed manner) the test-time $TT(l)$ is equal to the highest test-time among all the ND_l daisy-chains of layer l, i.e.

$TT(l) = \underset{j \in [1, ND_l]}{max} \{TD(DC_j^l)\}$. $TT(l)$ is minimized when the ND_l daisy-chains are as balanced, in terms of test-times $TD(DC_j^l)$, as possible. This problem is equivalent to the partitioning of the set of numbers $TC(C_1^l), TC(C_2^l), \ldots, TC(C_{NC_l}^l)$ into ND_l subsets, with the sum of the numbers in each subset be as nearly equal as possible to the sum of each of the other subsets (each subset corresponds to a set of cores connected in a separate chain).

The partitioning of the numbers into a) two, b) more than two sets can be tackled by the **K**armarkar-**K**arp [160] and the Complete **K**armarkar-**K**arp (K^2) [159] algorithms respectively. Specifically, the numbers $TC(C_1^l), TC(C_2^l), \ldots, TC(C_{NC_l}^l)$ are sorted in decreasing order, and a search following a k-ary tree is applied ($k = ND_l$). At each level of this tree a different number is assigned, and at each branch point one number is alternately assigned to one among the subsets. Each leaf of this tree corresponds to a complete partition. Let t be the sum of all the numbers, s the current largest subset sum, and d the difference of the best complete partition found so far. If $s - (t - s)/(k - 1) \geq d$ then the current branch is terminated. In order to minimize the time to find a good solution, at each branch the next number is put in one of the subsets by following increasing order of their sums. Every complete partition with a difference of zero or one is selected and the search is terminated. The K^2 algorithm is fast and provides very well balanced (in terms of test-time) daisy-chains, as it optimally solves the general number-partitioning problem [159].

The K^2 algorithm does not optimize the routing of the daisy chains as it does not consider the physical placement of the cores on the floorplan. For instance, in the daisy-chain of Figure 3.12 core A_1 should precede core A_2 ($A_1 \rightarrow A_2$) instead of the opposite, because WPO_{A_1} is physically closer to WPI_{A_2} than WPO_{A_2} is to WPI_{A_1}. Therefore, the cores of every daisy chain must be properly ordered in order to minimize the routing overhead of the TAM structure. To this end the interconnection overhead between any pair of cores A, B of the same daisy-chain is estimated based on the XY-coordinates of the WPIs and the WPOs of cores A, B on the floorplan. Specifically, the wire-length $WL(A_i, B_i)$ of the connection between the i^{th} bit of WPO_A with coordinates X_{A_i}, Y_{A_i} and the i^{th} bit of WPI_B with coordinates X_{B_i}, Y_{B_i} is estimated as the half perimeter (height plus width) of the minimum rectangle enclosing these two points, i.e., $WL(A_i, B_i) = |Y_{A_i} - Y_{B_i}| + |X_{A_i} - X_{B_i}|$. Depending on the routing congestion between these two points, this estimation can be a very close approximation due to the Manhattan routing adopted by all routers. The total

wire-length $WL(A,B)$ for connecting WPO_A to WPI_B (core A precedes core B) for a w-bit wide TAM is equal to $WL(A,B) = \sum_{i=1}^{w} WL(A_i, B_i)$.

For every pair of cores A, B that belong to daisy-chain DC_j^l the values $WL(A,B)$ and $WL(B,A)$ are calculated and a full directed weighted graph is generated as follows:

- every node corresponds to one core assigned to DC_j^l,
- every directed edge $A \rightarrow B$ and $B \rightarrow A$ corresponds to the connection between A, B where A precedes B and B precedes A respectively,
- the weight of the edge $A \rightarrow B$ and $B \rightarrow A$ is set equal to $WL(A,B)$, $WL(B,A)$ respectively.

The optimization objective is to identify the minimum-weight spanning-tree of this graph by the means of the **K**ruskal (K^1) algorithm [158]. Kruskal finds the minimum spanning tree of undirected graphs by adding increasing cost edges at each step [158]. On directed graphs the Kruskal algorithm selects the minimum cost edges $A \rightarrow B$ that do not form cycles, and for every directed edge $A \rightarrow B$ selected it discards all the outgoing edges from A to other nodes as well as all the incoming edges into B from other nodes. The complexity of Kruskal algorithm for E edges and V nodes is equal to $O(E \log V)$, which is very low for the problem at hand where $|V|$ is in the order of a few tens to a few hundreds at the most.

3.4.5 Stack-oriented (K^3) 3D TAM optimization.

The first step of the TAM optimization at the stack level (K^3) is to divide F_{GTC} into local test frequencies $F_1, F_2, \ldots, F_{N_{dies}}$ in a non-regular manner that depends on the specific test load and the test constraints of each layer. For example, let us assume that the three layers of the 3D SoC shown in Figure 3.12 require (approximately) $4/7$, $2/7$ and $1/7$ of the test data respectively (we assume that all TAM structures have the same bit-width). Then, F_{GTC} is divided as follows: the circular shift registers GCR_1, GCR_2, GCR_3 are 7-bit wide (7 is the common denominator of the ratios $4/7, 2/7$ and $1/7$), and they are loaded with the non-overlapping patterns "0101011", "1000100" and "0010000" respectively (each pattern has as many logic '1' values as the number on the nominator of the respective ratio). However, despite the fact that $F_1 = 4/7 \times F_{GTC}$, two successive clock edges of GTC are applied at LTC_1 (the last two

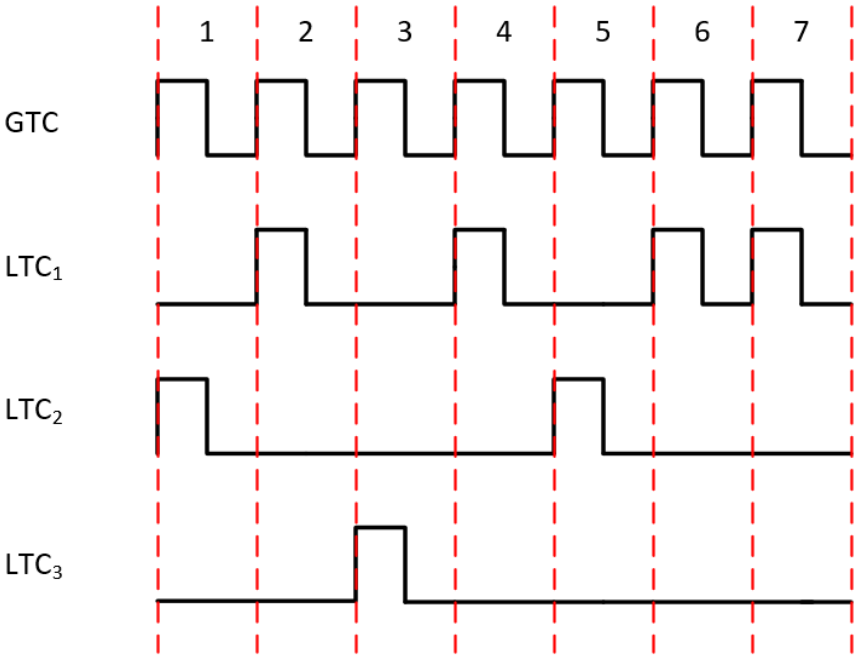


Figure 3.13: GTC division in layers.

bits of the pattern "0101011" are both equal to '1'), therefore F_1 switches between F_{GTC} (during the last clock cycle) and $F_{GTC}/2$ (during the rest of the clock cycles). In Figure 3.13 we present how LTC_1, LTC_2, LTC_3 are generated based on GTC and GCR_1, GCR_2, GCR_3. In the cases that the higher value of F_1 (F_{GTC} in the case at hand) cannot be supported by the local TAM design then another approximate division is selected, like for example $4/8$, $3/8$, $1/8$ for the case at hand.

After dividing F_{GTC} into $F_1, F_2, \ldots, F_{N_{dies}}$ the number ND_l of the daisy-chains for each layer l is set equal to $ND_l \geq \lceil F_l(max)/SF_{max} \rceil$, where $F_l(max)$ is the maximum shift frequency supported by the daisy-chains at layer l and SF_{max} is the max shift frequency supported by the cores of layer l. Then, the frequency for each daisy-chain is set equal to F_l/ND_l, the cores of every layer l are partitioned into ND_l daisy-chains by applying the K^2 algorithm, and the cores are ordered by applying the K^1 algorithm. Every ratio F_l/F_{GTC} for layer l must be nearly equal to the ratio of the test-data volume of layer l to the test-data volume of the whole stack.

The selected ratios provide nearly equal test-times for the dies $TT(1) \approx TT(2) \approx \cdots \approx TT(N_{Dies})$ and minimize the test time of the stack TT_{Stack} due to the parallel application of the tests using TDM, i.e. $TT_{Stack} = \max_{1 \leq l \leq N_{Dies}} \{TT(l)\}$. However, these ratios are very often not practical for frequency division, while there are also power constraints that prevent parallelization of certain tests. As a result, many values

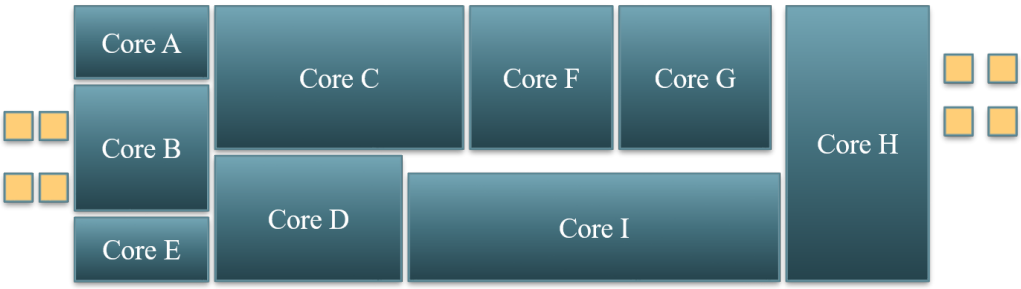


Figure 3.14: Cores layout for Example 1.

Table 3.3: Cores test time for Example 1.

Core	Test Time
A	22 ms
B	23 ms
C	8 ms
D	1 ms
E	8 ms
F	5 ms
G	15 ms
H	4 ms
I	2 ms

$TT(1), TT(2), \ldots, TT(N_{Dies})$ may deviate a lot from TT_{Stack} in practical applications. Nevertheless, this deviation provides a tolerance range for the lower test times to be increased without any impact on TT_{Stack}. This property is exploited to exchange cores among the ND_l daisy-chains with aim to shorten the long interconnections.

Let $A \rightarrow B$ be the longest daisy-chain connection of layer l. We begin from this connection and we try to exchange one of the cores A, B with a core C of another daisy-chain in order to replace this long connection with a short one. For all possible permutations $A \leftrightarrow C$ and $B \leftrightarrow C$ that do not violate the constraint $TT(l) \leq TT_{Stack}$ for $l = 1, 2, \ldots, N_{Dies}$, the new daisy-chains are generated by applying the K^2 algorithm and their total wire-length is evaluated as shown in Subsection 3.4.4. Among the permutations that reduce the wire-length of layer l, the permutation that offers either the highest wire-length reduction (criterion A) or the minimum increase of $TT(l)$ (criterion B) is selected, and the algorithm proceeds to the next permutation, until no further permutations are possible.

Example 1. In Figure 3.14 we present the layout of the cores composing a layer of a 3D SoC and in Table 3.3 we present the test time of each core. When we apply the K^2 algorithm the two daisy chains are formed as presented in Figure 3.15 and the total test time of each daisy chain is 44ms.

Afterwards, Kruskal algorithm is applied in order to define the sequence of the cores in each daisy chain. In Figure 3.16, we represent the positions of WPIs and WPOs on the layout with blue circles and red circles respectively. In Figure 3.17, we present the full directed weight graph that is generated initially. Kruskal algorithm selects the sequence of the cores step by step as presented through Figures 3.18-3.25.

In Figure 3.26 the sequence of the cores for both daisy chains is presented. At the stack-oriented 3D TAM optimization, if we exchange cores C, D, as presented in Figure 3.27 the total wirelength reduces, while the test time is increased from 44ms to 51ms. ■

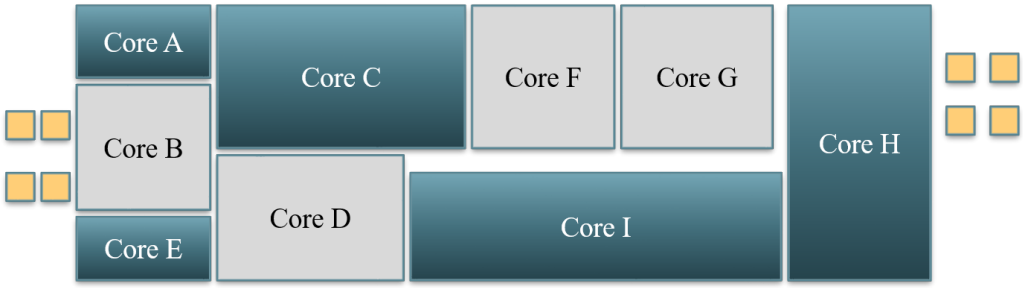


Figure 3.15: K^2 result for Example 1.

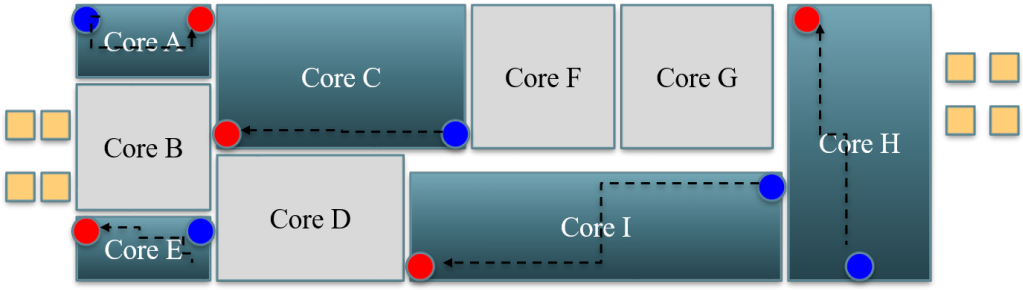


Figure 3.16: WPIs & WPOs of cores for Example 1.

The tests of the cores at each daisy-chain are scheduled in an order that adheres to the power constraints of the layer. For example, in the case of a layer with two daisy-chains the tests at each one of them are scheduled in reverse order of average power dissipation, i.e., ascending order in one daisy-chain and descending order in the other. If the power constraints are still violated, idle periods are inserted between

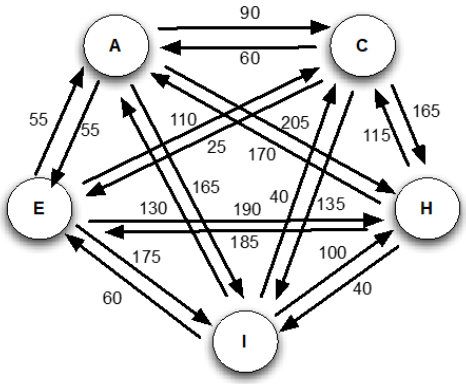


Figure 3.17: Initial graph for Example 1.

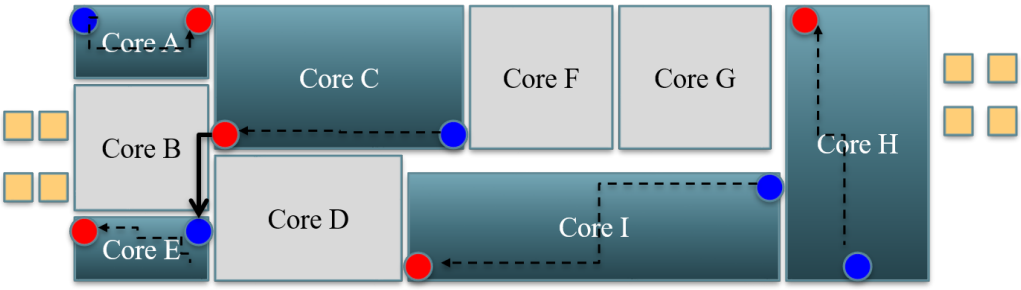


Figure 3.18: 1st step of Kruskal algorithm for Example 1.

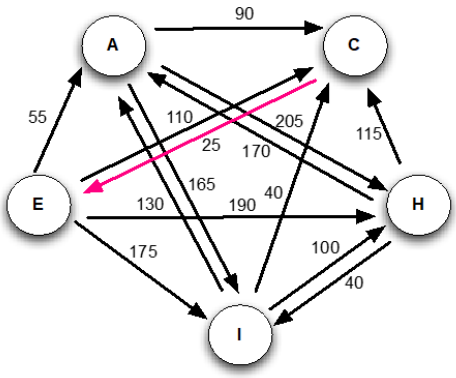


Figure 3.19: 1st edge selection of Kruskal algorithm for Example 1.

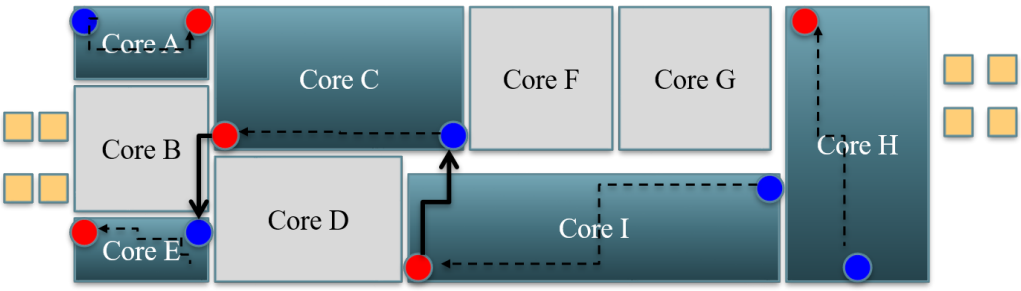


Figure 3.20: 2nd step of Kruskal algorithm for Example 1.

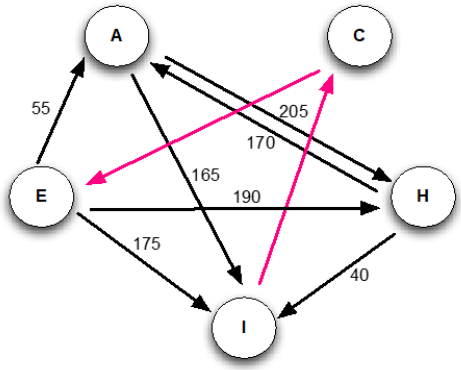


Figure 3.21: 2nd edge selection of Kruskal algorithm for Example 1.

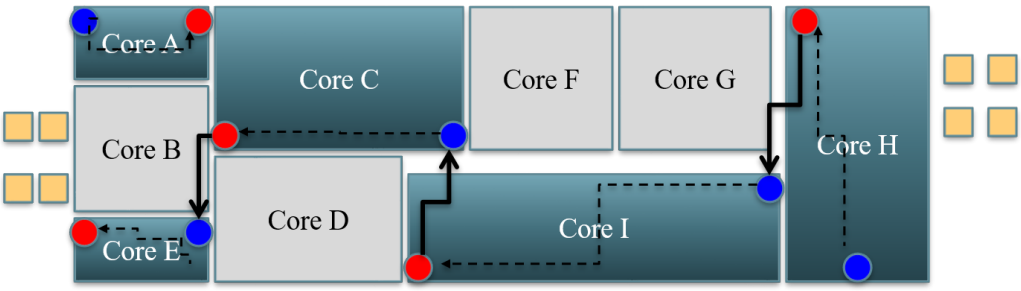


Figure 3.22: 3rd step of Kruskal algorithm for Example 1.

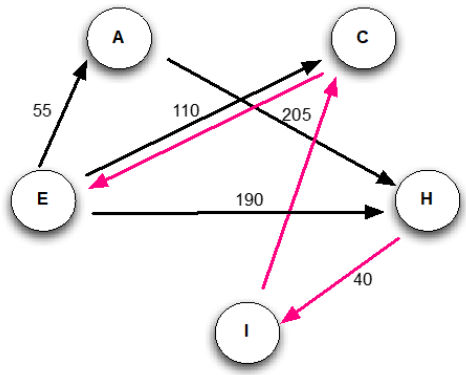


Figure 3.23: 3rd edge selection of Kruskal algorithm for Example 1.

tests, provided that $TT(l) \leq TT_{Stack}$. If for a certain value of TT_{Stack} a solution that adheres to both the wire-length and the power constraints of the design cannot be found then TT_{Stack} increases by small steps and the K^3 approach is applied repeatedly until a solution is found. At each step the number of possible permutations among the daisy-chains increases and the total wire-length drops. At the same time the power violations are gradually resolved (more idle periods can be inserted between the tests) thereby offering a trade-off between the test-time and the interconnection

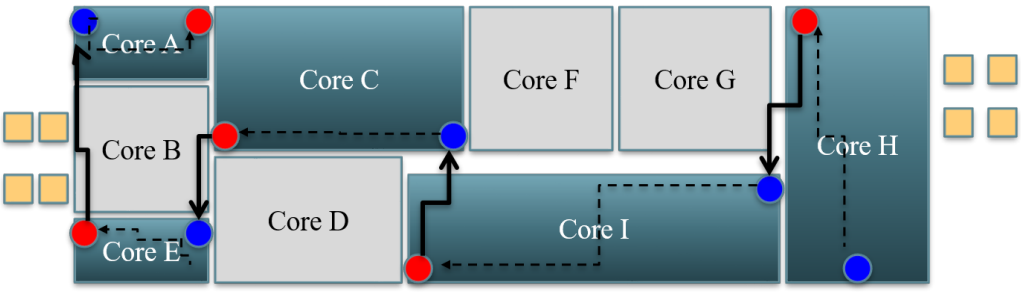


Figure 3.24: 4th step of Kruskal algorithm for Example 1.

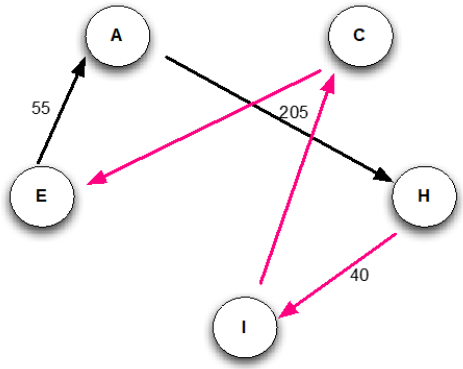


Figure 3.25: 4th edge selection of Kruskal algorithm for Example 1.

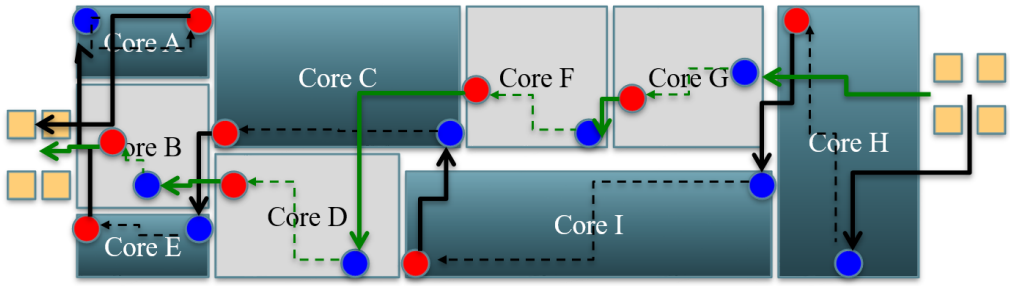


Figure 3.26: Cores sequence generated by Kruskal algorithm for Example 1.

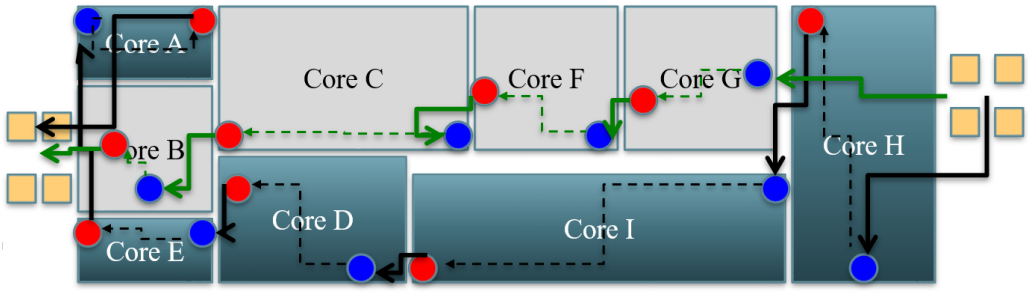


Figure 3.27: Daisy chains after permutation.

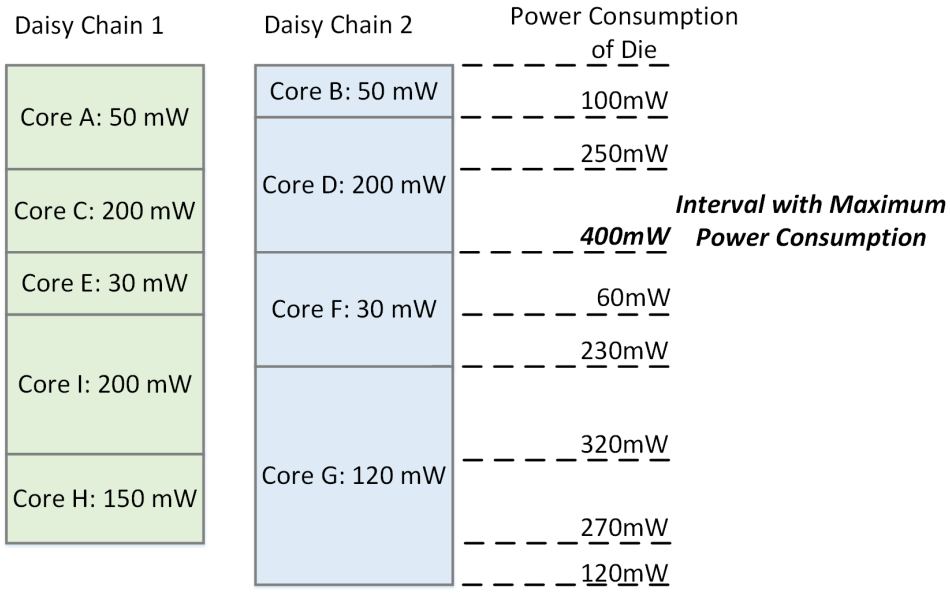


Figure 3.28: Daisy chains without power constraints.

overhead under power constraints.

Example 2. The cores of the two daisy chains we presented in Figure 3.26 have the power consumption presented in Figure 3.28. In Figure 3.28 there is no power constraint for the layer. If we limit the power consumption of the die to 270mW, we may change the order of the tests to satisfy the power constraint without increasing the total test time (Figure 3.29). However, if the power consumption of the die is further limited to 230 mW, we have to insert an idle period in the 2nd daisy chain (Figure 3.30), which results to increase the total test time. ■

The complexity of the optimization process is very low even for large stacks that consist of thousands of cores (note that the optimization process is applied at every layer separately and thus the running time of the proposed method depends on the number of cores at each layer). The complexity is very low even for very large future 3D stacks, because technology limitations favor the integration scaling at the vertical instead of the horizontal dimension, limiting thus the potential number of cores integrated at every layer of the stack.

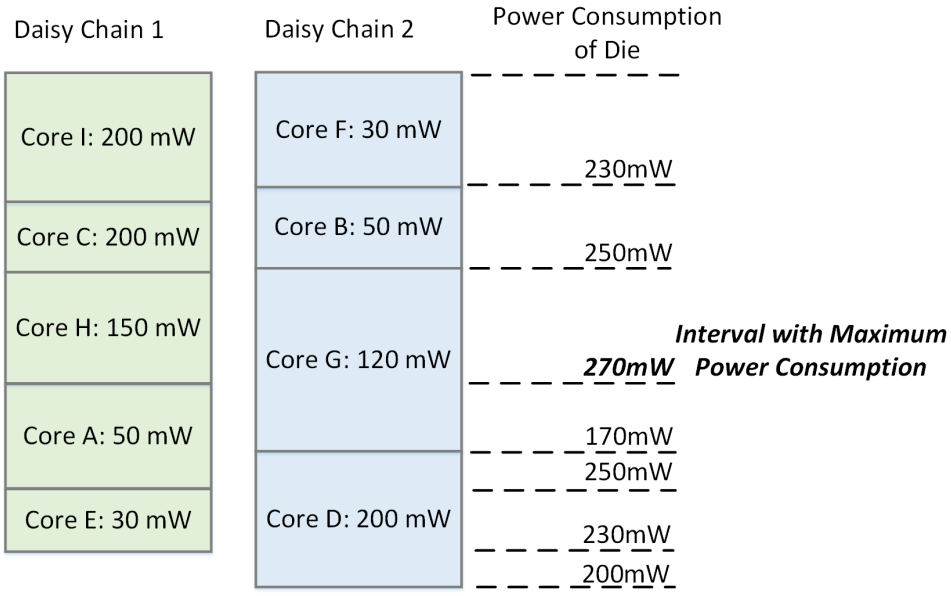


Figure 3.29: Daisy chains with power constrained to 270mW.

3.5 Fault-Independent Test-Generation for Software-Based Self-Testing

3.5.1 Introduction

Based on the SBST paradigm, several semiconductor and IP companies provide nowadays self-test libraries with their products, which can be easily integrated by their customer into the application code [161, 162, 163, 164, 165, 166]. The major challenge in the development of these self-test libraries is the generation of small test-programs that offer high fault coverage in short test-time [167, 168, 169]. SBST programs can be generated manually [170], semi-automatically [171] or automatically, targeting different processor architectures and fault models [172]. Various methods target microprocessors with caches [173], shared-memory schemes [174], floating-point units [175] and dual-issue processors [176]. Deterministic techniques exploit the regularity of sub-modules [177, 178, 179, 180, 181, 182, 183], while others use ATPG [184] and evolutionary algorithms [184, 185, 186]. Several methods explore the application of SBST to test peripheral modules [187, 188]. In [189] the effectiveness of SBST for a given level of dependability is evaluated.

The organization of the rest of the section is as follows. Subsection 3.5.2 presents background material and the motivation of this work. Subsection 3.5.3 presents the basic test generation flow. Subsection 3.5.4 presents the test generation for high delay-

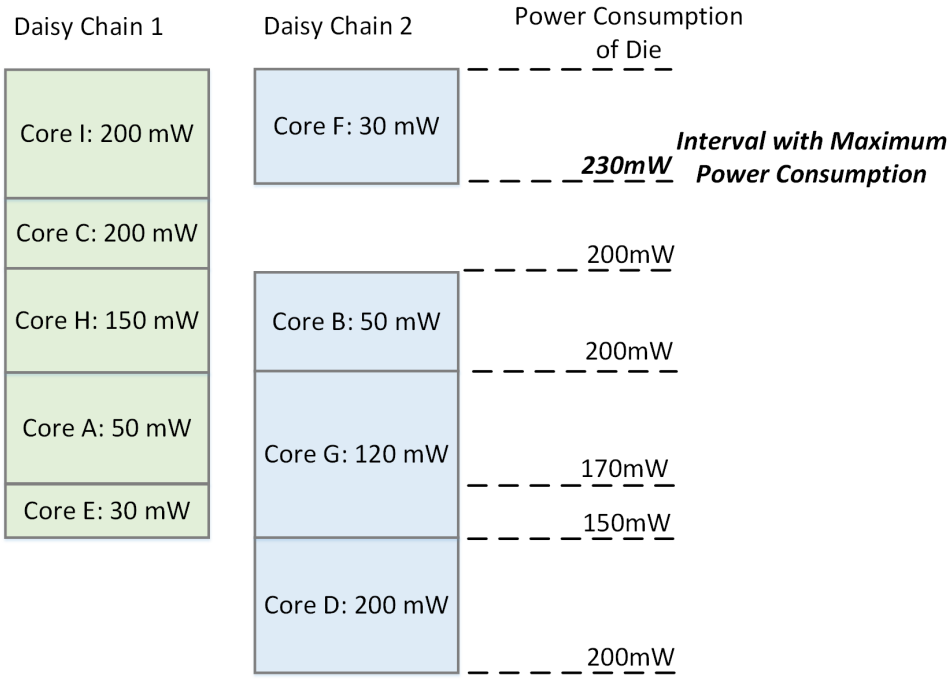


Figure 3.30: Daisy chains with power constrained to 230mW.

defect coverage. In Section 5.2 the proposed method is evaluated.

3.5.2 Motivation

The large computational overhead, the long running times and the high test-generation complexity prevent SBST methods from targeting other fault models than the stuck-at fault model. As it is shown in Section 5.2, even the simple task of fault simulating 50Kbytes of test code on the OpenRISC OR1200 processor using commercial tools, requires several days for stuck-at faults and even weeks for transition faults. By taking into account that test-generation is highly complex and even more CPU-intensive than fault simulation, we understand why most SBST methods target only the stuck-at fault model [136, 190, 172, 138, 169, 171, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 189, 191].

Even though targeting multiple fault models is a rather unrealistic goal for SBST, the defect coverage of the test programs can be enhanced by probabilistically evaluating their potential to detect arbitrary defects without targeting any particular fault model. Such an approach was proposed in [192, 193] for non-scan sequential circuits modeled at the register-transfer level (RTL). However, RTL models limit the effectiveness of these methods for detecting silicon defects. Moreover, these methods

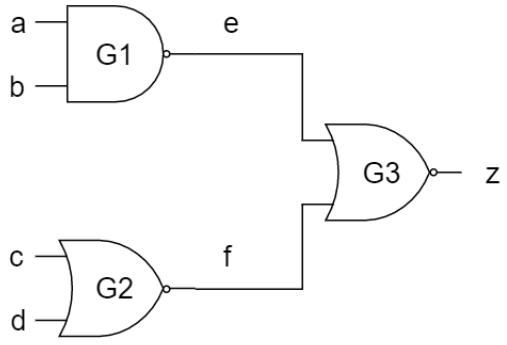


Figure 3.31: Logical circuit for Example 1.

Table 3.4: Signal probabilities for Example 1.

Input pattern	z	$p_{e,0}$	$p_{e,1}$	$p_{f,0}$	$p_{f,1}$	$p_{z,0}$	$p_{z,1}$
0000	0	0.1	0.9	0.2	0.8	0.886	**0.114**
0101	0	0.1	0.9	0.9	0.1	0.837	**0.163**
1111	1	0.8	0.2	0.9	0.1	**0.396**	0.604

require an ATE to apply the test sequences and to monitor the outputs at each clock cycle, whereas SBST implies the observation of the content of the data memory at the end of the test. Therefore, they are not suitable for SBST, which is by definition fully autonomous and ready to be applied in-the-field without the need of any ATE.

Gate-level output-deviations were shown to be very effective in detecting silicon defects in structural testing [194, 195, 196]. They are probability measures that reflect the likelihood of error detection at circuit outputs and they are computed without explicit fault grading, hence the computation is feasible for large circuits (the computational cost grows linearly with the number of gates). Initially, a probability map, the confidence-level (CL) vector, is assigned to every circuit gate. For every input pattern, each gate output i is assigned signal probabilities p_i^0, p_i^1 to be at logic 0 and 1, respectively. The CL vector R_i of gate G_i with m inputs has 2^m components $r_i^{0\ldots00}, r_i^{0\ldots01}, \ldots, r_i^{1\ldots11}$, each denoting the probability that the gate output is correct for the respective input combination. For example, r_i^{11} denotes the probability that the output y of the 2-input gate G_i is correct when the logic values at the inputs a, b of the gate are set equal to $ab = 11$. Every 2-input logic gate G_i is assigned a CL vector $R_i = (r_i^{00}\ r_i^{01}\ r_i^{10}\ r_i^{11})$. Let G_i be a NAND gate. Then the CL vector can be used to

define the probability that the output y is correct as follows:

$$p_y^0 = p_a^0 p_b^0 (1 - r_i^{00}) + p_a^0 p_b^1 (1 - r_i^{01}) + p_a^1 p_b^0 (1 - r_i^{10}) + p_a^1 p_b^1 r_i^{11}$$
$$p_y^1 = p_a^0 p_b^0 r_i^{00} + p_a^0 p_b^1 r_i^{01} + p_a^1 p_b^0 r_i^{10} + p_a^1 p_b^1 (1 - r_i^{11})$$

Note that $p_y^0 + p_y^1 = 1$. Signal probabilities can be computed for other gate types as well [194]. The gate-level CL vectors can be generated from simple transistor-level failure probabilities [194], or by using alternative ways, like layout information, inductive-fault analysis [197], and failure-data analysis.

For any gate G_i let its fault-free output value for input pattern t_j be d, with $d \in 0, 1$. The output deviation $\Delta G_{i,j}$ of G_i for t_j is defined as $P_{G_i}^{d'}$ (d' is the complement of d), and it is a measure of the likelihood that the gate output is incorrect for input pattern t_j. The deviation values at the circuit outputs are indicative of the probability for arbitrary defects to be detected at these outputs (the higher is the deviation value at an output, the higher is the likelihood of observing an error at the corresponding output).

Example 1. Figure 3.31 shows a circuit consisting of three gates G_1, G_2, and G_3, with two different confidence level vectors (R_1 and R_2) assigned to the NOR and NAND gates [194].

$$R_1(NOR) = (0.8\ 0.9\ 0.9\ 0.9)$$
$$R_2(NAND) = (0.9\ 0.9\ 0.9\ 0.8)$$

The first column of the Table 3.4 presents three test patterns and their respective fault free value at output z. The next six columns present the signal probabilities computed at the internal circuit nodes using the aforementioned confidence level vectors.

For each input pattern, the output-deviation is the probability for the output z to have incorrect value, which is shown in bold in the two last columns of the Table 3.4. Note that the deviation at output z is higher when the last test pattern is applied (it is equal to 0.396) as compared to the other two patterns (it is equal to 0.114 and 0.163 respectively). Therefore, the last test pattern $(a, b, c, d) = (1, 1, 1, 1)$ is the most promising for detecting defects as it provides the highest output deviation value. ■

A probabilistic-fault model F for a combinational circuit C is defined as follows: 1) Each gate G_i can fail independently of other gates and 2) the fault behavior of C is defined by R_i [194].

Table 3.5: Fault events for Example 1 under input pattern 0000.

Fault event	Fault event description	Event probability	Output value
E_0	G_1, G_2, G_3 fault-free	$0.9 \times 0.8 \times 0.9 = 0.648$	0
E_1	G_1, G_2 fault-free, G_3 faulty	$0.9 \times 0.8 \times 0.1 = 0.072$	1
E_2	G_1, G_3 fault-free, G_2 faulty	$0.9 \times 0.2 \times 0.9 = 0.162$	0
E_3	G_2, G_3 fault-free, G_1 faulty	$0.1 \times 0.8 \times 0.9 = 0.072$	0
E_4	G_1, G_2 faulty, G_3 fault-free	$0.1 \times 0.2 \times 0.8 = 0.016$	1
E_5	G_1, G_3 faulty, G_2 fault-free	$0.1 \times 0.8 \times 0.1 = 0.008$	1
E_6	G_2, G_3 faulty, G_1 fault-free	$0.9 \times 0.2 \times 0.1 = 0.018$	1
E_7	G_1, G_2, G_3 faulty	$0.1 \times 0.2 \times 0.2 = 0.004$	0

The probabilistic-fault model implies that the expected output values of the circuit in response to an input pattern are given by the signal probabilities at primary outputs. We should note that the circuit behavior is assumed to be deterministic after manufacturing and the probabilistic-fault model is only used during test development.

The circuit presented in Figure 3.31 can fail in a number of ways according to the probabilistic-fault model. Each failure way is termed a fault event. In Table 3.5, we list the various fault events $E_1, E_2, \ldots E_7$ and the fault free event E_0. Also, in Table 3.5, we compute the probability associated with each fault event and the corresponding output for input pattern 0000. The specific input pattern detects only the fault events E_1, E_4, E_5 and E_6. Let E be the event that the pattern 0000 detects a fault in the circuit. Then, it is

$$\begin{aligned} P[E] &= P[E_1 \bigcup E_4 \bigcup E_5 \bigcup E_6] \\ &= P[E_1] + P[E_4] + P[E_5] + P[E_6] \\ &= 0.114 \end{aligned}$$

as the fault events are mutually exclusive. Note that the output deviation for input pattern in Table 3.4 is also 0.114. This shows that the probability that an input pattern will produce an observable error at output for the probabilistic-fault model is directly proportional to the corresponding output deviation.

In this work we propose the first output-deviation-based metric that exploits the architectural and the gate-level models of the processor to evaluate SBST sequences.

Even though this metric can enhance the non-modeled fault-coverage of any SBST technique, we apply it on the particular case of *test macros* [171, 191] that have been proven to be very effective for stuck-at faults. A test macro is a sequence of assembly-level instructions, with one instruction executing a specific function and additional instructions that set the *macro parameters* (i.e., the operand values) and propagate the results to observable memory positions. At the architectural level, instruction-based *test macros* are synthesized that exercise various modules of the processor and observe the responses. Multiple instances of every test macro are generated by combining instructions that maximize the probability of detecting non-modeled faults, and by randomly varying their operands. Each test macro instance (TMI) is evaluated by means of a novel output-deviation-based metric computed on the gate-level netlist of the processor, and the most effective ones are selected to synthesize test programs, according to specific test-time and test-program-size constraints.

3.5.3 Basic Test Generation Flow

The basic test generation method consists of five steps and it is shown in Figure 3.32. Initially, one test macro is manually generated for every instruction of the processor following the guidelines presented in [171], which apply in every processor. This instruction is called hereafter the *Central-Instruction (CI)* of the macro and it exercises specific units of the processor. For example, the instruction *add* r_c, r_a, r_b of the OpenRISC OR1200 processor [198] executes the arithmetic operation $r_c = r_a + r_b$ (r_a, r_b, r_c are general purpose registers), and it exercises both the ALU and the control unit. If the result of the addition is observable, then this instruction constitutes a test for these units. The quality of this test depends on the contents of r_a, r_b, which are set by means of additional instructions, the *Wrapper-Instructions (WIs)*. The wrapper-instructions assign specific values to the operands of the central-instruction, and store the result at observable memory positions.

Example 2. The test macro generated for the instruction *add* r_c, r_a, r_b is shown in Figure 3.33. The first four instructions are wrapper-instructions that load the high/low 16-bit parts of registers r_a, r_b with the 16-bit values X_H, X_L, Y_H, Y_L. The next instruction is the central-instruction and the last instruction is a wrapper-instruction that stores the result at the physical address obtained by combining the immediate offset Z with register r_0 that holds always the value 0. ■

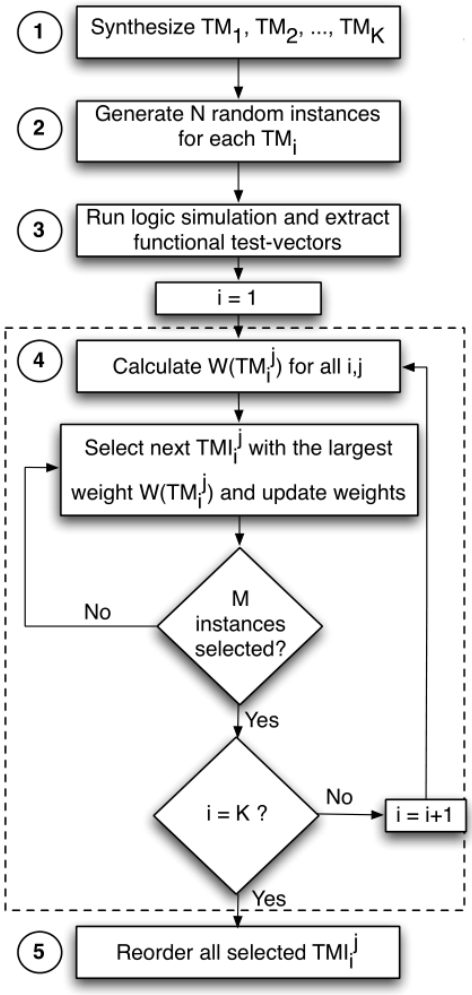


Figure 3.32: Test flow.

Similar macros are generated for every instruction of the processor, with two exceptions. The first one is related to wrapper-instructions. Such instructions do not necessarily become central-instructions, because the faults activated by them are observable when they are executed as wrapper-instructions. Second, the central-instructions that do not produce directly observable results (e.g. branch instructions) use wrapper-instructions to provide the observable results. For example, upon correct execution of a branch-based central-instruction, a wrapper-instruction stores a pre-determined value at an observable memory position, thereby making the results of the branch instruction observable. Even though the generation of the test macros requires some knowledge of the instruction set and the architecture of the processor, it is an one-time and rather simple task for every processor architecture.

The fault coverage of every test macro depends on the operands of the central and wrapper-instructions. Unfortunately, there is no straightforward method to select the

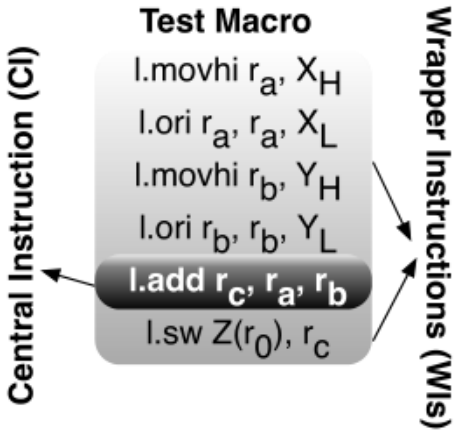


Figure 3.33: TMI example.

most effective operands for detecting non-modeled faults. To this end, we propose an almost fully automated process to generate test macro instances ($TMIs$) with high non-modeled fault coverage. Let K be the number of different macros TM_1, TM_2, ..., TM_K generated at the first step (one for every CI). Then, N random instances $TMI_i^1, TMI_i^2, \ldots, TMI_i^N$ are generated for every TM_i at the second step, by randomly varying every operand of TM_i. For the particular macro shown in Figure 3.33 the $TMIs$ are generated by varying registers r_a, r_b and the values for X, Y, and as a result register r_c and the value for Z also change (we note that whenever a TMI with invalid values is generated it is discarded).

At the third step, we run logic simulation on the gate-level netlist of the processor using the $N \times K$ $TMIs$ generated at the previous step, and the logic values generated at the inputs/outputs of selected units of the processor are recorded at the clock cycles when these units are excited by each TMI_i^j. For example, when the CI shown in Figure 3.33 is executed, the inputs/outputs of the ALU are recorded during the clock cycle when the arithmetic values stored into registers r_a and r_b are applied to the inputs of the ALU (see Figure 3.35). These logic values constitute one functional test-vector/response applied by the TMI. Every TMI generates a number of functional test-vectors/responses that excite various units of the processor and propagate the results to observable sites of the processor. These vectors are generated during the clock cycles when their responses are observable either immediately (e.g. the output of the ALU stored into register r_C) or later through the wrapper-instructions.

One example is shown in Figure 3.34 for the test macro of Figure 3.33. The first four instructions require two clock cycles to be executed and the functional test-

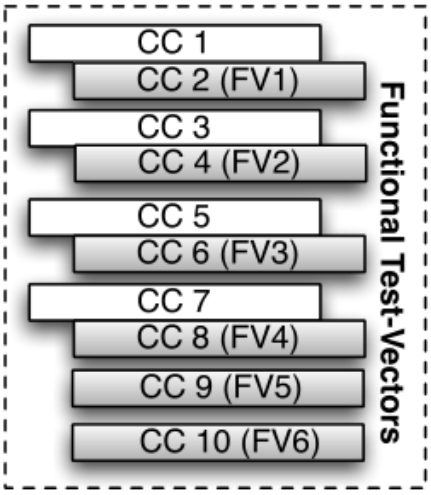


Figure 3.34: Functional vectors.

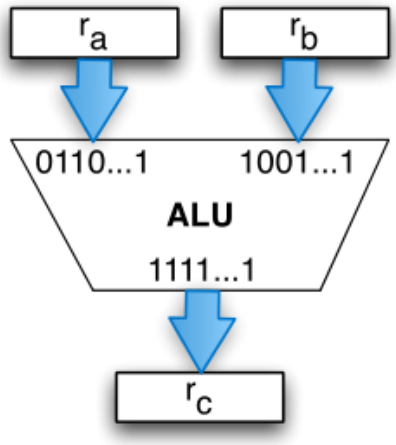


Figure 3.35: ALU test.

vectors are generated at their second clock cycles. The last two instructions require one clock cycle to be executed and the functional test-vectors are generated at both clock cycles. The more effective are the functional test-vectors of TMI_i^j in detecting defects, the higher is the test-quality of TMI_i^j. The potential of TMI_i^j for detecting non-modeled faults depends on the quality of the functional test-vectors/responses generated by TMI_i^j. When a functional test-vector generated by TMI_i^j activates a defect and propagates the error to the output of the exercised module, then there is a high probability that this error will propagate to an observable site of the processor.

In order to evaluate the functional test-vectors of the $TMIs$ we propose a new output-deviation-based metric. Specifically, the combinational logic of every processor unit exercised by a functional test-vector is used to calculate the deviations of its

outputs for this test-vector, as it is shown in Subsection 3.5.2. For example, in Figure 3.35 we show the functional test-vector $FV5$ generated during the execution of the CI shown in Figure 3.33, where the ALU receives inputs from registers r_a, r_b and it stores the result in register r_c. The computation starts from the inputs to the outputs of the ALU (note that, in the general case, the mapping between inputs and outputs of every combinational block may not be trivial).

According to the theory of output-deviations the most effective test-vectors are the test vectors that produce the highest deviation values at the outputs of the circuits under test [194]. In order to identify those vectors, we apply first all the generated functional vectors that excite one unit, and we calculate the maximum deviation value that is generated at every output bit p of that unit. This process is applied separately for every TM_i, because different test macros exercise different parts of the units. Let NV_i be the number of functional test-vectors generated by TM_i. In the particular case of TMI_i^j the functional vectors $FV(TMI_i^j, 1)$, $FV(TMI_i^j, 2)$, ..., $FV(TMI_i^j, NV_i)$ are generated. The output deviations of $FV(TMI_i^j, k)$ are computed for $j \in [1, N]$, $k \in [1, NV_i]$, and the proximity of each deviation value to the highest deviation value found at every output among all the generated functional vectors is calculated. Let $Max(TM_i, p, v)$ be the highest deviation value found for all functional test-vectors of every instance TM_i^j of TM_i ($j \in [1, N]$) at output p for logic response v ($v = 0, 1$). Then, for each $FV(TMI_i^j, k)$ all the pairs (p, v) with deviation values $Dev(FV(TMI_i^j, k), p, v)$ higher than a threshold value constitute the set $MS(TMI_i^j, k)$ (the rest of the pairs are not further considered for this functional test-vector). This threshold value THR is a percentage of the highest deviation value $Max(TM_i, p, v)$ at this output, i.e., $Dev(FV(TMI_i^j, k), p, v) \geq THR \times Max(TM_i, p, v)$, with THR usually in the range $90\% - 100\%$. Note that the higher is the value of THR, the more strict is the selection process towards pairs (p, v) with high deviation values.

Besides the high deviation values, the potential of $FV(TMI_i^j, k)$ to detect defects at the outputs p and logic response v with $(p, v) \in MS(TMI_i^j, k)$ depends on two additional parameters. The first one is the number of faults that are observable at output p, $Faults(p)$, which is proportional to the size of the logic cone driving p. This parameter is modeled by setting $Faults(p)$ equal to the number of gates in the fan-in cone size of p. The second one is the number of functional test-vectors generated by all random instances of TM_i that provide high deviation for each pair (p, v). The higher is this number, the higher is the probability that, eventually, some of the selected

$TMIs$ will provide functional test-vectors with high deviation values for this pair. Therefore, this output-logic value pair is considered as an easy pair, and it is given low priority $PR(TM_i, p, v)$ to bias the selection towards $TMIs$ that embed functional test-vectors with high deviation values at more difficult pairs. $PR(TM_i, p, v)$ is set equal to the inverse of the proportion of all functional test-vectors generated by every random instance of TM_i that offer high deviation value for (p, v). $FV(TMI_i^j, k)$ is assigned a weight equal to

$$WFV(TMI_i^j, k) = \sum_{(p,v) \in MS(TMI_i^j, k)} Faults(p) \times PR(TM_i, p, v) \tag{3.1}$$

Note that the weight increases as the fan-in cone-size of p and the priority of (p, v) increase. Then, a weight is assigned to TMI_i^j equal to the sum of the weights of its functional test-vectors

$$W(TMI_i^j) = \sum_{k=1...NV_i} WFV(TMI_i^j, k) \tag{3.2}$$

The higher is the value of $W(TMI_i^j)$, the more effective is the instance TMI_i^j for detecting non-modeled faults.

Every instance TMI_i^j with high weight $W(TMI_i^j)$ is expected to detect many defects at the outputs $p \in MS(TMI_i^j, k)$, i.e., the outputs with high deviation values at the functional test-vectors generated by TMI_i^j. When this instance is selected, the potential of the rest of the instances of the same test macro to detect defects at the same outputs $p \in MS(TMI_i^j, k)$ decreases (less defects are anticipated to remain undetected at the logic cones of these outputs). To reflect this fact the number of faults $Faults(p)$ at the logic cone of every output $p \in MS(TMI_i^j, k)$ is divided by a constant factor F after TMI_i^j is selected. This reduces the weight of all the functional test-vectors that provide a high-deviation value at output p, since their effectiveness for detecting defects drops after the selection of TMI_i^j. The higher is the value of F, the higher is the priority given to instances with high deviation values at other outputs. Therefore, all the weights are re-computed by applying again eq. (3.1), (3.2) and the next instance with the highest weight is selected. This process is iterated until M instances are selected for every test macro.

The M most effective instances selected from every test macro maximize the defect coverage at the particular processor units excited by the respective test macro under test-program-size constraints. Therefore, in order to achieve high defect-coverage

ramp-up for the whole processor, the test macro instances must be applied in the specific order that maximizes the defect-coverage at all processor units at the same time. To this end, after the M most effective instances of every test macro are selected, all the $M \times K$ instances are evaluated again and they are re-ordered by considering the whole processor circuit. In particular, all the $M \times K$ instances are evaluated together by considering all the targeted units of the processor. First, we reset the values of $Faults(p)$ at their initial values and the weights $W(TMI_i^j)$ are re-computed. Then, $M \times K$ iterations are applied, and at each iteration the instance with the highest weight is selected as the next in the order, and the values of $Faults(p)$ are updated as shown before. We note that the final order of the test macro instances depends on both the size of the excited units and the quality of the generated functional vectors, as it is evaluated by the proposed output-deviation based metric.

Even though the test program generation is not fully automatic, it requires only limited designer/test-engineer intervention, mostly during the development of the test macro templates. However, this intervention is at the architectural level and it does not require the specific gate-level or transistor-level model of the processor, which decouples the proposed method from any implementation details. We note that, such an intervention is an one-time task, while the rest of the test-generation method (which is usually applied multiple times during the development process) is fully automatic.

3.5.4 Test generation for High Delay-defect Coverage

Even though the test macros generated using the method proposed in Subsection 3.5.3 are very effective in detecting non-modeled faults, they offer limited detection of delay-faults in execution units like the ALU. Detection of delay-faults requires pairs of test-vectors to be applied to the inputs of the exercised units. Therefore, delay-fault oriented test macros are developed that embed pairs of central instructions exercising the processor units in successive clock cycles with different test-vectors. Such macros are called *Independent-Test Macros* (*I-TMs*).

Even though *I-TMs* can be generated for most of the processor's units, some units are accumulator-type units (like the Multiply-Accumulate unit of the OR1200 processor), which are designed to execute successive operations of accumulator-type only. Such operations require one operand of the second central instruction to be the

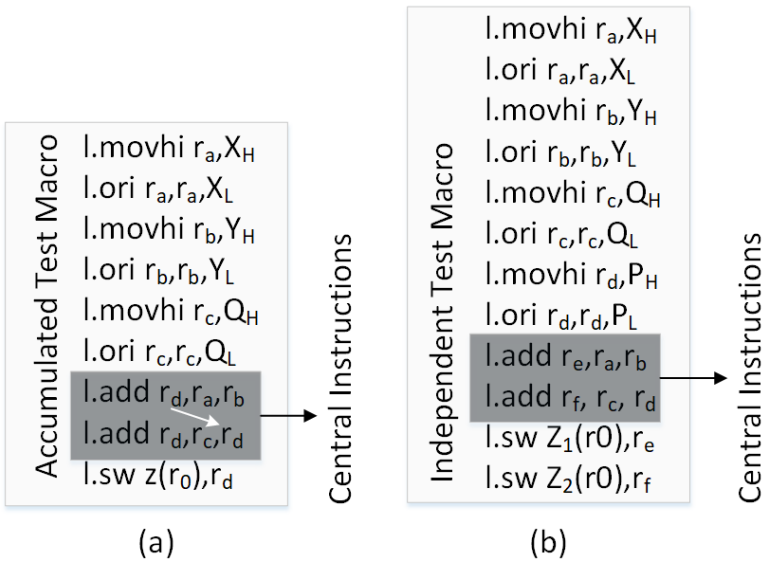


Figure 3.36: (a) A-TM example (b) I-TM example.

result of the execution of the first central instruction. For such units $I-TMs$ are inapplicable, and *Accumulated-Test Macros* (*A-TMs*) are used instead. Each $A-TM$ embeds two similar successive central-instructions, where one operand of the second *CI* is the result of the first *CI*.

Figure 3.36 presents an example of an *A-TM* and an *I-TM*. Since both *CIs* in every *A-TM* instance (*A-TMI*) consist of the same instruction, the test-generation flow is exactly the same with the flow shown in Figure 3.32 except of the additional functional test-vector that is generated by the second CI. However, in the case of *I-TMs* separate evaluation and selection of the first and second central-instructions is required (note that both of them detect defects, but most of the delay-defects are detected by the second central-instruction). To this end, different priority values PR and sets MS of outputs with high deviation values are manipulated for the first and the second central-instruction of all *I-TMIs*.

The generation of the *I-TMIs* is done as follows: multiple random instances of regular *TMIs* with a single central-instruction are generated, but only the functional test-vector of their central-instruction is evaluated using equations (3.1), (3.2). The two central-instructions with the highest weights are combined to generate an *I-TMI* (the central-instruction with the highest weight is used as second in the pair to favor the detection of delay defects). Then, the selected central-instructions are removed and the same process is repeated for generating the next N *I-TMIs* using the remaining central-instructions.

Even though $A-TMs$ can be potentially used to test all the units of the processor, $I-TMs$ are conjectured to offer higher delay fault coverage than $A-TMs$ because they do not suffer from the correlation between the operands of the first and the second central-instruction. Moreover, $A-TMs$ can be considered as a specialization of $I-TMs$, because every $A-TMI$ can be substituted by an equivalent $I-TMI$ with one of the operands of the second CI be a value equal to the expected result of the execution of the first CI. Therefore, $A-TMs$ are used only for accumulator-type units, and $I-TMs$ are used for the rest of the units.

The synthesis of $A-TMs$ and $I-TMs$ is straightforward for computational units like the ALU but there are various non-computational units that require the synthesis of special test macros in order to detect delay faults. One such example is the *Load-Store* unit of the OR1200 processor, which transfers the data between the processor and the memory. Even though some delay faults of this unit are exercised by the wrapper instructions of the test macros generated for other units, most of them remain undetected unless specific test macros are synthesized for this unit. One test macro example targeting delay-faults at this unit is presented in Figure 3.37a. The main purpose of this macro is to exercise the *Load-Store* unit by the means of different memory addresses and data transferred between the registers and the memory. Specifically, the wrapper-instructions load the registers r_a, r_b with the 32-bit values $X(X_H, X_L)$ and $Y(Y_H, Y_L)$, while the central instructions transfer these data between the registers and the cache memory of the processor (the value X is moved from register r_a to a memory position defined by the contents of r_b, then it is transferred to register r_c and finally from register r_c to memory position 0).

Another unit that requires the synthesis of special test macros is the *Control* unit. Similar to the *Load-Store* unit, the *Control* unit is also exercised by the wrapper and central instructions of all test macros, but it cannot be sufficiently tested for delay defects unless multiple combinations of different instructions targeting different units are applied in consecutive clock-cycles. In Figure 3.37b, a test macro example for the *Control* unit is presented. The first eight instructions are wrapper-instructions that load the registers r_a, r_b, r_c, r_d with the 32-bit values X, Y, Q, P, then two arbitrary central-instructions follow that apply an arithmetic and a logical computation using these registers, and the last two wrapper-instructions store the results of the computations at the physical addresses Z_1, Z_2. In a similar manner test macros can be synthesized for every unit of the processor.

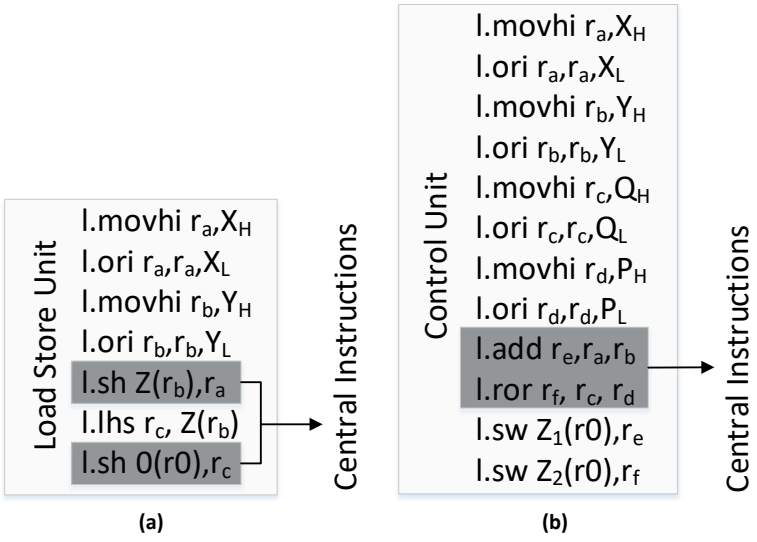


Figure 3.37: Load-Store Unit & Control Unit TM example

Chapter 4

Simulation Framework

4.1 Introduction

The simulation framework provides with an integrated 3D-IC design and test environment. A comprehensive set of script- and program- based flows for commercial and custom-made tools enable the efficient and reliable deployment and execution of experiments of the test methods presented in Chapter 3. The simulation framework consists of three main flows:

- The design flow for benchmark cores, that processes IWLS cores [148] and produces the required core- design and test data for our experiments.
- The 3D-SoC testing flow, which allows a) creation of artificial 3D-SoC and TAM configurations, b) execution of implementations of the proposed methods, and c) evaluation of the proposed test methods.
- The SBST flow, which allows a) simulation of test programs for processor based SoCs, b) reorder of the test programs by discovering the most effective test patterns, and c) evaluation of the proposed method.

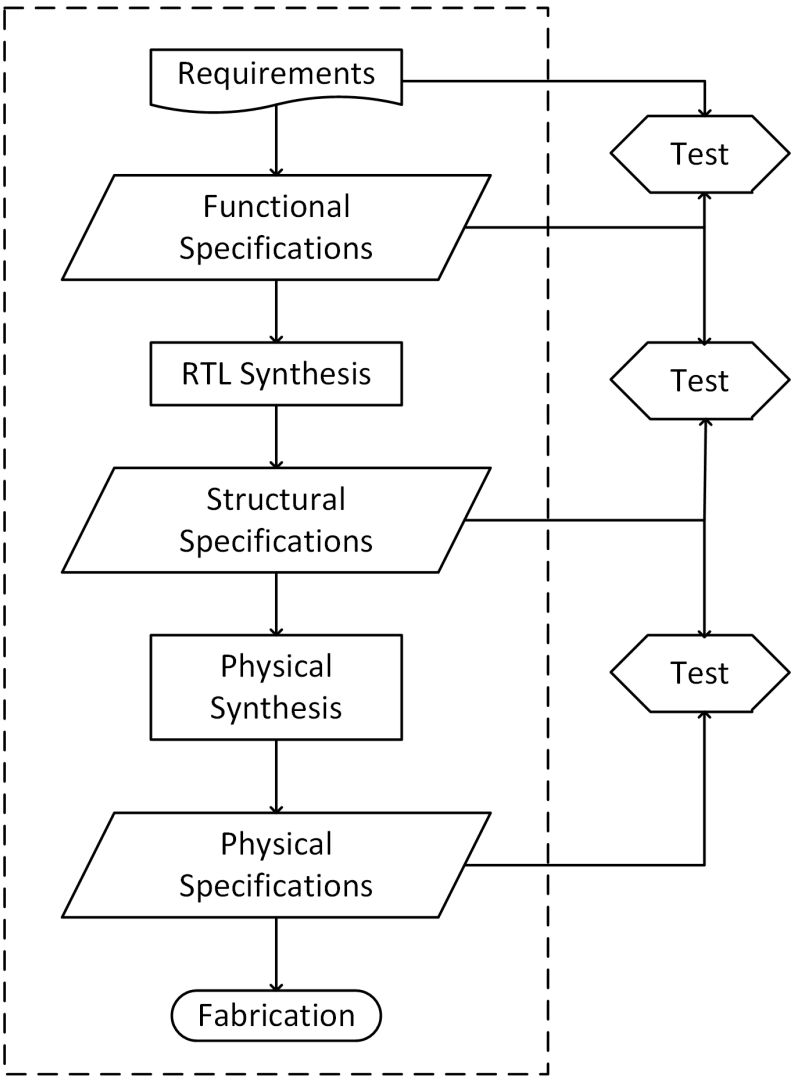


Figure 4.1: General design flow.

4.2 Design Flow for Benchmark Cores

A design flow, in the domain of the VLSI IC, is a set of procedures that allows designers to progress from chip specification to chip implementation. Figure 4.1 presents a general design flow. Design starts at the behavioral level and then proceeds to the structural level, by applying the RTL synthesis step. In RTL level the designs are captured in a Hardware Description Language (HDL). Then, the physical synthesis (or layout generation) step follows, where the HDL description is transformed to a physical description suitable for chip fabrication. The basic design flow, presented in Figure 4.1, applies to SoC design too, in the sense that the entire system needs to be specified, debugged, modified for testability, validated, and mapped to a technology, but in this case the whole flow needs to be done in an integrated framework.

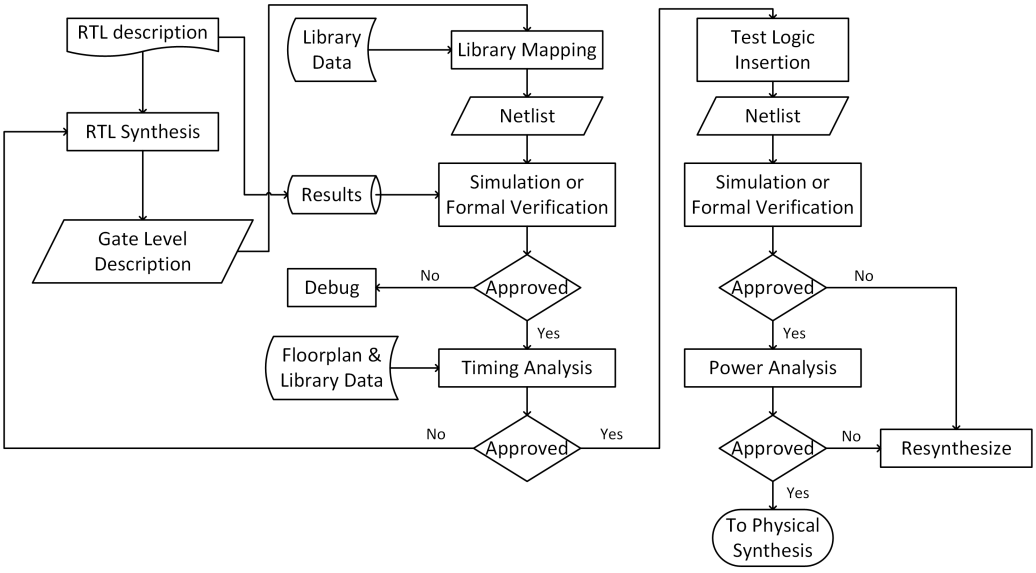


Figure 4.2: RTL flow.

In the case of a benchmark core, the behavioral level has already been implemented, tested and verified. At the point of RTL synthesis, specialized tools are used to directly transform the behavioral RTL description to a structural gate-level netlist. The tools that have been used in this research for RTL synthesis were provided from Synopsys and Cadence Design Systems.

The implemented RTL flow is shown in Figure 4.2. As it is presented, using an RTL synthesizer such as the Design Compiler of Synopsys or the RTL Compiler of Cadence, the verified RTL description of a IWLS core is transformed initially to a generic circuit of gates and registers, optimized in terms of speed and area. Then, the generic gates are mapped one-to-one to pre-designed cells of a standard cell library. In this research, the 45nm process nangate standard cell library [199] is used. The final result is a structural netlist of the benchmark core.

There are two main approaches to verify that the structural netlist performs the same function as the verified HDL product, the functional and the formal verification. In functional verification, a logic test bench is used to verify that exactly the same output is produced for the behavioral and structural descriptions. In formal verification, a formal verification program is created that compares the logical equivalence of the two descriptions. Formal verification mathematically proves that both descriptions have exactly the same Boolean functions [200], [201]. On the other hand, functional verification is based on the efficiency of the test vectors. Formality from Synopsys and Incisive Conformal from Cadence are examples of formal verifiers. Other types

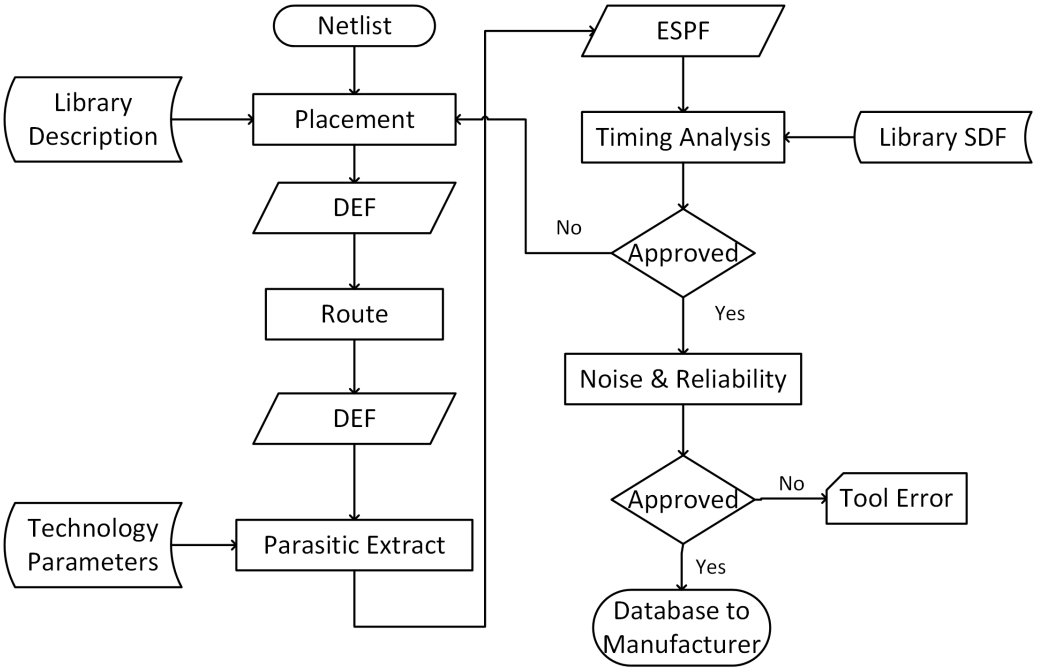


Figure 4.3: Automated layout generation.

of verification that can be run are semantic and structural checks on the HDL. For example, in a semantic check, it should be ensured that all bus assignments match in bit width, while in a structural check, it should be checked that all outputs are connected.

The next step in the flow is a static timing analysis that evaluates all timing paths in the core under scope. The inputs are derived from the basic timing of the library gates, due to intrinsic gate delays, and routing loads. The routing loads are estimated statistically or they are derived from floor-planning data. In this research, the Encounter from Cadence has been used for timing analysis. The final result is a report that includes timing information for the worst paths in the core.

At this point, to allow for efficient test, the DFT is implemented using specialized procedures either in the Synopsys Design Compiler or the Cadence RTL Compiler. Specifically, scannable registers are inserted and / or existing registers are modified so that the state of the design can be set and monitored. Then, ATPG is performed to generate tests for the scannable design. In this research, the required core test sets are generated using the Synopsys Tetramax and the Cadence ET ATPG tools.

The RTL flow concludes with the estimation of the design's normal and test power consumption. Commercial power analysis tools that have been used are the PrimePower from Synopsys.

Layout generation is the process of turning a design into a manufacturable database. Namely, it transforms a design from the structural to the physical domain. This process is sometimes called physical synthesis. Figure 4.3 presents a standard place and route layout generation design flow used in most ASICs and in this research too. The commercial tool that has been used in this research for the layout generation of the IWLS cores is the Encounter from Cadence.

The layout flow starts with the structural netlist describing gates, flip-flops, and their interconnections. The netlist is provided in the Design Exchange Format (DEF) as a Verilog netlist. Then, a semi-automated floor-planning step is performed, which produces a file that describes the floor-plan of the design. The next step is the placement process, where standard cells of constant-height and variable-width are arrayed in rows across a chip. A standard cell library definition describing cell dimensions and port locations, in the Library Exchange Format (LEF), summarizes the salient physical details of cells. A simple placement algorithm is used to minimize the length of wires. At the end of the placement phase, the used cells have been fixed in position in the overall array. The placed design is saved in a standard format (e.g., DEF) for routing.

After placement of standard cells, the signal nets in the design need to be routed. There are two phases of routing: a) global and b) detailed routing. A global router creates routes through channels according to a cost function. Wires can be changed from channel to channel if the density of wires in a channel becomes too high. The detailed router places the actual geometry required to complete signal connections. The results are written to another DEF file.

The placed and routed design is then passed to the circuit parasitic extractor. The placed and routed design is provided to the extractor in DEF format and the output is a file in the Standard Parasitic Exchange Format (SPEF) that describes the resistors and capacitors associated with all nets in the layout. The extractor uses another technology file defining the interlayer capacitances and layer resistances.

Static timing analysis is rerun with the actual routing loads placed on the gates. Multiple iterations of synthesis and placement/routing are usually necessary to converge on timing requirements. In modern high-speed designs the clock distribution strategy is the dominant strategy. To minimize skew, the clock and its buffers are pre-route before the main logic placement and routing is completed. This task is performed with a clock tree router. Normal and test power estimation is repeated for

the complete design. Similar tools and techniques to those in the RTL level are used.

4.3 Simulation Flow for 3D-SoCs Testing

The simulation flow for 3D-SoCs testing uses artificial SoCs, which are defined by a custom-made tool called *SoCTDMScheduler* [202]. This tool allows to a designer the following:

- Select the embedded cores in an artificial SoC from a set of available benchmark cores and/or user-defined cores.
- Define 3D-SoC layers and assign cores in each layer. Then, the SoCTDMScheduler using an external tool, the *hotfloorplanner* [203], derives the floorplan of the artificial SoC taking into account proximity relations among the embedded cores that belong to the same layer.

Then, the designer can proceed in the definition of the SoC test and TAM configuration files, which include:

- The cores in the 3D SoC.
- The layers of the 3D SoC.
- The assignment of the cores to the layers of the 3D SoC.
- The scan frequencies provided by the TDM scheme.
- The maximum TSV clock frequency.
- The maximum scan frequency that is allowed in each layer.
- The test time of each core per scan frequency used.
- The power consumption of each core per scan frequency used.
- The TDM-based TAM configuration, that is the number of the TAM buses, their bit-length and the set of the cores that are connected to each bus.

For the method presented in Section 3.3, SoCTDMScheduler is used to derive the test schedule, while we may impose or not power constraints per layer. For the method

presented in Section 3.4, we implemented our algorithm in C#. Our implementation uses as input the same configuration files as before, with the addition of the number of daisy chains per layer and their frequencies.

4.4 Simulation Flow for SBST

The simulation flow for the proposed SBST method consists of the following steps:

- Create test programs for a processor-based SoC based on the TMIs presented in Subsection 3.5.4 using custom-made Python scripts.
- Fault simulate the test programs, using Synopsys TetraMax.
- Select the most effective TMI instances, using a custom made tool, implemented in C++.
- Reorder the initial programs, using custom-made Python scripts and rerun fault simulation.
- Evaluate the results.

Chapter 5

Evaluation of the research work

5.1 Evaluation of 3D-SoCs Testing Methods

5.1.1 Cores & Artificial 3D-SoCs Characteristics

In order to evaluate our proposed methods on 3D-SoCs, we selected ten cores from the IWLS suite [148] and we synthesized them using the 45 nm Nangate technology [199]. Their function is presented in Table 5.1 and their structural information is presented in Table 5.2.

For testing each core, the full scan design methodology was adopted. The cores were wrapped using commercial DFT synthesis tools with IEEE 1500 Std. wrappers and 8, 16 and 32 wrapper and scan chains (denoted as WC in Tables 5.3, 5.4). The input (output) parallel ports of the wrappers were connected to 8-bit, 16-bit and 32-bit input (output) local buses. For each core, compacted test vectors targeting complete coverage of all detectable transition faults were generated using a commercial fault simulator and the Launch-on-Capture scheme. The number of test vectors for the various cores varies between 25 test vectors for core pci_bridge up to 2443 test vectors for core vga_lcd (the same test data are generated for identical cores, but they are not delivered simultaneously to the cores). These test vectors were applied

Table 5.1: Benchmark cores function.

Core	Function
tv80	TV80 8-Bit Microprocessor Core
mem_ctrl	WISHBONE Memory Controller
ac97	WISHBONE AC 97 Controller
usb_funct	USB function core
pci_bridge	PCI
aes_core	AES Cipher
wb_conmax	WISHBONE Conmax IP Core
ethernet	Ethernet IP core
des_perf	DES optimized for performance
vga_lcd	WISHBONE rev.B2 compliant Enhanced VGA/LCD Controller

Table 5.2: Benchmark cores information.

Core	Sequential	Inverter	Buffer	Logic	Tristate	Unresolved	Total
tv80	359	1101	97	5604	-	-	7161
mem_ctrl	1083	1462	221	8674	-	-	11440
ac97	2199	1525	111	8020	-	-	11855
usb_funct	1746	1865	33	9164	-	-	12808
pci_bridge32	3359	3095	100	10262	-	-	16816
aes_core	530	5589	274	14402	-	-	20795
wb_conmax	770	3366	86	24812	-	-	29034
ethernet	10544	3404	234	32557	32	-	46771
des_perf	8808	28372	1489	59672	-	-	98341
vga_lcd	17079	21397	2542	83013	-	-	124031

with increasing shift frequencies, until the timing simulation failed. This frequency is the maximum shift-frequency supported by each core, and it is reported in the first row for each core in Tables 5.3, 5.4. The next three rows show the test-time T and the total average power P at lower shift frequencies of 200, 100 and 50 MHz. The average leakage power is about 30% of the total average power at the minimum shift-frequency. The full scan-tests were thermally characterized by providing to the HotSpot thermal simulator the power trace of the test-schedule for each core for each time step, and the floorplan of the 3D IC [204].

We created three artificial 3D-SoCs in order to evaluate our methods, *SoC-A* with 90 cores, *SoC-B* with 285 cores and *SoC-C* with 283 cores. For SoC-A the cores were randomly placed in multiple copies in the three dies as follows: 36 cores at the bottom die, 25 cores at the middle die and 29 cores at the top die. For SoC-B the cores were placed into three dies using an unbalanced manner in terms of test-time and power consumption. Specifically, 30 copies of the 3 largest and most test-time consuming IWLS cores were placed in die-1, 170 copies of the 7 smallest and less time-consuming ones were placed in die-2, and 85 copies of all IWLS cores were placed in die-3. For SoC-C the cores were placed into three dies using an unbalanced manner in terms of test-time and power consumption. Specifically, 30 copies of the three largest and most test-time consuming IWLS cores (ethernet, des_perf, vga_lcd) were placed in die-1, 170 copies of the rest seven smallest and less-time consuming cores were placed in die-3, and 83 copies of all ten cores were placed in die-2.

5.1.2 2-D Time-Division Multiplexing

To highlight the benefits of the proposed method, we run experiments on SoC-A and SoC-B presented in Subsection 5.1.1, using 8, 16 and 32 wrapper and scan chains. For these SoCs we set a) $(F_{GTC}, F_{LTC}) = (600, 200)$ MHz, and b) $(F_{GTC}, F_{LTC}) = (1200, 400)$ MHz (the functional clock is used for the capture cycles). The average power dissipated by an 1-bit wide global channel toggling at 600 MHz and 1.2 GHz is equal to 0.31 mW and 0.60 mW, respectively.

The 2D TDM test-scheduling method was implemented in C and experiments were conducted on a TAM configuration with one input and one output global channel of 8, 16 and 32 TSVs. At each layer, one local input and one output bus were used with 8, 16 and 32 lines. The vertical TDM frequency was set equal to a) 600 MHz,

Table 5.3: Test Data (A) for the cores of the artificial 3D-SoCs (SF: Shift Frequency, WC: Wrapper Chains).

	SF	8 WC		16 WC		32 WC	
	(MHz)	P (mW)	T (μs)	P (mW)	T (μs)	P (mW)	T(μs)
tv80	222	3.8	79.1	3.6	43.8	3.6	26.1
	200	3.5	87.8	3.3	48.7	3.3	29.0
	100	1.8	175.7	1.7	97.4	1.7	58.0
	50	1.0	351.3	1.0	194.8	1.0	116.0
mem_ctrl	333	10.9	260.5	10.9	135.9	11.0	70.9
	200	6.7	434.2	6.7	226.4	6.8	118.2
	100	3.5	868.5	3.6	452.9	3.6	236.4
	50	1.9	1736.9	1.9	905.8	2.0	472.7
ac97	333	17.9	109.1	17.7	59.7	18.1	29.4
	200	11.0	181.9	10.9	99.6	11.2	49.1
	100	5.8	363.8	5.8	199.1	5.9	98.1
	50	3.2	727.6	3.2	398.3	3.2	196.3
usb_funct	400	23.0	112.9	22.9	54.5	23.1	27.8
	200	12.1	225.8	12.1	109.0	12.2	55.6
	100	6.4	451.7	6.3	218.0	6.4	111.3
	50	3.5	903.4	3.5	435.9	3.5	222.5
pci_bridge	400	22.7	21.5	22.8	11.0	22.8	5.2
	200	11.7	43.0	11.8	21.9	11.8	10.4
	100	6.2	86.0	6.2	43.9	6.2	20.9
	50	3.5	171.9	3.5	87.7	3.5	41.8

Table 5.4: Test Data (B) for the cores of the artificial 3D-SoCs (SF: Shift Frequency, WC: Wrapper Chains).

	SF	8 WC		16 WC		32 WC	
	(MHz)	P (mW)	T (μs)	P (mW)	T (μs)	P (mW)	T (μs)
aes_core	333	16.4	106.3	16.3	55.4	16.5	29.7
	200	11.1	177.1	10.9	92.4	11.1	49.6
	100	6.3	354.2	6.2	184.8	6.3	99.2
	50	3.6	708.4	3.5	369.6	3.6	198.3
wb_conmax	400	50	241	50	120.4	49.9	60.8
	200	26.6	482	26.7	240.7	26.6	121.6
	100	14.2	963.3	14.2	481.5	14.2	243.3
	50	7.8	1927.8	7.8	962.9	7.8	486.6
ethernet	333	76.6	5989.4	77.5	2628.8	79.1	1231.7
	200	47.3	9982.3	47.8	4377	48.8	2050.9
	100	25.0	19964.6	25.3	8754	25.8	4101.7
	50	13.8	39929.2	13.9	17507.9	14.1	8203.4
des_perf	400	143.2	254.2	143.9	120.6	144.1	59.5
	200	80.5	508.3	81	241.2	81.1	119
	100	43.9	1016.6	44.1	482.4	44.2	238
	50	23.8	2033.2	24	964.8	24	476.1
vga_lcd	333	119	17948.9	120.9	8317.6	122	4056.9
	200	73.4	29884.9	74.5	13848.9	75.2	6754.7
	100	38.6	59769.7	39.2	27697.7	39.5	13509.4
	50	21.1	119539.5	21.4	55395.5	21.6	27018.7

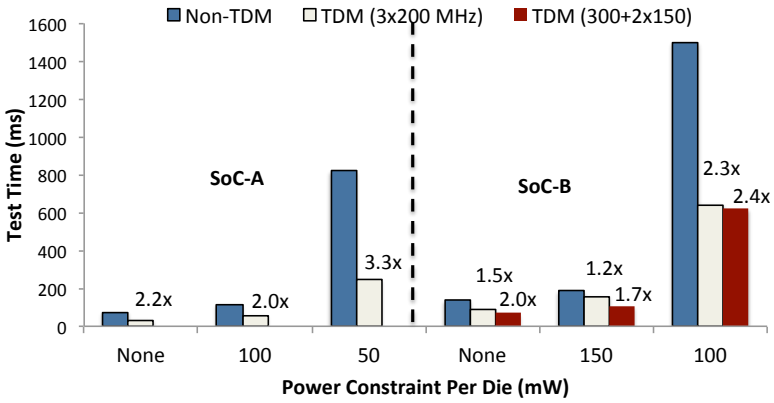


Figure 5.1: The effect of TDM on test-time.

b) 1.2 GHz, and the horizontal TDM frequency was set equal to a) 200 MHz and b) 400 MHz respectively. The horizontal TDM frequency is divided into 200, 100, 50 and 25 MHz. The run times for SoC-A were equal to 187 sec (no power constraints), 348 sec (50 mW power constraint per die) and 2.5 hours (with power and thermal constraints). The run times for SoC-B were equal to 41 minutes, 2.44 hours and 8.2 hours respectively. Every intermediate schedule generated requires in average 15 minutes for SoC-A and 38 minutes for SoC-B.

We also developed a non-TDM baseline approach using the RP heuristic. This approach was applied on multiple TAM configurations, and the result of the best configuration was reported every time. In the case of 32 TAM-lines, the non-TDM approach was applied to three different bus configurations: a) one 32-bit input and one 32-bit output bus, b) two 16-bit input and two 16-bit output buses, and c) four 8-bit input and four 8-bit output buses. In the non-TDM approach, the TAM frequency is always set equal to the highest scan shift-frequency that does not violate the power constraint for any of the dies of the 3D SoC (the SA optimization is not applicable in this case). In both the non-TDM and the TDM cases the test clock and other control signals for the 1500 standard are transferred to the cores. However, since the overhead of these signals is similar in both the TDM and the non-TDM methods, we do not consider this overhead for comparison purposes.

In Figure 5.1 we compare the TDM and the baseline approaches for various power constraints. The x-axis shows the power constraint per die, and the y-axis shows the test-time. At SoC-B the 600 MHz frequency supported by the global channels was divided among the dies a) uniformly with 200 MHz at each die, and b) non-uniformly with 300 MHz at the most loaded die and 150 MHz at each of the other dies. The TDM

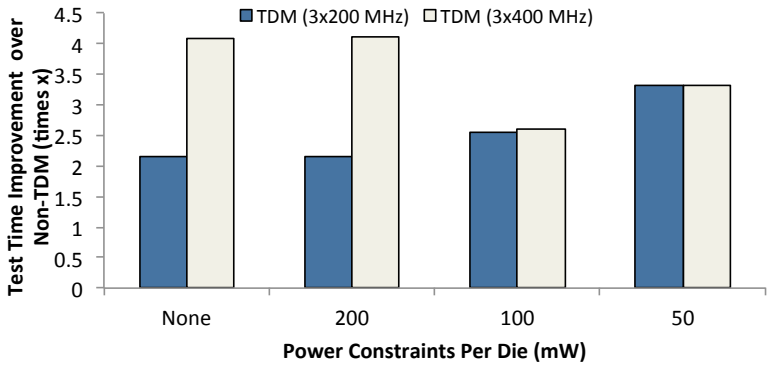


Figure 5.2: The effect of the vertical frequency on the test-schedule of SoC-A.

scheme clearly outperforms the baseline approach, and the improvement increases as the power constraint per die becomes more strict (above each column group we report the improvement for TDM over the best configuration for the non-TDM approach). This happens because the power constraint prohibits the use of high shift frequencies in many of the cores, and restricts the TAM frequency in the non-TDM case. TDM overcomes this limitation by (a) using different shift frequencies for the various cores, and (b) scheduling different tests in parallel with appropriate shift frequencies that do not violate the power constraint of the die. Therefore, as the power constraint becomes more tight, TDM is able to better exploit the higher frequency provided by the global channels. As shown in Figure 5.1 the improvement in SoC-B when the 600 MHz are uniformly divided among the dies is not as high as the improvement in SoC-A, because of the unbalanced placement of cores in SoC-B. However, when the most time-consuming layer of SoC-B is assigned 300 MHz of frequency, and the rest two layers are assigned 150 MHz each, then the improvement of the TDM over the Non-TDM method improves even more.

In Figure 5.2 we highlight the impact of the vertical TAM frequency on the test-time improvement achieved by TDM over the non-TDM approach for SoC-A. The x-axis shows again the power constraint set for each die, while the y-axis presents the ratio of the test-time provided by the non-TDM method to the test-time provided by the TDM method. When the power is not constrained, TDM fully exploits the high speed of the TSVs and offers lower test-time for increasing frequency. As the power constraints become more tight, TDM cannot derive the full benefit of the available frequency and the additional test-time benefits offered by increasing the vertical frequency become very small. Nevertheless, TDM offers over 2x test-time reduction

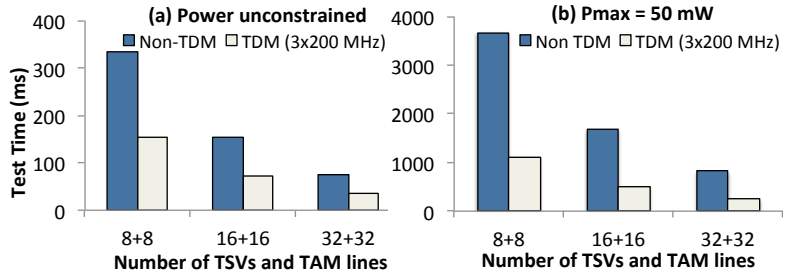


Figure 5.3: The effect of TDM on the number of TSVs and TAM-lines (SoC-A).

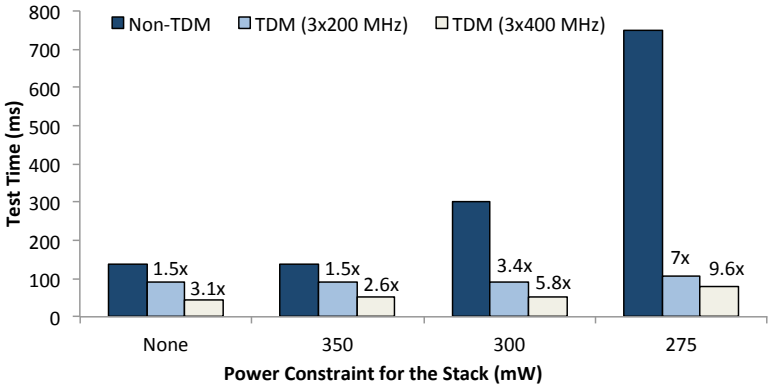


Figure 5.4: Test-time under power constraints on the whole stack of SoC-B.

even when only a part of the TSV frequency is exploited, which is very important in 3D SoCs with a large number of dies (thus long global channels).

In Figure 5.3 we vary the number of TSVs and the TAM-lines in the range 8+8, 16+16 and 32+32, and we apply both the non-TDM baseline and TDM at 600 MHz vertical frequency on SoC-A as follows: a) without power constraints, and b) for 50 mW maximum average power consumption for each die. The test-time for TDM is equal to the test-time for the non-TDM approach that uses double the number of TSVs and TAM-lines in case (a), and almost quadruple the number of TSVs and TAM-lines in case (b). The total TAM routing is reduced with TDM because half of the TAM lines are required in case (a) and a quarter of the TAM lines are required in case (b) for delivering the same test time. TAM routing is very expensive in 3D ICs because: a) it involves long horizontal and vertical connections, b) it requires TSVs that occupy silicon, and c) separate device pins are required to load the test data. Therefore, TDM exploits the frequency offered by TSVs to considerably reduce the total TAM cost and TSV count without any adverse impact on test-time.

In Figure 5.4 we present the test-time benefits of the TDM approach over the non-TDM baseline for SoC-B for 32 TSVs, when the power constraint is set for the whole

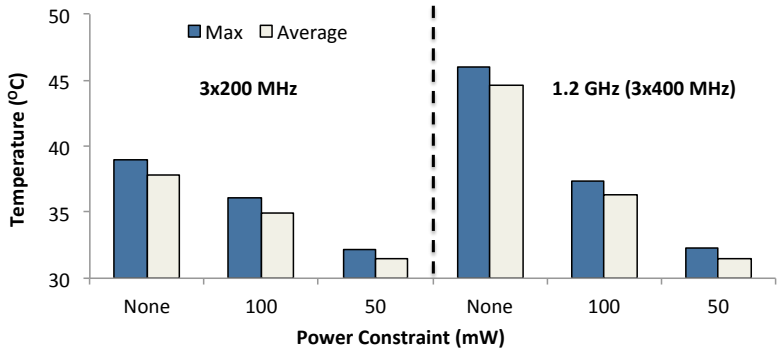


Figure 5.5: The temperature of SoC-A for various power constraints.

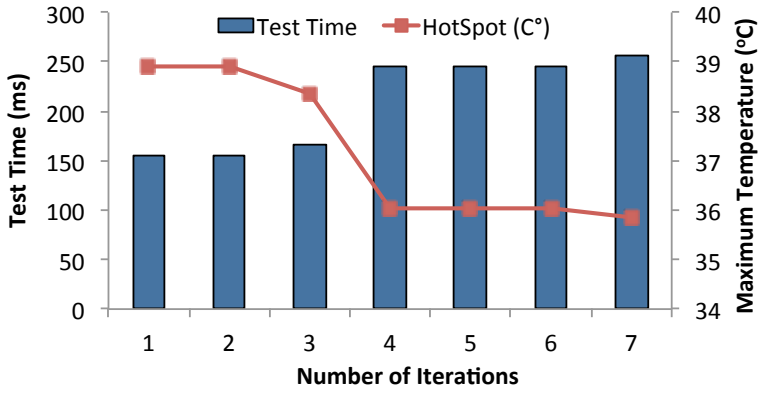


Figure 5.6: The effect of thermal constraint on test-time.

stack. The x-axis shows the total power constraint of the stack, and the y-axis shows the test-time. The TDM approach clearly outperforms the baseline approach, and the test-time improvement approaches 7.0x at 600 MHz, and 9.6x at 1.2 GHz. The improvement is higher in this case because the test-scheduling method has higher flexibility to schedule tests in parallel without violating the power constraints (some of the dies dissipate more power than the rest of the dies).

Figure 5.5 presents the effect of power constraints on the maximum and average temperature at each die of SoC-A. The highest temperature is observed when the scheduling of tests is not constrained by power, and the frequency of the global channels is relatively high at 1.2 GHz (note that a high frequency at the global channels permits high frequencies at the local channels). The temperature drops considerably as the power constraint becomes more strict. The same result is achieved by using a lower TDM frequency at the global channels, but this compromises the high frequency of TSVs.

Figure 5.6 shows the effect of the thermal-aware scheduling algorithm on the

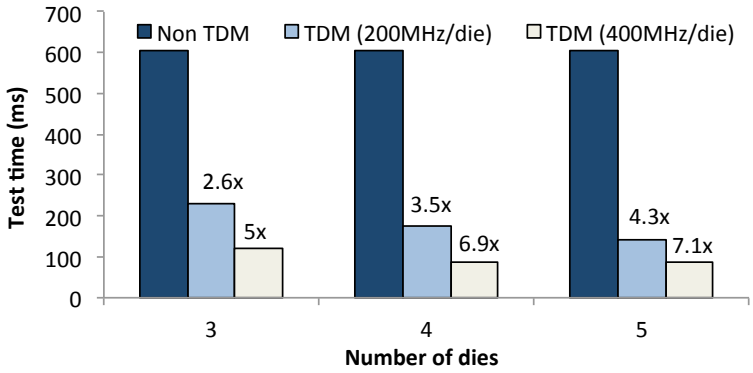


Figure 5.7: The efficiency of TDM for many-layer stacks (SoC-B).

maximum temperature of the dies of SoC-A for one 8-bit input and one 8-bit output global channel operating at 600 MHz. The maximum temperature of the initial thermally-unconstrained schedule was $38.91°C$. The thermal-aware scheduling algorithm reduced the maximum temperature below $35.84°C$ after 3 iterations, at the expense of an increase in test-time. Further test-time gains can be achieved by targeting only selected thermal-sensitive areas at dies that are far from the heat sink.

At the last experiment we partitioned the cores of SoC-B into 4 and 5 layers, and we used 200 MHz and 400 MHz horizontal TDM frequency per layer. The TAM consists of an 8-bit input and an 8-bit output global bus operated between 600 MHz and 2.0 GHz. The test-time results, assuming no power constraints, are shown in Figure 5.7. The test-time of the Non-TDM architecture remains the same, since the test-time overhead does not change with the number of layers (note that the number of cores remains the same in all cases). In contrast, each additional layer provides the opportunity for more parallelism in TDM-based test-scheduling (for each TDM case, Figure 5.7 reports also the improvement against the Non-TDM case). This happens because the 2.27 GHz vertical frequency suffices to accommodate up to 5 layers with 400 MHz shift-frequency per layer, while in terms of test-scheduling, each additional layer offers an additional level of parallelism. As a result the test-time drops, even for stacks with many layers, which makes the two-dimensional TDM approach very effective for testing 3D ICs.

Table 5.5: Test time and routing overhead of the proposed TAM architecture without power constraints.

	Optimize Test-Time			Optimize Routing	
Ratio	TT(w/o, w/)	WL(w/o)	WL(w/)	TT(w/)	WL(w/)
$R_{1,1,1}$	767.2 ms	875.3 mm	754.7 mm	819.5 ms	735.1 mm
$R_{2,1,1}$	605.5 ms	851.3 mm	725.3 mm	786.7 ms	689 mm
$R_{2,2,1}$	639.4 ms	834.2 mm	727.1 mm	703.5 ms	702.4 mm
$R_{3,2,1}$	**511.5 ms**	873.3 mm	740.7 mm	579.5 ms	723 mm
$R_{4,2,1}$	529.9 ms	850.4 mm	729.2 mm	**688.5 ms**	692.9 mm
$R_{3,3,1}$	596.6 ms	872.8 mm	743.4 mm	719.6 ms	714.2 mm
$R_{5,2,1}$	605.5 ms	860.3 mm	721.5 mm	786.7 ms	685.2 mm
$R_{3,3,2}$	682 ms	881.2 mm	738.8 mm	719.3 ms	**672.8 mm**

5.1.3 K^3 TAM Optimization Results & Comparisons

We implemented the TAM optimization approach in C#, and we run experiments on SoC-C using 8 wrapper and scan chains, and 8-bit wide daisy-chains at each layer.

In order to avoid the uniformity imposed by using a common test-set to test multiple copies of each core, the test-set of each copy was extended using additional N-detect test-vectors. Specifically, at the first die the test-set of each core was extended by 20%, 40%, 60% ... additional N-detect test-vectors to generate the test-set of each copy, at the second die it was extended by 10%, 20%, 40% ... and at the third die by 5%, 10%, 15% ... additional test-vectors. The second and the third row of Tables 5.3 and 5.4 report the test time T and the total average power P at lower shift-frequencies equal to 200 MHz, 100 MHz for the initial (not-extended) test-set of every core (the average leakage power is about 30% of the total average power at the minimum shift-frequency).

The $^{12}/_{20}$, $^{7}/_{20}$ and $^{1}/_{20}$ of the total test-data volume are used for testing the first, second and third die of the stack respectively. Therefore, beside the regular division $^{1}/_{3}, ^{1}/_{3}, ^{1}/_{3}$ of F_{GTC} we also applied non-regular division with ratios $^{a_1}/_{b}$, $^{a_2}/_{b}$, $^{a_3}/_{b}$ corresponding to die 1, 2 and 3, for $a_1 \geq a_2 \geq a_3$, $b = a_1 + a_2 + a_3$ and $b = 4, 5, 6, 7, 8$ (a_1, a_2, a_3 are integer values in the range $[1, 6]$). Each ratio is reported as R_{a_1,a_2,a_3} (e.g. $R_{1,1,1}$ is the regular division of F_{GTC}). The value of F_{GTC} was set equal to 600 MHz, 1.2 GHz and 1.8 GHz.

Table 5.6: Test time and routing overhead of the proposed TAM architecture with power constraints.

	Optimize Test-Time				Optimize Routing	
Ratio	TT(w/o)	WL(w/o)	TT(w/)	WL(w/)	TT(w/)	WL(w/)
$R_{1,1,1}$	767.2 ms	875.3 mm	767.2 ms	754.7 mm	819.5 ms	735.1 mm
$R_{2,1,1}$	770.0 ms	851.3 mm	762.2 ms	713.5 mm	819.5 ms	689 mm
$R_{2,2,1}$	639.4 ms	834.2 mm	639.4 ms	727.1 mm	**703.5 ms**	702.4 mm
$R_{3,2,1}$	770 ms	873.3 mm	762.2 ms	740.5 mm	740.4 ms	723 mm
$R_{4,2,1}$	788 ms	850.4 mm	710.5 ms	692.9 mm	710.5 ms	692.9 mm
$R_{3,3,1}$	**596.6 ms**	872.8 mm	**596.6 ms**	743.5 mm	719.6 ms	714.2 mm
$R_{5,2,1}$	720.3 ms	860.3 mm	718.9 ms	**684.4 mm**	786.7 ms	685.2 mm
$R_{3,3,2}$	682 ms	881.2 mm	682 ms	738.8 mm	719.3 ms	**672.8 mm**

In Tables 5.5, 5.6 we present the total test time (TT) and the wirelength (WL) of the proposed architecture with and without power-constraints respectively for two different goals: (a) *Optimize Test-Time* and (b) *Optimize Routing*. The proposed architecture was generated for every configuration R_{a_1,a_2,a_3} reported above using $F_{GTC} = 600$ MHz (the best configurations are reported in Tables 5.5, 5.6). In the *Optimize Test-Time* case we applied the K^3 optimization approach in two steps. At the first step two daisy-chains were generated at each die without (w/o) exchanging cores between them (the test time TT is minimum in this case and it is reported in the first column). At the second step we exchanged cores between the two daisy chains at each die (w/) in order to further reduce the routing overhead, with TT limited below the minimum value computed at the previous step. This limitation was removed in the *Optimize Routing* case, where we allowed the value of TT to increase up to 50% in order to enable additional core permutations between the daisy-chains at each die to further reduce the routing overhead.

As it was expected, in the power unconstrained case (Table 5.5) the minimum test-time (marked in bold) was provided by $R_{3,2,1}$ (note that the ratios $3/6, 2/6, 1/6$ are closer to the ideal values $12/20, 7/20, 1/20$ than the rest ones). Moreover, in the *Optimize Test-Time* cases the wirelength of the 'w/' TAM is considerably smaller than the 'w/o' TAM even though they both offer the same test time. In the *Optimize Routing* case the total wirelength reduces even further, while in most of the cases the additional

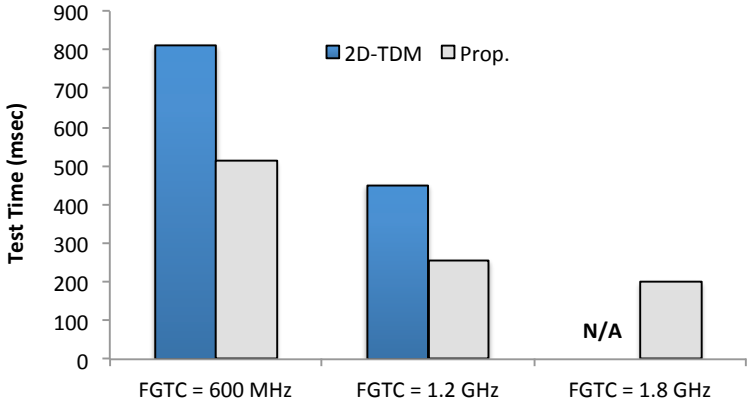


Figure 5.8: Comparisons against 2D-TDM [18].

test-time overhead is in the range of 5%-23%.

In the power-constrained case the average power-dissipation for each layer is limited below 175 mW (Table 5.6) and thus idle periods are inserted in the test-schedule. As a result the minimum test-time increases by 17% as compared to the power unconstrained case (the best configuration is $R_{3,3,1}$ in this case). Moreover, in the *Optimize Routing* case (last two columns in Table 5.6) the wirelength of the daisy chains is not affected by the power constraints, while the test times increase only marginally as compared to the *Optimize Test Time* case. Therefore, it is obvious that despite the very strict power-constraints, the proposed approach exploits the irregular division of the frequency among the dies in such a way as to optimize both the test-time and the wirelength of the TAM.

In Figure 5.8 we compare the proposed method against the 2-D TDM approach [18] assuming no power constraints. The 2-D TDM approach was applied on the 3D stack and it was compared in terms of test time against the $R_{3,2,1}$ configuration for $F_{GTC} = 600$ MHz, 1.2 GHz, 1.8 GHz (note that $F_{GTC} = 1.8$ GHz was not applicable in [18] due to timing violations). Besides the very large routing overhead of [18], which is more than four times (4x) larger than that of the proposed method, a large number of buffers is also required by 2D-TDM. At the same time the proposed TAM optimization exploits the high bandwidth of the global channels, and it offers 2x-4x reduction of the test-time compared to the 2D-TDM approach.

In Figure 5.9 we compare the proposed method against the conventional daisy-chain-based method used for testing 3D-ICs denoted as Non-TDM. We generated two TAM configurations for the Non-TDM method: Non-TDM(A), Non-TDM(B). In Non-TDM(A) we used a single 8-bit wide daisy chain that spans the entire stack and

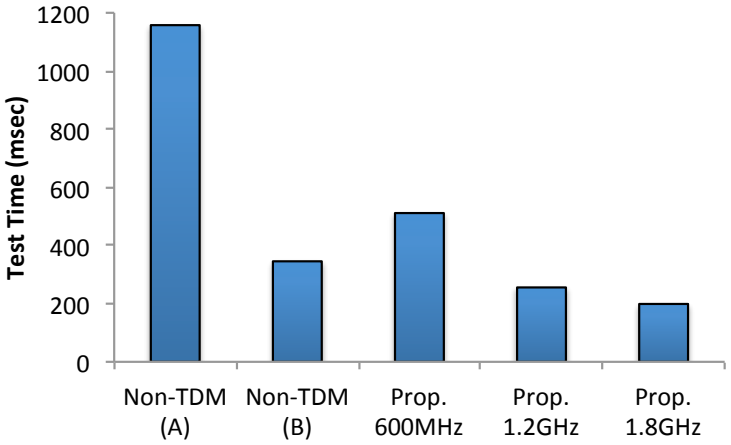


Figure 5.9: Test-time comparisons against Non-TDM approach.

connects all the cores at all dies. This configuration requires roughly the same routing overhead with the proposed TDM-based approach, as well as the same number of test TSVs ($2 \times (8+8)$ at the back side of the first and second layers) and the same number of test-pins at the bottom die. In Non-TDM(B) we used two independent 8-bit wide daisy-chains at each layer, which are connected using separate TSVs with dedicated test pins at the bottom die. In this case $2 \times (8+8)$ TSVs are required at the back side of the 3rd layer and another $2 \times (8+8) + 2 \times (8+8)$ at the back side of the 2nd layer (96 test-TSVs and 96 test-pins in total). This configuration offers comparable test-time and wirelength with the proposed scheme because all daisy chains can potentially shift test-data in parallel with the highest shift frequency supported by all cores (this frequency is equal to 222 MHz as shown in Tables 5.3, 5.4). The proposed method offers the lowest test time, which is almost 6x lower than Non-TDM(A). Non-TDM(B) offers comparable test-time with the proposed method running at 1.2 GHz, but it requires 3x more TSVs and 3x test-pins. Moreover, any increase of the number of dies to 4 and 5 increases both the TSV and test-pin overhead of Non-TDM(B) to 4x and 5x respectively, rendering this approach very expensive and not scalable for 3D-ICs.

5.2 Evaluation of SBST Approach

The proposed method, presented in Section 3.5, was developed using C++ and Python. Experiments were performed using the 32-bit scalar OpenRISC OR1200 RISC

processor, which has a Harvard micro-architecture, 5-stage integer pipeline, virtual-memory support (MMU) and basic DSP capabilities. The processor has one embedded instruction cache and one embedded data cache of 8KB each. The gate level netlist of the processor was synthesized using the NanGate 45nm technology and commercial tools. The gate-level netlist (excluding the memories) consists of 17.6K cells. One instruction is fetched from the cache at every clock cycle, while additional clock cycles are required when the data are fetched from the memory and/or conditional branch instructions are executed. The clock frequency for the OpenRISC OR1200 processor was set equal to 200 MHz.

Almost all units of the processor were targeted, i.e. the *ALU*, the *Multiply-Accumulate (MAC)*, the *Load-Store*, the *Program Counter Generator*, the *Instruction Fetcher*, the *Operands Mux*, the *Write-Back Mux* and the *Instruction-Decoder* (the *Exception-Handling* unit and the caches require special test generation mechanisms). 137 test-macros (including $A-TMIs$, $I-TMIs$, Load-Store and Control unit $TMIs$) were generated, each consisting of 3 to 17 instructions. For each test-macro 120 random instances were generated (overall 16,440 $TMIs$) and 6 test-programs were semi-automatically generated as follows:

- **Baseline**: 2, 4 and 8 instances were selected randomly out of the 120 instances generated for every test-macro (overall 274, 548 and 1,096 $TMIs$).
- **Proposed**: 274, 548 and 1,096 $TMIs$ were selected using the proposed method.

We note that the values of 2, 4 and 8 instances were intentionally chosen in order to generate small, medium and large test-programs, respectively. The running time of the proposed method on a single 64-bit CPU running at 1.2 GHz was less than one day in the worst case.

All test-programs were evaluated for detecting non-modeled faults using two surrogate fault models: the stuck-at and the transition-delay fault models. None of these models were explicitly targeted by the test-macro generation process. Instead, they were used to evaluate the potential of the proposed method to detect non-modeled faults. Even though the proposed method does not involve any fault simulations, such simulations were used to evaluate the generated test-programs and thus assess the effectiveness of the proposed method. These simulations were applied using commercial tools on a server with 48 CPUs running at 2.5 GHz. The stuck-at fault-simulation time for each test-program was between 5 hours (for 274 $TMIs$ on the

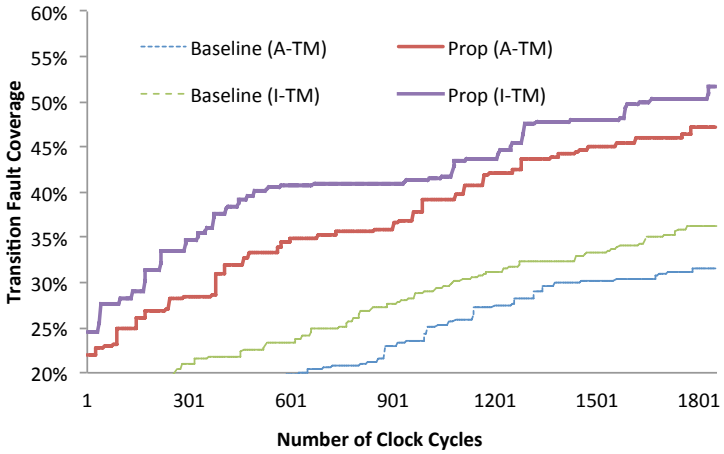


Figure 5.10: $I-TMs$ vs $A-TMs$ for unit (ALU).

Write-Back Mux unit) up to 5 days (for 1,096 $TMIs$ on the *Multiply-Accumulate unit*), and the transition-delay fault-simulation time was between 6 hours and 6.5 days on the respective cases. The stuck-at fault simulation and the transition-fault simulation on the whole processor (for 1,096 $TMIs$) require 11.5 days and 2 weeks, respectively.

As it was explained in Subsection 3.5.4, both $A-TMs$ and $I-TMs$ can be used for most of the processor-units, but the superiority of $I-TMs$ in detecting transition delay faults makes them more favorable as compared to $A-TMs$. As it is shown in Figure 5.10, $I-TMs$ are more effective than $A-TMs$ on the ALU unit of the OpenRISC OR1200 processor in both the proposed and the baseline approaches. Therefore, all the test-programs were composed of $I-TMs$ for every unit, except of the $Multiply-Accumulate$ unit, where only $A-TMs$ are applicable.

In order to show that the efficiency of the proposed method only slightly depends on the randomness of the initial set of $TMIs$, we generated (randomly) three different initial sets of 16,440 $TMIs$, and we run the proposed method three times selecting each time the 274, 548 and 1,096 most effective $TMIs$ from each one of these sets. We repeated the same approach for the baseline approach, and we generated 3 different baseline test-programs for every different number of 274, 548 and 1,096 $TMIs$. Figure 5.11 and Figure 5.12 present the stuck-at and the transition fault-coverage for the ALU unit for the three test-programs generated using the proposed approach, as well as the three test-programs generated using the baseline approach. Figure 5.13 and Figure 5.14 present the stuck-at and the transition fault-coverage for the *MAC* unit for the respective cases. In every chart the x-axis presents the test-time (in msec) and the y-axis presents the fault-coverage. It is obvious that there is no significant

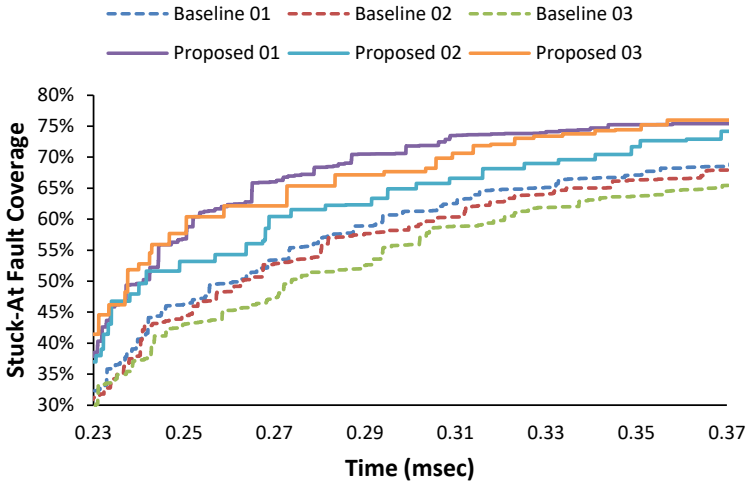


Figure 5.11: Stuck-at fault variation results of *ALU*.

variation on the fault coverage of the proposed test-programs, while in all cases the proposed method clearly outperforms the baseline approach. The proposed method offers very high coverage ramp-up, which can further reduce the test-time in strictly constrained manufacturing and periodic-test applications.

We note that the proposed method cannot achieve complete stuck-at fault coverage on the OpenRISC OR1200 processor, due to the existence of functionally untestable faults, but it achieves a stuck-at fault coverage that is close to the one achieved by the method in [179], which is based on manual test generation and represents one of the highest reported values in the literature for the OpenRISC OR1200 processor. Moreover, the use of a processor for specific applications may restrict the usage of certain parts of the processor [205], thereby further reducing the maximum attainable fault-coverage for these devices. For example, the stuck-at fault coverage for the MAC unit is 99.0% in [179] and 95.67% in the proposed method. The results for the ALU unit are 91.9% in [179] and 88.57% in the proposed method, and the results for the Instruction Fetch unit are 23.5% and 74.19%, respectively. The number of instructions composing the test according to [179] is equal to 31.728 while in the proposed method it was equal to 8,487. We have to note that these results were obtained using different synthesized netlists of the processors and different fault lists.

Table 5.7 presents the test-program size, the test-time, the stuck-at fault-coverage and the transition fault-coverage of the baseline (BSL) and the proposed approaches for each one of the three proposed and baseline test-programs. We note that the pro-

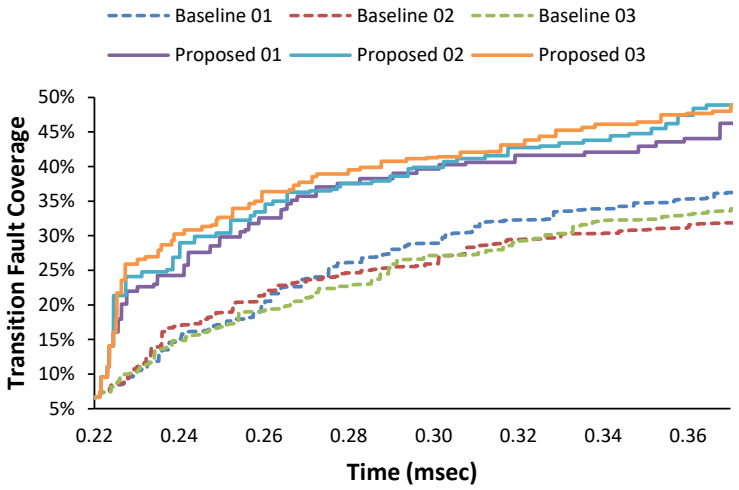


Figure 5.12: Transition-fault variation results of *ALU*.

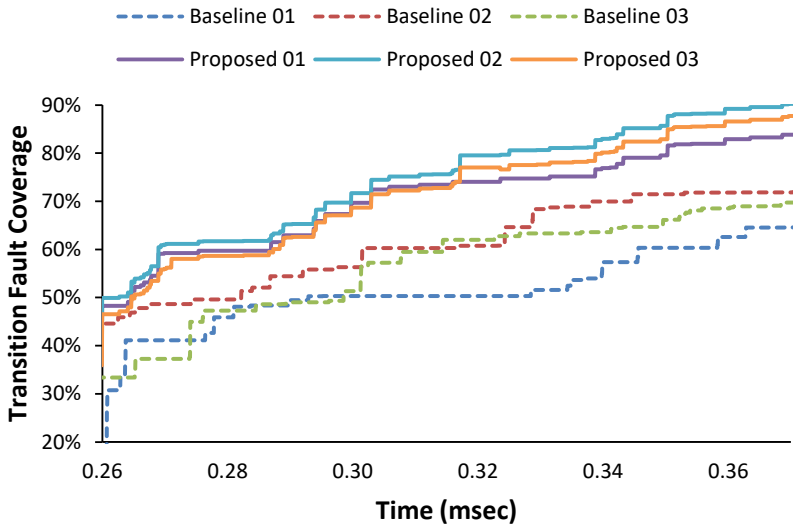


Figure 5.13: Stuck-at fault variation results of *MAC* unit.

posed method offers the highest benefits when a small number of $TMIs$ are selected because the proposed output-deviation based metric is very effective in identifying the $TMIs$ with the highest non-modeled fault coverage (when the number of the selected $TMIs$ increases some less effective $TMIs$ are inevitably selected and the large gap between the proposed and the baseline approach reduces). Therefore, its effectiveness depends mostly on the potential of the output-deviation based metric to identify the most effective $TMIs$, and less on the amount of randomization (the variation of the proposed method among the three test programs is less than 0.5% in almost all cases). Moreover, the proposed method tends to select $TMIs$ with large numbers of WIs, therefore it generates test-programs slightly larger than the base-

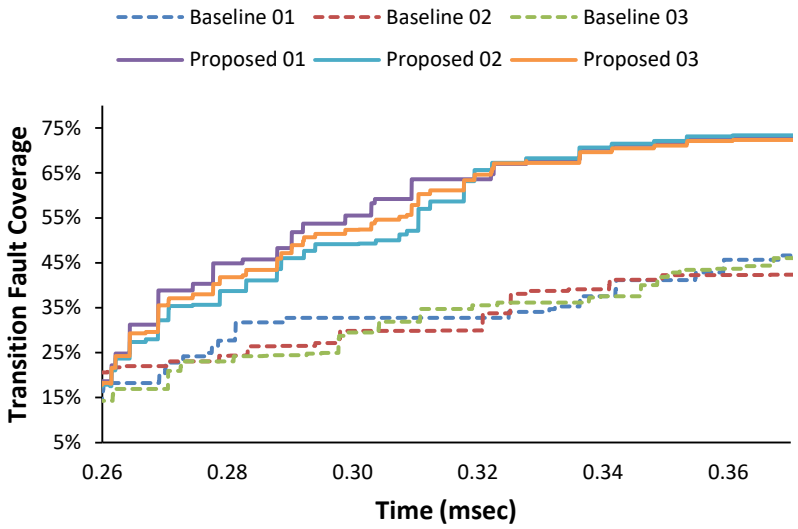


Figure 5.14: Transition-fault variation results of *MAC* unit.

line method, even though they both select the same number of $TMIs$. Nevertheless, the defect coverage of the proposed method remains higher even in cases that the baseline test-programs contain more $TMIs$.

Table 5.8 compares the proposed method against BSL in terms of the average (among the three test-programs) fault coverage achieved for every targeted unit of the processor. The results on the various units are reported in descending order of the size of the units (the large units are reported first). It is obvious that the proposed method achieves higher stuck-at and transition fault coverage in almost all cases, but the higher benefits are on the two largest units, the *ALU* and the *Multiply-Accumulate (MAC)* unit. We note that the variation of the fault coverage of the proposed method was very low (less than 1%) in all cases, while it was higher for the baseline method. For example, the stuck-at fault coverage for the *Multiply-Accumulate (MAC)* unit for 274 $TMIs$ was between 84.48% and 89.18% for the baseline method and between 94.48% and 95.17% for the proposed method. The corresponding ranges for the *Write-Back Mux* unit were $75.14\% - 75.68\%$ and $76.68\% - 76.86\%$ respectively.

In Figure 5.15 and Figure 5.16 we compare test-programs generated using the proposed method with 274 $TMIs$ against SBST programs P1 to P6 (details about these test programs can be found in [206, 207]) in terms of stuck-at and transition-fault coverage. Test programs P1 to P6 have been generated manually, requiring several weeks of significant test-engineering effort. The proposed test-programs clearly outperform the rest of the programs in terms of test-time and test-program size (the test-program size is proportional to the test-time in most of the cases), while they offer

Table 5.7: Software-Based Self-Testing results.

		Test Program Size (KBytes)		Test Time (msec)		Stuck-At FC (%)		Transition FC (%)	
TMIs	**#**	**BSL**	**Prop**	**BSL**	**Prop**	**BSL**	**Prop**	**BSL**	**Prop**
2x137	1	10.9	13.2	0.064	0.079	82.58	87.45	59.00	73.32
	2					83.01	87.84	60.04	73.79
	3					81.96	87.08	58.70	73.32
4x137	1	19.8	23.5	0.109	0.122	86.39	88.92	66.85	75.89
	2					86.53	88.91	66.91	76.06
	3					86.44	88.86	66.84	76.04
8x137	1	37.5	41.8	0.186	0.203	88.14	89.56	73.51	77.62
	2					87.81	89.67	72.62	77.36
	3					88.08	89.78	72.62	77.40

higher or similar (in some cases) stuck-at and transition fault-coverage. Nevertheless, in every case the proposed test-programs offer considerably higher defect-coverage ramp-up, and taking also into account that they were generated very fast and almost fully systematically, the superiority of the proposed method becomes apparent.

Table 5.8: Stuck-at & transition fault coverage per unit.

		Stuck-At Faults (%)		Transition Faults (%)		Number of Faults per Unit
	TMIs	**BSL**	**Prop**	**BSL**	**Prop**	
MULT	2x137	87.60	94.92	63.79	86.14	32,844
	4x137	92.54	95.41	73.64	86.96	
	8x137	94.00	95.67	80.87	88.23	
ALU	2x137	76.90	80.78	47.64	57.46	12,368
	4x137	82.41	85.86	57.97	67.19	
	8x137	85.34	88.57	67.71	70.05	
GENPC	2x137	57.78	58.84	31.71	33.56	4,072
	4x137	58.93	59.91	33.22	34.26	
	8x137	60.44	60.73	35.56	36.09	
CTRL	2x137	83.67	84.30	73.26	74.25	3,982
	4x137	83.94	84.80	74.14	74.40	
	8x137	84.19	84.87	75.03	75.11	
OPMUX	2x137	97.16	97.16	93.36	93.65	2,574
	4x137	97.16	97.16	93.64	93.71	
	8x137	97.16	97.16	93.73	93.73	
IF	2x137	71.41	73.42	50.84	52.74	2,322
	4x137	72.69	73.90	52.58	54.08	
	8x137	73.57	74.19	53.55	54.51	
LSU	2x137	82.45	84.69	55.95	59.89	2,282
	4x137	84.87	85.91	58.05	60.51	
	8x137	85.37	85.97	58.75	61.09	
WBMUX	2x137	75.28	76.78	53.81	56.30	1,826
	4x137	75.61	77.04	55.94	57.22	
	8x137	76.79	77.97	56.72	57.88	

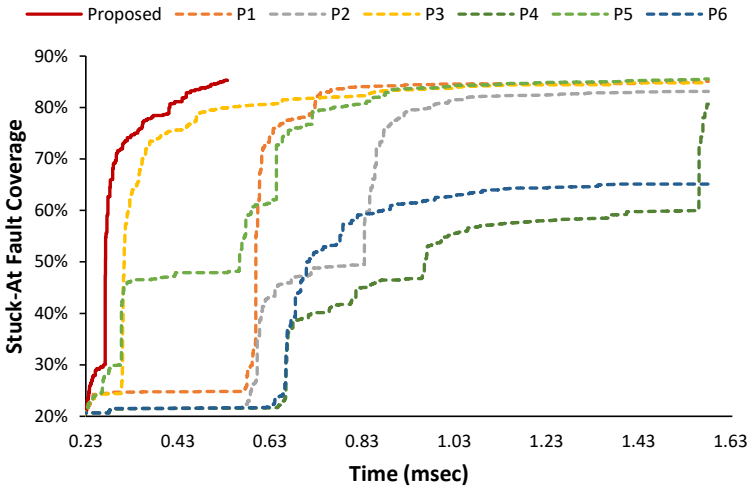


Figure 5.15: Comparisons with other SBST programs (stuck-at faults).

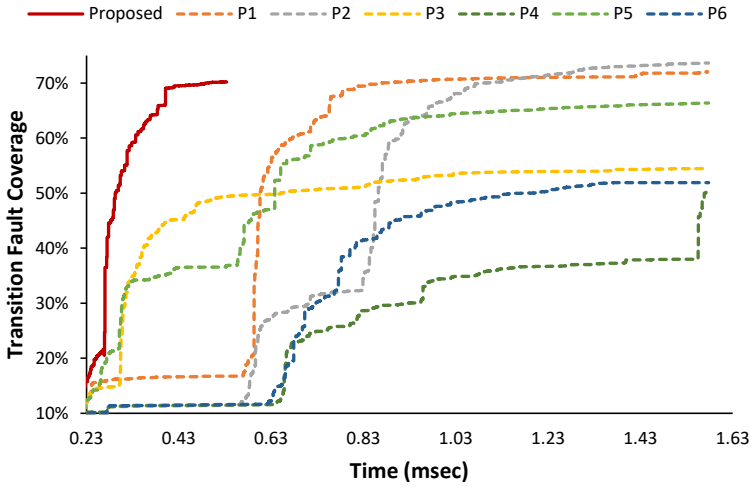


Figure 5.16: Comparisons with other SBST programs (transition faults).

Chapter 6

Conclusions

This research has focused in the development of new TAM architectures and test-scheduling methods for 3D-SoCs, which exploit the high speed offered by TSVs, while power and thermal constraints are met. We presented a new TAM architecture for 3D SoCs, which minimizes the test-time, the number of TSVs, and TAM lines used for transferring test-data to the cores [P.1], [P.3]. The proposed method exploits the high speed offered by the TSVs by means of a very effective TDM method, and a highly efficient optimization method based on rectangle-packing and simulated-annealing. Experiment results on two artificial 3D-SoCs have shown that significant test-time savings can be achieved using the proposed technique, especially under strict power and thermal constraints.

Even though 2D-TDM exploits the high transmission frequency of TSVs, the previous method is compatible only with bus-based TAMs, which are slow and don't exploit the high frequencies of the global channels. We have presented a new TDM-based TAM architecture for 3D SoCs, which supports non-regular division of the frequency among the various layers of the stack [P.4]. The proposed architecture reduces considerably the intra-die connections and permits very high test-time reductions by the means of a very efficient K^3 design-automation process. Experimental results on a 3D-SoC have shown that significant test-time and routing savings are achieved even under strict power-constraints.

This research also has focused on the improvement of the defect screening of processor-based devices, where not-intrusive methods, such as SBST, complement

the DFT solutions. We have presented a SBST test generation method that offers high non-modeled fault coverage in a semi-automatic manner and with short computational time [P.2], [P.5]. Instead of applying time-consuming fault-simulations using multiple fault-models, the proposed method uses logic simulation and a novel SBST-oriented probabilistic metric that exploits both the architectural and the gate-level model of the processor. The proposed method is fast, it is almost fully automated, and it achieves high non-modeled fault-coverage ramp-up. Experiments on the OR1200 processor demonstrate the advantages of the proposed SBST method.

Bibliography

[1] "Cisco IBSG, april 2011." https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf. Accessed: 2019-04-14.

[2] H. Hellmich, "Re-usable low power DSP IP embedded in an ARM based SoC architecture," in *IEE Seminar on Intellectual Property*, pp. 10–10, 01 2000.

[3] L.-T. Wang, C.-W. Wu, and X. Wen, *VLSI Test Principles and Architectures: Design for Testability*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2006.

[4] L.-T. Wang, *System-on-Chip Test Architectures: Nanometer Design for Testability*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2008.

[5] J. Aerts and E. J. Marinissen, "Scan chain design for test time reduction in core-based ICs," in *Proceedings International Test Conference 1998 (IEEE Cat. No.98CH36270)*, pp. 448–457, Oct 1998.

[6] H. S. Lee and K. Chakrabarty, "Test Challenges for 3D Integrated Circuits, year=2009, volume=26, number=5, pages=26-35, keywords=design for testability;integrated circuit design;integrated circuit manufacture;integrated circuit testing;microassembling;3D integrated circuit testing;DFT;IC manufacturing process;IC assembly;monolithic stacking;die stacking;Circuit testing;Integrated circuit testing;Three-dimensional integrated circuits;Automatic testing;Design for testability;Design automation;Manufacturing processes;Electronic design automation and methodology;Manufacturing automation;Stacking;3D ICs;3D integration;interconnect scaling;defects;CMOS;DFT;design and test, doi=10.1109/MDT.2009.125, issn=0740-7475, month=Sep.,," *IEEE Design Test of Computers*.

[7] D. L. Lewis and H. H. S. Lee, "A scanisland based design enabling prebond testability in die-stacked microprocessors," in *2007 IEEE International Test Conference*, pp. 1–8, Oct 2007.

[8] B. Noia, K. Chakrabarty, and E. J. Marinissen, "Optimization methods for post-bond die-internal/external testing in 3D stacked ICs," in *2010 IEEE International Test Conference*, pp. 1–9, Nov 2010.

[9] A. Hsieh, T. Hwang, M. Chang, M. Tsai, C. Tseng, and H. Li, "TSV redundancy: Architecture and design issues in 3D IC," in *2010 Design, Automation Test in Europe Conference Exhibition (DATE 2010)*, pp. 166–171, March 2010.

[10] X. Wu, P. Falkenstern, and Y. Xie, "Scan chain design for three-dimensional integrated circuits (3D ICs)," in *2007 25th International Conference on Computer Design*, pp. 208–214, Oct 2007.

[11] E. J. Marinissen, C. Chi, J. Verbree, and M. Konijnenburg, "3D DfT architecture for pre-bond and post-bond testing," in *2010 IEEE International 3D Systems Integration Conference (3DIC)*, pp. 1–8, Nov 2010.

[12] S. Deutsch, K. Chakrabarty, and E. J. Marinissen, "Uncertainty-aware robust optimization of test-access architectures for 3D stacked ICs," in *2013 IEEE International Test Conference (ITC)*, pp. 1–10, Sep. 2013.

[13] Y. Huang, J. Li, J. Chen, D. Kwai, Y. Chou, and C. Wu, "A built-in self-test scheme for the post-bond test of TSVs in 3D ICs," in *29th VLSI Test Symposium*, pp. 20–25, May 2011.

[14] C. Lung, Y. Su, S. Huang, Y. Shi, and S. Chang, "Fault-tolerant 3D clock network," in *2011 48th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 645–651, June 2011.

[15] D. Juan, S. Garg, and D. Marculescu, "Statistical thermal evaluation and mitigation techniques for 3D Chip-Multiprocessors in the presence of process variations," in *2011 Design, Automation Test in Europe*, pp. 1–6, March 2011.

[16] Y. Yu, C. Chou, J. Li, C. Lo, D. Kwai, Y. Chou, and C. Wu, "A built-in self-test scheme for 3D RAMs," in *2012 IEEE International Test Conference*, pp. 1–9, Nov 2012.

[17] M. Agrawal and K. Chakrabarty, "Test-cost optimization and test-flow selection for 3D-stacked ICs," in *2013 IEEE 31st VLSI Test Symposium (VTS)*, pp. 1–6, April 2013.

[18] P. Georgiou, F. Vartziotis, X. Kavousianos, and K. Chakrabarty, "Testing 3D-SoCs Using 2-D Time-Division Multiplexing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, pp. 3177–3185, Dec 2018.

[19] N. H. E. Weste and K. Eshraghian, *Principles of CMOS VLSI Design: A Systems Perspective*. Reading, Mass. : Addison-Wesley Pub. Co., 1993.

[20] T. C. May and M. H. Woods, "Alpha-particle-induced soft errors in dynamic memories," *IEEE Transactions on Electron Devices*, vol. 26, pp. 2–9, Jan 1979.

[21] M. Bushnell and D. V. Agarwal, *Essentials of Electronic Testing for Digital, Memory, and Mixed-Signal VLSI Circuits*. Springer US, 2000.

[22] T. W. Williams and N. C. Brown, "Defect Level as a Function of Fault Coverage," *IEEE Transactions on Computers*, vol. C-30, pp. 987–988, Dec 1981.

[23] C. E. Stroud, *A Designer's Guide to Built-In Self-Test*. Springer US, 2002.

[24] "Ieee std 1500 - standard for embedded core test." `http://grouper.ieee.org/groups/1500/index.html`. Accessed: 2019-04-14.

[25] "IEEE Standard Test Access Port and Boundary Scan Architecture," *IEEE Std 1149.1-2001*, pp. 1–212, July 2001.

[26] P. Varma and B. Bhatia, "A structured test re-use methodology for core-based system chips," in *Proceedings International Test Conference 1998 (IEEE Cat. No.98CH36270)*, pp. 294–302, Oct 1998.

[27] K.-T. Cheng and C.-J. Lin, "Timing-driven test point insertion for full-scan and partial-scan BIST," in *Proceedings of 1995 IEEE International Test Conference (ITC)*, pp. 506–514, Oct 1995.

[28] C.-J. Lin, Y. Zorian, and S. Bhawmik, "Integration of partial scan and built-in self-test," *Journal of Electronic Testing*, vol. 7, pp. 125–137, Aug 1995.

[29] H.-C. Tsai, S. Bhawmik, and K.-T. Cheng, "An almost full-scan BIST solution-higher fault coverage and shorter test application time," in *Proceedings International Test Conference 1998 (IEEE Cat. No.98CH36270)*, pp. 1065–1073, Oct 1998.

[30] "On-chip bus development working group," 2000.

[31] S. Hellebrand, H. J. Wunderlich, and A. Hertwig, "Mixed-mode BIST using embedded processors," in *Proceedings International Test Conference 1996. Test and Design Validity*, pp. 195–204, Oct 1996.

[32] Y. Xie, G. H. Loh, B. Black, and K. Bernstein, "Design Space Exploration for 3D Architectures," *J. Emerg. Technol. Comput. Syst.*, vol. 2, pp. 65–103, Apr. 2006.

[33] W. R. Davis, J. Wilson, S. Mick, J. Xu, H. Hua, C. Mineo, A. M. Sule, M. Steer, and P. D. Franzon, "Demystifying 3D ICs: the pros and cons of going vertical," *IEEE Design Test of Computers*, vol. 22, pp. 498–510, Nov 2005.

[34] "Intel lakefield." `https://www.digitaltrends.com/computing/intel-lakefield-news-rumors-architecture/`. Accessed: 2019-08-19.

[35] C. Zhang, M. Jung, S. K. Lim, and Y. Shi, "Novel crack sensor for TSV-based 3D integrated circuits: Design and deployment perspectives," in *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 371–378, Nov 2013.

[36] E. J. Marinissen, "Challenges and emerging solutions in testing TSV-based 2 1 over 2D- and 3D-stacked ICs," in *2012 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 1277–1282, March 2012.

[37] B. Noia and K. Chakrabarty, "Pre-bond probing of TSVs in 3D stacked ICs," in *2011 IEEE International Test Conference*, pp. 1–10, Sep. 2011.

[38] P. Chen, C. Wu, and D. Kwai, "On-Chip TSV Testing for 3D IC before Bonding Using Sense Amplification," in *2009 Asian Test Symposium*, pp. 450–455, Nov 2009.

[39] P. Chen, C. Wu, and D. Kwai, "On-chip testing of blind and open-sleeve TSVs for 3D IC before bonding," in *2010 28th VLSI Test Symposium (VTS)*, pp. 263–268, April 2010.

[40] S. Panth and S. K. Lim, "Scan chain and power delivery network synthesis for pre-bond test of 3D ICs," in *29th VLSI Test Symposium*, pp. 26–31, May 2011.

[41] L. Jiangs, Q. Xu, K. Chakrabarty, and T. M. Mak, "Layout-driven test-architecture design and optimization for 3D SoCs under pre-bond test-pin-count constraint," in *2009 IEEE/ACM International Conference on Computer-Aided Design - Digest of Technical Papers*, pp. 191–196, Nov 2009.

[42] J. Li and D. Xiang, "Dft optimization for pre-bond testing of 3D-SICs containing TSVs," in *2010 IEEE International Conference on Computer Design*, pp. 474–479, Oct 2010.

[43] A. Kumar, S. M. Reddy, I. Pomeranz, and B. Becker, "Hyper-graph based partitioning to reduce DFT cost for pre-bond 3D-IC testing," in *2011 Design, Automation Test in Europe*, pp. 1–6, March 2011.

[44] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, "Multilevel hypergraph partitioning: applications in VLSI domain," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 7, pp. 69–79, March 1999.

[45] M. Agrawal and K. Chakrabarty, "A graph-theoretic approach for minimizing the number of wrapper cells for pre-bond testing of 3D-stacked ICs," in *2013 IEEE International Test Conference (ITC)*, pp. 1–10, Sep. 2013.

[46] B. Noia and K. Chakrabarty, "Identification of Defective TSVs in Pre-Bond Testing of 3D ICs," in *2011 Asian Test Symposium*, pp. 187–194, Nov 2011.

[47] B. Noia, S. Panth, K. Chakrabarty, and S. K. Lim, "Scan test of die logic in 3D ICs using TSV probing," in *2012 IEEE International Test Conference*, pp. 1–8, Nov 2012.

[48] L. H. L. Jiang and Q. Xu, "Test architecture design and optimization for three-dimensional SoCs," in *2009 Design, Automation Test in Europe Conference Exhibition*, pp. 220–225, April 2009.

[49] E. J. Marinissen, J. Verbree, and M. Konijnenburg, "A structured and scalable test access architecture for TSV-based 3D stacked ICs," in *2010 28th VLSI Test Symposium (VTS)*, pp. 269–274, April 2010.

[50] B. Noia, S. K. Goel, K. Chakrabarty, E. J. Marinissen, and J. Verbree, "Test-architecture optimization for TSV-based 3D stacked ICs," in *2010 15th IEEE European Test Symposium*, pp. 24–29, May 2010.

[51] B. Noia, K. Chakrabarty, S. K. Goel, E. J. Marinissen, and J. Verbree, "Test-Architecture Optimization and Test Scheduling for TSV-Based 3-D Stacked ICs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, pp. 1705–1718, Nov 2011.

[52] B. Noia, K. Chakrabarty, and E. Marinissen, "Optimization methods for post-bond testing of 3d stacked ics," *J. Electronic Testing*, vol. 28, pp. 103–120, 02 2012.

[53] C. Chi, E. J. Marinissen, S. K. Goel, and C. Wu, "DfT Architecture for 3D-SICs with Multiple Towers," in *2011 Sixteenth IEEE European Test Symposium*, pp. 51–56, May 2011.

[54] Y. Lin, S. Huang, K. Tsai, W. Cheng, S. Sunter, Y. Chou, and D. Kwai, "Parametric Delay Test of Post-Bond Through-Silicon Vias in 3-D ICs via Variable Output Thresholding Analysis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, pp. 737–747, May 2013.

[55] J. Rajski and J. Tyszer, "Fault diagnosis of TSV-based interconnects in 3-D stacked designs," in *2013 IEEE International Test Conference (ITC)*, pp. 1–9, Sep. 2013.

[56] 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 *2012 IEEE International Reliability Physics Symposium (IRPS)*, pp. 5F.1.1–5F.1.12, April 2012.

[57] I. Loi, S. Mitra, T. H. Lee, S. Fujita, and L. Benini, "A low-overhead fault tolerance scheme for TSV-based 3D network on chip links," in *2008 IEEE/ACM International Conference on Computer-Aided Design*, pp. 598–602, Nov 2008.

[58] M. Tsai, A. Klooz, A. Leonard, J. Appel, and P. Franzon, "Through Silicon Via(TSV) defect/pinhole self test circuit for 3D-IC," in *2009 IEEE International Conference on 3D System Integration*, pp. 1–8, Sep. 2009.

[59] Y. Zhao, S. Khursheed, and B. M. Al-Hashimi, "Cost-Effective TSV Grouping for Yield Improvement of 3D-ICs," in *2011 Asian Test Symposium*, pp. 201–206, Nov 2011.

[60] S. Deutsch, K. Chakrabarty, S. Panth, and S. K. Lim, "TSV Stress-Aware ATPG for 3D Stacked ICs," in *2012 IEEE 21st Asian Test Symposium*, pp. 31–36, Nov 2012.

[61] S. Huang, Y. Lin, K. Tsai, W. Cheng, S. Sunter, Y. Chou, and D. Kwai, "Small delay testing for TSVs in 3-D ICs," in *DAC Design Automation Conference 2012*, pp. 1031–1036, June 2012.

[62] C. Metzler, A. Todri-Sanial, A. Bosio, L. Dilillo, P. Girard, A. Virazel, P. Vivet, and M. Belleville, "Computing detection probability of delay defects in signal line tsvs," in *2013 18th IEEE European Test Symposium (ETS)*, pp. 1–6, May 2013.

[63] Y. Cheng, L. Zhang, Y. Han, J. Liu, and X. Li, "Wrapper Chain Design for Testing TSVs Minimization in Circuit-Partitioned 3D SoC," in *2011 Asian Test Symposium*, pp. 181–186, Nov 2011.

[64] S. Kannan, B. Kim, and B. Ahn, "Fault Modeling and Multi-Tone Dither Scheme for Testing 3D TSV Defects," *J. Electronic Testing*, vol. 28, pp. 39–51, 02 2012.

[65] V. Pasca, L. Anghel, M. Nicolaidis, and M. Benabdenbi, "CSL: Configurable Fault Tolerant Serial Links for Inter-die Communication in 3D Systems," *J. Electronic Testing*, vol. 28, pp. 137–150, 02 2012.

[66] R. Rashidzadeh, "Contactless test access mechanism for TSV based 3D ICs," in *2013 IEEE 31st VLSI Test Symposium (VTS)*, pp. 1–6, April 2013.

[67] Y. Lin, S. Huang, K. Tsai, and W. Cheng, "Programmable Leakage Test and Binning for TSVs," in *2012 IEEE 21st Asian Test Symposium*, pp. 43–48, Nov 2012.

[68] M. Hashizume, T. Konishi, H. Yotsuyanag, and S. Lu, "Testable Design for Electrical Testing of Open Defects at Interconnects in 3D ICs," in *2013 22nd Asian Test Symposium*, pp. 13–18, Nov 2013.

[69] S. Huang, Y. Lin, L. Huang, K. Tsai, and W. Cheng, "Programmable Leakage Test and Binning for TSVs with Self-Timed Timing Control," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, pp. 1265–1273, Aug 2013.

[70] F. Ye and K. Chakrabarty, "Tsv open defects in 3D integrated circuits: Characterization, test, and optimal spare allocation," in *DAC Design Automation Conference 2012*, pp. 1024–1030, June 2012.

[71] C. Chi, C. Wu, M. Wang, and H. Lin, "3D-IC interconnect test, diagnosis, and repair," in *2013 IEEE 31st VLSI Test Symposium (VTS)*, pp. 1–6, April 2013.

[72] C. Shih, S. Hsieh, Y. Lu, J. C. Li, T. Wu, and K. Chakrabarty, "Test Generation of Path Delay Faults Induced by Defects in Power TSV," in *2013 22nd Asian Test Symposium*, pp. 43–48, Nov 2013.

[73] J. You, S. Huang, D. Kwai, Y. Chou, and C. Wu, "Performance Characterization of TSV in 3D IC via Sensitivity Analysis," in *2010 19th IEEE Asian Test Symposium*, pp. 389–394, Dec 2010.

[74] J. You, S. Huang, Y. Lin, M. Tsai, D. Kwai, Y. Chou, and C. Wu, "In-Situ Method for TSV Delay Testing and Characterization Using Input Sensitivity Analysis," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, pp. 443–453, March 2013.

[75] C. Pai, R. Cu, B. Cheng, L. Chen, and K. S. Li, "A Unified Interconnects Testing Scheme for 3D Integrated Circuits," in *2011 Asian Test Symposium*, pp. 195–200, Nov 2011.

[76] S. Deutsch and K. Chakrabarty, "Non-invasive pre-bond TSV test using ring oscillators and multiple voltage levels," in *2013 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 1065–1070, March 2013.

[77] D. L. Lewis and H. S. Lee, "Testing Circuit-Partitioned 3D IC Designs," in *2009 IEEE Computer Society Annual Symposium on VLSI*, pp. 139–144, May 2009.

[78] B. Noia, K. Chakrabarty, and Y. Xie, "Test-wrapper optimization for embedded cores in TSV-based three-dimensional SOCs," in *2009 IEEE International Conference on Computer Design*, pp. 70–77, Oct 2009.

[79] C. Chou, J. Li, J. Chen, D. Kwai, Y. Chou, and C. Wu, "A Test Integration Methodology for 3D Integrated Circuits," in *2010 19th IEEE Asian Test Symposium*, pp. 377–382, Dec 2010.

[80] C. Papameletis, B. Keller, V. Chickermane, E. J. Marinissen, and S. Hamdioui, "Automated DfT insertion and test generation for 3D-SICs with embedded cores and multiple towers," in *2013 18th IEEE European Test Symposium (ETS)*, pp. 1–6, May 2013.

[81] D. L. Lewis, S. Panth, X. Zhao, S. K. Lim, and H. S. Lee, "Designing 3D test wrappers for pre-bond and post-bond test of 3D embedded cores," in *2011 IEEE 29th International Conference on Computer Design (ICCD)*, pp. 90–95, Oct 2011.

[82] C. C.-H. Liao, A. W.-T. Chen, L. Y.-Z. Lin, and C. H.-P. Wen, "Fast Scan-Chain Ordering for 3-D-IC Designs Under Through-Silicon-Via (TSV) Constraints," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, pp. 1170–1174, June 2013.

[83] B. Noia and K. Chakrabarty, "Retiming for Delay Recovery After DfT Insertion on Interdie Paths in 3-D ICs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, pp. 464–475, March 2014.

[84] X. Wu, Y. Chen, K. Chakrabarty, and Y. Xie, "Test-access mechanism optimization for core-based three-dimensional SOCs," in *2008 IEEE International Conference on Computer Design*, pp. 212–218, Oct 2008.

[85] C. Lo, Y. Hsing, L. Denq, and C. Wu, "SOC Test Architecture and Method for 3-D ICs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, pp. 1645–1649, Oct 2010.

[86] S. Deutsch and K. Chakrabarty, "Robust optimization of test-architecture designs for core-based SoCs," in *2013 18th IEEE European Test Symposium (ETS)*, pp. 1–6, May 2013.

[87] M. Agrawal, K. Chakrabarty, and B. Eklow, "A distributed, reconfigurable, and reusable bist infrastructure for 3D-stacked ICs," in *2014 International Test Conference*, pp. 1–10, Oct 2014.

[88] M. Cho, C. Liu, D. H. Kim, S. K. Lim, and S. Mukhopadhyay, "Design method and test structure to characterize and repair TSV defect induced signal degradation in 3D system," in *2010 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 694–697, Nov 2010.

[89] Y. Huang and J. Li, "Built-In Self-Repair Scheme for the TSVs in 3-D ICs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, pp. 1600–1613, Oct 2012.

[90] V. Pasca, L. Anghel, and M. Benabdenbi, "Kth-aggressor fault (kaf)-based thru-silicon-via interconnect built-in self-test and diagnosis," *Journal of Electronic Testing*, vol. 28, 12 2012.

[91] B. SenGupta, U. Ingelsson, and E. Larsson, "Scheduling Tests for 3D Stacked Chips under Power Constraints," *J. Electronic Testing*, vol. 28, no. 1, pp. 121–135, 2012.

[92] C. Hou and J. Li, "Allocation of RAM built-in self-repair circuits for SOC dies of 3D ICs," in *2013 IEEE 31st VLSI Test Symposium (VTS)*, pp. 1–6, April 2013.

[93] L. Jiang, F. Ye, Q. Xu, K. Chakrabarty, and B. Eklow, "On effective and efficient in-field TSV repair for stacked 3D ICs," in *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, May 2013.

[94] C. Yang, C. Chou, and J. Li, "A TSV Repair Scheme Using Enhanced Test Access Architecture for 3-D ICs," in *2013 22nd Asian Test Symposium*, pp. 7–12, Nov 2013.

[95] E. G. Friedman, "Clock distribution networks in synchronous digital integrated circuits," *Proceedings of the IEEE*, vol. 89, pp. 665–692, May 2001.

[96] M. Mondal, A. J. Ricketts, S. Kirolos, T. Ragheb, G. Link, N. Vijaykrishnan, and Y. Massoud, "Thermally Robust Clocking Schemes for 3D Integrated Circuits," in *2007 Design, Automation Test in Europe Conference Exhibition*, pp. 1–6, April 2007.

[97] J. Minz, C. Zhao, and S. K. Lim, "Buffered clock tree synthesis for 3D ICs under thermal variations," in *2008 Asia and South Pacific Design Automation Conference*, pp. 504–509, March 2008.

[98] X. Zhao, D. L. Lewis, H. S. Lee, and S. K. Lim, "Pre-bond testable low-power clock tree design for 3D stacked ICs," in *2009 IEEE/ACM International Conference on Computer-Aided Design - Digest of Technical Papers*, pp. 184–190, Nov 2009.

[99] M. Buttrick and S. Kundu, "On testing prebond dies with incomplete clock networks in a 3D IC using DLLs," in *2011 Design, Automation Test in Europe*, pp. 1–6, March 2011.

[100] V. Arunachalam and W. Burleson, "Low-power clock distribution in a multi-layer core 3D microprocessor," in *2008 ACM Great Lakes Symposium on VLSI*, pp. 429–434, 01 2008.

[101] V. F. Pavlidis, I. Savidis, and E. G. Friedman, "Clock distribution networks for 3-D ictegrated Circuits," in *2008 IEEE Custom Integrated Circuits Conference*, pp. 651–654, Sep. 2008.

[102] T. Kim and T. Kim, "Clock tree embedding for 3D ICs," in *2010 15th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 486–491, Jan 2010.

[103] T. Kim and T. Kim, "Clock tree synthesis with pre-bond testability for 3D stacked IC Designs," in *Design Automation Conference*, pp. 723–728, June 2010.

[104] T.-Y. Kim and T. Kim, "Clock tree synthesis for tsv-based 3d ic designs," *ACM Trans. Design Autom. Electr. Syst.*, vol. 16, p. 48, 10 2011.

[105] T. Kim and T. Kim, "Resource Allocation and Design Techniques of Prebond Testable 3-D Clock Tree," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, pp. 138–151, Jan 2013.

[106] X. Zhao, D. L. Lewis, H. S. Lee, and S. K. Lim, "Low-Power Clock Tree Design for Pre-Bond Testing of 3-D Stacked ICs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, pp. 732–745, May 2011.

[107] X. Zhao, J. Minz, and S. K. Lim, "Low-Power and Reliable Clock Network Design for Through-Silicon Via (TSV) Based 3D ICs," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 1, pp. 247–259, Feb 2011.

[108] H. Park and T. Kim, "Comprehensive technique for designing and synthesizing TSV Fault-tolerant 3D clock trees," in *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 691–696, Nov 2013.

[109] J. Cong, J. Wei, and Y. Zhang, "A thermal-driven floorplanning algorithm for 3D ICs," in *IEEE/ACM International Conference on Computer Aided Design, 2004. ICCAD-2004.*, pp. 306–313, Nov 2004.

[110] D. Xiang, K. Shen, and Y. Deng, "A Thermal-Driven Test Application Scheme for 3-Dimensional ICs," in *2012 IEEE 21st Asian Test Symposium*, pp. 101–106, Nov 2012.

[111] K. Dev, G. Woods, and S. Reda, "High-throughput TSV testing and characterization for 3D integration using thermal mapping," in *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, May 2013.

[112] Y. Xie, "Future memory and interconnect technologies," in *2013 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 964–969, March 2013.

[113] W. Kang, C. Lee, K. Cho, and S. Kang, "A Die Selection and Matching Method with Two Stages for Yield Enhancement of 3-D Memories," in *2013 22nd Asian Test Symposium*, pp. 301–306, Nov 2013.

[114] N. V. Y.-F. Tsai, Y. Xie and M. J. Irwin, "Three-dimensional cache design exploration using 3DCacti," in *2005 International Conference on Computer Design*, pp. 519–524, Oct 2005.

[115] L. Jiang, Y. Liu, L. Duan, Y. Xie, and Q. Xu, "Modeling TSV open defects in 3D-stacked DRAM," in *2010 IEEE International Test Conference*, pp. 1–9, Nov 2010.

[116] L. Jiang, R. Ye, and Q. Xu, "Yield enhancement for 3D-stacked memory by redundancy sharing across dies," in *2010 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 230–234, Nov 2010.

[117] M. Taouil and S. Hamdioui, "Layer Redundancy Based Yield Improvement for 3D Wafer-to-Wafer Stacked Memories," in *2011 Sixteenth IEEE European Test Symposium*, pp. 45–50, May 2011.

[118] B. Lin, M. Lee, and C. Wu, "Exploration Methodology for 3D Memory Redundancy Architectures under Redundancy Constraints," in *2013 22nd Asian Test Symposium*, pp. 1–6, Nov 2013.

[119] M. Taouil, M. Masadeh, S. Hamdioui, and E. J. Marinissen, "Interconnect test for 3D stacked memory-on-logic," in *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 1–6, March 2014.

[120] Y. Chen, D. Niu, Y. Xie, and K. Chakrabarty, "Cost-effective integration of three-dimensional (3D) ICs emphasizing testing cost analysis," in *2010 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 471–476, Nov 2010.

[121] M. Taouil, S. Hamdioui, K. Beenakker, and E. J. Marinissen, "Test Cost Analysis for 3D Die-to-Wafer Stacking," in *2010 19th IEEE Asian Test Symposium*, pp. 435–441, Dec 2010.

[122] S. Hamdioui and M. Taouil, "Yield Improvement and Test Cost Optimization for 3D Stacked ICs," in *2011 Asian Test Symposium*, pp. 480–485, Nov 2011.

[123] Y. Chen, Y. Huang, and J. Li, "Test cost optimization technique for the pre-bond test of 3D ICs," in *2012 IEEE 30th VLSI Test Symposium (VTS)*, pp. 102–107, April 2012.

[124] M. L. Y.-W. Chou, P.-Y. Chen and C. Wu, "Cost modeling and analysis for interposer-based three-dimensional IC," in *2012 IEEE 30th VLSI Test Symposium (VTS)*, pp. 108–113, April 2012.

[125] J. Abraham and S. Thatte, "Test Generation for Microprocessors," *IEEE Transactions on Computers*, vol. 29, pp. 429–441, jun 1980.

[126] J. Abraham and D. Brahme, "Functional Testing of Microprocessors," *IEEE Transactions on Computers*, vol. 33, pp. 475–485, jun 1984.

[127] M. J. Flynn, "Trends and Problems in Computer Organizations," in *IFIP Congress*, pp. 3–10, 1974.

[128] W.-C. Lai, A. Krstic, and K.-T. Cheng, "On testing the path delay faults of a microprocessor using its instruction set," in *Proceedings 18th IEEE VLSI Test Symposium*, pp. 15–20, April 2000.

[129] W.-C. Lai, A. Krstic, and K.-T. Cheng, "Test program synthesis for path delay faults in microprocessor cores," in *Proceedings International Test Conference 2000 (IEEE Cat. No.00CH37159)*, pp. 1080–1089, Oct 2000.

[130] L. Chen and S. Dey, "Software-based self-testing methodology for processor cores," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, pp. 369–380, March 2001.

[131] L. Chen, S. Ravi, A. Raghunathan, and S. Dey, "A scalable software-based self-test methodology for programmable processors," in *Proceedings 2003. Design Automation Conference (IEEE Cat. No.03CH37451)*, pp. 548–553, June 2003.

[132] X. Bai, L. Chen, and S. Dey, "Software-based self-test methodology for crosstalk faults in processors," in *Eighth IEEE International High-Level Design Validation and Test Workshop*, pp. 11–16, Nov 2003.

[133] X. Bai, S. Dey, and J. Rajski, "Self-test methodology for at-speed test of crosstalk in chip interconnects," in *Proceedings 37th Design Automation Conference*, pp. 619–624, June 2000.

[134] R. Sharma and K. Iniewski, *Design of 3D Integrated Circuits and Systems*. CRC Press, 2015.

[135] ISO/DIS26262, "Road vehicles - functional safety," 2009.

[136] F. Reimann, M. Glaß, J. Teich, A. Cook, L. R. Gómez, D. Ull, H.-J. Wunderlich, P. Engelke, and U. Abelein, "Advanced Diagnosis: SBST and BIST Integration in Automotive E/E Architectures," in *Proceedings of the 51st Annual Design Automation Conference*, DAC '14, (New York, NY, USA), pp. 96:1–96:9, ACM, 2014.

[137] K. Constantinides, O. Mutlu, T. Austin, and V. Bertacco, "A Flexible Software-Based Framework for Online Detection of Hardware Defects," *IEEE Transactions on Computers*, vol. 58, pp. 1063–1079, Aug 2009.

[138] A. Paschalis and D. Gizopoulos, "Effective software-based self-test strategies for on-line periodic testing of embedded processors," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, pp. 88–99, Jan 2005.

[139] B. Vermeulen, C. Hora, B. Kruseman, E. J. Marinissen, and R. van Rijsinge, "Trends in testing integrated circuits," in *2004 International Conferce on Test*, pp. 688–697, Oct 2004.

[140] S. K. Roy, C. Giri, S. Ghosh, and H. Rahaman, "Optimizing Test Wrapper for Embedded Cores Using TSV Based 3D SOCs," in *2011 IEEE Computer Society Annual Symposium on VLSI*, pp. 31–36, July 2011.

[141] C. Chi, E. J. Marinissen, S. K. Goel, and C. Wu, "DfT Architecture for 3D-SICs with Multiple Towers," in *2011 Sixteenth IEEE European Test Symposium*, pp. 51–56, May 2011.

[142] D. Xiang, K. Chakrabarty, and H. Fujiwara, "Multicast-Based Testing and Thermal-Aware Test Scheduling for 3D ICs with a Stacked Network-on-Chip," *IEEE Transactions on Computers*, vol. 65, pp. 2767–2779, Sep. 2016.

[143] S. Deutsch, K. Chakrabarty, and E. J. Marinissen, "Robust Optimization of Test-Access Architectures Under Realistic Scenarios," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, pp. 1873–1884, Nov 2015.

[144] C. Chi, E. J. Marinissen, S. K. Goel, and C. Wu, "Post-bond testing of 2.5D-SICs and 3D-SICs containing a passive silicon interposer base," in *2011 IEEE International Test Conference*, pp. 1–10, Sep. 2011.

[145] I. Ndip, B. Curran, K. Lobbicke, S. Guttowski, H. Reichl, K. Lang, and H. Henke, "High-Frequency Modeling of TSVs for 3-D Chip Integration and Silicon Interposers Considering Skin-Effect, Dielectric Quasi-TEM and Slow-Wave Modes," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 1, pp. 1627–1641, Oct 2011.

[146] A. Sheibanyrad and F. Pétrot, *Asynchronous 3D-NoCs Making Use of Serialized Vertical Links*, pp. 149–165. New York, NY: Springer New York, 2011.

[147] F. Sun, A. Cevrero, P. Athanasopoulos, and Y. Leblebici, "Design and feasibility of multi-Gb/s quasi-serial vertical interconnects based on TSVs for 3D ICs," in *2010 18th IEEE/IFIP International Conference on VLSI and System-on-Chip*, pp. 149–154, Sep. 2010.

[148] "Iwls." `http://iwls.org/iwls2005/benchmarks.html`. Accessed: 2019-04-14.

[149] Y. Lee, I. Hong, and S. K. Lim, "Slew-aware buffer insertion for through-silicon-via-based 3D ICs," in *Proceedings of the IEEE 2012 Custom Integrated Circuits Conference*, pp. 1–8, Sep. 2012.

[150] M. Murugesan, Y. Ohara, T. Fukushima, T. Tanaka, and M. Koyanagi, "Low-Resistance Cu-Sn Electroplated–Evaporated Microbumps for 3D Chip Stacking," *Journal of Electronic Materials*, vol. 41, pp. 720–729, Apr 2012.

[151] P. Garrou, C. Bower, and P. Ramm, *Handbook of 3D Integration*. Wiley-VCH, 2012.

[152] B. Chazelle, "The Bottomn-Left Bin-Packing Heuristic: An Efficient Implementation," *IEEE Transactions on Computers*, vol. C-32, pp. 697–707, Aug 1983.

[153] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes: The Art of Scientific Computing* , ch. 10.12. New York: Cambridge University Press, 3rd ed., 2007.

[154] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science 220*, pp. 671–680, 1983.

[155] F. Vartziotis, X. Kavousianos, K. Chakrabarty, A. Jain, and R. Parekhji, "Time-Division Multiplexing for Testing DVFS-Based SoCs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, pp. 668–681, April 2015.

[156] F. Vartziotis, X. Kavousianos, P. Georgiou, and K. Chakrabarty, "A Branch--Bound Test-Access-Mechanism Optimization Method for Multi- v_{dd} SoCs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, pp. 1911–1924, Nov 2017.

[157] D. Xiang, K. Shen, B. B. Bhattacharya, X. Wen, and X. Lin, "Thermal-Aware Small-Delay Defect Testing in Integrated Circuits for Mitigating Overkill," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, pp. 499–512, March 2016.

[158] T. Cormen, L. Charles, R. Rivest, and C. Stein, *Introduction To Algorithms (Third ed.)*. MIT Press, 2009.

[159] R. E. Korf, "A complete anytime algorithm for number partitioning," *Artificial Intelligence*, vol. 106, no. 2, pp. 181 – 203, 1998.

[160] N. Karmarkar and R. M. Karp, "The differencing method of set partitioning," 1982.

[161] "Arm." `https://developer.arm.com/technologies/functional-safety`. Accessed: 2018-09-18.

[162] "Microchip." `http://ww1.microchip.com/downloads/en/DeviceDoc/52076a.pdf`. Accessed: 2018-09-18.

[163] "Renesas." `https://www.renesas.com/en-eu/products/synergy/software/add-ons.html#read`. Accessed: 2018-09-18.

[164] "Cypress." `http://www.cypress.com/file/249196/download`. Accessed: 2018-09-18.

[165] "St." `http://www.st.com/content/ccc/resource/technical/document/application_note/02/1a/91/78/e4/15/4d/35/CD00290100.pdf/files/CD00290100.pdf/jcr:content/translations/en.CD00290100.pdf`. Accessed: 2018-09-18.

[166] "Hitex." `https://www.hitex.com/software-components/selftest-libraries-safety-libs/pro-sil-safetcore-safetlib/`. Accessed: 2018-09-18.

[167] S. M. Thatte and J. A. Abraham, "Test Generation for Microprocessors," *IEEE Transactions on Computers*, vol. C-29, pp. 429–441, June 1980.

[168] A. Paschalis, D. Gizopoulos, N. Kranitis, M. Psarakis, and Y. Zorian, "Deterministic software-based self-testing of embedded processor cores," in *Proceedings*

Design, Automation and Test in Europe. Conference and Exhibition 2001, pp. 92–96, March 2001.

[169] C. H. P. Wen, L.-C. Wang, and K.-T. Cheng, "Simulation-Based Functional Test Generation for Embedded Processors," *IEEE Transactions on Computers*, vol. 55, pp. 1335–1343, Nov 2006.

[170] P. Singh, D. L. Landis, and V. Narayanan, "Test Generation for Precise Interrupts on Out-of-Order Microprocessors," in *2009 10th International Workshop on Microprocessor Test and Verification*, pp. 79–82, Dec 2009.

[171] F. Corno, M. Sonza Reorda, G. Squillero, and M. Violante, "On the test of microprocessor IP cores," in *Proceedings Design, Automation and Test in Europe. Conference and Exhibition 2001*, pp. 209–213, March 2001.

[172] M. A. Skitsas, C. A. Nicopoulos, and M. K. Michael, "Daemonguard: O/S-assisted selective software-based self-testing for multi-core systems," in *2013 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS)*, pp. 45–51, Oct 2013.

[173] S. Di Carlo, P. Prinetto, and A. Savino, "Software-Based Self-Test of Set-Associative Cache Memories," *IEEE Transactions on Computers*, vol. 60, pp. 1030–1044, July 2011.

[174] A. Apostolakis, D. Gizopoulos, M. Psarakis, and A. Paschalis, "Software-Based Self-Testing of Symmetric Shared-Memory Multiprocessors," *IEEE Transactions on Computers*, vol. 58, pp. 1682–1694, Dec 2009.

[175] G. Xenoulis, D. Gizopoulos, M. Psarakis, and A. Paschalis, "Instruction-Based Online Periodic Self-Testing of Microprocessors with Floating-Point Units," *IEEE Transactions on Dependable and Secure Computing*, vol. 6, pp. 124–134, April 2009.

[176] P. Bernardi, C. Bovi, R. Cantoro, S. De Luca, R. Meregalli, D. Piumatti, E. Sanchez, and A. Sansonetti, "Software-based self-test techniques of computational modules in dual issue embedded processors," in *2015 20th IEEE European Test Symposium (ETS)*, pp. 1–2, May 2015.

[177] F. Corno, E. Sanchez, M. S. Reorda, and G. Squillero, "Automatic test program generation: a case study," *IEEE Design Test of Computers*, vol. 21, pp. 102–109, March 2004.

[178] N. Kranitis, A. Paschalis, D. Gizopoulos, and G. Xenoulis, "Software-based self-testing of embedded processors," *IEEE Transactions on Computers*, vol. 54, pp. 461–475, April 2005.

[179] N. Kranitis, A. Merentitis, G. Theodorou, A. Paschalis, and D. Gizopoulos, "Hybrid-SBST Methodology for Efficient Testing of Processor Cores," *IEEE Design Test of Computers*, vol. 25, pp. 64–75, Jan 2008.

[180] E. Sanchez and M. S. Reorda, "On the Functional Test of Branch Prediction Units," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, pp. 1675–1688, Sep. 2015.

[181] D. Sabena, M. S. Reorda, and L. Sterpone, "A new SBST algorithm for testing the register file of VLIW processors," in *2012 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 412–417, March 2012.

[182] P. Bernardi, R. Cantoro, L. Ciganda, B. Du, E. Sanchez, M. S. Reorda, M. Grosso, and O. Ballan, "On the Functional Test of the Register Forwarding and Pipeline Interlocking Unit in Pipelined Processors," in *2013 14th International Workshop on Microprocessor Test and Verification*, pp. 52–57, Dec 2013.

[183] P. Bernardi, R. Cantoro, L. Ciganda, E. Sanchez, M. S. Reorda, S. De Luca, R. Meregalli, and A. Sansonetti, "On the in-field functional testing of decode units in pipelined RISC processors," in *2014 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, pp. 299–304, Oct 2014.

[184] M. Schölzel, T. Koal, and H. T. Vierhaus, "Systematic generation of diagnostic software-based self-test routines for processor components," in *2014 19th IEEE European Test Symposium (ETS)*, pp. 1–6, May 2014.

[185] P. Bernardi, L. Ciganda, M. de Carvalho, M. Grosso, J. Lagos-Benites, E. Sanchez, M. S. Reorda, and O. Ballan, "On-line software-based self-test of the Address Calculation unit in RISC processors," in *2012 17th IEEE European Test Symposium (ETS)*, pp. 1–6, May 2012.

[186] G. Squillero, "Artificial evolution in computer aided design: from the optimization of parameters to the creation of assembly programs," *Computing*, vol. 93, pp. 103–120, Dec 2011.

[187] A. Apostolakis, D. Gizopoulos, M. Psarakis, D. Ravotto, and M. S. Reorda, "Test Program Generation for Communication Peripherals in Processor-Based SoC Devices," *IEEE Design Test of Computers*, vol. 26, pp. 52–63, March 2009.

[188] M. Grosso, W. J. H. Perez, D. Ravotto, E. Sanchez, M. S. Reorda, and J. V. Medina, "A software-based self-test methodology for system peripherals," in *2010 15th IEEE European Test Symposium*, pp. 195–200, May 2010.

[189] P. Bernardi, M. Grosso, E. Sanchez, and O. Ballan, "Fault grading of software-based self-test procedures for dependable automotive applications," in *2011 Design, Automation Test in Europe*, pp. 1–2, March 2011.

[190] M. Psarakis, D. Gizopoulos, E. Sanchez, and M. Sonza Reorda, "Microprocessor Software-Based Self-Testing," *IEEE Design Test of Computers*, vol. 27, pp. 4–19, May 2010.

[191] N. Kranitis, A. Paschalis, D. Gizopoulos, and Y. Zorian, "Effective software self-test methodology for processor cores," in *Proceedings 2002 Design, Automation and Test in Europe Conference and Exhibition*, pp. 592–597, March 2002.

[192] A. Sanyal, K. Chakrabarty, M. Yilmaz, and H. Fujiwara, "RT-level design-for-testability and expansion of functional test sequences for enhanced defect coverage," in *2010 IEEE International Test Conference*, pp. 1–10, Nov 2010.

[193] H. Fang, K. Chakrabarty, A. Jas, S. Patil, and C. Tirumurti, "Functional Test-Sequence Grading at Register-Transfer Level," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, pp. 1890–1894, Oct 2012.

[194] Z. Wang and K. Chakrabarty, "Test-Quality/Cost Optimization Using Output-Deviation-Based Reordering of Test Patterns," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, pp. 352–365, Feb 2008.

[195] Z. Wang, H. Fang, K. Chakrabarty, and M. Bienek, "Deviation-Based LFSR Reseeding for Test-Data Compression," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, pp. 259–271, Feb 2009.

[196] X. Kavousianos, V. Tenentes, K. Chakrabarty, and E. Kalligeros, "Defect-Oriented LFSR Reseeding to Target Unmodeled Defects Using Stuck-at Test Sets," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, pp. 2330–2335, Dec 2011.

[197] F. J. Ferguson and J. P. Shen, "A CMOS fault extractor for inductive fault analysis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 7, pp. 1181–1194, Nov 1988.

[198] "Openrisc 1200." `https://opencores.org/ocsvn/openrisc/openrisc/trunk/or1200/doc/openrisc1200_spec.pdf`. Accessed: 2019-04-14.

[199] "Nangate." `http://www.nangate.com/?page_id=2325`. Accessed: 2016-06-16.

[200] D. Anastasakis, R. Damiano, H.-K. T. Ma, and T. Stanion, "A Practical and Efficient Method for Compare-point Matching," in *Proceedings of the 39th Annual Design Automation Conference*, DAC '02, (New York, NY, USA), pp. 305–310, ACM, 2002.

[201] D. Perry and H. Foster, *Applied Formal Verification*. McGraw-Hill, 2005.

[202] F. Vartziotis, *Advanced Methods for Testing Multi-Core SoCs*. PhD Thesis, University of Ioannnina, 2016.

[203] "Hotspot tool suite." `http://lava.cs.virginia.edu/HotSpot/index.htm/`. Accessed: 2019-08-19.

[204] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. R. Stan, "HotSpot: a compact thermal modeling methodology for early-stage VLSI design," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, pp. 501–513, May 2006.

[205] H. Cherupalli, H. Duwe, W. Ye, R. Kumar, and J. Sartori, "Bespoke processors for applications with ultra-low area and power constraints," in *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, pp. 41–54, June 2017.

[206] G. An, R. Cantoro, E. Sanchez, and M. S. Reorda, "On the detection of board delay faults through the execution of functional programs," in *2017 18th IEEE Latin American Test Symposium (LATS)*, pp. 1–6, March 2017.

[207] R. Cantoro, S. Carbonara, A. Floridia, E. Sanchez, M. S. Reorda, and J. Mess, "An analysis of test solutions for COTS-based systems in space applications," in *2018 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, pp. 59–64, Oct 2018.

Glossary

3D-IC Three-Dimensional Integrated Circuit.
A-TM Accumulated-Test Macro
ARM Advanced RISC Machines.
ASIC Application Specific Integrated Circuit.
ATE Automatic Test Equipment.
ATPG Automatic Test Pattern Generation.
BIST Built-In Self-Test.
BISR Built-In Self-Repair.
BL Bottom Left.
CL Confidence Level.
CR Cooling Rate.
CMOS Complementary Metal–Oxide–Semiconductor.
CUT Circuit Under Test.
D2D Die-to-Die.
D2W Die-to-Wafer.
DUT Device Under Test.
DFM Design for Manufacturability.
DFT Design For Testability.
DLL Delay Lock Loop.
DRAM Dynamic Random Access Memory.
DSP Digital Signal Processor.
FV Functional Vector.
GTC Global Test Clock.
HDL Hardware Description Language.
I-TM Independent-Test Macro
IC Integration Circuit.

IP Intellectual Property.
ILP Integer Linear Programming.
IoT Internet of Things.
KGD Known Good Die.
KOZ Keep-Out-Zone.
LTC Local Test Clock.
LOC Launch-On-Capture.
MMU Memory Management Unit.
NoC Network on Chip.
ORA Output Response Analyzer.
PCB Printed Circuit Board.
PPM Parts Per Million.
RAM Random Access Memory.
RP Rectangle Packing.
RISC Reduced Instruction Set Computer.
RTL Register-Transfer Level
SBST Software-Based Self-Testing.
SECT Standard for Embedded Core Test.
SA Simulated Annealing.
SoC System on Chip.
TAM Test Access Mechanism.
TAP Test Access Port.
TDM Time Division Multiplexing.
TMI Test Macro Instance.
TPG Test Pattern Generator.
TSV Through-Silicon Via.
TT Test Time.
VLSI Very-Large-Scale Integration.
W2W Wafer-to-Wafer.
WBC Wrapper Boundary Cell.
WBR Wrapper Boundary Register.
WBY Wrapper Bypass Register.
WC Wrapper Chain.

WIR Wrapper Instruction Register.

WL Wire-Length.

WPC Wrapper Parallel Control.

WPI Wrapper Parallel Input.

WPO Wrapper Parallel Output.

WSC Wrapper Serial Control.

WSI Wrapper Serial Input.

WSO Wrapper Serial Output.

WSP Wrapper Serial Port.

Author's Publications

1. P. Georgiou, F. Vartziotis, X. Kavousianos and K. Chakrabarty, "Two-dimensional time-division multiplexing for 3D-SoCs", *Proc. of IEEE European Test Symposium* (2016).

2. P. Georgiou, X. Kavousianos, R. Cantoro and M. Sonza Reorda, "Fault-Independent Test-Generation for Software-Based Self-Testing", *Proc. of IEEE International Symposium on On-Line Testing and Robust System Design* (2018).

3. P. Georgiou, F. Vartziotis, X. Kavousianos and K. Chakrabarty, "Testing 3D-SoCs Using 2-D Time-Division Multiplexing", *IEEE Trans. on CAD of Integrated Circuits and Systems* vol 37 (2018) 3177-3185.

4. P. Georgiou, I. Theodosopoulos and X. Kavousianos, "K^3 TAM Optimization for Testing 3D-SoCs using Non-Regular Time-Division-Multiplexing", *Proc. of IEEE European Test Symposium* (2019).

5. P. Georgiou, X. Kavousianos, R. Cantoro and M. Sonza Reorda, "Fault-Independent Test-Generation for Software-Based Self-Testing", *IEEE Trans. on Device and Materials Reliability* vol 19 issue 2 (2019) 341-349.

Short Biography


Panagiotis Georgiou received his Diploma from the Department of Computer Engineering & Informatics (University of Patras, GR), in 2012, and the M.Sc. degree in Hardware-Software Integrated Systems from the same University in 2014. Then, he started pursuing his PhD at University of Ioannina. His main research interests include DFT, 3D-IC testing, Software Based Self-Testing, power/thermal- aware test, low power design and testing. He has worked as a Junior Digital IC Design Engineer with Analogies S.A., where he was responsible for the design and verification of various blocks for a MIMO-OFDM telecommunication system of high complexity. Since 2018, he is working as a Software Engineer with Syntax IT Inc.