Design-for-Test and Test Optimization

Techniques for TSV-based 3D Stacked ICs

by

Brandon Noia

Department of Electrical and Computer Engineering
Duke University

Date: _____

Approved:

_____
Krishnendu Chakrabarty, Supervisor

_____
John Board

_____
Christopher Dwyer

_____
Hisham Massoud

_____
Patrick Wolf

Dissertation submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in the Department of Electrical and Computer Engineering
in the Graduate School of Duke University
2014

ABSTRACT

Design-for-Test and Test Optimization Techniques for TSV-based 3D Stacked ICs

by

Brandon Noia

Department of Electrical and Computer Engineering
Duke University

Date: _____

Approved:

_____
Krishnendu Chakrabarty, Supervisor

_____
John Board

_____
Christopher Dwyer

_____
Hisham Massoud

_____
Patrick Wolf

An abstract of a dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Electrical and Computer Engineering
in the Graduate School of Duke University
2014

Abstract

As integrated circuits (ICs) continue to scale to smaller dimensions, long interconnects have become the dominant contributor to circuit delay and a significant component of power consumption. In order to reduce the length of these interconnects, 3D integration and 3D stacked ICs (3D SICs) are active areas of research in both academia and industry. 3D SICs not only have the potential to reduce average interconnect length and alleviate many of the problems caused by long global interconnects, but they can offer greater design flexibility over 2D ICs, significant reductions in power consumption and footprint in an era of mobile applications, increased on-chip data bandwidth through delay reduction, and improved heterogeneous integration.

Compared to 2D ICs, the manufacture and test of 3D ICs is significantly more complex. Through-silicon vias (TSVs), which constitute the dense vertical interconnects in a die stack, are a source of additional and unique defects not seen before in ICs. At the same time, testing these TSVs, especially before die stacking, is recognized as a major challenge. The testing of a 3D stack is constrained by limited test access, test pin availability, power, and thermal constraints. Therefore, efficient and optimized test architectures are needed to ensure that pre-bond, partial, and complete stack testing are not prohibitively expensive.

Methods of testing TSVs prior to bonding continue to be a difficult problem due to test access and testability issues. Although some built-in self-test (BIST) techniques have been proposed, these techniques have numerous drawbacks that render them impractical. In this dissertation, a low-cost test architecture is introduced to enable pre-bond TSV test through

TSV probing. This has the benefit of not needing large analog test components on the die, which is a significant drawback of many BIST architectures. Coupled with an optimization method described in this dissertation to create parallel test groups for TSVs, test time for pre-bond TSV tests can be significantly reduced. The pre-bond probing methodology is expanded upon to allow for pre-bond scan test as well, to enable both pre-bond TSV and structural test to bring pre-bond known-good-die (KGD) test under a single test paradigm.

The addition of boundary registers on functional TSV paths required for pre-bond probing results in an increase in delay on inter-die functional paths. This cost of test architecture insertion can be a significant drawback, especially considering that one benefit of 3D integration is that critical paths can be partitioned between dies to reduce their delay. This dissertation derives a retiming flow that is used to recover the additional delay added to TSV paths by test cell insertion.

Reducing the cost of test for 3D-SICs is crucial considering that more tests are necessary during 3D-SIC manufacturing. To reduce test cost, the test architecture and test scheduling for the stack must be optimized to reduce test time across all necessary test insertions. This dissertation examines three paradigms for 3D integration - hard dies, firm dies, and soft dies, that give varying degrees of control over 2D test architectures on each die while optimizing the 3D test architecture. Integer linear programming models are developed to provide an optimal 3D test architecture and test schedule for the dies in the 3D stack considering any or all post-bond test insertions. Results show that the ILP models outperform other optimization methods across a range of 3D benchmark circuits.

In summary, this dissertation targets testing and design-for-test (DFT) of 3D SICs. The proposed techniques enable pre-bond TSV and structural test while maintaining a relatively low test cost. Future work will continue to enable testing of 3D SICs to move industry closer to realizing the true potential of 3D integration.

To my fiancée for her love, dedication, and support.

Contents

List of Tables

List of Figures

Acknowledgements

I would like to acknowledge the financial support received from the National Science Foundation and the Semiconductor Research Corporation. I thank Erik Jan Marinnisen for a very fruitful collaboration collaboration over the years. I also thank Sun Kyu Lim and Shreepad Panth for the 3D design benchmarks that they contributed for use in research. I acknowledge the contributions of the past and present students at Duke University, including Mukesh Agrawal, Sergej Deutsch, Hongxia Fang, Sandeep Goel, Yan Luo, Fangming Ye, Mahmut Yilmaz, Zhaobo Zhang, and Yang Zhao. Finally, I would like to thank the members of my committee for for their time and effort in reading this dissertation and attending my defense.

1

Introduction

The semiconductor industry has relentlessly pursued smaller device sizes and low-power chips in a broad range of market segments, ranging from servers to mobile devices. As transistors continue their miniaturization march through smaller technology nodes, the limits of device scaling tend to be reached. Interconnects, particularly global interconnects, are becoming a bottleneck in integrated circuit (IC) design. Since interconnects do not scale as well as transistors, long interconnects are beginning to dominate circuit delay and power consumption.

To overcome the challenges of scaling, the semiconductor industry has recently begun investigating 3D stacked ICs (3D SICs). By designing circuits with more than one active device layer, large 2D circuits can instead be created as 3D circuits with significantly shorter interconnects. 3D SICs will therefore lead to a reduction in the average interconnect length and help obviate the problems caused by long global interconnects [33, 38, 39]. This not only leads to large reductions in latency, but can also lead to lower-power, higher-bandwidth circuits with a higher packing density and smaller footprint. Since dies in a 3D stack can be manufactured separately, there are also benefits to the heterogeneous integration of different technologies into a single 3D stack.

This introduction will serve as an overview of testing and motivate the need for 3D DfT and test optimization. The remainder of this dissertation provides in-depth description, results, theoretical advances, architectures, and optimization methods for 3D circuit test as developed in this dissertation.

Chapter 2 provides a solution for pre-bond TSV test through TSV probing as opposed to built-in-self-test (BIST) or other techniques. A discussion of present probe card technology and the limitations of TSV probing are presented, along with die and probe card architectures for enabling pre-bond parametric TSV test. The feasibility and accuracy of probing are analyzed in detail. An algorithm for reducing pre-bond TSV test time through testing multiple simultaneously TSVs through a single probe needle are also examined.

A variety of optimizations for reducing the overhead and test cost of pre-bond testing are introduced in subsequent chapters. Chapter 3 demonstrates how the architecture presented in Chapter 2 can be reused for pre-bond structural test. A complete analysis of the feasibility, speed, and drawbacks of the method is provided. Chapter 4 shows how novel applications of register retiming and new retiming flows can be used to minimize the delay overhead of the designs in Chapter 2 and 3.

Finally, Chapter 5 presents a unified co-optimization for the 3D test-access-mechanism (TAM) and 2D TAMs to minimize test time through optimal test architecture design and test scheduling. The optimization can account for all possible post-bond test insertions. Furthermore, it covers a variety of 3D-specific test constraints, such as the number of dedicated test TSVs that can be added between any two dies.

This introduction will now continue with a general overview of circuit testing as it pertains to 3D SICs.

1.1 Basics of Testing

Testing of manufactured ICs by applying test stimuli and evaluating test responses is a necessary part of an IC manufacturing flow. This enables defect screening to discard or repair faulty products for low product return rates from customers. Testing of ICs is an expansive topic and we will cover only the basics in this section. To begin, we will look at the four categories of testing: *verification testing*, *manufacturing testing*, *burn-in*, and *incoming inspection* [1]. We will then examine the differences between functional, structural, and parametric tests.

1.1.1 Categories of Testing

Verification testing is applied to a design prior to production. Its purpose is to ensure that the design functions correctly and meets specification. Functional and parametric tests, which will be discussed later, are generally used for characterization. Individual probing of the nets in the IC, scanning electron microscopes, and other methods not commonly performed during other tests, may also be used. During this time, design errors are corrected, specifications are updated given the characteristics of the design, and a production test program is developed.

Manufacturing, or production, test is performed on every chip produced [1]. It is less comprehensive than verification test, designed to ensure that specifications are met by each chip and failing those that do not meet standards. Since every chip must be tested, manufacturing tests aim to keep test costs low, which requires that test time per chip be as small as possible. Manufacturing tests are generally not exhaustive, aiming instead to have high coverage of modeled faults and those defects which are most likely to occur. In the rest of this dissertation, discussion is limited to manufacturing testing.

Even when manufacturing tests are passed and the chips function to specification, some devices will fail quickly under normal use due to aging and latent defects. Burn-in tests,

which are often run at elevated voltages and temperatures, aim to push such devices to failure. This process removes chips that experience *infant mortality*.

Incoming inspection test takes place after devices are shipped and may not always be performed [1]. During incoming inspection, the purchaser of the device tests it once again before incorporating it into a larger design. The details of these tests vary greatly, from application-specific testing of a random sample of devices to tests more comprehensive than manufacturing tests, but the goal is to ensure that the devices function as expected prior to integration into a larger system when test becomes significantly more expensive.

1.1.2 **Functional, Structural, and Parametric Testing**

Functional tests aim to verify that a circuit meets its functional specifications [1]. Generally, functional tests are produced from the functional model of a circuit in which, when in a specific state and given specific inputs, a certain output is expected.

A benefit of functional testing is that the tests themselves can be easy to derive since they do not require knowledge of the low-level design. Since patterns used in verification testing are similar to functional tests, the patterns can be easily converted to functional patterns, which reduces costs in test development. Furthermore, functional tests can detect defects that are difficult to detect using other testing methods.

Despite these benefits, functional testing suffers from serious drawbacks [1]. In order to test every functional mode of a circuit, every possible combination of stimulus must be applied to the primary inputs. Thus, functional testing is prohibitively long unless, as is usually the case, a small subset of possible tests is used. This, however, leads to low defect coverage for functional tests. There is no known general method for efficiently and accurately evaluating the effectiveness of functional test patterns.

While it has its drawbacks, functional test is usually included in product testing along with structural tests. Unlike functional testing, structural tests do not treat the circuit itself as a black box, instead generating patterns based on faults in specific areas of the netlist [1].

There are a number of fault models, the specifics of which are not discussed here, that can be used in generating structural tests. These models allow for the testing of specific critical paths, delay testing, bridging tests, and more. Producing structure-aware patterns leads to high fault coverage and generally reduces test time, especially compared to functional tests. Since specific models are used, structural tests can be more easily evaluated for fault coverage. The drawbacks of structural tests are that gate-level knowledge of the circuit is needed and that structural tests sometimes fail good circuits due to overtesting.

Parametric tests aim to test the characteristics of a device or a part thereof, and are generally technology-dependent [1]. Parametric tests can be separated into two categories — DC test and AC test. DC tests can include leakage tests, output drive current tests, threshold levels, static and dynamic power consumption tests, and the like. AC tests include setup and hold tests, rise and fall time measurements, and similar tests.

1.2 Design for Testability

Given the complexity of today's IC designs, comprehensive testing is impossible without specific hardware support for testing. *Design for Testability* (DFT) refers to those practices that enable testing of VLSI ICs. In this section, we examine DFT techniques that enable the testing of digital logic circuits. Other methods are used for the testing of memory blocks, analog, and mixed-signal circuits, but these will not be discussed.

1.2.1 Scan Test

The vast majority of today's ICs are sequential circuits that rely on flip-flops and clocking to produce functionality. Testing of sequential circuits is very difficult because flip-flops must be initialized, must save states, and tend to have feedback. This greatly impacts both *controllability*, which is the ease with which a system can be placed in a desired state through its primary inputs, and *observability*, which is how easily internal states of the circuit can be propagated to primary outputs. In order to obtain controllability and

observability at flip-flops, scan chains are commonly designed in the circuit.

Scan chains are based on the idea that an IC can be designed to have a specific test mode, separate from functional mode, to which it can be switched. In test mode, groups of flip-flops are connected together to form a shift register called a *scan chain*. In order to do this, each flip-flop is designed as a scan-flop, which multiplexes between a functional input and a test input when in test mode. The test input is either a primary input for the circuit under test (CUT), or the previous scan-flop in the scan chain.

Scan chains enable controllability and observability at every scan-flop in the circuit. Test vectors are scanned into the scan chain from the inputs one bit at a time, with these bits shifted into scan-flops further in the chain. Test responses are then latched, and the response is scanned out through the output of the scan chain. Multiple scan chains can be tested in parallel to reduce test time, though each requires its own input and output in the CUT. There are overheads associated with scan test, including gate, area, and performance overheads, but the benefit to testability makes scan test common in most VLSI ICs.

1.2.2 **Modular Test, Test Wrappers, and Test Access Mechanisms**

A modular test approach to DFT separates a large system-on-chip (SOC), which may be composed of billions of transistors, into many smaller test modules. These modules are often partitioned based on functional groups, ranging from an entire core to analog circuit blocks. These modules may be considered stand-alone test entities from the rest of the modules in an SOC. This partitioning allows tests to be developed for each module independent of other modules in the SOC, which greatly reduces test generation complexity compared to developing tests for the top-level SOC. Furthermore, test reuse is possible if the same module is instantiated multiple times within the same SOC or between multiple IC designs. This also enables the purchase of modules from third-parties for incorporation into an SOC, such that tests for the module are provided and no knowledge of the implementation of the module is needed.

In order to easily test each module, the module must present a standardized test interface to the SOC integrator. It is for this reason that test standards, such as the IEEE 1500 standard [25], were developed. A *test wrapper* is used to provide controllability and observability at the boundary of the module. The test wrapper further enables appropriate test modes for the module and organizes scan chains in the module for test pattern delivery. We will briefly examine the 1500 standard as an example of wrapper design.

The 1500 test wrapper contains a wrapper instruction register (WIR) which can be configured to place the wrapper into specific functional or test modes. This is configured through the wrapper serial interface port (WSP) which contains the wrapper serial input (WSI), the wrapper serial output (WSO), wrapper clock (WRCK), wrapper reset (WRSTN), and other signals. A wrapper boundary register (WBR) is present into which patterns can be shifted for either external test of the logic between modules or internal test through the module's scan chains. A wrapper bypass register (WBY) allows test patterns or test responses to bypass the module on route to other modules in the SOC or to be output at the external SOC pins. Though all tests may be performed using the WSP, many wrappers also contain wrapper parallel in (WPI) and wrapper parallel out (WPO) busses. These consist of two or more bit lines for loading and unloading multiple internal scan chains at the same time. This design reduces test time, but requires more test resources.

In order to route test data from the external test pins to all modules in the SOC, a test access mechanism (TAM) is required. There are many ways to architect a TAM, including multiplexing, daisychain, and distribution designs [46]. Optimization of the TAM and test wrappers to minimize test time using limited test resources has been the subject of much research and will be discussed later in this dissertation.

1.3 3D Integration Technology

A number of 3D integration technologies have been considered, but two main technologies have emerged — *monolithic* integration and *stacking* integration [4]. Although the research presented in the following chapters is based on a stacking approach, in which 2D circuits each with their own active device layer are bonded one on top of the other, we will first briefly examine monolithic technology.

Monolithic integration was proposed as an alternative to stacking, because the mask count and process complexity increases significantly with each stacked die. With monolithic 3D ICS, the processing for the creation of active devices is repeated on a single wafer, resulting in the 3D stacking of transistors. Since the devices and their wiring are processed on a single substrate, the added manufacturing complexities of thinning, alignment, and bonding and the need for through-silicon vias (TSVs) are nonexistent.

Because monolithic integration involves the creation of devices in the substrate above devices that have already been manufactured, significant changes in fabrication processes and technologies would have to take place [5]. The heat currently involved in active device processing is high enough to damage deposited transistors and melt existing wiring. Therefore, advances in low temperature processing technologies are needed. Some recent advances in these technologies [6, 7] have allowed monolithic designs to be realized in the laboratory [5].

Unlike monolithic integration, the active device layers in stacking-based integration are manufactured in separate substrates. Thus, each set of active device layer and associated metal layers are processed on wafers using current fabrication technologies, and substrates are then stacked one on top of the other to produce a 3D circuit. Because no significant changes are required in fabrication technologies, stacking-based integration is more practical than monolithic integration and has therefore been the focus of 3D research [4].

Stacking-based integration can be further separated into three categories based on the

method of 3D stacking — wafer-to-wafer, die-to-wafer, and die-to-die stacking [4]. In wafer-to-wafer stacking, two or more wafers, each with many copies of a circuit, are stacked on top of one another and the resulting 3D stack is then diced to create the individual 3D stacked-ICs (SICs). In die-to-wafer stacking, two wafers are once again produced but one wafer is diced into individual dies, which are then stacked on the other wafer. More dies can be stacked after this process. In die-on-die stacking, wafers are diced into individual dies and then stacked. Die-to-die stacking is desirable as it allows for testing of individual die prior to being introduced to a stack. Aside from being useful for increasing stack yield by discarding bad dies, die-to-die stacking allows for binning of dies to match a stack for performance and power.

Some method must exist in a 3D SIC for dies in the stack to interconnect to one another. A number of methods have been proposed for this interconnection, including wire bonding, microbump, contactless, 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, manufacturing difficulties and insufficient densities limit these methods. TSVs hold the most promise as they have the greatest interconnect density, though they also require more manufacturing steps [38]. We assume stacking-based integration, whether wafer-to-wafer, die-to-wafer, or die-to-die, with TSVs for the rest of this dissertation.

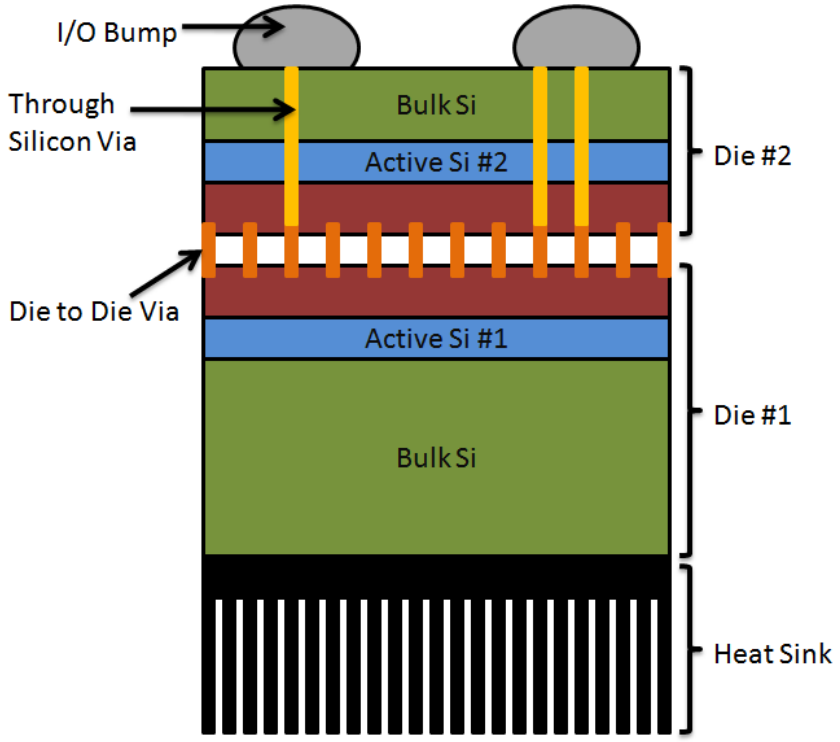


FIGURE 1.1: Example of a face-to-face bonded 3D SIC with two dies.

TSVs are vertical metal interconnects that are processed into a substrate at some point during manufacture. In a front-end-of-the-line (FEOL) approach, TSVs are implanted into the substrate first, followed by active devices and metal layers. In a back-end-of-the-line (BEOL) approach, the active devices are processed first, followed by the TSVs and metal layers [3]. In certain limited stacking approaches, TSVs may also be manufactured later in the process (post-BEOL). This can be done before bonding (vias first) or after bonding (vias last).

In all TSV manufacturing approaches, the TSVs are embedded in the substrate and need to be exposed. TSVs are exposed through a process called “thinning”, in which the substrate is ground away until the TSVs are exposed. This step results in dies that are much thinner than conventional 2D substrates, and are thus quite fragile and are commonly attached to carrier wafers before 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. During bonding, the dies are permanently (current technology does not support “unbonding” of dies) connected to one another, making contact between TSVs. Bonding can be done through a variety of methods, including direct metal-to-metal bonding, direct bonding of silicon oxide, or bonding with adhesives [3]. The processes of alignment and bonding continue until all thinned dies are integrated into the 3D SIC.

There are two different approaches to stacking dies — face-to-face and face-to-back. In face-to-back bonding, the outermost metal layer of one die (the face) is connected to the TSVs on the substrate side (the back) of the other die. Face-to-back allows for many die layers to be bonded in a stack. In face-to-face stacking, the faces of two die are connected to one another. This can reduce the number of TSVs needed for connections, but can support only two dies in a stack unless face-to-back bonding is used for other dies. Though back-to-back bonding is conceivable, this is not a common approach.

To better illustrate 3D integration, Figure 1.1 shows an example of a 3D SIC. This is an example of a stack with two dies bonded face-to-face. Only Die 2 has TSVs to connect to external I/O bumps, and so it is thinned while Die 1 is not. A heat sink is attached to the outside of the stack.

Commercial products using 3D stacks with TSVs are available [8, 9], but are limited to stacked memories. Due to the relative ease of test and repair for memory devices, built-in self-repair and wafer matching techniques can result in significantly high product yields. Faulty stacks that cannot be repaired are discarded. In order to realize the full potential of 3D integration — memories stacked on cores, 3D cores, mixed technology, and the like — testing techniques have to be developed for use during 3D manufacturing.

1.3.1 **3D Testing**

Compared to the testing of 2D ICs, 3D SICs introduce many new challenges for testing. Yield loss for each die in a 3D SIC is compounded during stacking, so stacking of untested die leads to prohibitively low product yields. This motivates the need for *pre-bond* testing, or the testing of dies prior to being bonded to a 3D stack. This allows for the stacking of die that are known to be defect-free and also enables die-matching, so that dies in the same stack can be chosen based on metrics such as speed or power consumption. It is also important to perform *post-bond* tests — testing of either a partial stack to which all dies have yet to be bonded or testing of the complete stack. Post-bond testing ensures that the stack is functioning as intended and that no errors have been made or new defects have been introduced during thinning, alignment, and bonding.

Pre-bond testing of dies offers many design-for-test (DFT) challenges. First, thinned wafers are far more fragile than their non-thinned counterparts, so a small number of contacts must be made during probing and low contact-force probes are a necessity. Because of design partitioning in 3D stacks, a die may contain only partial logic and not completely functional circuits. Currently, this limits the number of tests that can be applied to a circuit with partial logic, though future breakthroughs may make these dies more testable. TSVs also present problems of pre-bond testing because high densities and small sizes make them difficult to probe individually with current technology. Limitations on dedicated test TSVs, oversized test pads for probing and test signals, and the the availability of I/O pins only through one end of the stack make design and optimization tools important for proper test-resource allocation. Such resource constraints are present post-bond as well.

Like pre-bond testing, post-bond testing also presents difficulties not present in 2D IC testing. In order to ensure that no new defects are introduced during stacking, testing of a partial stack is needed. This requires a test architecture and appropriate optimizations to ensure that test time remains small and partial stack tests are possible. Embedded cores

and other parts of the stack may span multiple dies, further complicating test. Few test TSVs are available to the stack, since each additional TSV needed restricts the number of active devices and most TSVs are needed for clocks, power, and other functional signals. Furthermore, limitations in test access are present from few dedicated test pins that provide test access through only one end of the stack.

In [50], a die-level wrapper and associated 3D architecture are presented to allow for all pre-bond and post-bond tests. This approach proposes die-level wrappers and it leverages current standards, IEEE 1149.1 and IEEE 1500. In addition to functional and test modes, die-level wrappers allow bypass of test data to and from higher dies in the stack and reduced test bandwidth during pre-bond tests. While this provides a practical look at test architectures in 3D-SICs — what 3D wrappers and TAMS may be like — it offers no insight into optimization and test scheduling.

Other considerations may also impact the choices made for 3D SIC testing. Thermal constraints may limit which modules on which dies may be testable at any given time, or require low-power pattern generation. The stacking of substrates and oxide layers greatly increases the difficulty of heat dissipation, particularly for die farther from the heat sink. This leads to complications especially during test, when the stack is often heating faster than under functional operation.

TSVs themselves may also impact the operation of surrounding devices. Since a 'keep-out' area around each TSV may have a significant impact on the area overhead of 3D integration, active devices may be placed close to TSVs during layout. The stresses in the semiconductor crystal caused by the presence of a TSV may alter electron and hole mobility near the TSV. Depending on the distance and orientation of transistors near TSVs, this may cause them to run slower or faster than normal. Such considerations are important for test development.

1.3.2 **Die Wrappers for Standard Test Access of 3D Stacked ICs**

In order for a die-level wrapper to be utilized, every die in the stack should be wrapped in accordance with the standard. The die wrapper supports a reduced-bandwidth pre-bond test mode, post-bond testing for both partial and complete stacks, and board-level interconnect testing. The die wrapper is compatible with both the 1500 and JTAG 1149 standards, and is therefore modular in design. In other words, each die, its embedded test modules, TSV inter-die interconnects, and external pins, can all be tested separately. In this way, the scheduling of pre-bond, post-bond, and board-level tests can be flexible depending on the tests needed for a particular manufacturing flow. Although the die wrapper standard is still under development through the IEEE P1838 test work-group, the die wrapper discussed in this section will be referred to as the P1838-style standard wrapper for convenience.

The P1838 wrapper assumes that external connections to the stack are available only on either the top or the bottom of the stack. While it is possible to have I/O connections elsewhere in the stack, for example via wire-bonding, it is likely in the immediate future that I/O will be available only through the bottom of the stack. In this section, it is assumed that this is the case in order to simplify explanations, although by switching references from the bottom of the stack to the top of the stack the reader can understand how the wrapper would be implemented were I/O available through the top of the stack.

The P1838 wrapper is designed to accommodate a variety of pre-bond and post-bond test scenarios. For example, after the fabrication of a die one company may want to perform pre-bond KGD test, including the test of all internal modules, intra-die circuitry, and TSV test (although pre-bond TSV testing is not explicitly supported by the wrapper, Chapter 2 provides a possible solution). The company may then ship the good dies to a second company for integration into a stack, and the second company would like to perform post-bond testing of the partial and complete stack, perhaps retesting the internal modules of each die as well as the TSV interconnects between dies. The die wrapper is further integrated with

the 1149.1 standard for board-level test.

Each die in the stack is assumed to be equipped for scan testing, i.e. scanable digital logic, BIST circuits, etc. The wrapper interfaces with the internal scan chains, test control architectures (such as 2D TAMs), compression circuits, etc. In order to accommodate the wrapper and its functions, the addition of TSVs to the design may be necessary to allow for communication between the dies during test. These dedicated test TSVs are referred to as *test elevators* and will be discussed in more detail later.

Due to the availability of external pins only on the bottom (or top) of the die, all test signals must be routed to and from each die in the stack via dies lower in the stack. In other words, all test control signals and test data must be routed through the bottom die of the stack. When these signals are routed to or from the die for which they are intended without moving further in the stack, it is called a *test turn*. The die wrapper is tier-neutral in that a die equipped with the wrapper may be placed anywhere in a stack. Furthermore, the wrapper does not limit the number of dies that can be in a stack.

Similar to test standards for 2D circuits, the P1838 wrapper is scalable as required by the designer. A one-bit serial TAM is required, with an optional multi-bit parallel TAM. The serial TAM is utilized for debug and diagnosis. Similar to the 1500 serial ports, it provides a low-cost, low-bandwidth method of loading instructions for test configuration and routing test data. The serial ports can be used after the stack is integrated onto a circuit board. The optional parallel TAM provides a method for high-volume production testing. While more costly to implement, it can significantly reduce production test time.

Due to the modularity of the P1838 wrapper, various tests can be performed at different times or not at all—there is no requirement to test the stack as a single entity. All of the possible interconnect tests between dies and the dies themselves are considered as separate test insertions. Each die can be made up of any number of embedded test modules, and these too may be treated as separate entities in regard to test. The benefit of such a modular

approach to test is in ease of integration of IP modules or dies, the ability to optimize test for different fault models depending on the circuit under test, and the freedom in designing and optimizing the best test flow for any given stack.

The Die Wrapper Architecture

Each die in a stack is equipped with a die wrapper, and the wrappers of each die work together to enable test as shown in Figure 1.2. Figure 1.2 provides a conceptual overview of the die wrapper in a three-die stack soldered onto a circuit board, where each die in the stack is wrapped. The pins on the bottom die of the stack provide both functional I/O and, in this example, test inputs. Two types of TSVs exist between each die—FTSVs are functional TSVs, while TTSVs are dedicated test TSVs for use by the die wrappers, which are also referred to as *test elevators* or *TestElevators*. Each die contains some number of test modules that are individually wrapped via the 1500 wrapper, and each die contains its own 2D TAM connecting the internal test modules to one another and the modules to the die wrapper. The modules do not need to be wrapped—they can also include test compression, BIST, or other testable modules. An 1149.1 compliant test-access port (TAP) exists on the bottom die to enable board test.

The die wrapper added to each die in the stack is the additional DfT architecture that makes up the P1838 standard. The wrapper consists of serial I/O for configuring the wrapper test mode and possibly parallel test interfaces as well. Arrows show the movement of test data throughout the stack, and test turns exist on each die to accept data from and return data to the I/O pins on the bottom die. Large, dedicated probe pads are shown to enable a reduced-bandwidth pre-bond test mode. These large pads provide a landing point for probe needles to individually contact TSVs for test application during pre-bond test. Architecture extensions discussed in Chapter 2 and Chapter 3 provide compatible alternatives to these probe pads. The test elevators are used to route test signals up and down the stack from the bottom I/O pins. A 3D TAM, which consists of the test turns, test elevators, and a control

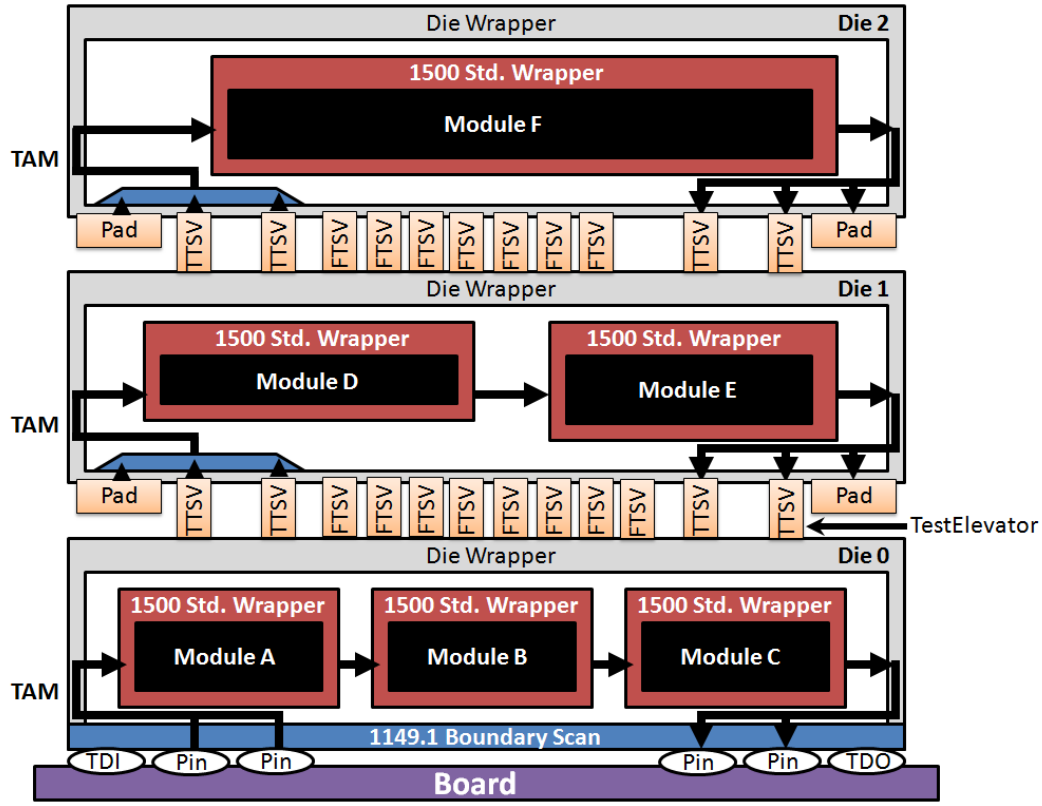


FIGURE 1.2: A conceptual example of a three-die stack using die wrappers.

mechanism for setting the individual die wrapper test modes and optionally the embedded module test modes also exists as part of the die wrappers.

1500-based Die Wrapper

The die wrapper can be designed to interface with either the 1500 or 1149.1 standard. To save space, only the 1500 implementation is discussed. Such an implementation is shown in Figure 1.3 for a three-die stack. Similar to 1500, it can have two test access ports—a mandatory single-bit serial port with a wrapper serial in (WSI) and wrapper serial out (WSO) port used for providing instructions to the wrapper and for low-bandwidth test, and an optional parallel access port for high-bandwidth test. The parallel port can be of arbitrary size, depending on the needs of the designer. The bits of the wrapper instruction register (WIR), combined with the wrapper serial control (WSC) signals, determine the

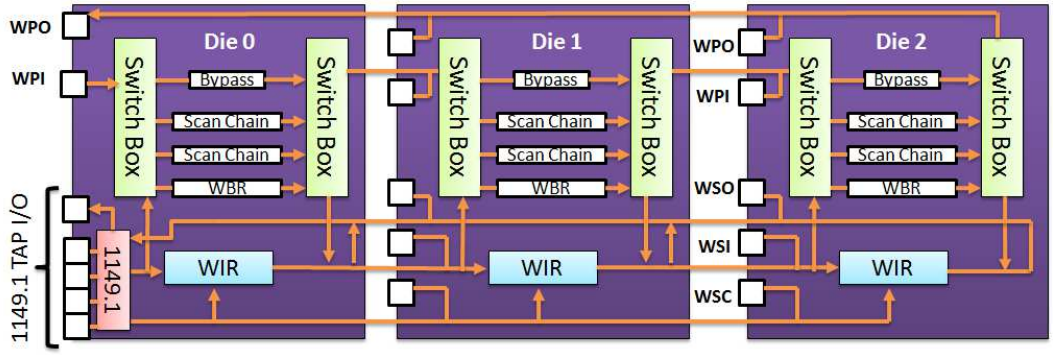


FIGURE 1.3: An example of a three-die stack utilizing 1500-based die wrappers.

mode of operation that the wrapper is in at any given moment. The wrapper boundary register (WBR) is used to apply test patterns and capture responses for both internal tests to the die itself (*intest*) and external tests for circuitry between dies (*extest*), such as TSVs. A bypass path exists to route test data past a die without testing the die or needing to utilize the WBR. Intest, extest, and bypass comprise three of the possible modes of operation in P1838, and are analogous to their 1500 counterparts.

As can be seen in the stack of Figure 1.3, the WSC control signals are broadcast to all the die WIRs. The serial and parallel test buses are daisy-chained through the stack. The highest die in the stack does not utilize test elevators, as there is no die above it. All external I/O pins are on the bottom die, as is an 1149.1 TAP controller to provide for board testing. The serial interface of the die wrapper on the bottom die is connected to the TAP. The only additional pins required for the P1838 wrapper are those for the standard JTAG interface as well as optional parallel ports.

There are four significant features of P1838 that differ from 1500 and are unique to 3D SICs. These features are:

- *Test turns*—Modifications are made to the standard 1500 interface, which exists on the bottom side of each die and is made up of WSC, WSI, WSO, WPI, and WPO. Pipeline registers are added at the output ports (WSO, WPO) to provide an appropriate timing interface between dies or the stack and the board.

- *Probe pads*—All dies on the stack except for the bottom die, which is already equipped with the I/O pin locations for the stack, have oversized probe pads added to some of their back-side TSVs or face-side TSV contacts. These provide a location for current probe technologies to touchdown on the die and individually contact each probe pad. In P1838, these probe pads are required on WSC, WSI, and WSO, which represent the minimal interface for the die wrapper. Probe pads may also be added on any or all of the WPI and WPO pins as necessary. If fewer probe pads are added for pre-bond test than there are parallel access ports for post-bond test, a switch box is utilized to switch between a low-bandwidth pre-bond test mode and a higher-bandwidth post-bond test mode. The state of the switchbox is controlled by the WIR. While the probe pads are currently necessary for testing through the die wrapper in the P1838 standard, previous chapters have discussed methods for providing a test interface that requires few probe pads while providing for pre-bond TSV and scan test.

- *Test elevators*—Additional, dedicated test TSVs for the die wrapper are necessary for routing test data and instructions between dies. These TSVs are referred to as test elevators.

- *Hierarchical WIR*—In the 1500 standard, each embedded test module on each die is equipped with its own 1500-compliant wrapper. In order to provide instructions to these internal wrappers, a hierarchical WIR is necessary. In order to load all WIRs, the WIRs are chained together similarly to scan chains. The length of the WIR chain depends on the number of dies in the stack, the number of 1500-compliant test modules per die, and the summed length of the WIR instructions. In P1838, the die wrapper WIRs are equipped with an extra control bit to bypass the WIRs of the embedded test modules on the die in order to only load die wrapper WIRs.

Figure 1.4 shows the possible operating modes of the P1838 wrapper. To read the diagram, start from either the serial or parallel test mode and follow a path to the end of the

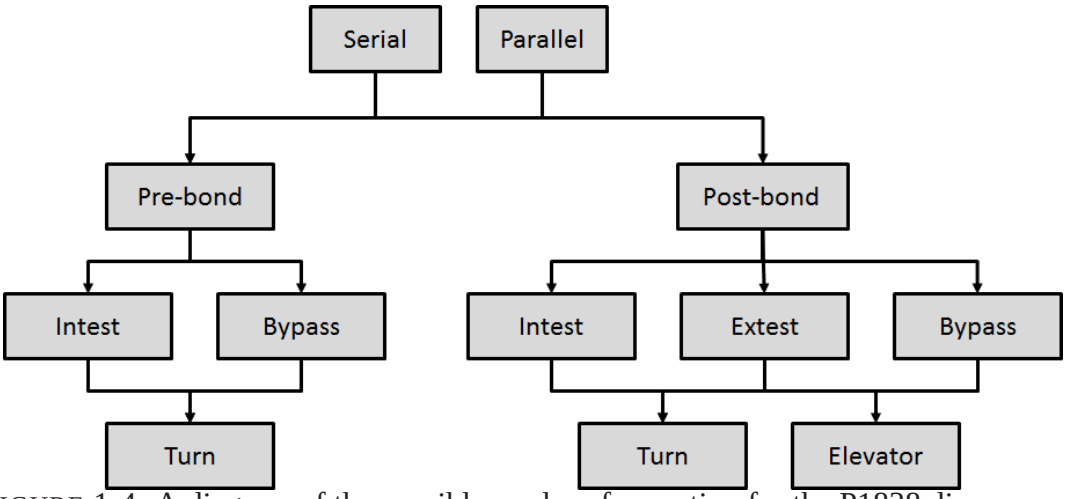


FIGURE 1.4: A diagram of the possible modes of operation for the P1838 die wrapper.

diagram. Each path that can be made is a possible operating mode. For example, several possible modes of operation include *SerialPrebondIntestTurn*, *ParallelPrebondIntestTurn*, *ParallelPostbondExtestTurn*, and *SerialPostbondExtestTurn*. There is a total of 16 operating modes, with four pre-bond modes and twelve post-bond modes.

Each die may be in a different operating mode, depending on what is tested within the stack at any given time. For example, any or all dies may be tested simultaneously. Likewise, any or all interconnect tests between dies can be performed simultaneously. To give an example, consider a four-die stack in which the interconnects between Dies 2 and 3 are tested in parallel with the internal circuitry of Die 4. All of these tests take place utilizing the parallel access ports. In this example, each die except for Dies 2 and 3 will be in a different operating mode. Die 1 will be in *ParallelPostbondBypassElevator* mode, as it is being utilized as a bypass to route test data up and down the stack. Die 2 and Die 3 are placed in *ParallelPostbondExtestElevator* mode, as they are performing their external test on the TSVs between them as well as routing test data further up the stack. Die 4 is in *ParallelPostbondIntestTurn* mode, performing its internal module tests and turning test data back down the stack.

1.3.3 **Conclusion**

This chapter has provided an overview of the testing challenges that must be overcome before 3D SICs can be widely adopted by industry adn the test standards that are currently being developed. Optimization techniques are needed to make the best use of limited resources, both in terms of test access and test scheduling. New methods and breakthroughs are needed to make TSV testing economical and to enable the testing of partial logic. Each chapter from this point on will provide in-depth examination of individual 3D SIC testing topics as well as describe prior work that allows understanding of research advances in an appropriate context.

2

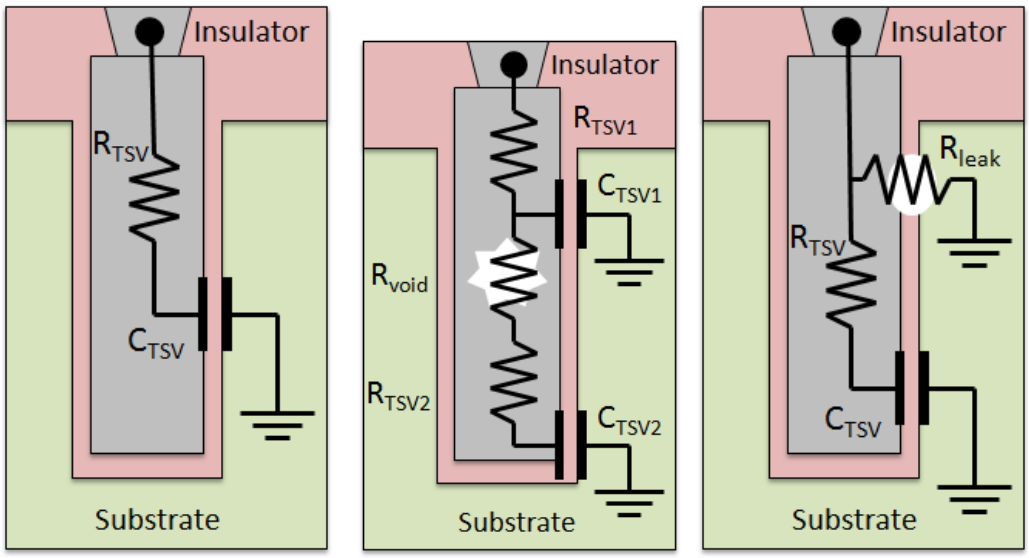
Pre-Bond TSV Test Through TSV Probing

2.1 Introduction

Pre-bond testing of individual dies prior to stacking is crucial for yield assurance in 3D-SICs [60, 58]. A complete known-good-die (KGD) test requires testing of die logic, power and clock networks, and the TSVs that will interconnect dies after bonding in the stack. This chapter, and Chapter 3 after it, will focus on pre-bond TSV test. In this chapter, we explore a probing technique for the rapid, pre-bond parametric test of TSVs.

TSV testing can be separated into two distinct categories—pre-bond and post-bond test [60, 58]. Pre-bond testing allows us to detect defects that are inherent in the manufacture of the TSV itself, such as impurities or voids, while post-bond testing detects faults caused by thinning, alignment, and bonding. Successful pre-bond defect screening can allow defective dies to be discarded before stacking. Because methods to "unbond" die are yet to be realized, even one faulty die will compel us to discard the stacked IC, including all good dies in the stack.

There are further benefits to pre-bond TSV testing beyond discarding faulty dies. Pre-bond testing and diagnosis can facilitate defect localization and repair prior to bonding,



(a) (b) (c)

FIGURE 2.1: Examples with electrical models for (a) a fault-free TSV, (b) a TSV with a void defect, and (c) a TSV with a pinhole defect.

for example in a design that includes spare or redundant TSVs. TSV tests that can be performed after microbump deposition can also test for defects that arise in the microbump or at the joint between the TSV and the microbump. Furthermore, dies can be binned for parameters such as power or operating frequency and then matched during stacking.

TSVs play the role of interconnects, hence there are a number of pre-bond defects that can impact chip functionality [53]. Figure 2.1 provides several example defects and their effect on the electrical characteristics of a TSV. Figure 2.1(a) shows a defect-free TSV modeled as a lumped RC circuit, similar to a wire. The resistance of the TSV depends on its geometry and the material that makes up the TSV pillar. For example, if we assume without loss of generality a TSV made from copper (Cu) with a diffusion barrier of titanium nitride (TiN), the bulk TSV resistance (R_{TSV}) can be determined as follows [35]:

$$R_{TSV} = \frac{4\rho_{Cu}h}{\pi d^2} \| \frac{\rho_{TiN}h}{\pi(d+t_{TiN})t_{TiN}} \tag{2.1}$$

where h is the height of the TSV pillar, d is the diameter of the pillar, t_{TiN} is the thickness of

the TiN diffusion barrier, and ρ_{Cu} and ρ_{TiN} are the resistivity of copper and titanium nitride, respectively. The resistance of the TSV is determined as the parallel resistance of the resistance of the copper pillar and the resistance of the titanium nitride barrier. Generally speaking, the diameter d of the TSV pillar is much larger than the thickness t_{TiN} of the TiN barrier, and ρ_{TiN} is almost three orders of magnitude larger than ρ_{Cu}. Therefore, R_{TSV} can be approximated as:

$$R_{TSV} \approx \frac{4\rho_{Cu}h}{\pi d^2} \tag{2.2}$$

The capacitance of the TSV can be similarly determined from the pillar and insulator materials and geometries. Once again we assume a copper pillar as well as silicon dioxide (SiO_2) for the dielectric insulator. For the TSV shown in Figure 2.1(a), which is embedded in the substrate with insulator on the sides and bottom of the pillar, the bulk TSV capacitance can be calculated as follows [35]:

$$C_{TSV} = \frac{2\pi\varepsilon_{ox}h_c}{\ln\left[\left(d+2t_{ox}\right)/d\right]} + \frac{\pi\varepsilon_{ox}d^2}{4t_{ox}} \tag{2.3}$$

where t_{ox} is the thickness of the insulator and ε_{ox} is the dielectric constant of SiO_2. The bulk capacitance C_{TSV} is the maximum capacitance under low-frequency, high voltage operation. In Equation 2.3, the first term on the right models the parallel plate capacitance formed between the TSV pillar and the sidewall. The second term on the right models the capacitance between the TSV pillar and the bottom disk. If the TSV is embedded in the substrate without insulation along the bottom, or after die thinning which exposes the TSV pillar, the TSV capacitance becomes:

$$C_{TSV} = \frac{2\pi\varepsilon_{ox}h_c}{\ln\left[\left(d+2t_{ox}\right)/d\right]} \tag{2.4}$$

Defects in the TSV pillar or the sidewall insulator alter the electrical characteristics of the TSV. Figure 2.1(b) shows the effect of a microvoid in the TSV pillar. A microvoid is a break in the material of the TSV pillar and can be caused by incomplete fills, stress cracking of the TSV, and other manufacturing complications. These microvoids increase the resistance of the TSV pillar and, depending on the severity of the defect, can manifest as anything from a small-delay defect to a resistive open. In the case of high resistance and open microvoids, the bulk capacitance of the TSV as seen from the circuit end may be reduced as a large portion of the TSV that contributes to the capacitance may be separated from the circuit. This is shown in Figure 2.1(b), as TSV bulk resistance and capacitance are each broken into two separate terms, R_{TSV1} and R_{TSV2}, and C_{TSV1} and C_{TSV2}, respectively.

Figure 2.1(c) shows a pinhole defect of the sidewall insulator. A pinhole defect refers to a hole or irregularity in the insulator around the TSV that results in a resistive short between the TSV and the substrate. Pinhole defects may be caused by impurities trapped in the insulator, incomplete insulator deposition, stress fractures in the insulator, and more. Depending on the severity of the defect, the leakage of the TSV may significantly increase due to the leakage path to the substrate.

Microvoid, pinhole, and other TSV defects that exist before bonding may be exacerbated over time. Electromigration, thermal, and physical stress can lead to early TSV failures. Many defects cause increased power consumption and heating and can increase physical stress before and after bonding, further aggravating early and long-life failures. A thorough pre-bond KGD test should include a burn-in test for TSVs to screen for these types of failures.

Pre-bond TSV testing is difficult due to a variety of factors. Pre-bond test access is severely limited due to TSV pitch and density. Current probe technology using cantilever or vertical probes requires a minimum pitch of 35 μm, but TSVs have pitches of 4.4 μm and spacings of 0.5 μm [69]. Without the introduction of large probe pads onto a TSV or

similar landing pads on the face side of the die [37], current probe technology cannot make contact with individual TSVs. Adding many landing pads is undesirable, as they significantly decrease the pitch and density of TSVs and TSV landing pads. Thus, the number of test I/O available to a tester during pre-bond test is significantly reduced compared to the I/O available during post-bond test. Furthermore, TSVs are single-ended before bonding, meaning that that only one side is connected to die logic and the other side is floating (or tied to the substrate in the case of a deposited TSV without bottom insulation). This complicates TSV testing because standard functional and structural test techniques, for example stuck-at and delay testing, cannot be performed on the TSV.

BIST techniques also suffer from drawbacks. No current BIST architecture for pre-bond TSV test is capable of detecting resistive defects near or at the end of the TSV furthest from the active device layer. This is due to the fact that these resistive defects result in no significant increase or decrease to the TSV capacitance because the bulk of the TSV closest to the test architecture is intact. Furthermore, BIST techniques do not provide an avenue for TSV burn-in tests, and so cannot screen for TSVs that would experience infant mortality failures. The test circuits utilized by BIST techniques, such as voltage dividers or sense amplifiers, cannot be calibrated before hand and are subject to process variation on the die. This can impact the accuracy of parametric tests.

To address the above challenges and offer an alternative to BIST techniques, this chapter presents 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 "TSV network". The proposed approach highlights the relevance of today's emerging test standards and test equipment, and the important role that tester companies can play in 3D SIC testing. Because the proposed method requires probing, it is assumed in this chapter that the die has already been thinned and that it is

supported by a rigid platter (carrier) to prevent mechanical damage during probing. During test, the probe needle must be moved once to allow testing of all TSVs in the chip under test. This method also allows for the concurrent testing of many TSVs to reduce overall test time. Furthermore, significantly fewer probe needles are required to test all TSVs, which reduces the cost and complexity of probe equipment.

2.2 Prior Work on BIST Techniques and their Drawbacks

Although interest in 3D-SIC testing has surged in the past year and a number of test and DFT solutions have been proposed [50, 51, 52, 54, 67, 66], pre-bond TSV testing remains a major challenge. Recent efforts have identified some possible solutions to pre-bond TSV testing. A discussion of TSV defects and several methods for pre- and post-bond testing are presented in [53]. In this work, twelve different TSV defect types are highlighted, five of which can arise post-bond from errors in alignment, bonding, or stress, while the rest involve defects that arise prior to bonding. A number of possible resistance and capacitance tests are shown, including average measurements taken from TSV matrices to tests on TSV chains. Direct leakage tests and AC tests assisted by on-die ring oscillators or phase-lock loops are also described. These methods are however conceptual; specific architectures, implementation details, and thorough characterization have been left for future work.

In [56], an on-chip voltage divider is connected to each TSV, and additional test circuitry is utilized. A sense amplifier is tuned such that only a certain range of voltages on the divider are acceptable, with these values being encoded as a 1 or 0, and these digital values can then be scanned out. During normal operation, the sense amplifier can be used to restore the signals from TSVs that are affected by only minor resistive defects. This method only detects TSV defects that result in large resistances. Capacitive defects and those defects that result in a small increase in resistance (such as small-delay defects) are not detected. Furthermore, the need for tuning and the inherent variability of on-die sense

amplifiers and voltage dividers may make accurate fault detection more difficult.

Another application of on-chip sense amplification was proposed in [59] for detecting capacitive TSV faults. Each TSV is treated as a DRAM cell that is charged and discharged. A tuned sense amplifier determines whether the capacitance on the TSV is within an acceptable range. As with [56], the sensitivity of on-chip circuitry for detecting small capacitive changes is limited. Not only must bounds on TSV capacitance be pre-determined, but errors due to circuit variability must be accounted for in an already sensitive environment.

Two more methods for detecting TSV defects are evaluated in [68]. The first is a leakage current sensor to detect the resistance between the TSV and the substrate. This approach utilizes a comparator connected to the TSV. The second method adds a capacitive bridge to each TSV, using a filtered clock signal to compare the TSV capacitance to a reference capacitance. Alhough more sensitive than sense amplification, this method requires more die area per TSV. It is also sensitive to variability in the reference capacitor, which must have a value in the tens of fF.

The BIST techniques examined in this section help to enable pre-bond TSV test by alleviating test access issues, e.g. limited external test access and probe technology limitations. A significant downside of these BIST techniques is that they suffer from limitations in terms of observability and the measurements that are feasible. Many BIST techniques cannot detect all types of capacitive and resistive TSV faults, and no BIST technique can detect resistive defects toward the far end of the TSV pillar that is embedded in the substrate. Furthermore, BIST techniques require careful calibration and tuning for accurate parametric measurement, but this is often infeasible, a problem exacerbated by BIST circuits themselves being subject to process variation. Furthermore, BIST techniques can occupy a relatively large die area, especially when considering the thousands of TSVs that are predicted per die [60] and that TSV densities of 10,000/mm^2 or more [58] are currently implementable.

In this chapter, we combine capacitance, resistance, leakage, and stuck-at pre-bond TSV tests in a single unified test scheme. Our goal is to minimize the amount of on-die circuits for measurements. We also avoid adding analog components or components that need considerable tuning. We achieve this goal by moving sensing circuitry off-chip, where it can be well characterized prior to use, which also allows us to use larger analog components that are not feasible for on-chip implementation. We further utilize the existing on-die test infrastructure that enables post-bond external test.

2.2.1 **Probe Equipment and the Difficulty of Pre-bond TSV Probing**

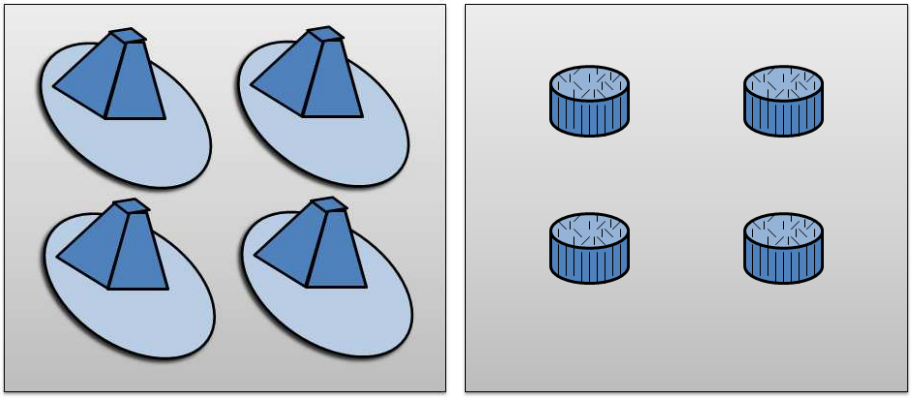
A TSV is a metal pillar that extends into a silicon substrate through the active device layer. A "keep out" area where no active devices may be included is therefore associated with each TSV [58]. Prior to wafer thinning, the TSV is embedded in the substrate and is inaccessible to external probing. During thinning, part of the substrate is removed, thereby exposing the TSVs. There are additional considerations that are important when probing thinned wafers. Due to the fragility of a thinned wafer, it needs to be mounted on a carry platter for testing. Probe cards that use low contact forces may also be required. Furthermore, the probe must not touch down too many times during testing, as this may cause damage to both the TSVs and the wafers. Many devices also lack the buffers needed to drive automated test equipment, particularly through TSVs. Thus, probe cards with active circuitry are necessary; this has been articulated recently as being a focus of research at a major tester company [69].

Although interest in 3D-SIC testing has surged in recent years and a number of test and DFT solutions have been proposed in the literature [50, 51, 52, 54, 67, 66], pre-bond TSV testing remains a major challenge. Recent efforts have identified some possible solutions for pre-bond TSV testing. A discussion of TSV defects and several methods for pre- and post-bond testing are presented in [53] and [58]. In [53], twelve different TSV defect types are highlighted, five of which can arise post-bond from errors in alignment, bonding, or

stress, while the rest involve defects that arise prior to bonding. Thus, many defects can be targeted at a pre-bond stage. For example, a microvoid in the TSV increases the resistance of the TSV, while a pinhole defect causes leakage between the TSV and the substrate, thus increasing the TSV capacitance. Most of the pre-bond TSV defect types are resistive in nature [53].

In pre-bond probing of TSVs, surface planarity of the TSVs or microbumps impacts the consistency of contact between the probe needles and the TSVs. Therefore, "spring-loaded" probe technologies can facilitate pre-bond probing of TSVs by providing varying degrees of individual control of contacts involving probe needles. Much attention has been devoted to the manufacturing of spring-loaded probe needles [99, 77, 100, 101], as surface non-planarity and other issues have made this a desirable technology for achieving good contact during wafer probing. Proposed techniques include membrane probe cards, thermally-actuated probe needles, and probe needles with electrostatic actuators. Furthermore, because non-planarity also impacts TSV connections made during bonding, recent research has explored the planarization of microbumps [102]. This approach can also reduce non-planarity when it is used for the testing of TSVs with microbumps.

It is also important to examine the contacts and contact resistances that may be expected between a probe card and a TSV/microbump. In [97], low-force contacts were made between probe needles and microbumps, and a worst-case contact resistance of 13 Ω was obtained. This is well within a reasonable range to achieve accurate parametric measurements for TSVs. In [74], the effect of probe needle wear on contact resistance is reported. The length of the needle decreased over time as material was worn away by touchdowns, adversely affecting contact force and the quality of contact. However, the reported results show that even with extensive wear after many touchdowns, contact resistances remained below 3 Ω at 30°C for certain needle heads, with worst-case resistance being no higher than 40 Ω. If these findings are extended to the problem of contacting a TSV network, in



(a) (b)

FIGURE 2.2: Example of the (a) pyramid probe card and (b) NanoPierce^{TM} probe card for pre-bond TSV probing.

which contact quality will be worse for some TSVs, similar changes in contact resistance can be expected.

Novel probe card architectures to enable pre-bond TSV probing are currently being examined in the literature. Cascade Microtech Inc. has introduced a pyramid probe card that has been demonstrated at a 40 μm array pitch [97]. An illustrative example of this probe card is shown in Figure 2.2(a) with four probe needles. The needles are deposited on a membrane substrate to allow for individual actuation to compensate for surface non-planarity between TSVs. The needles themselves present a flat square probe head. Form Factor Inc. has introduced the NanoPierce^{TM} contact head for 3D TSV probing, similarly demonstrated at a 40 μm array pitch [98]. An illustrative example of this probe card is shown in Figure 2.2(b). The probe needles are grown from many dense nanofibers that act together to make a contact. Both probe cards utilize low-force probing and show minimal microbump damage.

Despite these new advances in probe card technology, the demonstrated pitches and array placement of needles limit TSV placement and density if individual contact with each TSV required. Furthermore, scaling these probe cards is yet to be demonstrated, and it appears that microbumps may scale faster than probe technology. For example,

in [103] microbumps have already been manufactured at sizes of 5 μm with a 10 μm pitch, compared to the 40 μm pitch in [97, 98]. Furthermore, even if every TSV on a die can be contacted individually, the issue of routing test data to and from a probe card with the thousands of probe needles required to contact every TSV is likely prohibitive. If the number of probe needles is instead reduced to a more manageable number, then many touchdowns may be needed to test every TSV on a die, significantly increasing test time and the likelihood of damaging the thinned die or wafer. The probing technique introduced in this chapter ensures that probe technology, regardless of size and pitch, will be capable of testing TSVs with or without microbumps.

The rest of this chapter examines a novel combination of on-die architectures, pin electronic architectures, and test optimizations to enable fast pre-bond parametric TSV test. Section 2.3 introduces a cutting-edge test architecture for pre-bond probing and testing of TSVs. It discusses the benefits and drawbacks of TSV probing and demonstrates the effectiveness of the pre-bond probing architecture. Section 2.4 presents a heuristic approach to reducing pre-bond TSV test time by organizing TSVs into test groups in which multiple TSVs within the same TSV network are tested simultaneously. A examination of the considerable reduction in test time possible through parallel TSV test is provided. Finally, Section 2.5 concludes this chapter.

2.3 Pre-bond TSV Testing

In the pre-bond test method discussed in this chapter, a number of TSVs are shorted together through contact with a probe needle to form a network of TSVs. The network capacitance can be tested through an active driver in the probe needle itself, and then the resistance of each TSV can be determined by asserting each TSV onto the shorted net. More details on TSV testing are given in Subsection 2.3.1. In this section, the new test architectures used for pre-bond TSV testing will be discussed.

For post-bond external tests, a 1500-style die wrapper with scan-based TSV tests has been proposed [50]. It is assumed in this section that die-level wrappers are present. To enable pre-bond TSV probing, the standard scan flops that make up the die boundary registers between die logic and TSVs are modified to be gated scan flops (GSFs), as shown in Figure 2.3. Alternative methods to accessing a TSV on a die have been reported as in [62], but these methods are not compatible with die wrappers, and so are not considered in this chapter as die wrappers are likely to be utilized in future 3D stacks. As seen at the block level in Figure 2.3(a), the gated scan flop accepts either a functional input or a test input from the scan chain; the selection is made depending on operational mode. A new signal, namely the "open signal", is added; it determines whether the output Q floats or takes the value stored in the flip-flop.

In the GSF design, shown at gate-level in Figure 2.3(b) and at transistor-level in Figure 2.3(c), two cross-coupled inverters are used to store data. Transmission gates are inserted between the cross-coupled inverters and at the input (D) and output (Q) of the flop itself. The widths of the transistors in the first cross-coupled inverter stage are greater than the widths of those in the second cross-coupled inverter stage such that the second stage takes the value of the first stage when the buffer between them is open and they are in contention. An internal inverter buffer is added before the output transmission gate such that the gated scan flop can drive a large capacitance on its output net without altering the value held in the flop. The "open" signal controls the final transmission gate.

It is important to distinguish between sending and receiving TSVs and their impact on test circuitry. Sending TSVs are TSVs that are being driven by logic prior to bonding, whereas receiving TSVs are floating before bonding and meant to drive logic on their associated die. In both cases, the GSF can be utilized. For the testing of sending TSVs, the GSF is used to drive the TSV during probing. In functional mode, the gate remains in its low-impedance state. In the case of receiving TSVs, the GSF also drives the TSV during

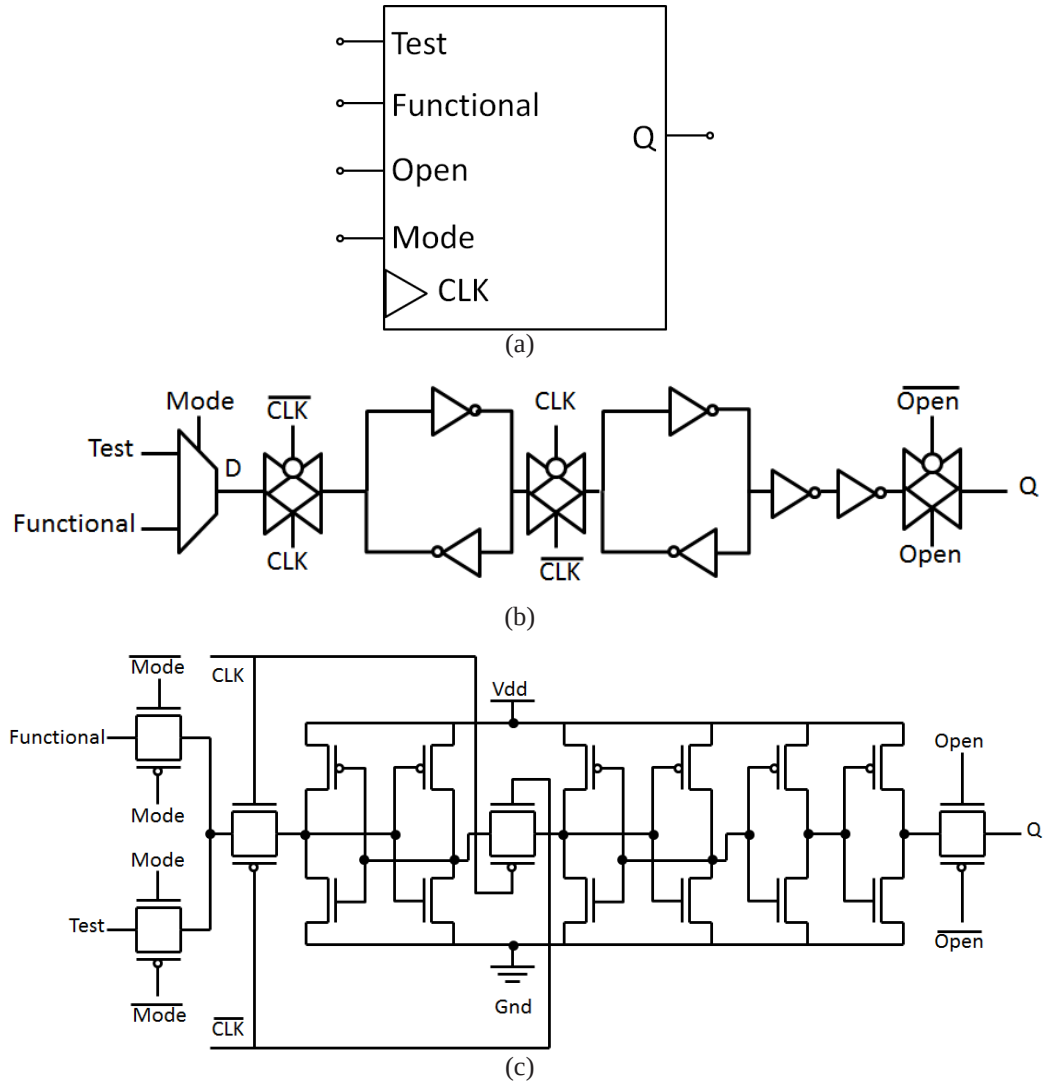


FIGURE 2.3: Design of a gated scan flop: (a) block-level; (b) gate-level; (c) transistor-level.

test. However, in functional mode, the gate remains in its high-impedance state because the TSV will be driven by logic on another die. The functional output of the GSF associated with receiving TSVs, if needed, can be connected to node “f” as denoted in Figure 2.3(b).

A controller is needed for determining which gates are open in the TSV network at any given time. One such controller may be a centralized gate controller that is routed through a decoder to control gates in each TSV network simultaneously as shown in Figure 2.4. Because each network is contacted by a separate probe needle, the TSVs in one network

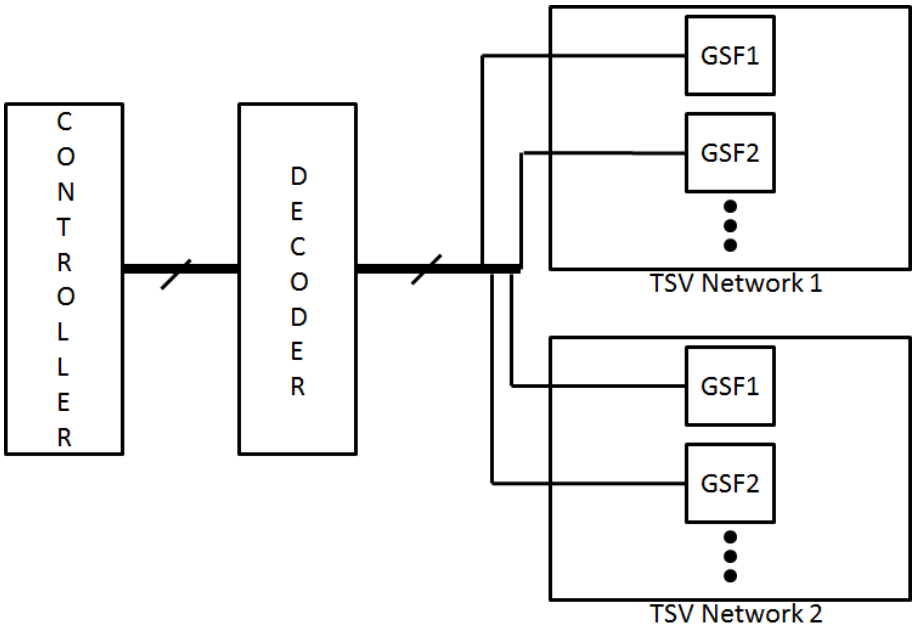


FIGURE 2.4: Overview of an example control architecture.

can be tested in parallel with the TSVs in each other network. Each TSV is driven by its own GSF.

For the example controller shown in Figure 2.5, a synchronous up counter based on J/K flip-flops is used that can also be utilized as a shift register. The example controller includes four bits; it only needs $\log_2(n)$ bits, where n is the number of gated scan-flops in the largest TSV network during test. During normal operation, the controller counts up, cycling through the gated scan-flops in each network for test. If a specific TSV must be tested or special test codes must be sent to the decoder, then the appropriate data can be shifted into the controller. A limitation of using a central controller is that outputs from the decoder must be routed to each TSV network. However, because it is only necessary to have as many wires leaving the decoder as there are TSVs in the largest network, routing can be greatly simplified, especially when compared to BIST techniques.

To determine the capacitance of the TSV network and the resistance of each TSV, the probe needle must be equipped with an active driver and a method of detection. In order to keep this circuitry simple, a design such as the one shown in Figure 2.6 can be used.

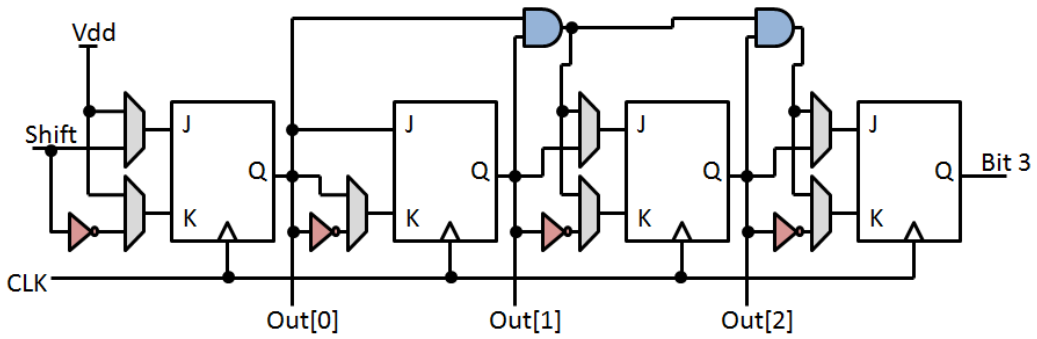


FIGURE 2.5: A shift counter.

This design consists of a DC source with a voltage on the order of the circuit under test. A switch, $S2$, is used to connect or disconnect the source from a capacitor (C_{charge}) of known capacitance. The voltage across the capacitor is continuously monitored through a voltmeter. A second switch, $S1$, allows the capacitor to be connected or disconnected from the probe needle itself.

It should be noted that the above charge sharing circuit facilitates design and analysis in HSPICE. In practice, this circuit can be prone to measurement error caused by leakage currents. Therefore, an AC capacitance measurement method can be used to mitigate the effects of leakage, for example with a capacitive bridge [70].

While digital testers are usually not equipped with the drivers and sensors needed to measure capacitance, analog and mixed-signal testers are known to have these capabilities [71]. Because pre-bond TSV testing requires accurate analog measurements (digital measurements are not feasible unless more complete functionality and I/O interfaces are available), it is necessary to either add capacitance-sensing circuits and drivers to digital testers or to utilize analog testers for pre-bond TSV defect screening.

In order to contact all TSV matrices, the probe card has to be moved at least once. In order to reduce the number of times the probe card must be moved (and ensure a one-time-only movement), a design such as that shown in Figure 2.7 can be utilized. By offsetting the probe needles as shown, the probe card has to be shifted up or down only once in order to

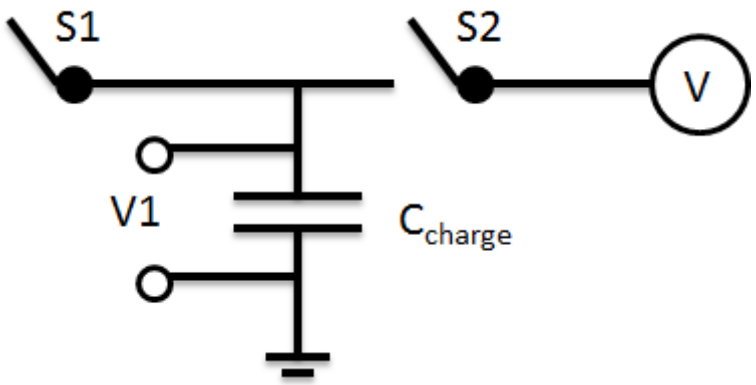
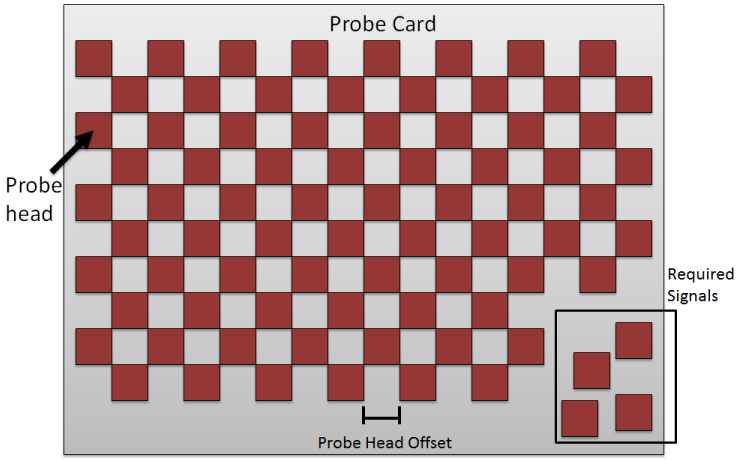


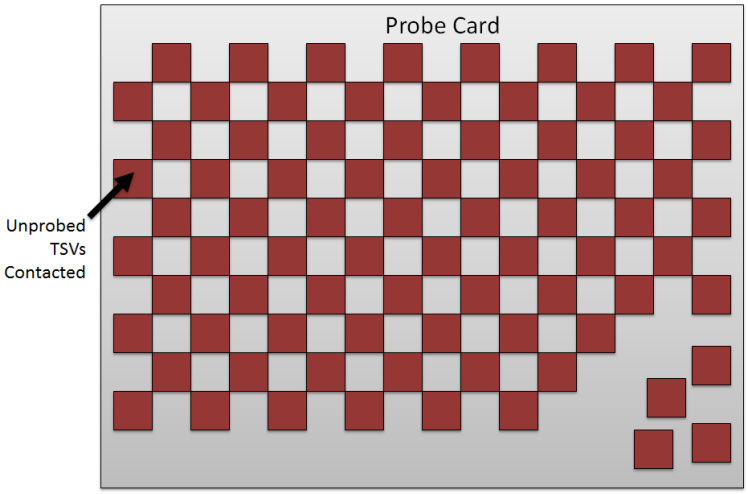
FIGURE 2.6: A charge-sharing circuit.

contact all the TSV networks. In Configuration 1, probe needles on the probe card contact some groups of TSVs, with adjacent TSV networks lacking a probe needle to contact. Once the probe card is moved to Configuration 2, the probe needles contact the previously untested TSVs. Contacting and supplying critical signals such as power and clocks to the die during test may require special probe needles to be added to the probe card; these are shown as required signals in Figure 2.7 and are not placed in the same configuration as the rest of the probe needles. It is assumed that these will be individually contacting TSVs with large probe pads added to them.

In order to illustrate how contact is made with all TSV networks on a die, Figure 2.8 shows a partial example of two rows of probe needles above a die with TSVs. The TSVs are spaced in an irregular manner in this example and it is assumed but not necessary that the TSVs have microbumps. Figure 2.8(a) shows the initial configuration of the probe card and probe needles. In Figure 2.8(b), the probe card is lowered and contact is made between the probe needles and the highlighted TSVs. Each group of TSVs contacted by one of the probe needles comprises a TSV network. The probe card is then lifted and shifted to its second configuration as in Figure 2.8(c), contacting the newly highlighted TSVs. As shown in Figure 2.8(d), a row of TSVs can be completely contacted with a single movement of the probe card. The probe card design attempts to limit the number of contacts the probe needle



(a) Configuration 1



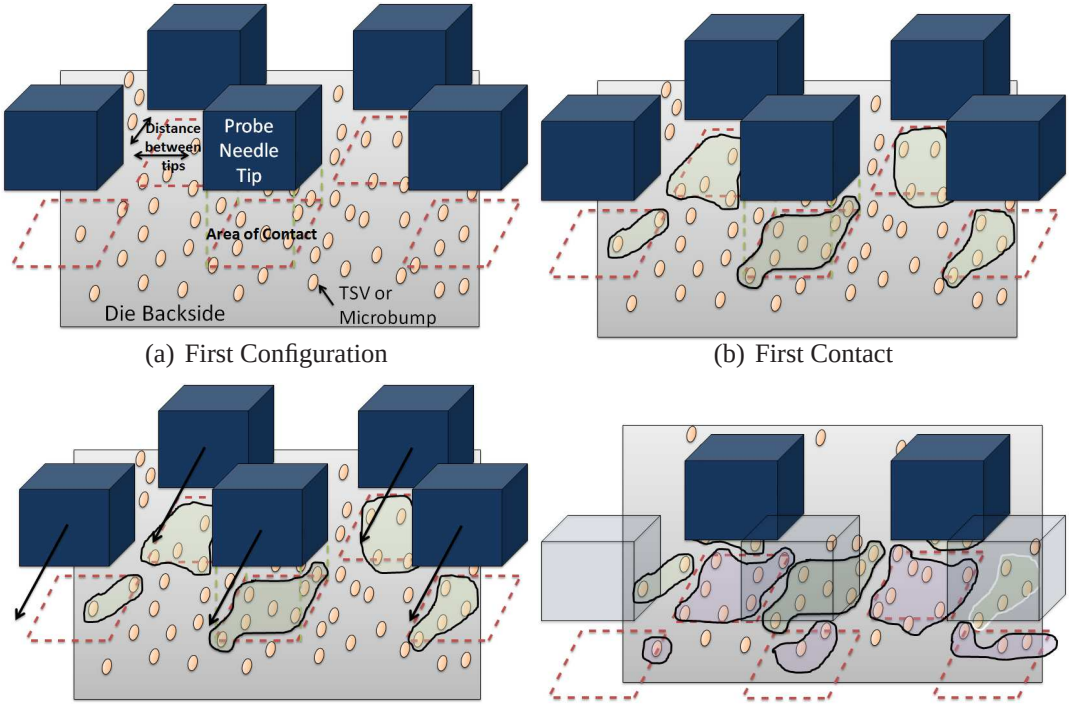
(b) Configuration 2

FIGURE 2.7: Two configurations of a probe card for TSV testing.

makes with each TSV to minimize damage that may occur during test, such as scrubbing. To prevent a single TSV from being contacted more than once, or in more than one TSV network during test, additional control signals can be included in the controller to close the gates for all TSVs tested in the first test period during the second test period, and vice versa.

2.3.1 **Parametric TSV Testing via Probing TSV Networks**

A TSV can be modeled as a wire with both a resistance and a capacitance. While a TSV may be manufactured from a number of different materials, copper is often used for metal



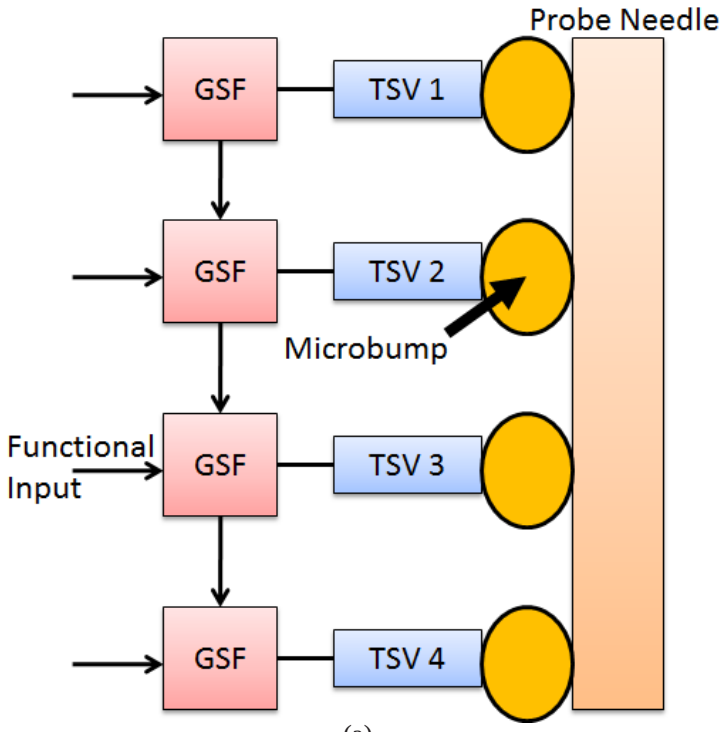
(a) First Configuration (b) First Contact

(c) Shift (d) Second Contact

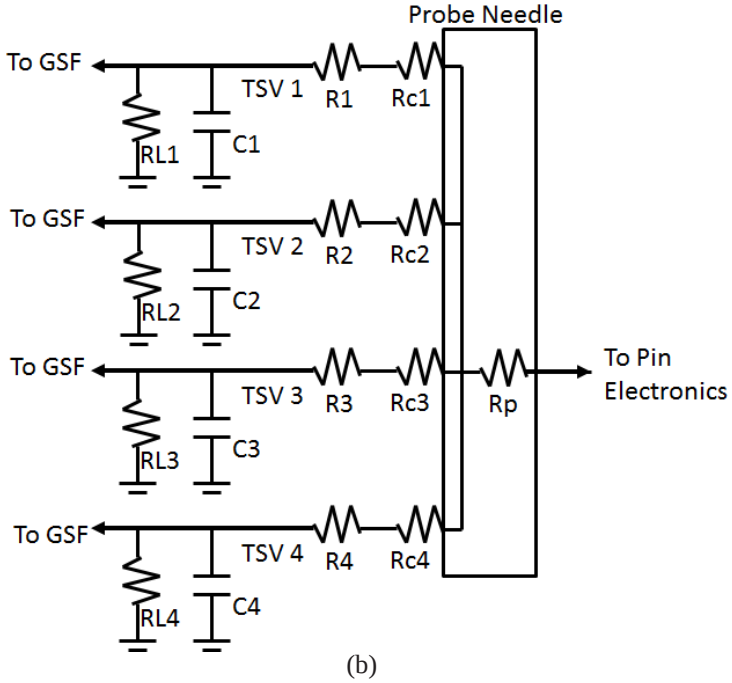
FIGURE 2.8: Example to illustrate the probing of TSV networks.

layers and polysilicon may be a non-metal alternative. The resistance of a TSV made from copper with a 2-5 μm diameter and 5 μm height is 80-200 mΩ. For a polysilicon TSV with a 28-46 μm diameter and 50 μm height, the resistance is 1.3-5.0 Ω [73]. The capacitance of a copper TSV with a 1-10 μm diameter and 30-100 μm height is 10-200 fF [35].

A probe needle makes contact with a number of TSVs at a time, as seen in Figure 2.9(a). The TSVs are connected to gated scan-flops, which are connected to form a scan chain. This circuit is modeled as seen in Figure 2.9(b). The probe needle has a known resistance R_p and a contact resistance (Rc_1-Rc_4) with each TSV. The contact resistance depends on the force with which the probe needle contacts each TSV, and may differ per TSV. Each TSV has an associated resistance (R_1-R_4) and capacitance (C_1-C_4). Furthermore, a leakage path modeled by a resistance (RL_1-RL_4) exists between each TSV and the substrate. The value of interest is the net capacitance C_{net}, which is the combined capacitance of all of the TSVs in parallel. C_{net} can be expressed as:



(a)



(b)

FIGURE 2.9: TSV network model.

$$C_{net} = C_1 + C2 + \cdots + C_n.$$

The net resistance, R_{net} is then the equivalent of the probe resistance, contact resistances, and TSV resistances, calculated as follows:

$$R_{net} = R_p + \left(\frac{1}{R_1 + R_c} + \frac{1}{R_2 + R_c} + \cdots + \frac{1}{R_n + R_c} \right)^{-1}.$$

The net leakage RL_{net} is simply all leakage resistances added in parallel.

Capacitance Measurements

The net capacitance must first be determined to characterize each TSV. From this measurement the capacitance of each TSV can be estimated and their respective resistances measured. The charge-sharing circuit of Figure 2.6 is connected to the probe needle which shorts together multiple TSVs as shown in Figure 2.10. There are three steps involved in measuring the net capacitance:

- Discharge the TSV network by loading a 0 into all gated scan-flops and then opening their gates. During this step, switch $S1$ is open. The charge sharing circuit is disconnected from the TSV network, and switch $S2$ is closed in order to charge capacitor C_{charge} to a known voltage V.

- Close all gated scan-flops and open switch $S2$. Close switch $S1$ to connect capacitances C_{charge} and C_{net}. This sets up the charge-sharing network as C_{charge} is discharged into C_{net}.

- Monitor the rate of change of $V1$ through the volt meter until it falls below a certain level. This level corresponds to the rate of change in a simulated charge curve that has reached 1% of its maximum charge during discharge. Once this rate is reached, then a final measurement of voltage $V1$ across capacitor C_{charge} is taken.

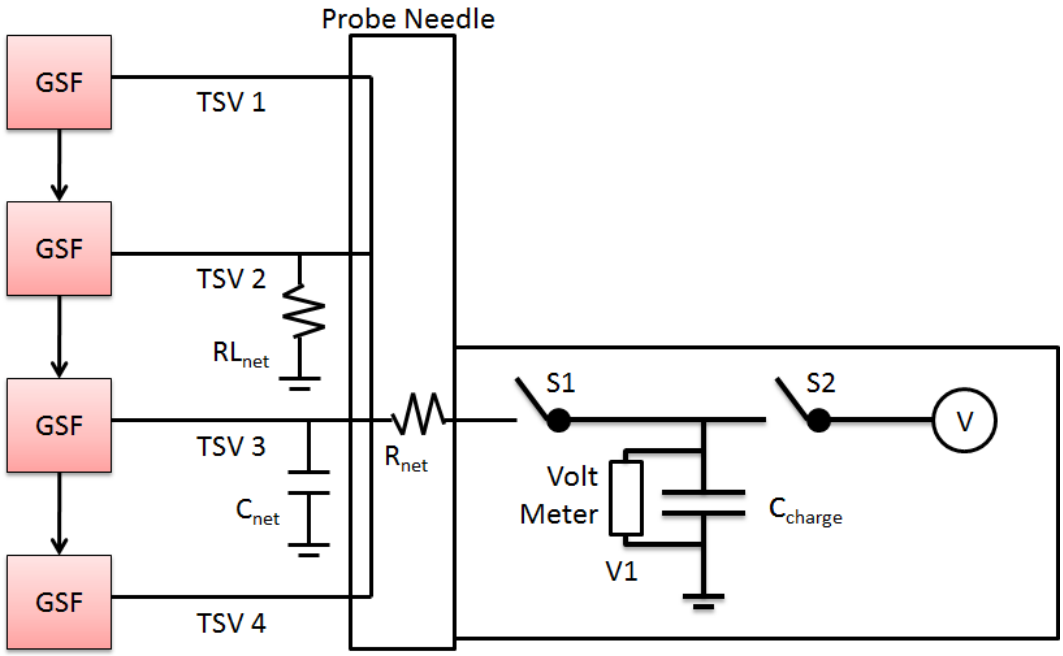


FIGURE 2.10: TSV network with charge-sharing circuit.

Once these steps are completed, the value of C_{net} can be determined from known values using the following charge-sharing equation:

$$C_{net} = C_{charge} \cdot \frac{(V - V1)}{V1} \tag{2.5}$$

From the network capacitance, the average capacitance of each TSV can be determined by dividing the network capacitance by the number of TSVs in the network. In this respect, the presence of fewer TSVs in the network will allow for a higher resolution in capacitance measurements, although this is not the case for resistance measurements or stuck-at/leakage tests (described below). Among the TSV defect types described in [53], only one of the pre-bond-testable defects results in capacitance changes (as opposed to resistance changes). This is the pinhole defect, which may also be detected through leakage tests. Although capacitance measurement using this method yields only an average value, significant increases in capacitance can be readily detectable if the number of TSVs in a network is not too large.

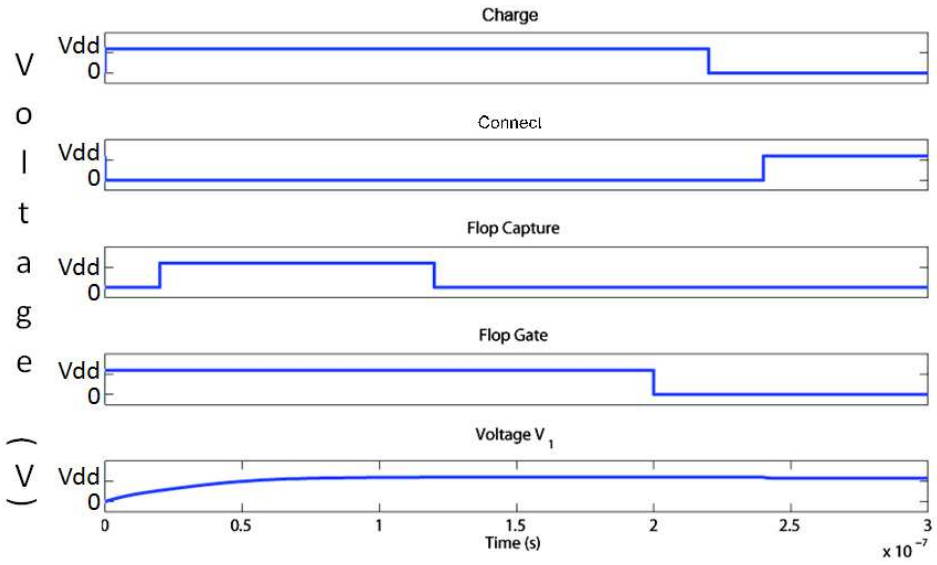


FIGURE 2.11: The process of net capacitance measurement.

Resistance Measurements

The bulk of TSV defects that can be tested pre-bond result in increased TSV resistance. For this reason, it is important that a pre-bond test be capable of accurately measuring TSV resistance. In order to measure resistance, the charge-sharing circuit of Figure 2.6 is once again utilized. The capacitor C_{charge} will be charged through each TSV, and the time needed to charge the capacitor to a chosen voltage (for example, 99% of V_{dd}) is recorded. Long charge times increase the resolution in resistance measurement, but they lead to higher test time. As a tradeoff, smaller voltage levels (such as 90% of V_{dd}) can be used to reduce test times if the resolution is acceptable—see Table 2.1 in Subsection 2.3.2 for more information.

The above measurement can be carried out by recording the start time when the control signal is asserted for the TSV under test to open, then the end time can be measured when $V1$ reaches the desired voltage. In order for resistance to be measured, the probing device must first be calibrated using a non-faulty TSV in a TSV network. This calibration can be

done off-chip prior to testing any circuits, for example by using a dummy silicon chip with double-ended TSVs in which the TSVs themselves can be completely characterized. One or more TSVs on this calibration chip can be used to calibrate the equipment. The charge time of C_{charge} in this environment is determined, and charging times on the test floor are then viewed in relation to the calibrated time.

The test begins by loading all of the gated scan-flops with 1 and discharging the TSV network using the probe. Switch $S2$ is opened and switch $S1$ is closed such that the capacitor C_{charge} is discharged as well. One of the gated scan-flops is then opened, allowing the scan-flop to charge C_{charge} through its connected TSV. When $V1$ reaches the pre-determined voltage, the time to charge C_{charge} is recorded. It is then compared to a calibrated charge curve for a non-faulty TSV. This process of charging and discharging continues for each TSV, which can be completed quickly by incrementing the controlling counter to open each subsequent TSV.

Leakage Tests

Leakage tests are an averaged measurement per TSV, similar to capacitance tests described earlier. In order to perform a leakage test, C_{charge} is disconnected from the TSV network and the network capacitance C_{net} is charged through the GSFs to V_{dd}. Next, all gates are switched to their high impedance state and the TSV network is left floating for a chosen period of time. After this period, a voltage measurement of the network is taken through the probe, and the change in voltage over the floating time period is determined. This is then compared to a calibrated curve to determine the leakage of the network.

Stuck-at Tests

Stuck-at and leakage tests, in which leakage is high enough to be similar to stuck-at faults, can be performed together and in parallel under this scheme. For strong stuck-at 0 faults or leakage with low resistances to ground, the TSV network can be charged with the gated

scan-flops closed and its voltage measured. If the rate of discharge is abnormally high, it can be inferred that a stuck-at 0 fault or a leakage-inducing defect exists on at least one of the TSVs. A parallel stuck-at 1 test can be performed by discharging the TSV network with the gated scan-flops closed and measuring the voltage on the net.

Individual stuck-at tests can also be performed quickly. This is done by loading the scan-chain with a pattern of alternating ones and zeros. The value on the gated scan-flop on the first flop in the control sequence for the TSV network determines whether the net is first charged or discharged. Then, each GSF is opened in sequence, making alternating assertions of high or low. The pattern is then shifted by one and the process is repeated.

2.3.2 **Simulation Results for Pre-bond Probing**

Simulation results are presented for a TSV network of 20 TSVs modeled in HSPICE. The number 20 was determined based on the relative diameter and pitch of probe leads and TSVs. Unless otherwise stated, the resistance of each TSV and contact resistance is 1 Ω and the TSV's associated capacitance is 20 fF. These numbers are based on data reported in the literature [73, 35]. The probe needle resistance is set at 10 Ω. This value is several Ω higher than contact resistances seen with probe cards today [74, 75] to account for the low contact force needed and unusually small features probed in our scheme. TSV leakage resistances for non-faulty TSVs were 1.2 TΩ, corresponding to a leakage of 1 pA at a 1.2 V V_{dd} [104]. The transistors were modeled using predictive low-power 45 nm models [76]. Transmission-gate transistor widths were set to 540 nm for PMOS and 360 nm for NMOS. These larger widths were chosen such that the gate, when open, would have little impact on signal strength. A strong and a weak inverter were used, with the strong inverter having widths of 270 nm for PMOS and 180 nm for NMOS, and the weak inverter having 135 nm for PMOS and 90 nm for NMOS. These were chosen such that the majority of transistor W/L ratios were 2/1 for NMOS and 3/1 for PMOS. The charge-sharing capacitor C_{charge} was modeled at 10 pF, chosen to be an order of magnitude larger than the fault-free

capacitance of the TSV network. This is sufficiently large to achieve good resolution in measurement without being so large that charge times are unreasonable or that leakage becomes a significant issue. The power supply voltage V_{dd} for both the probe electronics and the circuit under test was set at 1.2 V.

Inductance is not included in the model for two reasons. First, modern probe cards have little parasitic inductance on the probe tips [77]. Second, sampling takes place in the pin electronics and not through the TSV network itself, so pin electronics are the limiting factor for high-speed sampling and not the TSV network or its contact with the probe needle. Probes capable of GHz sampling frequencies have been available for some time [78].

Figure 2.11 demonstrates the process of net capacitance measurement, with high signals meaning that a gate is open or a switch is closed. To begin, switch S2 is closed, charging C_{charge} to V. During this time, the gated scan-flops capture a 1 (the Flop Capture signal captures on the falling edge). The flop gates are open as denoted by the Flop Gate signal. The flop gates are then closed, S2 is opened, and switch S1 is closed to begin charge-sharing. C_{charge} then begins discharging, and the voltage is measured after 250 ns when it has settled to 1.15 V. Using Equation (2.5) and the subsequent division step, each TSV capacitance can be determined to be 20.25 fF, very close to the actual value of 20 fF.

The determination of C_{net} is a robust measurement. Because the charge sharing system set up between C_{charge} and C_{net} is allowed to settle before measurements are taken, TSV and contact resistance do not affect the result. Only relatively high leakage currents prevent measurements, because the change in voltage of C_{charge} will remain large until the capacitor is discharged. For example, a 100-point Monte Carlo simulation was conducted assuming a Gaussian process variation with a 3-σ value of 20% around nominal TSV and leakage resistance values. TSV capacitances ranged from 10 fF to 50 fF, with a total net capacitance of 550 fF. In every simulation, C_{net} was calculated to be 569 fF regardless of process variation.

Figure 2.12 shows the charging behavior of capacitor C_{charge} through one TSV in the TSV network. The TSV resistance varies from 1 Ω to 3000 Ω in 500 Ω intervals. V_1 is recorded when the voltage across C_{charge} has reached 99% of V_{dd}, or 1.19 V. As shown, small variations in resistance alter the charge curve in measurable ways. Figure 2.13 shows the charge time to reach this voltage level for each TSV resistance and for one, two, or three TSVs under test in parallel. As can be seen, there is a linear relationship between capacitor charge time and the resistance of the TSV under test. For the capacitance value of 10 pF, each 500 Ω increment increase in TSV resistance results in about a 20 ns increase in charge time when considering only one TSV under test. Assuming a sample rate of 1 GHz and calibration at 1 Ω (the first waveform), a resolution r of about 25 Ω is achieved. In other words, each increase in charge time of 1 ns above the calibrated charge time corresponds to a 25 Ω increase in resistance on the TSV under test. In this scheme, smaller values of r preferable. Higher resolutions can be achieved at the cost of longer charge times by increasing the capacitance of C_{charge}. However, if the capacitance is too large then leakage could become a significant source of error. Generally, the resolution of measurement can be determined using the formula $\frac{\Delta\Omega}{S \cdot \Delta T}$ where ΔT is a change in charge time, $\Delta\Omega$ is the TSV resistance difference for that charge time, and S is the sample rate.

Table 2.1 shows the resolution of TSV resistance measurements at 500 MHz and 1 GHz sample rates for different chosen voltage levels, assuming a fault-free TSV with 1 Ω resistance and a faulty TSV with 500 Ω resistance. For example, the resolution achieved when charging C_{charge} to 99% of V_{dd} implies that resistances as small as 24.3 Ω above the nominal 1 Ω fault-free resistance can be detected. The lower the entries in the second column of Table 2.1, the larger the resolution and detectability of TSV defects. It can be seen that as the voltage level to which C_{charge} is charged decreases, the resolution achievable by resistance measurements also decreases.

Table 2.2 shows the calculated TSV resistance values of several faulty TSVs using the

Table 2.1: Resolution of TSV resistance measurements with a 500 MHz and 1 GHz sample rate for a fault-free 1 Ω and faulty 500 Ω TSV at various voltage levels.

Chosen Voltage Level (Percentage of V_{dd})	**Minimum Detectable Resistance Change**	
	at 1 GHz (Ω)	**at 500 MHz (Ω)**
99	24.3	48.6
95	40.4	80.8
90	55.6	111.2
60	161.3	322.6
50	221.2	442.4
40	324.7	649.4
10	2777.8	5555.6

calibration curve for a single TSV of Figure 2.12. As can be seen, high accuracy is achieved for a range of faulty resistances. Higher resolutions are achieved in the 400-600 Ω range, although this is based on a curve calibrated only at every 500 Ω. It is expected that more data points in the calibration curve would lead to more accurate results at other resistance values.

The test time for TSV resistance measurements can be estimated from the voltage level to which C_{charge} is charged and the number of TSVs and TSV networks that must be tested. For example, consider a die with 10,000 TSVs and 20 TSVs per network, for which C_{charge} is charged to 99% of V_{dd}. Due to bandwidth and current limitations of the probe card, it is assumed that only 100 TSV networks can be tested in parallel at a time. From simulations of fault-free TSVs, maximum currents of 46 μA are sunk through each probe needle during resistance measurement. This is well within the current limits of the smallest probe needles [79] ($\approx$120 mA for tip diameter of 1.0 mil and $\approx$400 mA for tip diameter of 5 mil), and it is thus likely that in these circumstances, more than 100 TSV networks could be tested at a time. The time required for measuring the resistance of all TSVs in this example is 80 μs, not including the time required to move the probe card.

It is also possible to test the TSV resistance of multiple TSVs in parallel at the cost of resolution of the result. Figure 2.13 shows charge times when two or three parallel TSVs

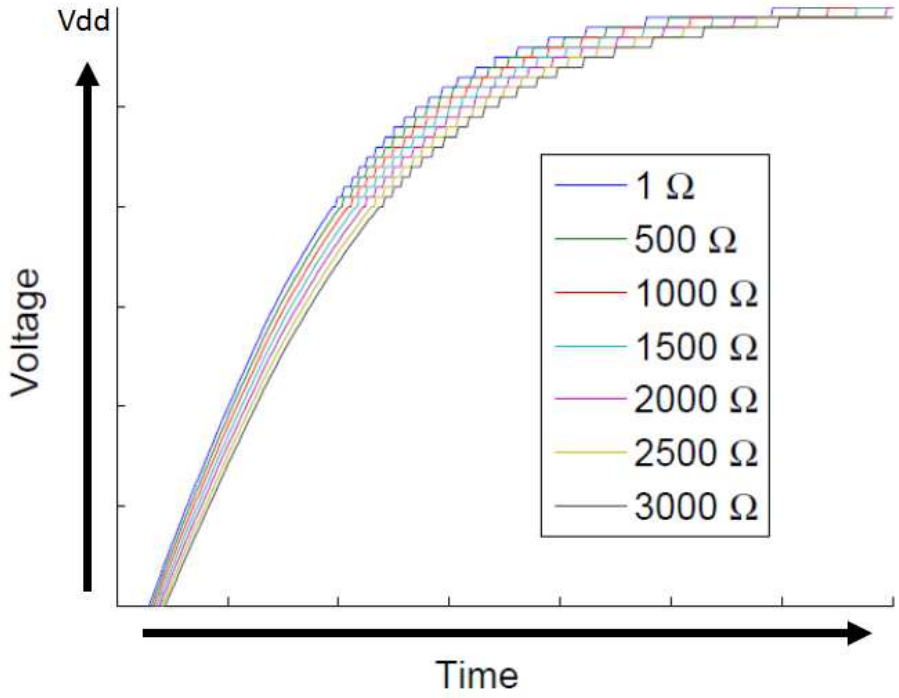


FIGURE 2.12: Capacitor charging through TSVs of varying resistance.

Table 2.2: Measurement accuracy at varying faulty TSV resistances.

Actual Resistance (Ω)	**Measured Resistance (Ω)**	**Percent Difference**
100	110.8	10.8
200	207.3	3.7
300	304.3	1.4
400	401.8	0.5
500	499.1	0.2
600	596.8	0.5
700	695.0	0.7
800	793.4	0.8
900	891.8	0.9
1000	990.8	0.9

are under test. In each case, the resistance for all TSVs in the group tested in parallel increases from 1 Ω to 3000 Ω in increments of 500 Ω. A loss of resolution is experienced on two fronts. The first is that the difference in charge times between the chosen TSV resistances decreases to 10 ns for two TSVs in parallel and 5 ns for three. This loss of resolution can be overcome to an extent with a larger capacitance C_{charge}, although larger

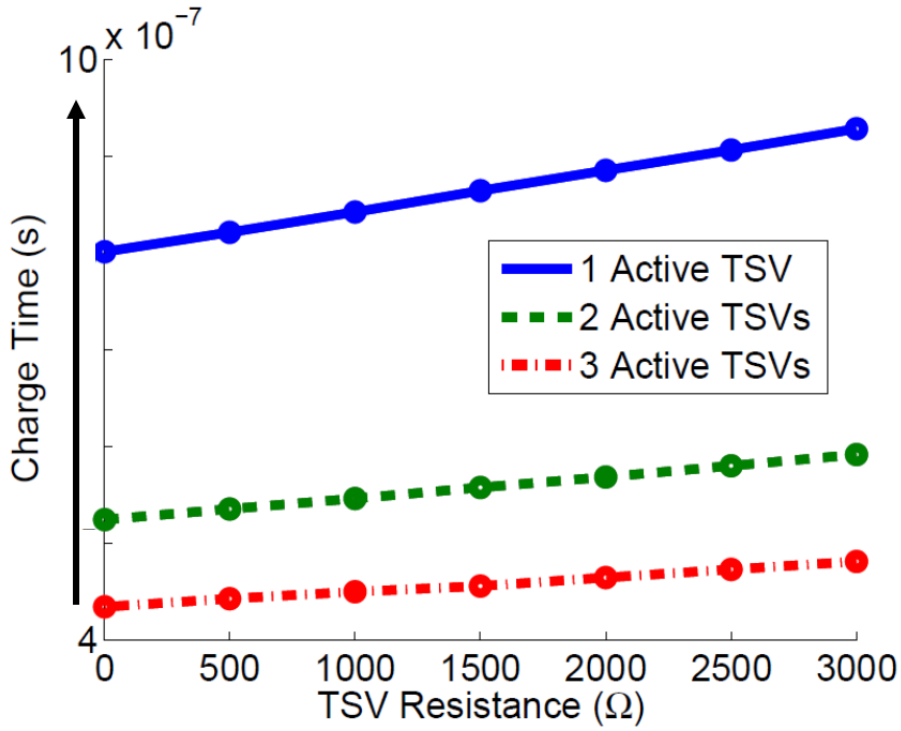


FIGURE 2.13: Capacitor charge time through one, two, and three TSVs to 0.99 V_{dd}.

capacitances are more susceptible to error from leakage. The second reason for loss of resolution lies in the averaging that must take place between the resistances of the TSVs tested in parallel. This problem cannot be easily alleviated using the proposed method. Some averaging may be desirable in the test environment, in which case it would be faster to test groups of TSVs in each TSV network in parallel with one another. The controller can be designed appropriately.

The robustness of TSV resistance measurements in a 20-TSV network under process variations is examined next. The TSV under test is considered to have a resistive fault with a total resistance of 50 Ω. Resistances on the other TSVs in the network are simulated with a Gaussian distribution in which 3-σ is a 20% spread from the nominal value of 1 Ω. All TSV capacitances are simulated with a similar Gaussian distribution using a nominal value of 20 fF, and leakage resistances are distributed around a nominal 1.2 TΩ. Charge times are then compared to a calibrated curve. As can be seen from a 100-trial Monte Carlo

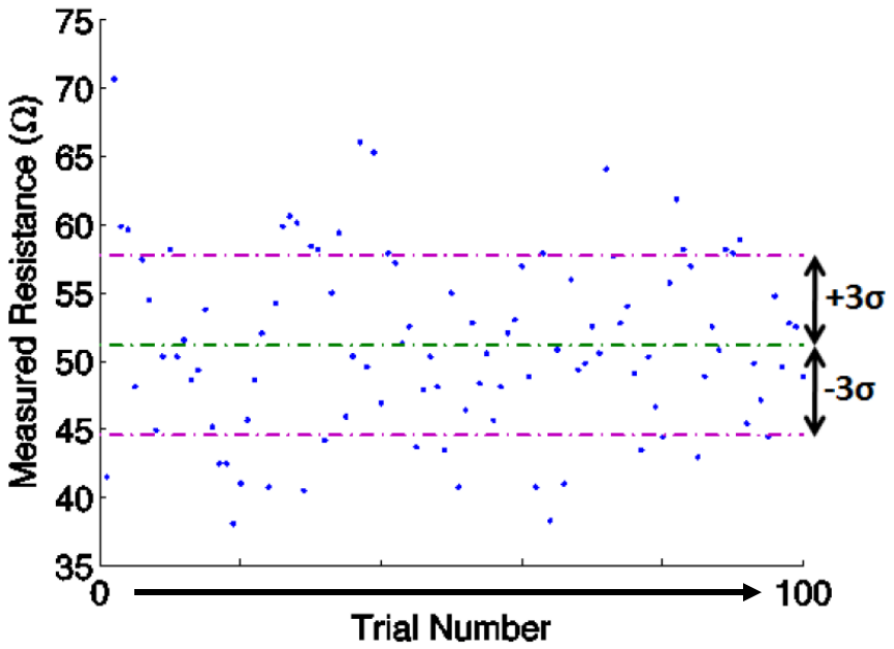


FIGURE 2.14: 100-point Monte Carlo simulation of TSV resistance measurements with 20% variation in the TSV resistance, leakage resistance, and capacitance of fault-free TSVs.

simulation in Figure 2.14, the resolution of resistance measurements remains high under process variations, with a mean measurement of 51.2 Ω and a standard deviation of 6.6 Ω.

The accuracy of TSV resistance measurements in a TSV network where more than one TSV is faulty is explored. The Monte Carlo simulations of Figure 2.14 are repeated, this time assuming that each TSV is defective with a Gaussian probability density function. For this example, let the 3-σ value for defective TSV resistance be 100 Ω around a 150 Ω nominal value. It is assumed that the 3-σ value of the TSV capacitance under process variation is 30f fF, with a nominal value of 20 fF. The leakage resistance 3-σ value was 400 GΩ around a 1.2 TΩ nominal value. Figure 2.15 presents results for a 100-trial Monte Carlo simulation in this scenario. Good resolution in resistance measurements continues to be achieved, with a mean of 141 Ω and a standard deviation of 54 Ω. While the defective TSVs are severe, their impact on resistance measurements is reduced because a capacitance C1 that is orders of magnitude larger than the TSV capacitance is chosen. The charge time

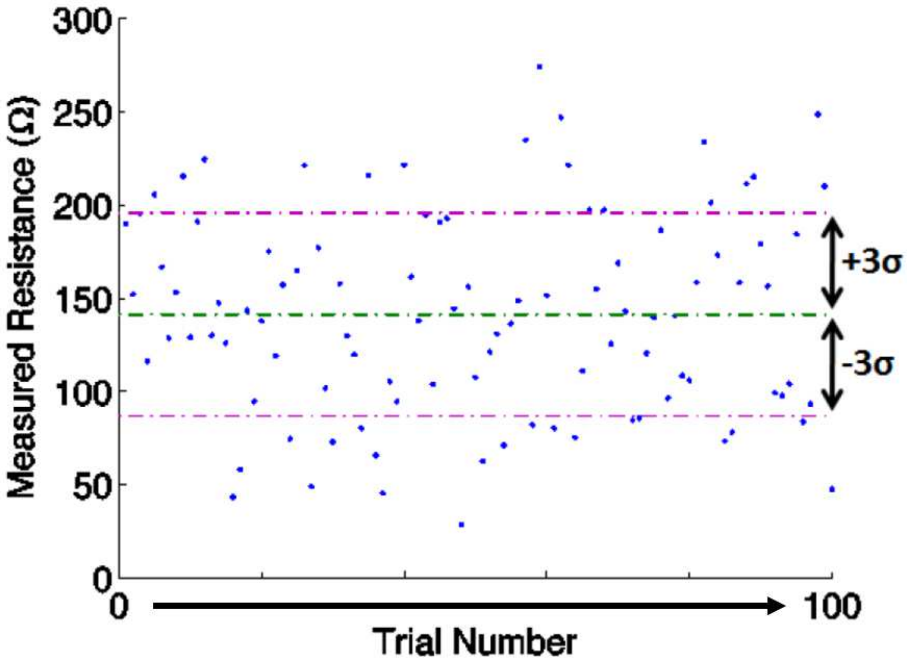


FIGURE 2.15: 100-point Monte Carlo simulation of TSV resistance measurements for multiple TSV resistive, leakage, and capacitive defects and under process variations.

of this capacitor dominates the much smaller changes in charge time caused by varying RC values in the TSV network.

Finally, the accuracy of TSV resistance measurements in a TSV network when the contact resistance varies between TSVs is examined. Many probe needles are not flat, for example they may terminate at tapered plateaus. Three different models of TSV contact resistance are explored. The first (static) profile assumes that contact resistance is independent of TSV location in the TSV network, with a Gaussian distribution for contact resistance with a 3-σ value of 10 Ω around a 40 Ω expected value. The second (linear) profile increases contact resistance linearly within the TSV network the further a TSV is from the center of the theoretical probe needle. The linear profile varies contact resistance with a Gaussian function per TSV with a 3-σ value of 2 Ω around a 5 Ω expected value for the innermost TSVs to a 3-σ value of 15 Ω around a 30 Ω expected value for the outermost TSVs. The last (exponential) profile increases contact resistance exponentially within the

TSV network the further a TSV is from the center of the network. The exponential profile varies contact resistance with a Gaussian function per TSV with a 3-σ value of 5 Ω around a 5 Ω expected value for the innermost TSVs to a 3-σ value of 20 Ω around a 100 Ω expected value for the outermost TSVs. A 100-point Monte Carlo simulation was performed for each profile in an attempt to measure a 50 Ω faulty TSV. Contact resistance is additive to TSV resistance, so the expected value of the contact resistance is subtracted from the measurement to obtain the TSV resistance. The results for the static profile are shown in Figure 2.16, with a mean measured faulty resistance value of 50.8 Ω and standard deviation of 3.3 Ω. The simulation results for a linear profile are shown in Figure 2.17, and results for an exponential profile are shown in Figure 2.18. The faulty resistance measurements for these simulations were a mean of 50.9 Ω and standard deviation of 0.7 Ω for the linear model and a mean of 50.8 Ω and standard deviation of 1.7 Ω for the exponential model. As can be inferred, as long as the expected value of contact resistance is near to the actual contact resistance, accurate measurements of TSV resistance can be obtained due to the additive nature of contact resistance.

Similar to the calibration curves for TSV resistance measurements, calibration curves can be determined for leakage resistance as shown in Figure 2.19. This calibration plots the voltage of C_{net} after 8 μs of the TSV network left in a floating state on the x-axis. The y-axis consists of the corresponding total leakage resistance RL_{net}. Due to the non-linear property of capacitance discharge, a calibration curve was created from this data using a logarithmic fit in base 10.

The effect of process variation on leakage resistance measurements is shown in Figure 2.20. As before, 100-point Monte Carlo simulations were performed. A Gaussian distribution was used with TSV resistance, leakage resistance, and TSV capacitance varying with a 3-σ of 20% around their nominal values. One faulty TSV was assigned a leakage resistance of 100 MΩ. As can be seen, this leakage was accurately determined in network

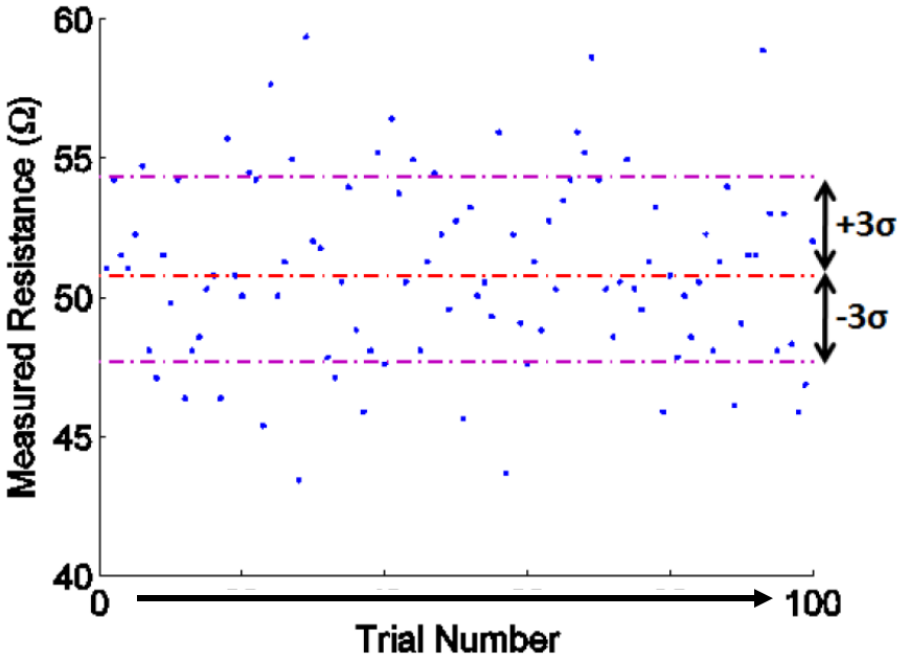


FIGURE 2.16: 100-point Monte Carlo simulation of TSV resistance measurements for a static profile of TSV contact resistances.

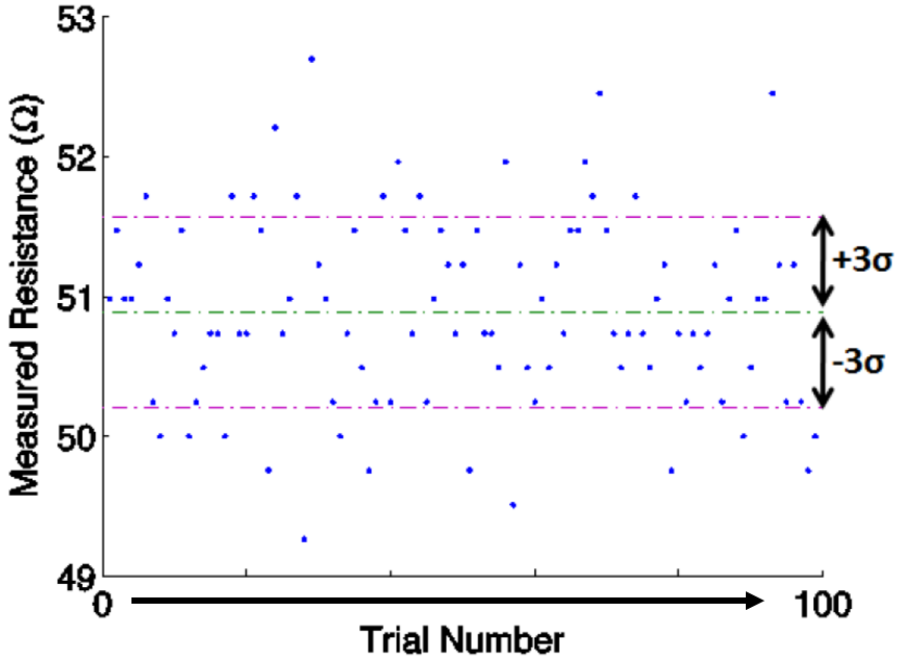


FIGURE 2.17: 100-point Monte Carlo simulation of TSV resistance measurements for a linear profile of TSV contact resistances.

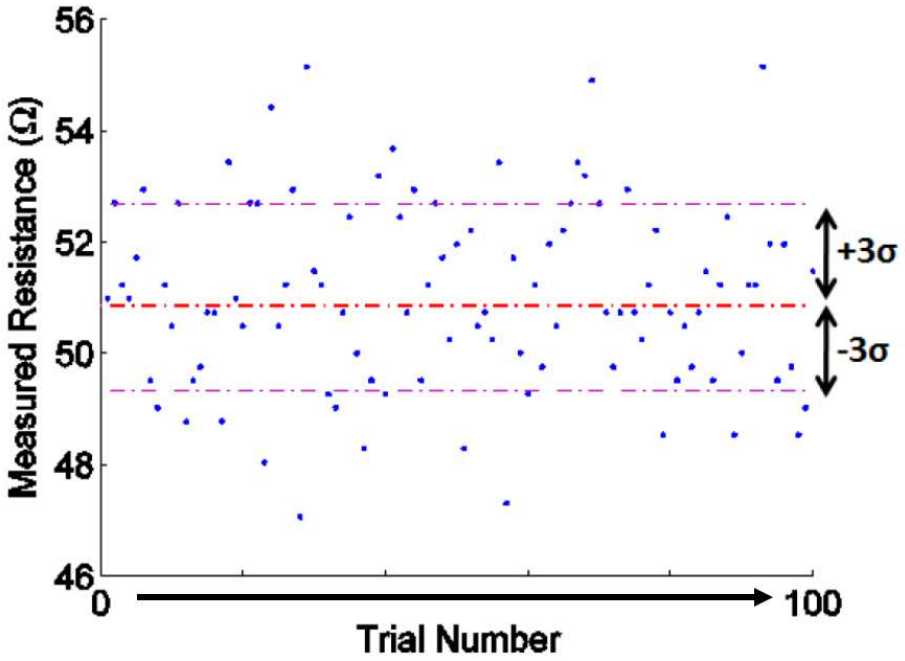


FIGURE 2.18: 100-point Monte Carlo simulation of TSV resistance measurements for an exponential profile of TSV contact resistances.

resistance measurements, with a mean of 100.5 MΩ and a standard deviation of 1.4 MΩ.

2.3.3 **Limitations of Pre-bond TSV Probing**

This section presented DFT and ATE-compatible measurement methods for pre-bond probing of TSVs. Several enhancements to this basic approach are possible. The need for averaging across all TSVs in a network implies that the resolution in capacitance measurement may be reduced in larger networks. This problem is not severe, however, as resistance and leakage tests can be used to detect most pre-bond defects in TSVs presented in [53]. The proposed method also requires that more than one contact be made with the thinned wafer during testing; therefore, it is important to minimize the number of times the probe needle must be moved during testing to avoid damage and to reduce test time. Pinpointing which TSVs in a network contribute to averaged errors, such as capacitance and leakage, is difficult with this method. Therefore, it is an open problem to identify and repair faulty TSVs.

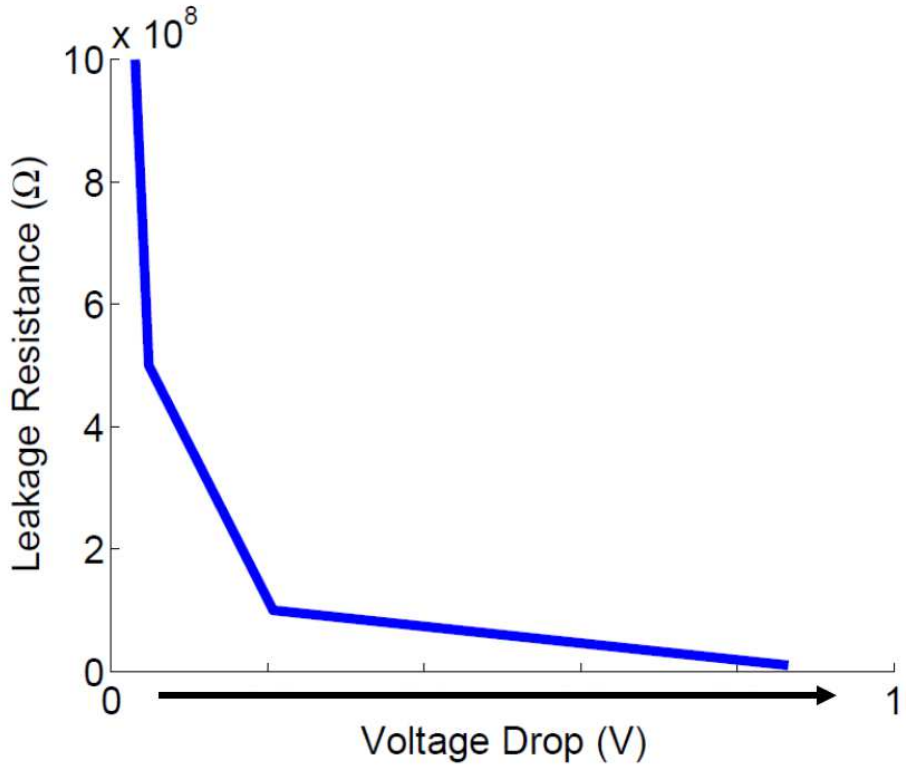


FIGURE 2.19: Plot of RL_{net} versus the voltage change of C_{net} over 8 us.

The addition of an analog tester to the test flow and the need for new probe card designs add to the test cost. Moreover, the need to move a die between testers will inevitably increase test cost. Next-generation testers might provide the required measurement capabilities. Furthermore, there is an area overhead associated with the proposed architecture, although existing test structures are reused where possible. Ultimately, the yield of TSVs versus the cost of test will have to be considered before test architecture decisions are made.

2.4 Reducing Test Time Through Parallel TSV Test and Fault Localization

Section 2.3 introduced a DFT architecture and techniques for pre-bond probing of TSVs for thinned wafers. The key idea in this probing technique is to use each probe needle to simultaneously contact multiple TSVs, forming a "TSV network". Recall Figure 2.13, which demonstrated that, within a TSV network, multiple TSVs can be tested in parallel

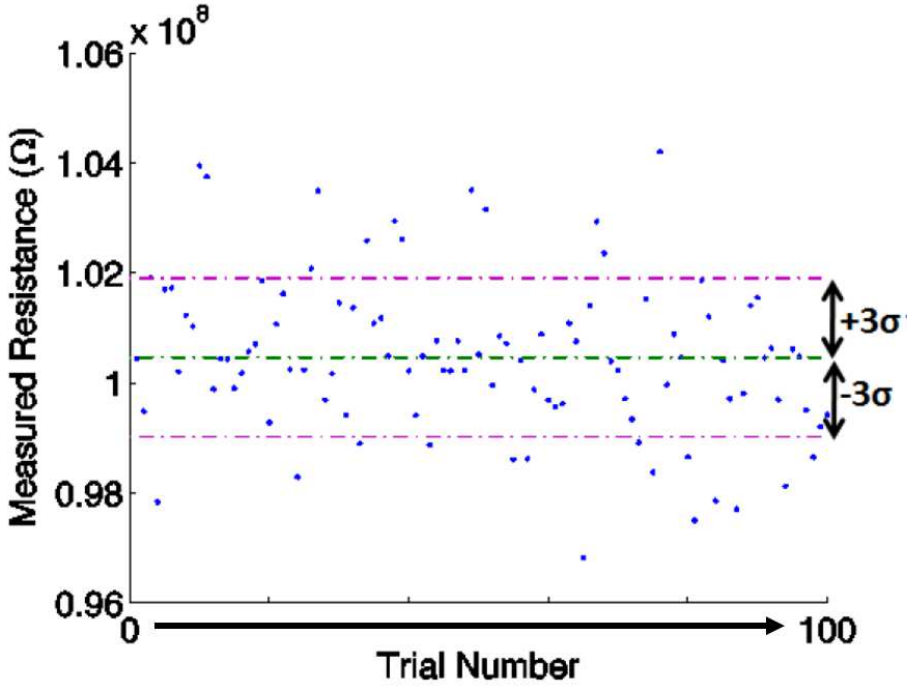


FIGURE 2.20: 100-point Monte Carlo simulation of leakage resistance measurements with 20% variation in the TSV resistance, leakage resistance, and capacitance of fault-free TSVs.

to decrease test time, but at the expense of reduced resolution of analog measurements for each TSV. In order to carry out diagnosis and TSV repair using methods such as those described in [55, 105], it is necessary to identify the individual TSVs that are faulty out of those TSVs that are tested simultaneously within a TSV network.

In this section, an algorithm will be developed for designing parallel TSV test sessions such that TSV test time is reduced and a given number of faulty TSVs within the TSV network can be uniquely identified under parallel test. The algorithm returns the sets of TSVs to test in parallel within a network. The algorithm is efficient and fast, and therefore can be used as a subroutine in a more general algorithm for optimizing TSV networks.

As the number of TSVs tested in parallel increases, there is a reduction in charge time for C_{charge} for both fault-free networks and networks with a single 1000 Ω faulty TSV, as shown in Figure 2.21. The difference in charge times between a faulty and fault-free net-

work decreases as more TSVs are tested in parallel (Figure 2.13), which adversely affects resolution. Therefore, while larger TSV networks allow more TSVs to be tested simultaneously to reduce test time, the number of TSVs per test session cannot be increased beyond a limit due to resolution constraints.

Consider a TSV network consisting of six TSVs. A simple solution for testing is to test each TSV individually, resulting in six test sessions. However, significant savings in test time can be achieved if multiple TSVs are tested in parallel and the repair mechanism can target the faulty TSVs in an individual network. Consider an upper limit m on the number of faulty TSVs that need to be identified in each network. This limit would be defined based on the capability of the on-die repair architecture or on the desired level of fault localization. If the goal of TSV test is to pinpoint only one faulty TSV in a TSV network, then each TSV i needs to be in two test groups such that the TSVs in the first network for i are different from the TSVs in the second network for i. In other words, let S_1 and S_2 be the set of TSVs that are grouped with i in the first and second test sessions, respectively. Then $S_1 \cap S_2 = \emptyset$. If $m = 2$, then three unique test sessions are needed for each TSV to distinguish m faulty TSVs. Thus, the number of unique test sessions needed per TSV is equal to $m + 1$.

The above reasoning can be explained conceptually using the example of $m = 2$. Any fault-free TSV i may be in a test session with faulty TSV f_1, faulty TSV f_2, or in a test session without either f_1 or f_2. Hence, i can be in 3 different test sessions. In a worst-case scenario, two of the test sessions for i will contain one of f_1 and f_2. Thus, these two sessions will fail. However, the third test session can then contain neither f_1 nor f_2 and will pass, indicating that i is fault-free. It is important to note that the condition presented above is sufficient but not necessary, as it is possible that none of the test sessions for i contain f_1 or f_2.

Table 2.3 shows an example of test groups that can be designed for a network of six

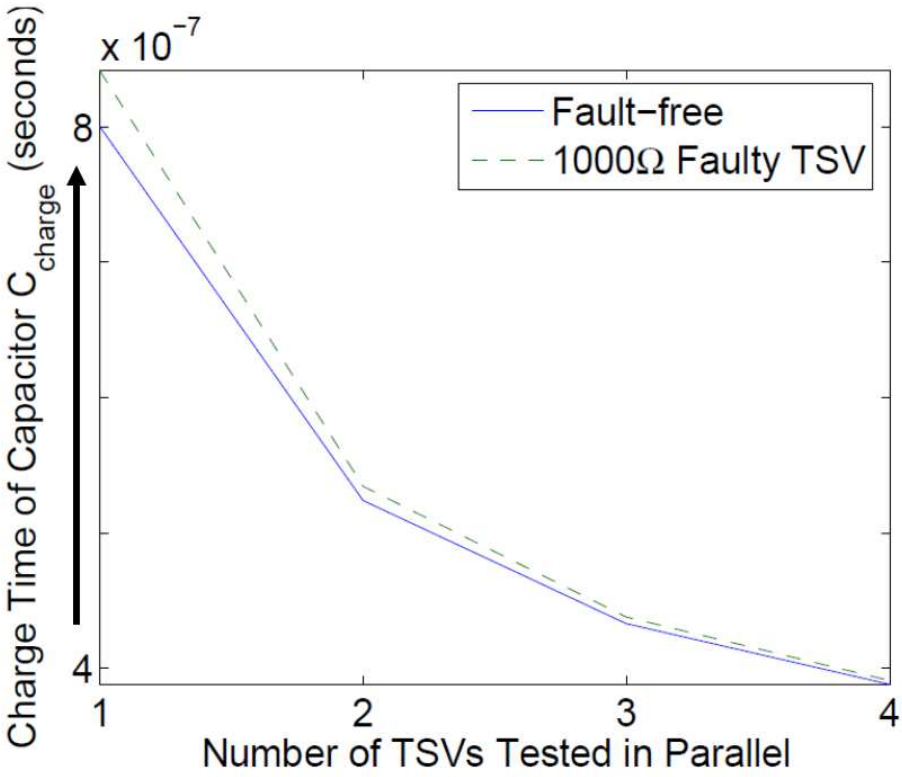


FIGURE 2.21: Reduction in capacitor charge time when driven through multiple TSVs.

Table 2.3: Parallel tests needed for pinpointing defects for one and two faulty TSVs in a 6-TSV network.

Test Session Number	Test Groups Created (m = 1)	Test Groups Created (m = 2)
1	{1,2,3}	{1,2,3}
2	{1,4,5}	{1,4,5}
3	{2,4,6}	{2,4,6}
4	{3,5,6}	{3,5,6}
5	—	{1,6}
6	—	{2,5}
7	—	{3,4}

TSVs in which at most three TSVs can be tested in parallel. Column 2 shows test groups if only one faulty TSV needs to be pinpointed, and Column 2 shows results for two faulty TSVs. When m = 1, the number of tests needed can be reduced by 2 with a resulting significant decrease in capacitor charging time, creating a 63.93% reduction in test time, based on Figure 2.21. For m = 2, one more test session is needed per TSV but a 31.19%

reduction in test time is still possible. If three or more faulty TSVs need to be identified per network, then for this example separate testing of the TSVs in a network is the best choice.

With the above example as motivation for parallel testing of TSVs, a formal problem statement can be developed. The problem of parallel-test-group creation for TSV networks is defined as follows.

Given the number of TSVs to be tested (T), the tester bandwidth B (the number of probe needles that can be active at one time, which determines how many TSV networks can be tested in each test period), a set P of the test times associated with testing different numbers of TSVs in parallel, the number of faulty TSVs m that must be identified per TSV network, and a minimum resistance resolution r, **determine** the parallel test sets for each TSV network in order to minimize overall test time while keeping the resolution of measurements at or above r and ensuring that up to m faulty TSVs in any given TSV network are uniquely identifiable.

2.4.1 Development of an Algorithm for Parallel TSV Test Set Design

Before describing the optimization algorithm, several constraints can be derived from the problem definition. These are done during algorithm initialization. First, for the purpose of this section, the TSVs are evenly distributed to networks, with the largest network having $\lceil \frac{T}{2B} \rceil$ TSVs. The "2" in the denominator results from the two separate test periods for all the TSV networks. This does not alter the generality of the algorithm, as the algorithm can be utilized for TSV networks of any size given the design and test constraints of an actual 3D design. The constant *numTests* takes the value of $m+1$, the number of test groups that are needed for each TSV. The variable *curRes*, which keeps track of the maximum number of TSVs that the algorithm tries to test in parallel with each other, is initialized to $\lceil \frac{T}{numTests} \rceil$ or r, whichever is lower. In most networks, *curRes* will equal r, except in small networks where r is a significant fraction of the TSVs in the network when compared to the number of test groups needed for each TSV. In these cases, combining TSVs into test groups at the

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \\ 3 & 4 & 1 & 2 \\ 4 & 1 & 2 & 3 \end{bmatrix} setMatrix$$

$$\begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 2 & 1 & 1 & 0 \end{bmatrix}$$
$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ - & 3 & 4 & 1 \\ 3 & 4 & 1 & 2 \\ 4 & - & 2 & 3 \end{bmatrix} \begin{bmatrix} - & 2 & 3 & 4 \\ - & 3 & 4 & - \\ - & 4 & - & 2 \\ - & - & 2 & 3 \end{bmatrix}$$

FIGURE 2.22: The setMatrix and initial steps for a TSV network with $T = 4$ and $m = 1$.

maximum resolution will result in sub-optimal test groups.

To keep track of each TSV and those TSVs that it has already been tested with, a TxT matrix, *setMatrix*, is initialized. In this matrix, the TSVs are labeled from 1 to T, with each column offset by a value of one from the previous column, as shown in Figure 2.22 at the top for a TSV network with $T = 4$.

A vector, *used*, is also initialized with length T, to track how many times each TSV has been used in a test group. All values in this vector are initialized to 0. A number of functions that act on *setMatrix* and *used* are further defined. The function **longIntersect(a)** takes a set *a* of columns from *setMatrix* and returns the set of values formed from the intersection of all of the columns. This function, if acting on N columns with T TSVs, has a worst-case time complexity of $O(N \cdot T)$. The function **bestIntersection(b)** takes one column, *b*, of *setMatrix* and determines which other column results in a set with the most intersections and contains the first value in each column. It will stop immediately if the number of intersections is equal to or greater than r. The complexity of this step is O(T^2).

The function **updateUsed(c)** takes a set of TSV numbers, nullifies corresponding values in *setMatrix* columns represented in the set, and increments the *used* value for each TSV. If the *used* value for a TSV equals *numTests*, then that column in *setMatrix* is removed completely from consideration in future test groups. This step removes the TSV number

associated with the column from all other columns. For example, Figure 2.22 shows two different iterations of the vector *used* (above each matrix) and *setMatrix* for a network with $T = 4$ and $m = 1$. On the bottom left, TSVs 1 and 2 have been added to a test group together. Therefore, the *used* value for each has been incremented to 1 and the TSV numbers have been removed from column 1 and 2. On the bottom right, TSV 1 is subsequently added to a test group with TSV 3. This increments *used* for TSV 1 to 2 and for TSV 3 to 1. Because TSV 1 has reached the value of *numTests*, the entire column associated with TSV 1 is nullified and is removed from all columns. TSV 3 would also be removed from column 1; however, the column was deleted when TSV 1 was placed in its second session.

Two other functions need to be defined. The function **notNull(***setMatrix***)** returns the number of columns in *setMatrix* that have not been nullified. The function **reduce(***d***)** takes a vector *d* and reduces the number of values in it to be equal to *curRes*. This is done with respect to intersection sets. The function preserves the TSV numbers in the set corresponding to the TSVs tested and returned by **bestIntersection()**.

Now, the algorithm **createTestGroups** can be described, (Algorithm 1). The algorithm begins with initializations, including the creation of *testGroups*, a set containing sets of TSVs that are tested in parallel. The algorithm runs iteratively through each TSV starting with TSV 1, assigning them to test groups until their corresponding *used* value is above *numTests*. To determine which TSVs have not yet been tested with each other, intersections between *setMatrix* columns are determined.

The final **if** statement in the algorithm exists to reduce the value of *curRes* that the algorithm is trying to match to avoid sub-optimal group assignments for the final TSVs in a network. For example, consider that for a network with $T = 20$, TSVs 17, 18, 19, and 20 remain to be tested, *curRes* is 4, and m is 1. The algorithm attempts to place all the TSVs into a test group $\{17, 18, 19, 20\}$, incrementing their *used* values to 1. However, each *used* value must equal 2 because *numTests* is 2. Therefore, each TSV must then be tested

Algorithm 1 createTestGroups(*T*,*B*,*P*,*m*,*r*)

```
Create and initialize setMatrix, used, numTests, curRes;
testGroups = {};
for i = 1 to T do
   while used[i] < numTests do
      inter ← bestIntersection(i);
      if (curRes ≥ 4) AND (size(inter) geq 4) then
         for each set b of curRes TSVs in inter do
            bestInter = {};
            if size(longInterb) > size(bestInter) then
               bestInter ← longInter(b);
               if size(bestInter) ≥ curRes then
                  break;
               end if
            end if
         end for
      else
         bestInter ← inter;
      end if
      if (T ∃ bestInter) AND (used(T) < numTests - 1) AND (notNull(setMatrix) < curRes - 1)
      then
         curRes ← ⌈ curRes / 2 ⌉;
         next;
      end if
      reduce(bestInter);
      testGroups ← testGroups + bestInter;
      updateUsed(bestInter);
   end while
end for
```

individually, even after testing them together. To avoid this, *curRes* is instead decremented to 2 and the algorithm tries again. Decrementing could continue if needed, but at 2 this yields the test groups $\{17,18\}$, $\{17,19\}$, $\{18,20\}$, and $\{19,20\}$, which reduces test time compared to testing each TSV individually.

The above procedure guarantees that $m + 1$ unique test groups are created for each TSV and that each faulty TSV can be uniquely identified. The *used* vector ensures that each TSV is placed in $m + 1$ unique test sessions. The one exception to this rule is when a TSV is placed in a test session with no other TSVs, in which case this single test session is sufficient for determining whether or not the TSV is faulty. To ensure that each test session contains a unique combination of TSVs, the *setMatrix* and associated column intersections identify those TSVs that have and have not been tested together. Each intersection returns those TSVs that can still be combined to form unique test sessions.

An example underlying the iterative nature of this algorithm is shown in Figure 2.23

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 \end{bmatrix} \textit{used}$$

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 3 & 4 & 5 & 6 & 1 \\ 3 & 4 & 5 & 6 & 1 & 2 \\ 4 & 5 & 6 & 1 & 2 & 3 \\ 5 & 6 & 1 & 2 & 3 & 4 \\ 6 & 1 & 2 & 3 & 4 & 5 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ - & - & - & 5 & 6 & 1 \\ - & - & 5 & 6 & 1 & 2 \\ - & 5 & 6 & - & 2 & 3 \\ 5 & 6 & - & - & 3 & 4 \\ 6 & - & - & - & 4 & 5 \end{bmatrix} \textit{setMatrix}$$

$$\begin{bmatrix} 2 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 2 & 2 & 1 & 1 & 2 & 1 \end{bmatrix} \textit{used}$$

$$\begin{bmatrix} - & 2 & 3 & 4 & 5 & 6 \\ - & - & - & 5 & - & - \\ - & - & 5 & 6 & - & 2 \\ - & 5 & 6 & - & 2 & 3 \\ - & 6 & - & - & 3 & 4 \\ - & - & - & - & 4 & - \end{bmatrix} \begin{bmatrix} - & - & 3 & 4 & - & 6 \\ - & - & - & - & - & - \\ - & - & - & 6 & - & - \\ - & - & 6 & - & - & 3 \\ - & - & - & - & - & 4 \\ - & - & - & - & - & - \end{bmatrix} \textit{setMatrix}$$

$$\begin{bmatrix} 2 & 2 & 2 & 1 & 2 & 2 \end{bmatrix} \begin{bmatrix} 2 & 2 & 2 & 2 & 2 & 2 \end{bmatrix} \textit{used}$$

$$\begin{bmatrix} - & - & - & 4 & - & - \\ - & - & - & - & - & - \\ - & - & - & - & - & - \\ - & - & - & - & - & - \\ - & - & - & - & - & - \\ - & - & - & - & - & - \end{bmatrix} \begin{bmatrix} - & - & - & - & - & - \\ - & - & - & - & - & - \\ - & - & - & - & - & - \\ - & - & - & - & - & - \\ - & - & - & - & - & - \\ - & - & - & - & - & - \end{bmatrix} \textit{setMatrix}$$

FIGURE 2.23: A step through of **createTestGroups** with $T = 6$, $m = 1$, and $r = 4$.

for $T = 6$, $m = 1$, and $r = 4$, i.e., up to four TSVs may be tested in parallel. The *used* vector is shown on top, with *setMatrix* below. After initialization, the value of *curRes* is four. The first iteration adds the set $\{1,2,3,4\}$ to *testGroups*, with *used* and *setMatrix* updated appropriately. The second set produced is $\{1,5,6\}$, followed by $\{2,5\}$, $\{3,6\}$, and finally $\{4\}$. This results in five test groups compared to six for a serial test case, and results in a test time reduction of 44.40%.

2.4.2 **Evaluation of the createTestGroups Algorithm**

In this section, simulation results for TSV networks with varying values of T, m, and r are presented. In order to determine test times, simulations using HSPICE were done on a

TSV network of 20 TSVs. The resistance of each TSV and contact resistance is 1 Ω and the TSV's associated capacitance is 20 fF. These numbers are based on data reported in the literature [73, 35]. The probe head resistance is 10 Ω. This value is several Ω higher than contact resistances seen with probe cards today [74, 75] to account for the low contact force needed and unusually small features probed in our scheme. The transistors are modeled using predictive low-power 45 nm models [76]. Transmission-gate transistor widths were set to 540 nm for PMOS and 360 nm for NMOS. These larger widths were chosen such that the gate, when open, would have little impact on signal strength. A strong and weak inverter were used, with the strong inverter having widths of 270 nm for PMOS and 180 nm for NMOS, and the weak inverter having 135 nm for PMOS and 90 nm for NMOS. These were chosen such that the majority of transistor W/L ratios were 2/1 for NMOS and 3/1 for PMOS. The charge-sharing capacitor C_{charge} was modeled at 10 pF, chosen to be an order of magnitude larger than the fault-free capacitance of the TSV network. This is sufficiently large to achieve good resolution in measurement without being so large that charge times are unreasonable or leakage becomes a significant issue. The power supply voltage V_{dd} for both the probe electronics and the circuit under test was set at 1.2 V.

All test time reductions shown in this section refer to the reduction in test time compared to the case of testing each TSV individually. This test time reduction considers only the time needed to charge C_{charge}, and not the time needed for control signals or the movement of the probe card. A reduction of 0% means that the algorithm could not determine a solution that resulted in test times lower than the sequential testing baseline case. To simplify the presentation, the resolution r is given as the maximum number of TSVs that can be tested in parallel.

Figure 2.24 lists the test time reduction versus the number of faulty TSVs that must be pinpointed, for different resolution values in a 20-TSV network. It can be seen that, in general, increasing m results in less reduction in test time. This is expected, as pinpointing

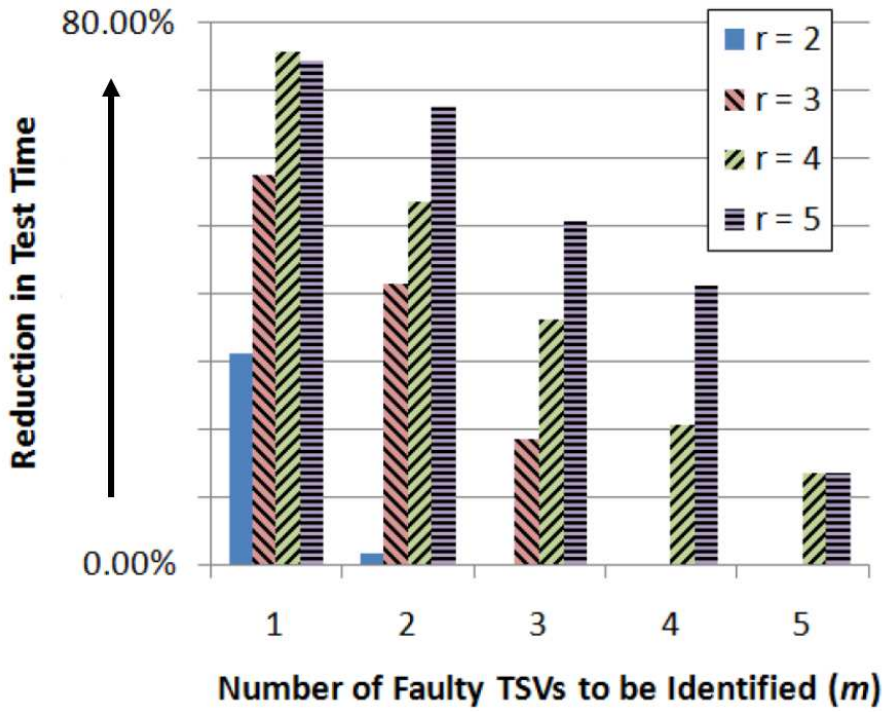


FIGURE 2.24: Reduction in test time for a 20-TSV network.

more faulty TSVs requires more test groups. For the 20-TSV network, an increase in resolution tends to result in a decrease in test time. This is because there are enough TSVs in the network to capitalize on larger test groups. An exception occurs for a resolution of four TSVs with $m = 1$, which results in a larger reduction in test time than a resolution of five TSVs with $m = 1$. Although both of these optimizations produce 10 test groups, a resolution of $r = 4$ creates 10 test groups, each with four TSVs. For a resolution of $r = 5$, only six test groups contain five TSVs, with two test groups of three TSVs and two test groups of two TSVs. Overall, this results in higher test time. Our algorithm allows for the entire design space to be quickly traversed, allowing the pinpointing of optimal values of r, given a limit on the maximum allowable resolution, to minimize test time.

The above effect can be seen more clearly in networks of fewer TSVs. Figure 2.25 reproduces the data from Figure 2.24 for an 8-TSV network. As can be seen, a resolution of $r = 4$ leads to significantly shorter test times when compared to $r = 5$ for $m = 1$. This is because for $r = 4$ the algorithm produces, on average, larger as well as fewer test groups

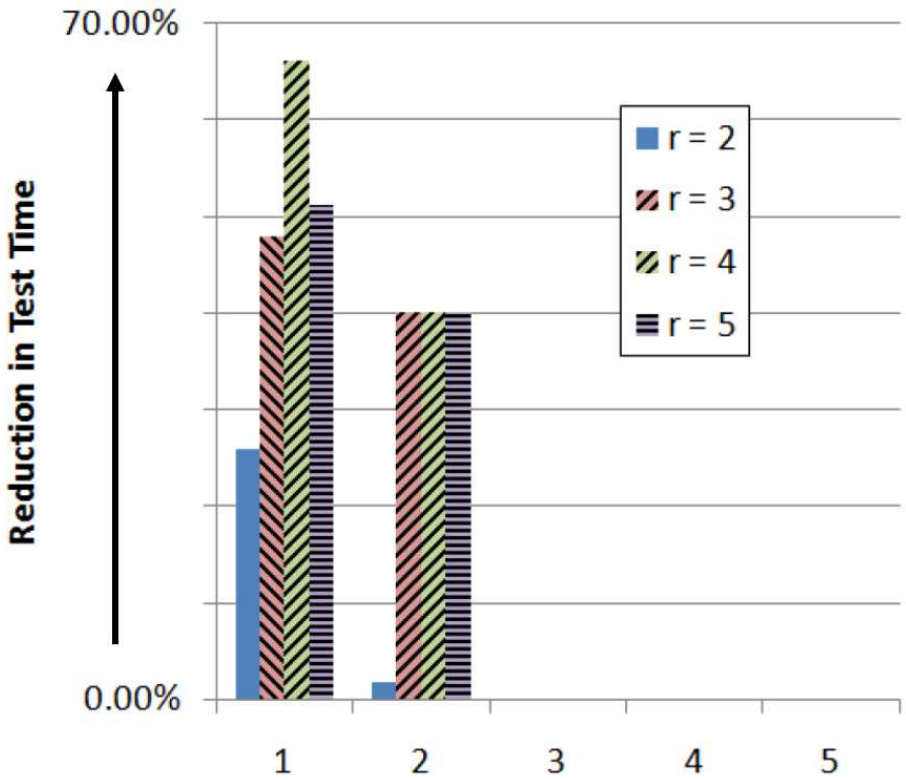


FIGURE 2.25: Reduction in test time for a 8-TSV network.

than for $r = 5$. Due to adjustments that the optimization algorithm made to *curRes* on account of the small size of the TSV network, test groups for higher resolutions and larger values for m were the same. Compared to the data for a 20-TSV network, higher values of m were a larger portion of the TSVs in the network. Thus sequential testing was more effective for $m \geq 3$.

It is necessary to examine the effect of the number of TSVs in the network on test groups. Figure 2.26 shows the reduction in test time with respect to m at a fixed resolution of $r = 3$ and various values for T. For a given resolution, larger reductions in test time are achieved when the values of T and m are such that most test groups contain the maximum number of TSVs that can be tested in parallel. For $m = 1$, this situation occurs for seven TSVs, i.e. $T = 7$. For $m = 2$ and $m = 3$, the greatest reduction in test time is obtained for 11 and 15 TSVs, respectively. These results further motivate the need for careful design and

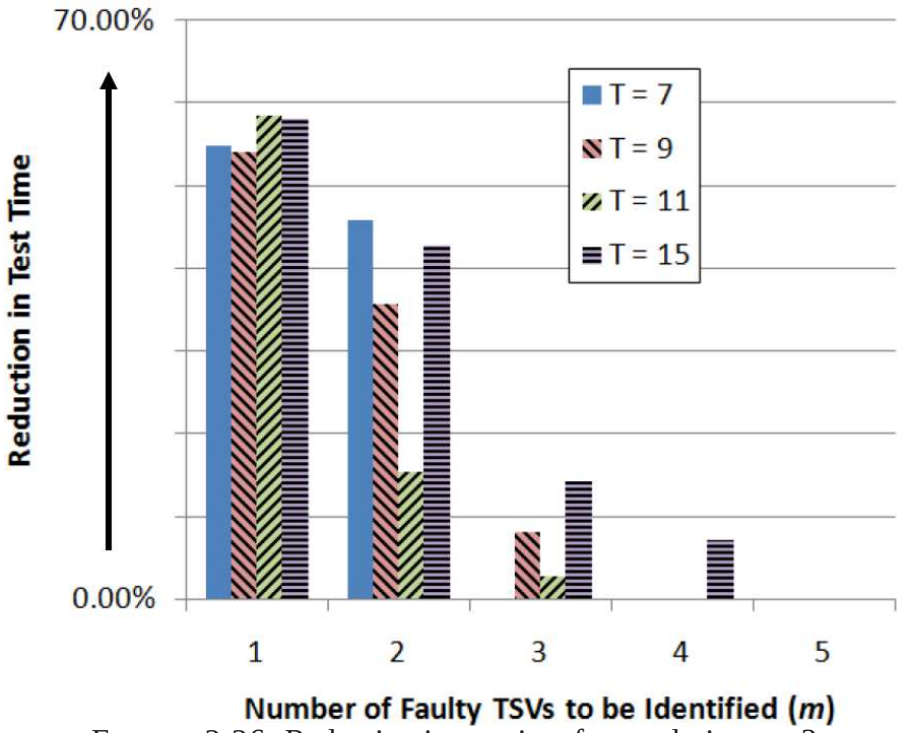


FIGURE 2.26: Reduction in test time for resolution $r = 3$.

optimization of all parameters. Automated design tools can use the fast algorithm described in this section for this purpose. The generation of an array of data across values of T from 5 to 20, m from 1 to 5, and r from 2 to 5 took less than 3 seconds of CPU time.

Finally, the number of test groups produced during optimization is explored. Figure 2.27 shows the number of test groups produced with respect to m at a resolution of $r = 4$ and for various values of T. Data points are not shown for values of m and T for which the algorithm could not reduce test time relative to the baseline case of sequential TSV testing. For smaller values of m, the algorithm often produced fewer test groups when compared to the number of tests needed for each TSV individually. With larger TSV networks, it is possible to reduce test time while increasing the number of groups needed. This reduction in test time (but with more test groups) increases controller and routing complexity. It remains an open problem to determine the best trade-off by considering implementation cost for the test application scheme.

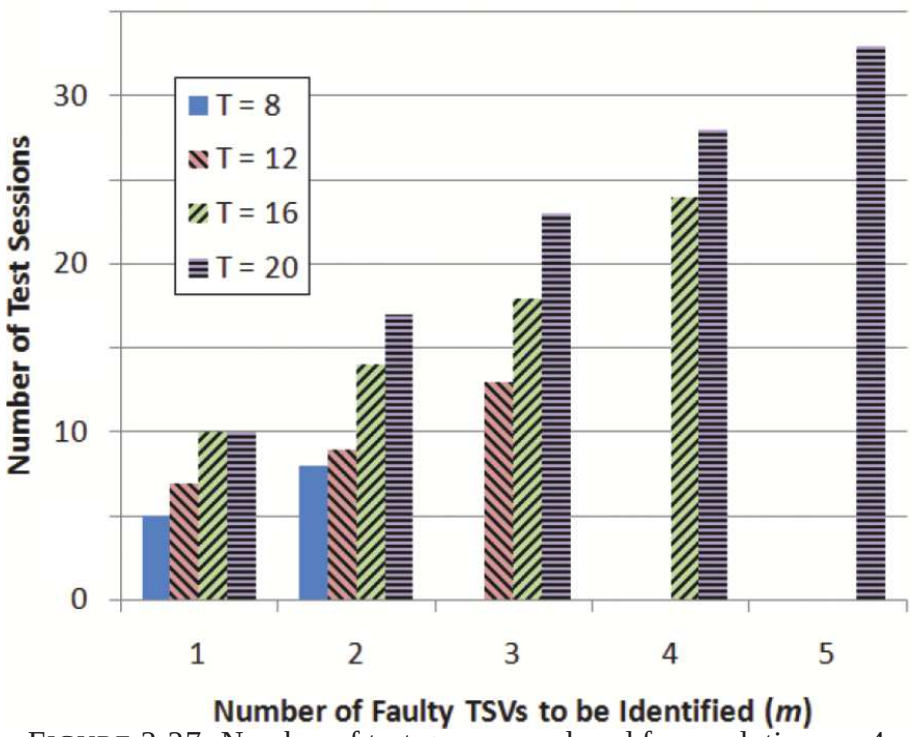


FIGURE 2.27: Number of test groups produced for resolution r = 4.

2.4.3 **Limitations of the createTestGroups Algorithm**

Though the **createTestGroups** can provide significant reductions in test time, it does not create optimal test groups in every design situation due largely to its greedy nature in assigning TSVs to test groups. Consider again the example of Figure 2.23 where $T = 6$, $m = 1$, and $r = 4$. The algorithm begins by placing four TSVs in the test set $\{1,2,3,4\}$. Unable to create any more 4-TSV test sets, it then produces the test set $\{1,5,6\}$. It once again is unable to create at test set at resolution $r = 3$, and so creates the test sets $\{2,5\}$ and $\{3,6\}$. Finally, it can only place the last TSV alone in the test set $\{4\}$. This does result in a considerable reduction in test time over serial TSV testing (44.40%), but this is not an optimal test time.

Further reduction in test time could be achieved if the test sets $\{1,2,3\}$, $\{1,4,5\}$, $\{2,4,6\}$, and $\{3,5,6\}$ were used instead. These tests result in a significant 63.93% reduction in test time over the serial test case, and a 35.11% reduction over the test solution

developed by the algorithm. It should be noted that the algorithm can produce the more optimal test set if its parameters are instead set to $T = 6$, $m = 1$, and $r = 3$, but this example does demonstrate the sub-optimality of the algorithm overall and how results can be improved if the algorithm is utilized as part of a larger optimization framework.

2.5 Conclusions

This chapter has examined novel on-die DFT methods and measurement techniques combined with pin electronics that allow probing-based pre-bond testing of TSVs. It has been demonstrated how a probe card can be used together with the DFT architecture to measure resistance and capacitance, as well as to perform stuck-at and leakage tests. These parametric tests are applied to a network of TSVs, and HSPICE simulation results highlight the effectiveness of this approach. It is possible to test not only multiple TSV networks in parallel, but also several TSVs in each network in parallel if some loss of resolution in the measured data is acceptable. The test method yields reliable and accurate results even in the presence of process variations and multiple defective TSVs. The proposed method highlights the relevance of commercial testers and the role that tester companies can play in the maturation of 3D SIC test methods. It also demonstrates the need for cheap, effective low-force probe techniques to minimize touchdowns and damage to dies and TSVs as well as to keep the cost of probing reasonable when compared to BIST techniques.

Furthermore, the problem has been formulated of identifying faulty TSVs when using pre-bond probing to test TSVs within the same TSV network simultaneously. This problem can be described in terms of test time, resolution for fault detection, and the number of test groups required to localize a given number of defective TSVs. An efficient algorithm has been introduced for calculating test groups for parallel TSV testing within TSV networks. Results have been provided to highlight the significant reductions in test time achievable with parallel test. The test time reduction depends on the number of TSVs in

a network, the number of faulty TSVs to detect, and the minimum resolution needed for measurements. The results highlight the need for a general multi-objective framework in which the proposed algorithm can be an important component.

3

Pre-Bond Scan Test Through TSV Probing

3.1 Introduction

Previous chapters have discussed the need for pre-bond KGD test to ensure high stack yields. Chapter 2 briefly discussed BIST methods and examined in detail probing to enable pre-bond TSV test. While TSV test is important for KGD test, it covers only a small fraction of the tests that must be performed to achieve complete KGD test. In particular, the majority of die area is dedicated to logic and associated memory.

Many test architectures are discussed in the literature [50, 37] to enable pre-bond test, including the die-level standard wrapper. However, these architectures rely on the deposition of oversized probe pads on either or both of those TSV pillars or face-side TSV contacts that will be utilized for pre-bond logic test. These probe pads are sized to allow single contact of probe needles to a TSV. They require a significant amount of space and considerably limit TSV pitch and density. Therefore, only a limited number of probe pads are utilized for pre-bond test. This significantly limits the pre-bond test bandwidth available for logic test, increasing test time and cost.

To address the above challenges, this chapter explores a novel method for pre-bond

testing of die logic through backside probing of thinned dies. It extends the test architecture discussed in Chapter 2. While the probing technique of Chapter 2 focused only on TSV test, this chapter focuses on scan-test of die logic utilizing scan chains that can be reconfigured for pre-bond test to allow scan-in and scan-out through TSVs. This method does not require many oversized probe pads, save for a few critical signals such as power, ground, and test/functional clocks. A significant benefit of the method outlined in this chapter is that, coupled with the architecture described in Chapter 2, it enables both pre-bond TSV/microbump test as well as pre-bond structural test under a single test paradigm. Furthermore, probe pads do not limit the test bandwidth of pre-bond test, so a high-bandwidth pre-bond test can be performed quickly through the methods outlined in this chapter. Several different scan configurations are examined in this chapter, each providing varying degrees of test parallelism depending on design constraints. A variety of simulations will be discussed that demonstrate the feasibility of scan test through TSV probing, including area overhead, power/current delivery needs, current density in TSVs, and scan clock frequencies.

The rest of this chapter is organized as follows. Section 3.2 examines the architecture of Chapter 2 in context of pre-bond scan test, and introduces two new kinds of GSFs depending on the type of TSV they are driving. Subsection 3.2.1 introduces the proposed scan architecture and test method for performing pre-bond scan test. Subsection 3.2.2 presents HSPICE simulation results for a number of dies with TSVs across two logic-on-logic 3D benchmarks, highlighting the feasibility of the method presented discussed in this chapter. Finally, Section 3.3 concludes the chapter.

3.2 Pre-bond Scan Test Through TSV Probing

In Chapter 2, a measurement and DFT technique was introduced to enable the pre-bond test of TSVs through probing. This method utilized a die wrapper similar to that discussed

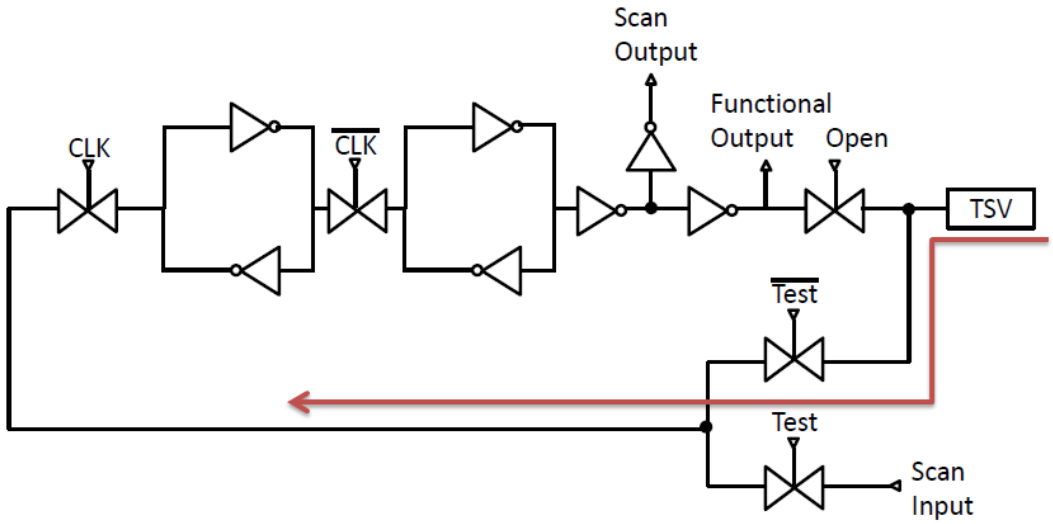
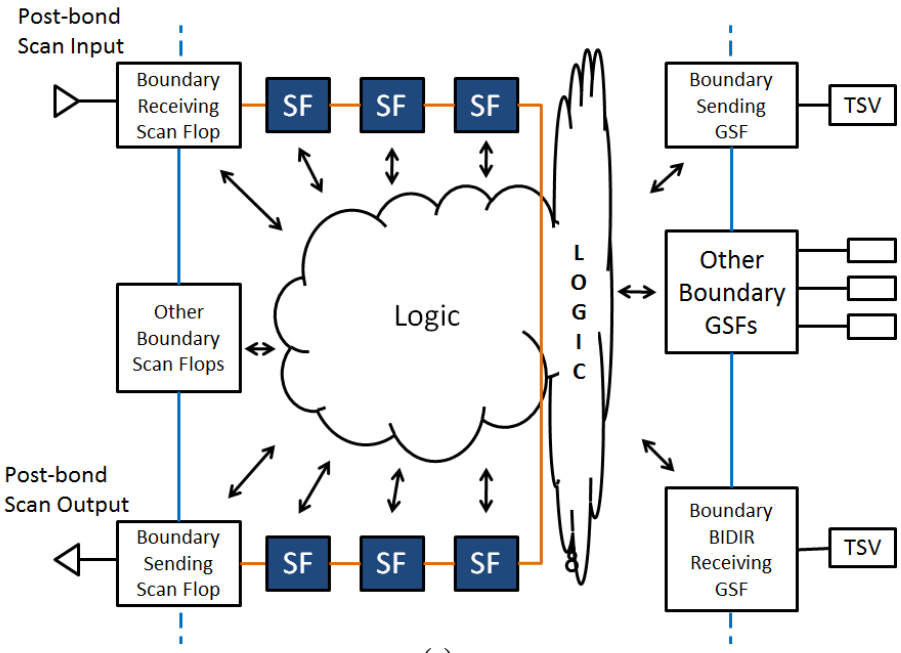


FIGURE 3.1: An example gate-level design for a bidirectional gated scan flop with receiving path highlighted.

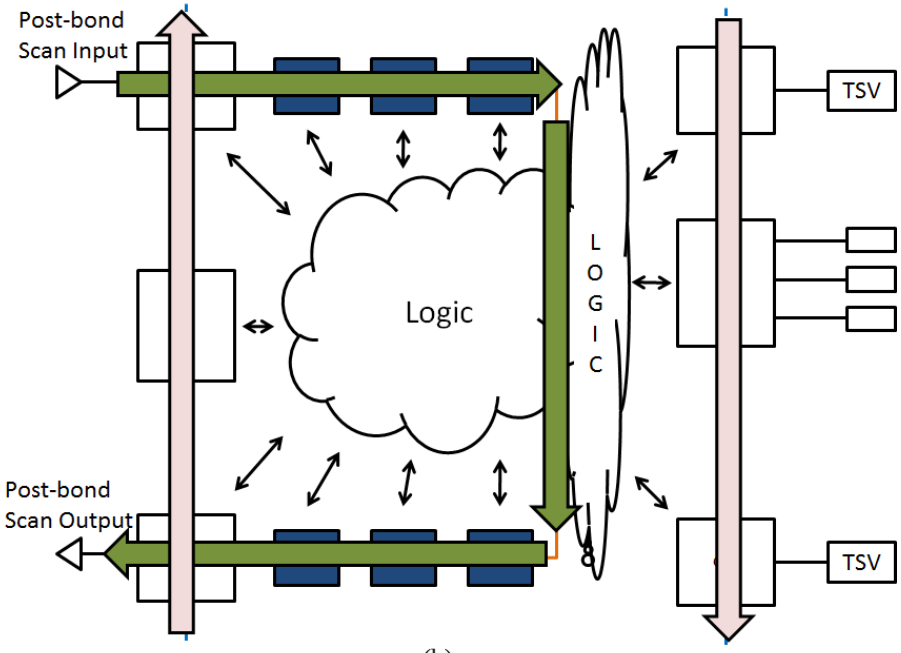
in the Introduction but replaced the boundary scan flops with gated scan flops (GSFs). For pre-bond scan test, the directionality of TSVs in a TSV network is important, so it is necessary to distinguish between GSFs on *sending* and *receiving* TSVs. A sending TSV is a TSV that is driven by logic on its own die during functional operation and sends a signal to another die. A receiving TSV is a TSV that is driven by logic on another die during functional operation and receives a signal.

Figure 3.1 shows the gate-level design of a bidirectional GSF. The receiving path of the GSF is highlighted with an arrow. As discussed in Chapter 2, a GSF multiplexes between a test input and a functional input and can be connected to other GSFs to form a scan chain. The difference is that the GSFs include a buffer of two inverters and a transmission gate at the output of the flop, which accepts an 'open' signal to switch between a low- and a high-impedance output. This design effectively allows the TSV to be driven by the GSF or to be left floating. GSFs on receiving TSVs must be bidirectional GSFs because the GSF must be able to drive the TSV during pre-bond TSV test.

In Chapter 2, the GSFs were included before each TSV to enable pre-bond probing of TSVs. It was shown that by using probe needles larger than an individual TSV, groups of



(a)



(b)

FIGURE 3.2: The assumed post-bond scan architecture: (a) scan chains and logic; (b) movement of test data.

TSVs can be deliberately shorted together to form a single circuit called a TSV network. Using the GSFs, the resistance of each TSV can be accurately determined, along with the

average capacitance of each TSV. Contact force and variations in contact quality between TSVs were shown to have little effect on the ability to accurately characterize TSVs.

To enable pre-bond scan test using the same architecture, scan chains are reconfigured into a pre-bond test mode in which scan inputs and scan outputs are connected to TSV networks. This allows the probe station to apply test patterns to the die and to read test responses through the scan chains and pre-bond TSV scan I/O. A key advantage of using TSVs for pre-bond scan test is that not all TSVs need to be contacted for die logic test. It is necessary to contact only those TSVs that are required for pre-bond scan. Results for a 3D benchmark that will be discussed further in Subsection 3.2.2 show that for 100 scan chains for pre-bond test, as few as 10.7% of the TSVs need to be contacted. Therefore, only one touchdown is likely needed for pre-bond scan test, and this can be the second touchdown required for pre-bond TSV test to allow for scan test after all TSVs have been tested to be fault free.

3.2.1 **Performing Pre-Bond Scan Test Through TSV Probing**

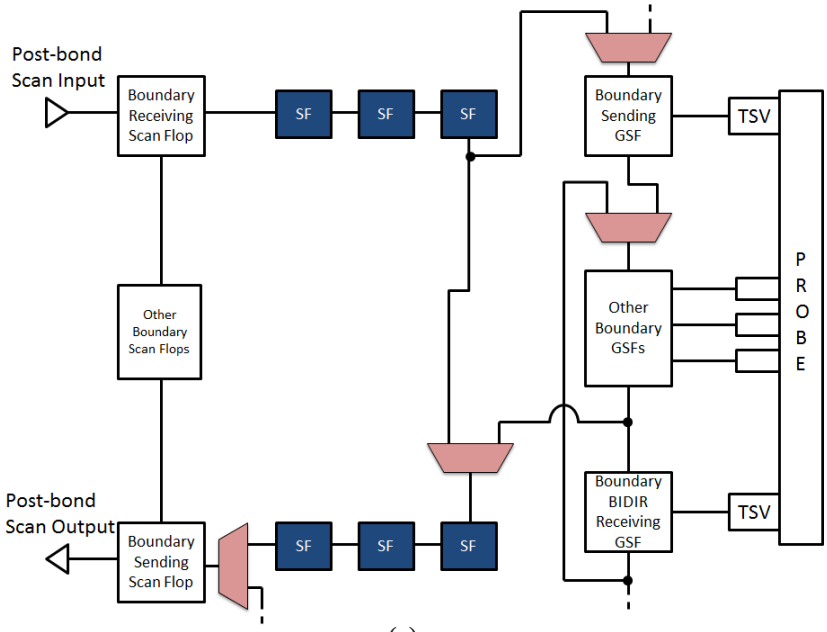
This section describes the test architecture and methods required to perform pre-bond scan test [113]. A post-bond scan architecture is assumed that is compatible with the die wrapper discussed in the Introduction, as shown in Figure 3.2. Figure 3.2(a) shows a single scan chain and a number of boundary scan flops. The scan chain consists of typical scan flops (SFs), while boundary scan registers at the TSV interface are GSFs. As with die wrappers, some landing pads must be supplied for providing essential signals to the die, such as power, ground, and clocks. The post-bond scan input and scan output for a scan chain enter the die through the boundary register. In the bottom die in a stack, this interface is through external test pins or a JTAG test-access port. For other dies in the stack, scan I/Os are connected to the dies below them in the stack. Parallel loading of the boundary registers decreases test time, but serial scan test is also available by shifting through the boundary scan chain. This is illustrated in Figure 3.2(b), which shows the post-bond movement of

test data. Test data can be shifted not only through the internal scan chain, but also around the boundary registers. All scan flops interact with die logic.

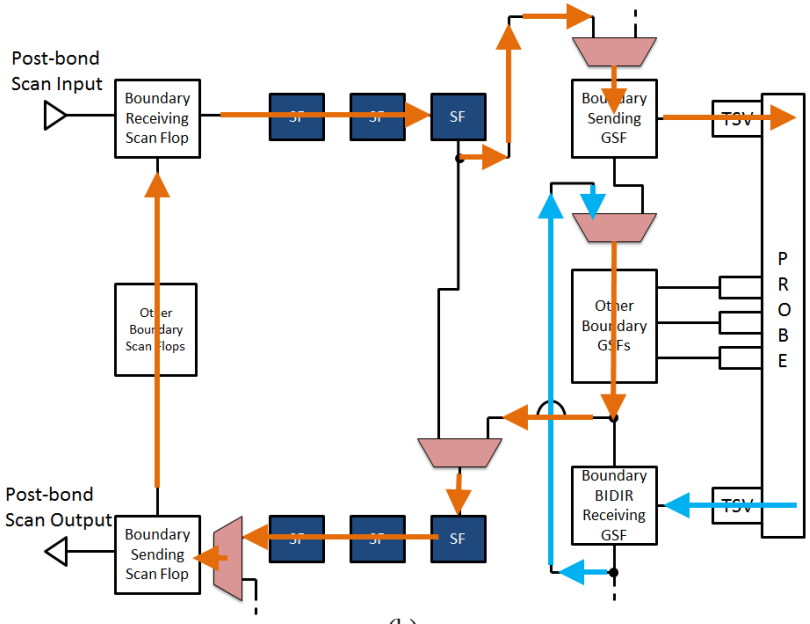
Multiplexers are added to the scan path to allow scan chains to be reconfigured to a pre-bond mode in which their scan-in and scan-out connections are through TSVs, as shown in Figure 3.3(a). A receiving GSF is chosen for the reconfigured scan-in and a sending GSF is chosen for the scan-out. Because many boundary scan registers are logically separated from internal scan chains in the post-bond mode, they need to be stitched to the scan path in pre-bond mode to enable testing. Multiplexers are added in as few places as possible to achieve access to all internal and boundary scan flops in order to minimize hardware overhead.

Consider the multiplexers added to a single scan chain in Figure 3.3(a). The receiving GSF, which now acts as the pre-bond scan input, is enabled to accept its functional input driven through the TSV. Its scan output is then multiplexed into the boundary scan chain. This is done such that the sending GSF used as a pre-bond scan output and the receiving GSF used as a pre-bond scan input will interface with scan flops that are adjacent to one another in the post-bond scan chain. The output of the boundary scan flop that is used to feed the pre-bond scan input is then multiplexed into the scan chain. The post-bond scan output, post-bond scan input, and other boundary registers are stitched into the scan chain. Finally, the sending GSF used as a pre-bond scan output is multiplexed to the end of the scan chain. The pre-bond movement of test data is shown in Figure 3.3(b). The combinational logic is not shown so as to retain clarity; it is the same as in Figure 3.3(a). Arrow color changes in the figure so as not to confuse the overlapping arrows.

The reconfigured pre-bond scan chain in Figure 3.3 demonstrates one of several possible pre-bond scan configurations (Configuration A). In this example, the pre-bond scan chain's scan-in and scan-out terminals are part of the same TSV network. Under these conditions, the scanning in of test data and the scanning out of test responses must be done



(a)



(b)

FIGURE 3.3: Reconfigurable scan chains for pre-bond test: (a) added multiplexers; (b) movement of test data.

separately. This is because, in order to scan in test data, the transmission gate on the receiving GSF must be set to its low-impedance state while all other gates must be set to their high-impedance states. Likewise, while scanning out, the sending GSF's gate must be set

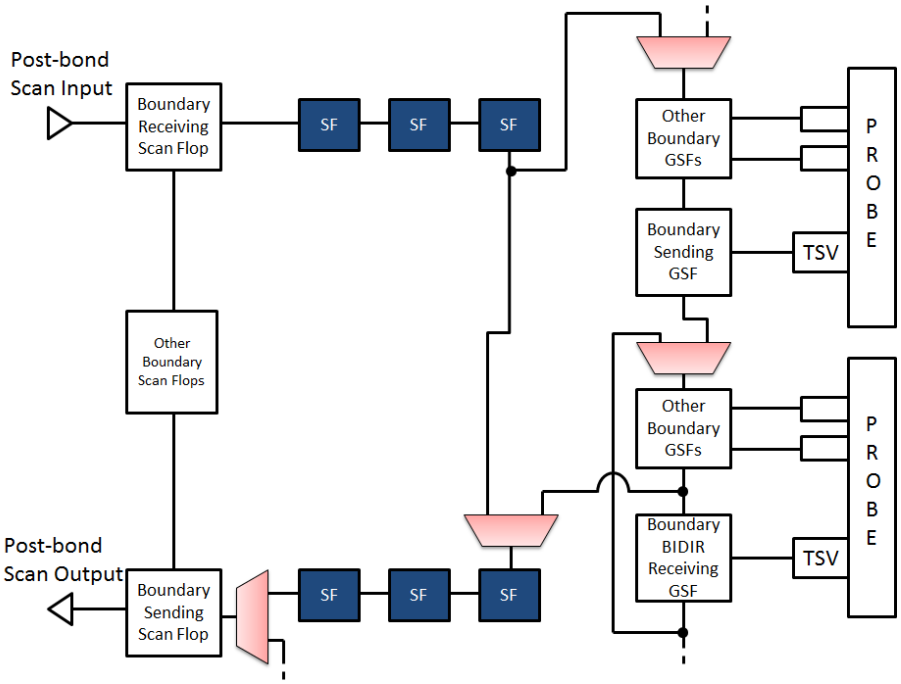


FIGURE 3.4: A reconfigurable scan chain with pre-bond scan input and scan output on different TSV networks.

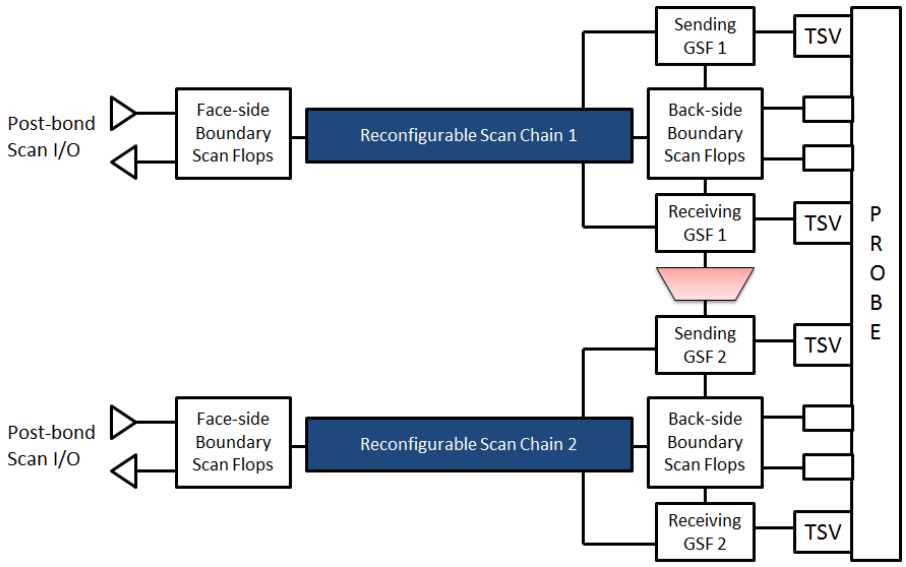
to low impedance while all others are set to high impedance. Because scan-in and scan-out occur on the same network, the maximum number of scan chains that can be tested in a single touchdown is equal to the number of TSV networks formed. In other words, the number of scan chains can at most be equal to the number of probe needles. Furthermore, if current or power limits cause the maximum scan clock frequency to be different for scan input and scan output, then the appropriate frequency must be used for the corresponding operation.

A second possible pre-bond scan configuration (Configuration B) involves the scan input and scan output on separate TSV networks, an example of which is shown in Figure 3.4. In this case, test responses can be scanned out while test patterns are scanned in. The maximum number of scan chains that can be tested per touchdown is reduced to half of the number of probe needles (or half of the number of TSV networks). Both scan input and scan output operations must occur at the lower of the possible scan frequencies, because

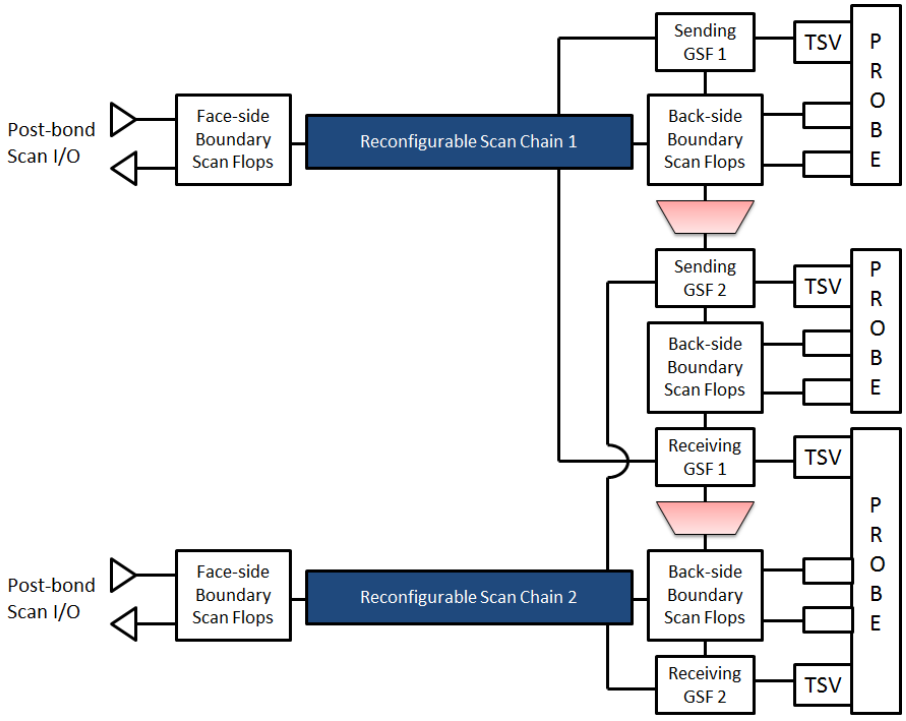
both operations occur simultaneously. It should be noted that pre-bond functional test cannot be conducted while using TSV networks, because it is not possible to supply individual inputs to TSVs within a network at the same time.

Pre-bond scan configurations can also be designed such that two or more scan inputs and/or scan outputs belong to the same TSV network. Such a configuration is desirable in a number of scenarios. Design constraints such as routing complexity or layout difficulty may prevent the routing of a scan chain's pre-bond I/O to an independent TSV network. In such a case, the scan chain may be required to share a pre-bond scan input, output, or both, with TSV networks that already have pre-bond scan I/O routed to them. In another scenario, there may exist more post-bond scan chains than there are pre-bond TSV networks in a single touchdown. Because realigning the probe card and performing a second touchdown significantly increases test time, it is preferable to test all scan chains in a single touchdown. In this case, sharing TSV networks between pre-bond scan I/O can result in test times shorter than if two scan chains are stitched together to form a single, longer scan chain.

Figure 3.5 shows a pair of examples where two separate scan chains share TSV networks. In Figure 3.5(a), the pre-bond scan inputs and outputs of both scan chains are routed to the same TSV network (Configuration C). In Figure 3.5(b), reconfigurable scan chains 1 and 2 share a TSV network for their pre-bond scan inputs, but have independent TSV networks for their scan outputs (Configuration D). When scan chains share a TSV network across their pre-bond scan inputs, patterns can be applied using a broadcast method to reduce test time. During the broadcast of test patterns, the scan chains must receive unique shift signals such that one or both can shift in bits depending on which bit is applied to the TSV network. Test patterns for both scan chains can then be combined into single patterns that require fewer test clock cycles to scan in than scanning in the patterns serially. When scan outputs share a TSV network, test responses must be scanned out serially. Therefore, the configuration of Figure 3.5(a) must utilize either a serial or broadcast scan-in and a



(a)



(b)

FIGURE 3.5: Reconfigurable scan chains with two pre-bond scan inputs on the same TSV network and (a) scan outputs on the same TSV network or (b) scan outputs on separate TSV networks.

serial scan-out, and scan-in and scan-out operations cannot occur simultaneously. For the configuration of Figure 3.5(b), scan-in must occur serially or through broadcast, but scan-

out from the two scan chains can occur in parallel. Scan-in and scan-out operations can occur simultaneously.

From a test time perspective, it can be determined which configuration is best to use for a given design. The following design constraints are utilized:

- s - The number of pre-bond scan chains created during reconfiguration.
- p - The number of patterns to be applied during pre-bond scan test.
- m - The number of scan cells in the longest pre-bond scan chain. This value is assumed to be constant across touchdowns for determining test times in this chapter, though it need not be.
- l_i - The number of bits in the length of the ith pattern, where $i = 0$ is the first pattern of a pattern set. This variable is only required for configurations 3 and 4, which utilize broadcast patterns. Thus, each pattern can be of a varying length and will generally be larger than m.
- n - The number of probe needles available for TSV networks on the probe card.
- t - The time required for the alignment and touchdown of the probe card.
- f_{in} - The maximum scan-in clock frequency that can be used.
- f_{out} - The maximum scan-out clock frequency that can be used.

The objective is to determine an equation for the test time T required for each configuration given the constraints above and choose the configuration that results in the shortest test time. Because the time required for alignment and touchdown is generally much longer than the time required to perform scan tests, it is often better to use configurations that require only a single touchdown.

For configuration A, scan-in and scan-out operations occur sequentially because scan I/O utilize a shared TSV network and probe needle. To speed this process, scan-in operations can use the maximum scan-in frequency f_in and scan-out operations can use the maximum scan-out frequency f_out. The equation for the test time for Configuration A is:

$$T_A = \left\lceil \frac{s}{n} \right\rceil \cdot \left(\frac{m \cdot p}{f_{in}} + \frac{m \cdot p}{f_{out}} \right) + \left\lceil \frac{s}{n} \right\rceil \cdot t \tag{3.1}$$

The number of touchdowns required to perform pre-bond scan test is given by $\lceil \frac{s}{n} \rceil$. This is then multiplied by the time required to apply all the test patterns and receive all test responses for each touchdown and added to the time required to perform all alignment and touchdown operations ($\lceil \frac{s}{n} \rceil \cdot t$).

For Configuration B, scan-in and scan-out operations can occur in parallel, which reduces the time required to apply patterns and receive test responses when compared to the time required by Configuration A. However, Configuration B can interface with half-as-many scan chains per touchdown as Configuration A. The test time for Configuration B is written as:

$$T_B = \left\lceil \frac{2s}{n} \right\rceil \cdot \left(\frac{m \cdot (p+1)}{\min\{ f_{in}, f_{out} \}} \right) + \left\lceil \frac{s}{2n} \right\rceil \cdot t \tag{3.2}$$

Configuration C allows for significant consolidation of scan chains across TSV networks, allowing for the test of twice as many scan chains per touchdown as Configuration A and four times as many as Configuration B. Scan-in and scan-out operations are performed sequentially, requiring two scan-out cycles for each scan-in cycle due to the need to scan out two scan chains worth of responses for each TSV network. Furthermore, patterns are of variable length due to the compression required to generate broadcast patterns. The test time for Configuration C is thus calculated as:

$$T_C = \left\lceil \frac{s}{2n} \right\rceil \cdot \left(\frac{\sum_{i=0}^{p} l_i \cdot p}{f_{in}} + 2 \cdot \frac{\sum_{i=0}^{p} l_i \cdot p}{f_{out}} \right) + \left\lceil \frac{s}{2n} \right\rceil \cdot t \tag{3.3}$$

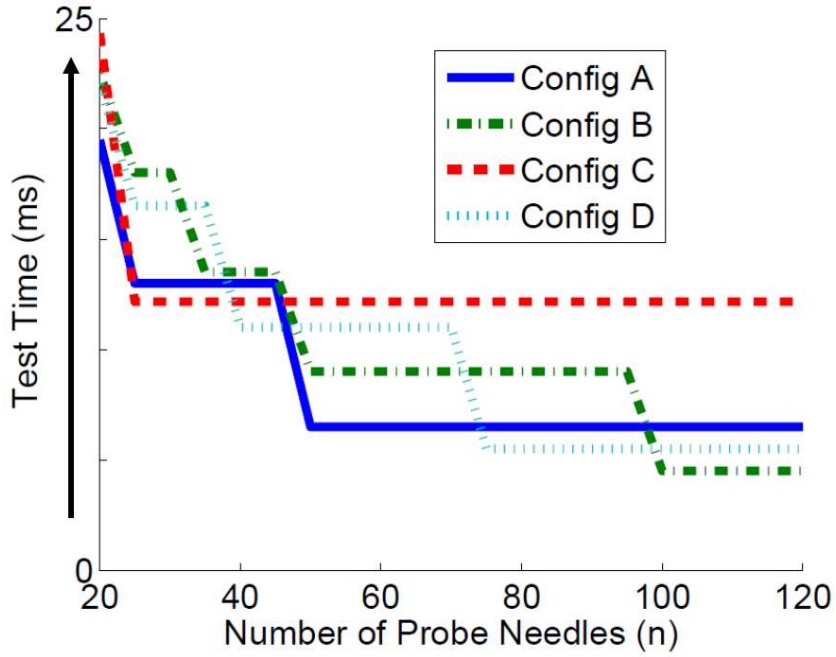
Lastly, Configuration D allows for parallel scan-in and scan-out operations while allowing for the test of more scan chains per touchdown than Configuration B. It utilizes the broadcast pattern set, and its test time is found to be:

$$T_D = \left\lceil \frac{\frac{3}{2}s}{n} \right\rceil \cdot \left(\frac{\sum_{i=0}^{p} l_i \cdot (p+1)}{\min\{ f_{in}, f_{out} \}} \right) + \left\lceil \frac{\frac{3}{2}s}{n} \right\rceil \cdot t \tag{3.4}$$

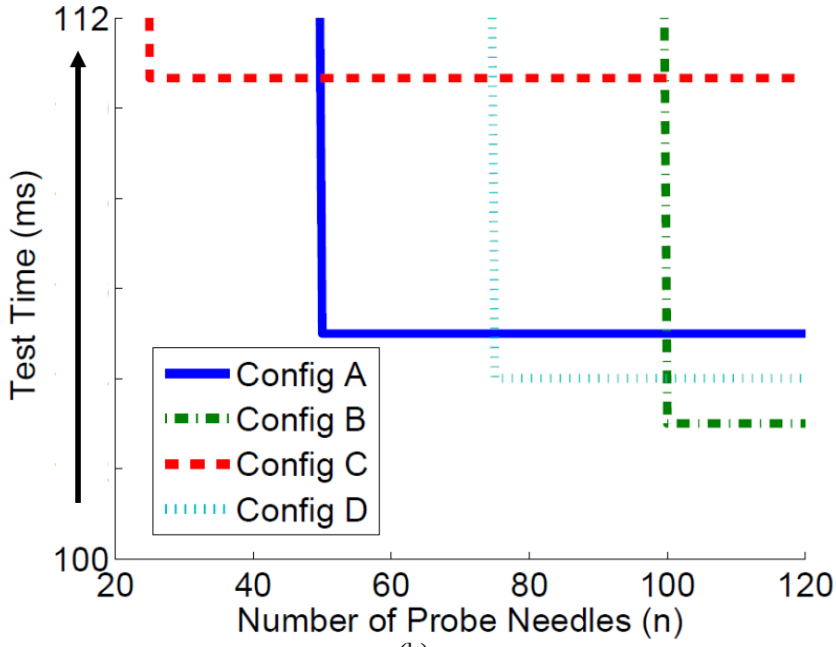
Though these equations can act as a guide in determining which configuration to use for a design, they only encompass test time considerations for creating the reconfiguration architecture. In reality, design and technology constraints, such as routing complexity, area overhead, and so forth, will also influence which configurations are feasible for a given design.

Figure 3.6 shows the test times of Configurations A, B, C, and D while varying the number of probe needles available for TSV network creation from 10 to 200. Parameter values were chosen to show the difference between the configurations, with s of 50, p of 1000, m of 300, each l_i value set to 400, f_{in} of 150 MHz, and f_{out} of 100 MHz. The alignment and touchdown time t for Figure 3.6(a) is 1.5 ms, which is relatively fast but is used to ensure that the test times of the die are not eclipsed by t to provide a complete picture of the various configurations. Figure 3.6(b) is produced when t = 100 ms, which is significantly longer than the time required to perform structural test using the given parameters, and provides a realistic look at the differences between the configurations in practice.

As Figure 3.6(a) demonstrates, there is pareto-optimality among the configurations with regard to n. For low values of n, Configurations C and A tend to result in lower test



(a)



(b)

FIGURE 3.6: The test times of Configurations A, B, C, and D with varying numbers of probe needles and alignment and touchdown time of (a) t = 1.5 ms or (b) t = 100 ms.

times as they provide a higher compression ratio for scan chains among TSV networks. At higher values of n, Configurations B and D result in lower test times as they provide

higher test parallelism with regard to pattern application. Figure 3.6(b) shows these effects in a more realistic environment, where utilizing the configuration that best matches the pre-bond scan test bandwidth and utilizes only a single touchdown has the greatest impact on test time. Which configuration will provide the lowest test times depends on both the design parameters and the probe card constraints. Section 3.2.2 will explore this issue further with results from benchmark circuits.

3.2.2 **Feasibility and Results for Pre-Bond Scan Test**

This section addresses a number of key criteria needed to demonstrate the feasibility of the proposed method:

- The current needed to be delivered to the device under test during pre-bond scan test must fall within the current-carrying capacities of TSVs and probe needles.

- The speed at which highly capacitive TSV networks are charged and discharged must be reasonable such that the pre-bond scan test time is low.

- The area overhead of the proposed method must be small.

- That boundary scan registers are necessary to achieve high coverage in pre-bond scan test.

Simulation results are presented demonstrating the feasibility of the methods presented in this chapter. Simulations were conducted in HSPICE on two 3D logic-on-logic benchmarks. The resistance and capacitance used for each TSV of 5 μm diameter were 1 Ω and 20 fF, respectively [73, 35]. Transistors were modeled using a predictive low-power 45 nm model [76] except where otherwise noted. Transmission-gate transistor widths were set to 540 nm for PMOS and 360 nm for NMOS. These larger widths were chosen such that the gate, when open, would have little impact on signal strength. For each GSF, a strong and

weak inverter were used, with the strong inverter having widths of 270 nm for PMOS and 180 nm for NMOS, and the weak inverter having 135 nm for PMOS and 90 nm for NMOS. These were chosen such that the majority of transistor W/L ratios were 2/1 for NMOS and 3/1 for PMOS. The power supply voltage for both the probe and the circuit was taken to be 1.2 V.

3D IC Benchmarks

Because 3D IC benchmarks are not available in the public domain, two benchmarks were created from cores available through the OpenCores set of benchmarks [106]. A Fast Fourier Transform (FFT) circuit and a Reconfigurable Computing Array (RCA) circuit were utilized. Both are synthesized using the Nangate open cell library [123] at the 45nm technology node [76]. The total gate count after synthesis is 299,273 with 19,962 flip-flops for the FFT circuit, and 136,144 gates with 20,480 flip-flops for the RCA circuit. Both designs were partitioned into 4 dies, with the gate counts in each die of the FFT stack being 78,752, 71,250, 78,367, and 70,904, respectively. For the RCA stack, gate counts for each die were 35,500, 34,982, 32,822, and 32,840, respectively. The logic gates in each die are placed using Cadence Encounter, and TSVs are inserted in a regular fashion, using a minimum spanning tree approach [107]. Back-to-face bonding is assumed, which means that TSVs are present only in the first three dies. The TSV counts for each die in the FFT stack are 936, 463, and 701, respectively, and for the RCA stack are 678, 382, and 394, respectively. The TSV diameters are 5 μm. The circuits were routed such that each TSV has a small microbump sized at 7 μm, and the total TSV cell size including the keep out zone is 8.4 μm, which corresponds to six standard cell rows. Each die is then routed separately in Cadence Encounter. The bottom die of the FFT 4-die layout is shown in Figure 3.7, with TSVs in white and standard cells in green.

Boundary scan cells were added at the TSV interface. The need for inserting boundary registers at the TSV interface can be motivated by examining die 0 of the 4-die FFT

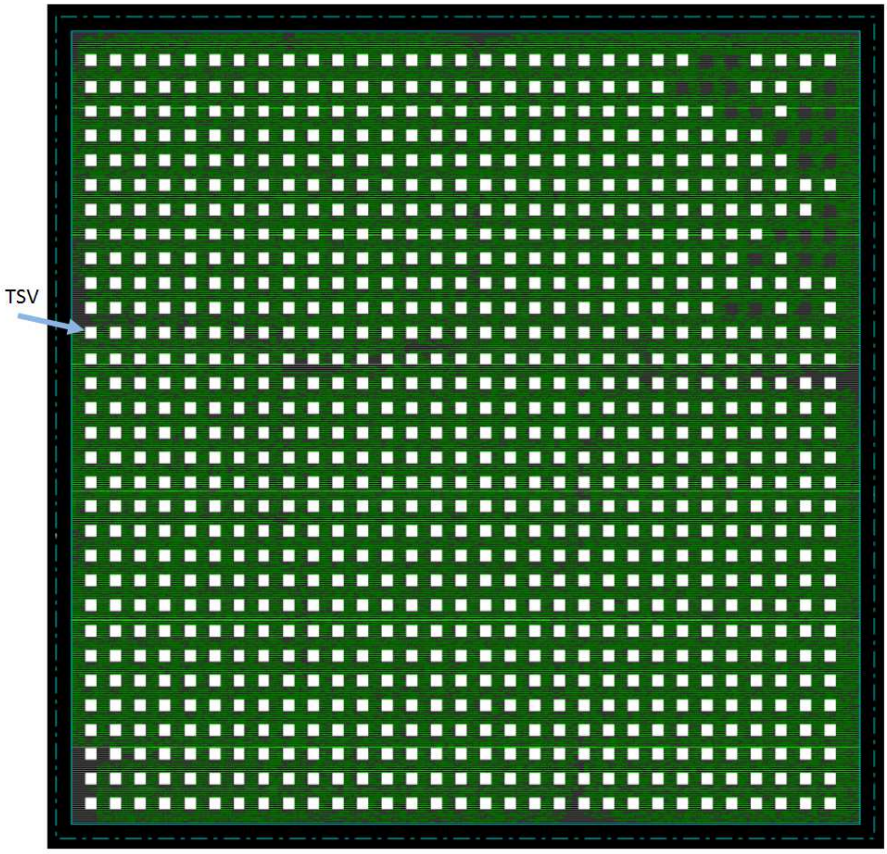


FIGURE 3.7: Layout of die 0 of the 4-die FFT benchmark, with standard cells in green and TSVs in white.

benchmark. Without boundary scan registers, the pre-bond stuck-at fault coverage is only 44.76%. With boundary registers added, the coverage increases to 99.97% for stuck-at test patterns and 97.65% for transition test patterns. This is a significant increase, especially considering that the die only contains 936 TSVs, and an industry design may contain tens of thousands of TSVs.

The area overhead of the boundary scan GSFs and scan chain reconfiguration circuits is shown for the FFT dies with TSVs in Table 3.1 and for the RCA dies with TSVs in Table 3.2. These results show area overheads between 1.0 % and 2.9 % of the total number of gates. Generally, the area overhead was higher for the RCA benchmark because the benchmark contains significantly fewer gates per die than the FFT benchmark, while at the

same time containing nearly as many flops and without a significant reduction in TSVs. This means that many boundary scan cells need to be added to the RCA dies, and there is a similar number of scan chains that need reconfiguration circuitry between the two benchmarks.

Simulation Results

The feasibility of performing scan test through probe needles is first examined in terms of sourcing and sinking currents. To determine an upper limit on the current drawn, scan chains were inserted into the benchmark. In order to manage the complexity of circuit-level HSPICE simulation, scan chains were limited to a length of 8 (6 internal scan cells and two boundary scan cells for pre-bond scan I/O per chain) for each die in each benchmark. Stuck-at and transition test patterns for this design were generated using a commercial ATPG tool and ordered based on toggle activity. Test generation yielded the toggle activity for each test pattern. For each die, two scan chains and associated logic were extracted for simulation in HSPICE based on toggle activity for the highest activity pattern and for an average activity pattern in the generated pattern set. By associated logic, it is meant that fan-in and fan-out gates for the scan cells of that scan chain were simulated in HSPICE up to primary I/O or other flip-flops. For the pattern with highest peak toggle activity, the scan chain and associated logic that yielded the largest number of transitions for that pattern were simulated. For the average pattern, a scan chain and associated logic based on the average number of toggling bits was simulated.

Figure 3.8 shows the current drawn for shifting in the highest-power stuck-at pattern for the worst-case scan chain and shifting out test responses at 25, 40, 50, 60, and 75 MHz shift frequency. The figure gives data for Die 0 of the FFT circuit. At 50 MHz, current drawn averaged at around 300 μA and, at all frequencies, peaked at almost 1 mA for about a tenth of a nanosecond. For a high toggle-activity transition fault pattern using launch-off-shift and a 1 GHz functional clock for the same die, an average current of 432 μA is drawn

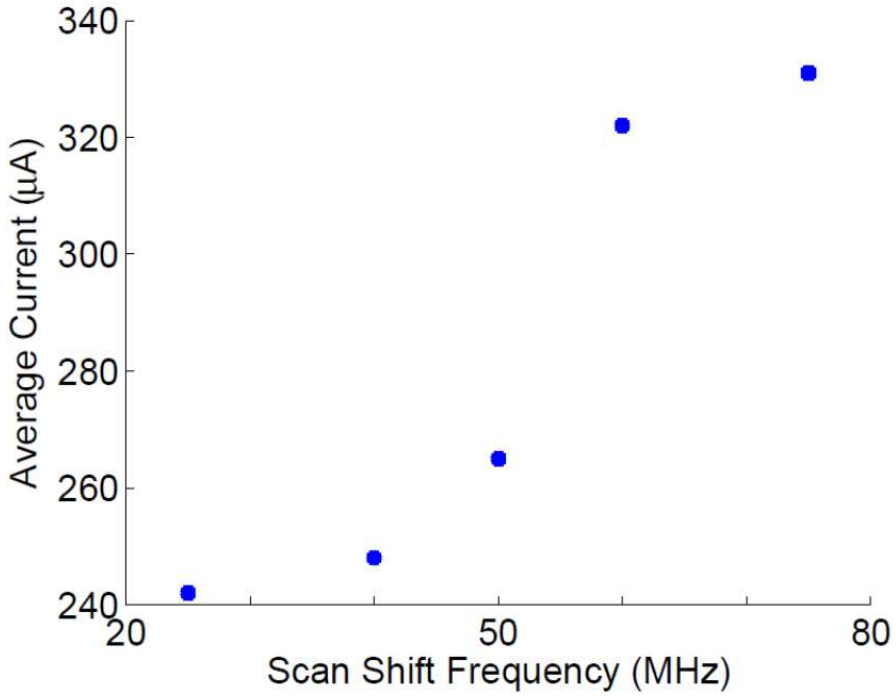


FIGURE 3.8: Average current drawn at 25, 40, 50, 60, and 75 MHz scan shift frequency for Die 0 of the FFT benchmark.

during capture and peak current similar to that of stuck-at patterns.

Table 3.1 shows the peak current and average currents drawn for the worse-case stuck-at and transition pattern for the three dies with TSVs in the FFT benchmark. For transition test, an average current draw is shown both for the shift and capture cycles, with all scan-in shift cycles averaged together. Only a single peak current is shown because the results were nearly identical for stuck-at and transition patterns. Table 3.2 shows the same results for the first three dies of the RCA benchmark. The simulations were performed with a scan-shift frequency of 50 MHz and a functional clock of 1 GHz. Because neither the driver strength nor the TSV network size were changed for these simulations, maximum scan-in and scan-out frequencies were equal for the dies. Table 3.3 and Table 3.4 show the same results for the average scan-chain and test pattern.

As is evident from the tables, the highest worst-case average current drawn for the stuck-at pattern was 327 μA, as experienced by Die 2 of the FFT benchmark. For the tran-

sition pattern, the highest worst-case current was 432 μA in Die 0 of the FFT benchmark during capture These worst-case currents were significantly less for the RCA benchmark, reaching as high as 288 μA for the stuck-at pattern and 331 μA during capture for the transition pattern. The average currents for the average scan chains and patterns were lower, as expected, though there is little change in the peak current draw.

It has been reported in the literature that a TSV can handle a current density higher than 70,000 A/cm^2 [108]. Published work on TSV reliability screening indicates that a sustained current density of 15,000 A/cm^2 is possible through a TSV without damage [109]. To sustain a peak current of 1 mA through a single 5 μm TSV in the pre-bond test method would require the TSV to be capable of handling a current density of 5093 A/cm^2. To handle a 300 μA average current, a TSV must be capable of sustaining a current density of 1528 A/cm^2. Both these numbers are well below the maximum allowable current density.

In addition to the current density limits of the TSVs, it is important to consider the amount of current that the probe needles can deliver. It has been shown in the literature that a 3 mil (76.2 μm) cantilever probe tip is capable of supplying 3 A of current for a short pulse time (less than 10 ms) [110, 79]. In the worst case, assuming that all scan chains and logic in the FFT benchmark draw the peak current at once, the probe tip would have to supply 3 A of current for less than 0.1 ns. This falls within the probe current-supply specification. If current supply from the probe is an issue, a variety of well-known methods can reduce peak and average test power on die during test, including partitioning the circuit into separate test modules, clock gating, and low-power patterns [111, 112, 114].

Table 3.5 shows the results of low-power pattern generation for Die 0 of the 4-die RCA benchmark in order to reduce test power. Column one shows the target peak toggle activity as a percentage of the unconstrained worst-case pattern toggle activity. Column two and column three provide the increase in pattern count as a percentage of the unconstrained pattern count and reduction in coverage as a percentage of the unconstrained coverage,

Table 3.1: A comparison of the worst-case results of three dies with TSVs in the FFT 3D stack.

Test Parameter (FFT Stack)	Die 0	Die 1	Die 2
Peak Current	1 mA	1 mA	1.1 mA
Avg. Current (stuck-at)	300 μA	294 μA	327 μA
Avg. Shift Current (transition)	387 μA	300 μA	335 μA
Avg. Capture Current (transition)	432 μA	341 μA	383 μA
Area Overhead	2.2 %	1.0 %	1.2 %

Table 3.2: A comparison of the worst-case results of three dies with TSVs in the RCA 3D stack.

Test Parameter (RCA Stack)	Die 0	Die 1	Die 2
Peak Current	0.8 mA	0.8 mA	0.8 mA
Avg. Current (stuck-at)	279 μA	288 μA	242 μA
Avg. Shift Current (transition)	287 μA	321 μA	261 μA
Avg. Capture Current (transition)	327 μA	331 μA	300 μA
Area Overhead	2.9 %	1.7 %	1.9 %

Table 3.3: A comparison of the average-case results of three dies with TSVs in the FFT 3D stack.

Test Parameter (FFT Stack)	Die 0	Die 1	Die 2
Peak Current	1 mA	1 mA	1 mA
Avg. Current (stuck-at)	289 μA	274 μA	291 μA
Avg. Shift Current (transition)	370 μA	281 μA	296 μA
Avg. Capture Current (transition)	412 μA	305 μA	344 μA

Table 3.4: A comparison of the average-case results of three dies with TSVs in the RCA 3D stack.

Test Parameter (RCA Stack)	Die 0	Die 1	Die 2
Peak Current	0.8 mA	0.7 mA	0.7 mA
Avg. Current (stuck-at)	270 μA	270 μA	241 μA
Avg. Shift Current (transition)	277 μA	291 μA	246 μA
Avg. Capture Current (transition)	298 μA	317 μA	261 μA

respectively. Column 4 gives the peak current draw for the worst-case pattern using the same scan chain from the simulations from Table 3.4. As can be seen from the table, peak

Table 3.5: Low-power pattern generation results for Die 0 of the 4-die RCA benchmark.

Activity Target (RCA Stack)	**Pattern Inflation**	Coverage Loss	Peak Current
90 %	1.4 %	0 %	0.75 mA
80 %	3.9 %	0 %	0.71 mA
70 %	8.1 %	0 %	0.66 mA
60 %	16.3 %	3.5 %	0.58 mA

toggle activity can be significantly reduced (to roughly 70% of the unconstrained toggle activity, resulting in a 0.66 mA peak current) without a loss of coverage and with at worst 8.1% additional patterns. A reduction to 60% results in some coverage loss (3.5%), but can reduce peak current draw to 0.58 mA. These results demonstrate how low-power pattern generation can be used to reduce test power if it exceeds TSV or probe constraints.

Figure 3.9 and Figure 3.10 show the average stuck-at current for Die 0 of the 4-die FFT benchmark with regard to changing TSV resistance and capacitance, respectively. Increases in TSV resistance resulted in an almost negligible increase in current draw, with a high TSV resistance of 5 Ω resulting in only a 0.13% increase over the baseline current draw of 300 μA. As seen in Figure 3.10, an increase in TSV capacitance had a slightly greater, though still minor, effect on stuck-at current draw, with a 500 fF TSV capacitance resulting in a 1.6% increase over the baseline current draw. These results indicate that power consumption during test is dominated by die logic and not the TSVs. The current draw is greater for capacitance increases than resistance increases because the capacitance of all TSVs in a TSV network impacts test, due to an increase in net capacitance, whereas only the resistance of TSVs in use in the TSV network impacts test.

The feasibility of the proposed method from a test-time perspective can also be discussed. The frequency at which scan-in and scan-out can take place depends on a number of factors. Scan-in speed depends on the strength of the probe-needle driver, while scan-out speed depends on the strength of the TSV driver in the sending GSF used as a scan output, which is the twin-inverter buffer shown in Figure 3.1, and the width of the GSF transmis-

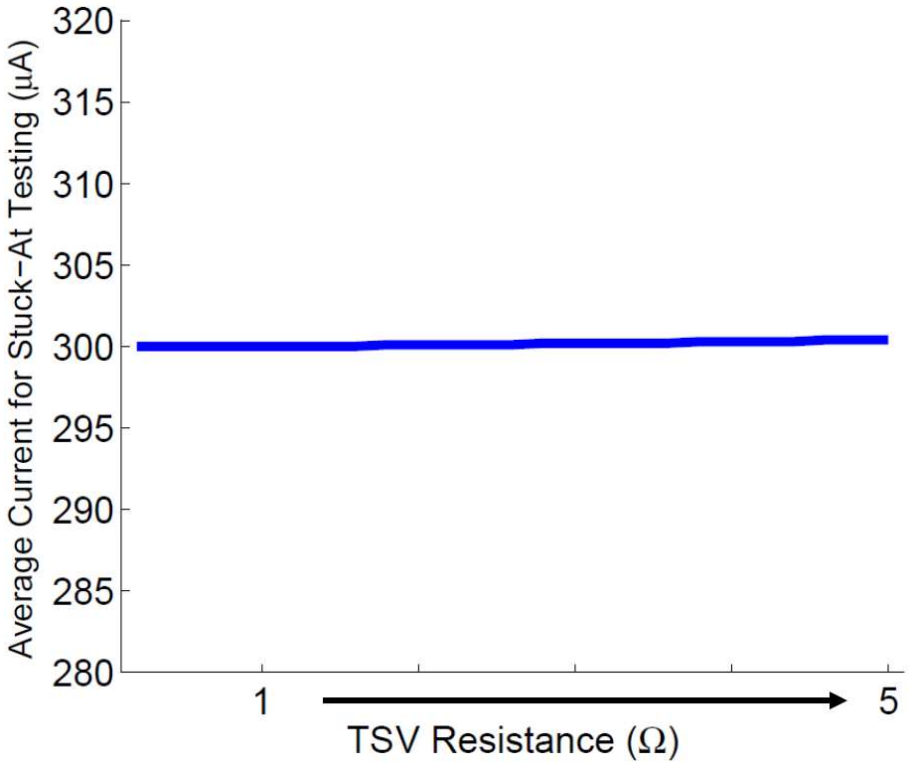


FIGURE 3.9: Change in average stuck-at current versus TSV resistance.

sion gate. Both the GSF driver and the probe driver must be able to charge and discharge the TSV network capacitance quickly enough to meet the setup and hold times of the scan flops given the test clock frequency. Therefore, the number and capacitance of TSVs in a network also influence maximum scan clock frequency.

Scan frequency simulations were performed on Die 0 of the FFT benchmark assuming a probe card with 100 probe needles [97]. The design contains 936 TSVs and it is assumed that TSV networks are roughly balanced, so a worse-case network of 11 TSVs was simulated. This results in a network capacitance of 220 fF. Simulations were performed using both the 45 nm low-power technology model and a 32 nm low-power technology model. It is assumed that drivers in the probe needle can be significantly stronger than drivers on the die itself, so simulations were performed only for the scan-out frequency as this would be the limiting factor to test time. The widths of the inverter driver and the transmission gate were varied and a maximum scan frequency was calculated by measuring the amount

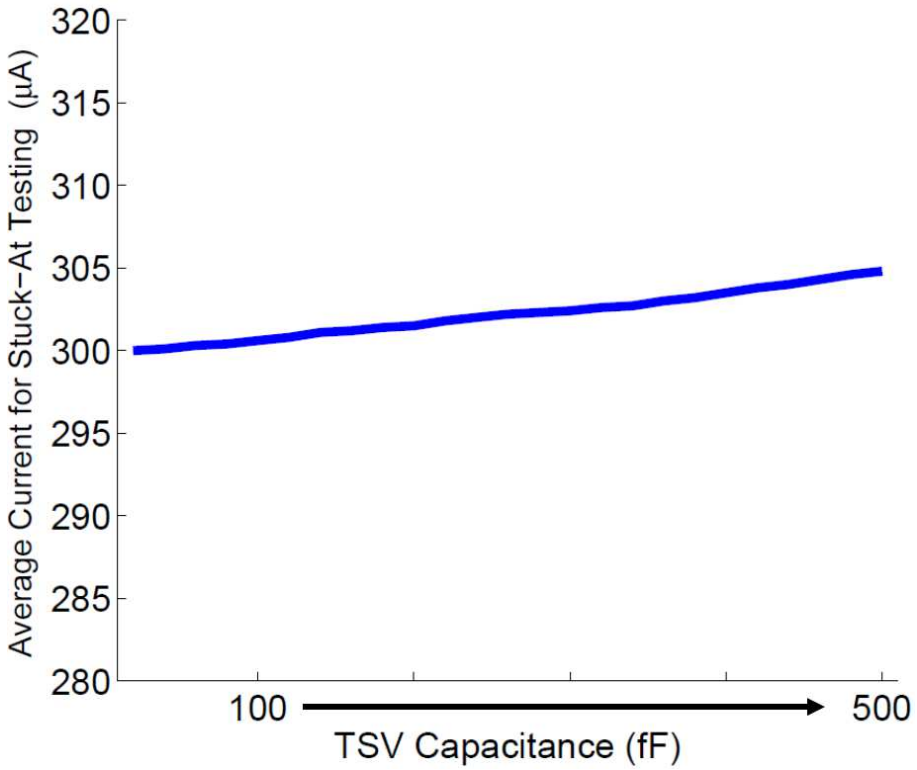


FIGURE 3.10: Change in average stuck-at current versus TSV capacitance.

of time required for a rising or falling signal to charge or discharge 75% or 25% of Vdd, respectively. The results of these simulations are shown in Figure 3.11 for the 45 nm technology node and in Figure 3.12 for the 32 nm technology node.

As can be seen from Figure 3.11 and Figure 3.12, maximum scan-out frequency depends strongly on both the width of the inverter buffer which drives data onto the TSV network and the width of the transmission gate. Small transmission gate widths limit the amount of current that can pass through the gate even in its low impedance state, drastically reducing the shift frequency even at large driver widths. Likewise, a small driver width limits its scan frequency even at large gate widths because it is incapable of sourcing or syncing enough current to quickly charge or discharge the TSV network capacitance.

As expected, at similar widths the 32 nm technology resulted in lower shift frequency when compared to the 45 nm technology, but both models showed that reasonable shift

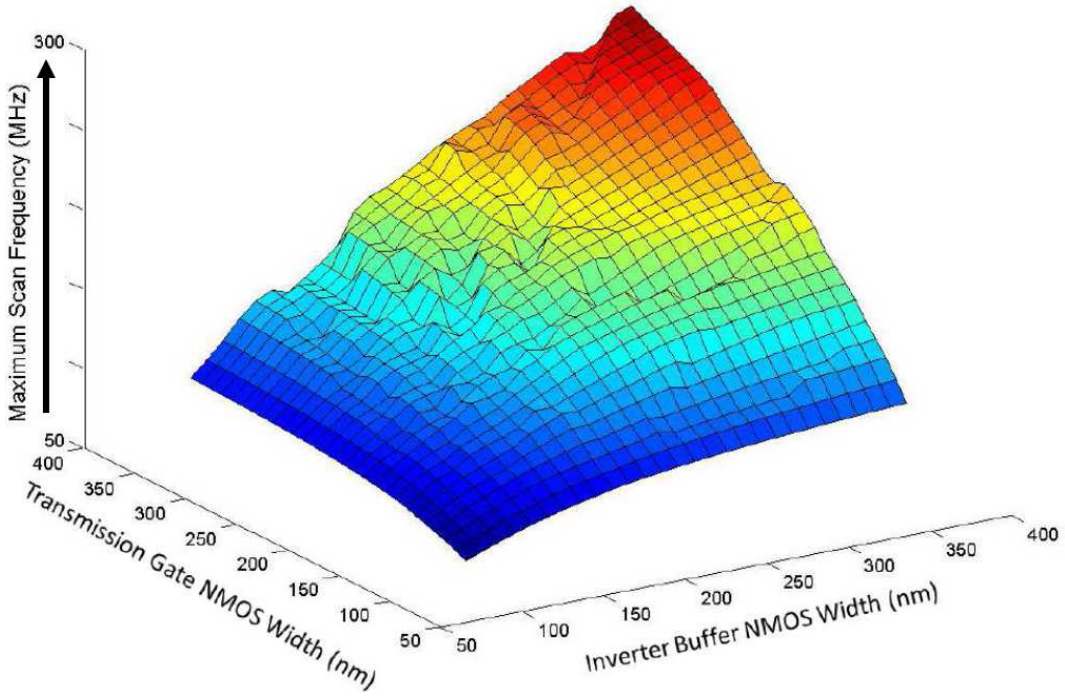


FIGURE 3.11: Maximum scan-out frequency in a 11 TSV network with varying driver and transmission gate width for a 45 nm technology.

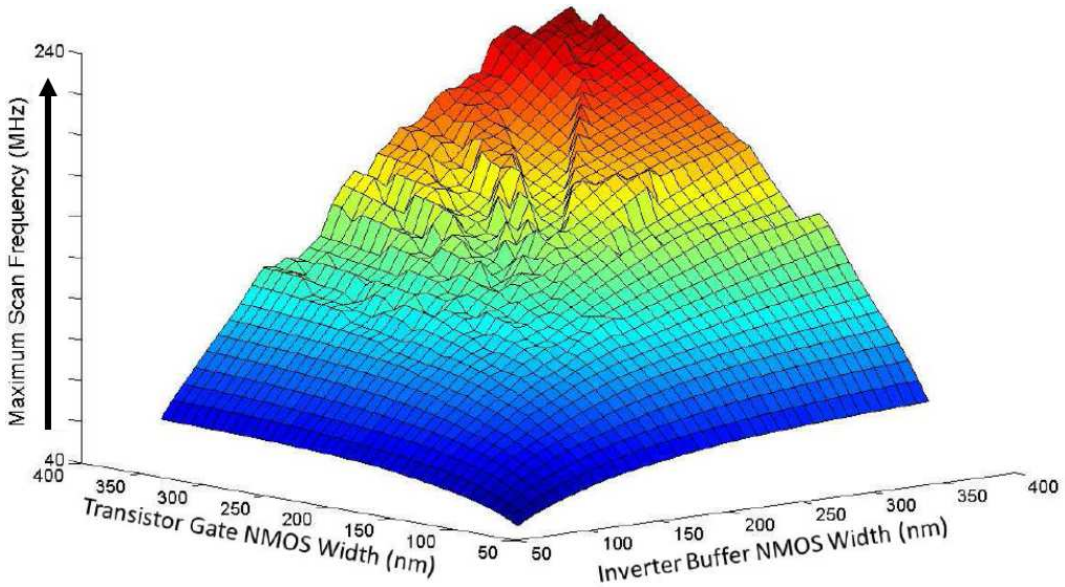


FIGURE 3.12: Maximum scan-out frequency in a 11 TSV network with varying driver and transmission gate width for a 32 nm technology.

frequencies can be achieved without significant increases in the buffer or gate W/L ratios. For example, for an NMOS W/L ratio of 2/1, at 45 nm the maximum achievable shift frequency is 71 MHz, while at 32 nm the maximum shift frequency is 44 MHz. Increasing the ratio to 3/1 results in shift frequencies of 86 MHz and 55 MHz, respectively, and at 4/1 the frequency becomes 98 MHz and 62 MHz, respectively.

The size of the NMOS and PMOS transistors in the TSV driver can be fine-tuned to coincide with the scan frequency achievable from the automated test equipment. The maximum scan frequency of the pre-bond scan test increases significantly with driver size. To achieve a scan frequency of 200 MHz at 45 nm requires NMOS and PMOS width to length rations of about 5/1. After a point, the drivers are made too large and there is a significant drawback in the higher power consumption and parasitic capacitance of the drivers. However, Figures 3.11 and 3.12 show that scan frequencies above 200 MHz are achievable without significantly larger drivers.

Next, the effect of scan configuration on test time is discussed. In Section 3.2.1, several possible scan configurations were described—one in which the scan I/Os for a scan chain are on the same TSV network (Configuration A), one in which they are on separate networks (Configuration B), and one in which multiple scan chains share the same TSV networks. While there are many possible ways that this third configuration can be constructed, this section will examine two examples, with the example from Figure 3.5(a) being Configuration C and Figure 3.5(b) being Configuration D. The scan frequency, scan chain length, number of scan chains, and number of TSV networks determine which configuration results in a lower test time. Three examples are presented to highlight this issue, with test times determined as per equations 3.1, 3.2, 3.3, and 3.4.

For Die 0 of the 4-layer FFT benchmark, if 50 scan chains are created, the result is a maximum scan chain length of 402 cells and 633 stuck-at test patterns. It is assumed that a probe card with 100 probe needles for contacting TSV networks is utilized. It is

further assumed that Configurations A and C utilize the maximum scan-in (185 MHz) and scan-out (98 MHz) clock frequencies. Configurations A and C can use different scan-in and scan-out frequencies because these two shift operations are not performed in parallel. However, scan-in and scan-out are not overlapped. Configurations C and D utilize a broadcast scan-in, where patterns for each scan chain are combined to form a single, longer pattern. In this case, Configuration A requires 4.0 ms to complete the stuck-at scan test. Configuration B, operating only at 98 MHz, requires 2.6 ms because it can scan out test responses while scanning in the next test pattern. Configuration C requires 7.2 ms, because larger test patterns and the need to serially scan out two sets of test responses requires significantly more test time. Configuration D requires 3.9 ms, because broadcast scan-in requires additional test clock cycles that require more time than the simultaneous scan-out operations. In this example, Configurations A, B, C, and D require 50, 100, 25, and 75 TSV networks, respectively. Thus, if the probe card only supported 75 TSV networks instead of 100, then Configuration D would result in the shortest test time because Configuration B would require multiple touchdowns.

On the other hand, if the die has 100 scan chains, the maximum scan chain length is 202 cells and ATPG results in 640 stuck-at patterns. Because Configuration A can handle a maximum of 100 scan chains in a single touchdown, it needs to contact the die only once. This results in a test time of 2.0 ms. Configuration B requires two touchdowns; each time it is only capable of loading and unloading 50 scan chains. It is assumed that the die is partitioned into separate test modules each of 50 scan chains such that coverage remains high. In this case, Configuration B requires 4.6 ms for test plus the time required to align the probe card and for the second touchdown. Configuration C requires 3.7 ms and a single touchdown. Configuration D requires 6.6 ms and two touchdowns.

Our final example contains 150 scan chains, with a maximum scan chain length of 134 cells and 637 stuck-at patterns. Under these conditions, Configuration A requires

Table 3.6: Percentage of TSVs that must be contacted for Die 0, as a function of the number of scan chains and scan configuration.

Scan Chain Number	% of TSVs to be Contacted	
	Config A	Config B
25	2.7	5.3
50	5.3	10.7
75	8.0	16.0
100	10.7	21.4

two touchdowns, Configuration B requires 3 touchdowns, and Configuration D requires 5 touchdowns. Therefore, test times utilizing these configurations will be significantly larger than for Configuration C. Discounting the number of touchdowns required by Configurations A, B, and D, they have test times of 2.6 ms, 2.7 ms, and 6.5 ms, respectively. Configuration C needs only 1.6 ms for test, and because it requires only a single touchdown it will be the most cost-effective configuration for performing pre-bond scan test.

As stated earlier, not all TSVs need to be contacted for die logic testing. This is an important advantage, especially if TSV or microbump damage due to probing is a concern. Table 3.6 shows what percentage of TSVs must be contacted depending on the number of scan chains present on the die and the scan configuration used. If oversize probe pads are used with the same number of scan chains, significant overhead must be incurred due to the large number of probe pads (even with test compression solutions). If the number of probe pads is limited, the test time will be higher because of constraints on the number of scan chains.

3.3 Conclusions

An extension of the TSV probing architecture of Chapter 2 has been introduced that can be used not only for pre-bond TSV test, but also for full-scan pre-bond die logic test. Scan chains are reconfigured into a pre-bond state to use TSV networks for scan I/O while preserving significant test parallelism and not requiring many oversized probe pads. HSPICE

simulation results highlight the feasibility and effectiveness of this approach. Simulations show that the current needed for testing can be supplied through TSVs and probe needle tips. Test clock frequencies also remain relatively high even with the increased TSV network capacitance. The clock frequency can be tuned by adjusting the strength of drivers of the TSV network. The area overhead of this approach was estimated to be between 1.0% and 2.9% across the dies with TSVs for the two 4-die logic-on-logic 3D stacks presented in this chapter.

4

Timing-Overhead Mitigation for DfT on Inter-Die Critical Paths

4.1 Introduction

As discussed in previous chapters, 3D ICs require both pre-bond and post-bond testing to ensure stack yield. The goal of pre-bond testing is to ensure that only known good die (KGD) are bonded together to form a stack. Post-bond test ensures the functionality of the complete stack and screens for defects introduced in alignment and bonding. In order to enable both pre- and post-bond testing, die level wrappers have been proposed in the literature [50, 115]. These die wrappers include boundary scan cells at the interface between die logic and TSVs to add controllability and observability at the TSV. Chapter 2 and Chapter 3 described how modified boundary scan cells, called gated scan flops (GSFs), can be used along with pre-bond probing for pre-bond TSV and structural logic tests. Together, published work such as [50, 97, 115, 116] shows that die wrappers can not only be used as a standard post-bond test interface, but also for pre-bond KGD tests.

Figure 4.1 gives an example of a two-die logic-on-logic stack, where TSVs are utilized as fast interconnects between logic on two separate dies. One of the drawbacks of inserting

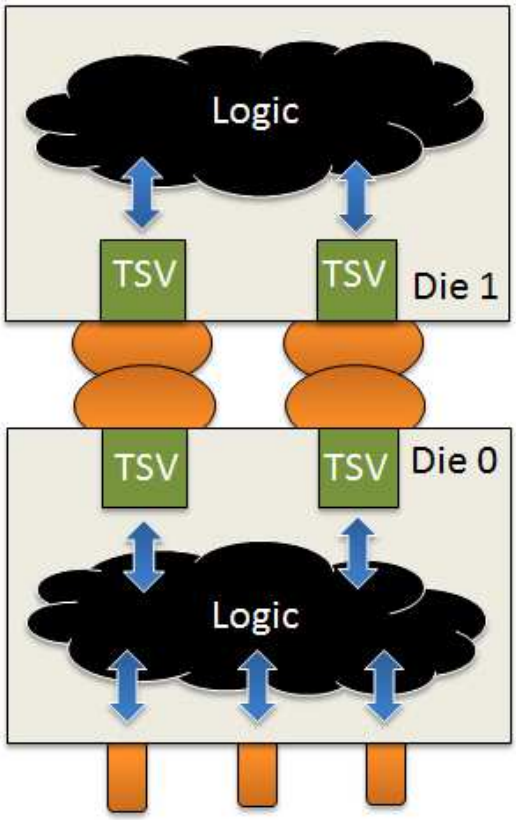


FIGURE 4.1: Example of a two-die logic-on-logic stack.

die-boundary cells is the adverse timing impact on these TSV functional paths. A reduction in latency using short TSV interconnects is one of the key drivers for 3D technology in both logic-on-logic and memory-on-logic applications. Adding boundary flops to TSVs between two layers adds two additional clocked stages to a functional path, which would otherwise not exist in a two-dimensional (2D) design. Bypass paths can be added to the boundary scan cells to multiplex the functional input between being latched in the flop or being output directly to/from the TSV. During functional mode, the bypass path is active and a signal traveling across dies is never latched in the boundary register; however, the bypass path still introduces additional path latency. It is important to note that additional latency is added by test architectures not just to logic-on-logic stacks, but also to memory-on-logic and memory-on-memory stacks that contain dies with die wrapper boundary registers.

Retiming is an algorithmic approach to improving multiple aspects of circuit design

post-synthesis by moving the positions of registers with respect to combinational logic while preserving circuit functionality [117]. Retiming methods have been developed to target a number of circuit features, including minimizing clock period [118], reducing power consumption [119], and improving testability by reducing feedback dependency [120]. Previous literature focuses on retiming in 2D circuits after synthesis but before test insertion. This is because before synthesis, the RTL behavioral circuit model does not contain register locations.

This chapter extends the concept of retiming to utilize retiming algorithms to recover the latency added by boundary scan cell bypass paths in a 3D IC. Retiming in this case is performed after synthesis and 3D boundary cell insertion. Two-dimensional retiming methods can be reused after test-insertion by fixing the location of die wrapper boundary flops so that they remain at the logic/TSV interface during retiming. This requirement ensures that die logic is not moved across dies when retiming is performed on a complete stack. A bypass path is added to each register so that in functional mode, data does not need to be latched, thereby replacing extra clock stages with added latency. Retiming is then performed to recover the added latency of the bypass along the TSV path. An additional step is added to standard retiming methods, whereby complex logic gates that prevent register relocation due to latency violations are decomposed into their basic logic cells from the circuit library. Furthermore, a logic redistribution algorithm, whereby certain logic cells on a critical path can be moved from one die to its adjacent die, is applied to die-level retiming in order to provide better delay reduction results. This step can allow for a redistribution of circuit delay to non-critical paths from low-slack paths during retiming.

4.1.1 The Impact of Die Wrappers on Functional Latency

Die wrappers are a means for standardizing the test interface at die level for pre-bond and post-bond test. Die wrappers leverage many of the features of the IEEE 1500 standard wrapper for embedded cores [25]. A Wrapper Instruction Register (WIR) is loaded with

instructions to switch the wrapper between functional mode and test states. Switch boxes multiplex between a post-bond test mode and a reduced pin-count pre-bond test mode. A Wrapper Boundary Register (WBR) consists of scan flops at the die interface between the next die in the stack (or the primary inputs and outputs in the case of the lowest die in the stack) and the die's internal logic.

The WBR provides the means to enhance pre-bond test coverage. During pre-bond scan test, die boundary registers provide controllability and observability at each TSV. Without boundary registers, untestable logic exists between die-internal scan chains and TSVs. In a 3D benchmark circuit without boundary registers, stuck-at and transition fault coverage for a die was shown to be as low as 45% [116]. With the addition of die boundary registers on the same die, the fault coverage for the die was shown to be above 99%.

Although boundary registers, with or without GSFs, are necessary for pre-bond fault coverage, prior work has not addressed the latency overhead that boundary registers contribute to the signal paths that cross die boundaries. One of the advantages of 3D stacked ICs is the reduction in interconnect delay gained by utilizing short TSVs in place of long 2D interconnects. For example, in a memory-on-logic stack, latency on the TSV path directly impacts memory access time. In logic-on-logic stacks, least-slack paths are split between die layers to increase operating frequency [61, 62].

In 3D circuits, especially logic-on-logic stacks, the latency improvements of 3D integration are most relevant when critical paths are split between two or more dies. The use of low-latency TSVs instead of long 2D interconnects increases the slack of critical paths and therefore allows for a faster functional clock. Unlike wrapping 2D test modules, for example as per the 1500 standard, if die-level WBRs are added to a 3D circuit, they must be unavoidably on critical paths. This is because they will exist on every path that utilizes a TSV or has to interface with a TSV on another die. This removes slack from the same critical paths that are split between dies. Through retiming, slack can be redistributed

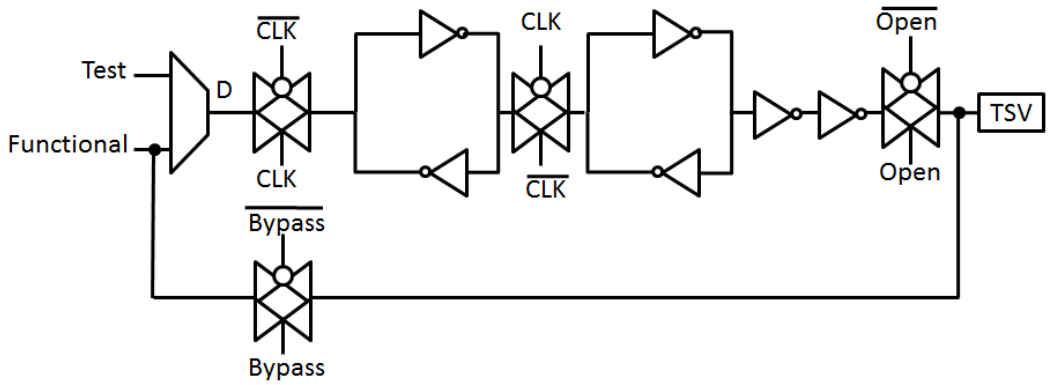


FIGURE 4.2: An example gate-level design of a gated scan flop with a bypass path.

throughout a circuit in such a way that cross-die critical paths can meet their pre-wrapper-insertion timing requirements.

In this chapter, it is assumed that a bypass path is added to the GSF boundary registers of the die wrapper, as shown in Figure 4.2. In functional mode, the bypass signal is asserted to route the functional input directly to the TSV, circumventing the need to latch functional data in the boundary registers. However, even when using a bypass path, the addition of boundary registers still increases latency on TSV paths. This chapter introduces methods to recover this latency through register retiming.

4.1.2 **Register Retiming and its Applicability to Delay Recovery**

Register retiming is a post-synthesis, algorithmic approach to circuit optimization first introduced in [117]. While retiming can be used to target a number of applications, including reducing power consumption [119] or enhancing testability [120], most retiming algorithms focus on clock-period reduction. During retiming, slack is moved from paths with excess slack to least-slack paths without altering the function of the circuit by moving the location of registers in relation to combinational logic. Transforms are defined to limit how and where registers can be moved by retiming algorithms to preserve circuit functionality [118].

In the simplest algorithms, retiming is performed by representing a circuit as a directed graph where vertices represent logic gates and edges represent connections between logic

elements [117]. The weight of each edge represents the number of registers present between two logic elements. The propagation delay through each vertex is calculated as a function of the number of logic elements through a directed path that does not contain a register. During retiming, registers are moved between graph edges to reduce the delay values for graph paths, a step that leads to a reduction in the clock period.

While most retiming algorithms are used after synthesis, accurately approximating path delay from a structural circuit definition is difficult without placement. Furthermore, the movement of registers can alter interconnect lengths. These changes are difficult to account for with only a structural description of a circuit. The authors of [121] carry out retiming after performing an initial placement in order to model interconnect delay more accurately. During retiming, it is assumed that combinational logic placement remains unchanged but that a retimed register would be placed at a range around the geometric center of its fan-in and fan-out cones based on slack.

Retiming has been utilized in the literature to recover the delay added to a path where a standard flip-flop is converted into a scan flip-flop [122]. The extra delay is caused by the addition of a multiplexer to select between a functional and test input. Retiming is then performed to move the flop portion of the scan flop in relation to the added multiplexer. This method is not applicable for die-boundary registers, however, as these registers cannot be moved. Furthermore, in functional mode, the wrapper boundary registers are not used to latch data and are instead set to a bypass mode that is independent of the multiplexer required to select between functional and test inputs.

In this chapter, retiming is utilized after synthesis and test-architecture insertion in a 3D stack to recover the additional delay added to a TSV path by boundary registers. In order to present simulations results, the retiming algorithm of Synopsys Design Compiler is used, which performs retiming first to minimize clock period and then to minimize register count. This is followed by a combinational logic optimization step in view of changes to loading

on cells and critical paths that may have occurred after the movement of registers. An additional step of logic decomposition is added to the retiming procedure—if a complex logic gate is preventing further retiming on a critical path that does not satisfy slack constraints, then the gate is decomposed into simple gates from the cell library that together perform the equivalent Boolean function. Retiming is then performed again to determine if logical decomposition further reduces latency. Furthermore, logic can be shifted between dies to achieve better retiming results.

The rest of this chapter is organized as follows. Section 4.2 provides a motivating example for post-test-insertion retiming and outlines the retiming methods used in this chapter. Subsection 4.2.1 provides a detailed description of the test insertion and retiming flow used to recover delay overhead. Subsection 4.2.2 discusses the design flow and algorithm used to redistribute logic during die-level retiming. Subsection 4.2.3 presents retiming results for two-, three-, and four-die stacks with TSVs in logic-on-logic 3D benchmarks as well as a modular processor benchmark. Finally, Subsection 4.3 concludes the chapter.

4.2 Post-DfT-Insertion Retiming in 3D Stacked Circuits

Figure 4.3 illustrates retiming performed on an example circuit on the bottom die of a 3D stack. Figure 4.3(a) shows the circuit prior to wrapper insertion and retiming. Inputs *A*, *B*, *C*, and *D* are primary inputs. The circuit contains four flip flops (FF), labeled f_1 through f_4. Several logic gates exist between the flops and the TSV that connect to the next die in the stack. These are part of an incomplete logic circuit without the connected logic in the other stack tiers and are not observable during pre-bond scan test. Basic and complex logic cells are annotated with a delay value representing delay through the cell as a fraction of the clock cycle. For example, a value of 0.5 means that rise and fall delays from the inputs to the output of a logic gate is half of one clock period. This example, and the timing data presented, is simplified for demonstration and does not consider additional data such as

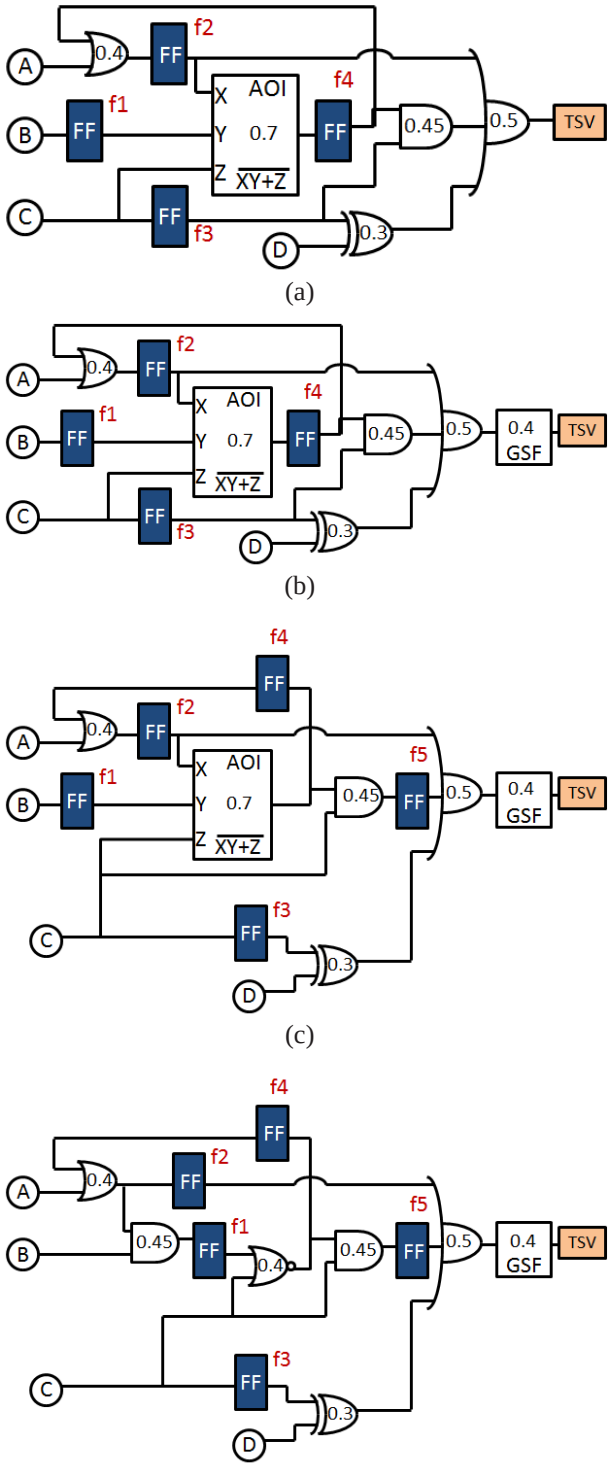


FIGURE 4.3: Boundary register insertion and retiming (including logic decomposition): (a) design before register insertion, (b) design after register insertion, (c) first register movement, (d) logic decomposition and second register movement.

interconnect delay, fan-out, clock skew, or flop setup and hold times. It is assumed for this example that no path from a primary input or flop to another flop or to the TSV can have a delay greater than one clock period. In Figure 4.3(a), the longest delay is 0.95 of a clock period from flop f_4 to the TSV, so this timing constraint is satisfied.

In order to provide controllability and observability, and to present a standard test interface, a wrapper is next added to the die. For the example circuit, a boundary GSF is inserted between the die logic and the TSV as shown in Figure 4.3(b). In its bypass mode, the GSF contributes an additional 0.4 delay to any path through the TSV. For the path from flop f_2 to the TSV, this is acceptable, because the overall path delay is only 0.9 clock cycles. However, three paths now violate the timing constraints—the paths from f_3 and f_4 to the TSV at a delay of 1.35 clock cycles each, and the path from *D* to the TSV at a delay of 1.2 clock cycles.

Retiming is performed to recover the latency added to paths through the TSV by moving registers to ensure that no path violates the timing constraint. In the first retiming step, flops f_3 and f_4 on the AND gate inputs are pushed forward to its output as shown in Figure 4.3(c). The AND gate's inputs now come directly from *C* and the complex AOI gate without latching, and its output is instead latched. This requires the addition of an extra flop f_5 at the AND gate's output. The extra flop is created because a flop (f_4) is still required on the feedback loop from the AOI gate to flop f_2 in order to preserve functionality. Likewise, a flop (f_3) is still required between *C* and the XOR gate. Now the path from flop f_5 to the TSV satisfies the timing constraint with a delay of 0.9 cycles. On the other hand, the path from flop f_2 to flop f_5 now violates the constraint with a delay of 1.15 clock cycles.

At this point, further retiming cannot satisfy the timing constraint. If f_2 is pushed to the output of the AOI gate, then the path from *A* to f_2 will violate timing constraints. If f_5 is moved to the inputs of the AND gate, then the circuit returns to its state in Figure 4.3(b). The AOI gate does not allow any further retiming, so in an effort to provide

greater flexibility to the retiming algorithm, it is decomposed into basic logic gates from the cell library—an AND and NOR gate. Compared to its decomposed gates, the AOI gate provided for a shorter delay for the Boolean function it performed and required less die area. However, the basic logic gates allow more avenues for retiming. Flops f_1 and f_2 can now be pushed from the inputs of the to the output of the AND gate added after decomposition, as shown in Figure 4.3(d). All paths now satisfy the timing constraint and retiming is complete. Although there is less slack in the circuit overall after the addition of the boundary GSF and the decomposition of the AOI gate, the slack is redistributed among all paths so that no single path experiences a timing violation.

It is important to note that the delay values given for GSFs and gates in the motivational example is only meant to illustrate why and how retiming may be utilized. These do not present realistic delay values. In simulation, the delay of the GSF in bypass mode that is added to each die on an inter-die path amounted to 3-4 inverters designed in the same technology node. Therefore, a path that spans one bonded interface between dies has an additional delay equivalent to 7 inverters. If a path crosses two bonded interfaces, the additional delay is equivalent to 10-11 inverters, and so forth. The additional delay caused by the GSFs in any given design will depend on the design of the GSF and the manufacturing technology used.

The rest of this chapter refers to a metric designated as "% delay recovered" to denote the quality of retiming results. This does not refer to the percentage of recovered paths. The delay recovered is calculated from the actual operational frequency of the circuit. Three minimum timing values are produced for a given design. The first (*original*) value is the frequency of the circuit before DfT insertion. The second (*inserted*) value is the frequency of the circuit after DfT insertion, which in all examined benchmarks was less than the original frequency due to slack violations. The third (*retimed*) value is the frequency of the circuit after retiming, after which slack is redistributed in an attempt to allow the circuit to

run at the original frequency value. From these variables, the equations

$$a = inserted - original$$

$$b = retimed - original$$

are defined and % delay recovered is calculated as

$$\frac{a-b}{a} \cdot 100.$$

When *inserted* equals *retimed*, then 0% of the delay is recovered. When *retimed* equals *original*, then 100% delay is recovered.

4.2.1 Method for Die- and Stack-level Retiming

Retiming can be performed either at the die- or stack-level. In the example of Figure 4.3, retiming was performed at the die-level, or in other words, on a single die of a 3D stack without knowledge of the circuitry on other dies. Die-level retiming allows greater control over the redistribution of slack throughout the stack. For example, a designer may not want to move any registers on a particular die *D* but would still like to recover the additional latency of adding a wrapper to that die. In this case, additional (dummy) delay can be added to the TSV paths on the dies adjacent to *D*. Retiming can then be performed on the adjacent dies in an attempt to recover the additional path delay due to the wrapper cells in the adjacent dies and the wrapper cells of *D*.

A flowchart of the steps required for post-wrapper insertion die-level retiming is shown in Figure 4.4. First, the design is synthesized into a structural circuit definition. In die-level retiming, paths crossing the TSV interface are incomplete and the total delay across these paths cannot be considered. Because paths crossing die boundaries are likely to be the least-slack paths in a 3D stack [61, 62], the clock period for the stack may be too large to provide a tight timing constraint when considering a single die. In order to determine

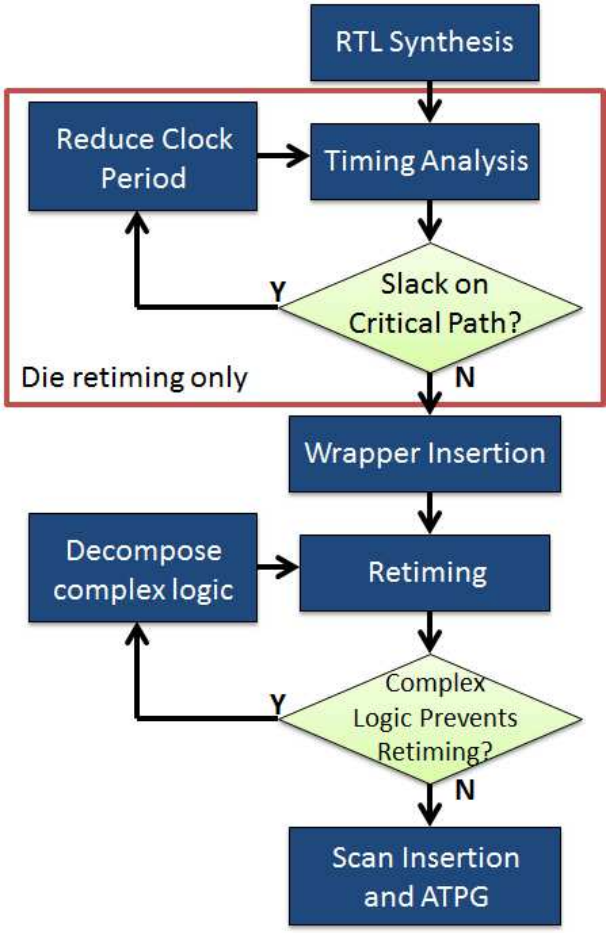


FIGURE 4.4: Flowchart for retiming of a 3D stack at either die- or stack-level.

an appropriate timing target, timing analysis is performed to identify the amount of slack on the least-slack path of the die. The target clock period for retiming is incrementally reduced until the least-slack path has no positive slack. Wrapper insertion is then performed, adding delay to the TSV paths equal to the bypass path through a boundary GSF. During retiming, boundary GSFs are fixed so that the retiming algorithm does not consider them as movable registers nor does it attempt to move logic or other registers past the GSFs. Timing information for the logic gates, flip flops, and GSFs are extracted from the cell library. After the retiming algorithm has executed, timing analysis is again performed to determine if all paths in the die satisfy the target timing constraint. If they do not, the path that has the most negative slack is examined to determine if complex logic gates on the path may be

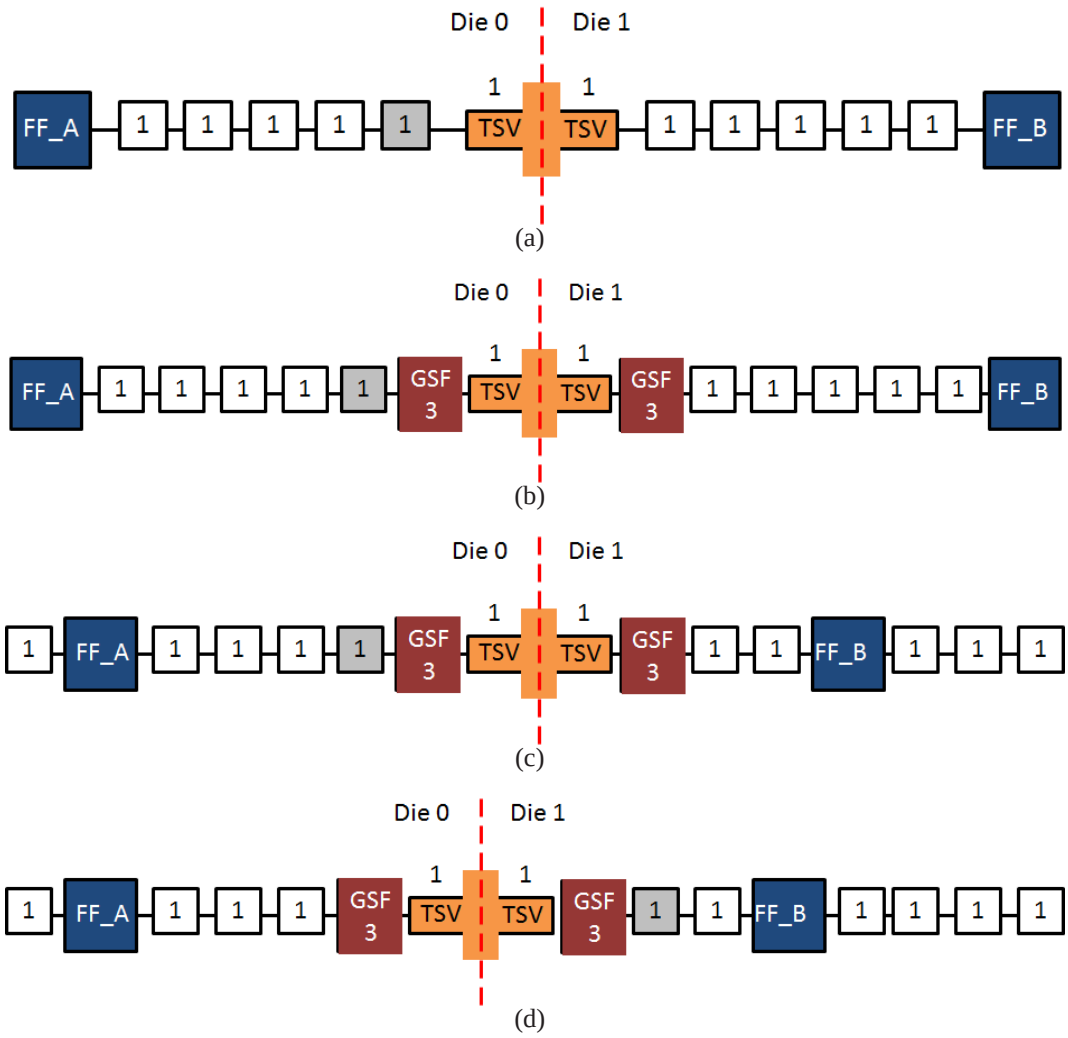


FIGURE 4.5: Logic redistribution example showing: (a) design before register insertion, (b) design after register insertion, (c) die-level retiming before logic redistribution, (d) die-level retiming after logic redistribution.

preventing retiming. If complex logic gates are preventing retiming, they are decomposed into simple logic cells from the cell library and retiming is performed again. This process continues until all paths meet the timing target or no complex logic gates are restricting retiming. Finally, scan insertion and ATPG are performed for the die.

While die-level retiming does not consider the total path delay for paths that cross die boundaries, stack-level retiming can exploit this added degree of freedom. In stack-level retiming, the complete stack is retimed as a monolithic entity. The boundary GSFs

are once again fixed during retiming to prevent the movement of logic from one die to another. During stack retiming, the intended clock frequency for the stack can be used as a timing target because all circuit paths are known. While die-level retiming provides more control over slack redistribution, stack-level retiming provides greater leeway to the retiming algorithm. To illustrate this point, consider a three-die stack where each boundary GSF adds an additional 0.2 clock period delay to a path. A path crossing all three dies from the bottom to the top of the stack would experience an additional 0.8 clock period delay after wrapper insertion—0.2 delay from both the bottom and top dies and 0.4 delay from the middle die. In die-level retiming of all the dies in the stack, 0.2 clock periods of delay would have to be recovered in each of the top and bottom dies of the stack, and 0.4 delay would have to be recovered in the middle die. In stack-level retiming, extra slack can be redistributed to the path regardless of which die can supply the slack. To recover the entire 0.8 clock period of delay added to the path, 0.1 delay can be recovered from the lowest die, 0.3 from the middle die, and 0.4 from the top die.

The flow for stack-level retiming is similar to the die-level retiming flow of Figure 4.4. Because the clock period of the stack is known, it can be used as a timing constraint for retiming. For this reason, no timing analysis or tightening of the clock period must be performed before wrapper insertion. Retiming, logic decomposition, scan insertion, and ATPG are performed as they were for die-level retiming.

The drawback of die-level retiming—namely that a die-level retiming may result in inter-die paths that continue to violate timing constraints after wrapper insertion when a stack-level retiming solution would satisfy timing constraints—motivates the need for an enhancement to die-level retiming. In this chapter, a logic redistribution algorithm, which is described in detail in Subsection 4.2.2, is introduced to better make use of excess slack during die-level retiming. Consider, for example, the inter-die path shown in Figure 4.5 between flip-flop A (FF_A) on Die 0 and flip-flop B (FF_B) on Die 1 in a hypothetical

stack. Figure 4.5(a) shows the example path before wrapper insertion or retiming. The full path consists of 10 logic cells, each with a delay of 1 ns, and two TSVs, each also with a delay of 1 ns. The entire path has a delay of 12 ns, and no path in the circuit can have a delay greater than 13 ns. Each die contains five logic cells and one TSV of the path. There exists enough slack on Die 0 such that FF_A can be moved toward its TSV by one logic cell, and on Die 1 there is enough slack for FF_B to be moved toward its TSV by four logic cells. Figure 4.5(b) shows the path after wrapper cell insertion, where each wrapper cell (GSF) contributes 3 ns of delay in functional mode. Now, the full path requires 18 ns for signal propagation, well above our limit of 13 ns. During die level retiming of Die 0, as shown in Figure 4.5(c), FF_A can only be moved to regain 1 ns of delay, thus violating its timing constraints. For Die 1, FF_B will be moved to regain 3 ns of delay, recovering all of the additional delay. Thus, with die-level retiming, the path violates the overall timing constraint by 1 ns, even though there is additional slack on Die 1 that would be appropriately allocated during stack-level retiming. A die-level solution to this problem would be to move one logic cell, the grayed cell, from Die 0 to Die 1 by pushing it across the die boundary as shown in Figure 4.5(d). Now, when die-level retiming is performed, Die 1 will recover 4 ns of delay as in Figure 4.5(d) and the timing constraint for the path will be met.

In all of the benchmarks presented in this chapter, the circuits have been partitioned for timing optimization. Before DfT insertion, the most critical path in each benchmark is an internal path in a die, since paths that would be critical in the stacked circuits utilize TSVs where available and are no longer critical paths. After DfT insertion, at least one inter-die path becomes the most critical path of the design and causes timing violations that were not present in the pre-test-insertion design. The retiming method described and utilized in this chapter does not attempt delay recovery on every path. It is an iterative process where the least-slack path that violates the timing constraint is first retimed, and then the next, and so

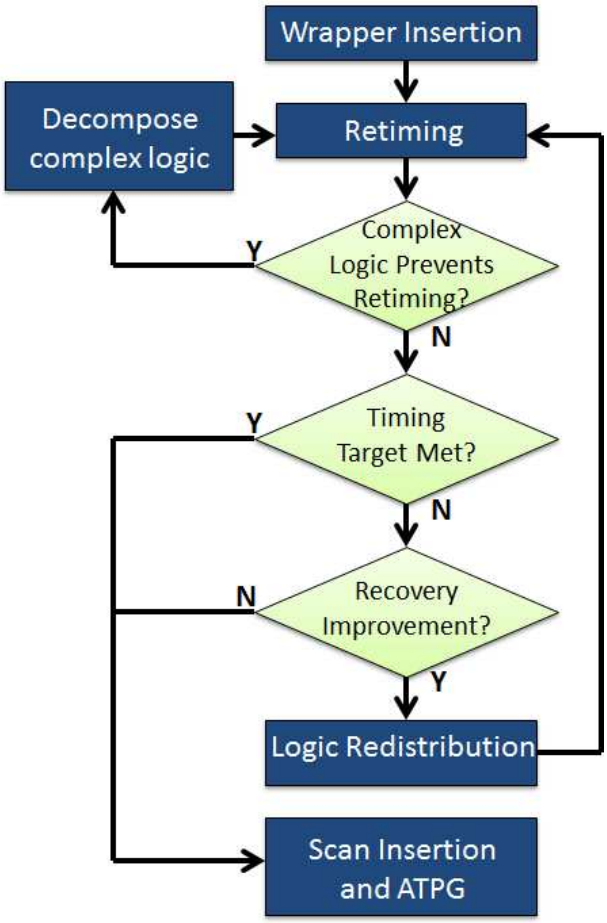


FIGURE 4.6: Flowchart for die-level retiming with logic redistribution.

forth until the timing constraint is no longer violated or the current least-slack path cannot be made to satisfy the timing constraint. Paths that do not violate the timing constraint after DfT insertion are not affected except in certain cases where they may be altered to retime a critical path.

4.2.2 **Algorithm for Logic Redistribution**

Figure 4.6 shows the insertion of the logic redistribution algorithm into the die-level retiming flow. A logic redistribution algorithm is inserted after complex logic decomposition. If timing targets are met, then no logic redistribution is necessary and scan insertion and ATPG can take place. If, on the other hand, timing targets are not met after die-level retiming, then logic redistribution can be performed in an attempt to achieve additional delay

recovery. Logic redistribution is performed, followed by retiming and logic decomposition if necessary, until either no recovery improvement is achieved, in which case scan insertion and ATPG are performed on the last netlist that resulted in delay recovery improvement, or until all timing constraints are met.

For any given iteration, the logic redistribution algorithm attempts to shift logic one cell at a time on the most critical inter-die path from one die to its adjacent die. Retiming is then performed again, and if necessary another iteration of the logic redistribution algorithm takes place. This allows the algorithm to target either the same path for further improvement or a new path if that path becomes the predominant critical path.

Algorithm 2 LogicRedis(*DieA, DieB, CritPathA, cellLibrary*)

```
cellToMove = getCell(DieA, CritPathA);
fanIn = getFanIn(DieA, cellToMove, cellLibrary);
controlSig = getControlSignal(cellToMove, cellLibrary);
All variables are initialized at this point.
if fanIn OR controlSig then
   return {DieA, DieB};
else
   {DieA, DieB} = moveLogic(DieA, DieB, CritPathA);
end if
return {DieA, DieB};
```

The pseudocode for the logic redistribution algorithm (LogicRedis) is given in Algorithm 2. The algorithm requires as input the netlists for two dies, *DieA* and *DieB*, that are adjacent to another and share the critical path *CritPathA*. The library of standard cells, *cellLibrary*, is also needed. The algorithm attempts to move one logic cell from the critical path on *DieA* to *DieB*.

The LogicRedis begins by identifying the logic cell that must be moved through the **getCell** function. This function takes the netlist for *DieA* and the critical path identification and finds the logic cell on the critical path on *DieA* closest to the TSV and returns its unique name and standard cell type, which are assigned to the data structure *cellToMove*. It is then necessary to determine whether or not the cell can be moved to *DieB*.

Logic redistribution is limited by the fact that TSVs between the adjacent dies cannot

be added to or subtracted from the stack design. Whether a logic cell is being moved from a transmitting die to a receiving die or vice versa, a logic cell with a fan-in greater than one or that requires a control signal (such as a transmission gate) cannot be moved without adding or removing TSVs. There are two exceptions to this rule. The first is a logic cell with fan-in greater than two but with all inputs except for the TSV input tied to either V_{dd} or ground. The second is a logic cell with a control signal that is tied to V_{dd} or ground. The **getFanIn** function returns true if the logic cell (*cellToMove*) passed to it has a fan-in greater than one and at least two fan-in nets are not connected to V_{dd} or ground. The **getControlSignal** function returns true if the passed logic cell requires a control signal and that signal is not tied to V_{dd} or ground.

If moving the logic cell would violate our constraint that TSVs cannot be added or removed—that is, if either $fanIn$ or *controlSig* is true—then the input netlists *DieA* and *DieB* are returned without modification. If, instead, the logic cell can be safely moved from *DieA* to *DieB*, then the **moveLogic** function performs the logic redistribution and returns new netlists for *DieA* and *DieB*. These new netlists are then returned in order to perform die-level retiming once again such that the new netlists may lead to better delay recovery.

The **LogicRedis** algorithm terminates under two circumstances. The first is after moving a single logic cell, in which case both $fanIn$ and *controlSig* are false and the *DieA* and *DieB* netlists are updated with the moved logic and returned. If the algorithm is terminated in this way, then retiming is performed again and the result is checked to see if the timing target is met or if there is recovery improvement as per Figure 4.6. The second termination criterion is that no logic cell can be moved without adding or subtracting a TSV, in which case either or both of the $fanIn$ and *controlSig* variables will be true. In this case, the *DieA* and *DieB* netlists are returned unchanged and no further retiming is possible.

The algorithm for logic movement is of complexity $\mathcal{O}(n)$ with regard to the size of the

netlist of *DieA*. The **getCell** function is $\mathcal{O}(n)$ as it must search the netlist for the cell to be retimed, whose inputs and outputs match those of the critical path identification. The functions **getFanIn** and **getControlSignal** are constant-time, as they only require checking the inputs and outputs of a standard cell definition. Likewise, the **moveLogic** function is constant time as it requires adding or removing the previously identified logic cell and two wire definitions from each netlist.

The limitation on adding or removing TSVs during logic redistribution limits how close die-level retiming can come to stack-level retiming. Consider a worst-case retiming scenario in a two-die stack where a cross-die critical path exists such that most of the path logic is on Die 0 while only a single logic cell of the path is on Die 1. During stack level retiming, the burden of recovering the timing latency of the path is almost entirely on Die 0, while the retiming of Die 1 may not consider the path to be critical at all if only a small portion of the path is on Die 1. In this case, it is beneficial to move some of the path logic from Die 0 to Die 1 to evenly distribute the retiming burden between the dies, or to move all of that path logic to Die 1 if Die 1 has a significant amount of slack that can be moved onto the path while Die 0 has none. If the entire path across both dies consists of logic cells that satisfy our TSV constraints—that is, if any or all cells can be moved between the dies—then die-level retiming can achieve the same results as stack-level retiming for that particular path. This is because a logic distribution can be found such that each die is capable of providing all the extra slack it can to that path. In real circuits, however, it is likely that only some logic on some critical paths will meet the TSV requirements for logic redistribution, and therefore stack-level retiming will outperform die-level retiming.

To implement the retiming flows in practice, the retiming tool utilized in this chapter (Synopsys Design Compiler) reports which gate on the least-slack path is preventing further retiming when a timing target is not met. If the timing target is not met and a gate is identified, the decomposition algorithm implemented in this dissertation checks to see if

the gate is a complex gate. If so, the gate is decomposed and retiming is attempted again. If no gate is flagged and the timing target is met, then 100% delay recovery is achieved and the retiming algorithm terminates. If the timing target is not met, but there was a recovery improvement over the previous retiming result, then logic redistribution is attempted. Only in the case where logic decomposition is not necessary and logic redistribution has been performed with no delay recovery improvement over the previous retiming attempt does the retiming algorithm terminate without 100% delay recovery.

4.2.3 **Effectiveness of Retiming in Recovering Test-Architecture-Induced Delay**

In this section, the effectiveness and impact of post-DfT-insertion retiming is examined in the context of several benchmark circuits that have been partitioned into two-, three-, and four-die stacks. Two benchmarks are used—the performance-optimized data encryption standard (DES) circuit and the Fast Fourier Transform (FFT) circuit from the IWLS 2005 OpenCore benchmarks [106]. The DES circuit contains 26,000 gates and 2,000 flops, while the FFT circuit contains 299,273 gates with 19,962 flops. They contain no embedded, wrapped modules or black boxes, so retiming can target any gates or flops in the design. The DES circuit was partitioned into two-, three-, and four-die stacks and the FFT circuit was partitioned into two- and four-die stacks using the Nangate open cell library [123] and a placement engine that optimized timing.

A third OpenCore benchmark—the OpenRISC1200 (OR1200) 32-bit scalar RISC processor—is used as a benchmark with modules that cannot be retimed. The OR1200 has a 5 stage integer pipeline and IEEE 754-compliant single precision floating point unit. The OR1200 implementation used in this chapter contains a one-way direct-mapped 8 KB data cache and one-way direct-mapped 8 KB instruction cache. The processor contains 15,000 cells with 1850 flip-flops and utilizes a 250 MHz clock frequency.

No 3D modular benchmarks yet exist in the open literature, so a benchmark for this chapter was created by partitioning the OR1200 processor across two dies. Many mod-

ules were fixed, meaning that they could not be retimed, and were placed in their entirety on one die or the other. Fixed modules generally include modules where retiming may move critical registers that can interfere with timing closure, such as the decode, execute, memory, and writeback stages of the processor pipeline. The modules that were allowed to be retimed are the debug unit, exception logic, floating point unit, freeze logic, instruction fetch, load/store unit, programmable interrupt controller, power management, and SPR interface. These modules were split across the dies in such a way that logic that may be retimed existed on both sides of each TSV. This was done because, barring interconnects, no logic existed between modules. The modules that may be retimed in this benchmark take the place of complex logic between modules. If a fixed module were on either side or both sides of a TSV belonging to a critical path, then retiming may have no impact on delay recovery.

To approximate the bypass mode delay of a GSF, a GSF was created in HSPICE using a low-power 45 nm process [76]. Input-to-output rise and fall times were determined using simulation. To model wrapper insertion, a number of inverters from the cell library were added before each TSV and each face pad for TSV bonding to approximate the delay of bypass mode GSFs. All of the data given in this section utilizes the inverter delay approximation for the GSF. In all benchmarks presented, the delay of the GSF was significant enough to cause slack violations after DfT insertion.

Table 4.1 shows retiming results for the (a) two-, (b) three-, and (c) four-die DES stacks. The columns present data first for die-level retiming, starting with the lowest die in the stack (Die 0) and moving to the highest die in the stack. The last column shows data for stack-level retiming. The first row of the table lists the percent of delay recovered during retiming on the least-slack TSV path. A value of 100 means that all delay on all TSV paths was recovered. The second row indicates the area overhead of DfT insertion as a percentage of total area, or the sum of cell area and interconnect area. Interconnect area is estimated

Table 4.1: A comparison of delay, area, and pattern count results for die- and stack-level retiming for the DES logic circuit partitioned across (a) two, (b) three, and (c) four dies.

(a) Two-Die Stack

	Die 0	**Die 1**	**Complete Stack**
% Delay Recovered	100	100	100
% Area Overhead Post-Wrapper Insertion	16.3	16.4	18.7
% Area Overhead Post-Retiming	12.4	13.4	16.6
% Change in Pattern Count	−7.4	3.0	10.9

(b) Three-Die Stack

	Die 0	**Die 1**	**Die 2**	**Complete Stack**
% Delay Recovered	100	100	100	100
% Area Overhead Post-Wrapper Insertion	20.0	29.8	26.2	26.2
% Area Overhead Post-Retiming	19.7	29.1	24.2	25.0
% Change in Pattern Count	3.3	6.2	−1.4	12.7

(c) Four-Die Stack

	Die 0	**Die 1**	**Die 2**	**Die 3**	**Complete Stack**
% Delay Recovered	100	100	60	100	100
% Area Overhead Post-Wrapper Insertion	22.7	35.5	35.6	28.5	31.9
% Area Overhead Post-Retiming	16.1	34.1	34.6	25.9	27.5
% Change in Pattern Count	−2.5	−4.2	0.8	5.3	8.1

from wire load models included with the cell library. The third row indicates the percentage of total area overhead after retiming. The last row presents the percentage change in pattern count for stuck-at patterns between ATPG before DfT-insertion and ATPG after retiming. A negative value indicates that pattern count decreased. Table 4.2 shows similar results for

Table 4.2: A comparison of delay, area, and pattern count results for die- and stack-level retiming for the FFT logic circuit partitioned across (a) two and (b) four dies.

(a) Two-Die Stack

	Die 0	**Die 1**	**Complete Stack**
% Delay Recovered	100	100	100
% Area Overhead Post-Wrapper Insertion	0.9	1.1	1.2
% Area Overhead Post-Retiming	1.1	1.1	1.2
% Change in Pattern Count	3.9	−3.0	6.4

(b) Four-Die Stack

	Die 0	**Die 1**	**Die 2**	**Die 3**	**Complete Stack**
% Delay Recovered	100	100	100	100	100
% Area Overhead Post-Wrapper Insertion	2.1	2.3	2.5	1.9	2.4
% Area Overhead Post-Retiming	2.0	2.1	2.4	1.5	2.2
% Change in Pattern Count	1.1	5.1	0.8	−0.4	7.8

Table 4.3: Change in SDQL and pattern count before and after retiming for the (a) two-, three-, and four-die DES stack and (b) two- and four-die FFT stack.

(a) DES Stack

# of Dies in Stack	**Pre-retiming SDQL**	**Post-retiming SDQL**	**% Change** in SDQL	**% Change in** Pattern Count
2-Die	183	172	-6.0	-2.1
3-Die	182	185	1.6	0.7
4-Die	178	182	2.2	3.0

(b) FFT Stack

# of Dies in Stack	**Pre-retiming SDQL**	**Post-retiming SDQL**	**% Change** in SDQL	**% Change in** Pattern Count
2-Die	52385	51861	-1.0	0.9
4-Die	51977	52741	2.5	2.8

the FFT two- and four-die benchmarks.

Table 4.3 shows the change in pattern count and change in the statistical delay quality level (SDQL) reported by a commercial tool for small-delay defects. Table 4.3(a) provides results for the two-, three-, and four-die DES stacks and Table 4.3(b) provides results for the two- and four-die FFT stacks. Each row shows results for a stack of different size. The first column shows the SDQL after boundary cell insertion and before retiming. Column II shows the SDQL after retiming, with column III providing a percent-change in SDQL between the pre- and post-retiming values. Column IV provides the overall percent-change in pattern count between pre- and post-retiming ATPG.

Table 4.4 provides retiming results for the four-die FFT benchmark when one or two dies are fixed, meaning that they cannot be retimed. Because there is added latency to the inter-die critical paths on the fixed dies, tighter timing constraints are added to the dies that may be retimed. This makes retiming more difficult on the dies, but ensures that the added delays on the fixed dies are taken into account during retiming. The first column displays which dies are fixed, and the results given are for percentage delay recovery. Table 4.5 shows results from the simulations of Table 4.4 with the addition of the algorithm to move logic between dies in an attempt to improve results. Results are given only for simulations where two adjacent dies are unfixed, as logic cannot be moved to fixed dies.

As can be seen from the results, retiming recovered 100% of the additional GSF bypass mode delay in most cases. The one exception to this is Die 2 of the four-die stack under die-level retiming.

For the benchmark circuit used in this simulation, retiming becomes more difficult as the circuit is partitioned across more dies. During both die-level and stack-level retiming, slack can only be redistributed from within a die, because logic and registers will not be moved between dies. As the circuit is partitioned between more dies, there are fewer paths on each individual die from which to take excess slack. Furthermore, more paths will cross die boundaries and be subject to additional delay after GSFs are added during DfT-

insertion. This increased difficulty of retiming is reflected in the lower recovered delay of Die 2 of the four-die stack.

Table 4.1(c) and the results of Tables 4.4, 4.5, and 4.6 also demonstrate the greater effectiveness of stack-level retiming versus die-level retiming with regard to slack redistribution. For example, if the DES four-die stack were to be assembled after die-level retiming, an additional delay would exist on some of the paths that cross into Die 2 from Dies 1 and 3. This delay would be no worse than 40% of the delay of a GSF in bypass mode. This delay exists because not enough positive slack paths are present on Die 2 or because the extra slack present on Die 2 could not be distributed to TSV paths that violated timing constraints during retiming. During stack retiming, Die 2 would present the same difficulties to the retiming algorithm. However, because all dies are retimed simultaneously and complete paths that cross dies are known, slack could be redistributed on the other dies in the stack to make up for the restrictions of Die 2.

For the DES benchmark and die partitioning used in this chapter, the area overhead of DfT-insertion ranged from 16.3% for Die 0 of the two-die stack to 35.6% for Die 2 of the four die stack. Area overhead decreased with retiming, as the retiming algorithm retimes for area minimization after clock period minimization. The area overheads are relatively large because with die partitioning many paths crossed dies, and so many TSVs were introduced to the circuit. Compared to the number of cells on each partitioned die before DfT-insertion, the number of GSFs added after DfT-insertion accounted for a large number of the total number of cells. As can be seen from the FFT benchmark, which is much larger and therefore has a larger cell to TSV ratio, the area overheads were significantly lower, with 2.5% being the highest area overhead for Die 2 of the four-die FFT stack.

As the number of dies in a partition increased, the number of cells on each die before DfT-insertion decreased with an often corresponding increase in TSV count. Thus, the area overhead of DfT-insertion generally increased as the stack became larger. The area

Table 4.4: A comparison of delay recovery with fixed dies for the four-die FFT logic circuit partitioned considering (a) one and (b) two dies cannot be retimed.

(a) One Fixed Die

	Delay Recovery when Retimed for:				
Fixed Die	**Die 0**	**Die 1**	**Die 2**	**Die 3**	**Complete Stack**
Die 0	—	100	100	100	100
Die 1	87.5	—	100	100	100
Die 2	100	100	—	100	100
Die 3	100	100	100	—	100

(b) Two Fixed Dies

	Delay Recovery when Retimed for:				
Fixed Dies	**Die 0**	**Die 1**	**Die 2**	**Die 3**	**Complete Stack**
Dies 0, 1	—	—	75.0	82.5	100
Dies 2, 3	50.0	62.5	—	—	71.9
Dies 0, 2	—	62.5	—	82.5	84.4
Dies 1, 4	37.5	—	50.0	—	68.8

overhead was also generally shown to be worse for the inner dies of a stack, as these require GSFs both on TSV paths and on paths that contain face-side pads that will be bonded to TSVs on another die during stack assembly.

To illustrate this effect, consider the DES circuit evenly distributed among dies in a four die stack. In this case, each die would have about 6500 cells. Assume that each die has 500 connections with each adjacent die. The Die 0 and Die 3 of the stack would each need 500 GSFs, or 8% of the cells on the die. In comparison, Die 1 and Die 2 would each need 1000 GSFs, which is over 15% of the cells on the die. The area impact of GSF insertion would be significantly less if the dies themselves were more complex, with a higher cell count relative to the number of TSVs per die. For example, if a die contained one-million cells and 10,000 TSVs, then adding GSFs to a die would only account for 1-2% of the number of cells on the die, depending on whether or not the die were in the middle of the stack.

Tables 4.4(a) shows that, generally, there is enough slack present in the FFT circuit such that three of the four dies are capable of recovering their own timing overhead as well as that of the fixed die. This is not the case when two dies are fixed, as seen in Tables 4.4(b),

although better stack-level retiming results hold true in this case. Furthermore, the logic-movement algorithm to push logic cells between dies can be used to improve die-level retiming under two fixed dies as seen in Table 4.5. In the case of Die 1 when Dies 2 and 3 are fixed, there is a delay recovery improvement from 62.5% recovery without logic movement to 75% recovery with logic movement, or a 16.7% increase in delay recovery. In the case where Dies 2 and 3 are fixed, logic movement has no effect. Utilizing the logic-movement algorithm to push logic cells between dies can improve die-level retiming under two fixed dies as seen in Table 4.5. In the case of Die 1 when Dies 2 and 3 are fixed, there is a delay recovery improvement from 62.5% recovery without logic movement to 75% recovery with logic movement, or a 16.7% increase in delay recovery. In the case where Dies 2 and 3 were fixed, logic movement had no effect.

This example of logic redistribution provides insight into when logic redistribution can improve results and how much improvement can be expected. As demonstrated by Table 4.5, stack-level retiming, in general, produces better results than die-level retiming even with logic redistribution, and logic redistribution does not change stack-level retiming results. This is because stack-level retiming already considers the complete inter-die paths, so logic redistribution will never result in delay recovery improvements (or delay recovery reductions), and at best die-level retiming can only match stack-level retiming.

Additional insights are gained when considering the effect of redistribution with Die 0 and Die 1 fixed, or Die 2 and Die 3 fixed. In the former case, no improvement is seen as the critical paths contain no logic that could potentially be moved. In the latter case, there is improvement in the delay recovery on Die 1, as some logic is moved to Die 0, which contain some additional slack to be provided to the path. However, there is no improvement on Die 0 because its most critical path has no movable logic. Therefore, it is demonstrated that logic redistribution is useful only when two conditions are satisfied—there is movable logic on a die's most critical path, and there is additional slack on the adjacent die's portion

Table 4.5: A comparison of delay recovery with fixed dies and the additional use of the logic movement algorithm for the four-die FFT logic circuit with two fixed dies.

	Delay Recovery when Retimed for:				
Fixed Dies	**Die 0**	**Die 1**	**Die 2**	**Die 3**	**Complete Stack**
Dies 0, 1	—	—	75.0	82.5	100
Dies 2, 3	50.0	75	—	—	71.9

of that path.

Delay recovery is also more difficult in a circuit with modules that cannot be retimed, as shown in Table 4.6 with the OR1200 results. Although die-level retiming provided less delay recovery compared to other benchmarks, with 37.5% recovery for Die 0 and 50% recovery for Die 1, stack-level retiming still provided good results, with an overall 92% delay recovery.

The effect of DfT-insertion and retiming on stuck-at fault pattern count was found to be negligible. In some cases, such as Die 0 of the two die DES stack or Die 2 of the three die DES stack, this resulted in fewer test patterns after retiming. In other cases, such as Die 1 of the two die DES stack or Die 0 and Die 1 of the three die DES stack, retiming incurred a small increase in pattern count. It should be noted that when scan insertion is performed for the complete stack, scan chains are allowed to span multiple die layers. This is not the case when scan insertion is performed on a per-die basis. Thus, there is a significant difference in the number of test patterns produced after stack-level scan insertion when compared to die-level scan insertion. Variability is seen in pattern count because registers can be added/removed as well as moved throughout the circuit during retiming, there can be significant changes in controllability/observability during scan test. This can have a wide range of impact on ATPG, and this impact produces the pattern count changes.

Similar conclusions can also be drawn for path delay test patterns, as evident in Table 4.3. The effect of retiming on SDQL was generally greater in the relatively small DES stacks than the much larger FFT stacks. For the DES stack, the worse-case change in SDQL is a 6% reduction, while for the FFT stacks at worst a 1% reduction is seen. The

Table 4.6: A comparison of delay, area, and pattern count results for die- and stack-level retiming for the OR1200 processor partitioned across two dies.

	Die 0	**Die 1**	**Complete Stack**
% Delay Recovered	37.5	50	92
% Area Overhead Post-Wrapper Insertion	6.1	4.6	7.2
% Area Overhead Post-Retiming	5.2	4.4	6.3
% Change in Pattern Count	0.1	0.6	2.9

greater volatility of SDQL for the smaller stack implies that, when dies are larger, most die-internal paths are untouched by retiming. Since there are far fewer die-internal paths in the DES benchmark, changes in paths at or near the die boundary make up a change in a larger percentage of the total die paths. Changes in SDQL in the same design depending on stack size is due to differences in logic partitioning, which alters which paths are targeted for test by the statistical ATPG model.

Run times for the retiming algorithm were generally in minutes per die, but were longer for a complete stack and larger benchmark. For example, the run time for the complete four-die DES stack was 12.4 minutes. For each die, the run times (in minutes) were 3.5, 2.7, 2.6, and 2.2 for Die 0, Die 1, Die 2, and Die 3, respectively. For the FFT four-die benchmark, run times in minutes for die-level retiming were 9.4, 10.6, 10.2, and 9.1 for Die 0, Die 1, Die 2, and Die 3, respectively, while stack-level retiming required 41.5 minutes.

4.3 Conclusions

The methods and results discussed in this chapter show that retiming can be used to recover the delay added to circuit paths that cross die boundaries during die DfT-insertion. Retiming has been demonstrated at both the die-level and stack-level, with stack-level retiming providing an upper limit on die-level retiming. Retiming results on a DES circuit and an FFT circuit partitioned into two, three, and four die 3D benchmarks are provided. In most

cases retiming can recover 100% of the delay added by die DfT-insertion when all logic and dies are unfixed and in cases where a quarter of dies are not unfixed. It is further shown that, for modular benchmarks and benchmarks in which half of the dies are not unfixed, stack-level retiming outperforms die-level retiming in terms of delay recovery. However, a logic redistribution algorithm can be utilized to improve die-level retiming results in some cases. It has also been demonstrated that test pattern counts are not significantly impacted by DfT-insertion or retiming.

5

Test-Architecture Optimization and Test Scheduling

5.1 Introduction

Previous chapters have discussed issues associated with pre-bond KGD test and test standards for pre-bond and post-bond test. Post-bond testing is in many ways a less complex issue than pre-bond test, because external test pins can be utilized for test access and TSVs in bonded dies can be treated as interconnects during test (although testing the TSVs themselves may require testing for failure modes unique to TSVs and neighboring active devices). Nevertheless, new constraints for post-bond testing of a 3D stack must be considered, such as limited test access to dies depending on their position in the stack, multiple post-bond test insertions, and limitations on the addition of test TSVs between dies. Just as for pre-bond test, optimizations are needed to design 3D test architectures and test schedules to minimize the cost of post-bond test in order to achieve cost-effective KGD, partial stack, and known-good-stack (KGS) test throughout the manufacture flow of a product.

Memories are easier to stack compared to logic due to high yields after repair and simplified testing and design [10], and as such 3D memory stacks have already been manu-

factured [10]. Stacks that include memory stacked on logic [64] or multiple logic dies [65] are likely to be seen in the near future. Although 3D design-and-test automation is not yet fully mature for commercial exploitation, it is well on its way [13] and many commercial design tools are beginning to support varying degrees of 3D design. These tools need to be able to exploit the benefits of 3D technologies while taking into account the various design-related trade-offs. For example, in a TSV-based 3D-SIC, the number of TSVs available for test access is limited because of their associated chip area costs. Most TSVs are likely to be dedicated to functional access, power/ground, and clock routing.

Post-bond testing of core-based dies in 3D-SICs brings forward new challenges [60, 58]. In order to test the dies and associated cores, a Test Access Mechanism (TAM) must be included on the dies to transport test data to the cores, and a 3D TAM is needed to transfer test data to the dies from the stack input/output pins. TAM design in 3D-SICs involves additional challenges compared to TAM design for 2D SOCs. In a 3D-SIC, a test architecture must be able to support testing of individual dies as well as testing of partial and complete stacks, and it is for this reason that the test standards discussed in the Introduction are being developed. These standards require compatible test architecture optimizations that must not only minimize the test length, but also minimize the number of TSVs used to route the 3D TAM, as each TSV has area costs associated with it and is a potential source of defects in a 3D-SIC. The test length is therefore dependent on the test architecture and test schedule and is constrained by a limit on the available test resources.

In this chapter, test architecture optimization for 3D stacked ICs implemented using TSVs is discussed. The optimizations are compatible with emerging die wrapper standards. A variety of design cases for 3D SICs with die-level test architectures are considered—including dies with fixed test architectures and dies whose test architectures have yet to be designed. Over the course of this chapter, mathematical programming techniques are derived to create optimal solutions for a variety of architecture optimization problems. These

mathematical models, in addition to being immediately useful for optimization, provide a clear framework for 3D optimization that serve as a foundation for future research and applications. The optimization will first be developed for a complete stack test, and then extended to include optimizations for any or all post-bond test insertions as well as post-bond TSV test. From this, it will be demonstrated that optimal test architecture solutions and test schedules for multiple test insertions are different from their counterparts for a final stack test alone.

5.1.1 3D Test Architecture and Test Scheduling

The problem of test-architecture optimization for 3D-SICs considers three different 3D integration cases—(1) *hard dies*, in which a test architecture already exists, (2) *soft dies*, for which the 2D (per die) and 3D (per stack) test architectures are co-optimized, and (3) *firm dies*, in which a test architecture already exists 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. For the sake of simplicity and ease of implementation, this chapter assumes session-based test scheduling [20], i.e., in which all tests which are executed simultaneously need to be completed before the next test session is started. Methods for minimizing the number of TSVs or test pins used for target test lengths are developed for both a total stack limit and a limit on TSVs between neighboring dies. While it is theoretically possible to have multiple dies on a given layer in a stack, for this chapter it is assumed that there is only one die per layer in a stack. Furthermore, a core is considered to be part of a single die only, i.e., "3D cores" are not considered. In addition to minimizing the test length for each soft die, the test length for the complete stack is minimized in all three problem instances.

Testing of 2D SOCs and the optimization of related test-access architectures have been well studied [27, 42, 46, 30]. Optimization methods have included integer linear programming (ILP) [27], rectangle packing [27, 45], iterative refinement [46], and other heuris-

tics [30, 47]. However, these methods were all originally developed for 2D SOCs, and the added test complexities related to 3D technology were not considered.

Recently, early work has been reported on testing of 3D-SICs. Heuristic methods for designing core wrappers in 3D-SICs were developed in [14]. These methods do not address the problem of 3D TAM design. ILP models for test architecture design for each die in a stack are presented in [66]. While these ILP models take into account some of the constraints related to 3D-SIC testing such as a TSV limit, this approach does not consider the reuse of die-level TAMs. A TAM wire-length minimization technique based on simulated annealing is presented in [67]. A drawback of this approach is that it implies a 3D test architecture that is not feasible in practice. Heuristic methods for reducing weighted test cost while taking into account the constraints on test pin widths in pre-bond and post-bond tests are described in [54]. An unrealistic assumption made in [54] is that TAMs can start and terminate in any layer.

In most prior work on 3D-SIC testing, TAM optimization is performed at die-level only, which leads to inefficient TAMs and non-optimal test schedules for partial-stack and complete-stack tests. Furthermore, all previous methods assume that the designer can create TAM architectures on each die during optimization, which may not be possible in all cases. In the Introduction a die-level wrapper and associated 3D architecture is presented to allow for all pre-bond and post-bond tests. This approach proposes die-level wrappers and leverages the current IEEE 1149.1 and IEEE 1500 standards. In addition to functional and test modes, die-level wrappers allow bypass of test data to and from higher die in the stack and reduced test bandwidth during pre-bond tests. This is a realistic and practical look at test architectures in 3D-SICs, but it offers no insight into optimization and test scheduling. The optimization methods presented in this chapter are compatible with die wrappers, and they do not make any unrealistic assumptions about die wrappers or the 3D TAM. Pre-bond testing is not included in the optimization in this chapter. If reconfigurable scan chains as

described in Chapter 3 are utilized, then pre-bond test configurations for each die can be considered as a separate optimization problem.

5.1.2 The Need for Optimization Considering Multiple Post-Bond Test Insertions and TSV Test

Compared to two-dimensional ICs that typically require two test insertions, namely wafer test and package test, 3D stacking introduces a number of natural test insertions [58]. Because the die-stacking steps of thinning, alignment, and bonding can introduce defects, there may be a need to test multiple subsequent (partial) stacks during assembly. Figure 5.1 shows an example manufacturing and test flow for a 3D stack. First, wafer test (i.e., pre-bond test) can be used to test die prior to stacking to ensure correct functionality, as well as to match die in a stack for power and performance. Next, Die 1 and Die 2 are stacked, and then tested again. This is likely to be the first time the TSVs between Die 1 and Die 2 will be tested due to technology limitations that make pre-bond test of TSVs infeasible [60]. This step also ensures that defects can be detected in the stack due to additional 3D manufacturing steps such as alignment and bonding. The third die is added to the stack and all dies in the stack, including all TSV connections, are retested. Finally, the "known good stack" is packaged and the final product is tested. Optimization methods are needed to minimize test time not only for the final stack test, i.e., if the intermediate (partial) stacks are not tested, but also to minimize the total test time if the final stack and partial stacks are tested during bonding.

In Section 5.3, previously discussed optimization methods for 3D-SICs with *hard dies* and *soft dies* will be extended to consider multiple post-bond test insertions. In addition to minimizing the test time for each soft die, the test time can be minimized by considering all possible stack tests and the complete stack, as well as die-external tests as well. These optimization methods allow for the efficient generation of multiple options for testing a 3D-SIC.

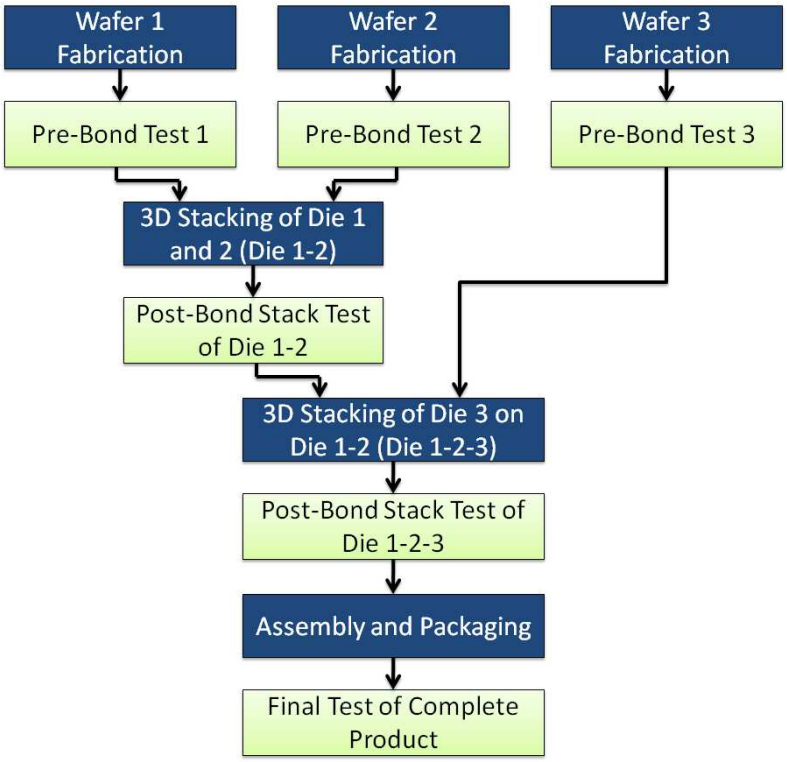


FIGURE 5.1: 3D-SIC manufacturing and test flow with multiple test insertions.

In [28], the authors presented an expanded wrapper architecture for 2D ICs using modified wrapper cells in which each wrapper cell can be connected to two TAMs. As opposed to the 1500-standard wrapper (referred to in the rest of this chapter as a "thin" or 1500-like wrapper), this expanded wrapper architecture, or "fat" wrapper, allows for core-external test (EXTEST) and core-internal test (INTEST) to be run in parallel. This chapter will consider both types of wrappers for EXTEST optimization; in particular, the use of fat wrappers in this chapter is a natural extension of die-level wrappers to allow for die-external tests (TSV tests) and die-internal tests in parallel.

The rest of the chapter is organized as follows. In Section 5.2, optimization techniques are introduced for minimizing the test time for final stack test. A global limit is set on the number of dedicated TSVs to be used for test access and constraints are imposed on test bandwidth due to a limited number of test pins on the lowest die in the stack. While

this optimization provides a sufficient starting point for designing a 3D test architecture, it does not consider multiple test insertions for testing the partial stack. Furthermore, the test time for TSVs and die-external logic is ignored in the optimization framework. Section 5.3 extends the model of Section 5.2 to allow for multiple test schedules and optimization for any number of or all post-bond stack tests. The test-bandwidth constraints use a more realistic model for dedicated test TSVs by considering a maximum number of TSVs per die, as opposed to a global limit. Furthermore, the test time for die-internal and die-external tests using both fat and thin wrappers is considered in the optimization. Section 5.6 concludes the chapter.

5.2 Test Architecture and Scheduling Optimization for Final Stack Test

In a 3D-SIC, which currently consist of anywhere from two to eight dies [18], the lowest die is usually directly connected to chip I/O pins and therefore can be tested using test pins. To test the non-bottom dies in the stack, test data must enter through the test pins on the lowest die. Therefore, to test other dies in the stack, the test access mechanism (TAM) must be extended to all dies in the stack through the test pins at the lowest die. To transport test data up and down the stack, "TestElevators" [50] need to be included on each die except for the highest die in the stack [58]. The number of test pins and TestElevators as well as the number of TSVs used affect the total test length for the stack.

Consider an example 3D-SIC with three dies with given test access architectures as shown in Figure 5.2. Suppose the test lengths for Die 1, Die 2, and Die 3 are 300, 800, and 600 clock cycles, respectively. The total number of available test pins at the bottom die is 100. Die 1 requires 40 test pins (TAM width of 20), and Dies 2 and 3 require 60 TestElevators and 40 TestElevators, respectively. The test length for each die is determined by its test architecture.

Figure 5.2(a) shows the TestElevator widths and the number of TSVs used if all dies are

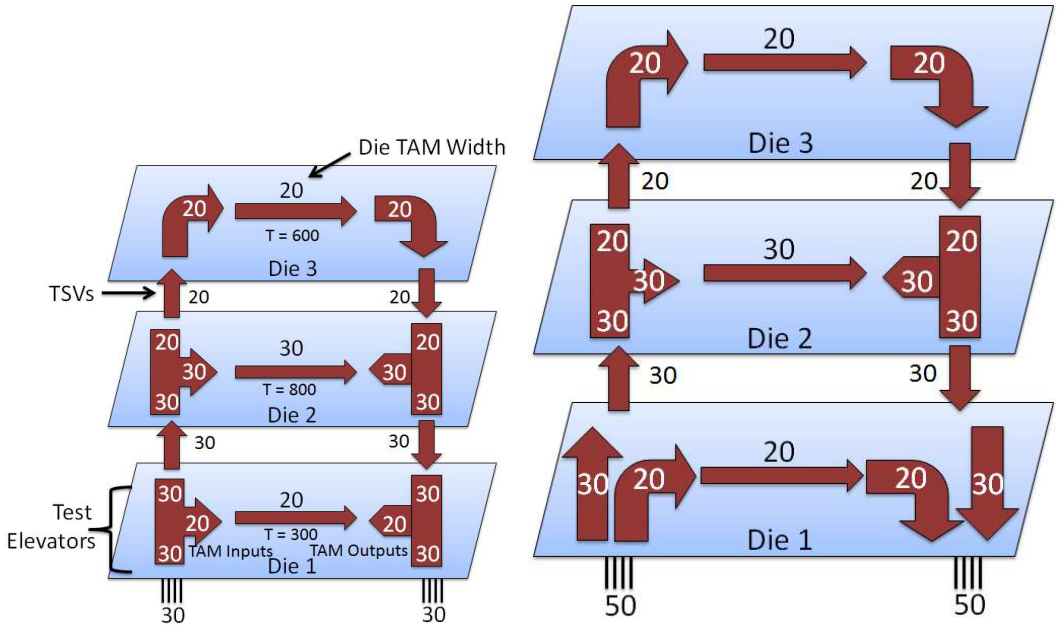
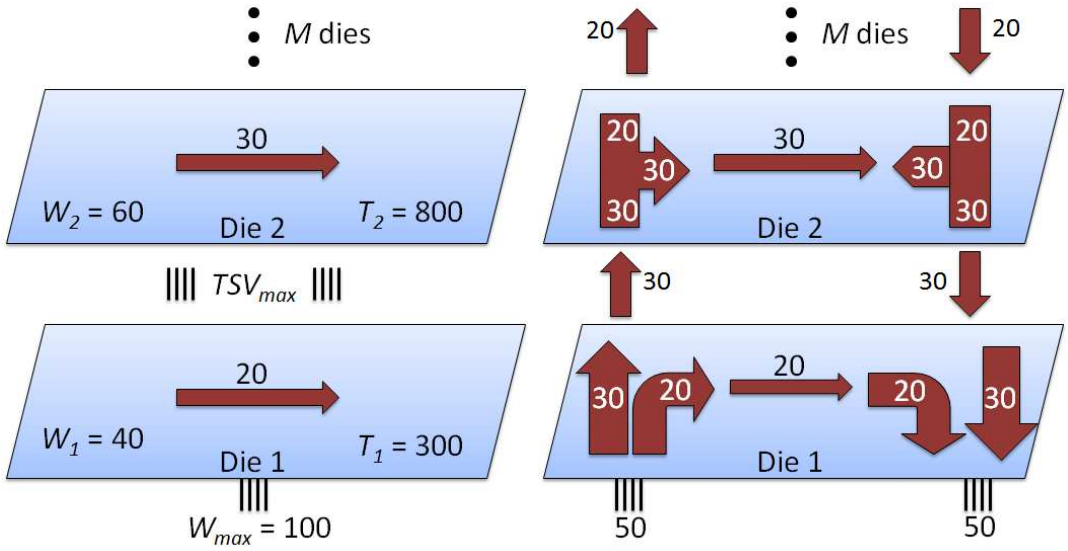


FIGURE 5.2: Example 3D-SIC with three hard dies.

tested serially. In this case, a total of 100 TSVs are used, and 100 test pins are available, of which only 60 are utilized. The total test length for the stack is the sum of the test lengths of the individual dies, i.e., 1700 cycles. Figure 5.2(b) shows the test architecture required if Die 1 and Die 2 are tested in parallel. In this case, the number of TSVs used is the same as in Figure 5.2(a). However, all 100 test pins are required to test Die 1 and Die 2 in parallel. Also, 60 TestElevators must pass between Die 1 and Die 2 in order to pass a separate 30-bit wide TAM to Die 2 for parallel testing. For this case, the total test length for the stack is $\max\{300, 800\} + 600 = 1400$ cycles. This example clearly shows that there is a trade-off between test length and the number of test pins and TSVs used. Therefore, a test-architecture optimization algorithm for 3D-SICs has to minimize the test length while taking into account upper limits on the number of test pins and TSVs used.

Test-architecture optimization for 3D-SICs with hard dies is illustrated in Figure 5.3. For a hard die, the 2D test architecture on the die is fixed. The only structure over which the designer has control is the 3D TAM. Hard dies offer less flexibility for optimization in the sense that each die must have exactly the pre-defined number of input and output TAM



(a) Problem instance (b) Architecture solution

FIGURE 5.3: Illustration of PSHD: (a) a problem instance; (b) an optimized architecture.

wires appropriated to it in the design of the 3D TAM. Therefore, the only decisions that can be made in designing the 3D TAM is which (if any) dies can be tested in parallel with one another given the limitations on test pins and test TSVs. Hard dies may be present in TAM design problems if vendors sell fabricated dies to a 3D integrator.

Figure 5.3(a) illustrates the variables that arise for the hard-die problem. As can be seen, a fixed 2D TAM width is given along with the known test time for each die. The given constraints are the number of test pins W_{max} and the number of test TSVs TSV_{max} available. A solution, therefore, can be given as in Figure 5.3(b). Here, each die receives the required and pre-defined test bandwidth, but Die 1 and Die 2 are tested in parallel through the 3D TAM.

The test-architecture optimization problem for hard dies is denoted as PSHD, where "PS" stands for "problem statement" and "HD" stands for "hard dies". The problem can be defined as follows.

3D-SIC with Hard Dies (PSHD)

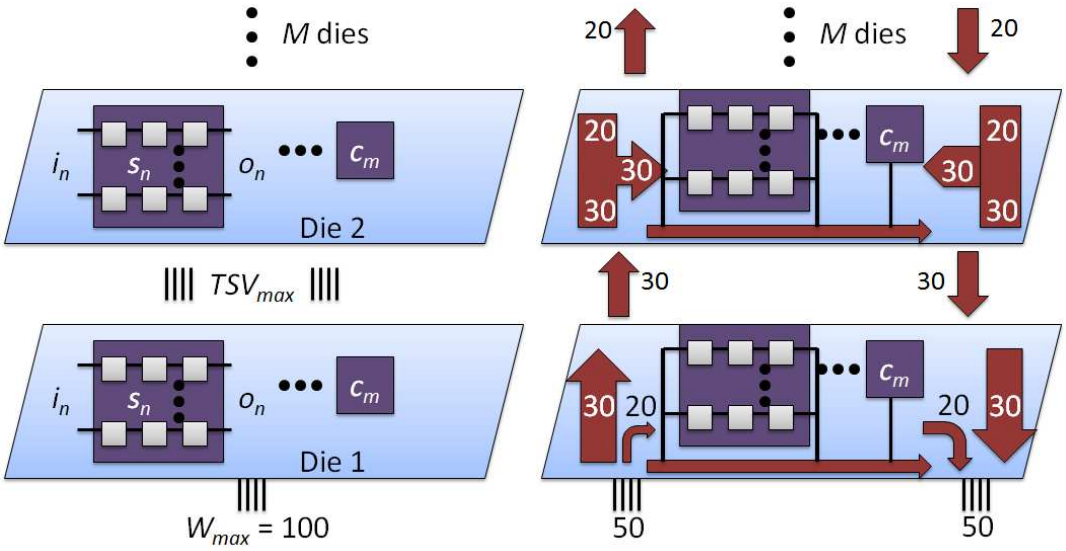
Given a stack with a set M of dies, total number of test pins W_{max} available for test, and a maximum number of TSVs (TSV_{max}) that can be used globally (throughout the entire stack)

for TAM design. For each die $m \in M$, the die's number corresponds to its tier in the stack (Die 1 is the bottom die, Die 2 is next, and so forth), the number of test pins on each die w_m ($w_m \leq W_{max}$) required to test the die is given, and the associated test length t_m (because the test architecture per die is given, t_m is also given). Determine an optimal TAM design and corresponding test schedule for the stack such that the total test length T for the stack is minimized and the number of TSVs used does not exceed TSV_{max}.

Two dual problems, PSHDT (the "T" stands for TSV minimization) and PSHDW (the "W" stands for test pin-count minimization), can be stated as follows. For PSHDT, determine an optimal TAM design and corresponding test schedule for the stack such that the total number of TSVs used for the stack is minimized and the upper limits on test length T_{max} and test pin count W_{max} are not exceeded. For PSHDW, determine an optimal TAM design and test schedule for the stack such that the total number of test pins used for the stack is minimized and the upper limits on test length T_{max} and total number of TSVs (TSV_{max}) are not exceeded.

The hard-die model is based on prior work on SOC testing [19] with additional constraints, while the firm and soft die models are considerably different and more complex. Besides simply adding 3D design constraints, each die must be considered across a range of many different possible TAM widths and many variations must be considered in which dies are tested in parallel along a 3D stack. These considerations require the addition of many more variables and constraints. Overall, these additions make the firm- and soft-die models significantly more complex than the hard-die model, potentially limiting the number of die that can be included in the model before run time becomes prohibitively high.

The above problem statement is different for a 3D-SIC with soft dies. In the case of soft dies, the test architecture for each die is not pre-defined, but is determined during the test-architecture design of the stack. In this case, both the 2D and 3D TAMs are co-designed. Scan chains for each test module are given, but the test wrappers for each module and



(a) Problem instance (b) Architecture solution

FIGURE 5.4: Illustration of PSSD: (a) a problem instance; (b) optimized architecture.

the TAM are designed during 3D TAM design. This allows the designer to develop the most efficient 2D/3D TAM designs given TSV-count and test pin-count constraints. Soft dies model the additional flexibility available for optimization when dies are fabricated in-house for 3D integration.

Test-architecture optimization for 3D-SICs with soft dies is illustrated in Figure 5.4. Figure 5.4(a) shows the known quantities associated with the soft die model, namely the number of modules per die, the pre-defined scan-chains per die, and W_{max} and TSV_{max}. Figure 5.4(b) shows the result of optimization, including wrapper, 2D TAM, and 3D TAM designs.

The test-architecture optimization problem for soft dies can be formally defined as follows.

3D-SIC with Soft Dies (PSSD)

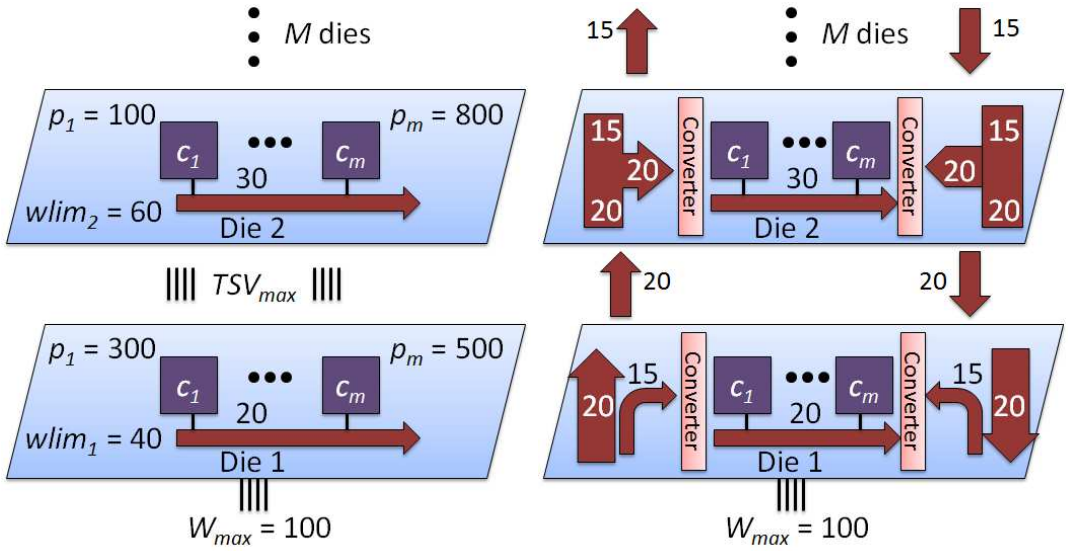
Given a stack with a set M of dies, the total number of test pins W_{max} available for test at the lowest die, and a maximum number of TSVs (TSV_{max}) that can be used for TAM design. For each die $m \in M$, the total number of cores c_m is given. Furthermore, for each core c,

the number of inputs i_c, outputs o_c, total number of test patterns p_c, total number of scan chains s_c, and for each scan chain v, the length of the scan chain in flip flops $l_{c,v}$ are given. Determine an optimal TAM design and test schedule for the stack, as well as for each die, such that the total test length T for the stack is minimized and the number of TSVs used does not exceed TSV_{max}.

Two dual problems, PSSDT and PSSDW, respectively, can again be stated as follows. For PSSDT, determine an optimal TAM design and test schedule for the stack and for each die such that the total number of TSVs used for the stack is minimized and the upper limits on test length T_{max} and test pin count W_{max} are not exceeded. For PSSDW, determine an optimal TAM design and test schedule for the stack as well as for each die such that the total number of test pins used for the stack is minimized and the upper limits on test length T_{max} and total number of TSVs (TSV_{max}) are not exceeded.

Finally, the problem statement is developed for a 3D-SIC with firm die. In the case of firm dies, the test architecture for each die is pre-defined as for a hard die, but additional serial/parallel conversion hardware may be added to the die to allow for fewer test elevators (or test pins in the case of the lowest die) to be used than in the case of the fixed 2D TAM width for the die. The conversion hardware is added before the inputs and after the outputs of the die wrapper. The input hardware multiplexes a smaller number of TAM wires to a larger number of die wrapper wires. Demultiplexers at the output of the die wrapper transfer test responses from a larger number of die wrapper wires to a smaller number of TAM wires. Compared to the scenario involving hard dies, this scenario allows the use of fewer test pins at the expense of higher test lengths, but also allows additional flexibility in test scheduling and test-time optimization.

The problem of test-architecture optimization for 3D-SICs with firm dies is shown in Figure 5.5. Figure 5.5(a) shows the known quantities for the firm die problem; these are similar to those of the hard die problem except that test times are given for certain se-



(a) Problem instance (b) Architecture solution

FIGURE 5.5: Illustration of PSFD: (a) a problem instance; (b) optimized architecture.

rial/parallel conversion bandwidths for each die. During optimization, one of these converters (or no converter) can be used, as seen in Figure 5.5(b). For Die 1, for example, a 3D TAM width of 15 bits is used, although the 2D TAM was designed for a width of 20 bits.

The test-architecture optimization for firm die is formally defined as follows.

3D-SIC with Firm Dies (PSFD)

Given a stack with a set M of dies, the total number of test pins W_{max} available for test at the lowest die, and a maximum number of TSVs (TSV_{max}) that can be used for TAM design. For each die $m \in M$, a fixed 2D TAM architecture with the total number of cores c_m is given along with which TAM partitions they utilize and their TAM widths. Furthermore, for each core n, the total number of test patterns p_n is given and the number of test pins $wlim_m$ required to test the die is given. Determine an optimal TAM design and test schedule for the stack, as well as possible serial/parallel conversion widths for each die, such that the total test length T for the stack is minimized and the number of TSVs used does not exceed TSV_{max}.

The above problems are all NP-hard (“proof by restriction”), as they can be reduced using standard techniques to the rectangle packing problem, which is known to be NP-

hard [21]. For example, for PSHD, if the constraints on the maximum number of TSVs are removed, each die can be represented as a rectangle with a width equal to its test length and height equal to the number of required test pins. Now all these rectangles (dies) must be packed into a bin with a width equal to the total number of test pins and a height equal to the total test length for the stack, which must be minimized. Similarly, for PSSD, a rectangle must also be selected for each die from a set of rectangles with different widths and heights, but a special case of the scenario is identical to PSHD. Despite the NP-hard nature of these problems, they can be solved optimally because the number of layers in a 3D-SIC is expected to be limited, e.g., up to four layers have been predicted for logic stacks [22].

The above problems are more general than the combinatorial problem of rectangle packing [21]. The added 3D design constraints and the greater design freedom available, especially for firm and soft dies, drastically increase the solution space. Rectangle packing is only a special and a considerably more simple case of our problem statements.

5.2.1 Test-Architecture Optimization for Final Stack Test

In this section, integer linear programming (ILP) is utilized to solve the problems defined in the previous section. Although ILP methods do not scale well with problem instance size, the problem instance sizes for P_{SHD} and P_{SSD} are relatively small for realistic stacks, and therefore ILP methods are good candidates for solving them.

5.2.2 ILP Formulation for PSHD

To create an ILP model for this problem, the set of variables and constraints must be defined. Consider a binary variable x_{ij}, which is equal to 1 if die i is tested in parallel with die j, and 0 otherwise. Constraints on variable x_{ij} can be defined as follows:

$$x_{ii} = 1 \qquad \forall i \tag{5.1}$$

$$x_{ij} = x_{ji} \qquad \forall i, j \tag{5.2}$$

$$1 - x_{ij} \geq x_{ik} - x_{jk} \geq x_{ij} - 1 \qquad \forall i \neq j \neq k \tag{5.3}$$

The first constraint indicates that every die is always considered to be tested with itself. The second constraint states that if die i is tested in parallel with die j, then die j is also tested in parallel with die i. The last constraint ensures that if die i is tested in parallel with die j, then it must also be tested in parallel with all other dies that are tested in parallel with die j.

Next, consider a second binary variable y_i, which is equal to 0 if die i is tested in parallel with die j on a lower layer ($l_i > l_j$), and 1 otherwise. The total test length T for the stack is the sum of the test lengths of all dies that are tested in series plus the maximum of the test lengths of each of the sets of parallel tested dies. Using variables x_{ij} and y_i, the total test length T for a stack with set of dies M can be defined as follows.

$$T = \sum_{i=1}^{|M|} y_i \cdot \left(\max_{j=i..|M|} \{x_{ij} \cdot t_j\} \right) \tag{5.4}$$

A proof of correctness of Equation 5.4 can be derived through induction as follows:

Base Case:

For the base case, consider two layers for which there are two possible optimization outcomes. Either both die are tested in series, or both die are tested in parallel. In the case of series testing, then $y_1 = 1, x_{11} = 1, x_{12} = 0$ and $y_2 = 1, x_{21} = 0, x_{22} = 1$. Using this information and equation 5.4, the test length is determined to be $y_1 \max\{x_{11} \cdot t_1\} + y_2 \max\{x_{22} \cdot t_2\} = \max\{t_1, 0\} + \max\{t_2\} = t_1 + t_2$. For parallel testing, the variables become $y_1 = 1, x_{11} = 1, x_{12} = 1$ and $y_2 = 0, x_{21} = 1, x_{22} = 1$. The equation becomes $1 \cdot \max\{t_1, t_2\} + 0 \cdot \max\{t_2\} = \max\{t_1, t_2\}$. These can both be demonstrated to be correct.

Induction Hypothesis:

It is assumed that (5.4) holds for M die.

Recursive Step:

It must be proven that the test length for die $M+1$ is properly considered in the overall test length. Either die $M+1$ is tested in serial with regard to the die in the stack, or it is tested in parallel with some die on a lower layer of the stack. When die $M+1$ is tested in series, $y_{M+1} = 1, x_{M+1,M+1} = 1$, and $x_{n,M+1}$ and $x_{M+1,n}$ are zero for all $n \neq M+1$. The test length becomes

$$\begin{aligned} & y_1 \cdot \max\{x_{11}t_1, x_{12}t_2, \cdots, x_{1M}t_M, x_{1,M+1}t_{M+1}\} + \\ & y_2 \cdot \max\{x_{22}t_2, x_{23}t_3, \cdots, x_{2M}t_M, x_{2,M+1}t_{M+1}\} + \\ & \qquad \cdots + y_M \max\{x_{MM}t_M\} \end{aligned} \tag{5.5}$$

In this equation, $x_{n,M+1}$ is 0 for all $n \neq M+1$, so the test length of die $M+1$ is only added to the total test length once, for:

$$y_{M+1} \max\{x_{M+1,M+1}t_{M+1} = t_{M+1}\} \tag{5.6}$$

For the parallel case, die $M+1$ is tested in parallel with one or more die below it. Let die k be the die lowest in layer that is tested in parallel with die $M+1$. Then $y_k = 1, x_{k,M+1} = 1, y_{M+1} = 0$, and $x_{M+1,k} = 1$. Then

$$y_{M+1} \max\{x_{M+1,M+1}t_{M+1}\} \tag{5.7}$$

goes to zero, and

$$y_k \max\{x_{kk}t_k, x_{k,k+1}t_{k+1}, ..., x_{k,M+1}t_{M+1}\} \tag{5.8}$$

takes into account the test length of dies k, $M+1$, and any other die tested in this parallel tested set.

It should be noted that Equation (5.4) has two non-linear elements, the *max* function, and the product of variable y_i and the *max* function. This equation is linearized by introducing two new variables. The variable c_i takes the value of the *max* function for each die i and the variable u_i represents the product $y_i \cdot c_i$. The variables u_i and c_i are defined using

standard linearization techniques as shown in Figure 5.6. The linearized function for total test length can be written as follows.

$$T = \sum_{i=1}^{|M|} u_i \tag{5.9}$$

As the number of test pins used for parallel testing of dies should not exceed the given test pins W_{max}, a constraint on the total number of pins used to test all dies in a parallel set can be defined as follows. In the inequalities, w_j refers to the TAM width for die j.

$$\sum_{j=1}^{|M|} x_{ij} \cdot w_j \leq W_{max} \qquad \forall i \tag{5.10}$$

Similarly, the total number of used TSVs should not exceed the given TSV limit TSV_{max}. The number of TSVs used to connect layer i to layer $i-1$ is the maximum of the number of pins required by the layer at or above layer i that takes the most test pin connections, and the sum of parallel-tested dies at or above layer i in the same parallel-tested set. Based on this, the constraint on the total number of TSVs used in a test architecture can be defined as follows:

$$\sum_{i=2}^{|M|} \{\max_{k=i}^{|M|} \{w_k, \sum_{j=k}^{|M|} w_j \cdot x_{kj}\}\} \leq TSV_{max} \tag{5.11}$$

The above set of constraints can be linearized by representing the *max* function by a variable d_i. Finally, to complete the ILP model for PSHD, constraints on binary variable y_i and the relationship between binary variable y_i and x_{ij} must be defined. For this purpose, a constant C is defined that approaches but is less than 1. It is then possible to define y_i as follows:

$$y_1 = 1 \tag{5.12}$$

$$y_i \geq \frac{1}{1-i} \sum_{j=1}^{i-1} \left(x_{ij} - 1\right) - C \qquad \forall i > 1 \tag{5.13}$$

Equation 5.12 forces y_1 to 1, because the lowest layer cannot be tested in parallel with any layer lower than itself. Constraint (5.13) defines y_i for the other layers. To understand this constraint, first make the observation that the objective function (as shown in Equation (5.4)) would be minimized if each y_i is zero. This would make the objective function value equal to 0, which is an absolute minimum test length. Thus, y_i must be restricted to 1 only where it is absolutely necessary, otherwise the objective function can be relied on to assign a value 0 to all unrestricted y_i variables. This equation considers the range of values that the sum of x_{ij} can take. The fraction in the equation normalizes the sum to a value between 0 and 1 inclusive, while the summation considers all possible cases for a die being tested in parallel with a die below it.

Equation 5.13 can be proven correct through induction as follows:

Base Case:

Consider for the base case a stack of 2 die. The variable y_1 is always equal to one, $x_{11} = 1$, and x_{22}=1. There are two possible configurations for testing the two die. The first configuration is that both die are tested serially. If this is true, then the variables take the values of $x_{12} = 0$ and $x_{21} = 0$. The equation for y_2 then becomes:

$$y_2 \geq \frac{1}{1-2} \left(x_{21} - 1\right) - M \tag{5.14}$$

$$y_2 \geq 1 - M \tag{5.15}$$

Because $M < 1$, $1 - M$ is some small fraction greater than zero. Thus, y_2 must be greater than zero, and because it is binary it must take the value of one. The second possibility is

that both die are tested in parallel such that $x_{12} = 1$ and $x_{21} = 1$. This defines y_2 as follows:

$$y_2 \geq \frac{1}{1-2}(x_{21} - 1) - M \tag{5.16}$$

$$y_2 \geq -M \tag{5.17}$$

This leaves y_2 unrestricted, as it can only take the value zero or one and is thus always greater than a negative number. Due to the objective, y_2 will become zero as desired, because it is tested in parallel with a die lower in the stack than itself.

Induction Hypothesis:

It is assumed that Equation (5.13) holds for the case of m die.

Case of $m+1$ die

For die $m+1$, Equation (5.13) becomes:

$$\begin{aligned} y_{m+1} \geq \frac{1}{1-(m+1)}([x_{(m+1)1} - 1] + [x_{(m+1)2} - 1] + \\ \ldots + [x_{(m+1)m} - 1]) - M \end{aligned} \tag{5.18}$$

If $m+1$ is tested serially, then the summation adds the quantity -1 a total of m times. This results in:

$$y_m \geq \frac{1}{-m}(-m) - M \tag{5.19}$$

$$y_m \geq 1 - M \tag{5.20}$$

This forces y_{m+1} to one. Now consider the range of values that can be taken by the right hand side of equation 5.18 for parallel testing cases of die $m+1$ by considering the extremes. If die $m+1$ is tested in parallel with only one die below it, then one of the terms of the summation becomes zero and the calculation becomes:

$$y_m \geq \frac{1}{-m}(-(m-1)) - M \tag{5.21}$$

$$y_m \geq \frac{1-m}{-m} - M \tag{5.22}$$

Objective:

Minimize $\sum_{i=1}^{|M|} u_i$

Subject to the Constraints:

$t_{max} = \max_{i=1}^{|M|} t_i$

$c_i \geq x_{ij} \cdot t_j \qquad \forall i, j = i..|M|$

$u_i \geq 0 \qquad \forall i$

$u_i - t_{max} \cdot y_i \leq 0 \qquad \forall i$

$u_i - c_i \leq 0 \qquad \forall i$

$c_i - u_i + t_{max} \cdot y_i \leq t_{max} \qquad \forall i$

$\sum_{j=1}^{|M|} x_{ij} \cdot w_j \leq W_{max} \qquad \forall i$

$x_{ii} = 1 \qquad \forall i$

$x_{ij} = x_{ji} \qquad \forall i, j$

$1 - x_{ij} \geq x_{ik} - x_{jk} \geq x_{ij} - 1 \qquad \forall i \neq j \neq k$

$\sum_{i=2}^{|M|} d_i \leq TSV_{max} \qquad \forall i$

$d_i \geq \sum_{j=k}^{|M|} w_j \cdot x_{kj} \qquad \forall i, k = i..|M|$

$d_i \geq w_j \qquad \forall i, j = i..|M|$

$y_1 = 1$

$y_i \geq \frac{1}{1-i} \sum_{j=1}^{i-1} (x_{ij} - 1) - C \qquad \forall i > 1$

FIGURE 5.6: ILP model for 3D TAM optimization PSHD.

The fraction clearly results in a positive number less than one, and subtracting M makes this a negative value, leaving y_{m+1} unrestricted. In the case of die $m+1$ being tested in parallel with every die below it, every term of the summation is zero and the calculation becomes:

$$y_m \geq \frac{1}{-m}(0) - M \tag{5.23}$$

$$y_m \geq -M \tag{5.24}$$

Thus, for all cases of die $m+1$ tested in parallel, the right hand side is in the range $[\frac{1-m}{-m} - M, -M]$, which are all negative values, thereby leaving y_{m+1} unrestricted.

The complete ILP model for problem PSHD is shown in Figure 5.6.

A benefit of using ILP is that the dual problems PSHDT and PSHDW can be easily tackled by appropriately modifying the model of Figure 5.6. For both PSHDT and PSHDW, a maximum test length constraint T_{max} is introduced and the following inequality is added

to the model:

$$\sum_{i=1}^{|M|} u_i \leq T_{max}.$$

As can be easily seen, the previous objective function is now transformed into a constraint. For PSHDT, the constraint on TSVs used is removed, which is the inequality involving TSV_{max}, and is replaced with the following objective function:

$$\text{Minimize} \sum_{i=2}^{|M|} d_i$$

For PSHDW, the constraint on the number of test pins used is removed, which is the inequality involving W_{max}, and the variable P is introduced to represent the number of test pins used by the stack. The following inequalities define P.

$$P \geq \sum_{j=1}^{|M|} x_{ij} \cdot w_j \qquad \forall i \tag{5.25}$$

Our objective for PSHDW is therefore to minimize P.

5.2.3 **ILP Formulation for** PSSD

The ILP formulation for 3D-SICs with soft cores is derived in a similar manner as that for 3D-SICs with hard cores. In this case, the test length t_i for die i is a function of the TAM width w_i assigned to it. Using the variables x_{ij} and y_i as defined in Section 5.2.2, the total test length T for the stack with the set of soft dies M can be defined as follows.

$$T = \sum_{i=1}^{|M|} y_i \cdot \max_{j=i..|M|} \{x_{ij} \cdot t_j(w_j)\} \tag{5.26}$$

It should be noted that Equation (5.26) has several non-linear elements. To linearize this equation, the test length function must first be defined. For this purpose, the binary variable g_{in} is introduced where $g_{in} = 1$ if $w_i = n$, and 0 otherwise. The expression is then linearized

Objective:

Minimize $\sum_{i=1}^{N} u_i$

Subject to the Constraints:

$t_{max} = \max_{i=1}^{|M|} t_i$

$c_i \geq v_{ij} \qquad \forall i, j = i..|M|$

$v_{ij} \geq 0 \qquad \forall i, j$

$v_{ij} - t_{max} \cdot x_{ij} \leq 0 \qquad \forall i, j = 1..|M|$

$-\sum_{n=1}^{k_i}(g_{jn} \cdot t_j(n)) + v_{ij} \leq 0 \qquad \forall i, j$

$\sum_{n=1}^{k_i}(g_{jn} \cdot t_j(n)) - v_{ij} + t_{max} \cdot x_{ij} \leq t_{max} \qquad \forall i, j$

$u_i \geq 0 \qquad \forall i$

$u_i - t_{max} \cdot y_i \leq 0 \qquad \forall i$

$u_i - c_i \leq 0 \qquad \forall i$

$c_i - u_i + t_{max} \cdot y_i \leq t_{max} \qquad \forall i$

$z_{ijk} \geq 0 \qquad \forall i, j, k$

$z_{ijk} - t_{max} \cdot x_{jk} \leq 0 \qquad \forall i, j, k$

$-wi + z_{ijk} \leq 0 \qquad \forall i, j, k$

$w_i - z_{ijk} + t_{max} \cdot x_{jk} \leq t_{max} \qquad \forall i, j, k$

$\sum_{i=2}^{|M|} d_i \leq TSV_{max}$

$d_i \geq \sum_{j=k}^{|M|} z_{jkj} \qquad \forall i, k = i..|M|$

$d_i \geq w_j \qquad \forall i, j = i..|M|$

$\sum_{j=1}^{|M|} z_{jij} \leq W_{max} \qquad \forall i$

$x_{ii} = 1 \qquad \forall i$

$x_{ij} = x_{ji} \qquad \forall i, j$

$1 - x_{ij} \geq x_{ik} - x_{jk} \geq x_{ij} - 1 \qquad \forall i \neq j \neq k$

$y_1 = 1$

$y_i \geq \frac{1}{1-i} \sum_{j=1}^{i-1} (x_{ij} - 1) - M \qquad \forall i > 1$

FIGURE 5.7: ILP model for 3D TAM optimization PSSD.

using the variable v_{ij} for $x_{ij} \cdot \sum_{n=1}^{k_i}(g_{jn} \cdot t_j(n))$. Similarly to Equation (5.9), the variable c_i takes the value of the max function for each die i and the variable u_i represents the product $y_i \cdot c_i$. Because w_j is now a decision variable, $x_{ij} \cdot w_j$ is linearized using a new variable z_{ijk} defined for all i, j, k. The max function is represented by the variable d_i as before. By using the variable z_{ijk}, the TAM width that can be given to each die can be constrained by an upper limit, which is the number of available test pins. This is represented with the following set of inequalities. The complete ILP model for PSSD is shown in Figure 5.7.

$$\sum_{j=1}^{|M|} z_{jij} \leq W_{max} \qquad \forall i \tag{5.27}$$

As before, alterations are made to the ILP model to solve the dual problems PSSDT and

PSSDW. For both PSSDT and PSSDW, as with the hard die dual problems, a maximum test length constraint T_{max} is introduced and the following constraint is added to the problem:

$$\sum_{i=1}^{|M|} u_i \leq T_{max} \tag{5.28}$$

For PSSDT, the constraint on TSVs used is removed and the following objective function is used:

$$\text{Minimize} \sum_{i=2}^{|M|} d_i.$$

For PSHDT, the constraint on the number of test pins used is removed, which is the inequality involving W_{max}, and variable P is once again utilized. The following inequality defines P.

$$P \geq \sum_{j=1}^{|M|} z_{jij} \qquad \forall i \tag{5.29}$$

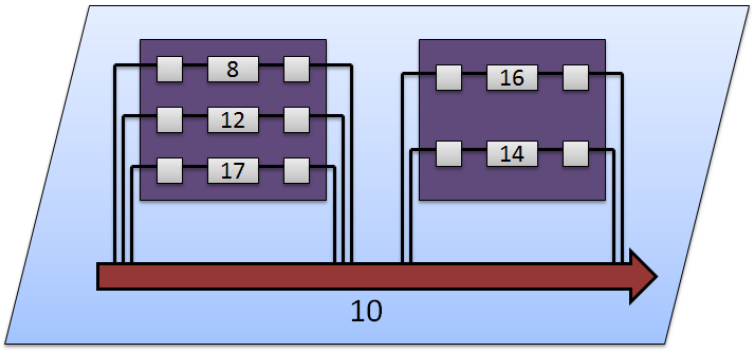
Our objective for PSHDT is therefore to minimize P.

5.2.4 **ILP Formulation for** PSFD

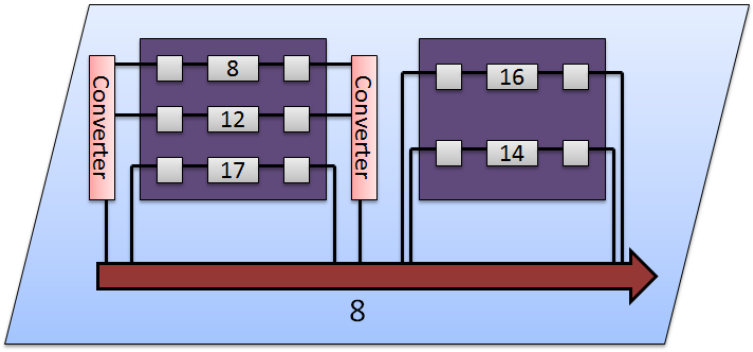
The ILP formulation for 3D-SICs with firm dies is an extension of the model for soft dies. A constraint is added to indicate that the number of test pins used for a die cannot exceed the number of test pins required by the fixed 2D TAM for that die. This constraint is expressed as:

$$w_i \leq wlim_i \qquad \forall i \tag{5.30}$$

where $wlim_i$ is the number of test pins required by the 2D TAM on each die i prior to any serial/parallel conversion. In order to accurately determine test lengths for the dies using serial/parallel conversion, the control-aware TAM design method of [23] is modified to allow the architecture to be fixed in terms of assignment of modules to TAM partitions



(a) Without Conversion



(b) With Conversion

FIGURE 5.8: Illustration of TAM width reduction using serial/parallel conversion.

for a die. The effective widths of the TAM partitions are then iteratively reduced and re-optimized, thereby determining the optimal serial/parallel conversion to use depending on the bandwidth given to that die as shown in Figure 5.2.4. Figure 5.2.4 (a) shows a die prior to TAM width reduction, where ten pins are required to test the die. There are two cores, one with three wrapper chains consisting of the given number of scan flops, and another with two wrapper chains. The amount of time needed to test each core is dependent on the length of the longest wrapper chain and the number of test patterns required by the core. In this example, it is assumed that both cores require the same number of test patterns. Therefore, the TAM width is reduced by two and it is best to combine the wrapper chains of length eight and twelve in the first core, resulting in a longest wrapper chain of twenty as seen in Figure 5.2.4 (b).

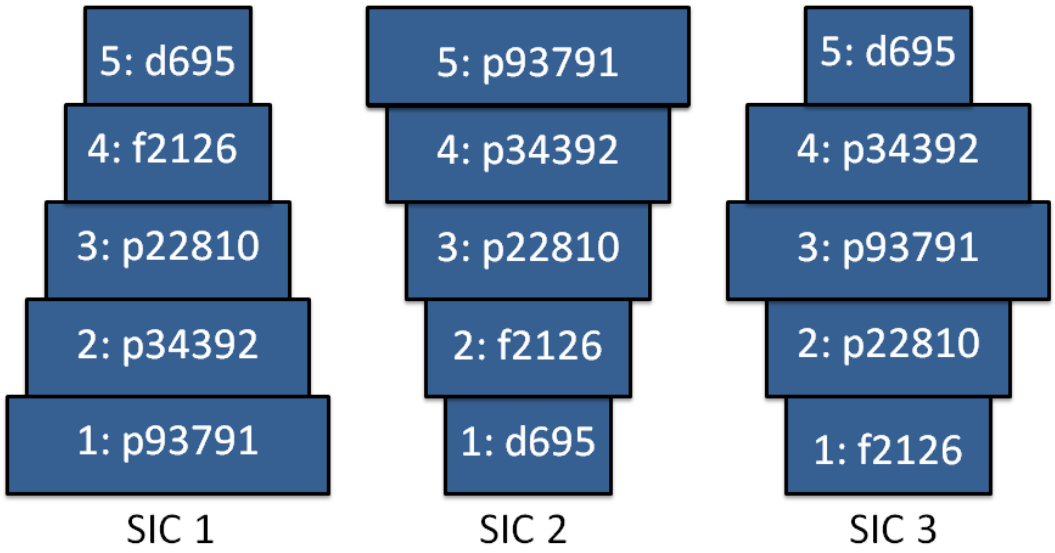


FIGURE 5.9: Three 3D-SIC benchmarks.

5.2.5 **Results and Discussion of ILP-based Final Stack Test Optimization**

In this section, simulation results are shown for the ILP models presented in the previous section. As benchmarks, three 3D-SICs (as shown in Figure 5.9) have been handcrafted from several SOCs from the ITC'02 SOC Test Benchmarks as dies inside the 3D-SICs. The SOCs used are d695, f2126, p22810, p34292, and p93791. In SIC 1, the die are ordered such that the lowest die is the most complex (p93791), with dies increasing in complexity as one moves higher in the stack. The order is reversed in SIC 2, while for SIC 3, the most complex die is placed in the middle of the stack, with dies decreasing in complexity moving out from that die. For equal test bitwidths, the dies lowest in the stack in SIC 1 have the highest test times. In Table 1, f2126 has a slightly higher test time than p22810 because it has a smaller test bitwidth. P22810, however, is still the more complex die from a test perspective. Because SIC 1 and SIC 2 are two extreme cases, they better illustrate the results that are generated. SIC 3 is included to demonstrate test times for an intermediate case of 3D stacking, as opposed to simply the opposite extremes.

To determine the test architecture and test length for a given die (SOC) with a given

Table 5.1: Test lengths and number of test pins for dies as required in PSHD.

Die	**d695**	**f2126**	**p22810**	**p34392**	**p93791**
Test Length (cycles)	96297	669329	651281	1384949	1947063
# of Test Pins	15	20	25	25	30

Table 5.2: Comparison of optimization results between PSHD and a greedy algorithm for SIC 1.

TSV_{max}	W_{pin}	PSHD (ILP)		PSHD (Greedy)		Percentage Difference in Test Length (ILP versus Greedy)
		Test Length (cycles)	Test Schedule	Test Length (cycles)	Test Schedule	
160	30	4748920	1,2,3,4,5	4748920	1,2,3,4,5	0.0
160	35	4652620	1,2,3,4‖5	4652620	1,2,3,4‖5	0.0
160	40	4652620	1,2,3,4‖5	4652620	1,2,3,4‖5	0.0
160	45	3983290	1‖5,2‖4,3	4001340	1,2‖5,3‖4	0.5
160	50	3428310	1‖4,2‖3,5	3428310	1‖4,2‖3,5	0.0
160	55	2712690	1‖2,3‖4,5	2712690	1‖2,3‖4,5	0.0
160	60	2616390	1‖2,3‖4‖5	2712690	1‖2,3‖4,5	3.7
160	65	2616390	1‖2,3‖4‖5	2712690	1‖2,3‖4,5	3.7
160	70	2616390	1‖2‖5,3‖4	2616390	1‖2‖5,3‖4	0.0
160	75	2598340	1‖2‖4,3‖5	2616390	1‖2‖5,3‖4	0.7
160	80	2598340	1‖2‖4,3‖5	2616390	1‖2‖5,3‖4	0.7
160	85	2598340	1‖2‖4,3‖5	2616390	1‖2‖5,3‖4	0.7
160	90	2598340	1‖2‖4,3‖5	2616390	1‖2‖5,3‖4	0.7
160	95	2043360	1‖2‖3‖4,5	2616390	1‖2‖5,3‖4	28.0
160	100	2043360	1‖2‖3‖4,5	2043360	1‖2‖3‖4,5	0.0

TAM width, the control-aware TAM design method in [23] has been used. Control-aware TAM design takes into account the number of scan-enable signals required for independent testing of TAMs in the architecture. For PSHD (3D-SIC with hard dies), the test lengths (cycles) and TAM widths for different dies are listed in Table 5.1. Note that test pins were assigned to dies based on their sizes in order to avoid very large test lengths for any individual die.

The minimal achievable test length for the hard die stack can be seen to be 1947063 cycles, which occurs when all dies are tested in parallel with one another. To investigate the effect of achieving this test length for a 3D stack, consider SIC 1 and SIC 2. For both SICs, this architecture requires 115 test pins on the bottom die. For SIC 1, this requires 195 test TSVs. For SIC 2, this requires 265 test TSVs.

Table 5.3: Comparison of optimization results between PSSD and a greedy algorithm for SIC 1.

TSV_{max}	W_{pin}	PSSD (ILP)		PSSD (Greedy)		Percentage Difference in Test Length (ILP versus Greedy)
		Test Length (cycles)	Test Schedule	Test Length (cycles)	Test Schedule	
140	30	4795930	1‖2‖3‖4,5	7842000	1‖2,3,4‖5	63.5
140	35	4237100	1‖2‖3‖4,5	7633580	1‖3,2‖4,5	80.1
140	40	3841360	1‖2‖3‖4,5	6846400	1‖3,2‖4,5	78.2
140	45	3591550	1‖2‖3‖4,5	6379510	1‖2‖3,4‖5	77.6
140	50	3090720	1‖2‖3‖4,5	6041270	1‖2‖3,4‖5	95.5
140	55	2991860	1‖2‖3‖4‖5	5873430	1‖2‖3‖4,5	96.3
140	60	2873290	1‖2‖3‖4,5	5821900	1‖2‖3‖4,5	102.6
140	65	2784050	1‖2‖3‖4‖5	5705410	1‖2‖3‖4,5	104.9
140	70	2743320	1‖2‖3‖4‖5	5638140	1‖2‖3‖4,5	105.5
140	75	2629500	1‖2‖3‖4‖5	5638140	1‖2‖3‖4,5	114.4
140	80	2439380	1‖2‖3‖4‖5	5496200	1‖2‖3‖4,5	125.3
140	85	2402330	1‖2‖3‖4‖5	5447190	1‖2‖3‖4,5	126.7
140	90	2395760	1‖2‖3‖4‖5	5447190	1‖2‖3‖4,5	127.4
140	95	2383400	1‖2‖3‖4‖5	5447190	1‖2‖3‖4,5	128.5
140	100	2369680	1‖2‖3‖4‖5	5351480	1‖2‖3‖4,5	125.8

Table 5.2 compares optimal results produced using ILP with those produced using a greedy algorithm for PSHD SIC 1. The greedy algorithm attempts to combine dies in parallel-tested sets, starting with those that would lead to the greatest reduction in test time. If more than one combination results in the same reduction, it prioritizes those combinations that result in the smallest test resource use. Compared to PSSD and PSFD, PSHD is a less complex problem. Thus, the greedy algorithm is capable of producing results that are sometimes optimal, although it often does not result in the minimum test time.

Table 5.3 shows the information in Table 5.2 for problem PSSD. The soft die problem is more difficult to solve using a greedy heuristic. The heuristic algorithm for PSSD uses the greedy algorithm from PSHD as a subroutine in test-architecture optimization. It begins with an assignment of an equal number of test pins to each of the dies and optimizes the 2D and 3D TAM under these constraints. It then randomly adds and removes random numbers of test pins from each die, each time balancing the result to use the maximum number of test pins, and optimizes again. It checks for reductions in test time, returning to the best

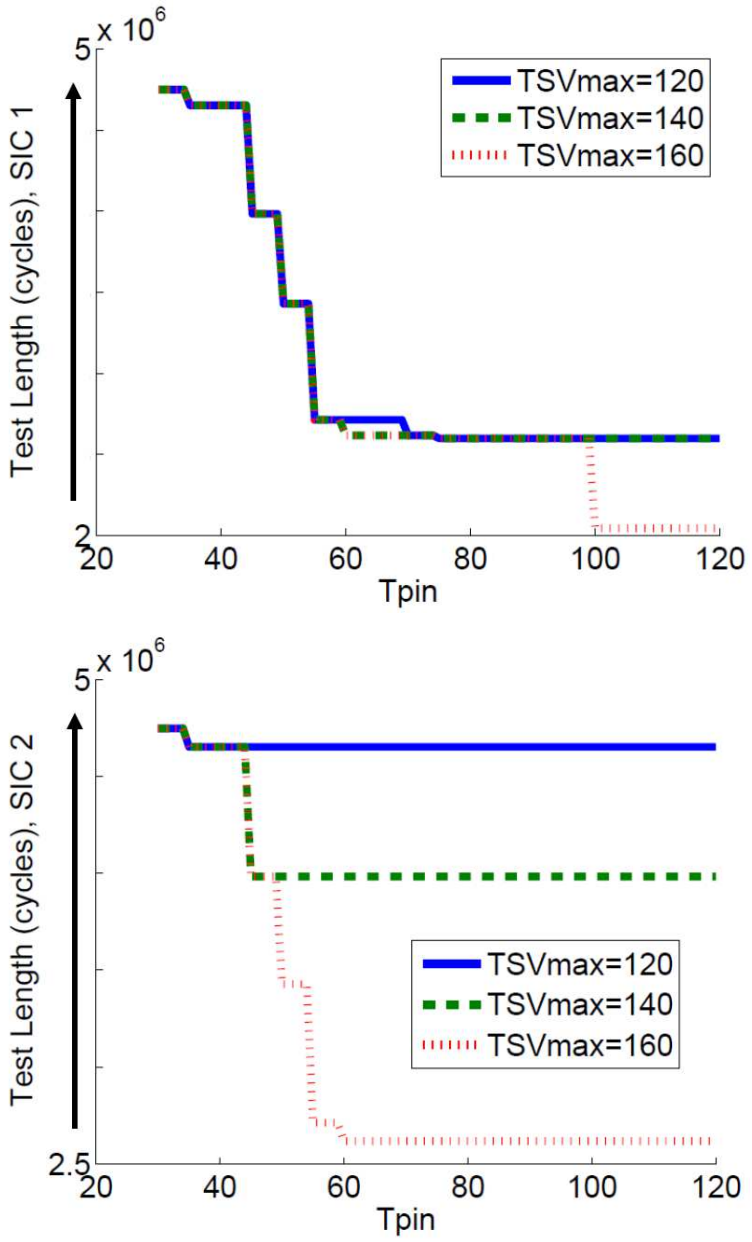


FIGURE 5.10: The test length with respect to TSV_{max} for SIC 1 and SIC 2 with hard dies.

solution so far if no test time reduction is produced or constraints are violated. It terminates after 10000 iterations of no improvement. As can be seen, the optimal ILP solution tends to be much better in terms of test length than the heuristic solution, simply because the solution space is so large.

It is useful to briefly discuss how the test architecture for a soft die is built using the

information produced from the ILP optimization. Prior to using the ILP model, 2D TAM architectures that minimize test time are produced for each die in the stack assuming a wide range of TAM widths available to that die. In this sense, the 2D architecture is already completed and a choice must be made regarding which architecture is to be used. The ILP formulation provides data regarding which TAM width to use for each die and the test schedule for testing the die, which lets the designer know which dies need to be tested in parallel. With this information, the design of the 3D TAM is elementary. The integrator simply provides the appropriate TAM width to each die, assuring that the number of test elevators between each die is sufficient for the bandwidth required for any parallel tests.

Table 5.4: Simulation results for PSHD.

TSV_{max}	W_{pin}	PSHD SIC 1			PSHD SIC 2			PSHD SIC 3		
		Test Length (cycles)	Test Schedule	Reduction (%)	Test Length (cycles)	Test Schedule	Reduction (%)	Test Length (cycles)	Test Schedule	Reduction (%)
160	30	4748920	1,2,3,4,5	0.00	4748920	1,2,3,4,5	0.00	4748920	1,2,3,4,5	0.00
160	35	4652620	1,2,3,4‖5	2.03	4652620	1‖2,3,4,5	2.03	4652620	1‖5,2,3,4	2.03
160	40	4652620	1,2,3,4‖5	2.03	4652620	1‖3,2,4,5	2.03	4652620	1‖5,2,3,4	2.03
160	45	3983290	1‖5,2‖4,3	16.12	3983290	1‖3,2‖4,5	16.12	3983290	1‖4,2‖5,3	16.12
160	50	3428310	1‖4,2‖3,5	27.81	3428310	1,2‖5,3‖4	27.81	3428310	1,2‖4,3‖5	27.81
160	55	2712690	1‖2,3‖4,5	42.88	2712690	1,2‖3,4‖5	42.88	2712690	1,2‖3,4‖5	42.88
160	60	2616390	1‖2,3‖4‖5	44.91	2616390	1‖2‖3,4‖5	44.91	2616390	1‖4‖5,2‖3	44.91
160	65	2616390	1‖2,3‖4‖5	44.91	2616390	1‖2‖3,4‖5	44.91	2616390	1‖4‖5,2‖3	44.91
160	70	2616390	1‖2‖5,3‖4	44.91	2616390	1‖2‖3,4‖5	44.91	2616390	1‖2‖3,4‖5	44.91
160	75	2598340	1‖2‖4,3‖5	45.29	2616390	1‖2‖3,4‖5	44.91	2616390	1‖2‖3,4‖5	44.91
160	80	2598340	1‖2‖4,3‖5	45.29	2616390	1‖2‖3,4‖5	44.91	2616390	1‖2‖3,4‖5	44.91
160	85	2598340	1‖2‖4,3‖5	45.29	2616390	1‖2‖3,4‖5	44.91	2616390	1‖2‖3,4‖5	44.91
160	90	2598340	1‖2‖4,3‖5	45.29	2616390	1‖2‖3,4‖5	44.91	2616390	1‖2‖3,4‖5	44.91
160	95	2598340	1‖2‖4,3‖5	45.29	2616390	1‖2‖3,4‖5	44.91	2616390	1‖2‖3,4‖5	44.91
160	100	2043360	1‖2‖3‖4,5	56.97	2616390	1‖2‖3,4‖5	44.91	2616390	1‖2‖3,4‖5	44.91
160	105	2043360	1‖2‖3‖4,5	56.97	2616390	1‖2‖3,4‖5	44.91	2616390	1‖2‖3,4‖5	44.91

Take for example the information provided for the ILP optimization of soft dies in Table 5.3 when W_{pin} is 60. Although not shown in the table, the ILP optimization provides the designer with the following widths for each die (these values show the number of test pins used by that die, so the TAM width is half the given value): W_1 = 30, W_2 = 20, W_3 = 6, W_4 = 4, W_5 = 30. The designer sees that Dies 1-4 are tested in parallel, followed by Die 5. Because the width of the top die dominates this stack, TSV routing simply requires 30 test elevators between each die. Die 1, 2, 3, and 4 each utilize a different test pin of the 60 test pins available, and Die 5 can utilize any 30 of the test pins. Which pin is routed to which die is up to the best judgment of the designer, as is wire routing and the like.

For a fixed TSV_{max} and range of W_{max}, Table 5.4 presents representative results for PSHD for the three benchmark 3D-SICs using hybrid TestRail architectures [46]. Additional values for TSV_{max} could be considered, but they do not provide any new insights. For PSHD and its comparison to PSHD, optimizations were done using hybrid TestBus [46] architecture for variety. The ILP models were solved using the XPRESS-MP tool [49]. In this table, Column 1 shows the maximum number of TSVs allowed (TSV_{max}), while Column 2 represents the number of available test pins W_{max}. Columns 3, 6 and 9 represent the total test length (cycles) for the stack for 3D-SIC 1, 2 and 3 respectively. Columns 4, 7, and 10 show the resulting test schedule for the 3D-SICs, where the symbol "||" indicates parallel testing of dies, and a "," represents serial testing. Finally, Columns 5, 8, and 11 show the percent decrease in test length over the serial testing case for the three 3D-SICs. From Table 5.4 it can be seen that compared to serial testing of all dies (first row in the table), the proposed method obtains up to 57% reduction in test length. Note that although identical test lengths were obtained for SIC 2 and SIC 3 for $TSV_{max} = 160$, different TAM architectures and test schedules were obtained from the optimization algorithm (see Columns 4 and 10).

For a different number of TSVs (TSV_{max}), Figure 5.10(a) and Figure 5.10(b) show the variation in test length T with an increase in number of test pins W_{max} for SIC 1 and SIC 2. From the figures, it can be seen that both TSV_{max} and W_{max} determine which dies should be tested in parallel, and thus determine the total test length for the stack. For a given value of TSV_{max}, increasing W_{max} does not always decrease the test length, showing the presence of pareto-optimal points. These have an important impact on test resource allocation, because

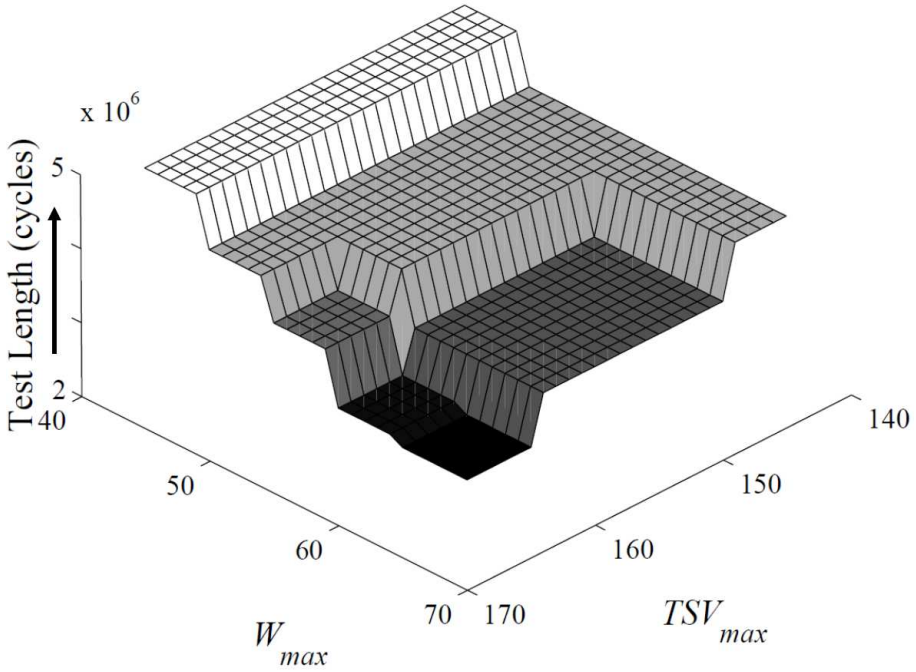


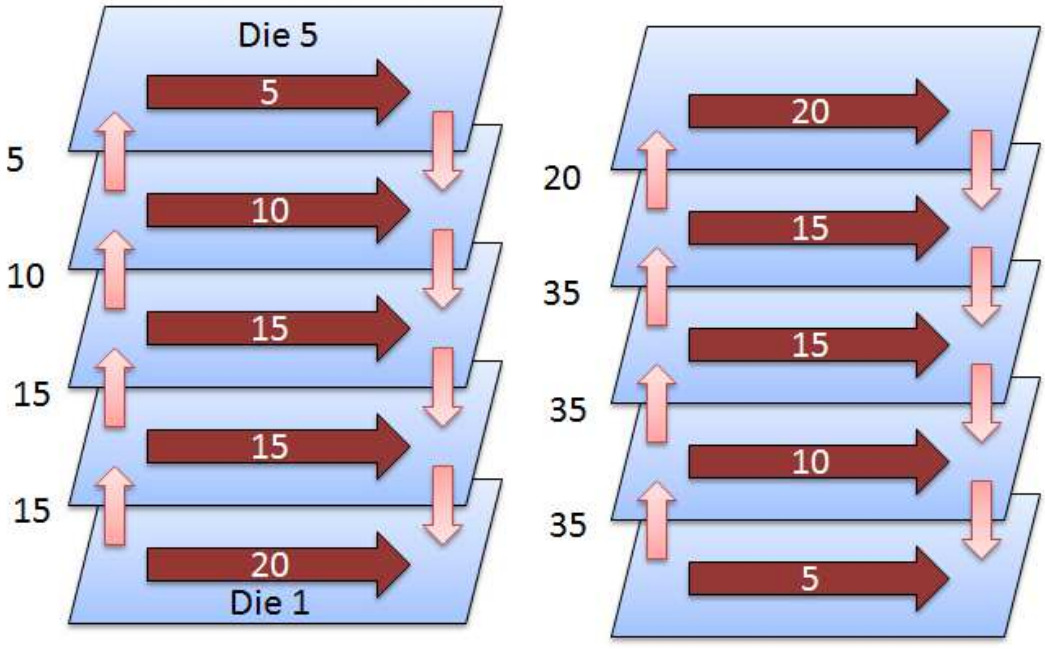
FIGURE 5.11: Variation in test length with W_{max} and TSV_{max} for SIC 2 with hard dies.

Table 5.5: Comparisons between PSHD and PSFD.

Optimization Framework	W_{max}	Test Length (cycles)	Reduction (%)	Test Schedule	No. of Test Pins used per Die
PSHD	35	4678670	0	1,2,3,4‖5	30,24,24,20,14
	44	4009340	0	1‖4,2,3‖5	30,24,24,20,14
	50	3381720	0	1‖3,2‖5,4,	30,24,24,20,14
	60	2658750	0	1‖5,2‖3,4,	30,24,24,20,14
	80	2658750	0	1‖5,2‖3,4,	30,24,24,20,14
PSFD	35	3828490	18.17	1‖4,2‖3,5	28,24,10,7,14
	44	2875900	28.27	1‖2,3‖4,5	28,16,24,18,14
	50	2641060	21.90	1‖2,3‖4‖5	30,18,24,18,4
	60	2335780	12.15	1‖2‖3‖4,5	28,16,10,6,14
	80	1971400	25.85	1‖2‖3‖4‖5	30,24,10,8,8

test resources for a target test length should be provided only to the extent that they align with the first point in a pareto-optimal plateau.

Figure 5.11 shows the variation in test length for SIC 2 when both TSV_{max} and W_{max} are varied. From the figure, it can be seen that a small increase in the number of test pins W_{max} for a given TSV_{max} reduces test length significantly, while to achieve the same reduction in test length with a fixed number of test pins W_{max}, a large increase in TSV_{max} is required.



(a) SIC 1 (b) SIC 2

FIGURE 5.12: Example of optimization for SIC 1 versus SIC 2.

Figure 5.12 demonstrates the differences between SIC 1 and SIC 2 during optimization. Two 3D stacks with five dies each are shown, with TAM widths displayed on each die and the number of TSVs used between each die shown to the left of the stack. Figure 5.12(a) shows the number of TSVs needed to test Die 1 and Die 2 in parallel followed by Dies 3, 4, and 5 for SIC 1. It is desirable to test Die 1 and Die 2 in parallel because they are the dies with the longest test lengths. This requires 90 TSVs. For SIC 2, this requires 250 TSVs as shown in Figure 5.12(b). This demonstrates why optimization produces better results for SIC 1 than for SIC 2.

Table V compares results for PSHD and PSFD for TestBus architectures. Table V demonstrates that by adding serial/parallel conversion of TAM inputs to hard dies, a reduction in test length as high as 28% can be obtained. This is because the conversion allows for an increase in the test length of individual die in order to minimize the overall SIC test length during test schedule optimization. It should also be noted that the test schedules and the number of test pins utilized for each die differ considerably between the hard-die and firm-die problem instances. Compared to a hard die, a firm die requires a small amount of extra hardware to convert a narrow TAM at the die input to a wider die-internal TAM, and vice

versa. The area cost of this additional hardware is negligible compared to the die area and the hardware required for the core and die wrappers.

Figure 5.13 shows comparative test lengths between PSHD and PSFD when W_{max} is varied, for two values of TSV_{max} and for two SICs. It is impossible to test the hard dies in these cases using fewer than 30 test pins without using serial/parallel conversion. As fewer test pins are used, the test lengths for individual dies greatly increase, resulting in a sharp increase in overall test length below certain values of W_{max}. It is important to note that the test length for a SIC with hard dies can never be shorter than the test length for the same SIC with firm dies; at best it can be equal. This is because, in the worst case with respect to test length, the optimization for firm dies is equivalent to the optimization for hard dies, i.e., no serial/parallel conversion is carried out. It can be seen that the use of serial/parallel conversion hardware can result in less use of test resources and shorter test time compared to hard dies without conversion. This observation is particularly valid in SIC 2, where the position of dies in the stack limits test time reduction.

Figure 5.15 shows PSHDT (3D-SIC with hard dies and TSV-count optimization) results for SIC 1. Under tight test length constraints, a solution to the optimization is unattainable for smaller values of W_{max}. In Figure 5.15, for example, a W_{max} value of 30 will not produce a feasible test architecture until the test length is above 470000 cycles. Once an optimized architecture is achievable, the minimum number of TSVs used is generally the same regardless of the value of W_{max}. There are two reasons for this. The first is that there are only a few configurations of the 3D TAM for the hard-die stack, and multiple configurations will give the same minimal TSV value. This is only a partial explanation, however, as equal minimal-TSV values for various W_{max} values are seen for soft dies as well. The primary reason for the results of Figure 5.15 is that in order to minimize the number of TSVs used by the stack, the ILP solver will tend toward testing all dies in series with each other. If this cannot be done, then it will attempt to test only those dies with the smallest TAM width in parallel. This test configuration—tending toward serial testing—also happens to be the configuration that results in the fewest number of test pins used. This is why the TSV-count values in Figure 5.15 overlap even for tight test pin constraints—minimizing the TSVs used also tends to minimize the number of test pins used. This is seen for both

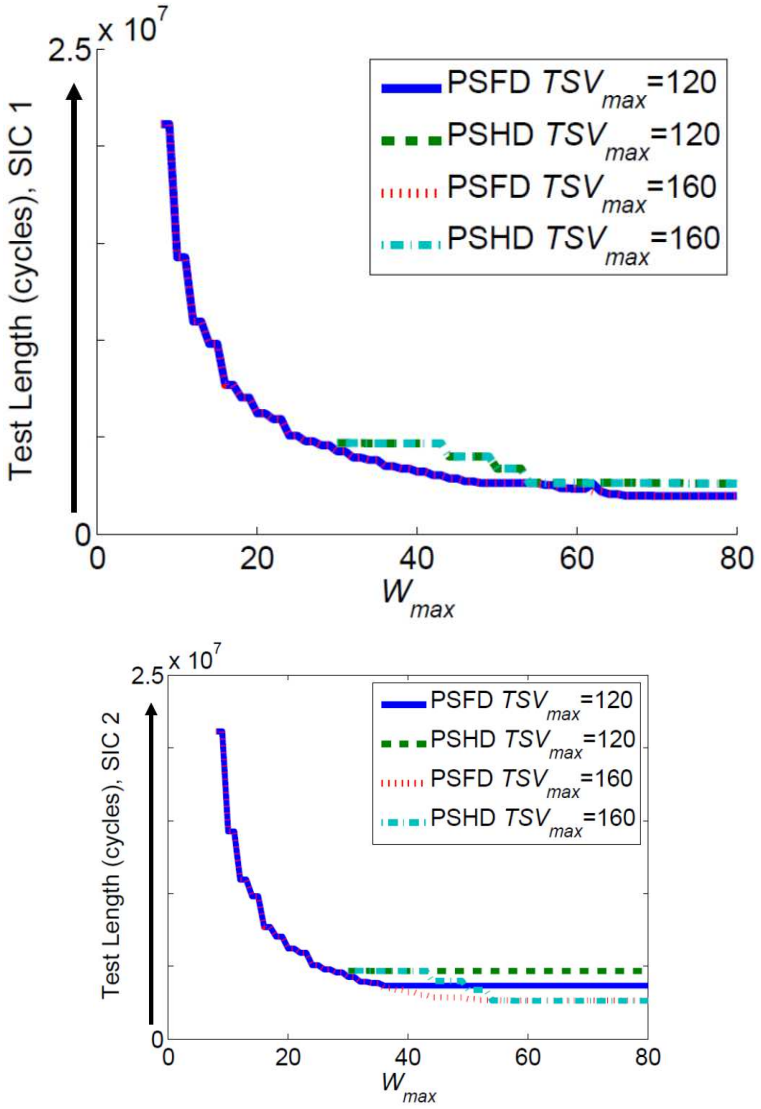


FIGURE 5.13: Comparison of variation in test length with W_{max} for SIC 1 and SIC 2 between firm dies and hard dies.

SIC 1 and SIC 2 (Figure 5.16), although the number of TSVs needed for testing in the less-optimized SIC 2 stack is higher. Results for PSHDW (3D-SIC with hard dies and test pin-use optimization) are also as expected—if minimizing TSV use also tends to minimize test pin use, then minimizing test pin use should also tend to minimize TSV use. As such, overlapping minimal test pin use for both tight and loose TSV_{max} constraints is again observed. Note that optimizing for minimum TSV or test pin use tends toward serial testing. Therefore, these optimizations result in very different test architectures than optimizing for

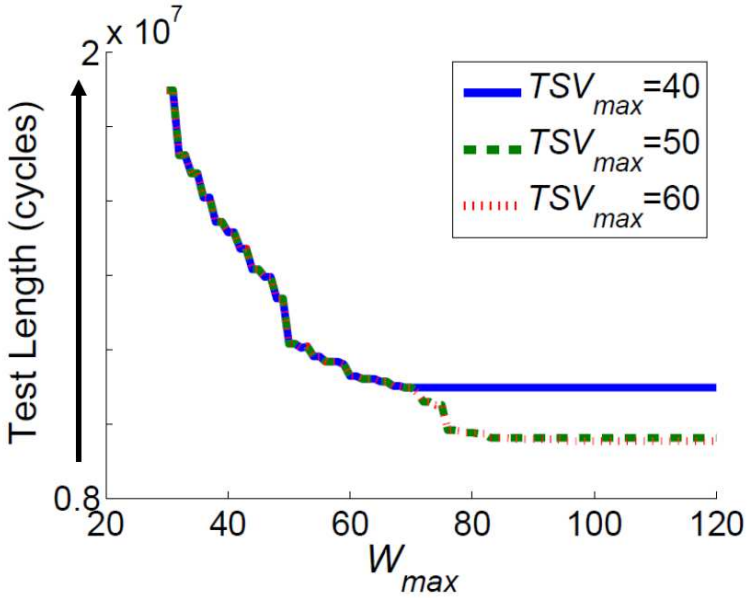


FIGURE 5.14: Variation in test length with W_{max} for SIC 1 with soft dies.

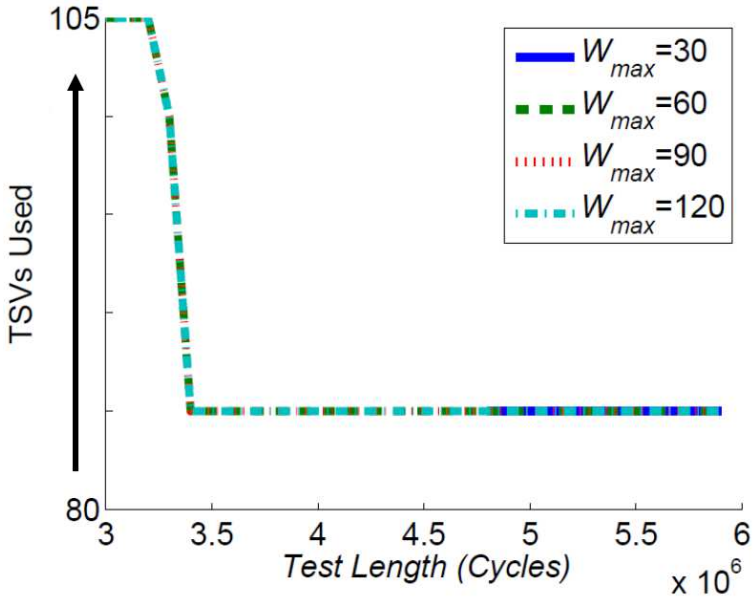


FIGURE 5.15: Variation in TSVs used with T_{max} for SIC 1 with hard dies.

test time. In contrast, solutions that minimize test time tend to result in the parallel testing of many dies, as can be seen for PSSD in Table 5.3.

For PSSD (3D-SIC with soft dies), Pareto-optimality is almost non-existent when W_{max} is varied; see Figure 5.14. This is due to the fact that as dies in the stack are soft, it is

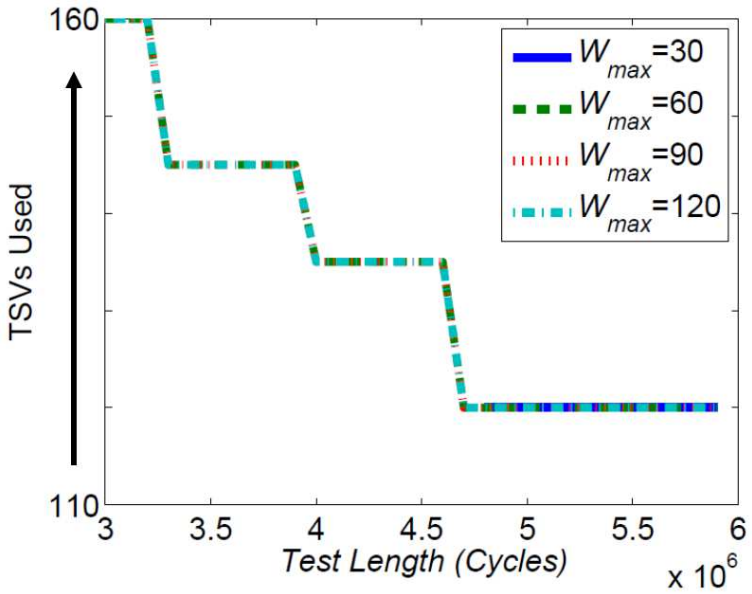


FIGURE 5.16: Variation in TSVs used with T_{max} for SIC 2 with hard dies.

always possible to find one die for which adding an extra test pin reduces the overall test length. Some Pareto-optimal points can be identified for SIC 2. This is because the most complex dies in a stack tend to be the bottleneck in reducing test length. Because these dies are stacked toward the top of the stack in SIC 2, TSV constraints are more restrictive; the addition of test pins to these dies requires more TSVs and TestElevators throughout the stack. However, for PSSD, although varying W_{max} does not create Pareto-optimal points, varying TSV_{max} results in various Pareto-optimal points as shown in Figure 5.17. Note that this effect is more pronounced in SIC 2 than in the other 3D-SICs. This is because the addition of test pins to the bottleneck die (at the highest layer) introduces a larger TSV overhead than in the other 3D-SICs. Furthermore, as long as W_{max} is sufficient, TSV_{max} is the limiter on test length. For PSHD, PSSD, and PSFD, the stack configuration (SIC 1) with the largest die at the lowest layer and the smallest die at the highest layer is the best for reducing test length while using the minimum number of TSVs. Figure 5.20 shows a comparison of optimized test lengths for soft dies between SIC 2 and SIC 3. As shown, SIC 3 leads to test lengths lower than or equivalent to SIC 2 at higher values for W_{max}. However, under tight test pin constraints SIC 2 results in better test lengths.

Figure 5.18 shows PSSDT (3D-SIC with soft dies and TSV-count optimization) results for SIC 1. Compared to the optimizations for hard dies, less pareto-optimality is present

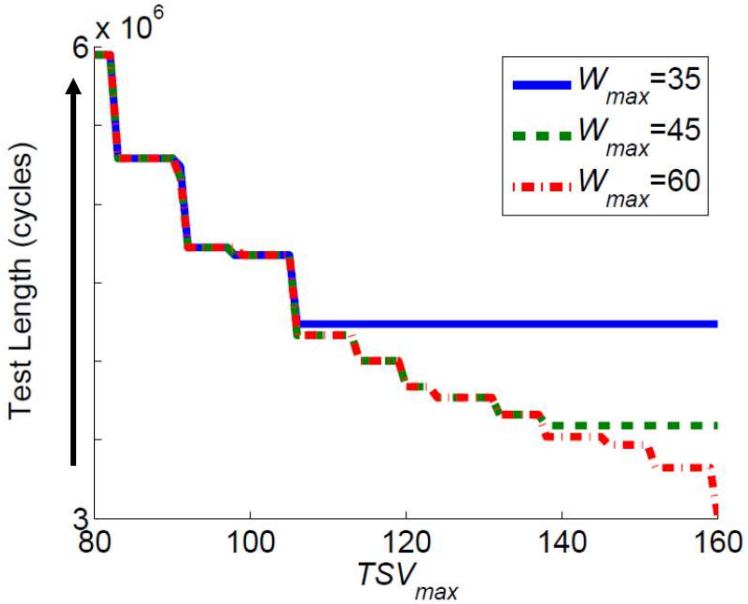


FIGURE 5.17: Variation in test length with TSV_{max} for SIC 2 with soft dies.

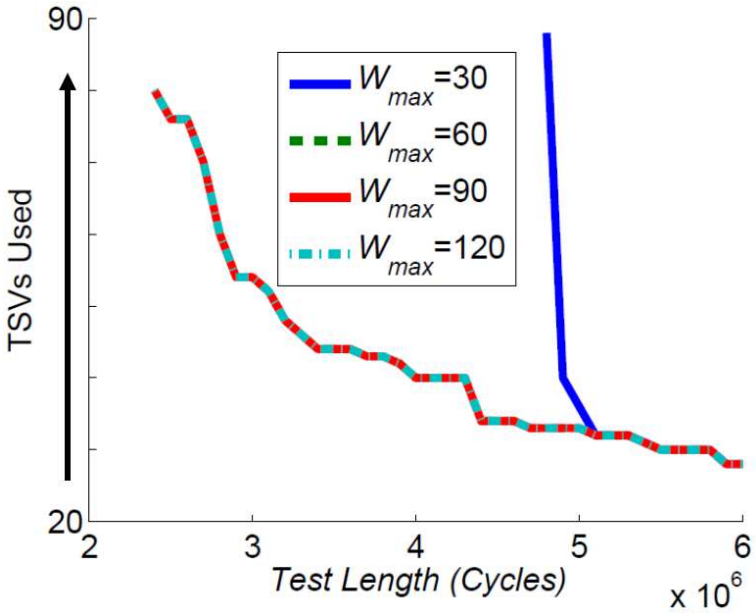


FIGURE 5.18: Variation in TSVs used with T_{max} for SIC 1 with soft dies.

as expected, because more leeway exists in the soft model. For the reasons described earlier, similar results as in Figures 5.15 are seen. Similar observations are made for test pin optimization, as seen in Figure 5.19, which shows PSSDW (3D-SIC with soft dies and test pin-use optimization) results for and SIC 2.

Consider the optimization of SIC 2 versus SIC 3 for problem PSSD; see Figure 5.20.

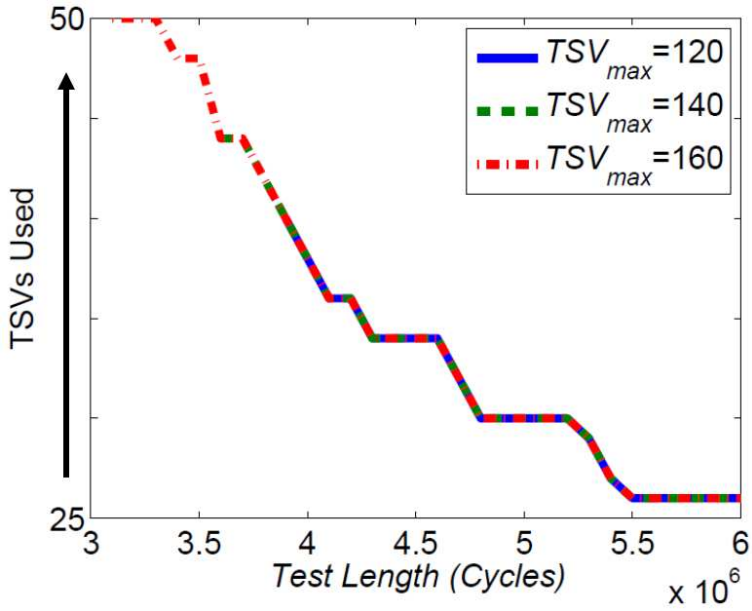


FIGURE 5.19: Variation in test pins used with T_{max} for SIC 2 with soft dies.

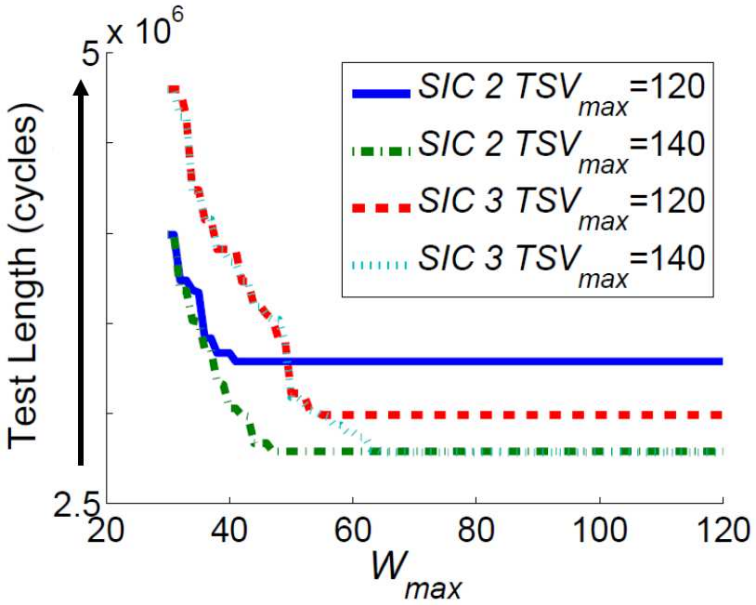


FIGURE 5.20: Comparison of test lengths for PSSD for SIC 2 and SIC 3

For hard dies, as shown in Table 5.4, similar test times with different architectures were produced. This is because the hard die model is too limited with 3D constraints to lead to different test times for SIC 2 and SIC 3. This is not the case for soft dies, where the additional degree of freedom leads to different architectures and better test times in SIC 3 when compared to SIC 2, as expected. From a design perspective, this means that a stack

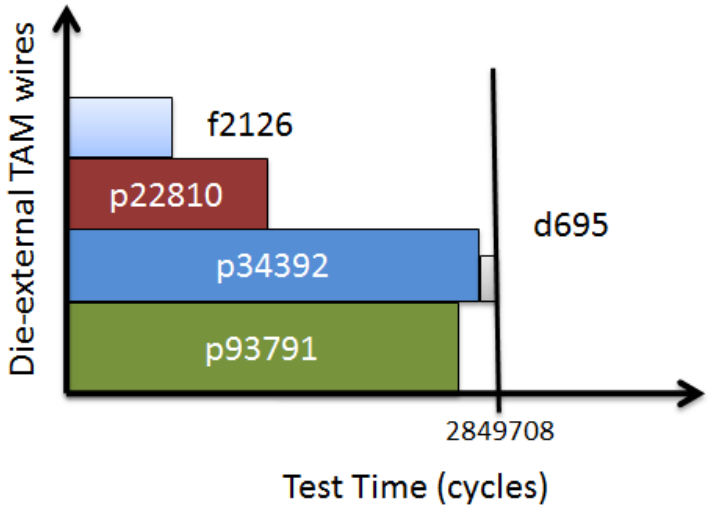


FIGURE 5.21: Visualization of test schedule for SIC 1 with hard dies, TSV_{max} = 160, and W_{pin} = 100.

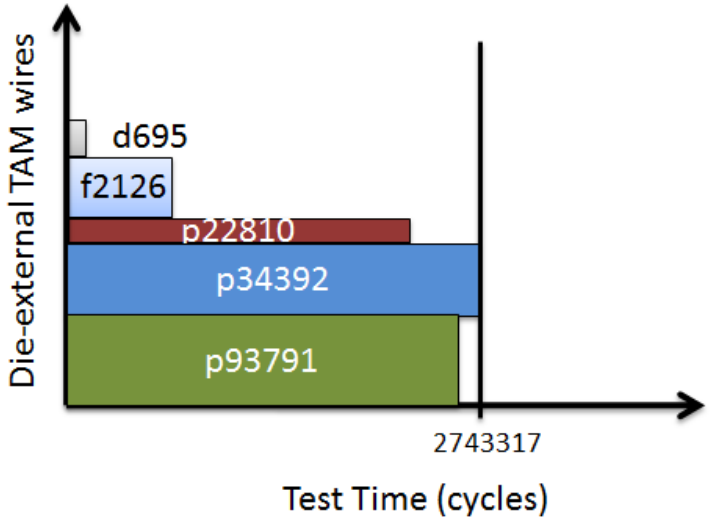


FIGURE 5.22: Visualization of test schedule for SIC 1 with firm dies, TSV_{max} = 160, and W_{pin} = 100.

layout with lower test time can be achieved if one can generally keep the most complex dies in lower layers in the stack.

Figure 5.21 demonstrates the optimization for hard dies in a 3D stack with TSV_{max} of 160 and W_{pin} of 100. As can be seen, with fixed dies TAMs there is limited opportunity for

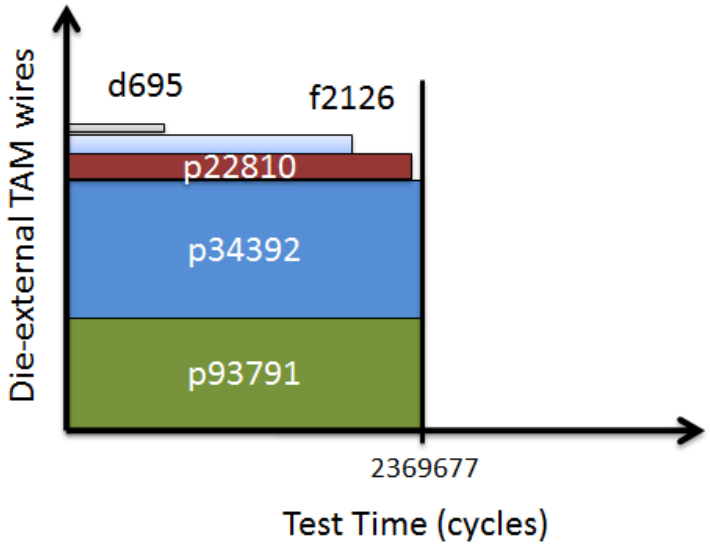


FIGURE 5.23: Visualization of test schedule for SIC 1 with soft die, TSV_{max} = 160, and W_{pin} = 100.

optimization given that the test lengths of each individual die cannot be altered. Thus, the test schedule has undesirable white spaces denoting wasted test resources. The use of firm and soft dies helps to eliminate wasted resources, as seen in Figure 5.22 and Figure 5.23, respectively. Firm dies allow for a modest reduction in test length by decreasing the number of TAM wires allotted to p22810 and utilizing those wires instead to test all the dies in parallel. With soft dies, even more test length is saved.

Finally, consider the scalability of the ILP-based optimization method by determining the stack size (number of layers) for which the ILP solver takes more than one day of CPU time. For PSHD, M = 16 is obtained, while for PSSD, M = 10. Because these values of M are unlikely for realistic stacks, it can be concluded that the proposed method is scalable and practical.

5.3 Extending Test Optimization for Multiple Test Insertions and Interconnect Test

In Section 5.2, optimization techniques were introduced for minimizing the test time for final stack test. A global limit was set on the number of dedicated TSVs to be used for test

access and constraints were imposed on test bandwidth due to a limited number of test pins on the lowest die in the stack. This section extends the mathematical model to allow for the optimizer to consider any or all post-bond test insertions desired by the 3D test engineer. This is a more general approach that still allows for optimization for the final stack test alone, or any number of post-bond test insertions. Furthermore, the test times for TSVs and die-external logic will be considered in the extended optimization framework. Finally, TSV limits for the 3D TAM are applied on a more realistic per-layer basis.

5.3.1 Modifying the Optimization Problem Definition

As mentioned earlier, the lowest die in a 3D SIC is usually directly connected to chip I/O pins, and therefore it can be tested using package pins. To test the other dies in the stack, TAMs that enter the stack from the lowest die should be provided. To transport test data up and down the stack, *test elevators* must be included on each die except for the highest die in the stack [58]. The number of test pins and test elevators, as well as the number of TSVs used, affect the total test time for the stack.

Many new manufacturing steps are needed for the production of 3D-SICs than for 2D-SICs, including TSV creation, wafer thinning, alignment, and bonding. These steps can introduce possible defects that do not arise for 2D-SICs [60]. Such defects include incomplete fill of TSVs, misalignment, peeling and delamination, and cracking due to back-side grinding. It is difficult to carry out pre-bond testing of TSVs due to limitations in probe technology and the need for contactless probing. Thus, post-bond partial-stack and in-stack TSV testing are needed to detect these new defects and reduce defect escapes.

The following two examples, for hard dies and soft dies, respectively, highlight the limitations of optimization techniques that are oblivious to multiple test insertions. In Section 5.2, optimization decisions were made considering only the final stack test after all dies have been bonded. These models cannot be directly applied to optimize for multiple test insertions, as shown in Figure 5.24.

Example 1: Consider an attempt to optimize for two test insertions, the first with three dies on the stack as in Figure 5.24(a) and the second with all four dies as in Figure 5.24(b), by building upon the test architecture created for the first test insertion. A global TSV limit

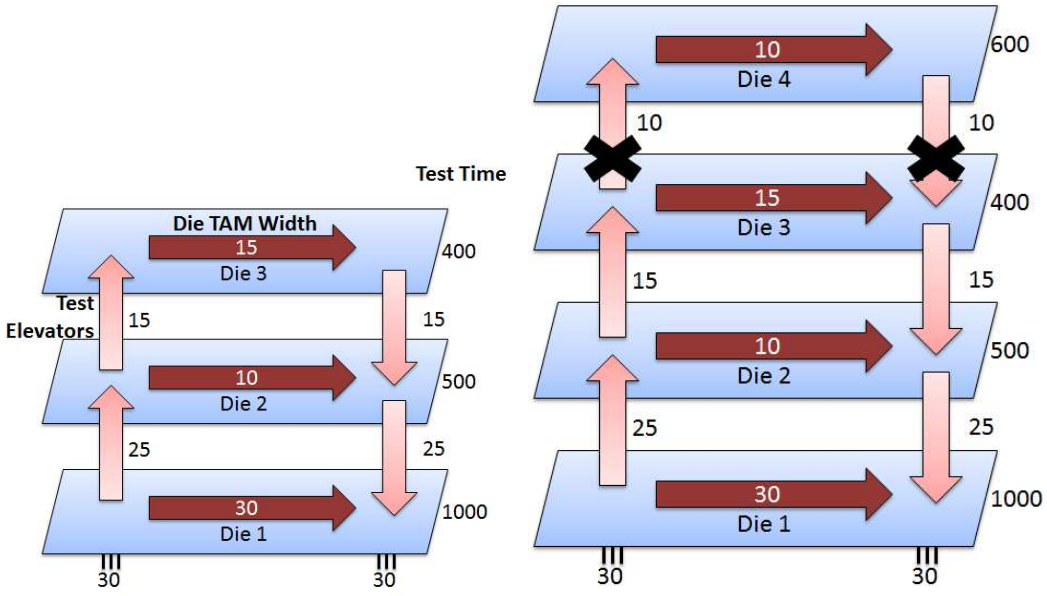


FIGURE 5.24: Example 1: 3D-SIC with three hard dies.

of 90 is imposed for this example. As seen in Figure 5.24(a), Die 1 is tested in series with Die 2 and Die 3, resulting in a minimized test time of 1500 cycles and the use of 80 TSVs as test elevators. It is then attempted to add the fourth die while preserving the previous architecture, but this leads to a violation of the TSV limit because 100 TSVs are now needed (the test elevators marked in the figure exceed the mandated upper limit). If instead optimization starts from the complete stack and works backwards, suboptimal results for the test time are obtained. This is because the architecture created for the final stack test cannot support optimal test schedules for other intermediate test insertions. Therefore, new optimization techniques are needed for partial stack testing. The following example highlights this problem.

Example 2: It can also be shown that optimizing only for the final stack test does not result in optimum test times for multiple test insertions. Consider an SIC with three dies from the ITC'02 SOC Benchmarks [24] as shown in Figure 5.25. There are 40 test pins available to the stack and a limit of 40 TSVs per side of each die as the maximum number of TSVs to use for the test infrastructure. The test time for each die is determined by its test architecture, which in this example relies on daisychaining.

Figure 5.25(a) shows the resulting test elevator widths and the number of TSVs used if

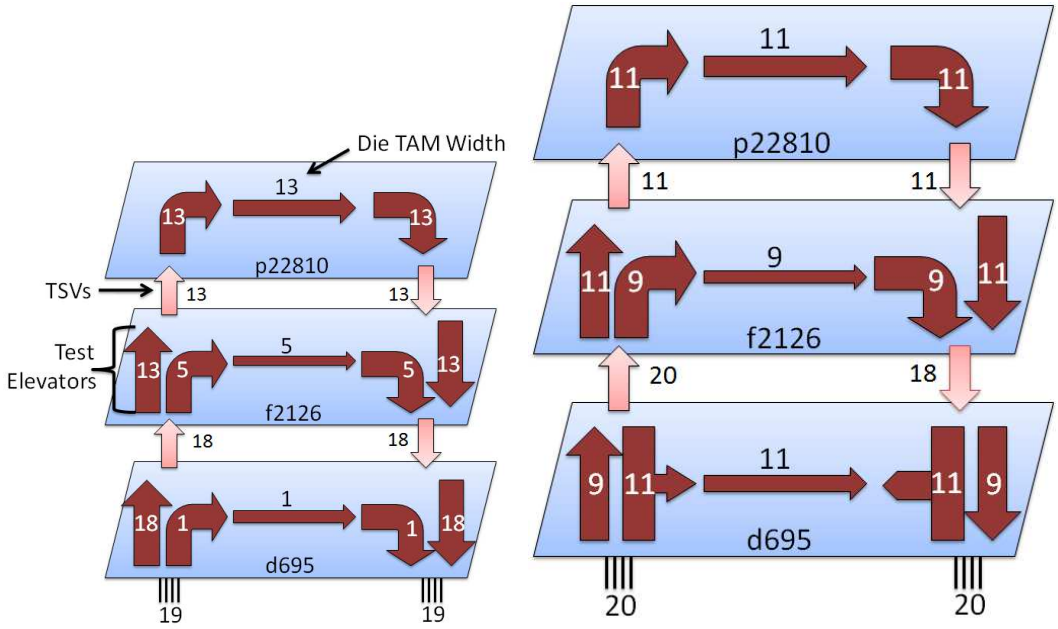


FIGURE 5.25: Example 2: 3D-SIC with three soft dies.

the stack is optimized to reduce the test time of the final stack test after all dies have been bonded. This architecture allows for the testing of all three dies in parallel, which results in a final stack test time of 1313618 cycles with individual die test times of 1313618, 1303387, and 658834 cycles, from top to bottom. The architecture uses 38 out of the 40 allowed test pins and 64 TSVs. When both possible stack tests are considered, with the first stack test done after the middle die is bonded to the lower die and the final stack test, the total test time becomes 2617005 cycles (Die 1 and Die 2 are tested in parallel for the first stack test).

Figure 5.25(b) shows the test architecture created if all stack tests are considered during optimization, which uses all 40 test pins and 62 TSVs. This architecture allows Die 1 and Die 2 to be tested in parallel for the first stack test, and then Die 2 and Die 3 to be tested in parallel after Die 1 is tested for the final stack test. The test times for the dies from top to bottom are 1338480, 700665, and 75004, respectively. This results in a final stack test time of 1413484 cycles, an increase in time over the previous example. However, when considering all stack tests, this architecture results in a total test time of 2114149 cycles (700665 cycles for the first stack test), a considerable reduction over the previous example. This example clearly shows the impact of the optimization target on the architecture and test

time. Therefore, a test-architecture optimization algorithm for 3D-SICs has to minimize the test time while taking into account 3D design constraints as well as all of the post-bond tests that will be run on the 3D-SIC. The resulting optimized 3D TAM must also allow for different test schedules for each post-bond test.

With the above examples as motivation for better optimization methods, the problems addressed in this section can be formally stated. The problem of test-architecture optimization for 3D-SICs with hard dies for all stack test insertions is defined as follows.

3D-SIC with Hard Dies and Multiple Test Insertions (P^H_{MTS})

Givens include a stack with a set M of dies and total number of test pins W_{max} available for test. For each die $m \in M$, its tier number l_m in the stack, the number of test pins w_m ($w_m \leq W_{max}$) required to test the die, the associated test time t_m, and a maximum number of TSVs ($TSVmax_m$) that can be used for TAM design between die $m-1$ and m ($m > 1$) are given. The goal is to determine an optimal TAM design for the stack and test schedule for each stage of stacking such that the total test time T, i.e., the sum of the test times for all the desired stack tests (final stack test or multiple test insertions), is minimized and the number of TSVs used per die does not exceed $TSVmax_m$.

For a 3D-SIC with soft dies, the test-architecture for each die is not pre-defined, but is determined during the test-architecture design for the stack. This scenario provides greater flexibility in terms of test-time optimization. The problem of test-architecture optimization for 3D-SICs with soft dies and multiple test insertions is formally defined as follows.

3D-SIC with Soft Dies (P^S_{MTS})

Givens include a stack with a set M of dies and the total number of test pins W_{max} available for test. For each die $m \in M$, its tier number l_m in the stack, a maximum number of TSVs ($TSVmax_m$) that can be used for TAM design between die $m-1$ and m ($m > 1$), and the total number of cores c_m are given. Furthermore, for each core n, the number of inputs i_n, outputs o_n, total number of test patterns p_n, total number of scan chains s_n, and for each scan chain k, the length of the scan chain in flip flops $l_{n,k}$ are given. The goal is to determine an optimal TAM design and test schedule for each stage of stacking, as well as for each die, such that the total test time T for the stack is minimized and the number of TSVs used per die does not exceed $TSVmax_m$.

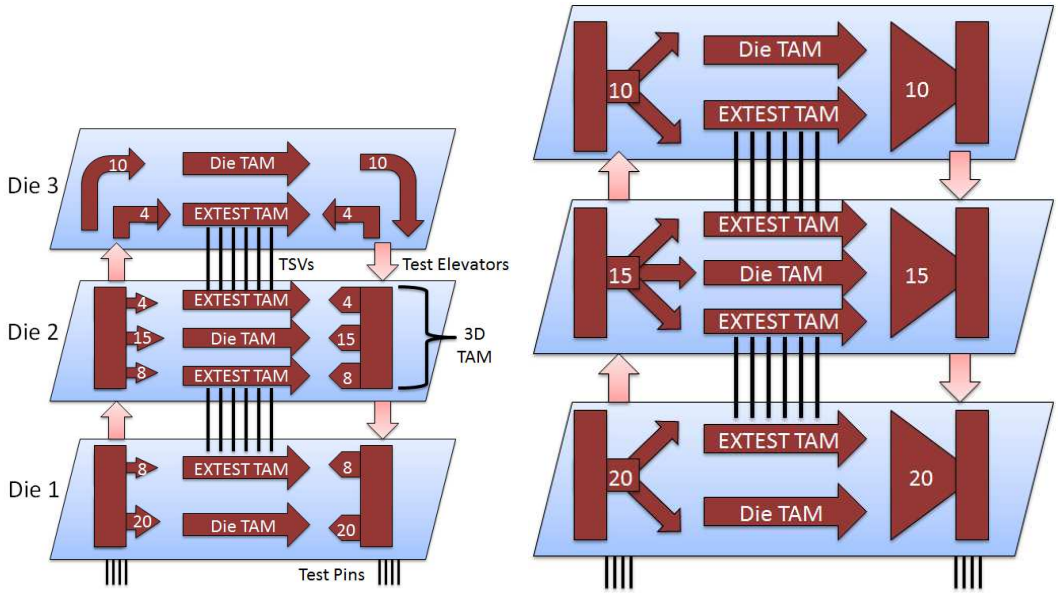


FIGURE 5.26: Example of a test architecture in a 3D-SIC including die-external tests.

Consider further the testing of TSVs and die-external logic along with the cores in each die. There are two variants to this problem; see Figure 5.26. Figure 5.26(a) shows a three-die stack with a test architecture using fat wrappers to allow TSV tests to take place in parallel with module testing on the dies. Each die has a separate TAM for die-external tests, both to higher and lower dies in the stack. Each TAM has its own width and utilizes different test pins. In this case, test pins used for TSV testing are not available for die testing. For example, internal test of Die 2 and external test of the TSVs and die-external logic between Die 2 and Die 3 can occur simultaneously, utilizing a total of 23 test pins. The second variant uses a thin wrapper which allows die-external tests to only take place serially with module testing on the die as seen in Figure 5.26(b). This design allows the TAMs for die-external test to access the full test width for the die, but test pins are shared between all TAMs on the die. In this architecture, the external tests for logic and TSVs between Die 2 and Die 3 can be carried out in parallel with the internal test of Die 1, but not in parallel with any other test.

Below, the problem definitions for these two variants is presented, where "||" refers to parallel and "−−" refers to serial. The problems of testing hard dies including TSV tests for a single test insertion (final stack test only) are stated below as Problem 3(a) and

Problem 3(b), respectively.

3D-SIC with Hard Dies and (a) fat wrapper ($P^H_{DTSV,||}$) or (b) thin wrapper ($P^H_{DTSV,--}$)

A stack with a set M of dies and the total number of test pins W_{max} available for test is given. For each die $m \in M$, its tier number l_m in the stack, the number of test pins w_m ($w_m \leq W_{max}$) required to test the die, the associated test time t_m, and a maximum number of test elevators ($TSVmax_m$) that can be used for TAM design between die $m-1$ and m are given. Furthermore, the number of functional TSVs (Tf_m) and test width given to the TSVs (Wt_{max}) between die $m-1$ and m, and the number of test patterns (Pf_m) for the functional TSVs between die $m-1$ and m (for $m > 1$) are given. The goal is to determine an optimal TAM design and a test schedule for the stack for die-internal and die-external tests such that the total test time T is minimized and the number of test TSVs used per die does not exceed $TSVmax_m$.

The problems of testing soft dies including TSV tests for a single test insertion are stated below as Problem 4(a) and Problem 4(b).

3D-SIC with Soft Dies and (a) fat wrapper ($P^S_{DTSV,||}$) or (b) thin wrapper ($P^S_{DTSV,--}$)

A stack with a set M of dies and the total number of test pins W_{max} available for test is given. For each die $m \in M$, its tier number l_m in the stack, a maximum number of test elevators ($TSVmax_m$) that can be used for TAM design between die $m-1$ and m, and the number of functional TSVs (Tf_m) between die $m-1$ and die m are given. Furthermore, the number of test patterns (Pf_m) for the functional TSVs between die $m-1$ and m (for $m > 1$), and the total number of cores c_m are given. For each core n, the number of inputs i_n, outputs o_n, total number of test patterns p_n, total number of scan chains s_n, and for each scan chain k, the length of the scan chain in flip flops $l_{n,k}$ are given. Determine an optimal TAM design for the stack and TSVs and a test schedule for the stack such that the total test time T is minimized and the number of test TSVs used per die does not exceed $TSVmax_m$.

All six problems presented above are NP-hard from "proof by restriction" [26], as they can be reduced using standard techniques to the rectangle packing problem, which is known to be NP-hard [21]. For example, for Problem P^S_{MTS}, if the constraints related to maximum number of TSVs are removed and only the final stack test insertion is considered, each die

can be represented as a set of rectangles with different widths and heights, where width is equal to its test time for a given TAM width and height equal to the number of required test pins. Now, all these rectangles (dies) must be packed into a bin with width equal to the total number of test pins and height equal to the total test time for the stack, which must be minimized. Despite the NP-hard nature of these problems, they can be solved optimally because the number of layers in a 3D-SIC is expected to be limited, e.g., up to four layers have been predicted for logic stacks [22].

5.4 Derivation of the Extended ILP Model

In this section, ILP is used to model and solve the problems defined in the previous section. The problem instances in practice are relatively small for realistic stacks with anywhere from two to eight dies. Therefore, ILP methods are good candidates for solving these optimization problems.

5.4.1 ILP Formulation for Problem P^H_{MTS}

To create an ILP model for this problem, the set of variables and constraints must be defined. To begin, define a binary variable x_{ijk}, which is equal to 1 if die i is tested in parallel with die j for a test insertion when there are k die in the stack, and 0 otherwise. There are $M-1$ test insertions, one for each additional die added to the stack, such that k ranges from 2 to M. Constraints on variable x_{ijk} can be defined as follows:

$$x_{iik} = 1 \qquad \forall k, i \leq k \tag{5.31}$$

$$x_{ijk} = x_{jik} \qquad \forall k, \{i, j\} \leq k \tag{5.32}$$

$$1 - x_{iqk} \geq x_{iqk} - x_{jqk} \geq x_{ijk} - 1 \qquad \forall k, \{i, j, q\} \leq k, i \neq j \neq q \tag{5.33}$$

The first constraint indicates that every die is always considered to be tested with itself for every test insertion. The second constraint states that if die i is tested in parallel with die j for insertion k, then die j is also tested in parallel with die i for insertion k. The last constraint ensures that if die i is tested in parallel with die j for insertion k, then it must also be tested in parallel with all other dies that are tested in parallel with die j for insertion k.

Next, define a second binary variable y_{ik}, which is equal to 0 if die i is tested in parallel with die j on a lower layer ($l_i > l_j$) for insertion k, and 1 otherwise. The total test time T for the stack is the sum of the test times of all dies that are tested in series plus the maximum of the test times for each of the sets of parallel tested dies for all test schedules at every test insertion. Using variables x_{ijk} and y_ik, the total test time T for all test insertions with the set of dies M can be defined as follows.

$$T = \sum_{k=2}^{|M|} \sum_{i=1}^{k} y_{ik} \cdot \max_{j=i..k} \{x_{ijk} \cdot t_j\} \tag{5.34}$$

It should be noted that Equation (5.34) has two non-linear elements, the *max* function, and the product of y_ik variable and the *max* function. This is linearized by introducing two new variables. The variable c_{ik} takes the value of the max function for each die i for test insertion k and the variable u_{ik} represents the product $y_{ik} \cdot c_{ik}$. The variables u_{ik} and c_{ik} are defined using standard linearization techniques. The linearized function for total test time can be written as follows.

$$T = \sum_{k=2}^{|M|} \sum_{i=1}^{k} u_{ik} \tag{5.35}$$

As the number of test pins used for parallel testing of dies should not exceed the given test pins W_{max} across all test schedules for every test insertion, a constraint on the total number of pins used to test all dies in a parallel set in any given test insertion can be defined as follows for all k.

$$\sum_{j=1}^{k} x_{ijk} \cdot w_j \leq W_{max} \qquad \forall i \leq k \tag{5.36}$$

Similarly, the total number of used TSVs should not exceed the given TSV limit ($TSVmax_m$) for each die face across all test insertions. It should be noted that $TSVmax_2$ is the limit for the upper face of die 1 and the lower face of die 2, $TSVmax_3$ is for the upper face of die 2 and lower face of die 3, and so forth. The number of TSVs used to connect layer i to layer $i-1$ is the maximum of the number of pins required by the layer at or above layer i that takes the most test pin connections, and the sum of parallel-tested dies at or above layer i

in the same parallel tested set across all test insertions. Based on this, the constraint on the total number of TSVs used in a test architecture can be defined as follows.

$$\max_{i<=k<=|M|} \{w_i, \sum_{j=i}^{k} w_j \cdot x_{kj}\} \leq TSVmax_i \qquad \forall i \geq 2, \tag{5.37}$$

The above set of constraints can be linearized by representing the *max* function with a variable d_i. Finally, to complete the ILP model for Problem P^H_{MTS}, constraints on the binary variable y_{ik} and the relationship between binary variables y_{ik} and x_{ijk} must be defined. For this purpose, a constant C is introduced that approaches but is less than 1. The variable y_{ik} can then be defined as follows:

$$y_{1k} = 1 \qquad \forall k \tag{5.38}$$

$$y_{ik} \geq \frac{1}{1-i} \sum_{j=1}^{i-1} \left(x_{ijk} - 1\right) - C \qquad \forall k, i \leq k, i \neq 1 \tag{5.39}$$

The first equation forces y_{1k} to 1, because the lowest layer cannot be tested in parallel with any layer lower than itself. Constraint 5.39 defines y_{ik} for the other layers. To understand this constraint, first make the observation that the objective function (as shown in Equation (5.34)) would be minimized if each y_{ik} is zero. This would make the objective function value equal to 0, which is an absolute minimum test time. Thus, y_{ik} must be restricted to 1 only where it is absolutely necessary, and then the objective function can be relied upon to assign a value 0 to all unrestricted y_{ik} variables. This equation considers the range of values that the sum of x_{ijk} can take. The fraction in the equation normalizes the sum to a value between 0 and 1 inclusive, while the summation considers all possible cases for a die being in parallel with dies below it. The complete ILP model is shown in Figure 5.27.

Figure 5.28 illustrates the ILP model for P^H_{MTS}. The 3D-SIC in Figure 5.28 consists of three dies, with test times of 1333098, 700665, and 106391 cycles moving up the stack. There are 22 test pins available, so W_{max} is 22. TSV_{max} is set to 20, such that there can be no more than 20 dedicated test TSVs between any two dies (this limits the TSVs per die to 40). There are two test insertions, the first when Die 1 and Die 2 are stacked and the second for the complete stack. In the first test insertion (k = 2), the optimal solution that Die 1 and

Objective:

Minimize $\sum_{k=2}^{|M|} \sum_{i=1}^{k} u_{ik}$

Subject to the Constraints:

$t_{max} = \max_{i=1}^{|M|} t_i$

$c_{ik} \geq x_{ijk} \cdot t_j \qquad \forall k, \{i,j\} \leq k$

$u_{ik} \geq 0 \qquad \forall k, i \leq k$

$u_{ik} - t_{max} \cdot y_{ik} \leq 0 \qquad \forall k, i \leq k$

$u_{ik} - c_{ik} \leq 0 \qquad \forall k, i \leq k$

$c_{ik} - u_{ik} + t_{max} \cdot y_{ik} \leq t_{max} \qquad \forall k, i \leq k$

$\sum_{j=1}^{k} x_{ijk} \cdot w_j \leq W_{max} \qquad \forall k, i \leq k$

$x_{iik} = 1 \qquad \forall k, i \leq k$

$x_{ijk} = x_{jik} \qquad \forall k, \{i,j\} \leq k$

$1 - x_{iqk} \geq x_{iqk} - x_{jqk} \geq x_{ijk} - 1 \quad \forall k, \{i,j,q\} \leq k, i \neq j \neq q$

$d_i \leq TSVmax_i \qquad \forall i \neq 1$

$d_i \geq \sum_{j=i}^{k} w_j \cdot x_{ijk} \qquad \forall i \neq 1, k \geq i$

$d_i \geq w_j \qquad \forall i, j = i..|M|$

$y_{1k} = 1 \qquad \forall k$

$y_{ik} \geq \frac{1}{1-i} \sum_{j=1}^{i-1} \left(x_{ijk} - 1\right) - C \qquad \forall k, i \leq k, i \neq 1$

FIGURE 5.27: ILP model for the 3D TAM optimization Problem P_{MTS}^{H}.

Die 2 are tested in parallel is calculated. As such, $x_{1,1,2}$, $x_{1,2,2}$, $x_{2,1,2}$, and $x_{2,2,2}$ are all equal to 1. Die 2 is tested in parallel with a die below it (Die 1), so $y_{2,2}$ is 0 and $y_{1,2}$ is 1. The test time for this test insertion is 1333098 cycles, because $u_{1,2}$ is 1333098 and $u_{2,2}$ is 0.

For the second test insertion (k = 3), the optimal solution is to test Die 1 and Die 2 in parallel, and then to test Die 3. Because Die 1 and Die 2 are tested in parallel again, $x_{1,1,3}$, $x_{1,2,3}$, $x_{2,1,3}$, and $x_{2,2,3}$ are equal to 1. For Die 3, $x_{3,3,3}$ is 1. All other x_{ijk} variables are 0. As before, $y_{2,3}$ is 0 since Die 2 is tested in parallel with Die 1. The variables $y_{1,3}$ and $y_{3,3}$ are 1. The test time for this test insertion is 1439489, as $u_{1,3}$ is 1333098, $u_{2,3}$ is 0, and $u_{3,3}$ is 106391. In this architecture, d_2 is 20 and d_3 is 14, neither of which violate the TSV limit. All 44 test pins are utilized.

The above ILP model is a generalization of the special case presented in Section 5.2, in which test-time was minimized for only the final stack test. If the variable k is constrained to take only one value, namely M, in P_{MTS}^{H}, then optimization will produce a test architecture and test schedule that minimizes test time only for the final stack, i.e., the objective in Section 5.2. Optimization for only the final stack test is referred to as P_{FS}^{H} and P_{FS}^{S} for hard

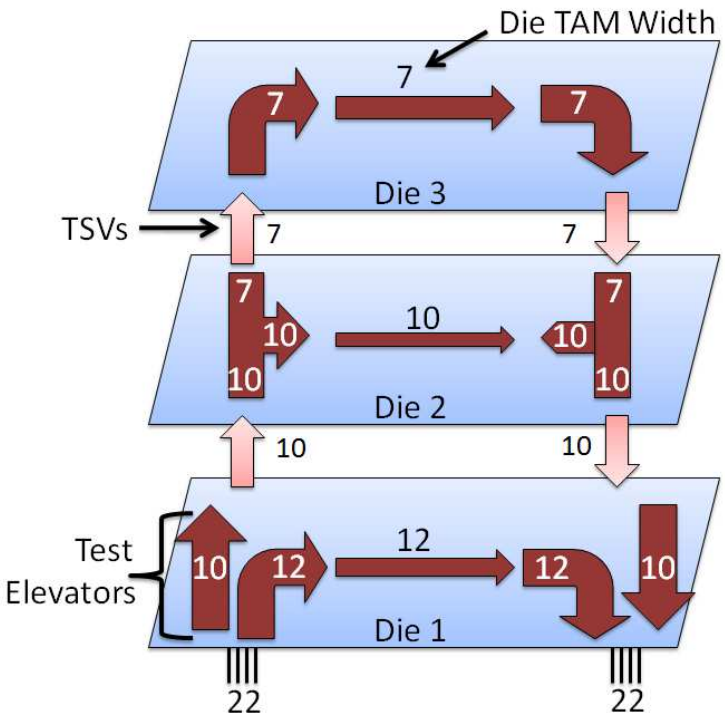


FIGURE 5.28: Example of a test architecture in an optimized 3D-SIC with hard dies.

and soft dies, respectively. Therefore, an advantage of the optimization model proposed here is that it is flexible—it can be easily tailored to minimize test time for any number of stack tests, from one final test to two or more intermediate test insertions. Multiple options can be automatically generated for testing the stack. For example, suppose there is interest in two test insertions—after the second die is bonded to the first die and the final stack test. By allowing k to now take two values, namely 2 and M, the test time for the stack can be minimized by considering only these two insertions.

5.4.2 **ILP Formulation for Problem P^S_{MTS}**

The ILP formulation for 3D-SICs with soft cores and multiple test insertions is derived in a similar manner as the 3D-SIC with hard cores above. In this case, the test time t_i for die i is a function of the TAM width w_i assigned to it. Using the variables x_{ijk} and y_ik as defined in Section 5.4.3, the total test time T for the stack with the set of soft dies M across all test

insertions can be defined as follows.

$$T = \sum_{k=2}^{|M|} \sum_{i=1}^{k} y_{ik} \cdot \max_{j=i..k} \{x_{ijk} \cdot t_j(w_j)\} \tag{5.40}$$

It should be noted that Equation (5.40) has several non-linear elements. To linearize this equation, first define the test time function. For this purpose, a binary variable g_{in} is introduced where $g_{in} = 1$ if $w_i = n$, and 0 otherwise. This expression is then linearized using the variable v_{ijk} for $x_{ijk} \cdot \sum_{n=1}^{k_i}(g_{jn} \cdot t_j(n))$. It should be noted that, although the variable x_{ijk} can change for different test insertions depending on the test schedule for each insertion, there must be a single value of test pins used for each die to reflect an architecture that can support all test schedules used throughout all test insertions. Similar to Equation (5.35), the variable c_{ik} takes the value of the max function for each die i for each test insertion k and the variable u_{ik} represents the product $y_{ik} \cdot c_{ik}$. Because w_j is now a decision variable, $x_{ijk} \cdot w_j$ is linearized using a new variable z_{ijkj}. The max function is represented by the variable d_i as before. By using the variable z_{ijkj}, the TAM width that can be given to each die can be constrained by an upper limit, which is the number of available test pins. This is represented by the following inequality.

$$\sum_{j=i}^{k} z_{ijkj} \leq W_{max} \qquad \forall k, i \leq k \tag{5.41}$$

The complete ILP model for Problem P^S_{MTS} is shown in Figure 5.29.

5.4.3 ILP Formulations for $P^H_{DTSV,||}$, $P^H_{DTSV,--}$, $P^S_{DTSV,||}$, and $P^S_{DTSV,--}$

The ILP formulations in Section 4.1 and Section 4.2 do not account for the TAM architecture and test time needed to test the TSVs. In order to include TSV tests, the TSV connections between two layers are treated as a "virtual" die. This die has a predefined test width for hard dies or a range of possible widths with associated test time for soft dies, just as an actual die for each model. In order to calculate the test time for the TSV "layer", the formula $\lceil T f_m / \lfloor (pins/4) \rfloor \rceil (P f_m + 1)$ is used, where *pins* refers to the number of test pins available for TSV testing.

Objective:

Minimize $\sum^{(}|M|)_{k=2}\sum_{i=1}^{k}u_i$

Subject to the Constraints:

$t_{max} = \max_{i=1}^{|M|} t_i$

$c_{ik} \geq v_{ijk} \qquad \forall k, i \leq k, i \leq j \leq k$

$v_{ijk} \geq 0 \qquad \forall k, \{i,j\} \leq k$

$v_{ijk} - t_{max} \cdot x_{ijk} \leq 0 \qquad \forall k, \{i,j\} \leq k$

$-\sum_{n=1}^{k_i}(g_{jn} \cdot t_j(n)) + v_{ijk} \leq 0 \qquad \forall k, \{i,j\} \leq k$

$\sum_{n=1}^{k_i}(g_{jn} \cdot t_j(n)) - v_{ijk} + t_{max} \cdot x_{ijk} \leq t_{max} \qquad \forall k, \{i,j\} \leq k$

$u_{ik} \geq 0 \qquad \forall k, i \leq k$

$u_{ik} - t_{max} \cdot y_{ik} \leq 0 \qquad \forall k, i \leq k$

$u_{ik} - c_{ik} \leq 0 \qquad \forall k, i \leq k$

$c_{ik} - u_{ik} + t_{max} \cdot y_{ik} \leq t_{max} \qquad \forall k, i \leq k$

$z_{ijkj} \geq 0 \forall k, \{i,j\} \leq k$

$z_{ijkj} - t_{max} \cdot x_{ijk} \leq 0 \forall k, \{i,j\} \leq k$

$-wj + z_{ijkj} \leq 0 \forall k, \{i,j\} \leq k \forall k, \{i,j\} \leq k$

$w_j - z_{ijkj} + t_{max} \cdot x_{ijk} \leq t_{max} \forall k, \{i,j\} \leq k$

$\sum_{i=2}^{|M|} d_i \leq TSVmax_i$

$d_q \geq \sum_{j=i}^{k} z_{ijkj} \qquad \forall 2 \leq q \leq |M|, k \geq q, q \leq i \leq k$

$d_i \geq w_j \qquad \forall i, j \geq i$

$\sum_{j=1}^{k} z_{ijkj} \leq W_{max} \qquad \forall k, i \leq k$

$x_{iik} = 1 \qquad \forall k, i \leq k$

$x_{ijk} = x_{jik} \qquad \forall k, \{i,j\} \leq k$

$1 - x_{iqk} \geq x_{iqk} - x_{jqk} \geq x_{ijk} - 1 \quad \forall k, \{i,j,q\} \leq k, i \neq j \neq q$

$y_{1k} = 1 \qquad \forall k$

$y_{ik} \geq \frac{1}{1-i}\sum_{j=1}^{i-1}\left(x_{ijk} - 1\right) - C \qquad \forall k, i \leq k, i \neq 1$

FIGURE 5.29: ILP model for 3D TAM optimization Problem P_{MTS}^{S}.

The above expression begins by taking the floor of $pins/4$. It is assumed that, between two dies, there is any number of functional TSVs that are used for either input to the dies or output from the dies. The TSV tips are latched and the corresponding flip-flops are connected to form one or more scan-chains. Thus, the scan-flops are also assumed to be unidirectional. In order to quickly test all TSVs, shift-in, shift-out, and capture on both sides of the TSVs must be allowed for simultaneously. This requires 4 test pins for each TSV scan-chain, i.e., an input and output pin for each scan-chain for both sides of the TSVs as seen in Figure 5.30(a). As can be seen from Figure 5.30(b), there is no reduction in test time for stack tests if unequal numbers of TSV scan-chains for dies on either side of the TSVs are considered, as the bottleneck in test time would then be the die with the fewest TSV scan-chains (TSV testing requires use of TSV scan-chains on both sides of the TSVs

in parallel). At least four more test pins must be added for a reduction in test time as shown in Figure 5.30(c). Thus, test pin use is evenly divided among the dies for TSV tests, and the number of scan-chains on either side of the TSVs is $\lceil T\, f_m / \lfloor (pins/4) \rfloor \rceil$. This is multiplied by the number of patterns required for TSV testing plus one to accommodate for shift-in and shift-out operations. Without loss of generality, the number of test pins required for control signals to a die-level wrapper for TSV testing is not considered.

Figure 5.31 shows the difference between fat and thin wrappers. In Figure 5.31(a), the die-internal and die-external TAMS utilize different test pins. Thus, EXTEST can be performed in parallel with either or both INTEST of Die 1 and Die 2. This is representative of fat wrappers. For thin wrappers, Figure 5.31(b) shows that the same test pins are utilized for die-internal and die-external tests, and as such test data must be multiplexed to the correct TAM. In this case, the INTEST of each die can be performed in parallel, but EXTEST cannot be performed in parallel with INTEST on Die 1 or Die 2 (although it can be be performed in parallel with INTEST of other dies in the stack).

The ILP formulation for 3D-SICs with hard cores including TSVs is derived in a similar manner as the 3D-SIC with hard cores in Problem 1. Begin by removing the subscript k from all variables that account for multiple test insertions—only consider the final stack test. As stated before, consider a set of TSVs between two layers to be a virtual die for purposes of optimization, such that Die 1 is the lowest die in the stack, Die 2 represents the TSVs between Die 1 and Die 3, Die 3 is the second die in the stack, Die 4 represents the TSVs between Die 3 and Die 5, and so on. In this way, odd-numbered dies are actual dise and even-numbered dies represent the TSVs between two odd-numbered dies.

Variables and constraints must be added in order to accurately model the dies representing TSVs. Begin by defining the variable $p_i = 1$ if a die i representing TSVs can be tested in parallel with the dies below and above it and 0 otherwise. In the case of fat wrappers, p_i is left as a decision variable and in the case of thin wrappers, p_i is forced to 0 for all dies representing TSVs. For actual (even-numbered) dies, p_i is always 1. If a die representing TSVs can be tested in parallel with the dies around it, then the number of test pins given to TSV testing reduces the number of pins available to the dies for testing. Otherwise, the TSVs can use all the test pins that are utilized by the dies around it. Define the set die to

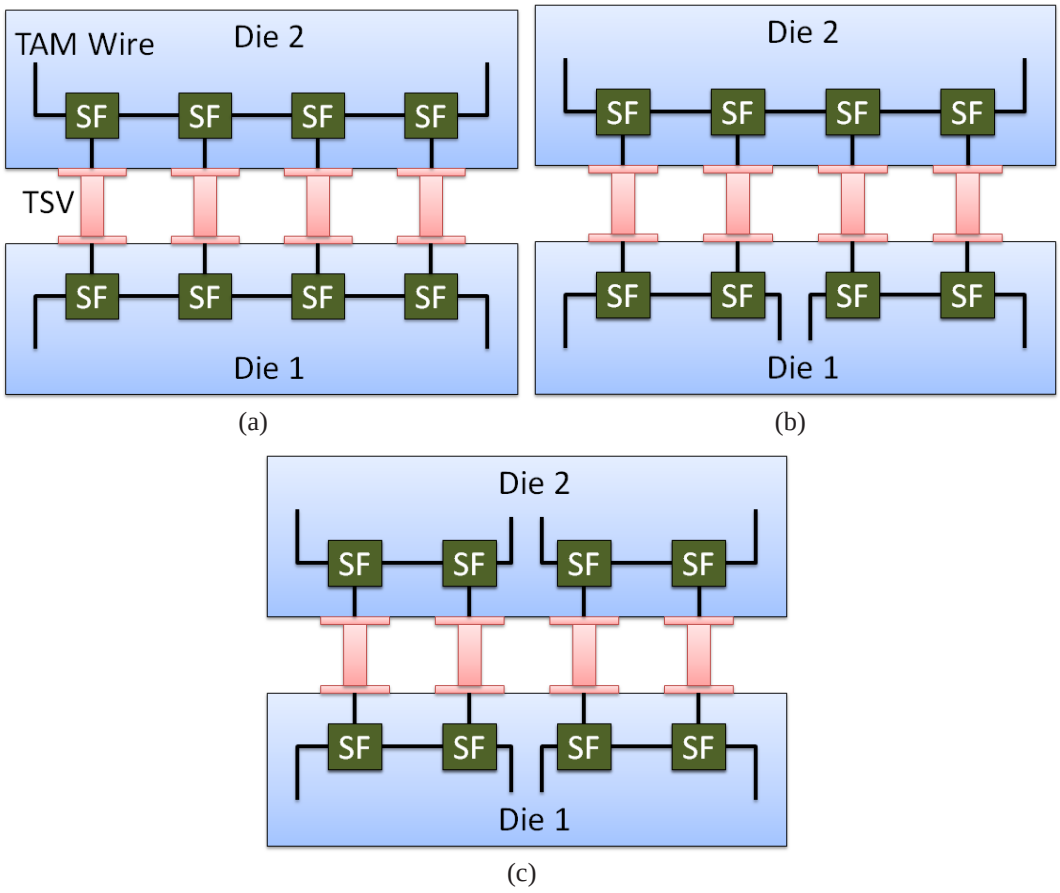


FIGURE 5.30: Example of scan chains for die-external tests.

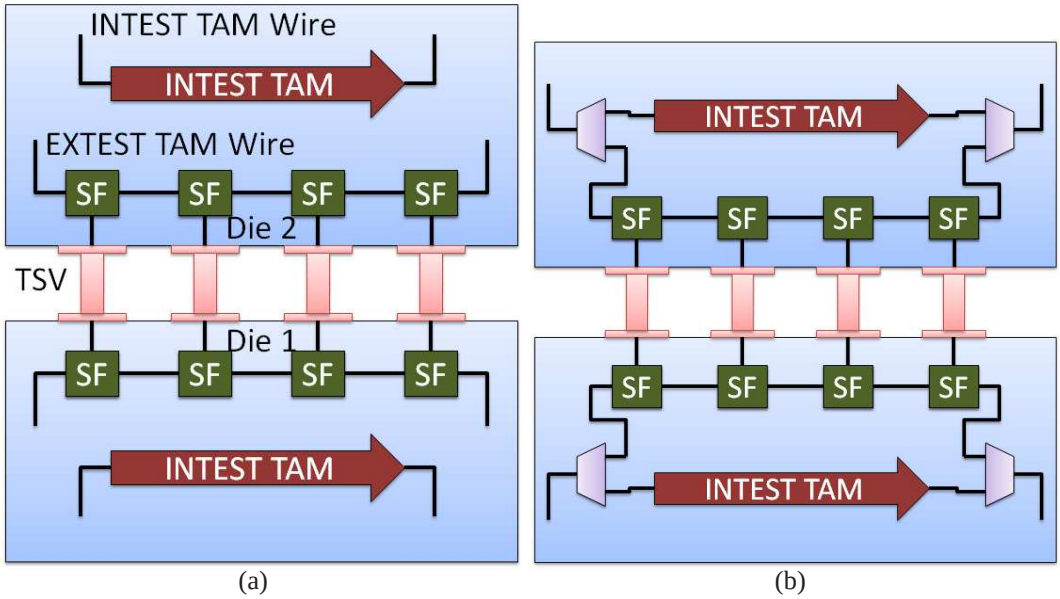


FIGURE 5.31: Simplified illustration of fat wrappers and thin wrappers.

contain all even-numbered dies, or the actual dies, in the stack. The following restrictions on p_i are added for fat wrappers (the first constraint applies for thin wrappers as well):

$$p_i = 1 \qquad \forall i \in die \tag{5.42}$$

$$p_{i+1} \leq w_i - w_{i+1} \qquad \forall i \in die, i \neq |M| \tag{5.43}$$

$$p_{i+1} \leq w_{i+2} - w_{i+1} \qquad \forall i \in die, i \neq |M| \tag{5.44}$$

This restricts p_i to 0 if all pins on a die are are shared for TSV tests, and otherwise leaves the variable open to optimization. The variable p_i is used to accurately represent the number of TSVs and test pins utilized in the stack and uses these constraints to further restrict the variable p_i. The variable d_i is redefined as follows:

$$d_i \geq \sum_{j=k}^{|M|} w_j \cdot x_{kj} \qquad \forall i \in die, i \neq |M|, k = i..|M| \tag{5.45}$$

$$d_i \geq w_j \qquad \forall i \in die, i \neq |M|, j = i..|M| \tag{5.46}$$

$$d_i \geq w_j + p_{j-1} \cdot w_{j-1} + p_{j+1} \cdot w_{j+1} \quad \forall i \in die, j \geq i \tag{5.47}$$

Thus, the test elevator use is tallied only between actual dies, while taking into account the number of extra test elevators used for the TSV layer considering parallel testing. For thin wrappers, the last constraint on d_i is removed. For Die 1, the last constraint is instead $d_i \geq w_j + p_{j+1} \cdot w_{j+1}$. Test pin constraints therefore reduce to:

$$\sum_{j=1}^{|M|} x_{ij} \cdot w_j \leq W_{max} \qquad \forall i \tag{5.48}$$

$$w_i + p_{i-1} \cdot w_{i-1} + p_{i+1} \cdot w_{i+1} \leq W_{max} \qquad \forall i \in die, i \neq |M| \tag{5.49}$$

The constraint 5.51 accounts for the combined test pin use by dies and TSVs when they are tested in parallel or in series. This constraint is removed for thin wrappers, and for the first die instead reads $w_i + p_{i+1} \cdot w_{i+1} \leq W_{max}$, $\forall i \in die, i \neq |M|$. It is necessary to further update the variable x_{ij} as follows:

$$x_{i,i+1} \leq p_{i+1} \qquad \forall i \in die, i \neq |M| \tag{5.50}$$

$$x_{i+2,i+1} \leq p_{i+1} \qquad \forall i \in die, i \neq |M| \tag{5.51}$$

Objective:

Minimize $\sum_{i=1}^{|M|} u_i$

Subject to the Constraints:

$t_{max} = \max_{i=1}^{|M|} t_i$

$c_i \geq x_{ij} \cdot t_j \qquad \forall i, j = i..|M|$

$u_i \geq 0$

$u_i - t_{max} \cdot y_i \leq 0$

$u_i - c_i \leq 0$

$c_i - u_i + t_{max} \cdot y_i \leq t_{max}$

$\sum_{j=1}^{|M|} x_{ij} \cdot w_j \leq W_{max} \qquad \forall i$

$w_i + p_{i-1} \cdot w_{i-1} + p_{i+1} \cdot w_{i+1} \leq W_{max} \qquad \forall i \in die, i \neq |M|$

$x_{ii} = 1 \qquad \forall i$

$x_{ij} = x_{ji} \qquad \forall i, j$

$1 - x_{ij} \geq x_{ik} - x_{jk} \geq x_{ij} - 1 \qquad \forall i \neq j \neq k$

$x_{i,i+1} \leq p_{i+1} \qquad \forall i \in die, i \neq |M|$

$x_{i+2,i+1} \leq p_{i+1} \qquad \forall i \in die, i \neq |M|$

$\sum_{i=2}^{|M|} d_i \leq TSVmax_i$

$d_i \geq \sum_{j=k}^{|M|} w_j \cdot x_{kj} \qquad \forall i \in die, i \neq |M|, k = i..|M|$

$d_i \geq w_j \qquad \forall i \in die, i \neq |M|, j = i..|M|$

$d_i \geq w_j + p_{j-1} \cdot w_{j-1} + p_{j+1} \cdot w_{j+1} \qquad \forall i \in die, j \geq i$

$y_1 = 1$

$y_i \geq \frac{1}{1-i} \sum_{j=1}^{i-1} (x_{ij} - 1) - C \qquad \forall i > 1$

$p_i = 1 \qquad \forall i \in die$

$p_{i+1} \leq w_i - w_{i+1} \qquad \forall i \in die, i \neq |M|$

$p_{i+1} \leq w_{i+2} - w_{i+1} \qquad \forall i \in die, i \neq |M|$

FIGURE 5.32: ILP model for 3D TAM optimization Problem $P^H_{DTSV,||}$.

The complete ILP model for the $P^H_{DTSV,||}$ problem is given in Figure 5.32.

For $P^H_{DTSV,--}$, a further constraint is added:

$$x_{i-1,i+1} = 0 \quad \forall i \in die, i \neq \{1, |M|\} \tag{5.52}$$

Problem $P^S_{DTSV,||}$ is formulated in a similar manner as P^S_{MTS}, with the subscript k removed and added constraints as defined in $P^H_{DTSV,||}$. In this way, the ILP model for $P^S_{DTSV,||}$ can be derived from the ILP model for P^S_{MTS} in the same way as $P^H_{DTSV,||}$ was derived from P^H_{MTS}. The model for $P^S_{DTSV,--}$ forces all p_i to 0 for dies representing TSVs. One new linearization constraint, pw_i is added to represent $p_i \cdot w_i$. The complete ILP model for the $P^S_{DTSV,||}$ problem is given in Figure 5.33.

Objective:

Minimize $\sum_{i=1}^{N} u_i$

Subject to the Constraints:

$t_{max} = \max_{i=1}^{|M|} t_i$

$c_i \geq v_{ij} \qquad \forall i, j = i..|M|$

$v_{ij} \geq 0 \qquad \forall i, j$

$v_{ij} - t_{max} \cdot x_{ij} \leq 0 \qquad \forall i, j = 1..|M|$

$-\sum_{n=1}^{k_i}(g_{jn} \cdot t_j(n)) + v_{ij} \leq 0 \qquad \forall i, j$

$\sum_{n=1}^{k_i}(g_{jn} \cdot t_j(n)) - v_{ij} + t_{max} \cdot x_{ij} \leq t_{max} \qquad \forall i, j$

$u_i \geq 0$

$u_i - t_{max} \cdot y_i \leq 0$

$u_i - c_i \leq 0$

$c_i - u_i + t_{max} \cdot y_i \leq t_{max}$

$z_{ijk} \geq 0$

$z_{ijk} - t_{max} \cdot x_{jk} \leq 0$

$-wi + z_{ijk} \leq 0$

$w_i - z_{ijk} + t_{max} \cdot x_{jk} \leq t_{max}$

$\sum_{i=2}^{|M|} d_i \leq TSV_{max}$

$\sum_{i=2}^{|M|} d_i \leq TSVmax_i$

$d_i \geq \sum_{j=k}^{|M|} z_{jkj} \qquad \forall i \in die, i \neq |M|, k = i..|M|$

$d_i \geq w_j \qquad \forall i \in die, i \neq |M|, j = i..|M|$

$d_i \geq w_j + pw_{j-1} + pw_{j+1} \qquad \forall i \in die, j \geq i$

$\sum_{j=1}^{|M|} z_{jij} \leq W_{max} \qquad \forall i$

$w_i + pw_{i-1} + pw_{i+1} \leq W_{max} \qquad \forall i \in die, i \neq |M|$

$x_{ii} = 1 \qquad \forall i$

$x_{ij} = x_{ji} \qquad \forall i, j$

$1 - x_{ij} \geq x_{ik} - x_{jk} \geq x_{ij} - 1 \qquad \forall i \neq j \neq k$

$x_{i,i+1} \leq p_{i+1} \qquad \forall i \in die, i \neq |M|$

$x_{i+2,i+1} \leq p_{i+1} \qquad \forall i \in die, i \neq |M|$

$y_1 = 1$

$y_i \geq \frac{1}{1-i} \sum_{j=1}^{i-1} (x_{ij} - 1) - M \qquad \forall i > 1$

$p_i = 1 \qquad \forall i \in die$

$p_{i+1} \leq w_i - w_{i+1} \qquad \forall i \in die, i \neq |M|$

$p_{i+1} \leq w_{i+2} - w_{i+1} \qquad \forall i \in die, i \neq |M|$

$pw_i \geq 0 \qquad \forall i$

$pw_i - W_{max} \cdot p_i \leq 0 \qquad \forall i$

$pw_i - w_i \leq 0 \qquad \forall i$

$w_i - pw_i + W_{max} \cdot p_i \leq W_{max} \qquad \forall i$

FIGURE 5.33: ILP model for 3D TAM optimization Problem $P^S_{DTSV,||}$.

5.5 Results and Discussion for the Multiple-Test Insertion ILP Model

In this section, simulation results are presented for the ILP models given in Section 5.4. SIC 1 and SIC 2 as shown in Figure 5.9 from Section 5.2 are utilized as benchmarks for

Table 5.6: Test lengths and number of test pins for hard dies used in optimization.

Die Name	**d695**	**f2126**	**p22810**	**p34392**	**p93791**
Test Length	106391	700665	1333098	2743317	2608870
Test Pins	15	20	25	25	30

this section. In SIC 1, the most complex die (p93791) is placed at the bottom, with die complexity decreasing as one moves up the stack. The order is reversed in SIC 2.

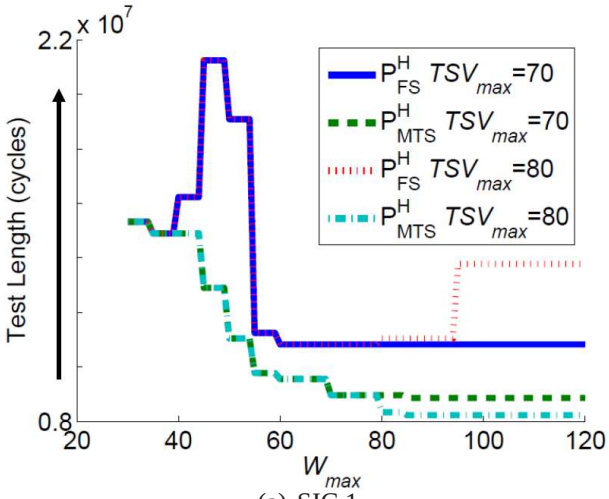
As before, to determine the test architecture and test time for a given die (SOC) with a given TAM width, the TAM design method in [23] for daisychain TestRail architectures [46] was utilized. For problem instances with hard dies, the test times (cycles) and TAM widths for different dies are listed in Table 5.6. Note that test pins were assigned to dies based on their sizes in order to avoid very large test times for any individual die.

For a fixed value of TSV_{max} and range of values of W_{max}, Table 5.7 presents results for P^H_{MTS} and P^S_{MTS} for the two benchmark SICs. They are compared against optimized results for only the final post-bond stack test (referred to as P^H_{FS} and P^S_{FS}). The ILP models were run and optimal results obtained using XPRESS-MP [49]. The CPU times for the simulations on an AMD Opteron 250 with four gigabytes of memory were in the range of a few seconds to eight minutes. In this table, Column 1 lists the optimization model that was run and Column 2 lists the benchmark SIC. Column 3 shows the maximum number of TSVs allowed for each die (TSV_{max}), while Column 4 lists the number of available test pins W_{max} at the lowest die. Column 5 represents the total test length (cycles) for all four post-bond test insertions in a five-die stack. Columns 6 gives the percent reduction in test time over the optimization case for only the final stack test. Let the test length for P^S_{FS} be T_1 and let the test length for P^S_{MTS} be T_2. Then, the percentage reduction is $((T_1 - T_2)/T_1) \cdot 100\%$. The calculation for hard dies is performed in the same manner. The percentage reductions for P^H_{FS} and P^S_{FS} are zero by default. Columns 7 through 10 show the resulting test schedule for the 3D-SIC at each test insertion, where the symbol "$\|$" indicates parallel testing of dies, and a "," represents serial testing. Finally, Column 11 gives the number of test pins needed for each die. From Table 5.7 it can be seen that the proposed method can provide

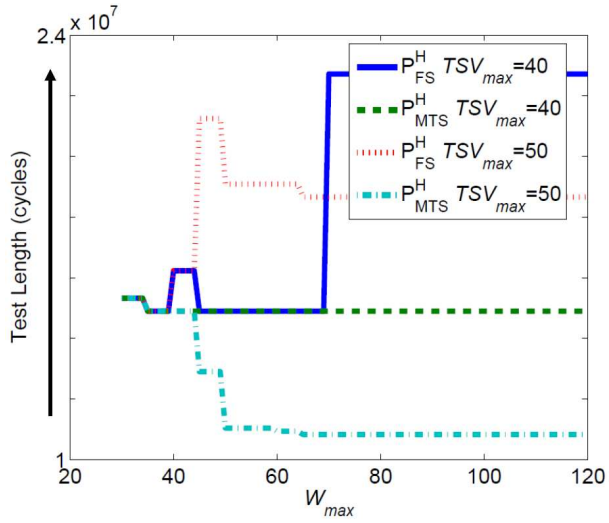
considerable reductions in test time over the optimization methods of Section 5.2 that are targeted toward final stack test.

For a different number of TSVs (TSV_{max}), Figure 5.34(a) and Figure 5.34(b) show the variation in test time T with an increase in the number of test pins W_{max} for SIC 1 and SIC 2. From the figures, it can be seen that both TSV_{max} and W_{max} determine which dies should be tested in parallel and thus the total test time for the stack. For a given value of TSV_{max}, increasing W_{max} does not always decrease the test time for P^H_{MTS}, although test time never increases. Similarly, increasing TSV_{max} for a given W_{max} does not always decrease the test time. These Pareto-optimal points can easily be seen in Figure 5.34 for both benchmarks. Furthermore, it can be seen that optimizing for the final stack test does not always reduce test time when multiple test insertions are considered. In fact, it is often the case that the test time is higher when increasing the values of TSV_{max} and W_{max}.

For optimization of 3D-SICs with soft dies, Pareto-optimality is almost non-existent when W_{max} is varied; see Figure 5.35. This is due to the fact that when dies in the stack are soft, it is almost always possible to find one die for which adding an extra test pin reduces the overall test time. When test time stops decreasing, the TSV limits for the problem instance have been reached and no further optimization is possible. Once again, as expected, the optimization method for multiple test insertions outperforms previous models.

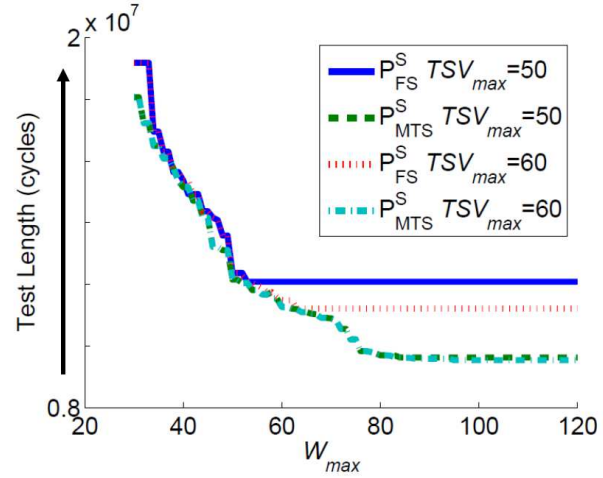


(a) SIC 1

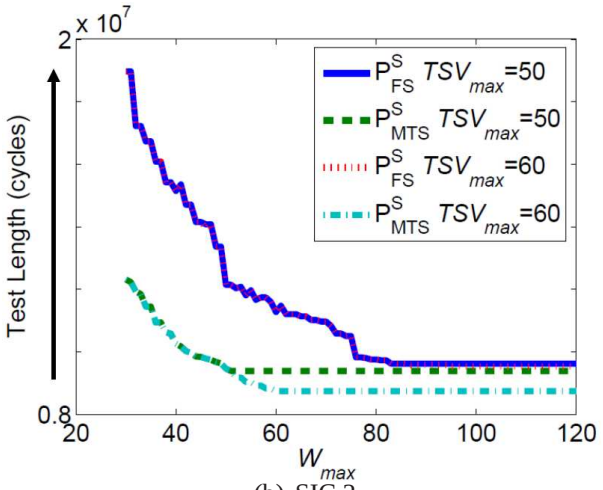


(b) SIC 2

FIGURE 5.34: The test time with respect to TSV_{max} and W_{max} for SIC 1 and SIC 2 with hard dies and multiple test insertions.

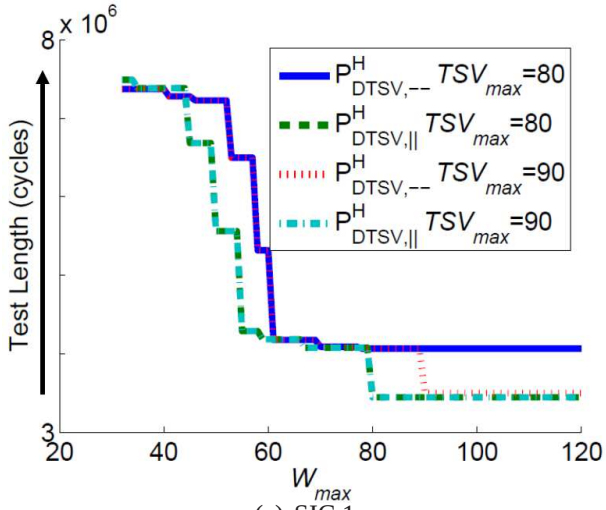


(a) SIC 1

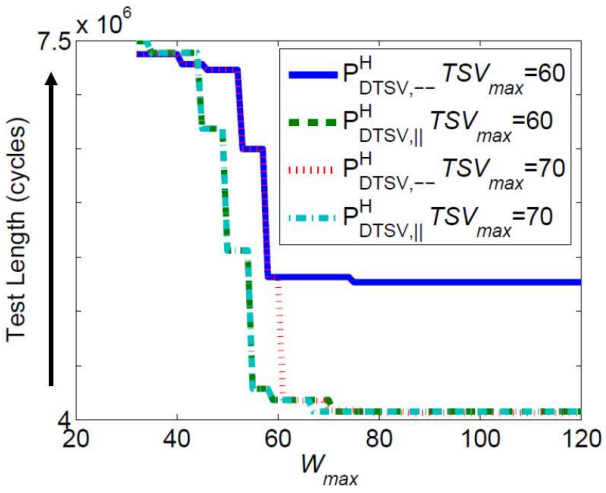


(b) SIC 2

FIGURE 5.35: The test time with respect to TSV_{max} and W_{max} for SIC 1 and SIC 2 with soft dies and multiple test insertions.



(a) SIC 1

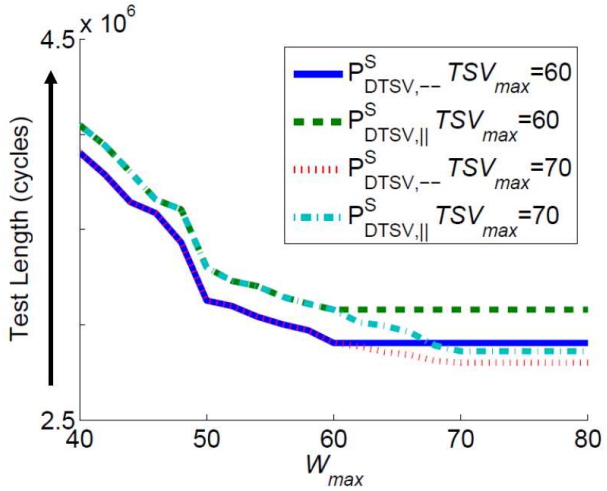


(b) SIC 2

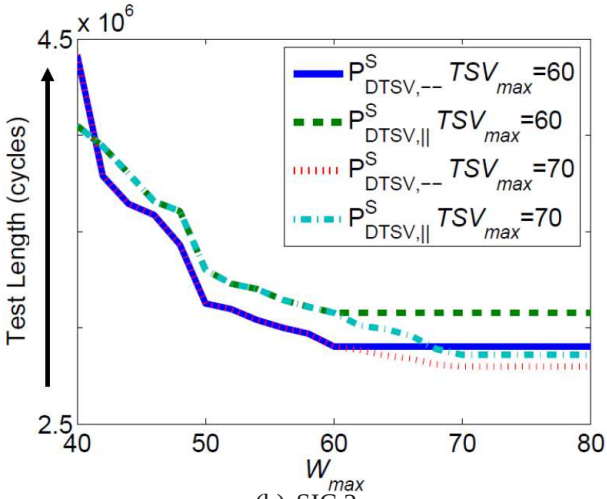
FIGURE 5.36: The test time with respect to TSV_{max} and W_{max} for SIC 1 and SIC 2 with hard dies including die-external tests.

Table 5.7: Simulation results and comparisons for multiple test insertions.

Optimization Framework	SIC	TSV_{max}	W_{max}	Test Length (cycles)	Reduction (%)	Test Schedule 1 (Die 1-2)	Test Schedule 2 (Die 1-2-3)	Test Schedule 3 (Die 1-2-3-4)	Test Schedule 4 (Die 1-2-3-4-5)	No. of Test Pins used per Die
P_{FS}^{H}	SIC 1	70	49	21254500	0.00	1,2	1,2‖3	1,2‖3,4	1‖5,2‖3,4	30,25,25,20,15
	SIC 1	70	50	19091000	0.00	1,2	1,2,3	1,2,3‖4	1,2‖5,3‖4	30,25,25,20,15
	SIC 1	70	69	10819000	0.00	1‖2	1‖2‖3	1‖2‖3,4	1‖2‖3,4‖5	30,25,25,20,15
	SIC 1	70	70	10819000	0.00	1‖2	1‖2‖3	1‖2‖3,4	1‖2‖3,4‖5	30,25,25,20,15
P_{MTS}^{H}	SIC 1	70	49	12901800	39.30	1‖2	1,2‖3	1‖4,2‖3	1‖4,2‖3,5	30,25,25,20,15
	SIC 1	70	50	11042700	42.16	1‖2	1,2‖3	1‖2,3‖4	1,2‖5,3‖4	30,25,25,20,15
	SIC 1	70	69	9554160	11.69	1‖2	1‖2‖3	1‖3‖4,2	1‖2‖3,4‖5	30,25,25,20,15
	SIC 1	70	70	8959880	17.18	1‖2	1‖2‖3	1,2‖3‖4	1‖4‖5,2‖3	30,25,25,20,15
P_{FS}^{S}	SIC 2	50	49	13366500	0.00	1‖2	1‖2‖3	1‖2‖3‖4	1‖2‖3‖4,5	22,16,6,4,28
	SIC 2	50	50	12149400	0.00	1‖2	1‖2‖3	1‖2‖3‖4	1‖2‖3‖4,5	24,16,6,4,30
	SIC 2	50	51	12149400	0.00	1‖2	1‖2‖3	1‖2‖3‖4	1‖2‖3‖4,5	24,17,6,4,30
P_{MTS}^{S}	SIC 2	50	49	9636860	27.90	1‖2	1‖2‖3	1‖2,3‖4	1‖2‖3,4‖5	3,18,28,20,29
	SIC 2	50	50	9467830	22.07	1‖2	1‖2‖3	1,2‖3‖4	1‖2,3‖4‖5	22,18,10,16,24
	SIC 2	50	51	9392830	22.69	1‖2	1‖2‖3	1‖2‖3‖4	1‖2,3‖4‖5	3,22,10,16,24



(a) SIC 1



(b) SIC 2

FIGURE 5.37: The test time with respect to TSV_{max} and W_{max} for SIC 1 and SIC 2 with soft dies including die-external tests.

It should be noted that for all optimizations with the same values of TSV_{max} and W_{max}, the stack configuration (SIC 2) with the largest die at the highest layer and the smallest die at the lowest layer is the best for reducing test time while using the minimum number of TSVs. This is because the most complex dies with the longest test times are tested in the fewest insertions. For example, the die at the top of the five-die stack is only tested once, while the die at the bottom is tested four times.

For $P^H_{DTSV,||}$, two extra test pins were added to the hard die for each EXTEST TAM on a die. For the highest and lowest dies in the stack, this implies an addition of two test pins, while for other dies it implies an addition of four test pins. This value was chosen because

it is the smallest addition of test pins necessary to result in reduced EXTEST times. It is possible to perform EXTEST with only two additional pins added to a die, but this results in unreasonably long EXTEST times. For the problem instances it is assumed, without loss of generality, that each die has 10,000 functional TSVs, requiring 20 patterns. It has been reported that the number of tests for TSVs is likely to grow logarithmically with TSV count [58]. In order to make comparisons between $P^H_{DTSV,--}$ and $P^H_{DTSV,||}$, either two or four test pins were added to the total TAM width of the hard dies. As can be seen in Figure 5.36, there is a large dependence of test length on both TSV_{max} and W_{max}. These parameters and the hard TAM widths determine whether serial testing or parallel testing of TSVs with the dies is adopted to lower test times. For a majority of values for TSV_{max} and W_{max} for the TAM widths chosen, parallel testing leads to lower test times. Note here that these optimizations apply only to the final stack test. As such, SIC 2 results in longer test times than SIC 1, as expected from Section 5.2.

In $P^S_{DTSV,--}$ and $P^S_{DTSV,||}$, there is more opportunity for test-time optimization and the results differ from those obtained with hard dies, as shown in Figure 5.37. Because full control is available over both die-internal and die-external TAM widths, trade-offs for parallel testing are magnified. Test pins that are not utilized for die-internal tests and instead used to support die-external tests tend to result in longer global test times. This is due to the complexity of the modules tested during die-internal tests when compared to TSV tests. Many more patterns are needed for die-internal tests, and as such, test pins devoted to these tests reduce test time more than test pins dedicated to die-external tests. Because it requires at least four test pins to result in test-time reduction for a single die-external test, this utilization of test pins does not provide much benefit. Therefore, thin wrappers and serial testing of TSVs result in lower test times than fat wrappers and parallel testing for 3D stacks with soft dies, as seen in Figure 5.37.

5.6 Conclusions

This chapter has dealt extensively with post-bond test architecture and test scheduling problems in the context of an ILP mathematical model and optimization solution. ILP solutions

were presented for three different, but realistic, 3D test-architecture optimization problems. These solutions produce optimal TAM designs and test schedules under test-access constraints for three design scenarios, referred to as hard, soft, and firm dies. The optimization models are rigorously derived and their optimality is formally established. It is shown that these methods result in significantly decreased test time over heuristic methods for 3D TAM design.

The effects of a number of parameters on the optimization results were examined; the results provide insights to designers on how to design for testability prior to 3D integration, how to design the 3D stack itself, and how to best allocate the limited test resources for the 3D SIC. The parameters of interest include the placement of dies in a stack, the availability of test TSVs and test pins. It has been demonstrated that, given test bandwidth and TSV constraints, pareto-optimality exists in the solution space. This finding is especially significant for the hard die case, although it is also present in the case of firm dies. These results imply that, using the derived models, designers can explore many test solutions for the hard and firm die scenarios in order to prevent sub-optimal allocation of test resources. Firm and soft dies are shown to provide lower test times with an increase in test resources. Because both scenarios result in lower test times than the case of hard dies, designers can consider 2D and 3D TAM design as related to optimization problems.

Generalized and rigorous optimization methods to minimize test time for multiple test insertions were further derived. These methods also provide optimal solutions for several additional problem instances of interest, e.g., final stack test only and testing of any subset of partial stacks. This extended optimization model is defined to be globally optimal in the case of multiple test insertions. Furthermore, optimization techniques were derived to consider die-external and die-internal tests during test-time minimization. Results show that optimization methods that only consider the final stack test provide significantly sub-optimal results (higher test times) if multiple test insertions are carried out. Optimizations for hard dies that take into account die-internal and die-external tests show that in general, fat wrappers reduce test time more than thin wrappers. This is not the case with soft dies, because the availability of fewer test pins for die-internal tests tends to lead to more test time.

6

Conclusions

This dissertation has covered an array of research relating to the design-for-test and test optimization for 3D SICs. These research topics aim to realize both pre-bond KGD test and low-cost post-bond test through test optimization. Probing techniques were explored for pre-bond TSV and scan test. Optimizations for reducing test cost were also covered, including flows to reduce the delay overhead of DfT architectures and to optimize TAM architectures and test schedules to reduce test time. Together, the topics covered by this dissertation aim to make the wide adoption of 3D SICs a reality.

Chapter 2 offered an alternative to BIST for pre-bond TSV test through probing. The present state-of-the-art in probe card design is provided, including a discussion of probing limitations. A combination of on-die and pin electronic architectures was explained to enable the probing of TSVs based on emerging standards and probe limitations. This was achieved by shorting together multiple TSVs with a single probe needle. Detailed results and analysis were provided to demonstrate the feasibility, accuracy, limitations, and cost of pre-bond TSV probing. Furthermore, an optimization technique, which reduced test time by allowing for the accurate test of multiple TSVs shorted through the same probe needle simultaneously, was discussed.

Chapter 3 extended the probing paradigm presented in Chapter 2 to enable both pre-bond TSV and scan test through the same probing paradigm. This was achieved by utilizing TSVs as test inputs and outputs along with reconfigurable scan chains. These scan chains were switched between pre-bond test modes utilizing TSVs and post-bond test modes that utilize the emerging die wrapper standard. An analysis of the feasibility, speed, and cost of the method was provided.

Chapter 4 explored a retiming-based test architecture optimization for reducing the impact of the test architectures described in Chapters 3 and 2 on the post-bond functional speed of the stack. Performance improvements were achieved by moving slack in the 3D circuit to offset the additional delay caused by test-related boundary registers at the interface between TSVs and die logic. A new retiming algorithm was developed to allow the limited movement of logic between dies and enable the decomposition of complex logic gates. The effectiveness, benefits, and drawbacks of retiming at both the stack- and die-level were detailed.

Finally, Chapter 5 developed an optimization method for the 3D TAM and test schedule to minimize the test time for post-bond stacks. A detailed and flexible ILP model was developed that enabled test optimization for any or all post-bond test insertions, for varying test pin and test TSV constraints, for external tests, and more. A detailed analysis of the test time reduction achievable through the optimization, including comparisons to other algorithms, was provided.

6.1 Future Research Directions

3D IC testing is a relatively new topic in the literature, and as such there are many challenges that require new or better solutions. While this dissertation mainly addressed known fault models and parametric test, 3D-specific faults could benefit from better modeling and targeting testing. One example of a 3D-specific concern is stress-related defects due to

TSV formation and bonding. The stress on the substrate caused by a TSV can influence the electrical characteristics of devices in close proximity, altering properties such as threshold voltage. To further complicate matters, the stress profile can change dramatically between the pre-bond die and the post-bond die. Fault models to enable structural pre-bond and post-bond TSV test considering such 3D-specific concerns, and the tests to capitalize on these fault models, should be developed.

TSV probing needs further study to ensure its viability for large-scale KGD testing. Though the probing technique described in this dissertation does allow for TSV and structural test, there remain limitations to the method. Capacitance and leakage parametric tests are averaged across the TSVs in a TSV network, and therefore the more TSVs in a network the less accurate the test will be. Furthermore, probing requires more time and investment in test equipment compared to BIST techniques. Though no BIST technique to date can perform such a wide range of accurate testing, and all seem to be limited by an inability to detect faults at the floating end of the TSV, better BIST techniques or a combination of BIST and TSV contact may result in cheaper and faster pre-bond test.

The work in this dissertation, in both design and optimization, does not address the issues of heating and heat dissipation during 3D test. Given that one die may be bonded between other dies with no clear route for heat dissipation, test optimization and testing methods should be developed to prevent overheating during test. For example, the test schedule could be designed in such a way that a high-temperature test is broken apart and run interspersed with low-temperature tests. Another option would be to optimize and compartmentalize test patterns and test structures such that test equipment can dynamically allocate tests depending on the reality of testing any given stack. Such ideas could warrant future exploration.

In summary, this dissertation has provided solutions for pre-bond and post-bond test of 3D SICs to enable accurate and cheap test. These solutions are aimed toward enabling

pre-bond TSV and scan test, to bring down post-bond test cost by minimizing test time, and to reduce the latency cost of 3D-specific test architectures. It is hoped that these advances can help to enable the wide-spread adoption of 3D SICs.

Bibliography

[1] M. L. Bushnell and V. D. Agrawal, “Essentials of Electronic Testing for Digital, Memory, and Mixed-Signal VLSI Circuits”, Boston; Kluwer Academic Publishers, 2000.

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

[3] P. Garrou, C. Bower, and P. Ramm. **Handbook of 3D Integration—Technology and Applications of 3D Integrated Circuits**. Wiley-VCH, Weinheim, Germany, August 2008.

[4] Yuan Xie, “Processor Architecture Design Using 3D Integration Technology”, *VLSI Design*, pp. 446-451, 2010.

[5] S. Wong, A. El-Gamal, P. Griffin, Y. Nishi, F. Pease, and J. Plummer, “Monolithic 3D Integrated Circuits”, *VLSI Technology, Systems and Applications*, pp. 1-4, 23-25, 2007.

[6] J. Feng, Y. Liu, P. Griffin, and J. Plummer, “Integration of Germanium-on-Insulator and Silicon MOSFETs on a Silicon Substrate”, *Electronic Device Letters*, pp.911-913, 2006.

[7] F. Crnogorac, D. Witte, Q. Xia, B. Rajendran, D. Pickard, Z. Liu, A. Mehta, S. Sharma, A. Yasseri, T. Kamins, S. Chou, and R. Pease, “Nano-Graphoepitaxy of Semiconductors for 3D Integration”, *Microelectronic Engineering*, pp. 891-894, 2007.

[8] Tezzaron Octopus.
http://www.tezzaron.com/memory/Octopus.html
Accessed March 2011.

[9] “Samsung Develops 8GB DDR3 DRAM Using 3D Chip Stacking Technology.”
http://www.samsunghub.com/2010/12/07/samsung-develops-8gb-ddr3-dram-using-

3d-chip-stacking-technology/
Accessed March 2011.

[10] U. Kang et al. “8Gb 3D DDR3 DRAM Using Through-Silicon-Via Technology”, In *Proc. International Solid State Circuits Conference (ISSCC)*, pp. 130-132, February 2009.

[11] C.C. Liu, I. Ganusov, M. Burtscher, and S. Tiwari, “Bridging the Processor-Memory Performance Gap with 3D IC Technology,” *IEEE Design & Test of Computers*, vol. 22, pp. 556-564, 2005.

[12] B. Black, D.W. Nelson, C. Webb, and N. Samra, “3D Processing Technology and its Impact on iA32 Microprocessors”, *Proc. International Conference on Computer Design (ICCD)*, pp. 316-318, 2004.

[13] S. Das, A. Chandrakasan, and R. Reif, “Design Tools for 3-D Integrated Circuits,” In *Proc. IEEE Asia South Pacific Design Automation Conference (ASP-DAC)*, pp. 53-56, 2003.

[14] B. Noia, K. Chakrabarty, and Y. Xie, “Test-Wrapper Optimization for Embedded Cores in TSV-Based Three-Dimensional SOCs”, *Proc. IEEE International Conference on Computer Design*, pp. 70-77, 2009.

[15] B. Noia, K. Chakrabarty, and E. J. Marinissen, “Optimization Methods for Post-Bond Die-Internal/External Testing in 3D Stacked ICs”, accepted for publication in *Proc. IEEE International Test Conference*, 2010.

[16] X. Zhao, D. Lewis, H.-H. Lee, and S. K. Lim, “Pre-Bond Testable Low-Power Clock Tree Design for 3D Stacked ICs”, *Proc. International Conference on Computer-Aided Design*, pp. 184-190, 2009.

[17] L. Jiang, Y. Liu, L. Duan, Y. Xie, and Q. Xu, “Modeling TSV Open Defects in 3 -D Stacked DRAM”, Accepted for publication in *Proc. International Test Conference*, 2010.

[18] Physorg.com
http://pda.physorg.com/_news170952515.html

[19] K. Chakrabarty, “Optimal Test Access Architectures for System-on-a-Chip”, *ACM Transactions on Design Automation of Electronic Systems*, vol. 6, pp. 26-49, January 2001.

[20] M.L. Flottes, J. Pouget, and B. Rouzeyre, "Sessionless Test Scheme: Power-Constrained Test Scheduling for System-on-a-Chip", *Proc. of the 11th IFIP on VLSI-SoC*, pp. 105-110, 2001.

[21] E.G. Coffman, Jr., M.R. Garey, D.S. Johnson and R.E. Tarjan. "Performance Bounds for Level-Oriented Two-Dimensional Packing Algorithms", *SIAM J. Computing*, vol. 9, pp. 809-826, 1980.

[22] X. Dong and Y. Xie. "System-level Cost Analysis and Design Exploration for 3D ICs", *Proc. of Asia-South Pacific Design Automation Conference (ASP-DAC)*, pp. 234-241, Jan. 2009.

[23] S.K. Goel and E.J. Marinissen, "Control-Aware Test Architecture Design for Modular SOC Testing", *European Test Workshop*, pp. 57-62, 2003.

[24] E.J. Marinissen, V. Iyengar, and K. Chakrabarty. "A Set of Benchmarks for Modular Testing of SOCs", In *International Test Conference*, pp. 519-528, Oct. 2002.

[25] *IEEE Std. 1500: IEEE Standard Testability Method for Embedded Core-Based Integrated Circuits*. *IEEE Press*, New York, 2005.

[26] M.R. Garey and D.S. Johnson, "Computers and Intractability—A Guide to the Theory of NP-Completeness", New York: Freeman, 1979.

[27] V. Iyengar, K. Chakrabarty, and E. J. Marinissen, "Test Wrapper and Test Access Mechanism Co-optimization for System-on-Chip", *JETTA*, vol. 18, pp. 213-230, 2002.

[28] S. K. Goel, E. J. Marinissen, A. Sehgal and K. Chakrabarty, "Testing of SOCs with Hierarchical Cores: Common Fallacies, Test-Access Optimization, and Test Scheduling", *IEEE Transactions on Computers*, vol. 58, pp. 409-423, March 2009.

[29] Q. Xu and N. Nicolici, "Resource-Constrained System-on-a-Chip Test: A Survey," *IEE Proc. Comp. Dig. Tech.*, vol. 152, no. 1, pp. 67-81, 2005.

[30] E. Larsson, K. Arvidsson, H. Fujiwara, and Z. Peng, "Efficient Test Solutions for Core-based Designs," *TCAD*, vol. 23, no. 5, pp. 758-775, 2004.

[31] K. Banerjee et al., "3-D ICs: a Novel Chip Design for Improving Deep-Submicrometer Interconnect Performance and Systems-on-Chip Integration," *Proc. IEEE*, vol. 89, no. 5, pp. 602-633, 2001.

[32] R. Weerasekera et al., “Extending Systems-on-Chip to the Third Dimension: Performance, Cost and Technological Tradeoffs,” in *ICCAD*, 2007.

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

[34] C.-Y. Lo, Y.-T. Hsing, L.-M. Denq, and C.-W. Wu, “SOC Test Architecture and Method for 3-D ICs”, *IEEE Transactions on Computer-Aided Design*, pp. 1645-1649, 2010.

[35] P.-Y. Chen, W. C.-W. Wu, and D.-M. Kwai, “On-Chip Testing of Blind and Open-Sleeve TSVs for 3D IC Before Bonding”, *Proc. IEEE VLSI Test Symposium*, pp. 263-268, 2010.

[36] S. Panth and S. K. Lim, “Scan Chain and Power Delivery Network Synthesis for Pre-Bond Test of 3D ICs”, Accepted for publication in *Proc. IEEE VLSI Test Symposium*, 2011.

[37] D. Lewis and H.-H. Lee, “A Scan-Island Based Design Enabling Pre-bond Testability in Die-Stacked Microprocessors”, *Proc. International Test Conference*, pp.1-8, 2007.

[38] Y. Xie, G. H. Loh, and K. Bernstein, “Design Space Exploration for 3D Architectures,” J. *Emerg. Technol. Comput. Syst.*, 2(2):65-103, 2006.

[39] G. Loh, Y. Xie, and B. Black, “Processor Design in 3D Die Stacking Technologies,” *IEEE Micro* Vol 27, No. 3, pp. 31-48, 2007.

[40] X. Wu, P. Falkenstern, K. Chakrabarty, and Y. Xie, “Scan-Chain Design and Optimization for 3D ICs,”’ *ACM Journal on Emerging Technologies in Computing Systems*, vol. 5, Article 9, July 2009.

[41] X. Wu, Y. Chen K. Chakrabarty, and Y. Xie, “Test-Access Mechanism Optimization for Core-Based Three-Dimensional SOCs,” *ICCD* 2008.

[42] E. J. Marinissen, S. K. Goel, and M. Lousberg, “Wrapper Design for Embedded Core Test,” *Proc. Int’l Test Conf.*, pp. 911-920, 2000.

[43] Williams, H. P. 1985. *Model Building in Mathematical Programming*. 2nd ed. John Wiley, New York, NY.

[44] D. Bertsimas and J. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1997.

[45] Y. Huang et al., "Optimal Core Wrapper Width Selection and SOC Test Scheduling Based on 3-D Bin Packing Algorithm,"' In *International Test Conference*, pp. 74-82, 2002.

[46] S. K. Goel and E. J. Marinissen. "SOC Test Architecture Design for Efficient Utilization of Test Bandwidth,"' *ACM Transactions on Design Automation of Electronic Systems*, 8(4):399-429, 2003.

[47] Q. Xu and N. Nicolici, "Resource-Constrained System-on-a-Chip Test: A Survey", *IEE Proc.: Computers and Digital Techniques*. vol. 152, pp. 67-81, Jan. 2005.

[48] L. Jiang, L. Huang, and Q. Xu, "Test Architecture Design and Optimization for Three-Dimensional SoCs,"' *DATE*, pp. 220-225, 2009.

[49] FICO. Xpress-MP.
http://www.fico.com/en/Products/DMTools/Pages/FICO-Xpress-Optimization-Suite.aspx

[50] E.J. Marinissen, J. Verbree, and M. Konijnenburg, "A Structured and Scalable Test Access Architecture for TSV-Based 3D Stacked ICs", *VLSI Test Symposium*, 2010.

[51] B. Noia, S.K. Goel, K. Chakrabarty, E.J. Marinissen, and J. Verbree, "Test-Architecture Optimization for TSV-Based 3D Stacked ICs", *European Test Symposium*, 2010.

[52] B. Noia, K. Chakrabarty and E. J. Marinissen, "Optimization Methods for Post-bond Die-Internal/External Testing in 3D Stacked ICs", *Proc. IEEE International Test Conference*, 2010.

[53] H. Chen, J.-Y. Shih, S.-W. Li, H.-C. Lin, M.-J. Wang, C.-N. Peng. "Electrical Tests for Three-Dimensional ICs (3DICs) with TSVs.", *International Test Conference 3D-Test Workshop*, 2010.

[54] L. Jiang, 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", *International Conference on Computer-Aided Design*, pp. 191-196, 2009.

[55] U. Kang et. al., "8 Gb 3-D DDR3 DRAM Using Through-Silicon-Via Technology", *IEEE Solid-State Circuits*, vol. 45, no. 1, pp. 111-119, 2010.

[56] M. Cho, C. Liu, D. Kim, S. Lim, and S. Mukhopadhyay, "Design Method and Test Structure to Characterize and Repair TSV Defect-Induced Signal Degradation in 3D System", *Proc. IEEE Conference on Computer-Aided Design*, pp. 694-697, 2010.

[57] M. Nicolaidis, V. Pasca, and L. Anghel, “Interconnect Built-In Self-Repair and Adaptive-Serialization (I-BIRAS) for 3D Integrated Systems”, *IEEE On-Line Testing Symposium*, p. 218, 2010.

[58] E.J. Marinissen and Y. Zorian, “Testing 3D Chips Containing Through-Silicon Vias”, *International Test Conference*, E 1.1, 2009.

[59] P.-Y. Chen, C.-W. Wu, and D.-M. Kwai, “On-Chip TSV Testing for 3D IC Before Bonding Using Sense Amplification,” *Proc. Asian Test Symposium*, pp. 450-455, 2009.

[60] H.-H. S. Lee and K. Chakrabarty, “Test Challenges for 3D Integrated Circuits”, *IEEE Design & Test of Computers*, vol. 26, pp. 26-35, September/October 2009.

[61] K. Puttaswamy and G. H. Loh, “The Impact of 3-Dimensional Integration on the Design of Arithmetic Units,” in *IEEE International Symposium on Circuits and Systems*, 2006.

[62] K. Puttaswamy and G. H. Loh, “Thermal Herding: Microarchitecture Techniques for Controlling Hotspots in High-Performance 3D-Integrated Processors,” *IEEE High Performance Computer Architecture*, pp. 193-204, 2007.

[63] K. Puttaswamy and G. H. Loh, “Scalability of 3D-Integrated Arithmetic Units in High-Performance Microprocessors,” *Design Automation Conference*, pp. 622-625, 2007.

[64] C.C. Liu, I. Ganusov, M. Burtscher, and S. Tiwari, “Bridging the Processor-Memory Performance Gap with 3D IC Technology,” *IEEE Design & Test of Computers*, 22(6):556-564, November/December 2005.

[65] B. Black, D.W. Nelson, C. Webb, and N. Samra, “3D Processing Technology and its Impact on iA32 Microprocessors”, *In Proc. International Conference on Computer Design (ICCD)*, pp. 316-318, October 2004.

[66] X. Wu, Y. Chen K. Chakrabarty, and Y. Xie, “Test-Access Mechanism Optimization for Core-Based Three-dimensional SOCs”, *IEEE International Conference on Computer Design*, pp.212-218, 2008.

[67] L. Jiang, L. Huang, and Q. Xu, “Test Architecture Design and Optimization for Three-Dimensional SoCs”, *Design, Automation, and Test in Europe*, pp. 220-225, 2009.

[68] Y. Lou, Z. Yan, F. Zhang, and P. Franzon, "Comparing Through-Silicon-Via (TSV) Void/Pinhole Defect Self-Test Methods", *International Test Conference 3D-Test Workshop*, 2010.

[69] K. Lee, "Trends in Test", Keynote talk presented at *IEEE Asian Test Symposium*, December 2010.

[70] P. Holmberg, "Automatic Balancing of Linear AC Bridge Circuits for Capacitive Sensor Elements", *Instrumentation and Measurement Technology Conference*, pp. 475-478, 2010.

[71] Qmax. QT2256 - 320 PXI.
http://www.qmaxtest.com/in/Automated%20Test%20Equipment/qt2256pxifea.html
Accessed January 2011.

[72] T. Olsen, AMD, Private Correspondence, December 2010.

[73] T. Yasafuku, K. Ishida, S. Miyamoto, H. Nakai, M. Takamiya, T. Sakurai, and K. Takeuchi, "Effect of Resistance of TSV's on Performance of Boost Converter for Low-Power 3D SSD with NAND Flash Memories", *3D System Integration*, pp. 1-4, 2009.

[74] J. Broz and R. Rincon, "Probe Needle Wear and Contact Resistance", talk presented at *Southwestern Test Workshop*, June 1998.

[75] K. Kataoka, S. Kawamura, T. Itoh, T. Suga, K. Ishikawa, and H. Honma, "Low Contact-Force and Compliant MEMS Probe Card Utilizing Fritting Contact", *IEEE Conference on Micro Electro Mechanical Systems*, pp. 364-367, 2002.

[76] 45nm PTM LP Model.
http://ptm.asu.edu/modelcard/LP/45nm_LP.pm
Accessed January 2011.

[77] J. Leung, M. Zargari, B.A. Wooley, and S.S. Wong, "Active Substrate Membrane Probe Card", *Electron Devices Meeting*, pp.709-712, 1995.

[78] D. Williams and T. Miers, "A Coplanar Probe to Microstrip Transition", *IEEE Transactions on Microwave Theory and Techniques*, pp. 1219-1223, 1988.

[79] O. Weeden, "Probe Card Tutorial",
http://www.accuprobe.com/Downloads/Probe%20Card%20Tutorial.pdf
Accessed January 2010.

[80] K. Puttaswamy and G. H. Loh, "Implementing Caches in a 3D Technology for High Performance Processors", *Proc. International Conference on Computer Design*, 2005.

[81] J. W. Joyner and J. D. Meindl, "Opportunities for Reduced Power Dissipation Using Three-Dimensional Integration", *Proc. International Interconnect Technology Conference*, pp. 148-150, 2002.

[82] C. Zhu, Z. Gu, L. Shang, R. P. Dick, and R. Joseph, "Three-Dimensional Chip-Multiprocessor Run-Time Thermal Management", *IEEE Transactions on CAD*, pp. 1479-1492, 2008.

[83] A. Coskun, J. Ayala, D. Atienza, T. Rosing, and Y. Leblebici, "Dynamic Thermal Management in 3D Multicore Architectures", *Proc. Conference on Design, Automation and Test in Europe*, pp. 1410-1415, 2009.

[84] S. Das, A. Chandrakasan, and R. Reif, "Timing, Energy, and Thermal Performance of Three-Dimensional Integrated Circuits", *Proc. 14th ACM Great Lakes Symposium on VLSI*, pp. 338-343, 2004.

[85] K. Puttaswamy and G. H. Loh, "Thermal Analysis of a 3D Die-Stacked High-Performance Microprocessor", *Proc. 16th ACM Great Lakes Symposium on VLSI*, pp. 19-24, 2006.

[86] T.-Y. Chiang, S. J. Souri, C. Chui, and K. C. Saraswat, "Thermal Analysis of Heterogeneous 3D ICs with Various Integration Scenarios", *Electron Devices Meeting*, pp. 31.2.1-31.2.4, 2001.

[87] W.-L. Hung, G. M. Link, Y. Xie, N. Vijaykrishnan, and M. J. Irwin, "Interconnect and Thermal-Aware Floorplanning for 3D Microprocessors", *Quality Electronic Design*, pp. 98-104, 2006.

[88] K. Balakrishnan, V. Nanda, S. Easwar, and S. K. Lim, "Wire Congestion and Thermal-Aware 3D Global Placement", *Proc. Asia and South Pacific Design Automation Conference*, pp. 1131-1134, 2005.

[89] J. Cong, G. Luo, J. Wei, and Y. Zhang, "Thermal-Aware 3D IC Placement Via Transformation", *Proc. Asia and South Pacific Design Automation Conference*, pp. 780-785, 2007.

[90] B. Goplen and S. Sapatnekar, "Efficient Thermal Placement of Standard Cells in 3D ICs using a Force-Directed Approach", *IEEE Conf. on Computer-Aided Design*, p. 86, 2003.

[91] Tessent Fastscan.
http://www.mentor.com/products/silicon-yield/products/fastscan/
Accessed March 2011.

[92] Hotspot 5.0.
http://lava.cs.virginia.edu/HotSpot/
Accessed March 2010.

[93] L.-R. Huang, S.-Y. Huang, S. Sunter, K.-H. Tsai, and W.-T. Cheng, "Oscillation-Based Prebond TSV Test", *IEEE Transactions on Computer-Aided Design*, vol.32, no.9, pp. 1440-1444, 2013.

[94] S. Deutsch and K. Chakrabarty, "Non-Invasive Pre-Bond TSV Test Using Ring Oscillators and Multiple Voltage Levels", *Proc. Design, Automation, and Test Conference in Europe*, pp. 18-22, 2013.

[95] T. Thorolfsson et al., "Design Automation for a 3DIC FFT Processor for Synthetic Aperture Radar: A Case Study", *Proc. IEEE Design Automation Conference*, 2009.

[96] L. Jiang, Y. Liu, L. Duan, Y. Xie and Q. Xu, "Modeling TSV Open Defects in 3D-Stacked DRAM", *Proc. IEEE International Test Conference*, pp. 1-9, 2010.

[97] K. Smith et al., "Evaluation of TSV and Micro-Bump Probing for Wide I/O Testing", *Proc. IEEE International Test Conference*, pp. 1-10, 2011.

[98] O. Yaglioglu, and N. Eldridge, "Direct Connection and Testing of TSV and Microbump Devices Using NanoPierce^{TM} Contactor for 3D-IC Integration" *Proc. IEEE VLSI Test Symposium*, pp. 96-101, 2012.

[99] B. Leslie and F. Matta, "Wafer-Level Testing with a Membrane Probe", *IEEE Design and Test of Computers*, pp. 10-17, 1989.

[100] Y. Zhang, Y. Zhang, and R. B. Marcus, "Thermally Actuated Microprobes for a New Wafer Probe Card", *Microelectromechanical Systems*, vol. 8, pp. 43-49, 1999.

[101] Y.-W. Yi, Y. Kondoh, K. Ihara, and M. Saitoh, "A Micro Active Probe Device Compatible with SOI-CMOS Technologies", *Microelectromechanical Systems*, vol. 6, pp. 242-248, 1997.

[102] W. Zhang, P. Limaye, R. Agarwal, and P. Soussan, "Surface Planarization of Cu/Sn Micro-bump and its Application in Fine Pitch Cu/Sn Solid State Diffusion Bonding", *Proc. Conf. on Electronics Packaging Technology*, pp. 143-146, 2010.

[103] Y. Ohara et al., "10 μm Fine Pitch Cu/Sn Micro-Bumps for 3-D Super-Chip Stack," *Proc. 3D System Integration*, pp. 1-6, 2009.

[104] G. Katti et al., "Temperature Dependent Electrical Characteristics of Through-Si-Via (TSV) Interconnections," *International Interconnect Technology Conference*, pp. 1-3, 2010.

[105] D. Velenis, E. J. Marinissen, and E. Beyne, "Cost Effectiveness of 3D Integration Options," *Proc. 3D Systems Integration Conference*, pp. 1-6, 2010.

[106] Open core circuits.
http://www.opencores.org
Accessed March 2012.

[107] D. H. Kim, K. Athikulwongse, and S. K. Lim, "A Study of Through-Silicon-Via Impact on the 3D Stacked IC Layout", *Proc. IEEE International Conference on Computer-Aided Design*, 2009.

[108] A. D. Trigg et al., "Design and Fabrication of a Reliability Test Chip for 3D-TSV," *Proc. Electronic Components and Technology Conference*, pp.79-83, 2010.

[109] S. L. Wright, P. S. Andry, E. Sprogis, B. Dang, and R. J. Polastre, "Reliability Testing of Through-Silicon Vias for High-Current 3D Applications," *Proc. Electronic Components and Technology Conference*, pp. 879-883, 2008.

[110] L. J. Thomas, H. K. Kow, and S. Palasundram, "Current Capacity Evaluation of a Cantilever Probe," *Proc. Electronic Manufacturing Technology Symposium*, pp.1-6, 4-6, 2008.

[111] K. M. Butler, J. Saxena, A. Jain, T. Fryars, J. Lewis, and G. Hetherington, "Minimizing Power Consumption in Scan Testing: Pattern Generation and DFT Techniques," *Proc. IEEE International Test Conference*, pp. 355-364, 2004.

[112] X. Wen, "Towards the Next Generation of Low-Power Test Technologies," *Proc. IEEE International Conference on ASIC*, pp.232-235, 2011.

[113] B. Noia, S. Panth, K. Chakrabarty, and S.K. Lim, "Scan Test of Die Logic in 3D ICs Using TSV Probing", accepted for publication in *Proc. IEEE Transactions on VLSI Systems*, 2014.

[114] D. Gizopoulos, K. Roy, P. Girard, N. Nicolici, and X. Wen, "Power-Aware Testing and Test Strategies for Low-Power Devices," *Proc. Design, Automation and Test in Europe*, pp. 10-14, 2008.

[115] B. Noia and K. Chakrabarty, "Pre-bond Probing of TSVs in 3D Stacked ICs", *Proc. IEEE International Test Conference*, pp. 1-10, 2011.

[116] B. Noia, S. Panth, K. Chakrabarty and S. K. Lim, "Scan Test of Die Logic in 3D ICs Using TSV Probing", accepted for publication in *Proc. IEEE International Test Conference*, pp. 1-8, 2012.

[117] C. E. Leiserson and J. B. Saxe, "Optimizing Synchronous Systems", *Journal of VLSI Computing Systems*, vol. 1, pp. 41-67, 1983.

[118] G. De Micheli, "Synchronous Logic Synthesis: Algorithms for Cycletime Minimization", *IEEE Trans. Computer-Aided Design*, vol. 10, pp. 63-73, 1991.

[119] J. Monteiro, S. Devadas, and A. Ghosh, "Retiming Sequential Circuits for Low Power", *Proc. IEEE International Conference on CAD*, pp. 398-402, 1993.

[120] S. Dey and S. Chakradhar, "Retiming Sequential Circuits to Enhance Testability", *Proc. IEEE VLSI Test Symposium*, pp. 28-33, 1994.

[121] T. C. Tien, H. P. Su, and Y. W. Tsay, "Integrating Logic Retiming and Register Placement", *Proc. IEEE Intl. Conf. Computer-Aided Design*, pp. 136- 139, 1998.

[122] O. Sinanoglu and V. D. Agrawal, "Eliminating the Timing Penalty of Scan", *Journal of Electronic Testing: Theory and Applications*, vol. 29, Issue 1, pp. 103-114, 2013.

[123] Nangate Open Cell Library.
http://www.si2.org/openeda.si2.org/projects/nangatelib
Accessed March 2012.

[124] Kenneth P. Parker. *The Boundary Scan Handbook*. Springer-Verlag, 2003.

[125] IEEE 3D-Test Working Group.
http://grouper.ieee.org/groups/3Dtest/
Accessed August 2013.

[126] E.J. Marinissen, C.-C. Chi, J. Verbree, and M. Konijnenburg, "3D DfT Architecture for Pre-Bond and Post-Bond Testing", *Proc. IEEE 3D Systems Integration Conference*, pp. 1-8, 2010.

[127] F. Brglez, D. Bryan, and K. Kozminski, "Combinational Profiles of Sequential Benchmark Circuits", *Proc. IEEE International Symposium on Circuits and Systems*, pp. 1924-1934, 1989.

Biography

Brandon Noia was born in Syracuse, NY, on the 13^{th} of September, 1985. He earned his Bachelor's degree from Duke University in 2008 and his Ph.D. in Electrical and Computer Engineering from Duke University in 2014.

Brandon earned an SRC/Global Research Collaboration Master's Scholarship in 2008 to work in the area of 3D test. In 2010, he was awarded an SRC Graduate Research Fellowship to continue his work.

Brandon earned 2^{nd} place in the ACM DAC Student Research Competition in 2012. He won the Best Oral Presentation at the Duke ECE Graduate Research Workshop in 2012, as well as the Best in Session Award at TECHCON in 2012 for his work on pre-bond TSV probing. He also received the ACM SIGDA Turing Celebration Travel Scholarship in 2012 and was a Design Automation Conference Young Student Support recipient in 2008.

Brandon has numerous conference and journal publications, including publications in IET Computers and Digital Techniques, IEEE Transactions on Computer-Aided Design, and the Journal of Electronic Testing: Theory and Applications. He has presented his work at conferences across the world, including the North Atlantic Test Workshop, the IEEE International Conference on Computer Design, the IEEE European Test Symposium, the IEEE Asian Test Symposium, the IEEE International Test Conference, the IEEE VLSI Test Symposium, and the IEEE International 3D System Integration Conference. Brandon's journal publications include:

1. B. Noia and K. Chakrabarty, "Test-wrapper optimisation for modular testing of em-

bedded cores in TSV-based three-dimensional SOCs", *IET Computers and Digital Techniques*, vol. 5, pp. 186-197, May 2011.

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

3. B. Noia, K. Chakrabarty and E. J. Marinissen, "Optimization methods for post-bond testing of 3D stacked ICs", *Journal of Electronic Testing: Theory and Applications*, vol. 28, pp. 103-120, February 2012.

4. B. Noia and K. Chakrabarty, "TSV probing for pre-bond testing of 3D stacked ICs", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, pp. 547-558, April 2013.

5. B. Noia and K. Chakrabarty, "Retiming for delay recovery after DfT insertion on inter-die paths in 3D ICs", accepted publication in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2013.

6. B. Noia, S. Panth, K. Chakrabarty, and S.K. Lim, "Scan Test of Die Logic in 3D ICs Using TSV Probing", accepted for publication in *Proc. IEEE Transactions on VLSI Systems*, 2014.

Brandon's conference and workshop publications include:

1. B. Noia and K. Chakrabarty, "Test-wrapper optimization for embedded cores in TSV-based three-dimensional SOCs", *IEEE North Atlantic Test Workshop*, May 2009.

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

3. B. Noia, S. K. Goel, K. Chakrabarty, E. J. Marinissen, and J. Verbree, “Test-architecture optimization for TSV-based 3D stacked ICs”, *Proc. IEEE European Test Symposium*, pp. 24-29, 2010.

4. B. Noia, K. Chakrabarty and E. J. Marinissen, “Optimization methods for post-bond die-internal/external testing in 3D stacked ICs”, *Proc. IEEE International Test Conference*, 2010.

5. B. Noia and K. Chakrabarty, “Pre-bond probing of TSVs in 3D stacked ICs”, *Proc. IEEE International Test Conference*, 2011.

6. B. Noia and K. Chakrabarty, “Identification of defective TSVs in pre-bond testing of 3D ICs”, *Proc. IEEE Asian Test Symposium*, 2011.

7. B. Noia and K. Chakrabarty, ”Pre-bond probing of TSVs in 3D stacked ICs”, presented at the *ATE 2020 Workshop*, 2011 (no formal publication).

8. B. Noia and K. Chakrabarty, “Testing and design-for-testability techniques for 3D integrated circuits“ (invited paper), *Proc. IEEE Asian Test Symposium*, 2011.

9. B. Noia and K. Chakrabarty, “Pre-bond testing of die logic and TSVs in high performance 3D-SICs”, *Proc. IEEE International 3D System Integration Conference*, 2012.

10. B. Noia, S. Panth, K. Chakrabarty and S. K. Lim, “Scan test of die logic in 3D ICs using TSV probing”, *Proc. IEEE International Test Conference*, pp. 1-8, 2012.

11. B. Noia and K. Chakrabarty, “Post-DfT-insertion retiming for delay recovery on inter-die paths in 3D ICs”, *Proc. IEEE VLSI Test Symposium*, pp. 264-269, 2013.

12. Z. Zhang, B. Noia, K. Chakrabarty and P. Franzon, “Face-to-face bus design with built-in self-test in 3D ICs”, *Proc. IEEE 3D Systems Integration Conference*, pp. 1-7, 2013.

13. B. Noia and K. Chakrabarty, “Pre-bond probing of TSVs in 3D stacked IC”, *SRC TECHCON Conf.*, 2011.

Brandon has three patents pending. These patents cover several topics in this dissertation, including pre-bond KGD test and 3D retiming flows. These patents are:

1. B. Noia and K. Chakrabarty, “Method and Architecture for Pre-Bond Probing of TSVs in 3D Stacked Integrated Circuits”, US Patent Application No. US-2013-0006557-A1, January 2013.
2. B. Noia and K. Chakrabarty, “Scan Test of Die Logic in 3D ICs using TSV Probing”, US Patent Application No. 13/666,696, November 2012.
3. B. Noia and K. Chakrabarty, “Retiming-Based Design Flow for Delay Recovery on Inter-Die Paths in 3D ICs”, US Patent Application No. 13/919,022, June 2013.

Brandon has further co-authored a book:

Noia, Brandon and Chakrabarty, Krishnendu. *Design-for-Test and Test Optimization Techniques for TSV-based 3D Stacked ICs*. Springer, 2014.