# Experimental Study of Scan Based Transition Fault Testing Techniques

**Vinay B. Jayaram**

Thesis submitted to the Faculty of
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Engineering

Dr. Michael S. Hsiao: Chair
Dr. James M. Baker: Member
Dr. Sandeep K. Shukla: Member

January 29, 2003

Bradley Department of Electrical and Computer Engineering
Blacksburg, Virginia.

Keywords: Transition faults, ATPG, pattern volume, test coverage, broad-side, skewed load

# Experimental Study of Scan-Based Transition Fault Testing Techniques

**Vinay B. Jayaram**

## Abstract

*The presence of delay-inducing defects is causing increasing concern in the semiconductor industry today. To test for such delay-inducing defects, scan-based transition fault testing techniques are being implemented. There exist organized techniques to generate test patterns for the transition fault model and the two popular methods being used are Broad-side delay test (Launch-from-capture) and Skewed load delay test (Launch-from-shift). Each method has its own drawbacks and many practical issues are associated with pattern generation and application. Our work focuses on the implementation and comparison of these transition fault testing techniques on multiple industrial ASIC designs. In this thesis, we present results from multiple designs and compare the two techniques with respect to test coverage, pattern volume and pattern generation time. For both methods, we discuss the effects of multiple clock domains, tester hardware considerations, false and multi-cycle paths and the implications of using a low cost tester. We then consider the implications of pattern volume on testing both stuck-at and transition faults and the effects of using transition fault patterns to test stuck-at faults. Finally, we present results from our analysis on switching activity of nets in the design, while executing transition fault patterns.*

# Acknowledgements

This section is dedicated to all the people who have been instrumental not only in the completion of this thesis but also during my entire graduate studies.

First, I'd like to thank my advisor, Dr. Michael S. Hsiao for his support, advice and understanding throughout the latter part of my stay at Virginia Tech. I'm very thankful for his timely help and encouragement and shall always cherish my interactions with him.

I'm also thankful to Dr. James M. Baker and Dr. Sandeep K. Shukla for agreeing to serve on my committee.

I am deeply indebted to Texas Instruments for having provided me with an opportunity to participate in the co-op program at their headquarters in Dallas, TX. I'm grateful to the ASIC division for allowing me to work on my thesis during this time. In particular, I'd like to thank Dr. Kenneth M. Butler and Dr. Jayashree Saxena for their invaluable help, advice and encouragement throughout my stay at TI. Without them, this work would not have been possible. I also appreciate the support and encouragement provided by Michael Sump and Dr. Bret Stewart.

A heartfelt thank you to Dr. Paul Dickinson at Sun Microsystems, for introducing me to DFT in the industry and for all the valuable advice, encouragement and knowledge imparted to me during my stay at Sun. I shall always cherish my association with him.

Before I conclude, I would like to mention the unhindered love, support and encouragement provided by my parents, brother and Antara. Finally, I wish to express my gratitude to God, for his countless blessings.

Vinay B. Jayaram
January 2003

# Table of Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

In recent years, a multitude of advances have been made in semiconductor technology, thus enabling industries to design and develop highly complicated integrated circuits with an ever-increasing number of components and logic gates per chip. These increased densities now enable designers to implement an array of complex digital logic functions, often on a single chip. From the viewpoint of industries and consumers, two points are of interest. Firstly, the designed product has to work correctly for the desired application, fully meeting all the functional requirements it was originally designed for. Secondly, it should continue to function correctly, for the life-time of its application. To make sure that these two goals are met, the components are to be completely tested before being released to the market. The principle goal of *testing* would thus be to check the manufactured product completely, making sure that it functions according to its requirements and *detect* malfunctions and defects in the product.

## 1.1 Testing environment

In the industry today, automatic test equipment (ATE) is being used to test large and complex digital circuits assembled on a chip. A model of an ATE with related components is shown in Figure 1. Such ATEs are loaded with a set of test patterns (vectors), which are generated by test engineers. These test patterns are designed to test different parts of the circuitry for many kinds of manufacturing defects. The ATEs are responsible for the application of these test vectors to the *Unit under Test* (UUT). The ATEs also make use of a fault-free model of the circuit to calculate fault-free responses to the same patterns. The fault-free responses can either be available from the functional specification of the product, or can be calculated using a hardware emulation or software simulation of the designed system. Once the vectors are applied to the UUT, the ATE obtains responses from outputs of the UUT and compares them with responses from the fault-free model. If there is any discrepancy, errors are reported and logged by the test controller.

## 1.2 Types of defects

Manufacturing defects are responsible for a majority of chip-failures that are detected at various stages. An *error* in the operation of a device is the manifested behavior of an underlying *defect*. To make the testing of devices easier, these defects are modeled as *faults*. Testing for such defects is usually carried out on all or part of the chips manufactured, depending on the cost of the chip, cost of test and other factors. Defects that can occur during manufacturing can involve silicon defects, mask contamination, process variation, defective oxide, interconnect defects and photolithography defects. The *electrical effects* of such defects involve shorts or opens in the designs, or transistors being stuck-open or stuck-closed. *Logical effects* of defects include nets that are stuck at values of 1 or 0, wired AND or OR effects or delayed switching effects. The goal of testing is to test for such manufacturing defects and physical failures, so that we can ascertain whether or a not a fabricated chip is functioning properly.

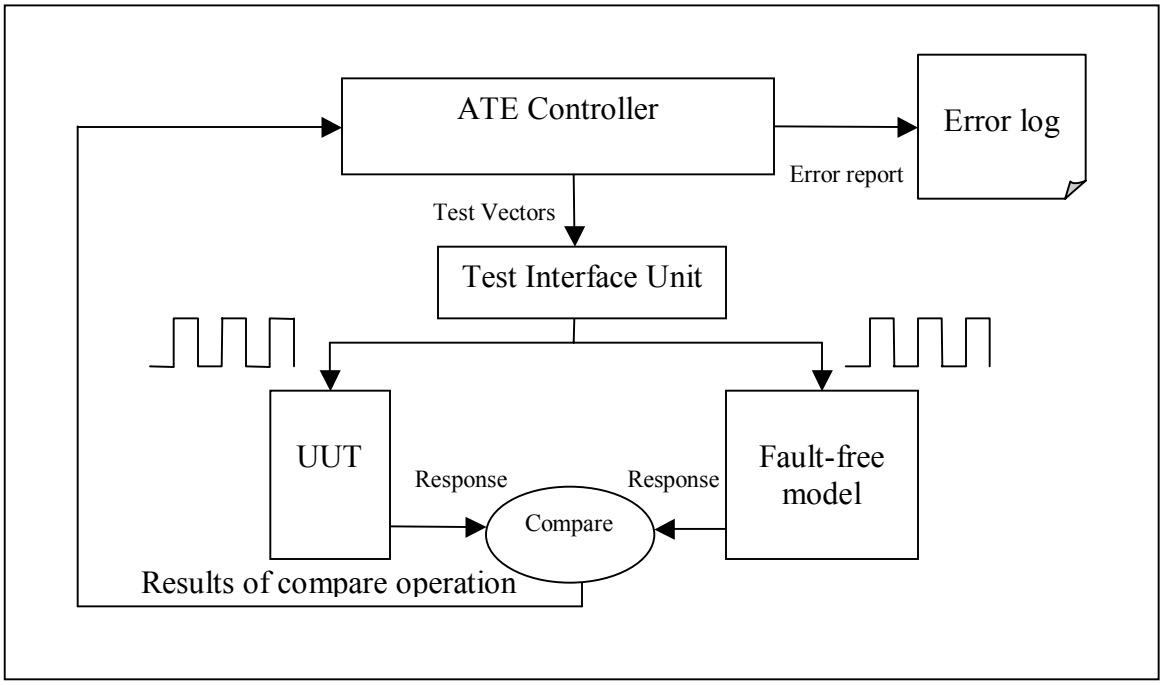



**Figure 1: Automatic Test Equipment (ATE) Model**

## 1.3 Types of testing

In the industry, two phases of testing are carried out in the life-cycle of a product. The first phase consists of *characterization tests*, while the next phase consists of *production tests*.

### 1.3.1 Characterization tests

After the completion of design and verification, the first set of chips (also referred to as parts) manufactured are for experimentation use only. These are also referred to as first-pass silicon and are usually tested in wafer form. These wafers are subjected to characterization tests, which include:

- Measurements of AC, DC and voltage characteristics.
- Probing of internal nodes to observe electrical failures.
- Critical path testing to see if all timing requirements are met.
- Reliability tests, which measure the speed of gate switching and operating values.

These tests are usually ad hoc, and are geared towards initial analysis and fine tuning purposes.

### 1.3.2 Production tests

Defects in the fabrication process can occur randomly across wafers during manufacturing and production tests are geared towards identifying all such defects. Test patterns are applied using ATE and a variety of faults are tested. To ensure high reliability, production tests also include burn-in tests, where the manufactured units are tested across a range of high temperatures and high voltages.

## 1.4 Fault models

Physical failures and fabrication defects cannot be easily modeled mathematically. As a result, these failures and defects are modeled as *logical faults*. **Structural faults** relate to the structural model of a system and these affect interconnections among components of a design. **Functional faults** relate to a functional model, for example an RTL/HDL model and these affect the nature of operation of the components in a design. Testing for functional faults validates the correct operation of a system, while testing of structural faults targets manufacturing defects.

Two popular structural fault models are prevalent in the industry today and they are the **stuck-at fault model** and the **transition fault model**. Stuck-at faults affect the logical behavior of the system, while transition faults affect the timing/temporal behavior of the system. An additional fault model being used is the **path-delay fault model**, which is also based on the timing behavior of the system, but cumulative delays along paths are considered, in lieu of delays at each net as in the transition fault model. In this thesis, we deal with structural faults alone, and mainly concentrate on transition faults.

### 1.4.1 Stuck-at fault model

The main assumption made by the stuck-at fault model is that the components of a circuit are fault-free and only their interconnections are defective. However, it has been shown that stuck-at fault tests are effective in capturing a wide range of defects on manufactured chips. This model represents faults caused by opens, shorts with power or ground, and internal faults in the components driving signals that keep them stuck-at a logic value [1].

To test for stuck-at faults, two steps are involved. The first step is to generate a test vector that excites the fault and the next step is to propagate the faulty effect to a primary output or a scan flip-flop. Automatic test pattern generation (ATPG) tools are typically used to generate the test vectors. The stuck-at fault model is being used virtually everywhere in

the industry to screen defects caused by stuck-at faults. It is relatively easy to generate patterns for stuck-at faults and pattern volume is also comparatively low.

### 1.4.2 Transition fault model

The transition fault model is similar to the stuck-at fault model in many respects. The effect of a transition fault at any point P in a circuit is that any transition at P will not reach a scan flip-flop or a primary output within the stipulated clock period of the circuit. According to the transition fault model [6], there are two types of faults possible on all lines in the circuit: a slow-to-rise fault (STR) and a slow-to-fall fault (STF). A slow-to-rise fault at a node means that any transition from 0 to 1 on the node does not produce the correct result when the device is operating at its maximum operating frequency. Similarly, a slow-to-fall fault means that a transition from 1 to 0 on a node does not produce the correct result at full operating frequency.

In any circuit, *slack* of a path can be defined as the difference between the clock period when the circuit outputs are latched and the propagation delay of the path under consideration. For a gate level delay fault to cause an incorrect value to be latched at a circuit output, the size of the delay fault must be such that it exceeds the slack of at least one path from the site of the fault to the site of an output pin or scan flip-flop. If the propagation delays of all paths passing through the fault site exceed the clock period, such a fault is referred to as a *gross delay fault* [17].

Any test pattern that successfully detects a transition fault comprises of a pair of vectors $\{V_1, V_2\}$, where $V_1$ is the initial vector that sets a target node to the initial value, and $V_2$ is the next vector that not only launches the transition at the corresponding node, but also propagates the effect of the transition to a primary output or a scan flip-flop [25].

In other words, a set of test vectors that test for a delay fault at the output or input of a gate are such that:

- A desired transition is launched at the site of the fault.

- If the fault is a slow-to-rise fault, the final pattern is a test for a corresponding stuck-at-0 fault, and if the fault is a slow-to-fall fault, the final pattern is a test for a corresponding stuck-at-1 fault.

When compared with tests for stuck-at faults, it can be seen that the only additional requirement to test for transition faults is the presence of a pattern that initializes a node to the required value, just before the application of a stuck-at fault pattern. One might expect that the fault coverage attained by testing transition fault patterns will be close to that attained by testing stuck-at fault patterns. However, we should remember that the fault coverage obtained for transition fault patterns represent only gross delay faults. More detailed analysis will be necessary to evaluate for smaller delay faults [26].

There are three main methods that can be used to generate and apply transition fault tests. The first method, termed **Broad-side delay test**, is also referred to as *functional justification* or the *launch-from-capture technique*. In this technique, the first vector of the pair is scanned into the chain and the second vector is derived as the combinational circuit's response to the first vector [16]. The second method, termed **Skewed load transition testing**, is also referred to as the *launch-from-shift technique*. In this method, both the first and second vectors of the pair are delivered through the scan cells themselves [16]. If the scan-chain is N bits long, an N-bit vector is loaded by scanning in the first (N-1) bits. The last shift clock is used to launch the transition, followed by a quick capture. In the third method, termed **Enhanced-scan transition testing**, the two vectors (V1, V2) are stored in the tester memory. Vector V1 is first applied and this initializes the circuit. Vector V2 is then scanned in, followed by applying it to the circuit under test and capturing its response. The important point is that it is assumed that the initialization provided by V1 is not lost while loading V2. Therefore, this type of test assumes a hold-scan design [27]. For inclusion of hold-scan cells, an area overhead is evident and there is an additional routing requirement for the control signal. As a result, such hold-scan cells are not used in the ASIC industry and thus, enhanced scan-design is not always useful in a practical environment. The main focus of this thesis is to compare

the implementation, performance and tradeoffs with using Broad-side and Skewed-load transition fault testing techniques on industrial designs.

### 1.4.3 Path delay fault model

The path delay fault model [5] takes the sum of all delays along a path into effect, while the transition fault model accounts for localized faults (delays) at the inputs and outputs of each gate. There may be cases where the gate delays of individual faults are within specified limits, but the cumulative effect of all faults on a path may cause an incorrect value to be latched at the primary outputs, if the total delay exceeds the functional clock period. The transition fault model cannot account for such defects, but the path delay fault model can. However, in a design containing $n$ lines, there can be a maximum on $2n$ transition faults (a slow-to-rise and slow-to-fall fault on each line), but there can potentially be $n^2$ (or exponential) path delay faults (considering all possible paths) [17]. Since all the paths cannot be tested, the path delay model requires identification and analysis of critical paths in the design. This makes it more complicated to use on large designs and hence, the transition fault model has been accepted as a good method to test for delay faults in the industry [7], [8].

## 1.5 Use of functional patterns for testing

Before scan-based structural test techniques were implemented, functional pattern testing was the only test method being employed in manufacturing test. Many problems are associated with using functional at-speed patterns. The major problems include the cost associated with developing them, difficulty in debugging them and the high cost of test hardware. Fault simulation is a major problem as very few tools exist that can fault grade functional patterns for delay fault coverage. Since the quality cannot be easily assessed, the level of fault coverage will be unknown. Even if such tests are generated, a large pattern volume is required to achieve the necessary coverage. As clock rates and design sizes increased, testing using functional patterns became more impractical.

## 1.6 Testing approaches – Scan based vs. logic BIST

In the past few years, structured scan-based methods, most often full-scan, are being increasingly used to generate test patterns that are capable of achieving high coverages. The advantage lies in the short cycle time for developing them, combined with the relative ease in debugging them. Test application times may not be short if many flip-slops are included in long serial scan chains. Designers often make use of multiple parallel scan chains to reduce the test application time. However, as design sizes increase, most types of automatic test equipment (ATE) are incapable of supporting the increased number and depth of scan chains. Scan based methods still remain the most viable alternative to functional tests and many improvements are being devised for such methods [13] [14].

Another approach to test delay faults is logic BIST. In such an approach, both test pattern generation and output response comparison functions are built into the chip [9]. With a BIST-only approach, high coverages have been reported [10]. Hybrids of logic BIST and deterministic ATPG methods have been proposed recently, which have the benefits of low tester data volume and high fault coverage [11] - [13]. Many problems are associated with logic BIST and other related approaches. The common problems among them include:

- There is an incremental area penalty for the BIST controller, which may not be easily justifiable.
- The requirement of an at-speed scan-enable calls for added changes to the design and test hardware.
- There is an added routing complexity.
- Complications with timing closure are always a concern.

Further, to adhere to commercial BIST vendor tool requirements, the clock distribution and clock tree balancing tasks can be further complicated [14]. Such design demands can

be overwhelming for most designs and hence, simpler structural test techniques may be the only viable option in the current industry scenario.

## 1.7 Previous work

Tremendous amount of work has been done both in academia and the industry to better understand the testing of delay faults. Savir [28] gives an excellent summary on the early developments in delay testing and provides a comprehensive bibliography on early work on transition fault testing. His work [28] gives a good introduction to concepts, algorithms and circuits that are used in conjunction with delay test, also referred to as AC Test.

Savir and Patil introduced scan-based transition test in [18] and [19] and gave exhaustive information on skewed load testing. In [18], the calculus that allows one to compute a complete set of skewed-load transition fault test vectors is introduced. The calculus is capable of computing both the first and second vectors constituting the delay-test pair. The techniques used to generate skewed load transition tests (SLOTT) vectors is illustrated in this work, and it is shown that this method has a relatively low complexity compared to other methods. A single time frame calculation of SLOTT, based on the Boolean difference technique is shown. Undetectability of transition faults due to shift dependencies is also shown. A *displaced function* is included in the calculation, which separates the detection probabilities of stuck-at and transition faults.

The work reported by Savir and Patil in [19] follows up with [18] in discussing the coverage issues associated with SLOTT. This work shows how some faults may not be detected by SLOTT due to shift dependency originating in the scan chains. Based on the topology of the circuit, a practical lower bound on coverage is calculated and demonstrated on a family of ISCAS benchmark circuits. This work also discusses methods to improve test coverage. One such approach examines the improvements in coverage achieved by reordering the inputs. It is also shown that input ordering tries to reduce the effect of shift dependency on creating valid SLOTT pairs for transition fault tests.

In [29], Savir and Patil introduce Broad-side delay test as another method to test for transition delay faults. They term this method Broad-side, as the second vector of the transition test pair is derived as the combinational circuit's response to the first vector. In this work, the effectiveness of this technique is examined and methods to generate these vectors using existing stuck-at fault pattern generation tools are shown. This work also explains the calculation of the detection probability of a transition fault, subjected to broad-side patterns. It is finally shown that the broad-side technique is inferior to the skewed- load technique and the reasons for this are identified. One reason is the limited capability of the broad-side method to generate a rich set of two-pattern sets in networks where the sequential depth is not greater than 1. These are also referred to as single section networks. A major problem is that nothing can be done in terms of *Design for Testability (DFT)* to counter this problem. It is also shown that a hybrid of the broad-side and skewed-load methods can be used to improve testability, in cases where the skewed-load tests fail to detect some faults, even after input re-ordering. It is pointed out in this work that the broad-side technique is predominantly a sequential ATPG problem and as a result, pattern volumes are typically high, owing to the fact that sequential pattern compression engines are not as powerful as combinational pattern compression engines. Further, simulation time is significantly higher than the skewed-load method. Finally, it is shown that no lower bound can be placed on the coverage that can be obtained using the broad-side method, as can be for the skewed-load method.

In [2], Maxwell et al. showed results of manufacturing test on an industrial ASIC and presented the relative effectiveness of scan based AC tests. They showed that transition fault testing is absolutely essential to capture at speed defects. In [3]-[4], Nigh et al. further showed that any one type of test is not enough to cover a broad range of defects and that all tests, stuck-at, transition and IDDQ are required to constitute a comprehensive test strategy.

In [3], Nigh et al. make some interesting observations about the amount of time that has to be allocated to each testing strategy. Given a fixed amount of time, say X seconds, this

work tries to identify the portions of time to be allocated to stuck-at test, transition test, functional test and IDDQ test. The time required is directly related to the pattern volume and hence, this is work is of interest to us. In subsequent sections of our thesis, we deal with reusing transition patterns for stuck-at test and analyze the tradeoffs of using transition patterns for stuck-at test.

Maxwell et al. analyzed the effectiveness of functional tests and compared them with scan test vectors in [30]. They observe that some chips that fail AC test on executing scan tests actually pass functional tests. It so happens that in a lot of cases, the AC tests that fail are actually functionally unsensitizable. In other words, scan tests in some cases may lead to over-testing and thus result in good parts being discarded as faulty. This is referred to as yield loss. In our work, we address the yield loss problem with respect to skewed-load test vectors.

Tumin et al. present results from industrial experiments in [33], where they address the question: "Can scan patterns replace functional patterns?" This is a very important question facing engineers in the industry now, especially for high performance designs. It is well known that the development of functional patterns involves a significant investment in cost, time and other resources. Results from this work show that scan patterns can detect all or a majority of the faults detected by functional patterns, thus instilling confidence in the industry's approach to using scan patterns for testing most faults . This is important for our work, as we rely mainly on scan based patterns to detect transition faults in our designs.

Tendolkar et al. present the implementation details and results from their work on generating and applying transition fault patterns on Motorola's microprocessors based on the PowerPC$^{TM}$ instruction set architecture [7]. They implement transition fault testing using the broad-side model (launch-from-capture) and show that they require about 15,000 vectors to achieve 76% transition fault coverage for the MPC7400 microprocessor, consisting of 10.5 million transistors. In contrast, we deal with ASIC

designs in our thesis and present implementation details for both the launch-from-capture and launch-from-shift methods and infer the pros and cons of each approach.

Saxena et al. present an overall approach for implementing a scan based delay test strategy, with particular reference to low cost test equipment in [15]. They explain the basics of transition fault testing using the launch-from-capture method, design challenges they faced and how they overcome them. They deal in detail with low cost test equipment and provide the problems they face with implementing a transition fault testing strategy on such low cost testers. They follow up with solutions on how they overcome these hurdles and present results on industrial ASIC designs. They however, do not deal with the implications of using the launch-from-shift strategy, tradeoffs involved and advantages/disadvantages of each method. In our thesis, we follow up on the work reported in [15] and analyze the effects of decisions made to support launch-from-capture and launch-from-shift strategies. We quantify the effects of using low cost test equipment and draw conclusions from them.

Moving away from scan based testing, Kusko et al. present implementation details of transition test methodology using only logic BIST [10]. In a logic BIST approach, the features of test pattern generation and output response comparison are built into the chip itself. The obvious advantage is the savings achieved in test application time, as the test generation circuitry is present on the chip itself. Unfortunately, there are many disadvantages with logic BIST also. First up, there is an incremental area penalty for the logic BIST controller, which has to be present on the chip. Further, additional area penalty incurs with supporting modifications to the scan-enable signal, added routing, extra controllability/observability and other design changes. In addition, logic BIST can complicate the delicate task of timing closure. In this work [10], the authors present the challenges involved in supporting a logic BIST only approach on a 1 GHz IBM zSeries 900 chip for AC test capability.

In [9], Hetherington et al. discuss the implementation of logic BIST for industrial ASIC designs. They discuss the implementation of a BIST compatible core along with a logic

BIST controller for at-speed testing. They discuss the techniques to automatically identify and bound X generators, bypass RAMS, bound I/Os and insert test points. They also show the implementation of the BIST controller that can support at-speed testing of multiple clock frequencies, thus enabling the testing of various intra-clock and inter-clock domains at-speed.

## 1.8 Organization of this thesis

In this thesis, we present the details of implementing a structured scan-based delay test strategy using the transition fault model on industrial ASIC designs. We consider both the launch-from-capture and launch-from shift approaches and outline the pros and cons of each technique.

To begin with, we describe the transition fault model in brief and explain the different techniques used in detecting transition faults in Chapter 2. In addition, we detail the practical implications of supporting these two techniques, principles to be followed and test pattern information.

Next, in Chapter 3, we compare the different techniques from an implementation viewpoint and cover the practical issues that influence the effectiveness of these transition fault tests. We then move on to design features that make transition fault testing challenging and present the solutions we use to conquer these challenges. We cover the implications of using low cost test equipment and analyze the capabilities of tools to generate patterns focused towards low cost test equipment.

In Chapter 4, we present the implementation details and the capabilities of commercial tools we use. Further, we outline the methods to be followed while using these commercial tools. We then present results from multiple industrial ASIC designs and compare the two techniques, launch-from-capture and launch-from-shift with a variety of metrics. The main metrics we compare are test coverage, pattern volume and pattern generation time. We then continue to quantify the implications of design decisions we made and the tradeoffs we experience by using low cost test equipment.

In Chapter 5, we address the implications of pattern volume on manufacturing test in detail. We analyze pattern volume for stuck-at faults and transition faults individually and then consider possibilities for combining the two testing approaches. We discuss methods to use transition fault patterns to test for stuck-at faults and quantify the benefits acquired. We present results from two industrial ASIC designs and show how transition fault patterns can be used to test stuck-at faults, thus reducing pattern volume.

In Chapter 6, we touch upon the issue of power consumption and switching activity in designs while testing for transition faults. We show the implementation of a method to measure switching activity while executing transition fault patterns. We present results from two industrial ASIC designs and discuss methods that can be used to reduce switching activity.

Finally, in Chapter 7, we summarize with the observations we have made along the course of this thesis and present possible topics for future research. We believe the methods described in this thesis will aid others in developing/improving methods to test for transition faults in the industry.

# Chapter 2

# Testing for Delay Faults using the Transition Fault Model

As mentioned in Section 1.4.2, the effect of a transition fault at any point P in a circuit is that any transition at P will not reach a scan flip-flop or a primary output within the stipulated clock period of the circuit. According to the transition fault model [6], there are two types of faults possible on all lines in the circuit: a slow-to-rise fault and a slow-to-fall fault. Any test pattern that successfully detects a transition fault comprises of a pair of vectors $\{V_1, V_2\}$, where $V_1$ is the initial vector that sets a target node to the initial value, and $V_2$ is the next vector that not only launches the transition at the corresponding node, but also propagates the effect of the transition to a primary output or a scan flip-flop [25].

The main advantage of using the transition fault model is that it is similar to stuck-at ATPG and as a result, the tools being used already have the capability. One more major advantage is that a majority of the logic area of the die is covered, as required for manufacturing test. In this chapter, we deal with different techniques for testing transition faults and the challenges posed by these techniques in industrial designs. The two prevalent structured transition fault testing techniques are the "launch-from-capture" technique [16], [17] and the launch-from-shift technique [18], [19].

## 2.1 Launch-from-capture technique

This technique is also known as the "broad-side" or "functional justification" technique. As we know, transition fault tests require a pair of vectors - one to set a target node to an initial value and the next to launch the transition and propagate the effect to a primary output or scan cell [6], [20]. In this technique, the first vector of the pair is scanned into the chain and the second vector is derived as the combinational circuit's response to the first vector [16].

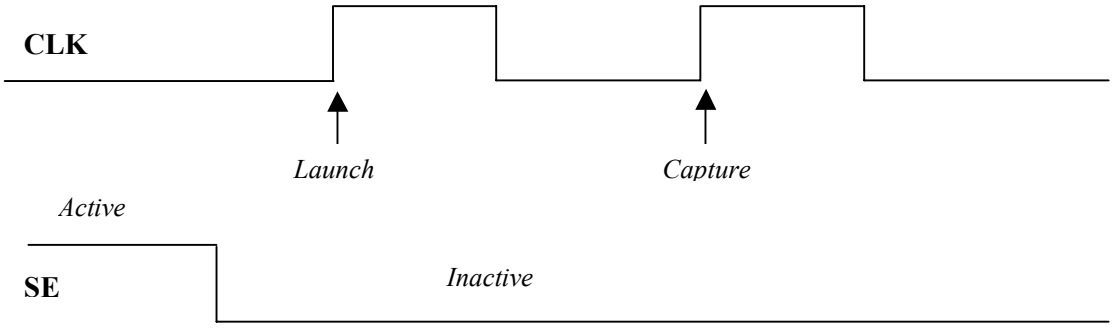



**Figure 2: Timing diagram for the launch-from-capture technique**

In a scan-based design, if the scan chain contains N cells, a vector pair is obtained by applying the following steps:

- Shift the data into the scan-chain N times.
- Toggle the scan-enable signal and allow the circuit to settle (new PI values may be applied if required).
- Pulse the clock twice. The first pulse will launch the transition and the second pulse will capture the response from the combinational portion of the circuit.
- If required, primary input (PI) or primary output (PO) changes could be made with the application of the first clock pulse.
- If the tester hardware does not support at-speed PI changes, the PI values across launch and capture cycles will have to be held constant. If at-speed output strobing is not supported, the effects of all faults have to be observed only at flip-flops on the scan chain.

The timing diagram for this method is shown in Fig 2. The important point to note here is that the launch and capture are performed with the scan-enable signal set to functional mode. The scan-shift frequency is much slower than the functional operation frequency in most industrial designs. The scan-shift speed may also be limited by the maximum frequency supported by the tester hardware being used. As a result, two different waveforms (or timesets), one to enable scan-shift and the other to perform the at-speed capture, may need to be applied to the same clock pin, while the device is being tested.

This can be implemented with three different approaches and they are detailed below.

### 2.1.1 Launch-from-capture using a Late-Late-Early waveform

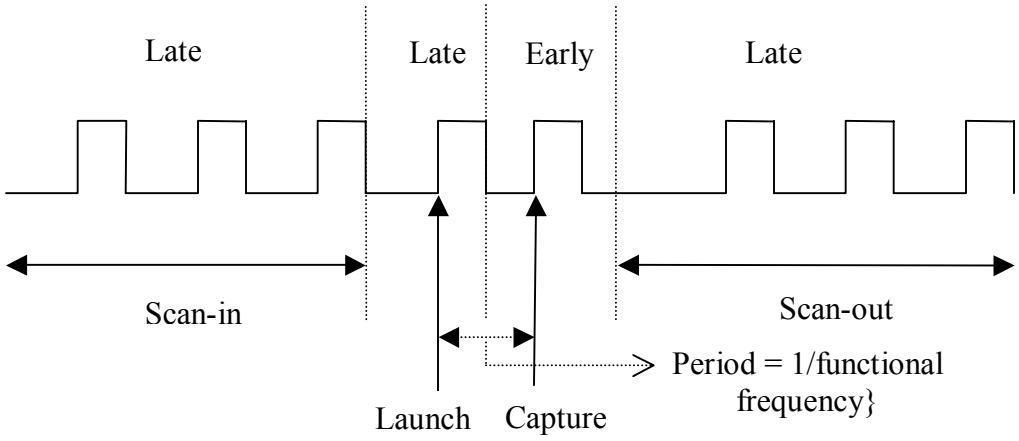



**Figure 3: Launch-from-capture: Late scan-in –Late launch - Early capture – Late scan-out**

The two-timeset waveform, one for scan-shift and launch and the other for capture is shown in Figure 3. It can be seen that scan-in and launch are performed using one time-set, labeled "Late", with the riding edge (leading edge) of the clock occurring towards the end (later half) of the cycle. The capture operation uses a different timeset, where the leading edge of the clock occurs during the early half of the cycle and this time-set is labeled "Early" in Figure 3. The time interval between the launch and capture is adjusted such that it corresponds to the functional operating frequency of the device, thus enabling at-speed testing.

### 2.1.2 Launch-from-capture using a Early-Late-Early waveform

The waveform for this approach is shown in Figure 4. In this case, an "Early" timeset is used for shifting in the data. If the design makes use of slower clock trees, the usage of this "Early" pulse for shift proves helpful, by permitting the shift pulse width to be wide enough. Following the scan-in, a pulse that has a leading edge is applied late in the cycle, thus providing the launch. This is followed by an "Early" pulse, that arrives early in the

cycle and this works as the capture pulse. Once again, the interval between the launch and capture cycles is fine tuned to reflect the functional operating frequency. An additional requirement with this method is that the scan-in data supplied at the scan-in pins of the device have to reach the scan data ports of the first flip-flops in the chain, before the early launch is applied.

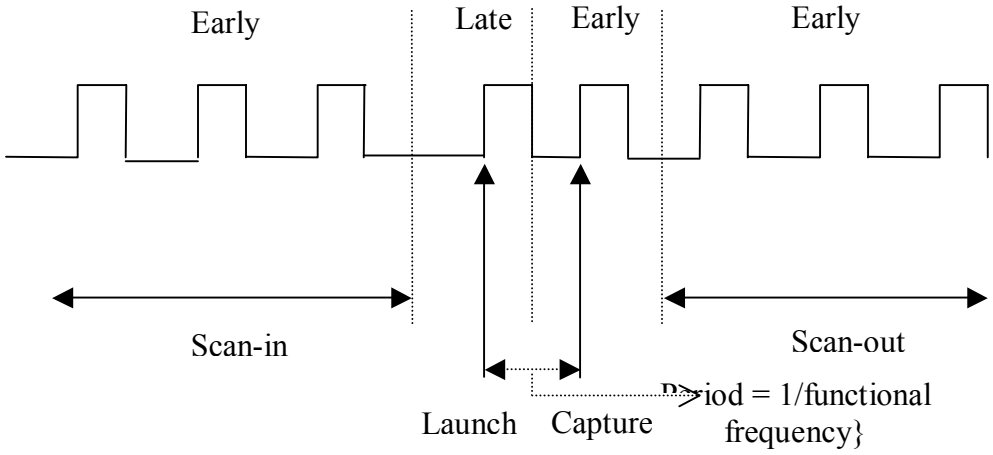



**Figure 4: Launch-from-capture: Early scan-in + Late launch - Early capture – Early scan-out**

For both the methods discussed above, a constant tester period is used for the entire duration of the test. An alternative is to use two different time periods, one for the scan-shift and the other for launch and capture.

### 2.1.3 Launch-from-capture using two different periods

The waveform for the two-period approach is depicted in Figure 5. This approach uses two different time periods, one for shift and one for launch and capture. The scan operation is completely decoupled from the launch and shift mechanism and thus offers maximum flexibility. Scan-in and scan-out are performed using the required period and clock width.

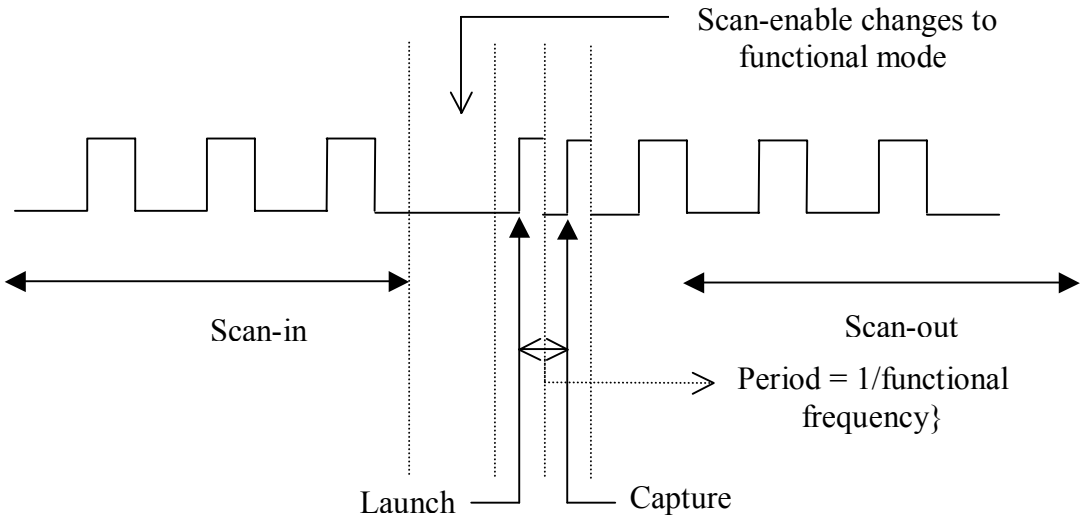



**Figure 5: Launch-from-capture: Two different periods**

In this case, the scan-in operation is performed using the maximum shift speed of the scan chains. This is followed by a cycle with no clock pulse. The period of this cycle is the same as that of the scan cycles. During this period, the scan-enable is toggled, such that the circuit enters functional mode. The PIs are then applied as required. Since this period is equal to that of the scan shift cycles, the scan-enable signal has enough time to propagate to all flip-flops, as do the PI values. Once this is done, two quick clock pulses with a period corresponding to the functional operating frequency of the circuit are pulsed. These form the launch and capture pulses. This operation is then followed by the scan-out phase, all of which is done with the maximum scan-shift frequency available.

The advantage of using two separate periods is that the launch and capture waveforms are completely decoupled from the shift waveforms. This gives the added benefit of flexibility during shift and thus, the method can be applied across a whole range of devices and test equipment.

Another major advantage is the ease with which the tester can program the same timing generator to create both edges if the launch and capture pulses are identical. When this is done, the edge placement accuracy errors are eliminated. The only source of inaccuracy

then is jitter. The maximum offset due to jitter is often less than the maximum offset due to edge placement accuracy error and hence, this approach is more favored.

If the same time period is used for all operations, the imposing of such strict pulse width requirements to keep the shift, launch and capture waveforms the same is not always possible on the test equipment available. For all the launch-from-capture results reported in this thesis, the two-period approach was used.

## 2.2 Launch-from-shift technique

This technique is also known as the "Skewed-Load" or "Transition Shifting" technique. Here, both the first and second vectors of the pair are delivered through the scan cells themselves [16]. In a scan chain containing N cells, this approach consists of the following steps:

- Shift the scan-chain (N-1) times to obtain the first vector in the pair.
- Simultaneously, apply the first of the two sets of PI values to the non-scan pins.
- Most designs consist of a muxed data scan cell, where a mux is used to choose between the value from the combinational logic and the value from the scan-chain. The scan-enable signal is used to control this mux. In such designs, setting the scan-enable signal to scan mode and shifting the scan chain once more generates the second of the two vectors.
- Toggle the scan-enable pin
- Change the PI values as required.
- Pulse the clock to capture the response data into the scan flip-flops.
- If the tester hardware supports at speed output strobing, the PO pins are strobed during this cycle to detect transition faults propagating to the POs.

The timing diagram for this method is shown in Fig 6. The most important difference between the two techniques described above with respect to muxed data scan designs is the need for at-speed scan-enable operation in the launch-from-shift technique. Further,

the launch-from-capture technique requires a sequential ATPG algorithm, while launch-from-shift patterns can be generated with a purely combinational ATPG algorithm.

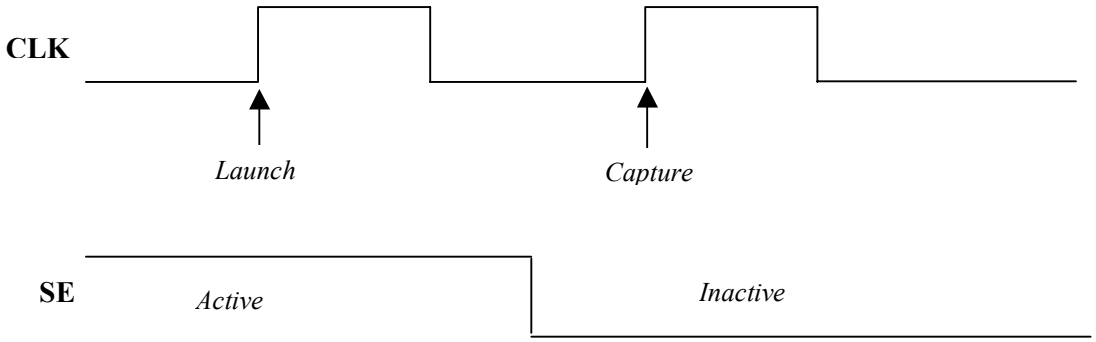



**Figure 6: Timing diagram for the launch-from-shift technique**

## 2.3 Pattern format

The format of patterns produced by the launch-from-capture and launch-from-shift techniques has been illustrated in Table 1 below.

| Launch-from-capture technique | Launch-from-shift technique |
|---|---|
| Scan in values<br>Force PIs<br>Pulse launch CLK<br>Force PIs<br>Measure POs<br>Pulse capture CLK<br>Scan out results | Initialize (force) PIs<br>Scan in values<br>Force PIs<br>Measure POs<br>Pulse CLK<br>Scan out values |

**Table 1: Pattern format in launch-from-capture and launch-from-shift methods**

From the pattern formats, it can be inferred that launch-from-capture uses a form of sequential ATPG and there are two clear launch and capture clock pulses. An observant reader may note that there is only one Pulse CLK statement in the launch-from-shift pattern template. So, how does this constitute a transition test vector?

The fact is that in the launch-from-shift technique, the transition occurs because of the last shift while the scan chains are being loaded (Scan-in values step), or the next step where primary inputs are forced.

For all the designs reported in this thesis, only the "launch-from-capture" technique is being employed for transition fault testing during manufacturing test. The lack of need of an at-speed scan enable makes the "launch-from-capture" technique the most viable option for all our designs. However, we present results from both techniques to make comparisons on test coverage, pattern volume and other metrics.

# Chapter 3

# Practical Issues that Influence Test Effectiveness

All the designs used in this set of experiments had scan capability and features of DFT already built into them. At TI, multiple tester platforms are being used for manufacturing test and a number of practical issues are associated with supporting different designs on our existing flows and tools. In this section, we describe how we deal with design features that make testing for transition faults using the launch-from-capture method difficult.

## 3.1 Implications on test equipment

As described in sections 2.1.1 - 2.1.3, there is a requirement for tester platforms to support multiple timeset switching. One class of tester used in manufacturing test has been designed as shown in Figure 7. There are two tester channels available and these can drive two different clocks with different time periods. These two channels are then "muxed" onto one device clock pin. To support the waveforms shown in Figures 3 and 4, Channel 1 marked C1 can be used to drive the "early" waveform and Channel 2 marked C2 can be used to drive the "late" waveform.

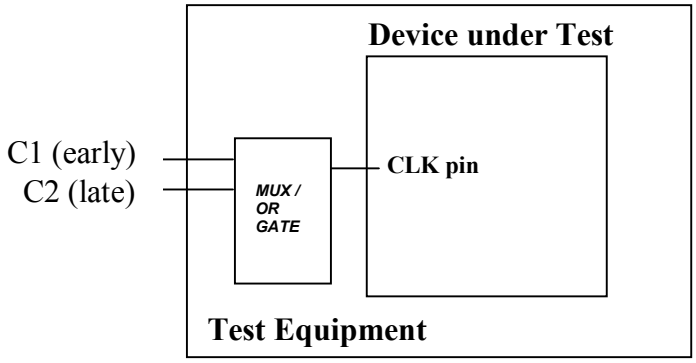



**Figure 7: Tester that supports multiple channels with a Mux or OR gate.**

On some of the older test equipment being used, the two channels can be hooked to an OR gate, with the output of the OR gate feeding the device CLK pin. In designs with

clocks having an off-state of '0', this can work correctly. One channel, say C2 is held at '0' while C1 drives the "early" waveform and then, C1 is held at '0' while C2 drives the late waveform. This may not work correctly if the CLK being driven has an off-state of '1'. In such cases, the test program has to be edited such that a '0' is driven on the unused clock input of the OR gate during the required cycles.

## 3.2 Implications of false and multi-cycle paths

Typically, designs contain paths in the circuitry that do not operate at the full functional frequency of the device. This can be attributed to slower logic, false paths or multi-cycle paths in the design. These design features can make at-speed testing challenging. Solutions to these scenarios are presented below.

### 3.2.1 False paths

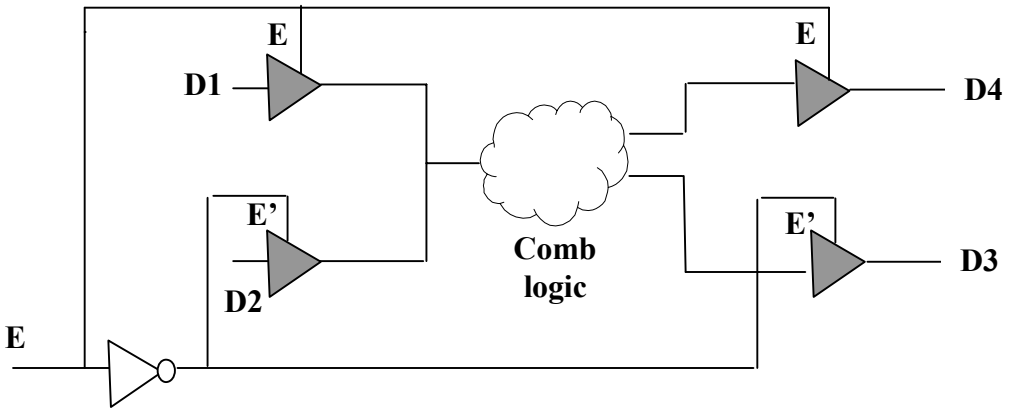



**Figure 8: Illustration of false paths**

Figure 8 shows the illustration of false paths. There are four tri-state devices in the above circuit, two of them enabled by E and two enabled by E'. Due to the presence of the inverter, E and E' are always mutually exclusive. As a result, the paths D1 – D3 and D2-D4, along with the combinational logic in-between constitute false paths.

Such false paths can pose problems while performing at-speed ATPG. The solution is to mask all the flip-flops that are present at the destinations of such false paths. By doing so, we will not be *observing* any values at the outputs of such paths, thus avoiding testing

such false paths. However, the disadvantage is that such masked flip-flops may also constitute the destination of other non-false paths. We may lose some amount of coverage due to flip-flop masking. A better approach may be to *mask* entire paths themselves, but this is not possible during ATPG. As a result, the feature provided by commercial ATPG tools to mask individual flip-flops was used.

### 3.2.2 Multi-cycle paths

Figure 9 illustrates a circuit segment where multi-cycle paths exist. It can be seen that the flip-flops (or registers) in Counter B are enabled only at a rate that is $1/8^{th}$ of the overall clock rate. This is because, the "Clock-enable" line in Counter B is turned on only when Counter A finishes counting from 0 through 7. Once the carry-bit of Counter A is set, Counter B's clock is enabled. The combinational paths in Counter B are now multi-cycle paths.

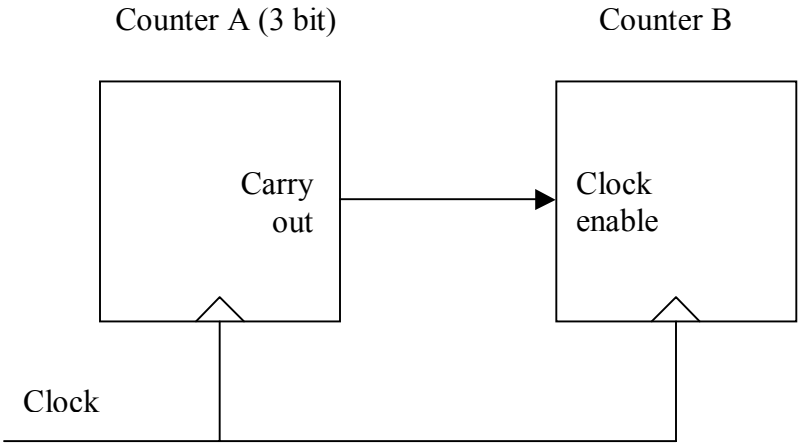



**Figure 9: Illustration of multi-cycle paths**

For such designs, at-speed testing can again be a challenge. The solution we use comprises of identifying such paths and masking all destination flip-flops of such paths. ATPG tools provide the capability to mask the outputs of some flip-flops. When this is done, no faults are observable at these flops. The tool will then try to generate patterns that are observable at other flip-flops or primary outputs. The source flip-flops of such paths are also constrained to Xs. By doing so, the outputs of such source flip-flops are

always set to X during ATPG. Then, ATPG is performed at full frequency. The next step is to remove the X constraints and output masks and perform ATPG at $1/n^{th}$ the frequency, where "n" is the length of each multi–cycle path.

Now that techniques exist to test designs with false and multi-cycle paths, how are such paths identified? One efficient approach is to use the reports generated by static timing analysis (STA). STA reports contain lists of such false and multi-cycle paths and this information was used. An additional problem is that STA reports may not contain an exhaustive listing of all false paths, as this is the same as the ATPG problem, which is N-P -complete.

## 3.3 Issues with clock-domains

Most of the designs in the industry today have more than one clock domain. Separate device pins can be used to clock such different domains or a common device clock can also be used to clock all domains simultaneously.

If the tester period is made equal to the slowest of all clocks and if all clocks are pulsed in this period sequentially, the fault coverage can be improved. However, this requires a lot of change, as sequential ATPG has to be supported. To minimize changes to our setup, we generate tests for one clock domain at a time. One clock pin is used for launch and capture while the scan-in and scan-out operations are done using all clocks.

The procedure is then to scan-in data using all the clocks, use one clock pin to fire the launch and capture pulses and finally scan out using all the clocks. When ATPG for one clock domain is complete (coverage peaks), the entire process is repeated for a different device clock. The disadvantage is that any "cross-interacting logic" remains untested in this case [15].

If one common clock is being used to test designs, all flip-flops in domains not being tested have to be masked. This places a requirement that the clock tree for this test clock

must be able to run at the fastest frequency being tested. Further, all the flip-flops in other domains need to be identified beforehand, so that they can be masked accordingly.

In multiple clock domain designs, it may so happen that flip-flops triggered by different clock domains are on the same scan-chain. In such a case, it becomes imperative to place LOCKUP latches at the boundaries of these clock domains, to eliminate clock skews between domains while shifting through the scan chain. Such lockup latches delay the data for half a clock cycle, from the rising to the falling edge, thus providing a high tolerance to skew between domains. This requirement is especially necessary for transition fault tests, though desirable for stuck-at tests [15].

In some designs, designers do not use LOCKUP latches, but intentionally skew the device clocks to make up for the use of LOCKUP latches. This technique works while testing for stuck-at faults using low frequencies. However, this method will most often not work for at-speed testing, as it is almost impossible to simultaneously shift the scan chain, perform launch and at-speed capture in the same test pattern.

As a result, the only alternative is to use LOCKUP latches in the scan chain at domain crossings. Additionally, it is imperative to place the trailing edge-triggered flip-flops prior to the leading edge-triggered flip-flops in the scan chain.

Some designs may consist of both leading-edge and falling-edge triggered flip-flops. It will be a lot easier if only one edge in the test mode. If both the edges are being used in test mode, care has to be taken while performing simulation to update values at both edges of the clock. If this is not possible, all the falling-edge flip-flops are masked and set to X during ATPG.

The effects of clock domains on test coverage were examined on a range of designs and these results will be presented in Section 4.2.

## 3.4 Low Cost Test

Semiconductor companies have been making use of low cost testers to offset ATE costs [21] – [24]. Accuracy and speed requirements are reduced and this contributes to the low cost. Texas Instruments has developed such a low cost tester, that is available in a 512 pin configuration, with a maximum functional operating frequency of 30 MHz. Typically, this 30 MHz channel is used for scan shifting, while there is another single channel available that is capable of clock bursts up to 500 MHz. This feature makes it possible to use these low cost testers in testing at-speed faults. Flexibility has been provided to map this high-speed channel onto any of the device pins as needed. All other device pins are dependent on relatively low-speed tester channels.

In these low cost testers, only the "launch-from-capture" technique can be used, as there is only a single pin available for high-speed operation. The "launch-from-shift" method warrants the need for an at-speed scan enable in addition to the need for a pin driving the clock at functional frequency. Since this requirement cannot be met, the "launch-from-shift" method is incompatible with TI's current generation low cost test equipment. However, work is currently in progress to address this issue.

While testing for transition faults, some tools may generate patterns that require at-speed changes in the primary inputs. Further, they may also produce patterns that require at-speed strobing at the primary outputs. Since there are no other at-speed channels available on low cost testers, such patterns cannot be supported. ATPG tools support two important commands: "Hold PI" and "Mask Outputs" options. When the "Hold PI" option is turned on, the commercial tools generate patterns such that: for any pair ($v_1$, $v_2$) of transition fault vectors, the PI values are held constant across the application of $v_1$ and $v_2$. When the "Mask Outputs" option is turned on, the ATPG tool will not generate patterns in which faults are not detected at the POs. Instead, the effects of all detected faults are propagated to a flip-flop on the scan chain. The disadvantage of these two commands is that all PI-to-scan-flip-flop and scan-flip-flop-to-PO faults remain undetected. The extent of this disadvantage was analyzed on a range of designs and the

results are presented in Section 4.3. It was observed that the effect is not great, if pin values are registered close to the chip boundary.

# Chapter 4

# Implementation and Results

Multiple industrial designs were selected for experimentation purposes, with many parameters in mind. The gate count of chosen designs ranged from 760K to 4.17M. The functional clock frequencies varied from 155 MHz to 300 MHz and the number of clock pins across designs varied from as low as 2 to as high as 9. Scan shift frequencies varied from 10 MHz to 30 MHz. Designs with both low and high pin counts were chosen. The number of input pins varied from 66 to 329, while the number of output pins varied from 39 to 305. The total number of scan-cells varied from 21K to 300K.

All the ATPG runs were performed on a compute ranch comprised of UltraSPARC III $^{TM}$ machines and as a result, the configuration and load of individual machines varied from one to another. The typical configuration of each machine consisted of 8 processors running at 900 MHz, with a total of 32 banks of 512MB interleaved memory. However, resources were being shared with other jobs at the time of execution.

Two commercial ATPG tools were used for generating test patterns on all the designs. The emphasis of the tools was on different aspects. Tool 1 placed more emphasis on pattern volume and execution time while Tool 2 was geared more towards attaining the maximum coverage possible. From the results obtained, it was noted that Tool 2 is able to attain a slightly higher coverage on most of the designs, at the cost of pattern volume and execution time. Tool 1 on the other hand seemed to help in gaining coverage close to that obtained by Tool 2, but with significantly lesser number of patterns and lesser execution time.

The main focus of the exercise was to compare the two techniques, launch-from-capture and launch-from-shift from multiple viewpoints. Both the commercial tools had the capability to support commands relevant to each approach and they are summarized below:

**Launch-from-capture technique**

- Set the fault type to transition faults and add all faults.
- Set the tool to perform sequential ATPG, using two pulses of the capture clock.
- Turn off the combinational ATPG algorithm.
- Since the scan-enable signal need not switch at-speed for the launch-from-capture technique, the scan-enable pin has to be constrained to a 0 during the two capture cycles.
- In between two adjacent scan operations, some tester platforms may require that the primary inputs be held constant. This is the case if the tester platform does not support at-speed operation of all the primary inputs.
- All tester platforms may not support at-speed operations of the primary output pins. In such cases, the output pins have to be masked during ATPG. The only observability available would be via the circuit response data captured into the scan flip-flops and these become visible during the scan-out operation.

**Launch-from-shift technique**

- Set the fault type to transition faults and add all faults.
- Set the tool to perform pure combinational ATPG, with a moderate abort limit
- A prerequisite for this method is the availability of an at-speed scan-enable pin. So, no constraint needs to be placed on the scan-enable pin.
- Depending on whether the tester supports at-speed primary inputs and at-speed strobing of the primary outputs, constraints have to be placed on the primary input and output pins.

## 4.1 Launch-from-capture vs. launch-from-shift

The first set of experiments was geared towards comparing three parameters: test coverage, pattern volume and tool execution time with respect to both launch-from-capture and launch-from-shift techniques. The results obtained from Tool 1 have been summarized in Table 2. All the designs chosen were semi-custom ASIC designs, with

scan capability already inserted. The same designs were then run through Tool 2 and these results have been documented in Table 3.

| ID | Design size (total gates) | Scan cells {scan chains} | *Tool 1* | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | *Launch-from-capture* | | | *Launch-from-shift* | | | *Comparison* | | |
| | | | *TC (%)* | *Num of pats (pre/post compression)* | *ATPG time (hrs)* | *TC (%)* | *Num of pats (pre/post compression)* | *ATPG time (hrs)* | *Diff TC* | *Diff Pat. Vol* |
| A | 0.76 M | 21,238 {8} | 62.44 | 2569/2514 | 31.76 | 89.07 | 1226/1224 | 4.13 | 1.43x | 0.48x |
| B | 0.86 M | 60,669 {32} | 75.47 | 16770/16744 | 8.47 | 80.98 | 7094/7086 | 8.10 | 1.07x | 0.42x |
| C | 1.07 M | 71,113 {8} | 74.37 | 4472/4468 | 9.16 | 79.97 | 1231/1231 | 12.34 | 1.07x | 0.27x |
| D | 1.36 M | 74,673 {8} | 81.44 | 23509/23227 | 43.01 | 89.78 | 7787/7781 | 22.17 | 1.10x | 0.33x |
| E | 1.41 M | 53,895 {7} | 79.39 | 10304/10303 | 54.39 | 84.96 | 3843/3843 | 23.16 | 1.07x | 0.37x |
| F | 4.17 M | 300,974 {165} | 88.89 | 23635/23541 | 45.67 | 89.12 | 2020/2019 | 63.79 | 1.00x | 0.08x |

**Table 2: Launch-from-capture vs. Launch-from-shift – PIs held constant, no PO strobing (Tool 1)**

| ID | Design size (total gates) | Scan cells {scan chains} | *Tool 2* | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | *Launch-from-capture* | | | *Launch-from-shift* | | | *Comparison* | |
| | | | *TC (%)* | *Num of pats (pre/post compression)* | *ATPG time (hrs)* | *TC (%)* | *Num of pats (pre/post compression)* | *ATPG time (hrs)* | *Diff TC* | *Diff Pat. Vol* |
| A | 0.76 M | 21,238 {8} | 64.26 | 9554/6951 | 5.20 | 90.56 | 9004/7329 | 16.76 | 1.41x | 1.05x |
| B | 0.86 M | 60,669 {32} | 75.53 | 74697/61031 | 18.70 | 81.66 | 26291/21864 | 21.38 | 1.08x | 0.36x |
| C | 1.07 M | 71,113 {8} | 77.73 | 102744/86387 | 40.75 | 79.62 | 41479/36660 | 30.84 | 1.02x | 0.42x |
| D | 1.36 M | 74,673 {8} | 89.55 | 193488/149247 | 287.34 | 99.67 | 59310/57220 | 148.32 | 1.11x | 0.38x |
| E | 1.41 M | 53,895 {7} | 75.84 | 94769/80425 | 58.42 | 86.49 | 46431/45826 | 1302.72 | 1.14x | 0.57x |
| F | 4.17 M | 300,974 {165} | 91.69 | 100457/80565 | 180.33 | 91.90 | 24263/17059 | 38.16 | 1.00x | 0.21x |

**Table 3: Launch-from-capture vs. Launch-from-shift – PIs held constant, no PO strobing (Tool 2)**

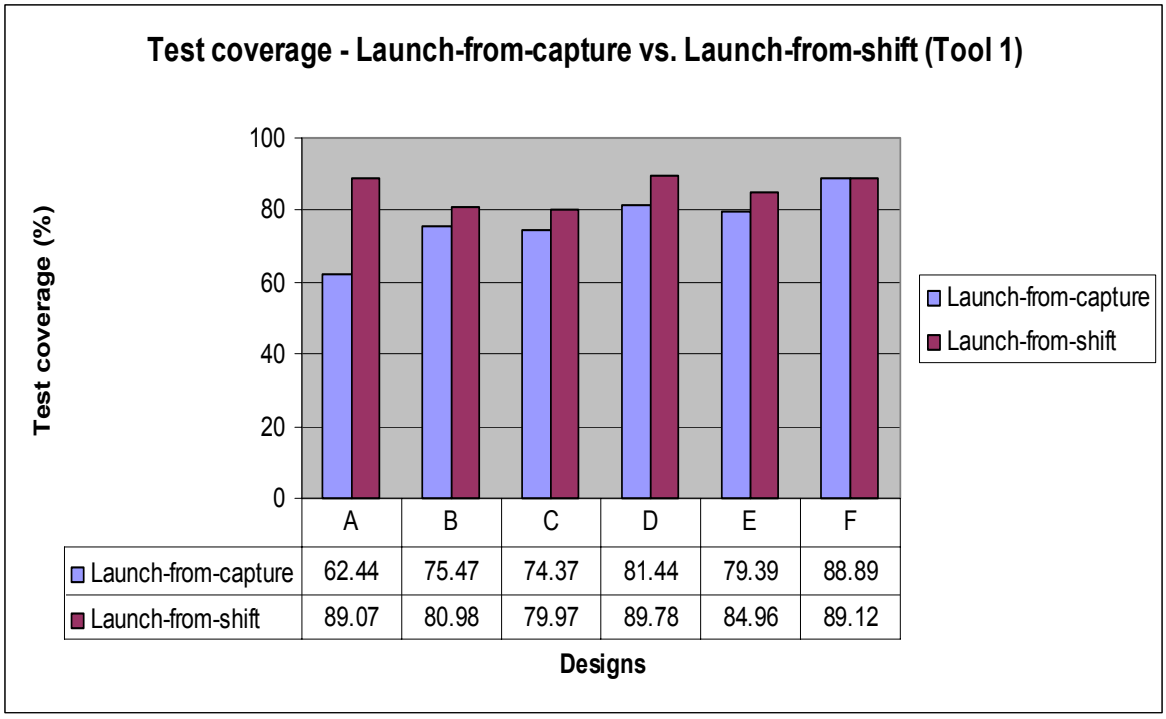
Total gate count, number of count of scan cells and number of scan chains in the design have been included in Tables 2 and 3. The designs were geared towards different applications and are thus not similar. The column marked TC gives the maximum test

coverage reached by each of the tools. Test coverage is a measure of the quality of the test set generated and is defined as the ratio of total number of faults detected by the tool to the total number of testable faults. ATPG untestable faults are not considered in this metric.

The number of patterns required to reach a given test coverage is included, along with the total time taken towards ATPG. The patterns generated by both tools were subjected to static compression post ATPG. The column showing time taken for ATPG does not include the time spent on static compression. Time taken is not very deterministic, as the machines executing these jobs were sharing resources with other unrelated jobs as well.

The final two columns indicate the magnitude of difference seen with respect to test coverage and pattern volume in the two approaches. Column marked 'Diff TC' is the magnitude of coverage by which launch-from-shift performs better than launch-from-capture approach. The column marked 'Diff Pat Vol.' indicates the extent by which pattern volume reduces by using the launch-from-shift approach.

The results clearly show that the launch-from-shift technique delivers higher test coverage as compared to the launch-from-capture technique. The maximum attainable coverage cannot be related to the design size. From Tables 2 and 3, we can see that Design F with 4.17 million gates reaches a much higher coverage than Design A with just 0.76 million gates. The launch-from-capture technique is predominantly a sequential ATPG algorithm, while launch-from-shift uses a pure combinational ATPG algorithm.




| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| Launch-from-capture | 62.44 | 75.47 | 74.37 | 81.44 | 79.39 | 88.89 |
| Launch-from-shift | 89.07 | 80.98 | 79.97 | 89.78 | 84.96 | 89.12 |

**Figure 10: Comparison of Test coverage (Tool 1)**

The difference in coverage is more pronounced in sections of the circuit where the sequential depth is not higher than 1. The coverage numbers reported here are the maximum attainable test coverages. No constraints were placed on pattern volume or coverage. The tools usually spend a majority of the time trying to generate patterns in the final 0.1 % to 0.5 %, before aborting ATPG. While using Tool 2 on design E, the ATPG process took 316 hours to reach test coverage of 81.47%, but the tool took significant CPU effort and time to reach the maximum of 86.49%.

Figure 10 shows a graphical representation of the comparison of test coverage across designs over both launch-from-capture and launch-from-shift techniques. Figure 11 shows the comparison of post-compression pattern volume and Figure 12 shows a comparison of ATPG times.

It has been shown that combinational pattern generation and compression techniques are superior to sequential pattern generation and compression techniques [17]. As a result,

the execution time of the launch-from-capture method is higher than the launch-from-shift method for most cases. In cases where the launch-from-shift method takes more time, it can be attributed to the significant improvement in coverage attained and comparative reduction in pattern volume. Overall, the launch-from-shift method performs significantly better than the launch-from-capture method, both in terms of coverage and pattern volume in most of the designs, irrespective of gate count, scan cell count and other parameters. However, as described in Section 3, the launch-from-shift method is not always practical and a number of requirements like the presence of an at-speed scan-enable and at-speed tester pins for the PIs and POs have to be addressed, if warranted by the tester.

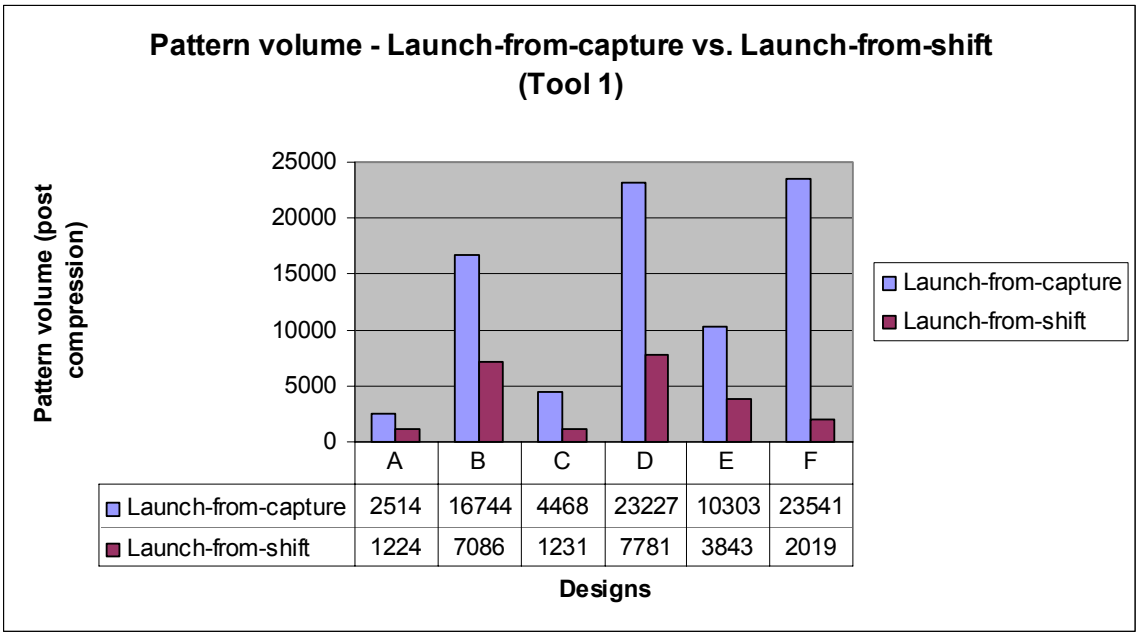



**Figure 11: Comparison of Pattern volume (Tool 1)**

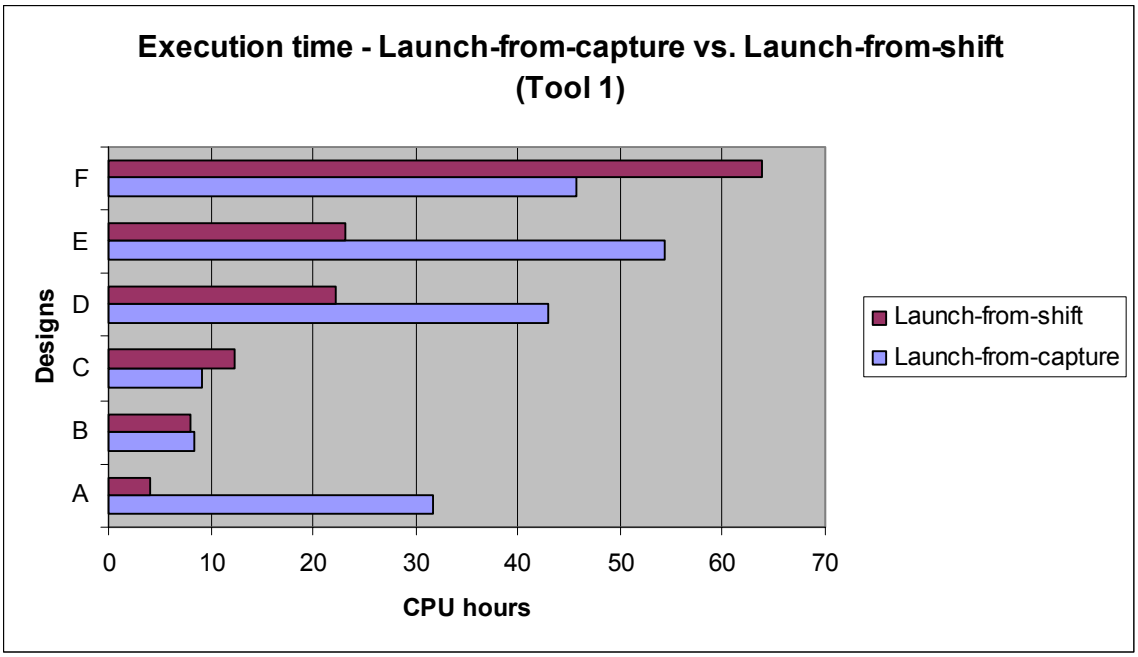



**Figure 12: Comparison of ATPG-time (Tool 1)**

## 4.2 Effect of multiple clock domains

The next set of experiments was to find out the implications of testing for transition faults in the presence of multiple clock domains. When multiple clock domains are present in a design, as is the case with most of the industrial designs, transition fault tests are usually generated for one clock domain at a time. The typical approach to this is to scan in using all clocks, use one device clock pin (at a time) to pulse both the launch and capture and then scan out using all clocks. In such cases, all logic "between" the two domains remains untested. If some clocks cover only a minor proportion of the flops, they can be left untested as well.

To gauge the effect of this approach, an experiment was conducted on some designs by combining all clocks together and the common clock was used to pulse all sequential elements. Instead of modifying the netlist, pin equivalences were placed on all clock pins. ATPG was conducted using one common clock and these results have been summarized

in Tables 4 and 5. Results from the designs with "unmodified clock domains" are also presented side-by-side for comparison.

| ID | Design size | CLK dom | | Launch-from-capture | | | |
|---|---|---|---|---|---|---|---|
| | | | | *N CLKs* | | *1 CLK* | |
| | | | | *TC (%)* | *Pats (pre/ post)* | *TC (%)* | *Pats (pre/ post)* |
| **A** | **0.76 M** | **2** | | 62.44 | 2569/ 2514 | **62.45** | **2551/ 2515** |
| **B** | **0.86 M** | **3** | | 75.47 | 16770/ 16744 | **75.68** | **16542/ 16533** |
| **C** | **1.07 M** | **3** | | 74.37 | 4472/ 4468 | **78.71** | **2691/ 2691** |
| **G** | **3.11 M** | **2** | | 72.24 | 32337/ 31670 | **72.28** | **27086/ 26690** |
| **F** | **4.17 M** | **9** | | 88.89 | 23635/ 23541 | **89.11** | **22356/ 22145** |

**Table 4: Effect of Multiple Clock Domains – Launch-from-capture (Tool 1)**

| ID | Design size | CLK dom | | Launch-from-shift | | | |
|---|---|---|---|---|---|---|---|
| | | | | *N CLKs* | | *1 CLK* | |
| | | | | *TC (%)* | *Pats (pre/ Post)* | *TC (%)* | *Pats (pre/ post)* |
| **A** | **0.76 M** | **2** | | 89.07 | 1226/ 1224 | **89.07** | **1180/ 1180** |
| **B** | **0.86 M** | **3** | | 80.98 | 7094/ 7086 | **83.65** | **6849/ 6845** |
| **C** | **1.07 M** | **3** | | 79.97 | 1231/ 1231 | **79.97** | **931/ 931** |
| **G** | **3.11 M** | **2** | | 64.94 | 528/ 528 | **69.83** | **534/ 532** |
| **F** | **4.17 M** | **9** | | 89.12 | 2020/ 2019 | **89.12** | **1909/ 1909** |

**Table 5: Effect of Multiple Clock Domains – Launch-from-shift (Tool 1)**

The results from Tables 4 and 5 show that the coverage is improved by pulsing all flip-flops with a common clock. In some designs (A, B, G), the improvement is marginal, while for other designs (C, F) the improvement is substantial for the launch-from-capture method. While performing ATPG on one clock domain at a time, constraints are placed

on the other domains, thus potentially affecting coverage. Some improvement can be seen in this "common clock" experiment, as such constraints are not applicable here. Another point to note is the reduced pattern volume. This is intuitive, as patterns need not be regenerated to test common logic, when an equivalent clock is being used.

Figures 13 and 14 show a graphical illustration on the effects of multiple clock domains in designs.

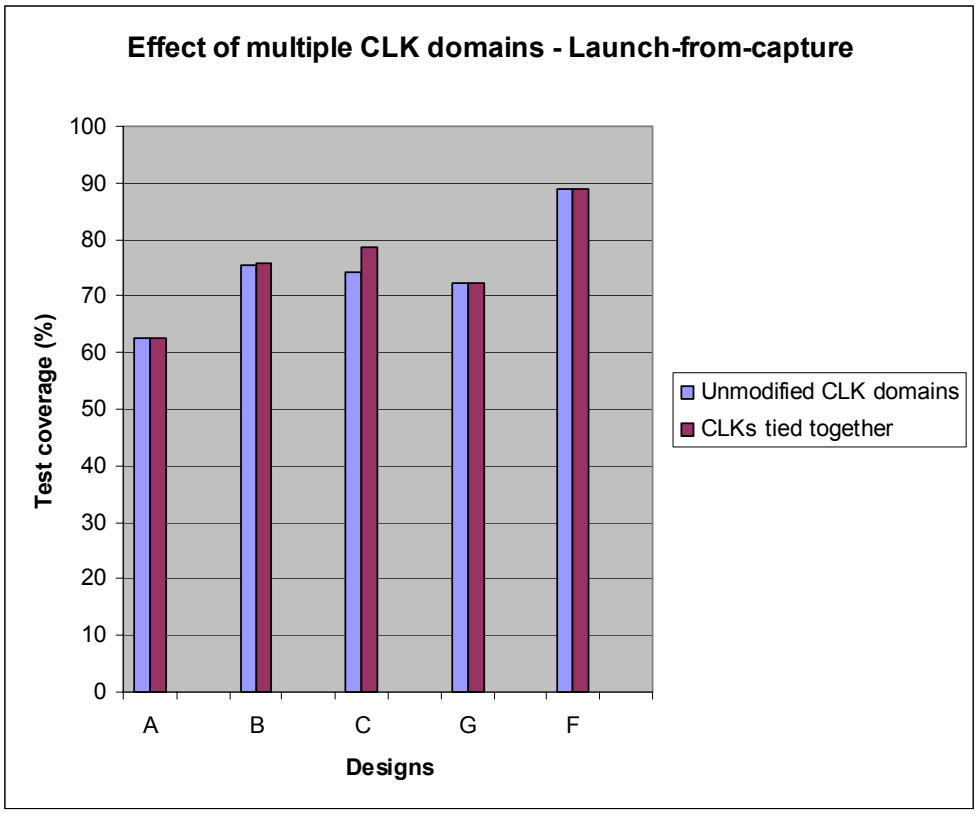



**Figure 13: Illustration on effect of multiple clock domains (launch-from-capture)**

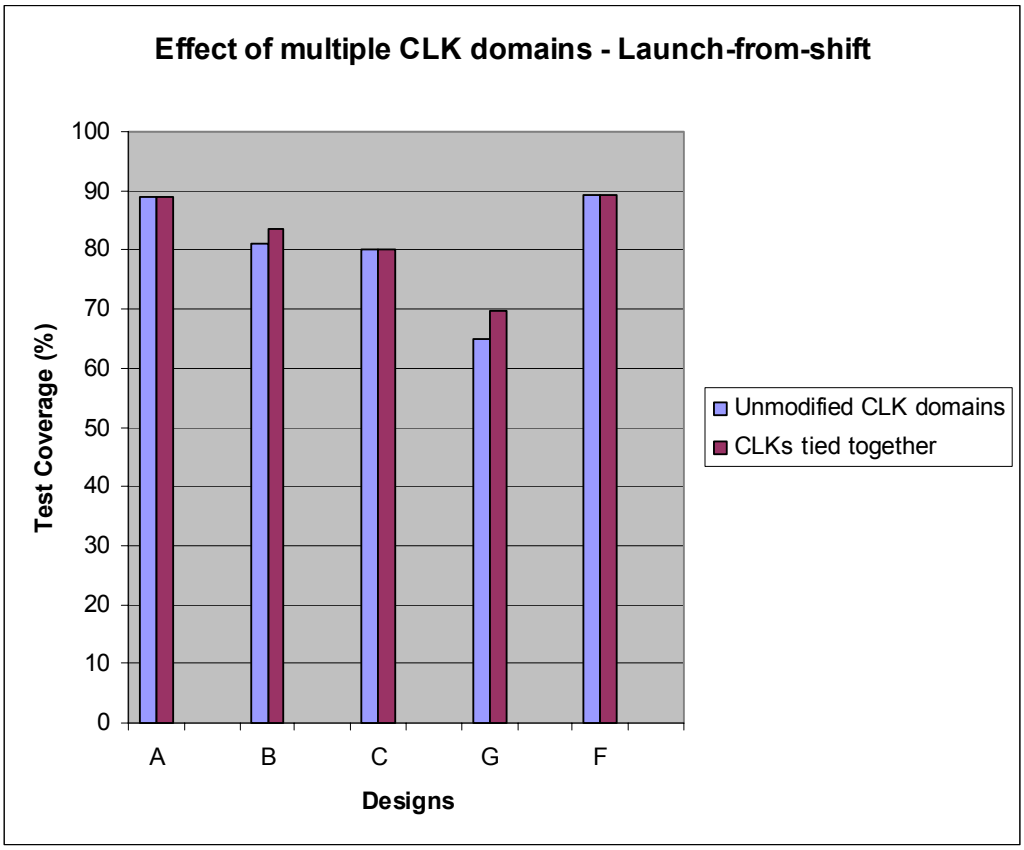



**Figure 14: Illustration on effect of multiple clock domains (launch-from-shift)**

## 4.3 Effect of constraining PIs and masking POs

The next step was to analyze the effect of supporting at-speed PI or PO operations. Supporting fast PI changes or PO strobes places additional and complex requirements on the tester and such requirements may be hard to meet. Even if met, the changes required can make the ATEs more expensive, which is definitely not desirable. From Tables 6 and 7, we can see that the coverage improvement depends on the total PI/PO count. The improvement is negligible in cases like designs A, B and C where the pin counts are relatively low. In designs like D and G, the improvement is significant. The main reason for such change in coverage can be attributed to the amount of logic between PIs and the nearest scannable flip-flop and the logic between end flip-flops and POs. Considering the complexity and cost of the test equipment, it is feasible to use testers that do not support at-speed PI and PO operations.

| ID | Design size | PIs | POs | | Launch-from-capture | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | *With PI/PO constraints* | | *No PI/PO constraints* | |
| | | | | | *TC (%)* | *Pats (pre/ post)* | *TC (%)* | *Pats (pre/ post)* |
| **A** | **0.76 M** | **66** | **39** | | *62.44* | *2569/ 2514* | **62.45** | **2556/ 2516** |
| **B** | **0.86 M** | **160** | **97** | | *75.47* | *16770/ 16744* | **75.62** | **16713/ 16682** |
| **C** | **1.07 M** | **127** | **101** | | *74.37* | *4472/ 4468* | **74.41** | **2803/ 2803** |
| **D** | **1.36 M** | **307** | **260** | | *81.44* | *23509/ 23227* | **86.15** | **60893/ 55484** |
| **E** | **1.41 M** | **160** | **143** | | *79.39* | *10304/ 10303* | **79.46** | **10400/ 10397** |
| **G** | **3.11 M** | **329** | **305** | | *72.24* | *32337/ 31670* | **77.36** | **21012/ 20706** |

**Table 6: Effect of Presence of At-speed PIs/POs – Launch-from capture technique (Tool 1)**

| ID | Design size | PIs | POs | | Launch-from-shift | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | *With PI/PO constraints* | | *No PI/PO constraints* | |
| | | | | | *TC (%)* | *Pats (pre/ post)* | *TC (%)* | *Pats (pre/ Post)* |
| **A** | **0.76 M** | **66** | **39** | | 89.07 | 1226/ 1224 | **89.07** | **1231/ 1230** |
| **B** | **0.86 M** | **160** | **97** | | 80.98 | 7094/ 7086 | **81.09** | **7111/ 7111** |
| **C** | **1.07 M** | **127** | **101** | | 79.97 | 1231/ 1231 | **79.98** | **1235/ 1235** |
| **D** | **1.36 M** | **307** | **260** | | 89.78 | 7787/ 7781 | **89.79** | **7824/ 7817** |
| **E** | **1.41 M** | **160** | **143** | | 84.96 | 3843/ 3843 | **84.99** | **3825/ 3825** |
| **G** | **3.11 M** | **329** | **305** | | 64.94 | 528/ 528 | **65.45** | **682/ 660** |

**Table 7: Effect of Presence of At-speed PIs/POs – Launch-from shift technique (Tool 1)**

Table 8 further attempts to illustrate the effect of PI-PO pins on the difference in coverage. As can be seen, the difference in coverage is almost related to the number of PI-PO pins. However, the major differentiating factor is the distance between these boundary pins and the nearest scannable flip-flops.

| Design | Total (PI + PO) | Coverage difference Launch-from-capture (%) | Coverage difference Launch-from-shift (%) |
|---|---|---|---|
| A | 105 | 0.01 | 0.00 |
| C | 228 | 0.04 | 0.01 |
| B | 257 | 0.15 | 0.11 |
| E | 303 | 0.07 | 0.03 |
| D | 567 | 4.71 | 0.01 |
| G | 634 | 5.12 | 0.51 |

**Table 8: Comparison of total PI-PO pins vs. change in coverage**

## 4.4 Disadvantages of the "launch-from-capture" technique

Though it is one of the more feasible approaches in practical terms, the launch-from-capture technique suffers from some inherent disadvantages. These same disadvantages can be attributed to the reason why the launch-from-shift method performs better than the launch-from-capture method. The main problem is its inability to generate tests in sections of designs where the sequential depth is not higher than 1 (single section networks). Logic lying between PIs and scan-flops can also be considered as single section networks. Simulation of patterns produced by the launch-from-capture technique is a two time-frame event. As a result, pattern generation takes more time as compared to the launch-from-shift method. Finally, it has been shown that the launch-from-shift method guarantees transition fault coverage of at least 50% in full-scan designs. No such metric exists for the launch-from-capture technique as it inherently depends on the individual designs [3].

From a different perspective, it can be noted that in a pair of transition fault vectors ($v_1$, $v_2$), the state reached by vector $v_2$ is reachable from the state obtained by $v_1$. As a result,

during the launch-from-capture technique, additional constraints are enforced by the ATPG tools, thus accounting for a lower coverage. This state constraint is not present in the launch-from-shift technique, as can be seen from the higher coverage obtained. An interesting observation is that the two consecutive states generated by launch-from-shift vectors are not actually reachable in functional mode at all. As a result, some of these additionally detected faults may be functionally untestable and a higher coverage may actually translate to yield loss.

# Chapter 5

# Dealing with Pattern Volume in Manufacturing Test

During manufacturing test, pattern volume plays a significant role as test application time is directly proportional to the number of patterns to be applied. As pattern volume increases, it becomes more and more difficult to store the patterns in the tester memory. Different test platforms have different tester memory capacities and it is definitely beneficial to have the minimum number of patterns.

When testing for multiple fault models, as is often done in the industry, the target fault coverage for each model has to be decided. It is common knowledge that a large number of patterns are required to detect the last 1%-2% of the faults in any model. Stuck-at fault coverage is always higher than transition fault coverage. While generating patterns using ATPG, designers need to decide on when to abort pattern generation. This decision is made on a variety of factors like target fault coverage, tester memory constraints, number of parts to be manufactured, test application time allotted for each part, etc.

A common debate in manufacturing test is to decide on which types of patterns go first on the tester. In reality, the transition pattern volume is at least 4x-5x times stuck-at pattern volume. A number of questions and arguments now come to mind.

- Is it better to execute the stuck-at patterns first, catch any possible defects, discard those parts and then execute transition fault patterns? If this is not done, would it be wasteful to test for all transition fault patterns first, which constitute a high pattern volume. Assuming we do so and they all pass, then what if we test for stuck-at faults and failures are detected? Will this not waste valuable test time?

- However, it should be borne in mind that the transition-fault patterns can also detect stuck-at faults. After all, the method to test for transition faults also involves excitation and propagation, which is exactly what is needed for testing

stuck-at faults. So, if we test transition faults first, what percentage of stuck-at faults will be tested? Will all stuck-at faults be detected? If not, how many additional patterns will be required to detect the remaining stuck-at faults? In other words, what is the *top-off pattern volume*?

Both the arguments above are perfectly valid points and there is no way to declare one method better, if we do not perform the corresponding experiments.

In this chapter, we present the implementation details and results from the experiments we conducted. Results are presented and the pros and cons of each method are discussed.

## 5.1 Implementation details

To analyze the implications of pattern volume from different fault models, the following main steps were done:

**Step 1**: Generate patterns for stuck-at faults alone. Note the coverage reached and the number of patterns required. This will be the maximum stuck-at fault coverage attainable by stuck-at ATPG alone.

**Step 2**: Generate patterns for transition faults alone. The launch-from-capture method was used for this purpose. Save patterns generated externally. Use the same patterns and fault-simulate stuck-at faults against these transition patterns. Note the coverage reached and pattern volume. This will be the maximum stuck-at coverage attainable using pure transition fault patterns alone.

**Step 3**: Identify the untested stuck-at faults from Step 2. Execute pure stuck-at fault ATPG for these faults alone. The patterns generated here will constitute the *top-off patterns*. Note the pattern volume again.

**Calculations:**

Savings in pattern volume can be calculated from the following equation:

```
# Savings = (N1 + N2) – (N2 + N3) = (N1 – N3)
```

where:

# Savings: Savings in pattern volume by using transition fault patterns for stuck-at ATPG (for the same/better fault coverage)

N1: Number of patterns required for stuck-at ATPG (from Step 1)
N2: Number of transition fault patterns (from Step 2)
N3: Number of top-off patterns (from Step 3)

Then algorithm for Step 1 is straightforward, as it just involves stuck-at ATPG. It consists of the following steps:

1. Set fault type to stuck-at faults.
2. Add all faults.
3. Run stuck-at ATPG.
4. Compress patterns.
5. Report statistics and collect coverage and pattern volume information.

**Figure 15: Algorithm for Step 1**

The algorithm for Step 2 consists of the following steps:

1. Set fault type to transition faults.
2. Add all faults.
3. Run transition fault ATPG.
4. Save all patterns externally.
5. Reset state, so that all faults and patterns are deleted from the tool's internal memory.
6. Reset all ATPG untestable faults (Otherwise, the tool may not take these faults into consideration during fault-simulation).
7. Set the fault type to stuck-at faults.
8. Add all faults.
9. Read in the patterns saved in Step 4 from the external source.

```
10. Fault-simulate these patterns against the stuck-at
    faults added in Step 8.
11. Compress patterns.
12. Report statistics and collect coverage and pattern
    volume information.
```

**Figure 16: Algorithm for Step 2**

Step 3 is a logical continuation of Step 2 and consists of the following steps.

```
13. Report all uncontrollable (UC) or unobservable (UO)
    faults and save them externally.
14. Reset state to eliminate all faults and patterns
    from the tool's internal memory.
15. Reset all ATPG untestable faults (Otherwise, the
    tool may not take these faults into consideration
    during ATPG).
16. Set the fault-type to stuck-at faults.
17. Read in the fault-list which was saved in Step 13.
18. Invoke stuck-at ATPG.
19. Compress patterns.
20. Report statistics and collect coverage and pattern
    volume information.
```

**Figure 17: Algorithm for Step 3 (numbering of steps retained to show continuity)**

## 5.2 Results

| Transition fault ATPG only | | Stuck-at fault sim using transition fault patterns | | Top-off stuck-at patterns for remaining faults | | Pure stuck-at fault patterns | |
|---|---|---|---|---|---|---|---|
| Num of patterns | Test Coverage | Num of patterns | Test Coverage | Num of patterns | Test Coverage | Num of patterns | Test Coverage |
| 0 | 0.00 | 0 | 0.00 | 0 | 0.00 | 0 | 0.00 |
| 64 | 49.50 | 64 | 79.32 | 64 | 60.40 | 64 | 85.07 |
| 128 | 55.02 | 128 | 84.68 | 128 | 72.35 | 128 | 89.83 |
| 192 | 57.32 | 192 | 86.89 | 192 | 82.81 | 192 | 91.51 |
| 256 | 58.64 | 256 | 88.13 | 256 | 84.62 | 256 | 92.47 |
| 320 | 59.47 | 320 | 89.18 | | | 320 | 92.93 |
| 384 | 60.02 | 384 | 89.84 | | | 384 | 93.25 |
| 448 | 60.43 | 448 | 90.17 | | | 448 | 93.4 |
| 512 | 60.75 | 512 | 90.72 | | | 512 | 93.47 |
| 576 | 61.00 | 576 | 91.05 | | | 576 | 93.5 |
| 640 | 61.19 | 640 | 91.28 | | | 640 | 93.52 |
| 704 | 61.37 | 704 | 91.42 | | | 704 | 93.52 |
| 768 | 61.51 | 768 | 91.57 | | | | |
| 832 | 61.66 | 832 | 91.66 | | | | |
| 896 | 61.77 | 896 | 91.75 | | | | |
| 960 | 61.87 | 960 | 91.83 | | | | |
| 1024 | 61.94 | 1024 | 91.9 | | | | |
| 1088 | 62.01 | 1088 | 91.96 | | | | |
| 1152 | 62.06 | 1152 | 92.05 | | | | |
| 1216 | 62.12 | 1216 | 92.11 | | | | |
| 1280 | 62.16 | 1280 | 92.16 | | | | |
| 1344 | 62.2 | 1344 | 92.2 | | | | |
| 1408 | 62.23 | 1408 | 92.23 | | | | |
| 1472 | 62.26 | 1472 | 92.27 | | | | |
| 1536 | 62.28 | 1536 | 92.31 | | | | |
| 1600 | 62.3 | 1600 | 92.34 | | | | |
| 1664 | 62.31 | 1664 | 92.37 | | | | |
| 1728 | 62.33 | 1728 | 92.39 | | | | |
| 1792 | 62.34 | 1792 | 92.41 | | | | |
| 1856 | 62.36 | 1856 | 92.44 | | | | |
| 1920 | 62.37 | 1920 | 92.47 | | | | |
| 1984 | 62.38 | 1984 | 92.49 | | | | |
| 2048 | 62.39 | 2048 | 92.51 | | | | |
| 2112 | 62.4 | 2112 | 92.53 | | | | |
| 2240 | 62.41 | 2240 | 92.56 | | | | |
| 2304 | 62.42 | 2304 | 92.57 | | | | |
| 2368 | 62.43 | 2368 | 92.59 | | | | |
| 2496 | 62.44 | 2496 | 92.62 | | | | |
| 2560 | 62.44 | 2521 | 92.63 | | | | |

**Table 9: Design X - Tradeoff between Transition-fault and Stuck-at fault Patterns**

| Design X | | Stuck-at fault simulation using transition patterns | | Top-off stuck-at patterns | | Pure Stuck-at patterns | |
|---|---|---|---|---|---|---|---|
| **Fault Class** | **Description** | **#faults collapsed** | **#faults total** | **#faults collapsed** | **#faults total** | **#faults collapsed** | **#faults total** |
| **FU** | (full) | 1728461 | 2860898 | 56547 | 99255 | 1728461 | 2860898 |
| | | | | | | | |
| **UC** | (uncontrolled) | 2589 | 6930 | 137 | 408 | 124 | 337 |
| **UO** | (unobserved) | 53958 | 92325 | 2636 | 4987 | 3029 | 5947 |
| **DS** | (det_simulation) | 1455964 | 2497702 | 40651 | 72695 | 1462029 | 2510708 |
| **DI** | (det_implication) | 104684 | 132416 | | | 104684 | 132416 |
| **PT** | (posdet_testable) | 8197 | 8627 | | | | |
| **UU** | (unused) | 16989 | 17696 | | | 16989 | 17696 |
| **TI** | (tied) | 1682 | 3037 | | | 1682 | 3037 |
| **BL** | (blocked) | 581 | 641 | | | 581 | 641 |
| **AU** | (atpg_untestable) | 83817 | 101524 | 5063 | 7816 | 131167 | 176577 |
| | | | | | | | |
| **PU** | (posdet_untestable) | | | 2 | 2 | 86 | 104 |
| **RE** | (redundant) | | | 8058 | 13347 | 8090 | 13435 |
| | | | | | | | |
| | | | | | | | |
| **TC** | Test coverage | 91.31% | 92.63% | 83.84% | 84.62% | 92.10% | 93.53% |
| **FC** | Fault coverage | 90.29% | 91.93% | 71.89% | 73.24% | 90.64% | 92.39% |
| **Eff** | ATPG effectiveness | 96.25% | 96.23% | 95.10% | 94.56% | 99.82% | 99.78% |
| | | | | | | | |
| **#Pats** | Number of patterns | | 2282 | | 230 | | 672 |

**Table 10: Design X – Statistics report**

Two industrial ASIC designs were considered and the above experiments conducted. A commercial ATPG tool was used to run the above experiments. The tool supports all the relevant commands and the summary reported by the tools is used for analysis.

The designs chosen had two different features. Design X's transition fault coverage was very low, while that of Design Y was high. Design X's transition fault coverage topped off at 62.44%, while design Y's coverage went up to 91.53%. The maximum stuck-at fault coverage attainable was similar in both designs. Design X's stuck-at coverage topped off at 93.52%, while Design Y's coverage topped off at 95.23%. These two diverse designs were selected to see if any similarities could be drawn.

ATPG run-time statistics from Design X have been shown in Table 9. Detailed statistics for Design X are shown in Table 10.

From Table 9 and Table 10, we can make the following observations with respect to Design X.

- Transition fault coverage tops off at 62.44% and 2282 patterns are required.
- Pure stuck-at coverage tops off at 93.53% and 672 patterns are required.
- By fault-simulating transition fault patterns against stuck-at faults, stuck-at coverage of 92.63% could be reached.
- For the remaining stuck-at faults that went undetected by transition fault patterns, but are really detectable, 230 extra patterns are required.

The number of patterns reported in Table 9 and Table 10 may vary slightly as Table 9 displays the coverage and pattern volume while performing dynamic compaction. Table 10 displays the final values, with maximum possible coverage and minimum number of patterns required.

Statistics of individual fault classes are summarized in Table 10 and their definitions are given below:

FU: A consolidated number representing all the faults in the design.

UC: Faults that are uncontrollable and hence remain undetected. In other words, during ATPG, these points can never be excited to the value of the fault, given the necessary constrains.

UO: Faults that are unobservable and hence remain undetected. In other words, the fault effect cannot be propagated to an observable point, which can either be a primary output pin or a scan cell.

DS: Faults that are detected by simulation.

DI: Faults that are detected by implications.

PT: All faults that are identified as possible-detected, but not hard-detected. This is usually due to a 0-X or a 1-X difference at an observable point. The ATPG tool gives 50% credit for such possible-detected faults.

UU: Unused faults. These result from unconnected points in the design.

TI: Tied faults. This class includes faults on logic gates where the point of the fault is tied to a value, thus making the fault identical to a stuck-at fault.

BL: Blocked faults result when all the paths to an observable point are blocked due to a tied value along the path.

AU: ATPG Untestable faults - This class includes all faults for which the test generator is not able to generate a test pattern. However, these faults cannot be proven redundant by the tool. The most common reason for such faults being ATPG untestable is because of constraints placed on the design. If such constraints are removed, the fault class can change to PT or detectable, but the class cannot be altered by increasing the abort limit of the tool.

PU: These are potentially untestable faults. They are proven ATPG untestable and are also hard undetectable possible-detected faults.

RE: These are redundant faults, meaning faults that the test generation tool considers undetectable. A special analysis is performed by the tool post test generation to verify that these faults are truly undetectable under any conditions.

DT: Detected faults that include all DS and DI faults.

For all calculations, the testable fault class (#testable) includes the following:

- Detected faults - DT = (DS + DI)

- Post detected - PD = (PU + PT)
- ATPG Untestable - AU
- Undetected - UD = (UC + UO)

The untestable fault class (UT) includes the following:

- Unused faults - UU
- Tied faults – TI
- Blocked faults - BL
- Redundant faults - RE

Coverage calculations are made by the tool using the following equations:

```
Test coverage (TC) = {#DT + (#PD * pos_det credit)} / # testable

Fault coverage (FC) = {#DT + (#PD * pos_det credit)} / # full

ATPG effectiveness (Eff) = {#DT + #UT + #AU + #PU + (#PD *
                pos_det credit)} / #full

          where pos_det credit is usually 50%
```

From Table 10, the following observations can be made.

- Total number of faults in column 1 and column 3 are the same.

- The undetected faults from column 1 (stuck-at fault simulation on transition fault patterns) are carried over to column 2 (top-off stuck-at faults)
    - #UC from column 1 + #UO from column 1 = #FU in column 2

- After the UC and UO faults from column 1 are carried forward to column 2, top-off patterns are generated. After the stuck-at top-off ATPG, 408 UC and 4987 UO faults remain.

  Total (UC + UO) faults after top-off ATPG = 408 + 4987 = 5395

But, total (UC+UO) faults from pure stuck-at ATPG = (337 + 5947) = 6284

Thus, we can conclude that by using transition fault patterns for stuck-at ATPG and then topping off the stuck-at coverage, we can *detect* more stuck-at faults.

- #Pats from Column 2 < #Pats from Column 3
  [230 < 772]

- Savings in pattern volume = (672-230) = 442

This is very interesting because it proves that there *is* a benefit in using transition fault patterns for stuck–at faults. Not only is there a decrease in total pattern volume, but an increase in stuck-at coverage also.

Figure 18 shows the graph of Test coverage vs. Pattern volume for the stuck-at and transition fault models. It also illustrates the curve for stuck-at fault simulation with transition fault patterns. It can be seen that each of the fault models have their own "knee", the point where fault coverage stops increasing and remains almost the same. An important exercise in each test strategy is to identify this "knee", so that the test coverage can be constrained, thus limiting pattern generation time to a minimum and pattern volume optimal.

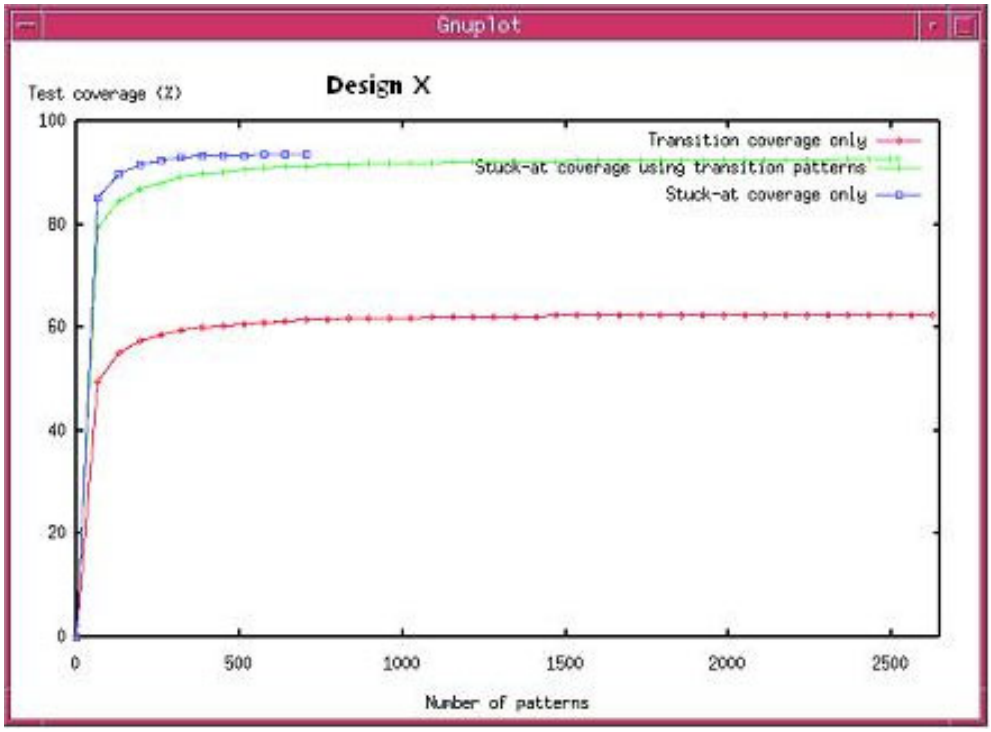



**Figure 18: Test coverage vs. Number of patterns (Design X)**

Once again, From Table 11 and Table 12, we can make the following observations with respect to Design Y.

- Transition fault coverage tops off at 91.53% and 1797 patterns are required.
- Pure stuck-at coverage tops off at 95.23% and 660 patterns are required.
- By fault-simulating transition fault patterns against stuck-at faults, stuck-at coverage of 93.69% could be reached.
- For the remaining stuck-at faults that went undetected by transition fault patterns, but are really detectable, 220 extra patterns are required.

| Transition fault ATPG only | | Stuck-at fault sim using transition fault patterns | | Top-off stuck-at patterns for remaining faults | | Pure stuck-at fault patterns | |
|---|---|---|---|---|---|---|---|
| Num of patterns | Test Coverage | Num of patterns | Test Coverage | Num of patterns | Test Coverage | Num of patterns | Test Coverage |
| 0 | 0.00 % | 0 | 0.00 % | 0 | 0.00 % | 0 | 0.00 % |
| 64 | 73.60% | 64 | 88.74% | 64 | 2.54% | 64 | 87.47% |
| 128 | 81.05% | 128 | 91.04% | 128 | 4.14% | 128 | 91.45% |
| 192 | 84.20% | 192 | 91.91% | 192 | 5.74% | 192 | 92.89% |
| 256 | 86.14% | 256 | 92.40% | 256 | 8.73% | 256 | 93.65% |
| 320 | 87.29% | 320 | 92.67% | 320 | 9.69% | 320 | 94.21% |
| 384 | 88.08% | 384 | 92.88% | | | 384 | 94.46% |
| 448 | 88.53% | 448 | 93.05% | | | 448 | 94.60% |
| 512 | 88.91% | 512 | 93.16% | | | 512 | 94.75% |
| 576 | 89.25% | 576 | 93.24% | | | 576 | 95.13% |
| 640 | 89.52% | 640 | 93.31% | | | 640 | 95.22% |
| 704 | 89.76% | 704 | 93.38% | | | 704 | 95.23% |
| 768 | 89.93% | 768 | 93.42% | | | | |
| 832 | 90.10% | 832 | 93.45% | | | | |
| 896 | 90.25% | 896 | 93.48% | | | | |
| 960 | 90.39% | 960 | 93.51% | | | | |
| 1024 | 90.50% | 1024 | 93.54% | | | | |
| 1088 | 90.58% | 1088 | 93.55% | | | | |
| 1152 | 90.67% | 1152 | 93.57% | | | | |
| 1216 | 90.74% | 1216 | 93.58% | | | | |
| 1280 | 90.81% | 1280 | 93.60% | | | | |
| 1344 | 90.88% | 1344 | 93.61% | | | | |
| 1408 | 90.94% | 1408 | 93.62% | | | | |
| 1472 | 91.01% | 1472 | 93.63% | | | | |
| 1536 | 91.05% | 1536 | 93.64% | | | | |
| 1600 | 91.10% | 1600 | 93.64% | | | | |
| 1664 | 91.14% | 1664 | 93.65% | | | | |
| 1728 | 91.18% | 1728 | 93.65% | | | | |
| 1792 | 91.22% | 1792 | 93.66% | | | | |
| 1856 | 91.25% | 1856 | 93.66% | | | | |
| 1920 | 91.29% | 1920 | 93.67% | | | | |
| 1984 | 91.32% | 1984 | 93.67% | | | | |
| 2048 | 91.36% | 2048 | 93.67% | | | | |
| 2112 | 91.38% | 2112 | 93.67% | | | | |
| 2176 | 91.41% | 2176 | 93.68% | | | | |
| 2240 | 91.43% | 2240 | 93.68% | | | | |
| 2304 | 91.45% | 2304 | 93.68% | | | | |
| 2368 | 91.47% | 2368 | 93.69% | | | | |
| 2432 | 91.48% | 2432 | 93.69% | | | | |
| 2496 | 91.49% | 2496 | 93.69% | | | | |
| 2560 | 91.50% | 2560 | 93.69% | | | | |
| 2624 | 91.51% | 2624 | 93.69% | | | | |
| 2688 | 91.52% | 2684 | 93.69% | | | | |
| 2752 | 91.53% | | | | | | |

**Table 11: Design Y - Tradeoff between Transition-fault and Stuck-at fault Patterns**

| Design Y | | Column 1 Stuck-at fault simulation using transition patterns | | Column 2 Top-off stuck-at patterns | | Column 3 Pure Stuck-at patterns | |
|---|---|---|---|---|---|---|---|
| **Fault Class** | **Description** | **#faults (collapsed)** | **#faults (total)** | **#faults (collapsed)** | **#faults (total)** | **#faults (collapsed)** | **#faults (total)** |
| FU | (full) | 1674453 | 2507628 | 34803 | 48745 | 1674453 | 2507628 |
| | | | | | | | |
| UC | (uncontrolled) | 22772 | 26768 | 125 | 289 | 146 | 391 |
| UO | (unobserved) | 12031 | 21977 | 1019 | 1744 | 1850 | 3969 |
| DS | (det_simulation) | 1387079 | 2179061 | 1577 | 3162 | 1406192 | 2201751 |
| DI | (det_implication) | 131082 | 138915 | | | 131082 | 138915 |
| PU | (posdet_untestable) | | | 2 | 4 | 2165 | 2182 |
| PT | (posdet_testable) | 33242 | 37261 | 1 | 1 | 5 | 9 |
| UU | (unused) | 29444 | 30692 | | | 29444 | 30692 |
| TI | (tied) | 1155 | 2320 | | | 1155 | 2320 |
| BL | (blocked) | 467 | 562 | | | 467 | 562 |
| RE | (redundant) | | | 8657 | 16115 | 8639 | 16094 |
| AU | (atpg_untestable) | 57181 | 70072 | 23422 | 27430 | 93308 | 110743 |
| | | | | | | | |
| TC | Test coverage | 92.38% | 93.69% | 6.03% | 9.69% | | 95.23% |
| FC | Fault coverage | 90.67% | 92.44% | 4.53% | 6.49% | | 93.34% |
| Eff | Atpg effectiveness | 95.94% | 96.57% | 96.71% | 95.83% | | 99.83% |
| | | | | | | | |
| #pats | #Number of patterns | | 1797 | | 220 | | 660 |

**Table 12: Design Y – Statistics report**

Similarly, from Table 12, the following observations can be made.

- Total number of faults in column 1 and column 3 are the same.

- The undetected faults from column 1 (stuck-at fault simulation on transition fault patterns) are carried over to column 2 (top-off stuck-at faults)

    - #UC from column 1 + #UO from column 1 = #FU in column 2
      [26768 + 21977 = 48745]

- After the UC and UO faults from column 1 are carried forward to column 2, top-off patterns are generated. After the stuck-at top-off ATPG, 289 UC and 1744 UO faults remain.

Total (UC + UO) faults after top-off ATPG = 289 + 1744 = 2033

But, total (UC+UO) faults from pure stuck-at ATPG = (391 + 3969) = 4360

Thus, we can reaffirm our conclusion that by using transition fault patterns for stuck-at ATPG and then topping off the stuck-at coverage, we can *detect* more stuck-at faults.

- #Pats from Column 2 < #Pats from Column 3
  [220 < 660]

- Savings in pattern volume = (660-220) = 440

From the above two experiments, we can conclude that if we use transition fault patterns for stuck-at ATPG and then generate the top-off patterns, we still can reach the level of stuck-at coverage reached by pure stuck-at ATPG alone or in some cases exceed it as well. The added benefit is the reduction in pattern volume as well.

However, the technique to be applied varies from design to design. We will have to take many factors like test application time, number of parts to be manufactured, stuck-at pattern volume, transition pattern volume, etc. into consideration before embarking on a test strategy.

# Chapter 6

# Switching Activity and Power Reduction Techniques

In the industry today, scan-based tests are the most popular method to test for stuck-at and delay faults. A typical scan-based test consists of the following steps.

- Scan-in a previously generated (using an ATPG tool) test pattern to all the memory elements in the design (scan cells). These are usually flip-flops or master-slave latches, with a scan-enable input. In functional mode, the D inputs are fed by the logic itself, and in scan-mode, the scan-in pins feed the flip-flops. In scan-mode, the scan chains are configured as a shift-register, thus forming one or more scan-chains.
- Apply primary input (PI) values at the input pins and measure outputs.
- Apply a capture pulse that causes the combinational circuit response to be *captured* into the sequential elements (scan cells).
- Shift out the values in the scan-chain for observation.

During the shifting phase, the magnitude of switching activity in the circuit is of interest. The switching activity is typically comprised of three variants.

1. Switching in the scan-chain itself.
2. Switching in the clock tree.
3. Switching in the combinational logic.

## 6.1 Problems with switching activity

Switching activity directly relates to power consumption and power is consumed both during shift and during capture. Depending on the size of the circuit and the frequency at which it operates, the power consumption during scan-testing can increase to levels that overload the power distribution systems and exceed package power limits [31]. As design

sizes and frequencies increase in current designs, power consumption is becoming a major problem, this making designing of clock trees more complicated.

Further, when testing for delay faults using the transition fault model, some parts were failing manufacturing test. It was determined that a possible reason was increased power consumption due to a burst in switching activity while testing for transition faults.

It was determined that the possible cause could be due to an increase in switching activity while the launch and capture pulses are applied while testing for transition faults.

Our aim was to analyze multiple designs and measure the switching activity during the launch and capture phase. Figure 19 shows the 'window' of interest, where we attempt to quantify the number of nets that switch during testing.

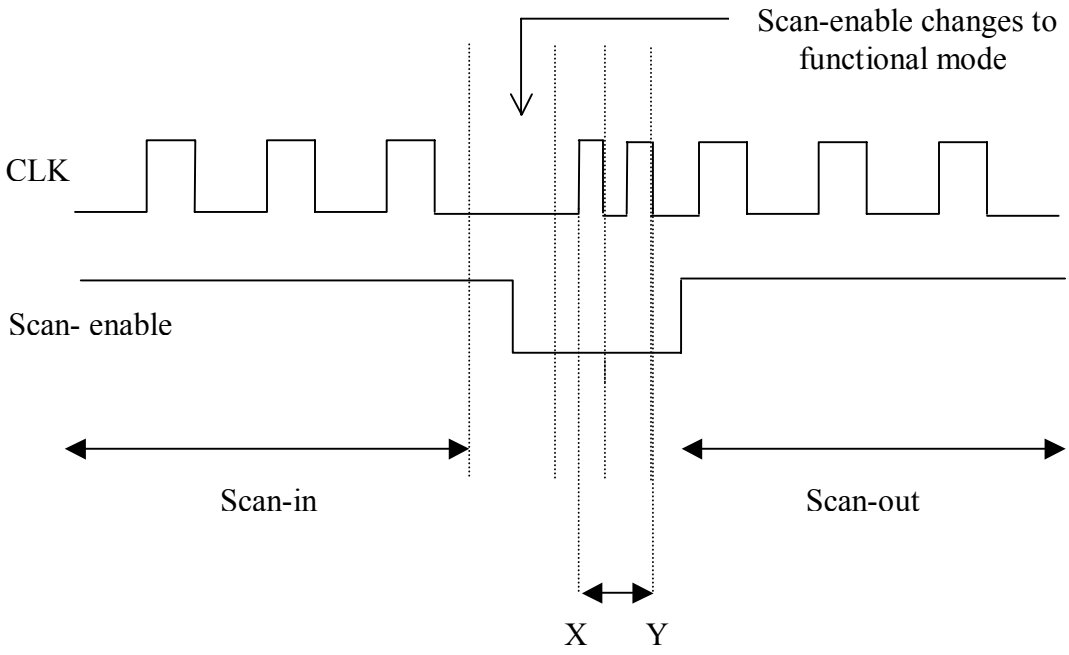



**Figure 19: Illustration of 'window' where switching activity is to be measured**

## 6.2 Evaluation of switching activity

A commercial ATPG tool was used to generate transition fault patterns and the patterns were saved in TI-TDL format. A TI-proprietary tool was used to generate the test-bench and the test-bench was then manually modified to include information on where the switching activity was to be observed.

The exact number of cycles to be skipped during scan-in was calculated. Immediately after this, the launch clock is pulsed. Let us refer to this instant as X. This point is also illustrated in Figure 19. Let us refer to the instant after the capture pulse as Y. Now, we concentrated on the nets that toggle between these two regions X and Y.

Cadence simulators support an option to record "value change information", also referred to as "value change dump" (VCD). In the test-bench, the following segment was inserted.

```
#X;
$dumpon;
$dumpvars;
#n;
$dumpoff
```

Here, "X" is the time instant where we would like to start measurement of the toggle activity. "n" is the sum of the width of the launch and capture pulses. The sum of "X" and "n" is "Y".

The designs were then simulated using Cadence NC-Verilog. The simulator recognizes the $dumpon and $dumpoff statements in the test-bench and dumps a list of all nets that toggle between time instant X and time instant Y. All types of transitions: 1->0, 0->1, 1->X, X->1, 0->X and X->0 are accounted for. Nets that toggle twice (0->1, ….., 1-> 0) are also accounted for.

Once the simulation is complete, a simple line count of the dump file between time intervals X and Y will give a count of all nets in the design that toggle between the launch and capture cycle. The dump file will also include the total number of nets in the design. With this the percentage of nets in the design that toggle can be calculated.

Two designs were used for this analysis. In Design A, 38.08% of the nets toggle between the launch and capture cycles. In Design B, 40.33% of the nets toggle. Along with results collected from earlier designs, it was concluded that the power distribution grids of circuits have to be designed to support approximately 40% of switching activity.

A lot of research has been done to come up with techniques to reduce power consumption during scan testing. An interested reader is directed to [32] for a description of such approaches and results from industrial designs.

# Chapter 7

# Conclusions and Future Work

In the semiconductor industry, there has been an increasing emphasis on the necessity for at-speed testing and transition fault testing is being widely adapted to screen delay defects. In this thesis, we have attempted to capture many facets of transition fault testing, their implications and how they can be supported in an industrial environment.

In particular, we have addressed the following topics:

An overview of manufacturing test and fault models used in the industry today to screen defects. We compared the different scan based testing approaches and discussed the advantages and disadvantages with alternatives like functional tests. We gave an overview of logic BIST and a brief discussion on how it is implemented and what the advantages are.

A primer on the different techniques to test for transition faults. The two main techniques covered are the launch-from capture and launch-from-shift techniques. A comparison of both the testing techniques was made and the pros and cons of each approach were discussed in length. A number of "design tricks" are being used in deep sub-micron technologies that make testing for transition faults very challenging. We have discussed the implications of these design decisions and shown what aspects need to be considered before deciding on a test strategy. We addressed low cost test and analyzed the tradeoffs we have to make to support at-speed testing in such low cost test environments. We then presented results from multiple ASIC designs and showed how each technique performs and the effect of design decisions we made earlier.

Next, we addressed the issues with pattern volume in testing for multiple fault models. We presented details on how we can use tests from one fault model like the transition fault model to test for stuck-at faults. We observed the implications on pattern volume,

test application time and test quality. We quantified the savings in pattern volume and the improvement in coverage we observed in two industrial designs.

Finally, we analyzed the effects of switching activity in circuits while testing for transition faults. We mentioned how switching activity is related to power consumption and presented a flow to measure the switching or toggle activity in industrial designs. Further, we presented results from two such ASIC designs.

Overall, we touched upon multiple issues related to transition fault tests and showed how each of them relate to a practical industrial environment. We hope these discussions and results will aid others in developing an effective delay test strategy in industrial circuits.

## 7.1 Future Work

A number of improvements to transition fault testing methods have been reported, but it remains to be seen if they can be used in a practical environment. Commercial ATPG vendors are constantly making improvements to their tools as well. A good technique would be a combination of both the launch-from-capture and launch-from-shift techniques, such that the maximum faults are detected. The order in which patterns are generated or how they are ordered is an interesting topic for future research.

It also needs to be determined if the additional faults detected by the launch-from-shift technique are really meaningful. It may be the case that all such “extra” faults detected are not functionally testable at all, thus amounting to yield loss. As a result, parts that are actually good may be discarded during manufacturing test.

Alternatives to expensive ATE are being explored and low cost testers are one of them. Testing for at-speed faults are always challenging with minimum resources and improvements to the process are in progress. The launch-from-shift technique has not been supported yet on low cost test equipment and test engineers are trying to counter this challenge.

Pattern volume and tester memory are always big issues and an efficient model to test multiple fault models needs to be proposed yet. Until then, clever tricks to combine testing of such models are being developed.

Finally, analysis of power consumption and switching activity in designs is still at a latent stage. As design sizes increase and frequencies ramp up, designers are working hard to come up with methods to reduce power consumption.

# Bibliography


[1] R.D. Eldred, “Test Routing Based on Symbolic Logical Statement”, Journal of ACM, January 1959, Vol. 6, pp.33-36.

[2] P.C. Maxwell, R.C Aitken, K.R. Kollitz and A.C Brown, “$I_{DDQ}$ and AC Scan: The War Against Unmodelled Defects”, Proceedings of IEEE International Test Conference, Oct 1996, pp. 250-258.

[3] P. Nigh, W. Needham, K. Butler, P. Maxwell, and R. Aitken, "An Experimental Study Comparing the Relative Effectiveness of Functional, Scan, IDDQ, and Delay-fault Testing, Panel discussion at IEEE VLSI Test Symposium, Apr. 1997, pp. 459-464.

[4] P. Nigh, W. Needham, K. Butler, P. Maxwell, R. Aitken, and W. Maly, "So what Is an Optimal Test Mix? A Discussion of the SEMATECH Methods Experiment," Panel discussion at IEEE International Test Conference, Nov. 1997, pp. 1037-1038.

[5] G.L. Smith, “Model for Delay Faults Based Upon Path”, in Proceedings of IEEE International Test Conference, Nov 1985, pp. 342-349.

[6] J.A. Waicukauski, E.Lindbloom, B. Rosen and V.Iyengar, “Transition Fault Simulation by Parallel Single Fault Propagation”, in Proceedings of IEEE International Test Conference, Sept 1986, pp. 542-549.

[7] N. Tendolkar, R. Raina, R. Woltenberg, X.Lin, B. Swanson and G. Aldrich, “Novel Techniques for Achieving High At-speed Transition Fault Test Coverage for Motorola’s Microprocessors Based on PowerPC™ Instruction Set Architecture”, in Proceedings of IEEE VLSI Test Symposium., Apr-May 2002, pp. 3-8.

[8] T. L. McLaurin and F. Frederick, “The Testability Features of the MCF5407 Containing the 4th Generation Coldfire Microprocessor Core”, in Proceedings of IEEE International Test Conference, Oct. 2000, pp.151-159.

[9] G. Hetherington, T.Frayars, N. Tamarapalli, M. Kassab, A.Hassan and J. Rajski, “Logic BIST for Large Industrial Designs: Real Issues and Case Studies”, in Proceedings of IEEE International Test Conference, Sept 1999, pp. 458-466.

[10] M.P. Kusko, B.J. Robbins, T.J Koprowski and W.V.Huott, “99% AC Test Coverage Using only LBIST on a 1 GHz IBM S/390 zSeries 900 Microprocessor”, in Proceedings of IEEE International Test Conference, Oct-Nov. 2001, pp. 586-592.

[11] R. Dorsch and H-J. Wunderlich, Accumulator Based Deterministic BIST," in Proceedings of IEEE International Test Conference, Oct. 1998, pp. 412-421.

[12] G. Kiefer, H. Vranken, E. J. Marinissen and H-J. Wunderlich, Applications of Deterministic Logic BIST on Industrial Circuits," in Proceedings of IEEE International Test Conference, Oct. 2000, pp. 105-114.

[13] D. Das and N. Touba, Reducing Test Data Volume using External/LBIST Hybrid Test Patterns," in Proceedings of IEEE International Test Conference, Oct. 2000, pp. 115-122.

[14] F.F. Hsu, K.M. Butler and J.H. Patel, “A Case Study of the Illinois Scan Architecture”, Proceedings of IEEE International Test Conference, 2001, pp.538-547.

[15] J. Saxena, K.M. Butler, J.Gatt, R.Raghuraman, S.P Kumar, S. Basu, D.J. Campbell and J. Berech, “Scan-Based Transition Fault Testing – Implementation

and Low Cost Test Challenges", Proceedings of IEEE International Test Conference, 2002, pp. 1120-1129.

[16] J.Savir and S.Patil, "On Broad-Side Delay Test", VLSI Test Symposium, Sept 1994, pp 284-290.

[17] J.Savir and S. Patil, "Scan-based Transition Test", IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, Vol. 12, No. 8, Aug 1993.

[18] J. Savir, "Skewed-Load Transition Test: Part 1: Calculus", Proceedings of IEEE International Test Conference, Oct 1992, pp. 705-713.

[19] J. Savir, "Skewed-Load Transition Test: Part 2: Coverage", Proceedings of IEEE International Test Conference, Oct 1992, pp. 714-722.

[20] M.H. Schultz and F.Brglez, "Accelerated Transition Fault Simulation", Design Automation Conference, June 1987, pp. 237-243.

[21] G.D. Robinson, "DFT-focused Chip Testers: What Can They Really Do?", Proceedings of IEEE International Test Conference, Oct 2000, p. 1119.

[22] S.Comen, "DFT-focused Chip Testers – What Can They Do?", in Proceedings of IEEE International Test Conference, Oct 2000, p. 1120.

[23] A.L. Crouch, "Position Statement: DFT-focused Chip Testers", in Proceedings of IEEE International Test Conference, Oct 2000, p. 1121.

[24] P. Muhmenthaler, "Enough Test with DFT-focused Chip Testers", Proceedings of IEEE International Test Conference, Oct 2000, p. 1122.

[25] M. L.Bushnell and V. D. Agrawal, "Essentials of Electronic Testing for Digital, Memory, and Mixed-Signal VLSI Circuits", Kluwer Academic Publishers, Boston, 2000.

[26] V.S. Iyengar, B.K. Rosen and J.A. Waicukauski, "On Computing the Sizes of Detected Delay Faults", IEEE Transactions on Computer Aided Design, Vol. CAD-9, pp.299-312, March 1990.

[27] B. Dervisoglu and G.Stong, "Design for Testability: Using Scan-path Techniques for Path-delay Test and Measurement", Proceedings of IEEE International Test Conference, 1991, pp. 365-374.

[28] Savir, J, "Developments in Delay Testing", VLSI Test Symposium, 1992 – "10th Anniversary - Design, Test and Application: ASICs and Systems-on-a-Chip", Digest of Papers, 7-9 Apr 1992, pp. 247-253.

[29] Savir, J, Patil, S., "Broad-side Delay Test**",** IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Volume: 13, Issue: 8, Aug 1994, pp. 1057 -1064.

[30] Maxwell, P.; Hartanto, I, Bentz, L, "Comparing Functional and Structural Tests", Proceedings of IEEE International Test Conference, 2000. pp. 400-407.

[31] B. Pouya, A.L. Crouch, "Optimization Tradeoffs for Vector Volume and Test Power", Proceedings of IEEE International Test Conference, October 2000, pp. 873-881.

[32] Saxena, J., Butler K.M., Whetsel, L, "An Analysis of Power Reduction Techniques in Scan Testing, Proceedings of IEEE International Test Conference, 2001, pp. 670-677.

[33] K. Tumin, C. Vargas, R. Patterson and C. Nappi, “Scan vs. Functional Testing – A Comparative Effectiveness Study on Motorola’s MMC2107$^{TM}$”, in Proceedings of IEEE International Test Conference, Oct.-Nov. 2001, pp. 443-450.

# Vita

Vinay B. Jayaram was born in Bangalore, India. He received his bachelor's degree in Computer Science and Engineering from R.V. College of Engineering, Bangalore University in August 1999. For a year, he worked at Oracle India Development Center, Bangalore in the Server Technologies Group. In the fall of 2000, he joined Virginia Polytechnic Institute and State University for a Master's degree in Computer Engineering and worked as a teaching assistant in the Bradley Department of Electrical and Computer Engineering between 2000 and 2002. He joined the *PROACTIVE* research group at Virginia Tech in 2001 and has been involved in research on DFT and ATPG since then. He held internship positions at Ericsson IP Infrastructure Inc., Rockville, MD during the summer of 2001 and at Sun Microsystems Inc., Sunnyvale CA in the UltraSPARC™ DFT group during the summer of 2002. During the fall of 2002, he worked on a co-op program at Texas Instruments, Dallas TX in the ASIC DFT group. He graduated from Virginia Tech in January 2003 and rejoined the ASIC DFT group at Texas Instruments. His technical interests include test generation for delay faults, DFT, memory testing and fault simulation.