Copyright
by
Jing Zeng
2005

The Dissertation Committee for Jing Zeng
certifies that this is the approved version of the following dissertation:

Improving Timing Verification and Delay Testing Methodologies for IC Designs

Committee:

Jacob A. Abraham, Supervisor

Magdy S. Abadir

Anthony P. Ambler

Nur A. Touba

Martin D. F. Wong

Improving Timing Verification and Delay Testing Methodologies for IC Designs

by

Jing Zeng, B.S., M.S.

DISSERTATION

Presented to the Faculty of the Graduate School of
The University of Texas at Austin
in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN
May 2005

Dedicated to my families

- with love and gratitude

Acknowledgments

This has been the longest journey I have embarked on in my life. Now it is coming to an end. I want to thank everybody in my family for going through all the ups and downs of school, work and life with me. With this journey as a constant backdrop, my life has seen many changes over the years. The most recent change is that I became a mother to my 6 month old son Connor. I want to thank my son for bringing me so much joy with his beautiful, amazing presence during some of my most stressful days. In his own way, Connor encourages me to finish my dissertation so that he could spend more time with his mother. I want to thank my husband Dan for the sacrifice, patience, and loving kindness he showed me during the final finishing of my thesis. I would like to thank my parents for their love for me, for instilling values of the importance of knowledge in me and always encouraging me to strive for the best. I am also grateful to my parents-in-law who are tremendously helpful with Connor and many other things. I thank my sister for wishing me well from Canada.

I am grateful to my supervisor, Prof. Jacob Abraham, for agreeing to supervise my research topics, for offering constructive criticism and helping me to stay the course over the years. I appreciate that Dr. Magdy Abadir, my manager at Freescale Semiconductor, who is also the co-supervisor of my committee, has been an invaluable mentor to me over many years. Dr. Abadir provides a work environment very conducive to research and development. From our technical discussions, many ideas sprung forth and became part of this thesis. I appreciate other committee members, Prof. Martin D. F. Wong, Prof. Nur Touba and Prof. Tony Ambler for taking the time to read and evaluate this dissertation. A special thanks to Prof. Wong for agreeing to staying on my committee after joining the faculty at University of Illinois at Urbana-Champaign and for his kind words of encouragement.

My friend and lab mate, Rob Sumners, has helped me greatly over the years through his friendship, wisdom, perspective, and technical discussion. I would like to thank Nari, Jayanta and Robin for their help, support and encouragement during this process. I also thank Melanie Gulick, Linda Frost, Debi Prather, Diana Perez, Shirley Watson, Barbara Schleier, Andrew Kieschnick and many other helpful staff members at the ECE department. My journey would not have been possible without their kindness and involvement.

Improving Timing Verification and Delay Testing Methodologies for IC Designs

Publication No. ____________________

Jing Zeng, Ph.D.
The University of Texas at Austin, 2005

Supervisor: Jacob A. Abraham

The task of ensuring the correct temporal behavior of IC designs, both before and after fabrication, is extremely important. It is becoming even more imperative as the demand for performance increases and process technology advances into the deep sub-micron region.

This dissertation tackles the key issues in the timing verification and delay testing methodologies. An efficient methodology is presented to identify false timing paths in the timing verification methodology which utilizes ATPG technique and timing information from an ordered list of timing paths according to the delay information. This dissertation also presents a speed binning methodology which utilizes structural delay tests successfully instead of functional tests. In addition, it establishes a methodology which quantifies the correlation between the timing verification prediction and actual silicon measurement of timing paths. This quantification methodology lays the foundation for further research to study the impact of deep submicron effects on design performance.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

High performance Very Large Scale Integration (VLSI) circuits are characterized by high complexity and large variations in component delays. They are also required to operate at increasingly high clock speeds which do not allow for much design margin. In such a scenario, it is imperative and more challenging to ensure the correct operation of such circuits within the allowed timing constraints.

Timing verification refers to pre-fabrication verification of the temporal behavior of a design database. Delay testing, on the other hand, refers to post-fabrication timing verification which subjects each chip to functional or structural test vectors at the rated clock speed.

Timing analysis is at the core of the timing verification and delay testing methodologies. The objective of timing analysis is to improve the accuracy of critical path identification in a design cycle. For testing, timing analysis improves the selection of critical paths for delay test generation. This goal of testing is important as it guarantees that the device fully meets customer performance expectations.

1.1 Scope of the Thesis

We first describe the problems that are being addressed in this thesis, then define the scope of the thesis.

- False Path Identification

  In industry, static timing analysis or structural timing analysis (STA) is the dominating approach in the design methodology to guarantee that the chip design meets the specified clock frequency before the actual fabrication. This structural approach can be very efficient. It scales well with complex, high performance circuits.

  The downside of STA is that a transition is assumed to propagate along a path without verifying that there is indeed some pattern of input stimuli that could sensitize the path. A path is considered *sensitizable* if a transition at the head of the path (*launch point*) can be propagated to the tail of the path (*capture point*). The transition may not be sensitized due to unsatisfiable logic conditions. This is the **false path** problem with the structural analysis [21]. The false path problem could result in a large percentage of identified paths being unsensitizable.

- Speed Binning Utilizing Structural Tests

  Speed binning is used to sort working ICs based on their performances and to assign them to appropriate price points. The use of functional vectors has been an industry standard for speed binning of high-performance ICs. This practice can be prohibitively expensive as ICs become faster and more complex. In comparison, structural patterns target performance related faults in a more systematic manner, which makes structural testing an attractive alternative for speed binning.

- Pre-silicon and Post-silicon Timing Path Correlation

  The lack of correlation between pre-silicon prediction and post-silicon measurement of timing paths highlights the disconnect between the design methodology and silicon bring-up efforts. During pre-silicon timing verification, a set of critical timing paths, S_1, is identified. Engineering efforts are then spent optimizing S_1. During the post-silicon phase, often a different set of critical timing paths S_2, which gates the speed of the silicons, is identified. Trade-offs of power and area to speed up timing paths in S_1 during optimization do not necessarily translate into speed-up for timing paths in S_2, which gate the performance of the actual silicons. This is caused by process variations, power noise, crosstalk, thermal effects etc., [24] [27][56], and also sometimes certain design practice contributes to the situation as will be described in Chapter 4.

To tackle the above listed problems, the principal contributions of this thesis are detailed below.

- Efficient utilization of automatic test pattern generation (ATPG) technique and timing information for false path identification in the timing verification methodology.
- Replacing functional tests with structural tests for speed binning.

- Quantification of the correlation between the pre-silicon predictions and post-silicon measurements of timing paths.

Before we discuss the specifics of the research of this thesis, we describe the background and related work on timing verification and delay testing.

1.2 Background and Related Work

1.2.1 Timing Verification

Timing verification determines the maximum delay of the circuit under some given component delay model. It also reports the critical timing paths with the maximum delay in the circuit. A *critical (timing) path* is a path in a circuit, which due to its cumulative delay, limits the operation of the circuit at the desired speed [73]. This critical (timing) path information can be used to decrease the delay of the circuit during performance optimization if the circuit violates some timing constraints. The terms *timing paths* and *paths* will be used interchangeably in this thesis.

The problem under investigation is the determination of the maximum delay at which the sequential circuit can be clocked, given the individual component delays, set-up and hold constraints, clock skews, etc. It has been generally carried out in two phases: determining the delays of the combinational blocks of the circuits, then calculating the minimum feasible clock period of the whole circuit.

To determine the delay of a combinational logic block, there are two broad categories of techniques [73]: timing simulation (also called dynamic timing verification) and timing analysis.

1.2.1.1 Timing Simulation

The timing simulation approach simulates the logic functions on each gate and propagates signals through the gates based on the gate functions and the signal delays. This approach relies on input patterns which are impossible to generate for all of the paths since the number of paths can be exponential with respect to the number of inputs. Accurate gate delay estimation depends on many factors from the gate-specific capacitance and slew rate to factors like voltage, temperature and process variations [55]. It is also prohibitively expensive to perform delay simulation of paths at the chip-level. Due to the above reasons, timing simulation is generally used for detailed timing characterization for library cells and custom designed circuits.

1.2.1.2 Timing Analysis

Structural timing analysis obtains bounds on the maximum delay of a circuit by analyzing the topological structures. To estimate the delay of timing paths in a circuit, certain assumptions of gate delay models need to be made. The simplistic gate delay model is the *fixed delay model*, where the delay of each gate is assigned a single delay value. A *bounded delay model* assumes that each gate delay can vary independently of each other in a range [22]. More refined delay models take into consideration that the delay may depend on the fanout [60][62], or on the transition propagated through the gate (rising or falling) [67][74]. In [12], authors present a delay model that includes gate inertia, i.e., the impulse at an input must be long enough to change the value at the output of a gate. In [63], authors take into consideration electrical phenomena at the transistor-level and determine the gate delay based on values and transitions at nodes that do not belong to the path, or that are visible only at the transistor-level.

Structural timing analysis attempts to efficiently process the exponential number of timing paths in the circuit using linear structural algorithms which do not take into consideration the circuit functionality. It is generally used iteratively to allow the timing issues with a certain number of critical timing paths fixed before it is run again to generate another set of critical paths. Unfortunately, not all the timing paths identified using such analysis can be sensitized since functional information of the circuit is not taken into consideration. The cost of optimization and iteration is expensive without any performance improvement when spent on optimizing unsensitizable paths. It leads to underestimation of the circuit speed, and becomes worse as the demand for the higher performance circuits grows. An unsensitizable critical timing path becomes a bottleneck to time-to-market and translates into trading power and area for a perceived but nonexistent performance gain.

There has also been much work on sensitization criteria and algorithms to find the maximum circuit delay. Benkoski et al. [6] have proposed a static path sensitization criterion. A *viable* criterion is defined in [54] where the side-inputs of the path under consideration either settle to non-controlling values or terminate a viable path under the test with larger (or equal) delay. In [15], authors have defined conditions for *vigorous* sensitization of a path.

In industry, designers perform timing analysis to find the frequency at which the design should work correctly after manufacturing. Timing analysis requires information about a design's logic gate and interconnect delays. For a design using standard cell libraries, the library description provides information about logic gate delays. For custom-designed logic, a thorough circuit simulation provides the logic gate delays. Interconnect delays are either estimated from the interconnect's dimensions (leading to an estimate of the associated resistance and capacitance), or the resistance and capacitance values are extracted from the design layout with extraction tools. Because a design must operate over a range of voltages and temperatures, and component delays vary with the operating voltage and temperature, timing analysis is usually performed at various temperature and voltage corners.

1.2.1.3 Statistical Timing Verification

As CMOS technologies continue to evolve and advance and minimum feature sizes continue to decrease, circuit timing reflects many important effects such as process variations, power noise, crosstalk, thermal effects, etc. These effects are hard to predict and model deterministically. Statistical timing analysis and timing simulation approaches attempt to better handle the deep sub-micron timing effects [2][9][45][46][49][56][59].

In the statistical approach, instead of fixed delays, delay variables with correlated variations among themselves, will be used to produce a statistical distribution, rather than best-case and worst-case models. This approach attempts to predict the percentage of circuits that will run at a given speed, while allowing the adjustment of timing criteria to reach the acceptable mix of yield vs performance for a given IC design.

Statistical timing verification is not the focus of this thesis.

1.2.2 Delay Fault Testing

At 0.13 micron and below, IC manufactures are starting to see more defects that are not caught by traditional stuck-at fault testing [3]. Defects like high impedance metal, high impedance shorts and crosstalk are not caught by traditional stuck-at tests. Instead they show up as timing failures that can only be caught by at speed testing. These defects are called *timing defects*, compared to logic defects which cause the failures of the circuit functionality. One study on a microprocessor design shows that if scan-based at-speed tests were removed from the test program, the escape rate went up nearly 3% [71]. Another study carried out at Standford University showed that, for 0.7 and 0.35 micron technology respectively, the detection of 30% to 35% of defective parts of circuits designed with standard cells depended on at-speed tests. Also 3 out of 116 defective parts would be missed when tested at slower than the expected operating speed at normal operating voltage [78].

The two at-speed fault models most widely used today include the *path delay* model and the *gate fault* model [18][32][33][34][67]. Path delay tests target faults which model manufacturing defects or process problems that can cause cumulative delays along the design's critical paths. Gate fault tests target faults which model manufacturing defects at the inputs and output of a gate. Under the *gross delay fault model*, a gate fault can be detected along any path, while under the *small gate fault model*, faults of different sizes are detected along different paths. Research has been done to propagate transitions along the longest paths [35][60]. A comparison between the gate fault model and the path fault model is presented in [38]. Other at-speed fault models include *segment delay fault* [29][30].

At-speed path delay or gate fault tests require test patterns with two parts. The first part launches a logic transition value along a path, and the second part captures the response at a specified time determined by the system clock speed. If the captured response indicates that the logic involved did not transition as expected during the cycle time, the path fails the test and is considered to be defective.

Two types of scan methods can be used for at-speed testing: *launch-off-shift* and *broadside* [44]. In the *launch-off-shift* method (Fig. 1.1), the last shift of the scan chain load also launches the transition event. The critical timing is the time from the last shift (or launch) clock to the capture clock. The figure also shows the launch clock is skewed so that it is late in its cycle, and the capture clock is skewed so that is is early in its cycle. This skewing creates a higher launch-to-capture clock frequency than the standard shift clock frequency. The main advantage of launch-off-shift method is that it only requires the ATPG tool to create combinational patterns, which are quicker and easier to generate. The main disadvantage (for mux-DFF designs) is that it imposes restrictions on the routing of the SE signal. The SE signal must switch from 1 to 0 very quickly, because after the second pattern (a shifted version of the first pattern) appears at the flip-flop outputs, the tester must capture the circuit response in a time equal to the system clock cycle time. One way to meet this requirement is to route the SE signal as a clock tree. However, routing another high-speed signal with clock-like accuracy, in addition to the clocks required for system operation, can be expensive. An alternative is to use a pipelined

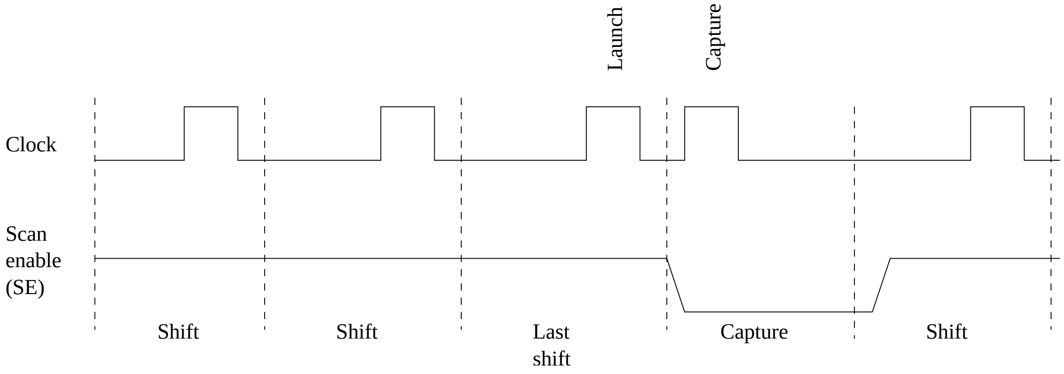


Figure 1.1: Launch-off-shift pattern timing

approach for SE signal distribution. Also the nonfunctional logic related to operating SE at a high frequency can contribute to yield loss. Another downside is that it could detect non-functional delay faults and scan chain shifting is usually done at lower frequencies.

In the *broadside* method (Fig.1.2), the entire scan data shifting can be done at slow speeds in test mode, and then two at-speed clocks are pulsed for launch and capture in functional mode. Once the values are captured, the data can be shifted out slowly in test mode. This method also allows using the late and early skewing of the launch and capture clocks within their cycles. The main advantage of this broadside approach is that the timing of the SE transition is no longer critical, because the launch and capture clock pulses occur in functional mode. Adding extra *dead cycles* after the last shift can give the SE additional time to settle. Another advantage of broadside approach is that it does not require scan chains to shift at-speed. From the design point of view, this makes broadside method simpler to implement.The ATPG problem is now sequential, as a result more faults can be detected in functional mode. The downside is

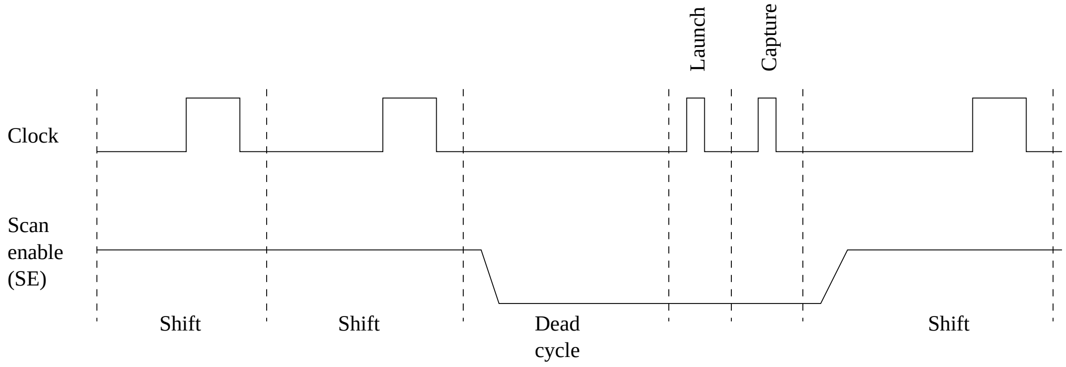


Figure 1.2: Broadside-pattern timing

that the test generation time can be longer and a higher pattern count might result. In this thesis, we use broadside scan method to apply our at-speed tests.

There are two main sources for the at-speed test clocks for test application: external ATE and on-chip clocks. Traditionally, ATE has always supplied the test clocks. However, the cost of the tester increases as the clocking speeds and accuracy requirements rise. The second source of clocks come from inside the chip itself. An IC design includes a phase-locked loop (PLL) or other on-chip clock generating circuitry. Because the purpose of a delay test is to verify that the circuitry can operate at a specified clock speed, it makes sense to use the actual on-chip functional clocks. On-chip functional clocks are not only more accurate, but also avoid the problem of importing high-speed clock signals into the device.In this scenario, the tester provides the slower test shift clocks and control signals, and the programmable on-chip clock circuitry provides the at-speed launch and capture clocks. Using these functional clocks reduces the ATE requirements, enabling the use of a less expensive tester. An example of

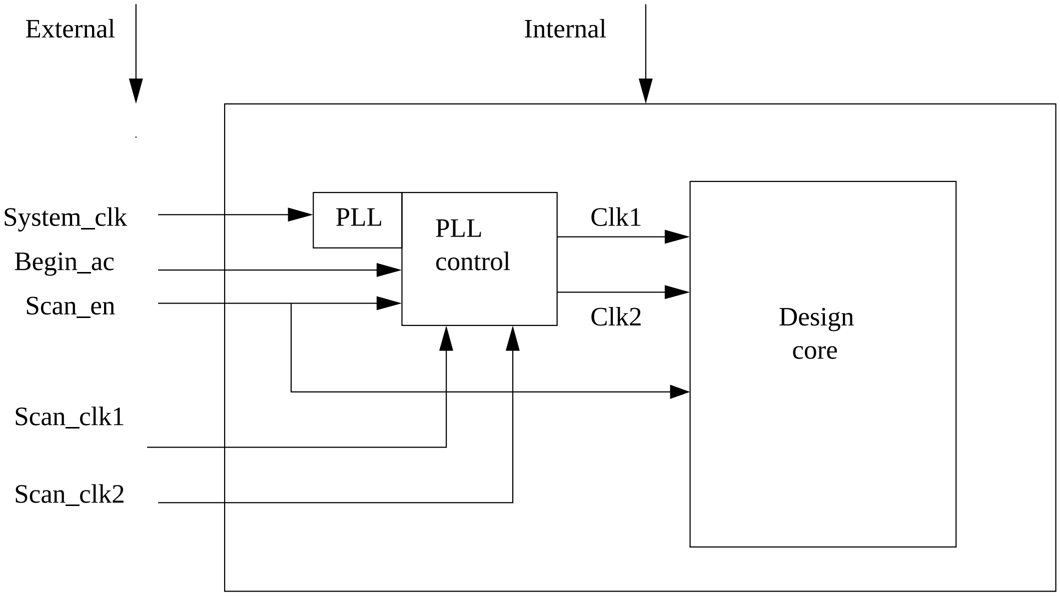


Figure 1.3: Phase-Locked Loop (PLL) Clock Generation with Internal and External Clocks

a programmable PLL is shown in Fig.1.3, combinations of system clock ($System\_clk$) and scan clocks ($Scan\_clk1$ and $Scan\_clk2$) are used to generate the internal clock signals feeding to the design core (Clk1 and Clk2).

1.2.2.1 Transition Fault Tests

Transition faults include slow-to-rise or slow-to-fall faults placed at the inputs and output of each gate. The number of transition faults in a design scales linearly with the size of the design. In order to detect a transition fault, an ATPG tool operates under the gross delay fault assumption and does not necessarily travel the longest path which limits the maximum operating frequency. So in general, a transition test may not always expose a small delay fault along a critical path. As a result, in a fixed-cycle testing scheme, a good portion of the transition tests are not suitable for delay defect testing if the paths being tested are too short compared to the functional cycle time. There have been many research attempts to overcome this problem [35][60][79].

1.2.2.2 Path Delay Tests

The path delay fault model concerns itself with propagation delays along one or more paths exceeding the timing constraint. A path is said to have a delay fault if the delay of the path is greater then the given timing constraint, and this excess delay is the size of the path delay fault. This model does not try to assign the excess delay to any sub-path or point-to-point connection. It also makes no assumption about the individual component delays. Path delay fault model is general enough to include the gate delay fault model.

An often-cited hurdle for path-delay testing is the large number of paths that a combinational circuit can have. In [65], it is shown that the number of paths can be an exponential function of gates. For example, the benchmark circuit c6288 [10] is known is have 1.98 x 10^{20} paths [73]. To contain the so-called *path explosion* problem in path delay testing, several criteria for selecting paths have been proposed. In [43], an algorithm is given that identifies a set of paths including the longest delay path through every net. Realizing that some selected paths may be untestable, authors generate tests for longest delay *testable* paths through all the nets [60]. *Vigorous* path sensitization criteria are developed where the timing of side inputs of delay paths are taken into consideration [15].

There are different types of path delay tests. Some of the path delay tests are called *robust* tests [17][67]. A robust test is supposed to detect a delay fault it targets regardless of the presence of other delay faults in the circuit under test. A non-robust test [17][67] is guaranteed to detect the fault it targets only if no other delay faults affect the circuit. Since non-robust tests are prone to fault masking, test generation procedures select robust tests (e.g., [14][50][53][60][64][66]). A non-robust test for a path will still be valid if we can guarantee that the side-paths which can invalidate the test are delay fault-free. Such a non-robust test is called a *validatable non-robust testable*. It has been shown that up to 43% of the paths in the ISCAS85 benchmark circuits are not even non-robustly testable [17]. Considerable work has been done to classify these remaining non-robustly testable paths based on delay fault testability. One such classification is shown in Fig. 1.4.

The set of path delay faults in a circuit can be divided into those that are *functionally sensitizable* and those that are *functionally redundant*. A non-robust *un*testable delay fault can still affect the timing of a circuit if it may be sensitized along with other non-robust *un*testable delay faults. These delay faults are considered *functionally sensitizable* [17], or *multiply-testable* [26]. Functionally redundant path delay faults correspond to those paths that cannot be sensitized for any combination of component delay values.

1.2.2.3 At-Speed Memory BIST Test

Testing of on-chip memories requires delivery of a huge number of pattern stimuli to the memory and the readout of an enormous amount of cell information [11]. As the number of memory bits per chip continues

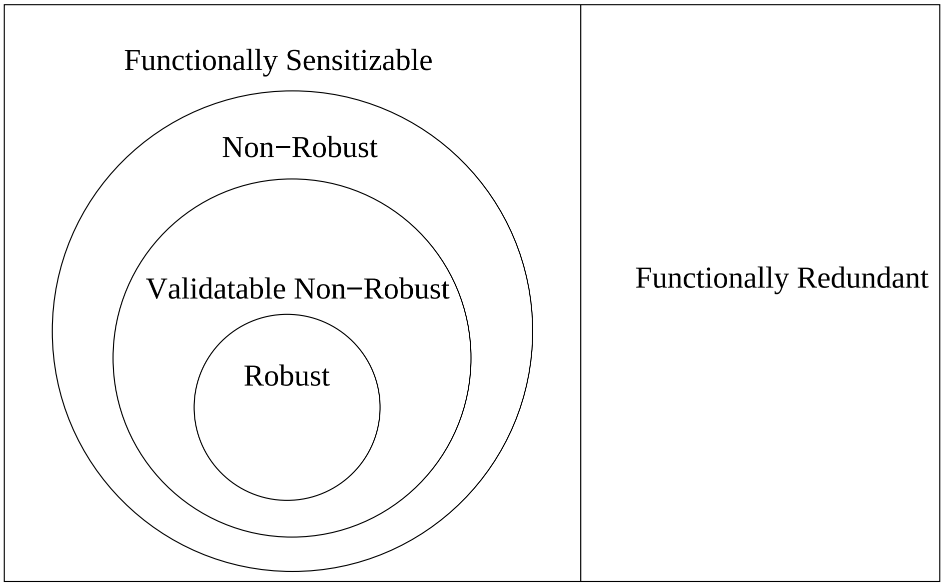


Figure 1.4: A Classification of Path Delay Faults

to increase exponentially and fault sensitivity increases, memory faults become more complex. As a result, applying any memory testing algorithm of complexity higher than O(n) becomes prohibitively expensive [11].

With memory BIST, the entire memory testing algorithm is implemented on-chip, and operates at the speed of the circuit, which is 2 to 3 orders of magnitude faster than a conventional memory test [28]. The following test hardware is needed for memory BIST (Fig. 1.5):

- A memory BIST *controller*.
- An *Address decoder*.
- A MUX circuit feeding the memory during self-test from the controller.
- A *comparator* for response checking.

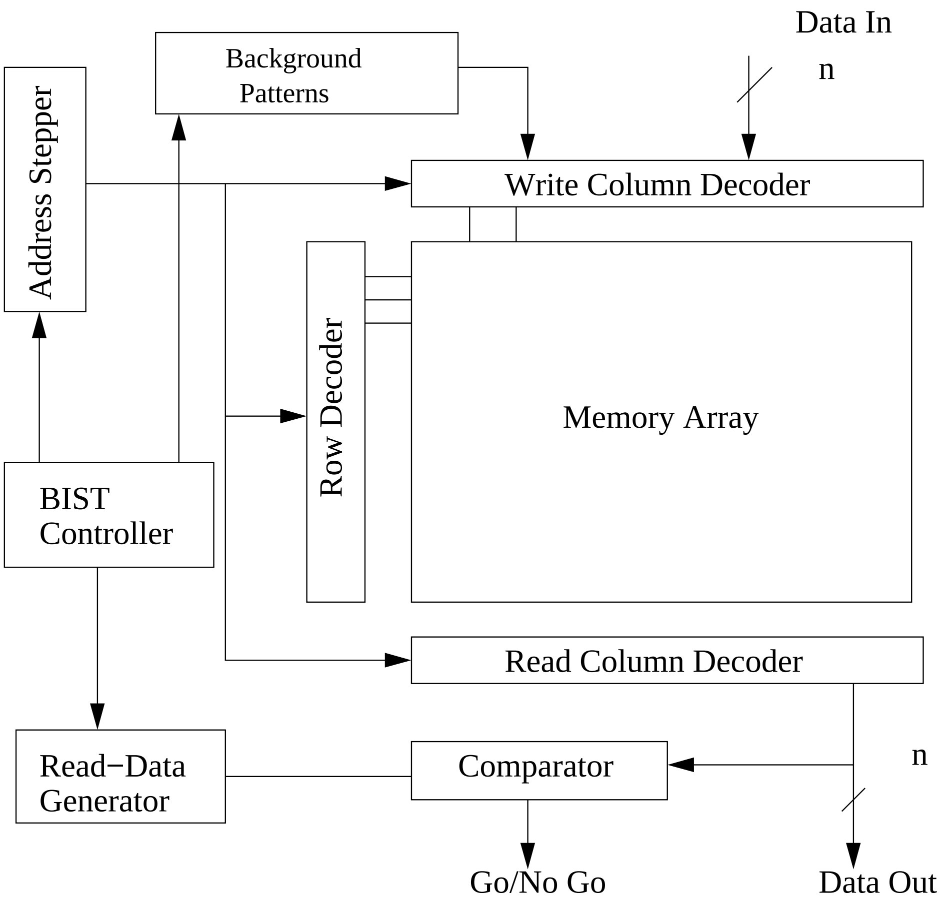


Figure 1.5: Memory BIST

- A *Background Pattern* inserter or *Data Generator* for inserting test patterns into memory columns.

If memory BIST related logic is conditioned to run at functional speed, memory BIST can be used to catch timing related faults inside the memories. This is especially important for high performance ICs in which the custom designed memories gate the frequency.

1.2.2.4 BIST for Delay Faults

It is also possible to test circuits for timing delays using BIST [11]. A delay fault BIST testing system has the standard BIST architecture, but with a hybrid pattern generator optimized to test both stuck-at faults and delay faults. While pseudo-random vectors provide good coverage of stuck-at faults, they can also cover a large percentage of transition faults if applied at high speed. Coverage of path delay faults frequently requires additional circuit modifications. When pseudo-random patterns are applied to the circuit, some long combinational paths that are non-functional in the sequential mode can be activated. When this happens, BIST can produce timing failures even in a circuit that meets the functional timing requirements. In such cases, the clock rate of BIST should be lowered below the specification. A suitable clock rate can be found by timing simulation. The power consumption of at-speed BIST can exceed the power rating of the chip. This is because of the high signal activity that random vectors cause in some circuits. Both peak and average power for BIST should be analyzed and corrected. BIST for delay faults technique is not the focus of this thesis.

Chapter 2

Utilizing ATPG Technique and Timing Information for False Path Identification

In this chapter, false path identification(FPI) techniques are discussed. First we have the following definitions.

A **critical** timing path (P) is characterized by a set of n nodes $x_1, x_2, \ldots, x_n$ and a set, $T = \{t_1, t_2, \ldots, t_n\}$, of signal transitions such that $t_i \in T$ represents the logic value transition on x_i. Node x_i is called a **path input** for path P. Many of these path inputs are associated with gate devices g_i of path **P** which can have other inputs which are called **side inputs**. The transition t_i of each node x_i is characterized by a pair of booleans $\langle b_i, a_i \rangle$ where b_i and a_i are the initial (or ***b***efore) and final (or ***a***fter) boolean values at node x_i, respectively. b_i and a_i are always complementary to each other, since we are concerned with the signal transition on every node along the path. $\{b_1, b_2, \ldots, b_n\}$ is called the $Before$ set and $\{a_1, a_2, \ldots, a_n\}$ is called the $After$ set. The time frame associated with the application of the before set is considered the **previous time frame**, while that associated with the application of the after set is the **current time frame**.

In Fig. 2.1, there are 8 library cells and/or custom macros in the circuits, path inputs x_1, x_2, ..., x_7 of gates g_1, g_2, ..., g_8, go through transi-

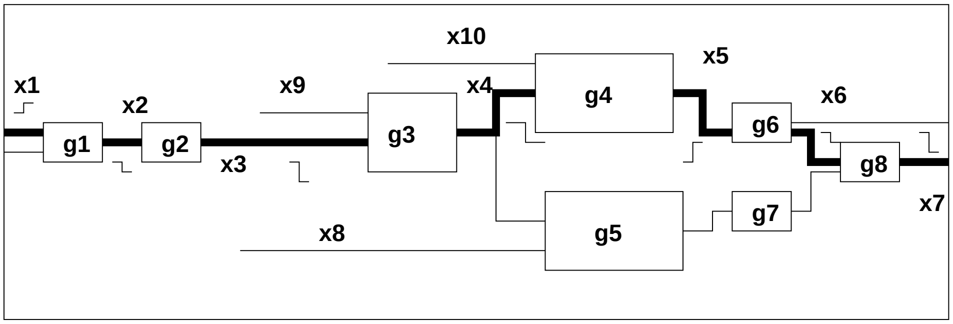


Figure 2.1: Critical Timing Path

tions $< 0,1 >$, $< 1,0 >$, $< 1,0 >$, $< 1,0 >$, $< 0,1 >$,$< 1,0 >$ and $< 1,0 >$. Gates g_i can have other inputs like x_9 for gate g_3, x_{10} for gate g_4 which are side inputs of the bold-faced path in the figure.

Definition: a *false* path is a path along which a specified logic transition cannot be sensitized.

Timing paths identified using structural timing analysis are considered to start from the **launch point** and end at the **capture point**. Each node x_i is either the primary input or output of a sequential element, library component or custom macro. x_i depends combinationally upon a set of sequential elements and primary inputs. Timing paths with outputs of sequential elements as their launch points and inputs of sequential elements as their capture points are called **latch to latch timing paths**.

Timing paths are generally listed in the timing reports based on their timing slacks. **Timing slack** is defined as the required arrival time minus the actual arrival time at a capture point of a timing paths. Static timing analysis can be configured to run so that only the single worst case timing path is generated for each capture point in the circuit. These paths are called **main paths**. Main paths can be ordered based on their timing slacks at the capture points with the path with the worst slack showing up first in the report.

Besides the main path for a particular capture point, paths converging to the same capture point, with differences in timing slacks from that of the main path within a given threshold can be generated. These paths are called **subpaths**. Subpaths and their corresponding main path form a group of **converging paths**.

2.1 Environment for False Path Identification

Certain assumptions need to be made about the environmental condition of a logic block before analyzing the stand-alone operation of such a block. Such assumptions include the range and arrival of primary input vectors to the block. A *single-stepping transition mode of operation* [22][23] is used where all circuit nodes are assumed to have stabilized to their final values under the first input vector before a second primary input vector is applied. This reflects the assumption that the block of interest is embedded in a scan latch/flip-flop based synchronous sequential system (Fig. 2.2). In this setting, a particular timing path could be false due to the following.

- **local constraints**: logic constraints of the block containing the path,
- **external/environmental constraints**: constraints imposed by the logic feeding the latches which surround the block containing the path.

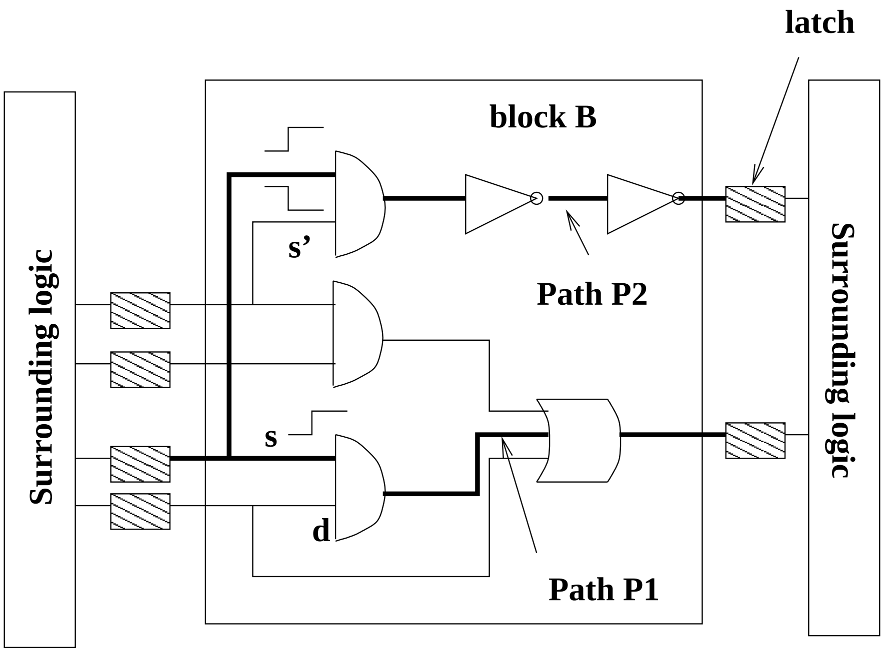


Figure 2.2: Logic Constraints

As shown in Fig. 2.2, paths P_1 and P_2 are two timing paths under consideration. Designs built with a test methodology which inserts scan latches between a block and its surrounding logic are considered. The latches are illustrated using shaded boxes in the figure. We can see the following.

- Path P_1 is false since d feeds into both the *and* and *or* gates. It violates *local constraints*.
- Path P_2 is false and violates *external constraints* if s and s' are constrained by the surrounding logic to be one-hot.

In general, in a System On Chip (SOC) setting, block B can be a processor itself or a peripheral device. Currently the analysis is restricted to local constraints within combinational blocks. An enhanced scan latch/flip flop based environment is further assumed, where an arbitrary vector pair can be applied to the combinational portion of a sequential circuit. With this assumption, we are being conservative in identifying false paths.

2.2 Logic Path Sensitization

In order to check whether a transition can be sensitized through a path P, we need to check if every x_i on the path can take up the values b_i as well as a_i, which is equivalent to checking the satisfiability of the following Boolean functions.

- e_a = True iff for all i, $x_i = a_i$ can be justified simultaneously when evaluated in the current time frame,
- e_b = True iff for all i, $x_i = b_i$ can be justified simultaneously when evaluated in the previous time frame.

The subscripts in the above terms e_a, e_b help to indicate the evaluation of different criteria. We call the boolean function which checks the satisfiability of all the a_is the **a**fter value criterion, while the corresponding one for b_is the **b**efore value criterion.

Failing to satisfy any of the a_i implies that P is combinationally false. In Fig. 2.3, it shows a 2-input AND gate with inputs i_1 and i_2 and output *out*. Let us consider the critical path section P (i_1, *out*) and the $0 \rightarrow 1$ transition at i_1. If at the same time, the transition on i_2 can only be $1 \rightarrow 0$, then *out* can not assume its final value 1. A delay of the $0 \rightarrow 1$ transition along path p cannot manifest at the capture point.

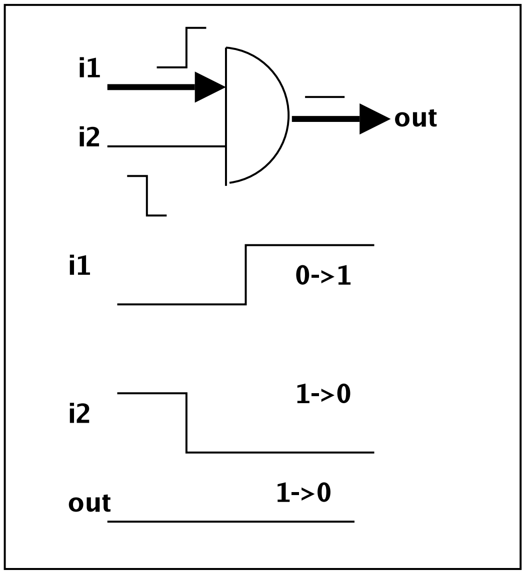


Figure 2.3: False Path due to Violation of the After Value Criterion

On the other hand, failure to satisfy a given b_i does not necessarily make a timing path false. For example, in Fig. 2.4, for a 2-input AND gate, having inputs i_1 and i_2 and output *out*. Let us take the critical path $\{i_1, out\}$ and the $1 \rightarrow 0$ transition on both nodes. If i_1, which is on the critical path, is undergoing a $1 \rightarrow 0$ and the side input, i_2, is undergoing a $0 \rightarrow 1$ transition then the before value criterion is not satisfied. But if the transition on i_2 happens before the transition on i_1 (which is a possibility since the path $\{i_1, out\}$ is the critical path) then there exists a functional test for this delay path. To account for the effect of the side inputs, the values in the before set only need to be checked when controlling values on the side inputs of the corresponding gates are assumed.

The following criteria take into consideration the violation of the *before set* when the side-nodes of path P have controlling values.

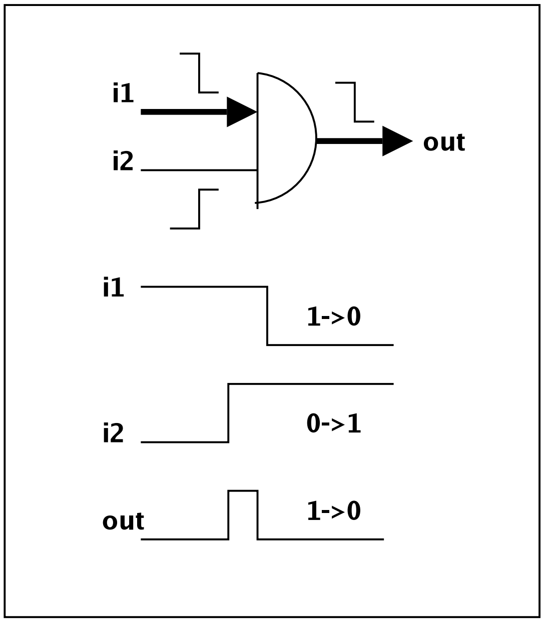


Figure 2.4: Effect of side-inputs

- e_n = True iff respective non-controlling values can be assigned simultaneously at all side-nodes when evaluated in the current time frame,
- e_{rb} = True iff x_i=b_i can be justified simultaneously in the previous time frame for every x_i where e_n is violated.

e_n corresponds to the evaluation of **n**on-controlling value criterion while e_{rb} corresponds to the evaluation of **r**evised **b**efore value criterion.

Using this notation, the following algorithm, which we will refer to as algorithm A, was presented in [7], [80].

```
Given a path $P$ with $e_a$, $e_{rb}$ and $e_n$
if $e_a = false$, then
```

```
        $P$ is a false path ($f_a$)
    else if $e_n = true$ then
        $P$ is an active critical path
    else if $e_{rb} = false$
        then $P$ is a false path ($f_{rb}$)
    else $P$ is an active critical path
```

The labels in parenthesis represent the group of paths which are identified false at a particular step in the algorithm. We also call the *false paths* identified in algorithm A *logically false paths* since only logic information is utilized for FPI. If a path is not identified as logically false, we call it a *logically active path*. Besides *logical false paths*, there will be other kinds of false paths identified as shown later.

In the above algorithm, an *active* path is a path which has not been identified as a *false path*. An *active* path is not necessarily a true path since the algorithm only identifies combinational false paths at this point.

2.2.1 Example Circuit

The circuit in Fig. 2.5 will be used to demonstrate the FPI process using logic path sensitization criteria. There are 6 paths in Fig. 2.5.

- P_1: rising transition through nodes “a”, “m”, “x”,
- P_2: falling transition through nodes “a”, “m”, “x”,
- P_3: rising transition through nodes “b”, “m”, “x”,
- P_4: falling transition through nodes “b”, “m”, “x”,

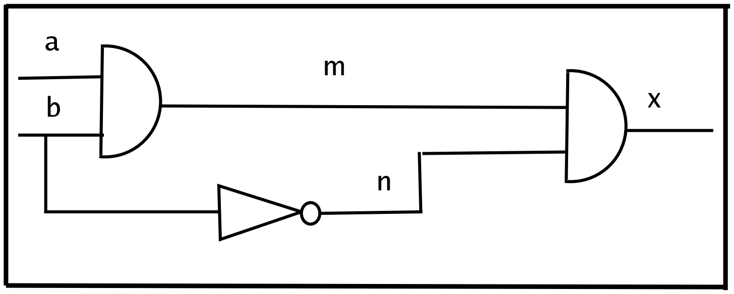


Figure 2.5: Example Circuit

- P_5: rising transition through nodes “b”, “n”, “x”,
- P_6: falling transition through nodes “b”, “n”, “x”.

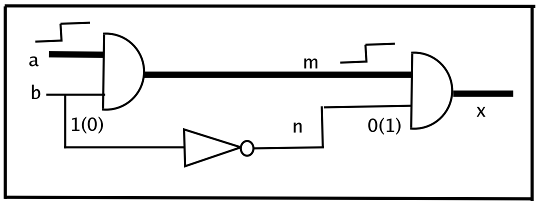
Based on algorithm A, *After value criterion* (e_a) is checked first. As a result, paths P_1, P_3, P_6 are identified as false (Fig. 2.6), while path P_2, P_4, P_5 are still considered *active. non-controlling value criterion* (e_n) is then checked. Paths P_4 and P_5 both satisfy e_n and can be tested non-robustly (Fig. 2.7). Path P_2 violates e_n, but not *revised before value condition* (e_{rb}) (Fig. 2.8). Based on logic sensitization criteria presented so far, path P_2 is still an active path.

2.3 Slow Path Condition: False Path Identification Using Logic and Delay Information

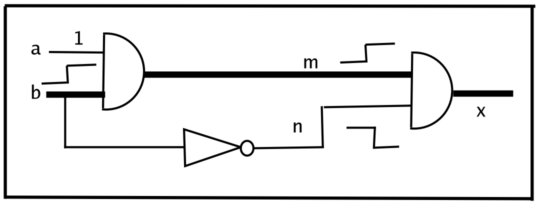
In general, due to reconvergent fanouts in the circuits, race conditions between timing paths exist. In Fig.2.9, both side input nodes of the AND gate $S(i11)$, $S(i12)$, along with the on-node i_1, transition from 1 (non-controlling value) to 0 (controlling value) of the AND gate. Here e_a and e_{rb} are satisfied for both the inputs and the output of the gate, but e_n

is violated since $S(i11)$ and $S(i12)$ assume the controlling value during the current time frame.

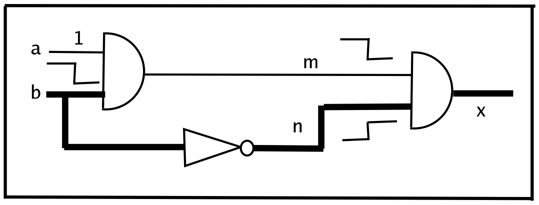
In a **race condition**, more than one inputs of a gate on the path under consideration (PUC) transition from the NCV (non-controlling value) to the CV (controlling value). The input(s) of the gate in a race condition



(a) Path P_1



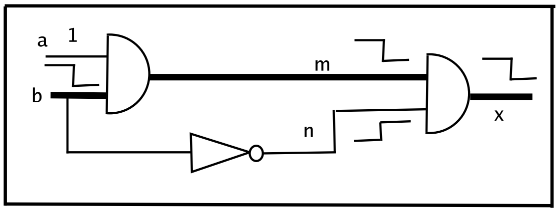
(b) Path P_3



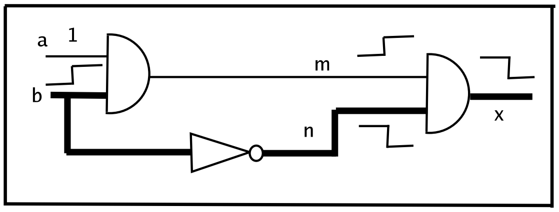
(c) Path P_6

Figure 2.6: Logical False Paths: Path P_1, P_3 and P_6

which are side-node(s) corresponding to the on-node n_i of the PUC are called $S(n_{ij})$. All the $S(n_{ij})$ for a PUC form a set **SN**. The path which forms a race condition with the PUC at $S(n_{ij})$ is called $P/S(n_{ij})$. All the $S(n_{ij})$ corresponding to $P/S(n_{ij})$ with longer delay than PUC form a set **SSN**. SSN is a subset of SN.



(a) Path P_4



(b) Path P_5

Figure 2.7: Non-Robust Testable Paths: Path P_4 and P_5

In Fig. 2.9, due to the race condition between the transitions of the paths associated with the inputs of the AND gate, the transition at the output of the gate is associated with that of the *faster* path. So unless the arrival times of the transitions associated with the side nodes of the AND gate, $S(i_{11})$, $S(i_{12})$ are slower than that associated with the on-node i_1,

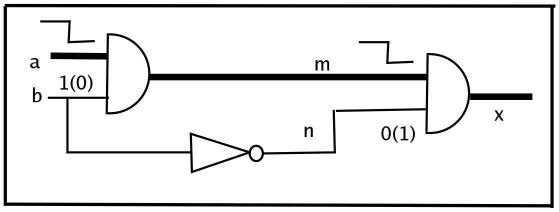


Figure 2.8: Path P_2: violates e_n

the delay effect of the on-path will not be sensitized.

One way of deciding whether the transition associated with the on-node is sensitized, is to keep track of the arrival time information at each node during the race condition analysis. But since we have a timing report generated already, we can utilize the delay information of timing paths from the timing report efficiently.

Notice in Fig.2.9, since the path section from the output of the AND gate to the capture point is shared among all $P/S(n_{11})$, $P/S(n_{12})$ and P, a slower arrival time at the side node $S(n_{11})$, $S(n_{12})$ implies that paths $P/S(n_{11})$, $P/S(n_{12})$ have longer delays than path P.

Thus the condition when the on-path transition can be sensitized in a race condition can be checked based on the delay information of converging timing paths. For any timing path P, which may or may not be a main path, we have the following theorem.

Theorem I: A timing path P with race condition(s), is a false path if there is not a logically active $P/S(n_{ij})$ of longer delay than P at any $S(n_{ij})$.

Proof: In Fig. 2.10, an "x" indicates a $S(n_{ij})$ node.

Take for example, two side-nodes $S(n_{11})$, $S(n_{12})$ and the corresponding on-node n_1 in a race condition. If either of the two side paths

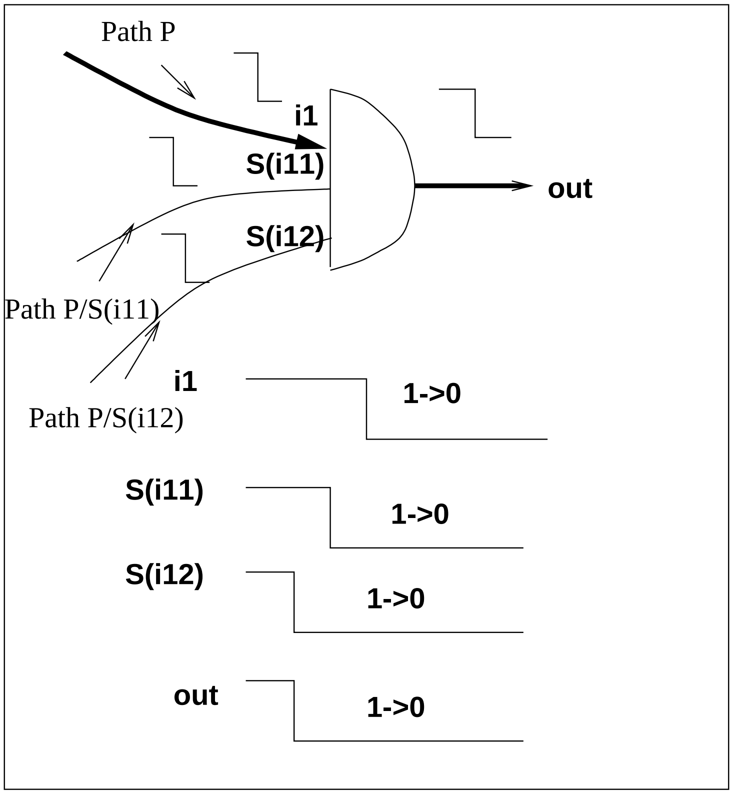


Figure 2.9: Path with Race Condition

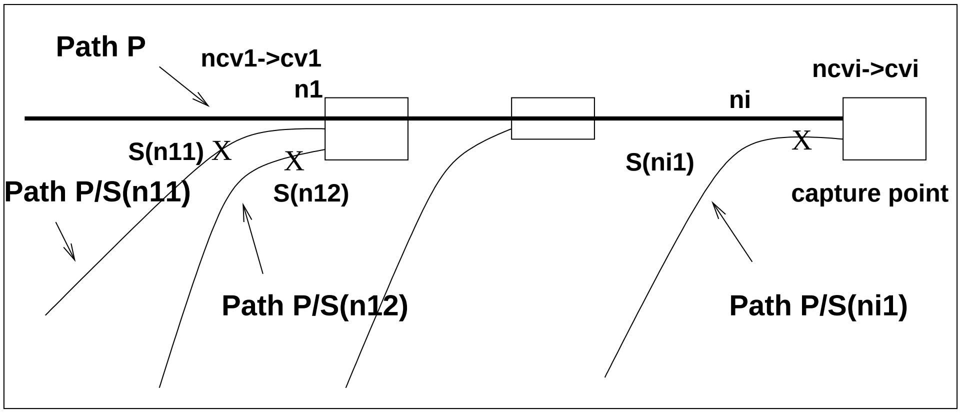


Figure 2.10: General Multiple Race Condition

$P/S(n_{11})$, $P/S(n_{12})$ corresponding to $S(n_{11})$, $S(n_{12})$ has been identified as logically false, then it would not help to sensitize the transition of path P. If either of the two side paths $P/S(n_{11})$, $P/S(n_{12})$ corresponding to $S(n_{11})$, $S(n_{12})$ is of shorter delay than path P, it would kill the propagation of the transition on node n_1.

The same condition needs to be true for all $S(n_{ij})$s *simultaneously*, otherwise P is a false path.

$\square$.

We call the condition defined in Theorem I the *Slow Path Condition*. The false paths identified based on the slow path condition are not logically false paths. Whether they can be sensitized depending on the delays of other logically active paths.

It is straightforward to reason that a main path with a race condition violates Theorem I.

Corollary I. A main path P with a race condition is false.

To check Theorem I, we can identify the exact locations of race conditions where e_a, e_b are satisfied, but e_n is violated. We can then check if there are logically active paths of longer delays associated with these nodes. This can be expensive. Instead, we identify the upper bound of the locations of race conditions while allowing the PUC sensitization using efficient structural analysis as follows. First we have the following definition. **Race condition intersection**: an intersection node where path P forms race condition with one of its converging path P' based on the transitions propagated through path P and P'.

Corollary II. A timing path P with race condition(s) is false if a race condition happens at a node which is not among the race condition intersection nodes between path P and its converging paths with longer delays.

The converging paths of P with longer delays can be identified from the delay information in the timing report. Their race condition intersections with path P can be identified by comparing path nodes and checking the transitions along the paths.

Corollary II avoids explicitly stating exactly where the race conditions are, but provides an upper limit for them for path P to be sensitized. We call this restricted subset of race condition intersection nodes **RCIN**.

If a converging path P' with longer delay is logically false, it cannot help propagate path P at their intersection node(s). A path P with race conditions is false if none of its converging paths with longer delays is logically active. Thus we have Corollary III.

Corollary III. A timing path P with race conditions is false if the corresponding converging paths of RCIN nodes are not logically active paths.

We call the FPI condition defined by Corollary II, III a *revised slow path condition* since it does not attempt to identify the exact locations of race conditions, only the upper bound. We called the final upper bound identified using the above corollaries **RSPN**. Fig. 2.11 illustrates the boundary of **RSPN**.

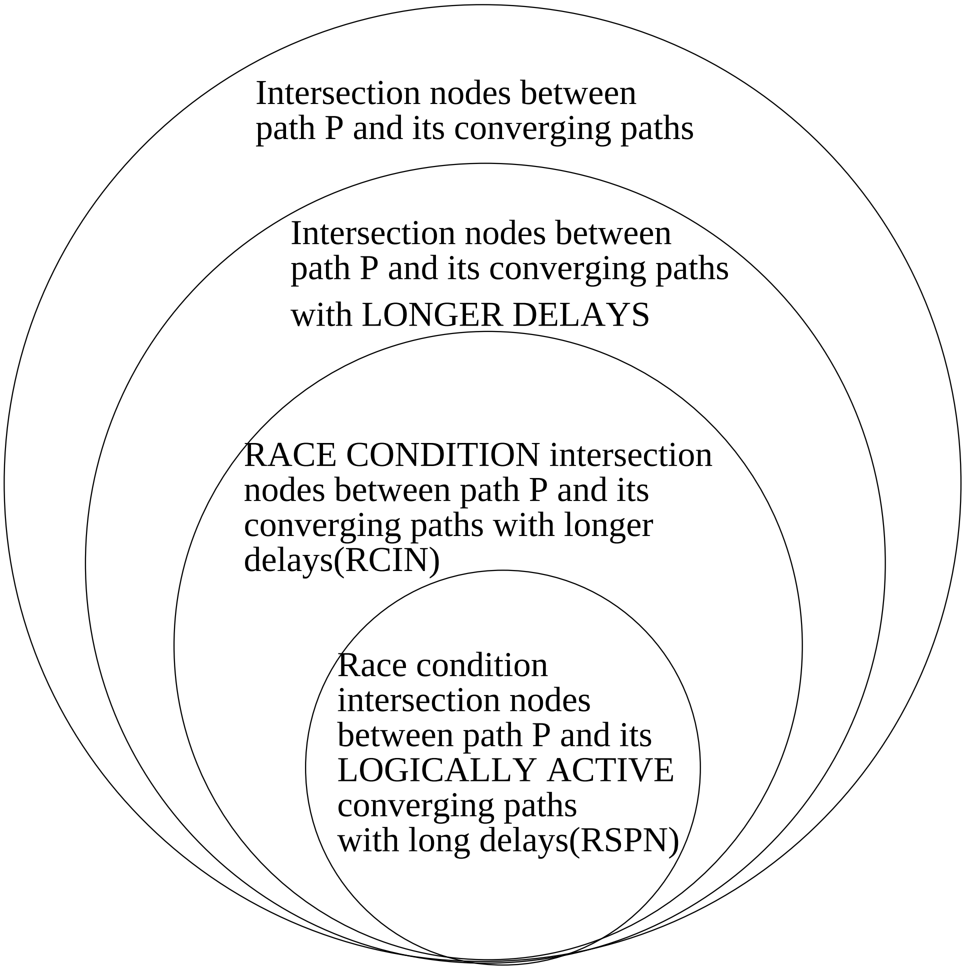


Figure 2.11: Upper Bound Identified for Race Conditions

2.4 Improved False Path Identification Algorithm

Besides e_a, e_{rb}, and e_n already defined while describing Algorithm A, we have the following additional definitions.

- e_s = True iff the set RSPN is non-empty,
- e_{rn} = True iff respective NCVs can be assigned simultaneously at non-S_n nodes.

Item e_s is for the evaluation of the revised **s**low path criterion, while e_{rn} is for the evaluation of the **r**evised non-controlling value criterion.

We check if RSPN is empty when e_n is violated, to see if there are race condition locations which *still allow* timing path P to stay *sensitizable*.

Corollary IV. A timing path P which violates e_n is false if RSPN is empty.

In Corollary IV, when e_n is violated:

- either e_a, e_{rb} is violated also, thus P is a logic false path,
- or race condition exists. Since RSPN is empty, P is a false path.

Additional logic checks need to be performed at nodes outside RSPN to further perform FPI on path P.

Corollary V. A timing path P is false if e_{rn} is not satisfied for all of its side-nodes outside RSPN.

Corollary IV, V allow efficient identification of a false path without specifying the exact cause of it being false. This allows earlier and more efficient identification of false paths.

Given a path P with e_a, e_s, e_{rn} and e_{rb}, Fig. 2.12 shows our improved algorithm B. The labels in parenthesis represent the group of paths which are identified false at a particular step in the algorithm.

The main advantages over algorithm A are the following.

- Additional false paths (f_s and f_{rn}) are identified.
- The simultaneous checking of the satisfiability of e_n and e_a to ensure that a specific transition can be sensitized.
- The locations where e_{rb} is checked are identified efficiently using the corollaries which do not take additional ATPG run time, compared to being checked exactly where e_n is violated in algorithm A.
- The checking of e_{rn} after e_s is done at a node $\notin$RSPN
- Paths which fail e_{rb} can be identified false as early as during the checking of e_s.

2.4.1 Example Circuit

We reconsider the circuit in Fig. 2.5. The checking of *After value Criterion* and *Non-Controlling Value Criterion* in our improved algorithm B remains the same as that in algorithm A. We conclude that paths P_1, P_3 and P_6 are false paths, while paths P_4 and P_5 can be tested non-robustly.

```
if $e_a$ = false, then P is false($f_a$)
else
    if $e_a$ and $e_n$ are true, then P stays active
    else
        if $e_s$ = false, then P is false($f_s$)
        elsif $e_{rn}$ = false, while $e_s$ is true
            then P is false($f_{rn}$)
        elsif $e_{rb}$ = false, while $e_{rn}$, $e_s$ are true
            then P is false($f_{rb}$)
        else P stays active
```

Figure 2.12: Algorithm B

Path P_2 is identified to violate e_n. To further decide whether path P_2 is a false path, we need to utilize timing information. We will analyze timing information under several scenarios.

Since there is only a single output of the circuit, all the paths from the input signals converge at the output. Based on the transitions at the path nodes, path P_2 could form race conditions with path P_4 and P_5 at node "b", "n" (Fig. 2.13). The transitions associated with paths P_2, P_4 and P_5 are labeled in Fig. 2.13. For example, at node "m", path P_2 and P_4 converge. The falling transition at node "m" is labeled with $t(p_2)$ and $t(p_4)$.

Even though the potential race conditions at node "b" and "n" could not happen simultaneously, that information is functional and is not available to timing analysis tool. Note both path P_4 and P_5 are non-robustly testable, thus *logically active.* Consider the following scenarios of timing information.

1. if path P_2 is the main path at the output port, i.e., it has the worst case timing out of all the converging paths, then *revised slow path criterion* is violated (e_s is false). As a result, path P_2 is false.
2. if path P_2 is *faster* than either P_4 or P_5 or both, *revised slow path criterion* (e_s is satisfied) path P_2 is active.

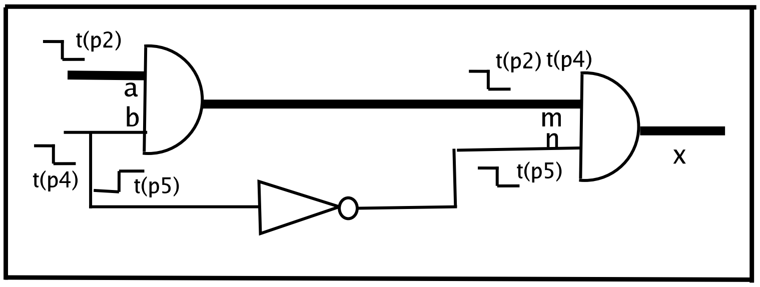


Figure 2.13: Path P_2 and Converging Paths: P_4 and P_5

We see since path P_2 violates e_n, whether it stays active depends on its timing relative to other timing paths.

2.4.2 Efficiency of Revised Slow Path Criterion

Revised slow path condition identifies the upper bound of locations where race conditions could happen. This is especially true for our example circuit since race conditions could not happen simultaneously at nodes “b” and “n” due to the logic of the circuit.

To appreciate the efficiency of our *revised slow path criterion*, we perform the following calculation. Suppose a path P with 15 path nodes. Only 4 of the intersecting path nodes of path P with other converging paths, transition from NCVs to CVs of the corresponding gates. These are fairly reasonable, conservative numbers based on the timing analysis of the industrial circuit we experimented with. We need to perform the following logic checking if we are to determine exactly where the race condition(s) happen.

- Assume exactly 1 out of 4 intersecting path nodes forms race condition with another converging path (4).
- Assume exactly 2 out of 4 intersecting paths nodes form race conditions with 2 other converging paths (6).
- Assume exactly 3 out of 4 intersecting path nodes forms race conditions with 3 other converging paths (4).
- Assume exactly 4 out of 4 intersecting path nodes forms race conditions with 4 other converging paths (1).

The numbers in the parentheses are for the number of logic checks that need to be performed. We would need to perform a total of 15 (4 + 6 + 4 + 1) logic checks to determine the exact locations of race condition before checking whether the timing information at these race condition locations satisfy the *slow path condition* requirements. We can see that the number of logic checks required for *slow path condition* can quickly add up and become prohibitively expensive.

It should be noted that if we still decide to determine the exact locations of the race conditions after the tight upper bound of these locations is identified using *revised slow path condition*, the number of corresponding logic checking would be much reduced.

2.4.3 The Effectiveness and Limitation of Revised Slow Path Criterion

In order to make sure timing requirements are met at all the capture points of a circuit, the timing path with the worst delay at each capture point, i.e., main path, is included in the timing analysis report. For each corresponding capture point, other converging paths with differences in timing slacks from that of the main path within a given threshold are also included in the timing analysis report.

When handling the timing paths with worst timing slacks at each capture point, *revised slow path criterion* prove to be very effective as will be shown in our experimental results. We can normally identify tighter upper bounds for race condition locations for a timing path P, compared to just the intersecting path nodes of path P with other converging paths, transitioning from NCVs to CVs of the corresponding gates.

On the other hand, the lower the rank of a timing path P is, or the faster a timing path is, the less effective the criterion is. There is because there would be many timing paths which are slower than path P, which converge at the same capture point. As a result, an upper bound of race condition locations could not be achieved as effective as in the case of more critical timing paths. Notice the application of our criterion is for false timing path identification of critical timing paths where optimization of false paths is a more pressing issue.

2.4.4 Timing Accuracy

The comparison of timing slacks of paths depends on how accurately delays are estimated. A threshold is needed to estimate how close the timing slacks of two timing paths can be and still be differentiated. In a race condition, if timing path P_1 has a slightly faster timing than P_2, P_1 might still be able to activate P_2 if P_1 ends up with slower timing in real silicon. To accurately determine the threshold in timing analysis which can also be used to differentiate two timing paths in silicon, correlation of timing prediction of timing paths in timing analysis and timing measurement of the corresponding paths in real silicon needs to be established.

2.5 A More Efficient Version of the Improved Algorithm

In the first two steps of Algorithm B, logic checking of e_a and the satisfiability of e_a, e_n simultaneously is carried out before the structural checking e_s. To minimize ATPG run time, we need to minimize the amount of logical checking and perform the structural checking as early as possible. We observe that the set of timing paths which fail the simultaneous satisfaction of e_a and e_n is the upper bound of all the false timing paths we identified in algorithm B.

With e_a, e_s, e_{rn} and e_{rb}, we propose algorithm C (Fig.2.14) for path P. Similarly, the labels in parenthesis represent the group of paths which are identified false at a particular step in the algorithm. The main efficiency advantage of algorithm C over Algorithm B is that the upper bound of all

```
if $e_a$ and $e_n$ are true,
    then P stays active
else
    if $e_s$ = false, then P is false($f_s$)
    elsif $e_a$ = false, while $e_s$ is true
        then P is false($f_a$)
    elsif $e_{rn}$ = false, while $e_a$, $e_s$ are true
        then P is false($f_{rn}$)
    else
        if $e_{rb}$ = false, the P is false($f_{rb}$)
        else P stays active
```

Figure 2.14: Algorithm C

the false paths is identified by e_a and e_n. This reduces the amount of paths that need be checked by further logic conditions, especially e_a. As in algorithm B, the upper bound is further reduced by checking e_s.

Note that we perform e_s on more timing paths in algorithm C than in algorithm B. This trade-off is minimal for structural analysis. Also, paths which fail e_a by itself and e_b can be identified as early as during the checking of e_s.

2.6 Bridging the Gap between the Physical Design and Testing Model

To check criteria e_a, e_{rb} and e_{rn}, we can set the corresponding values at the nodes along the path simultaneously using commands in the ATPG

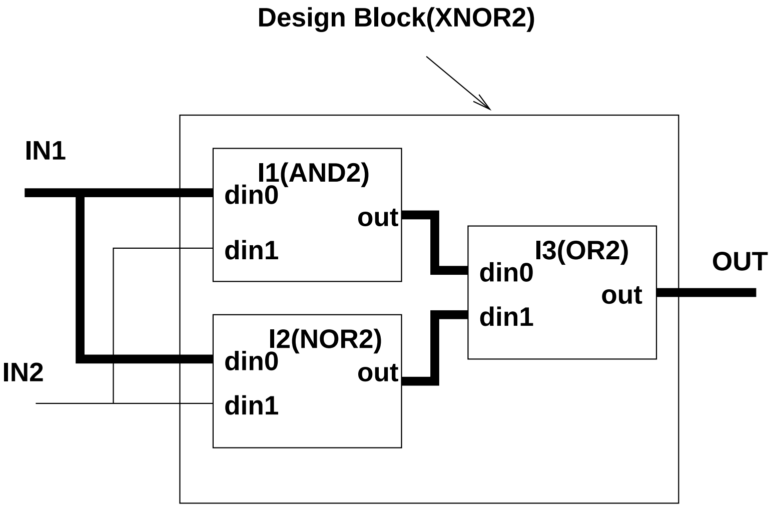


Figure 2.15: Ports of Custom Design Blocks

tool. In the current methodology, most of the pre-characterized design blocks are not gate level primitives understood by the ATPG tool. To specify the nodes which are ports for these design blocks, we analyze the gate-level models for the blocks to determine the gate primitives inside the blocks which are connected to these ports.

We implemented a path extractor which analyzes all of the custom designs and extracts all of the paths associated between each pair of input/output pins in a design.

For example, the design block in Fig. 2.15 is not a gate-level primitive. It contains gate-level primitives I_1, I_2 and I_3. Its input/output pins IN_1,IN_2 and OUT would then be specified as the following.

1. IN_1: /I1/din0, where $din0$ is an input port for $I1$,
2. IN_2: /I2/din0, where $din0$ is an input port for $I2$,

3. *OUT*: /I3/out, where *out* is an output port for *I*3.

There are 2 paths between $IN1$ and OUT, one is activated when $IN2$ is 0, the other when $IN2$ is 1. Our path extractor would extract both. We call a timing path with only library components along it a fully specified **logic path**. A timing path through a complex custom design can correspond to more than one logic paths. We will not label such a timing path false unless we can identify every one of its logic paths to be false. In this way, we are being conservative.

2.7 Experimental Results

We considered a circuit as being consisted of sub-circuits and RC networks. The timing behaviors of the sub-circuits were pre-characterized under numerous environmental parameters using transistor-level simulation. The RC nets were for the estimation of interconnect timing behavior using an extraction tool. Timing analysis was then performed on the whole chip. The output of the timing analysis consisted of a set of critical paths.

We show our results on latch to latch timing paths, but our techniques are generally enough to identify any type of false timing path. The output of the timing analysis consisted of a set of critical paths along with the transition for every node on each path. Our FPI engine post-processed the generated outputs from the timing analyzer.

Our FPI engine translated timing paths into delay paths for ATPG tools [80]. It checked the satisfiability of different criteria for the path by setting the corresponding values at the nodes along the path simultaneously using the ATPG tool commands. Based on the status returned after running the ATPG tool with command files, we used our FPI engine again to identify a list of false paths. For further logic checking, additional command files were fed to the ATPG tool and the log files from the ATPG tool were analyzed.

# of Transistors	Frequency	# of IO pins	# of latches	Stuck-at faults
33 million	1Ghz+	281	90k	6.2M

Table 2.1: Statistics for MPC7455

We ran our experiments on the circuit description for the MPC7455 microprocessor. Statistics are shown in Table 2.1. All runs were performed on a 400MHz Ultra60 running Solaris 5.6 with 1GB memory. Three sets of most critical timing paths were generated using the timing analyzer. We simplified the issue by analyzing only most critical latch to latch timing paths, but it was straightforward to extend the analysis to other kinds of paths. The cycle time target was 950ps. The threshold for generating subpaths was 3ps. The threshold for differentiating between the timing slacks of two timing paths was set to be 0.5ps. Any two timing paths with timing slacks of less than 0.5ps difference were considered to be of comparable delay. False paths identified in different groups following the convention described earlier are shown in Table 2.2.

Our additional checking of f_s and f_{rn} was effective in identifying false paths. This highlights the need of taking into consideration delay information of the paths besides logic value justification when performing FPI. The last column in Table 2.2 shows the number of false paths identified using algorithms B and C, both utilizing f_s and f_{rn}, over algorithm

Latch to latch timing paths	Algorithms									False Path Identification improvement of B, C over A
	A	B				C				
	fa	fa	fs	fn	fb	fs	fa	fn	fb	
61	6	6	9	5	0	12	3	5	0	3.3
332	26	26	36	6	0	54	8	6	0	2.6
566	54	54	56	6	0	95	15	6	0	2.1

Table 2.2: False Paths Identified using Different Algorithms

A. At least twice as many false paths were identified using f_s and f_{rn}, or at least 10% additional false timing paths out of the total timing paths analyzed in all three sets of latch to latch timing paths. Note that only f_a was identified in algorithm A since identifying f_{rb} without structural analysis information was expensive.

We can see that the number of false paths in different columns identified using algorithm C were different from those identified using algorithm B. Structural analysis e_s was performed earlier in algorithm C allowing it to identify false paths which were under category f_a in algorithm B. This results in the number under the f_a using algorithm C being much less than that using algorithm B. The number of timing paths which needed to be checked using f_a was also reduced. In general, algorithm C performed much less logic checking than algorithm B, thus algorithm C took much less time than algorithm B since the run time of the algorithms mainly came from running the ATPG tool for logical checking. The run time for the 3 sets of timing paths was listed in Table 2.3. The ATPG abort limit was set to 100.

latch to latch timing paths	A (mins)	B (mins)	C (mins)
61	30	63	33
332	68	141	73
566	94	195	101

Table 2.3: Comparison of Algorithms

We see algorithm C requires comparable amount of time to algorithm A, which is about 50% of what it took for algorithm B. With a set abort limit, in the worst case, the overall run time scaled linearly with the total number of paths. If the logic checking for most timing paths took much less time than the abort limit, the ratio of overall run time for two sets of timing paths could be smaller than the ratio of the numbers of timing paths of two different runs. From Table 2.3, between the first two data groups, the ratio of timing path number was 5.4 (332/61), while the ratio of run time was 2.3. The same was true for comparison between all of the data groups. The loading of the gate-level model required around 1 hour and was not included in the run time since it was an one-time cost and was the same for all algorithms.

2.8 Conclusions

We demonstrate new techniques to perform false path identification utilizing both logic and delay-based information of the paths in the circuit. We were able to identify at least 20% false paths out of latch to latch main paths and subpaths within the different groups of timing paths. The false path identification criteria using delay-based information identify at least 10% more false paths out of all the latch to latch timing paths.

Our algorithms can be applied using any commercial ATPG tool and utilizing timing information from any static timing analyzer. Our algorithm using delay-based information is particularly effective for identifying the false paths among the most critical timing paths where most optimization efforts are spent.

Chapter 3

Correlating Structural Tests with Functional Tests for Speed Binning

3.1 Definition of Speed Binning

In general, the purpose of testing is to bin the tested ICs based on certain criteria. Most of the research and industry work on testing has focused on binning to identify defective ICs. Binning requires an efficient and affordable mechanism to categorize the ICs. The mechanism to evaluate the ICs needs to correlate with the characteristic that the tester wishes to identify.

Speed binning is used to sort working ICs based on performance to assign ICs to appropriate price points. It focuses on how fast the IC can perform its intended functionalities. Typically, companies choose an IC's operating frequency to meet customer and marketing requirements, as well as to make an economic trade-off between yield loss below the operating frequency and the higher value of a faster design. In addition, it is necessary to distinguish faster ICs to be sold as a higher operating frequency. Speed binning runs tests at the highest rated functional speed for each part. Parts are then separated according to the highest speed test they pass, and then priced accordingly.

3.1.1 Types of Speed Failures

There are two types of speed failures, one due to process variations, another due to random defects. In a defect-free IC, relatively few critical paths dominate the maximum frequency over a range of parametric variations in manufacturing and operating conditions. When parts fail because of process-related speed variation, many parts have their maximum frequency dominated by a few critical paths.

Even when the process parameters of a manufactured part are within an acceptable range, a delay defect caused by random events can appear. If the delay added by a defect exceeds the timing slack of the path it lands on, it may cause a timing failure also. Unlike process variation, a delay defect can appear anywhere on the die and can turn any path into the critical path gating the frequency of a IC, depending on how much extra delay the defect introduces. Random delay defects can also exist on critical paths from normal process variation. Normal process variation has a Gaussian distribution. Outliers to this Gaussian distribution fall outside the envelope of normal process variation and are probably due to random delay defects.

Random delay defects which increase interconnect resistance or cross-talk sensitivity or that decrease supply voltage or clock skew can cause very small delay defects. Many of the random delay defects can occur in ICs which continue to operate above the desired frequency. These defects have no impact on a product sold at a single frequency, but they can change the bin a particular IC belongs in. So speed-binning strategy must account for random delay defects as well as defects induced by process variations.

Speed binning is typically achieved with functional test patterns.

Functional patterns target systematic speed failures caused by process variation. It is expected that systematic delay problems caused by process variation will affect several paths of the chip (including critical paths) and will hence be detected by functional test patterns targeted at exercising the critical paths. However, functional patterns are expensive to develop. The development of functional patterns require significant effort by the design team. Furthermore, there is no formal measure of their effectiveness in assessing the true speed of a complex design. Applying functional patterns at full system speed demands an expensive, full-function tester that can provide a high degree of timing and edge accuracy. As designs become faster and more complex, the cost associated with the development of functional patterns and the cost of functional testers can become a tremendous burden for the design team. So even though functional testing has the advantage of exercising the part in the way it can be exercised in the system, writing functional tests that target thousands of paths would be beyond the means of most projects.

In contrast to systematic defects, random delay defects can appear anywhere on a part. Functional tests that exercise a design's timing-critical paths are useful for detecting random delay defects that happen to fall along the targeted paths, but such tests are generally not sufficient. Structural tests, on the other hand, can be generated systematically to cover the entire design.

3.1.2 Structural Tests for Defect Detection vs Speed Binning

Structural tests are generated automatically by ATPG tools. By selecting a transition fault model and/or a path delay fault model, ATPG patterns can be used to measure the design's overall performance. The fault coverage of these ATPG patterns provides a succinct measure of pattern quality. Moreover, it is much easier to debug ATPG patterns. Scan-based structural tests can be applied with low-cost testers. While scan data may require large amount of tester memory, fewer tester channels are required and few, if any, high-accuracy functional channels are required.

Structural tests have been generally used for defect detection. In [36], delay testing techniques are used to characterize delay-related defects. The authors show that a reasonable defect level would not be achievable without delay testing. In another paper [20], pragmatic application of structural delay tests is described from early design cycle to post-silicon phase. In [47], structural delay tests are applied at a wide variety of operating conditions for exploring the relationship between test frequency and defect coverage. In [61], the author describes BIST implementation techniques to achieve structural at-speed testing and he addresses important timing setup issues from a logic BIST perspective.

The difference between utilizing structural tests for defect detection and for speed failure detection or speed binning is the clock frequency at which the test patterns are applied and the outputs of the IC are captured. In general, test-only paths can be exercised during scan-based delay testing. Delays along test-only paths are not necessarily guaranteed to meet the clock cycle time and can be significantly slower. Thus when the tester captures the response of the IC circuit after one clock period or very close to that, it is possible that the destination latch will not correctly capture the response of a transition sensitized along a very long test-only path. The launch-on-capture approach to delay testing can minimize, but not eliminate this problem because it is difficult during ATPG to determine whether the first pattern scanned into the latch is a valid system state that can appear in the latches during functional operation.

If our goal is defect detection, we would allow the test-only paths to be included in the delay tests if they provide additional delay fault coverage. Even if these delay faults covered by test-only paths might not affect the speed of the IC in functional mode, they can still pose a liability issue to the overall lifetime of the IC circuit. As a result, the overall delay tests including the test-only paths, will be applied at a frequency lower than that of the clock frequency to allow signals propagating along these test-only paths to have enough time to settle down before the logic values at the outputs of the IC under test are captured. Structural delay tests run at lower than full speed can still be effective in total defect control by screening out a significant, measurable portion of defects.

If our goal is speed binning, we would only be interested in the speed of the IC circuit under the normal mode of functional operation. As a result, we would seek to eliminate test-only paths completely so that the clock frequency can be applied at full speed to determine the overall number of parts which meet frequency, and at higher than full speed to determine parts to be sold at higher frequency.

To utilize structural tests for speed binning, previous results [4] has shown the correlation between the frequencies defined by functional test (*functional frequencies*), and frequencies defined by at-speed structural tests (*structural frequencies*), include at-speed memory tests, transition fault tests which do not go through arrays, and latch-to-latch path delay tests. There are coverage holes left due to the lack of transition tests going through arrays, and due to the exclusion of timing critical paths other than the latch-to-latch paths. In another paper [19], a linear relationship between the frequency determined by functional tests and latch-to-latch path delay tests is described. The authors establish a linear relationship between the two frequencies given by the functional tests and the path delay tests. Using that linear relationship, latch-to-latch path delay vectors can be used for speed binning during the production phase. In this thesis, we investigate the correlations between functional frequency and the frequencies obtained from various types of structural patterns.

3.2 Different Types of Structural Tests

For the purpose of our study, we consider the following types of structural tests:

- **At-speed memory BIST tests**: also called ABIST for A(array)BIST. We use the terms *array* and *memory* interchangeably in this thesis.
- **Transition tests without going through arrays**: also referred as *simple transition tests* in this thesis.
- **Transition tests going through arrays**: also referred to as *complex transition tests* in this thesis.
- **Latch-to-latch path delay tests**: also referred to as *simple path tests* in this thesis.
- **Memory and cycle-stealing path delay tests**: also referred to as *complex path tests* in this thesis.

3.2.1 At-Speed Memory BIST Test

Memory BIST focuses exclusively on testing the logic inside the embedded memory arrays and signals that touch the BIST circuitry, such as BIST address/data muxes and fail/redundancy data collection downstream of the memories. Memory BIST running at functional speed can catch timing related faults inside the memories. This is especially important for high performance ICs in which the custom designed memories gate the frequency.

3.2.2 Transition Fault Tests without Going through Arrays

As noted in [4], typical transition test sets do not provide high coverage in the logic associated with the custom memory arrays. The difficulty of detecting transition faults that require going through arrays stems from the fact that: (a) it is generally difficult to model memories to ATPG tools, (b) those faults require longer test sequences and more complex clocking. The detection of some transition faults requires going through embedded arrays, either for control or for propagation.

3.2.3 Transition Fault Tests Going Through Arrays

A different set of transition fault patterns was generated that provided coverage in the logic associated with the custom memory arrays (as well as the rest of the logic). It is important to note that many of the frequency limiting paths in most industrial microprocessors are associated with the custom memory arrays [4].

3.2.4 Path Delay Tests

Static timing analysis (STA) can be used to generate a list of critical paths. A path delay ATPG engine can then be used to generate path delay tests for these critical paths. The advantage of this methodology is that the critical paths identified by STA are the best candidates to expose subtle timing issues. Note that the path delay fault universe is exponential with respect to the size of the circuit. This makes it impossible for all the paths to be tested completely. A limitation of path delay test based on critical paths is that the critical paths identified by STA do not always determine the IC's speed. The accuracy of STA is affected by several physical factors such as temperature, clock skew, interconnect delays and crosstalk effects. To perform meaningful path delay tests, all the delay path types need to be included. This includes not only the latch-to-latch paths, but also memory and cycle stealing paths.

- **Latch to Latch Path Delay Tests**:

  Latch to latch delay paths are launched from latch outputs and captured at latch inputs. This is the simplest type of delay path.

- **Memory Path Delay Tests**:

  Memory paths are launched from the data output ports of the memories and captured at the latches, or they can be launched from outputs of the latches and captured at the data input ports, address ports or control ports of the memories. For example, Fig. 3.1 shows a rising transition is launched from the data output port $do\_7[2]$ of a memory with multiple read/write ports. The memory has a read-off value of

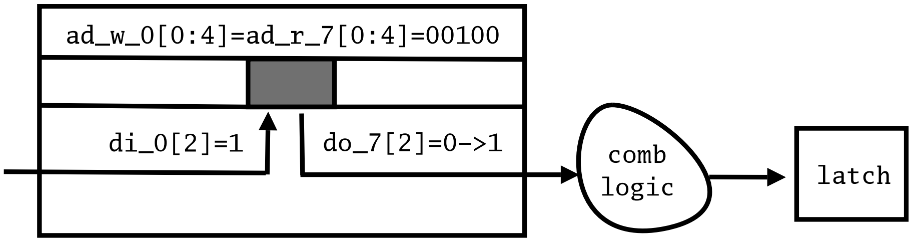


Figure 3.1: Memory Path Delay with Transition Launched from Data Output Port

0 at its output ports. As an example, to generate a test for this path, an ATPG engine writes a 1 to one of the data input ports ($di\_0[2]$) at some address ($ad\_w\_0[0:4] = 00100$), then performs a read operation of the data value at the same address ($ad\_r\_7[0:4] = 00100$). Since $do\_7[2]$ has a read-off value of 0, a rising transition is formed as the 1 is read out. The ATPG engine also needs to propagate the transition at the data output through the combinational logic to the input of a latch. The complete test sequence obviously requires several clocks to perform the write, read and latch capture operations.

- **Cycle Stealing Path Delay Tests**:

Cycle stealing is a necessity in many high-speed ICs. As an example, consider a two-phase clock, latch-based design as shown in Fig. 3.2. Here the transition on the delay path is launched on the rising edge of the first c2 pulse. Depending on the delay of the combinational logic in P1, the time at which the transition arrives at the first (C1) latch varies. Likewise, the delay along P2 affects the arrival time at the second (C2) latch. The combined path needs to meet a timing of three half-cycles. In the simple case, P1 is a full cycle path (rise of

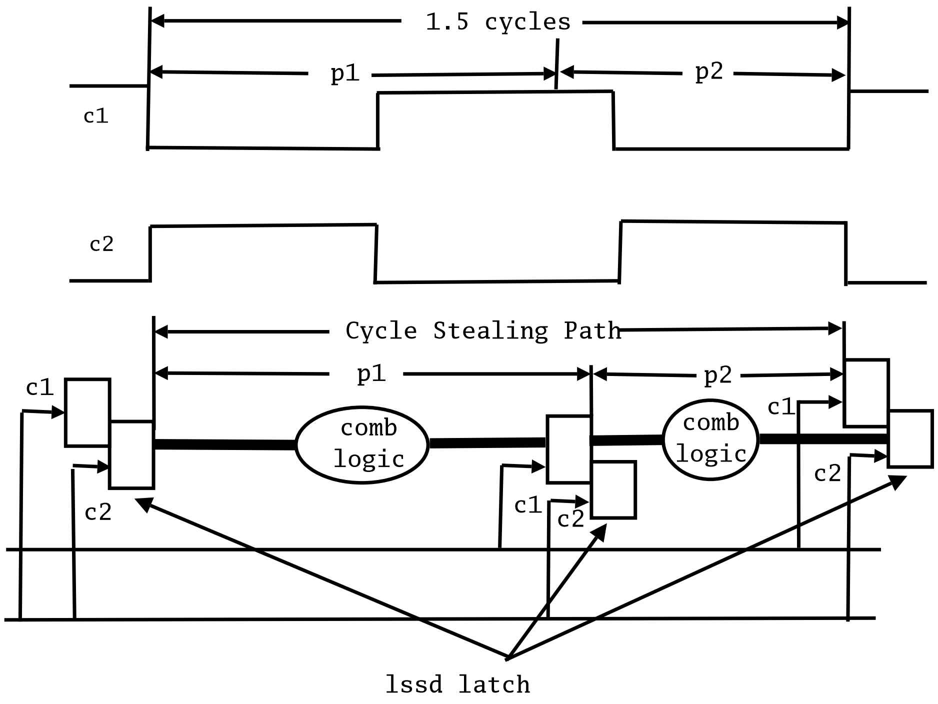


Figure 3.2: A Cycle Stealing Path

> C2 through fall of C1) while P2 is a half-cycle path (must propagate during second C2 pulse). If P2 fails to meet half-cycle timing, we can steal time from P1 if P1 propagates in less than a full cycle. If P1 propagates in say, 0.8 cycles, then the data held in the middle (C1) latch will be valid before the fall of C1. This can be used to get an early start on P2 and thus close P2's timing. When cycle stealing is used, testing the individual paths separately does not suffice to prove that the combined path will operate correctly at speed. An at-speed test for P1 proves that P1 meets full-cycle timing. A test would have to be applied at 20% beyond functional speed to validate the 0.8 cycle timing in this example. Testing the combined path at speed is a simpler way of validating the timing for these paths.

3.3 Correlating Structural with Functional Frequencies

Our experiment targeted the same MPC7455 microprocessor, with statistics shown in Table 2.1. Again, we used similar overall DFT strategy which was described in [1][68][69].

Different types of scan tests were generated by several ATPG tools. A simple set of transition test patterns which did not go through memories had a fault coverage of 70% with around 13k patterns. A second, complex set of transition test patterns which went through memories had a fault coverage of 78% with around 12k patterns. Since the value of paths going through test-only logic is highly controversial in the DFT community [13], we eliminated these paths from these transition test patterns. The test-only paths do not need to operate at the maximum chip frequency.

We ran STA and selected the top 2490 critical timing paths. 1463 of them were core latch-to-latch delay paths, 91 of them were memory paths, 230 were cycle-stealing paths. The rest of the 700 or so paths were miscellaneous paths including those in the clock regen or precharge logic, which were not suitable for delay test.

The path delay test coverage numbers are shown in Table 3.1. The third column, *path cov.*, was defined as the number of robustly and non-robustly detected paths over the total number of paths under consideration (PUC). The fourth column, *Test eff.*, was defined as the number of detected or undetectable paths over the total number of PUC.

Path types	Paths	Path cov.	Test eff.
Latch to latch	1463	60%	96.7%
Memory	91	95%	100%
Cycle stealing	230	63%	100%

Table 3.1: Path delay coverage nums

3.3.1 Sample Module Results

14 production modules were used as an initial sample for this study. The maximum frequencies of the various test types (functional, transition, etc) were measured. The measurements of the frequencies were carried out through shmooing on an Agilent 93000 tester. The parts were tested at 1.3v under room temperature. We normalized all the frequencies using the corresponding functional frequencies of the modules. The normalized maximum frequencies of each module measured using different tests were plotted in Fig. 3.3 through Fig. 3.5.

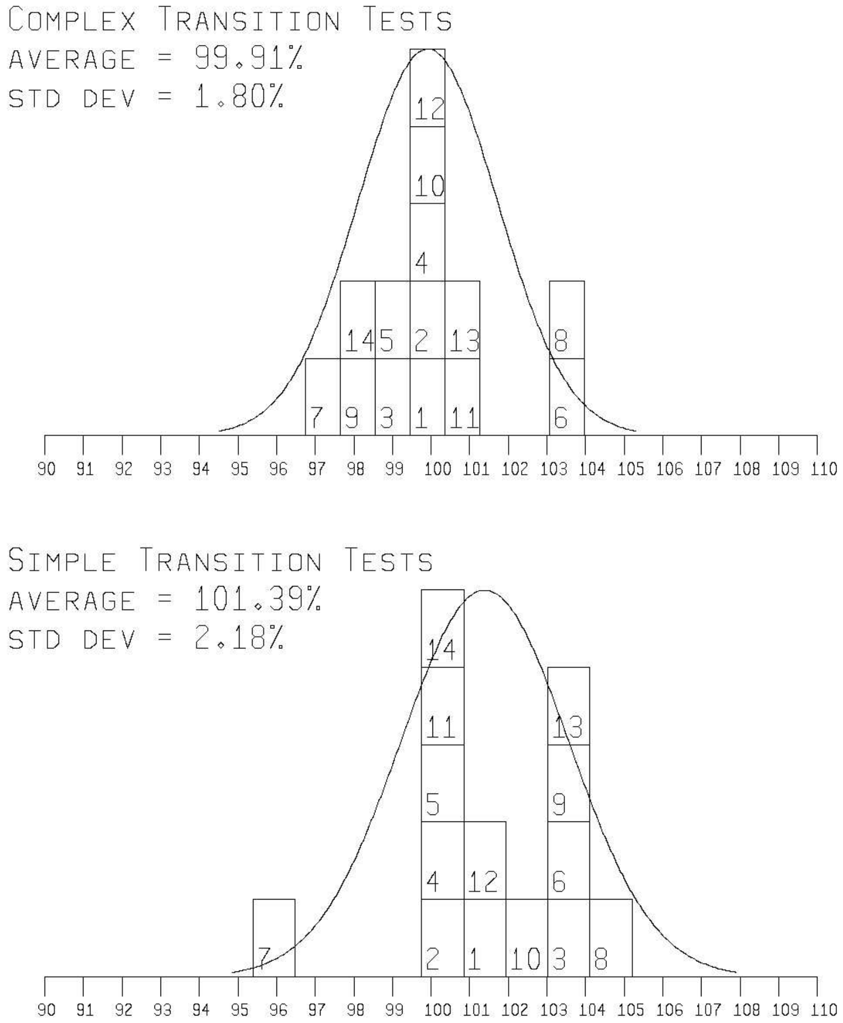


Figure 3.3: Transition Test Frequency Distribution

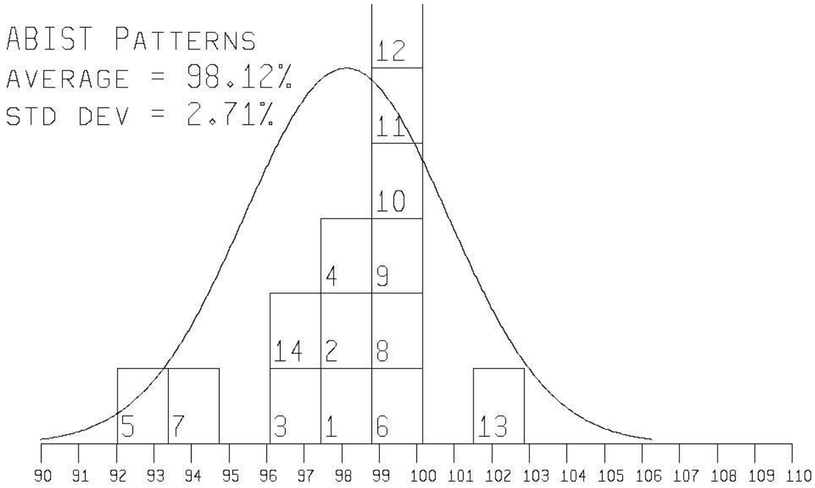


Figure 3.4: ABIST Frequency Distribution

The averages (*AVE*) and standard deviations (*STD DEV*) of the normalized frequency data were calculated and shown in Fig. 3.3 through Fig. 3.5. The corresponding Gaussian distribution curves were drawn. The x axis of the Gaussian curves were divided into speed bins based on their standard deviations. Here we used 0.5 of *STD DEV* as the width of the speed bins. For example, in Fig. 3.3, the complex transition tests showed a standard deviation of 1.80%. For these patterns, the bins would be 0.9% wide. We then plotted the histogram of the normalized frequency data of the modules. The numbers inside the boxes of the speed bin histogram were the module numbers in the respective bins. The complex transition tests which went through the memories provided the closest match to the functional frequencies, 99.91% on average with 1.8% standard deviation. The simple set of transition tests, which did not go through memories, on the other hand, provided speed binning results 1.39% higher than the functional results. This is understandable since the memories were associated

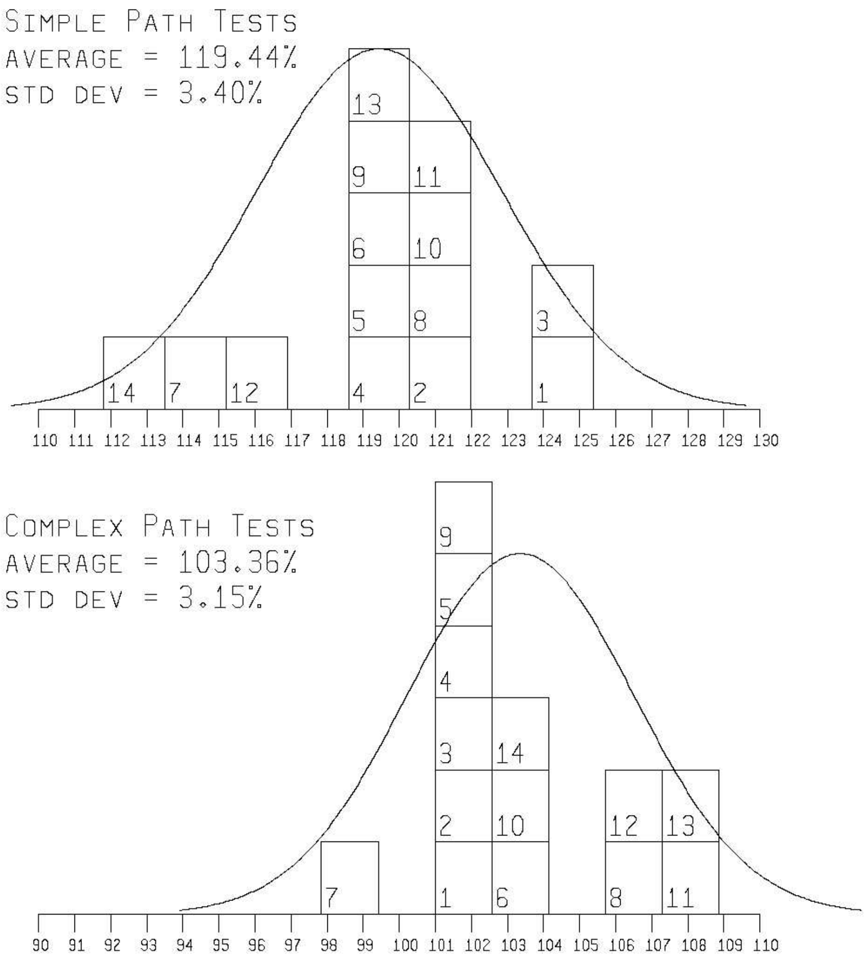


Figure 3.5: Path Test Frequency Distribution

with the most speed limiting paths. Note that these results are somewhat improved over the earlier results reported in [4]. ABIST tests provided speed binning results slightly lower and wider spread than those provided by the transition tests. This validated the fact that the on-chip memories gate the frequency of the chip.

The latch-to-latch path tests (Fig. 3.5) ran about 20% faster than the functional tests. This is understandable in that most of the critical paths are associated with the custom memory blocks. Clearly, generating path delay tests through the custom memory blocks provided a better indicator of module performance.

The complex path tests (memory paths plus cycle stealing paths) ran about 3% faster than the functional tests. Given the limited number of complex paths tested (145), these results seem to match the static timing analysis results.

3.3.2 Wafer Probe Results

A larger data sample was collected from various sites on seven wafers from a recent manufacturing lot. The data was collected from arbitrary sites during wafer test on a Teradyne J973 tester. The raw data from the tester showed a modest number of sites failing one or both sets of transition tests (the transition tests failed at the slowest speed applied). A small number of modules showed transition test results far slower than their functional test results. These are believed to be caused by delay defects (mostly gross-delay defects, a few small-delay defects). This underscores the need for AC testing in order to achieve acceptable quality levels. Data for these sites was discarded. Valid data was received for a total of 411 die.

Due to a data collection error, complex path test data was only available for 207 of these 411 die. Distribution plots of different structural frequencies normalized to the functional frequencies were shown in Fig.3.6 to Fig.3.9.

The complex transition tests still provided the closest match to the functional frequencies (99.01% on average). At wafer probe, these patterns ran 0.9% slower than the original 14 module sample (based on normalized averages). The results were more variable with a 3.06% standard deviation versus 1.80% from the module sample. The simple transition tests ran an average of 1.9% slower than functional tests (98.10%). This differed from the module results where these patterns ran 1.39% faster than the functional tests. The simple transition tests ran 3.29% slower at wafer than at module. The ABIST tests averaged 3.83% slower than functional test (compared with 1.88% from the module sample). The simple path tests ran much slower at wafer test. Where the module results had simple Pmax averaging 19.44% faster than functional test, the wafer data shows simple Pmax averaging 9.28% faster than functional. The complex path tests also ran much slower at wafer test - averaging 7.88% slower than functional tests (versus 3.36% faster in the module sample). Both types of path tests ran over 10% slower at wafer probe than they had in the original module sample. It can be observed that all of the test types ran slower (compared with Fmax) at wafer probe compared to the original 14 module sample. The complex transition tests were affected the least (losing less than 1%) while the path tests were hit the hardest (running 10% slower). Possible explanations for this include differences in hardware vintage, electrical environment, and thermal control. The wafer data was collected from newer, faster lot than was used in the original module sample. The electrical en-

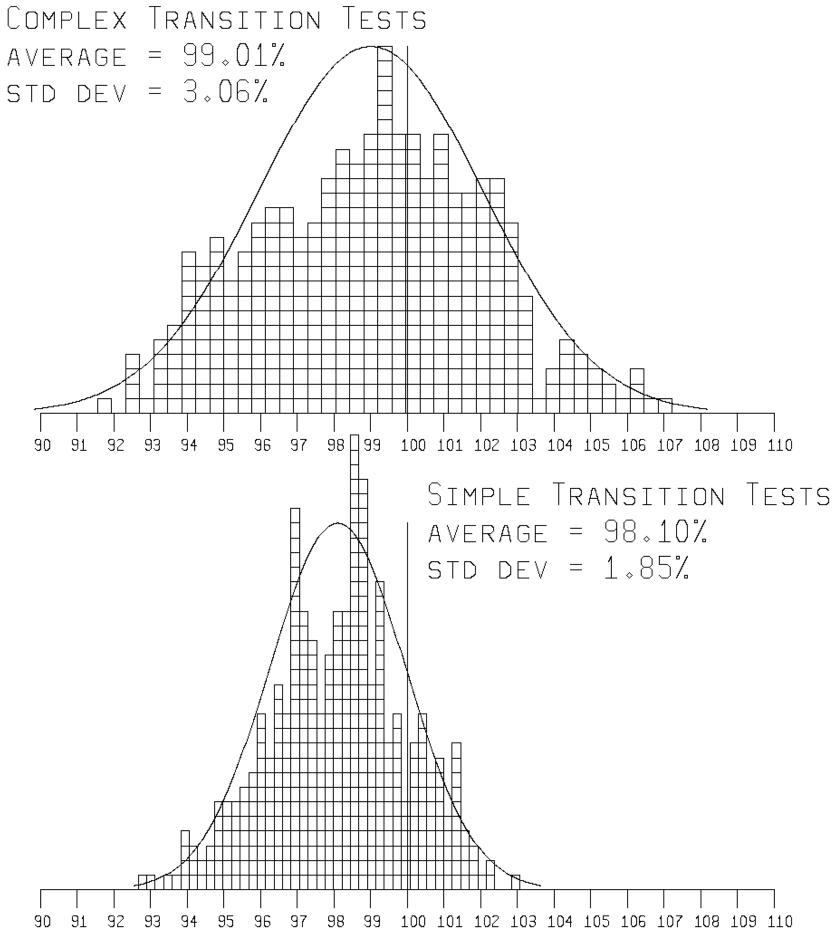


Figure 3.6: Transition Test Frequency Distribution

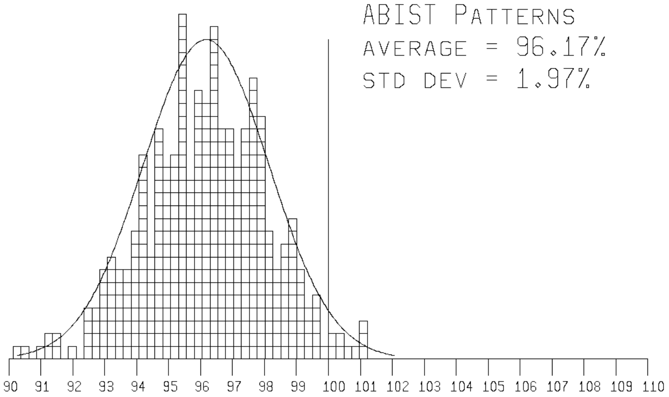


Figure 3.7: ABIST Frequency Distribution

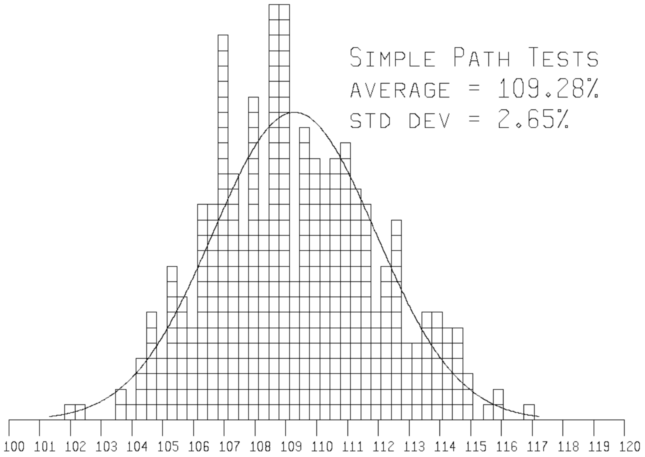


Figure 3.8: Simple Path Test Frequency Distribution

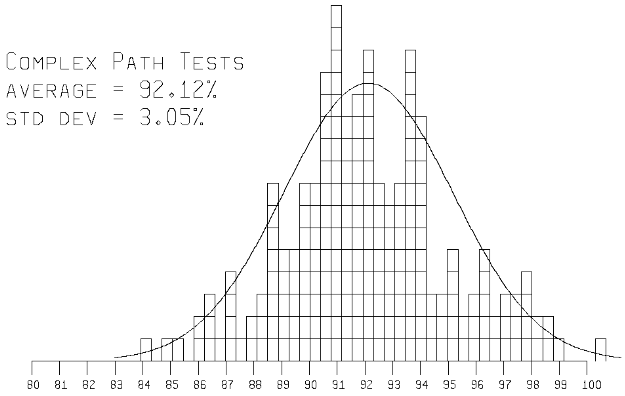


Figure 3.9: Complex Path Test Frequency Distribution

vironment is not as robust at wafer test as it is for module test. There are also differences in cooling between wafer and module test.

3.3.3 Validation of Structural Tests

Fail data analysis could be carried out for parts/dies for which structural tests provided different frequency predictions than the functional test. We carried out fail data analysis to see whether the structural tests caught additional at-speed defects which were missed by functional tests. In other words, whether structural tests were needed in order to achieve better at-speed quality levels.

We saw from the data of normalized structural frequencies for out sample module results, that part 7 was consistently predicted slow by all the structural tests. Fail data analysis was carried out for part 7. The speed limiting complex transition test for part 7 exercised BTIC logic which was normally frequency limiting.

Analysis of fail data of many dies on wafer probe was carried out where the frequencies of complex transition tests were noticeably slower than that of the functional tests. A few speed limiting critical paths were identified related to the memory transactions of the processor. With the process variations among all the dies used in our study, these few speed liming paths gated the speed of over 88% of them. This demonstrated the effectiveness of our complex transition tests, which included speed limiting critical paths along with other tests which provide a good coverage of the entire design.

3.4 Correlation of Structural with Functional Tests for Speed Binning

Consider the simplest situation of speed binning using only 2 speed bins: *slow* and *fast* bins, which are divided by cut-off frequency f. Ideally, all the parts with frequencies faster than f can be placed correctly in the *fast* bin, while all the parts with frequencies slower than f can be placed correctly in the *slow* bin. Parts in the *fast* bin are sold for x amount of money more per part than the parts in the *slow* bin.

In reality, it is possible for speed binning algorithm replying on a particular type of test to misplace parts with frequencies faster than f into the *slow* bin, while on the other hand, parts with frequencies slower than f into the *fast* bin. Fast parts placed in the *slow* bin, called *under* parts, are sold for x amount of money less than what they would have sold for. This resulted in loss in profit. The slow parts placed in the *fast* bin, called *over* parts, are sold for more money, but pose a potentially more serious problem of parts being returned by customers because they ran at slower than rated frequency. To avoid the more serious problem of customer returns, it is unavoidable to have some *under* parts. This is shown in Fig. 3.10. In fact, a guardband is set to place all the *over* parts along with additional *fast* parts into the *slow* bin. Essentially the guardband is a hedge against speed binning errors.

As seen in Fig. 3.10, the tighter the guardband, the fewer *fast* parts are under sold in the *slow* bin. As a result, a smaller profit margin is lost. For example, if one million parts meet the minimum speed requirement and are being speed binned before shipment. If the guardband can be reduced from 3% to 2%, as a result, approximately 8% of the parts for shipment moved into the next higher speed bin commanding an estimated $50 higher price. This gives an average of $4 per part, a total of $4 million. We can see that the tightening of guardband brings huge financial gains of a product.

3.4.1 Sample Module Results

We compared the speed binning data of the structural tests with the functional tests. Based on 2 speed bins among the 14 parts, their distribution was shown in Fig.3.11. In Fig.3.11, the ticks on x-axis (frequency axis) were 10MHz apart. The frequency space between two nearby ticks on the frequency axis was considered a speed bin, which we would use to plot our frequency data for different parts. The production frequencies for the 2 bins were fs and ff, where fs is for slow frequency and ff is for fast frequency. They were decided by the customers. The guard band for fs was 30Mhz, for ff was 36Mhz. These were set by the functional tests. We used them for all the tests under consideration. Based on the bin frequencies and the corresponding guard bands, the parts which fell in each bin by

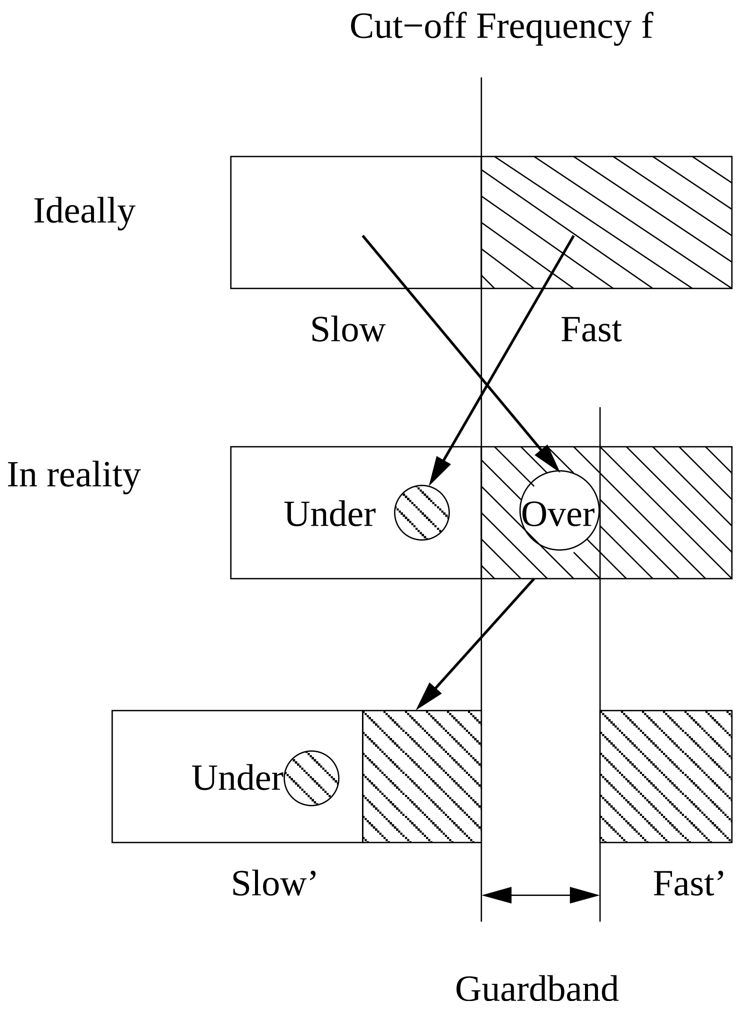


Figure 3.10: Illustration of Guardband

different tests were shown in the Fig. 3.11. According to the functional tests, there were 7 parts in *bin1*, 7 in *bin2*. Notice the guard bands resulted in pessimistic frequency predictions. That was the reason why parts 11, 12, 14, which were within the guard band above ff, ended up in the slow bin.

Among all the tests, we see that ABIST tests were pessimistic and ended up putting more parts in the lower bin. The complex transition tests had the same speed binning results as the functional tests. It is interesting to note that even parts 7, 6, 8 which were outside the region of $3STD\ DEV$ from the AVE in Fig. 3.6, the final binning results was not affected. The more optimistic simple transition tests placed only part 12 differently from the functional tests by predicting its frequency above the guard band of the faster bin. We scaled the frequency axis of latch to latch tests by 1.2 based on the ratio of its AVE and functional test AVE shown in Fig. 3.8. *fs*, *ff* and the corresponding guard bands are scaled. The speed binning result using scaled frequency for latch to latch tests was different from that using functional test in: a) part 13 failed both bins and would be thrown away; b) part 11 was placed in the faster bin. On the other hand, when we used complex path tests for binning, all the parts fell into *bin2*. Scaling speed bins for the complex path tests did not make any difference. We saw even complex path tests gave better frequency prediction for each individual part than the latch to latch path tests, they gave worse speed binning results if we scaled the speed bins for latch to latch path tests.

3.4.2 Wafer Probe Results

To assess the suitability of the various structural tests for use in speed binning, the wafer test data was used to speed bin the 411 die into one of two bins (fast or slow). We will experiment with different settings of speed bins. First the cut-off between slow and fast die was defined arbitrarily as the average of the measured Fmax of the dies. For example, if the average measured Fmax value was 1030 Mhz, parts at or above 1030 Mhz would be defined as fast while parts below 1030 Mhz would be defined as slow. In practice, parts right around 1030 Mhz could be offered for sale as operating at 1.0 Ghz - reserving a 3% guardband. The following tables give the speed binning results for each of the test types. The 411 dies used yield 232 "fast" parts and 179 "slow" parts based on the functional test and the 3% guard band. Table 3.2 uses the average Fmax values as the cut-off between slow and fast bins for all the structural tests. Table 3.3 uses the average of the corresponding structural test to assign a cut-off for each particular test type.

The *Under* column lists the percentage of the 411 parts incorrectly assigned to the "slow" bin based on a particular test type ("fast" part errantly classified as "slow"). The *Over* column lists the percentage of the 411 parts incorrectly assigned to the "fast" bin. The *GB* column lists the guardband that would be required in order for all of the parts in the *Over* column to operate at their offered speed determined by the functional tests.

It can be seen that several of the test types produced moderately good results (on the order of 5% each *under* and *over* with a 3% guardband sufficient to protect against errors). Only the complex transition tests produced acceptable results using the average functional frequency to pick

Test Type	Under	Over	GB
Complex Transition	**7.5%**	**5.4%**	**1.8%**
Simple Transition	18.7%	0.8%	1.1%
ABIST	34.8%	0.5%	1.8%
Simple Path Delay	0%	35.3%	10.3%
Complex Path	55.6%	0%	0%

Table 3.2: Speed Binning Results with the Average of the Measured Fmax

Test Type	Under	Over	GB
Complex Transition	4.4%	6.6%	2.2%
Simple Transition	3.2%	6.1%	2.2%
ABIST	3.9%	5.4%	2.2%
Simple Path Delay	5.8%	7.3%	6.4%
Complex Path Delay	**1.9%**	**4.8%**	**2.2%**

Table 3.3: Speed Binning Results with the Average of the Corresponding Structural Tests

the frequency cut-off. The simple path tests were the only tests that did not produce acceptable results even when the average frequency of simple path test was used. The complex path tests (with the average frequency of complex path test as the cut-off frequency) produced the best results.

3.5 Conclusions

We show that good correlation can be achieved between the structural frequencies and functional frequency. Complex transition tests which go through the memory arrays, not only provide speed limiting paths, but also provide tests with good structural coverage of the entire IC. These complex transition tests can catch additional speed failures which are missed by the functional tests. Good speed binning results can be achieved using scan delay test patterns. For both the transition tests and path delay tests, testing through the custom memory arrays provided better results. Our results demonstrate that the usefulness of structural tests extends beyond defect detection, and into the characterization of the speed prediction of high performance ICs.

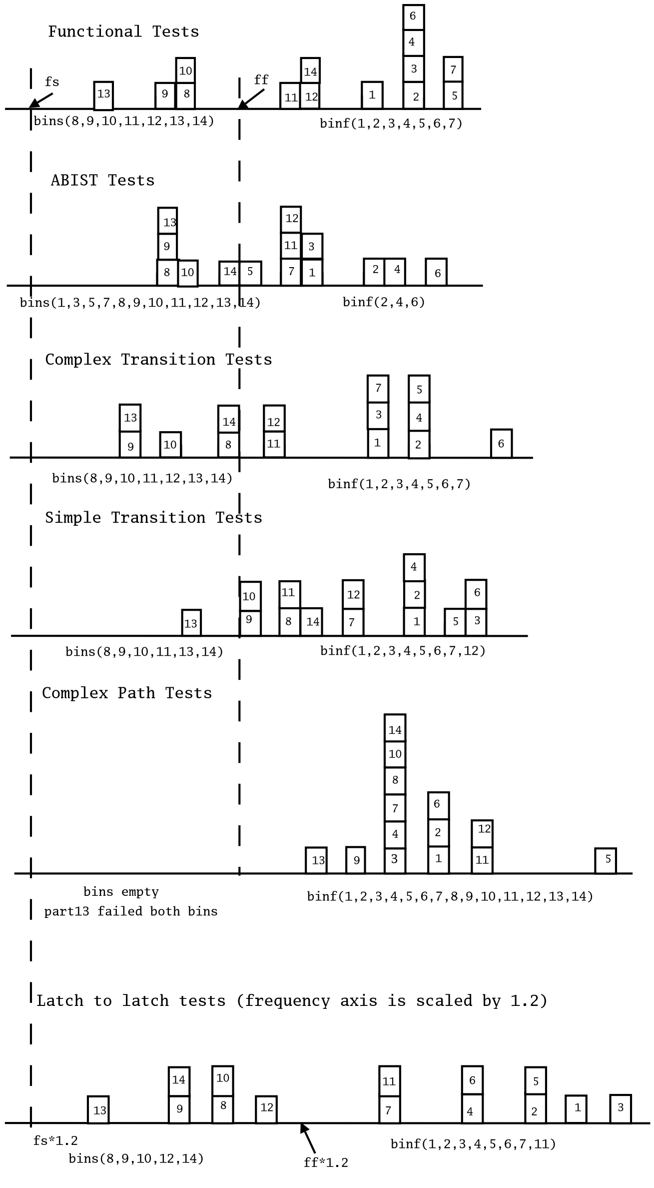


Figure 3.11: Speed Binning Data

Chapter 4

Correlation between the Pre-Silicon Path Prediction and Post-Silicon Path Measurement

In order for STA to provide a good guidance for the selection of critical paths for path delay test generation during the post-silicon testing, it is important that the STA prediction of critical timing paths correlate with the actual silicon measurement.

In general in a pre-silicon design environment, timing models used for STA may not be correct and 100% complete due to many contributing factors such as clock skew, process variations, inaccurate interconnect delay models, impact of noise on delay, supply voltage drop, etc [24] [27] [56]. These factors affect the timing yield even more as the processing technology advances into the submicron region.

In addition, due to the complexity and cost issues, the standard industry design practice adds to the inaccuracy. For example, during manufacturing, silicon undergoes many iterations to improve the speed of the parts, like shrinkage of gate oxide and changing of transistor nodes. Often these changes are not fed back to the characterization and timing analysis process, as shown by the dashed line in Fig. 4.1, due to complexity and resource issues. As a result, the delay numbers used by STA, which heavily depend on the accurate characterization of the transistors and wire delay

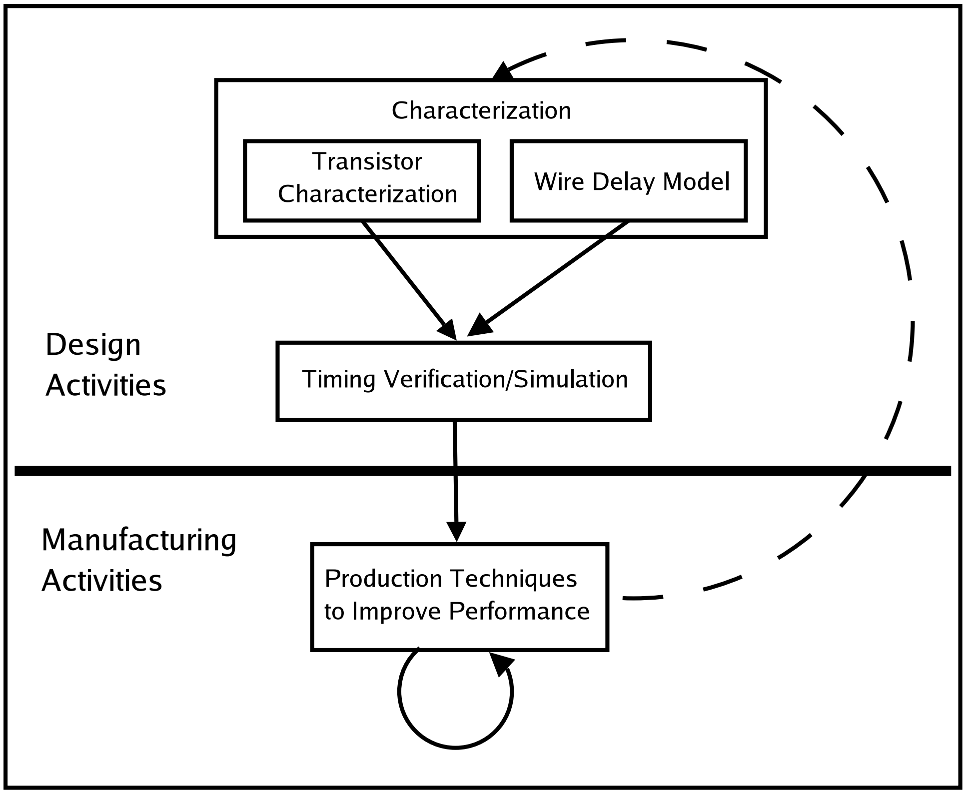


Figure 4.1: Deviation Between Timing Models and Silicons

models, are not likely to reflect those of real silicon. This would affect the validity of STA runs at all design corners. As a result, the correlation between the timing estimation of the pre-silicon timing models and the timing measurements of the actual silicons is expected to be poor in general. In order to predict the timing behavior of ICs accurately, a set of test chips are needed to figure out the timing path with the worst delay. In this thesis, we establish a methodology to study the correlation between the STA prediction of the critical timing paths and the post-silicon delay measurements of the same set of timing paths.

4.1 Methodology for Correlation Study

To study the correlation of pre-silicon and post-silicon timing paths, we use the following delay path selection and test generation methodology (Fig. 4.2).

- Run chip-level STA to identify n paths with the worst timing slack. n must be large enough (e.g. 2000-3000) to provide valid coverage.
- For each of the path P_i, i = 1,..., n, identified in the first step, perform path delay ATPG to generate a sequence of patterns T_i that can be used to load needed values into the scan chains, then to launch the right signal values that will sensitize a given path, and finally capture the response of the path into a latch that can be observed via scan.
- Each delay pattern is applied to each chip while varying the capture frequency to determine the maximum frequency at which the test will pass. Thus for every chip, we obtain a measured speed for each path P_i, for i = 1, ..., n.
- Compare the predicted speed of Pi versus the average measured speed of Pi for every chip, for i = 1, ..., n.

4.2 Study the Correlation between the Pre-silicon Path Prediction and Post-silicon Path Measurement

We carried out our experiment of correlation study using the same MPC7455 microprocessor, with statistics shown in Table 2.1. Similar

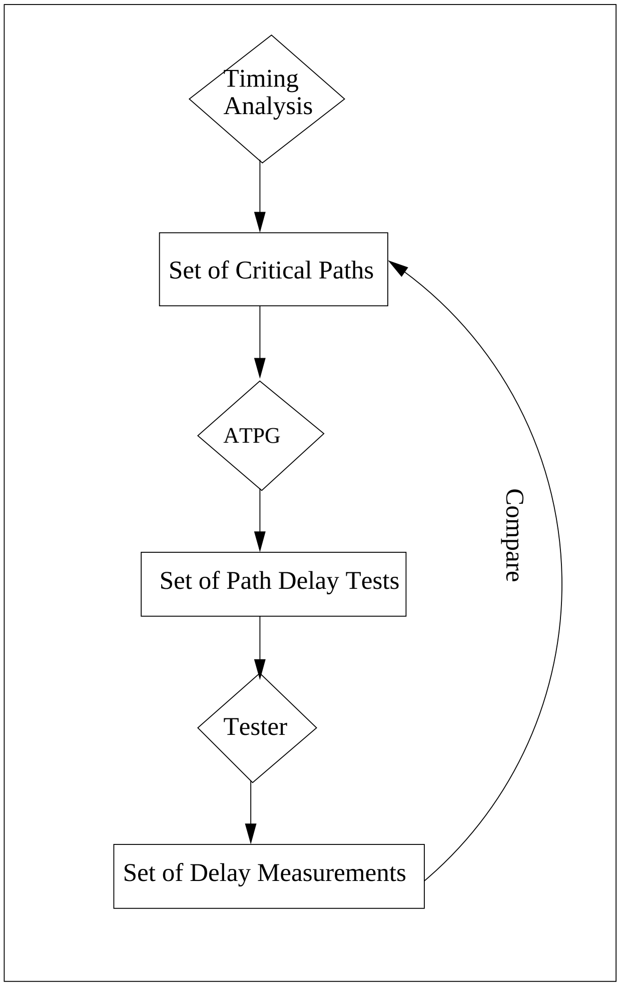


Figure 4.2: Correlation Study Methodology

overall DFT strategy which was described in [1][68][69] was used for path delay testing of timing critical paths predicted by STA.

STA was used to generate a set of critical timing paths (Fig. 4.2). In general, STA could be run at different design corners with different parameters. We ran STA using nominal voltage and temperature conditions. Also nominal pin-to-pin delays of library cells were assumed for the run. After ATPG, the same path delay set as in Table 3.1 was generated. We came up with 878 simple path delay tests and 231 complex path delay tests [82][83].

Our experiment for correlation study was performed using 254 parts from a recent production lot. Every path delay test was run individually on every part. For each path delay test, the capture frequency was varied to determine the shortest cycle time at which the test would pass. This was repeated for each part. We collected data for the top 556 simple delay paths and 206 complex delay paths for analysis. We could not collect the data for all the path delay tests generated because many of them ran faster than the maximum PLL frequency when run individually [77]. We presented our data and data analysis based on the following series of questions they answered.

4.2.1 Do the Measured and STA Predicted Speed Data Correlate?

To see the overall correlation between the STA prediction and the measured speed data from the tester for all the paths, the measured speed data from the tester is recorded and the average from different parts is used as an indication of the speed of the path. The correlation data is

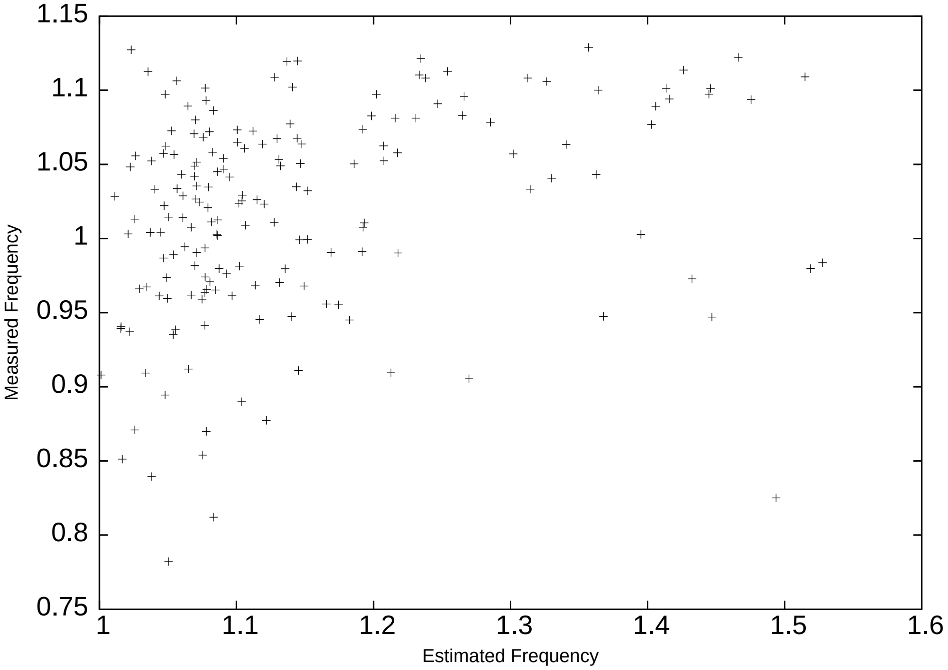


Figure 4.3: Correlation between Measured and STA Predicted Speed Data

shown in Fig. 4.3. We see the correlation is generally poor. Note that both frequency axises have been augmented.

4.2.2 Do the Measured and STA Predicted Path Ranks Correlate?

We will examine the correlation between the path ranks based on STA, and those based on post-silicon measurements. At the pre-silicon side, the ranks for paths can be inferred from STA. We will only consider the path ranks of delay paths suited for delay testing. As a result, the path ranks for our delay paths are different than those in the STA.

For example, consider the case when STA contains a total of 10 paths, where paths 1, 2, 3, 4, 6, 7, 8 can be tested through scan delay tests. The rest of the paths 5, 9, 10 are not suited for delay testing,like the paths in the clock regen logic. The final ranks for the 7 delay paths under our consideration will be 1, 2, 3, 4, 5, 6, 7. We call the path ranks predicted by STA *old* ranks.

On the other hand, we rank the post-silicon paths based on their delay measurements from the tester. We call the resulting path ranks based on the delay measurements *new* ranks.

To check the correlation distribution of the path ranks between the STA predicted and the post-silicon measured ones, concordance coefficients are calculated for the delay paths including the simple and complex delay paths.

The correlation of the two path ranks will be measured based on the *coefficient of concordance* [25]. Given a set of k-tuples (r_{11}, r_{21},...,r_{k1}), (r_{12}, r_{22},..., r_{k2}),..., and (r_{1n},r_{2n},...,r_{kn}), which represents the k ranks assigned to the n paths as a result of k different ranking experiments. Hence, r_{ij} represents the rank assigned to the j^{th} path during experiment i. The *coefficient of concordance* (W) is a measure of the extent of association or agreement of various k-tuples. W is defined as:

$$W = 12/k^2n(n^2-1) * \sum_{j=1}^{n}[(R_j - k*(n+1)/2)^2]$$

where R_j is the sum of the ranks assigned to r_{1j}, r_{2j},..., r_{kj}. When $W = 0$, there is no correlation between the two ranks. When $W = 1$, complete correlation exists between the ranks. In our case, $k = 2$ corresponding to the two different ways of ranking the paths. For each part, the difference between the two different path ranks would be path dependent. Concordance coefficients can be calculated considering all the paths of each part to show the overall correlation for the part.

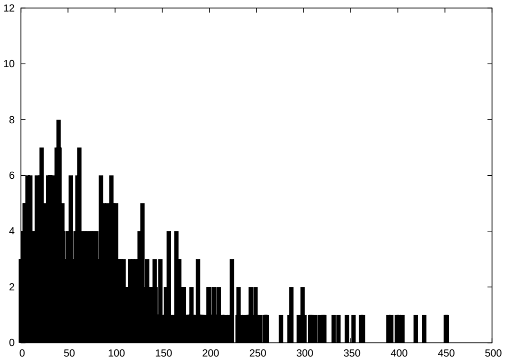
We also refer to *coefficient of concordance* as *correlation coefficients*.

The distribution of the difference between the predicted and measured path ranks can be investigated based on individual parts.

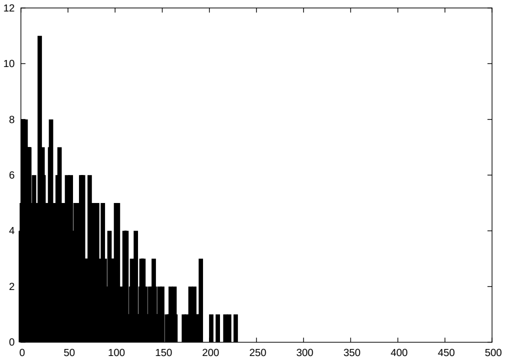
We treat rank difference as bins and plot the occurrences of paths in them. Fig. 4.4(a), Fig. 4.5(a) show the occurrences of paths vs the corresponding rank differences for two different parts. We see the distribution of Part 1 peaked around 40, while that of Part 2 have several peaks. Part 1 have better concordance coefficient (0.82) than that of Part 2 (0.63).

Note that paths with the biggest difference between the old and new path ranks are at the tail end of the histogram distributions in Fig. 4.4(a) and Fig. 4.5(a). We expect if we remove a given number of paths with the biggest difference between the old and new path ranks from the calculation of coefficients, the values of the concordance coefficients of the parts will improve. Also when we remove a given number of paths from the coefficient calculation, both the old and new ranks of the rest of the paths can also be affected. The rationale is if a path was removed, then the paths which are less critical than the removed path, become more critical upon the removal. Thus the overall histogram distribution can change.

We show in Fig. 4.4(b) and Fig. 4.5(b), the new histogram of rank differences if we remove the top 70 paths (about 10% out of 762 paths including both simple and complex delay paths) for the same two parts. We see the correlation between the two different path ranks increase significantly for both parts, especially for Part 1. After removing the top 70 paths with the largest rank differences, the histogram for Part 1 becomes



(a) Before Path Removal



(b) After Path Removal

Figure 4.4: Histogram of Rank Differences for Part 1

tighter. This indicates that there are significant differences in ranks for a small group of paths, which are largely responsible for the rank difference for the rest of the paths. Once this small group of paths is removed, the correlation of the path ranks for the rest of the paths improves greatly. In comparison, Part 2 shows worse overall correlation between the 2 types of path ranks since more paths correlate poorly.

4.2.3 What Do the Correlation Distribution between Measured and STA Predicted Path Ranks Look Like?

The concordance coefficients are calculated for all the parts. We divide up the range of the concordance coefficients into bins. We show the number of parts in different bins in Fig. 4.6(a).

The distribution of concordance coefficients for all the delay paths peak around 0.74. This show that despite the differences between the timing models and the real silicon behaviors, correlations exist between the path ranks predicted by STA and those measured using silicon.

The concordance coefficients are recalculated, after the top 70 paths with the largest path rank difference are removed for all the parts, and are shown in Fig. 4.6(b).

Compare the results in Fig. 4.6(b) to those in Fig. 4.6(a), the distribution peak shift about 0.1, which is a significant improvement for the concordance coefficient as a measurement of correlation.

Even though STA is run under voltage v_1, the frequency measurements are carried out under 3 different voltage design corners: v_0, v_1, v_2 for the delay paths in our experiment. To avoid redundancy, we show only the concordance coefficient distributions under the voltages v_0 and v_2 after the removal of the top 10% of paths with the largest path rank differences in Fig. 4.7(c) and Fig. 4.7(d).

The averages of the coefficient distributions are given in the sub captions of in Fig. 4.6 and Fig. 4.7. We see after the path removal, the averages are 0.83 for both voltages v_1 and v_2, 0.84 for v_0. It is interesting to note that the correlation of predicted and measured path ranks is slightly better under voltage v_0 even though the STA prediction is made with voltage v_1.

4.2.4 Are the Parts with Better Correlation Coefficients Faster Parts?

We divide up the whole region of the concordance coefficients into 3 sections.

- Low correlation section: the values of concordance coefficients are smaller than 0.7.
- Medium correlation section: the values of concordance coefficients are greater than or equal to 0.7, but smaller than 0.74.
- High correlation section: the values of concordance coefficients are greater than 0.75.

We call the parts with concordance coefficients in the *low* correlation section *lcc* parts. We name *mcc* and *hcc* parts similarly.

One interesting observation is *hcc* parts are not necessarily the faster ones out of all the parts. By the same token, the *lcc* parts are not necessarily the slower ones. In Fig. 4.8 and Fig. 4.9, we plot the histogram over normalized functional frequencies. The functional frequencies are normalized against the maximum functional frequency. The histogram show the distribution of the *hcc*, *mcc* and *lcc* parts in the normalized functional frequency bins.

Based on work in [82][83], two speed bins are formed based on the average of the functional frequency (0.92). The parts which are faster than 0.92 form the *faster* speed bin, while those slower form the *slower* speed bin. The breakdown of the above 3 category of parts in the 2 speed bins is shown in Table 4.1. We see from Table 4.1, that a larger percentage of *lcc* parts go into the faster speed bin.

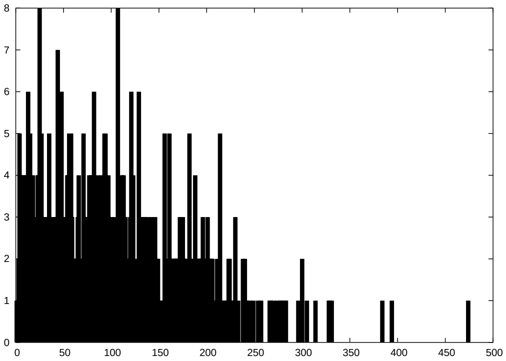
parts	Portion of parts in the faster bin
lcc	66.7%
mcc	56.9%
hcc	32.7%

Table 4.1: Breakdown of Parts in Different Speed Bins

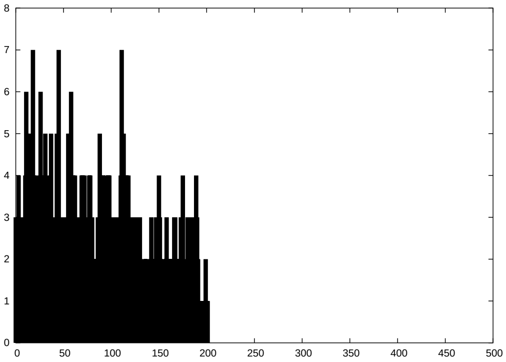
4.3 Conclusions

We establish a methodology for carrying out our study of correlation between the pre-silicon timing prediction and post-silicon timing measurement of the same set of critical timing paths. We call our methodology the *methodology of Correlation Study.* Our experimental data allows us to gain a general understanding regarding the correlation between the predicted and measured timing data.

Our correlation study methodology includes all the different types of delay paths which help to reflect the impact of different process options on different types of logic. Based on our methodology of correlation study, further research can be carried out to study the impact of different effects on timing paths, including clock skews, process variations, coupling noise etc. The resulting information can be utilized to improve a timing verification methodology.

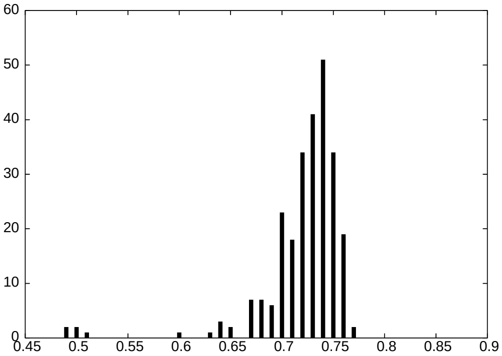


(a) Before Path Removal

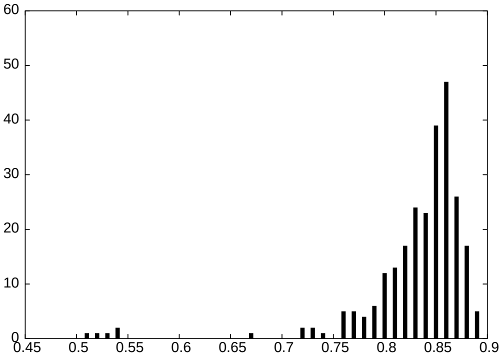


(b) After Path Removal

Figure 4.5: Histogram of Rank Differences for Part 2

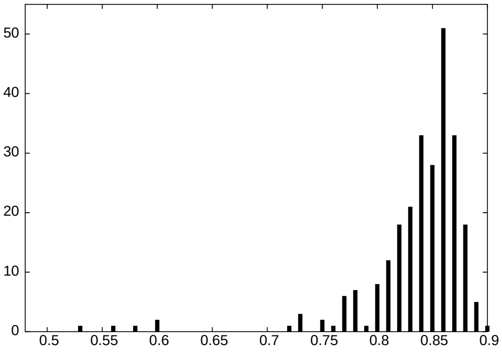


(a) Before Path Removal (Voltage: v_1, ave = 0.73)

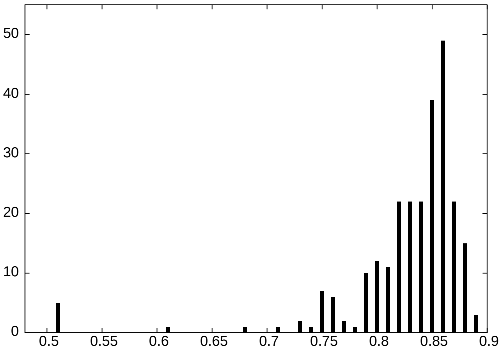


(b) After Path Removal (Voltage: v_1, ave = 0.83)

Figure 4.6: Histogram of Concordance Coefficients for the Delay Paths (Voltage: v_1)

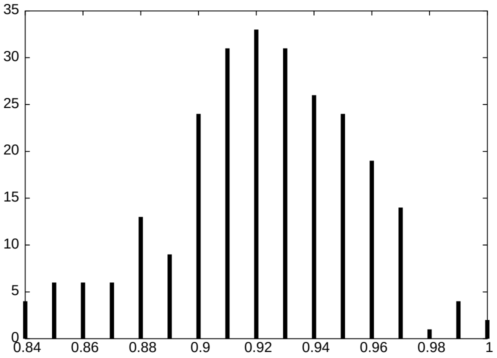


(a) After path removal (Voltage: v_0, ave = 0.84)

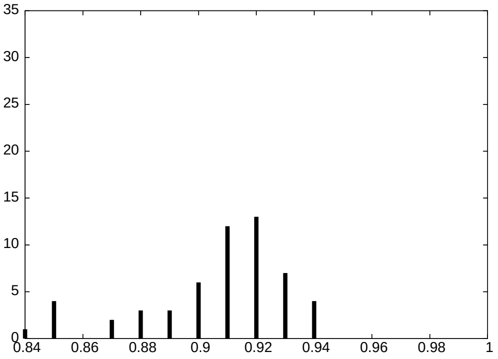


(b) After path removal (Voltage: v_2, ave = 0.83)

Figure 4.7: Histogram of Concordance Coefficients for the Delay Paths (Voltage: v_0, v_2)

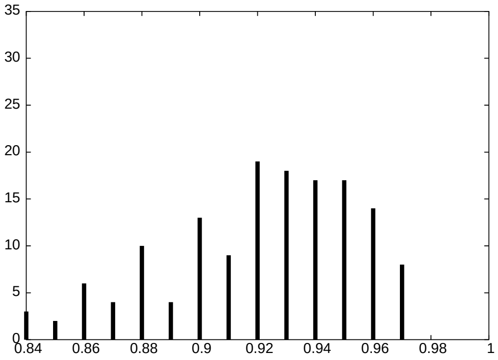


(a) All the Parts

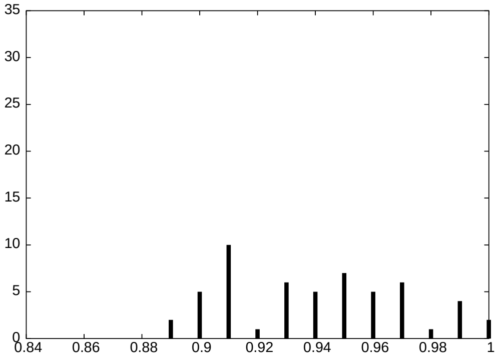


(b) *hcc* Parts

Figure 4.8: Histogram of Parts in Normalized Functional Frequency Bins



(a) *mcc* Parts



(b) *lcc* Parts

Figure 4.9: Histogram of Parts in Normalized Functional Frequency Bins

Chapter 5

Conclusions and Future Research

This dissertation addresses issues in timing verification and delay testing methodologies. An efficient technique utilizing ATPG and timing information to identify false timing paths in the timing verification methodology is presented. In addition, it seeks to quantify the correlation of timing path ranks between the timing verification prediction and delay fault testing measurement. This dissertation also presents a speed binning methodology utilizing structural delay tests.

5.1 Future Research Directions

Improvement of False Timing Path Identification Method. Currently we have demonstrated how to identify combinational false timing paths using ATPG techniques and delay information. We can further improve our techniques in the following manner.

1. **Sequential False Timing Path Identification.** Since static timing analysis does not concern itself with functional information at all, many timing paths identified by a static timing analyzer may be false sequentially even though they might be active combinationally. Further techniques need to be developed to efficiently identify sequentially false timing paths.

2. **Characterization of Custom Circuits.** Due to the complexity and wide data bus inputs of custom circuits, it is difficult to determine the worst case delay between any input/output pair of a custom circuit. Designers generally decide the vector which activates the worst case delay for a custom circuit between one of its input/output pairs based on their intuition, which may not be accurate. As a result, the worst timing delay between an input/output pair may not be identified for accurate timing verification.

   Also, the characterization process allows only recording worst case pin to pin delays of blocks, but not necessarily the activation vectors associated with them. This makes it harder to determine whether a timing path is false if it contains more than one logic paths. Conservative approach is taken at this point so that such a timing path will not be identified false unless every logic path associated with it is false.

   It is desirable to have a systematic way of estimating the worst case delay between any input/output pair of a custom circuit accurately along with the activation vector. This can expose any shortcomings of the current custom circuit characterization process and improve the effectiveness of our false path identification technique.

Incorporation of False Timing Path Identification Method in a Timing/Noise Methodology. Algorithms have been developed to remove false timing paths once they are identified [8]. Based on our algorithms, false timing paths identification techniques can be incorporated into the timing verification flow as indicated in Figure.5.1.

Currently in industry, coupled noise analysis also largely depends on proximal structural information to divide up the nets into **aggressor** and **victim** classes. A victim net is a net on which noise is injected by one or more neighboring nets, i.e. aggressor nets, through cross-coupled capacitances. The noise may propagate to a state element, changing its state and causing a functional failure.

If the path sections between a victim net and some of its aggressors are false, the noise cannot be propagated from the aggressors to a victim net and can thus be ignored. This reduces the pessimism of noise analysis through the use of logic correlations in the circuit. BDD-based algorithms have been attempted in [42]. ATPG techniques and delay information can also be applied to the noise analysis methodology.

Study of the Impact of Different Effects on Timing Paths. Currently, the critical timing paths predicted by STA are generally not the ones which gate the frequencies of the actual silicons. This discrepancy is due to many deep sub-micron effects like process variations, noise, supply voltage drop, as well as issues in design methodology which fail to capture process options of manufacturing. Our correlation study methodology provides a unique mechanism to study the impact of different effects and process options on timing paths.

For example, we could measure the different delays of the same timing path among the different dies within a wafer and among dies from the same locations of different wafers. This allows us to study the effect of process variations among all the dies within a wafer, and among different wafers within a lot.

Path Rank Correlation Based Path Delay Testing Methodology. Even though the exact measured delays of the timing paths are generally not the same as predicted by STA, if the measured timing path ranks are in 100% correlation with the predicted ones, then all the timing paths are affected proportionally. In this case, the worst timing path measured would be the same as the worst timing path predicted. Thus we only need to include the worst timing path predicted by STA for path delay testing in an ideal situation.

As the correlation between the path ranks of timing paths predicted by STA and those measured on silicons gets worse, it is understandable that more timing paths predicted by STA are needed for path delay testing in order to have the most critical timing path on silicon included in the test with certain confidence level. Sample parts can be used to develop an analytical model, which predicts the number of top timing paths needed for path delay testing to cover the most critical timing path on silicon during the test, based on the correlation between the predicted and measured path ranks. This model can then be used to provide guidance to the final delay testing during manufacturing.

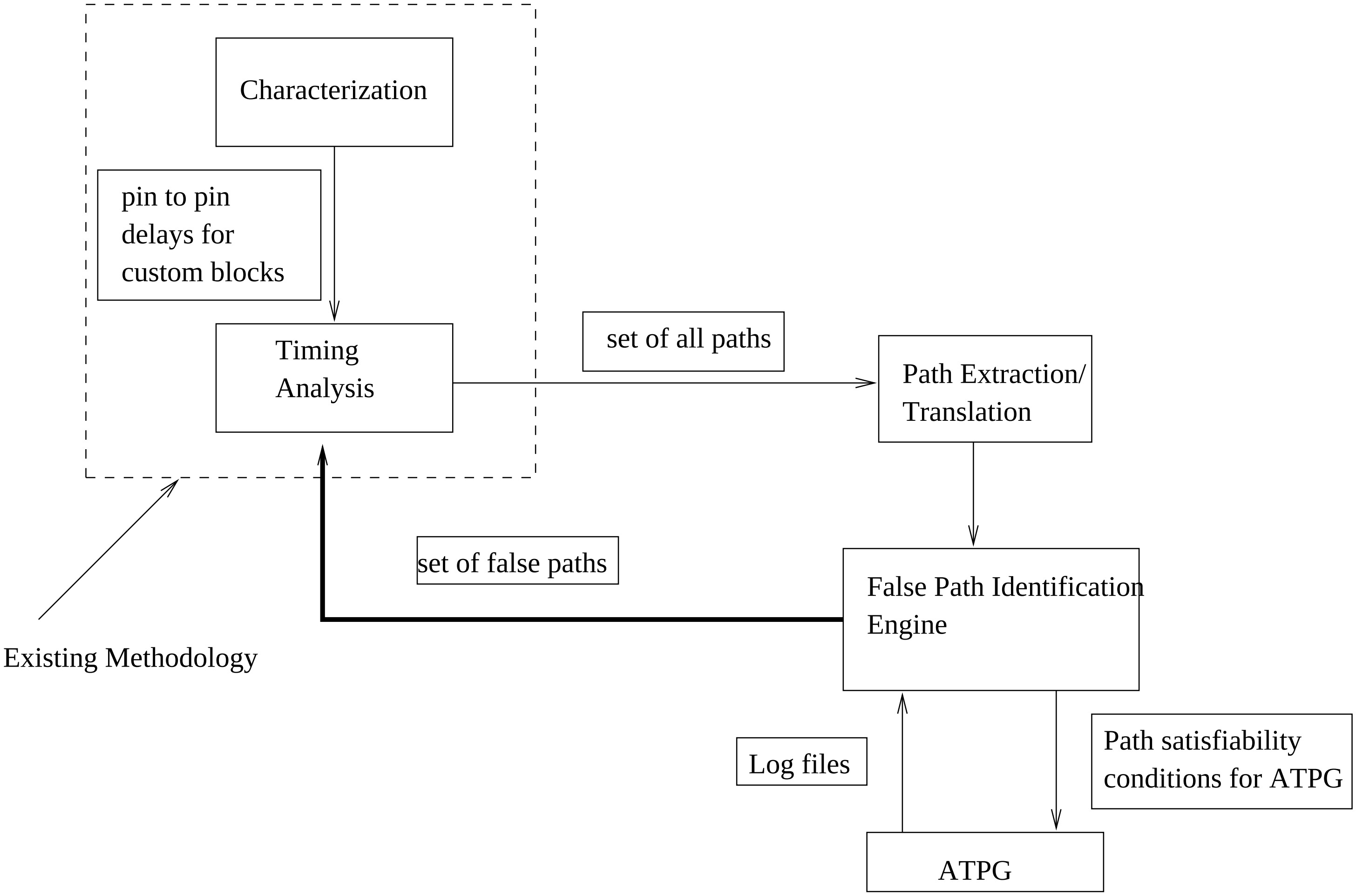


Figure 5.1: New Static Timing Verification Methodology

Bibliography

[1] M. Abadir, R. Raina. "Design-For-Test Methodology for Motorola PowerPC Microprocessors" in *Proceedings of International Test Conference*, pp.810-819, 1999.

[2] A. Agarwal, D. Blaauw and V. Zolotiv. "Statistical Timing Analysis for Intra-die Process Variations with Spatial Correlations" in *Proceedings of International Conference on Computer-Aided Design*, pp.900-907, 2003.

[3] G. Aldrich and B. Cory. "Improving Test Quality and Reducing Escapes" in *Proc. Fabless Forum*, Fabless Semiconductor Assoc., pp.34-35, 2003.

[4] D. Belete, A. Razdan, W. Schwarz, R. Raina, C. Hawkins and J. Morehead. "Use of DFT Techniques In Speed Grading a 1GHz+ Microprocessor", in *Proceedings of International Test Conference*, pp.1111-1119, Oct. 2001.

[5] K. P. Belkhale, A. J. Suess. "Timing Analysis with Known False Subgraphs" in *Proceedings of the International Conference on Computer-Aided Design*, pp.736-739, 1995.

[6] J. Benkoski, E. V. Meersch, L. J. M. Claesen, and H. De Man. "Timing Verification using Statically Sensitizable Paths" in *IEEE Trans. Computer-Aided Design*, vol.9, pp.1073-1084, Sept.1990.

[7] J. Bhadra, M. S. Abadir, J. A. Abraham. "A Quick and Inexpensive Method to Identify False Critical Paths Using ATPG Techniques: an Experiment with a PowerPC Microprocessor" in *Proceedings of IEEE 2000 Custom Integrated Circuits Conference*, pp.71-74, May. 2000.

[8] D. Blaauw, R. Panda and A. Das. "Removing User-Specified False Paths from Timing Graphs" in *Proceedings of 37th IEEE/ACM Design Automation Conference*, pp.270-273, 2000.

[9] M. A. Breuer, C. Gleeson and S. Gupta. "New Validation and Test Problems for High Performance Deep Sub-Micron VLSI Circuits" in *Tutorial Notes, IEEE VLSI Test Symposium*, April 1997.

[10] F. Brglez and H. Fujiwara. "A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortran", distributed on a tape to participants of the Special Session on ATPG and Fault Simulation, International Symposium on Circuits and Systems, June 1985.

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

[12] J. L. Carter, V. S. Iyengar, and B. K. Rosen. "Efficient Test Coverage Determination for Delay Faults" in *Proceedings of IEEE International Test Conference*, pp.418-427, Sep. 1987.

[13] T. J. Chakraborty and V. D. Agrawal. "Effective Path Selection for Delay Fault Testing of Sequential Circuits" in *Proceedings of International Test Conference*, pp.998-1003, 1997.

[14] S. T. Chakradhar, M. A. Iyer and V. D. Agrawal. "Energy Minimization Based Delay Testing" in *Proceedings of IEEE European Design Automation Conference*, pp.280-284, Sep. 1992.

[15] H. Chang. "Strategies for Design and Test of High Performance Systems", *Ph.D. Dissertation*, The University of Texas at Austin, August 1993.

[16] H. C. Chen, D. and H. C. Du. "Path Sensitization in Critical Path Problem" in *Proceedings of IEEE International Conference of Computer-Aided Design*, pp. 208-211, 1991.

[17] K. T. Cheng and H. C. Chen. “Delay Testing for Non-Robust Untestable Circuits” in *Proceedings of IEEE International Test Conference*, pp.954-961, Oct. 1993.

[18] K.-T. Cheng. “Transition Fault Testing for Sequential Circuits” in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, no.12, pp.1971-1983, Dec. 1993.

[19] B. Cory, R. Kapur and B. Underwood. ”Speed Binning with Path Delay Test in 150-nm Technology” in *IEEE Design & Test of Computers*, pp.41-45, Sep-Oct 2003.

[20] A. Crouch, J. Potter and J. Doege. “AC Scan Path Selection for Physical Debugging” in *IEEE Design & Test of Computers*, pp.34-40, Sep-Oct 2003.

[21] D. H. C. Du, H. Yen, and S. Ghanta. “On the General False Path Problem in Timing Analysis” in *Proceedings of 26th IEEE/ACM Design Automation Conference*, pp. 555-560, 1989.

[22] S. Devadas, K. Keutzer, S. Malik, and A. Wang. “Certified Timing Verification and the Transition Delay of a Logic Circuit” in *Proceedings of the 29th ACM/IEEE Design Automation Conference*, pp.549-555, 1992.

[23] S. Devadas, K. Keutzer, S. Malik, and A. Wang. “Certified Timing Verification and the Transition Delay of a Logic Circuit” in *IEEE Trans. VLSI Systems*, vol.2, pp.333-342, Sept. 1994.

[24] A. Ferzli, and F. N. Najm. “Statistical estimation of leakage-induced power grid voltage drop considering within-die process variations” in *Proceedings of ACM/IEEE 40th Design Automation Conference*, pp 856-859, 2003.

[25] J. E. Freund and R. E. Walpole. *Mathematical Statistics*, Prentice Hall, 1987.

[26] M. A. Gharaybeh, M. L. Bushnell, and V. D. Agrawal. “Classification and Test Generation for Path-Delay Faults Using Single Stuck-at Fault Tests” in *Journal of Electronic Testing: Theory and Applications*, vol. 11, no. 1, pp.55-67, Aug. 1997.

[27] A. Gattiker, et al. ”Timing Yield Estimation from Static Timing Analysis” in *Proceedings of the IEEE ISQED*, pp 437-442, 2001.

[28] A. J. van de Goor, *Testing Semiconductor Memories: Theory and Practice*, Chichester, UK: John Wiley & Sons, Inc., 1991.

[29] K. Heragu, J. H. Patel, and V. D. Agrawal. “Segment Delay Faults: A New Fault Model” in *Proceedings of 14th IEEE VLSI Test Symposium*, pp. 32-39, May 1996.

[30] K. Heragu, J. H. Patel, and V. D. Agrawal. “SIGMA: A Simulator for Segment Delay Faults” in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pp.502-508, November 1996.

[31] S. T. Huang, T. M. Parng and J. M. Shyu. “A Polynomial-Time Heuristic Approach to Approximate a Solution to the False Path Problem” in *Proceedings of the 30th ACM/IEEE Design Automation Conference*, pp.118-122, 1993.

[32] V. S. Iyengar, B. K. Rosen, and I. Spillinger. “Delay Test Generation 1 - Concepts and Coverage Metrics” in *Proceedings of IEEE International Test Conference*, pp.857-866, Sep. 1988.

[33] V. S. Iyengar, B. K. Rosen, and I. Spillinger. “Delay Test Generation 2 - Algebra and Algorithms” in *Proceedings of IEEE International Test Conference*, pp.867-876, Sep. 1988.

[34] V. S. Iyengar, B. K. Rosen, and J. A. Waicukauski. “On Computing the Sizes of Detected Delay Faults” in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol.9, pp.299-312, March 1990.

[35] R. Jayabharathi, H. Chang and J. A. Abraham. “Practical Considerations for Improving Delay Fault Tests using a Unified Delay Model” in *Proceedings 7th IEEE North Atlantic Test Workshop*, pp.84-91, May 1998.

[36] K. Kim, S. Mitra and P. Ryan. “Delay Defect Characteristics and Testing Strategies” in *IEEE Design & Test of Computers*, pp.8-16, Sep-Oct 2003.

[37] T. I. Kirkpatrick and N. R. Clark. “PERT as an Aid to Logic Design” in *IBM Journal of Research and Development*, vol.10, pp.135-141, 1966.

[38] A. Krasniewski and L. Wronski. “Coverage of Delay Faults: When 13% and 99% mean the same” in *Lecture Notes in Computing Science: Dependable Computing* (K. Echtle, D. Hammer, and D. Powell, eds), vol.852, pp.178-195, Springer Verlag, 1994.

[39] A. Krishnamachary, J. A. Abraham and R. S. Tupuri. “Timing Verification and Delay Test Generation for Hierarchical Designs” in *Proceedings of the 14th International Conference on VLSI Design*, pp.157-162, 2001.

[40] W. K. Lam et al., “Delay Fault Coverage, Test Set Size, and Performance Trade-Offs” in *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*,vol.14, no.1, pp.32-44, Jan.1995.

[41] K. T. Lee and J. A. Abraham. “Critical Path Identification and Delay Tests of Dynamic Circuits” in *Proceedings of the International Test Conference*, pp.421-430, 1999.

[42] R. Levy, D. Blaauw, G. Braca, A. Dasgupta, A. Grinshpon, C. Oh, B. Orshav, S. Sirichotiyakul and V. Zolotov. “ClariNet: a Noise Analysis Tool for Deep Submicron Design” in *Proceedings of the 37th IEEE/ACM Design Automation Conference*, pp.233-238, 2000.

[43] W.-N. Li, S. M. Reddy, and S. K. Sahni. "On Path Selection in Combinational Logic Circuits" in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 8, no.1, pp.56-63, January 1989.

[44] X. Lin, R. Press, J. Rajski, P. Reuter, T. Rinderknecht, B. Swanson and N. Tamarapalli. "High-Frequency, At-Speed Scan Testing" in *IEEE Design & Test of Computers*, pp.17-25, Sep-Oct 2003.

[45] J.-J. Liou, A. Krstic, L.-C. Wang and K.-T. Cheng. "False-Path-Aware Statistical Timing Analysis and Efficient Path Selection for Delay Testing and Timing Validation" in *Proceedings of the 39th IEEE/ACM Design Automation Conference*, pp.566-569, June 2002.

[46] J.-J. Liou, L.-C. Wang and K.-T.Cheng. "On Theoretical and Practical Considerations of Path Selection for Delay Fault Testing" in *Proceedings of International Conference on Computer-Aided Design*, pp.94-100, Nov. 2002.

[47] B. Madge, B. Benware and R. Daasch. "Obtaining High Defect Coverage for Frequency-Dependent Defects in Complex ASICs" in *IEEE Design & Test of Computers*, pp.46-53, Sep-Oct 2003.

[48] A. K. Majhi, J. Jacob, L. M. Patnaik, and V. D. Agrawal. "On Test Coverage of Path Delay Faults" in *Proceedings of 9th International Conference on VLSI Design*, pp.418-421, January 1996.

[49] M. C-T. Chao, L.-C. Wang, K. -T. Cheng. "Pattern Selection for Testing of DSM Timing Defects" in *Proceedings of Design, Automation and Test in Europe*, pp.1060-1065, March 2004.

[50] W. Mao and M. D. Ciletti. "Robustness Enhancement and Detection Threshold Reduction in ATPG for Gate Delay Faults" in *Proceedings of the International Test Conference*, pp.588-596, Sep 1992.

[51] F. S. Marques, R. P. Ribas, S. Sapatnekar and A. I. Reis. "A New Approach to the Use of Satisfiability in False Path Detection" in *Proceedings of Latin American Test Workshop*, pp.59-64, 2005.

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

[53] P. C. McGeer, A. Saldanha, P. R. Stephan, R. K. Brayton, and A. L. Sangiovanni-Vincentelli. "Timing Analysis and Delay-Fault Test Generation using Path-Recursive Functions" in *Proceedings of International Conference on Computer-Aided Design*, pp.180-183, Nov. 1991.

[54] P. C. McGeer and R. K. Brayton. *Integrating Functional and Temporal Domains in Logic Design*, Kluwer Academic Publishers, 1991.

[55] L. W. Nagel. "SPICE2, A Computer Program to Simulate Semiconductor Circuits", *Memo UCB/ERL M520, Univ. California, Berkeley*, May 1975.

[56] S. R. Nassif. "Modeling and Analysis of Manufacturing Variations" in *Proceedings of IEEE Custom Integrated Circuits Conference*, pp.223-228, 2001.

[57] S. Natarajan, S. K. Gupta and M. A. Breuer. "Switch-Level Delay Test" in *Proceedings of the International Test Conference*, pp. 171-180, 1999.

[58] P. Nigh and A. Gattiker. "Test Method Evaluation Experiments and Data" in *Proceedings of International Test Conference*, pp.454-463, 2000.

[59] M. Orshansky and K. Keutzer. "A General Probabilistic Framework for Worst Case Timing Analysis" in *Proceedings of the 39th IEEE/ACM Design Automation Conference*, pp.556-561, June 2002.

[60] E. S. Park and M. R. Mercer. "An Efficient Delay Test Generation System for Combinational Logic Circuits" in *IEEE Trans. CAD*, vol. 11, pp.926-938, July 1992.

[61] S. Pateras. "Achieving At-Speed Structural Test" in *IEEE Design & Test of Computers*, pp.26-33, Sep-Oct 2003

[62] S. Patil and S. Reddy. “A Test Generation System for Path Delay Faults” in *Proceedings of IEEE International Conference on Computer Design*, pp.40-43, Oct. 1989.

[63] A. Pierzynska and S. Pilarski. “Pitfalls in Delay Fault Testing”in *IEEE Trans CAD*, vol. 16, pp.321-329, March 1997.

[64] I. Pomeranz and S. M. Reddy. “Test Generation for Path Delay Faults Based on Learning” in *Proceedings of International Conference on Computer-Aided Design*, pp.428-435, 1993.

[65] I. Pomeranz and S. M. Reddy. “An Efficient Nonenumerative Method to Estimate the Path Delay Fault Coverage in Combinational Circuits” in *IEEE Trans. on Computer-Aided Design*, vol. 13, no. 2, pp.240-250, Feb. 1994.

[66] I. Pomeranz and S. M. Reddy. “NEST: A Nonenumerative Test Generation Method for Path Delay Faults in Combinational Circuits” in *IEEE Trans. on Computer-Aided Design*, vol. 14, pp.1505-1515, Dec. 1995.

[67] A. K. Pramanick and S. M. Reddy. “On the Detection of Delay Faults” in *Proceedings of IEEE International Test Conference*, pp.845-856, Sep. 1988.

[68] C. Pyron, J. Prado and J. Golab. “Next Generation PowerPC Microprocessor Test Strategy Improvements” in *Proceedings of International Test Conference’*, pp.414-423, 1997.

[69] R. Raina, R. Bailey, D. Belete, V. Khosa, R. Molyneaux, J. Prado and A. Razdan. “DFT Advances in Motorola’s Next Generation 74xx PowerPC Microprocessor” in *Proceedings of International Test Conference*, pp.131-140, 2000.

[70] R. Raimi and J. A. Abraham. “Detecting False Timing Paths” in *Proceedings of the 35th IEEE/ACM Design Automation Conference*, pp.737-741, 1999.

[71] J. Saxena et al., "Evaluating ATE Features in Terms of Test Escape Rates and Other Cost of Test Culprits" in *Proceedings of International Test Conference*, pp.1040-1048, 2002.

[72] A. D. Singh et al. "Binning for IC Quality: Experimental Studies on the SEMATECH Data" in *Proceedings of International Symposium on Defect and Fault Tolerance in VLSI Systems*, pp.4-10, 1998.

[73] M. Sivaraman and A. J. Strojwas. *A Unified Approach for Timing Verification and Delay Fault Testing*, Kluwer Academic Publishers, 1998.

[74] G.L. Smith. "Model for Delay Faults Based upon Paths" in *Proceedings of IEEE International Test Conference*, pp. 342-349, November 1985.

[75] R. Tekumalla and P. R. Menon. "Test Generation for Primitive Path Delay Faults in Combinational Circuits" in *Proceedings of International Conference of Computer-Aided Design*, pp.636-641, 1997.

[76] N. Tendolkar, R. Molyneaux, C. Pyron and R. Raina. "At-speed Testing of Delay Faults for Motorola's MPC7400, a $PowerPC^{TM}$ Microprocessor" in *Proceedings of VLSI Test Symposium*, pp.3-8, 2000.

[77] N. Tendolkar, R. Raina, R. Woltenberg, L. Xijiang, B. Swanson and G. Aldrich . "Novel Techniques for Achieving High At-speed Transition Fault Test Coverage for Motorola's Microprocessors based on $PowerPC^{TM}$ instruction set architecture" in *Proceedings of VLSI Test Symposium*, pp.3-8, 2002.

[78] C. W. Tseng, J. Li and E. J. McClusky. "Experimental Results for Slow-Speed Testing" in *Proceedings of VLSI Test Symposium*, pp.37-42, 2002.

[79] C. W. Tseng and E. J. McClusky. "Multiple-Output Propagation Transition Fault Test" in *Proceedings of International Test Conference*, pp.358-366, 2001.

[80] J. Zeng, M. S. Abadir, J. Bhadra, J. A. Abraham. “Full Chip False Timing Path Identification: Applications to the PowerPC Microprocessors ” in *Proceedings of Design, Automation and Test in Europe*, pp.514-518, 2001.

[81] J. Zeng, M. S. Abadir, J. A. Abraham. “False Timing Path Identification using ATPG Techniques and Delay-Based Information” in *Proceedings of the 39th IEEE/ACM Design Automation Conference*, pp. 562-565, 2002.

[82] J. Zeng, M. Abadir, G. Vandling, Li-C, Wang, A. Kolhatkar, J. Abraham. “On Correlating Structural Tests with Functional Tests for Speed Binning of High Performance Design” in *Proceedings of International Test Conference*, pp. 31-37, 2004.

[83] J. Zeng, M. Abadir. “On correlating structural tests with functional tests for speed binning” in *Proceedings of International Workshop on Current and Defect Based Testing*, pp. 79-83, 2004.

Vita

Jing Zeng was born to TianYuan Zeng and LanYing Liu on the 7th of November 1967 in Beijing, China. She graduated with a B. S. in physics from Beijing University, Beijing, China in 1989. She obtained her M.S. in Physics from the University of Arizona in Tucson, Arizona in 1991. She transferred to University of Texas at Austin in 1992 to study biomedical engineering and received a second M. S. in Electrical Engineering in 1994. She started pursuing her Ph.D in computer engineering at Computer Engineering Research Center in fall 1994. From summer 1995 to spring 1996, she worked at Advanced Micro Devices as an intern performing DFT and functional verification related tasks. Since 1997, she has been working as a CAD engineer at the PowerPC Design Center in Freescale Semiconductor Inc.(formerly Motorola Inc.) in Austin, while continuing working on her PhD at the University of Texas at Austin.

Permanent address: 1110 Barton Creek Dr
Dripping Springs, Texas 78620

This dissertation was typeset with LaTeX[†] by the author.

[†] LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.