The Dissertation Committee for Arun Krishnamachary
certifies that this is the approved version of the following dissertation:

Test Generation for Realistic Defects

Committee:

Jacob. A. Abraham, Supervisor

Anthony. P. Ambler

Margarida Jacome

Nur Touba

Steve Keckler

Raghuram Tupuri

Test Generation for Realistic Defects

by

Arun Krishnamachary, B.E., M.S.E.E

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

December 2003

To my parents

Acknowledgments

I would like to begin by offering my sincerest appreciation to everyone who has encouraged me in my pursuit of this research. First, I would like to thank my advisor Dr. Jacob Abraham, whose ideas and knowledge of practical applications of the problem has kept me in focus for the real solution. I would like to thank my committee members Tony Ambler, Steven Keckler, Margarida Jacome, Nur Touba and Raghuram Tupuri whose inputs proved very useful. My friends at the university, who have proved vital in helping me stay focused and providing me such good company. The staff at CERC Linda, Shirley, Andrew, Ruth who have helped me throughout the stay here. My colleagues at Intel Phoenix have been very encouraging throughout the last year I have been working here... thanks guys!!

I would also like to thank the Semiconductor Research Corporation who funded my research throughout my Ph.D., Dr Trevor Mudge, University of Michigan for the ARM model, LSI logic for the process libraries that we used.

And of course, a special thanks goes to my parents, my sister, my brother , my brother-in-law whose constant encouragement helped me pursue my goals. Last but certainly not least, my fiance Swathi with whom I will be getting married soon, ever since we have known each other, you have been a constant source of support.

ARUN KRISHNAMACHARY

The University of Texas at Austin

December 2003

Test Generation for Realistic Defects

Publication No. _____

Arun Krishnamachary, Ph.D.
The University of Texas at Austin, 2003

Supervisor: Jacob. A. Abraham

The rapidly evolving process technologies and device complexity that have fueled the exponential growth in the performance of microprocessors have made the manufacturing test of these devices a hard problem. In addition to making the detection of defects modeled by the classical fault models like the stuck-at and the transition fault model more complex, these process technologies have resulted in additional types of defects (like the resistive opens, defects due to the process parameter variations and crosstalk defects) becoming more prominent.

The requirement for an effective delay test framework which involves an effective fault model, optimized test generation procedure and efficient test application has become even more urgent in the current scenario. This framework also needs to address the issues with yield and complexity (due to the large number of faults) that are associated with a delay test strategy.

In this dissertation, we provide a strategy to help address many of the issues outlined above. An improved delay fault model is first proposed which enables better detection of resistive open defects and also yields a good oppurtunistic coverage

of defects due to process parameter variations. This is coupled with an optimized test generation strategy, which facilitates efficient delay test generation under the fault model. A fault collapsing technique helps reduce the number of faults that need to be targeted. To improve the yield of a scan based test application, a technique is provided to identify the functional sensitizability of paths across multiple latch boundaries, and the effect of this strategy on yield is then calculated. Finally a technique to enable use of ATPG to evaluate the chip level sensitizability of paths which enables the use of tighter timing bounds in chips is presented.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The performance of microprocessors has been increasing at a rapid pace. This rate of increase in the performance of the microprocessors is forecast to continue for more processor generations. The Graph 1.1 shows the rate of increase in the operating frequencies over the next decade as predicted by the SIA roadmap. This has been enabled by new process technologies that have been introduced. The process nodes are shown in Graph 1.2.

Chip design flows have introduced significant levels of automation to enable the designers to handle this increase in the number of components in a design and the evolving process technologies. Though automation helps in speeding up the overall design process, the methodologies have lagged behind the accelerating technology changes. The rate of the reduction in the process sizes has gone down (forecast to reduce further after 2004) and there has been a corresponding decrease in the growth in the number of transistors in the chip, but the complexity still remains.

The traditional chip design flow is illustrated in the Figure 1.3. The design flow involves an initial specification of the design in a (reasonably) high level

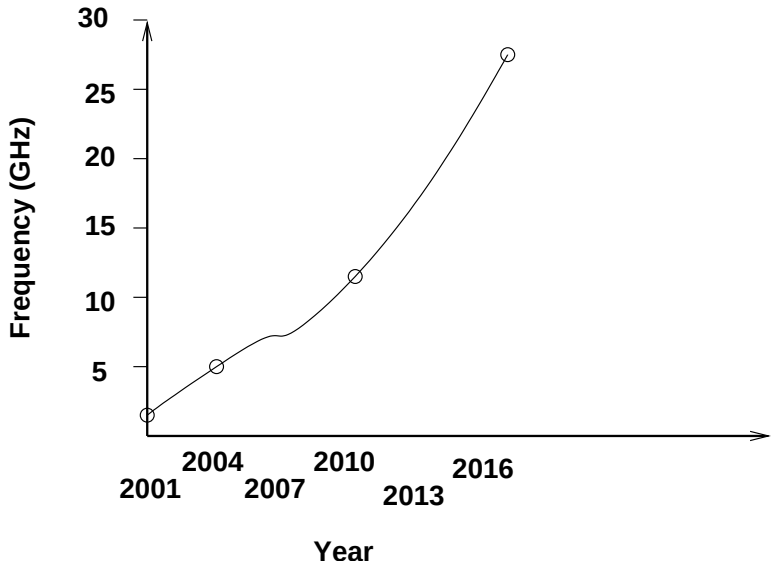


Figure 1.1: Increase in Frequency

Hardware Description Language (HDL); these languages which include Verilog and VHDL enable an abstract description of the design. Once this specification is complete, these designs are synthesized or hand crafted to less abstract gate/transistor level design of the circuit.

The timing of the circuit under consideration is decided by determining the maximum delay among the combinational blocks in the design. In a sequential design, the latch to latch delay is considered as an atomic block for timing. In Figure 1.4, we show the combinational blocks as logic clouds; of the maximum path delays in each of the combinational blocks, the worst is chosen as an initial estimate of the timing of the circuit. In some cases, the presence of transparent latches makes the consideration of delays over multiple latch boundaries necessary; this is a more complex process especially in cases where the combined delay is

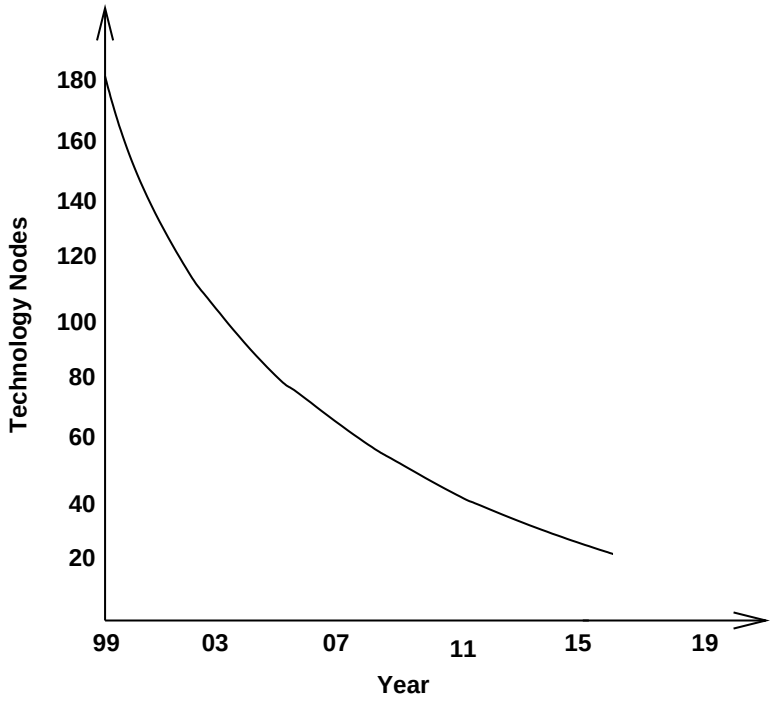


Figure 1.2: Change in Process

close to the threshold.

Significant simulation is performed on the initial abstract specification of the design to validate the design, and to verify the equivalence of the design in two contiguous levels of the flow. Significant testing is also done in the manufacturing process to check for correctness of operation.

As the design progresses from abstract to logic design to circuit design to fabrication, the cost of fixing bugs increases exponentially. For example when a logic bug is found in the HDL description, it requires a simple fix to the HDL code to fix the bug, but if the same bug is found after fabrication, the design needs to be validated through one fabrication step. This is true of both logical bugs and

timing marginalities. Thus it is desirable to perform significant logic simulation and timing marginality estimation early in the flow. Estimating the timing marginalities requires that an accurate identification of the critical paths be made early in the design and sufficient slack be provided in the design to account for manufacturing marginalities.

The validation steps preceding the manufacture validate the correctness of the design. The primary techniques used in this evaluation are simulation based and formal verification based. Simulation based validation involves tests that are hand-crafted or generated randomly to test the correct operation of the design. These tests are often applied using a test-bench. Formal techniques use model checking or theorem proving based techniques to verify that the design conforms to specifications expressed as a set of properties. Formal equivalence based techniques are also used to validate the design against a golden model of the design.

Timing verification which involves the identification of the critical paths in the design and evaluating whether their delay conforms to the timing requirement of the circuit is then performed before the chip is fabricated. This pre-fabrication timing verification is necessary because of the tight timing bounds and the negligible design margins that are allowed by the current process technologies. Techniques have been developed to perform static and dynamic timing verification and delay models have been proposed to enable identification of critical paths that have close correlation to the critical paths that are seen after fabrication.

The timing verification step validates all the critical paths in the circuit for a delay while factoring in slack that allows for inaccuracies in the delay model used in evaluating the critical paths in the design. In-spite of this slack, an inaccurate delay model would cause timing failures, thus causing additional silicon steppings. Thus an effective critical path identification technique and an accurate delay estimation

model is essential to minimize the debug steps post-silicon.

Manufacturing test is used during the manufacturing process to test for the manufacturing defects that could have appeared due to process inadequacies, environment or just due to random effects. These tests are run on a sample of the manufactured chips. The manufacturing tests are applied using structural and functional testers depending on the test application technique. The main goal of the manufacturing test step is to detect maximum defects using an optimal set of vectors which minimizes the test application time. The test time becomes critical as the volume of manufacture increases.

Defects per Million (DPM) [18] is a measure used to determine the quality of the chips being manufactured. Any defect that escapes a test in any stage of the manufacturing flow contributes to an increase in the DPM. DPM measures are associated will all the screening processes to measure the effectiveness of the screen. This data is used to reduce cost and improve the throughput time. There is a clear correlation between the fault coverage and the DPM measured, and the primary methodology to reduce the DPM is to introduce more tests into the manufacturing test flow. Other techniques adopted to achieve a reduction in DPM involve the following.

- Process changes for robustness
- Improved handling methods to minimise the damage
- Design for robustness

When new tests are introduced to reduce the DPM, they are evaluated based on effectiveness. Adding a marginally effective test limits the number of cycles available on the additional test steps, so this tradeoff is evaluated before adding the

test. Overall, the total cost of test is weighted against the cost of test escapes. This cost of test escapes is uniformly high both in tangible and intangible cost. Tangible costs are high because at the customer side, assembly of the end-product is done before testing and the cost of isolating a defect in this scenario is high. A graph showing the representative relationship between test cost and test escape cost is shown in Figure 1.5.

The classical fault models used in the industry for manufacturing tests is the stuck-at fault model. This fault model along with the transition fault model is used extensively in the manufacturing test process and these models have proved effective in providing target DPM (Defects Per Million) in the manufacturing process. But as new process technologies are introduced, new defect types which cannot be detected by these fault models are becoming more prominent. These defect types include resistive opens, crosstalk, power supply noise.

For example in the copper manufacturing technology, the interconnect breaks occur with high probability. One of the main reasons for this is the difficulty of depositing copper without voids and cracks in high aspect ratio trenches [20]. Further, during the chemical mechanical polishing for planarizing the metal layers, copper tends to be worn out much faster than the neighboring dielectrics thus causing dishing, leading to higher resistance at the site of this dishing phenomenon.

As newer manufacturing technologies emerge, the instances when manufacturing actually happens with a process that cannot be classified as mature are becoming common. This leads to significant process parameter variation in the actual manufacturing of the chip. Thus the manufacturing test step should also enable targeting the parameter variations that could affect the performance of the circuit.

The opportunistic coverage that is provided by the vectors generated using the stuck-at and the transition fault model are not adequate to reduce DPM which

might be affected by these new defect types. Newer fault models are thus required to help target these new defect types.

In this dissertation, techniques to enable efficient test for these new kinds of defects are presented. The techniques presented start with a useful fault model to maximize the most common defect seen currently in the manufacturing flow which is the resistive open defect and a test generation technique to efficiently generate test vectors for this fault model. A technique is then presented to prune the number of paths based on sensitizability information. This is presented both in the timing verification and the manufacturing test domain.

In Chapter 3, a new fault model that yields high coverage of resistive open defects and significant opportunistic coverage of the process parameter variations is presented. This fault model maximizes the coverage of resistive open faults by targeting the longest sensitizable paths through the fault sites. Further, an implicit sensitization based methodology for test generation based on the test generation tool presented in [71] is presented. In Chapter 4, using learning on the unsensitizable sub-paths in the circuit, techniques to limit the complexity seen by the test generation tool are outlined. Also presented is a fault collapsing methodology to reduce the number of faults that are targeted by the delay test vectors and a method to reduce the test application time while still having limited area overhead over standard scan, this method is based on reconfigurable scan. An intelligent threshold selection mechanism to further reduce the test generation time is then presented. In Chapter 5, we present a timing verification strategy that enables evaluating the functional sensitizability of critical paths at the full-chip level using ATPG with a simplified model of the design to deal with, this helps in easily reaching the timing bounds that are increasingly imposed on current generation designs because the methodology helps remove functionally unsensitizable paths which affect the timing of the

design. Chapter 6 presents the yield savings that can be achieved by considering the the sensitizability of critical paths targeted under the path delay fault (PDF) based delay test approach. This deals with the savings that can be achieved in the manufacturing step of the design flow using a technique similar to the one presented in Chapter 5.

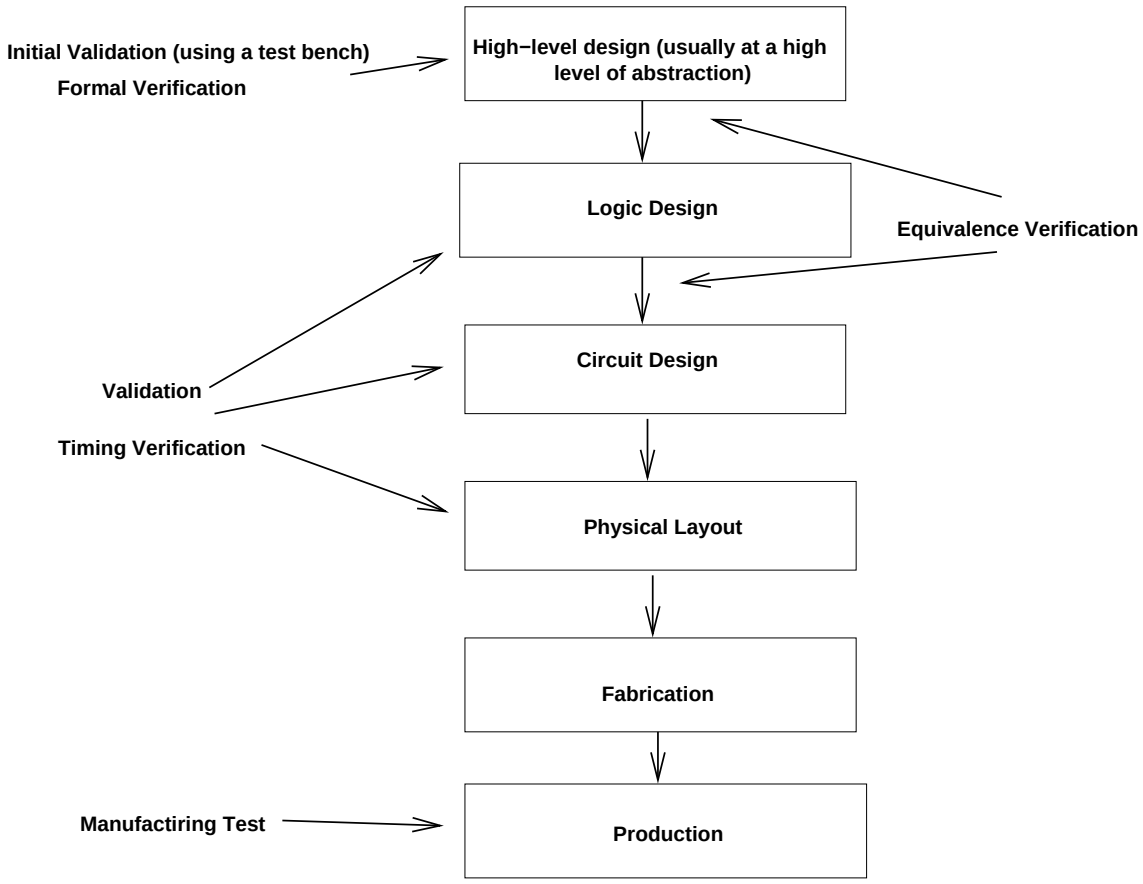


Figure 1.3: VLSI Design Flow

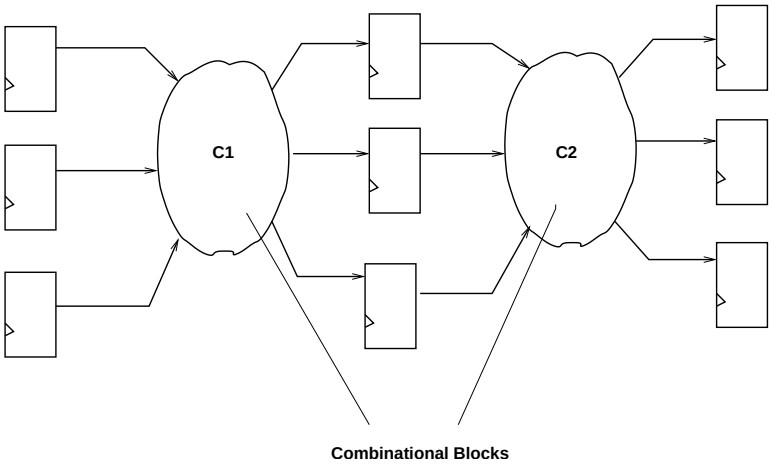


Figure 1.4: Sequential Circuit

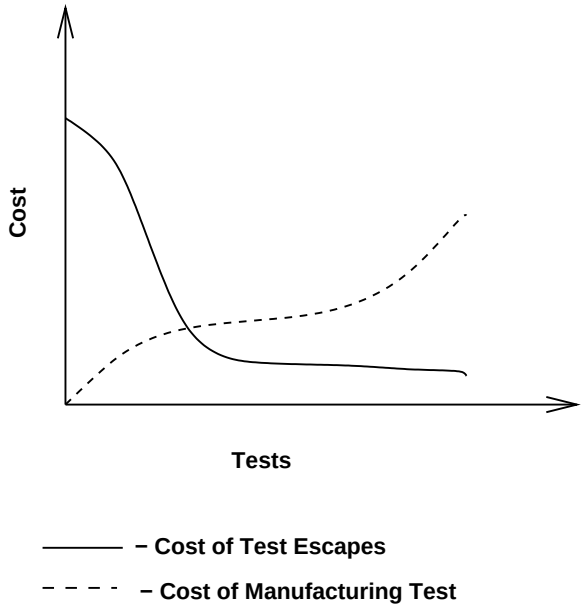


Figure 1.5: Relationship Between Test Cost and Escape Cost

Chapter 2

Background

2.1 Basic Definitions

Consider a sequential circuit shown in the Figure 1.4. The blocks represented by C_1, C_2... C_n are the combinational blocks that are targeted in any timing verification/ delay test Generation methodology.

Definition 1 *Any combinational circuit can be represented by $\langle V, E \rangle$, where V = $i\{g_1, g_2, ..., g_n\}$ represent the set of all the gates in the block and E = $\{\{g_1, g_2\}, \{g_1, g_3\}...\}$ represents the set of edges in the block.*

Each of the gates in the circuit has an associated functionality (AND,OR etc.) and has a delay (D(g_i)) associated with it, this is dependent on the various factors including the input transitions. The maximum delay possible through the gate is ($D_{max}(g_i)$). The delays seen by edges is given by $D(g_x, g_y)$ and the maximum delay is given by ($D_{max}(g_i, g_j)$).

Thus for example, consider the circuit shown in Figure 2.1. Here the V = $\{g_0, g_1, g_2\}$ and E = $\{\{g_0, g_1\}, \{g_0, g_2\}, \{g_4, g_0\}, \{g_5, g_0\}, \{g_3, g_1\}\}$.

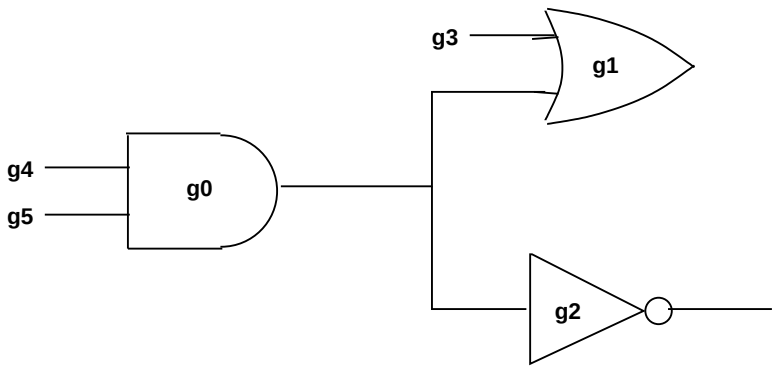


Figure 2.1: Example Circuit

Definition 2 *A path P = an ordered set of nodes* $\{g_0, g_1, g_2 \ldots g_n\}$ *where* g_0 *corresponds to the input of the combinational block and* g_n *is a primary output of the circuit. The delay of the path D(P) under consideration is the sum of delay on each gate and each edge in the path under consideration. When the* D_{max} *is considered for each gate and net, D(P) =* $D_{max}(P)$.

A subset of this ordered set of nodes is referred to as a sub-path.

Definition 3 *The Structurally Longest Path Delay of a circuit is the delay of the path with maximum* D_{max} *among all paths in the combinational path.*

Definition 4 *A path is sensitized if there exists an input assignment such that a transition is enabled at the input of the path and this results in transitions on all gates along the path and a transition at the output of the path.*

A path is unsensitizable if for all input assignments, the path cannot be sensitized.

Figure 2.2 shows a path being sensitized for an input value assignment. The path highlighted in bold is sensitizable by the input vector shown. Figure 2.3 shows a

path that is unsensitizable. The path shown in bold is unsensitizable because of the reconvergent fanout from the gate g_0.

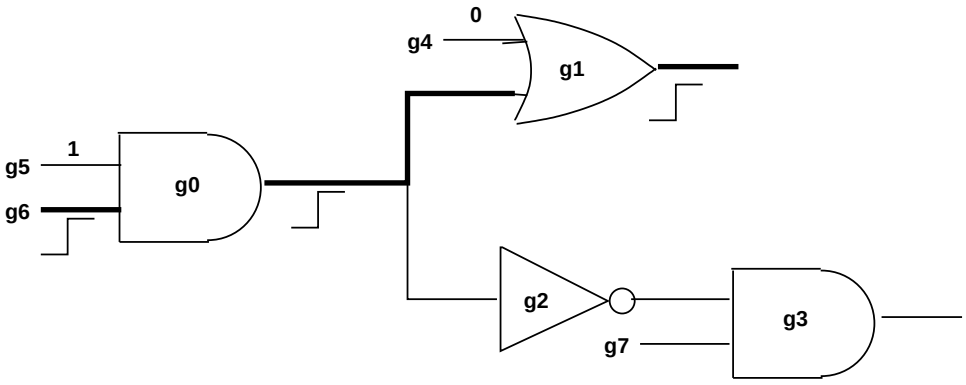


Figure 2.2: Sensitizable Path

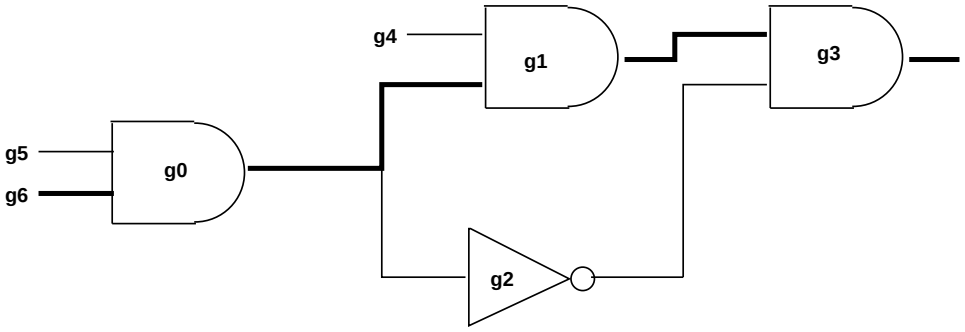


Figure 2.3: Unsensitizable Path

Definition 5 *Given a path P=*$\{g_0, g_1 ..., g_n\}$*, an edge* $\{g_x, g_y\}$ *such that* $\{g_x, g_y\} \in P$*, is an on-path input of gate* g_y*.*

An edge g_i, g_j *such that* $g_j \in P$ *and* $\{g_i, g_j\} \ni P$*, is an off-path input to gate* g_j*.*

Considering the circuit in Figure 2.2, input g_5 is on-path and the input g_6 is the off-path input.

Definition 6 *A controlling value of a gate is a value at the input of the gate that determines the output value of the gate independent of the other inputs of the gate.*

Figure 2.1 shows the truth tables for an "and" gate and an "or" gate. As can be seen, a value of 1 is a controlling value for an "or" gate and a value of 0 is the controlling value for an "and" gate. Each of these values when present even in one of the inputs of the gates determines the output independent of the other.

Table 2.1: Truth Tables for AND and OR Gates

INPUT A	INPUT B	OUTPUT (AND)	OUTPUT(OR)
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

2.2 Timing Verification

Timing verification is done using two distinct approaches:

- Timing Analysis estimates the maximum delay in a circuit in an input independent manner
- Timing Simulation estimates the maximum delay in a circuit by simulating different input vectors on the circuit

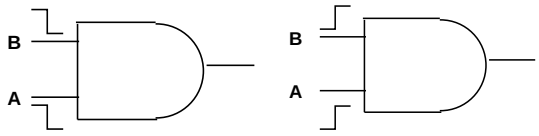
The simplest although conservative way to estimate the maximum delay of the circuit is to estimate the longest structural delay of the circuit (PERT). The maximum delay that is obtained under this approach is a very conservative estimate because the critical paths yielding the maximum delay might not be sensitizable for any value on the primary inputs of the circuit. Consider Figure 2.3; the path $\{g_0 - g_1 - g_3\}$ is the longest path in the circuit (assuming a unit delay of 1 for each of the gates), but this path is unsensitizable, because of the re-convergent fanout from the gate g_0.

Algorithms in literature assume a floating mode of operation which imposes the condition that all the nodes in the circuit are at unassigned values till the effect of the currently applied input vector reaches the node. This enables an accurate estimate of the maximum delay of a block under the assumption that the block is embedded in a larger sequential design. Many algorithms have been proposed for timing verification under the floating mode of operation. These include SENV, Dynamic, Viability, Brand-Iyengar and DYG [36]. CRITIC was a timing verification engine based on "vigorous sensitization". This will be explained further in Chapter 3.

Another important component of a timing verification system is the component delay models. The *fixed delay model* is a very common component delay model; this model assumes a fixed delay value for all components in the given circuit. The *unbounded delay model* [6] assumes that given a value D_c, the value of the delay of each component can vary from 0 to D_c. The bounded delay model [7] assumes that the delay of each component in the circuit always falls in a bound [Min, Max]. The actual delay also depends on other parameters which include the internal component state, output loading, input transition time and the neighboring node activity.

The timing verification algorithms at worst provide a conservative estimate of the delay of a combinational block; a path that is true will never be classified as false. The algorithms for sensitization evaluation of the gates in the circuit are classified based on their accuracy. The criteria for exact sensitization include the following.

- The on-path input is the earliest arriving controlling value, i.e., all side-inputs settle to non-controlling values or settle to controlling values after the on-path input
- The on-path is a non-controlling value and the off-path inputs settle to non-controlling values before the on-path transition arrives



(a) (b)

Table 2.2: Transitions on an AND Gate

An example of the first condition is shown in Figure 2.2(a); the on-path input is A which has a transition to controlling value. The off-path inputs settling to controlling value before the on-path input artificially causes an earlier transition on the gate which is not worst-case. Off-path inputs settling to non-controlling values have no arrival time considerations imposed on them. Figure 2.2(b) shows an example of the second condition, the off-path input settling to a non-controlling value after the on-path input inflates the delay of the path, so this is not suitable in an exact framework, when the off-path input settles to a controlling value before the on-path, the transition is masked so this needs to be avoided.

These excitation conditions can be further loosened with a consideration of the run time of the timing verification engine; for example the second condition is relaxed to the off-path inputs just settling to non-controlling values rather than settling at a particular point in time.

All the techniques that were outlined above target just the combinational portion of the design. This assumption of full controllability at the input of the combinational block and full observability at the output of the combinational block forces a very conservative estimate of the timing of the design. Determining the functional constraints on the combinational block becomes essential to capitalize on the accuracy improvements that have been enabled by the timing verification schemes outlined above.

2.3 Defect Characteristics

The fault models that are currently being used in the manufacturing test of microprocessors are the stuck-at fault model and the transition fault model. But, as outlined in Chapter 1, these fault models are not adequate in providing the required detection in the modern process technologies where new kinds of defects are becoming more prominent [15, 16]. These new kinds of defects which include resistive opens, crosstalk defects, power supply noise and process parameter variations have excitation and propagation conditions different from both the stuck-at and transition fault models.

Resistive opens occur because of a discontinuity in the metal constituting a wire in the circuit. The resistance seen at the point of the defects characterizes the severity of the defect; this resistance in turn causes an increase in the propagation delay of a transition through the net under consideration. The reason for this

prevalence of opens can be seen in the process technologies being used. Aluminum technology is based on depositing metal and then removing the excess metal, while copper interconnects involve depositing a seed-liner followed by the electro-plating of the metal; this causes aluminum to have more bridges and copper to have more opens [18].

Consider the example shown in Figure 2.4, which shows the transistor model of a two-input NAND gate. The stuck-open fault implies any one of the four transistors could be stuck-open. For example, in the case where transistor A is stuck open, the pull-up through this transistor is affected as the transistor never switches on, and this causes a sequential behavior. However, this is not always the case when one of the terminals of the transistor is floating. In such cases where the opens are resistive opens, the resistance combines with the circuit capacitance to cause an increase in the delay of signal propagation in the circuit. Table 2.3 shows delay caused in a 2-input NAND gate for defects at locations X and Y as shown in Figure 2.4; this also shows increases in delays for various resistance values for a 0.35 micron process. A rising transition caused by a falling transition on I1 has a greater delay than in the fault-free case. Similarly, defect W in the pull-down path of the gate can be tested by driving a rising transition on the appropriate transistor. Thus, a methodology that drives both rising and falling transitions through the gate and through all the inputs of the gate enables the activation of all the defects in the gate. In this paper, we show results on circuits with primitive gates AND, OR, NAND, NOR and inverters. These gates are similar structurally to the example given above and can be tested using similar techniques.

Table 2.3 also shows the increase in delay due to a break (or partial break) in the interconnect between gates as shown in Figure 2.5. The phenomenon is illustrated with a series of inverters with a defect at location Z. Detecting these

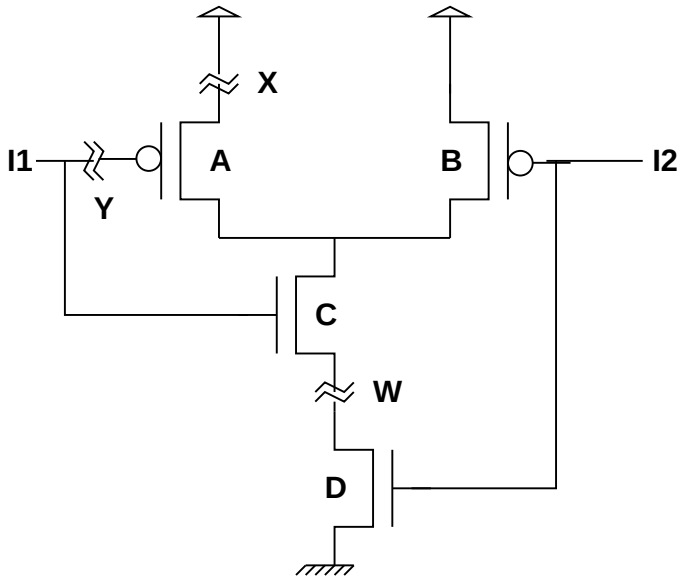


Figure 2.4: 2-input NAND gate with Breaks

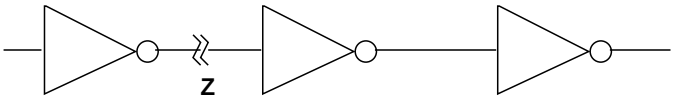


Figure 2.5: Chain of inverters with Break

defects requires the propagation of both rising and falling transitions through these nets. Further, delay tests are required to enable the detection of these resistive opens. For example, an increase in resistance of 100Ω causes the delay to increase by 170 picoseconds. This increase will only cause a problem if the path has a slack less than this value.

Process parameter variations occur due to irregularity in the lithographic patterns; this could occur due to a new process technologies being used in the man-

Table 2.3: SPICE Results

Fault location	Delay (ns)			
	Fault free	100Ω	500Ω	1KΩ
X	0.17	0.6	2.2	3.2
Y	0.17	0.18	0.19	0.20
W	0.15	0.22	0.88	1.68
Z	0.6	0.77	0.79	0.82

ufacturing. This aspect which is becoming very common now, causes variations in the interconnect and the intra-layer dielectric thicknesses due to lack of precise controls on the process. This lack of precise controls also causes variations in threshold voltage, effective channel length and temperature dependence. All these translate to variation in loads and cross-coupling capacitances seen by components; this in turn leads to delay and noise sensitivity variation across dies. In manufacturing test it becomes more important to test for such marginalities as newer process technologies are used in manufacturing. The defect causes an increase in delay (and noise) over areas of the chips rather than at points in the circuit (as in resistive opens).

Crosstalk defects occur when there is capacitive coupling between lines in the circuit. This occurs when circuit marginalities and design constraints cause long nets to be very close to each other. This kind of effect is very common in busses. One of the coupled nets acts as the aggressor and the other net acts as the victim in the capacitive coupling. The capacitive coupling causes a glitch on the victim when it has a steady state signal and a transition happens on the aggressor, and causes a delay effect when a transition in the opposite direction happens on the victim.

2.4 Coverage metrics for Realistic Defects

The fault models traditionally used in modeling the resistive open defects include the stuck-open fault model, floating gate model. The stuck open defect and the stuck-on defect models are switch level fault models which model the individual transistors in the gates as being either stuck open or stuck-on; this leads to a memory effect, thus requiring a two-vector test which first initializes the fault location and then propagates the fault effect. For example consider the nand gate shown in Figure 2.4; the test for transistor C stuck-open would initialize the gate output to 1 followed by a vector to drive the gate to 0. A defect is detected when the second vector is unable to drive the gate to 0.

The defects that cause the behavior modeled by the stuck-open fault model can be detected by using a vector set for transition fault detection. But the fact that this fault model assumes a very high resistance associated with the defect renders it unusable in the general case.

IDDQ testing [87, 88, 89] has also been proposed to detect these open defects. This technique involved the measurement of the quiescent power supply current to detect the presence of a defect. The presence of a open defect on a node in the circuit causes an elevated IDDQ; this elevated IDDQ measurement can be used to detect the presence of a defect. The drawback of this methodology is that in the current generation processes, distinguishing this elevated IDDQ from the background noise and the significantly lower measurement time due to the higher operating frequencies, and these factors render this technique almost unusable in the current scenario.

Process parameter variations cause a distributed delay effect where there is increase in the propagation delay on multiple gates. In the ideal case testing all

paths for delay defects would detect all the faults due to the process parameter variations. But due to the number of paths being exponential on the number of gates, this methodology is not feasible. The variations in the process parameters over dies and over lots plays an important part in determining an effective strategy to detect these kinds of defects. A uniform process parameter variations assumption over a die is inaccurate, we thus assume uniform values over combinational blocks which constitute atomic blocks for our test purposes.

The behavior of both the defects outlined above point to the need for a delay test vector set for adequate detection. Chapter 3 outlines a fault model that enables the detection of these defects.

2.5 Test Application Techniques

The test application techniques most commonly followed to detect delay defects are scan based. This is primarily because of the difficulty of generating functional test patterns. The scan based techniques enable a two-vector application by using various strategies.

- Enhanced scan
- Skewed Load
- Broadside scan

In the enhanced scan based methodology, the scan latch is modified to enable storage of two bits per latch. This enables storage of bits corresponding to the first and second test vectors in a single scan load, thus enabling the application of these test vectors in successive clock cycles. The fact that the bits corresponding to the

two vectors can be stored permits any combination of vectors to be generated. The modification of the design and the subsequent validation required and the timing penalties involved has limited the use of this methodology in actual manufacturing flows. The area overhead involves the 3 latches per enhanced scan latch; the premium on area has rendered this technique unsuitable in high performance designs. The requirement is for a test application technique that does not impose such a major area overhead.

In order to allieviate the effect on the delays seen in the circuit, and the area overhead outlined above, a technique was proposed to perform a selective enhanced scan addition to latches that did not affect critical paths in the design [8].

The skewed load test application [68, 69] enables the application of tests using a standard scan design. The second test vector in this case is obtained by a scan shift following the application of the first test vector. This imposes constraints on the vectors that can be generated, thus requiring modifications to the test generation tool. Additinally, the constraints imposed by the skewed load scheme are not the actual functional constraints as seen in the circuit. If these constraints become significant, the coverage of the test set might be affected.

Functional justification based delay test application [97] also enables delay test application in a standard scan based design. This method is illustrated in Figure 2.6. Here, the module under test is C2. The first test vector and a justified second vector (justified through C1) are loaded on to two successive latch boundaries. When the design is clocked once to enable the application of the first test vector, the second test vector becomes available at the latch boundary preceding the combinational block under test. A primary advantage of this test application technique is the implicit imposition of functional constraints for the generation of delay test. This dependency serves to refine the test vectors generated for this test

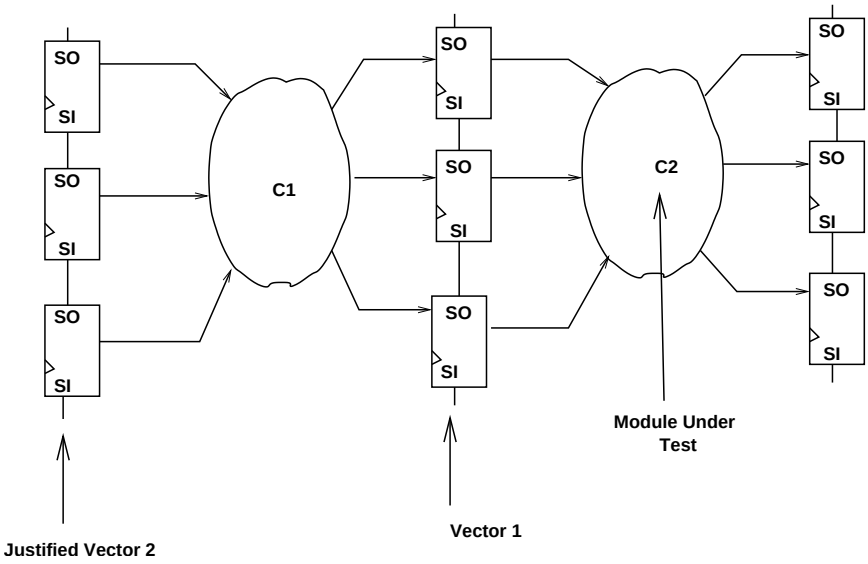


Figure 2.6: Functiona Justification Based Test

application technique.

Chapter 3

An Improved Fault Model

The requirement for a fault model yielding good detection of realistic defects arises from the fact that the stuck-at fault model and the transition fault model do not provide adequate coverage when targeting opens, process parameter variations and crosstalk defects. In this chapter we present an improved fault model to enable good detection of resistive open defects and good opportunistic coverage of defects due to process parameter variations.

3.1 Delay Fault Models

The primary fault models for delay defects are the transition fault model, the gate delay fault model and the path delay fault model. The gate delay first proposed in [24, 27, 28, 29, 25, 30] and the transition delay fault model first proposed in [23] target driving both rising and falling transitions through all the nets in the given combinational block. The criterion for coverage is the propagation of transitions through nets in the circuit and the observation of the fault effect on the output of the circuit. Here the information of the path through which the transition is propagated

is not considered for evaluating the coverage of the fault model; the fault model is thus useful for only the gross delay defects that could affect the combinational block. The transition fault model is similar to the gate delay fault model and is enabled in many commercial test generation tools.

The segment delay fault [31, 32] model tries to account for defects causing increase in propagation delay on multiple gates in the design. This model aims to cover distributed delay effects that are more localized. This model defines segments which correspond to an ordered set of nets that have a directed path between them as shown in the Figure 3.1. The segment lengths can vary from 1 to L where L corresponds to the number of nets in the longest path in the combinational block. When the segment length is 1, the segment delay fault model corresponds to the gate delay fault model and when the segment length is L, the segment delay fault model corresponds to the path delay fault model. In Figure 3.1, consider the path indicated in bold; the net corresponding to input g_6, corresponds to a segment of size 1, the segment consisting of g_6, edge $\{g_0, g_2\}$, corresponds to a segment of size 2.

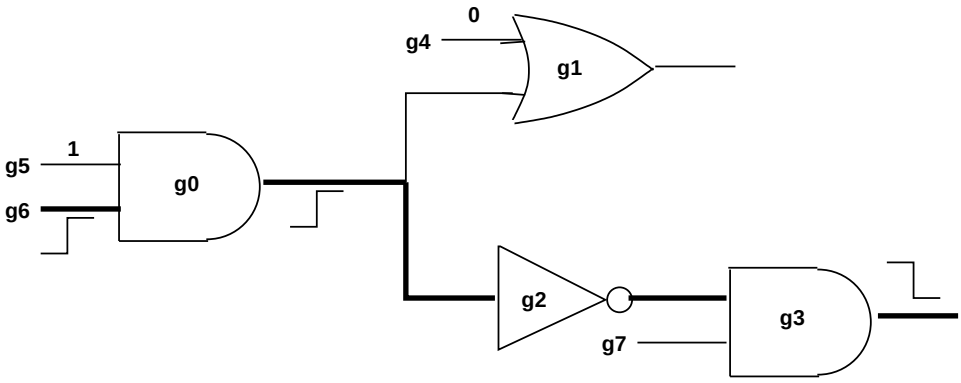


Figure 3.1: Segment Delay Faults

The main advantage of this fault model is that it gives a realistic measure of coverage, under a distributed fault assumption when compared to the gate/transition delay fault model. Further, the number of faults that need to be covered under this fault model is significantly less than the path delay fault model when the segment length considered is less than L. There is also more probability of sensitizing a segment rather than a whole path. This is shown in the Figure 3.2 consider the path shown in bold ($\{g_0, g_1, g_3\}$). This full path is unsensitizable, but just the segment $\{g_0, g_1\}$, can be propagated through g_7 instead, thus making the segment delay fault sensitizable.

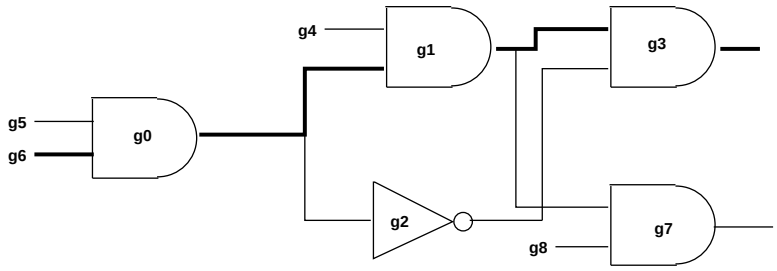


Figure 3.2: Segment Sensitization

The path delay fault model [24] aims to overcome the disadvantages of the gate and transition delay fault models. In particular, the fact that the longest paths in the combinational block might not be covered by a vector set generated with the gate/transition delay models. Thus ideally under the path delay fault model, the coverage metric would be the coverage of all the paths in the given combinational circuit. Because the number of paths in a given circuit is exponential on the number of gates in the circuit, this coverage metric is not suitable for the manufacturing test scenario as a large number of paths to cover would drastically increase the test

generation time and the test application time.

In order to reduce the number of paths that need to be tested to get satisfactory coverage of the possible defects in the circuit, some path selection techniques are used, the selection of all paths that are above a given threshold being the most preferred. This is based on the fact that paths closer to the critical path delay of the circuit have greater probability of detecting failure in the presence of delay defects and as a result greater probability of detecting these delay defects.

Surrogate fault models were developed to estimate the possible defect coverage of test vectors generated for an alternate fault model. Thus, a stuck-at fault model can be used to generate tests which can then be fault graded on a path delay fault model or a transition fault model depending on which fault model accurately models the defects being targeted. This type of fault model was used in [4, 5, 9] to enhance the defect detection of ATPG vectors.

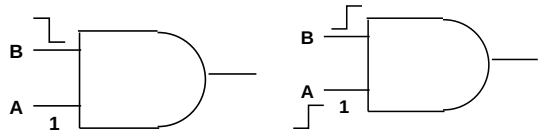
3.2 Test Generation

The test generation for the gate delay and the transition delay fault model is of very low complexity. The slow-to-rise fault can be detected by a vector driving the fault site to value 0 followed by a test vector for a stuck-at-0 fault at the site of the fault. The test generation for the slow-to fall fault is similar. The test generation for these fault models can thus be easily integrated into a stuck-at test generation scheme.

The test generation scheme [90, 91, 92, 96] for the path delay fault model is dependent on what is termed as the quality of test vectors that are required. The tests for the path delay fault model are classified as robust and non-robust path delay tests. Consider a 2-vector test set $\langle v_1, v_2 \rangle$; the path delay fault targeted is detected if the output seen during test is different from the expected value. A two-vector test

$\langle v1, v2 \rangle$ is classified as a robust test if the vector combination launches the required transition at the input of the targeted path and, for all the gates in the circuit,

- if the on-path input settles to controlling values, the off-path inputs must be at stable non-controlling values,
- if on-path input settles to non-controlling values, the off-path inputs should eventually settle to non-controlling values.



(a) (b)

Figure 3.3: Robust Test Generation

Examples of the robust sensitization criterion is shown in Figures 3.3(a,b) for an AND gate.

The set of requirements for a non-robust path sensitization, relaxes the conditions for the off-path inputs for a to-controlling transition on the on-path input of a gate. The non-robust sensitization criterion requires the off-path inputs to settle to non-controlling values before the on-path inputs. This requirement for the off-path input to settle to non-controlling values before the on-path inputs eliminates the possibility of the to-controlling transition being masked, which causes the right value to be captured at the output latch even in the presence of a defect. This condition is shown in Figure 3.4(a,b); the Figure (a) shows that the off-path input settles to the non controlling value before the on-path input, and Figure (b) shows the output having no transition when the off-path input is delayed.

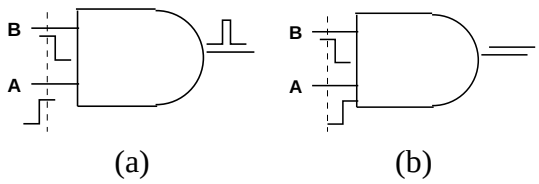


Figure 3.4: Nonrobust Test Generation

Though the non-robust sensitizability condition introduces a certain amount of uncertainty because the off-path inputs might be delayed due to process marginalities which leads to an escape, the higher probability of detecting a longer path with a non-robust test makes it very useful [35]; this is required especially with the prevalence of resistive open type of defects, where greater stress needs to be placed on the length of paths through which we propagate the fault effect. A lot of research has focussed on the generation of robust tests, but because of the advantages of the non-robust tests outlined above, we focus on the generation of non-robust tests in this paper.

The robust and the non-robust criterion can also be used in generating the test vectors to target the segment delay faults. These have the same advantages and disadvantages as the path delay fault model.

The test generation techniques used to detect the faults with the delay fault models have relied on identifying a set of paths and then trying to determine their sensitizability and generate the corresponding test vectors to detect the faults that are targeted. These are very high in complexity because of the number of paths that need to be evaluated for sensitizability before a sensitizable path can be identified.

Using the concept of a surrogate delay model, some methodologies were proposed to generate multiple vectors for each fault in a given fault model to improve the probability of *fortituous* detection of actual defects by the fault model.

This type of test generation referred to as generation of n-detect test sets for a fault model, aims to generate N vectors to detect a fault. If total number of vectors that detect the fault(T) is such that $T \geq N$. If $T < N$, T vectors are generated to enable the detection of additional defects.

N-detect test sets using the stuck-at fault model studied in [10, 11, 12], are more suitable for logic defects in the circuit based on the fact that they use a single vector to detect each of the faults associated with the fault model. The transition fault based on the fact that it is a two-vector test set, is more suited for use as a source for surrogate model based coverage estimation when targeting delay defects; this was shown in [13]; this study also showed that the coverage of long paths also increased as n was increased. This fault model is still inadequate for the purpose of detection of the realistic defects because, in-spite of its improvements, the coverage of paths is still opportunistic.

An additional test generation methodology where the biasing of the test vectors generated due to the stuck-at fault model is removed by using an unbiased delay model is proposed in [14].

In Chapter 2 the manifestations of realistic defects were outlined, the resistive opens require a fault model that enable the detection of the smallest possible delay caused by the defect. This requires the propagation of the effect of the resistive open through the longest possible path, thus giving the fault the least possible slack by which it can remain detected. The transition fault model by virtue of the fact that it does not consider the path through which it propagates the fault to the output, is not adequate for this fault model. For example, consider the circuit that is shown in the Figure 3.1. If the path chosen is through gates g_0, g_1 rather than g_0, g_2, g_3, the amount of resistance on the resistive open defect on input g_0 that can be detected is limited; these resistive opens that would be missed could potentially

affect the functioning of the circuit.

The path delay fault model due to its already mentioned shortcomings of being exponential in the number of paths targeted would be unsuitable to detect resistive opens. Even considering paths longer than a given threshold would be ineffective because not all points where defects could occur are covered by a 100% coverage vector set under this fault model.

Detecting a good portion of the process parameter variations requires a specialized delay test set. Under the assumption of uniform process parameter variation over combinational blocks under consideration, considering the longest paths through the circuit would yield good opportunistic coverage of the process parameter variations that could cause a circuit marginality failure in the circuit.

It has been shown in [35] that the paths that can be targeted by using the non-robustness criterion are much longer than the paths targeted using the robustness constraint. Many of these non-robust tests have very low likelihood of failing, thus enabling us to have good confidence in the tests that are generated.

It is thus very important to atleast identify all the longest testable paths in the design even if some of the paths can only be tested non-robustly.

3.3 Unified Delay Fault Model

The Unified Delay Fault model is targeted to provide good coverage of resistive open defects and significant opportunistic coverage of process parameter variations.

The aim here is to combine the advantages of the transition and the path delay fault models and get good coverage of spot defects and distributed defects with reasonable complexity.

This fault model's goal is to drive both rising and falling transition through

all the nets in the circuit to enable detection of smallest possible delay effects and to enable the detection of defects affecting the longest paths in the design

3.3.1 Defect Coverage

Test generation for our fault model has two phases.

- **Phase 1** targets the paths whose delays exceed a given threshold.
- **Phase 2** targets the nets that are not covered in Phase 1 through their respective longest sensitizable paths.

The test vectors that are generated in Phase 1 also enable us to target the effects of process variations that occur across the chip apart from targeting point defects on individual nets. In current device technologies, the intra-die process parameter variation has a very low probability of occurrence. Thus for a given chip there is very high probability that in the case of variation, the process parameters vary uniformly. This is especially relevant as individual blocks considered are usually combinational blocks within latch boundaries in the case of synchronous sequential circuits. Thus we can assume an increase in the delay of all the gates forming the combinational block.

Thus testing the top few paths under these conditions ensures the detection of moderate process parameter variations even if the ordering of the long paths in the design cannot be guaranteed. This is because a path reordering that could cause these tests to fail must elevate a path under the threshold to a high enough value in the presence of process parameter variations and this would be detected by the process monitors.

Theorem 1 *All nets covered in Phase 1 are covered through their longest sensitizable path.*

Proof: *Phase 1 targets the top few longest paths in the design for test generation.*
Let I=$\{P_0, P_1, P_2, ...\}$ *be the set of paths covered in Phase 1.*
Consider a net in path P_i *which is covered in Phase 1 :* $n \in P_i$.
If $\exists Q$ *ST,* $n \in Q$ *and* $D(Q) > D(P_i)$*, where D(X) corresponds to delay of a path X,* $Q \in I$
As all the longest paths are covered in Phase 1, Q is also covered.

Point defects such as resistive opens result in increases in delay of a particular net/gate. When we deal with primitive gates, enabling both rising and falling transitions on all the inputs of the gates enables detection of all resistive opens in the gate as has been outlined in Section 3.3.1. Our fault model drives transitions through each of the gate inputs as shown in Theorem 2.

Theorem 2 *All the nets in the circuit are covered by the test generation algorithm.*
Proof*: The set of all nets in a combinational circuit can be given by,*
$S = \{PI, GO, GF\}$ *here, PI corresponds to the set of all primary inputs, GO corresponds to the output of gates, GF corresponds to the fanouts of the gate outputs.*

- *The primary inputs and the gate outputs can be targeted by weighting individual gates*
- *The fanouts of gates are targeted by addition of the weights on two gates in the circuit*

Let the set of all primary outputs be given by PO; it can be easily shown that $PO \in GO, GF$*; thus all the nets in the circuits are covered.*

Further, propagating all the tests through the longest path possible through the net for the particular transition minimizes the increase in delay due to the delay faults that can be detected.

There are two major cases that we need to consider when generating tests to detect effects of resistive opens in the CMOS circuits. The first case is when we assume that there is a maximum of one resistive open defect in the circuit under consideration. The more general case concerns the possible presence of multiple resistive open defects in the circuit.

The fact that we generate vectors to drive both rising and falling transitions through each net in the circuit and propagate it through the longest testable path in the circuit, enables the detection of all possible single resistive open defects that can occur in the given combinational block. In the presence of multiple resistive open faults, each resistive open fault can be considered independently of the other resistive open faults in the circuit. Thus the tests that are generated are still effective. The defect occurrence not targeted is the occurrence of multiple resistive opens, many of which should be included in a given path to be detected. Invalidation of tests can also occur due to the presence of delay on the side input which has a $C_i \rightarrow NC_i$ when the on-path input has a $NC_i \rightarrow C_i$ transition.

This model differs from the line delay fault model proposed in [82] in that we target paths and generate tests under the non-robustness criterion to enable an improvement in the detection probability, and also due to the fact that we target the longest paths exceeding a given threshold which gives us a way to target cumulative effects of delays along a path.

3.4 Test Generation Methodology

Test generation for the fault model is derived from the vigorous sensitization procedure outlined in [56].

3.4.1 Implicit Identification of Critical Paths

Consider a combinational block C, consisting of gates of types AND, OR, NAND, NOR, NOT. The methodology incrementally identifies sub-paths that are sensitizable from the primary inputs. Once the sub-paths are identified, all the sensitizable paths associated with any currently identified sub-path are evaluated to identify the longest path through the current subpath. This provides the advantage of not having to repeatedly determine the sensitizability of a given sub-path as they might occur in multiple paths in the combinational block. The procedure uses a five valued logic consisting of $\{0, 1, X, R, F\}$; R(F) refer to rising(falling) transitions.

The progressive identification of the longest path in the combinational block is guided by the sum of *source delay* and the *sink delay* of the candidate gates which are chosen from the set of fanouts of the gate under consideration (G_i).

Definition 7 *Sink Delay of a gate in a given combinational block is the delay of the longest path from the gate to one of the primary outputs.*

Definition 8 *Source delay is the delay of the longest path from a primary input to the gate under consideration.*

These delay measures are calculated using the maximum possible delay of the gate. These two static measures help in biasing the implicit path search algorithm towards the longest paths (although structurally longest). Thus a combination

of implicit sensitizability evaluation and structural preprocessing results enable this technique to yield significant savings in complexity.

A complementary measure is used to evaluate the actual delay of paths in the combinational circuit, this is the arrival time.

Definition 9 *The Arrival Time is the time required for the output of a gate G_i to settle to a stable value.*

This delay value is dependent on the arrival time of the signal in the gate preceding the current gate in the path under consideration. It is also dependent on whether some of the inputs settle to a controlling values. When some of the inputs i_1, i_2 settle to controlling values, the arrival time is calculated as min(Ar(i_1),Ar(i_2)), where Ar(x) denotes the arrival time of a signal X in the circuit. Arrival time is a measure which is based more on functionality than the source and sink delays that were outlined earlier. The functional delay measures can be derived for each of the gate types that would be used in the circuit representations.

The gate identified as a candidate gate to follow the current gate in a path is called the *successor gate*. The conditions that need to be satisfied for a fanout to be classified as a successor are the following.

a) the gate has ($C_{G_i} \rightarrow NC_{G_i}$) transition with a settling time before the on-path transition or a NC_{G_i} or an X value at the off-path input, and

b) the gate has the maximum sink delay among all the gates satisfying the above condition.

Figure 3.5 gives an example gate with a fanout of 3. There is an option of propagating the transition either through gate X or gate Y. If the transition on line

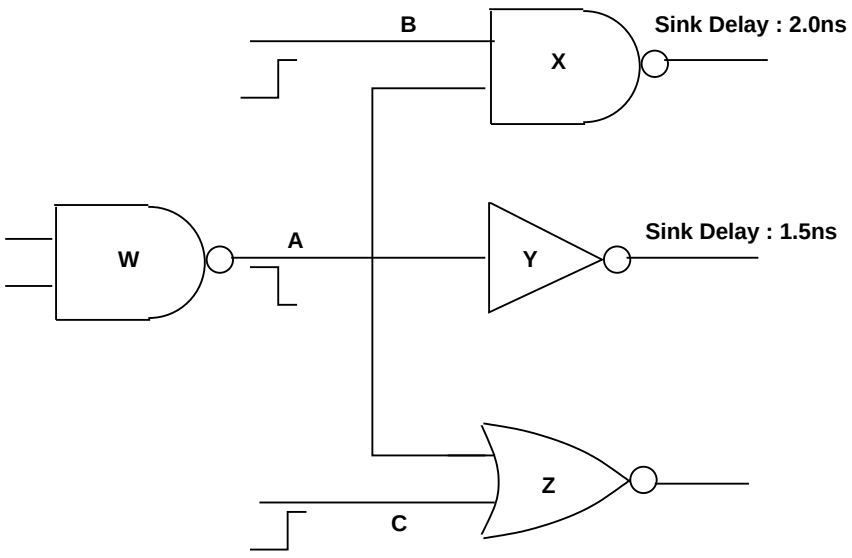


Figure 3.5: Successor Selection

B stabilizes before the transition on line A, gate X is chosen because it has larger sink delay. If the transition on line B stabilizes after A, gate Y is chosen.

Thus a transition initiated at a particular primary input is progressively propagated through subsequent successor gates till a primary output is reached. The side input of the successor gates are justified and implication is performed such that the transition is propagated through the successor gate. When we exhaust evaluating all the candidate gates, backtracking is performed where a recently assigned primary input is inverted and the implication is performed.

For the off-path gates in the circuit which have a transition at the inputs, the following conditions are imposed. Let us consider gate G_i which does not lie on the path being tested by the two-vector delay test. Let N (N $>$ 1) represent the number of inputs of G_i. Let $ST_{max}(C_{G_i} \rightarrow NC_{G_i})$ and $ST_{min}(C_{G_i} \rightarrow NC_{G_i})$ represent maximum and minimum stabilization time for $C_{G_i} \rightarrow NC_{G_i}$ transitions on the

input of the gate. $ST_{max}(NC_{G_i} \rightarrow C_{G_i})$ and $ST_{min}(NC_{G_i} \rightarrow C_{G_i})$ represent the same for $NC_{G_i} \rightarrow C_{G_i}$ transitions. The conditions that mark the stabilization times for the off-path gates with transitions in the case of the two-vector test include the following.

a) When N - 1 inputs of G_i have NC_{G_i} with one input having a $C_{G_i} \rightarrow NC_{G_i}$ or a $NC_{G_i} \rightarrow C_{G_i}$ with a stabilization time ST, the stabilization time at the output of the gate is (ST + $w_{G_i}(x_{G_i})$).

b) When the number of $C_{G_i} \rightarrow NC_{G_i}$ is greater than 1 with no $NC_{G_i} \rightarrow C_{G_i}$ transitions the stabilization times of the outputs can be given by ($ST_{max}(C_{G_i} \rightarrow NC_{G_i})$ + $w_{G_i}(x_{G_i})$).

c) When the number of $NC_{G_i} \rightarrow C_{G_i}$ is greater than 1 with no $C_{G_i} \rightarrow NC_{G_i}$ transitions, the stabilization times of the outputs can be specified by ($ST_{min}(NC_{G_i} \rightarrow C_{G_i})$ + $w_{G_i}(x_{G_i})$).

d) When the number of $NC_{G_i} \rightarrow C_{G_i}$ transitions and the number of $C_{G_i} \rightarrow NC_{G_i}$ transitions are greater than zero, the stabilization time at the output is given by ($ST_{min}(NC_{G_i} \rightarrow C_{G_i})$ + $w_{G_i}(x_{G_i})$).

x_{G_i} corresponds to the transition propagating through the gate. This procedure uses an ATPG engine based on the PODEM algorithm.

The identification of paths through individual nets is achieved by adding a biasing parameter $\rho \in R^+$ to the rising/falling delays of the gate. This affects the sink delay seen by all the gates on all paths passing through the gate and that appear before the gate under consideration in the paths. The biasing parameter is applied to the rising and falling transition delay separately to drive particular transitions through the nets. The size of ρ can be determined to be a value greater than the delay

of the longest sensitizable path in the combinational block under consideration. When the biasing parameter is added to only one gate, the ability to propagate the transition through a net is still lacking. In order to bias the path tracing through a net, the biasing parameter is added to two gates in the combinational block under consideration.

When identifying the longest path, in order to identify all the critical paths that are above a given threshold, all the fanout points are identified. Once the paths are identified, the fanout stems at these fanout points which have the required delay characteristics (*sink delay* being above the chosen threshold) are targeted by adding biasing parameters to these "interesting" fanout points.

In the first phase, all the critical paths above a given threshold are identified. The two-pattern test vectors that sensitize the critical paths are also generated. The nets that are not covered by the first phase of the process are then identified. The biasing parameter is added corresponding to the target nets and to the rising or falling transition based on the transition required at the particular net. This biases the vigorous sensitization approach to generate the longest sensitizable path through the target net.

3.4.2 Determining the Real Delay Values

In our methodology, a technique to estimate the real delay values of the individual gates is used to enable the identification of the true critical paths in the design. This eliminates the need to use unit delay values for each gate which does not yield the true critical paths in the design [85].

The estimation of the delay of paths in the circuits become important and become even more relevant because of the testers lagging behind in the frequencies

they can handle. As a result, the accuracy of testers in determining the failing frequency is limited.

We incorporate actual delay values in our model by lumping the delays of gates into delay buffers which are added at the inputs of the gates. This lumping of the delays at the gate inputs allows us to account for the input-output path delays that are associated with the actual delay values. The Standard Delay Format (SDF) [46] description of the circuit is used to identify these actual delay values.

The actual delay values of circuit elements are available from the SDF description of the circuit. The SDF description was obtained using Synopsys Design Analyzer [47], the delay values were represented using (MAXIMUM / NOMINAL / MINIMUM) values for rising and falling transitions. As the test generation scheme here uses a single value for the rise and the fall delay values corresponding to the gate, the nominal values from the SDF was used as an estimate for the delay of the gate.

The nominal delay was chosen because the maximum value would have given a very pessimistic estimate of the delay of the circuit and choosing the minimum value would give a very optimistic estimate of the path delay of the circuit. Seen in the context of manufacturing test, having an optimistic estimate of the critical path delay would leave the possibility of defects not being tested for, and a pessimistic estimate might lead to too many paths having to be tested.

The values determined were used in producing circuits in a format suitable for CRITIC, the format also requires the representation of all complex gates in terms of primitive gates. This conversion of complex gates to primitive gates is specified in a library file that is specified by the user. In the case of complex gates with only one output, the lumped delay values are associated with buffers at each input of the complex gate.

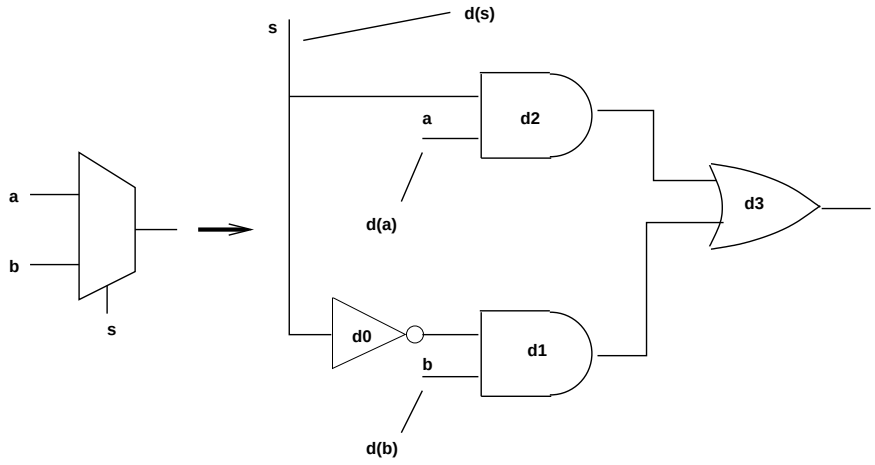


Figure 3.6: Complex Gate Transformation

An example of mapping the behavior of complex gates to primitive gates is shown in Figure 3.6; here a two 2x1 multiplexer is transformed to a block consisting or AND, OR and NOT gates. The d_0, d_1, d_2, d_3 represent the delays of the gates (could be rising/falling delays), and the lumped delay values at the inputs of the gates correspond to the input-output delays seen in the SDF specification of the circuit.

3.4.3 Effectiveness of Tests

We deal with synchronous sequential designs. The timing in these designs is fixed by determining the longest sensitizable path in the design. A given two-vector test detects a delay fault on a path if the effect of the point/distributed delay defect on the path exceeds the slack of the path. Considering the longest paths that pass through a net detects the defects on the lines with maximum probability.

With an aim to try and determine the smallest possible delay defects in the

circuits, we identify the sensitizability of the paths based on the non-robust sensitizability criterion. Various invalidation conditions have been outlined for delay faults in [86]. We outline the conditions to overcome these invalidation mechanisms in the context of our fault models.

Lemma 1 *The suppression of transitions on the targeted path in the absence of delay faults is eliminated in our test generation methodology.*
Proof: *The actual delay values used in conjunction with the settling time conditions outlined in the Section 3.4.1 enable us to accurately specify the arrival time of transitions on the side inputs of the on-path gates in the absence of other delay faults. The conditions imposed on the side inputs of the gates of candidate on-path gates enable them to settle to NC_i values before the on-path input. Thus the suppression of transition on the path is eliminated.*

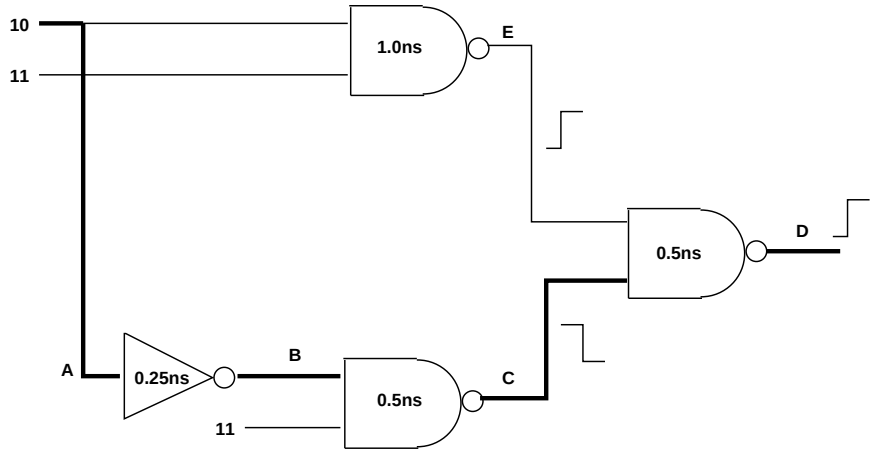


Figure 3.7: Transition Suppression

We use an example provided in [86] to illustrate this case. The example circuit along with the input vectors is shown in Figure 3.7; the path under test is shown

in bold. The invalidation for the test shown for the path {A, B, C, D} happens when the $C_{NAND} \rightarrow NC_{NAND}$ on E occurs after the $NC_{NAND} \rightarrow C_{NAND}$ transition on C. Our methodology avoids this case because of the conditions imposed on the selection of the successor gates outlined in Section 3.4.1.

Lemma 2 *Early launching of transitions is also avoided in the test generation methodology.*

Proof: *Getting an accurate estimate of these final settling times is achieved by the conditions imposed to evaluate the settling time of gates not on the path being tested. These conditions which were outlined in Section 3.4.1 enables the last possible settling time to be considered for the purpose of test generation. The requirement for the side inputs to settle to NC_i values eliminates the condition under which this early launch of transitions occurs.*

The other major cause for the invalidation of delay tests is the presence of path delay faults on the side inputs of the gates on the target path. This is the case only when the side inputs have a $C_i \rightarrow NC_i$ transition with the on-path inputs having a $NC_i \rightarrow C_i$ transition. This delay fault could occur due to a variety of causes. These include the presence of a distributed delay defect on the side inputs of the paths and the presence of an additional resistive open on the path leading to the side input.

The scenarios under which a distributed delay defect on a path leading to the side input leads to the invalidation of the delay test is considerably reduced because the test set also targets the longest paths in the given combinational block. Only conditions where the defects due to these variations do not affect the longest paths but still cause a reordering of transitions at the sub-path levels could result in the delay tests being invalidated. One condition is a non-uniform process parameter

variation across the combinational block under consideration which as mentioned earlier we can safely assume not to happen. Another contributing condition is a presence of very small process parameter variation. The masking of delay tests due to multiple delay faults with the individual delay faults remaining undetected remains fairly remote.

This delay fault model has a good tradeoff between the test quality and the test application difficulty. The test application difficulty can gauged by the number of vectors required to test the given circuit, this difficulty is minimized because the number of vectors in our fault model is proportional to the number of nets in the circuit rather than the number of paths. The test quality is reflected in the size of the delay defects that can be detected by the test set unlike the classical stuck-at fault model, and our fault model targets this aspect effectively.

3.5 Results

As described above, the test generation tool for the fault model was obtained by modifying the timing verification tool, CRITIC. In order to enable the use of real delay values in determining the longest paths in the circuit, we first generate the Standard Delay Format (SDF) file and then use it to generate the delay buffers in the file which is in a format suitable for the tool that we use. The experiments are based on a 0.35 micron process.

We generated tests for some of the ISCAS combinational benchmarks. We first determined the structurally longest path in the circuits. The threshold for determining the longest sensitizable critical paths in Phase 1 was fixed at 95% of the delay of the structurally longest path. This yielded only very few paths because of the fact that the circuits were not timing optimized. The times were measured on a

Sun UltraSparc II with 1 GB of RAM.

The nets that were not covered in the paths that were identified in Phase 1 in our methodology form the fault set for Phase 2. In this phase, weights are assigned to the delay buffers according to the nets that have to be covered by the appropriate longest sensitizable path. The maximum number of faults in this case, and the number of paths identified in this phase, is twice the number of nets, since we target both the rising and falling transitions for each net.

Table 3.1 gives the results of the test generation. Column 2 gives the number of nets in the design, Column 3 gives the fault coverage which is the percentage of the nets covered for rising and falling transitions.. The cases where the fault coverage is less than 100% are a result of no sensitizable paths being found and, in some cases, the test generation tool aborting due to limited backtrack. Column 4 gives the percentage of the nets for which the test generation tool aborted, Column 5 gives the percentage of nets for which no sensitizable path could be found and Column 6 gives the CPU time taken to generate the test vectors.

Table 3.1: Test Generation Results

Circuit	# of Nets	Cov (%)	Aborts (%)	Untestable (%)	Time (mins)
c880	598	100	0	0	3.5
c1908	603	86.23	8.14	5.63	363.15
c2670	1118	94.82	0	5.18	263.88
c5315	2751	98.33	0	1.67	311.85
s1196	865	99.77	0	0.23	70.25
s1238	886	95.94	0	4.06	107.56
s1423	951	96.21	3.05	0.74	233.45
s5378	2165	98.57	0.65	0.78	299.88
s38584	17936	98.28	0.25	1.47	9256.14

The number of faults using this fault model is twice the number of nets in the circuit. The phase 1 of this methodology adds a few more paths to be targeted. The number of paths targeted is linear on the number of nets in the circuit. This number of faults is thus proportional to the number of faults with the gate delay/transition delay model and the segment delay fault model (with segment of length 1) while giving us the ability to target the smallest detectable delay defect. As mentioned earlier, when compared to the path delay fault model, the number of faults is significantly lesser, and scales less drastically. Graph 3.8 shown the significant difference in the number of faults between the gate delay fault model and the path delay fault model [31].

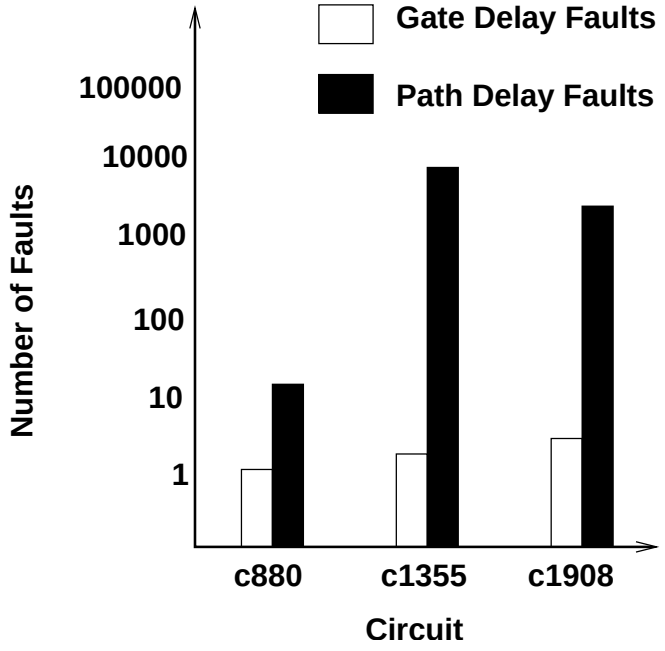


Figure 3.8: Fault Count

Further, in all the paths that were identified, the percentage of side inputs of

the gates in the path that made the test generated non-robust were less than 2% of the total number of side inputs in all gates having a $NC_i \rightarrow C_i$ transition at the on-path input, only c1908 had a value higher than this at 7%. This directly points to a very low probability of a path failing due to delay faults (distributed and point) at the side inputs.

The test generation complexity is dependent on the number of faults that need to be targeted and the number of gates in the circuit. The complexity scales linearly on the number of faults that are targeted and exponentially on the number of gates in the circuit. The number of faults in the proposed fault model is linear on the number of gates, this minimizes the contribution of this parameter to the complexity minimal. Using implicit sensitization for identifying the sensitizable long path also yields an additional complexity improvement over enumeration based methodology. Though the worst-case complexity is similar for implicit sensitization and explicit sensitization based ATPG, the average-case complexity of implicit sensitization based ATPG (which was proposed here) is significantly better because for the majority of cases, evaluation of a subpath to be unsensitizable eliminates multiple paths.

When compared to other recent approaches to delay test generation [93, 94, 95, 44], the proposed methodology is superior because the methodologies still identify all the paths and then try to generate tests for the paths. The complexity of these methodology are analogous to the complexity of delay test generation based on explicit sensitization.

As a comparison with existing approaches, Table 3.2 gives the percentage of the longest paths identified by our fault model that are detected by test patterns targeting sequence dependent defects. We show the coverage for four representative sets of test patterns generated using the following approaches.

A. Single stuck-at (SSA) test generation.

B. Tests generated using the transition fault model.

C. SSA patterns padded with zeros.

D. SSA patterns padded with ones.

E. Union of all the above patterns.

The stuck-at test vector set has a 100% stuck-at fault coverage and the transition vector set has a 100% transition fault coverage. The stuck-at patterns were then padded with zeros and ones under the heuristic that this would generate transitions at the inputs of the targeted combinational blocks. The final pattern set is a union of the preceding pattern sets thus targeting all the paths that were exercised by the above fault models.

As can be seen, only a fraction of the longest paths identified are exercised by these test sets. This is the case even with a test set consisting of the union of patterns generated under the first four methodologies. This demonstrates the inadequacy of existing techniques for detecting delays due to resistive opens, and the usefulness of the proposed approach.

3.6 Conclusions

An improved fault model was presented that yielded potentially high resistive open coverage and a good opportunistic coverage of defects due to process parameter variations. This fault model helps in overcoming the deficiencies in the path, gate and the transition delay fault model and helps target defects that are found more commonly in current generation process technologies. The test generation approach

Table 3.2: Longest Path Coverage

Circuits	Path Coverage (%) Test Generation Methodology				
	A	B	C	D	E
c880	19.13	25.56	23.62	21.52	31.24
c2670	11.80	22.49	14.99	15.59	29.48
s1196	33.44	45.56	32.31	31.01	68.66
s1238	27.32	48.11	46.53	31.41	59.29
s1423	16.87	23.20	17.58	19.29	27.40
s5378	32.52	51.58	34.32	35.77	61.17

presented helped in limiting the complexity of generating tests by using implicit sensitization. The coverage obtained using the fault model was evaluated against re-ordered stuck-at fault models and transition fault models and the coverage of long paths in all these fault models was low, thus the unified fault model which targeted the longest paths through nets was more effective in the defects we are targeting.

Chapter 4

Reducing Test Application and Test Generation Complexity

Though the test generation procedure presented in the Chapter 3 is efficient, some further optimizations can be made to reduce the test generation time and the test application time.

In this chapter, a technique to help reduce the number of faults that need to be targeted and consequently reduce the number of vectors to be generated is presented. A technique to use reconfigurable scan to further reduce the test application time is then introduced. Also presented are techniques to reduce the number of backtracks by storing the subpath sensitization information and by a better threshold selection methodology.

4.1 Test Application Complexity

The test application techniques used for delay testing as outlined earlier are primarily scan based. In a scan-based delay test methodology, an enhanced-scan based ap-

proach to test application is the ideal configuration to keep the dependence between the ATPG methodology and the test application methodology to a minimum. The advantages of an enhanced-scan based methodology stems from the high degree of controllability and observability that it provides on the design; this is required especially in the case of delay test application where the requirement is to apply a two-pattern vector in consecutive cycles on a latch boundary.

In-spite of the benefits of a enhanced-scan based methodology, the test application time in a enhanced-scan design is high under high volume production [83, 84]. Consider the number of scan elements in the design under test to be N_S, and the number of test vectors to be N_V; the total test application time (T) can thus be given as $T = N_V(N_S + 1) + N_S$; the test application time is thus proportional to the number of vectors and the number of latches on the scan path. To reduce the number of scan elements, multiple scan chains, parallel scan have been proposed. Though these techniques are effective, the area overhead incurred in using these offsets the benefits. Reducing the number of vectors offers an alternative to reduce the test application time. Fault dominance/equivalence rules can be used to eliminate the faults in a delay test framework. The following techniques enables the reduction of the number of faults to be targeted and consequently the number of vectors that need to be applied in the test application phase.

4.2 Reducing Test Application Complexity

4.2.1 Reducing the Number of Faults

The number of faults for a given circuit is proportional to the number of nets in the circuit for the Unified Delay Fault model. This set of test vectors obtained (or the

faults targeted) can be drastically reduced if certain fault equivalence conditions are considered among the nets that are targeted.

The Unified Delay fault model targets the longest paths through nets for both the rising transitions and the falling transitions. Thus for example in a single fanout gate, the longest path among the paths found at the inputs of the gate also serves as the longest paths through the output. Thus the test generation tool need not attempt to generate a separate test for the output of the gate when the longest paths through the inputs have been determined.

Fault Equivalence Rules

The number of faults that are targeted can be reduced by applying some rules to the set of nets that are targeted.

a) For gates with single outputs, the longest path over all the inputs of the gate covers the output

b) For gates with multiple fanouts, the fanout stem included in the longest path over all the inputs is ignored but the longest paths have to be reevaluated for the other gates in the circuit.

Lemma 3 *Considering a single output gate, the longest among all paths through the inputs covers a fault on the output.*

Proof : A gate can be considered to be a vertex in a directed graph, with multiple incoming edges and one outgoing edge as shown in the Figure 4.1. The set of inputs correspond to the vertices $I = \{I_0, I_1, I_2\}$*, and the outputs correspond to* $O = \{O_0, O_1, O_3\}$*. A path in this circuit would correspond to a path in the graph from vertex* $X \in I$ *to vertex* $Y \in O$*. Consider a vertex V as shown in the Figure 4.1,*

when V is contained in a path the edge $\{V, V_0\}$ and an incoming edge is included in the path. Thus, the longest path through the vertex is the longest path containing one of the inputs of V also contains the outgoing edge from the vertex. Further, only a path through the vertex contains the output edge $\{V, V_0\}$; thus the longest path through $\{V, V_0\}$ always passes through the vertex. Thus the longest of the paths through the inputs is also the longest path through the output.

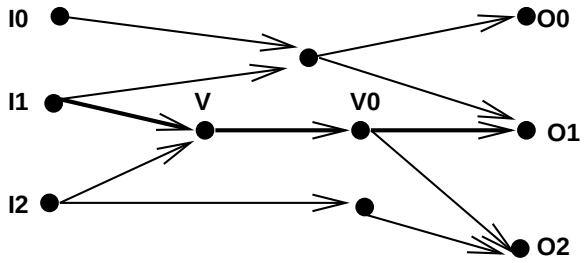


Figure 4.1: Fault Equivalence

Lemma 4 *In the case of the output of a gate having a fanout, the proof is a simple extension of the preceding proof.*

An example of this fault collapsing is shown in Figure 4.2; the initial set of defect locations that need to be tested for number 8. When the faults are collapsed, the set of fault locations that need to be targeted correspond to $\{f_0, f_1, f_2, f_3, f_5\}$. The faults f_4, f_6, f_7 are respectively covered by the paths targeting the inputs of the gate.

The first type of fault collapsing can be done statically by considering the circuit as a graph with the gates being vertices. Faults on the outputs of all the gates with a single input and single fanout are dropped under the equivalence conditions.

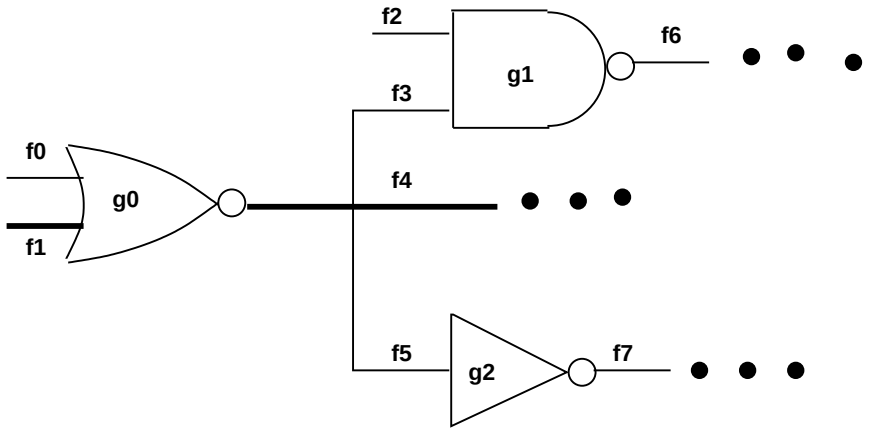


Figure 4.2: Fault Collapsing

No information on the faults that are detected is required; this is because the search for a critical path through a particular net implicitly searches through all possible paths till a given threshold is reached and from the proof given above; when both the inputs are thus covered, all the paths through the output are also targeted.

The second type of equivalence needs to be done dynamically as the longest paths are evaluated. This kind of a **dynamic fault collapsing** requires an ordering to be maintained in generating tests for each net in the circuit. The faults need to be ordered such that for all the gates in the circuit, the inputs of the gates are targeted before the outputs of the gates. This enables identifying the fanouts of the gate that need to be evaluated before a given net are taken up for evaluation. In the case of a single output gate with multiple fanouts, the minimum number of faults that can be dropped can range from 1 to N_f where N_f is the number of fanouts in a particular gate. Only one fault can be dropped when all the long paths through the inputs pass through a single fanout and N_f faults can be dropped if $N_i = N_f$ where N_i is the

number of inputs and the longest path through each input passes through a distinct fanout. Examples of the two cases are shown in the Figures 4.3,4.4. In Figure 4.4, the various paths passing through the gate are highlighted by solid, dashed and dotted lines respectively.

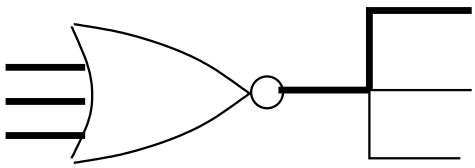


Figure 4.3: One Fanout Eliminated

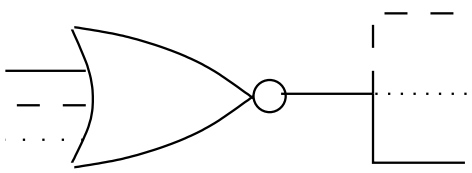


Figure 4.4: Multiple Fanouts Eliminated

When gates used in the combinational circuit have multiple outputs, the conditions that arise are similar to gates with single outputs and multiple fanouts.

Results

The given approach for the fault reduction is applied to some of the ISCAS benchmark circuits. Table 4.1 shows the reduction in the number of faults that need to be targeted when the number of faults are reduced based on the equivalence conditions specified above.

Table 4.1: Reduction in the Number of Faults

Circuit	Orig. Num of Faults	Maximum faults to Target	Reduction (%)
c880	598	338	43.5
c1355	868	471	45.8
c1908	603	322	46.6
c2670	1118	597	46.6
c3540	1396	763	45.4
c5315	2751	1494	45.7
c6288	4716	2387	49.4

The table looks just at the number of fanouts at each of the gates and gets a value of the number of the nets that need to be targeted. The reduction in the test application time when these circuits are assumed to be combinational blocks in a sequential design with a scan chain length of 1000 is shown in the Graph 4.5. Further reduction in the number of nets targeted can be obtained using the dynamic fault collapsing methodology. The methodology yields good reduction in the number of vectors to generate even without the dynamic fault collapsing.

4.2.2 Reducing the Test Application Complexity Using Reconfigurable Scan

When functional justification is used for application of delay test vectors to test a given circuit for delay faults, there is a need to adhere to the requirement to test the combinational block between the PI and the first latch boundary before the second and successively for each successive combinational block. This is to avoid the effect of a delay fault on a module preceding it to affect the tests generated for the combinational block under test. This is because the vector V_2 is propagated from

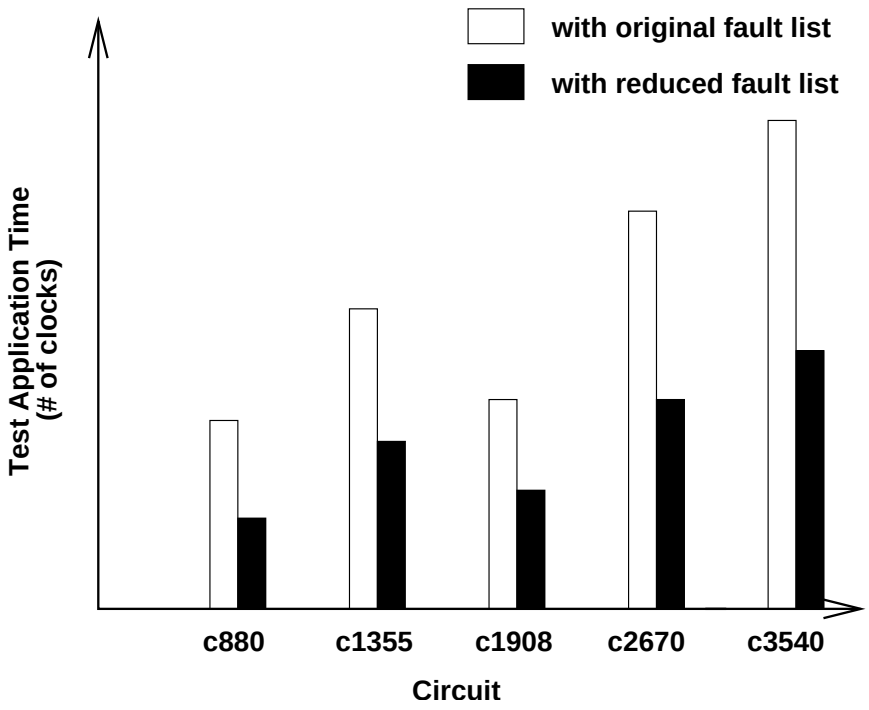


Figure 4.5: Reduction in Test Application Time

the preceding latch boundary when the clock is applied for the first vector.

This requirement imposes the condition where only one block can be tested at a particular time when using functional justification based test application; this could cause an increased test application time. Reconfigurable scan offers a useful methodology to considerably reduce the test application times without the use of enhanced-scan latches, thereby avoiding the area penalty associated with it.

Further, this method obviates the need for multiple scan chains in the circuit which leads to the need for multiple pins to be added to the primary inputs of the chip under test. We achieve this by using a multiplexer to connect scan-chains at the output of each succeeding block to the scan-out pin of the chip. Further,

we can perform a similar multiplexing for the scan-in pin. These along with a select pin which enables switching from one scan in/scan out to the next makes the requirement of additional input pins at 3.

This approach gives us a considerable advantage over using functional justification approach using standard scan methodology without any reconfiguration modes added to the scan chain. Compared to enhanced scan methodology, we achieve a reduction in the area complexity.

Functional Justification under standard-scan

When a given circuit has single scan chain with standard scan latches, a functional justification based test application approach becomes costly in terms of test application time because of the requirement mentioned above. Consider a sequential circuit(**S**) which is a full-scan design with a single scan chain. Let the length of the scan chain be **L**. Let the number of combinational blocks demarcated by the latch boundaries formed by these scan chains be **N**. Let the number of two-vector tests forming the test set for the circuit be **NV**. Let the number of vectors to test each of the combinational blocks be NV_i S. T. i = $\{1, ... ,N\}$, where i represents the combinational block's position from the primary input.

Under this base case, the number of cycles required to apply the test vectors so that we can avoid the possibility of masking outlined in the previous section is

$$Number of cycles = \sum_{i=1...N} NV_i \times L \tag{4.1}$$

For comparison, the number of cycles required to apply the same set of vectors under enhanced scan is $max(NV_i) \times L$ where, $max(NV_i)$ is the maximum number of test vectors required among all the combinational blocks in the design. Assuming that the number of latches in each of the latch boundaries to be equal in number **n**,

under our reconfigurable scan methodology, the test application time can be given as

$$Number of cycles = \sum_{i=1...N} NV_i \times 3 \times n \tag{4.2}$$

As can be seen, the test application time for test vectors based on functional justification is much reduced with a reconfigurable scan chain when compared a framework with standard scan.

Multiple scan chains can be provided in order to obtain a reduced test application time, but there is a requirement to add multiple input pins to the chip in order to enable this.

Reconfigurable scan chain

Reconfigurability can be built into the scan chain with the addition of only one test pin apart from the scan-in and scan-out pins. The reconfigurability can be achieved by adding a multiplexer which chooses between various inputs and outputs of the scan-chains formed by each of the latch boundaries identified.

This can be achieved by using a multiplexer design as shown in Figure 4.6. The test pin is connected to the clock of a simple ripple carry adder which increments by one whenever a transition is received on the test pin. This in turn is connected to the select inputs of the multiplexer, thus enabling the successive scan chains for various combinational blocks in the design.

Thus the current value in the counter refers to the combinational block for which the associated scan latches need to be enabled.

A reset line of the counter is connected to the reset line of the chip such that the line is reset at the beginning of the test procedure. This enables the testing of the combinational blocks successively from the primary inputs of the circuit under

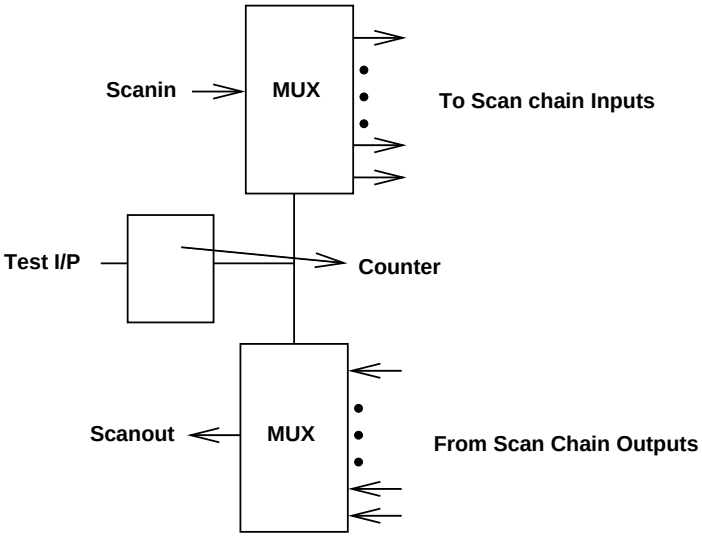


Figure 4.6: Circuit to Enable Reconfiguration

consideration.

As shown in the figure the only additional pin needed for this configuration is a test pin which is fed with a signal similar to a clock signal which causes the counter to increment.

4.3 Decreasing Test Generation Complexity

4.3.1 Storing the Sub-path Sensitizability Information

The test generation approach yielded a drastic reduction in complexity when compared with the techniques where, the structurally longest paths are identified first and each of the structurally longest paths is evaluated for sensitizability.

Because of the implicit path sensitizations used in generating the delay tests,

the complexity of the technique was highly dependent on the way sensitization of sub-paths was handled while determining the longest paths through given nets. In this section, we improve the determination of the sensitizability of the sub-paths, thus drastically reducing the complexity of the tool in general, especially for cases when we deal with circuits with high fanout.

The implicit enumeration based test generation algorithm does not store the sub-path information; thus the search causes an additional complexity in the case of circuits with high fanout. As shown in Figure 4.7, when there are a large number of fanouts for nodes in the circuit, the sub-path preceding each of the fanout points has to be reevaluated multiple times on the way to evaluating the sensitizability of the entire path. As can be seen in Figure 4.7, the subpath {A, B, C} has to be reevaluated for each of the fanouts of the gate C; when such information can be remembered, a drastic reduction in complexity can be achieved.

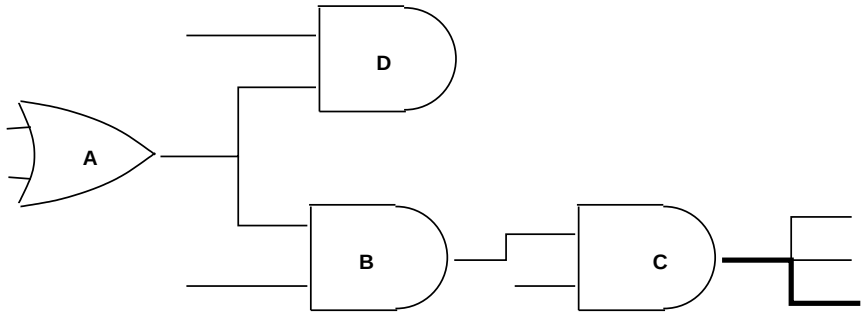


Figure 4.7: Sub-path elimination

This complexity can be avoided and the re-evaluations of the critical paths can be minimized by storing the results of one attempt at sensitization of the sub-path. Thus when this sensitization information is stored, further attempts at sensiti-

zations can be avoided after tracing through the targeted subpath only once. When a sub-path is traced and it finds a conflicting assignment, the tool evaluates all the possibilities before it considers alternate paths. Thus, when a path is abandoned in one search, it is a proven unsensitizable path for the particular value of the input transition.

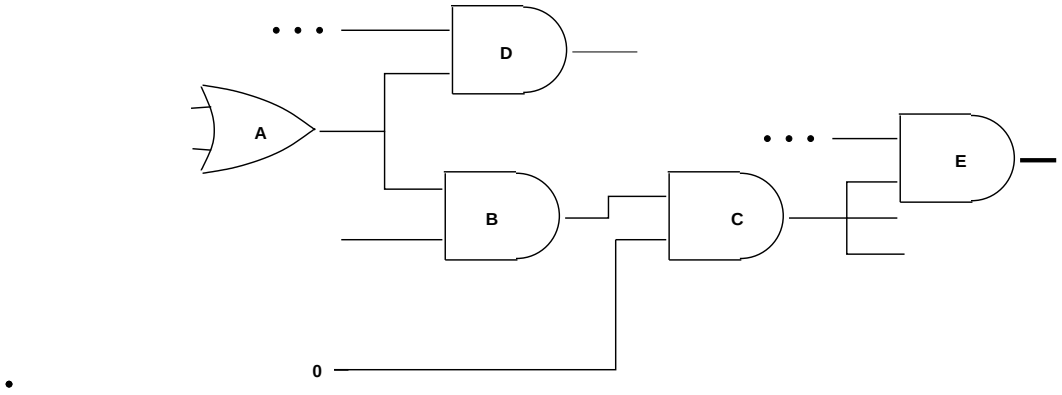


Figure 4.8: Sub-path elimination example

Therefore, as can be seen in Figure 4.8, the path {A,B,C} is not sensitizable because of the 0 at the off-path input of C. Thus when we try to generate a longest path through the output of the gate E (which is shaded for better illustration), due to the sink delays that are seen by the tool, the tool starts tracing through the gates and initially follows the sub-path {A,B,C}. The proposed methodology disables any additional evaluation of the path for sensitizability because of the fact that the sub-path was already evaluated and found to be unsensitizable for the transition targeted at the input of the path.

The sensitization information is stored for each of the sub-paths indexed by the path and the transition at the input of the sub-path. At the start of every sensi-

tization routine, the gate is checked for a match with all the sub-path information that is is already stored in the database.

When a match occurs between the path being sensitized and an already determined unsensitizable path, the sensitization ignores the current path. Any path that contains this sub-path would be unsensitizable and can be ignored.

This fact saves a considerable amount of complexity that is caused by the sequence of backtracks that is required in determining that sub-paths are unsensitizable. These accumulate for each of the sub-paths that evaluate to being unsensitizable, and eliminating the redundant searches yields drastic savings.

Combining the delay test with the subpath sensitization approach proposed in this method helps us achieve further savings over what is provided by the path selection approach (where the exponential complexity of considering all the paths was eliminated).

4.3.2 Reordering the Lengths of Paths that are Targeted

In the search for the sensitizable paths above a given threshold, the order in which the algorithm chooses thresholds significantly affects the number of backtracks encountered and consequently the complexity of the algorithm. The current algorithm starts from a value of the threshold close to the Structurally Longest Path Delay (SLPD) and progressively searches for sensitizability for smaller delays.

When determining the existence of unblocked paths, the implicit enumeration algorithm also evaluates whether the delay of the projected path is greater that the targeted threshold. In our methodology, the existence of a sensitizable path with delay longer than a small threshold is first evaluated before the identification of the longest path through the net is made. This reduces the number of backtracks by the

implicit sensitization tool, and consequently the complexity seen by the tool. This is because for many of the nets which are targeted, the longest sensitizable path through the net is much shorter than the SLPD.

The overall strategy is given in the Figure 4.9; here, the threshold is chosen at a low value to determine the presence of atleast on path close to the critical path delay. The implicit sensitization approach is then used in conjunction with the sub-path storing mechanism to get an overall saving in the test generation time.

4.3.3 Results

The methodology was evaluated on the ISCAS benchmark circuits. Table 4.2 gives the reduction in the number of backtracks that are seen when the preceding two optimizations were incorporated into the implicit path enumeration framework. The reduction in the test generation time is given in Column 5.

Table 4.2: Complexity Savings

Circuit	Number of Faults	Original Number of Backtracks	Final Number of Backtracks	TG Time Red (%)
c880	598	24821	5772	64.28
s1196	865	143807	18033	69.75
s1238	885	238046	24136	47.69
s1432	951	98913	29085	33.66
s5378	2165	95348	68383	22.68
s9234	6183	134911	88684	34.97
s38584	17936	183206	98681	29.32

The reduction of the backtracks that is seen here contributes to an improvement in the performance of the test generation tool.

4.4 Conclusions

Techniques to improve the efficiency of test generation were proposed. A dynamic fault collapsing approach which drastically reduced the number of faults to be targeted was first proposed; this helps in reducing the test generation and the test application complexity. A reconfigurable scan based methodology for functional justification based test application technique helps in further reducing the test application time. Two techniques for further reducing the test complexity, one based on smart threshold reduction and the other based on storing sub-path information, were also presented and results were presented on circuits of medium complexity.

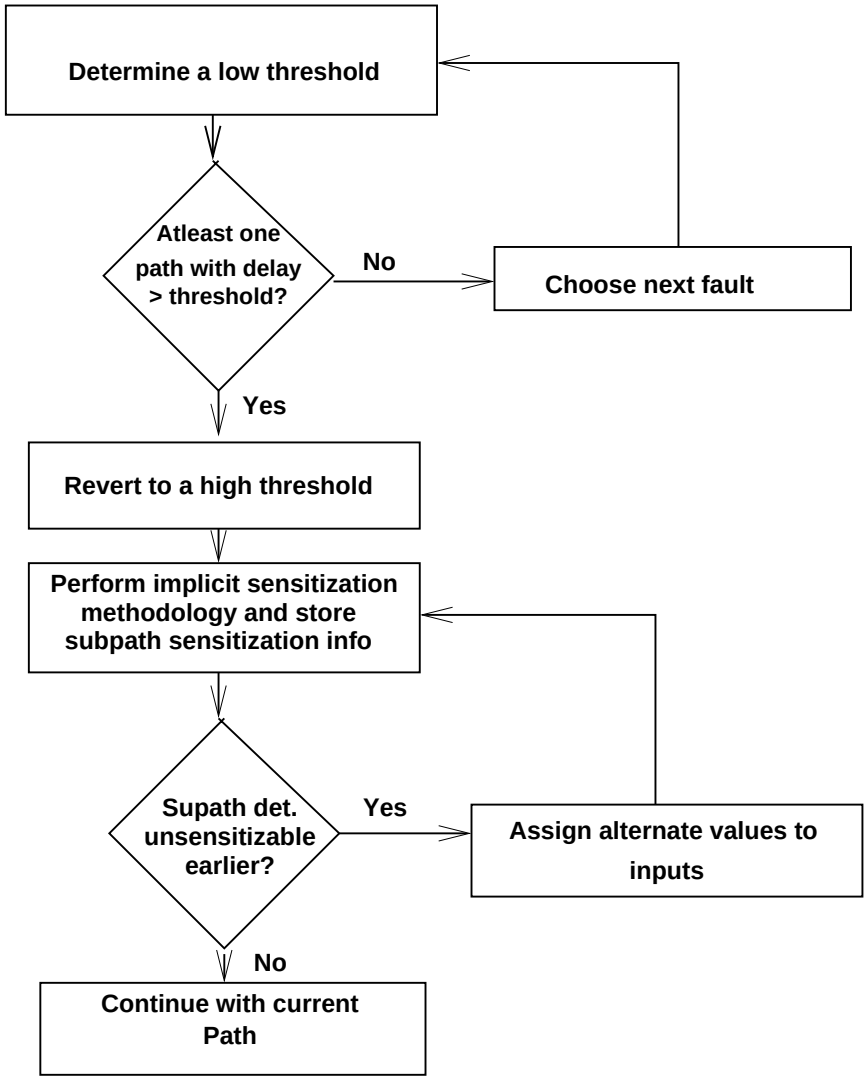


Figure 4.9: Test Generation Optimization Flow

Chapter 5

Identification of Full-chip level Critical Paths

Industry relies on static analysis for timing verification today since commercial tools only support this capability. This approach identifies the structurally longest paths in the combinational modules and estimates the maximum clock frequency from this information. It is well known that many of these structurally longest paths are false paths, and it is impossible to propagate a signal along the false paths. The issue of identifying false paths has been addressed in [51, 52, 53]. Relying on static analysis underestimates the maximum clock speed of the chip. In addition, attempts to optimize the circuit based on the structurally longest paths increases chip area and power unnecessarily. Another problem with existing static analysis based timing verification at the module level is that the true paths at this level may become false when the module is instantiated in a bigger design, since there may not be any input sequences that drive the module inputs to values that sensitize the critical path. It may be possible to simplify the delay test problem using design for

testability (DFT) techniques like scan (as long as it is possible to apply a particular sequence of vectors). However, the chip-level timing verification problem has to be addressed at the functional level, since we need to determine the operating clock frequency of the chip; this has to be done during design – it is not possible to optimize the chip after manufacture.

Timing verification is based on the path delay fault model, and many delay fault test generation techniques have been proposed for this model. One hurdle in dealing with path delay faults is the exponential number of paths that need to be considered. Various path selection techniques have been proposed for pruning the number of critical paths that need to be considered [54, 55]. These include considering all paths whose delay exceed a given threshold. Considering only sensitizable critical paths would help in further reducing the number of paths which have to be analyzed, and more pertinently, eliminating the analysis of structurally long paths that turn out to be false paths. This sensitization can be both at the module level (the assumption here is of full controllability at the inputs of the combinational block and full observability of the output) and across sequential boundaries (including signal constraints imposed by the design in the preceding logic blocks).

In [60] the authors propose to generate delay faults by adding fault injection circuitry to the circuit under test such that testing for a stuck-at fault at the output of the injected circuit sensitizes the critical path. This allows the use of ATPG tools for finding the critical path, and if a test is generated for the stuck-at fault, the critical path is sensitizable. This approach was proposed for non-scan circuits, but could also be used for circuits with scan.

Since there are potentially an exponential number of paths in a circuit, it would not be economical to first generate the longest paths and then check whether they are sensitizable. In [56], a methodology is proposed to identify the longest

sensitizable paths at the module level, without enumerating the longest paths; the tool "CRITIC" does this by generating all the sensitizable paths that exceed a given delay threshold. This methodology thus enables the determination of module level sensitizability the set of paths identified can be further refined by considering the constraints imposed by the preceding blocks.

In [48] the authors provide a methodology to obtain a tight superset of the functionally testable paths in the microprocessor. Here, they analyze the RTL description of the microprocessor and symbolically derive all possible vector pairs that can be applied on the datapath logic and in the case of the controller logic, they specify input transition constraints, legitimate input patterns to the controller and output constraints. They then propose an implicit enumeration algorithm to classify the functionally testable paths in the datapath. They implement this implicit enumeration algorithm for combinational and sequential logic blocks.

An attempt to identify the critical paths (and eliminate false paths) when the combinational module is instantiated in a bigger design was dealt with in [65]. This technique attempted to automatically determine the sensitizability of module-level critical paths at the full-chip level. The sensitizability of the critical paths identified by CRITIC was verified by adding a fault-injection block that sensitized the critical path under consideration, and checking for equivalence of the fault-free and the fault-injected circuit; equivalence of the two circuits point to the non-sensitizability of the critical path under consideration. Checking the equivalence of the sequential circuits was done using the Extracted Control Flow Machine (ECFM) [65]. Although this is an elegant approach, it is only applicable to small designs due to the complexity of the computations necessary. The approach fails for the large designs such as the ARM processor considered in this paper.

The high-complexity involved when looking at the full-chip level also pre-

cludes the use of existing sequential ATPG in order to determine the sensitizability of a given critical path from the chip boundaries. An approach to reduce the complexity seen by sequential ATPG in hierarchical designs was proposed in [49, 50]. Here, given a hierarchical circuit and a module in the design, the instantiating constraints of the module are automatically extracted, and a much simpler "transformed module", which includes only the logic relating to the embedded module, replaces the remainder of the logic for the purpose of ATPG. Results show that the transformed module with lower sequential depth and fewer logic gates, reduced the ATPG complexity drastically. The results were demonstrated for stuck-at faults and yielded spectacular results in terms of fault coverage and ATPG time at the full-chip level.

In this chapter, we apply the hierarchical methodology to chip-level timing verification, where we identify the module level critical paths that are sensitizable at the full-chip level. This enables using the significant optimizations provided by ATPG over other techniques like equivalence checking apart from the abstraction defined above. In this methodology, we identify the module level critical paths using CRITIC. For each of the critical paths identified, we add fault injection circuitry so that if a stuck-at fault at the output of the fault injection circuit is testable, the critical path is sensitizable at the full-chip level. We perform ATPG on the transformed module, thus drastically reduce the complexity seen by the ATPG tool, making it possible to apply this technique to large designs. Further, we can use commercial ATPG tools to perform the test generation.

5.1 Automatic Functional Constraint Extraction

A module in a hierarchical design is a combination of state machines and data-path elements. Many of the states of the module can become unreachable when the module is instantiated in a design. This is due to functional constraints that are defined by the instantiating environment of the module in the hierarchical design. It is possible to identify these constraints in a high-level representation of the circuit but it is not apparent in the gate level representation of the circuit and thus to the ATPG tool. Thus during state-space search in ATPG, there is a very high number of backtracks while the tool resolves these functional constraints, leading to a very high complexity.

In [49, 50] a technique to use the RTL level description of the circuit to extract the instantiating constraints of the module under test was proposed. We extract unique justification (propagation) paths for each PI (PO) of the module under consideration. We thus generate an RTL level description of the justification constraints on each of the module inputs and the propagation constraints on the outputs. We then synthesize these instantiating constraints of the module which yields logic of much lesser complexity than what was originally surrounding the module under test. This logic can be visualized as virtual logic which has been introduced into the model as a guide to the ATPG tool even though this is not present in the original design. We build a transformed module with the synthesized constraints and the module under test as shown in Figure 5.1.

In this technique, we identify a set of registers accessible through primary inputs/outputs using the processor instruction set (public or private instructions). We classify these registers as PIER registers. We extract justification and propagation paths from the module till a PIER and in case a PIER is not encountered,

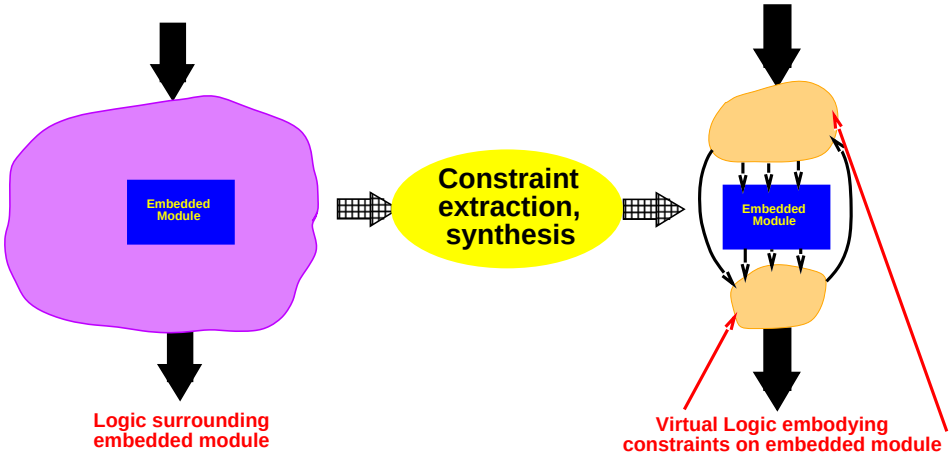


Figure 5.1: Transformed Module

till the PI/POs of the design. Thus the transformed module has PIERs and PI/POs as its boundaries. This helps in reducing the sequential depth of the transformed module which is the ATPG view of the module under test. The fact that the PIERs can be loaded and unloaded from using the instruction set enables us to translate the vectors generated by the ATPG tool to full-chip level vectors.

The PIERs are registers that can be loaded using the instruction set of the processor. These could be registers defined in the design or could be registers which were defined in processors for debug purposes. Thus each register has a sequence of instructions defined to load values into the registers. Generally, once PIERs have been identified, an assumption can be made that the PIERs are fully controllable, in the few cases that they are not, a set of constraints can be specified either a virtual constraints or as ATPG constraints.

The ATPG complexity, which is a function of the sequential depth and the number of gates, is reduced drastically due to the use of the transformed module.

Modifications were performed on this extraction methodology to make it suitable for speedpath sensitization. This is required because there is a requirement for exploring all possibilities in the surrounding logic to try and sensitize the path under consideration. Thus instead of trying to identify just one justification and propagation path from the module of interest, the methodology extracts all possible justification and propagation paths surrounding the module. In the original methodology, the pruning of some of the justification/propagation path restricts the set of vectors possible at the input of the module; this might contribute in classifying some of the paths as unsensitizable while they might be sensitizable which constitute false negatives. This alteration helps us eliminate the possibility of false negatives.

This is required because of the large set of constraints that need to be satisfied to determine path sensitizability.

A major factor contributing to the complexity reduction in the functional constraint extraction framework is still maintained because we still have PIERs that are identified and loaded. This minimizes the sequential depth thus helping maintain a low complexity for the ATPG tool.

5.2 Application to finding chip-level critical paths

We first identify critical paths at the module level which are sensitizable from the module inputs. These **sensitizable critical paths** are identified by CRITIC [56]. CRITIC determines all the sensitizable critical paths at the module level exceeding the provided threshold, and generates input vector sequences for each critical path. These vectors can be applied at the module boundaries.

Once the module level vectors for the critical path excitation are obtained,

along with the information as to whether the critical path is excited on a rising edge or a falling edge, we add a fault injection logic block to the circuit such that finding a test for a stuck-at fault at the output of the logic block determines the sensitizability of the critical path at full-chip level. For example, Figure 5.2 shows a combinational block, where the nets forming the critical path are highlighted. The module level input vector (generated by CRITIC) that sensitizes the critical path is also shown in the circuit. The logic block that was added by us is shown in the dotted box.

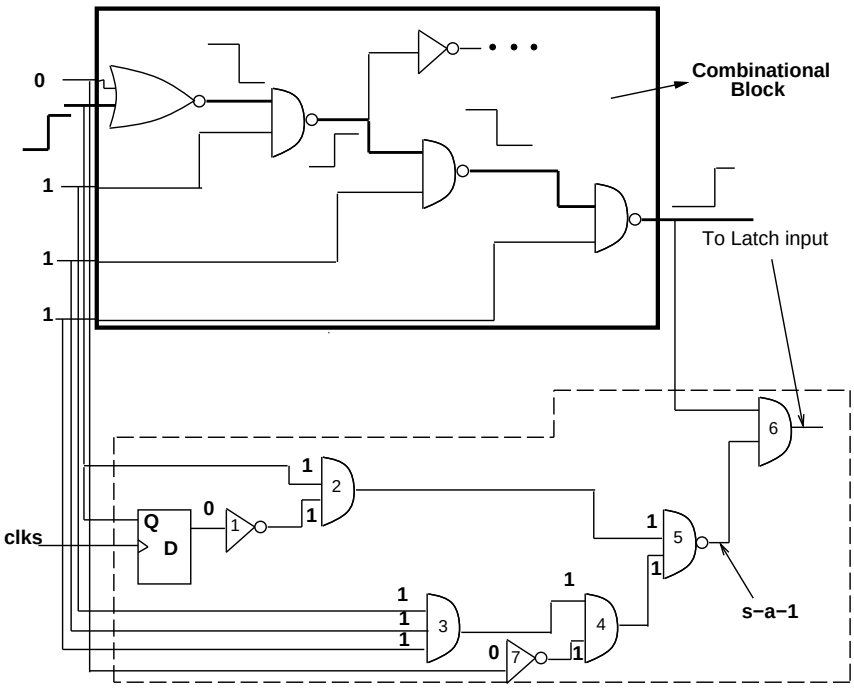


Figure 5.2: Circuit with Fault Injection Logic Block

We use the vectors at the module inputs which are generated by CRITIC, the transition at the input of the critical path, and the inversion parity of the critical path to construct the logic block. The output drives the output latch of the combinational

block such that the vectors generated by the commercial test generator make the faulty value at the output latch visible at the primary output or at the PIER registers.

In the example, latch in the fault injection logic block ensures that a transition is initiated at the input of the critical path. This is done by imposing a zero/one for a rising/falling transition respectively (this affects the fault because it passes through a latch which affects the fault site in the second cycle) and requiring its inversion in the second cycle. The gate 6 which follows the line where the stuck-at fault is placed feeds the latch that is at the output of the critical path, and enables the propagation of the fault effects to the primary output of the transformed module. The rest of the circuit (gates 1 through 5) establishes the input vector generated by CRITIC on the module inputs which the ATPG tool justifies to the primary input of the transformed module. We generate the sequential test patterns using the transformed module to reduce the ATPG complexity. Thus for each critical path, we generate a vector pair that sensitizes the critical path and initiates a transition at the input of the critical path and which can be applied at the PIs of the transformed module. The vector also serves to propagate the faulty value that is obtained at the output of the critical path.

The overall methodology is outlined in the Figure 5.3. Though the timing verification tool CRITIC is used here for the identification of the critical paths, the techniques is modular so as to allow the use of any available timing verification technique. The sensitization criterion and the delay model of the timing verification tool determines the number of critical paths with the lowest being the exact number of paths sensitizable at the combinational block level. As the sensitization criterion used becomes looser, the number of critical paths increasing thus increasing the evaluation time of the methodology defined here.

5.3 Results

We applied the technique to hierarchical designs which were of reasonable complexity. The designs that we considered are three processors, Viper, DLX, and ARM. Viper is a simple processor, DLX is a RISC processor with a 5-stage pipeline, and ARM is a model of the commercial ARM-2 processor. We used the ALU of all the three benchmarks for our experiments. The specifications of the various designs are shown in the Table 5.1 below.

Table 5.1: Details of Target Modules

Processor	Combinational Gates	Sequential Elements	Primary Inputs	Primary Outputs
Viper	5863	438	34	53
Viper ALU	1784	0	72	34
DLX	15177	1610	69	100
DLX ALU	2965	0	70	34
ARM	16029	1270	63	67
ARM ALU	3850	0	77	36

We first generated the input files for CRITIC of the Viper, DLX and ARM ALUs using a unit delay model; this is adequate to demonstrate this technique because we aim for a reduction in the number of paths seen. In order to obtain the module level critical paths, we ran CRITIC for various threshold values for the path delay. Here the delay was calculated in terms of unit delay model where the delay of the gate was 1. This was adequate because the goal is to estimate the reduction in the number of paths. Table 5.2 gives the values of the threshold values, the number of critical paths obtained and the time taken to identify these module level critical paths. Here, SLPD refers to the structurally longest path delay, CPD threshold refers to the threshold provided to CRITIC. For example, in the ARM ALU,

the structurally longest path had 137 units of delay, there were 8 sensitizable paths longer than 130 units and 125 sensitizable paths greater than 115 units of delay. The time taken is on a 300MHz Pentium II system with 64 MB RAM.

Table 5.2: Critical Paths

Module	SLPD	CPD Threshold	#CP	Time (secs)
Viper ALU	145	130	4	332.75
		115	15	638.90
DLX ALU	161	130	18	395.26
		125	35	443.05
ARM ALU	137	130	8	399.46
		125	26	558.16
		115	125	549.05

We then extracted the transformed module for the ALUs, this enabled us to reduce the logic complexity that we present to the ATPG tool. Table 5.3 shows the reduction obtained as a result of extracting the transformed module compared to the number of gates in the original design. We built the fault injection block as described in Section 5.2 and added it to the circuit.

Table 5.3: Reduction in Complexity

Module Name	# Original Gates	# Gates in Transf. Module	Reduction %
Viper ALU	3920	2087	47.8
DLX ALU	11881	4637	71
ARM ALU	16734	8143	52

We first used a commercial ATPG tool on the full-chip design with the fault injection logic block. The results are shown in Table 5.4, which gives the number of

critical paths that are sensitizable at full-chip level when using ATPG on the original design. Owing to the complexity of the designs, the commercial tool aborted on every case, giving no indication whether the module-level critical paths are sensitizable at the full-chip level. The experiments were run on an UltraSparc II 200MHz system with 1GB RAM.

In contrast, Table 5.5 gives the number of paths that are identified to be sensitizable at the full-chip level when we used the transformed module for ATPG with the same commercial ATPG tool. Column 3 is the number of critical paths at module level that have been identified by CRITIC and Column 4 gives the number of paths that are sensitizable at the full-chip level. The use of the transformed module reduces the ATPG time considerably, and there were no aborted ATPG runs.

Table 5.4: Sensitizable Critical Paths at Full-Chip Level

Module	CPD Threshold	Number of Critical Paths	Number of Sensitizable Paths	Number Aborted	Time (secs)
Viper ALU	130	4	0	4	208.9
	115	15	0	15	932.7
DLX ALU	130	18	0	15	5811.2
	125	35	0	30	2113.6
ARM ALU	130	8	0	8	934
	125	26	0	26	8486.8
	115	125	0	125	40655.8

Table 5.6 shows how the number of sensitizable critical paths for our benchmarks decreases as we move from the module level to full-chip level. The column SLP, refers to the Structurally Longest Paths that are identified when using current commercial static analysis tools. As can be seen, the reduction in the number of sensitizable paths is drastic. For example, for the ARM processor, the designers

Table 5.5: Sensitizable Critical Paths at Full-Chip Level Using Transformed Module

Module	CPD Threshold	Number of Critical Paths	Number of Sensitizable Paths	Number Aborted	Time (secs)
Viper ALU	130	4	3	0	117.0
	115	15	7	0	428.9
DLX ALU	130	18	0	0	2115.7
	125	35	1	0	5269.3
ARM ALU	130	8	0	0	933.6
	125	26	0	0	3060.1
	115	125	3	0	21924.3

only need to really worry about optimizing three long paths in the ALU. Using existing commercial tools, they would have to deal with optimizing over a thousand paths, and much of the work would be for naught, since most of the paths would never be exercised functionally at the chip level.

Table 5.6: A Comparison

Module	Delay Threshold	Number of SLPs	Number of True Paths	
			Module level	Full Chip level
Viper ALU	115	160	15	7
DLX ALU	125	404	35	1
ARM ALU	115	1143	125	3

The use of ATPG along with the constraint extraction technique helps in limiting the complexity increase as the circuit complexity increases. Using formal techniques [65] results in significant memory complexity; the ATPG techniques which are structural effectively overcome these drawbacks. Further, the current generation ATPG techniques provide a significantly mature capability when compared to the

satisfiability based techniques. The methodology proposed is modular enough to allow the use of commercial tools for module level timing verification and ATPG; and the methodology can thus be easily integrated into existing methodologies.

5.4 Conclusion

We have proposed a powerful technique for full-chip timing verification. The results of applying the technique on processor-level benchmarks show a steep reduction in the number of critical paths that need to be considered by the designer.

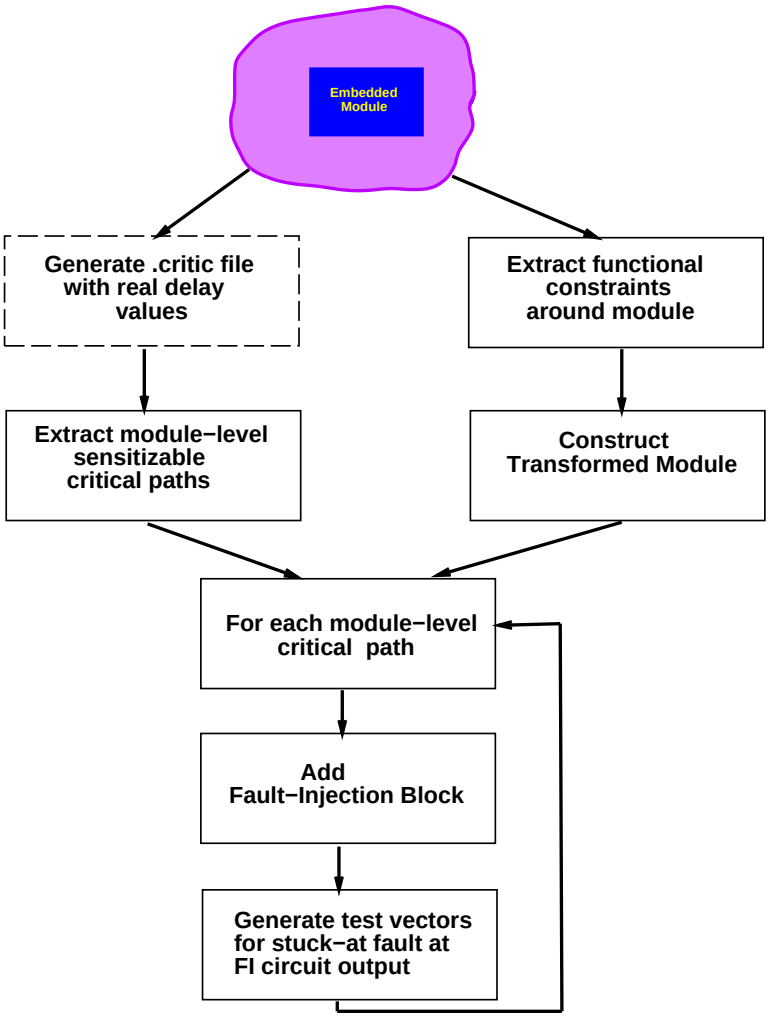


Figure 5.3: Timing Verification Flow

Chapter 6

Multicycle Sensitization of Critical Paths

In this chapter, we study the increase in test quality that could be achieved when the sequential sensitizability of the paths identified for test are considered. This quality of the test set is evaluated from the potential of defects (that are detected) to cause circuits to fail because of delay defects. This further enables an increase in the yield achievable and the speed (via speed binning) that the design can be clocked. We also show the effect of considering sequential sensitizability on the paths that are considered for delay test under a given fault model.

Any delay test technique generates tests on a combinational block in the design. These strategies assume full controllability at the input latch boundary and observability at the output latch boundary. This leads to overtesting where paths which are functionally unsensitizable are tested for delay defects which in turn leads to a decrease in the yield of the manufacturing process. Considering the functional constraints imposed by the preceding logic blocks in the design on the target block

helps in alleviating some of the yield losses.

The technique that is outlined in this chapter enhances the quality of test vectors generated; this is demonstrated using the improvement in yield that is achieved; the yield is calculated in terms of the reduction of the delay of the paths that are targeted. The technique used to achieve this is similar to the one outlined in Chapter 5. This yield value is achieved without affecting the DPM goals of the process. This contributes to the manufacturing flow as compared to the previous technique which contributed to the timing verification step in the design flow.

6.1 Yield and Test quality

The yield of a manufacturing process is calculated as the probability of a product having no faults. The fault is defined to be a manufacturing defect that manifests itself in the circuit, thus affecting the operation of the circuit . Estimates of yield of a manufacturing process, incorporate the parameter – Probability of Failure (POF) – to account for this [38]. The POF corresponds to the fraction of the manufacturing defects that manifest themselves as faults (thus affecting the circuit behavior).

When the yields of manufacturing processes are calculated the goal is to calculate the effects of localized defects; this is because of the very low probability of faults affecting the chip globally. The POF thus depends on type of defect, the size of the defect and the circuit. The yield calculation employs a probability spatial distribution to estimate the distribution of the defects in the circuit. The yield of the process is thus calculated as the probability that the number of defects in a given circuit is 0 (under the assumption that there is no fault tolerance incorporated into the circuit of interest).

For modeling the yield of the manufacturing process, a statistical distribu-

tions is assumed for the defects over the chip area. The most common of such models is the Poisson model. The simplicity of this model make it very suitable for our purposes and this model has been shown to be a good fit with the empirical data found.

The parameters that are used in modeling the yield are the average number of faults in the chip (λ); n denotes the number of statistically independent areas the chip is divided into. Thus the number of faults occurring in each of the sections is given by (λ/n). Thus the probability of the chip having k faults under this independent section assumption is given by

$$p(x = k) = \left(\begin{array}{c} n \\ k \end{array} \right) ((\lambda)/N)^k (1 - \lambda/N)^{n-k} \tag{6.1}$$

When $n \to \infty$ this results in a Poisson distribution

$$P(x = k) = \frac{e^{-\lambda} \lambda^k}{k!} \tag{6.2}$$

This assumption of statistically independent areas resulted in very low predicted yields. This was because in actual circuits, faults tend to be clustered in portions of the chip rather than being distributed evenly as the independent section assumption implies.

To account for these clustering effects, rather than a pure Poisson distribution, a compound Poisson distribution is modeled. Thus λ is modeled as a random variable rather than a constant as mentioned earlier. This λ is the expected value of a random variable L with values l and a density function $f_L(l)$ where $f_L(l)dl$ determines the probability that the chip average probability lies between l and $l + dl$. Thus the probability of k defects in a chip can be given by

$$P(x = k) = \int_0^\infty \frac{e^{-l} l^k}{k!} f_L(l) dl \tag{6.3}$$

The function $f_L(l)$ is known as the mixing function. Using the gamma function with two parameters λ and α. The parameter λ is the average number of faults per chip and the parameter α is the amount of fault clustering in the chip; this typically ranges from 0.5 to 5. Thus $f_L(l)$ is given by

$$f_L(l) = \frac{\alpha^\alpha}{\lambda^\alpha \tau(\alpha)} l^{\alpha-1} e^{-\frac{\alpha}{\lambda} l} \tag{6.4}$$

Substituting Equation 6.2 in Equation 6.4 results in the Equation:

$$Prob(x = k) = \frac{\tau(\alpha + k)}{k!\tau(\alpha)} \frac{\lambda/\alpha^k}{(1 + \lambda/\alpha)^{\alpha+k}} \tag{6.5}$$

Thus, the yield of the chip can be given by

$$Y_{chip} = Prob(x = 0) = (1 + \lambda/\alpha)^{-\alpha} \tag{6.6}$$

6.2 Path Sensitizability

The set of paths in a circuit when the circuit is simply considered as a directed graph, is exponential on the number of gates in the circuit. But when the functionality of the various gates in the circuit is considered along with their respective propagation delays, many of the paths become false. Research on the development of automated methodologies to identify the true paths in a given combinational circuit has been ongoing in the timing verification field.

Further, as mentioned earlier, combinational blocks in the complex sequential circuit are embedded within logic which imposes certain instantiating constraints on the combinational block. The example in Figure 6.1 shows a path (shown in bold) rendered false when sensitizability is considered over multiple cycles. Thus the number of combinational paths are further reduced. This reduction from the

structurally longest paths to the combinational block level sensitizable critical paths and then to the paths that are functionally valid is high [57].

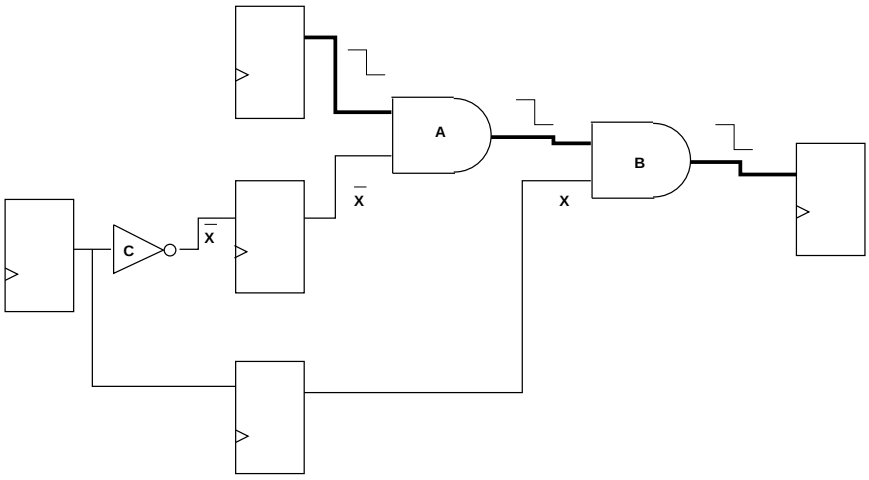


Figure 6.1: Multi-cycle Unsensitizability

6.3 Speed Binning and the Benefits of Multi-cycle Sensitization

Speed Binning is increasingly pursued in current generation design houses to identify the maximum possible frequency in which the given chip can operate. This is done by using critical paths in the design and using vectors to test them at-speed at various frequencies to determine the highest possible frequency that the chip can operate. Here the critical paths are identified at the combinational block level and the tests are generated correspondingly.

This has become increasingly important as the thrust on frequency became

paramount. The large number of paths that were marginal dramatically increased the probability that a chip would be slotted in a lower frequency bin, thus reducing it's value. The ability to prune the number of paths that are targeted here would dramatically increase the probability that a given chip can be shipped in the highest frequency bracket available, while still maintaining the fact that the all the functionally relevant paths have been tested.

As mentioned in Section 6.2 many of the critical paths at the combinational block level may not be true when they are considered over one or multiple latch boundaries. This causes many of the possible critical paths that were considered to be inconsequential, thus helping in the classification of the design in a higher frequency category compared to when the path sensitizability was not considered.

The ability to identify the paths that are false over multiple cycles enables us to accommodate the presence of delay defects on certain paths without in anyway compromising the timing correctness of the circuit under consideration. This enables a dramatic improvement in the yield of the chip and, as a result of the reduction in the delays of paths that are targeted, an increased number of chips slotted at a higher frequency during speed binning.

This methodology enables us to get a yield number that is closer to the ideal number – which can be achieved using a completely functional test set which could be applied at the primary inputs or as a sequence of instructions.

Further, multi-cycle sensitization allows the functional justification based test application of all the test vectors that were generated using this methodology, thus enabling the use of standard scan designs and reducing the area overhead due to DFT.

6.3.1 Determination of Sensitizability of critical paths

Once the paths to be targeted by the test vectors have been identified, they are evaluated for sensitizability over multiple combinational blocks preceding them. This is done by adding a fault injection block to the circuit under test that drives the required transition through the targeted path. Thus testing for a stuck-at fault generates vectors that drive the required transition through the path. An untestable stuck at fault indicates that the path is sequentially false. This technique is similar to the one outlined in [60].

Using the fault injection block on one path is facilitated by the fact that one distinct path is targeted by the test generated by the tool outlined in Chapter 3. This is done by specifying only the latch corresponding to the output of the critical path while maintaining the other paths to X's. This ensures that the test causes the failure of only the path that is primarily targeted. This disables the case shown in Figure 6.2, where the path B-F which is not targeted fails due to all the outputs being specified; this helps in making an accurate determination of the failure mode and optimize the yield achieved in the process.

6.3.2 Results

In order to demonstrate the possible increase in test quality that is possible using sequential path sensitization we have run the experiments on multiple ISCAS benchmarks. These were synthesized into primitive gates using a 0.25 micron CMOS library. The designs were further optimized for timing to increase the number of paths close to the clock timing as is the case in any commercial design.

The circuits were then broken into their various component combinational blocks and a commercial timing analysis tool was used to identify the sensitizable

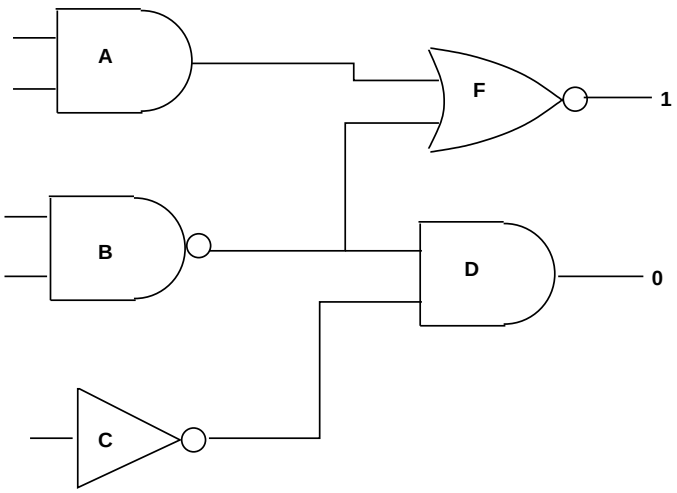


Figure 6.2: Spurious Detection

(at the combinational block level) critical paths. Using the commercial static timing analysis tool, we were able to get a handle on optimized delay modeling for identification of the critical paths.

The path sensitization technique that was outlined in the Section 6.2 was then applied in order to justify the given set of paths over combinational blocks that precede the block under consideration. The Table 6.1 gives the upper limit on the number of paths that were identified to be sensitizable. The Column 2 gives the threshold chosen for the identification of the critical paths. Column 3 gives the number of critical paths that are sensitizable at the module level. Columns (4,5), (6,7), (8,9) give the number of sensitizable paths and the number of paths for which the path sensitization aborted over 1, 2, 3 preceding combinational blocks, respectively.

As can be seen in the table, there is a good reduction in the number of paths

that are targeted. Therefore we see a much higher probability of the design being slotted to a higher frequency. The Columns 5, 7, 9 give the number of paths for which the ATPG tool aborted; this is because of the inability of the ATPG tool to handle the logic seen. In the case of aborts, because there was not clear indication that the paths were unsensitizable, they should be assumed to be sensitizable. The effect is not present when sensitizability is evaluated over one level, but becomes prominent over multiple levels. Further, accounting for this, the incremental benefits as we go across multiple levels keeps reducing as the number of levels increases.

Table 6.1: Reduction in Number of Critical Paths

Module	Thres.	Number of Sensitizable Paths						
	(nsec)	Module	1 Level	Aborts	2 Levels	Aborts	3 Levels	Aborts
s641	4	118	49	0	-	-	-	-
s713	4	93	20	0	-	-	-	-
s1238	6	104	92	0	-	-	-	-
s9234	5	2718	281	0	263	4	260	-
s15850	8.7	1290	648	0	354	224	137	173
s38417	3	414	52	0	50	0	50	0

6.4 Improvement in Test Quality with a Fault Model

In Chapter 3, a test generation methodology was demonstrated on designs of moderate complexity and a high degree of coverage was obtained on designs composed of primitive gates. The presence of scan was assumed for the application of all the generated test patterns for the results shown in Table 3.1. Of all the test application techniques, enhanced-scan is the most suitable for this scheme.

6.4.1 Test Vector Enrichment using Sequential Sensitizability

When a path that helps target a net in the combinational block for a given transition is found to be sequentially unsensitizable, a shorter path through the net is identified and its sensitizability determined. The identification of shorter paths through the given net is done by marking the fanout points when identifying the original path. When the sink delay of the alternate fanout nodes exceeds the chosen threshold, weights are added to the alternate fanout nodes to force the path tracing through these alternate nodes. This is performed recursively for all the fanouts in all the paths successfully identified.

6.4.2 Results

The ATPG engine for the Unified Delay fault model was derived from CRITIC, a timing verification engine proposed in [56]. Actual timing values for the various gates in the circuits used were derived from process libraries. This was done by using the input file in the Standard Delay Format (SDF format) and deriving the rise and fall delays and adding them as the delays of buffers at the inputs of the gates under consideration.

For each of the paths that were identified for testing by the fault model proposed in Chapter 3, a fault injection circuit was added which forced the propagation of a transition through the path for test generation to be successful. The design was broken down into multiple combinational blocks divided by latch boundaries. The delay test vectors (for the proposed fault model) were generated and the related paths were identified for each combinational block and were successively justified over the preceding combinational blocks. Table 6.2 gives the number of paths from the initial set identified that were rendered false. Here the set of paths identified

Table 6.2: Initial Reduction in the Number of Paths

Circuit	Number of Paths Identified	Number of Sensitizable Paths	
		Block 1	Block 2
s526	452	62	-
s641	733	191	-
s1238	1163	1043	-
s5378	827	514	500
s15850	1356	401	363
s38417	1283	286	243

corresponds to a sample set of 300 nets in one combinational block in each of the designs. These paths adequately serve our purpose of demonstrating the reductions in delay.

As can be seen, there is a reduction in the number of paths that can actually be exercised by the functional patterns. This is also shown in the graph shown in Figure 6.3.

Once the false paths among those identified by the test generation tool are determined, paths shorter than the ones identified need to be used for further test generation and the sensitizability identification. This causes a reduction in the length of the paths that are targeted for test generation.

We used the 0.25 micron CMOS library to determine the delays of the longest paths in the combinational blocks under consideration. Table 6.3 shows the number of nets for which there was a reduction in the delay of the longest paths passing through them. This implies that paths that are longer than the path that is actually sensitizable need not be targeted for delay faults and allows us to target the realistic delay defects (i.e. the defects on the paths that are actually sensitized). During the identification phase of the critical paths, when there are multiple long

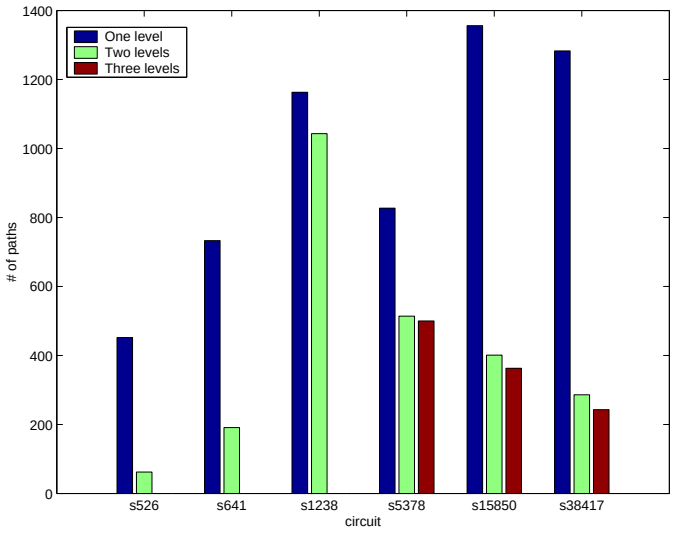


Figure 6.3: Reduction in the # of Critical Paths

paths through a net, the delay of the critical paths identified were limited to 70% of the longest sensitizable paths because this was just a demonstration of the possible improvements in the frequencies that could be achieved.

Figure 6.4 shows the change in the delays of the paths used for the delay test when sensitizability over one combinational block was considered for s526 and s641. The charts show the change in the delay distributions over various percentages of the delay of the maximum delay path in the combinational block under consideration. The paths considered were for nets which had a reduction in the longest sensitizable path through them when sequential sensitizability was also considered. As can be seen, there is a drastic reduction in the number of paths that have a delay greater than 80% of the delay of the longest path in the combinational block.

The graph shown in Figure 6.5 shows the yield as a function of the number of faults per $mm^2$of the chip. The yield goes down as the number of faults

Table 6.3: Reduction in the Path Delay

Circuit	Number of Paths Identified by ATPG	# faults with reduction in delays targeted	
		1 Level	2 Levels
s526	452	75	-
s641	733	24	-
s1238	1163	51	-
s5378	827	63	63
s15850	1356	955	993
s38417	1283	77	112

Table 6.4: Yield Improvement

Circuit	λ Red %	Yield Imp %
s526	2.4	0.2
s641	0.45	0.2
s1238	0.60	0.2
s5378	1.1	0.2
s15850	10.5	0.7
s38417	0.9	0.2

increases. Here a fault is defined as a defect that manifests itself in the circuit. In the results above, we have identified functionally false manifestations of defects. Thus, considering the results shown in Table 6.3, conservatively assuming an average reduction in delay of 15%, Table 6.4 gives the improvement in yield. The value λ is the average number of faults per chip. The reduction in λ is calculated as the fraction of paths with reduction in delay factored by 0.15 which is the average percentage reduction in the critical paths.

The yield values shown in the Table 6.4 is a conservative estimate of the

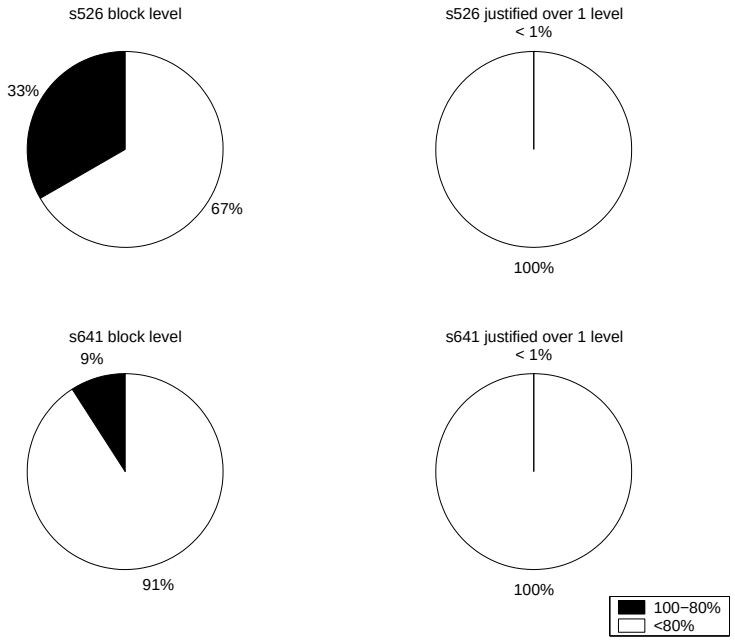


Figure 6.4: Reduction in the Delay

improvement in yield that can be achieved. We expect the average percentage reduction in the delay of critical paths to be much higher and thus contribute to a greater reduction in λ. Further the longest critical paths are most likely to cause failures in the circuit, thus eliminating the longest paths would in reality cause a much larger reduction in λ (because as compared to the stuck-at fault model, the probability of failure increases dramatically as the slack on a path being tested increases).The evaluation of sensitizability over multiple such levels would also result in much higher savings in yield.

The methodology outlined above, thus gives a structured technique to prune the number of faults targeted. Though some of the previously proposed techniques [60, 48] aim to identify the functionally sensitizable paths, the lack of an

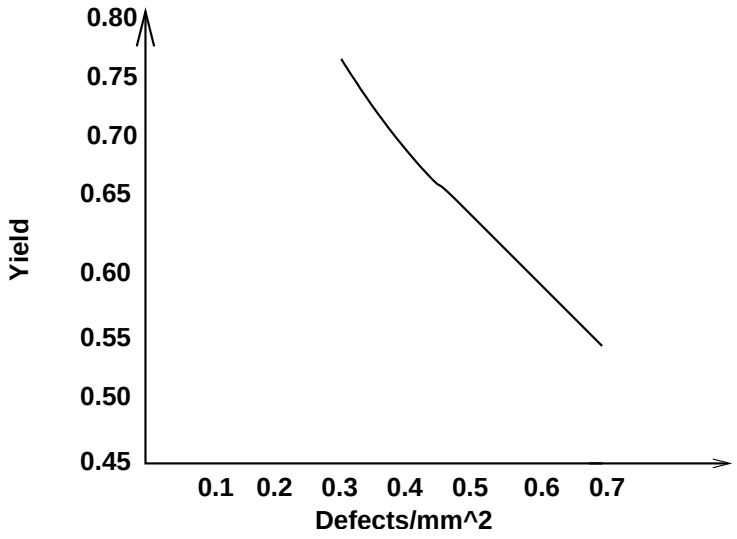


Figure 6.5: Yield as a Function of # of Defects per Unit Area

incremental methodology imposes considerable complexity on the constraint solving tools.

6.5 Conclusion

The advantages of evaluating the multicycle sensitizability in delay test generation were outlined. A technique to evaluate the sensitizability of the paths was also presented. This technique enables the application of the test using a standard scan design as compared to the rather complex scan strategies traditionally required for delay test application. The reduction in the number of paths and the resultant improvement in yield obtained was also presented.

Chapter 7

Conclusions and Future Work

The work outlined in this dissertation enabled providing a better coverage of the realistic defects that are not targeted by the traditional fault models used in manufacturing test generation. A directed methodology is more suitable than using an alternate less directed one because of the requirement to detect the smallest possible defects. The unified fault model provided exactly this in the delay test space. The fact that most of the realistic defects manifest themselves as delay defects lends more credence to this fault model.

The test generation methodology proposed helps overcome the complexity issues that arise with the prevalent delay test generation tool. This, along with the optimizations proposed help in the feasibility of using the unified fault model for better defect detection. As the methodology proposed is primarily scan-based, the issue of over-testing and the resultant yield loss becomes more relevant; this requirement along with the requirement for an enabler for delay test application using a standard scan design yielded the methodology for multi-cycle sensitizability evaluation. The calculations for yield showed that the benefits of such a methodology were tangible.

In the realm of timing verification, the evaluation of full-chip sensitizability of critical paths in the design, proved to be a key enabler for a more accurate critical path identification, thus allowing for more aggressive delay targets. The primary outcome of the methodology was enabling the use of ATPG for timing verification with many of the complexity traps mitigated.

7.1 Future Work

7.1.1 Accurate Delay Estimation

The delay model assumed in the test generation methodology assumed a constant rise and fall delay associated with each gate in the design. Though the constant delay was derived from the "Standard Delay Format" of the circuit and was the most accurate values under the constant delay model assumption, the fact that the value of the delay through the gate is not a constant for various input stimuli and the activity in neighbouring nodes points to a requirement for a better delay model.

The Table 7.1 gives the variations in the delay of the circuits as a function of the transitions or the lack of it at the inputs of the gate. The table shows the values for a NOR gate [35]. As can be seen, the variation delay from the standard delay is considerable depending on the transitions that are seen at the input of the gate. Thus, even assuming a nominal constant delay value is not accurate for all the values that can be seen at the inputs of the gate. In the case of transitions happening on both the inputs of the gate under consideration the relative delay between the transitions arriving at the gate also result in varying delays through the gate. As shown in [35], the maximum delay through a gate is seen when the transitions are closely spaced in the case of controlling to non-controlling transitions at the inputs

Table 7.1: Input Dependance of Gate Delay

Input A	Input B	Output	Delay Increase (%)
↑	↑	↓	0
0	↑	↑	+97.4
↑	0	↑	+119.2

of the gate under consideration

Further, the delay of gates on a critical paths considered are also dependent on the value on the side input of a fanout gate. This delay also depends on the input of the gate that the fanout is connected to (i.e. whether the gate input is close to V_{SS} or V_{DD}).

All the above mentioned conditions call for a more dynamic delay estimation methodology where the delay in a gate and thus the delay of the critical path seen is dependent on the input transitions, delay between transitions and the neighbouring conditions. This is important to help determine the critical paths more accurately and determine critical paths closer to the ones seen in silicon.

7.1.2 Test Generation for Crosstalk Defects

In Chapter 2 we gave an outline of the set of realistic defects that are becoming prominent in nowadays. One of the most prominent of these defects which is not covered in this thesis is the crosstalk defects. These defects are becoming more common with the decreasing feature size and closer spacing between the wires that cause a capacitive coupling between wires. Another cause is the high aspect ratios of adjacent wires that expose a greater surface area to capacitive coupling. This

effect is most prominent in the case of wires being adjacent over long distances which is the case in buses.

The two nets that are cross coupled are called aggressor and the victim. A transition on the aggressor has an effect on the victim as shown in the Figure 7.1. Figure 7.1 shows the crosstalk glitch, which is called when there is a transition on the aggressor while a constant value is maintained on the victim. Figure 7.2 shows a crosstalk delay where there are transitions in opposite directions.

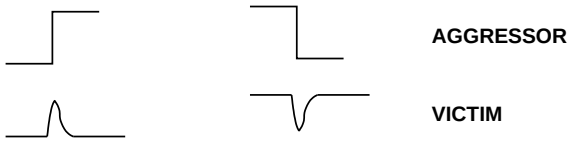


Figure 7.1: Crosstalk Glitch

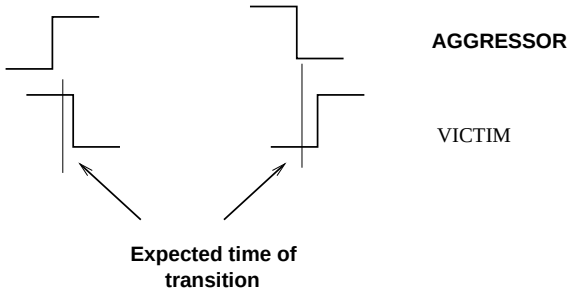


Figure 7.2: Crosstalk Delay

Methodologies have been proposed to detect crosstalk glitches [76, 75], as transition is required only through the aggressor while a constant value is required at the victim. This requires a test generation methodology similar to delay test generation conservative propagation to propagate the glitches to an observable point.

A fault model was proposed [74] to detect crosstalk delay in the case of crosstalk in long lines such as buses; this was called the maximal aggressor fault model. The goal was to drive transitions on multiple lines in the vicinity of the victim to increase the delay that would be caused by the crosstalk. BIST techniques to bias test vectors to maximize the detection of crosstalk defects under the maximal aggressor model have also been outlined.

The most complex case is the case of crosstalk delay when the above condition of the aggressor and the victim being long lines is not satisfied. This requires an accurate modeling of the delay of gates using the library of techniques mentioned above; this would enable lining up the transitions close together thus creating a crosstalk delay with maximum probability, which in the worst case would yield detection.

7.1.3 Improvement in Test Application Techniques

The traditional test application techniques for delay test vectors which include enhanced-scan and standard scan with skewed load are either too complex or are not very effective in yielding the required coverage. The broadside scan based technique would lead us a good deal towards a more efficient test application. But more efficient test application techniques are required to further ease the test application complexity.

The limitations in the frequencies of the testers are limiting the accuracy of the testers to accurately measure the failing frequency of the path in a given chip. This necessitated a functional delay test application strategy and an associated test generation strategy.

7.1.4 Modeling Process Parameter Variations

As was outlined earlier, because of the increasing use of technologies that are not mature in manufacture, the process parameter variations are becoming increasingly common and a greater source of defects. The significance of the delay tests in the current manufacturing framework significantly increases the role that these variations play on the testing of microprocessors, especially in dealing with the identification of critical paths.

The preceding delay test methodology assumed a uniform variation in the process parameters over combinational portions of the chip. But as technology progresses and the process dimensions become smaller, the assumption might no longer hold and thus the critical paths that are identified might no longer be valid and as a result, the level of confidence on the delay tests generated goes down.

In order to improve the level of confidence on the delay test vectors that are generated, a statistical analysis can be performed and a delay measure can be assigned at a location based on the probability of process parameter variation at the location currently under consideration. Techniques like the Principal Component Analysis [80], where a set of independent principal components are identified and a set of parameters that correlate to the process variables are classified as macro components; these are fitted using higher order polynomial representation built with techniques such as Response Surface Modeling(RSM) [81]. Thus given a value for a macro parameter (μ) all other macro parameter values can fluctuate in a range conditioned by μ. The electrical properties such as delay are functions of these macro parameters. This can be used in conjunction with statistical estimate of the ”actual” coverage of the delay test vectors and the test set can be tuned accordingly.

Bibliography

[1] Semiconductor Industry Association, “International technology roadmap”, 2002.

[2] L. W. Nagel, “SPICE32 : A computer program to simulate semiconductor circuits”, ERL Memorandum # ERL-M520, University of California, Berkeley 1975.

[3] D. Williams and A. P. Ambler, “System manufacturing test cost model”, *Proceeding of IEEE International Test Conference*, 2002, pp. 482–490.

[4] K. M. Butler and M. R. Mercer, “Quantifying Non-target defect detection by target fault test sets”, *Proceedings European Test Conference*, 1991, pp. 91–100.

[5] L. C. Wang, M. R. Mercer, and T. W. Williams, “On the decline of testing efficiency as fault coverage approaches 100%”, *Proceedings VLSI Test Symposium*, 1995, pp. 74–83.

[6] P. C. McGeer and R. K. Brayton, “Integrating functional and temporal domains in logic design”, *Kluwer Academic Publishers*, 1991.

[7] S. Devadas, K. Keutzer, S. Malik and A. Wang, “Certified timing verification

and the transition delay of a logic circuit", *Proceedings of IEEE/ACM DAC*, 1992, pp. 549–555.

[8] K-T. Cheng, S. Devadas and K. Keutzer, "A partial enhanced-scan approach to robust delay fault test generation for sequential circuits", *Proceedings IEEE International Test Conference*, 1991, pp. 403–410.

[9] L. C. Wang, M. R. Mercer and T. W. Williams, "Using target faults to detect non-target defects", *Proceedings IEEE International Test Conference*, 1996, pp. 629–638.

[10] S. C. Ma, P. Franco and E. J. McCluskey, "An experimental chip to evaluate test techniques experiment results", *Proceedings of IEEE International Test Conference*, 1995, pp. 663–672.

[11] S. M. Reddy, I. Pomeranz and S. Kajihara, "On the effects of test compaction on defect coverage", *Proceedings of IEEE VLSI Test Symposium*, 1996, pp. 430–435.

[12] J. T. Chang, C. W. Tseng, C. M. J. Li, M. Putnell and E. J. McCluskey, "Analysis of pattern dependent and timing-dependent failures in an experimental test chip", *Proceedings of IEEE International Test Conference*, 1998, pp. 184-193.

[13] I. Pomeranz, S. M. Reddy, "On n-detection test sets and variable n-detection test sets for transition faults", *IEEE Transactions on Computer-Aided Design of Intergated Circuits and Systems*, 2000, pp. 372–384.

[14] L. C. Wang, R. M. Mercer, T. W. Williams, "Reducing defective part level via unbiased test set", *Proceedings European Design and Test Conference*, 1995.

[15] R. C. Aitken, “Nanometer Technology Effects on Fault Models for IC Testing”, *IEEE Transactions on Computer*, 1999, pp. 46–51.

[16] K. T. Cheng and A. Krstic, “Current Directions in Automatic Test-Pattern Generation”, *IEEE Transactions on Computers*, 1999, pp. 58–64.

[17] A. Krstic, J. J. Liou, K. T. Cheng and L. C. Wang, “On structural vs. functional testing for delay faults”, *Proceedings of International symposium on Quality Electronic Design*, 2003.

[18] W. Needham, C. Prunty and E. H. Yeoh, “High Volume Test Escapes, An Analysis of Defects our Tests are Missing”, *Proceedings of IEEE International Test Conference*, 1999, pp. 25–34.

[19] J. Rearick and P. Maxwell, “Deception by Design: Fooling Ourselves with Gate-level Models”, *Proceedings of IEEE International Test Conference*, 2000, pp. 921–929.

[20] C. H. Lee, K. H. Shen, T. K. Ku, C. H. Luo, C. C. Tso, H. W. Chou and C. Hsia, “CVD Cu Technology Development for Advanced Cu Interconnect applications”, *Proceedings o‘f IEEE Interconnect Technology Conference*, 2000, pp. 242–244.

[21] J. Soden, R. Treece, M. Taylor and C. Hawkins, “CMOS IC stuck-open Fault Electrical Effects and Design Considerations”, *Proceedings of IEEE International Test Conference*, 1989, pp. 423–430.

[22] W. Maly, P. K. Nag and P. Nigh, “Testing Oriented Analysis of CMOS ICs With Opens”, *Proceedings of IEEE International Conference on Computer Aided Design*, 1988, pp. 344–347.

[23] J. A. Waicukauski, E. Lindbloom, B. K. Rosen and V. S. Iyengar, “Transition Fault Simulation”, *IEEE Design and Test of Computers*, Apr. 1987, pp. 32–38.

[24] J. L. Carter, V. S. Iyengar and B. K. Rosen, “Efficient Test Coverage Determination for Delay Faults”, *Proceedings of IEEE International Test Conference*, 1987, pp. 418–427.

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

[26] C. J. Lin and S. K. Reddy, “On delay fault testing in logic circuits”, *Proceedings of IEEE International Conference on Computer-Aided Design*, 1986, pp. 148–151.

[27] V. S. Iyengar, B. K. Rosen and L. Spillinger, “Delay test generation 1 – concepts and coverage metrics”, *Proceedings of IEEE International Test Conference*, 1988, pp. 857–866.

[28] V. S. Iyengar, B. K. Rosen and L. Spillinger, “Delay test generation 2– algebra and algorithms”, *Proceedings of IEEE International Test Conference*, 1988, pp. 867–876.

[29] V. S. Iyengar, B. K.. Rosen and J. A. Waicukauski, “On computing sizes of detected delay faults”, *IEEE Transactions on Computer-Aided Design on Integrated Circuits and Systems*, 1990, pp. 299–312.

[30] A. K. Pramanick and S. M. Reddy, “On the fault coverage of gate delay fault detecting tests”, *IEEE Transactions on Computer-Aided Design on Integrated Circuits and Systems*, 1997, pp. 78–94.

[31] K. Heragu, J. H. Patel and V. D. Agrawal, “Segment Delay Faults : A new fault model”, *Proceedings of IEEE VLSI Test Symposium*, 1996, pp. 32–39.

[32] K. Heragu, J. H. Patel and V. D. Agrawal, “SIGMA: A simulator for segment delay faults”, *Proceedings if IEEE/ACM International Conference on Computer-Aided Design*, 1996, pp. 502–508.

[33] G. L. Smith, “Model for Delay Based Upon Paths”, *Proceedings of IEEE International Test Conference*, 1985, pp. 342–349.

[34] K. Baker, G. Gronthoud, M. Lousberg, I. Schanstra and C. Hawkins, “Defect-Based Delay Testing of Resistive Vias-Contacts : A Critical Evaluation”, *Proceedings of IEEE International Test Conference*, 1999, pp. 467–476.

[35] A. Pierzynska, “Accounting for Electrical Phenomena in Delay Fault Testing”, *Ph.D. Dissertation, Simon Fraser University*, 1999.

[36] M. Sivaraman, “Signal stablization analysis for timing verification and delay fault testing”, *Ph.D, Dissertation, Carnegie Mellon University*, 1997.

[37] H. Chang, “Strategies for design and test of high-performance systems”, *Ph.D. Dissertation, University of Texas at Austin*, 1993.

[38] I. Koren and Z. Koren, “Defect tolerance in VLSI circuits: techniques and yield analysis”, *Proceedings of IEE*, 1998, pp. 1819–1837.

[39] W. Mao and M. D. Ciletti, “Robustness enhancement and detection threshold reduction in ATPG for gate delay faults”, *Proceedings of IEEE International Test Conference*, 1992, pp. 588–596.

[40] 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”, *Proceedings of International Conference on Computer-Aided Design*, 1991, pp. 180–183.

[41] I. Pomeranz and S. M. Reddy, “Test generation for path delay faults based on learning”, *Proceedings, International Conference on Computer-Aided Design*, 1993, pp. 428–435.

[42] I. Pomeranz and S. M. Reddy, “NEST : A nonenumerative test generation method for path delay faults in combinational circuits”, *IEEE Transactions on Computer-Aided Design*, 1995, pp. 1505–1515.

[43] S. T. Chakradhar, M. A. Iyer and V. D. Agrawal, “Energy Minimization Based Delay Testing”, *Proceedings of IEEE European Design Automation Conference*, 1992, pp. 280–284.

[44] E. S. Park and M. R. Mercer, “An efficient delay test generation system for combinational logic circuits”, *IEEE Transactions on Computer-Aided Design*, 1992, pp. 926–938.

[45] S. Patil and S. Reddy, “A test generation system for path delay faults”, *Proceedings of IEEE International Conference on Computer Design*, 1989, pp.40–43.

[46] Standard Delay Format Specification, *http://www.eda.org/SDF*.

[47] Design Compiler, *http://www.synopsys.com/products/logic/designcompiler.html*.

[48] W-C. Lai, K. T. Cheng and A. Krstic, “Test program synthesis for path delay faults in microprocessor cores”, *Proceedings of IEEE International Test Conference*, 2000, pp. 1080–1089.

[49] R. S. Tupuri and J. A. Abraham, “A Novel Functional Test Generation Method for Processors”, *Proceedings of IEEE International Test Conference*, November 1997, pp. 743–752.

[50] R. S. Tupuri, A. Krishnamachary and J. A. Abraham, “Test Generation for Gigahertz Processors Using an Automatic Functional Constraint Extractor”, *Proceedings of the 36th ACM Design Automation Conference*, 1999, pp.647–652.

[51] D. Brand and V. S. Iyengar, “Timing Analysis Using Functional Analysis”, *IEEE Transactions on Computers*, 1988, pp. 1309–1314.

[52] R. B. Hitchcock, G. L. Smith and D. Cheng, “Timing Analysis of Computer Hardware”, *IBM Journal of Research and Development*, 1982, pp. 100–105.

[53] S. Perremans, L. Claesen and H. De Man, “Static Timing Analysis of Dynamically Sensitizable Paths”, *Proc. 26th ACM/IEEE Design Automation Conference*, 1989, pp. 568–573.

[54] W. N. Li, S. M. Reddy and S. K. Sahni, “On Path Selection in Combinational Logic Circuits”, *IEEE Transactions on Computer-Aided Design*, 1989, pp. 56–63.

[55] G. M. Luong and D. M. H. Walker, “Test Generation for Global Delay Faults”, *Proceedings of IEEE International Test Conference*, 1996, pp. 433–442.

[56] H. Chang and J. A. Abraham, “Vigorously Sensitizable Path Extractor”, *Proceedings of the 30th ACM Design Automation Conference*, 1993, pp. 112–117.

[57] A. Krishnamachary, J. A. Abraham and R. S. Tupuri, “Timing Verification and Delay Test Generation for Hierarchical Design”, *Proceedings of IEEE International Conference on VLSI Design*, January 2001, pp. 157–162.

[58] A. Krishnamachary and J. A. Abraham, “Test Generation for Resistive Opens in CMOS”, *Proceedings of ACM Great Lakes Symposium on VLSI Design*, April 2002.

[59] A. Krishnamachary and J. A. Abraham, “Effects of Multi-cycle Sensitization on Delay Tests”, *Proceedings of IEEE International Conference on VLSI Design*, January 2002.

[60] P. Agrawal, V. D. Agrawal , S. C. Seth, “Generating Tests for Delay Faults in Non-Scan Circuits”, *IEEE Design & Test of Computers*, March 1993, pp. 20–28.

[61] R. Jayabharathi, K. T. Lee and J. A. Abraham, “A Novel Solution for Chip-Level Functional Timing Verification”, *Proceedings of the IEEE VLSI Test Symposium*, 1997, pp. 137–142.

[62] J. L. Carter, V. S. Iyengar and B. K. Rosen, “Efficient Test Coverage Determination for Delay Faults”, *Proceedings of the IEEE International Test Conference*, September 1987, pp. 418–427.

[63] G. L. Smith, “Model for Delay Faults Based upon Paths”, *Proceedings of the IEEE International Test Conference*, September 1985, pp. 342–349.

[64] A. K. Mahji and V. D. Agrawal, “Tutorial: Delay Fault Models and Coverage”, *Proceedings of IEEE International Conference on VLSI Design*, January 1998, pp. 364–369.

[65] Y. Hoskote, D. Moundanos and J. A. Abraham, “Automatic Extraction of the Control Flow Machine and Application to Evaluating Coverage of Verification Vectors”, *Proceedings of the IEEE International Conference on Computer Design*, 1995, pp. 532–537.

[66] J. Rearick, “Too Much Delay Fault Coverage is a Bad Thing”, *Proceedings of IEEE International Test Conference*, 2001, pp. 624–633.

[67] B. Dervisoglu and G. Stong, “Design for Testability: Using Scanpath Techniques for Path-Delay Test and Measurement,”, *Proceeding of IEEE International Test Conference*, 1991, pp. 365–374.

[68] J. Savir, “Skewed-Load Transition Test: Part I, Calculus,” *Proceedings of IEEE International Test conference*, 1992, pp. 705–713.

[69] J. Savir, “ “Skewed-Load Transition Test: Part I, Calculus,” *Proceedings of IEEE International Test conference*, 1992, pp. 714–721.

[70] J. Savir and S. Patil, “On Broad-Side Delay Test”, *IEEE Transactions on Very Large Integration System*, 1994, pp. 368–372.

[71] R. Jayabharathi, “Hierarchical Timing Verification and Delay Fault Testing”, *Ph. D. Dissertation, The University of Texas at Austin*, 1999.

[72] E. Malavasi, S. Zanella, J. Uschersohn, M. Misheloff and C. Guardiani, “Impact analysis of process variability on deigital circuits with performance

limited yield", *Proceedings of IEEE International Workshop on Statictical Methodology*, 2001, pp. 60–63.

[73] J. J. Liou, K. T. Cheng and D. A. Mukherjee, "Path selection for delay testing of deep sub-micron devices using stastical performance sensitivity analysis", *Proceedings of International Test Conference*, 2000.

[74] M. Cuviello, S. Dey, X. Bai and Y.Zhao, "Fault modelling and simulation for crosstalk on system-on-chip interconnects", *Proceeding of International Conference on Computer-Aided Design*, 1999, pp. 297–303.

[75] A. Rubio, N. Itazaki, X.Xu and K. Kinoshita, "An approach to the analysis and detection of crosstalk faunts in digital VLSI circuits", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 1994, pp. 387–395,.

[76] K. T. Lee, C. Nordquist and J. A. Abraham, "Automatic test pattern generation for crosstalk glitches in digital circuits", *Proceeding of IEEE VLSI Test Symposium*, 1998, pp. 34–39.

[77] W. Chen, S. Gupta and M. Breuer, "Test generation for crosstalk induced delay in integrated circuits", *Proceedings of IEEE International Test Conference*, 1999, pp. 191–200.

[78] P. Goel, "An implicit enumeration algorithm to generate tests for combinational logic circuits", *IEEE Transactions on Computers*, 1981, pp. 215–222.

[79] J. A. Cunningham, "The use and evaluation of yield models in integrated circuit manufacturing", *IEEE Transactions on Semiconductor Manufacturing*, 1990, pp. 60–71.

[80] L. E. Jackson, “A user guide to principal component analysis”, 1991.

[81] G. E. Box and N. R. Draper, “Empirical Model Building and Response Surfaces”, *J. Wiley & Sons*, 1987.

[82] A. K. Majhi, V. D. Agrawal, J. Jacob and L. M. Patnaik, “Line Coverage of Path Delay Faults”, *IEEE Transactions on Very Large Scale Integration Systems*, Oct. 2000, pp. 610–613.

[83] I. Pomeranz and S. M. Reddy, “Static Test Compaction for Scan-Based Designs to Reduce Test Application Time”, *Proceedings of IEEE Asian Test Symposium*, 1998, pp. 198–203.

[84] E. M. Rudnick and J. H. Patel, “A Genetic Approach to Test Application Time Reduction for Full Scan and Partial Scan Circuits”, *Proceedings of IEEE Internation Conference on VLSI Design*, 1995, pp. 288–293.

[85] L. G. Silva, J. P. M. Silva, L. M. Silveira and K. A. Sakallah, “Realistic Delay Modeling in Satisfiability-Based Timing Analysis”, *Proceedings of IEEE International Symposium on Circuits and Systems*, 1998, pp. 215–218.

[86] H. Konuk, “On Invalidation Mechanisms for Non-Robust Delay Tests,” *Proceedings of IEEE International Test Conference*, pp. 393–399, 2000.

[87] M. W. Levi, “CMOS is most testable”, *Proceeding of International Test Conference*, 1981, pp. 217–220.

[88] D. Baschiera and B. Courtois, “Testing CMOS : A challenge”, *IEEE Transactions of VLSI Design*, 1984, pp. 58–62.

[89] M. E. Turner, D. G. Leet, R. J. Prilik and D. J. McLean, “Testing CMOS VLSI : Tools, Concepts and experimental results”, *Proceedings of International Test Conference*, 1985, pp. 322–328.

[90] M. Breuer, “Effects of races, delay and delay faults on test generation”, *IEEE Transactions on Computers*, 1974, pp. 1078–1092.

[91] T. Hayashi et al, “A delay test generator for logic LSI”, *Proceedings of fault-tolerant computing symposium*, 1984, pp. 146–149.

[92] J. D. Lesser and J. J. Shedletsky, “An experimental delay test generatorfor LSI logic”, *IEEE Transactions on Computers*, 1980, pp. 235–248.

[93] Y-C. Hsu and S. K. Gupta, “An automatic test pattern generator for at-speed robust path delay testing”, *Proceedings IEEE Asian Test Symposium*, 1998, pp. 88–95.

[94] A. Krstic, Y-M. Jiang and K. T. Cheng, “Delay testing considering power supply noise effects”, *Proceedings International Test Conference*, 1999, pp. 181–190.

[95] S. Tani, M. Teramoto, T. Fukazawa and K. Matsuihiro, “Efficient path selection for delay testing based on partial path evaluation”, *Proceedings IEEE VLSI Test Symposium*, 1998, pp. 188–193.

[96] Y. K. Malaiya and R. Narayanaswamy, “Modeling and testing of timing faults in synchronous sequential circuits”, *IEEE Design and Test of Computers*, 1984, pp. 62–74.

[97] P. Varma, “On path delay testing in a standard scan environment”, *Proceedings of IEEE International Test Conference*, 1994, pp. 164–173.

Vita

Arun Krishnamachary was born in Madras, India, on December 24th, 1975 to Malathi and J.Krishnamachary. He graduated from Padma Seshadri Bala Bhavan Senior Secondary School in Madras, Tamil Nadu in 1993 . He received his Bachelor of Engineering degree in Computer Science and Engineering from The University of Madras in 1997 and MSEE from The University of Texas at Austin in 1999. Since then he has been pursuing his Ph.D. at The University of Texas at Austin. Between August 1997 to August 2002, he was a Research Assistant in the Computer Engineering Research Center with Dr. Jacob A. Abraham. Since August 2002 he has been working at Intel Corporation at Phoenix, AZ.

Permanent Address: 3 Singara Mudali Street Apt #5, T.Nagar, Madras-600017, India

This dissertation was typeset with LaTeX 2_ε [1] by the author.

[1] LaTeX 2_ε is an extension of LaTeX. LaTeX is a collection of macros for TeX. TeX is a trademark of the American Mathematical Society. The macros used in formatting this dissertation were written by Dinesh Das, Department of Computer Sciences, The University of Texas at Austin, and extended by Bert Kay and James A. Bednar.