Copyright
by
Ramyanshu Datta
2006

The Dissertation Committee for Ramyanshu Datta
certifies that this is the approved version of the following dissertation:

Parametric Testing, Characterization and Reliability of Integrated Circuits

Committee:

Jacob A. Abraham, Supervisor

Anthony P. Ambler

Nur A. Touba

Zhigang Pan

Kevin J. Nowka

Robert K. Montoye

Parametric Testing, Characterization and Reliability of Integrated Circuits

by

Ramyanshu Datta, B.E., M.S.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 2006

Dedicated to my family.

Acknowledgments

First and foremost, I would like to thank my supervisor, Jacob Abraham, for providing me with the guidance, support and various opportunities he has provided me with throughout the duration of my graduate studies. His insights and suggestions have been a guiding force behind this work, and he continues and will continue to remain a source of inspiration for me in all my professional endeavors.

I would like to thank Anthony Ambler, Nur Touba and David Pan from The University of Texas at Austin, and Kevin Nowka and Robert Montoye from IBM Corporation for agreeing to be members of my dissertation committee.

During my stay at CERC at UT Austin, I had the opportunity to work and interact with a number of colleagues. I would like to thank Antony Sebastine, Ravi Gupta, Whitney J. Townsend, Adam Tate, Ashwin Raghunathan, Hong Joong Shin, Paul Chun, Byoung Ho Kim, Akshay Gupta, Sriram Sambamurthy, Sankaranarayanan Gurumurthy, Ramtilak Vemu, Rajeshwary Tayade, Chaoming Zhang, Ji Seon Park, Joon Sung Park, Tung-yeh Wu, Qingqi Dou, Roopsha Samanta and Jyotirmoy Deshmukh for all the support they have provided me during my staty at CERC. I would also like to thank Linda Frost,

Debi Prather, Andrew Kieschnick, Melanie Gulick, Melissa Campos, Shirley Watson and Ruth Kubacka for all the assistance in administrative matters that they have provided me.

I held several internship positions at IBM during my graduate studies, where I had the opportunity to interact professionally with a large number of people who contributed greatly to my knowledge. I would like to thank Peter Hofstee, Sani Nassif, Gary Carpenter, Mack Riley, Matthew Fernsler, Wendy Belluomini, Damir Jamsek, Joy Acharyya, Jun Sawada Hung Ngo, Chandler Mcdowell and J.B. Kuang for all their help and support.

During my tenure as a graduate student, I have had the good fortune of forging some really good friendships. I would like to thank Narayan Nair, Vinit Ogale, Jay Shah, Divi Lohiya, Alok Lohiya, Apurva Gupta, Robin Ozah, Anubhav Jain, Debarshi Basu, Suvid Nadkarni and Vipin Sachdeva for making my stay in Austin a pleasant one. A special thank you goes to Priya Chandra for making every day appear as a joyful and special one.

Above all, I would like to thank my parents Sushmita and Biswajit Datta, my sister Dibyadyuti for the countless sacrifices, never-ending affection and continuous encouragement that have been the motivation behind any accomplishment I have made, and to them I owe every good thing I have today. I would also like to thank my grandparents, Indira Dutt, Jaba Sinha, Jibanananda Sinha and Santosh Kr. Dutt and my whole family for always being a great source of support and inspiration for me.

Parametric Testing, Characterization and Reliability of Integrated Circuits

Publication No. ____________________

Ramyanshu Datta, Ph.D.
The University of Texas at Austin, 2006

Supervisor: Jacob A. Abraham

This work deals with the problem of parametric failures in Integrated Circuits (ICs), focussing specifically on timing, which is one of the most important parameters in modern ICs. Two approaches to tackling timing violations are explored, the first being efficient timing characterization, involving delay test and debug, to screen out defective parts, and the second, timing oriented adaptive design for variability related failures.

Timing violations are a major source of defective silicon for ICs designed in Deep Sub-micron (DSM) technologies. This is because the performance requirements of such ICs are very high, leading to reduced slack margins, and also because defects and variations in process parameters significantly impact their behavior. However, smaller feature sizes and higher levels of integration, which are characteristic of DSM ICs, have severely limited their controllability and observability, hence hindering efficient timing characterization. In this work, techniques to enhance controllability and observability for timing characterization of ICs, using novel Design for Test and Design for Debug techniques are presented.

In addition to defects, variations in process parameters also impact the behavior of DSM ICs, and can cause a large number of defect free parts to fail the test process and be discarded, leading to reduction in manufacturing yield. An approach for combatting variations is the use of adaptive or variation aware design. In this work, adaptive design techniques with a focus on timing, i.e., performance-optimized adaptive design, are explored. These techniques ensure that adaptation does not cause a chip to violate timing specifications, and also enable a chip to reconfigure itself to reduce or eliminate variability related timing violations, hence enabling parametric reliability in ICs.

Table of Contents

Acknowledgments **v**

Abstract **vii**

List of Tables **xi**

List of Figures **xii**

Chapter 1. Introduction **1**

1.1 Controllability Issues in Timing Characterization of ICs 5
1.1.1 Controllability Issues in Static CMOS circuits 5
1.1.2 Controllability Issues in Dynamic Circuits 11
1.2 Observability Issues in Timing Characterization of ICs 18
1.3 Variability and Parametric Reliability 24
1.4 Summary of Contribution 26
1.5 Outline of Dissertation 27

Chapter 2. Controllability of Static CMOS Circuits for Timing Characterization **29**

2.1 Tri-Scan Scheme 30
2.2 Simulation Results 46
2.3 Summary 58

Chapter 3. Controllability of Dynamic Circuits for Timing Characterization **60**

3.1 Dynamic Circuit Families 61

3.2 Precharge Control Schemes 66
3.3 Delay Test and Debug of Scan Based Dynamic Circuits 75
3.4 Simulation Results 80
3.5 Summary 85

Chapter 4. Response Analysis for Timing Characterization 87
4.1 Modified Vernier Delay Line 90
4.2 Delay Scan Chain 106
4.3 Skewed Inverter Delay Line 112
4.4 Simulation Results and Discussion 122
4.5 Summary 133

Chapter 5. Adaptive Design for Parametric Reliability 135
5.1 Process Variation Sensing Schemes 139
5.2 Process Compensation Mechanisms 145
  5.2.1 Process Compensation for Performance-Optimized Robustness 145
  5.2.2 Correction of Process Variation Related Timing Failures 150
5.3 Simulation Results 153
5.4 Summary 161

Chapter 6. Conclusion and Future Work 163

Appendices 171

Appendix A. List of Acronyms and Abbreviations 172

Bibliography 174

Vita 185

List of Tables

2.1 Performance overhead of Tri-Scan scheme 57
2.2 Fault coverage of regular scan and Tri-Scan schemes 58

4.1 Example to illustrate behavior of MVDL considering metastability issues 94
4.2 Handling all possible paths and transitions using the same MVDL 101
4.3 Modes of operation of DSCAN flip-flop 108
4.4 Propagation delay measurement of critical paths of test circuits using MVDL 123
4.5 Behavior of MVDL at different technologies 124
4.6 Propagation delay measurement of critical paths of test circuits using *Delay Scan Chain* 126
4.7 Propagation delay measurement of critical paths of test circuits using SIDL 130
4.8 Behavior of SIDL across different technologies 131
4.9 Behavior of SIDL in presence of L Variations 132

5.1 Variation of MVDL resolution with variation in channel length L 157
5.2 Variation of MVDL resolution with variation in supply voltage 160

List of Figures

1.1 Two stage logic to illustrate delay fault testing with complete access to all inputs 6
1.2 Combinational logic with launch and capture scan chains to illustrate the controllability problem of delay fault testing in scan based designs 7
1.3 Skewed load transition test scheme for delay testing in scan based designs 9
1.4 Broad side delay test scheme for delay testing in scan based designs 10
1.5 Circuit level schematic of two level AND gate using domino circuit family to illustrate the problem of delay fault testing and debug of dynamic circuits 13
1.6 Circuit level schematic to illustrate problem with using precharge as the initialization vector 15

2.1 Two stages of combinational logic with scan in between, and driver shown explicitly 31
2.2 a) Regular inverter b) Tri-state inverter 32
2.3 Tri-Scan scheme 33
2.4 Tri-Scan scheme with scan enable being reused for controlling the tristate buffer 34
2.5 Example circuits for comparing skewed-load transition test and Tri-Scan 37
2.6 Minimum voltage value at output of tristate inverter with different sized loads 48
2.7 Voltage value at the output of the tristate inverter over 2000ns with a capacitive load equivalent to a) 200μm transistor width b) 36μm transistor width 49

2.8 Voltage value at the output of the tristate inverter over 20000ns with a capacitive load equivalent to a) 200μm transistor width b) 36μm transistor width 50
2.9 Flip-flops used for scan design 51
2.10 Area overhead of 48 scan and tri-scan chains of different sizes using a) flip-flop f1 b) flip-flop f2 52
2.11 Area overhead of 48 scan and tri-scan chains of different sizes using a) flip-flop f3 b) flip-flop f4 53
2.12 Power dissipation during scan operation of 48 scan and tri-scan chains of different sizes using a) flip-flop f1 b) flip-flopf2 54
2.13 Power dissipation during scan operation of 48 scan and tri-scan chains of different sizes using a) flip-flop f3 b) flip-flopf4 55
2.14 a) Relative area overhead of scan and tri-scan in a chip b) Relative power overhead of scan and tri-scan in a chip during scan operation 56

3.1 Domino operation 62
3.2 LSDL NAND gate 65
3.3 *Precharge Control Scheme 1* 67
3.4 Timing diagram for *Precharge Control* schemes 68
3.5 *Precharge Control Scheme 2* 70
3.6 Drawback of using a single transistor for precharge control . . 71
3.7 Case study: *LSDL 4-to-2 Carry Save Adder* 72
3.8 Two stage combinational logic with scan 76
3.9 a) Dynamic buffer b) *AND Gated Flop Output* scheme 77
3.10 *Evaluate Control* scheme 78
3.11 *ANDed Evaluate Clock* scheme 80
3.12 *Precharge Control Scheme 1* a) Area-performance curve b) Power-performance curve 81
3.13 *Evaluate Control* scheme a) Area-performance curve b) Power-performance curve 83
3.14 *ANDed Flop Output* scheme a) Area-performance curve b) Power-performance curve 84
3.15 Power-performance curve illustrating power reduction during scan operation 84

4.1 *Modified Vernier Delay Line* 91

4.2 *Modified Vernier Delay Line* with capability to handle all possible paths and transitions 100
4.3 Timing characterization using MVDL in a scan based environment 105
4.4 a) Regular scan flip-flop and b) DSCAN flip-flop 107
4.5 *Delay Scan Chain* 109
4.6 Timing characterization using *Delay Scan Chain* 110
4.7 a) Pulse generator b) Pulse shaper 112
4.8 *Skewed Inverter Delay Line* scheme 114
4.9 Timing characterization using SIDL 119
4.10 Percentage area and power overhead of a *Delay Scan Chain* with respect to a regular scan chain 128
4.11 Driver size vs. interconnect length for the SIDL scheme 131

5.1 MVDL-PD 140
5.2 MVDL-OHE 144
5.3 Noise Tolerance Mechanisms 146
5.4 Process compensation for a) static CMOS circuits b) dynamic circuits 147
5.5 Correction of process variation related timing failures 151
5.6 Latency variation across process corners for a) Static CMOS w4 circuit with sleep transistors b) Dynamic w4 circuit with keepers 154
5.7 Variation of resolution of a) stage 1 and b) stage 2 of an MVDL across process corners 155
5.8 Variation of resolution of a) stage 3 and b) stage 4 of an MVDL across process corners 156
5.9 Variation of average MVDL resolution across process corners . 156
5.10 Delay obtained by varying L and sweeping a) Width of sleep transistor in static CMOS version of w4 b) Width of keeper in dynamic version of w4 158

Chapter 1

Introduction

Parametric failures, caused by violation of one or more of the parameter specifications in Integrated Circuits (ICs), have become a major source of failure in modern ICs designed in Deep Sub-micron (DSM) technologies. The focus of this work is on timing, which is one of the most important of the various parameters of an IC. Violations of timing specification of an IC, are known as delay faults [68], and hence the terms timing violations and delay faults are used interchangeably in this work.

Delay faults are a major source of failure in modern ICs. This is due to a multitude of factors. Firstly, aggressive timing requirements of modern chips which for microprocessors are in the multi-GHz range (e.g., [92], [102]), require designs to be pushed for maximal performance, leading to reduced slack margins. Secondly, because of shrinking feature sizes and higher level of integration, obtained by technology scaling, defects have a much greater impact on chip behavior, including timing behavior, in DSM technologies, than they did earlier. Additionally, in DSM technologies, limited control on the manufacturing process leads to variations in process parameters [95]. Such variations have a significant impact on behavior of DSM chips and make it difficult to correctly predict their timing behavior, which, in turn, could lead to timing violations in manufactured chips. Designers are moving towards aggressive design styles like statistical design to tackle variability [47], [95], which further increases the possibility of timing violations. Delay faults could be caused by a large number of factors, such as under-performing gates, defects like hard opens or shorts, open or highly resistive vias and contacts, resistive interconnects, weak opens, process variations, and environmental factors like crosstalk and variations in supply voltage and temperature [6], [14], [17], [47], [58], [98], [121].

Delay faults are defined as faults that cause the timing of a chip, or a path within a chip, to fall out of specification. Unlike stuck-at faults, testing for delay faults requires application of two patterns, namely, an initialization vector to bring the circuit or path under test to a known and desired state, and a transition vector, which acts upon this initialized state of the circuit to activate the delay fault, and propagate its impact to the nearest output or observable point [68], [77]. Several techniques have been proposed to model delay faults, which range from lumped models like transition [75] and gate [61] delay fault models, to distributed models like the path delay fault model [74], [77]. Delay faults can be categorized as gross delay faults, or delay faults that cause a chip to fail at-speed, and small delay faults, which cause a small increment in the delay of certain paths within a chip [97], [101]. An efficient test strategy needs to detect both classes of delay faults, the reasons for which are elucidated later in this chapter, in Section 1.2.

In addition to efficient testing of chips, there is also a need for expeditious silicon debug, especially for first silicon. This is primarily because first silicon debug contributes significantly to time-to-market [7], [41], [136]. Additionally, an increase in the number of silicon spins causes an increase in overall cost of the chip. A systematic debug methodology is thus required for expeditious silicon debug [7], [41], [136].

In addition to screening of defective parts, and ensuring fewer silicon spins, another role of test and debug is in the area of Design for Manufacturability (DFM) [138]. This is because the data obtained from testing of high volume chips as well as from silicon debug, can be used to understand necessary improvements in the design flow to enhance yield [80].

In this work, two approaches are utilized to tackle timing violations in ICs. The first pertains to development of techniques to enable efficient delay test and debug, which are jointly referred to as timing characterization in this work. An efficient timing characterization scheme can effectively screen out defective parts during the test process, but in DSM technologies it may also result in a lot of defect free parts being discarded. This is primarily because variations in process and environmental parameters in such technologies can cause such a defect free die to fall out of specification. This in turn, could lead to significantly low yield numbers, and reduce the overall profitability of a product. Hence the second approach in this work pertains to timing oriented compensation for variability in ICs, to minimize chances of timing violations caused by process and environmental variations. Such tolerance to process and environmental variations is termed as parametric reliability in this work.

It is clear from the material and literature survey presented till now, that efficient timing characterization is imperative for modern ICs designed in DSM technologies. However, a significant hurdle for test and debug of such ICs is the problem of limited controllability and observability, which stems primarily from shrinking feature sizes and higher levels of integration in DSM ICs. This is more so for timing characterization, because it also suffers from the problem of slow speed testers. A large number of ICs are well in the GHz range, but most testers still operate at frequencies in the order of few hundred MHz. High speed testers are very expensive. Traditionally, controllability and observability limitations in test have been overcome by incorporation of Design for Test (DFT) structures in a chip. The most common of these are scan chains [48], which are very effective for stuck-at fault testing, enabling the same level of controllability for sequential circuits as that of combinational circuits with complete access to all inputs.

However, delay fault testing is different from stuck-at fault testing, since it requires use of two vectors. Hence a regular scan design makes the task of pattern application (controllability) for delay fault testing difficult, and these problems are explained in detail in Section 1.1, along with existing techniques and their drawbacks. Observability limitations for delay fault testing in scan based designs, (also known as AC scan tests) [85], [101] pertain to gross margins used to for such tests, which curtails its ability to detect small delay faults. To enhance the resolution, iterative techniques like clock schmooing are applied, which has several drawbacks like increased test time, that are discussed in Section 1.2, along with other existing solutions and their drawbacks.

As has been mentioned earlier, process and environmental variations can also cause timing violations. These variations occur due to limited control on the manufacturing process, and continue to become more significant with each new generation of technology. It was estimated in [17], that in sub-90nm technologies, almost a generation of performance could be lost in some cases due to process parameter variations. In Section 1.3, an overview of the impact of variability in DSM ICs, and approaches to ensure reliability in face of variations are discussed.

1.1 Controllability Issues in Timing Characterization of ICs

1.1.1 Controllability Issues in Static CMOS circuits

The problem of delay fault testing for circuits with complete access to all inputs is explained using Figure 1.1. The figure shows two stages of logic comprised of AND gates, with A, B, C, and D as primary inputs. The delay fault exists in the Path Under Test (PUT) from A to *out* (the path delay fault model [74], [77] is assumed in this chapter, but the concepts can easily be extended to other models). As mentioned before, delay fault testing requires two vectors, namely an initialization vector (V_1) and a transition vector (V_2). In Figure 1.1, V_1 is logic value 0 on input A and logic value 1 on inputs B, *C*, and *D*. This brings the output *out* to its initialized state, i.e., logic value 0. Subsequently, the transition vector for this test is applied, which is a logic value 1 on all 4 inputs, and the requisite transition for delay fault testing of the PUT is caused on the path from *A* to *out*.

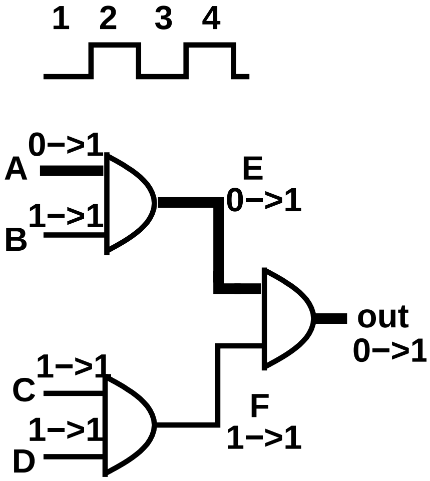


Figure 1.1: Two stage logic to illustrate delay fault testing with complete access to all inputs

However, the problem becomes more complex in a scan based design. Consider a combinational logic as Circuit Under Test (CUT) with launch and capture scan, i.e., scan chains at the input and output of the logic, as shown in Figure 1.2. The initialization vector is scanned in through the launch scan chains (the scan chain on the left in Figure 1.2) and applied to the CUT to bring it to a desired state. However, as soon one starts scanning in the transition vector, the initialized state of the CUT is disturbed, precluding application of patterns for delay fault testing.

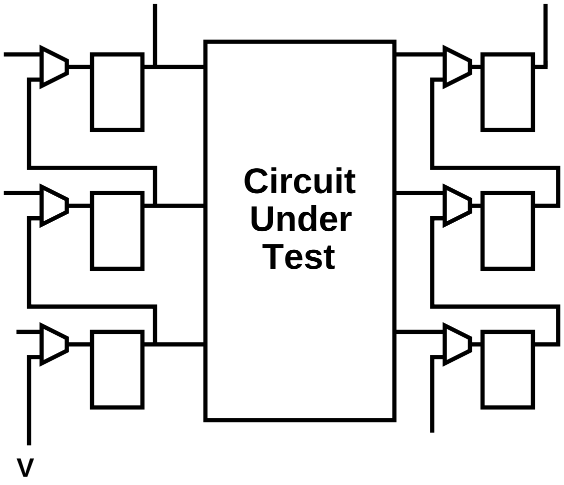


Figure 1.2: Combinational logic with launch and capture scan chains to illustrate the controllability problem of delay fault testing in scan based designs

Hence, the scan chain structure, which is very useful for stuck-at fault testing, cannot be used for delay fault testing in its existing form. Solutions to this problem include *enhanced scan* [42], [81], which utilizes an extra storage element for every scan flop to enable holding bits of V_1 and V_2. However, such a scheme is very expensive in terms of hardware overhead, since it would require an extra latch at every scannable storage element.

Another technique used for applying delay test patterns in scan based designs is called *skewed-load transition test* [117], [118]. This scheme is shown in Figure 1.3. The transition vector is obtained by shifting the initialization vector by one bit. This is shown in Figure 1.3, where V_1 is made up of bits *x0*, *x1*, and *x2*, and V_2 is made up of bits *x1*, *x2*, and *x3*. The advantage of this scheme is that only one cone of logic needs to be assessed for test generation.

However, only a fraction, i.e., $2^{-(n-1)}$ of the possible pairs of patterns (V_1, V_2) can be applied [132], leading to poor coverage [28]. Scan chain reordering [28], [84], [117] has been shown to improve coverage, but it adds substantial routing overhead while not providing sufficient coverage at all times [28]. Also, a particular set of paths may be sensitized by one particular ordering, whereas another set may be sensitized by another set of ordering, making it difficult to determine the right order. Additionally the timing of the *scan_enable* signal is critical and hence, the signal has to be routed carefully to ensure high speed operation [78], [120]. However, in [2], the authors present techniques that are practiced in the industry to address the problem of timing criticality of the *scan_enable* signal for *skewed-load transition test*, and also suggest some optimizations.

Broad side delay test, also known as *functional justification* [28], [119], is another technique utilized for delay fault testing in scan based designs [119]. The basic idea is to scan in V_1, and obtain V_2 as the response of a combinational logic block. An example implementation of this scheme using two pipeline stages is illustrated in Figure 1.4. In this scheme, the initialization vector V_1 is scanned into the launch scan path of the CUT. Simultaneously, another vector V'_1 is scanned into the launch scan path of the combinational logic stage preceding the CUT. This vector V'_1 is such that the response of the preceding combinational logic to this vector forms the transition vector V_2 for the CUT. However, this scheme has its drawbacks, which include the fact that two cones of logic need to be assessed for ATPG purposes, and also, from

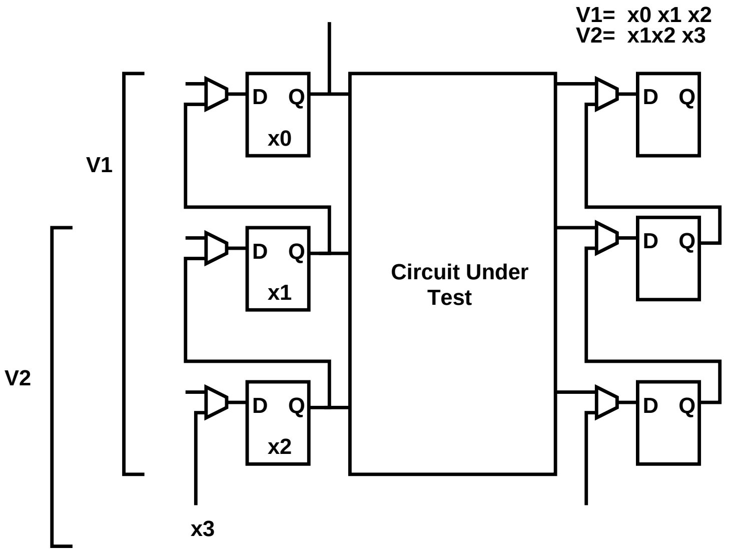


Figure 1.3: Skewed load transition test scheme for delay testing in scan based designs

a debug perspective, if there is a delay fault in the preceding combinational logic, then it is very difficult to determine whether a delay fault exists in the CUT.

Other solutions presented in the literature include [54], [130], [132]. A method based on clock control, which requires an overhead commensurate to *enhanced scan* is presented in [130]. The authors in [132] present a method called *scan-mapping*, wherein the initialization vector is scanned in, and the transition vector generated utilizing mapping logic that is added on to flip-flops. This scheme requires generation of complex mapping logic, which increases complexity of test generation, and adds overhead to scan flip-flops, with the amount of overhead being dependent on the mapping function. Ad-

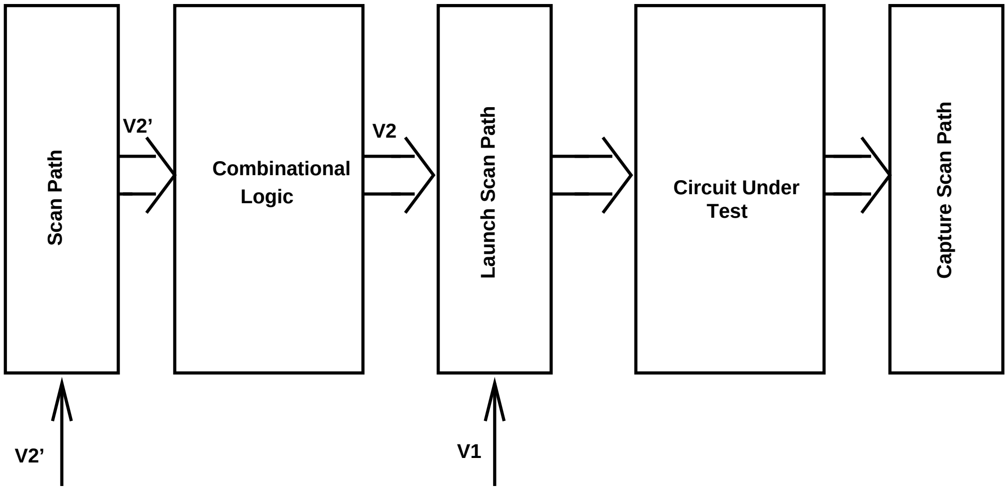


Figure 1.4: Broad side delay test scheme for delay testing in scan based designs

ditionally, *scan-mapping* does not always provide high coverage due to lack of complete accessibility. In [54] a technique is presented for two-latch systems, wherein the second latch is cut off during the scan operation, and the scan path is dynamic and occurs through the dynamic node in between the two latches through pass transistors, which could be susceptible to leakage charge sharing and other noise sources [122] which are prevalent in DSM technologies. Solutions were presented for testability of stuck-open faults (which are effectively delay faults with infinite delay) in [18], wherein the authors developed two new circuit families which are variants of static CMOS circuit family. These circuit families, called *DFCMOS* and *TFCMOS* required insertion of transistors in between the output node and the PMOS and NMOS networks, i.e., a total of two additional transistors per gate to enable testability. This solution was presented for enhancing stuck-open testability of circuits in general, without any mention of scan. However, the problem of using such a design is that the area overhead would be too high, since every gate would have two additional transistors. Additionally, every gate or cell in the library would have to be modified, which may not be a feasible option.

1.1.2 Controllability Issues in Dynamic Circuits

The operation of dynamic circuits are very different from static CMOS circuits, thus making it imperative to address their testing issues differently. In dynamic circuits, all computation takes place during a phase of the clock called *evaluate*, and the output of all dynamic circuits are reset during the other phase of the clock, which is called *precharge*. Usually *precharge* occurs during the period when the clock signal is at logic value 0, and *evaluate* occurs during the period when the clock signal is at logic value 1. Dynamic circuits implement positive unate circuit functions, (complement logic can be implemented using dual-rail), and hence an even number of inverting gates always feeds into a dynamic gate (Limited Switching Dyanmic Logic or LSDL circuit family is an exception). In Chapter 3, the operation dynamic circuits is explained in more detail. In this section, the problem of delay fault testing in dynamic circuits and drawbacks of existing solutions are presented.

Due to the difference in the way dynamic and static CMOS circuits operate, it is not possible to carry out the two-pattern delay fault tests that has been explained above for static CMOS circuits, in dynamic circuits even with complete access to all inputs. This stems from the fact that response of a dynamic CUT to a vector occurs only during the *evaluate* phase, and in the period between two *evaluate* phases corresponding to application of the two vectors V_1 and V_2, a reset phase exists, which destroys the initialized state of the CUT.

The above phenomenon is illustrated with an example. Consider the two level AND gate circuit shown in Figure 1.1, and assume the PUT is from A to *out*, with the transition to be tested being a 0-to-1 transition at input A, and assuming complete access to all inputs. Consider the pattern application process at the gate level without making any assumptions about the underlying circuit family. The two clock cycles, divided into four phases numbered accordingly, are shown in the Figure 1.1. During the first cycle of the clock (which consists of phases 1 and 2), input A, B, C, D=0, 1, 1, 1, is applied as V_1, and during the second clock cycle (comprising phases 3 and 4), A, B, C, D=1, 1, 1, 1, is applied as V_2 to the circuit to test for the fault.

However, even with complete access, the problem is more complicated in dynamic circuits. Consider the circuit level implementation of the 2 stage AND gate circuit of Figure 1.1, shown in Figure 1.5. This circuit level implementation is shown using domino circuit family, which is a common dynamic circuit family [134]. The primary inputs, internal nodes, and the output of this circuit have names that correspond to those in the gate level schematic of Figure 1.1. Assume complete access to all inputs, i.e., A, B, C, and D.

The phase by phase operation of the dynamic circuit in Figure 1.5,

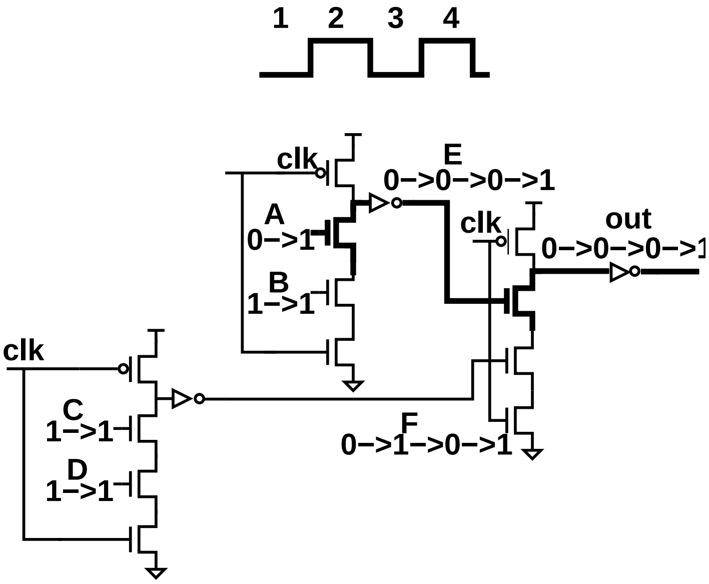


Figure 1.5: Circuit level schematic of two level AND gate using domino circuit family to illustrate the problem of delay fault testing and debug of dynamic circuits

upon application of the two vectors discussed for the circuit of Figure 1.1, is explained here. In phase 1, or the precharge phase of V_1, the internal nodes *E, F*, and the output *out* are *precharged* to a logic value 0 irrespective of the applied input vector (although it is a general practice to keep the pull down tree of a dynamic circuit switched off during precharge). In phase 2, i.e., the *evaluate* phase of V_1, the initialization vector $A, B, C, D=$ 0,1,1,1, acts upon the circuit, causing internal nodes E and F and output *out* to be initialized to their desired logic values, i.e., 0, 1 and 0 respectively.

However, phase 2 is immediately succeeded by phase 3, the *precharge* phase of V_2. This leads to internal nodes E and F, as well as output *out* to be *precharged* to logic values that they were at before application of the initialization vector. In the CUT, for the given pattern, this causes the logic value of the internal node F to be opposite of what is desired before application of transition vector. Thus we can see that in the given example, node F toggles between 0 and 1 in every phase. This condition precludes efficient delay fault testing of the circuit.

Existing techniques for delay fault testing utilize the *precharge* phase as the initialization vector [18], [62], [96]. Although this scheme works effectively when two dynamic gates drive parallel transistors of a third gate, the effectiveness of this scheme comes to question whenever two dynamic gates drive two transistors of a third gate that are connected in series. This can happen when two dynamic gates are driving a dynamic AND gate, static AND gate, series transistors of an XOR gate, or any other complex gate.

This problem is illustrated with the help of an example shown in Figure 1.6. Instead of a two cycle operation, delay test is just a single cycle operation now, as shown in the figure. This itself causes a problem, because, there is now an incompatibility with static testing tools. As before, the single cycle is divided into two phases. However, phase 1, which is a *precharge* phase, now doubles up as the initialization vector phase or V_1 phase, while phase 2, which is the *evaluate* phase, now doubles up as transition vector phase or V_2 phase. During the V_1 phase, internal nodes *E, F* and output *out* get initialized to a logic value 0, irrespective of the input vector. However, when the transition vector $A, B, C, D = 1,1,1,1$ is applied during the V_2 phase, it causes the dynamic node of both the upper and lower gates to discharge simultaneously.

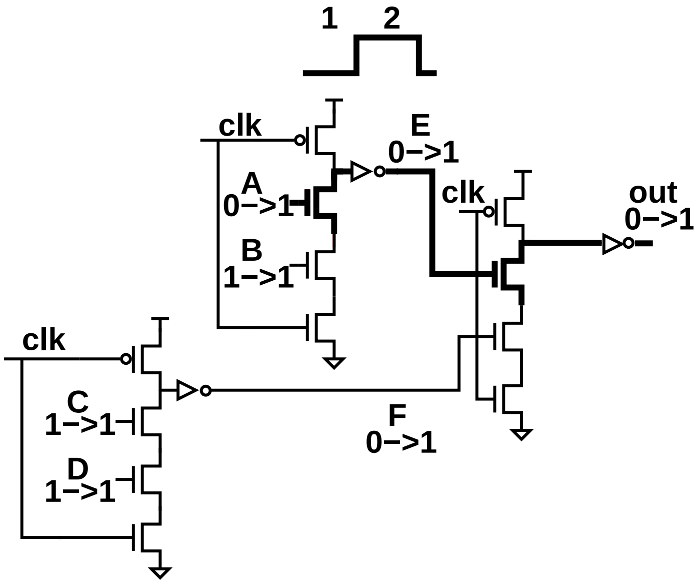


Figure 1.6: Circuit level schematic to illustrate problem with using precharge as the initialization vector

This causes both the internal nodes E and F to transition from logic value 0 to logic value 1 which in turn causes the output *out* to switch from a logic value 0 to logic value 1. However the problem occurs if there is a delay fault because then it becomes impossible to ascertain which set of paths is faulty, i.e., whether the PUT A to *out* is faulty, or whether a path like C to *out* is faulty.

Thus, in the above example, there is an ambiguity between any two given paths in delay fault testing of dynamic circuits, which does not exist for the static CMOS version of the above circuit. This situation will occur whenever two dynamic gates drive two transistors that are in series. In some cases, the probability of finding the faulty path would be $1/2^{n-1}$ where n is the number of levels of logic in a pipeline stage (and assuming all gates are two input gates).

One may consider that this is a characterization problem, and not of consideration in a go-no go kind of test. This is because, due to the way dynamic circuits operate, the aforementioned delay testing methodology that uses *precharge* phase as initialization vector is sufficient for determining presence of a failure in a CUT. However, the inability to determine which path failed, is a serious impediment to the post-silicon timing characterization process. This is because if the source of failure is unknown, the cause will be even more difficult to determine, and this will impede the rectification process before subsequent silicon spins, which in turn could lead to several other issues, like yield loss. Additionally, in the DSM era, with the significant time required for debug, and also with yield problems, it is necessary for the test process to facilitate easier silicon debug, and results of the test process needs to utilized to improve yield [80], requiring manufacturing test to evolve into more of a characterization test rather than being a go-no go test. Inability to determine which path failed every time a timing violation occurs, is a serious impediment to all the aforementioned objectives. For these reasons, the use of *precharge* as an initialization vector cannot be considered an effective solution for timing characterization of dynamic circuits.

ATPG techniques for delay testing dynamic circuits using different delay models have been presented in [18], [62], [70], [71], [73], [86], [96]. Test generation for delay faults in dynamic circuits under the impact of noise has been presented in [70]. All of these techniques utilize the *precharge* phase as initialization vector.

DFT techniques for dynamic circuits have been limited. In [23], a technique was presented to test dynamic circuits using low frequency clocks, wherein an additional transistor was introduced in the pull-down (NMOS) stack of the dynamic circuit. This transistor was controlled by a signal which was inverted and phase shifted with respect to the test clock. In [114], a DFT technique was presented that enabled current based testability of dynamic circuits by placing a parallel path to supply from the dynamic node, which could only be activated in test mode. However, to the best of knowledge, there has been no technique presented yet to address the aforementioned ambiguity.

Until now only problems for timing characterization of dynamic circuits with complete access to all inputs have been considered. However, dynamic circuits also suffer from the same problems as static CMOS circuits for timing characterization in scan based designs. The drawbacks of techniques like *enhanced scan* [42], *skewed-load transition test* [117], [118] and *broad-side delay test* [28], [119] have already been discussed earlier in this chapter. The *Tri-Scan* technique which was developed as a part of the work in this dissertation and which is presented in Chapter 2 is suitable for static CMOS circuits, but may not always be suitable for dynamic circuits since it involves insertion of static components.

Another impediment to testing of dynamic circuits stems from test economics. Unlike the static CMOS circuit family, there are not many CAD tools that support dynamic circuit families. This is primarily because dynamic circuits are generally custom design, have limited usage, and also most static test tools do not work very well with dynamic circuit families. Additionally, it may not be economically feasible for many semiconductor manufacturers to maintain an entirely separate suite of tools for dynamic circuit families. This problem is exacerbated by the fact that there are a variety of dynamic circuit families, and each may need its own set of tools.

1.2 Observability Issues in Timing Characterization of ICs

From an observability perspective, the requirements of an efficient timing characterization scheme are

- Detection of gross delay faults, i.e., delay faults that cause violation at rated speed.
- Detection of small delay faults, which are becoming a significant problem in ICs designed in DSM technologies and if and can cause reliability hazards if they go undetected [58], [98], [121].
- Determination of magnitude of violation to enable systematic silicon debug.

Gross delay faults can be detected using AC scan tests [85]. However, a large number of the previously mentioned sources of delay faults, like resistive vias and contacts and weak opens are very difficult to detect. Although most of these defects cause an increase in the delay of the path which they lie on, existing delay fault testing methodologies are not very effective in detecting them because of gross margins used in AC scan techniques [58], [121]. These form a class of delay faults, known as small delay faults [97], [101]. For example, it has been reported that the resistance of a via needs to increase from a nominal value of few Ω to tens or hundreds of kΩs to enable detection using at-speed testing. Current-based testing methods, like IDDQ [82], which were used to detect such defects earlier, are not very effective in chips designed in DSM technologies due to the presence of high leakage current in such chips [115]. Techniques like burn-in are expensive, can damage chips, and can have test escapes [66]. Parts that escape the manufacturing test flow can fail in the field and lead to reliability hazards and customer returns [6], [58], [98], [121]. Variations in process and environmental conditions can also cause small delay faults. Detection of small increments in propagation delay of critical paths is also essential for binning of parts into different frequencies [101], and efficient characterization of parts under different operating conditions, like the multi-parameter characterization schemes presented in [58], [121].

Very low voltage delay fault testing [22] has been proposed as a method for detection of small delay faults. However, the drawback for low voltage delay fault testing was pointed out in [16], wherein it was recognized that the variability of a circuit increase significantly as voltage level goes down, thus putting the validity of such a test methodology into question. Additionally, this methodology is not very effective for silicon debug, since it is difficult to obtain the magnitude of violation. Another technique for delay fault testing was proposed in [140], wherein the output of PUTs are connected to their input to form an oscillator, and their oscillation frequency measured by applying patterns that cause these circuits to oscillate. However, such a scheme has a severe drawback in that it requires modification of paths in the IC to enable converting them into oscillators during test mode, which may not be feasible in complex ICs.

In [83] and [103], techniques to detect small delay faults was presented, wherein the sample time is shifted to determine the size of delay faults. In a sequential circuit, such a technique would require schmooing of the system clock to determine the clock period at which various PUTs fail. However, the resolution of such a scheme is limited by the granularity with which the clock period can be changed, and the size of delay fault that can be detected is limited by the maximum achievable clock frequency in a chip.

Coupling of a clock schmooing mechanism with AC scan testing leads to another problem, namely, increase in test time. Use of AC scan in conjunction with clock schmooing is an iterative process, wherein, for each PUT, every test pattern has to be scanned in repeatedly and applied to the PUT at different frequencies. It has been reported for a microprocessors that scan chains are about 1800-2000 stages long [139], and for scan chains of such length (or longer), this process can be a serious bottleneck to test time. This is further exacerbated by the fact that a chip has to be tested under different environmental conditions like voltage and temperature. The resolution of delay test and debug is also limited in this methodology. Another drawback of techniques based on shifting of the sampling interval is that the clock edges need to be controlled accurately, as stated in [51]. This may not be a simple process in modern ICs, where clock speeds are in the multi-GHz range.

In [51], techniques were proposed to detect gross and small delay faults based on continuous checking of the output of various PUTs during delay test. However, the circuit proposed to detect gross delay faults was highly susceptible to noise, which could invalidate the test [104]. In order to detect small delay faults, the authors in [51] suggest analog integration of the output waveform, which is continuously monitored. This requires complex circuitry, namely, an on-chip integrator with multiple circuits with different RC constants connected to the output of a PUT [51]. In [50] a sampling circuit is proposed based on two clocks which are 180° phase shifted with respect to each other. However, the method requires insertion of two transistors in every gate which are controlled by the two clocks, and any skew between these clocks can cause the method to fail [50]. Additionally, the scheme requires careful design wherein the capacitors at different nodes of the gate have a certain ratio, and it also requires sampling of not only the outputs, but also the internal nodes of gates. Also, this sampling circuit can detect only delay faults where the delayed transition on a path occurs after the sampling time [104]. These drawbacks reduce the practicality of using this scheme in ICs. In [104], techniques were presented to continuously monitor the output of various PUTs. For detecting small delay faults this scheme relies on shifting of sampling time (the drawbacks of which have already been discussed earlier), and it requires insertion of one or more transistors inside the gates and also monitoring of internal nodes of the gates.

In [97], [101] a sampling technique based on *scanout chains* has been proposed for efficient delay test and debug. *Scanout chains* are small length scan chains that sample selected internal nodes of the chip, and capture response of the chip when functional tests are applied to them [101]. However, the drawback of this technique is that it relies on the ability to shift sampling time in order to detect small delay faults as well as determine magnitude of violation for silicon debug. The drawbacks of a scheme that depends on the ability to shift sampling time, i.e., vary clock period, has been discussed earlier.

Instead of just sampling the PUTs in a chip, as is done for schemes like those in [97], [101], a sampling mechanism coupled with delay measurement, i.e., on-chip delay measurement of critical paths in an IC [35], [38], [39], [60], [111], [127], can be considered as an attractive alternative to the above mechanisms for detection of small delay faults as well as for silicon debug. This is because, the resolution is determined by the delay measurement circuit, independent of the clock period. Consequently, such schemes can detect small increments in delay of the PUT, as well as determine the magnitude of this increment. Delay measurement of critical paths can also be used for determining available slack, maximum operating frequency and speed binning [8], [101].

In [60], a technique for on-chip delay measurement based timing characterization is presented, wherein a complex delay measurement module consisting of comparators and registers and analog components like integrators are used. The scheme converts the delay of a PUT into a voltage value, and this voltage is compared with a reference voltage. Thus, in addition to the drawback of using analog circuitry, the scheme also requires computation of different reference voltage values that correspond to different values of propagation delay of a PUT. A similar technique is explored in [111], wherein the voltage value on a capacitor depends on the path delay. Both these schemes are basically delay sensing schemes, and require computation of a reference voltage that corresponds to a particular delay. Another drawback of a scheme based on charge stored in a capacitor is that leakage of this charge can cause errors during test. However, an interesting result that was stated in [111] is that an on-chip sampling scheme coupled with delay sensing/measurement leads to an improvement in transition fault coverage, suggesting that, even some gross delay defects that can go undetected using existing delay test strategies can be detected using intermediate sampling points. In [127], an on-chip delay measurement based timing characterization technique is presented, wherein the propagation delay of a path is converted into a pulse using an XOR gate. Subsequently, this pulse at the output of the XOR gate is sampled using what the author call an "asynchronous clock", and at every sample edge of the clock, if the output of the XOR gate is at logic value 1, a counter is incremented. The value of this counter indicates the delay of the path. There are two main drawbacks of this scheme. First, the accuracy of delay measurement depends on the frequency of the sampling clock. If the resolution desired is $\frac{1}{5}^{th}$ of the period of the system clock, the sampling clock has to have a frequency approximately 10 times (5 sampling edges) that of the system clock. Such a constraint will significantly increase the complexity of timing characterization using this scheme, and make it ineffective in modern ICs, where clocks are in the GHz range. The second drawback of the scheme presented in [127] is that is requires pre-computation of a reference counter value against which the value of the counter used during delay measurement is compared. In order to use the scheme for timing characterization, a large number of reference counter values would need to be established, which could impact the efficiency of the scheme.

1.3 Variability and Parametric Reliability

Variations in process parameters signifcantly affect the behavior of DSM ICs. Such variations can be categorized into inter-die and intra-die process variations. Inter-die variations affect parameters in different regions of a die uniformly, and were the dominant source of variations in earlier technologies. However, with gate lengths scaling down to dimensions that are well below the wavelength of light used in the optical lithography process, intra-die variations have begun to play a significant impact on the performance of chips [17]. Lithography and scaling induced variations include factors like variation in threshold voltage, channel length and oxide thickness, which have been shown to create about 30% variation in chip frequency [14], and factors like line edge roughness, which can cause a variation in line width to the order of 5nm [40]. Additionally, environmental factors like drop in supply voltage and temperature variations can also have a significant impact of behavior of DSM ICs [14], [47].

Such variations can often lead to an otherwise fault free IC being marked defective during the test process, which in turn leads to significant yield loss. Traditionally, designers addressed performance fluctuations due to variations by keeping margins in their designs to account for them. This kind of design methodology is not effective in DSM technologies because the margins required for such a design would need to account for large amount of variability (the chip frequency can vary by about 30% as mentioned above), which would make these margins significant. This in turn would result in the chip being operated at a much lower speed than what is achievable. Such over-design would significantly impact the economic viability of the chip as well as scaling in general, by leading to increased design efforts to meet margins, as well as possible delays in time-to-market [47], [95]. Statistical analysis methods have been attempted to analyze the performance of the circuit for different combinations of process parameters [47], [95]. Adaptive design schemes, that enable reconfiguration of a chip to enable operation in face of process and environmental variations have been suggested as another potential solution to this yield loss problem, and actually complements statistical analysis [3], [14], [69], [15], [32]. Two kinds of adaptive design mechanisms have been reported in the literature, the first kind being an iterative post-silicon tuning based adaptation, e.g. [32], and the second kind being a non-iterative adaptation on mathematical/statistical analysis [69]. The latter has its advantages in being faster, and can utilize the results of statistical analysis, which gives parameters at different operating points.

Adaptive design schemes have two main components, namely, process variation sensing mechanisms, and process compensation mechanisms. Process variation sensing schemes determine the process corner in which a chip or a region of the chip is working, and in turn drive the process compensation mechanisms which could be multiple voltage settings, redundant components, different body-bias voltages [15]. Adaptive design techniques enable a die to work in the process corner in which it lies, or in face of environmental variatons, thus ensuring parametric reliability.

1.4 Summary of Contribution

This work explores techniques to enhance controllability and observability of ICs for timing characterization to detect and debug timing violations, and timing oriented adaptive design techniques to compensate for process variations, that ensure that adaptation does not cause a chip to violate minimum acceptable timing specifications. A low Design for Test (DFT) technique called *Tri-Scan* has been developed, that enhances controllability for delay fault testing of scan based sequential static CMOS circuits, to a level equivalent to fully combinational static CMOS circuits with complete access to all inputs. As mentioned earlier, delay test and debug of dynamic circuits have different requirements from that of static CMOS circuits, and to this end, Design for Test and Debug (DFTD) techniques for enhancing controllability of dynamic circuit have been developed, that enhance controllability of both dynamic circuits with full access, as well as scan based dynamic circuits. To overcome the aforementioned observability limitations for timing characterization of ICs, techniques to measure delay of a PUT for on-chip delay measurement based timing characterization are explored. These techniques enable detection of small increments in propagation delay of a PUT for testing of small delay faults, and also enable determination of magnitude of violation for efficient debug of delay faults. These delay measurement techniques are based on the principle of time-to-digital conversion using digital delay lines. Although such a technique was alluded to in [60], no details were presented regarding the same. In order to tackle process variations, adaptive design techniques for ensuring performance-optimized compensation for variability are explored. Such techniques ensure that adaptation does not cause the IC to violate minimum acceptable timing specifications. Techniques that can enable rectification of process variation related timing failures are also proposed. Both process perturbation sensing schemes as well as compensation mechanisms for performance-optimized adaptive design that enable parametric reliability are explored in this work.

1.5 Outline of Dissertation

In Chapter 2 the *Tri-Scan* scheme for enhancing controllability of static CMOS circuits is presented and techniques for enhancing controllability of dynamic circuits are presented in Chapter 3. In Chapter 4, an overview of existing delay line based time-to-digital conversion techniques are presented. Subsequently, three new on-chip delay measurement schemes and corresponding timing characterization methodologies are presented. In Chapter 5 techniques that provide performance-optimized adaptation to variations are presented, and this dissertation is concluded in Chapter 6.

Chapter 2

Controllability of Static CMOS Circuits for Timing Characterization

In this chapter, a DFT technique that enables pattern application for timing characterization in scan based static CMOS circuits is presented. The difficulty of delay fault testing in scan based designs was presented in Chapter 1, along with the existing techniques to overcome these problems, namely, *enhanced scan* [42], [81], *skewed-load transition test* [117], [118] and *broad-side delay test* [119], [28]. *Enhanced scan* is very expensive in terms of hardware overhead, whereas *skewed-load transition test* and *broad-side delay test* techniques increase complexity of test generation and provide limited coverage, in addition to other drawbacks discussed in Chapter 1. In [28], it was stated that a combination of *skewed-load transition test* and *broad side delay test* schemes (along with scan chain reordering) still left certain faults undetected, which could only be detected by *enhanced scan*.

Clearly, the level of accessibility provided by *enhanced scan* is tremendous, and in this chapter, a scheme is presented that enables the same level of accessibility for static CMOS circuits as is possible with *enhanced scan*, but with much lower overhead. The proposed scheme, called *Tri-Scan* (this technique was published in [36]), exploits the state holding property of static CMOS circuits, and reduces the problem of delay fault testing in scan based static CMOS circuits to that of delay fault testing in combinational circuits with complete access to all primary inputs.

2.1 Tri-Scan Scheme

Consider a scan chain in between two blocks of combinational logic, i.e., as capture chain of *Comb. Logic 1* and as launch chain of *Comb. Logic 2*, as shown in Figure 2.1 [28]. In a practical circuit, in order for the scan flip-flops to be able to drive the combinational logic (*Comb. Logic 2*), there would need to be a buffer (inverter) at the output of (and integrated into) the flip-flop. In Figure 2.1, this buffer is shown explicitly at the output of the flip-flop, as a stage following the latching logic in the flip-flop.

The implementation of the inverter buffer using static CMOS circuit family is shown in Figure 2.2a. Now if we add two transistors *M2* and *M3* to this inverter, control them using a signal *tri* as shown in Figure 2.2b, then this circuit becomes a tristate inverter. At the gate/logic level, a tristate inverter has its output in one of the 3 possible states, i.e., logic value 0, logic value 1 or logic value Z (high impedance state) at any given time. This is basically because, if the *tri* signal is de-asserted, then the tristate inverter acts like a regular CMOS inverter, by producing an output logic value opposite to the

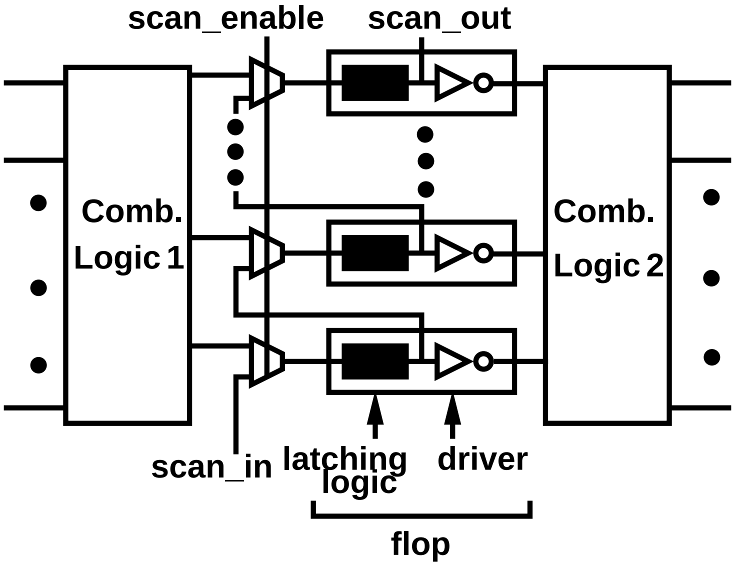


Figure 2.1: Two stages of combinational logic with scan in between, and driver shown explicitly

one at its input. However, when the *tri* signal is asserted, the output is said to be at high-impedance, denoted as Z, because the output node is cut-off from both the V_{dd} and ground due to transistors *M2* and *M3* being off (so no low impedance path exists from the output to the supply rails). However, CMOS logic has the property of holding its last defined state, and hence, logic value Z is nothing but the last held state of the output. This state is preserved in the capacitor shown explicitly at the *out* node of the tristate buffer in Figure 2.2b, and this capacitor is usually the fan-in capacitance of the stages of logic and interconnect that follow the tristate buffer.

The *Tri-Scan* scheme utilizes this fundamental state holding property of CMOS circuit family. The basic scheme is shown in Figure 2.3. The output

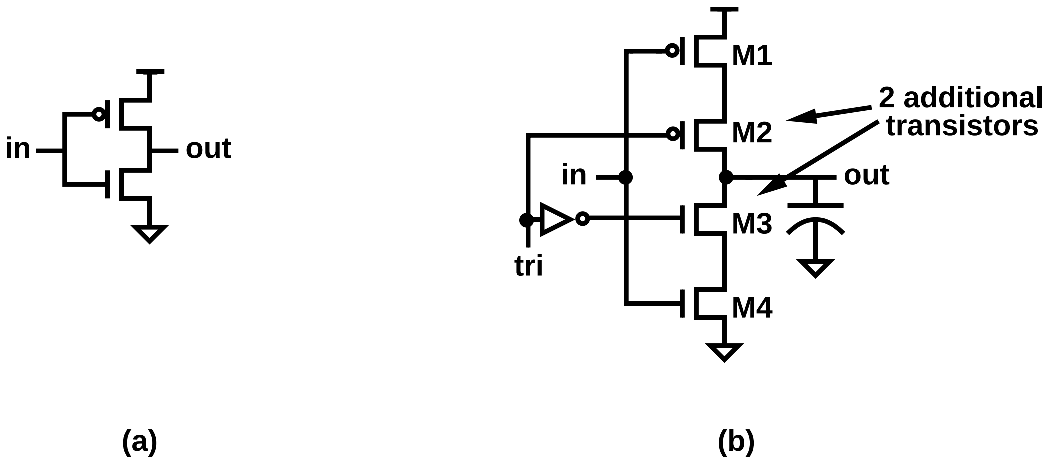


Figure 2.2: a) Regular inverter b) Tri-state inverter

inverter of every flip-flop is replaced with a tristate inverter. During normal mode of operation, the tristate buffer is made transparent by de-asserting the *tri* signal. Consequently, it acts like a regular static CMOS inverter. Even during scan operation for regular test (i.e., stuck-at testing), the tristate buffer can be kept transparent, and the flip-flops would work like a regular scan flip-flop with a driver buffer feeding into the corresponding combinational logic. During delay fault testing, the initialization vector V_1 can be scanned in by keeping the tristate buffer transparent. However, once V_1 has been applied to the CUT (*Comb. Logic 2* in Figure 2.3), then the *tri* signal is asserted before one begins to scan in the transition vector V_2. By asserting the *tri* signal, the output of the tristate buffer continues to hold the state it was in before assertion of *tri*, irrespective of the value at its input. The state which it holds is nothing but its output after application of V_1. Thus the CUT continues to be in the same state which it was in after application of V_1, because its input does not change in spite of multiple bit changes in the scan flip-flops when V_2 is being scanned in. In this manner, one achieves complete isolation between the scan path and the combinational logic for the duration of the time that V_2 is being scanned in. Once V_2 has been scanned in completely, the *tri* signal can be de-asserted, and V_2 can be applied to the CUT to cause the requisite transition for testing the delay fault. This gives the same level of accessibility for delay fault testing in scan-based sequential circuits as that of combinational circuits with complete access to all inputs.

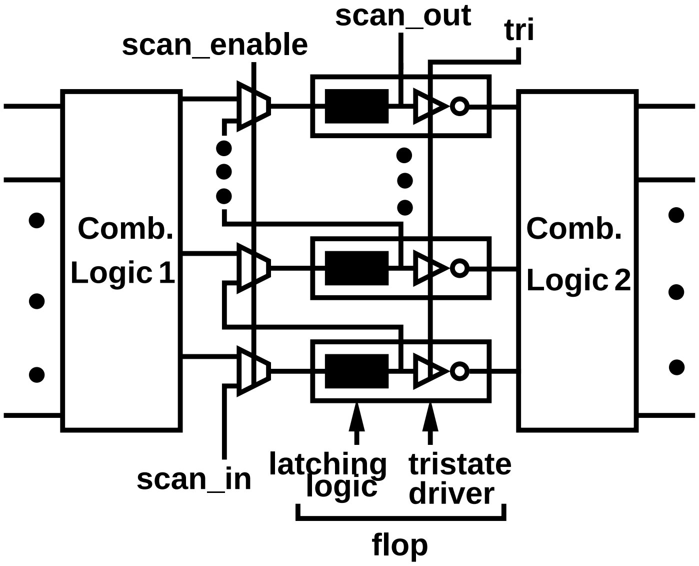


Figure 2.3: Tri-Scan scheme

In the scheme shown in Figure 2.3, there is a need for a separate *tri* signal. This will be expensive in terms of hardware, both because of the additional overhead of routing a separate signal all across the chip, and also due to the pin overhead associated with the need of a separate pin to control the *tri* signal externally. This additional overhead can be avoided by reusing the *scan_enable (SE)* signal for controlling the tristate buffer as shown in Figure 2.4. In such a scheme, whenever there is a scan operation taking place, (i.e., the SE signal is asserted), the output of the tristate inverter continues to hold the state it was in before the scan operation began. Hence there is complete isolation between the scan path and the combinational logic during any scan operation.

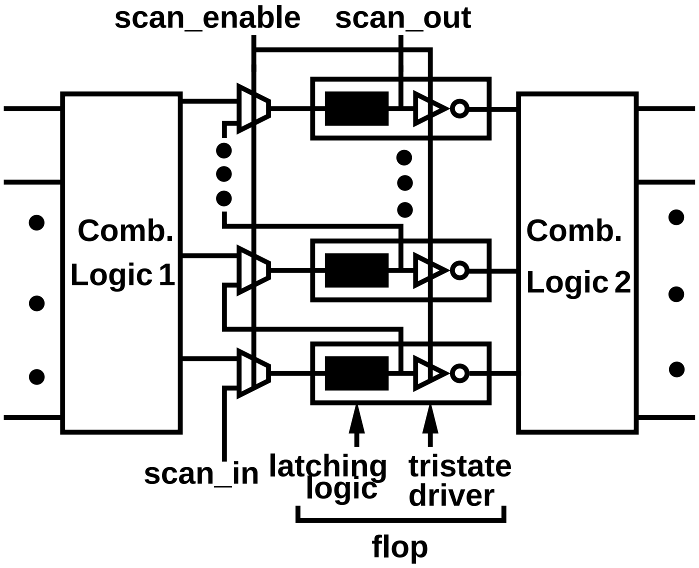


Figure 2.4: Tri-Scan scheme with scan enable being reused for controlling the tristate buffer

The tristate inverter in the *Tri-Scan* scheme also serves the dual purpose of reducing switching power of a processor during the scan operation.

Since the output of the scan flip-flops are isolated from the logic they drive during scan operation, there is no unwanted switching in the functional logic driven by the scan flip-flops when scan-in or scan-out operation takes place. Hence all the switching during scan operation happens in the scan flip-flops, while there is no switching activity in the combinational logic.

In the absence of DSM phenomena like leakage [91], the output of the tristate buffer in the *Tri-Scan* scheme can continue to hold its state when tristated, until the tristating control is de-asserted, and a value change at the input causes the output to switch. However, in DSM circuits the charge held at the output will discharge over time due to leakage. A successful practical implementation of this scheme requires that the duration for which the charge is held at the output of a tristated buffer be greater than the time required for scanning in a vector into a regular sized scan chain. This issue and techniques to address it are discussed in more detail later in this section.

The properties of *Tri-Scan* and its benefits over existing methods like *skewed-load transition test* and broad side delay test can be analyzed by extending the calculus for *skewed-load transition test*, because the former is reported to have better coverage than *broad side delay test* [119]. Boolean Difference based calculus [4], [29] was used for computing V_1 and V_2 in [117]. Boolean Difference is a technique that gives the condition under which the output of a circuit depends only on a particular input. In other words, it gives the logic values that all other inputs of a circuit need to have, so that any change in the input of interest reflects on the output. This condition is exploited for test to obtain patterns for testing of a particular fault. Unlike path tracing methods, Boolean Difference technique gives all possible patterns for testing a particular fault in a CUT [135]. However, it is not a commonly used technique for ATPG because it is computationally very expensive, and impractical for large circuits [135].

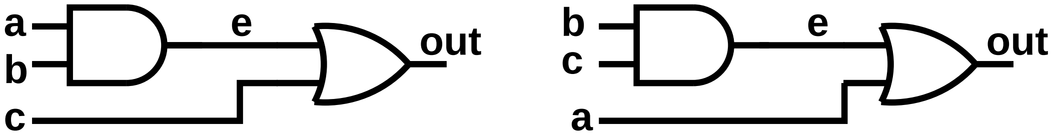
In a CUT where the fault to be detected is in a line f (which may be a primary input or internal line of the CUT), and the output of the CUT is *out*, Boolean Difference based test pattern generation requires using a combination of

1. Fault activation at the line f by creating a logic value at the line which is opposite to that created by the fault.
2. Propagation of the fault from line f to the output *out* by finding input values that enable $\frac{\partial out}{\partial f} = 1$.

The term $\frac{\partial out}{\partial f}$ in turn, is calculated as an XOR of

1. The output *out* as a function of inputs that make *out* true when f is at logic value 1.
2. The output *out* as a function of circuit inputs that make *out* true when f is at logic value 0.

For example, if there is a stuck-at 1 fault at f, then to activate this fault, f needs to be at logic value 0 (i.e., $\bar{f} = 1$), and to propagate the fault to the output, $\frac{\partial out}{\partial f} = 1$. Hence a pattern that detects the stuck-at 1 fault at f ϵ (solution set of $\bar{f}\frac{\partial out}{\partial f} = 1$) [135].



(a) (b)

Figure 2.5: Example circuits for comparing skewed-load transition test and Tri-Scan

Consider the circuit in Figure 2.5a. It is a simple circuit comprising a two input AND gate driving an input of a two input OR gate. This circuit was the one used in [117] to illustrate the salient features of *skewed-load transition test* and the same example is utilized here to demonstrate the salient features of *Tri-Scan* theoretically. Comparison is done with *skewed-load transition test*, because it requires assessment of a single cone of logic, just like *Tri-Scan*. The same example circuit as in [117] is utilized for this purpose to ensure comparison is done with *skewed-load transition test* using a circuit presented by the author who proposed the scheme.

In the circuit of Figure 2.5a, the signals a, b, and c are the inputs of the circuit, coming from 3 different scan flip-flops of the same scan chain, and *out* is the output of the circuit. The faults we consider are slow-to-rise (STR) and slow-to-fall (STF) faults on the signal e assuming the transition fault model [76]. In [117], a displaced function d(e) was suggested as a change in Boolean function of e due to a single bit shift in its inputs, to represent the shifting of V_1 needed in *skewed-load transition test* to derive V_2.

In case there is a STR fault on line e, then the vector V_2 is nothing but a test vector for stuck-at 0 fault on line e. This is obtained by setting the inputs such that 1) the fault at e is activated and 2) the effect of the fault at e is propagated to *out*, which is obtained by finding the solution set of $\frac{\partial out}{\partial e} = 1$. Hence, it can be said that

$$V_2 \; \epsilon \; solution\ set\ of\ e\frac{\partial out}{\partial e} = 1 \tag{2.1}$$

i.e., V_2 is obtained from the solution set of the equation for setting e to logic value 1, and propagating this effect to the output. Now in case of *skewed-load transition test*, there is another constraint on V_2, i.e., V_2 should be obtained by 1 bit shift over V_1, which in this case would have set e to logic value 0. Hence V_2 depends on the displaced function d(e) and the following equation represents this dependence [117]

$$V_2 \; \epsilon \; solution\ set\ of\ d(\bar{e}) = 1 \tag{2.2}$$

Combining equation 2.1 and equation 2.2, the final Boolean difference equation for V_2 can be arrived at, which is given in equation 2.3 [117].

$$V_2 \; \epsilon \; solution\ set\ of\ d(\bar{e})e\frac{\partial out}{\partial e} = 1 \tag{2.3}$$

Solving the Boolean Difference equation 2.3 will yield all possible combinations of V_2 that can test this fault using *skewed-load transition test*. The solution of this equation will also yield V_1, since V_2 is one shift away from V_1.

In case of *Tri-Scan*, V_2 depends only on the circuit topology and is independent of V_1. Hence V_1 and V_2 can be computed concurrently using equation 2.4. It is much less complex to solve these two separate equations rather than solving the single equation needed for *skewed-load transition test*. Hence it is evident that the complexity of test generation for *Tri-Scan* scheme is lower than that for *skewed-load transition test*, even though both require assessment of a single cone of logic, unlike *broad side delay test*.

$$\begin{aligned} V_1 \,\epsilon\, solution\ set\ of\ \bar{e}\frac{\partial out}{\partial e} &= 1 \\ V_2 \,\epsilon\, solution\ set\ of\ e\frac{\partial out}{\partial e} &= 1 \end{aligned} \tag{2.4}$$

The test set for a STF fault for *skewed-load transition test* and *Tri-Scan* can be obtained using equation 2.5 [117] and equation 2.6 respectively.

$$V_2 \,\epsilon\, solution\ set\ of\ d(e)\bar{e}\frac{\partial out}{\partial e} = 1 \tag{2.5}$$

$$\begin{aligned} V_1 \,\epsilon\, solution\ set\ of\ e\frac{\partial out}{\partial e} &= 1 \\ V_2 \,\epsilon\, solution\ set\ of\ \bar{e}\frac{\partial out}{\partial e} &= 1 \end{aligned} \tag{2.6}$$

Now let us consider and extend an example from [117] to project the benefits of *Tri-Scan* more clearly.

Example 1. Consider the circuit in Figure 2.5a [117]. The logic equations for various signals in the circuit are given below

$$e = ab,\ d(e) = bc,\ \bar{e} = \bar{ab} = \bar{a} + \bar{b}$$

$$d(\bar{e}) = \bar{b} + \bar{c},\ out = e + c$$

The Boolean Difference equation relating *out* to the line e where the faults to be detected exist, is obtained as follows.

$$\frac{\partial out}{\partial e} = out(e = 0) \oplus out(e = 1)$$

$$\frac{\partial out}{\partial e} = c \oplus 1$$

$$\frac{\partial out}{\partial e} = \bar{c}$$

Now consider a STF fault on line e.

If skewed-load transition test is used to detect this fault, then the patterns are obtained as follows [117]

$$d(e)\bar{e}\frac{\partial out}{\partial e} = bc(\bar{a} + \bar{b})\bar{c} = 0 \Rightarrow \neq 1$$

$$V_2(e_{STF}) = \phi$$

Hence an STF fault at line e cannot be tested using skewed-load transition test. However, if the Tri-Scan scheme is used, then the patterns are obtained as follows.

For obtaining V$_1$

$$e\frac{\partial out}{\partial e} = ab\bar{c}$$

and for obtaining V$_2$

$$\bar{e}\frac{\partial out}{\partial e} = (\bar{a} + \bar{b})\bar{c} = 1$$

Hence the STF fault at line e can be tested using Tri-Scan, and the patterns are as follows.

$$V_1(e_{STF}) = 110 \;\; and V_2(e_{STF}) = 0X0 \; or \; X00$$

where X represents don't care condition.

Now consider and STR fault on line e.

Using skewed-load transition test, the patterns are obtained as follows [117].

$$d(\bar{e})e\frac{\partial out}{\partial e} = (\bar{b} + \bar{c})ab\bar{c} = 1$$

$$V_2(e_{STR}) = 110, V_1(e_{STR}) = 10X,$$

For pattern application using Tri-Scan, the vectors are obtained as follows.

For V$_1$

$$\bar{e}\frac{\partial out}{\partial e} = (\bar{a} + \bar{b})\bar{c} = 1$$

and for V_2

$$e\frac{\partial out}{\partial e} = ab\bar{c}$$

$$V_1(e_{STR}) = X00 \; or \; 0X0 \;\; V_2(e_{STR}) = 110$$

From the above example, it can be concluded that the *Tri-Scan* can detect faults which *skewed-load transition test* cannot and gives higher delay fault coverage as compared to *skewed-load transition test*. The *Tri-Scan* scheme also reduces the complexity of test pattern generation since both patterns can be computed concurrently and independently and depend only on the circuit topology.

Now let us consider and extend another example from [117] to project some other benefits of *Tri-Scan*.

Example 2. Consider the circuit in Figure 2.5b [117]. It is the same as circuit in Figure 2.5a in terms of topology but its inputs are ordered in a different manner (for example, from scan chain reordering). The logic equations for various signals in the circuit are given below.

$$e = bc,\; d(e) = cf,\; \bar{e} = \bar{bc} = \bar{b} + \bar{c}$$

$$d(\bar{e}) = \bar{c} + \bar{f},\; out = e + a$$

Here f represents the bit that gets shifted out of the scan chain when single scan shift occurs to generate V_2 from V_1.

The Boolean Difference equation relating *out* to the line *e* where the faults to be detected exist is obtained as follows.

$$\frac{\partial out}{\partial e} = out(e=0) \oplus out(e=1)$$
$$\frac{\partial out}{\partial e} = a \oplus 1$$
$$\frac{\partial out}{\partial e} = \bar{a}$$

Consider a STF fault in line e. If skewed-load transition test is used for pattern application, then the test vectors are generated as follows [117].

$$d(e)\bar{e}\frac{\partial out}{\partial e} = cf(\bar{b}+\bar{c})\bar{a} = \bar{a}\bar{b}cf = 1$$

$$V_2(e_{STF}) = 001 \; V_1(e_{STF}) = 011$$

When Tri-Scan is used to detect the same fault then the vectors are obtained as follows.

For V_1

$$e\frac{\partial out}{\partial e} = bc\bar{a}$$

and for V_2

$$\bar{e}\frac{\partial out}{\partial e} = (\bar{b}+\bar{c})\bar{a} = 1$$

$$V_1(e_{STF}) = 011 \;\; V_2(e_{STF}) = 0X0 \; or \; 00X$$

So it can be seen that the corresponding bits in the test vector have the same values as in the earlier example if Tri-Scan is used for pattern application.

Now consider a STR fault at line e.

If skewed-load transition test is used for pattern application then the test vectors are generated as follows.

$$d(\bar{e})e\frac{\partial out}{\partial e} = (\bar{c} + \bar{f})bc\bar{a} = 1$$

$$V_2(e_{STR}) = 011, V_1(e_{STR}) = 110,$$

However, if Tri-Scan is the pattern application scheme then the test patterns are obtained as follows.

For V_1

$$\bar{e}\frac{\partial out}{\partial e} = (\bar{b} + \bar{c})\bar{a} = 1$$

For V_2

$$e\frac{\partial out}{\partial e} = bc\bar{a}$$

$$V_1(e_{STR}) = 0X0 \; or \; 00X \quad V_2(e_{STR}) = 011$$

Again it can be seen that the corresponding bits in the test vector have the same values as in the earlier example if Tri-Scan is used for pattern application.

Thus, scan chain reordering [28], [117] does not affect the delay fault coverage since the patterns depend only on the circuit topology. Hence expensive routing overheads associated with scan-chain reordering, can be avoided using *Tri-Scan* without paying the penalty of reduced coverage.

There are some drawbacks of *Tri-Scan* scheme. The state held on the output of the tristate inverter is critical, and the operation of the scheme depends entirely on the ability of the tristate inverter to hold its state for the entire duration of the scan. Simulation results presented in the next section show a negligible voltage drop in the amount of time required to scan in a vector into a regular sized scan chain in 0.18μm technology. However, in more advanced technologies like 90nm and 65nm, charge retention could become an issue. In such cases a keeper could be used to hold the state of the output of the tristate buffer. Keepers are relatively common in dynamic circuits [134] (the operation of such a keeper is also explained in chapter 5). The the concept of a keeper for retention of charge in an element similar to a tristate inverter has also been in explored in [88], where a keeper is used to retain state of a node for a C element. The size of the keeper is a function of the amount of leakage a given node experiences, and the environment in which the CUT is tested. This is because, transient effects like crosstalk can affect the charge on the node, and if such a noisy environment is anticipated, then the keeper would need to be strong enough to compensate for the change of state due to the transient effects.

The charge loss or gain in the output of the tristate buffer during delay test could also create a degree of optimism in the test. For example, if V_1 has created a logic value 1 at the output of the tristate buffer, then some of the charge can leak off and the delay of the tristate buffer at the input of a Path Under Test (PUT) will be comparatively less when V_2, which causes the output of this tristate buffer to go to logic value 0 is applied. However, the conjecture here is that this optimism will be compensated fully or partly by the additional delay induced due to the switching of the tristate buffer when there is a switch from scan to functional (i.e., capture) mode. This is because, during regular mode, the the two intermediate transistors in the tristate buffer, i.e., *M2* and *M3* in Figure 2.2b, are fully on. Consequently, there is a degree of pessimism in delay fault testing because of the switching of these transistors. Table 2.1 presented in the next section illustrates this pessimism, which can occur during delay test for the test circuits by comparing the performance overhead during regular mode with performance overhead during switching for several test circuits.

Use of *Tri-Scan* scheme makes the timing of the *scan_enable* signal critical, since the launch of the transition vector occurs upon a shift. This is even more so when the *scan_enable* signal is used to control the tristate buffer. In [2], it has been stated that pipelining of the *scan_enable* signal is a standard practice in the industry to tackle the issue of routing the *scan_enable* signal at high speeds, and this solution could be applied here too.

2.2 Simulation Results

The *Tri-Scan* scheme was designed in 0.18μm CMOS technology [19] and simulated using HSPICE, with 1.8V supply voltage, under nominal process conditions and at a temperature of 25°C. The correct operation of this scheme depends on the ability of the tristate inverter to hold the state at its output while a vector is being scanned in. Figure 2.6 shows the minimum voltage level at the tristated output for different sized loads, over a period of approximately 2000ns. This period corresponds to the length of the longest possible scan chain in a commercial processor [139], assuming scanning is done at 1 GHz. The capacitive load is created by placing a NMOS transistor whose gate terminal is connected to the output of the tristate inverter, and drain and source terminals are connected to ground.

We can see that even for capacitive loads equivalent to 300 μm transistor width, the voltage level does not discharge below V_{dd}-V_t. At certain points, the minimum voltage level is above the supply voltage level. This is because of the Miller coupling effect between the input and output [105]. When the tristate signal is asserted, a part of the charge gets coupled to the output of the tristate inverter, and hence causes the voltage level to go above V_{dd}. It should be noted that in this case Miller effect is proving to be useful, because the output is at the same logic value towards which the tristate signal is switching. However, when the output is at the opposite value, then the Miller effect will affect the operation of the scheme negatively. This factor has to be taken into consideration during design, and techniques like those suggested in [105] to overcome or mitigate the effects of this phenomenon may have to applied.

Figure 2.7a shows the value of the tristate inverter output across time, with a capacitive load equivalent to a 200μm transistor. The voltage drop

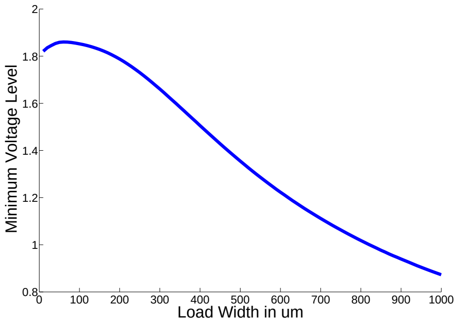
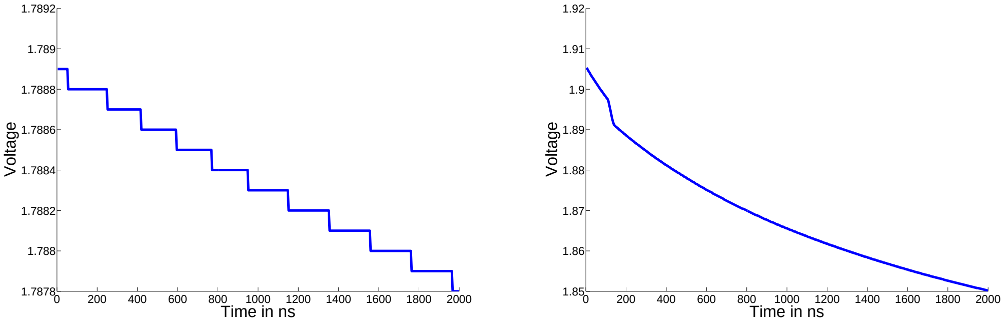


Figure 2.6: Minimum voltage value at output of tristate inverter with different sized loads

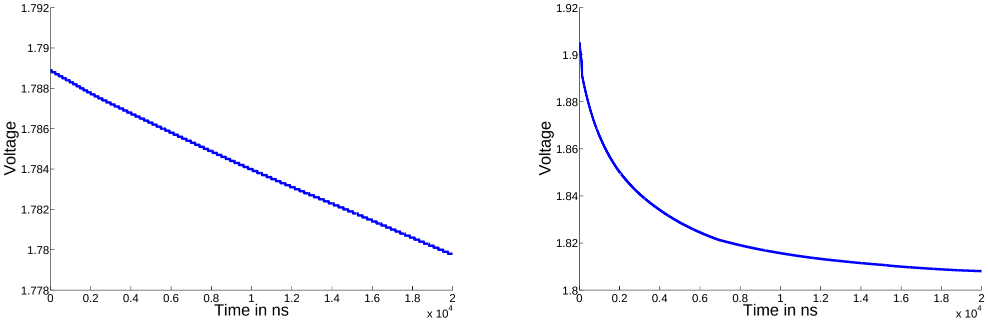
in approximately 2000ns is about 1mV (since the voltage drop is very small, it appears to be in the form of discrete steps in Figure 2.7a). This voltage drop corresponds to the voltage drop from the point when tristating is done (around 3ns), and the voltage level at that point is already about 11mV below the supply voltage. However, if the load is replaced by a capacitive load equivalent to a 36μm transistor, which approximately is a fanout-of-4 (fo4) load in our case, then the voltage level at the output of the tristate inverter actually increases by about 0.105V above the supply voltage, and then drops by about 55mV over 2000ns. However the final output at the end of 2000ns is still about 50mV above the supply level. The value of the voltage at the output of the tristate inverter with a 36μm load is shown in Figure 2.7b. Hence we can see, that if we use a larger load at the output of the tristate inverter, then the voltage drop over time is lower, because there is now a larger capacitor at the output to retain the charge, whereas, if we use a small load at the output of the tristate inverter, then the voltage drop is large over a given period of time, but the Miller effect has a greater impact on it, and in this case, helps in enhancing the voltage at the output. However, in both cases, the voltage drops are in the order of a few millivolts, and hence will not create a problem during testing.



(a) (b)

Figure 2.7: Voltage value at the output of the tristate inverter over 2000ns with a capacitive load equivalent to a) 200μm transistor width b) 36μm transistor width

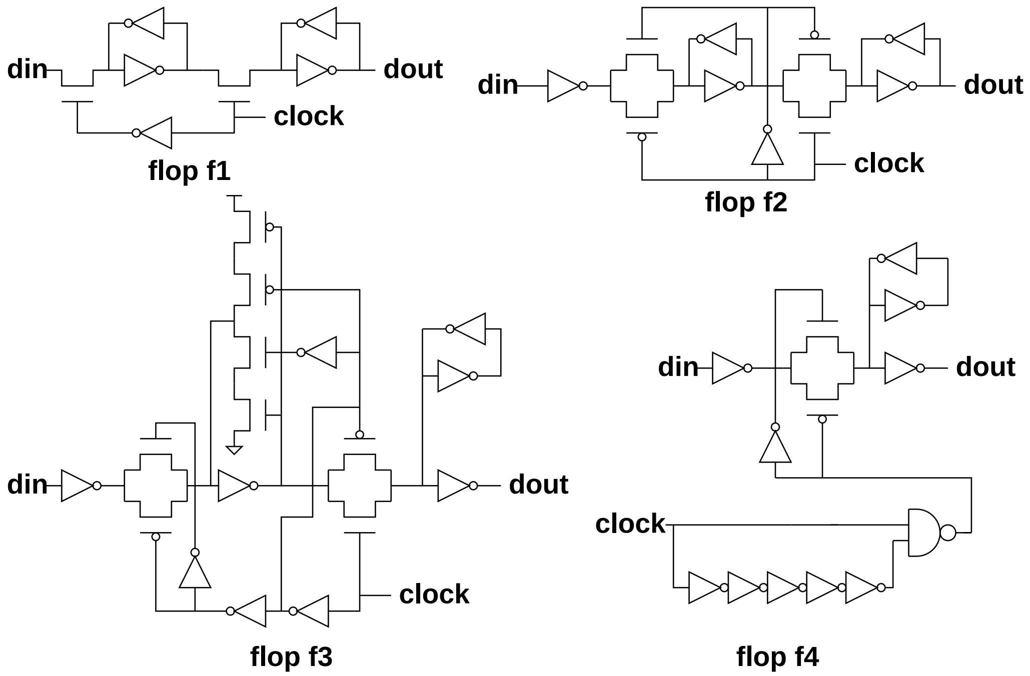
A crucial point here is that, because of the isolation and consequently lower power dissipation during scan operation enabled by the use of *Tri-Scan*, scan operation can be performed at higher than usual speeds, if the tester has the capability. However, if scanning is done at a slower speed, say 100MHz, then the output of the tristate inverter would have to hold its state for about 20000ns for a scan chain that is 2000 stages long. Figure 2.8a shows the voltage level at the output of the tristate inverter across 20000ns with a 200μm load. The voltage drop is about 9mV from the point at which the output is tristated, at which time the voltage level at the output is already 11mV below the supply voltage level. Figure 2.8b shows the voltage level at the output of the tristate inverter across 20000ns with a 36μm load. The voltage level at the output after tristating is done is 0.105V over the supply level due to Miller effect, and finally discharges by 97mV over 20000ns. hence the output is approximately at the level of the supply voltage after 20000ns.



(a) (b)

Figure 2.8: Voltage value at the output of the tristate inverter over 20000ns with a capacitive load equivalent to a) 200μm transistor width b) 36μm transistor width

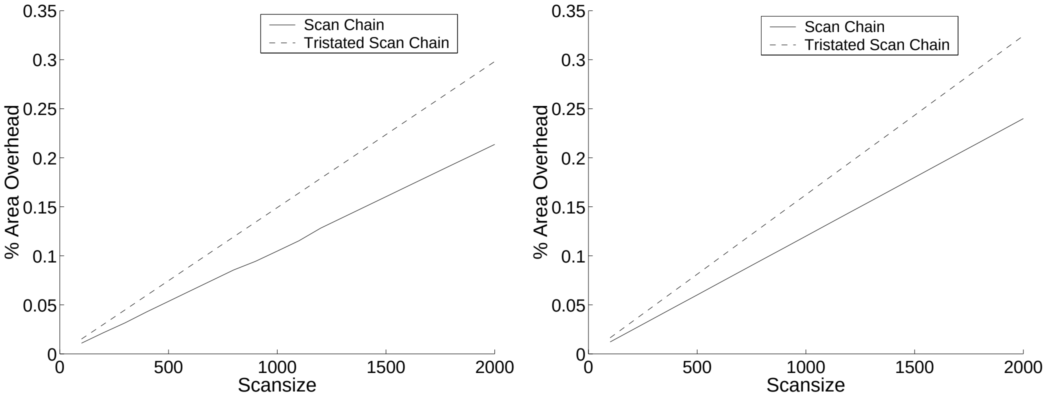
A representative commercial processor has been reported to have 48 scan chains [139], and has an area of 184 mm^2 [59]. A representative 1 GHz processor would have a switching power dissipation of about 500mW [134]. Relative area and switching power during scan operation of scan and *Tri-Scan* schemes in such a processor were compared. The computed area of both schemes were the total pre-layout transistor area (i.e., transistor width multiplied by length). This area was computed for different length (100-2000 stage) scan and *Tri-Scan* chains using four different flip-flops shown in Figure 2.9



flop f1 flop f2 flop f3 flop f4

Figure 2.9: Flip-flops used for scan design

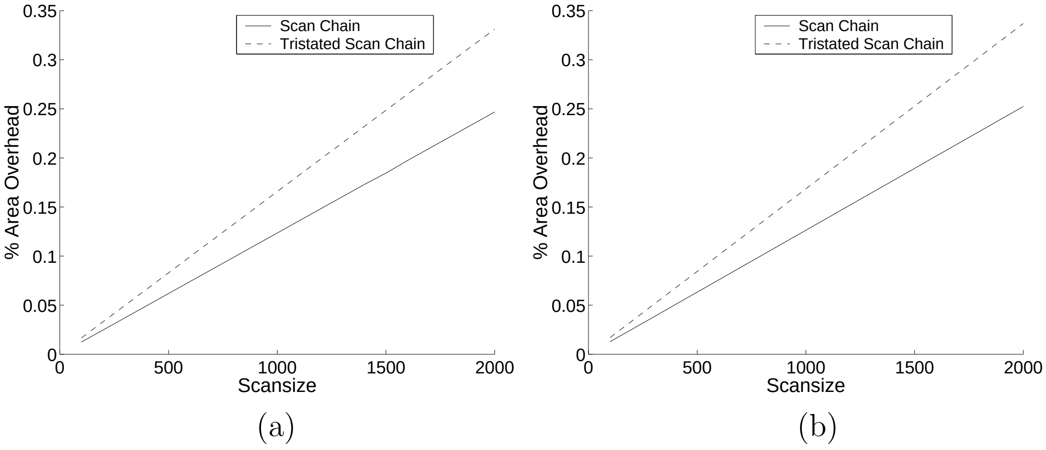
[134]. To obtain the area overhead, this computed area for different length scan and *Tri-Scan* chains was multiplied by 48 (i.e., the number of scan chains reported in the aforementioned processor [139]) and normalized with respect to the area of the aforementioned processor and expressed as a percentage. The idea of this analysis is to provide a comparative estimate of area overhead of regular scan and *Tri-Scan* schemes rather than providing absolute overhead numbers. Figures 2.10 and 2.11 show the estimates of relative area overhead of scan and *Tri-Scan* schemes with 48 scan chains of lengths ranging from 100 to 2000 stages using the four different flip-flops, and using the aforementioned assumptions and analysis. To obtain power dissipation results, power



(a) (b)

Figure 2.10: Area overhead of 48 scan and tri-scan chains of different sizes using a) flip-flop f1 b) flip-flop f2

dissipation of smaller length scan and *Tri-Scan* chains during scan operation was obtained using HSPICE simulations. Based on this, the power dissipation during scan operation for different sized scan chains and *Tri-Scan* chains (100-2000) was calculated. This power was normalized with respect to the power rating mentioned above, and multiplied by 48 and expressed as a percentage for both scan and *Tri-Scan* schemes. The idea of this simulation and analysis is to provide a comparative estimate of power dissipation during scan operation for regular scan and *Tri-Scan* schemes. Figures 2.12 and 2.13 show the estimates of relative power dissipation during scan operation at 25°C, under nominal process conditions with 1.8V supply voltage for regular scan and *Tri-Scan* schemes with 48 scan chains of lengths ranging from 100 to 2000 stages using the four different flip-flops and the aforementioned assumptions and analysis. We can see that the power during scan operation due *Tri-Scan*

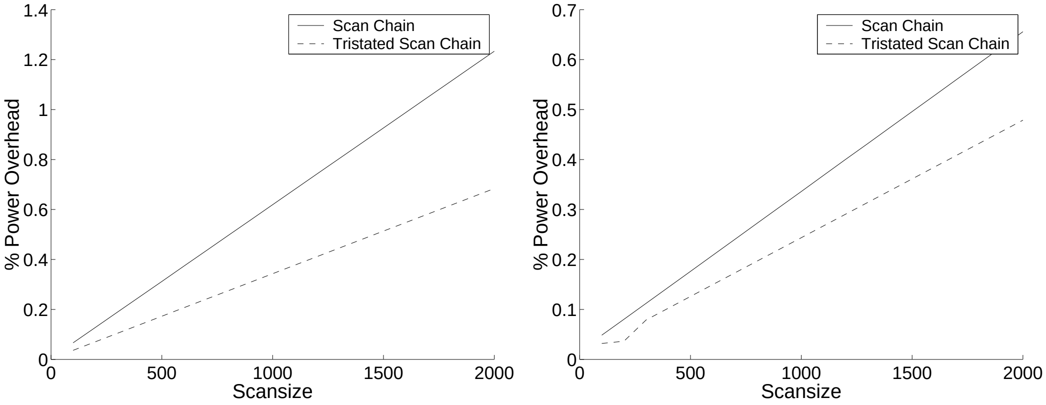


(a) (b)

Figure 2.11: Area overhead of 48 scan and tri-scan chains of different sizes using a) flip-flop f3 b) flip-flop f4

is significantly lower than that induced by a regular scan scheme. In the best case, the *Tri-Scan* scheme implemented using flip-flop f1 gives nearly a 2X (about 1.8X) improvement in power.

Figure 2.14a and 2.14b show the estimates of relative global area and power overhead respectively of 48 worst case length (in our case 2000 stages) [139] scan and *Tri-Scan* chains for the 4 different flip-flops, for a processor having an area of 184mm^2 [59] and power 0.5W at 25°C and 1 GHz operation [134], and using the other assumptions and analysis as explained above for computing relative area and power. The numbers 1-4 in the x-axis of both figures represent flip-flop f1-flop f4, respectively. There is a significant reduction in estimated power dissipation using *Tri-Scan* scheme, for different flip-flops. In case of flip-flop f1 the reduction is 1.8X, for flip-flop f2 it is 1.4X, for flip-

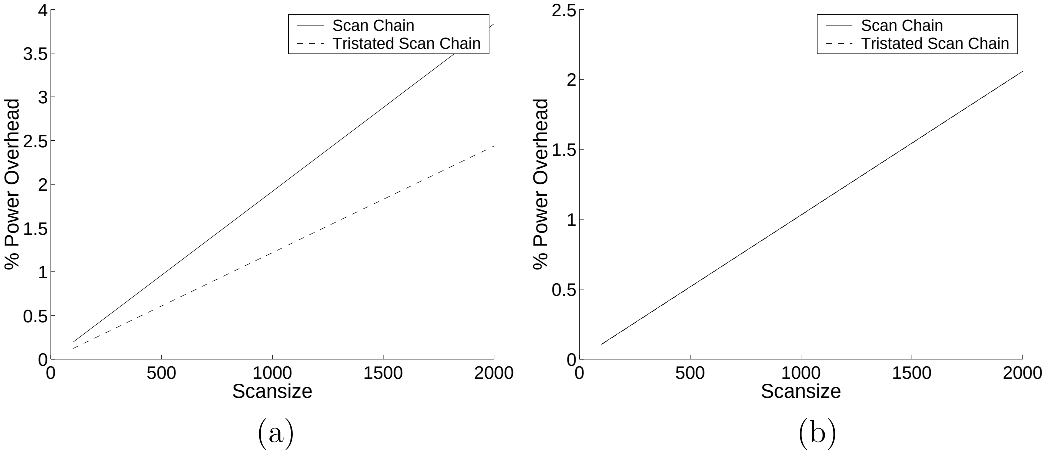


(a) (b)

Figure 2.12: Power dissipation during scan operation of 48 scan and tri-scan chains of different sizes using a) flip-flop f1 b) flip-flopf2

flop f3 it is 1.6X and for flip-flop f4 it is 1.005X. Hence we can see that if the switching power in the flip-flop dominates switching power in the combinational logic, then *Tri-Scan* will not give significant benefits. The additional area overhead induced by *Tri-Scan* scheme over a regular scan scheme with respect to area overhead of a processor ranges from 0.08% for flip-flop f1 to 0.09% for flip-flop f3.

Replacing a regular inverter with a tristate inverter in a path will lead to some performance overhead in those paths. A simulation experiment was designed to determine this overhead on non-inverting versions of top critical paths of different sized Wallace [137] and Dadda [31] multiplier circuits as well as ISCAS 85 Benchmark and 74 series [57] circuits (extracted using a commercial static timing analysis tool [128]). An inverter buffer was placed



(a) (b)

Figure 2.13: Power dissipation during scan operation of 48 scan and tri-scan chains of different sizes using a) flip-flop f3 b) flip-flopf4

before the critical path, which represents the output inverter buffer of a flip-flop and another inverter was added at the output of the paths to maintain polarity. The delay of this path was measured using HSPICE simulation, and subsequently the inverter buffer at the input was replaced with a tristate inverter buffer, and the delay measured using HSPICE simulation under two situations, 1) the tristate inverter is transparent throughout to obtain delay of the path in regular mode and 2) the tristate inverter becomes transparent simultaneously with the transition at the input of the path to obtain delay of the path during switching from scan to regular mode of operation. The additional delay over that of the corresponding path with regular inverter buffer incurred in the above two cases was calculated as a percentage of the delay of the corresponding path with regular inverter buffer. The results of this

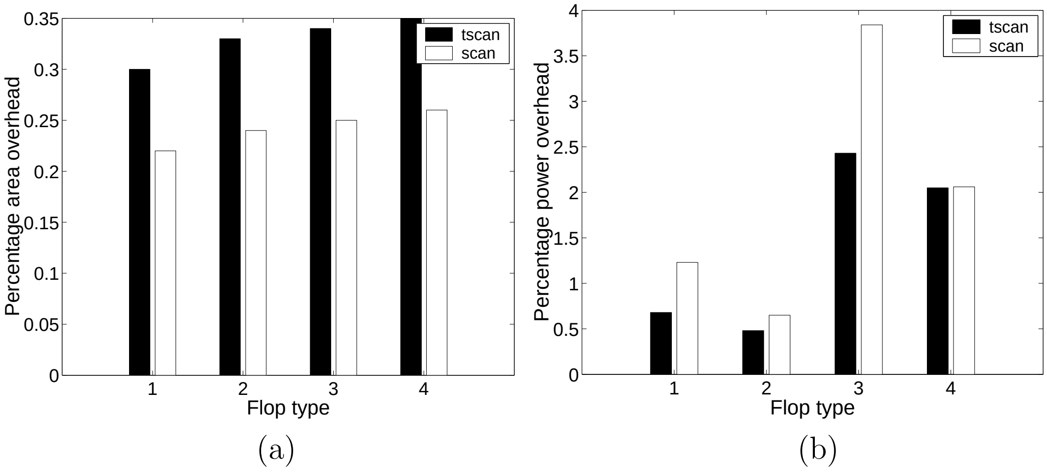


Figure 2.14: a) Relative area overhead of scan and tri-scan in a chip b) Relative power overhead of scan and tri-scan in a chip during scan operation

simulation experiment are shown in Table 2.1. The worst case performance overhead would be when the tristate inverter is switching, which is basically when scan flip-flops in the chip move from scan mode to regular mode. This is shown in column 2 and 3 of Table 2.1, and can be up to 2.9% for the test circuit considered. This overhead does not take into consideration overhead due to routing of *scan_enable*. This overhead basically manifests itself as a degree of pessimism during delay test. There will also be performance overhead in the regular mode, because the drive capability of the tristate inverter is reduced. This is shown in columns 4 and 5 of Table 2.1, and can be up to 1.8% for the test circuits considered. This overhead could be mitigated by appropriate sizing of the transistors, albeit at the expense of area and power.

In [132] the transition fault coverage numbers using standard scan for

Test Circuit	Performance Overhead while Switching (ps)	% Overhead	Regular Mode Performance Overhead (ps)	% Overhead
4 bit Wallace	30	2.7%	18	1.6%
4 bit Dadda	28	2.9%	14	1.4%
8 bit Wallace	36	1.7%	28	1.4%
8 bit Dadda	33	1.9%	31	1.8%
16 bit Wallace	30	0.88%	10	0.29%
16 bit Dadda	43	1.4%	36	1.2%
32 bit Wallace	25	0.36%	23	0.33%
32 bit Dadda	39	0.73%	27	0.51%
c432	21	0.54%	12	0.31%
c499	16	1.05%	4	0.26%
74181	33	2.9%	19	1.7%

Table 2.1: Performance overhead of Tri-Scan scheme

ISCAS 89 benchmarks were given. Most of the benchmarks had fault coverage in the range of 98%-99%. Therefore, in order to demonstrate the benefits of *Tri-Scan* Wallace [137] and Dadda [31] multipliers of different sizes were selected for comparing the transition fault coverage obtained using complete accessibility, as available with a scheme like *Tri-Scan* and transition fault coverage obtained using a standard scan scheme. Completely combinational versions of these multipliers were used and scan chains were placed at their primary input and outputs. Coverage results were obtained using a commercial tool. Table 2.2 shows experimental results of fault coverage and the number of patterns that were simulated for this experiment, for multipliers of various sizes with standard scan and *Tri-Scan*. The columns for *Tri-Scan* in Table 2.2

Test Circuit	Total number of Possible Faults	Fault Coverage		Number of Simulated Patterns	
		Standard Scan	Tri-Scan	Standard Scan	Tri-Scan
4 bit Wallace	1044	68.4%	100%	320	192
4 bit Dadda	1044	68.4%	100%	384	192
8 bit Wallace	4130	83%	100%	480	320
8 bit Dadda	3994	84.6%	100%	608	320
16 bit Wallace	16432	91.2%	100%	576	384
16 bit Dadda	15658	92.3%	100%	640	352
32 bit Wallace	64736	95.3%	100%	896	512
32 bit Dadda	62026	96.1%	100%	864	448

Table 2.2: Fault coverage of regular scan and Tri-Scan schemes

pertain to experimental results obtained by enabling complete accessibility to all inputs of the circuit (i.e., no scan chains). This was done because the commercial tool would not understand the tristating logic. It can be seen that having complete access to all inputs gives a higher coverage for all the circuits (listed under the columns for coverage of *Tri-Scan*). In case of larger circuits, the increase in coverage obtainable using *Tri-Scan* was smaller. This is because the total number of undetected faults do not increase proportionately with the increase in the total number of possible faults, as the size of multipliers increase.

2.3 Summary

There are severe limitations in the controllability of complex ICs designed in DSM technologies for delay fault testing. Existing scan based testing techniques either increase the complexity of test generation or have very high overhead. In this chapter a simple low overhead solution is presented that enables complete accessibility, and reduces the complexity of test generation for scan based sequential static CMOS designs to that of combinational designs with complete access to all inputs. The use of such a technique enhances controllability and delay fault coverage, and also leads to reduction is switching power during scan operation.

The technique presented in this chapter was published in [36], and the material in this chapter is based on the same paper. The multipliers used as test circuits were designed by Antony Sebastine and Whitney J. Townsend.

Chapter 3

Controllability of Dynamic Circuits for Timing Characterization

The problem with delay test and debug of dynamic circuits with complete access was explained in Chapter 1, and so were the drawbacks of the existing solution, wherein the *precharge* phase is used as the initialization vector (V_1). In addition to this, dynamic circuit face the same problems as static CMOS circuits when they are used in sequential designs with scan chains, and the drawbacks of existing solutions for this problem were also presented in Chapter 1. In the last chapter, a scheme for enabling two pattern application for timing characterization of static CMOS circuits was presented. The scheme, called *Tri-Scan*, is an effective solution for static CMOS circuits, but may not always be suitable for dynamic circuits, since it involves insertion of static CMOS components. In this chapter, two sets of Design for Test and Debug (DFTD) schemes, tailored towards dynamic circuits are presented. The first set, called *Precharge Control Schemes*, enable two pattern application for delay test and debug of dynamic circuits with complete access to all inputs, as easily as in corresponding static CMOS circuits. The second set of schemes are targeted towards scan-based dynamic circuits, and need to be used in conjunction with *Precharge Control Schemes*. Together, these schemes reduce the problem of delay test and debug of dynamic circuits in scan-based sequential designs to that of delay test and debug of static CMOS circuits with complete access to all inputs. This would also enable higher compatibility of dynamic circuits with existing tools for testing static CMOS circuits.

The rest of this chapter is organized as follows. In Section 3.1, a brief overview of some dynamic circuit families is presented. The *Precharge Control Schemes* are presented in Section 3.2, and DFTD techniques for enabling pattern application in scan-based dynamic circuits are presented in Section 3.3. In Section 3.4, simulation results on a dynamic 4-to-2 Carry Save Adder [34] using Limited Switch Dynamic Logic circuit family [11], [34], [90], [99] are presented, and a brief summary is presented in Section 3.5.

3.1 Dynamic Circuit Families

In this section an overview of of some dynamic circuit families is presented. Only those circuit families that are referred to in this work are covered in this section, and it is not a comprehensive reference of dynamic circuit families. Interested readers can look at [12], [105] and other excellent references for more information.

The fundamental operating principle of dynamic circuit families is that clocking is an integral part of the combinational logic, and hence is not limited to sequential circuit elements, as is the case in static CMOS circuits. A common dynamic circuit family is the Domino circuit family, an example of which is shown in Figure 3.1 [134]. Domino circuits are a clocked circuit family, having two phases, namely, *precharge* and *evaluate*. During the *precharge* phase, the *precharge transistor* is switched on, and the dynamic node is pulled up, as shown in Figure 3.1a. During this phase, no computation occurs and hence the *data* does not propagate to the output *out*.

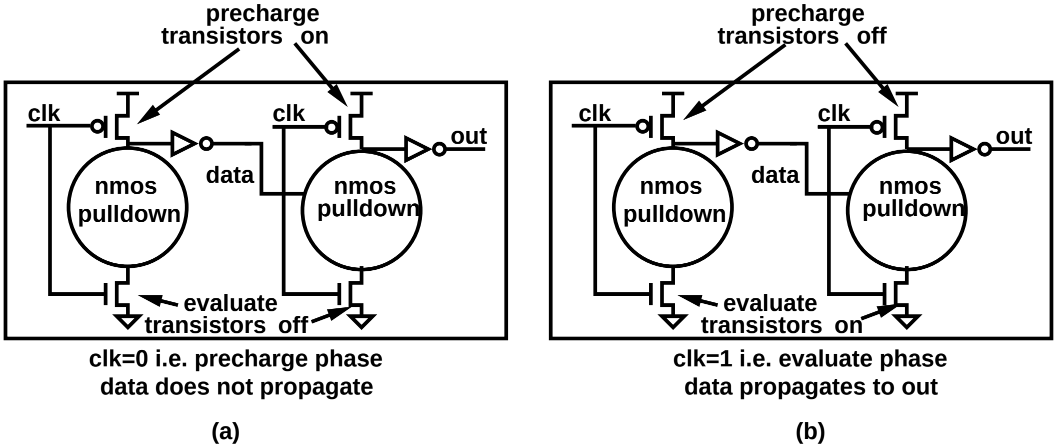


Figure 3.1: Domino operation

During the *evaluate* phase the *evaluate transistor* is switched on and depending on whether the NMOS pulldown is on or off, the dynamic node is either pulled down or stays the same, i.e., *data* propagates to *out*. Domino circuit families require the use of static inverters or other static inverting circuits between consecutive stages. That is to say, it a requirement of Domino circuit families that an even number of stages feed into a dynamic stage. This is necessary in order to prevent false triggering and discharge of a dynamic node during the *precharge* phase of the previous node, because a charge once lost is never regained until the next *precharge* phase, and would lead to erroneous values at the output. Domino is inherently fast because, like most other dynamic circuit families, it minimizes use of PMOS networks doing computation, and *precharge* of all cells in a stage (i.e., driven by a single clock) is simultaneous and not dependent on one another. Also there is only a (0 to 1) transition at the output of a gate, thus halving the number of transitions in a stage [134].

A potential optimization to domino is to remove footers in selected stages. This circuit style, called footless domino, has its advantages in being up to 5%-15% faster and enabling implementation of more complex gates [134]. However footless domino requires a hard setup on the falling edge of any input to a gate. Otherwise there could be a false discharge,and there is no possible recovery from that in domino circuits. Also the clock to such a gate needs to be at logic value 1 when the inputs are high, else there will be a path from supply to ground, leading to short circuit currents and power dissipation.

A way around the hard setup requirements of footless domino circuits was suggested in [100]. This circuit family called *Delayed Reset Domino* was used to build the first 1 GHz processor [123]. The basic idea is to provide each level of logic with its own complete reset or *precharge* phase of the clock. This is done by propagating the clock along with the data, and inserting delay elements between the clocks of two consecutive logic stages. In this manner, the *precharge* clock wave propagates through the logic prior to the computation wave. The *precharge* clock is shaped as it proceeds through the circuit to ensure that the computation wave does not collide with the *precharge* wave [100].

The main drawbacks of the domino circuit family and its variants are that they have much higher power dissipation due to higher switching factor than static CMOS, are sensitive to noise, and require complex structured like dual rail for implementing inverting logic. The Output Prediction Logic (OPL) [87] is a circuit family that was developed in order to overcome the noise susceptibility of the domino logic family, while maintaining the speed benefits [134]. OPL utilizes the alternating nature of logical output values for inverting gates on a critical path. OPL predicts that every inverting gate output on a critical path will be at logic value 1 after the transitions are completed. Since all gates are inverting (same as in static CMOS), the OPL predictions will be correct half of the time, and every other gate will not have to make a transition. Since a logic value 1 at the output of every gate is not a stable state, all gates are *precharged* (by making clock=0) till their inputs are ready to evaluate. When the inputs are ready to evaluate, the clock signal is made high and the circuit behaves like a normal static CMOS circuit. This reduces the switching in the circuits by 50% [87], [134]. The clock distribution is done using a delayed clock scheme as in *Delayed Reset Domino.*

Limited Switch Dynamic Logic (LSDL) [11], [34], [90], [99], is a dynamic circuit family with a domino front end to maximize performance, followed by a latch, which minimizes power dissipation by reducing switching at the output.

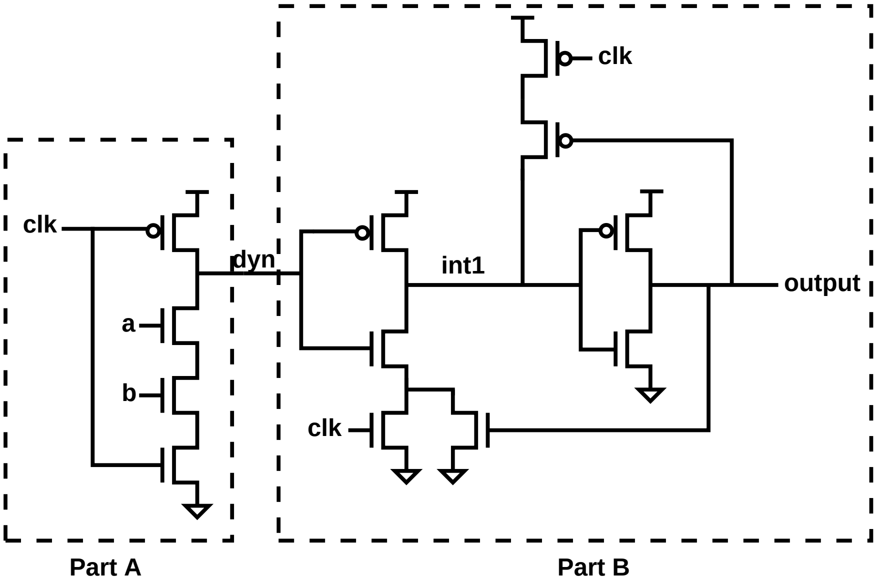


Figure 3.2: LSDL NAND gate

This merged logic latch approach is explained using a LSDL NAND gate shown in Figure 3.2 [34]. *Part A* in the figure forms the dynamic portion of the cicuits, where the computation of the NAND of a and b is performed. The dynamic node *dyn* is fed into the latch or *Part B*. The latch provides two level of amplification to the dynamic node before it drives other gates. This facilitates smaller device sizes in the computational tree, which leads to lower area, lower supply voltage requirements, higher computational stacks, and consequently lower leakage. Additionally, latch overhead can be mitigated by carrying out computation within the latch i.e., the latch can also be used to perform useful logical computation, instead of just providing isolation and scan features [34], [90]. Additionally, unlike domino, logic inversion in LSDL can be performed very easily by just inserting static gates between latches, wherein in domino, it would have required structures like dual-rail [34].

3.2 Precharge Control Schemes

Dynamic circuits are different from static CMOS circuits from the perspective of delay test and debug, since they have a *precharge* phase between any two *evaluate* phases, and consequently, between application of any two vectors. This would preclude application of initialization vector (V_1) followed by transition vector (V_2) for delay test and debug in dynamic circuits with complete access to all inputs, as is done in corresponding static CMOS circuits. In this section solutions to this problem are presented, which rely upon the capability to stop this *precharge* phase between application of V_1 and V_2, and reduce the problem of timing characterization of dynamic circuits with complete access to that of static CMOS circuits with complete access. This capability, is provided by DFTD schemes called *Precharge Control Schemes.*

The first such scheme, called *Precharge Control Scheme 1* is shown in Figure 3.3. In this scheme, an additional transistor is added in every *precharge* path, i.e., the path from the dynamic node to V_{dd}, and this transistor is controlled by a separate signal called *precntrl*. This additional transistor is shown by thicker lines in Figure 3.3, which is basically the example circuit, i.e., the two level AND gate circuit that was used to illustrate the problem of delay test and debug in dynamic circuits, as well as drawbacks of previous work in Chapter 1.

Although addition of an extra transistor has the potential of slowing down the *precharge* process, it can be considered feasible since the *precharge* path is generally the less critical path in a dynamic circuit. This is more so in case of high performance industrial dynamic circuit families like *Delayed Reset Domino* [100], where each *precharge* transistor is given an entire half cycle to *precharge*, during which all other inputs to that gate are switched off.

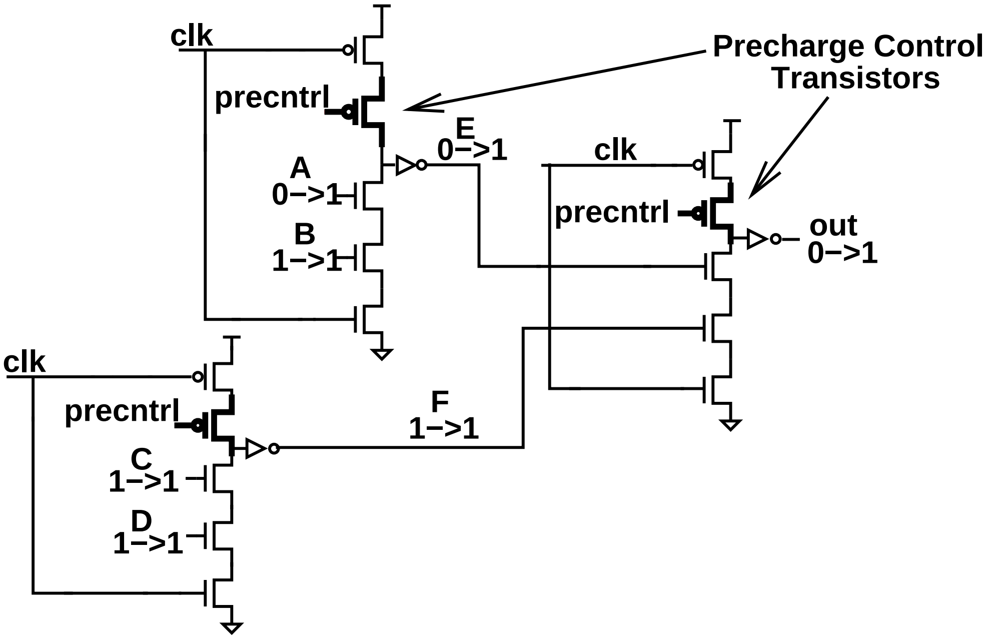


Figure 3.3: *Precharge Control Scheme 1*

The transistors controlled by the *precntrl* signal are basically switched off in the *precharge* phase between the application of the initialization and transition vectors. The operation of the *Precharge Control* schemes is better explained with the help of a timing diagram for Figure 3.3, which is shown in Figure 3.4. As before (i.e., in illustrations of the dynamic two level AND gate in Chapter 1), the two cycles of clock used for delay test are divided into four phases. Phase 1 and 3 are called the *precharge* phases of the initialization vector and transition vector respectively, while phases 2 and 4 are called *evaluate* phases of V_1 and V_2 respectively.

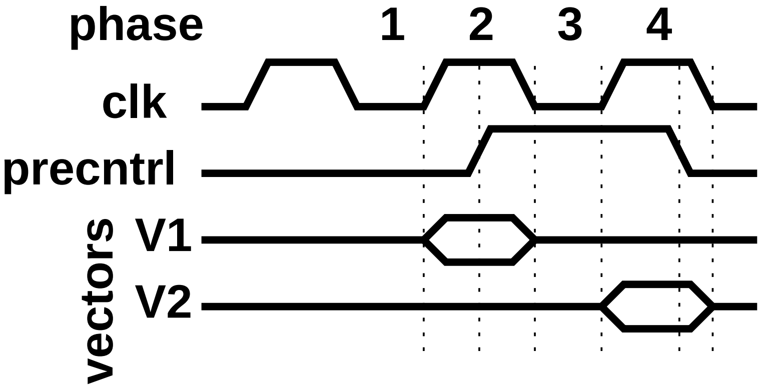


Figure 3.4: Timing diagram for *Precharge Control* schemes

In phase 1, the *precntrl* signal is kept at logic 0 value, allowing a normal precharge, irrespective of the input vector. During phase 2, the initialization vector is applied, which in this example PUT is *A,B,C,D*=0,1,1,1. Also, sometime during this phase, the *precntrl* signal needs to be asserted, switching off the *Precharge Control* transistors. The timing of assertion of the *precntrl* signal is flexible within phase 2, that is to say it can be asserted anytime after the rising edge of the first cycle, but the *Precharge Control* transistor must be completely switched off before the beginning of phase 3, i.e., the falling edge of the second cycle.

In phase 3, since the *Precharge Control* transistors are off, internal nodes *E, F* and output *out* remain at their initialized values, i.e., logic value 0 and 1 respectively. At the beginning of phase 4, the transition vector is applied, which in this example PUT case is *A,B,C,D*=1,1,1,1, and the 0 to 1 transition from input *A* propagates to the output *out*, just as in case of a static CMOS implementation of this circuit. Hence if there is a delay fault, it will be detected, and also it will be ascertained without ambiguity that the PUT is the faulty path. This will make the entire process of delay test, debug and diagnosis of dynamic circuits more efficient, effective and streamlined.

In Figure 3.4, the initialization and transition vectors are shown to be applied at the beginning of their corresponding *evaluate* phases. In case of regular domino, this is flexible, i.e., they could be applied sometime during the preceding *precharge* phase too. However, this can make the precharge slower, and is usually avoided. In case of footless domino families, including the high performance *Delayed Reset Domino* circuit family, this is a hard requirement. That is to say, the pull down tree cannot be on during the *precharge* phase. Hence V_1 and V_2 are applied only during *evaluate* phases.

Although *Precharge Control Scheme 1* is effective, it involves placing of additional transistors on each logic gate. Instead of that, one could achieve the same end by gating the clock to the *precharge* transistor with *precntrl* using an OR gate as shown in Figure 3.5. This scheme, called *Precharge Control Scheme 2*, will have significantly lower area overhead as compared to *Precharge Control Scheme 1*, while maintaining the same level of efficiency in timing characterization of dynamic circuits. However, there would be a problem of additional loading on the clock signal, and there will also be a skew between the clock input to the *precharge* and *evaluate* transistors of

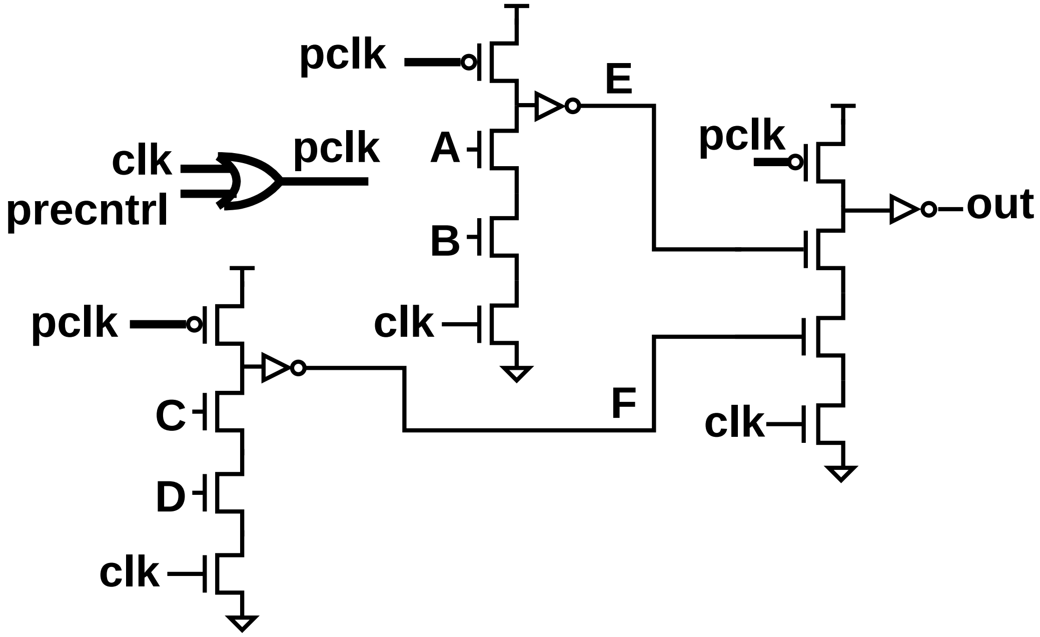


Figure 3.5: *Precharge Control Scheme 2*

every dynamic gate. The clock loading problem can be overcome by suitable buffering, but skew could wreak havoc in circuits with significantly low noise margins. However, this scheme could be effective for delay fault test and debug of chips built using footless domino and *Delayed Reset Domino* circuit families.

An optimization over *Precharge Control Scheme 1* was speculated, wherein a single transistor would be used per logic stage as shown in Figure 3.6. This scheme is referred to as *Precharge Control 3.* The single transistor is controlled by the *precntrl* signal, just like the *Precharge Control* transistors used in every gate in *Precharge Control Scheme 1.* This scheme has very low hardware overhead, but suffers from a significant drawback. In case the *Precharge Control* transistor is shared by two gates that drive series transistors of the same gate, then this scheme cannot be used. This is because, under such a circumstance, the dynamic nodes of the two gates will need to have opposite logic values after application of the initialization vector, i.e., after phase 2 in the above examples, and subsequently, i.e., in phase 3, there will be a path between the two dynamic nodes, through the *precharge* transistors (which are controlled by the clock), leading to charge sharing. This path is shown using the curved line in Figure 3.6. This in turn could lead to either one or both the dynamic nodes to have erroneous logic values before the transition vector can act on them, and hence this optimization is not recommended.

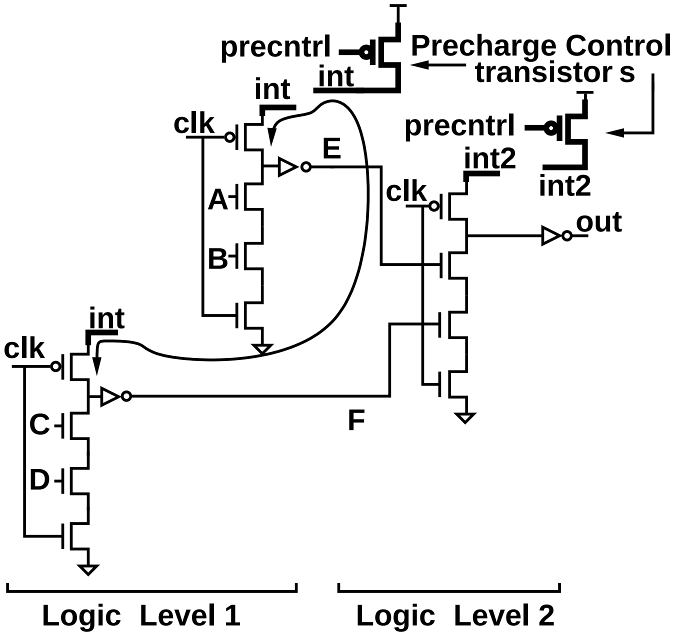


Figure 3.6: Drawback of using a single transistor for precharge control

The importance of *Precharge Control* schemes are projected with the help of a case study of a dynamic 4-to-2 Carry Save Adder (CSA) [34] designed

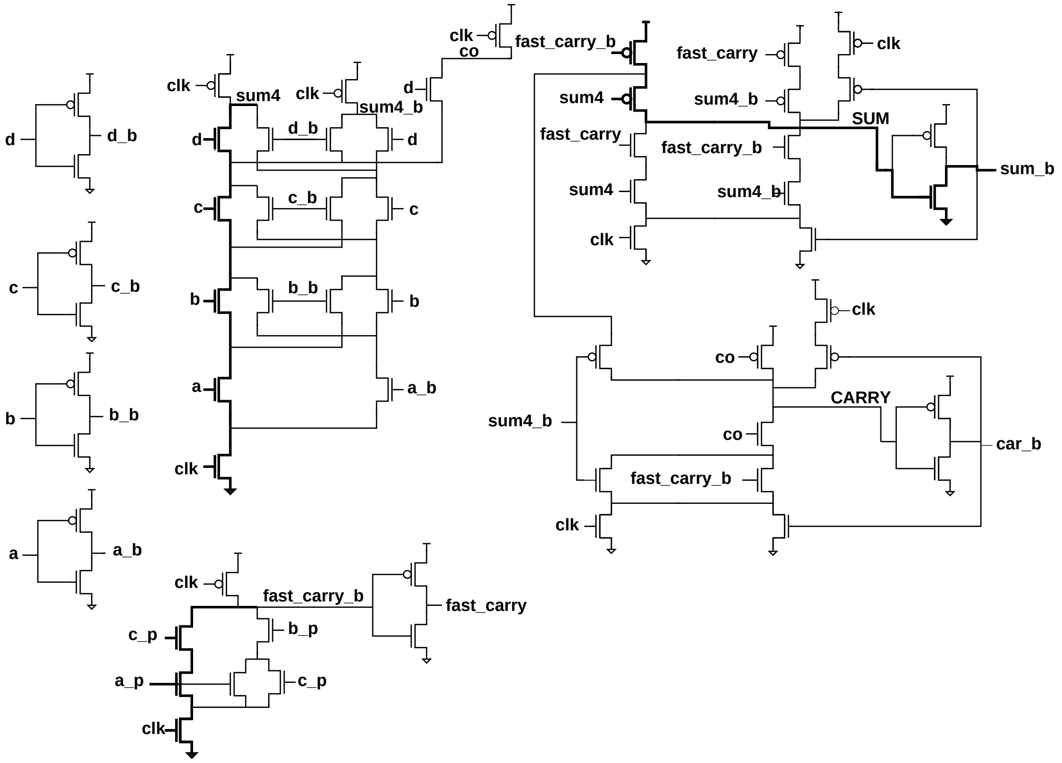


Figure 3.7: Case study: *LSDL 4-to-2 Carry Save Adder*

using the LSDL circuit family. This circuit was used for performing 4-to-2 compression in an 8 GHz multiplier [11]. The circuit diagram of the 4-to-2 is shown in Figure 3.7 [34]. Although LSDL circuits are known for their simplicity, having only two levels of logic per pipeline stage (with the second one being part of the latch), they still suffer from the ambiguity in determining the faulty path. One such case is shown in Figure 3.7. Let the PUT be *a* to *sum_b*. The segment *a-b-c-d-sum4* feeds into the pull-up path through *fast_carry_b* and *sum4* (i.e., pulling down *sum_b*), shown by the thicker lines in the figure. However there are two different dynamic nodes feeding into this pull-up path, i.e., the segment mentioned above and an off-path segment that pulls down *fast_carry_b* shown by thicker lines in the figure. The precharge phase causes both *sum4* and *fast_carry_b* to be precharged to a logic value of 1. After application of the transition vector both *sum4* and *fast_carry_b* discharge simultaneously. Therefore, if there is a delay fault, it will not be possible to ascertain if the PUT is the fault path, or the path through *fast_carry_b* is the faulty one. Over 70% of such combinations of segments in the paths of the 4-to-2 CSA are subject to this kind of ambiguity if *Precharge Control* is not used.

In the above cases, only logic 0 to logic 1 transitions on the input have been considered. This is primarily because in dynamic circuit families like *Domino* and *Delayed Reset Domino*, there is always an even number of stages feeding into a dynamic gate (i.e., the circuits are non-inverting), and all computations depend on the rising edge of the input propagating to the output. However a timely transition from logic 1 to logic 0 is also critical for correct operation of a dynamic circuit. During the *precharge* phase, all evaluation trees need to go off (i.e., all inputs need to transition to logic 0), so that the dynamic node can *precharge* fully before the beginning of the next *evaluate* phase. In footless dynamic circuits, this is a hard requirement for the operation of the circuit. In regular dynamic circuits with footers, like regular domino, this is usually preferred, since charge sharing might not allow complete *precharge*, which in turn causes a wrong value to be latched in.

Thus, a delay fault in a falling transition on the input manifests itself as a delay fault in the *precharge*. Hence the mechanism for delay test of *precharge* logic given in [96] can be used to handle this. The *Precharge Control* schemes are not needed for *precharge* test, so they are de-activated. An initialization vector needs to be applied during an *evaluate* phase, that will completely discharge the node, i.e., we need to keep the input of the PUT, as well as all inputs between the transistor that the input to the PUT drives and ground, at logic 1. Subsequently, an at-speed *precharge* phase is applied, during which the input to the PUT is made to transition from logic 1 to logic 0. If there is a delay fault during this transition, the concerned dynamic node will not *precharge* suitably, and the inverter or static gate following it will treat it as a stuck at 0 fault on the dynamic node. Now this *precharge* phase will be followed by a slow *evaluate* phase, during which this stuck-at-0 fault will be propagated to the output [96] (which may be a Primary Output or input of a scan flop, also known as Pseudo-Primary Output [135]).

The above test is not applicable for LSDL circuit family, since it is a complementing dynamic circuit family (i.e., not positively unate). However, in LSDL circuits, a latch immediately follows a single level of dynamic logic. Hence an error in *precharge* due to delayed falling transition on an input will immediately propagate to the corresponding Pseudo-Primary Output.

When using *Precharge Control* with *functional justification* [28] in a scan based test environment, a question may arise about a situation where the transition vector to a CUT needs the previous state to be at a *precharged* value. However in Domino or any of its variants, such a circumstance will only occur if we are trying to test for a falling transition of input, and as mentioned earlier, the *Precharge Control* scheme will be de-activated in such a case. In LSDL, this situation can be tackled by applying an initialization vector that causes the preceding stage to remain *precharged* after the *precharge* phase of the initialization vector of CUT, i.e., phase 1 in our examples. If a completely functional test is used for delay testing, then the method for tackling functional justification can be extended accordingly.

3.3 Delay Test and Debug of Scan Based Dynamic Circuits

Scan design forms an integral part of the Design for Test (DFT) infrastructure of most modern integrated circuits. The basic advantage of incorporating scan feature on a chip is that it reduces the problem of test generation for stuck-at faults in a sequential circuit, into test generation for stuck-at faults in a combinational circuit with complete access to all primary inputs (PIs) [48], [135]. However, as has been explained earlier, scan design poses significant problems for timing characterization, both of static CMOS and dynamic circuits. This is because, the process of scanning in V_2 after V_1 has been applied to the CUT, creates the effect of multiple unwanted intermediate vectors being applied to the CUT between V_1 and V_2.

In this section, DFTD techniques techniques are presented that help overcome this problem, specifically in dynamic circuits. However for these techniques to be effective, they need to be coupled with one of the *Precharge Control* schemes presented in the previous section. The first such technique

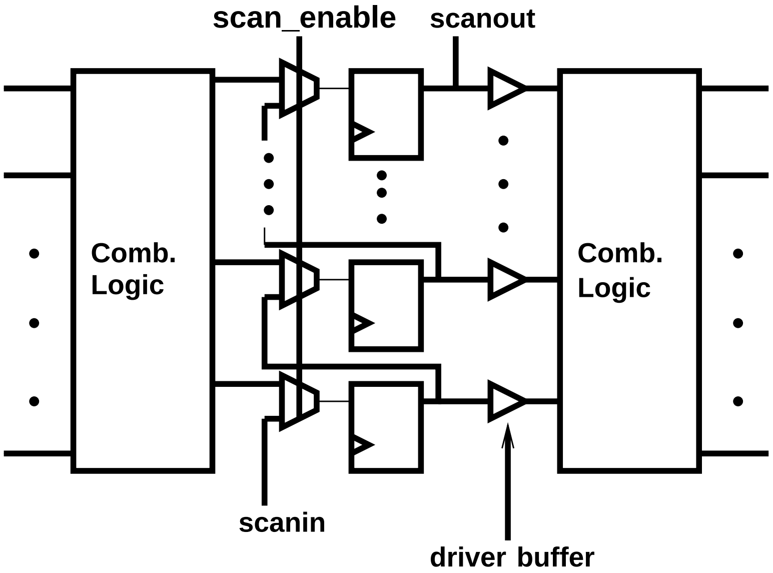


Figure 3.8: Two stage combinational logic with scan

involves placement of dynamic AND gates at the output of scan flops. Figure 3.8 shows two stages of combinational logic with a scan chain between them (this figure is similar to the earlier two stage figure except for a non-inverting buffer). Flops need buffers to drive the combinational logic, and we have shown these buffers explicitly in the figure. Now in case the buffer is dynamic, it will manifest itself as shown in Figure 3.9a, along with *Precharge Control Scheme 1* [1]. Now if an additional transistor is placed in this circuit, then at the logical level, the circuit will be converted from a buffer into an AND gate. The scheme in shown in Figure 3.9b. The *scan_enable* (SE) signal

---

[1] In [37], *Precharge Control Scheme 1* has been erroneously referred to as *Precharge Control Scheme 3* and vice versa in cases where they have been used in conjunction with DFTD schemes for scan based dynamic circuits. The appropriate corrections have been made here.

can be used to control this additional transistor.

Thus, whenever a vector is being scanned into the scan chain, the combinational logic or CUT that is being driven by the scan chain is completely isolated from the scan path. Hence the CUT remains at the state it was in after application of the previous vector, which in case of delay fault testing will be the initialization vector. When the complete transition vector has been scanned in, it is applied to the CUT and its response propagated to the primary outputs (POs) or pseudo-primary outputs (PPOs). Hence the problem of multiple unwanted intermediate vectors is overcome. Reuse of the *scan_enable* signal obviates the need for routing an additional global signal.

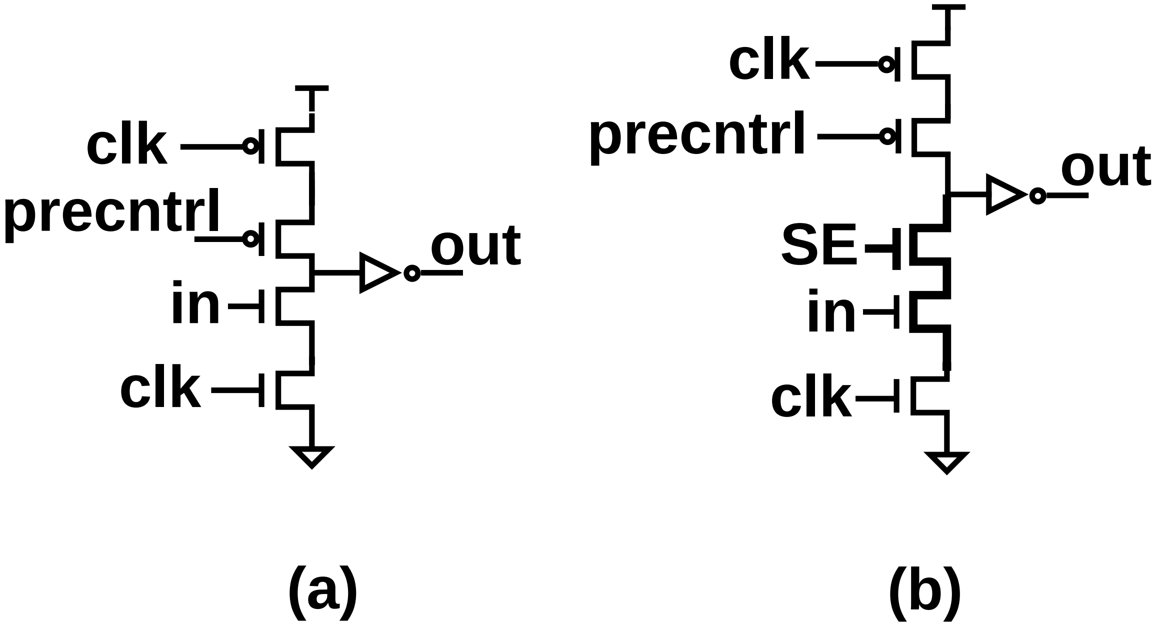


Figure 3.9: a) Dynamic buffer b) *AND Gated Flop Output* scheme

In certain cases, where there is a fast pull down stack in the dynamic

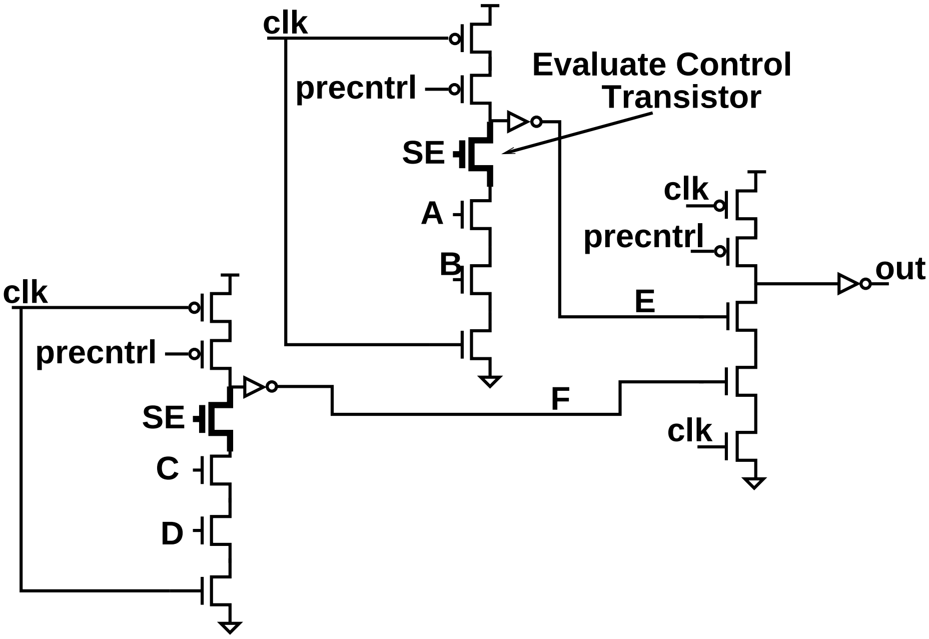


Figure 3.10: *Evaluate Control* scheme

portion of the circuit, like the 4-to-2 Carry Save Adder, an *Evaluate Control* technique can be used, wherein the pulldown tree of a dynamic gate or complex circuit block is connected to the dynamic node using a single NMOS transistor, known as *Evaluate Control Transistor*. This technique is illustrated with the help of a two level logic shown in Figure 3.10. In the figure, inputs *A, B, C* and *D* form pseudo-primary inputs (PPIs) of the combinational logic block (i.e., outputs of scan flops) [135]. During scan operation, the joint action of the *Precharge Control* scheme (in this case we have used *Precharge Control Scheme 1*) and *Evaluate Control* technique, isolates the CUT from the scan chain, and the *SE* signal is reused for controlling the *Evaluate Control* transistor. A point to be noted here is that the *Evaluate Control* transistors are required only in the first level of logic in any combinational stage. Considering that there are usually six to twelve levels of logic in a regular combinational logic stage, the area overhead of this scheme is minimal, and so is the performance penalty.

Another scheme that can be used for delay testing of scan based dynamic circuits is *ANDed Evaluate Clock*, wherein the clock going to the footer or *evaluate* transistor is controlled with an AND gate as shown in Figure 3.11. The operation of this scheme and the effect it achieves is the same as the *AND Gated Flop Output* and *Evaluate Control* schemes. However from a circuit designers' perspective, this scheme would not be considered very effective in spite of its low hardware overhead. This is primarily because this scheme is susceptible to charge sharing noise [122], [134], which might render it ineffective for delay fault testing. Nevertheless, this scheme can find its use in delay test and debug of logic stages where stack heights of evaluation trees are small, and the allowable hardware overhead for test purposes is limited.

Both *AND Gated Flop Output* and *Evaluate Control* schemes have one more significant advantage, namely, reduced power in test mode, specifically during scan operation. When a vector is being shifted into a scan chain, the switching in the combinational logic could become very high. In the worst case, it would be switching its state every cycle, and this may be higher than the switching factor the circuit was originally designed for. In case the scan in operation is done at a high frequency, then this could lead to very high power dissipation and possible damage to the chip. The aforementioned schemes isolate the combinational logic from the scan path when a vector is being

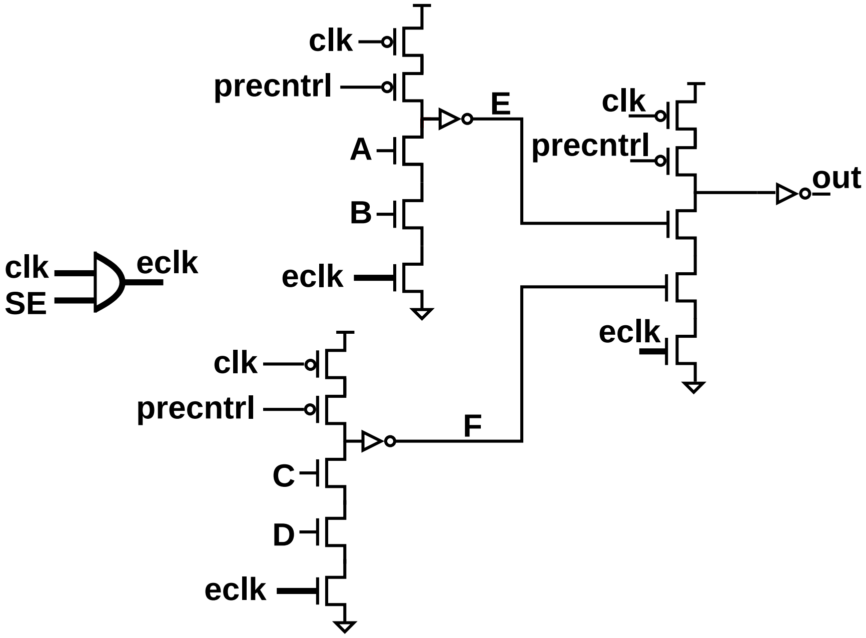


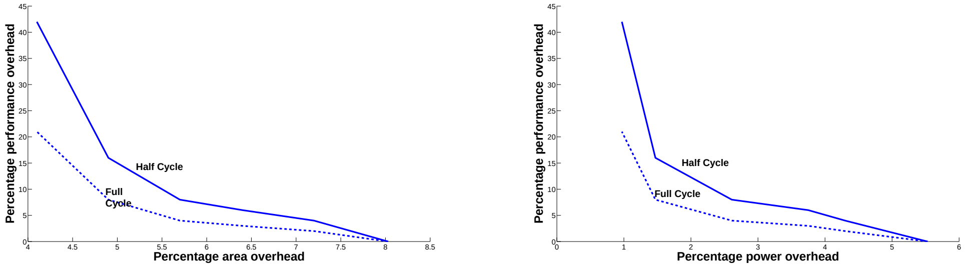
Figure 3.11: *ANDed Evaluate Clock* scheme

shifted into it. Hence there is no switching activity in the combinational logic block, leading to a significant reduction in power dissipation during scan operation.

3.4 Simulation Results

Simulation results for the Design for Test and Debug schemes presented in this paper, were obtained using a transistor level schematic of a high performance test circuit, namely, a 4-to-2 Carry Save Adder designed in 65nm Silicon-On-Insulator (SOI) [13] technology, using LSDL circuit family. The simulations were carried out using nominal process conditions, with a supply voltage of 0.9V and temperature of 85°C. A circuit diagram of the 4-to-2 Carry

Save Adder is presented in Figure 3.7 [34]. It is built entirely in dynamic logic, hence minimal number of PMOS transistors. Additionally, some portion of the computation is carried out in the latch circuit, which is characteristic of the LSDL circuit family. These factors make this circuit inherently fast. Another factor which enhances performance is the relocation of the *fast_carry* computation to where it is needed, i.e., the present bit. The *fast_carry* is a majority function of inputs A, B, and C of the previous or lesser significant bit, and its computation is relocated by inputting those signals into the present bit. The SUM is an XOR of inputs A, B, C, D and the carry_in or the *fast_carry*. The $CARRY$ is a majority function and is obtained as CARRY $=$ Maj $[(A \oplus B \oplus C), D, Cin]$ [34].



(a) (b)

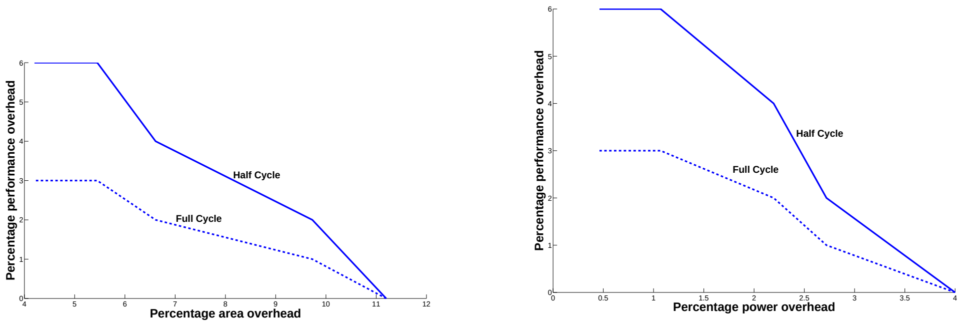
Figure 3.12: *Precharge Control Scheme 1* a) Area-performance curve b) Power-performance curve

Performance and power overhead numbers are given for the various DFTD schemes for a cycle time of 100ps. However, in LSDL pipelines, two phase clocking is sometimes attempted, wherein alternate stages operate in the high and low phase of the same cycle [11], [90]. Thus many stages will have only half the cycle time for operation which in this case in 50ps. Hence the correlation of area and power overheads with the associated performance overhead is presented for both half cycle and full cycle operation.

Figure 3.12a shows the area-performance curve for *Precharge Control Scheme 1*, and Figure 3.12b shows the power-performance curve for the same. A *precharge* performance overhead of 42%, was the highest that could be tolerated without violating cycle time and associated tolerance margins. This is tantamount to a 21% performance overhead in full cycle operation and for this upper bound on performance overhead, the area overhead is 4.1% and the overhead in switching power is 0.97%. The area and power penalty incurred in order to have the same level of *precharge* performance as obtained without insertion of *Precharge Control Scheme 1* are 8.03% and 5.53% respectively.

The area overhead incurred by incorporating *Precharge Control Scheme 2* into the 4-to-2 CSA is 2.8% and the power overhead is 2.5%.

The area overhead of the *Evaluate Control* scheme is 4.2% for a 3% performance penalty in full cycle operation, and 6% performance overhead in half cycle operation. The power overhead for this performance penalty is 0.46%. In order to ensure 0% performance overhead in the 4-to-2 CSA while using this scheme, an area overhead of 11.2% and a power overhead of 4.0% is incurred. When this scheme is used in circuits where there are multiple levels of logic in a stage, unlike LSDL circuits, the percentage area and power overhead will be much lower since the scheme will only be used in the

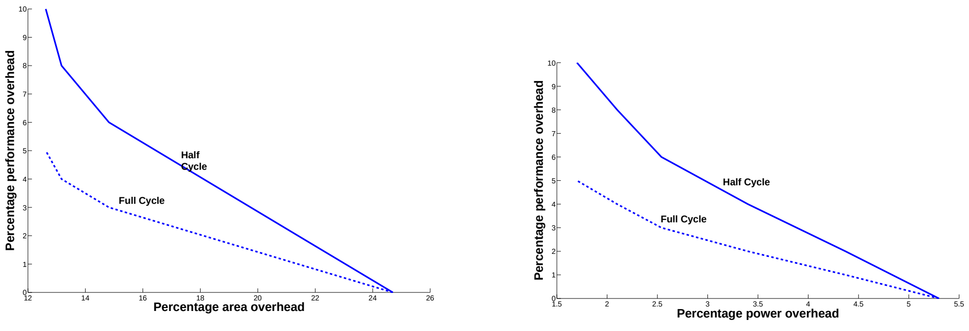


(a) (b)

Figure 3.13: *Evaluate Control* scheme a) Area-performance curve b) Power-performance curve

first level of logic. The area-performance and the power-performance curves for the *Evaluate Control* scheme are shown in Figure 3.13a and Figure 3.13b respectively.

The *AND Gated Flop Output* scheme results in maximum area overhead when used in the 4-to-2 CSA. This is mainly because the 4-to-2 CSA has only two levels of logic, and we are effectively adding one more level to it. Hence to limit performance overhead to 5% we need to incur an area penalty of 12.62%, whereas in order to have no performance overhead, we need to incur an area penalty of over 24%. Hence this scheme or its static CMOS variant *Tri-Scan* [36] are more suitable for use in designs that have multiple levels of logic, and addition of another level does not create a significant performance overhead. However, since dynamic circuits are generally custom design, the *Evaluate Control* scheme will be a better choice for them. The area-performance and



(a) (b)

Figure 3.14: *ANDed Flop Output* scheme a) Area-performance curve b) Power-performance curve

the power-performance curves for the *AND Gated Flop Output* scheme are shown in Figure 3.14a and Figure 3.14b respectively.

The area overhead and power overhead associated with the use of the *ANDed Evaluate Clock*, in the 4-to-2 CSA is 2.96% and 2.45% respectively.

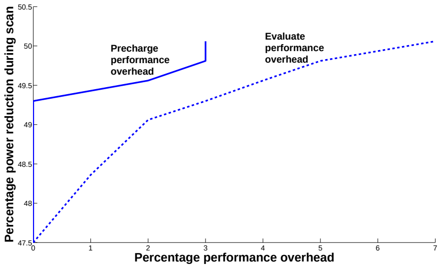


Figure 3.15: Power-performance curve illustrating power reduction during scan operation

Finally, the use of a *Precharge Control* scheme along with either *Evaluate Control* or *AND Gated Flop Output* will lead to significant savings in power dissipation during scan operation. Figure 3.15 shows the power-performance curve, illustrating the power saving during scan operation, for *Precharge Control Scheme 3* used in conjunction with *Evaluate Control* scheme in the 4-to-2 CSA. The two curves illustrate the reduction in power dissipation during scan operation, and the associated performance penalty incurred in during *precharge* and *evaluate* modes, for full cycle operation. It can be seen that in the best case, a power savings of over 50% over standard scan methodology can be obtained using a combination of these two schemes. In this experiment, scan environment was simulated by switching inputs in every cycle. The dynamic node was isolated from supply and ground when the *Precharge Control* and *Evaluate Control* schemes were activated. The circuit of Figure 3.7 was used, which has no launch latches. The capture latches do not switch when the dynamic node is isolated by activation of the *Precharge Control* and *Evaluate Control* schemes. Hence, as mentioned in the previous chapter, isolation techniques like those presented in this and the previous chapter may not provide significant benefits in terms of power savings during scan operation, if the power dissipation during scan operation is dominated by power dissipation in the latches.

3.5 Summary

Dynamic circuits are used in a large number of high-performance chips to speed up critical paths. Consequently, their timing characterization is imperative, and current solutions have significant drawbacks. Two sets of Design for Test and Debug (DFTD) schemes have been presented in this chapter.

The first set of schemes enable pattern application for delay test and debug in dynamic circuits with complete access to all inputs, and the second set of schemes, when used in conjunction with the first, enable pattern application in scan based dynamic circuits. The use of these schemes reduces the problem of delay test and debug of dynamic circuits to that of delay test and debug of scan based static CMOS circuits with complete access to all inputs. This in turn also makes dynamic circuits more compatible with existing tools for testing of static CMOS circuits.

The work presented in this chapter was published in [37], and the material in this chapter is based on the same. The test circuit used in this chapter was published in [34].

Chapter 4

Response Analysis for Timing Characterization

In Chapters 2 and 3, the focus was on techniques to enhance controllability for timing characterization, i.e., delay test and debug of ICs. The focus of this Chapter is on observability enhancements for timing characterization. As has been mentioned in Chapter 1, observability requirements for an efficient timing characterization scheme include detection of gross as well as small delay faults, and determination of magnitude of violation for efficient debug. In Chapter 1, the drawbacks of existing timing characterization schemes were described, and the use of delay line based time-to-digital conversion techniques for on-chip delay measurement based timing characterization was proposed. In this chapter three such timing characterization schemes are presented.

The concept of time-to-digital conversion is an old one, and a lot of literature exists in this area. Some schemes for time-to-digital conversion were listed in [46], which include methods based on CMOS tapped delay lines [30], [55], [56], [79], [108] or otherwise [116], [125]. The latter included a technique where the delay to be measured is converted into a pulse, which in turn is used to charge a capacitor. The final voltage of the capacitor indicates the delay between the two signals [125]. The advantage of using delay line based techniques over others was also highlighted in [46]. Comparison of different techniques for time-to-digital conversion has been reported in the literature, for example, in [30], [108]. A single ended tapped delay line based on CMOS technology was presented in [9], [10]. In [52], [53], the use of a single ended, as well as a differential or balanced tapped delay line (also known as a *Vernier Delay Line*) scheme for time-to-digital conversion were presented. A balanced or *Vernier Delay Line* circuit is made up of two delay chains, with intermediate tap points from both delay chains feeding into different inputs of storage elements (which may be clock and data inputs or S and R inputs of flip-flops) [46], [52]. The primary advantage of a balanced delay line over a single ended one is that the resolution of delay measurement in the latter is limited by the minimum achievable delay in a given technology, whereas, in the former, the minimum achievable delay difference between two delay elements places a lower bound on resolution, consequently, enabling lower values of resolution [56]. *Vernier Delay Line* circuits were also reported in [46], [55], [56], [107], [108]. In some of these papers, techniques to make *Vernier Delay Lines* insensitive to variations in process, voltage and temperature were presented, which included Delay Locked Loop (DLL) based techniques to fix the resolution of various stages of the circuits by providing different control voltages to voltage controlled buffers in the delay lines. Most of these papers suggested the use of

Vernier Delay Line for time-to-digital conversion in physics experiments. In [56] the use of such a circuit in instrumentation, clock and data recovery in communication applications was suggested.

The use of *Vernier Delay Lines* for jitter measurement has also been reported in the literature, for example in [1], [20], [21], [27]. In [20], [21] a single stage *Vernier Delay Line* was presented as a solution to the problem of mismatch in loading, and also for reducing the area overhead of *Vernier Delay Line* schemes. To enable robustness to variations, calibration technique was presented to determine resolution of the *Vernier Delay Line*. Pulse shrinking based delay lines have also been reported in the literature as another method for time-to-digital conversion, and these are covered later in this chapter.

As mentioned earlier, in this chapter, three delay line based time-to-digital converters are presented, which can be used for on-chip delay measurement based timing characterization of ICs. The rest of this chapter is organized as follows. In Section 4.1 the first such scheme, titled *Modified Vernier Delay Line* is presented, which overcomes drawbacks which existing *Vernier Delay Lines* would suffer from, if they are used for timing characterization of ICs. In Section 4.2, a second balanced delay line based timing characterization scheme is presented, which re-uses Design for Test (DFT) infrastructure already present in ICs. In Section 4.3, the third timing characterization scheme is presented, which overcomes certain drawbacks of the schemes presented in Sections 4.1 and 4.2, as well as those of existing *Vernier Delay Lines*. This scheme, called *Skewed Inverter Delay Line* is based on pulse shrinking, and in Section 4.3, an overview of the existing literature on pulse shrinking based delay lines is also presented, in addition to the description of the *Skewed Inverter Delay Line* scheme, and its use for timing characterization of ICs. Simulation results are presented in Section 4.4 and a brief summary is presented in Section 4.5.

4.1 Modified Vernier Delay Line

The primary drawbacks of using existing balanced delay line schemes like *Vernier Delay Line* for delay fault test and debug of critical paths in a chip, is that they can only handle only cases where a path has a single kind of transition on its input and output. That is to say, a given delay line could handle either rising transition or falling transition on input, but not both. Hence separate delay lines would be required for such transitions, as well as for measuring delay of inverting or non-inverting paths when either of these transitions are applied to them. In integrated circuits, critical paths can be inverting or non-inverting, and either rising or falling transitions can cause the worst case delay. Hence there would be a need for multiple types of delay lines if existing *Vernier Delay Lines* were used for timing characterization, and also there will be limitations on the use of a single delay line for measuring delay of multiple paths, thus limiting the practiality of using on-chip delay measurement based timing characterization. Additionally, in existing *Vernier Delay Line* techniques, reading out of the values from the storage elements for calculating delay is a difficult process, and existing solutions can have high overhead. In this section, a scheme called *Modified Vernier Delay Line*

(MVDL) is presented, which overcomes these drawbacks of existing balanced delay line schemes with very low overhead.

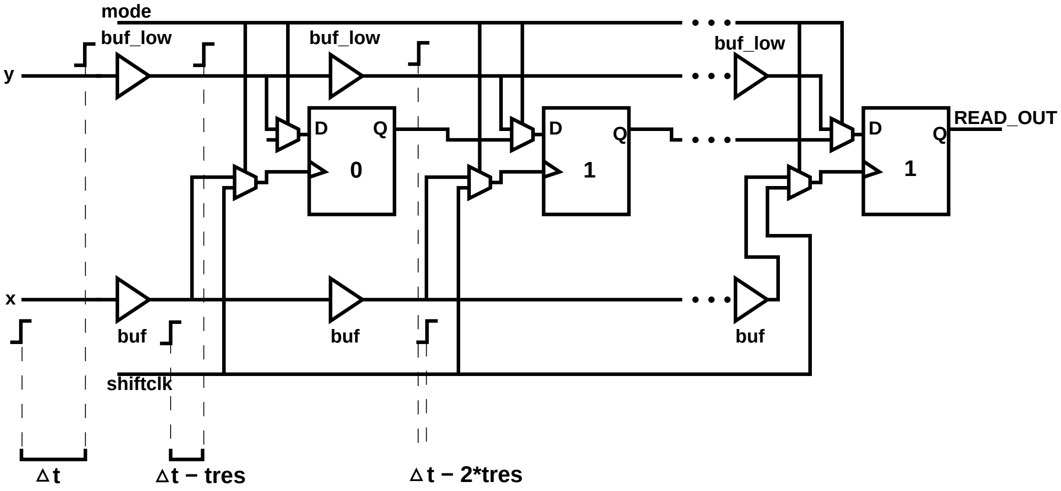


Figure 4.1: *Modified Vernier Delay Line*

The MVDL scheme is shown in Figure 4.1. It is basically a balanced delay line scheme, having two buffer chains and associated latching elements. Each individual buffer *buf*, in the lower buffer chain, has a delay t_{buf} that is higher than delay t_{buf_low} of the corresponding buffer *buf_low* of the upper buffer chain. A combination of a *buf*, *buf_low* and an edge-triggered (rising edge triggered in this work) latching element or flip-flop forms a stage of the MVDL, with the resolution t_{res} of the stage being

$$t_{res} = t_{buf} - t_{buf\_low}$$

In the absence of variations, this value t_{res} would be the same for all the stages, and hence would be the resolution of the MVDL. The issue of behavior of MVDL in face of variations and tackling such behavior is handled later in this section. Another factor to note in Figure 4.1 is that each flip-flop has its clock and data inputs coming through two different multiplexers. The reason for this is explained later in this section. Also, the MVDL shown in Figure 4.1 can handle only paths where a rising transition on the input causes a rising transition on the output, just like existing schemes. This has been done to facilitate ease of explanation of the scheme, and later in this section the technique for handling all possible combinations of paths and transitions using the same MVDL is presented.

Consider a non-inverting PUT whose delay is measured using the MVDL, with rising transition at its input. The input of the PUT is routed to input x of the MVDL, and the output of the PUT is routed to input y of the MVDL. Before delay measurement, all the flip-flops in the MVDL need to be initialized to hold a logic value 0 in this case. Let the delay of the PUT be Δt. Then the difference between a rising transition at input x and a rising transition at input y is Δt (assuming that the signals are delayed equally when they are routed to MVDL inputs from the PUT). At each stage, this delay difference between signals at x and y is reduced progressively by an amount that equals t_{res}. Finally, at a particular stage, the two signals catch up, i.e., the signal travelling along the lower buffer chain (which drives the clock input of the flip-flops), arrives after the signal travelling through the upper buffer chain (which drives the data input of the flop-flop). This stage captures in a logic value 1, and is known as the *indicator stage* (referred to as n in this chapter). The *indicator value* for this PUT is logic value 1. The flip-flops in all stages preceding the *indicator stage* continue to hold a logic value 0, and the flips-flops in all stages that follow the *indicator stage* latch in a logic value 1. The *indicator stage* gives the delay of the PUT as a range illustrated by the following equation.

$$(n-1) * t_{res} < \text{Path delay} < n * t_{res} \tag{4.1}$$

The above equation does not take into consideration timing overheads of flip-flops, i.e., setup time and hold time. The behavior of the scheme with consideration of these issues is explained with the help of examples. Consider a MVDL scheme, with resolution equal to 10 (no units are used in this example to maintain generality). Consider 10 different PUTs, with delay ranging from 31-40. These are shown in Table 4.1. The signal coming from the input of the PUT, travelling along the lower buffer chain, and feeding into the clock input of the flip-flops is referred to as *clk* for the rest of this discussion. The signal coming from the output of the PUT, travelling along the upper buffer chain, and feeding into the data input of the flip-flops is referred to as *D* for the rest of this discussion. The *indicator stage* for the range of path delays considered with the aforementioned resolution is stage 4 of the MVDL. Any other range of delays for PUTs shows similar behavior as in Table 4.1, but with a different indicator stage.

The stage preceding the *indicator stage* is referred to as *preceder stage* and the stage following the *indicator stage* is referred to as *follower stage* in

PUT Number	**Path Delay**	**Delay difference between signals**				
		Stage 1	**Stage 2**	**Stage 3**	**Stage 4**	**Stage5**
1	31	21	11	1	-9	-19
2	32	22	12	2	-8	-18
3	33	23	13	3	-7	-17
4	34	24	14	4	-6	-16
5	35	25	15	5	-5	-15
6	36	26	16	6	-4	-14
7	37	27	17	7	-3	-13
8	38	28	18	8	-2	-12
9	39	29	19	9	-1	-11
10	40	30	20	10	0	-10

Table 4.1: Example to illustrate behavior of MVDL considering metastability issues

this work. In this example, stage 3 is the *preceder stage* and stage 5 is the *follower stage*. Column 2 gives the delay of various paths considered. and columns 3-7 give the delay difference between the *clk* and *D* signals for stages 1-5 respectively. A positive delay difference indicates that the rising edge of *D* arrives after the rising edge of *clk* at the flip-flop of the corresponding stage, and a negative difference indicates that the rising edge of *D* arrives before the rising edge of *clk* at the flip-flop of the corresponding stage. Six different cases are now considered to illustrate the impact of different setup and hold times on the accuracy of delay measurement using the MVDL. In this example, setup and hold time definitions given in [126] are used. Setup time is the minimum time before the clock edge for the data to become valid, and hold time is the minimum time after the clock edge for which the data should remain valid in order to avoid metastability.

- **Case 1: setup time = 6 and hold time = 6**. In this case, there will be metastability due to hold time violation in stage 3, i.e., the *preceder stage* for PUTs 1-5, and the corresponding flip-flop will not be able to definitively capture in a logic value 0. There will also be metastability due to setup time violation in stage 4 or the *indicator stage* for PUTs 5-10 and the corresponding flip-flop will not be able to definitively capture a logic value 1. There is no metastability in the *follower stage* for any of the PUTs. Thus, for one PUT, namely PUT 5 there will be an indefinite value due to metastability in two stages, the *indicator stage* and the *preceder stage*, and for other PUTs, there will be metastability in 1 stage, which could be either in the *indicator stage* (due to setup time violation) or in the *preceder stage* (due to hold time violation).

- **Case 2: setup time = 0 and hold time = 8**. In this case, there is no metastability in the *indicator stage* for any of the PUTs considered, since the rising edge of D, which arrives at the flip-flop of the *indicator stage* before the rising edge of *clk* or at the same time, and hence always gets captured in that flip-flop definitively because the setup time is zero. However, there is metastability in the *preceder stage* for PUTs 1-7 due to hold time violation. There is no metastability in the *follower stage* for any of the PUTs. Hence there is metastability in 0 or 1 stages and it would occur in the *preceder stage* due to hold time violation.

- **Case 3: setup time = 8 and hold time = 0**. In this case, there is metastability in the *indicator stage* for PUTs 3-10 due to violation of setup time. There is no metastability in the *preceder stage* for any of the PUTs considered. There is no metastability in the *follower stage* for any of the PUTs considered. Hence there is metstability in 0 or 1 stages and it would occur in the *indicator stage* due to setup time violation.

- **Case 4: setup time = 8 and hold time = -6**. In this case there is metastability in the *indicator stage*, for PUTs 3-10 due to violation of setup time. There is also a hold time violation in the *indicator stage* for PUTs 5-10. There is no metastability in the *preceder stage* or the *follower stage* for any of the PUTs considered. Hence there is metastability in 0 or 1 stages, namely the *indicator stage* for the PUTs considered, and could be due to setup time violation or hold time violation.

- **Case 5: setup time = -6 and hold time = 8**. In this case, there is metastability in the *preceder stage* for PUTs 1-7 due to hold time violation, and there is metastability in the *preceder stage* for PUTs 7-10 due to setup time violation. There is no metastability in the *indicator stage* or the *follower stage*, for all the PUTs considered. Thus there is metastability in 0 or 1 stages for the PUTs considered, namely, in the *preceder stage*, and it could be due to setup time violation or hold time violation.

- **Case 6: setup time = 12 and hold time = 0**. In this case, there is metastability in the *indicator stage* due to setup time violation for all PUTs considered, and there is metastability in the *follower stage* due to setup time violation for PUTs 9-10. There is no metastability in the *preceder stage*. Thus there is metastability in 1 or 2 stages for the PUTs considered, and it could be just in the *indicator stage*, or in the *indicator stage* and the *follower stage* due to violation of setup time.

A few inferences can be made from the above example. Firstly, irrespective of the values of setup and hold times, there will be some PUTs for which metastability can occur in the MVDL (since both setup and hold time cannot be zero in a practical case). Hence there will be need for a metastability detector in the flip-flops to indicate if there is any metstability during delay measurement of a given PUT. The metastability detector can be a simple one like the one used in [32]. If the value of t_{res} is less than the setup time or the hold time, then there can be metastability in multiple stages. If both setup time and hold time are greater than zero but less than t_{res}, then there is possibility of metastability in 1 or 2 stages, i.e., either the *indicator stage* or the *preceder stage* or both can have metastability. If the setup time is zero or negative, but hold time is greater than zero but less than t_{res}, then there is a possibility of metastability in at most one stage, namely the *preceder stage*. If the setup time is greater than 0 and less than t_{res}, while the hold time is 0 or negative, then there is a possibility of metastability in at most one stage, namely, the *indicator stage*. The last kind of flip-flop is the most common type in ICs [126].

With this example and subsequent analysis, it can be safely said, that the design of MVDL should be done with flip-flops that either have 1) hold time greater than 0 and less than t_{res} and setup time equalling 0 or negative or 2) setup time greater than 0 and below t_{res}, and hold time 0 or negative. In case either setup time of hold time have negative values, the absolute magnitude should be less than t_{res}. Under these circumstances, there will either be no metastability in the MVDL or there would be a metastability in 1 stage only, and that stage will be either 1) *preceder stage* or 2) *indicator stage.* The second type of flip-flop, i.e., setup time greater than 0 and hold time 0 or negative is more common [126].

Once the selection of a flip-flop is done, then the metastability issue can be handled in different ways. Suppose the flip-flop with setup time greater than 0 and less than t_{res}, and hold time 0 or negative is selected. In that case, if there is metastability, the metastability detector will detect the stage where it occurs. One can either consider the stage with metastability, and treat it as the *indicator stage*, or ignore the stage with metastability, and consequently consider the *follower stage* to be the *indicator stage.* However, either one of the strategies has to be decided upon beforehand, and consistently adhered to, both during calibration to determine resolution, and all subsequent measurements, irrespective of the *indicator stage* having or not having metastability for any particular measurement. That is to say, for this kind of flip-flop and the aforementioned delay measurement, one either needs to consistently treat the first stage storing a logic value 1 or having metastability as the *indicator stage*, or consistently treat the *follower stage* as the *indicator stage.*

Another factor that can be inferred from the above example and subsequent analysis is that the probability of metastability increases proportionately with increase in setup time as a percentage of t$_{res}$ (for the case using the most common flip-flop, i.e., the one with zero hold time and non-zero setup time, with setup less than t$_{res}$). That is to say, if hold time is 0, and if setup time is 80% of t$_{res}$, then metastability occurs almost 80% of the time, and if setup time is 90% of t$_{res}$, then metastability occurs 90% of the time. Thus the probability of metastability increases as the setup time gets closer to t$_{res}$. Of course, if the hold time is negative, then it also plays a role in the metastability of the indicator stage, as is seen in case 4, where all PUTs have metastability in the *indicator stage*. An analogous reasoning can be made for hold time.

Until now, explaination of the MVDL assumed a rising transition on the input and output of a non-inverting PUT. As has been mentioned earlier, the worst case delays in paths could be due to a falling transitions, and/or the PUT may be an inverting PUT. Hence all such possible combinations need to be handled during delay test and debug. The MVDL scheme can handle all such combinations and it uses a multiplexer-inverter arrangement before the x and y inputs as shown in Figure 4.2 to do so.

Consider the case where the transition at the input of the PUT is a falling one, and the PUT is non-inverting. As before, all the flops are RESET, i.e., initialized to logic value 0 before delay measurement. The input and output of the PUT are routed into x and y respectively, through the multiplexer-inverter arrangement. However, instead of feeding them in as they are into the MVDL, the multiplexer selects the inverted version of input and output of

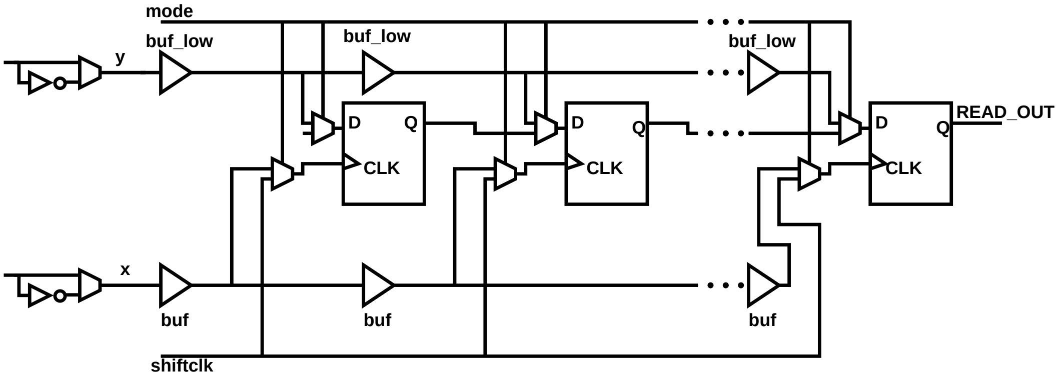


Figure 4.2: *Modified Vernier Delay Line* with capability to handle all possible paths and transitions

PUT since there is a falling transition on both. Since both signals undergo the same additional delay, i.e., an inverter and a multiplexer, the relative separation between them remains the same, causing no error in delay measurement. Now the inverted version of the signals, which are nothing but rising transitions, are fed into the MVDL, and the delay measurement takes place exactly as before. The *indicator value* is also the same as for the PUT described at the beginning of this section (non-inverting with rising transition at the input and output), i.e., logic value 1.

Now consider the case of an inverting path with a rising transition at the input. Inverting paths can be handled using the same circuit shown in Figure 4.2, by just changing the initialization mechanism. The MVDL is initialized for measuring propagation delays of inverting paths by SETing all the flops, i.e., storing a logic value 1 in them. Now the input and output are fed in as they are (i.e., uncomplemented) through the multiplexer-inverter arrangement into x and y inputs of the MVDL respectively. The time difference between the signals gets reduced at every stage by t_{res} as before. However the difference here is that the *indicator stage* is the one in which the flip-flop latches in a logic value 0, and all subsequent stages will latch in a logic value 0. Hence logic value 0 is the *indicator value* for inverting paths.

The second possible case for inverting paths is when the input is falling and the output is rising. Since the input is being used to clock the flops in the MVDL which are positive edge triggered, and we intend to use a single type of MVDL for all possible cases, we have to feed in the input as a rising edge. So the multiplexer selects the inverted version of input and output of PUT before feeding them through multiplexer-inverter arrangement into x and y of MVDL respectively. Thus relative delay difference remains the same again. Since the path is inverting, the MVDL is initialized by SETing it. Now all the flops initially store a logic value 1, so the path delay is indicated by the presence of the first flop which latches in a logic value 0.

Path type	**Input Transition**	**x**	**y**	**Initialization**	**Indicator Value**
Non-inverting	Rising	Non-inverted	Non-inverted	RESET	1
Non-inverting	Falling	Inverted	Inverted	RESET	1
Inverting	Rising	Non-inverted	Non-inverted	SET	0
Inverting	Falling	Inverted	Inverted	SET	0

Table 4.2: Handling all possible paths and transitions using the same MVDL

Table 4.2 summarizes the techniques explained above for handling all possible transitions and paths. The control logic for the multiplexer is very simple. It just selects the inverted version of the input of PUT when it is falling, and non-inverted version when input of PUT is rising. In Table 4.2, the x and y columns show the relation of the x and y inputs of the MVDL with the input and output of the PUT respectively. For inverting paths, initialization is done by SETing the MVDL and for non-inverting paths, initialization is done by RESETing the MVDL. The above explaination and summary given in Table 4.2 pertains to an MVDL designed using positive edge-triggered flip-flops. A similar strategy will be required for negative edge-triggered flip-flops, wherein, the input of the PUT would be fed as is into the x input of the MVDL if the transition at input of the PUT is falling, and the the input of the PUT would be inverted and fed into the x input of the MVDL if the transition on the input of the PUT is rising.

The issue of reading out the *indicator values* stored in a MVDL for a PUT is addressed by using a readout scheme which has minimal hardware and pin overhead. As shown in Figures 4.1 and 4.2, the data input to every flip-flop comes through a multiplexer, which in turn gets its inputs from a corresponding tap point in the upper buffer chain, and the output of the flip-flop of the previous stage. Similarly, the clock input of every flip-flop comes through a multiplexer, which in turn gets its inputs from a corresponding tap point in the lower buffer chain and a clock signal *shiftclk*. This kind of topology is like a scan chain, and once delay measurement for the PUT has been completed, the values stored in the flip-flops can be shifted out using this scan chain like arrangement, and the number of *indicator value* counted to determine measured delay of the PUT. The *READ_OUT* outputs of multiple MVDLs can be multiplexed and connected to a single output pin.

In earlier discussion on the MVDL, it has been mentioned that all the flip-flops need to be either SET i.e., initialized to logic value 1 or RESET, i.e., initialized to logic value 1. The scan chain like facility enables this to be done very easily. Hence, whenever the results of delay measurement of a PUT is being scanned out, a sequence of all 0s or all 1s can be scanned in to perform initialization for delay measurement of the next PUT.

As has been mentioned earlier, the resolution t_{res} of the MVDL or any similar balanced delay line scheme is susceptible to variations. This warrants a calibration technique to determine t_{res} before the MVDL can be used for timing characterization. In [46], the problem of variability in resolution is handled by utilizing a sophisticated scheme, wherein voltage controlled delay buffers are used in the upper buffer chain, and the bias voltage to these buffers is generated using a Delay Locked Loop (DLL). The basic idea is to ensure that the product of the average resolution and number of stages of the delay line (this product is called "dynamic range" in [46]) equals a predetermined value, which in turn is the delay difference between two signals. In other words, the resolution of the *Vernier Delay Line* was forced to be such that the delay difference between the two calibration signals equals the "dynamic range" of the *Vernier Delay Line*.

However, for timing characterization, only determining the value of the average resolution is sufficient, and there is no need to set the resolution to some specific value. Hence, the same two signals separated by a pre-determined value, as used in [46] can be utilized to calibrate the MVDL, and the *indicator stage* determined for measurement of the delay difference between the two signals. Based on this, average t_{res} can be determined for the MVDL. In [52], such a technique has been applied to calibrate a balanced delay line, wherein, two signals that are separated by a magnitude equalling the maximum propagation delay measurable by the delay line are used, and the resolution is averaged of the number of stages of the delay line.

One issue with the above calibration technique is that it implicitly assumes all stages to have identical resolution. This may not be the case due to mismatches in loading, and intra-die variability issues in DSM technologies. In [35] a suggestion was made to calibrate each individual stage for a delay line used for delay measurement, and a technique was proposed for the same, which will be presented later in this chapter. A similar technique can be applied here, wherein each stage will be calibrated individually using signals that are separated deterministically. A technique for generating such signals was presented in [21] for testing of the *Vernier Delay Line* presented in that paper, and consisted of an on-chip voltage controlled delay cell to generate phase delay. In case a digital circuit is desired, a digital variable delay cell like the one presented in [129] could be used.

Due to the "observability only" nature of MVDL, the process of timing characterization using MVDL is similar to that that using scanout chain [97],

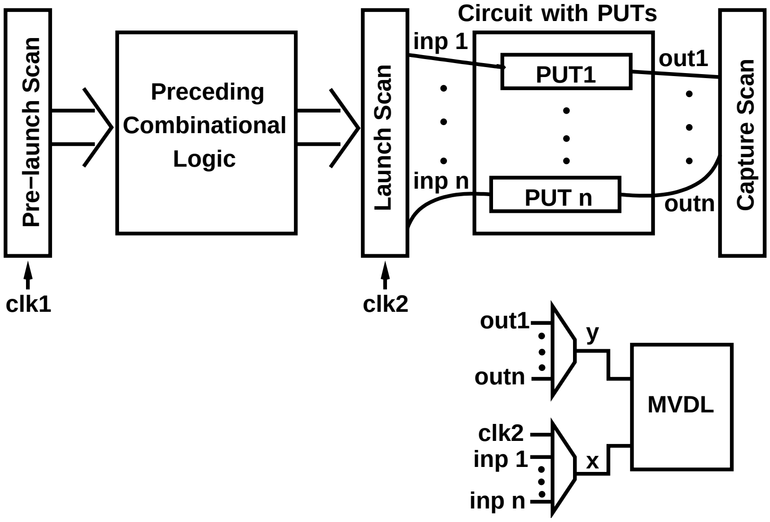


Figure 4.3: Timing characterization using MVDL in a scan based environment

[101]. Figure 4.3 shows the use of MVDL for timing characterization in a scan based environment. The PUTs to be tested are in between the *Launch Scan* and *Capture Scan*. The pattern application scheme can be *skewed-load transition test* [117], [118], *Tri-Scan* [36] or *broad-side delay test* [119] (for the last one, *Pre-launch Scan* will also be used). The delay of the path is measured by routing the tap points to the x and y inputs of the MVDL. For measuring propagation delay of the combinational portion, *inp1 ... inpn* is selected to be the x input of the MVDL, whereas for measuring propagation delay of the entire path, *clk2* or the system clock at the launch flop of the PUT is selected to be the x input of the MVDL. In both cases, the output of the corresponding PUT, i.e., *out1 ... outn* is selected to be the y input of the MVDL. Every time the delay measurement for a path is done, the values stored in the MVDL need to be scanned out, and this process can be coupled with the scan in of

the initialization values for testing the next PUT, i.e., scanning in all 0s or all 1s. To reduce test time overhead due to the scan operation, the scanning out of values from the MVDL can in turn be coupled with scanning in of the test vector to be applied to the PUT. For timing characterization using functional tests [101], there will be some latency between two consecutive tests using the same MVDL, to facilitate scanning out of values for delay measurement of the last PUT, and initialization of the MVDL for testing the next PUT.

Since small increases in propagation delay of a PUT can be detected using MVDL, gross as well as small delay faults can be detected. Additionally, for silicon debug, the magnitude of the delay fault can be obtained as follows

$$|n_e - n + 1| * t_{res} < FaultSize < |n_e - n| * t_{res} \tag{4.2}$$

where n_e is the indicator stage for the expected value of the delay of the PUT and n is the *indicator stage* for delay measurement of the PUT using MVDL.

4.2 Delay Scan Chain

The MVDL module presented in the previous section is a low overhead technique that can be used for delay test and debug. One interesting thing about the MVDL is that it has a bank of flip-flops connected in the form of a scan chain. Now, scan chains are a common feature in modern ICs, and if it can be reused for delay measurement, then the the total overhead at the chip level could possibly be much lower than MVDL, since we are now reusing existing DFT infrastructure for delay measurement. This is the motivation behind the *Delay Scan Chain* presented in this section, which is a slightly modified version of a regular scan chain, and can be used for on-chip delay measurement based timing characterization of critical paths in an IC.

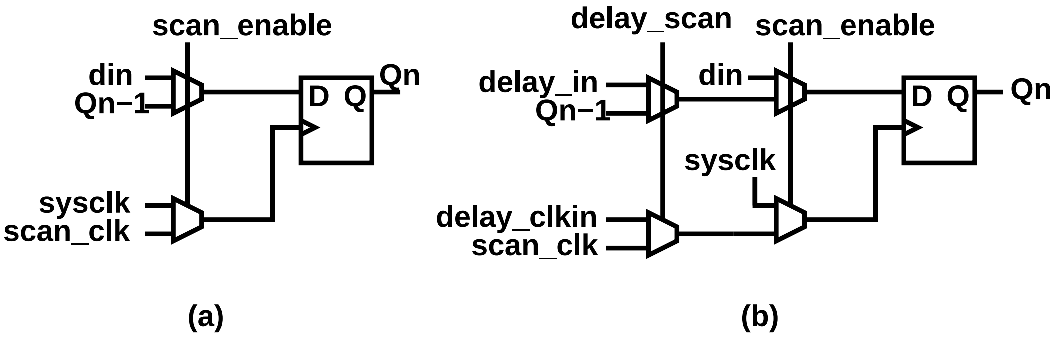


Figure 4.4: a) Regular scan flip-flop and b) DSCAN flip-flop

The fundamental structure within a *Delay Scan Chain* is a flip-flop called *Delay Scan* flop or DSCAN flop. A regular scan flop is shown in Figure 4.4a and a DSCAN flop is shown in Figure 4.4b. A regular scan flop has 2 modes of operation. It operates in the regular or functional mode when the *scan_enable* signal is at logic value 0, during which the scan flip-flop receives data from the combinational logic through input *din* as shown in Figure 4.4a. The flip-flop operates in the test or scan mode when *scan_enable* signal is at logic value 1, during which, the scan flip-flop received data either from the previous stage scan flip-flop in the scan chain, or from the scan-in input of the chip.

The DSCAN flop is obtained by adding a second level of multiplexers

scan_enable	**delay_scan**	**operation**
0	X	Regular D flip-flop
1	1	Regular scan flip-flop
1	0	DSCAN flip-flop for delay measurement

Table 4.3: Modes of operation of DSCAN flip-flop

before a regular scan chain, as shown in Figure 4.4b. These multiplexers are added in such a way, that they do not cause any additional delay to the data path feeding into the scan flip-flop through the *din* input. The second level of multiplexers are controlled using a separate signal called *delay_scan*. The DSCAN flop has three modes of operation and these are shown in Table 4.3. When the *scan_enable* signal is at logic value 0, the DSCAN flop operates like a regular flop, i.e., in functional mode, irrespective of the value of the *delay_scan* signal, and receives data from the datapath through *din*, and is clocked by the system clock or *sysclock*. When the *scan_enable* signal is at logic value 1, then there can be two modes of operation of the DSCAN flop, based on the value of the *delay_scan* signal. If the *delay_scan* signal is at logic value 1, then the DSCAN flop works in scan mode, wherein it receives data from the DSCAN (or regular scan) flip-flop of the previous stage, or from the scan-in input of the IC, through the Q_{n-1} input, and is clocked by the scan clock *scanclk* (here n-1 represents the number of the flip-flop that precedes the current flip-flop whose number is n, in the scan chain under consideration, and Q_{n-1} representes the output of the preceding flip-flop). However, if the *delay_scan* signal is at logic value 0, then the DSCAN flop operates in delay measurement mode. In this mode, when the DSCAN flop is placed in between two buffer chains and receives inputs from different tap points within the buffer chains, as a flip-flop did in the MVDL scheme. The tap point from the upper buffer chain feeds into the DSCAN flop through the *delay_in* input and the tap point from the lower buffer chain feeds into the DSCAN flop through the *delay_clkin* input.

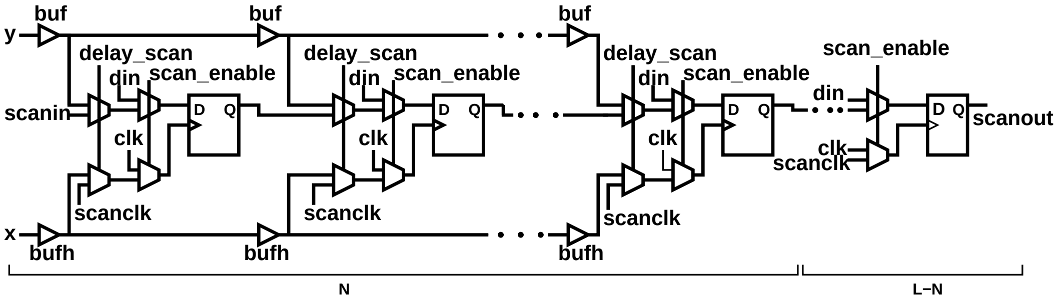


Figure 4.5: *Delay Scan Chain*

The *Delay Scan Chain* scheme is shown in Figure 4.5. Selective groups of consecutive regular scan flip-flops are replaced using DSCAN flip-flops in the figure. In Figure 4.5 N represents the number of flip-flops that have been converted to DSCAN flip-flops, and L represents the length of the scan chain. This group need not be in the beginning of the scan chain. Instead, they should be placed at positions which maximizes the number of PUTs that are close to them. The input of the PUT whose delay is to be measured is fed into the x input, and the output is fed into the y input. Individual buffers *buf* in the lower buffer chain have higher than corresponding buffer *buf* in the upper buffer chain. The *Delay Scan Chain* performs delay measurement exactly like the MVDL scheme, and is used for delay measurement by operating the

DSCAN flip-flops in delay measurement mode as explained above. In order to enable handling of all kinds of paths and transitions, a multiplexer-inverter arrangement just like the one in the MVDL can be placed before the x and y inputs of the *Delay Scan Chain*. The values stored in the DSCAN flip-flops can be shifted out and initialization values can be scanned in just as it would be done during a regular scan operation. The main advantage of this scheme over MVDL is that existing DFT infrastructure in a chip is being reused with slight modification. However, the drawback here is that the DSCAN flops also form state holding elements during normal operation of a chip. Consequently, the *Delay Scan Chain* scheme can be used only when the chip is in test mode and cannot be used in conjunction with functional testing techniques like [101], unlike the MVDL.

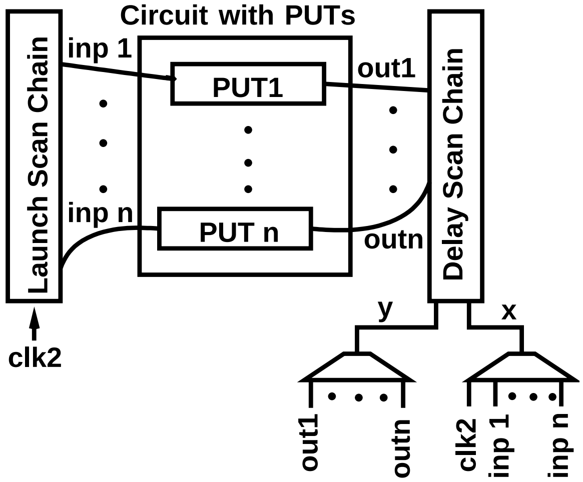


Figure 4.6: Timing characterization using *Delay Scan Chain*

The proposed timing characterization scheme using *Delay Scan Chain* is illustrated using Figure 4.6. Since the *Delay Scan Chain* is not an "observability only" circuit, the timing characterization scheme would be more complex, and would vary depending on the pattern application scheme. Let us assume that in Figure 4.6 the pattern application scheme is one that enables complete accessibility, like *Tri-Scan* [36]. Then, for timing characterization purposes, initialization vector can be scanned in into the launch scan chain, and the CUT initialized. However, initialization of the *Delay Scan Chain* will take place when the transition vector is being scanned into the launch scan chain. Subsequently, the *Delay Scan Chain* will be switched from scan mode to delay measurement mode, and delay of the PUT measured. This delay could be measured considering the clock signal (*clk2* in Figure 4.6 as input), or input of the PUT, in case delay of only the combinational portion of the PUT needs to be measured.

In case the pattern application scheme is *skewed load transition test* [117], [118] or *broad-side delay test* [119], the initialization of the *Delay Scan Chain* needs to be done when the scan-in operation of the intialization vector V_1 for the CUT is taking place. Subsequently, when V_1 is applied and the transition vector V_2 is generated, the *Delay Scan Chain* must be kept in regular scan mode. Once V_2 is applied to the CUT, the *Delay Scan Chain* is switched from regular scan mode to delay measurement mode, and the propagation delay of the PUT measured. The multiplexer in front of the *Delay Scan Chain* is used to select the path whose delay is to be measured, and will do so when V_2 is applied to the CUT.

4.3 Skewed Inverter Delay Line

The delay measurement schemes presented earlier in this chapter suffer from a major drawback in that they require balanced routing across long distances, i.e., from the PUT to the delay measurment unit. This is because the input and output of the PUT will have to be routed to the module, and that would require ensuring that signals from the input and output of the PUT are delayed by an exact amount when they reach the delay measurement module. This would require an extremely careful custom layout process, which may not be viable for Design for Test (DFT) and Design for Debug (DFD) structures. Also, in presence of intra-die variations, the signals could traverse through regions that lie in different process corners, leading to potential inaccuracies.

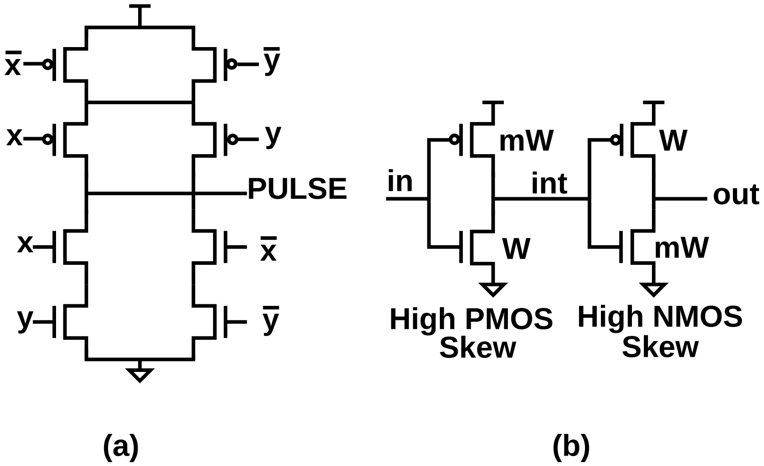


Figure 4.7: a) Pulse generator b) Pulse shaper

In this section, a scheme called *Skewed Inverter Delay Line* (SIDL) is presented, that overcomes the drawbacks of existing on-chip delay measurement schemes for timing characterization. The SIDL scheme is shown in Figure 4.8, and consists of a pulse generator and an *Integrated Pulse Shaping and Latching* (IPSL) circuit. The pulse generator is shown in Figure 4.7a, and is basically an XOR gate. The IPSL circuit consists of multiple stages, with each stage having a pulse shaper driving the *Latch Enable (LE)* input of a level triggered latch of that stage. In addition, the latches in each stage have a multiplexer at their data input, with one input of the multiplexer connected to a stable logic value 1, and the second input connected to either the output of the previous stage or in case of the first stage, to an external *scanin* pin. The output of the latch of the final stage is connected to an external *scanout* pin.

The pulse shaper used in the IPSL circuit is shown in Figure 4.7b. It consists of two inverters, the first one being a *High PMOS Skew* inverter, i.e., the PMOS width m times the NMOS width, followed by a *High NMOS Skew* inverter. If a pulse is fed into the input of such a pulse shaper, then the falling edge of the input pulse causes the intermediate node *int* to be pulled up very fast due to the *High PMOS Skew* inverter, which in turn causes the output of the pulse shaper *out* to be pulled down very fast, since the NMOS of the second inverter gets switched on faster, and also because it is a *High NMOS Skew* inverter. Thus the falling edge of the input pulse is propagated much faster to the output of the pulse shaper than its rising edge, which causes the pulse at the output of the pulse shaper to shrink as compared to the pulse at its input.

The skew ratio controls the magnitude by which the pulse is shrunk, and if a greater skew ratio is used, i.e., m is made larger, then the pulse will be shrunk by a greater amount. However, if the skew ratio is made too high, then there is a higher possibility of unbalanced PMOS and NMOS variations, the impact of which is discussed later in this section. Additionally, in Figure 4.7b, the skew ratio m of the *High PMOS Skew* inverter is shown to be same as that of the *High NMOS Skew* inverter. This is done to mitigate the impact of unbalanced PMOS and NMOS variation on a pulse.

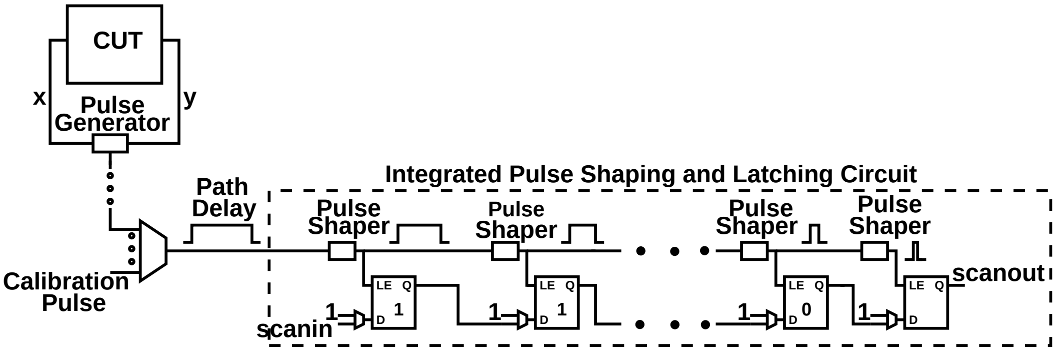


Figure 4.8: *Skewed Inverter Delay Line* scheme

The working of the scheme is explained by considering the PUT to be a non-inverting path in the CUT shown in Figure 4.8, with a rising transition at the input of the PUT. The input and output of each PUT is fed into its corresponding pulse generator, shown by x and y inputs in Figure 4.8. Delay fault testing requires application of an initialization vector V_1 and a transition vector V_2. Hence, in order to test for a rising transition on this PUT, V_1 will initialize the PUT input and output to logic value 0. Subsequently, when

V_2 is applied to the PUT, there will be rising transition at its input, which will cause the output of the pulse generator to go to logic value 1. When this rising transition propagates to the output of the PUT, the output of the pulse generator will be pulled down to logic value 0. This process will result in the pulse generator producing a pulse whose width equals the delay of the PUT. If the PUT is an inverting one, an XNOR gate will be used as the pulse generator.

This pulse is transmitted through an interconnect to the IPSL circuit for the SIDL. Since the pulse is transmitted through a single line, there is no need for carefully balanced routing when transmitting the pulse to the IPSL circuit, thus eliminating the need for careful custom layout, as well as reducing the possibility of inaccuracies due to process variations. The width of the pulse should remain fairly constant (In the simulation experiments carried out in this work, the tolerance of pulse width variation was limited to 5ps, and this number is about 1% of the cycle time of a commercial processor reported in the nearest technology [92]. However, the tolerance level could be higher than this.) even when the pulse is transmitted across regions lying in different process corners. This is because most process variations will speed up or slow down the rising and falling edge of the pulse equally, thus keeping the relative distance between them, and consequently the pulse width, constant.

An exception to this situation occurs when the interconnect through which the pulse is transmitted to the IPSL module is either too long for the source driver, or has intermediate inverters having unbalanced PMOS-NMOS variations, i.e., the process variations that affect PMOS and NMOS differently, and cause the pulse to shrink. The way around this problem would be to eliminate the use of intermediate inverters. Simulation results shown later in this chapter support this solution by demonstrating that the pulse can be transmitted with no change in width, over a long distance (about 500μm) without need of intermediate buffers, and without the source driver being too large. Also, if inverter pairs are used for intermediate buffering, then the some of impact on the pulse due to unbalanced PMOS-NMOS variation in one of the inverters would be offset by the similar unbalanced variation in the second inverter of the pair. Another issue that can affect the integrity of the pulse during transmission is that of crosstalk that has an unbalanced impact on the pulse. However, since this scheme is used during testing of a chip, when the chip is relatively quiescent, crosstalk is not anticipated to be an issue.

The latches in the IPSL circuit are all initialized to store a logic value 0 before running the delay measurement test. This is done by using the scan chain like structure of the latches of the IPSL circuit, wherein a logic value 0 is scanned into all the latches. The pulse, whose width equals the delay of the PUT is multiplexed into the IPSL and is chopped (i.e., it width shrunk) before it drives the *LE* of the first stage latch. If the chopped pulse is large enough, then a logic value 1 gets latched into the latch of the first stage of the IPSL. The pulse is chopped progressively by pulse shapers of the subsequent stages as it travels down the IPSL circuit, and at each stage, after shrinking the pulse, the pulse shaper feeds it to the *LE* of the latch of that corresponding stage. The first stage where a logic value 1 is not latched in, due to the pulse width at the *LE* input being too small, indicates the delay of the PUT and is called the *indicator stage (n).*

The amount by which a pulse is shrunk in each stage is called the resolution (t_{res}) of that stage, and depends on the difference between the pull-up and pull-down times of the skewed inverters of that stage. It is given by the following equation

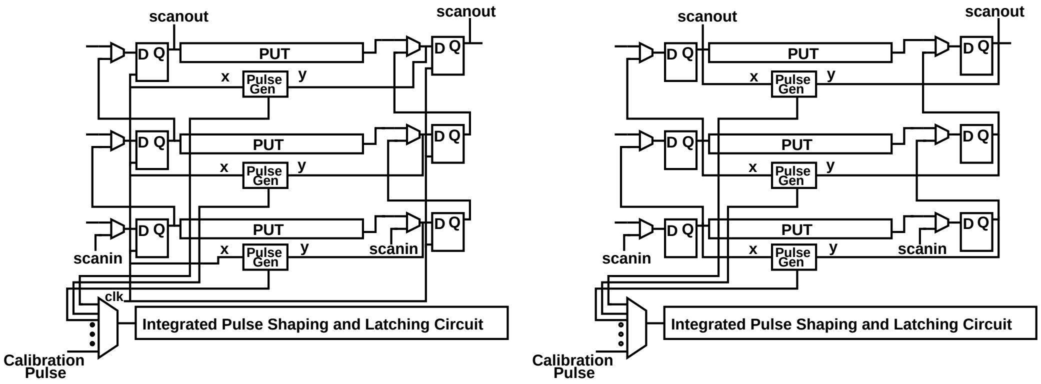
$$t_{res} = (t_n - t_p)_{HighPMOSSkew} + (t_p - t_n)_{HighNMOSSkew}$$

where t_p and t_n are the pull-up and pull-down times through the PMOS and NMOS transistors of that particular inverter. Hence the minimum resolution, and consequently the minimum propagation delay/delay fault that can be measured is limited by the minimum delay difference achievable between pull up and pull down transistors in a given technology.

The hold time of the latches (t_{hold}) can have an impact on t_{res}, if it is lower than the hold time. If t_{res} is slightly lower than t_{hold} then there could be an error of a single stage in delay measurement. However, if t_{res} is significantly lower than hold time of latches, then there could be a error of multiple stages. Thus, t_{hold} places another lower bound on the minimum achievable resolution, and a metastability detector would be needed just as suggested for MVDL and *Delay Scan Chain* to detect metastability in the latches. The *indicator stage n* stage gives the delay of the PUT as a range between (n-1)*t_{res} and n*t_{res}.

The resolution of the pulse shaper of each stage is theoretically the same, if identical pulse shapers are used. However, variations in process parameters and loading will cause this resolution to vary. Hence before using this scheme for timing characterization, there needs to be a calibration phase. The input multiplexer to the SIDL, shown in Figure 4.8 has a *Calibration Pulse* input for this purpose. The calibration process involves generating a pulse from two signals, the phase difference between which can be adjusted deterministically, i.e., the delay between them can be measured. The width of this calibration pulse is increased till it is wide enough to ensure that stage 1 of the SIDL is the *indicator stage*. The measured delay between the phase shifted deterministic signals which generates this size pulse is the resolution of stage 1 (t_{res1}) of the SIDL. Now the two phase shifted signals are separated further to generate a pulse that is wide enough to ensure that stage 2 of the SIDL is the *indicator stage*. The delay between the phase shifted signals which generates this size pulse is t'_{res2}, and the resolution of stage 2 (t_{res2}) is the difference between t'_{res2} and t_{res1}. In this manner, the resolution of all stages of the SIDL can be determined, and this kind of calibration will ensure robustness of the scheme to process variations. If the width of the calibration pulse changes while it is being routed to the SIDL, then stage 1 is used as a zero point, and resolution of subsequent stages is determined with respect to resolution of stage 1.

Additionally, the *Calibration Pulse* input can also be used during the initialization of the the SIDL latches. A very wide pulse will be sent in through this input so that the *LE* of all the latches is at logic value 1, enabling the scanned values, i.e., logic 0 values, to be captured into the latches.



(a) (b)

Figure 4.9: Timing characterization using SIDL

Figure 4.9 shows the proposed timing characterization scheme using SIDL in a scan based environment. Each PUT has its own pulse generator, and the output of multiple pulse generators is multiplexed into an IPSL module. The pulse generator is very small, i.e., a single XOR or XNOR gate, so it can be placed very close to the PUT, thus precluding the need for routing of signals across long distances. The input and output of a PUT are actually the launch clock and input to the capture storage element respectively. In case of edge triggered storage elements, these two signals have to be routed to the pulse generator as shown in Figure 4.9a. However, if level sensitive latches are used as storage elements, then, for long paths only, one can select the input of the PUT to be the output of the launch latch, and output of the PUT to be the output of the capture latch of the PUT, as shown in Figure 4.9b. Since almost all of the latches used in a chip are identical, this measured delay would represent the entire delay of the PUT, i.e., delay through flop plus the delay through the combinational logic. This solution would however be ineffective if the path delay increases so much that it misses the capture level of the capture flop. Also, if intra-die process variations are significant, then we would have to revert to the original method of using launch clock and input of capture latch as inputs to the pulse generator for accuracy.

Any existing pattern application technique can be applied to the PUT and its delay measured as per the above scheme, due to the "observability only" nature of SIDL. If the delay of the PUT is larger than what was anticipated, then there is a delay fault, the magnitude of which will also be known. In this manner, all small delay faults can also be detected and hence any part that has any defect or parametric failure mechanism, and would have escaped normal speed testing, would be identified. Additionally, since the magnitude of the delay fault is known, debug will be much easier, since the amount of rectification required before future silicon spins will be known.

In addition to eliminating the need for balanced routing over long distances, the SIDL scheme will also have significantly lower overhead than the schemes presented in [38], [39], [111] (these schemes were also discussed in the earlier sections and Chapter 1). This is because it will require routing of only one wire instead of 2 from each PUT to a delay measurement module, and also, since it has a single delay line per module as compared to two, used in [38], [39]. This SIDL could also possibly be used for adaptive design schemes that are based on on-chip delay measurement, an example of which is presented in [89].

The use of a pulse shrinking delay line for delay measurement has reported earlier in the literature. In [106], [108], [109], [110] pulse shrinking delay lines were used for delay measurement in physics and communication applications. All of them require addressing of individual latches in the delay line, and global reset signals which are needed to reset the SR latches used in these papers, making them expensive in terms of hardware overhead. The scan like structure in the SIDL overcomes these drawbacks, and facilitates easy initialization (reset) as well as easy readout of the results, thus enabling efficient timing characterization, i.e., delay fault testing and silicon debug. Additionally, it was stated in [26] that the scheme presented in [110] has limited accuracy due to mismatches since it requires two identical pulse shrinking delay lines. In [24], [25], [26], [131] pulse shrinking based delay measurement techniques were presented that enable easier readout than in the delay measurement techniques in [106], [108], [109], [110]. However, the schemes in [24], [25], [26], [131] require establishing a reference value against which the measured delay is compared. This is problematic for timing characterization since it would require establishing a large number of reference values. The SIDL actually measure the delay of the output of a PUT, referenced to its input, making the unit a self-referenced one. Additionally, the counter in [24], [25], [26], [131] would require a very fast clock, which may not always be practical.

4.4 Simulation Results and Discussion

The MVDL module was designed in 0.18μm technology [5] and simulation based delay measurement experiments were carried out to validate the operation of the scheme and determine the number of stages required for measuring delay of top critical paths of different sized Wallace [137] and Dadda [31] multiplier circuits. The top critical paths of completely combinational versions of these multipliers were extracted using a commercial static timing analysis tool (STA) [128], and non-inverting version of the paths were used. The average resolution was calculated by measuring delay of the first test circuit, and it was approximately 97ps. In all the delay measurement experiments (including the one used for calibration), the stage with metastability was ignored, and hence the *indicator stage* for delay measurement was actually the *follower stage* the detailed explaination for which has been provided earlier in this chapter. The results of delay measurement of the critical paths of different sized multipliers are shown in Table 4.4. In almost all cases the MVDL measures delay accurately. However, the averaging of resolution can cause an error in some cases, especially ones requiring a large number of stages.

In Table 4.4, we can see that in some cases a very large number of stages are needed. However, in a real chip, a resolution of about 5%-10% of the cycle time can be considered reasonable, and under such circumstances, a 10 stage - 20 stage MVDL would be required. Another 2-4 stages may be needed if delay faults that cause a PUT to have a latency of about 20% more than the cycle time are to be measured for silicon debug. Area of a 12 stage

Test Circuit	**Delay measured with STA**	**Delay measured with MVDL**	
		Indicator stage	**Delay range**
4 bit Wallace	2420	25	2328-2425
4 bit Dadda	1980	21	1940-2037
8 bit Wallace	4320	45	4268-4365
8 bit Dadda	3740	39	3686-3783
16 bit Wallace	7630	79	7566-7663
16 bit Dadda	6870	71	6790-6887
32 bit Wallace	14430	149	14356-14453
32 bit Dadda	12160	125	12028-12125

Table 4.4: Propagation delay measurement of critical paths of test circuits using MVDL

MVDL module is about 0.0029% of a chip with 200mm^2 die area and area of a 20 stage MVDL module is about 0.0047% of a chip with 200mm^2 die area. This area is obtained using an Auto Place and Route tool [124]. Although the MVDL module itself is compact, the real overhead of this scheme would come from routing of inputs and outputs of different PUTs to an MVDL. This overhead is dependent on the number of paths that need to be measured, number of MVDLs that would be utilized in a chip as well as their placement within the chip, and hence, is hard to estimate. This problem is actually a combination of path selection and floorplanning and routing problems, and would be hard to determine without actually having the complete layout of a prototype commercial chip with MVDL.

To analyze the behavior of MVDL and obtain an approximate idea of the possible resolution across different technologies, an experiment was run wherein the delay of a 10 stage inverter chain was measured using a MVDL at 4 different technologies, namely 0.18μm, 0.13μm, 0.10μm and 0.07μm [19], using nominal process corners and at 25°C operating temperature. The high delay buffers that constitute the lower buffer chain were made up of a chain of 4 inverters and the low delay buffers were made up of 2 inverters, thus giving a gross resolution of two inverters. The same MVDL and test circuit was used for running HSPICE simulations using the four different technology files, and just the scaling factor and supply voltage were changed. The flip-flops however were designed individually, because setup and hold times of flip-flops do not necessarily scale proportionately with technology. In certain cases additional inverters had to be introduced in the flip-flops.

Tech.	**Delay using HSPICE simulation (ps)**	**Supply Voltage (volts)**	**Indicator Stage**	**Resolution (ps) (ps)**	**Delay measured using MVDL (ps)**
0.18μm	281	1.8	5	59	236-295
0.13μm	213	1.3	5	45	180-225
0.10μm	144	1.0	5	30	120-150
0.07μm	124	0.9	5	26	104-130

Table 4.5: Behavior of MVDL at different technologies

Table 4.5 shows the results of this experiment. In column 1 the four technologies are listed, and the corresponding supply voltages are listed in column 3. The delay of the inverter chain test circuit obtained using HSPICE simulations is listed in column 2, and the *indicator stage*, i.e., the stage which is the first to capture in a logic value 1 is listed in column 4. The average resolution of the MVDL in this case is basically the average of individual resolution of the first five stages, and is listed in column 5. Based on this average resolution and the *indicator stage*, the value of delay measured for the inverter chain test circuit is obtained, and that is listed in column 6. It can be seen that the MVDL circuit tracks variations in technology well, and the *indicator stage* remains constant across different technologies. This is because the delay of the test circuit as well as the resolution of the MVDL scale proportionately. For ensuring robustness to variations, appropriate calibration needs to be done to determine this resolution. Also, we can see that even with a gross resolution equalling two inverters, the actual value of resolution is quite low, ranging from 59ps in 0.18μm technology to about 26ps in 0.07μm technology. This is indicative of the low resolutions of delay measurement possible using MVDL. In real hardware, one can lose some resolution due to unwanted loading or other effect, but at the same time, can get finer resolution by means of buffer sizing instead of using the gross two inverter resolution presented here. The flip-flop used for this experiment was a pulsed latch [126], [134].

The *Delay Scan Chain* scheme is basically obtained by placing another level of multiplexers before the MVDL scheme, and hence during simulation, behavior of the scheme is similar to the MVDL, and so is the resolution, since the loading on the buffers is the same. This is illustrated in Table 4.6, where the propagation delay of the non-inverting versions of critical paths of different sized Wallace [137] and Dadda [31] multipliers as well as those of some ISCAS benchmark circuits and 74 series circuits [57] is measured using *Delay Scan*

Chain, using the same assumptions and conditions as those mentioned for the MVDL earlier in this section.

Test Circuit	**Delay measured with STA (ps)**	**Delay measured with Delay Scan Chain**	
		Indicator Stage	**Delay Range (ps)**
4 bit Wallace	2420	25	2328-2425
4 bit Dadda	1980	21	1940-2037
8 bit Wallace	4320	45	4268-4365
8 bit Dadda	3740	39	3686-3783
16 bit Wallace	7630	79	7566-7663
16 bit Dadda	6870	71	6790-6887
32 bit Wallace	14430	149	14356-14453
32 bit Dadda	12160	125	12028-12125
c432	4950	51	4850-4947
c499	1590	17	1552-1649
74181	1500	16	1455-1552
74182	760	8	679-776

Table 4.6: Propagation delay measurement of critical paths of test circuits using *Delay Scan Chain*

Area and power overhead of a *Delay Scan Chain* as compared to a regular scan chain was computed (these overheads do not include overhead due to routing of the PUT to the delay measurement module and overheads due to routing of *delay_scan* signal) for different sized scan chains, with all of them having the first 20 stages as DSCAN flip-flop stages, and the remaining stages as regular scan flip-flop stages. That is to say, area and power dissipation due to use of 20 stages of DSCAN flip-flops in different scan chains ranging in length from 100-2000 was computed and compared to scan chains of corresponding lengths where no DSCAN flip-flop is used. The computed area is basically just the pre-layout transistor area (product of transistor width and length). For computing power dissipation, HSPICE simulations was carried out using smaller length scan chains and *Delay Scan Chains* (40 and 100 stages), and based on these, the power dissipation f or different sized scan chains and *Delay Scan Chains* was calculated. The idea of this experiment and analysis is to provide an estimate of comparative overheads rather than absolute numbers. The simulations were carried out using 0.18μm technology [19], under nominal process conditions, with a supply voltage of 1.8V and at 25°C operating temperature. A sequence of all 1s was scanned into both chains for computing power dissipation. Additionally, for the *Delay Scan Chain*, a transition was created at the input and the output of the upper buffer chain to do a more pessimistic analysis (since the scan operation and the delay measurement operation never take place simultaneously). The results of this analysis are shown in Figure 4.10. It is evident that, although the use of 20 DSCAN stages in scan chains of smaller length like 100-200 stages leads to significant area overheads, this overhead is much lower in more practical length scan chains which are about 2000 long [139], and in those the area overhead per scan chain is about 0.29% and power overhead is about 1.27% at 25°C.

In order to validate the use of the *Skewed Inverter Delay Line* (SIDL) scheme for on-chip timing characterization, the SIDL was designed in 0.18μm CMOS [19] technology, using a chain of pulse shapers, the output of each being connected to the *latch enable* (LE) of a level triggered latch. Each pulse shaper

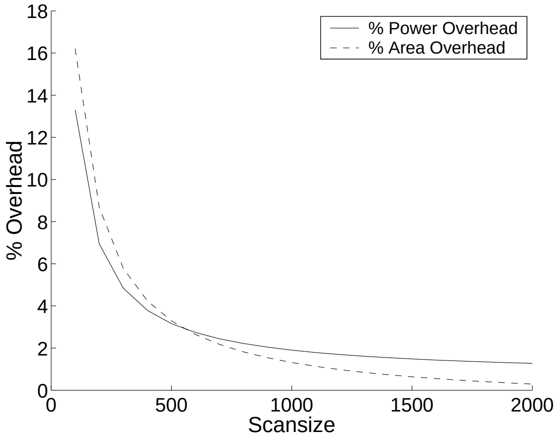


Figure 4.10: Percentage area and power overhead of a *Delay Scan Chain* with respect to a regular scan chain

consisted of a pair of skewed inverters, and the dimension of the all shapers were identical for all simulation experiments. The latch used was a pair of cross-coupled inverters with a transmission gate at the front end, and a logic value 1 was placed at the D input of these latches. The gate terminals of the transistors of this transmission gate get their input from the LE terminal of the latch, through a pair of inverters, with skew ratio similar to those used in the pulse shapers. Delay measurement experiments were run on non-inverting versions of top critical paths of different sized Wallace [137] and Dadda [31] multiplier circuits (extracted using a commercial STA tool). Table 4.7 shows the results of this simulation experiment. This simualtion experiment was carried out under nominal process conditions, with 1.8V supply voltage and 25°C operating temperature. The average resolution for delay measurement was approximately 60ps, and was calculated based on pulse width at output of some stages while doing delay measurement for the first test circuit. It can be seen that SIDL measures delay with reasonable accuracy. In case of 32 bit Wallace [137] and Dadda [31] multipliers, there is an overestimate of one stage. This is because the resolution considered was an approximate average one, and there is a margin of error of about 1-2ps, during HSPICE simulations. Additionally the pulse generator induces some inaccuracy. The cumulative effect of these factors, causes an error in delay measurement of about one stage. This phenomenon is referred to as the *sim-avg effect* in this work. However, during characterization of a chip, each stage will be calibrated to obtain its resolution, and typically there would be 10-20 stages (with t_{res} being 5-10% of cycle time), and this error will not arise. The error induced in delay measurement due to the pulse generator is indicated in column 4 of Table 4.7. This error is very low, ranging from 0.27% to 1.67% for the circuits considered, and should not be an issue in timing characterization. However, if it does become an issue then a margin of error has to used to account for it in a chip.

The area of a single 20 stage SIDL in 0.18μm technology [5] is about 0.0024% of a 200mm^2 chip, about half that of the MVDL. This area is obtained using an Auto Place and Route tool [124]. This is just the area of module, and not of the scheme, which, just like the MVDL will depend upon number of modules used, number of paths being tested and so on, and is outside the scope of this work.

As mentioned before, process variations can affect the pulse width and

Test Circuit	HSPICE Delay (ps)	Pulse Width (ps)	Inaccuracy due to Pulse Generator	Indicator Stage	Delay Range using SIDL ps
4 bit Wallace	1197	1181	1.34%	20	1140-1200
4 bit Dadda	1016	999	1.67%	17	960-1020
8 bit Wallace	1948	1919	1.48%	32	1860-1920
8 bit Dadda	1849	1833	0.87%	31	1800-1860
16 bit Wallace	3674	3658	0.44%	62	3660-3720
16 bit Dadda	3385	3369	0.47%	57	3360-3420
32 bit Wallace	6402	6373	0.45%	108	6420-6480
32 bit Dadda	5976	5960	0.27%	101	6000-6060

Table 4.7: Propagation delay measurement of critical paths of test circuits using SIDL

accuracy of delay measurement in the SIDL scheme if there are intermediate inverters used as buffers, with unbalanced PMOS and NMOS variations that could change the width of the pulse. Hence simulations were run in 0.10μm [19] technology to determine how far the pulse can be routed with only one inverter-pair (pre-driver + driver) as buffer at the output of the pulse generator, while ensuring that pulse width does not vary more than 5ps, which is about 1% of the cycle time of a commercial processor reported in the nearest technology [92]. Figure 4.11, shows the total driver width (pmos+nmos of 2^{nd} inverter) in μm required to route the pulse across different lengths of wire. It is clear that the pulse can be routed across 500μm of wire without the driver size being very high. This would be sufficient to meet the needs for timing characterization in most processors. The interconnect model used was a lumped pi model, with resistance r=0.7 ohms/um and capacitance c1 and c2 = 0.12fF/um.

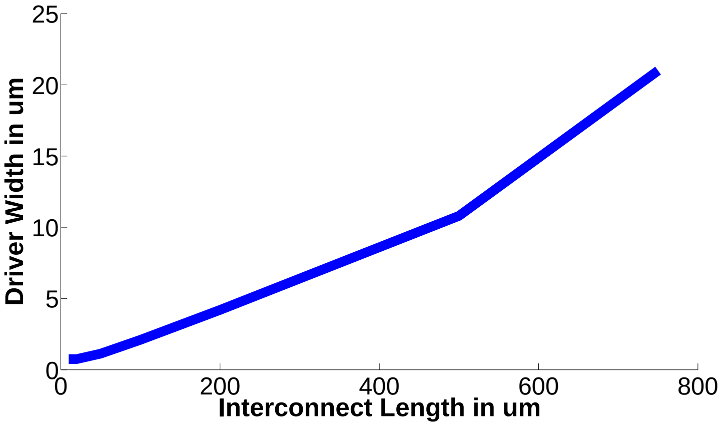


Figure 4.11: Driver size vs. interconnect length for the SIDL scheme

In order to analyze the response of the SIDL scheme to technology variations and obtain an idea of the approximate average resolution at different technologies [19], the critical path of the 4 bit Wallace multiplier was used as a test circuit, and delay measured using the SIDL scheme across four different technologies [19]. The technology file and scaling factor were changed to run these HSPICE simulations. The results of this simulation experiment are

Tech.	**Supply Voltage (volts)**	**Delay using HSPICE simulation (ps)**	**Indicator Stage**	**Average Resolution (ps)**	**Delay measured using SIDL (ps)**
0.18μm	1.8	1197	20	60	1140-1200
0.13μm	1.3	821	18	45	765-810
0.10μm	1.0	664	19	36	648-684
0.07μm	1.0	486	19	26	468-494

Table 4.8: Behavior of SIDL across different technologies

shown in Table 4.8. The technologies and the corresponding supply voltages are listed in columns 1 and 2 respectively, and the simulation experiment was run under nominal process conditions, and at an operating temperature of 25°C. The delay of the test circuit obtained using HSPICE simulations is shown in column 3. The *indicator stage* for delay measurement using SIDL is shown in column 4, and the delay of the SIDL circuit, obtained using this *indicator stage* and the corresponding average resolution is shown in column 6. The average resolution, shown in column 5, is an approximate one calculated based on pulse width at output of some stages for different technologies. It can be seen that with scaling of technology, the resolution scales down proportionally, and hence, the delay measured using SIDL remains reasonably accurate. Hence, determining the resolution of individual stages using calibration will ensure robustness to process variations.

Variation in L	**HSPICE Delay (ps)**	**Indicator Stage**	**Avg. Res. (ps)**	**SIDL Delay (ps)**
-20%	643	16	39	585-624
-10%	900	19	48	864-912
0%	1197	20	60	1140-1200
+10%	1518	21	74	1480-1554
+20%	1851	21	90	1800-1890

Table 4.9: Behavior of SIDL in presence of L Variations

Another simulation experiment was carried out using the critical path of the 4 bit Wallace multiplier as a test circuit to further analyze the response of the SIDL scheme to process variations in 0.18μm technology [19], using 1.8V supply voltage and 25°C operating temperature. One of the major process parameters, namely transistor length, was varied globally by ±20% for this experiment, and the delay of the test circuit measured using SIDL. The latches were sized separately. Such a large variation is a bit pessimistic, but the objective was to analyze behavior of SIDL under such pessimistic conditions of variability. Table 4.9 shows the results of this experiment[1]. It is clear that the SIDL scheme tracks process variations well, and the delay range measured using SIDL represents the path delay with reasonable accuracy.

In both the above experiments, the *sim-avg* effect can cause an error of 1 stage in some cases, which should not be present in real hardware measurements if appropriate calibration is done to determine resolution of individual stages.

4.5 Summary

Response analysis for timing characterization requires detection of gross and small delay faults and determination of magnitude of violation for silicon debug. In modern ICs, current techniques for timing characterization are unable to efficiently accomplish these objectives. In this chapter, schemes for efficient timing characterization of such ICs, based on on-chip delay measurement have been presented. These schemes are robust in face of process variations, and can effectively detect small as well as gross delay faults, and determine magnitude of violation.

---

[1] There was a simulation error in the corresponding table in [35]. In the corrected version presented here, the SIDL scheme tracks variations slightly better than in [35].

The ideas presented in this chapter were published in [35], [38], and [39]. The material presented in this chapter is based on these papers. The multipliers used as test circuits were designed by Antony Sebastine and Whitney J. Townsend.

Chapter 5

Adaptive Design for Parametric Reliability

Until this point in the dissertation, techniques to detect and debug timing violations in chips have been explored, which can enable detection of bad parts, which in turn are discarded. However, a large number of chips can have delay faults that are caused by variations in process and environmental parameters, and these chips, which are otherwise defect free, will fail the test process and be discarded along with the defective parts. This can lead to significantly low yield numbers, and affect the economic viability of a product. As technology scales further into the nanometer domain, the impact of variability on a chip increases. The effect of variability on nanometer scale chips has been elucidated in Chapter 1. Traditionally, designers tackled variations by keeping margins in their designs, but such a design methodology is not feasible in Deep Sub-micron (DSM) technologies, the reasons for which have been discussed in Chapter 1. In Chapter 1, the use of adaptive design as a solution to tackle variability was also discussed, along with the two categories, namely iterative and non-iterative. Adaptive design techniques enable a particular instance of a die to work optimally in face of process variations thus enabling parametric reliability in chips. Since timing is one of the most important parameters of an IC, and forms the focus of this work, hence in this chapter, adaptive design techniques that enable timing-optimized or performance-optimized parametric reliability in ICs are presented. Such techniques will mitigate yield loss die to variability, while giving priority to performance, hence ensuring that adaptation does not cause a chip to violate minimum performance specifications. This is done by providing optimal compensation within predefined performance bounds.

Adaptive design methodologies for process variation tolerance have process perturbation sensing scheme(s), which in turn drive the process compensation mechanism(s) in a circuit. Examples of process perturbation sensing schemes include on-chip delay sensing, on-chip leakage sensing, and on-chip delay measurement schemes. Compensation mechanisms include an appropriate voltage setting , frequency setting, or a particular component from a set of redundant components [15]. An example of such adaptive design for inter-die variations is presented in [65], where a set of keepers are connected to a dynamic node, and an on-chip leakage measurement module is used to determine the process corner of the chip. If the particular instance of a die lies in a low leakage process corner, then a narrow keeper width is selected to ensure that the die has good performance, whereas, if an instance of the die lies in a high leakage process corner, then a wide keeper is selected to enhance robustness of the die. The optimal keeper width is one-time programmable via fuses [65].

The drawback of the above scheme is that there is no direct performance monitoring to ensure that the chip meets the performance specifications. Also this technique is complex, requiring the generation of 3 bitlines from a leakage measurement module, and will require a large area overhead, especially if multiple modules are used per die to compensate for intra-die process variations. A second approach to adaptive design is the use of on-chip delay measurement or delay sensing to determine the process perturbations, and subsequently drive the process compensation mechanism. An example of such design is presented in [89], where the delay of a path is measured to drive an Adaptive Body-Bias (ABB) scheme. This type of adaptive design mechanism ensures that the process compensation that is provided is optimal within the given performance constraint. However, the process perturbation sensing scheme presented in [89] has high overhead, since the delay measurement module is a complex one requiring comparators and a decoder. This scheme is highly area-inefficient in compensation schemes, and even more so for process compensation for intra-die process variations, because of the large number of process perturbation sensing modules that would be needed. Another scheme, which is basically a delay sensing scheme and has low overhead has been presented in [32]. In this scheme, values of a capture latch and a shadow latch whose clock is a delayed version of the capture latch, are compared, and in case of a difference, the supply voltage value is increased (starting from the minimum value). This kind of mechanism for sensing process perturbations would be iterative and can take a lot of time to converge, and also places timing constraints on the path whose delay can be measured.

If delay measurement is used for sensing process perturbations, and a performance bound is set on the amount of adaptation, based on minimum acceptable performance, then the amount of adaptation provided will have a timing constraint, and can be considered performance-optimized adaptation. Also, in order to overcome the aforementioned drawbacks of the existing delay measurement schemes, one can use delay line based time-to-digital conversion techniques [35], [39], [46], [52], for on-chip delay measurement based process perturbation sensing schemes. The adaptive design schemes presented in this chapter use such simple and low overhead on-chip delay measurement modules for sensing process perturbations. These modules directly drive the process compensation mechanism, thus obviating the need for complex decoder circuitry, and has a direct bound for the degree of adaptation based on minimum performance specification, thus enabling performance-optimized parametric reliability. The schemes presented in this chapter can be easily used for adaptation to inter-die variations, but for intra-die variations, it has to be ensured that the process perturbation sensing schemes and process compensation mechanisms lie in the same corner, and this could be a source of inaccuracies.

The rest of this chapter is organized as follows. In Section 5.1 techniques for sensing of process perturbations based on on-chip delay measurement are presented. The compensation mechanisms driven by these process perturbation sensing schemes are presented in Section 5.2. Simulation results are presented in Section 5.3, and a brief summary is presented in Section 5.4.

5.1 Process Variation Sensing Schemes

The on-chip delay measurement module used for process perturbation sensing is a variant of the *Modified Vernier Delay Line* (MVDL) module presented in [39] and in the last chapter, and is shown in Figure 5.1. This circuit, which called MVDL-PD in this work, measures the delay of a path based on the principle of time-to-digital conversion using balanced delay lines, just like MVDL and its predecessors [46], that were presented in the last chapter. The difference between this circuit and the MVDL is that it has programmable delay buffers instead of regular buffers, and the outputs of certain flip-flops in the module are combined into a bus as shown in Figure 5.1. The reason for these alterations will become clear when the working of the MVDL-PD is explained.

The working of the MVDL-PD is quite similar that of the MVDL. It is a balanced delay line module, and has two chains of programmable delay buffers and associated edge triggered latching elements or flip-flops. A combination of a *buf* with delay t_{buf}, *buf_low* with delay t_{buf_low} and a flip-flop whose clock and data inputs are driven by the output of *buf* and *buf_low* respectively through multiplexers, form a stage of the MVDL-PD. The delay of *buf* of a given stage is higher than the delay of *buf_low* of that particular stage. The difference between the delay of *buf(s)* and *buf_low(s)* of any given stage is called the

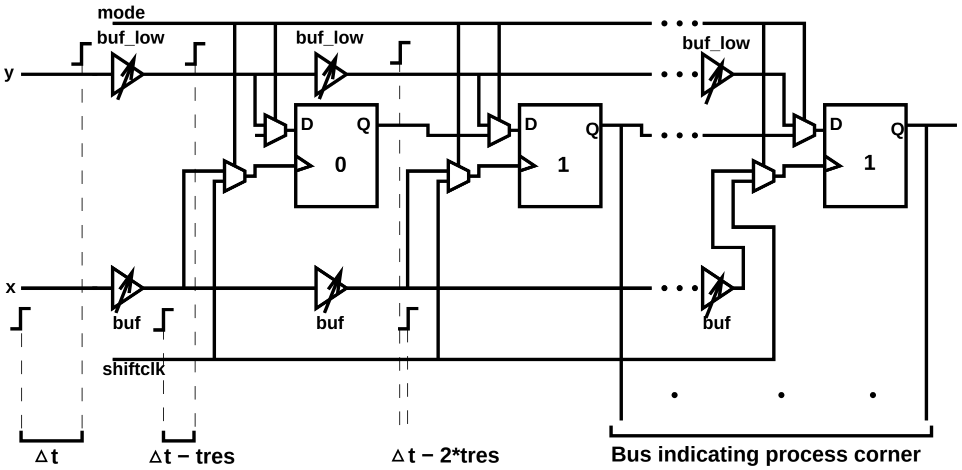


Figure 5.1: MVDL-PD

resolution of that stage, and is given by the following equation.

$$t_{res}(s) = t_{buf}(s) - t_{buf\_low}(s) \tag{5.1}$$

The (s) indicates the stage whose resolution is under consideration, since different stages of the MVDL-PD can be programmed to have different resolution, using the programmable delay buffers. In order to sense process perturbations and determine the corner in which a particular instance of the die lies, the delay of a *Representative Critical Path* (RCP) in the die is measured. This path basically would represent the longest path in the circuit. The input and output of the RCP are routed to the x and y inputs of the MVDL-PD, respectively. The MVDL-PD is initialized by making the values stored in all the flip-flops equal to a logic value 0. This could be done by scanning in a sequence of zeros into the MVDL-PD, since the output of every flip-flop in the

MVDL-PD is connected to the input of the next flip-flop, leading to a scan chain like setting.

Post-initialization, the process of measuring delay of the RCP begins by applying a rising transition at the input of the RCP, which in turn, causes a rising transition at the input x of the MVDL-PD. After a period Δt, which equals the delay of the RCP, a rising transition also occurs on input y of the MVDL-PD. At this point, the signals at the input of the MVDL-PD are separated by a time interval Δt. After crossing the buffers of the first stage, this time interval is reduced by $\mathrm{t}_{res}(1)$, i.e., the resolution of the first stage. However, if $\mathrm{t}_{res}(1)$ is not greater than or equal to Δt, then the rising transition at the data input of the stage 1 flip-flop arrives later than the rising transition at its clock input, and consequently, this flip-flop continues to hold its initialized value.

The time interval between the signals travelling along the upper and lower buffer chains of the MVDL-PD is progressively reduced as the signals propagate through the module. At a particular stage, the rising transition on the data input of the flip-flop of that stage arrives earlier than the rising transition on the clock input, and this flip flop is the first one that latches in logic value 1. This stage is known as the *indicator stage (n)*, since it indicates the delay range of the RCP as follows.

$$(n-1) * t_{res} < \text{Path delay} < n * t_{res} \tag{5.2}$$

For simplicity of representation in the above equation, the resolution of all stages have been assumed to be equal. If resolution is different for some stages, then resolution of individual stages will need to be determined, and the path delay measured using those values of resolution. All stages after the *indicator stage* latch in a logic value 1. There is a possibility of metastability due to setup and/or hold time violation, and the stage or stages in which metastability can occur depends on the values of setup and hold times of the flip-flops, as well as the value of t_{res}. In the common case of a flip-flop with non-zero setup time and zero hold time [126], if t_{res} is higher than the setup time, there is a possibility of metastability only in the *indicator stage* for certain paths. The probability of occurrence of this metastability increases as the setup gets closer to t_{res}. A metastability indicator would need to be used with the flip-flops, an example of which is the one used in [32].

As mentioned earlier, the original MVDL scheme, presented in [39] operates in a manner similar to the one explained above. It was designed for test and debug purposes, and with appropriate calibration to determine resolution of each stage, is robust to process variations. However, such a scheme would not be effective when used for driving process compensation mechanisms, because different components of the process compensation circuitry would be connected (hardwired) to different stages of the MVDL, and these connections cannot be changed after a die has been fabricated. In [46], a technique was proposed to make a balanced delay line tolerant to variations, by making the upper buffer chain using voltage controlled buffers, and regulating the control voltage using a Delay-Locked Loop structure. Instead of such a complex scheme with high overhead, a similar but simpler scheme with a lower overhead is presented here, wherein programmable delay buffers are used in both buffer chains as shown in the MVDL-PD, and a calibration phase using signals separated by known values is coupled along with the process of setting the resolution of each stage to a desired value, irrespective of the process corner in which the MVDL-PD lies. This tuning of the resolution will be done one-time for each fabricated die (since this work targets only manufacturing induced process variation issues and not environmental variation or lifetime degradation issues), and can be done during testing of the part, hence obviating the need of a complex scheme like the one in [46]. In this manner, by clamping the resolution of the MVDL-PD to a known value, delay fluctuations of the RCP and consequently, process perturbations can be tracked. The programmable delay buffers could be analog ones like those used in [46] or digital ones like those used in [129].

To utilize the MVDL-PD to drive a process compensation mechanism, the output of selected flip-flops are combined to form a bus that indicates variations in process parameters (based on the measured delay of the RCP), as shown in Figure 5.1. The selection of the flip-flops is done in order to incorporate a range of latencies of the RCP. The flip-flop whose output forms the left most part of the bus is the one which is the *indicator stage* for the RCP when the die is working in the fastest process corner. The flip-flop whose output forms the right most part of the bus is the one that forms

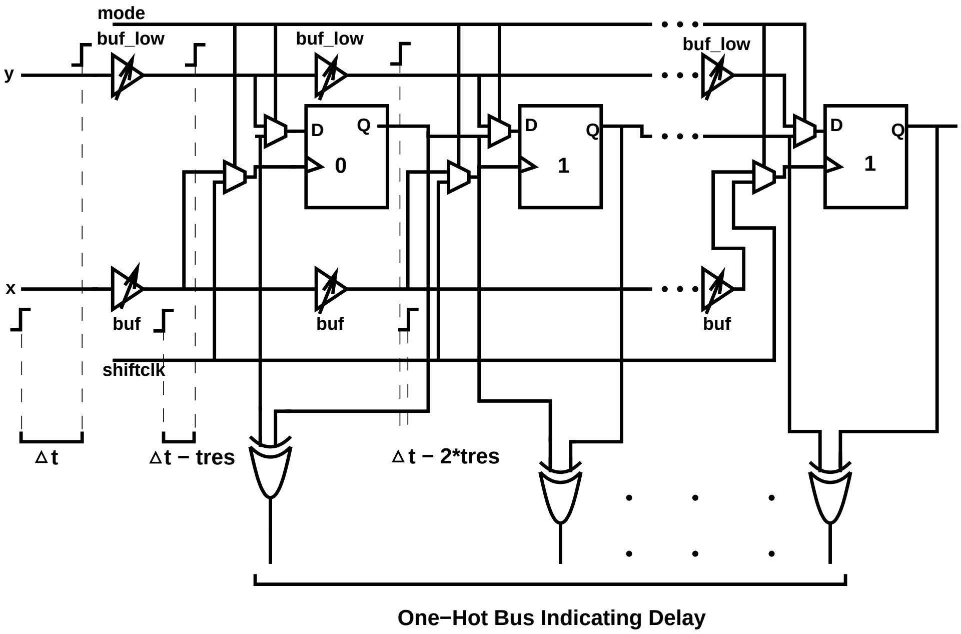


Figure 5.2: MVDL-OHE

the *indicator stage* for the maximum permissible latency of the RCP. If the die lies in a process corner that is any slower than this, then the die will be considered beyond minimum permissible performance specifications, and no further adaptation will be provided. In this manner, priority is given to chip performance. Also, this kind of mechanism ensures that the adaptive design scheme does not attempt to compensate for manufacturing defects. Metastability issues in the flip-flops need to be handled based on the type of flip-flop used.

A modification to the MVDL-PD can be made wherein the input and output of every flip-flop of the MVDL-PD can be connected using XOR gates, and the outputs of the XOR gates can be combined to form a bus which, in turn, can be used to drive the process compensation circuitry. This kind of XORing will make the bus one-hot, with the single output which is at logic value 1 indicating the delay of the RCP, and consequently, the process corner. Such a module is referred to as *MVDL-One Hot Encoded* (MVDL-OHE) in this work, and it is shown in Figure 5.2.

5.2 Process Compensation Mechanisms

In this section, process compensation techniques, that are driven by the process perturbation sensing schemes presented in the last section are presented. These techniques enable post-manufacturing reconfiguration of a chip to compensate for manufacturing induced process variations.

5.2.1 Process Compensation for Performance-Optimized Robustness

In this subsection process compensation techniques are presented for dynamic and static CMOS circuits, that ensure a given circuit has optimal robustness to certain noise mechanisms in a chip, while not violating performance constraints. The primary noise mechanisms that affect digital integrated circuits are leakage, charge sharing, crosstalk, power supply noise, and substrate noise [122]. In this work, techniques for compensating variations in process parameters that affect 1) sub-threshold leakage noise in static and dynamic circuits, and 2) charge sharing noise in dynamic circuits are targeted.

Sub-threshold leakage noise constitutes a major portion of the current

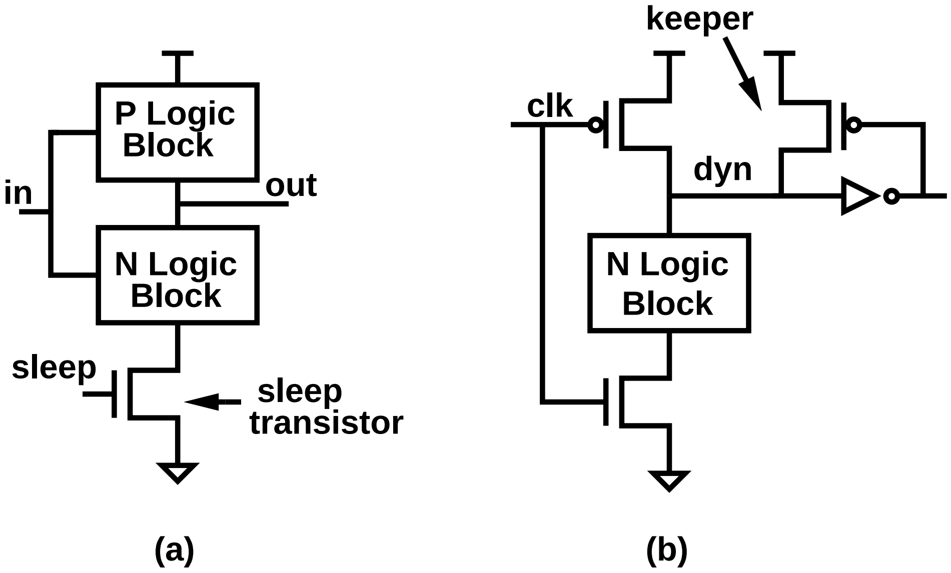
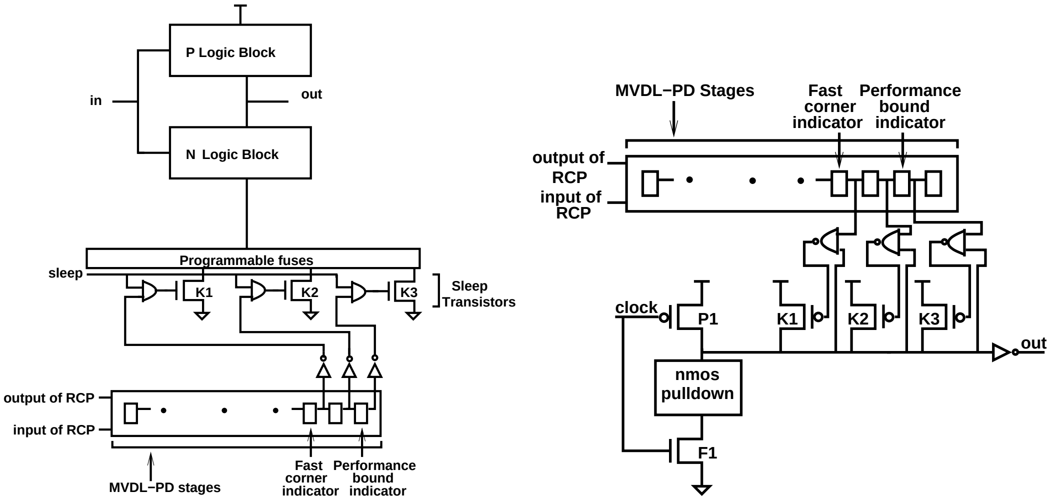


Figure 5.3: Noise Tolerance Mechanisms

flowing through a transistor in the off-state and is a significant problem in DSM technologies. Leakage current causes a charged node of a static circuit to discharge, or a discharged node to get charged through an off transistor. The magnitude of leakage current increases with reduction in the threshold voltage of transistors, which is the normal trend in scaling. The primary sources of sub-threshold leakage current in transistors are weak inversion and drain induced barrier lowering [91].

Charge sharing noise primarily affects dynamic circuits, and is caused by redistribution of charge on the dynamic node of a circuit with other internal nodes in the circuit, with which the dynamic node is connected through an on transistor. The on transistor may be in such a condition due to a desired signal at its input, or an undesired transition caused by factors like crosstalk [122].

In static CMOS circuits, a circuit level technique to mitigate leakage constitutes addition of a transistor between the circuit whose leakage needs to be mitigated, and the corresponding supply rail [91]. This transistor is known as the sleep transistor, and is shown in Figure 5.3a. It is switched off in the off state of the circuit using the *sleep* signals, in order to mitigate the flow of leakage current. The sleep transistor is sized keeping in mind the magnitude of leakage of the chip, and performance constraints.



(a) (b)

Figure 5.4: Process compensation for a) static CMOS circuits b) dynamic circuits

However, when variations in process parameters affect the circuit, then the sizing of the sleep transistor becomes a difficult task. For such circumstances, a scheme is presented here wherein a bank of sleep transistors are connected to the logic block using programmable fuses as shown in Figure 5.4a. Each transistor in the bank gets its input from a signal obtained by ANDing the *sleep* signal with a bit from the bus that comes out of a MVDL-PD and indicates the process corner. This results in selection of an optimal sized sleep transistor. As an example, assume that the 11^{th} stage of the MVDL-PD is made to be the *indicator stage* for the RCP under a nominal process corner, the 10^{th} stage is made to be the *indicator stage* when the die is working in the fast process corner, and the 12^{th} stage is made to be the *indicator stage* for an upper bound on the latency allowable under the specification of the chip (these configurations can be done using the programmable delay buffers). Then the process corner indicators in the adaptive sleep transistor scheme shown in Figure 5.4a are basically the stages that precede the *indicator stage* for delay of RCP in a given corner. For instance, in the above example, the 9^{th} stage is the *fast corner indicator*, 10^{th} stage the *nominal corner indicator*, and the 11^{th} stage is the *performance bound indicator*.

The (inverted) outputs of the corner indicator flip-flops (three bits in this example) are fed into the three sleep transistors *K1, K2, K3* through their corresponding AND gates as shown in Figure 5.4a. If the die is working in the fast corner, then stage 10 of the MVDL-PD is the *indicator stage* (i.e., stage 9 is the corner indicator), and only *K1* will be selected as the sleep transistor to be controlled using *sleep* signal. However, if the chip lies in a very slow process corner, then all three transistors *K1, K2, K3* will be selected as sleep transistors. If the chip is any slower, i.e., the *indicator stage* is beyond stage 12 of the MVDL-PD, then no additional robustness would be provided. Once the right sleep transistor width is selected, all others will have to be disconnected from the circuit using the programmable fuses, an example of which are the compact, electromigration based, electrically programmable fuse presented in [67]. The signals routed from the MVDL-PD to the process compensation mechanisms would also need to be routed to the fuse programming mechanism for such a scheme. The hardware overhead of the above scheme can be reduced by making a logic block share a compensation mechanism.

As mentioned before, dynamic circuits are susceptible to leakage as well as charge sharing noise, and a common technique used to enable robustness of the dynamic node is the use of a keeper transistor as shown in Figure 5.3b. The sizing of this keeper is a tradeoff between performance and robustness, and hence is difficult in face of process variations. In [65], it was suggested that a bank of keepers be used for process compensation, and the same mechanism is used here, wherein each keeper in the bank gets its input from a NAND gate, which in turn has one input driven by the dynamic node, and a second one driven by a signal from the process perturbation sensing mechanism. However, the difference is that adaptive keeper scheme presented here is driven by a delay measurement based process perturbation sensing mechanism as shown in Figure 5.4b. The working of this scheme is very similar to that of the adaptive sleep transistor scheme. However, the process corner indicator here is the same as the *indicator stage* for RCP delay in the given corner (instead of being the preceding one as in the adaptive sleep transistor case). If the circuit lies in the fast corner, all three transistors *K1, K2, K3* are selected as keepers, whereas if the circuit lies in a process corner where the delay of the RCP equals to the pre-determined upper bound, then only *K3* is selected as keeper. If the corner is any slower, no keepers are selected, because the circuit is too slow for robustness compensation.

In this manner, it can be ensured directly that the chip has optimal robustness within the given performance constraint, and this constraint is not violated due to improper keeper or sleep transistor selection. Thus, this scheme gives performance the highest priority when compensating for process variation, and is also a simple one, since selected outputs of the MVDL-PD are directly routed to the process compensation mechanism. The MVDL-OHE scheme can also be used instead of the MVDL-PD for driving such an adaptive sleep transistor scheme, wherein the transistor selection mechanism would be one-hot.

5.2.2 Correction of Process Variation Related Timing Failures

In this section, a technique is presented that would enable fixing of timing violations in a chip by automatically delaying launch and/or capture clocks in pipeline stages, based on an on-chip delay measurement based process perturbation sensing. This will take care of timing violations caused by paths being slower than anticipated due to process variations, as well as timing violations caused by short paths (i.e., hold time violations), hence making the chip a self-correcting one in face of process variation related errors. Only

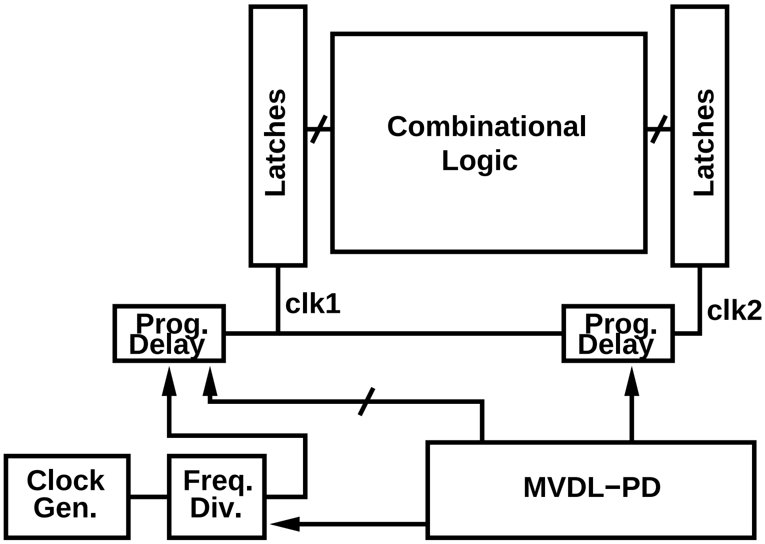


Figure 5.5: Correction of process variation related timing failures

the concept of the scheme is presented in this dissertation, and the actual implementation is future work and is not incorporated here.

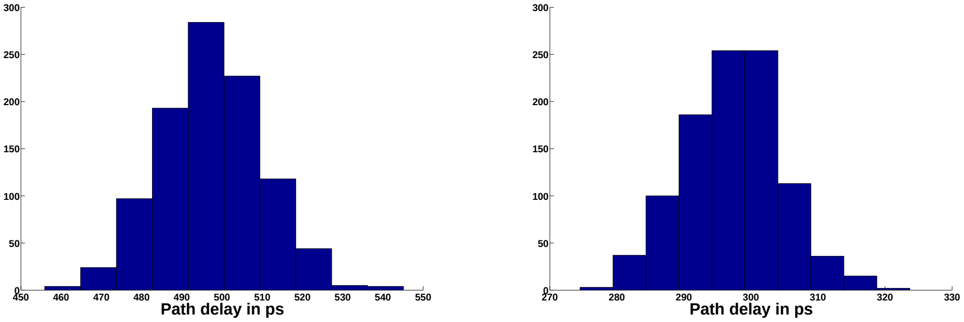
The proposed adaptive design scheme is shown in Figure 5.5. The clock is distributed through programmable delay modules to different launch and capture flops of various pipeline stages. These programmable delay modules would be made up of digital programmable delay buffers like those presented in [49] and/or [129]. The digital word which is the output of the MVDL-PD module is used to select the right delay. The MVDL-PD in turn makes this decision based on delay measurement of a RCP. In case there is a timing violation, i.e., the delay of the RCP is longer than anticipated, then the MVDL-PD automatically selects a higher value of skew between launch and capture clocks of a path. Subsequently, a test is run to see if there is a hold time violation. If there is no such violation, then the timing problems of this pair of launch and capture flop is fixed. Otherwise, the launch clock is also delayed sufficiently so that the hold time violation gets fixed, while the adjustment to overcome timing violation of the the long path is not destroyed. Since most paths in a pipeline stage and/or a chip have comparable propagation delay, such a scenario will not happen very frequently. In this manner, all timing related violations on a chip, that are caused by process variations can be eliminated. A similar procedure will be carried out to correct hold time violations. Also, this entire process will be carried out during the manufacturing test period, to configure the chip before shipping it. However, this timing correction scheme will fail in cases where the launch and capture latches of a pipeline stage are the same, i.e., a feedback circuit.

The idea of tackling clock skews has been explored earlier in the literature, and a discussion of certain techniques to tackle process variation related skews follows. In [64], a technique is presented wherein a single phase detector is placed inside each chip, and for each clock path also called "forward path" in the paper, there is a "return path" that is exactly matched to the clock path in terms of delay. De-skewing is done based on phase comparison of the "forward path" and "return path" signals. The drawback of this scheme is that there is a significant increase in area overhead due to "return paths", and also, it relies upon the ability to exactly match the "forward" and "return paths" of the clock, which may not be feasible in DSM technologies, especially over the long distances across which the clock signals are routed. In [45] local phase detectors were used to compare clock signals between two domains, and subsequently, perform de–skewing. In [129], techniques to deliberately add or remove skew are utilized, and this is achieved by varying the delay of various programmable delay buffers through which the clock is routed. However, the scheme in [129] controls the delay of the buffers using a digital word that is scanned in externally, whereas the aforementioned scheme using MVDL-PD utilizes on-chip sensing of process perturbations to do the same. A similar technique as in [129] is applied in [49], for controlling timing issues between different cores of a SOC. In [93] and [92], techniques for correcting timing failures related to variability are explored, wherein the clock edges are moved (i.e., skewed) either to fix timing violations or to optimize frequency. Programmable delay buffers are either controlled using a digital work that can be scanned in, or using the firmware during regular operation of the chip. The use of firmware enables online correction of timing failures, but the scheme is complex compared to the MVDL-PD based scheme presented above (which however cannot be used for reconfiguration in the field), where the outputs of different flip-flops are directly connected to the compensation mechanism.

5.3 Simulation Results

The test circuit used for the adaptive keeper and adaptive sleep transistor schemes is the top critical path (non-inverting version) of a 4x4 Wallace multiplier (referred to as w4 in this paper), extracted using a static timing analysis tool. All simulations were carried out using 70nm BSIM technology [19] with 0.9V supply voltage and 25°C operating temperature. In the static

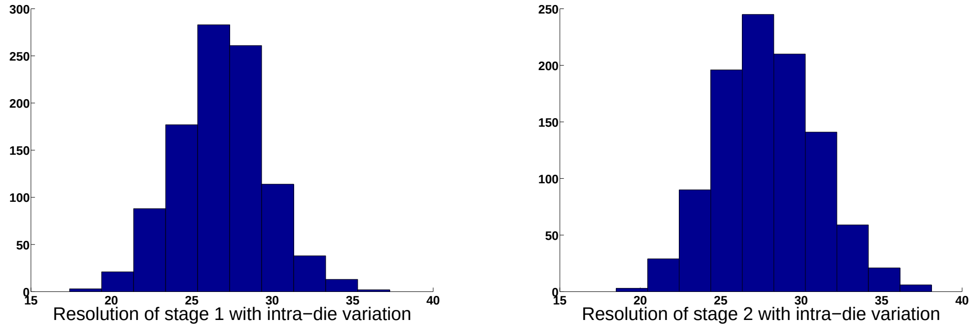
CMOS version of w4 (static w4), a NMOS footer used as a sleep transistor was inserted in every gate. To obtain a dynamic version of the test circuit (dynamic w4), the w4 circuit was re-designed using complex dynamic gates to ensure that the maximum number of gates have three inputs (and remaining gates having less than 3 inputs). Monte Carlo simulations were carried out across 1000 different random combinations of process parameters (L, V$_{th}$, t$_{ox}$), for both versions of the test circuit, and the path delay distribution of both are shown in Figure 5.6. The y-axis in the graphs represent the number of different combinations of process parameters. Figure 5.6a shows the path delay distribution of the static CMOS version of w4 with 1μm NMOS footers used as sleep transistors, and the mean (μ) value of the delay for this circuit is 497ps, and the σ = 12.7ps. Figure 5.6b shows the path delay distribution of the dynamic version of w4 using minimum sized keepers (0.07μm), which has a mean value of 297ps, and σ = 7.3ps.



(a) (b)

Figure 5.6: Latency variation across process corners for a) Static CMOS w4 circuit with sleep transistors b) Dynamic w4 circuit with keepers

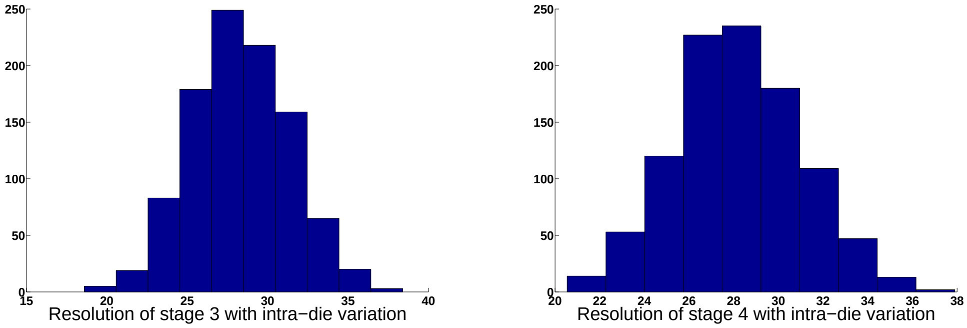
The resolution of the MVDL also varies with variations in process parameters. The distribution of resolution of first four stages of an MVDL for 1000 different random combinations of process parameters using Monte Carlo simulation are shown in Figures 5.7 and 5.8. These resolutions are calculated based on delay difference between two signals, i.e., the signal at the output of the high delay buffer and the signal at the output of the low delay buffer for each stage of the MVDL. Additionally, the resolution of each stage is a gross one, equalling two buffer delays. Finer resolutions can be obtained if desired, by using buffer sizing instead of gross buffer delays.



(a) (b)

Figure 5.7: Variation of resolution of a) stage 1 and b) stage 2 of an MVDL across process corners

The mean value of resolution of stages 1, 2, 3 and 4 are 27ps, 28ps, 28ps and 28ps respectively, and the corresponding σ values are 2.8ps, 3.1ps, 3.1ps and 2.8ps respectively. The average resolution, computed based on the aforementioned resolution of the four stages at each corner, has a mean value



(a) (b)

Figure 5.8: Variation of resolution of a) stage 3 and b) stage 4 of an MVDL across process corners

of 28ps and σ of 1.6ps for the same Monte Carlo simulation experiment as above. The distribution of the average resolution of four stages of the MVDL, with random variations in process parameters is shown in Figure 5.9.

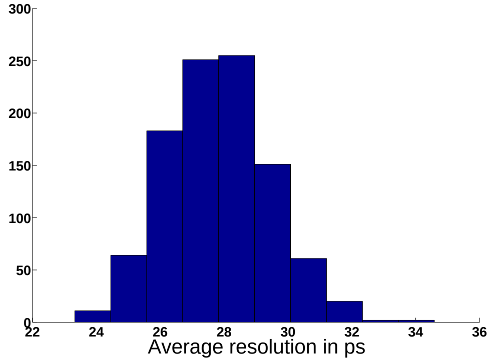


Figure 5.9: Variation of average MVDL resolution across process corners

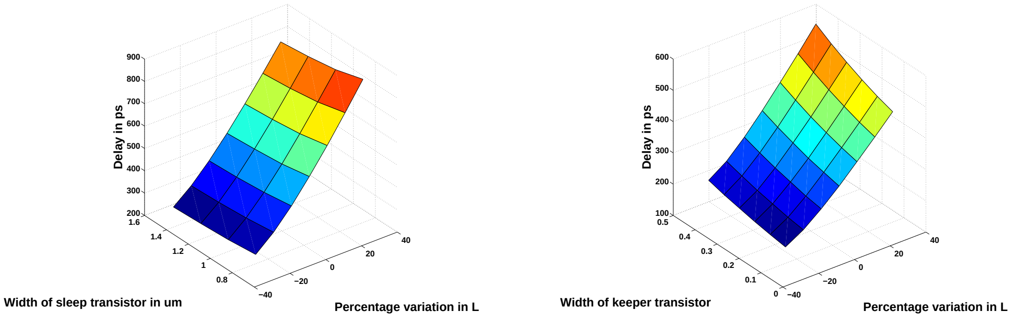
The response of the MVDL to gross variations was analyzed by varying the channel length globally by 30% above and below its nominal value (a 30%

Variation in L	**Resolution of MVDL stages**				**Average Resolution (ps)**
	Stage 1 (ps)	**Stage 2 (ps)**	**Stage3 (ps)**	**Stage4 (ps)**	
-30%	12	13	14	13	13
-20%	16	19	17	17	17
-10%	22	24	21	23	23
0%	27	28	28	28	28
10%	33	34	35	35	34
20%	39	42	41	41	41
30%	44	48	48	48	47

Table 5.1: Variation of MVDL resolution with variation in channel length L

variation in L is pessimistic in typical chips). The results of this simulation experiment are shown in Table 5.1. Variation in resolution of individual stages, calculated based on delay difference between two signals, i.e., the signal at the output of the high delay buffer and the signal at the output of the low delay buffer for each stage of the MVDL, are shown in columns 2-5 of Table 5.1. The mean value of resolution for stages 1, 2, 3 and 4 for this simulation experiment are 28ps, 30ps, 29ps and 29ps respectively, and the corresponding σ values are 11.8ps, 12.5ps, 12.7ps, 12.8ps respectively. The average resolution is calculated based on the resolution of these four stages of the MVDL. The variation in average resolution is shown in column 6 of Table 5.1, and has a mean value of 29ps and σ value of 12.5ps, thus yielding a $\frac{\sigma}{\mu}$ of 0.4 which is much higher than that for random variations.

Based on the results obtained from the above two simulation experiments carried out on the MVDL, it can be said that the resolution of the regular MVDL is a trade-off between granularity and process variation sensitivity. It is also clear that the variation in resolution is significant enough to justify use of programmable delay buffers suggested for the MVDL-PD. The use of a module like MVDL-PD enables the resolution to be set by means of variable delay buffers to a pre-determined value, independent of process variations, thus enabling the module to directly drive different compensation mechanisms in the IC.



(a) (b)

Figure 5.10: Delay obtained by varying L and sweeping a) Width of sleep transistor in static CMOS version of w4 b) Width of keeper in dynamic version of w4

Figure 5.10 illustrates the impact of different sized sleep transistors and different sized keepers on the latency of the test circuit across process corners, simulated by varying L by 30% above and below its nominal value. Figure 5.10a shows the impact of different sleep transistor sizes on the latency of the static CMOS version of w4 at different process corners, by varying the width of the sleep transistor from 0.75μm to 1.5μm in steps of 0.25μm, while Figure 5.10b shows the impact of different keeper sizes on the latency of the test circuit at different process corners, by varying the keeper width from 0.07μm to 0.42μm.

When L is 30% lower than the nominal value, the delay of static w4 with a 0.75μm sleep transistor is 5.5% higher than the delay of static w4 with a 1μm sleep transistor. On the other hand, the delay of static w4 with a 1.5μm sleep transistor is 5.5% lower than the delay of static w4 with a 1μm sleep transistor. At this value of L, the use of a 0.75μm sleep transistor instead of a 1.5μm one leads to an increase of 11.7% in the delay of static w4. At nominal value of L, the delay of static w4 with a 0.75μm sleep transistor is 4.8% higher than the delay of static w4 with a 1μm sleep transistor. On the other hand, the delay of static w4 with a 1.5μm sleep transistor is 4.6% lower than the delay of static w4 with a 1μm sleep transistor. At this value of L, the use of a 0.75μm sleep transistor instead of a 1.5μm one leads to an increase of 9.9% in the delay of static w4. When L is 30% higher than the nominal value, the delay of static w4 with a 0.75μm sleep transistor is 4.5% higher than the delay of static w4 with 1μm sleep transistor, whereas the delay of static w4 with a 1.5μm sleep transistor is 4.3% lower than the delay of static w4 with a 1μm sleep transistor. At this value of L, the use of a 0.75μm sleep transistor instead of a 1.5μm one leads to an increase of 9.2% in the delay of static w4. Thus it can be seen that there is a significant performance penalty for selection of a wrong sized sleep transistor.

A similar performance penalty also exists for selection of a wrong sized keeper. At nominal value of L, the delay of dynamic w4 with a 0.14μm keeper is 3.9% higher than the delay of dynamic w4 with a 0.07μm keeper. At the same value of L, the delay of dynamic w4 with a 0.42μm keeper is 26.7% higher than the delay of dynamic w4 with a 0.07μm keeper. The former however is more representative of a realistic difference in keeper sizes for the dynamic gates in the circuit. When L is 30% lower than the nominal value, the delay of dynamic w4 with a 0.14μm keeper is 4.6% higher than the delay of dynamic w4 with a 0.07μm keeper. When L is 30% higher than the nominal value, the delay of dynamic w4 with a 0.14μm keeper is 3.5% higher than the delay of dynamic w4 with a 0.07μm keeper.

These results project the significance of performance optimized adaptive design schemes for robustness presented in this paper, and justify their incorporation in ICs that use sleep transistors and/or keepers for robustness.

Supply Voltage volts	**Resolution of MVDL stages**				**Average Resolution (ps)**
	Stage 1 (ps)	**Stage 2 (ps)**	**Stage3 (ps)**	**Stage4 (ps)**	
0.6	48	47	51	51	49
0.7	35	38	38	36	37
0.8	30	30	31	31	31
0.9	26	28	27	28	27
1.0	24	24	23	23	23
1.1	21	24	23	23	23
1.2	20	21	22	21	21

Table 5.2: Variation of MVDL resolution with variation in supply voltage

An analysis of the response of the MVDL to environmental variations, specifically voltage variations, was carried out and the results of this simulation experiment are shown in Table 5.2. Supply voltage values are shown in column

1, and the resolution of stages 1 to 4 at that corresponding supply voltage are shown in columns 2 to 5 respectively. These resolutions are calculated based on delay difference between signals at the output of the two buffers just as in the previous simulation experiments in this section. The mean value of resolution for stages 1, 2, 3 and 4 for this simulation experiment are 29ps, 30ps, 31ps and 31ps respectively, and the corresponding σ values are 9.8ps, 9.2ps, 10.5ps, 10.4ps respectively. The average resolution is calculated based on the resolution of these four stages of the MVDL. The variation in average resolution is shown in column 6 of Table 5.2, and has a mean value of 30ps and σ value of 9.9ps. The key inference from this experiment is that the MVDL will be ineffective for online compensation, i.e., compensation in the field for environmental variations. In the next chapter, an overview of existing techniques for online sensing is presented along with a discussion of the advantages and disadvantages of such schemes.

5.4 Summary

In this chapter, adaptive design techniques that ensure parametric reliability in ICs in face of process variations are explored. These techniques use on-chip delay measurement of a RCP for sensing process perturbations, and have a performance-bound on the degree of adaptation, hence enabling performance optimized parametric reliability. Additionally, the process perturbation sensing schemes presented here are simple, enable one-shot adaptation instead of an iterative adaptation, which would enable faster post-silicon tuning to compensate for process variations. Part of the work presented in this chapter was published in [33] and corresponding material in the chapter is based on the same paper. The multipliers used as test circuits were designed by Antony Sebastine and Whitney J. Townsend.

Chapter 6

Conclusion and Future Work

An efficient and systematic timing characterization technique is imperative for Integrated Circuits (ICs) designed in Deep Sub-micron (DSM) technologies. This is driven by factors like stringent timing specifications, reduced slack due to aggressive design, and a larger impact of defects and process variations on behavior of such ICs. Timing characterization involves delay fault testing and silicon debug. Delay fault testing requires techniques to detect gross delay faults in order to ship quality parts, as well as techniques to detect small delay faults. Detection of small delay faults is important in ICs designed in DSM and nanoscale technologies, because it can help detect a large number of defects, which if undetected during the test process can become reliability hazards in the field.

Current based testing methods used earlier to detect defects are less effective in doing so in ICs designed in DSM technologies due to the presence of leakage. However, most of these defects do affect the timing behavior of paths, but may not cause gross delay failures, which necessitates detection of small delay faults. Additionally, there is a need for techniques to facilitate structured debug to enable fewer silicon spins and lower time-to-market. The effectiveness of traditional timing characterization techniques are challenged in ICs designed in DSM technologies due to a plethora of reasons. Firstly, small feature sizes and very high level of integration have severely limited controllability and observability of such ICs. Secondly, most ICs operate well in the GHz range, whereas testers at most facilities still operate at a few hundred MHz, and upgrading these testers would be an expensive proposition, even if it could be possible to do so. All of these factors point to the use of special hardware modules for timing characterization, i.e., Design for Test (DFT) and Design for Debug (DFD) structures. However, existing DFT and DFD structures have their limitations for timing characterization, which include controllability limitations like lower coverage and higher complexity of test generation for delay fault testing in regular scan based designs, as well as observability limitations like gross margins used in current delay fault testing techniques.

In Chapter 2, a technique to overcome the controllability limitations for delay test and debug in scan based static CMOS circuits was presented. This technique, called *Tri-Scan*, provides the same level of accessibility for timing characterization for scan based sequential static CMOS circuits as that of combinational circuits with complete access to all inputs, and does so without the excessive area overhead associated with other techniques like *enhanced scan*. A comparison of transition fault coverage for different circuits with complete access as well as in a scan based design methodology was provided in Chapter 2, and it was illustrated that complete access enables significantly higher coverage for all the test circuits considered and in some cases the increase in coverage was over 30%. The *Tri-Scan* scheme also provides reduction in the amount of switching power dissipated during scan operation by enabling isolation between combinational logic and scan flip-flops during the operation, and in the best case, this reduction is nearly 2X. An analysis of area and performance overheads was also carried out and the results presented in Chapter 2. In nanoscale technologies, there might be a need to use a keeper in conjunction with the *Tri-Scan* scheme, and the impact of that keeper on the performance of the path would be an interesting idea to explore. Based on the outcome of that analysis, a decision could be made on the need for incorporating a mechanism in the chip to de-activate the keeper during regular mode operation of the chip. Also, integration of the *Tri-Scan* scheme with existing commercial testing tools would pose some challenges, like making the tool understand the new functionality of the *scan_enable* signal.

Dynamic circuits are used in a number of ICs, primarily to speed up critical paths. Hence, timing characterization for such circuits is important, which is hindered by the fact that dynamic circuits have a *precharge* or reset phase between application of initialization and transition vectors for delay fault testing. Existing solutions cause ambiguity in determining the faulty path. Additionally, dynamic circuits face the same controllability issues for timing characterization as static CMOS circuits, when used in sequential designs with scan. However, the use of a scheme like *Tri-Scan* involves insertion of static CMOS components, that may slow down the path. In Chapter 3, techniques to enhance controllability for efficient timing characterization were explored, and two sets of novel Design for Test and Debug (DFTD) schemes are presented. The first set of schemes, namely, *Precharge Control Schemes* enable efficient delay test and debug of dynamic circuits in general (i.e. with complete access), while the second set facilitates pattern application in scan based dynamic circuits. Together, they reduce the problem of delay test and debug of scan based dynamic circuits to that of delay test and debug of static CMOS circuits with complete access to all primary inputs. The use of these schemes also enables greater compatibility of dynamic circuits with existing test and debug tools for static CMOS circuits.

Gross margins used for timing characterization create an observability limitation for DSM ICs, since a large number of defects could go undetected. Existing techniques to enhance observability for timing characterization were explored in Chapter 4, and three techniques for on-chip delay measurement based timing characterization were presented. A timing characterization scheme based on on-chip delay measurement can detect small increments in delay of a path, as well as measure the magnitude of the increment. Hence such a timing characterization scheme can detect small delay faults (and consequently, defects causing such delay faults), as well as the magnitude of violation for silicon debug. It can also enable quick characterization of a chip across various operating conditions, and help in speed binning. The first module presented in Chapter 4 is a *Modified Vernier Delay Line* (MVDL), which is a balanced delay line scheme, and overcomes drawbacks of existing balanced delay lines for timing characterization, by facilitating easier reading out of the values of the measured delay, and enabling measurement of delay for all kinds of paths and transitions using a single MVDL module. The second module is called *Delay Scan Chain*, and it operates in a manner similar to the MVDL, but its benefits lie in the fact that it reuses existing scan chains on a chip with slight modification for delay measurement purposes. The third module, called *Skewed Inverter Delay Line* (SIDL), overcomes drawbacks of existing delay measurement modules by eliminating the need for matched routing over long distances, and hence creates fewer sources of inaccuracies for timing characterization. Calibration techniques presented in Chapter 4 make these schemes robust in face of process variations. The modules themselves have very low area overhead. Future research will need to focus on selection of paths in an IC whose delay needs to be measured. Static timing analysis based selection may not be effective in DSM technologies because top critical paths may change due to variability. Additionally, an analysis to determine how many of these on-chip delay measurement modules would be used in a chip is an interesting problem. Using a large number of such modules, and placing them near critical paths would increase area overhead due to the modules, while reducing area overhead due to routing of the signals from critical path to the delay measurement module. Similarly, using fewer modules would lead to a lower overhead due to the modules, but increase routing overhead. Optimal placement of these modules within a chip is also an interesting problem for future work.

Efficient timing characterization can ensure better screening of defective parts and ensure fewer such parts are shipped. However, in DSM technologies a large number of dies failing the test process may not be defective. It is possible that they failed to meet timing requirements due to variations. It was mentioned that process variations can impact the performance of an IC by 30%, [14], and hence there is a very high possibility of defect free dies failing the test process, and this could lead to significantly low yield numbers and reduce the economic viability of manufacturing a chip in DSM technologies. A potential solution to this problem is to use adaptive design techniques that enable a chip to configure itself to work optimally in face of variations. In Chapter 5, such adaptive design techniques have been explored, with a focus on ensuring that minimum acceptable timing specifications are not violated due to adaptation.

The techniques presented in Chapter 5 utilized on-chip delay measurement (with facility for an in-built performance bound) for process perturbation sensing, thus enabling performance-optimized adaptive design to ensure parametric reliability in ICs. The techniques presented are simple, drive the compensation circuitry directly, and carry out adaptation in a non-iterative manner. However, a drawback of the adaptive design techniques presented in Chapter 5 is that they can only compensate for manufacturing induced process variations, and not variations in environmental factors like supply voltage, temperature etc. Compensation for such variations would require online variability sensing. There have been some techniques reported in the literature for such purposes. In [63], [72] and [133] the delay of a RCP was monitored using a phase detector or signal comparator circuits, and based on the output of these circuits, the supply voltage or in some cases, the body-bias was adjusted iteratively. A similar technique is used in [32], that has already been explained in the previous chapter. In [94] a tapped delay line with intermediate flip-flops is used, and the outputs of these flip-flops are encoded and fed into a circuit that iteratively increases or decreases the supply voltage based on the data bits coming from the encoder. The concept of non-iterative or one-shot adaptation was explored in [43] and [44]. Additionally, in [112], a variability sensing module was presented that could drive non-iterative adaptive design. However, a major drawback of all these technqiues is that online sensing and compensation could cause the IC to go into a loop. For example, the IC could loop between a higher and lower voltage. In [43] and [44], this problem was alluded to, and the authors stated that their scheme was effective for monotonic variations in supply voltage. However, in a chip it cannot be guaranteed that variations in environmental factors would always be monotonic, especially when a compensation mechanism is used. This provides an area for future exploration, i.e., non-iterative techniques for online variability sensing and compensation for non-monotonic variations to ensure parametric reliability in DSM and nanoscale ICs. Additionally, analysis techniques to determine which paths/logic blocks would be suitable candidates for the compensation mechanisms presented in Section 5.2, is an interesting area to investigate. These could be based on slack sensitivity analysis techniques like those presented in [113] or any other analysis technique that can be utilized or extended to help determine if the path/logic block requires compensation to variations.

Appendices

Appendix A

List of Acronyms and Abbreviations

ATPG - Automatic Test Pattern Generators

CAD - Computer-Aided Design

CUT - Circuit Under Test

DFD - Design for Debug

DFM - Design for Manufacturability

DFT - Design for Test

DFTD - Design for Test and Debug

DLL - Delay-Locked Loop

DSCAN - Delay Scan

DSM - Deep Sub-micron

IC - Integrated Circuit

IPSL - Integrated Pulse Shaping and Latching

LE - Latch Enable

LSDL - Limited Switching Dynamic Logic

MVDL - Modified Vernier Delay Line

MVDL-PD - Modified Vernier Delay Line - Programmable Delay
MVDL-OHE - Modified Vernier Delay Line - One Hot Encoded
OPL - Output Prediction Logic
PI - Primary Input
PO - Primary Output
PPI - Pseudo-Primary Input
PPO - Pseudo-Primary Output
PUT - Path Under Test
RCP - Representative Critical Path
SE - Scan Enable
SIDL - Skewed Inverter Delay Line
SOC - System on a Chip
SOI - Silicon-On-Insulator
STA - Static Timing Analysis
STR - Slow-to-Rise
STF - Slow-to-Fall
t_{res} - Resolution of delay measurement circuits
V_1 - Initialization vector for delay fault testing
V_2 - Transition vector for delay fault testing
VDL - Vernier Delay Line

Bibliography

[1] N. Abaskharoun, M. Hafed, and G. W. Roberts. Strategies for On-Chip Sub-nanosecond Signal Capture and Timing Measurements. In *International Symposium onCircuits and Systems*, pages 174–177. IEEE, May 2001.

[2] N. Ahmed, C. Ravikumar, M. Tehranipoor, and J. Plusquellic. At-Speed Transition Fault Testing with Low Speed Scan Enable. In *VLSI Test Symposium*, pages 42–47. IEEE, May 2005.

[3] R. Aitken. Redundancy-It's notjust for defects anymore. In *International Workshop on Memory Technology Design and Testing*, pages 117–120. IEEE, August 2004.

[4] S. B. Akers. On a Theory of Boolean Functions. *Journal Society of Industrial Mathematics*, 7(4):487–497.

[5] Artisan Components Inc. *TSMC 0.18μm Process 1.8 Standard Cell Library Databook*, February 2002.

[6] K. Baker, G. Gronthoud, M. Lousberg, I. Schanstra, and C. Hawkins. Defect-Based Delay Testing of Resisitive Vias-Contacts: A Critical Evaluation. In *International Test Conference*, pages 467–476. IEEE, September 1999.

[7] H. Balachandran, K. M. Butler, and N. Simpson. Facilitating Rapid First Silicon Debug. In *International Test Conference*, pages 628–637. IEEE, October 2002.

[8] S. Balajee and A. K. Majhi. Automated AC (Timing) Characterization for Digital Circuit Testing. In *International Conference on VLSI Design*, pages 374–377. IEEE, January 1998.

[9] M. Bazes. A Novel Precision MOS Synchronous Delay Line. *IEEE Journal of Solid State Circuits*, 20(6):1265–1271, December 1985.

[10] M. Bazes and R. Ashuri. A Novel CMOS Digital Clock and Data Decoder. *IEEE Journal of Solid State Circuits*, 27(12):1934–1940, December 1992.

[11] W. Belluomini, D. Jamsek, A. Martin, C. McDowell, R. Montoye, T. Nguyen, H. Ngo, J. Sawada, I. Vo, and R. Datta. An 8 GHz floating point multiply. In *International Solid State Circuits Conference*, pages 374–375. IEEE, February 2005.

[12] K. Bernstein, K. M. Carrig, , C. M. Durham, P. R. Hansen, D. Hogenmiller, E. J. Nowak, and N. J. Rohrer. *High Speed CMOS Design Styles*. Springer.

[13] A. Bhavnagarwala, S. V. Kosonocky, S. P. Kowalczyk, and R. V. Joshi. A Transregional CMOS SRAM with Single, Logic Vdd and Dynamic Power Rails. In *Symposium on VLSI Circuits*, pages 291–293. IEEE, June 2004.

[14] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De. Parameter Variations and Impact on Circuits and Microarchitecture. In *Design Automation Conference*, pages 338–342. IEEE, June 2003.

[15] P. Bose. Variation-Tolerant Design. *IEEE Micro*, 25(2):5–5, March-April 2005.

[16] S. A. Bota, M. Rosales, J. L. Rossello, and J. Segura. Low VDD vs. Delay: Is it really a good correlation metric for nanometer ICs. In *VLSI Test Symposium*, pages 358–363. IEEE, May 2006.

[17] K. A. Bowman, S. G. Duvall, and J. D. Meindl. Impact of Die-to-Die and Within-Die parameter fluctuations on the maximum clock frequency distribution for gigascale integration. *IEEE Journal of Solid State Circuits*, 37(2):183–190, February 2002.

[18] L. Bruni, G. Buonanno, and D. Sciuto. Transistor Stuck-at and Delay Faults Detection in Static and Dynamic CMOS Combinational Gates. In *International Symposium on Circuits and Systems*, pages 431–434. IEEE, May 1992.

[19] Y. Cao, T. Sato, M. Orshansky, D. Sylvester, and C. Hu. New paradigm of predictive MOSFET and interconnect modelling for early circuit simulation. In *Custom Integrated Circuits Conference*, pages 201–204. IEEE, June 2000.

[20] A. H. Chan and G. W. Roberts. A Synthesizable Fast and High-Resolution Timing Measurement Device using a Component-Invariant Vernier Delay Line. In *International Test Conference*, pages 858–867. IEEE, November 2001.

[21] A. H. Chan and G. W. Roberts. A Deep Sub-micron Timing Measurement Circuit Using a Single Stage Vernier Delay Line. In *Custom Integrated Circuits Conference*, pages 77–80. IEEE, May 2002.

[22] J. T.-Y. Chang and E. J. Mccluskey. Detecting Delay Flaws by Very-Low-Voltage Testing. In *International Test Conference*, pages 367–376. IEEE, October 1996.

[23] B. Chatterjee, M. Sachdev, and A. Keshavarzi. A DFT Technique for Low Frequency Delay Fault Testing in High Performance Digital Circuits. In *International Test Conference*, pages 1130–1139. IEEE, October 2002.

[24] P. Chen, S.-I. Liu, and J. Wu. A Low Power High Accuracy CMOS Time-to-Digital Converter. In *International Symposium onCircuits and Systems*, pages 281–284. IEEE, June 1997.

[25] P. Chen, S.-I. Liu, and J. Wu. Highly accurate cyclic CMOS time-to-digital converter with extremely low power consumption. *IEEE Electronics Letters*, 33(10):858–860, May 1997.

[26] P. Chen, S.-I. Liu, and J. Wu. A CMOS Pulse-Shrinking Delay Element for Time Interval Measurement. *IEEE Transactions on Circuits and Systems II*, 47(9):954–958, September 2000.

[27] K.-H. Cheng, S.-Y. Jiang, and Z.-S. Chen. BIST for Clock Jitter Measurements. In *International Symposium on Circuits and Systems*, pages V 577–V 580. IEEE, May 2003.

[28] K.-T. Cheng, S. Devadas, and K. Keutzer. Delay-Fault Test Generation and Synthesis for Testability under a Standard Scan Design Methodology. *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, 12(8):1217–1231, August 1993.

[29] A. C. L. Chiang, I. S. Reed, and A. V. Banes. Path Sensitization, Partial Boolean Difference and Automated Fault Diagnosis. *IEEE Transactions on Computers*, C-21(2):189–195, February 1972.

[30] J. Christiansen. An Integrated CMOS 0.15ns Digital Timing Generator for TDCs and Clock Distribution Systems. *IEEE Transactions on Nuclear Science*, 42(4):753–757, August 1995.

[31] L. Dadda. Some Schemes for Parallel Multipliers. *Alta Frequenza*, 34:349–356, 1965.

[32] S. Das, D. Roberts, S. Lee, S. Pant, D. Blaauw, T. Austin, K. Flautner, and T. Mudge. A Self-Tuning DVS Processor using Delay-Error Detection and Correction. *IEEE Journal of Solid State Circuits*, 41(4):792–804, April 2006.

[33] R. Datta, J. A. Abraham, A. U. Diril, A. Chatterjee, and K. Nowka. Adaptive Design for Performance-Optimized Robustness. In *International Symposium on Defect and Fault Tolerance*, pages 3–11. IEEE, October 2006.

[34] R. Datta, J. A. Abraham, R. Montoye, W. Belluomini, C. Mcdowell, H. Ngo, J. Kuang, and K. Nowka. A Low Latency Low Power 4-to-2 Carry Save Adder. In *International Symposium on Circuits and Systems*, pages 477–480. IEEE, May 2004.

[35] R. Datta, G. Carpenter, K. Nowka, and J. A. Abraham. A Scheme for On-Chip Timing Characterization. In *VLSI Test Symposium*, pages 24–29. IEEE, May 2006.

[36] R. Datta, R. Gupta, A. Sebastine, J. A. Abraham, and M. Dabreu. TriScan: A Novel DFT Technique for CMOS Path Delay Fault Testing. In *International Test Conference*, pages 1118–1127. IEEE, October 2004.

[37] R. Datta, S. Nassif, R. Montoye, and J. A. Abraham. Testing and Debugging Delay Faults in Dynamic Circuits. In *International Test Conference*, pages 101–110. IEEE, November 2005.

[38] R. Datta, A. Sebastine, and J. A. Abraham. Delay Fault Testing and Silicon Debug Using Scan Chains. In *European Test Symposium*, pages 46–51. IEEE, May 2004.

[39] R. Datta, A. Sebastine, A. Raghunathan, and J. A. Abraham. On-Chip Delay Measurement for Silicon Debug. In *Great Lakes Symposium on VLSI*, pages 145–148. ACM, April 2004.

[40] G. Declerck. A look into the future of nanoelectronics. In *Symposium on VLSI Circuits*, pages 6–10. IEEE, June 2005.

[41] B. Dervisoglu. Design for Testability: It is time to deliver it for Time-to-Market. In *International Test Conference*, pages 1102–1111. IEEE, September 1999.

[42] B. I. Dervisoglu and G. E. Strong. Design for Testability: Using ScanPath Techniques for Path-Delay Test and Measurement. In *International Test Conference*, pages 365–374. IEEE, October 1991.

[43] S. Dhar and D. Maksimovic. Switching Regulator with Dynamically Adjustable Supply Voltage for Low Power VLSI. In *Annual Conference of the IEEE Industrial Electronics Society*, pages 1874–1879. IEEE, November-December 2001.

[44] S. Dhar, D. Maksimovic, and B. Kranzen. Closed-Loop Adaptive Voltage Scaling Controller for Standard-Cell ASICs. In *International Symposium on Low Power Electronic Design*, pages 103–107. IEEE, August 2002.

[45] C. E. Dike, N. A. Kurd, P. Patra, and J. Barkatullah. A Design for Digital Dynamic Clock Deskew. In *Symposium on VLSI Circuits*, pages 21–24. IEEE, June 2003.

[46] P. Dudek, S. Szczepanski, and J. V. Hatfield. A High-Resolution CMOS Time-to-Digital Converter utilizing a Vernier Delay Line. *IEEE Transactions on Solid State Circuits*, 35(2):240–247, February 2000.

[47] S. G. Duvall. Statistical Circuit Modeling and Optmization. In *International Workshop on Statistical Metrology*, pages 56–63. IEEE, June 2000.

[48] E. B. Eichelberger and T. W. Williams. A Logic Design Structure for Design for Testability. In *Design Automation Conference*, pages 462–468. ACM, June 1977.

[49] Y. Elboim, A. Kolodny, and R. Ginosar. A Clock-Tuning Circuit for System-on-Chip. *IEEE Transactions on VLSI*, 11(4):616–626, August 2003.

[50] M. Favalli, P. Olivo, M. Damiani, and B. Ricco. Novel Design for Testability Schemes for CMOS IC's. *IEEE Journal of Solid State Circuits*, 25(5):1239–1246, October 1990.

[51] P. Franco and E. J. McCluskey. Delay Testing of Digital Circuits by Output Waveform Analysis. In *International Test Conference*, pages 798–807. IEEE, October 1991.

[52] J.-F. Genat. High resolution time-to-digital converter. *Nuclear Instruments and Methods in Physics Research*, A315.

[53] J.-F. Genat and F. Rossel. Ultra High-Speed Time-to-Digital Converter. United States Patent no. 4719608, January 1988.

[54] C. T. Glover and M. R. Mercer. A Method for Delay Fault Test Generation. In *Design Automation Conference*, pages 90–95. IEEE, June 1988.

[55] M. S. Gorbics, J. Kelly, K. M. Roberts, and R. L. Sumner. A High-Resolution Multihit Time-to-Digital Converter Integrated Circuit. *IEEE Transactions on Nuclear Science*, 44(3):379–384, June 1997.

[56] C. T. Gray, W. Liu, W. A. V. Noije, T. A. Hughes, and R. K. CavinIII. A Sampling Technique and its CMOS Implementation with 1 Gb/s Bandwidth and 25 ps Resolution. *IEEE Journal of Solid State Circuits*, 29(3):340–349, March 1994.

[57] M. C. Hansen, H. Yalcin, and J. P. Hayes. Unveiling the ISCAS-85 Benchmarks: A Case Study in Reverse Engineering. *IEEE Design and Test of Computers*, 16(3):72–80, July-September 1999.

[58] C. Hawkins, A. Keshavarzi, and J. Segura. A View from the Bottom: Nanometer Technology AC Parametric Failures - Why, Where, and How to Detect. In *International Symposium on Defect and Fault Tolerance*, pages 267–276. IEEE, November 2003.

[59] S. Hesley, V. Andrade, B. Burd, G. Constant, J. Correll, M. Crowley, M. Golden, N. Hopkins, S. Islam, S. Johnson, R. Khondker, D. Meyer, J. Moench, H. Partovi, R. Posey, F. Weber, and J. Yong. A 7th-Generation x86 Microprocessor. In *International Solid State Circuits Conference*, pages 92–93. IEEE, February 1999.

[60] M.-J. Hsiao, J.-R. Huang, S.-S. Yang, and T.-Y. Chang. A Built-In Timing Parametric Measurement Unit. In *International Test Conference*, pages 315–322. IEEE, November 2001.

[61] E. P. Hsieh, R. A. Rasmussen, L. J. Vidunas, and W. T. Davis. Delay Test Generation. In *Design Automation Conference*, pages 486–491. ACM, June 1977.

[62] N. K. Jha. Testing for Multiple Faults in Domino-CMOS Logic Circuits. *IEEE Transactions on Computer Aided Design*, 7(1):109–116, January 1988.

[63] V. V. Kaenel, P. Macken, and M. G. R. Degrauwe. A Voltage Reduction Technique for Battery-Operated Systems. *IEEE Journal of Solid State Circuits*, 25(5):1136–1140, October 1990.

[64] A. Kapoor, N. Jayakumar, and S. P. Khatri. A Novel Clock Distribution and Dynamic De-skewing Methodology. In *International Conference on Computer Aided Design*, pages 626–631. IEEE, November 2004.

[65] C. H. Kim, K. Roy, S. Hsu, A. Alvandpour, R. K. Krishnamurthy, and S. Borkar. A Process Variation Compensating Technique for Sub-90nm Dynamic Circuits. In *Symposium on VLSI Circuits*, pages 205–206. IEEE, June 2003.

[66] A. Kinra. Towards Reducing Functional Only Fails for the UltraSPARC Microprocessors. In *International Test Conference*, pages 147–154. IEEE, September 1999.

[67] C. Kothandaraman, S. K. Iyer, and S. S. Iyer. Electrically Programmable Fuse (eFUSE) using Electromigration in Silicides. *IEEE Electron Device Letters*, 23(9):523–525, September 2002.

[68] A. Krstic and K. Cheng. *Delay Fault Testing for VLSI Circuits*. Kluwer Academic Publishers, Boston, Massachusetts, 1998.

[69] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar. Mathematically Assisted Adaptive Body Bias (ABB) for Temperature Compensation in Gigascale LSI Systems. In *Asia and South Pacific Conference on Design Automation*, pages 559–564. IEEE, January 2006.

[70] R. Kundu and R. D. Blanton. Timed Test Generation for Crosstalk Switch Failure in Domino CMOS. In *VLSI Test Symposium*, pages 379–385. IEEE, April 2002.

[71] R. Kundu and R. D. Blanton. Path Delay Test Generation for Domino Logic Circuits in the Presence of Crosstalk. In *International Test Conference*, pages 122–130. IEEE, September 2003.

[72] T. Kuroda, K. Suzuki, S. Mita, T. Fujita, F. Yamane, F. Sano, A. Chiba, Y. Watanabe, K. Matsuda, T. Maeda, T. Sakurai, and T. Furuyama. Variable Supply-Voltage Scheme for Low-Power High-Speed CMOS Digital Design. *IEEE Journal of Solid State Circuits*, 33(3):454–462, March 1998.

[73] K. T. Lee and J. A. Abraham. Critical Path Identification and Delay Tests of Dynamic Circuits. In *International Test Conference*, pages 421–430. IEEE, September 1999.

[74] J. P. Lesser and J. J. Shedletsky. An Experimental Delay Test Generator for LSI Logic. *IEEE Transactions on Computers*, 29(3):235–248, March 1980.

[75] Y. Levendel and P. Menon. Transition Faults in Combinational Circuits: Input Transition Test Generation and Fault Simulation . In *International Fault Tolerant Computing Symposium*, pages 278–283. IEEE, July 1986.

[76] Y. Levendel and P. R. Menon. Transition Faults in Combinational Circuits: Input Transition Test Generation and Fault Simulation . In *International Fault Tolerant Computing Symposium*, pages 278–283. IEEE, July 1986.

[77] C. J. Lin and S. M. Reddy. On Delay Fault Testing in Logic Circuit. *IEEE Transactions on Computer Aided Design*, 6(5):183–190, September 1987.

[78] X. Lin, R. Press, J. Rajski, P. Reuter, T. Rinderknecht, B. Swanson, and N. Tamarapalli. High-Frequency, At-Speed Scan Testing. *IEEE Design and Test of Computers*, 20(5):17–25, September-October 2003.

[79] C. Ljuslin, J. Christiansen, A. Marchioro, and O. Klingsheim. An Integrated 16-channel CMOS Time to Digital Converter. *IEEE Transactions on Nuclear Science*, 41(4):1104–1108, August 1994.

[80] R. Madge. New test paradigms for yield and manufacturability. *IEEE Design and Test of Computers*, 22(3):240–246, May-June 2005.

[81] Y. K. Malaiya and R. Narayanaswamy. Modeling and Testing for Timing Faults in Synchronous Sequential Circuits. *IEEE Design and Test of Computers*, 1(6):62–74, November-December 1984.

[82] W. Maly and P. Nigh. Built-In Current Testing - Feasibility Study. In *International Conference on Computer Aided Design*, pages 340–343. IEEE, November 1988.

[83] W. Mao and M. D. Ciletti. A Variable Observation Time Method for Testing Delay Faults. In *Design Automation Conference*, pages 728–731. ACM/IEEE, July 1990.

[84] W. Mao and M. D. Ciletti. Arrangement of Latches in Scan-path Design to Improve Delay Fault Coverage. In *International Test Conference*, pages 387–393. IEEE, September 1990.

[85] P. Maxwell, I. hartanto, and L. Bentz. Comparing Functional and Structural Tests. In *International Test Conference*, pages 400–407. IEEE, October 2000.

[86] P. C. McGeer. Robust Path Delay-Fault Testability on Dynamic CMOS Circuits. In *International Conference on Computer Design: VLSI in Computers and Processors*, pages 206–211. IEEE, October 1991.

[87] L. McMurchie, S. Kio, G. Yee, T. Thorp, and C. Sechen. Ouput Prediction Logic: a high-performance CMOS design technique. In *International Conference on Computer Design*.

[88] S. Mitra, N. Seifert, M. Zhang, Q. Shi, and K. S. Kim. Robust System Design with Built-In Soft-Error Resilience. *IEEE Computer*, 38(2):43–52, February 2005.

[89] M. Miyazaki, G. Ono, and K. Ishibashi. A 1.2-GIPS/W Microprocessor using Speed-Adaptive Threshold-Voltage CMOS with Forward Bias. *IEEE Journal of Solid State Circuits*, 37(2):210–217, February 2002.

[90] R. Montoye, W. Belluomini, H. Ngo, C. McDowell, J. Sawada, T. Nguyen, B. Veraa, J. Wagoner, and M. Lee. A double-precision floating point multiply. In *International Solid State Circuits Conference*, pages 104–105. IEEE, February 2003.

[91] S. Mutoh, T. Douseki, Y. Matsuya, T. Aoki, S. Shigematsu, and J. Yamada. 1-V Power Supply High-Speed Digital Ciruit Technology with Multithreshold-Voltage CMOS. *IEEE Journal of Solid State Circuits*, 30(8):847–853, August 1995.

[92] S. Naffziger, B. Stackhouse, and T. Grutkowski. The Implementation of a 2-core Multi-Threaded Itanium Processor. In *International Solid State Circuits Conference*, pages 182–184. IEEE, February 2005.

[93] S. Naffziger, B. Stackhouse, T. Grutkowski, D. Josephson, J. Desai, E. Alon, and M. Horowitz. 1-V Power Supply High-Speed Digital Ciruit Technology with Multithreshold-Voltage CMOS. *IEEE Journal of Solid State Circuits*, 41(1):197–209, January 2006.

[94] M. Nakai, S. Akui, K. Seno, T. Meguro, T. Seki, T. Kondo, A. Hashiguchi, H. Kawahara, K. Kumano, and M. Shimura. Dynamic Voltage and Frequency Management for a Low-Power Embedded Microprocessor. *IEEE Journal of Solid State Circuits*, 40(1):28–35, January 2005.

[95] S. R. Nassif. Modeling and Forecasting of Manufacturing Variations. In *International Workshop on Statistical Metrology*, pages 2–10. IEEE, June 2000.

[96] S. Natrajan, S. K. Gupta, and M. A. Breuer. Switch Level Delay Test of Domino Logic Circuits. In *International Test Conference*, pages 367–376. IEEE, November 2001.

[97] W. Needham and N. Gollakota. DFT Strategy for Intel Microprocessors. In *International Test Conference*, pages 396–399. IEEE, October 1996.

[98] Needham et. al. High Volume Microprocessor Test Escapes, An Analysis of Defects Our Tests are Missing. In *International Test Conference*, pages 25–34. IEEE, October 1998.

[99] H. Ngo, W. Belluomini, and R. K. Montoye. Circuits and systems for limited switch dynamic logic. United States Patent Application no. 20030189445, October 2003.

[100] K. J. Nowka and T. Galambos. Circuit design techniques for a gigahertz integer microprocessor. In *International Conference on Computer Design*, pages 11–16. IEEE, October 1998.

[101] P. Parvathala, K. Maneparambil, and W. Lindsay. FRITS - A Microprocessor Functional BIST Method. In *International Test Conference*, pages 590–598. IEEE, October 2002.

[102] D. Pham, S. Asano, M. Bolliger, M. Day, H. P. Hofstee, C. Johns, J. Kahle, A. Kameyama, J. Keaty, Y. Masubuchi, M. Riley, D. Shippy, D. Stasiak, M. Suzuoki, M. Wang, J. Warnock, S. Weitzel, D. Wendel, T. Yamazaki, and K. Yazawa. The Design and Implementation of a First generation CELL

processor. In *International Solid State Circuits Conference*, pages 184–186. IEEE, February 2005.

[103] A. K. Pramanick and S. M. Reddy. On the Computation of the Ranges of Detected Delay Fault Sizes. In *International Conference on Computer Aided Design*, pages 126–129. IEEE, November 1989.

[104] K. Raahemifar and M. Ahmadi. Design for Testability Techniques for Detecting Delay Faults in CMOS/BiCMOS Logic Families. *IEEE Transactions on Circuits and Systems -II*, 47(11):1279–1290, November 2000.

[105] J. M. Rabaey, A. Chandrakasan, and B. Nikolic. *Digital Integrated Circuits: A Design Perspective.* Pearson Education, Upper Saddle River, New Jersey, 2002.

[106] T. Rahkonen and J. Kostamovaara. Pulsewidth Measurement using an Integrated Pulse Shrinking Delay Line. In *International Symposium onCircuits and Systems*, pages 578–581. IEEE, May 1990.

[107] T. Rahkonen, J. Kostamovaara, and S. Saynajakangas. CMOS ASIC Devices for the Measurement of Short Time Intervals. In *International Symposium on Circuits and Systems*, pages 1593–1596. IEEE, June 1988.

[108] T. Rahkonen and J. T. Kostamovaara. The Use of CMOS Delay lines for Digitization of Short Time Intervals. *IEEE Journal of Solid State Circuits*, 28(8):887–894, August 1993.

[109] T. Rahkonen, E. Malo, and J. Kostamovaara. A 3V Fully Integrated Digital FM Demodulator Based on a CMOS Pulse-Shrinking Delay Line. In *International Symposium onCircuits and Systems*, pages 572–575. IEEE, May 1996.

[110] E. Raisanen-Ruotsalainen, T. Rahkonen, and J. Kostamovaara. A Low-Power CMOS Time-to-Digital Converter. *IEEE Journal of Solid State Circuits*, 30(9):984–990, September 1995.

[111] A. Raychowdhury, S. Ghosh, S. Bhunia, D. Ghosh, and K. Roy. A Novel Delay Fault Testing Methodology Using On-Chip Low Overhead Delay Measurement Hardware at Strategic Test Points. In *European Test Symposium*, pages 108–113. IEEE, May 2005.

[112] A. Raychowdhury, S. Ghosh, and K. Roy. A Novel On-Chip Delay Measurement Hardware for Efficient Speed-Binning. In *International On-Line Testing Symposium*, pages 287–292. IEEE, July 2005.

[113] H. Ren, D. Z. Pan, and D. S. Kung. Sensitivity Guided Net Weighting for Placement Driven Synthesis. In *International Symposium on Physical Design*, pages 10–17. ACM, April 2004.

[114] R. Rosing, A. M. D. Richardson, Y. E. Aimine, H. G. Kerkhoff, and A. J. Acosta. Clock Switching: A New Design for current Testability (DcT)

Method for Dynamic Logic Circuits. In *International Workshop on IDDQ Testing*, pages 20–25. IEEE, November 1998.

[115] M. Sachdev, P. Janssen, and V. Zieren. Defect Detection with Transient Current Testing and its Potential for Deep Sub-micron CMOS ICs. In *International Test Conference*, pages 204–213. IEEE, October 1998.

[116] O. Sasaki, T. Taniguchi, T. K. Ohska, H. Mori, T. Nonaka, K. Kaminishi, A. Tsukuda, H. Nishimura, M. Takeda, and Y. Kawakami. 1.2ghz GaAs Shift Register IC for Dead-Time-Less TDC Application. *IEEE Transactions on Nuclear Science*, 36(1):512–516, February 1989.

[117] J. Savir. Skewed Load transition Test: Part I, Calculus. In *International Test Conference*, pages 705–713. IEEE, September 1992.

[118] J. Savir. Skewed Load transition Test: Part II, Coverage. In *International Test Conference*, pages 714–722. IEEE, September 1992.

[119] J. Savir. On Broad-Side Delay test. In *VLSI Test Symposium*, pages 284–290. IEEE, April 1994.

[120] J. Saxena, K. M. Butler, J. Gatt, R. Raghuraman, S. P. Kumar, S. Basu, D. J. Campbell, and J. Berech. Scan-Based Transition Fault Testing - Implementation and Low Cost Test Challenges. In *International Test Conference*, pages 1120–1129. IEEE, October 2002.

[121] J. Segura, A. Keshavarzi, J. Soden, and C. Hawkins. Parametric Failures in CMOS ICs - A Defect-Based Analysis. In *International Test Conference*, pages 90–99. IEEE, October 2002.

[122] K. L. Shepard and V. Narayanan. Noise in Deep Submicron Digital Design. In *International Conference on Computer-Aided Design*, pages 524–531. IEEE, November 1996.

[123] J. Silberman, N. Aoki, D. Boerstler, J. L. Burns, S. Dhong, A. Essbaum, U. Ghoshal, D. Heidel, P. Hofstee, K. T. Lee, D. Meltzer, H. Ngo, K. Nowka, S. Posluzny, O. Takahashi, I. Vo, and B. Zoric. A 1.0 GHz single-issue 64-bit PowerPC integer processor. *IEEE Journal of Solid State Circuits*, 33(11):1600–1608, November 1998.

[124] Silicon Ensemble. *Auto Place and Route Tool.*

[125] A. Stevens, R. P. Vanberg, J. V. D. Spiegel, and H. H. Williams. A Time-to-Voltage Converter and Analog Memory for Colliding Beam Detectors. *IEEE Journal of Solid State Circuits*, 24(6):1748–1752, December 1989.

[126] V. Stojanovic and V. Oklobdzija. Comparative Analysis of Master-Slave Latches and Flip-Flops for High-Performance and Low Power Systems. *IEEE Journal of Solid State Circuits*, 34(4):536–548, April 1999.

[127] C. Su, Y.-T. Chen, M.-J. Huang, G.-N. Chen, and C.-L. Lee. All Digital Built-in Delay and Crosstalk Measurement for On-Chip Buses. In *Design*

Automation and Test in Europe Conference and Exhibition, pages 527–531. IEEE, March 2000.

[128] Synopsis Inc. *Primetime Reference - Version 2000.11*, November 2000.

[129] S. Tam, R. D. Limaye, and U. N. Desai. Clock Generation and Distribution for the 130-nm Itanium 2 Processor with 6-MB On-Die L3 Cache. *IEEE Journal of Solid State Circuits*, 39(4):636–642, April 2004.

[130] R. C. Tekumalla and P. R. Menon. Delay Testing with Clock Control: An Alternative to Enhanced Scan. In *International Test Conference*, pages 454–462. IEEE, November 1997.

[131] S. Tisa, A. Lotito, A. Giudice, and F. Zappa. Monolithic Time-to-Digital Converter with 20ps resolution. In *European Solid State Circuits Conference*, pages 465–468. IEEE, September 2003.

[132] N. A. Touba and E. J. McCluskey. Applying Two Pattern Tests usign Scan-Mapping. In *VLSI Test Symposium*, pages 393–397. IEEE, May 1996.

[133] J. W. Tschanz, J. T. Kao, S. G. Narendra, R. Nair, D. A. Antoniadis, A. P. Chandrakasan, and V. De. Adaptive Body Bias for reducing Impacts of Die-to-Die and Within-Die Parameter Variations on Microprocessor Frequency and Leakage. *IEEE Journal of Solid State Circuits*, 37(11):1396–1402, November 2002.

[134] University of Texas at Austin. *EE382M VLSI-2 Class Notes “http://www.ece.utexas.edu/ mcdermot”*.

[135] University of Texas at Austin. *VLSI Testing Class Notes “http://www.ece.utexas.edu/ touba”*.

[136] B. Vermeulen and S. K. Goel. Design for Debug: Catching Design Errors in Digital Chips. *IEEE Design and Test of Computers*, 19(3):37–45, May-June 2002.

[137] C. Wallace. A Suggestion for a Fast Multiplier. *IEEE Transactions on Computers*, 13:14–17, 1964.

[138] K. P. White, R. N. Athay, and W. J. Trybula. Applying DFM in the Semiconductor Industry. In *International Electronics Manufacturing Technology Symposium*, pages 438–441. IEEE/CPMT, October 1995.

[139] T. J. Wood. The Test and Debug Features of the AMD-K7^{TM} Microprocessor. In *International Test Conference*, pages 130–136. IEEE, April 1999.

[140] W. C. Wu, C. L. Lee, M. S. Wu, J. E. Chen, and M. S. Abadir. Oscillation Ring Delay Test for High Performance Microprocessors. *Journal of Electronic Testing: Theory and Applications*, 16(1-2):147–155, February 2000.

Vita

Ramyanshu Datta was born on March 15th, 1980 in Calcutta (now renamed Kolkata) in the state of West Bengal in India, the son of Sushmita and Biswajit Datta. He did his primary schooling in Calcutta and moved to Bombay (now renamed Mumbai) at the age of 9. He completed his schooling from Greenlawns High School in 1995 and Jai Hind Junior College in 1997, both in Mumbai and joined Pune Institute of Computer Technology, affiliated to Pune University in India for a Bachelor's degree in Electronics and Telecommunications Engineering. He graduated with a Bachelor of Engineering degree in 2001 and joined the graduate program in Electrical Engineering at University of Southern California in Los Angeles, CA in January 2002. Subsequently, in September 2002 he transferred to The University of Texas at Austin for pursuing graduate studies in Electrical and Computer Engineering. He received his Master's degree in Electrical and Computer Engineering from The University of Texas in December 2004. He has held several internship positions with IBM Corporation during his graduate studies, both in the Research Division at IBM Austin Research Laboratory as well as in the Sony-Toshiba-IBM (STI) Design Center for the CELL processor. His interests lie in the area of VLSI design and test and he has several patents (issued or pending) and publications in these areas.

Permanent address: B 702 People's Cosmopolitan,
St. Teresa Road, Bandra (W)
Mumbai 400050, India

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

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