On Using Programmable Delay Tuning Elements To Improve Performance, Reliability, and Testing of Digital ICs

ON USING PROGRAMMABLE DELAY TUNING ELEMENTS TO IMPROVE PERFORMANCE, RELIABILITY, AND TESTING OF DIGITAL ICS

BY

ZAHRA LAK,

B.A.Sc., (Computer Engineering),

M.A.Sc., (Computer Architecture)

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

AND THE SCHOOL OF GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

Doctor of Philosophy (2012) McMaster University
(Electrical and Computer Engineering) Hamilton, Ontario, Canada

TITLE: On Using Programmable Delay Tuning Elements To Improve Performance, Reliability, and Testing of Digital ICs

AUTHOR: Zahra Lak
B.A.Sc., (Computer Engineering)
Amirkabir University of Technology, Tehran, Iran
M.A.Sc., (Computer Architecture)
Amirkabir University of Technology, Tehran, Iran

SUPERVISOR: Dr. Nicola Nicolici

NUMBER OF PAGES: xvi, 151

To Kaveh,

the everlasting inspiration of my life.

Abstract

The number of speed-limiting paths in modern digital integrated circuits (ICs) is in the range of millions. Due to un-modelled electrical effects and process variations in advanced fabrication technologies, it is difficult for pre-silicon timing analysis tools to provide accurate delay estimates. Hence, programmable delay elements are commonly inserted in high-performance circuits in order to provide a tuning mechanism at the post-silicon phase. Due to the large number of such tuning elements, finding the appropriate configuration bits for each element mandates an automated approach.

In this thesis we present three contributions to the field of digital IC design automation that leverage the presence of programmable delay tuning elements. These new automated approaches are geared toward three distinct objectives. The first one is to maximize the circuit performance using a scalable algorithmic framework. The second objective is to combat the lifetime performance degradation caused by circuit aging. The final objective is to improve the timing of the scan enable signal during the at-speed testing of digital ICs.

As the programmable delay tuning elements will become prevalent in the future generations of digital ICs, the contributions from this thesis will help improve the design methodologies that are expected to evolve in order to address at runtime the timing problems introduced by the increased fabrication process variability.

Acknowledgements

This work would not have been possible without several individuals who in one way or another contributed and extended their valuable assistance.

I am appreciative to my committee members, Dr. David Capson and Dr. Shahram Shirani for their valuable time and feedback during my research which helped to mature this work. I am also thankful to my external examiner Dr. Vikram Iyengar who provided very nice and insightful comments on my thesis which helped to improve the work.

I am also thankful to many friends who have been always supportive and helpful during the course of my study, especially those in the Computer Aided Design and Test research group of McMaster: Ho Fai (Henry) Ko, Adam Kinsman, Ehab Anis, Kaveh Elizeh, Roomi Sahi, Mark Jobes, Jason Thong, Phillip Kinsman, Xiaobing Shi, Amin Vali, and Pouya Taatizadeh. To them I owe special thanks for much time spent on interesting technical discussions, advice, and support. To my supervisor, Dr. Nicola Nicolici, I am deeply grateful for his thoughtful advice, insightful criticism and patient encouragement through the duration of my studies. He was always providing the greatest environment to work within and has spent much time and effort guiding me in research and in life, and this work would not have been possible without him.

I owe thanks to my brother-in-law Khosrow and my sister Zohreh for their emotional support and providing a relaxing environment. To my parents I am appreciative for teaching

me the important values of life and for supporting me through difficult moments in life. My husband Kaveh deserves a great deal of thanks for his everlasting support which has made this work possible.

I am also thankful to the administrative and technical staff of the ECE department at McMaster for much practical assistance.

Above all I am grateful to God for giving me the ability and passion to do research, and for placing these people in my life.

Glossary

Abstraction removal of detail to facilitate design description

ATE automated test equipment, used to apply test patterns to ICs

ATPG automatic test pattern generation, deriving of fault based test patterns

At-speed operating with the same clock period as used in normal operation

Automation handing over of control to computers for management of design details

Branch and bound an methodology for finding optimal solution an optimization problem

CAD computer aided design, tools for design automation (also EDA)

Calibration pattern a vector of circuit input values applied with the intent of calibrating the scan enable signal

Circuit aging degradation of circuit performance over time

Cluster a group of state elements sharing an element in circuit

Core a pre-designed and pre-verified module which can be reused

Critical path slowest path in a circuit determining performance

CUT circuit (or core) under test

CVD clock vernier device, an example of delay tuning elements

CVD configuration value of configuration elements in a CVD

CVD step minimum delay between two discrete CVD values

CVD value delay value provided by the configuration assigned to a CVD

Design verification comparison of design against specification

DFT design for test, steps/hardware added in design to facilitate test

EDA electronic design automation, tools for design automation (also CAD)

Fast path a path with short propagation delay

Fault a model which captures the effect of one or many types of defects

Fault coverage the percentage of faults detected by a test set

Functional test tests focused on circuit operation, in contract to structural test

HVP hold-time violation prediction

IC integrated circuit, many transistors integrated in a single package, a device

ISCAS'89 benchmarks benchmark circuits widely used in literature on test

Layout low level description of a circuit in terms of transistor placements

LFSR linear feedback shift register, a type of PRPG

LOS launch-on-shift, one of the at-speed testing techniques

Manufacturing test comparison of manufactured devices against the design

MISR multiple input signature register

Moore's law an empirical observation of exponential transistor density growth

Netlist a low-mid level description of a circuit in terms of logic gate connections

Post-silicon after fabrication, the actual product is available

Pre-silicon before fabrication, the actual product is not available

Reliability reliable operation of a circuit after fabrication

Scalability the ability to cope with increasing circuit size, VTD, etc.

Scan access to the internal state registers of an IC

Scan chains shift register structure facilitating scan

Scan segment a group of consecutive scan cells from scan chain

SCR signature of the captured response

SISR single input signature register

Slow path a path with long propagation delay

Specification the formal description of the design requirements for a system

Speedpath a path with long propagation delay, speed limiting path

SRR signature of reference response

STA static timing analyzer

Stuck-at-1 a fault model assuming a node shorted to power

SVP setup-time violation prediction

Synthesis transformation of a description to a lower level of abstraction

Test application process of applying test vectors to a circuit and collecting the responses

Test pattern a vector of circuit input values applied with the intent of testing

Test set a collection of test patterns

VLSI very large scale integrated circuits, containing millions of transistors

Contents

List of Figures

Chapter 1

Introduction

Advanced process technologies enable very large scale integrated (VLSI) circuits to be built with hundreds of millions and even billions of transistors. However, design and verification of digital integrated circuits (ICs) in the range of millions of transistors are not tasks that can be fulfilled manually and they need to be performed by means of sophisticated design methods and tools. In addition, after the design of a circuit is completed, each manufactured sample needs to be tested rapidly in order to screen out the failing devices caused by fabrication defects. Furthermore, because digital ICs are at the core of the computational engines that are pervasive in the modern society, it is also important to ensure that this type of devices operate reliably over time. In order to facilitate a better understanding of the challenges that have motivated the contributions presented in this thesis, it is essential to introduce the basics of digital ICs, as well as the design and test flows.

In this chapter we briefly cover the basic background and the trends that have enabled the proliferation of computational engines. Subsequently, we elaborate on the design flow for digital ICs and then we discuss a few challenges that are faced before and after a circuit is built, which are the key motivations for the work presented in this thesis.

1.1 Computational Engines

The invention of computational engines dates back to the 19th century [59]. In 1822, Babbage planned to build a mechanical calculator, which removes the sources of mistakes in mathematical computations and he called it *Difference Engine* [30]. Difference engines were based on decimal number systems rather than binary number systems that are used in todays electronic systems. However, high design and implementation cost as well as high area and power consumption made the growth of mechanical engines impractical.

Afterwards, the electric relay-based computers were introduced. Most of the computational machines at that time were based on discrete components such as resistors, capacitors, or diodes. Thereafter, digital electronic computing machines were invented based on vacuum tubes. However, the excessive design cost and power consumption of these computational machines was placing an upper limit on the *integration density* (the number of computational devices per unit of area), thus making it impractical to build machines that could handle much more computations than before. This stagnation was happening at the same time when the transistor was invented [10].

After the invention of transistors, the first *integrated circuit* was created at 1958, in which all components were integrated on the same semiconductor substrate [59]. Then, different types of IC families were implemented with objectives such as working at higher speed or consuming less power. It was in 1965 when Gordon Moore predicted that the transistor count on a single die (a surface that contains an IC) will double each year, as shown in Figure 1.1. This prediction that was later called *Moore's Law* [45] has proven to be visionary, as it has reliably predicted the growth for the transistor count per die over the past four and a half decades [59]. Nonetheless, when the integration density increases and designs become more complex, the design and manufacturing techniques need to evolve.

For the early designs for which the circuit structure was simple and the number of transistors per die was small, it was possible for the designer to lay out every single transistor manually. However, when considering the growth of the circuit size and the time to market pressure, designers needed to improve their design methods to handle these constraints, while also keeping up with the requirements for improving the circuit speed and power.

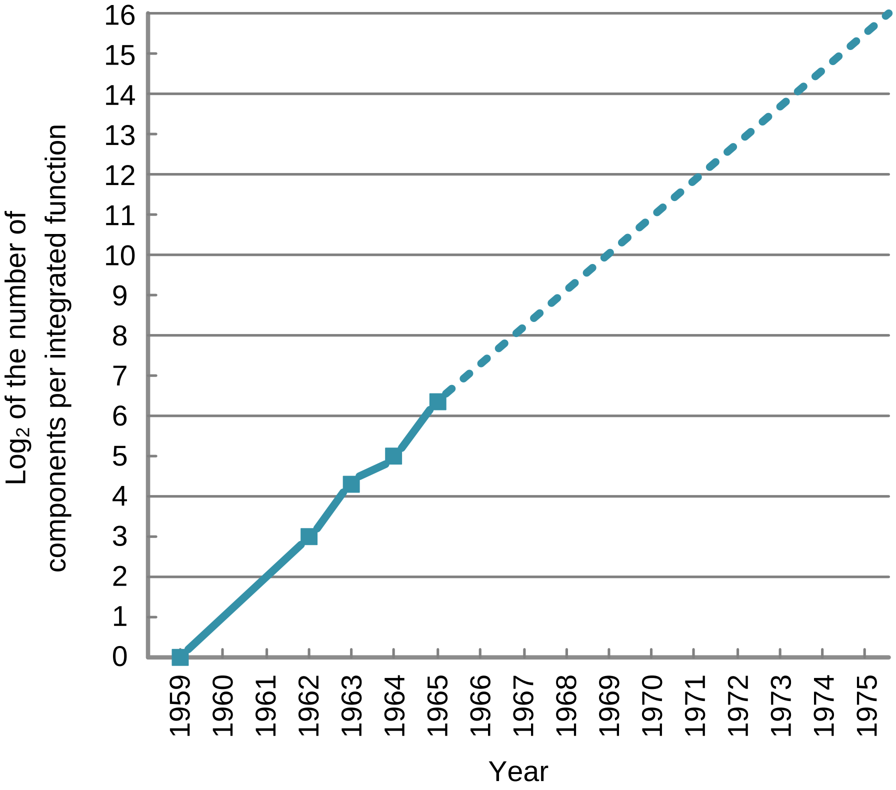


Figure 1.1: Trends in the number of components per integrated function [45].

Computer-Aided Design (CAD) techniques have naturally evolved together with design methodologies in order to facilitate the data handling during the design phase, reduce time to market and help optimize the circuit [44] in terms of area, operational speed, and power consumption. Many CAD techniques have been developed for multiple steps for the design and optimization of electronic circuits and they are still expanding and improving. The reason is that by the continuous growth of the circuit size, more problems will arise and the demand for extended capability of CAD tools increases [51]. As a consequence, over the past five decades there has been continuous research in the field of CAD that has addressed different stages in the design and optimization of digital ICs, with a constant emphasis on scalable tool development [44].

In the following, we will explain different steps in the design of digital ICs and their challenges, which motivate the need for quality metrics that guide the design process. All these steps, almost exclusively, are performed by the use of CAD tools.

1.2 Lifecycle of A Digital Circuit

In the following, we will discuss the lifecycle of a digital circuit from the time that its requirements are provided until the digital circuit is fabricated and it operates in a system.

1.2.1 Design Flow

The conceptual digital IC design flow is shown in Figure 1.2.

- **Behavioural Design:** Initially, based on the digital circuit requirements, its specifications are extracted. Then, the design flow starts with the designer generating a behavioural model such as a flow chart or a pseudocode based on the circuit's specifications [51]. Afterwards, by providing the required inputs to the model and observing the output, the designer specifies the correctness of the functionality of the circuit's behavioural model.

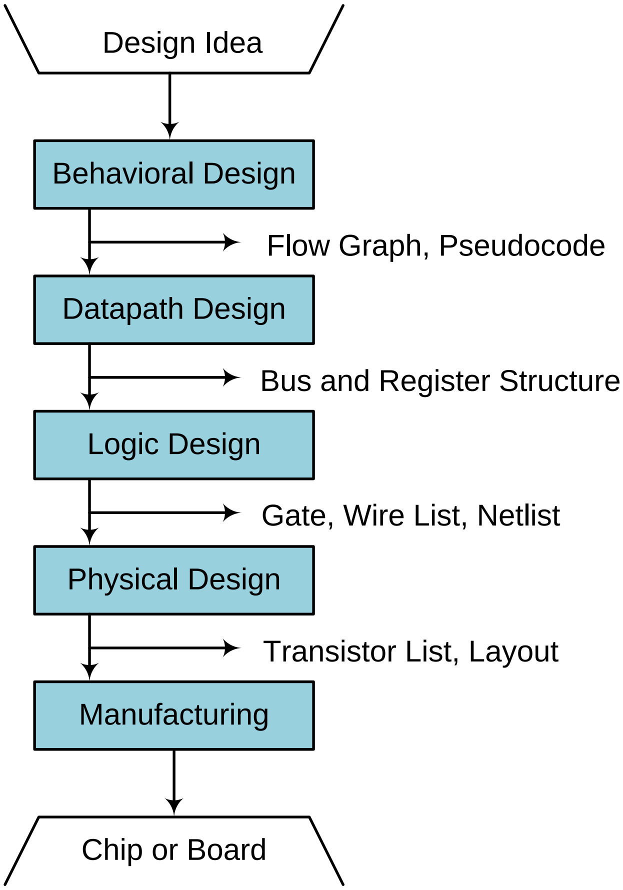


Figure 1.2: A digital circuit design flow [51].

- **Datapath Design:** In this step of the design flow, the datapath of the circuit is designed. In fact, the components of the digital circuit, registers and the interconnections between them are determined [51] and the functionality of each component is specified without providing implementation details. Also, the control logic that manages the movement of data between components is developed. An example of the output of datapath design phase is displayed in Figure 1.3, which illustrates the architecture of the digital circuit.

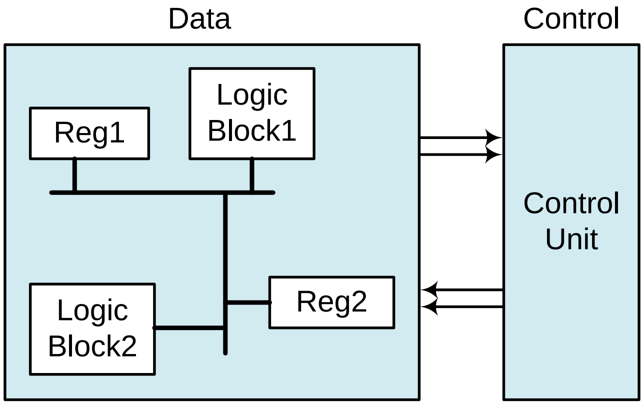


Figure 1.3: An example of datapath phase from Figure 1.2.

- **Logic Design:** In this phase of the design flow, more detailed information about the different components mentioned in the previous phase will be provided. For example, details about the usage of gates and flip-flops will be provided for logic blocks and registers. Also, busses and interconnections as well as the control unit will be implemented and the netlist of the design with gates, flip-flops, and their interconnections will be generated [51].

- **Physical Design:** In this stage of the design flow, gates and flip-flops of the netlist will be transformed into their implementation with transistors, and a layout of transistors including the interconnections between them will be produced.

- **Manufacturing:** In the manufacturing step, a set of photomasks derived from the circuit layout will be used to guide the fabrication steps (e.g., deposition, etching, etc [17]) in order to manufacture the circuit samples.

After a digital circuit is designed and fabricated, there is a screening process to figure out whether each fabricated sample is functional or not. This process will be briefly explained next.

1.2.2 Testing of Digital Circuits

The verification step that checks if each fabricated circuit sample matches the implementation is called *testing*. VLSI test is a well-researched topic with numerous techniques and algorithms used to facilitate the testing process [11], [44], [73]. To provide the *testability* feature to the circuits, additional functions may be added to the circuit during the design-time; this task is called *design for testability (DFT)* [11]. It should be noted that the testability of a circuit determines the level of assurance for the correctness of a circuit [70]. It means that the more a circuit is testable, the more it can be tested effectively and thus achieve a high coverage for different types of faults, such as the stuck-at fault model [73].

After a circuit passes the testing process, it can start operating in the target application environment. Next, we summarize the expectations during the lifetime operation of an IC.

1.2.3 Reliability

After a digital circuit starts its operation, it is expected to operate reliably and maintain correct behaviour [14]. If a circuit cannot operate as expected, it might have been affected by factors such as noise, timing issues, or other problems that have influenced its reliability. One of the most common issues affecting the reliability of a circuit is *circuit aging* [7], [28] and it refers to the degradation of circuit functionality over time. Circuit aging can, for example, slow down the operation speed of transistors, which may affect the operating frequency of the circuit as well. In extreme cases, circuit aging may even cause functional failures to occur over time. Hence, similar to the extra functions added for testability [8], [9], [78], designers may add functions during the design-time to improve the reliability during circuit's lifetime.

In the following, we discuss some of the challenges faced during the design, testing, and lifetime of digital circuits to improve their quality.

1.3 Challenges in Digital ICs' Lifecycle

Each digital circuit has its own characteristics such as cost, energy consumption, functionality, size, operational speed, and robustness [59]. Based on the application that the digital circuit has been designed for, one or multiple of these properties will have varying levels of importance. Hence, each digital circuit should satisfy the quality metrics that are defined for its particular application. For example, for a handheld mobile device, energy consumption is a crucial metric; and, in a server computer, speed is the dominant metric that should be optimized.

Traditionally, properties such as speed could be estimated during the design-time of a digital circuit and those estimations were precise enough compared to the real circuit speed after fabrication. However, advanced manufacturing process technologies enabled VLSI circuits to be built with millions of transistors. This increased design complexity, combined with the aggressive circuit optimizations, as well as variations caused by fabrication process of digital circuits, increase the gap between the design-time *(pre-silicon)* delay estimations provided by static timing analyzers (STA) and the real delay values observed after fabrication *(post-silicon)*. This inaccuracy of pre-silicon delay estimations will result in unpredictable circuit characteristics related to timing after fabrication. For instance, the operational speed calculated by pre-silicon timing analyzers does not account precisely for process variations. Hence, even if no fabrication defects occur, this lack of accurate estimations may cause different values between pre-silicon estimates and post-silicon measurements for a vast majority of fabricated devices.

To improve the accuracy of circuit parameters that have not been acquired at pre-silicon because of the mismatches between circuit models and their implementation, design-time planning is required to facilitate post-silicon characterization.

1.3.1 Performance Maximization

One of the key properties of a digital circuit is its performance. Based on a digital designer's perspective, the computational ability of a digital circuit is called *performance*, which is the clock period or the clock frequency of a synchronous sequential circuit [59]. In flip-flop-based circuits, the slowest paths determine the maximum operating frequency of digital IC. During design-time, estimations of delays are made and, based on them, the performance of the IC will be calculated. However, the performance of the fabricated IC may not be equal to the performance estimated at pre-silicon because of the inherent variability during the fabrication process.

One method to enhance the design flow of digital circuits is to improve the accuracy of pre-silicon estimates of slow paths that determine the frequency of an IC. This can be achieved using measurements from the first silicon samples, which are fed back in the design flow [37]. However, this method cannot provide the required information for all the fabricated silicon samples. The extra control feature that can facilitate performance tuning for a digital circuit at post-silicon can be achieved using *programmable delay tuning elements*. These programmable delay tuning elements, specific details of which will be provided in chapter 2, are inserted at the design-time as extra circuitry and they can be programmed at post-silicon to tune the timing of slow paths after fabrication. In chapter 3 we will present our new method for improving the performance of an IC at post-silicon by the use of programmable delay tuning elements.

As mentioned in subsection 1.2.2, after the design is fabricated it should be tested and, for a design to be testable, additional circuitry should be placed in the circuit during the design-time. However, the test infrastructure itself should operate correctly in the first place. In the following, we will explain a potential problem that may arise during testing.

1.3.2 Testing

One of the most adopted structured design-for-test (DFT) techniques is *scan design* [73]. In this technique, all flip-flops of a circuit are connected to each other serially forming one or multiple *scan chains*. Scan technique has been traditionally used for stuck-at [73] testing and, later on, it has been adopted for *at-speed testing* [72], which means that the circuit is verified whether it can operate correctly at its target performance. There are two basic approaches for at-speed testing [73] using scan: launch-on-capture (LOC) and launch-on-shift (LOS) (Figure 1.4) in both of which all flip-flops of the circuit are initialized through their serial connections with pre-defined values. Then, the response of all paths will be stored in flip-flops within the time interval provided by the clock period defined by the target performance. Finally, the responses will be shifted-out of the circuit serially and they will be investigated to check whether an incorrect value is generated by a path or not.

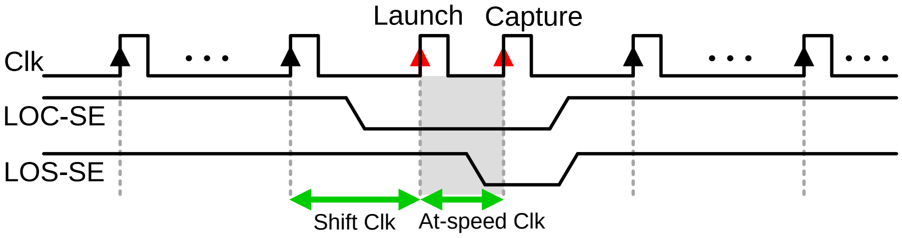


Figure 1.4: SE in LOC testing vs. SE in LOS testing.

In order to switch the circuit from the shift mode to the functional mode, a synchronizer signal is required. This signal is called the *scan enable (SE)* signal. It should be noted that shifting the initialization values in the flip-flops is carried out with the shift clock, which is known to be slower that the functional clock.

As shown in Figure 1.4, when the value of SE is high, flip-flops are in shift mode, and when it is low, flip-flops capture the responses of functional paths. In LOC testing [79] (to be elaborated in chapter 2), both de-assertion and assertion of SE are within the shift clock. Hence, SE will have sufficient time to transition from high to low and low to high. However, in LOS testing the de-assertion of the SE signal is within the functional clock pulse, which is faster than the shift clock. Therefore, the arrival time of the negative edge of SE signal that is connected to all flip-flops may be affected by un-modelled process variations, thus resulting in incorrect LOS testing. Therefore, a challenge is to guarantee that the negative edge of SE arrives at the same time to all flip-flops. In chapter 5 of this thesis, we will present a new technique that calibrates the timing of the SE signal for LOS testing at post-silicon by using programmable tuning elements.

In the following we discuss challenges facing circuit reliability, which are concerned with guaranteeing correctness of the behaviour of a circuit during its lifetime.

1.3.3 Reliability

Reliability degradation due to circuit aging is one of the common issues in the lifetime of a circuit. One of the sources of circuit aging is *negative bias temperature instability (NBTI)* that affects the timing of the circuit during its lifetime. As a consequence, some paths, which were not expected to fail based on pre-silicon estimates, may fail after fabrication due to delay degradation. This implies that the traditional pre-silicon delay estimation methods performed by static timing analyzers (STA) are not sufficient to accurately model the circuit aging at pre-silicon. This is because information about post-silicon variation that can vary from die to die, from chip to chip, and from path to path on the same chip is not available at pre-silicon.

In order to control the reliability degradation, the impact of the circuit aging on different circuit elements should be studied. Based on [3] and [16], the initial impact of NBTI is on the temporal increase of the threshold voltage of PMOS transistors in the circuit resulting into the operational frequency slow down of the circuit over time. In the work presented in [4], *sensor flip-flops* are presented that can predict delay degradation of a path.

In this thesis, in chapter 4, we will present a new technique to combat lifetime degradation that exploits the above mentioned sensor flip-flops to predict the delay degradation. These sensor flip-flops are used to predict the setup-time violation due to delay degradation. Then, by using the programmable delay tuning elements, we will tune the timing of degraded paths to avoid performance slow down of the circuit and improve the reliability.

1.4 Thesis Organization and Summary of Contributions

This thesis provides new techniques for post-silicon delay measurement and tuning, as applied to performance maximization, reliability improvement and facilitating a robust infrastructure for at-speed scan testing.

In chapter 2, the background knowledge and the known art on performance, testing, and reliability improvement will be elaborated. Then, in chapter 3 we will present our contribution for performance maximization, which can be summarized as follows.

- **Motivation:** As will be elaborated in chapter 2, multiple methods have been proposed to improve the performance based on pre-silicon information. Also, a group of works have emerged with the focus of improving the performance based on post-silicon delay information of slow paths. However, the majority of them fail to account for the post-silicon measured delay value of fast paths that may fail due to hold-time violation. Also, an exact method based on post-silicon delay measurements has been proposed that only targets small circuit sizes, making it impractical for large circuits because of its excessive storage requirements and run-time. As a consequence, lack of an exact yet scalable method that accounts for post-silicon delay values of fast paths was the motivation of the work that will be presented in our first contribution chapter.

- **Technical Challenges:** In order to tune the timing of slow paths after fabrication, the programmable delay tuning elements can be used. However, they can result in hold-time violations for fast paths, as stated above. To narrow down the group of fast paths that may be affected, the unique feature of delay tuning elements should be noted, which is the *tuning range of delay tuning elements is limited*. This feature means that those fast paths whose delay is smaller than the tuning range of delay tuning elements may fail due to hold-time violations and we only need to find the delay of those fast paths.

  After finding the delay of fast paths, the timing of slow paths should be tuned for which the programmable delay tuning elements can be used. Knowing the fact that the tuning range of delay tuning elements is limited, tuning the timing of only a limited group of slow paths is sufficient to maximize the operational performance of a circuit and there is no need to adjust the timing of all paths in a circuit. This will result in the significant reduction in run-time and the size of solution space for our proposed approach used to find the maximum performance.

- **Advantage:** By exploiting the above mentioned natural feature of the delay tuning elements, we are able to reduce the solution space and also break it into multiple small pieces that can be processed very fast. We have also been able to propose an exact performance maximization methodology that can scale to large circuits.

In our next contribution to be presented in chapter 4, we focus on improving the reliability of digital circuits during their lifetime.

- **Motivation:** A group of prior works in this area have focused on pre-silicon solutions to improve the lifetime reliability of a circuit. Post-silicon approaches have also been proposed such as voltage and frequency scaling methods to improve the lifetime reliability of circuits. However, to the best of our knowledge, none of the existing techniques presented in the public domain has employed programmable delay tuning elements to improve the circuit reliability. Therefore, in our new technique presented in chapter 4, we will leverage the presence of programmable delay tuning elements, which are commonly inserted in high-speed circuits for performance maximization, as an alternative to the known art for reliability enhancement.
- **Technical Challenges:** When the delay of a slow path is degraded in a circuit, it results in the performance slow down or even circuit failure. In our proposed technique, we combine sensor flip-flops and programmable delay tuning elements to avoid the circuit failure and maintain the maximum operational performance of the circuit by adjusting the timing of the paths whose delay has degraded. However, as the programmable delay tuning elements are used for time adjustment of the degraded paths, another subtle issue may arise, which is due to the difference between clock arrival

time to different flip-flops of the same circuit. Hence, if there is a fast path between two flip-flops with different clock arrival times, the timing requirements for it may not be met [59] resulting in the failure of the fast path. As a result, it is critical to distinguish fast paths that may fail due to the time adjustment of degraded paths. All of the above must be achieved in-system during circuit operation, which posed an additional challenge that needed to be addressed.

- **Advantage:** Based on the methodology to be elaborated in chapter 4, our approach is able to preserve the maximum operational performance of a circuit during its lifetime without the need to stall the operation of the circuit. Also a circuit structure has been proposed that can predict fast paths hold-time violations caused by delay tuning elements, thus facilitating in-system timing adjustments that can be achieved seamlessly without execution interruption.

In the last contribution chapter, which is chapter 5, we will discuss our technique to improve the LOS testing of VLSI circuits.

- **Motivation:** One of the positive features of LOS testing is that it detects higher number of faults vs. LOC testing for the same number of test patterns [79], making it one of the common techniques for at-speed testing of circuits through scan. However, based on the explanations provided in subsection 1.3.2 and Figure 1.4, the negative edge of the SE signal in LOS testing should arrive within the at-speed clock pulse. It is possible that due to un-modelled process variations introduced during the fabrication process, the timing requirement of the SE signal cannot be met, thus resulting into incorrect LOS testing. This problem has motivated us to investigate an alternative approach for improving the timing of the SE signal after fabrication by the use of programmable delay tuning elements.

- **Technical Challenges:** In order to find a circuit-specific configuration of delay elements that fixes the timing of SE for each individual circuit sample, every single CUT should have its own calibration process that happens before LOS scan testing. Hence, in order for the calibration process per chip to be practical, it must be short in time. The calibration time comprises the time for finding the timing issues of SE, if any, and tuning the programmable delay tuning elements to fix the timing problems.
- **Advantage:** By the use of programmable delay tuning elements, we are adding an extra feature that can facilitate the calibration of the timing of SE at post-silicon, and thus compensate for the timing variations caused by the fabrication process. An interesting property of the calibration process proposed in chapter 5 is that every individual chip will have its own unique configuration for its delay tuning elements, which is used to fix the specific SE timing problem that affects only the respective circuit sample.

Finally, after the three contribution chapters, the conclusion and suggestions for future works will be provided in chapter 6.

Chapter 2

Background and Related Works

A key metric that is considered in the design of digital integrated circuits is *performance*. From a digital system designer's point of view, the performance of a circuit lies in its computational speed. Based on [59], performance maximization usually refers to minimizing the duration of the clock period, which is equivalent to maximizing the clock frequency. Performance maximization of a digital circuit is a key objective during the *design stage* before the circuit is fabricated (pre-silicon). Performance is then considered after manufacturing (post-silicon) by *testing* the circuit for its maximum speed. Then, the performance needs to be accounted for *during the lifetime* of the circuit to improve its reliability, as the circuit's speed degrades due to aging.

In the following, we provide the background knowledge and related works on performance maximization, test, and reliability during the design stage, post-silicon, and the lifetime of a product. Later, in chapters 3, 4, and 5 we explain our approaches to improve performance, reliability, and test application in digital circuits.

2.1 Background: Performance Maximization

State elements are central to the implementation of the memory functionality in sequential circuits. The most commonly used state elements in sequential circuits are latches and flip-flops. In this section, we explain one of the performance improvement techniques called *time borrowing* in latch-based designs and flip-flop-based designs.

2.1.1 Performance Improvement in Latch-Based Designs

Performance improvements can be achieved through the *latch-based* design methodology in which transparent latches are placed within the combinational logic. The use of latches enables flexible timing by slack passing and time borrowing between combinational stages [59]. An illustrative example is shown in Figure 2.1 to clarify the concept of time borrowing in latch-based designs.

As displayed in Figure 2.1, *Path A* is between latches A and B, and *Path B* is between latches B and C. Based on Case 1 in this figure, latches A and C are negative level sensitive and latch B is positive level sensitive and the time allocated to each path is half a clock cycle ($\frac{1}{2} * T_{clk}$). Latch A will be open at clock edge 1, and latch B will be open at clock edge 2. If the propagation delay of *Path A* is less than $\frac{1}{2} * T_{clk}$ as shown in Case 2, the data provided by *Path A* will be ready before clock edge 2 when latch B becomes active. Then, at clock edge 2, latch B will be open and *Path B* will provide data to latch C. However, as shown in Case 3, if the propagation delay of *Path A* is longer than $\frac{1}{2} * T_{clk}$, it can utilize time that is left over from *Path B*. This is a consequence of latch B being transparent during its on-time. This phenomenon is called time borrowing and it does not need any design modification to improve the performance [59] because it is automatic. Hence, the performance (clock frequency) of this example can be more than what the critical path (*Path A*) implies.

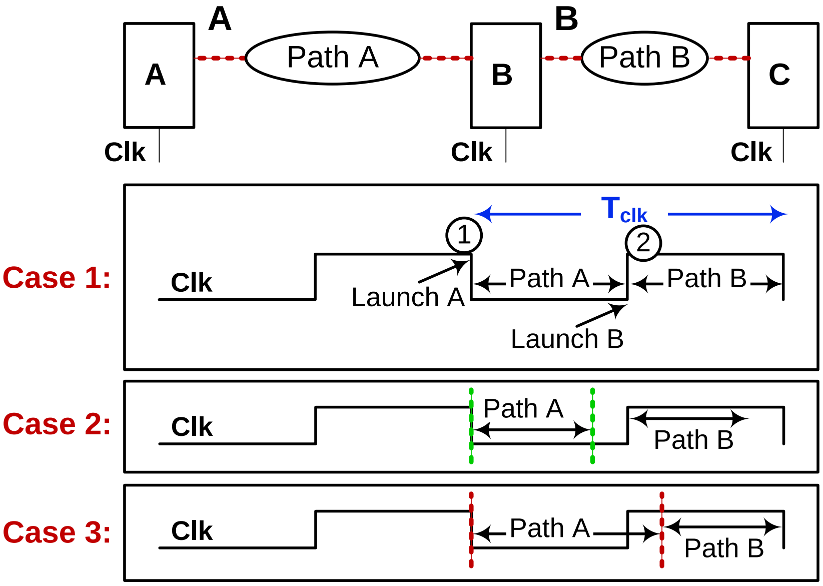


Figure 2.1: Time borrowing in a latch-based design.

The key advantage of time borrowing is that when multiple paths are cascaded and the propagation delay for a logic block between clock cycle boundaries is longer than the clock cycle, the combinational logic path can utilize the slack time from the neighbour paths without stretching the clock period and without explicit changes to the design.

In addition to enabling time-borrowing in pipelined designs, latches also require a smaller transistor count and less power than flip-flops. Nonetheless, one of the limitations introduced by latches is caused by their transparency during the active pulse of the clock signal. In order to avoid undesired transitions to be transmitted to the next logic stage, the hold-time constraint needs to account for the shortest propagation delay between any two state elements; this, in turn, adds a constraint on the shape of the clock pulse and its duty cycle. On the other hand, flip-flops change their state only at clock edges, thus simplifying both clocking and timing analysis. Consequently, flip-flop-based designs are dominant in practice, in particular for standard cell implementations, and hence they are discussed next.

2.1.2 Performance Improvement in Flip-Flop-Based Designs

In a flip-flop-based design, clocking elements are edge-triggered and the performance of the design is determined by the slowest combinational logic path between any pair of flip-flops. This is unlike latch-based designs where combinational paths that are longer than the clock period, and they can borrow the leftover time from their neighbour paths without any particular design modification. To improve the frequency of flip-flop-based designs, time borrowing-like techniques have been investigated. In the following, architectural modifications to implement the time borrowing in flip-flop-based designs are illustrated.

In Figure 2.2 assume there is a combinational logic block between $FF1$ and $FF2$ with delay $D1$ and another one between $FF2$ and $FF3$ with delay $D2$ smaller than $D1$. There is also a *cloud* on the clock signal of each flip-flop, which represents a clock tuning element for each flip-flop. In 'zero-skew' configuration (all the clock tuning elements will be assigned '0', and the clock edge arrival time to all flip-flops is the same), the minimum clock period would be equal to $T_{zero-skew} = T1 = D1$. Nonetheless, the performance of this design can be improved if *datapath 1* can borrow time from *datapath 2*. The performance can be improved if the clock arrival time of $FF1$ and $FF3$ are fixed and the clock arrival time of $FF2$ is delayed by δt (Figure 2.2), the clock period will shrink to $T2 = T1 - \delta t$ while the available time for *datapath 1* to prepare data for next stage ($FF2$) is still $D1 = T1 = T2 + \delta t$. To calculate the value of δt, the difference between $D1$ and $D2$ should be divided between the two paths, therefore, $\delta t = (D1 - D2)/2$ and the new clock period will be equal to $T2 = D1 - (D1 - D2)/2 = D1 - \delta t$. Note, however, that no path from $FF1$ to $FF2$ should be excited at any time with a delay less than δt.

An important property of the tuning elements is that there is a maximum of predefined delay that they can add to or subtract from clock arrival time of flip-flops connected to them.

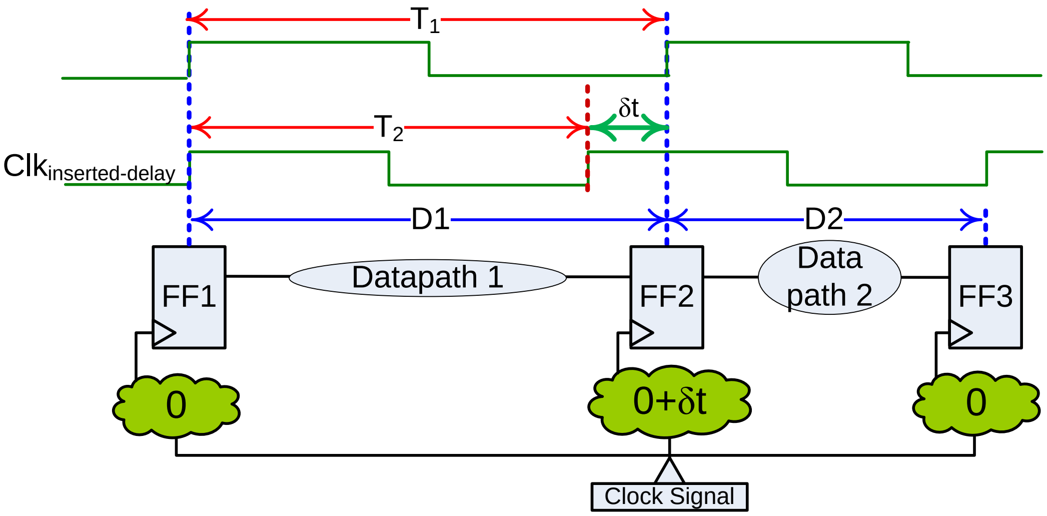


Figure 2.2: Clock tuning elements used for time borrowing and performance improvement.

In Figure 2.2, assume this maximum delay is $m < \delta t$. Therefore, the clock signal arrival time of $FF2$ can only increase as much as m and the new timing window for *datapath 1* will change to $(T2+m) < D1 = T1 = T2+\delta t$. It means that under new constraints, *datapath 1* will fail if the clock period remains $T2$. To make the above example work under the new constraints, equation (2.1) must be satisfied, which shows that the maximum tuning range of clock tuning elements limits the performance gain.

$$T_{new} = T2 + \delta t - m \tag{2.1}$$

An example of programmable clock tuning elements are Clock Vernier Devices (CVDs) [47]. Each CVD has a clock input and a clock output pin which is the delayed clock signal. The schematic of a CVD is shown in Figure 2.3. Each CVD has several configuration bits. For the CVD circuit described in [47] and illustrated in Figure 2.3, each CVD has three configuration bits and can have eight different configurations in total. The configurations are loaded through serial scan chains. Each of these configuration values provides a delay value to be added to the 'In Clock' and results in a new/delayed arrival time for 'Out clock'. The minimum delay value to be added to 'In Clock' is '0' generated by configuration '000' for the three bits. Likewise, each CVD provides a maximum delay to be added to its input clock signal through configuration '111' and the input clock signal cannot be delayed more than what '111' configuration provides. The CVDs' configuration values are determined by CVD configuration algorithms after all the speedpaths of the design are detected.

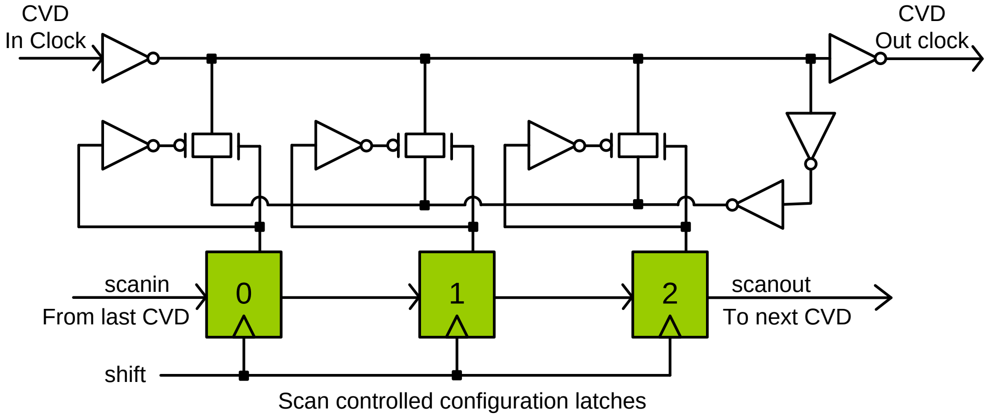


Figure 2.3: Clock Vernier Device (CVD).

In this section, we have discussed how the performance can be improved in latch-based designs without explicit design modifications. Also, we explained how CVDs can be exploited to improve the performance in flip-flop-based designs. In the following, we will discuss time borrowing techniques to improve the performance both at the design stage (pre-silicon) and after fabrication (post-silicon). Later in chapter 3 we will explain our technique to maximize the performance based on the real path delays at post-silicon by exploiting CVDs.

2.2 Related Works: Performance Maximization

Clock tree structures have been investigated since clock networks have been used in digital integrated circuits. A fundamental clock tree structure is the H-Tree structure [60], which guarantees the same clock signal arrival time for all the clocking elements in a design [59] (zero-skew structure). However, this clock tree structure does not result in the maximum achievable frequency (as elaborated in subsection 2.1.2, Figure 2.2). As a consequence, approaches based on clock tuning elements (examples of which are CVDs) have been introduced to take advantage of the difference in clock signal arrival times (useful skew) and therefore improve the operational clock frequency. The use of clock tuning elements implies two tasks; the first one is *insertion*, which is a design-time task, and the second one is *configuration* performed after the first silicon samples are available, if the clock tuning elements are programmable.

The works presented in [71] and [80] try to compensate the clock skew at pre-silicon by inserting clock buffers while routing the clock tree. They determine the clock buffers' locations and sizes using wire load models. In these two works, both insertion and adjustment of the tuning elements (which are the clock buffers) happen at pre-silicon. However, to provide another degree of freedom in adjusting the clock edge arrival time, programmable clock tuning elements can be inserted in pre-silicon and they can be programmed at post-silicon. The insertion of these *programmable* clock tuning elements has been investigated in [25], [26], [49], and [69]. Our research is concerned primarily with the configuration of programmable clock tuning elements. Although it is compatible with any existing insertion algorithms, we have explained our own CVD insertion algorithm in later sections which is closely interrelated with our proposed CVD configuration method. The unique challenges for configuration have motivated research on the topic elaborated below.

As mentioned earlier, CVDs are examples of *programmable* clock tuning elements. CVD configuration for clock skew scheduling has been discussed as early as in [29], relying on pre-silicon delay estimates. Other techniques such as [53] have followed based on the same reliance on pre-silicon estimates, which, though can be employed to some extent to performance tuning, they bring little value to speed binning [21], [23] or speedpath diagnosis for which post-silicon measurements are needed. In addition to that, pre-silicon estimations also are not sufficiently accurate for the state-of-the-art designs, for which the speedpath identification is not obvious [40] due to the process variations. For these reasons, post-silicon data is required for CVD configuration (e. g., [26], [46], [48], [65], [68]).

The work from [48] assumes one CVD per flip-flop resides in the circuit and it operates as follows. It identifies the failing outputs from the data gathered from the tester and, using the path tracing method [2], it tries to find the inputs that have triggered this failure. Because it relies on path tracing and it assumes that every single flip-flop is tuneable, the method does not scale to large designs. Furthermore, this technique does not exploit the quantization levels provided by the CVDs (as introduced in subsection 2.1.2). The work from [48] relies on a list of flip-flops for which their clock edge arrival time has to be earlier and another list of flip-flops for which their clock edge arrival time has to be later. If a flip-flop is in both the early list and the late list, its clock edge arrival time will not change, which misses the opportunity provided by the quantization levels in CVDs, which can facilitate variable delays to be added to the clock arrival times. Another work from [46] introduces a genetic algorithm for CVD configuration where CVD values are grouped into a solution that is iteratively transformed based on genetic operators. The results are reported on only two very small examples and for one of them it is assumed, as in [48], that there is one CVD per flip-flop.

As the circuit size grows, one cannot assign one clock tuning element per flip-flop because of the constraints on area, increased power consumption and increased complexity for configuring all clock tuning elements per flip-flop. Therefore, to avoid the limitations caused by one CVD per flip-flop, each clock tuning element should be connected to a *group* of flip-flops and assigning a new arrival time to it implies that all the flip-flops in that group will have the same clock arrival time. In the following, we discuss the approaches which have multiple flip-flops assigned to each CVD in order to understand how they are tackling the CVD configuration problem.

The work presented in [26] uses CVDs for post-silicon clock adjustment. It presents the clock distribution strategy for system-on-a-chip (SOC) devices which are integrating intellectual property (IP) blocks. It assumes that the clock root of each IP has its own clock tuning element; if the designer plans to substitute any IP with another IP with equivalent functionality, there is no need to redesign the entire clock tree of the SOC. Using clock tuning elements, new clock schedules can be achieved. By providing the coarse-grain control of CVDs (one CVD per IP block), the objective in [26] is to meet the timing constraints for the SOC when replacing IP blocks late in the implementation flow, and not to compensate for the lack of accuracy of the pre-silicon timing estimates caused by process variations in advanced fabrication technologies.

The work reported in [68] uses false path aware STA to provide an estimation of paths' delays in pre-silicon and creates a collection of flip-flops that either will fail, or may fail if variation is taken into consideration. Then, it assigns CVDs to the flip-flops in this collection. After the design is fabricated, CVD configuration is determined using linear programming and branch and bound algorithms. In this work, since only a few percentage of flip-flops are tuneable and the selection of flip-flops and CVDs which are tuneable is fixed and it is based on pre-silicon delay estimates, the maximum operational frequency may not be found. This is because if excessive process variations or systemic delay defects cause paths that were not accounted for during design to fail in silicon (whose source or sink flip-flops do not have CVDs), this method cannot be used.

All the above mentioned approaches find the CVD configuration based on the measurements on a single chip. In practice, the measurements can be done either on each individual chip or, if the tester time is prohibitively high, the same configuration can be used for the entire volume of fabricated devices. In [65], the CVD configuration is determined using an approach based on Satisfiability Modulo Theory (SMT) [19], [54] and using measurements on a *batch* of chips, as representatives of all the fabricated devices. The motivation for this approach lies in the focus of the work, which is on speed binning, and the main objective is to determine a CVD configuration that maximizes the number of devices from the batch that will be placed in the highest frequency bin. Because the focus of [65] is on speed binning and not on performance tuning, only a limited number of predicted speedpaths are taken into consideration; and CVD configurations for the source or sink CVDs of these speedpaths are adjusted. Also, if there are some paths in the neighborhood of these predicted speedpaths, whose delays are close to speedpaths' delays, adjusting only the CVDs for the speedpaths may negatively impact the timing of these neighbor paths; this can eventually convert these neighbor paths to speedpaths and consequently, worsen the operational frequency of design.

Since the focus of the work from [65] is on speed binning, not trying to borrow time from the CVDs which are not directly connected to speedpaths, but are reachable, may not have a significant impact on the binning process; however for performance tuning, accounting for the signal dependencies between the flip-flops in the design can become important, as discussed in this thesis. As a final remark, although [65] does consider the potential failures caused by hold-time violations, its solution, based on an off-the-shelf constraint solver, relies on the pre-silicon estimates of the fast paths, an approach similar to the one employed in [53].

Based on the above discussions, time borrowing can help improve the performance of a digital circuit either at the design stage or after the first silicon samples are available. Nonetheless, there is a need to *preserve* the maximum performance during the lifetime of a chip. In the following, we explain the aging process and how the circuit speed is affected. We will also explain briefly how the aged circuitry can be diagnosed on the chip. Then, the prior works related to preserving the performance will be discussed. Later in chapter 4 we will explain our technique proposed to preserve the performance during the lifetime of a circuit.

2.3 Background: Reliability

Negative bias temperature instability (NBTI) is one of the dominant factors in circuit aging as discussed in [3], [16], [41], [56], [62], [74], [75]. The initial impact of NBTI is on the temporal increase of the threshold voltage of PMOS transistors in a design, which will reduce the drive current and slow down the switching of the circuit [12]. As a consequence, the operational frequency of the circuit will reduce over time.

To mitigate the performance degradation over the lifetime of a circuit, NBTI effects have been considered in circuit analysis and design. For example, to quantify the performance degradation, [4] has shown that factors such as the supply voltage, age of the circuit, temperature and in-field activity (on-time) of the PMOS transistor affect the threshold voltage of a PMOS transistor. In addition to threshold voltage degradation, the chip-level impact of NBTI can be different on different paths of the same chip or same paths of different chips [4], [6]. Therefore, varying from one circuit sample to another, some paths that have not been critical may become critical if they degrade more than the paths that have initially been critical.

In order to tackle the reliability issues [35], [81], [31], [42] caused by circuit aging, those paths that have been affected by aging should be diagnosed at the first place. To diagnose the paths that have been degraded, [4] has presented sensor flip-flops, which predict failures due to aging during the normal operation of a circuit before the failures take place. Based on the sensor flip-flop displayed conceptually in Figure 2.4, assume that the delay of the combinational path leading to signal *Out* has been degraded due to aging such that a transition is observed by the sensor flip-flop during the guard-band interval introduced by the sensor flip-flop. In this case, the sensor flip-flop predicts a failure because if the delay of the path leading to *Out* increases more than the guard-band interval due to aging, a setup-time violation will occur. As a consequence, anytime that a sensor flip-flop receives a transition in its guard-band interval, it declares that it has predicted a setup-time violation.

In the following, we will discuss in detail sensor flip-flops and their operation. Also, we will explain how the known art deals with the reliability issue. Our unique approach for improving the reliability in digital circuits will be explained in detail in chapter 4.

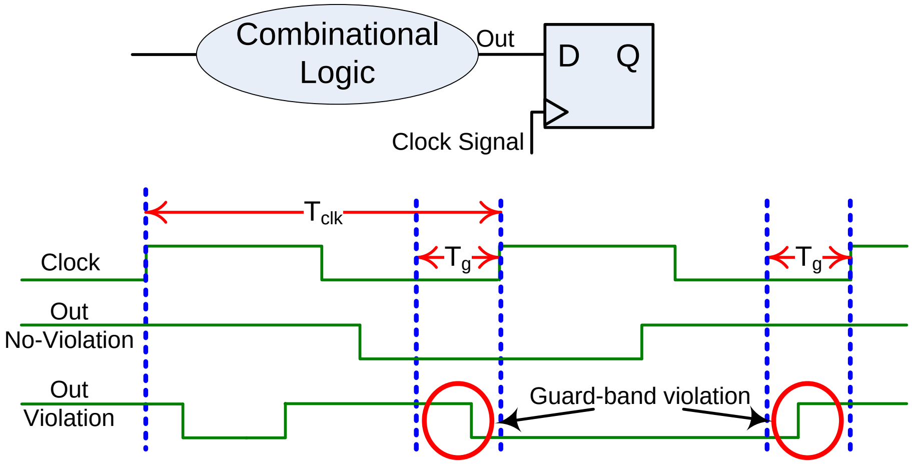


Figure 2.4: Sensor flip-flop to predict setup-time violation.

2.4 Related Works: Reliability

In this section, we first discuss the works ([4] and [76]) that have studied the aging issue and its sources. Afterwards, methods that estimate the degradation of an aged circuit will be introduced [74], [76] and then different approaches for dealing with degradation at pre-silicon or post-silicon will be discussed [3], [4], [16], [18], [27], [41], [55], [56], [63], [75], and [76].

As explained in section 2.3 NBTI is one of the dominant factors in circuit aging as discussed in [4], [41], and [76]. NBTI reduces the circuit's operational frequency by time. To mitigate the performance degradation over the lifetime of a circuit, many works have studied impacts of NBTI in circuit analysis and design. Approaches such as [41], [74] and [76] have estimated the magnitude of degradation due to aging. For example, [41] presents an aging analysis flow able to calculate the timing of the degraded circuit at the gate-level and it shows that the degradation in the worst case conditions is more than the degradation in case of probabilistic workload.

The work presented in [74] studies the impact of aging on gradual changes of circuit parameters such as threshold voltage, operational frequency, path delay, and drive current to statically model the overall degradation of the digital circuit under variation in the life time of the chip. The work from [76] has also developed short term and long term threshold voltage degradation models for different types of input vectors (such as periodic or random input vectors). And finally, the work presented in [16] has shown that aging is dependent on duty cycle of the input signals, supply voltage, and temperature. The works presented in [16] and [76] have considered the circuit parameters that influence the speed of circuit aging. Specific techniques that help control the speed of circuit aging, such as dynamic voltage scaling (DVS), lifetime awareness, dynamic instruction scheduling and power gating, are detailed in [16]. The work from [76] has also studied design techniques during the early design stage such as, optimizing the node activity, tuning the supply voltage, reducing temperature and resizing in order to minimize the NBTI effect.

The works that have been covered so far have dealt with the sources of aging and they have studied the impact of aging on different circuit parameters, as well as the design and circuit techniques that can affect the speed of aging. In the following, the works that have focused on tackling the degradation problem are discussed. A majority of the early works have relied on pre-silicon worst-case estimation of degradation as discussed next.

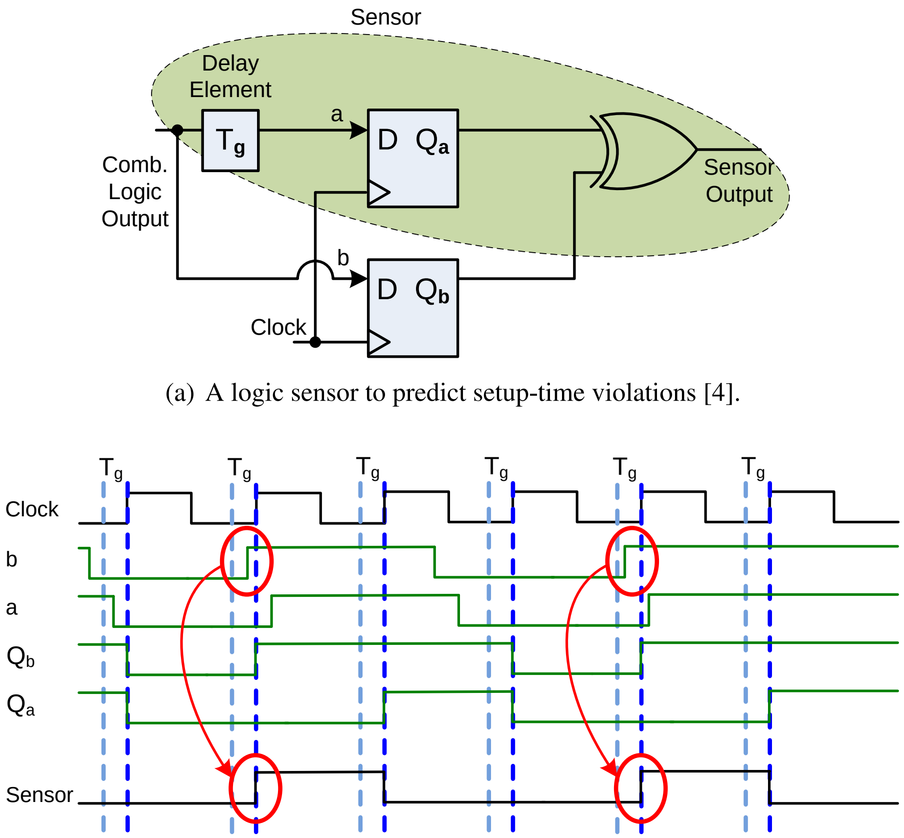
The work presented in [56] provides an early estimation of performance degradation [36], which is used as a design parameter to improve the reliability of the circuit during its lifetime and to guarantee a targeted frequency with an area overhead of 5.8%-14.8%. Their algorithm finds the threshold voltage degradation for all the PMOS transistors based on their on-times and performs circuit sizing to obtain a desired operational frequency. However, applying this method to large designs may be impractical and very time-consuming because the threshold voltage degradation of every single PMOS transistor should be calculated and the entire circuit should be resized for reliability assurance. The work presented in [75] relies on the fact that not all the transistors in a circuit need the estimation of threshold voltage degradation and resizing. In fact, the delay degradation estimation [57] is only required for the paths that are either critical, or may be critical if the delay degradation is taken into account. Then, the gates on these paths are sorted based on their maximum amount of delay degradation and their impact on the delay degradation of these paths. The gates with highest impact on delay degradation of the above-mentioned paths are selected as critical gates and they are replaced with fresh gates [75].

Both works presented in [56] and [75] have investigated design techniques at pre-silicon based on worst-case assumptions to reduce delay degradation due to circuit aging during the lifetime of a chip. The worst-case path delay degradation is calculated based on extreme conditions such as maximum on-time (activity) for PMOS transistors, high temperature and high supply voltage. However, this worst-case situation may not happen during the in-field operation and the paths may not get degraded at the calculated time. Also, the real-time degradation varies from one chip to another and from one path to another path on the same chip [6]. Therefore, there is a need for the circuits to be monitored for their degradation during their normal operation. One of the methods to monitor the failures caused by aging is to predict the time when a failure may occur by the means of on-chip hardware, as explained next.

The work from [4] *predicts* the failures due to aging during normal operation of the circuit before they occur. There are two benefits of failure prediction: (i) failures are predicted before they actually occur in-system and corrupt the circuit's state; (ii) the circuit does not have to operate based on a worst-case *estimated* performance. In fact, the circuit keeps on operating in its maximum possible performance until a failure is predicted at which point in time the circuit has to adapt itself (e.g., relaxes the operating frequency) before the failure affects the circuit state. As detailed below, [4] uses logic sensors to detect the setup-time violations that may cause a failure in the near future. When the failures are predicted, [4] discusses several self-correcting options, such as adapting the supply voltage (which increases power consumption and speeds-up the aging process) or relaxing the clock frequency (which degrades performance). As shown in Figure 2.4, if any transition is detected during a guard band interval, the sensor will turn on. The basic idea is to modify a standard flip-flop by adding the following three components: (i) a delay element which defines the guard-band interval; (ii) a logic circuitry to check the transition during the guard-band interval; (iii) an additional latch or flip-flop which works with the transition checker and holds its output. A simple method for designing a sensor flip-flop is captured in Figure 2.5(a). Figure 2.5(b) explains the operation of the logic sensor using waveforms.

As displayed in Figure 2.5(b), anytime a transition occurs in the guard-band interval of the sensor flip-flop, it will be detected and a rising edge will appear in its output. It should be noted that the logic sensors introduced in [4] are not the only types of sensors that can predict timing failures. Other types of specialized flip-flops have also been presented in the literature (e.g., [18], [27], [55] or [63]). While we use the sensors presented in [4] to discuss our flow, extensions of other types of sensors can similarly be used in our work, however the details are beyond the scope of the contribution presented in this thesis.

Sensor flip-flops can assist with failure prediction or detection, however they are not sufficient to guarantee the correct behaviour of a digital circuit during its lifetime. Therefore, based on the work presented in [4], self-correction or self-adaption techniques are



(a) A logic sensor to predict setup-time violations [4].

(b) An illustrative example of sensor operation.

Figure 2.5: Sensor flip-flop schematic [4] and waveforms.

required for NBTI-induced transistor aging. The methods for self-correction include frequency relaxation or adaptive power supply. Each of these methods have their own side effects. For example, frequency relaxation will directly impact performance, whereas increasing the power supply unnecessarily speeds up parts of the circuit that have not aged with a direct impact on power dissipation; it also indirectly accelerates aging [4]. As a conclusion, the actions taken to control aging have side effects that negatively affect the operation of the circuit. These side effects are the initial motivation for our work to be presented in chapter 4.

So far in this chapter, we have covered two common issues in the lifecycle of digital circuits, which are performance maximization and reliability. In the following, we describe the background knowledge and related works for testing the VLSI designs.

2.5 Background: Test

After a digital design is fabricated, it is tested to screen for defects caused by the fabrication process. Testing of VLSI circuits consists of applying test stimuli to the circuit under test (CUT) and collecting and analyzing the output responses [73]. Circuits that generate correct responses to all the test stimuli will pass the test and circuits that generate at least one faulty output for one of the input stimuli are considered faulty.

2.5.1 Functional and Structural Testing

To find functional errors caused by fabrication process, circuits should be placed under functional tests. To test a circuit with n inputs, multiple test stimuli should be applied to the CUT and the response to each of them should be compared to the known correct response of a fault-free circuit. To apply all possible input combinations to the CUT all possible 2^n different input combinations can be generated, each of which is called a *test vector*. Nonetheless, if the input size n is large, as it is the case for most state-of-the-art circuits, applying all possible input combinations will be impractical [73]. Hence, not all test vectors should be applied to the CUT; instead, only a small group of selected test vectors should be applied. Knowing the fact that each test vector targets one or multiple faults, the group of test vectors applied to a circuit are selected based on the faults that they target to sensitize. This method of testing is called structural testing, as explained next.

Structural tests depend on the structure of the netlist of a design. Based on the logic and timing behaviour of electrical defects, different fault models can be introduced allowing test generation, test application and test evaluation to be performed by automated methods. The most commonly used fault model is the single stuck-at fault model. It transforms the correct value on a faulty signal to be stuck-at a constant value (stuck-at-0 or stuck-at-1).

2.5.2 Test Pattern Generation

After selecting a fault model for structural test, the next step is to generate a group of test vectors (test patterns) to target all the faults, as defined by the fault model at hand. Test generation consists of creating test stimuli (input test vector) that sensitize one or multiple targeted faults (fault sensitization) and propagate the resulting fault effect to primary outputs (fault propagation). A group of expected correct responses should also be generated to be compared with the actual circuit responses to determine whether the CUT has failed or passed.

Automatic Test Pattern Generation (ATPG)

ATPG is the process of test pattern generation that is performed automatically based on different methodologies proposed for test pattern generation (such as *D-Algorithm* and *PODEM* [1], [15], [73]). The effectiveness of ATPG is measured by the number of faults that are detected (determines test quality that is considered to be higher when more faults are detected); and, the number of generated patterns that can impact the test application time.

ATPG allows testing of the combinational paths between flip-flops without considering the native mode behaviour of the circuit. To enable this type of testing, each source flip-flop needs to be initialized and the output of the combinational logic needs to be observed. To achieve this, flip-flops should be initialized and, after the data produced by the combinational logic is loaded into destination flip-flops, the responses must be observed in order to check which paths have produced incorrect results. One way to achieve this goal is to initialize a target fault over a sequence of clock cycles; likewise the fault effect needs to be propagated over a sequence of clock cycles to an observable output. This procedure is commonly referred to as sequential ATPG and it is known to be significantly more time consuming, as well as leading to lower test quality, when compared to combinational ATPG [73]. Combinational ATPG assumes that every flip-flop can be initialized and observed, which can be achieved through *scan* design.

Scan Testing

Scan design is a design for testability (DFT) technique that addresses the test cost [73] by reducing the complexity of ATPG and providing controllability and observability to the internal state elements. In the scan design technique, all flip-flops are connected to each other through serial connections forming a shift register structure called scan chain. In the scan test design there are two modes, a functional mode and a test mode in which the flip-flops (also called as scan cells) are accessed via their serial connection in the scan chain. Each circuit can have one or multiple scan chains which are activated in the test mode.

It should be noted that shifting-in the test vector and shifting-out the circuit response occurs in the test mode when flip-flops in scan chains are connected serially and shifting is performed by the shift clock. In the functional mode, the values stored in source flip-flops through the shift-in process are propagated via the functional logic to destination flip-flops with the functional clock. If the frequency of the functional clock is selected as high as the maximum frequency of the circuit, this method of testing will be called *at-speed* testing.

At-Speed Testing

As mentioned in chapter 1, there are two basic approaches for scan-based at-speed testing [73] that are launch-on-capture () and (LOS) (Figure 1.4) [58], [77]. In LOC testing, the SE signal does not need to transition at-speed because it is asserted and de-asserted by the shift clock. However, in LOS testing, the falling transition of the SE signal needs to happen within the at-speed clock pulse. Because SE is a global signal with high fan-out, the arrival time of its negative transition may not be the same for all the flip-flops. Therefore, a challenge is to guarantee that the negative edge of SE arrives at the same time to all the flip-flops.

Logic Built-In Self-Test (BIST)

Logic built-in self-test is a DFT technique in which a part of a chip is devoted to test the digital logic itself [73]. In a typical BIST system, the test pattern generator generates test patterns to be applied to the inputs of the circuit under test. Then, after the test vectors are applied, the output response analyzers generate a signature for the output responses of the circuit under test. Once the test application process is complete, a logic BIST controller indicates whether the circuit under test has passed or failed.

Next we will discuss how other approaches have dealt with the issue related to the timing of the SE signal in LOS testing. It should be noted that although there is a significant amount of literature that covers a broad range of topics concerned with testing VLSI circuits, in the following we discuss only prior works that are strictly related to the timing issue on the SE signal for LOS testing. Then in chapter 5 we explain our proposed method for adjusting the timing of SE in LOS testing.

2.6 Related Works: Scan Enable Timing for At-Speed Test

A commonly used technique to avoid timing slack of the SE signal received by the scan flip-flops is to pipeline the scan enable signal [50]. By pipelining it, the SE signal in LOS test mode does not need to transition at-speed. If multiple levels of pipelining are inserted on the path of the SE signal, then the SE signal should be de-asserted for a number of shift clock cycles before the launch edge equal to the levels of pipelining. Another technique that has tackled the slow switching of the SE signal, is the local generation of the SE signal [5]. This technique introduces last transition generator (LTG) cells which are inserted in scan chains. The advantage of this work is that it integrates the pipelining of the SE signal with local generation of the SE signal.

The above mentioned works have been employed to improve the timing of the SE signal in practice. Nonetheless at the final stage of the pipelined SE tree or the SE routed from LTG cells, there still might be slacks due to the process variations that appear only in some chip samples. Unless the timing of the SE is fixed, these chip samples will be regarded as faulty. To address this problem, we investigate an alternative approach to the SE timing problem by relying on delay tuning elements placed within the SE tree (similar to the ones placed in the clock tree for high-performance microprocessors [47]) that help compensate for the process variations. Our proposed method can be used as an alternative to the prior works [5] and [50], or it can be combined with them to adjust the timing of the SE signal before the LOS tests are applied. More details on our proposed method can be found in chapter 5.

2.7 Summary

In this chapter, we have explained how process variations increase the gap between the pre-silicon delay estimations and post-silicon real delay values and how this gap impacts the operation of digital ICs after fabrication. For instance, the performance obtained after fabrication may not be the same as the performance estimated at the pre-silicon stage because the pre-silicon delay estimation tools are not sufficiently accurate to take into account all the impacts of process variations. Also, the known art to deal with this problem has been explained and the use of clock tuning elements at the post-silicon stage has been covered in the related works section.

Reliability concerns caused by delay degradation due to circuit aging was the second topic of discussion in this chapter. The known art to improve the reliability of a circuit during its lifetime has been covered and sensor flip-flops have been discussed to predict delay degradation due to circuit aging.

The issues raised by the timing of the SE signal during LOS testing have also been introduced. To address the timing inaccuracies at post-silicon and tune SE for LOS testing, the related works on local generation of the SE signal and pipelining the SE signal have been explained.

In summary, this chapter has provided the background knowledge and it has discussed the relevant prior works that are related to the contributions presented in the next three chapters of this thesis.

Chapter 3

Post-Silicon Delay Measurement and Clock Tuning

In this chapter, a method for post fabrication delay measurement and clock tuning will be presented. All the prior works mentioned in chapter 2 regarding the configuration have tackled the configuration problem for different purposes, e. g. performance tuning, speed binning, and getting the circuit running at a target frequency. Furthermore, the known arts for the CVD configuration have been proposed based on underlying assumptions such as: clock tuning elements adjustment in pre-silicon or in post-silicon; having one flip-flop per CVD or multiple flip-flops per CVD, and finding CVD configuration based on measurements on a single chip or a batch of chips. Although these works have provided the foundations for tackling CVD configuration algorithmically, in this chapter we exploit a previously overlooked property of CVDs that can enable the implementation of an exact yet scalable CVD configuration algorithm. This algorithm is compatible with the following requirements that need to be accounted for when leveraging the presence of a large number of CVDs for performance tuning:

- The clock tuning flow supported by the post-silicon CVD configuration should converge to the maximum operating frequency based on the accuracy of the tester measurements, regardless of the process variations and systemic defects that affect an unknown number of speedpaths;
- The CVD configuration algorithm should be scalable to large designs without compromising the performance gains, and it should be equally applicable to data collected from either a single circuit-under-test or a batch of chips;
- No detailed knowledge from pre-silicon about the speedpath distribution should be required, although if it is available, it can speed up the clock tuning flow;
- The CVD configuration algorithm should be independent of the pattern set and the CVD insertion.

It should be noted that satisfying all the above mentioned requirements in a CVD configuration algorithm will lead to an exact and scalable method that is applicable to the today's large designs. In the work proposed in this chapter, we meet all the above mentioned requirements; however, other approaches have failed to satisfy all the requirements that will finally result in a non-optimum solution. For example, the work presented in [65] fails to meet the first condition by having worst case assumptions for paths that their delay information is not available. Also, the work presented in [48] is not scalable to the large designs by not meeting the second condition because the number of CVDs used in this work is as many as the number of flip-flops. Also, as mentioned in chapter 2, this work uses path tracing method, which makes it unfit for large designs. More details about these works can be found in chapter 2 as well as this chapter.

In the following section, we will discuss a method for post-silicon fast paths delay measurement and then we will discuss our proposed CVD configuration performed based on the measured delay values with the objective of performance enhancement.

3.1 The Proposed Flow and Algorithm

Based on the example shown in subsection 2.1.2, to improve the performance of a design by clock tuning, information from slow paths that slow down the frequency is required. Also, fast paths should be known because while adjusting the CVDs, the slack provided by them should not exceed the delay of fast paths to avoid hold-time violations. To find the CVD configuration which determines the maximum frequency of a circuit under test (CUT), an iterative flow as well as an exact algorithm is presented in this section. In the following we give an overview of our proposed *flow* shown in Figure 3.1 and then the CVD configuration *algorithm* will be explained.

3.1.1 General Overview of the Proposed Flow

Before CVDs can be used for post-silicon clock tuning, they need to be inserted in the CUT at the design phase. For the work and results presented in this chapter, we rely on our CVD insertion algorithm which will be elaborated in subsection 3.1.3. Nonetheless, our CVD configuration algorithm is compatible with any approach used for CVD insertion. It should be noted that since our proposed CVD insertion algorithm is driven by our proposed CVD configuration algorithm, we will discuss it after explaining the CVD configuration algorithm. In the following, an overview of the proposed flow will be provided and then, in the following subsections, each step of the flow will be illustrated in detail.

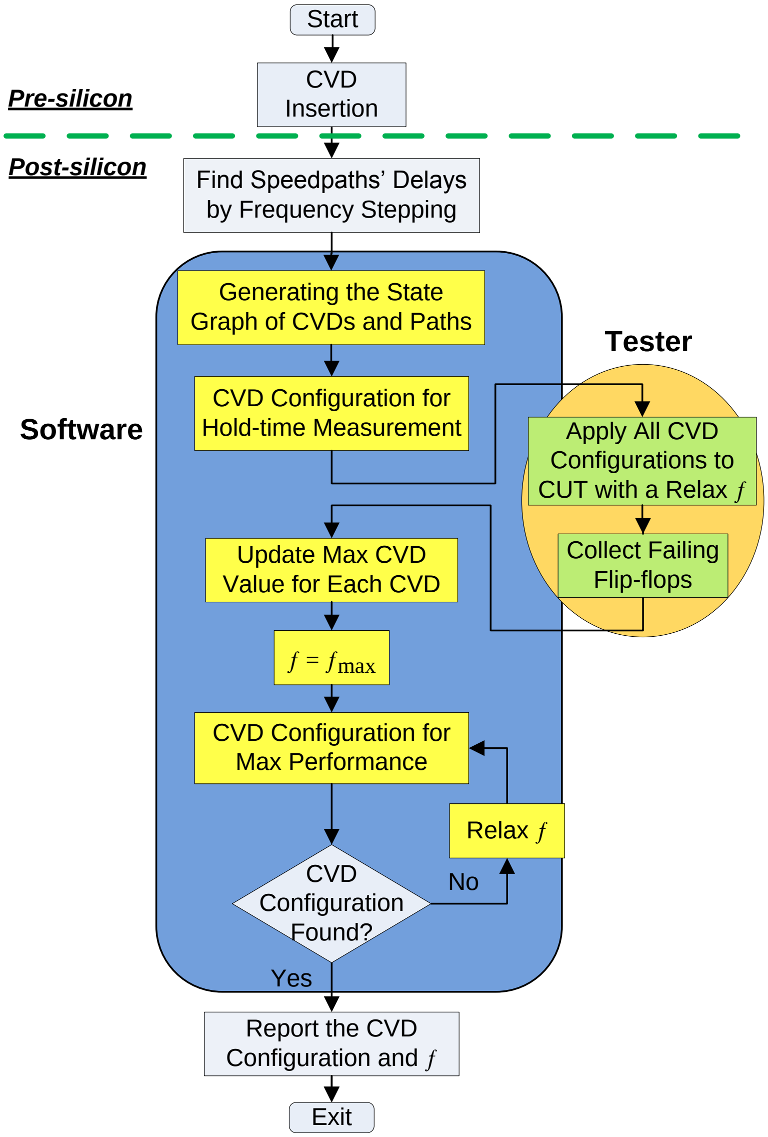

Start
CVD Insertion
Pre-silicon
Post-silicon
Find Speedpaths' Delays by Frequency Stepping
Generating the State Graph of CVDs and Paths
Tester
Software
CVD Configuration for Hold-time Measurement
Apply All CVD Configurations to CUT with a Relax f
Collect Failing Flip-flops
Update Max CVD Value for Each CVD
$f = f_{max}$
CVD Configuration for Max Performance
Relax f
No
CVD Configuration Found?
Yes
Report the CVD Configuration and f
Exit

Figure 3.1: The flow for post-silicon delay measurement and CVD configuration.

The flow shown in Figure 3.1 starts with CVD insertion at which groups or clusters of flip-flops are generated to share the same CVD. Afterwards, the CUT with the inserted CVDs will be the input to the delay measurement and CVD configuration algorithms that works in the post-silicon stage. The flow is continued with a pre-processing step "Find speedpaths Delays by Frequency Stepping" in which the slow paths (speedpaths) of the design are identified through the frequency stepping method performed on the tester, as it is the case also for the work presented in [65].

In the next step, as elaborated in subsection 2.1.2, for the flip-flops whose clock arrival times need to be delayed to compensate for the required setup-time, the fast paths leading to them should be found. The reason for finding these fast paths is that they will constrain the maximum configuration value that a CVD can get due to potential hold-time violations. To achieve this, in the box "Generating the State Graph of CVDs and Paths", the state graph of the CVDs of the design and a subset of fast and slow paths will be generated; afterwards, a set of CVD configurations will be generated through the box "CVD configuration for Hold-time Measurement" (as detailed later in this section). These CVD configurations will be applied to the CUT with a slow frequency to make sure that if any failure is observed, it is a hold-time violation failure. After all the above CVD configurations are applied to the CUT and all the failing information is collected, the real delay values of these fast paths that may fail due to hold-time violations will be calculated. Then, based on these real delay values of fast paths, the maximum applicable configuration value for each CVD will be found in "Update Max CVD Value for Each CVD".

As mentioned earlier in this section, we assume that all the slow paths' delay values are measured in a pre-processing step. We have also indicated that the real delay values of the paths that may fail due to hold-time violations can be measured. Using these data, the maximum achievable performance of the design will be found by the steps taken in the loop shown in Figure 3.1. In this loop, the software finds CVD configurations with an approach different from the one taken for fast paths measurement. In this loop, the purpose of the software is to find a CVD configuration which helps in achieving the maximum desired frequency '$f = f_{max}$' as stated in box "CVD configuration for Max Performance". If such a configuration can be found, the maximum frequency and the corresponding CVD configurations will be reported. Otherwise, the frequency is relaxed in "Relax f", and the software will try to find a new CVD configuration for the new target 'f'. The iterative flow will continue in this loop until it finds CVD configurations and the maximum *achievable* frequency. It is important to note that the CVD configuration values are dependent on the slow paths' delay values measured by the tester. Therefore, the precision of tester measurement affects the CVD configuration values. Furthermore, the granularity of CVD steps, which is the minimum delay difference between two CVD configuration steps also has an impact on the achievable frequency. The finer the CVD steps, the more improvement on the frequency can be achieved.

In the following, the algorithmic steps to find the maximum operational frequency are explained by the use of two CVD configuration algorithms. The first CVD configuration algorithm is to calculate the post-silicon fast paths delay; and, the second CVD configuration is to find the maximum achievable frequency of the design after fabrication.

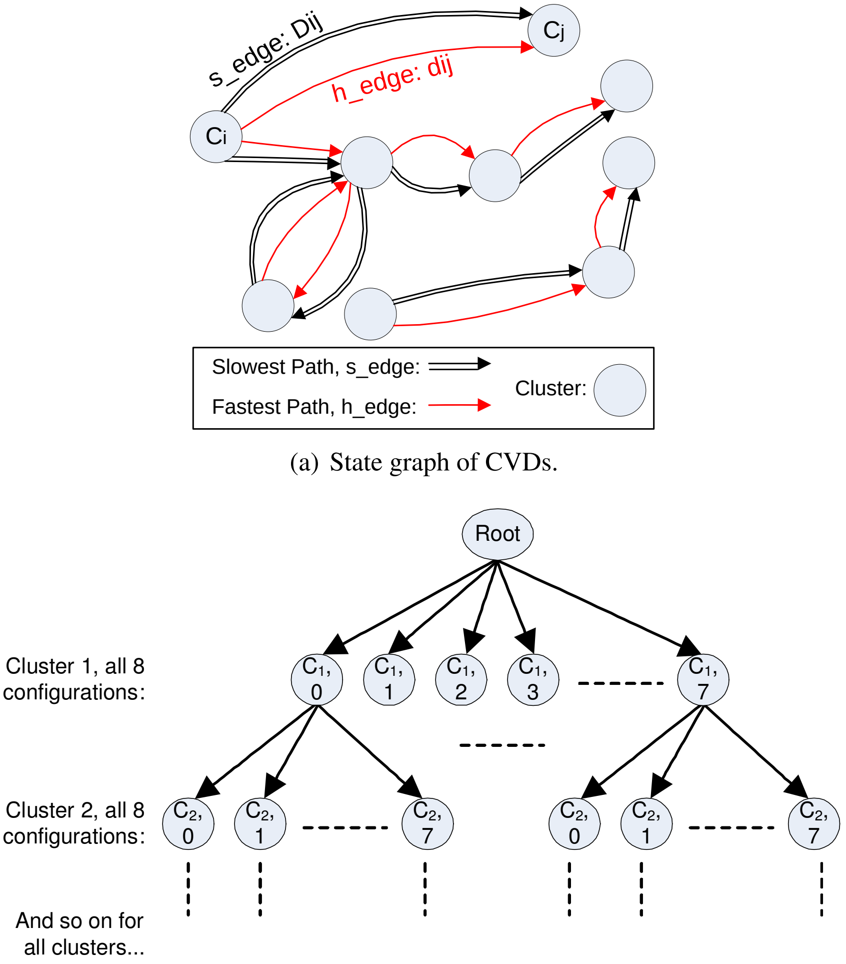
3.1.2 Finding the CVD Configuration to Measure the Delay of Fast Paths and Maximize the Frequency

As mentioned earlier in this section, to maximize the operational frequency, real delay values of the fast and slow paths are required. The delay of slow paths is measured through the frequency stepping method, which is a preprocessing step in the flow of Figure 3.1. In this subsection, we discuss the CVD configuration method proposed to calculate the delay of fast paths after fabrication. Afterwards, we discuss another CVD configuration method that maximizes the frequency of the CUT. The initial step to measure the post-silicon fast paths delays is to generate the graph of clusters (or their associated CVDs) and the combinational connections between them as explained next.

Generating the State Graph of CVDs and Paths

The first step of the software in the flow as shown in Figure 3.1 is creating the state graph of the CUT with CVDs inserted. Each node in this state graph represents one CVD (or its corresponding cluster of flip-flops); there will be two edges between two nodes CVD_i and CVD_j if there is at least one combinational path between any flip-flop from the group covered by CVD_i and any flip-flop from the group covered by CVD_j. The first edge is called *s_edge* and the weight of this edge is the delay of the slowest path between the two CVDs. The second edge is called *h_edge* and its weight is the delay of the fastest path between the same two CVDs. An example of a state graph is shown in Figure 3.2(a). s_edges represent the paths that may fail due to a 'S'etup-time violation if the frequency is too high and h_edges represent the paths that may fail due to a 'H'old-time violation resulting from skew provided by the difference in the clock arrival time of their source and destination CVDs.

Based on the explanations provided earlier in this section, the real delay value of s_edges are measured in the pre-processing step and the initial delay values for h_edges are estimated by false path aware STA. The real delay values of h_edges will be determined as explained in the next step. After all the real delay values are available, the CVD configuration for performance optimization will be found to help the circuit work in the maximum achievable frequency.



(a) State graph of CVDs.

(b) Complete search tree.

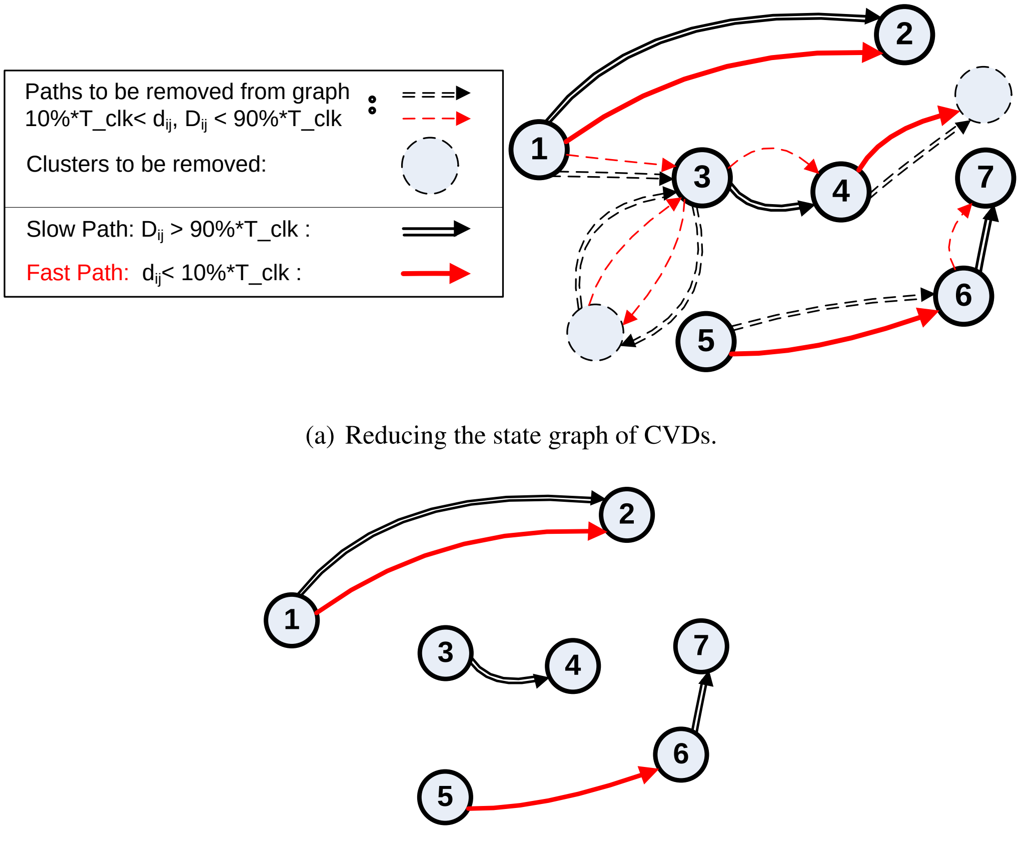
Figure 3.2: Complete state graph and complete search tree.

As mentioned in chapter 2, CVDs with three configuration bits can have eight different configuration values. If a branch and bound method is used for CVD configuration, the search tree will contain all the possible values for all the CVDs as shown in Figure 3.2(b), and then a valid CVD configuration for the circuit will be found from this search tree. However, the size of such a tree is impractical to be explored for large circuits.

Reducing the Size of the State Graph

To reduce the tree size by pruning, we need to determine if all CVDs need to be configured or not. *A key detail exploited by our algorithm is that CVDs can increase or decrease arrival times only by a limited value* [47] (e.g., 5% to 20% of the clock period). As a direct consequence, we remove those paths from the state graph for which adding or subtracting this limited value neither brings them to, nor removes them from the critical paths range or hold-time violation range.

To illustrate, if the maximum delay added to the clock arrival times provided by CVDs is $10\% * T_{clk}$, the paths that are removed from the state graph are the ones whose delays are less than 90% of the clock period and greater than 10% of the clock period. The resulting graph comprises single nodes that are not connected, as well as many disconnected subgraphs as shown in Figure 3.3(a). The disconnected subgraphs consist of nodes (clusters) and edges between pairs of nodes, which can be only h_edges, only s_edges, or both. In these subgraphs there will be many nodes connected to the rest of the graph only through h_edges. All h_edges incoming to 'CVDs without s_edges' will also be removed. The reason for removing them is that their destination CVDs do not have any s_edges and the configuration for the destination CVDs will not be changed. Furthermore, if the configuration for their source CVDs changes, it always increases (from '0' up to '7'), which implies that the particular h_edges cannot be violated and they have no effect on the configuration of other CVDs; therefore, those h_edges will be removed to reduce the graph size (Figure 3.3). In addition, all single nodes in the resulting graph will be removed since assigning any CVD configuration value to them does not affect the timing of other paths and CVDs which, reduces the graph size even further.


Paths to be removed from graph
10%*T_clk< d_{ij}, D_{ij} < 90%*T_clk :
Clusters to be removed:
Slow Path: D_{ij} > 90%*T_clk :
Fast Path: d_{ij}< 10%*T_clk :
1
2
3
4
5
6
7
1
2
3
4
5
6
7

(a) Reducing the state graph of CVDs.

(b) Reducing the size of the state graph.

Figure 3.3: Reduced state graph.

CVD Configuration

In our flow there are two different CVD configurations. The objective of the first one is to calculate the delays of the fast paths to accurately back-annotate h_edges. Then, after the real delay values of h_edges are calculated, the maximum CVD values should be adjusted accordingly to avoid hold-time violations in fast paths. Then, the second CVD configuration will be explained with the objective of finding the maximum operating frequency. In the following, the details of both CVD configuration algorithms are discussed.

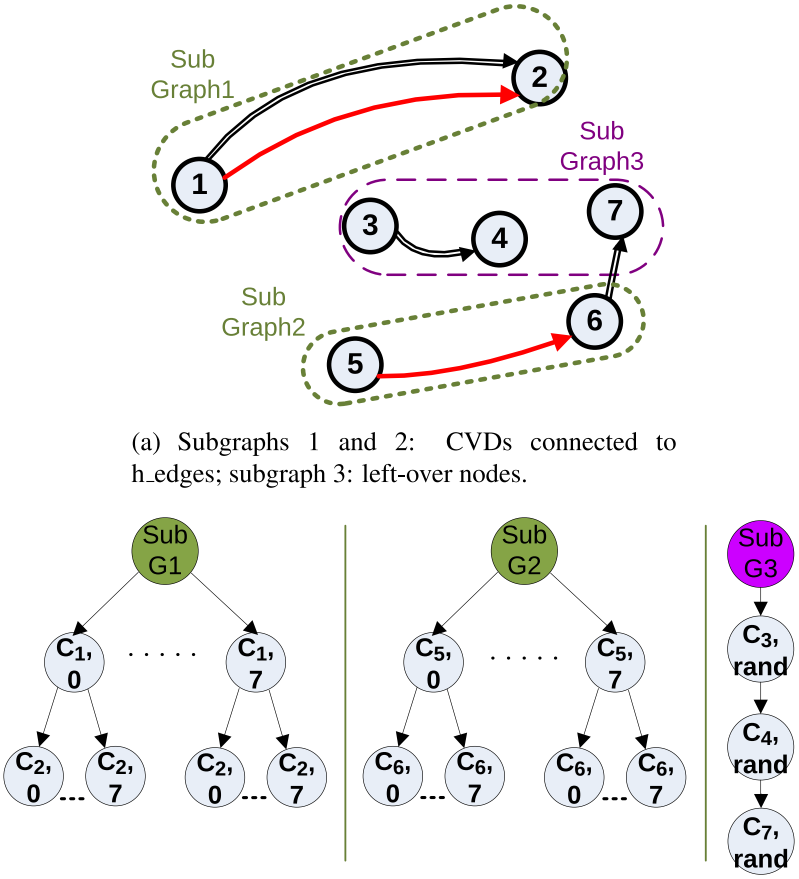
- **CVD configuration for Hold-time Measurement:** After the state graph is generated and reduced, the next step is finding the CVD configurations that will help in determining the real delay values of h_edges in the reduced state graph. Before getting to the details of this CVD configuration, we should emphasize that the CVD configurations found in this step will be applied to the CUT with a slow frequency to make sure that no path fails due to setup-time violation and the reason for any failure is the excessive skew between the source and the destination of h_edges which results is hold-time violations.

  To generate these CVD configurations, the search tree related to the state graph of Figure 3.3(b) should be traversed. However, with a smart pruning technique, we can shrink the search tree related to the state graph of Figure 3.3(b) to reduce the run-time of the CVD configuration for path delay measurement of h_edges. To prune the search tree, all the CVDs connected to h_edges (in subgraphs 1 and 2, Figure 3.4(a)) will be assigned all possible values, and the CVDs for which there is no h_edge (subgraph 3, Figure 3.4(a)) will be assigned a random value between the minimum and the maximum CVD value (‘0’ and ‘7’). This way, we prune the search tree of the reduced state graph to reduce the overall number of CVD configurations which directly reduces the time spent on the tester to apply all the CVD configurations to the CUT.

The pruned search tree of the graph in Figure 3.3(b) is shown in Figure 3.4(b). It is important to note that as of each node without h_edges, the size of the search tree is reduced 8 times (assuming there are 3 configuration bits). For this example, the size of the search tree is reduced $8 * 8 * 8$ times compared to the size of the search tree for the reduced state graph because there are three nodes in the reduced state graph without h_edges for which assigning any value to their CVD does not affect the timing of h_edges. Each path from root to leaf in each of the search trees for subgraphs is a CVD configuration for the corresponding subgraph which helps in finding the delay values of h_edges in that subgraph. After all CVD configurations are generated, they will be applied to the CUT and failing information will be collected by the tester. Then, based on failing range and passing range for each h_edge, the maximum CVD value for all the CVDs connected to h_edges will be calculated.

- **CVD configuration for Max Performance:** In this step, a constrained search tree with constraints on maximum CVD values (forced by real delays of h_edges) and also real delay values of s_edges will be used for finding the CVD configuration that helps the circuit work with the maximum achievable frequency. As an illustration, in the search tree displayed in Figure 3.5(b) that is driven from the subgraphs of Figure 3.5(a), the maximum value for CVDs 2 and 6 which are the destination CVDs of fast paths *may* be constrained to values smaller than ‘111’.

  It should be noted that this constrained search tree is generated based on the subgraphs displayed in Figure 3.5(a), which are different from the subgraphs of Figure



(a) Subgraphs 1 and 2: CVDs connected to h_edges; subgraph 3: left-over nodes.

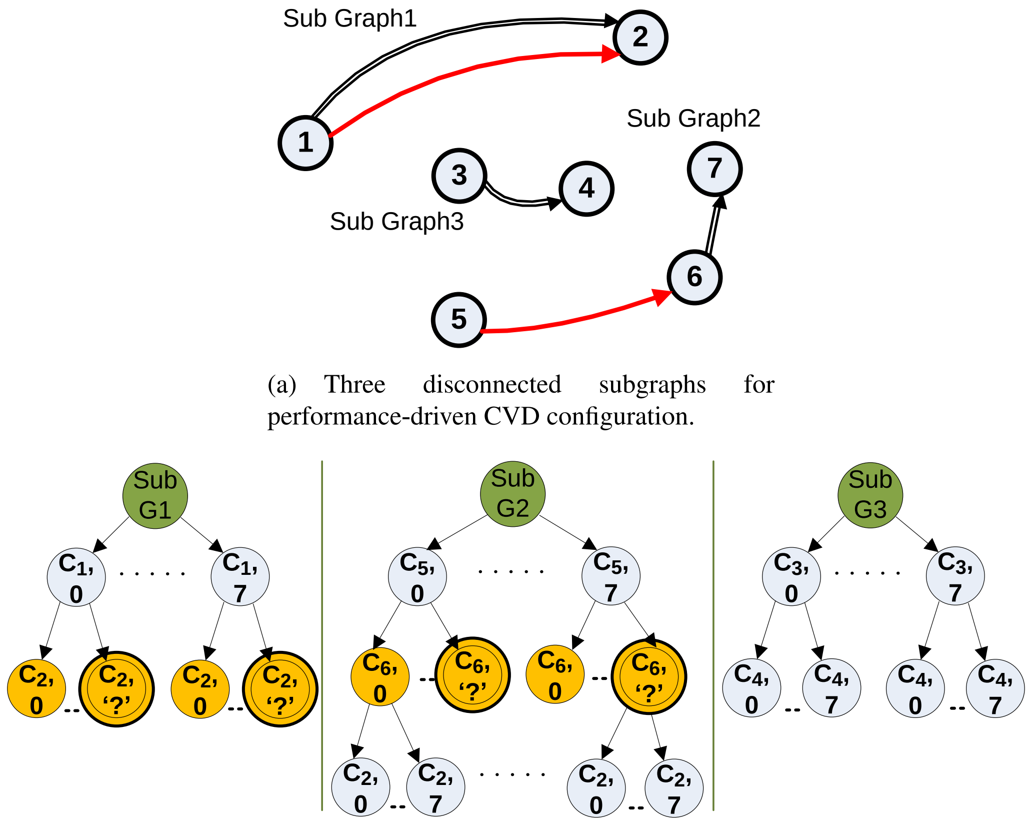
(b) Pruned search tree of the reduced state graph.

Figure 3.4: CVD configuration search tree for measuring the fast paths.

3.4(a). There is no connecting edge between the subgraphs shown in Figure 3.5(a), which means that the timing of subgraphs is also not related. Hence, the CVD configuration for maximum frequency for each subgraph can be found independently and concurrent with other subgraphs.

In addition to the above mentioned difference between search trees, the constrained search tree used for finding the maximum achievable performance (Figure 3.5(b)) is traversed with a different objective compared to the search tree used for CVD configuration for finding h_edges' delays. In the search tree of Figure 3.4(b), all the CVD configuration values are valid and they can be applied to the CUT for finding the real delay value of h_edges. However, the constrained search tree in Figure 3.5(b) is traversed based on the target frequency. Therefore, some CVD configuration values in the constrained search tree may not be valid based on the target frequency. As an example, assume that the maximum delay value provided by CVDs is $10\% * T_{clk}$. Therefore, the maximum frequency that may be achieved theoretically is equal to $f_{max} = \frac{1}{90\% * T_{clk}}$. However, with the constraints brought by the delay values of s_edges and h_edges, there is a possibility that no CVD configuration value exists to achieve the maximum frequency. It means that no CVD configuration for all CVDs can be found from the constrained search tree to make the circuit work with the target frequency. Therefore, the target frequency should be relaxed and the constrained search tree should be investigated for a valid CVD configuration.

As explained earlier in this section, we have assumed that the delay value of the slow paths can be measured at an early silicon stage by using a frequency stepping method. However, if these delay values are not available, we can modify our flow such that we can also calculate the delay values of slow paths. To be able to find the delay value of slow paths, after we have calculated the delay value of h_edges, the CVD configuration algorithm for maximizing the frequency will find the CVD configuration with real delay values of h_edges and the estimated delay values of s_edges provided by false path aware STA. The resulting CVD configuration will be applied to the CUT and the failing information will be collected by the tester. It is important to note that all the failures observed at this point are due to setup-time violations. After failures are observed, we relax the frequency on the tester step by step until no more failures are detected by the tester. Then, based on the



(a) Three disconnected subgraphs for performance-driven CVD configuration.

(b) Constrained search tree of the reduced state graph used to find the performance-driven CVD configuration.

Figure 3.5: Performance-driven CVD configuration.

frequency step in which each failure has not been observed anymore, the delay value for each s_edge will be calculated and updated in the state graph and a new CVD configuration will be found based on the updated delay values. This iterative process continues until no more failures are detected and all the values for the s_edges are updated in the state graph. In the last step, the CVD configuration corresponding to the maximum achievable frequency with real delay values for s_edges will be found similar to the loop of the flow in Figure 3.1.

After discussing the CVD configuration algorithms used for delay calculation of hold paths and also for maximizing the operational frequency, in the following subsection, we will discuss the CVD insertion algorithm that has been driven by the proposed CVD configuration algorithm.

3.1.3 CVD Insertion

In this subsection, the proposed CVD insertion or clustering algorithm is detailed. The output of this algorithm is the circuit with the inserted CVDs, which is the input to the flow from Figure 3.1, as well as to the CVD configuration algorithm.

Referring to the beginning of this chapter, the first condition for the CVD insertion algorithm is based on the fact that one cannot assign a CVD per flip-flop because of the power and area overhead caused by having as many CVDs as flip-flops in a design. Therefore, clusters or groups of flip-flops have to share CVDs. Furthermore, as explained in subsection 2.1.2, improvement in performance is achieved only if $FF1$ and $FF2$, which are the source and destination of a slow path, have been assigned to two different CVDs and therefore they could have two different arrival times. In the light of this example, the second condition of clustering is extracted, which is the source and destination flip-flops of slow (critical) paths should not go in the same cluster [49]. If they do, their clock edge arrival times would be equal and no timing improvement could be achieved.

The last condition for clustering can be extracted from the specific usage of individual flip-flops at the register transfer level of abstraction. For example, in case of shift registers, the designer would rather provide the same timing for all the flip-flops of a shift register; therefore, if the flip-flops of a shift register are assigned the same CVD, they will have the same clock signal arrival time.

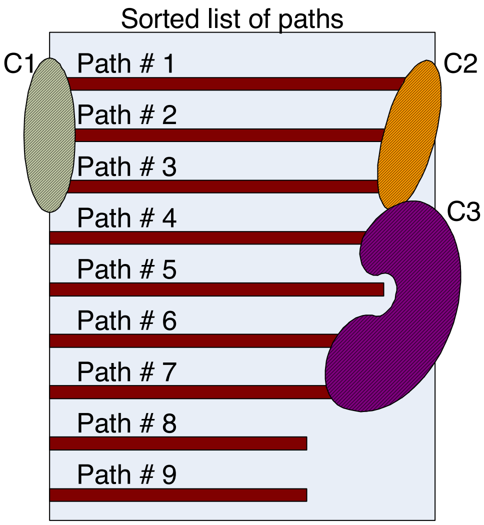


Figure 3.6: List of paths sorted based on delays. C1, C2, and C3 are examples of clusters.

Our CVD insertion or clustering method is performed based on the three above conditions. To meet these conditions, some information should be prepared. For the first condition, the maximum number of CVDs or the maximum cluster size are provided by the user of this algorithm and they can be determined based on custom constraints, such as the allowable budget for extra area and power. For the second condition, a list of paths sorted based on their delays should be prepared as shown in figure 3.6. All the path delays are obtained using a pre-silicon static timing analyzer and from this list, only the critical (slowest path) and semi-critical paths (top X% of slowest paths where X% is related to the tuning range of CVDs) are used for clustering and CVD insertion. For the third condition, all shift registers of design should be detected and provided as an input to the CVD insertion algorithm. After all the required information for clustering is provided, clusters of flip-flops will be created while the above conditions are satisfied.

3.2 Results

To validate our flow and algorithm, we have synthesized International Symposium on Circuits and Systems 89 (ISCAS) benchmark circuits [13] in a 90nm standard cell library. Then, pre-silicon delay estimations are provided by extracting the speedpaths' data using Synopsys Design Compiler's timing tool. The number of flip-flops to be assigned to each CVD is determined based on the analysis performed on the work from [47], where there is an average of 100 flip-flops per CVD, and more than twenty thousand CVDs reside in the design. In our case, since the number of flip-flops in ISCAS89 circuits is much less than that for the circuit reported in [47], we have applied the constraint of 25 flip-flops per CVD. Hence, assuming that the total number of calibration bits per CVD is k and the number of flip-flops sharing the same CVD is p, the overall area overhead for inserting CVDs will be $\frac{k}{p} \times 100\%$. Then, towards the end of this section, the impact of different cluster sizes on the operational frequency of s38417 benchmark circuit has been studied.

Initially, we have assumed that the range for CVD configuration values is between [0 .. 7], which has a total of 8 values and this range is 10% of the clock period [47]; i.e., if the CVD configuration value for a cluster is 7 (maximum), it means that the clock signal arrival time of that cluster should come later as much as 10% of the clock period. Therefore, the top/bottom 10% slowest paths are in the range that adding or subtracting the maximum CVD value (10% of the clock period), may push/remove them into/from the setup-time/hold-time violation range.

It is important to point out that in order to validate the effectiveness of our novel algorithm for performance tuning, we do not need to rely on a rather complex experimental setup with tester measurements for high-performance microprocessor designs, as it was the case in [47]. As a consequence, the variability or systemic delay defects that affect the post-silicon clock period have been generated randomly in virtual experiments that emulate the flow explained in Figure 3.1. Note, however, the virtual experiments are consistent with the tester methodology used in practice (e.g., [47]). Therefore, the effectiveness of the clock-tuning flow and the associated CVD configuration algorithm from section 4.1 are demonstrated; that is, regardless of the speedpath distribution reported by pre-silicon tools, we can use the post-silicon measurements (that capture the post-silicon effects) to tune the clock edges and maximize the operational frequency without relying on compute-intensive algorithms on the circuit's state graphs.

For a random set of failing flip-flops, we compare the clock period from our method against the zero-skew mode in Table 3.1. Note, the highest gain we expect to see is the maximum time that can be added to the slowest path, which is dependent on the maximum tuning range of CVDs, the accuracy of tester measurements, and the precision of the delay values provided by CVDs. In this experiment, the maximum additional time for the slowest path is equal to $max\_CVD\_Value - min\_CVD\_Value = 10\% * T$, which is consistent with the approach from [47]; therefore, the maximum theoretical gain (within the limits of the precision of CVD steps and tester measurements, as stated in section 4.1) is $10\% * T$.

Table 3.1: Performance comparison for ISCAS89 circuits.

	Zero-Skew	**Our CVD Configuration Algorithm**	
Circuit	**Clock Period (ns)**	**Clock Period (ns)**	**gain%**
s13207	3.356	3.082	8.2%
s15850	5.829	5.370	7.9%
s35932	94.835	86.834	8.4%
s38417	4.742	4.318	8.9%
s38584	18.465	16.783	9.1%

As shown in Table 3.1, our gains are very close to the maximum achievable gain. Furthermore, referring to Table 3.2, it should be noted that the CPU run-times are, as expected, very low. Even for the largest ISCAS89 benchmark circuits, which, to the best of our understanding, are of similar size to logic blocks in larger designs that contain critical paths, the run-times are below one second (as reported in Table 3.2). This is obviously a direct consequence of the pruning techniques that led to processing sparse state graphs where only tens to hundreds of edges were present. Interestingly, the same pruning techniques can be integrated into other algorithmic approaches for CVD configuration, such as constraint solvers, as employed in [53, 65].

In terms of comparison to the two related works on post-silicon clock tuning [48] and [65], the following should be noted. The work from [48] assumes that all the flip-flops are tuneable, which is also validated by the results reported only on small ISCAS89 circuits (with maximum of 19 flip-flops). On the other hand, one of our main objectives was to design a scalable method that can find the CVD configuration - given the accuracy of tester measurements and the granularity of CVD steps, as stated in section 4.1 - to improve the performance of the digital circuit (the runtime for our proposed method has been reported in Table 3.2). Furthermore, the work presented in [48] used critical path tracing method to find the source of each failure and no details are given on how to deal with hold-time violations. With respect to [65], it is important to note that it overcomes the practical limitations from [48]. However, because the focus is on speed binning for a microprocessor design, there

Table 3.2: CPU run-times for ISCAS89 circuits.

Circuit	**No. of flip-flops**	**CPU run-times in seconds**
s13207	669	0.240
s15850	597	0.346
s35932	1728	0.929
s38417	1636	0.846
s38584	1452	0.761

is no report on run-times for benchmark designs and no results on the performance gains for individual CUTs; when combined with the fact that the microprocessor design was not available to us, all of the above reasons pre-empt the possibility to perform a direct comparison. Nonetheless, it is essential to note that the key contribution from our work, which is pruning the state graphs to search the pruned CVD configuration tree exhaustively, as well as measuring the fast paths to account for hold-time violations when configuring the CVDs, can be used as add-on steps to improve any computational approach that uses post-silicon measurements for performance maximization.

Figure 3.7 explains how the maximum operating frequency changes when varying the number of configuration bits per CVD. More configuration bits lead to finer CVD steps, which will help update the delays of h_edges more accurately; also, more configuration bits provide more options to choose from in the search tree, however this can increase the CPU run-times. Figure 3.7 also shows how when the tuning range of the CVDs (X% of clock period) increases, the frequency also increases. The reason is that the number of paths accounted in the state graph increases and the extra timing from these paths can be used for the setup-time required by the slow paths. Figure 3.8 shows the frequency gain, as related to the upper bound (determined by the tuning range); as expected, when increasing the tuning range, the CVD steps (for a given number of configuration bits) become coarser, hence limiting the frequency gain.

In the following, the impact of the clustering algorithm for different cluster sizes has been studied and displayed in Figure 3.9. As can be noted in this figure, when the number of flip-flops per cluster increases, the frequency of the circuit decreases. The reason for this reduction in frequency is that when flip-flops are grouped to share the same CVD, any change in the clock arrival time provided by the CVD will result in a change in the clock

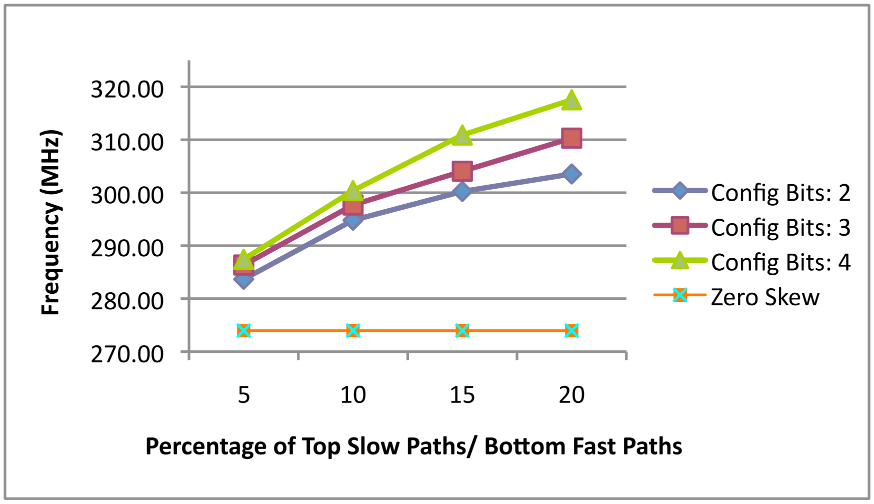


Figure 3.7: Impact of varying configuration bits and tuning range of CVDs on the frequency.

signal arrival time of all the associated flip-flops. Therefore, there are some flip-flops that although they may not need any change in their clock signal arrival time, their clock arrival time changes because they are part of the cluster. Since the timing for some of the flip-flops has been changed unnecessarily, the timing of the paths connected to these flip-flops may be restricted in two ways. It means that the timing provided to some of the paths connected to those flip-flops may not be enough, or, those flip-flops may not be able to lend time to other paths because their clock arrival time has been changed unnecessarily. As the number of flip-flops per cluster increases, the number of flip-flops in the same cluster that their timing changes unnecessarily increases and based on the above explanations, the timing of paths that need improvement cannot be fixed by time borrowing only, which means that the frequency needs to be relaxed. Hence, there is an inherent trade-off between ‘having more CVDs and better timing at the expense of larger area overhead’ vs ‘having a reasonable area overhead with less CVDs, however with lower performance’.

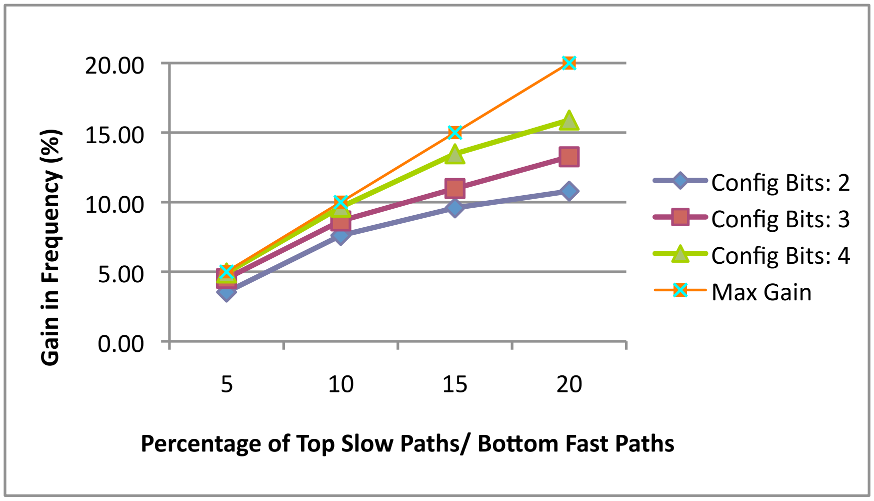


Figure 3.8: Impact of varying configuration bits and tuning range of CVDs on the frequency.

As a final remark, depending on the logistic details of the practical environment, our clock tuning flow can be applied to either a single device or a batch of devices. More specifically, if the goal is to performance tune one device at a time, the flow described in this chapter is applied as detailed. Otherwise, based on the data collected from the tester for a representative sample (batch) of devices, we will prioritize the failing information to be used by our CVD configuration algorithm. For example, if a flip-flop fails in nine out

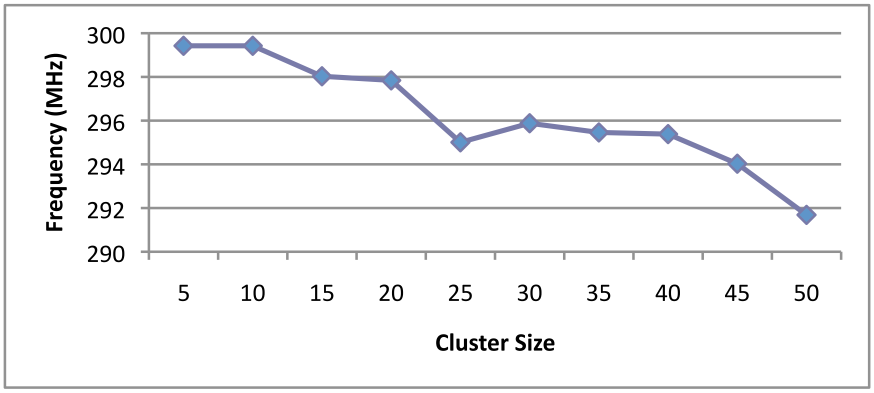


Figure 3.9: Results for ISCAS89 circuit s38417.

of ten devices and another one fails in one out of ten devices, then the former one will be used by our algorithm and the latter one not. Based on this reasoning, after a single CVD configuration is found it is expected that it will maximize the number of devices that pass at the maximum frequency. If the economics dictate it, the main advantage of using measurements on a sample set of devices is that the single CVD configuration can subsequently be reused for an entire volume of devices, which will not be subjected any more to the tester measurements.

3.3 Summary

The pre-silicon timing estimation tools find it more difficult to estimate speedpaths accurately in advanced process technologies. This has motivated the usage of clock tuning elements, such as CVDs, to perform post-silicon clock measurement and tuning [47]. However, finding CVD configurations without crunching huge amounts of data in state graphs (that capture the dependencies between the sources and sinks of the potential speedpaths) is not a straightforward problem for a large number of CVDs. In this chapter of the thesis we have introduced an algorithmic solution for post-silicon clock tuning that prunes the size of the solution space effectively.

Chapter 4

Using On-Chip CVDs to Address the Lifetime Performance Degradation

As overviewed in chapter 2, there have been a number of prior works that have studied the circuit aging effects and have provided solutions to address them. However, to the best of our knowledge, none of the existing techniques have exploited clock tuning elements, e.g., CVDs, to tackle this problem. In this chapter we investigate if and how CVDs can be used in-system for preserving the circuit performance when the delay degradation is predicted.

The existing techniques for determining CVD configurations rely on pre-silicon estimates and/or post-silicon tester measurements performed on a group of sample chips [26], [47], [65], [68]. For example, the tester data is processed and a CVD configuration is calculated, which determines the maximum achievable frequency for the group of sample chips. This pre-computed CVD configuration will be loaded through serial scan chains, with the expectation that it will result in the maximum performance for the majority of the chip samples. Circuits that do not work at this maximum frequency, but they are still operating correctly at a lower frequency, are speed-binned accordingly. Nonetheless, when high-performance is a critical objective and degradation caused by circuit aging is a concern, post-silicon measurements only on the first silicon samples are not sufficient to find the CVD configuration that can provide the maximum operating frequency throughout each circuit sample's lifetime.

When considering circuit aging, one has to account for the fact that each chip will be used in its unique environment and the applications running on one chip may be different from the ones running on the other chips. As a result, the path delay degradation will be different for the same path on different chips, as well as different paths on the same chip. This motivates us to find a unique CVD configuration for every single chip and this CVD configuration should have the possibility of being modified in-system, as needed. Thus, information from sensors that predict the failures in-system [4] can be leveraged to update the CVD configurations on-the-fly, in order to compensate for the extra circuit delays caused by aging. Nonetheless, as presented in the following sections, the ramifications of updating the CVD configurations in-system must be understood. As a consequence, we have developed a new hold-time violation prediction sensor and we present CVD insertion algorithms that are aware of the overhead that is introduced by both setup and hold-time violation prediction sensors. In addition, considering the fact that aging effects are not reversible, we do not focus on error detection and recovery; rather the focus is on prediction and self-adaptation. Toward this goal, we present a new on-the-fly CVD configuration mechanism with simple architectural features that are embedded on-chip and operate on-line without circuit interruption.

4.1 On-Chip Clock Tuning Mechanism

Before presenting our new clock tuning mechanism, we first provide a simple illustrative example to explain its intuition. This example is based on the original one presented in subsection 2.1.2.

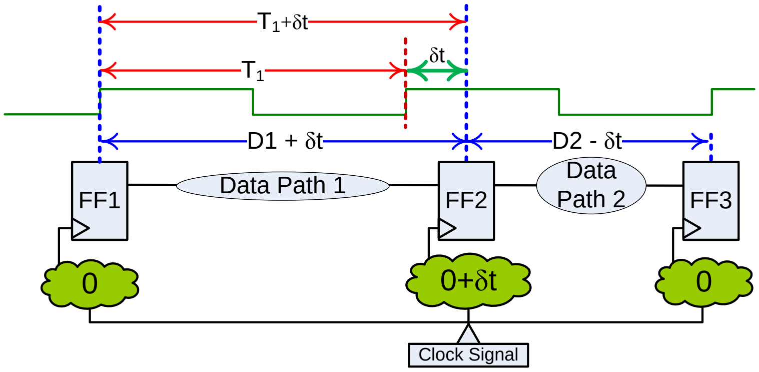


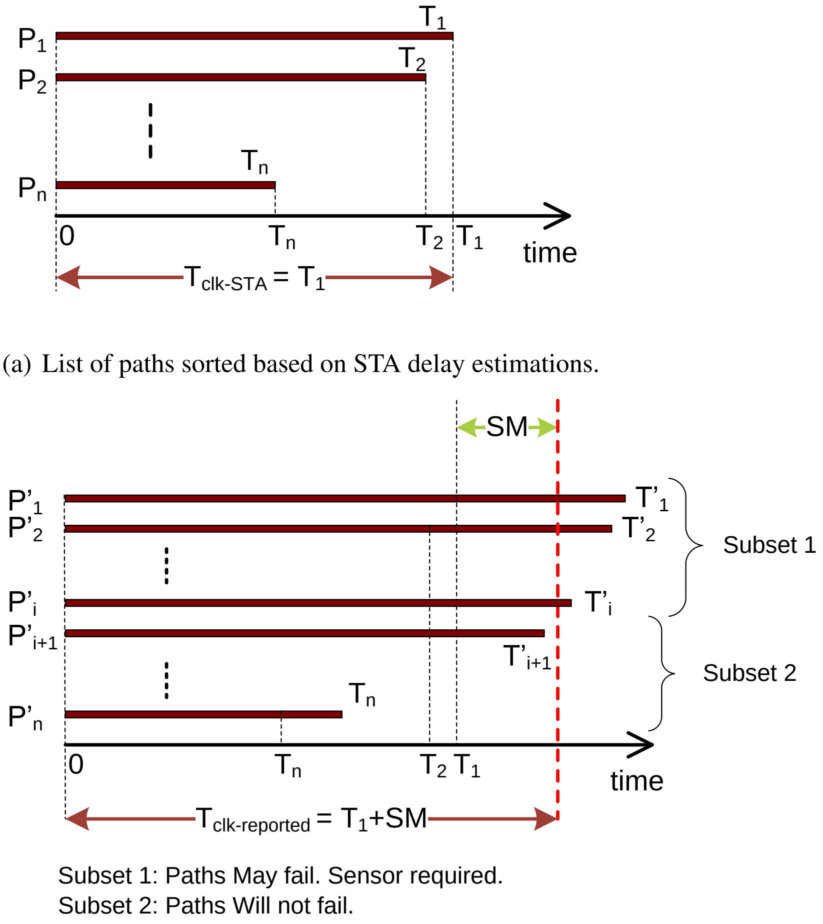
Figure 4.1: Using programmable clock tuning elements to improve the frequency.

In Figure 4.1, assume that the delay of the combinational logic between $FF1$ and $FF2$ is $D1$ and the delay of the combinational logic between $FF2$ and $FF3$ is $D2$, $D2 < D1$. There is also a clock tuning element on the clock signal of each flip-flop shown by a *cloud*. In ‘zero-skew’ configuration (clock signal arrives to all the flip-flops at the same time and all the clock tuning elements are assigned ‘0’), the minimum clock period is equal to the delay of the slowest path which is $D1 = T1 = T_{zero-skew}$. Now, assume that $D1$ is degraded by δt due to aging, and we want to keep the clock period as it was before: $T1$. To achieve this, we will exploit the timing slack available between the two paths. Therefore, the clock edge arrival time of $FF2$ should be delayed by δt (shown in Figure 4.1). In this case, the clock period will remain $T1$ and the timing window available for the path between $FF1$ and $FF2$ will be equal to $D1 + \delta t = T1 + \delta t$. In addition, the following condition must hold: $D2 < T1 - \delta t$. Note also that in order to avoid hold-time violations, no path from $FF1$ to $FF2$ should be excited at any time with a delay less than δt.

In the remainder of this section, we discuss the steps that are needed to build a circuit with the ability to self-predict the failures caused by setup/hold-time violations and also self-tune the performance in-system to achieve the maximum performance allowable by the characteristics of CVDs. It is important to note that the initial two steps which are *Insertion of sensor flip-flops* and *Insertion of CVDs*, are performed at pre-silicon (or design stage) and the last step which is *CVD configuration*, is performed at the post-silicon stage.

4.1.1 Insertion of Sensor Flip-Flops (Pre-Silicon)

The initial step for sensor flip-flops insertion is to identify which flip-flops in the design need sensors. To achieve this, we start with a Static Timing Analyzer (STA) to provide an estimation of the delay of the paths in the circuit. Then, the paths are sorted based on their delay value estimations. As shown in Figure 4.2(a), the delay estimation for Path1 (P_1) is equal to T_1, delay of P_2 is T_2, and so on and so forth. As discussed in reference [76], the maximum amount of timing degradation for each path is 20% of its original delay value. Therefore, if all the paths in Figure 4.2(a) are degraded by their maximum possible value (as shown in Figure 4.2(b)), the ones that violate the targeted clock period, are the ones that their destination flip-flops need sensors (Figure 4.2(b), subset 1). The targeted clock period is also reported by the designer as the maximum delay value estimated by STA ($T_{clk\_STA} = T_1$) plus a safety margin (SM). These concepts are shown in Figure 4.2(b). As shown in this figure, although the clock period has been extended to account for the safety margin, some paths may still fail due to timing degradation.



(a) List of paths sorted based on STA delay estimations.

(b) $T' = T(1+20\%)$.

Figure 4.2: Selecting the destination flip-flops that need sensors.

4.1.2 Insertion of CVDs (Pre-Silicon)

As shown in the illustrative example at the beginning of this section, the CVDs inserted at the clock path of flip-flops can provide non-identical clock arrival times and improve the performance. The works presented in [25], [49], [64], and [69] have investigated in CVD insertion for the circuits, without taking the degradation into account. However, in this work, we plan to insert the CVDs in the circuit, by considering degradation as a parameter.

Due to circuit limitations such as area overhead and extra power consumption, it is not practical to assign one CVD per flip-flop. Therefore, CVDs should be assigned only to the flip-flops that need them. To identify these flip-flops, it should be noted that the destination flip-flops of slow paths are the ones that may fail, or may be critical if timing degradation is taken into account [4], [65], [68], [75]. Therefore, they are the ones that may need their clock edge to be delayed. Detection of these flip-flops is similar to the previous subsection regarding sensors insertion. In fact, the flip-flops that need CVDs in their clock path, are the ones for which sensor flip-flops have already been inserted. To reduce the number of CVDs even further, multiple flip-flops can share the same CVD, similar to the works at [26], [47], [65], and [68]. Now, the first question that may be asked is 'Which flip-flops should be grouped together to share the same CVD?' (also called clustering).

Referring to the circuit shown in Figure 4.1, performance improvement can be achieved if $FF1$ and $FF2$ have different clock arrival times meaning that they should belong to separate clusters and CVDs. Therefore, the most important factor in clustering is that the source and the destination flip-flops of a slow path should not go to the same cluster to share the same CVD (Figure 4.3). Furthermore, if the source of a slow path is not the destination of another slow path, it does not need to go into a cluster, and it will not need a CVD. Further details of the CVD insertion algorithm can be found in subsection 3.1.3.

After clusters are generated and CVDs are assigned to each cluster, the circuit with inserted sensor flip-flops and CVDs is the input to our CVD configuration algorithm that works at post-silicon. Note, for the work and results presented in this chapter, we rely on the CVD insertion algorithm summarized in this subsection. However, it is important to point out that our CVD configuration algorithm is compatible with any method used for CVD insertion and it is not restricted to this particular CVD insertion approach.

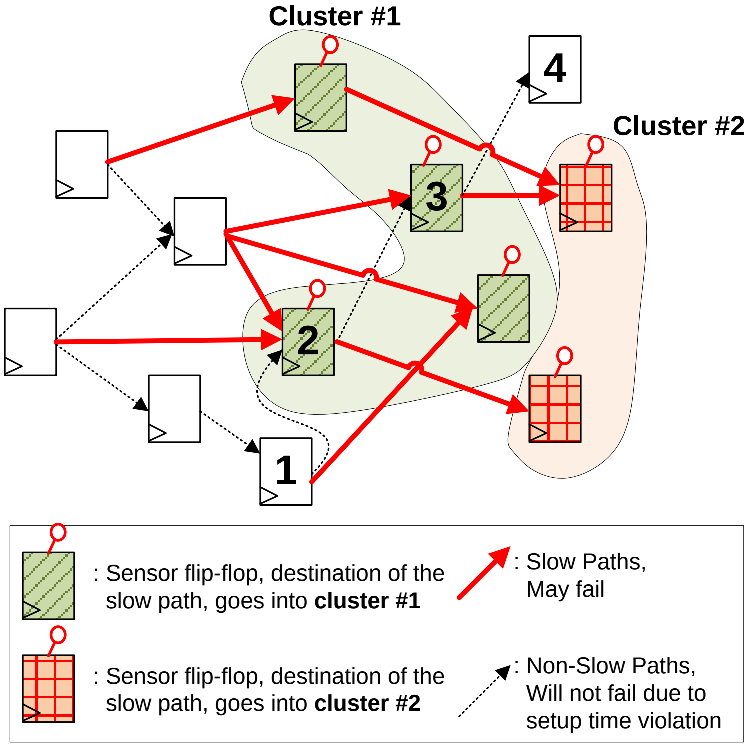

Cluster #1
Cluster #2
1
2
3
4
: Sensor flip-flop, destination of the slow path, goes into **cluster #1**
: Sensor flip-flop, destination of the slow path, goes into **cluster #2**
: Slow Paths, May fail
: Non-Slow Paths, Will not fail due to setup time violation

Figure 4.3: Clustering and CVD insertion.

4.1.3 CVD Configuration (Post-Silicon)

In this subsection, the in-system and on-the-fly CVD configuration mechanism is presented, while the objective is to find the maximum achievable performance. The reader should note that the maximum achievable performance is limited to the following three parameters: (i) the maximum delay value that a CVD can provide to the clock signal; (ii) the resolution of the delay that a CVD can add to the clock signal; (iii) the cluster size. The impact of all these three parameters will be elaborated in the results section.

During the in-system operation of a digital circuit equipped with sensor flip-flops and CVDs, anytime that a sensor predicts a setup-time violation, it implies that the particular flip-flop connected to the sensor may receive incorrect values in the near future because one of its incoming slow paths has been degraded. Therefore, the timing window provided for the paths leading to this flip-flop should be extended. However, we do not have to relax the frequency for the circuit since it will extend the timing for all the paths. In fact, extending the timing window *only for the paths leading to the flip-flop that is close to failure*, not only avoids the failure, but also helps the circuit maintain its operating frequency.

To extend the timing window of the paths leading to the flip-flop that is close to failure, the clock arrival time of the flip-flop can be delayed. Referring to Figure 4.1, delaying the arrival time of the destination flip-flop is equivalent to increasing its CVD configuration value. Therefore, each CVD can have a '*plus 1*' circuitry attached to it. Anytime that the sensor corresponding to a flip-flops is activated, it will enable the '*plus 1*' circuitry, which increments the CVD value by 'one', if the CVD value is less than its maximum configuration value which is '111'. Otherwise, the CVD cannot delay the clock signal more than the maximum CVD value and the frequency should be relaxed.

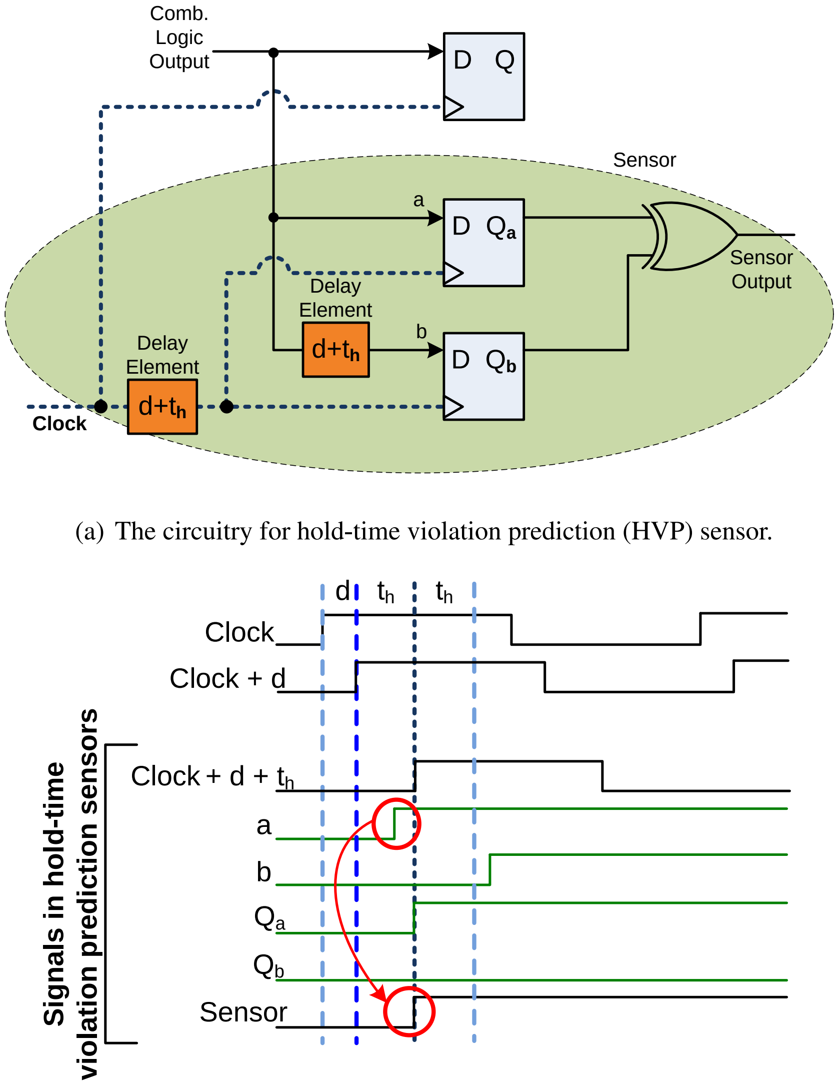
An important detail to be taken into account is the side-effect of CVD configurations that provide timing slack and may cause hold-time violations. Every time that the value of a CVD is to be incremented, it will delay the clock arrival time for all the flip-flops in its cluster. Hence, before incrementing the value of a CVD, we have to make sure that no fast path (FP) with its destination flip-flop in this cluster exists with its path delay less than the timing provided by the CVDs plus hold-time. This implies that the following inequality must hold: $t_{FP} > t_h +$ delay of $(CVD_{val} + 1)$. As a consequence, if no hold-time violation occurs and the current value of CVD is less than the maximum CVD value, if a setup-time violation prediction (SVP) sensor is activated in the corresponding cluster, the CVD value should be incremented to preserve the frequency; otherwise, the frequency must be relaxed to maintain the correct behaviour of the circuit.

As a result of the above discussion, since CVDs change the clock arrival times, monitoring the hold-time violations is a must in the circuits with CVDs. Therefore, we propose a new hold-time violation prediction (HVP) sensor, for which the schematic is displayed in Figure 4.4(a). In this Figure, d is *one CVD step* and t_h is the hold-time of the flip-flop.

Figure 4.4(b) is an example of the operation of the sensor shown in Figure 4.4(a). Through this example, we illustrate how this proposed sensor predicts the hold-time violation if a clock signal is to be delayed by one CVD step: d. In this figure, the original clock signal is named *clock*, and the clock signal in case the CVD associated with it increments will be labelled as $clock + d$. As shown in this figure, if the clock signal is delayed to $clock + d$, the hold-time of signal a will be violated. To predict this hold-time violation before delaying the *clock* signal, the following values should be assigned to the signals of the sensor shown in Figure 4.4(a).

The delayed *clock* signal, which is $clock + d + t_h$ will be sent to both flip-flops of the sensor: Q_a and Q_b. Signal a will be sent to Q_a flip-flop and the delayed version of it called b will be sent to Q_b flip-flop. Therefore, as shown in Figure 4.4(b), the values of the two inputs of the XOR gate that are Q_a and Q_b will be different at the edge of $clock + d + t_h$, which will set the value of *Sensor* to '1'.

Based on these explanations, HVP sensors must be inserted for the flip-flops which are connected to the fast paths and are in the clusters with CVDs. As a consequence, the number of HVP sensors will be smaller than the number of SVP sensors. The insertion of HVP sensors is performed before fabrication, at the same time when SVP sensors are inserted, and they are activated only if by incrementing the value of a CVD a hold-time might be violated (as illustrated in Figure 4.4(b)).



(a) The circuitry for hold-time violation prediction (HVP) sensor.

(b) An example for the operation of the above sensor.

Figure 4.4: Hold-time violation prediction sensor.

In terms of scalability, the following points should be made: (i) only the flip-flops that may receive incorrect values will need sensors; (ii) by avoiding one CVD per flip-flop, the number of CVDs is further reduced; (iii) the power consumption of the sensors can be controlled as follows. Based on the work from [4], the degradation is a gradual process. Hence, SVP sensors should not be turned on throughout the lifetime of a circuit.

A ‘Monitor’ signal can be defined, which keeps SVP sensors off for most of the times and turns them on for a short period of time. For example, they can be on for 10% of the lifetime of a circuit [3].

The power consumption for HVP sensors is even less than the power consumption of SVP sensors. The first reason is that the number of HVP sensors is much smaller than the number of SVP sensors because the flip-flops with HVP sensors are a subset of flip-flops with SVP sensors that are connected to fast paths. In addition to that, HVP sensors do not need to be on for all the time that SVP sensors are on. HVP sensors can always be off, unless an SVP sensor is activated in their common cluster, and the value of the CVD associated to that cluster is less than the maximum CVD value (‘111’). Only under these conditions HVP sensors will turn on until one of the HVPs in the associated cluster predicts a hold-time violation when they can be off until next SVP activation.

In the following, we will discuss how SVP and HVP sensors are integrated with CVDs into the circuit. This is illustrated in Figure 4.5. The pseudocode displayed in this figure captures the operation of the circuit shown by its left-hand side. As displayed, if at least one of the SVP sensors in a cluster is active, it means that a flip-flop may receive faulty values due to a setup-time violation in the near future. To avoid setup-time violations, extra time should be provided for that flip-flop by either time-borrowing from other paths using CVDs or by relaxing the frequency. If borrowing the time from other paths does not cause hold-time violation for the paths leading to that flip-flop, and the configuration value of the CVD associated with the cluster of that flip-flop can be increased, the value of the CVD will be incremented to provide the time required by that flip-flop without slowing down the circuit. Otherwise, the frequency should be relaxed to avoid hold-time violations that may be caused by incrementing the CVDs.

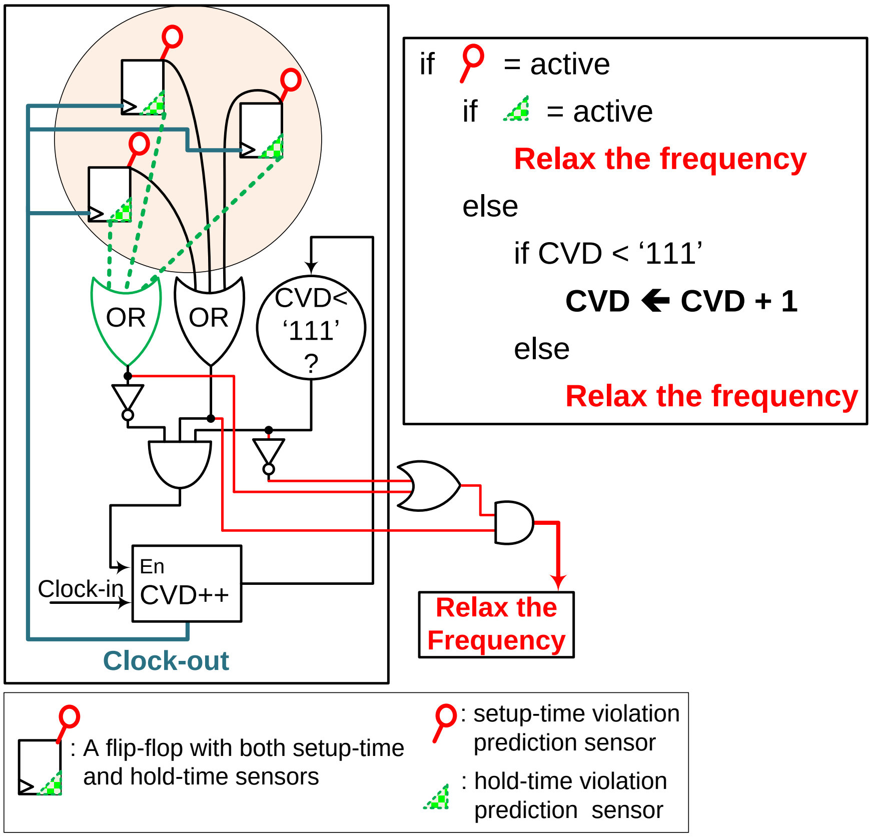


Figure 4.5: Integration of sensors with CVDs for on-chip CVD configuration.

It is important to note that the above pseudocode and the circuit will work as mentioned only if, when a SVP sensor has predicted a setup-time violation, the fast paths leading to that cluster are sensitized. Those fast paths are guaranteed to be sensitized after SVP activation if an online test is applied to sensitize them. In this case, HVPs only need to be on when SVPs are activated. Nonetheless, if there is no on-the-fly testing to sensitize fast paths when a SVP sensor is activated, the normal operation of the circuit has to continue and HVP sensors have to stay on all the time. Once HVP sensors predicted a hold-time violation without any active SVPs, the CVD increment for that cluster has to be undone and the frequency has to be relaxed.

Off-Chip CVD Configuration

If the area overhead for adding HVP sensors is considered too high, we can avoid their insertion and perform the evaluation of hold-time violation off-line. In this case, every time that an SVP sensor is activated, the system should be paused and all the information for SVP sensors and CVD values should be scanned out of the circuit. The evaluation of hold-time violation will be performed off-line based on STA estimates for fast paths. If no hold-time is violated in the corresponding cluster and its CVD value is less than the maximum CVD value, the configuration value of the CVD can be incremented. Otherwise, the frequency should be relaxed. After all these calculations are performed, all the CVD values will be uploaded through scan chains and the circuit will resume its operation.

It is important to note that there are two limitations in using off-line CVD configuration: (i) for some critical applications we may not be able to pause the system; (ii) the frequency found by this method is lower than the frequency found by the on-the-fly configuration method. The reason is that in the off-line configuration, the evaluation of hold-time violation is based on STA estimates. However, in reality, all the paths in the circuit, even the fast paths that may violate the hold-time, will degrade and their delay increases. As a consequence, STA estimates, which are smaller than the real degraded delay values, provide tighter bounds for CVDs to be incremented. Therefore, in the off-line method, the frequency may be relaxed before it is required (as shown in the results section). In order to maintain the benefits of the on-line CVD configuration method, yet to contain the number of HVP sensors, a clustering method will be presented in the next section.

4.2 Hold-Time-Aware Clustering and CVD Insertion

As mentioned in subsection 4.1.2, the clustering algorithm that has been used in our work so far, is based only on the delay information from the slow paths in a design. The base for this clustering algorithm is that the source and the destination flip-flops of a slow path should not be placed in the same cluster and should not share the same CVD as displayed in Figures 4.3 and 4.6(a). In this clustering algorithm, the slow paths where the delay values are close to each other will be selected and then their source flip-flops will form a cluster to share the same CVD. Also, the destination flip-flops of those slow paths form another cluster to share a CVD (Figure 4.6(a)).

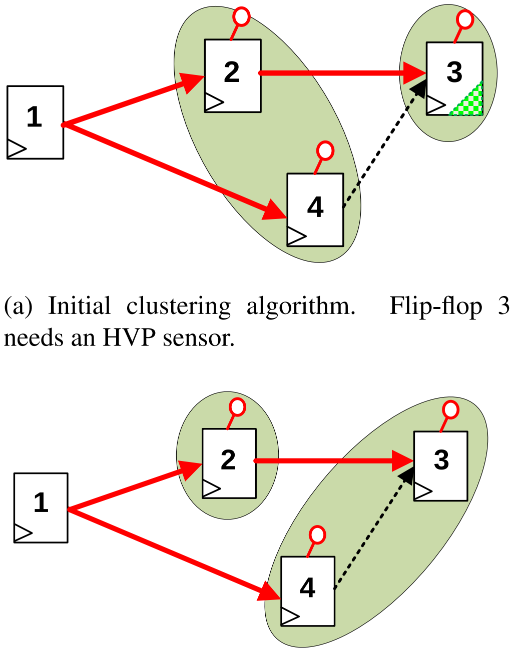
In the new hold-time aware clustering algorithm, in addition to using the delay information for slow paths, we use also the information from the fast paths. There are three groups of fast paths related to the speed limiting (slow) paths that are relevant to our algorithm:

- Group 1 - consists of the fast paths that only their source flip-flops are from the destination flip-flops of slow paths (Figure 4.3, fast path from flip-flop 3 to flip-flop 4). These groups of fast paths will not fail due to the slack provided by the CVD configuration, since their destination flip-flop will not have a CVD. Therefore, the clock signal arrival time for their destination flip-flop will not be delayed. Also, if the clock arrival time for their source flip-flop is delayed, it will reduce the slack between the source and the destination of these fast paths. As a result, these paths will not fail due to hold-time violation and the destination flip-flops for these paths will not need HVP sensors.
- Group 2 - consists of the fast paths that only their destination flip-flops are from destination flip-flops of slow paths (Figure 4.3, fast path from flip-flop 1 to flip-flop

2). It means that the destination flip-flops of these paths will have CVDs and their clock signal arrival times may be delayed if needed based on other paths' timings. However, the clock arrival time for the source flip-flops of these fast paths will not change because they are not destinations of slow paths and they do not have CVDs. Therefore, the source and destination flip-flops of these fast paths cannot be placed in the same cluster and may not have the same clock arrival time. As a result, these paths may fail due to hold-time violation and the destination flip-flops for these paths will need HVP sensors.

- Group 3 - consists of fast paths that both their source and destination flip-flops are from destination flip-flops of slow paths (Figure 4.3, fast path from flip-flop 2 to flip-flop 3). It means that both their source and destination flip-flops of these paths will have CVDs and their clock signal arrival times may be delayed and different. The source flip-flop of these fast paths will not need HVP sensors unless it is a destination of another fast path from Groups 2 or 3. The destination flip-flops of such fast paths will not need HVP sensors unless either of the two following conditions is valid for them: 1. If the destination flip-flops of fast paths are from Group 2 they will need HVP sensors; 2. If the source and the destination flip-flops of these fast paths are in two different clusters, their destination flip-flops will need to have HVP sensors.

Based on the above classification of fast paths, destination flip-flops from Group 1 do not need HVP sensors. Destination flip-flops from Group 2 will need HVP sensors. Also, the destination flip-flops of Group 3, when not clustered with their source flip-flops, will need HVP sensors (Figure 4.6(a)); otherwise, they will not need HVP sensors (Figure 4.6(b), flip-flop 3). The details of this hold-time aware clustering method are summarized as follows.



(a) Initial clustering algorithm. Flip-flop 3 needs an HVP sensor.

(b) Hold-time aware clustering algorithm. Flip-flop 3 will not need an HVP sensor.

Figure 4.6: The hold-time aware clustering impacts the number of HVP sensors.

The slow paths are sorted in the descending order of their delay values. Then, the source/destination flip-flops of different slow paths that have a fast path between them are placed in the same cluster if there are no slow paths between them. This will help in removing the HVP sensor required for the destination flip-flops of the fast paths (Figure 4.6(b)). In this clustering method, although the first priority for clustering is to place the source and destination flip-flops of fast paths in the same cluster (if there are no slow paths between them), the flip-flops are chosen from the descending list of slow paths in such a way that their delay is close to each other. The positive impact of this sorted list is that the delay difference between the slow paths, for which the source and destination flip-flops are clustered together, is smaller, which provides larger slack to the other slow paths that originate from a different CVD cluster.

In the results section, we will compare the impact of this hold-time aware clustering method with the initial clustering algorithm proposed in [38]. We will also compare this method against off-line CVD configuration method and the case when no CVD is applied for clock tuning purposes.

4.3 Results

To evaluate the proposed clock tuning mechanism, we have synthesized ISCAS89 benchmark circuits [13] using a 90nm standard cell library. Synopsys Design Compiler's timing tool is employed to provide pre-silicon delay estimates. Based on [76] and [24], the maximum amount of timing degradation for each path is assumed to be 20% of its original delay value. Therefore, we insert setup-time violation prediction (SVP) sensors for the destination flip-flops of the slow paths that may violate the targeted frequency if they are degraded by their maximum value. Also, we insert hold-time violation prediction (HVP) sensors for those flip-flops with SVP sensors that are connected to the fast paths and may fail due to hold-time violation.

The reader should note that there is a difference between HVP insertion in this work and the one presented in [39]. In [39], HVP sensors are inserted for all the flip-flops with SVP sensors to control the hold-time violation for all the flip-flops; therefore, the number of HVP sensors would be equal to the number of SVP sensors. However, a key observation is that not all the flip-flops with SVP sensors need HVP sensors. Only the ones which are connected to fast paths should be equipped with HVP sensors to predict the hold-time violation. The reduced number of HVP sensors is shown in Table 4.1 for three different safety margins: *SM* = 5%, 10%, and 15%. In this table, the purpose of clustering is to improve the performance, not to reduce the number of HVP sensors.

Table 4.1: The number of flip-flops that require SVP sensors and the extra area for inserting SVP sensors.

		SM=5%				SM=10%				SM=15%			
Circuit	**No. of flip-flops**	No. of potentially failing flip-flops	% of extra flops for SVPs	No. of HVPs	% of extra flops for HVPs	No. of potentially failing flip-flops	% of extra flops for SVPs	No. of HVPs	% of extra flops for HVPs	No. of potentially failing flip-flops	% of extra flops for SVPs	No. of HVPs	% of extra flops for HVPs
s15850	597	75	12.5%	1	0.3%	45	7.5%	1	0.3%	4	0.6%	1	0.3%
s13207	669	14	2.0%	0	0.0%	12	1.8%	0	0.0%	9	1.3%	0	0.0%
s38417	1636	141	8.6%	1	0.0%	105	6.4%	1	0.0%	71	4.3%	1	0.0%
s35932	1728	593	34.3%	186	21.5%	436	25.2%	186	21.5%	96	5.5%	186	21.5%
s38584	1452	1098	75.6%	9	1.2%	1090	75.1%	9	1.2%	907	62.4%	9	1.2%

As displayed in Figure 4.2(b), it can be seen that as the *SM* increases, the frequency is relaxed and the number of paths that may violate the more relaxed frequency will be reduced. Hence, the number of potentially failing flip-flops decreases and accordingly, the number of SVP sensors required for those flip-flops reduces. In Table 4.1, we have shown how different values assigned to *SM* (5%, 10%, or 15% of T_{clk_STA}) affects the number of SVP sensors.

In order to validate the effectiveness of our new CVD configuration algorithm, we have emulated the path delay degradations, which affect the post-silicon clock period, randomly in virtual experiments. Nonetheless, it should be noted that our virtual setup resembles the real world usage of a chip. The reason for this claim is that each chip sample is likely to be used in a unique environment, which is not identical to the environment of other fabricated samples of the same design. Furthermore, the types of applications that will run on one chip will be different from the applications that will run on the other chips. Therefore, each chip is degraded distinctly from the other ones. Thus, our virtual setup, which is generating the delay degradations for each path randomly, can capture the effectiveness of our CVD configuration method. Also, it should be noted that although some of the results provided in this section are only presented for a few large ISCAS89 benchmark circuits, consistent behaviour has been observed for the other large ISCAS89 benchmark circuits. For the results shown in Figures 4.7, 4.10, 4.11, and 4.12, the degradation model is assumed to be linear. It means that for each point in time, the path delay degradation is a linear function of time. As explained later in this section, we have also explored another degradation model, which essentially is independent from and orthogonal to the in-system clock tuning method proposed in this chapter.

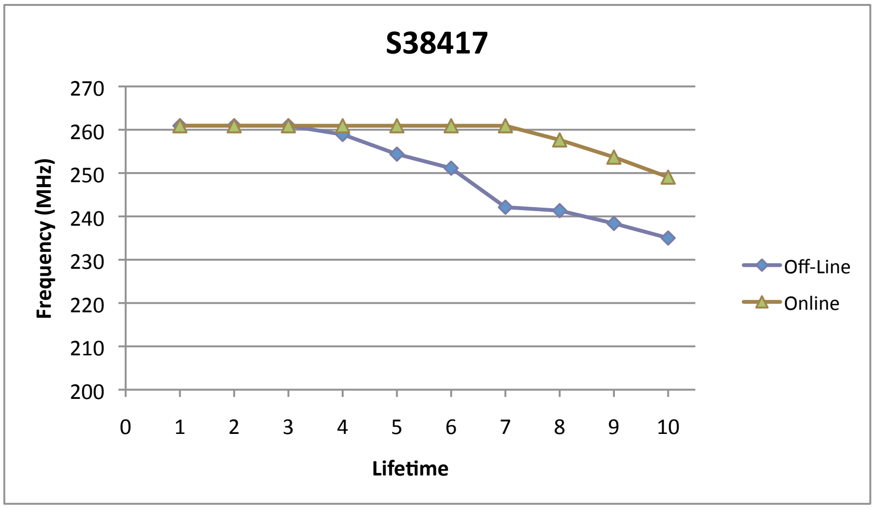


Figure 4.7: Off-line and on-line CVD configuration affect the frequency of s38417 differently during its lifetime.

The first experiment to be discussed is observing the performance when on-the-fly (on-line) CVD configuration and off-line CVD configuration are used. As shown in Figure 4.7, The frequency provided by the on-line CVD configuration is maintained in higher values compared to the frequency provided by the off-line CVD configuration, as anticipated based on the discussion from the previous section. For this experiment, the maximum tuning range for CVDs is 10% of the clock period and there are three configuration bits for each CVD, which is consistent with the circuit implementation presented originally in [47]. Also, SM is assumed to be $10\% * T_{clk\_STA}$.

For the experiments discussed next, we will focus only on the effectiveness of the on-line clock tuning mechanism. In the next experiment, we have studied how different safety margins can affect the operational frequency during the lifetime of the benchmark circuit. It can be observed that having larger SMs will result in a slower starting frequency (Figure 4.8) at the beginning of the lifetime of each design.

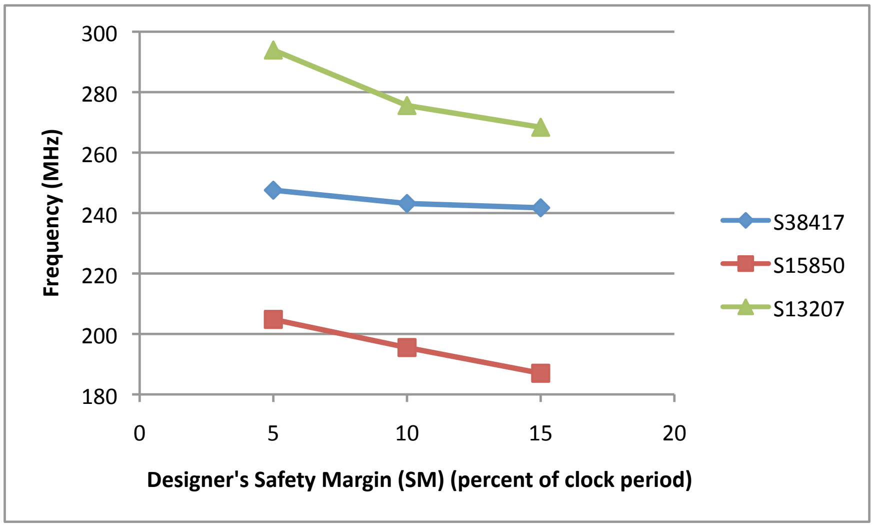


Figure 4.8: Frequency vs. safety margin.

As discussed in section 4.1, having one CVD per flip-flop is not practical for large designs. Therefore, clusters of flip-flops should be created, with a CVD shared between the flip-flops of the same cluster. In Figure 4.9, we show how different cluster sizes affect the performance. As displayed in this figure, when the cluster size is smaller, the operational frequency is higher. For the sake of this discussion, assume that the cluster size is ‘1’. It means that every single flip-flop is tuneable individually and if paths leading to one flip-flop need more time, by delaying the CVD of that flip-flop, we can provide the required timing margin for it. However, if cluster size is ‘n’, and the paths leading to only one flip-flop of that cluster need more time, the clock for all flip-flops of that cluster should be delayed. It means that for the other ‘n-1’ flip-flops in the cluster that do not necessarily need extra time, the clock signals are delayed. Therefore, the slow paths whose sources are in this cluster, may also fail because of less slack time.

Based on the above reasoning, the unwanted slack provided to some slow paths due to CVD configuration results in smaller improvement in frequency (Figure 4.9).

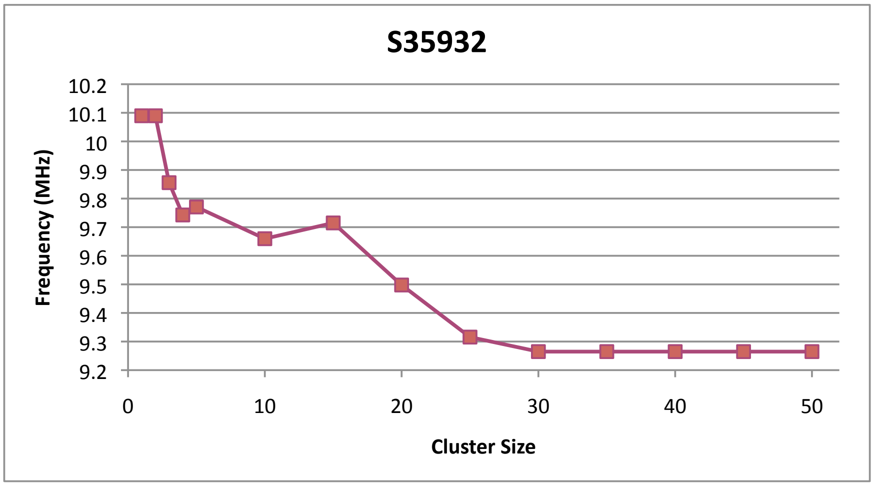


Figure 4.9: Frequency vs. cluster size for s35932.

Another important parameter (discussed in section 4.1) is how much a CVD can delay the clock of the destination flip-flop for a potential failing path (also called the CVD tuning range). The work from [47] assumes that if a CVD is configured at its maximum value, in can delay the clock signal by $10\% * T_{clk}$.

To evaluate our proposed method, we have performed our experiments for three different tuning ranges of CVDs which are: $5\% * T_{clk}$, $10\% * T_{clk}$, and $15\% * T_{clk}$ (Figure 4.10). First, as can be noted in this figure, the starting maximum operating frequency when the CVDs are employed is higher than the case when no CVDs are used. Second, in the first four points of measurements, although the degradation occurs and it is detected, for all the circuits with CVDs there is no need to relax the frequency; however, for the circuit without the CVDs, any time a failure is predicted, the frequency must be relaxed. Third, for the later points of measurements, it can be seen that the larger the CVD tuning range, the more improvement in performance can be achieved as the circuit ages.

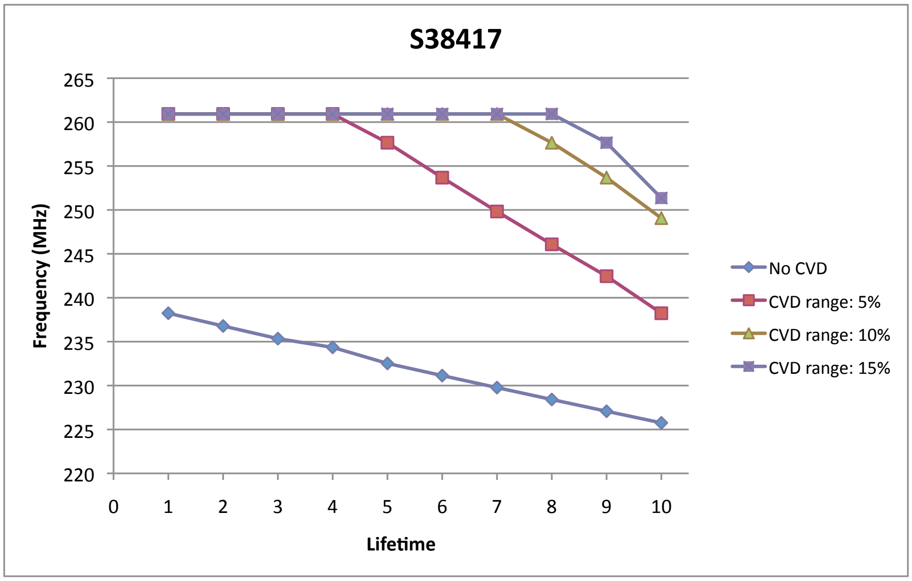


Figure 4.10: Frequency during the lifetime for different CVD tuning ranges.

The resolution of CVDs is also another important parameter for finding the maximum operating frequency. The CVDs presented at [47] have three configuration bits or eight configuration values. Assuming that the maximum CVD value is $10\% * T_{clk}$, the resolution for the delay added to the clock signal will be $resolution = \frac{10\% * T_{clk}}{7-0} = \frac{1}{70} * T$. However, assume that we only have one configuration bit for each CVD. It means that if we want to delay a CVD by half of its maximum value, we cannot do it because the only option is to delay it to its maximum value. In the experiments shown in Figure 4.11, it can be seen how finer resolution (more configuration bits) results in improved performance.

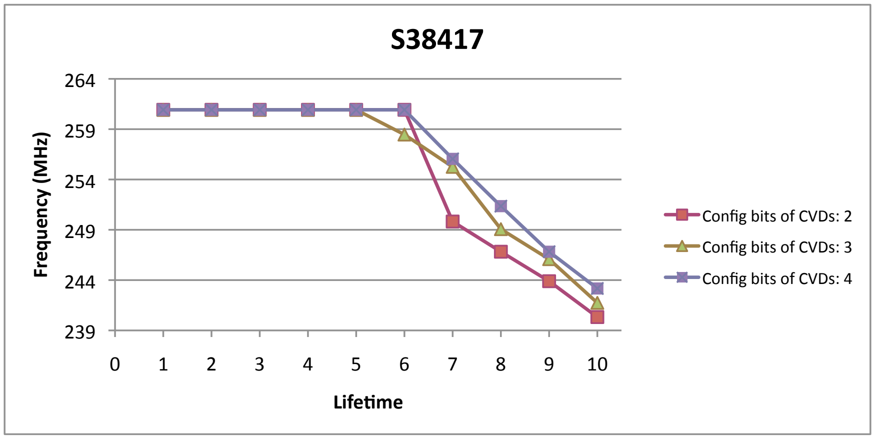


Figure 4.11: Frequency during the lifetime for different number of CVD configuration bits.

To study the effectiveness of our proposed hold-time aware clustering algorithm, which is based on both fast and slow paths delay values, Table 4.2 has compared the number of HVP sensors for the slow-path-aware and fast- and slow-path-aware clustering method. As explained in section 4.2, a new hold-time aware clustering algorithm has been presented to reduce the number of HVP sensors while the CVD configuration can be performed in-system and on-the-fly. This hold-time aware clustering method classifies the fast paths of a design in three groups as discussed in section 4.2. Then it focuses on reducing the number of required HVP sensors in group 3 by grouping the source and the destination flip-flops of the fast paths in the same cluster. The results for the reduced number of HVP sensors are shown in Table 4.2.

As displayed in this table, the number of HVP sensors is reduced. However, this saving in area comes at the cost of losing on the operational performance. As displayed in Figure 4.12, the CVD configuration algorithm with the clustering method based on both fast and slow paths information cannot maintain the operational frequency as high as the CVD configuration that is based only on slow paths information.

Table 4.2: Number of HVP sensors for the two clustering algorithms.

Circuit	**No. of flip-flops**	**No. of HVP sensors for slow path aware clustering**	**No. of HVP sensors for fast and slow-path aware clustering**
s15850	597	1	0
s13207	669	0	0
s38417	1636	1	0
s35932	1728	186	94
s38584	1452	9	0

In the comparison of the frequency provided by the hold-time aware clustering method against the off-line CVD configuration method, although in most of the points at the lifetime of the circuit they achieve the same operational frequency and off-line method does not use any HVP sensors, the hold-time aware clustering method has obtained better results at the end of the lifetime of the circuit (Figure 4.12). The reason for that is that in the on-line CVD configuration method based on hold-time aware clustering, the degradation of hold-paths is also considered in the CVD configuration, which relaxes the unnecessary tight bounds that over-constrain the CVD configuration algorithms. More importantly, in CVD configuration based on hold-time aware clustering method, when a failure is predicted, no application needs to be paused and no data needs to be processed off-chip, which is critical for the applications that cannot be interrupted during their operation.

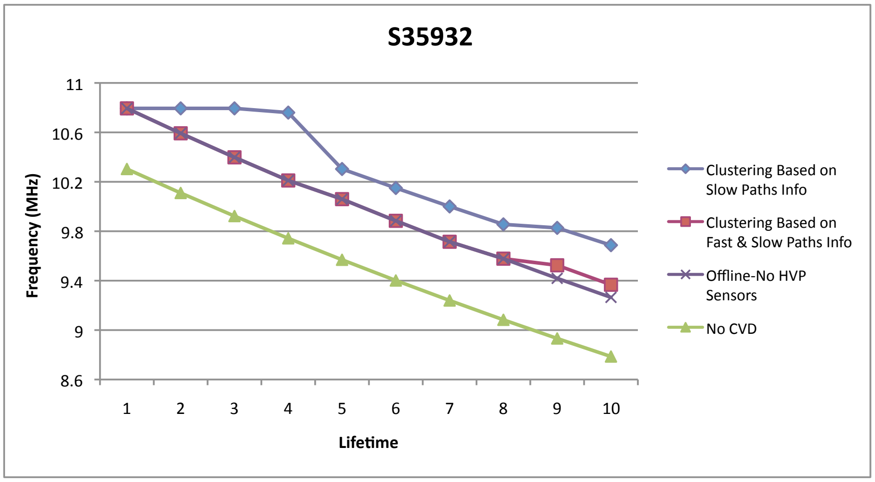


Figure 4.12: Frequency at different points in the lifetime of s35932 circuit.

In all the graphs that have been displayed so far based on 10 points in the life-time of the ISCAS89 benchmark circuits, we have assumed that the path delays are degraded linear to the time. However, the path delay degradation model may not be linear. Though our CVD configuration methodology is orthogonal to the delay degradation model of the design, we show the benefits of our method on another degradation model provided in [3] and [4]. As discussed in these works, delay degradation is relative to the temperature and voltage profiles, workload, and the amount of time elapsed (t^n, n=0.25 [4], or n=0.16 [3]). To study the behaviour of our CVD configuration method when the above degradation model is used, we have assumed that the multiplication of all the factors other than the t^n is constant and the paths delay degradation is relative to t^n, n=0.25 [4]. Based on this degradation model, the behaviour of our CVD configuration method is captured in Figure 4.13.

In terms of comparison of our work to the previous works, to the best of our knowledge, none of the works presented on delay degradation (such as [3], [4], and [74]) have worked on dynamically preserving the *performance* by using clock tuning mechanisms. The focus

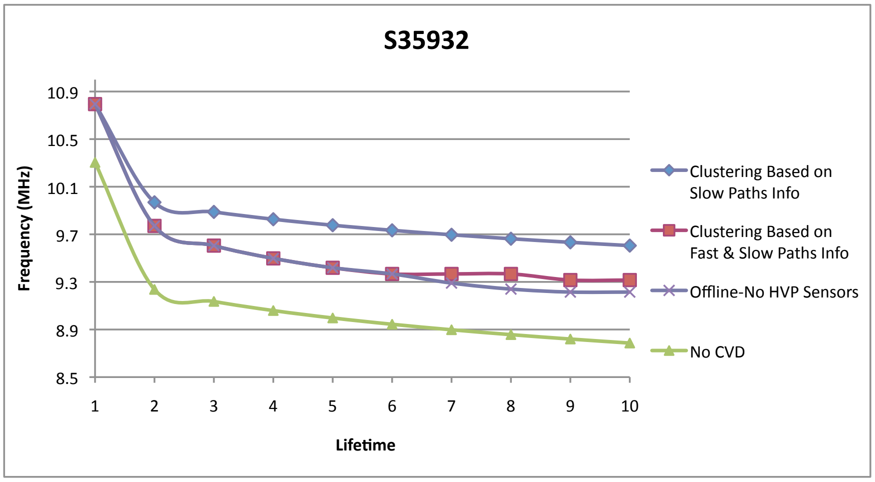


Figure 4.13: Behaviour of our CVD configuration method for the degradation model provided in [4].

in those works is more on finding when and where a failure may occur. Furthermore, the works presented on CVD configuration for performance tuning or speed-binning maximize the performance or the yield of high frequency bin only once after fabrication (i.e., at the beginning of circuit's lifetime). To the best of our understanding, these prior works do not account for the delay degradations caused by circuit aging. All of the above reasons pre-empt us from carrying out a direct comparison.

4.4 Summary

The research community has undertaken an effort over the past decade to improve the understanding on how circuit aging and timing degradation are related. In this chapter of the thesis, we have provided a new perspective on limiting the degradation of circuit's performance, as its path delays slow down due to aging. We have investigated how to combine the techniques for on-line failure prediction with the programmable clock tuning elements, in order to prolong the maximum post-fabrication frequency of a circuit in-system throughout its lifetime. To achieve this goal, we have proposed an effective on-chip mechanism for on-the-fly updates of the clock tuning elements, which facilitate uninterrupted circuit operation.

Chapter 5

Tuning the Timing of Scan Enable (SE) at Post-Silicon

In this chapter, we investigate an approach for addressing the scan enable (SE) timing problem by relying on delay tuning elements placed within the SE tree, similar to the ones placed in the clock tree for high-performance microprocessors. To the best of our knowledge, the works presented for tuning the timing of SE, including [5] and [50], have been proposed to operate at pre-silicon stage. However, process variations introduced during the fabrication process affect the timing of SE, which introduces a slack in the SE arrival time to different flip-flops of the design. In this chapter, we explain how clock vernier devices (CVDs, explained in chapter 2) can be exploited to tune the timing of SE at the post-silicon stage to compensate for timing slacks caused by process variations. Our proposed method can also be combined with the approaches presented in [5] and [50].

Our proposed method can be used to adjust the timing of the SE signal before the launch-on-shift (LOS) tests are applied. Conceptually our approach is illustrated in Figure

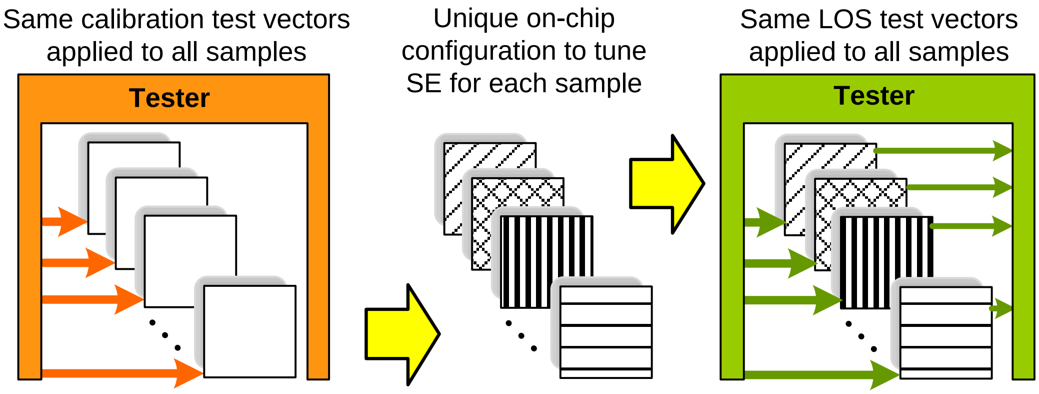

Same calibration test vectors applied to all samples
Tester
Unique on-chip configuration to tune SE for each sample
Same LOS test vectors applied to all samples
Tester

Figure 5.1: Tuning the SE signal for each circuit sample.

5.1. Before applying the LOS patterns, each chip will be subjected to a set of calibration patterns. Using on-chip structures, the calibration patterns will determine the unique timing configuration for the SE signal for the particular circuit under test (CUT). These configurations will be stored on-chip and the unmodified LOS test set will be subsequently applied.

5.1 Faulty SE and Its Impacts on LOS Testing

As introduced in chapter 2, in LOS testing the SE signal should transition at-speed. To detect the timing issues on the SE signal connected to a flip-flop, the impact of the faulty SE signal on the initialization data and the captured response of that flip-flop and the flip-flops connected to it should be studied.

Based on the waveforms displayed in Figure 5.2, early falling edge of the SE signal will result in: 1. one shift less of the test stimuli; and 2. double capture of the response. Having one shift less of the test stimuli at a flip-flop affects the initialization data for the flip-flops that receive their test stimuli from this flip-flop. Also, double capture of the response in a flip-flop may invalidate the captured response. Therefore, early falling edge of SE observed by a flip-flop affects both the same flip-flop and the flip-flops that are destinations of this flip-flop. A late falling transition on SE for a flip-flop is similar to the stuck-at-1 fault on the same line and results in no-capture of the response to the test vector [22]. As a consequence, late falling edge observed by a flip-flop affects only the same flip-flop.

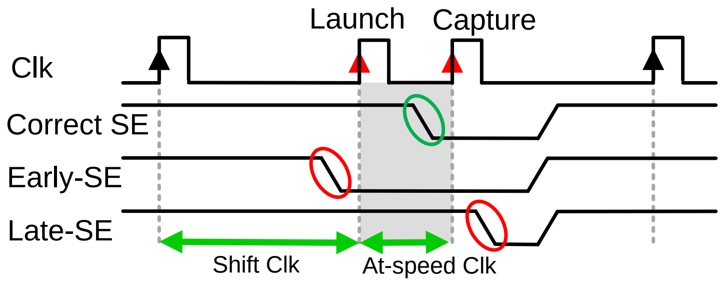


Figure 5.2: Early/late falling edge of the SE signal.

In this chapter, we explain how CVDs can be exploited to tune the timing of the SE signal and reduce the timing slack for arrival of the SE signal to the scan cells. The overview of our proposed sequence of tasks performed at the pre-silicon and post-silicon stages is captured in Figure 5.3. To provide the on-chip ability for tuning the timing of the SE signal, four steps should be taken both in the design stage and after fabrication. Initially, the two pre-silicon tasks which are clustering and architectural modifications to facilitate the on-chip detection of the timing issues of the SE signal should be performed. Then, the two post-silicon tasks including test application to find the timing issues of SE, and, on-chip CVD configuration will be performed to calibrate the timing of the SE signal.

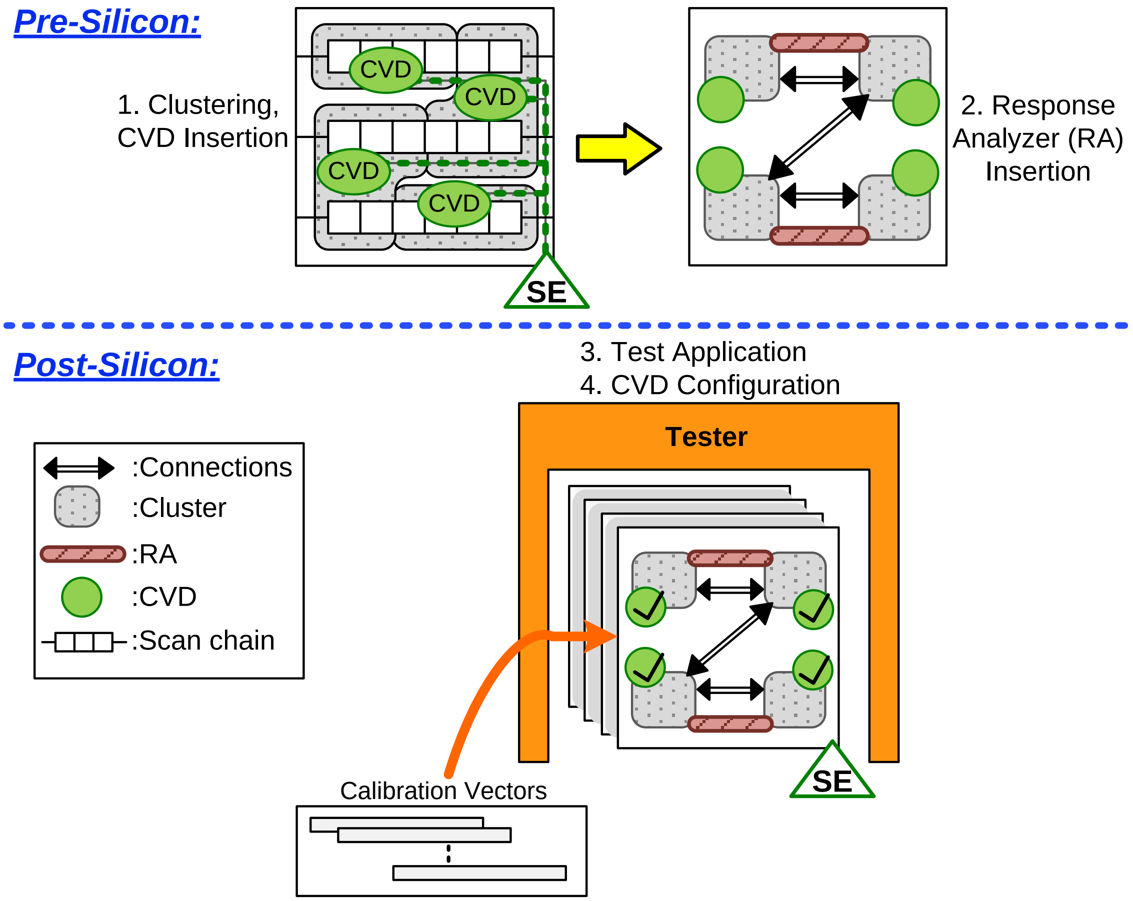


Figure 5.3: The sequence of pre/post-silicon tasks performed to tune SE by CVDs.

5.2 Design-Time Tasks

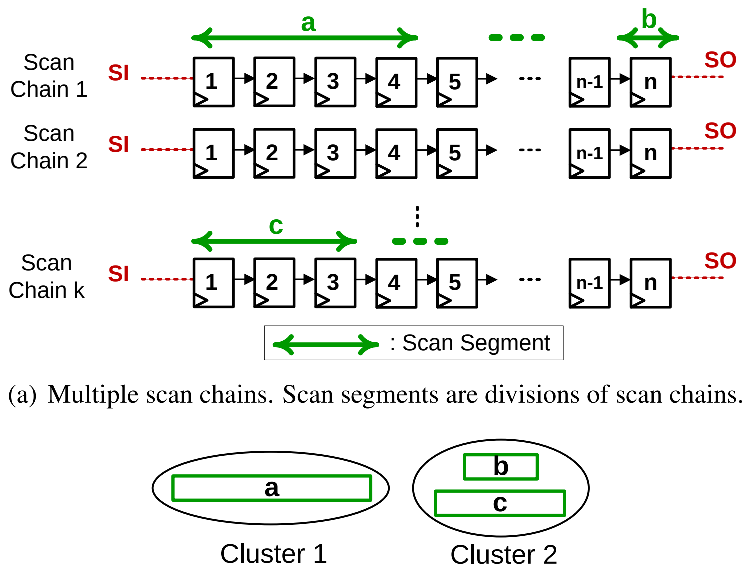
To provide the on-chip tuning ability for the SE signal in the LOS test mode, two tasks should be performed during the design stage: 1. generating groups of scan cells that share the same CVD; 2. providing the extra circuitry that facilitates the on-chip detection of timing issues on the SE signal.

5.2.1 Clustering and CVD Insertion

Based on the reasoning provided in chapter 3 and in [47] regarding the number of CVDs in clock trees, not every single flip-flop can have its own CVD due to excessive area and power overhead. Similar reasoning can be applied to SE trees, which is, one cannot assign a CVD to each scan cell. Therefore, groups of scan cells should be created to share a CVD; this task is called *clustering*.

To generate clusters of scan cells with the purpose of tuning the timing of SE, we should note which scan cells can be clustered together. If the SE net is generated through the pipelining method [50] or it is locally generated [5], [79], at the final stage of the SE tree, the SE signal is provided for a group of scan cells. Since there might be one or multiple scan chains in each digital circuit, these scan cells that receive their SE from the same stage of the SE tree may or may not be from the same scan chain (Figure 5.3-1).

In our clustering algorithm, since we do not have information for the routed pipelined SE tree and the local generation of SE, we will not know which scan cells receive their SE from the same stage of the SE tree. Therefore, we rely on the clustering algorithm that chooses scan cells in each cluster randomly. In the clustering algorithm that we use, each cluster can have one or two scan segments. Scan segments are divisions of scan chains and they are selected as consecutive scan cells. If a cluster has only one scan segment, that scan segment is chosen as consecutive scan-cells from one scan chain and the size of the scan segment should be equal to the cluster size that is assumed to be fixed and it is determined by the designer. If a cluster has two scan segments, they are selected from two different scan chains and the sum of the lengths of scan segments should not exceed the cluster size. These concepts are displayed in Figure 5.4.



(a) Multiple scan chains. Scan segments are divisions of scan chains.

(b) Clusters consist of single or multiple scan segments.

Figure 5.4: Clustering.

After the clustering task is completed, a CVD will be assigned to each cluster of scan cells. CVDs inserted at the design-time will be used to calibrate the timing of the SE signal after fabrication. To tune the timing of SE, first, we should know which clusters receive faulty SE signal. In the following, details of the additional circuitry required for detection of faulty SE will be explained.

5.2.2 Additional Circuitry to Detect the Faulty SE On-Chip

It should be noted that in our proposed calibration process displayed in Figure 5.3, the objective is to diagnose the clusters that receive faulty SE signal on-chip and tune the timing of their SE by on-chip configuration of their associated CVDs. This is to avoid offloading the captured responses of clusters and performing the diagnostic calculations off-chip, which will speed up the SE tuning process.

To be able to diagnose the faulty SE received to a cluster on-chip, we should be able to distinguish between the correct and incorrect captured responses. Hence, the correct response should be known on the chip and the architectural features should also be provided to distinguish between the correct and the incorrect responses. As a consequence, anytime that a calibration pattern is applied to the circuit, the correct (reference) response should also be uploaded on the chip to be compared with the captured response. After the capture clock edge (Figure 5.2), the generated response will be compared against the uploaded reference response. If there is a mismatch between them, it declares that the timing of SE has not been correct and the CVD corresponding to the failing cluster should be exploited to tune the timing of SE (Figure 5.5).

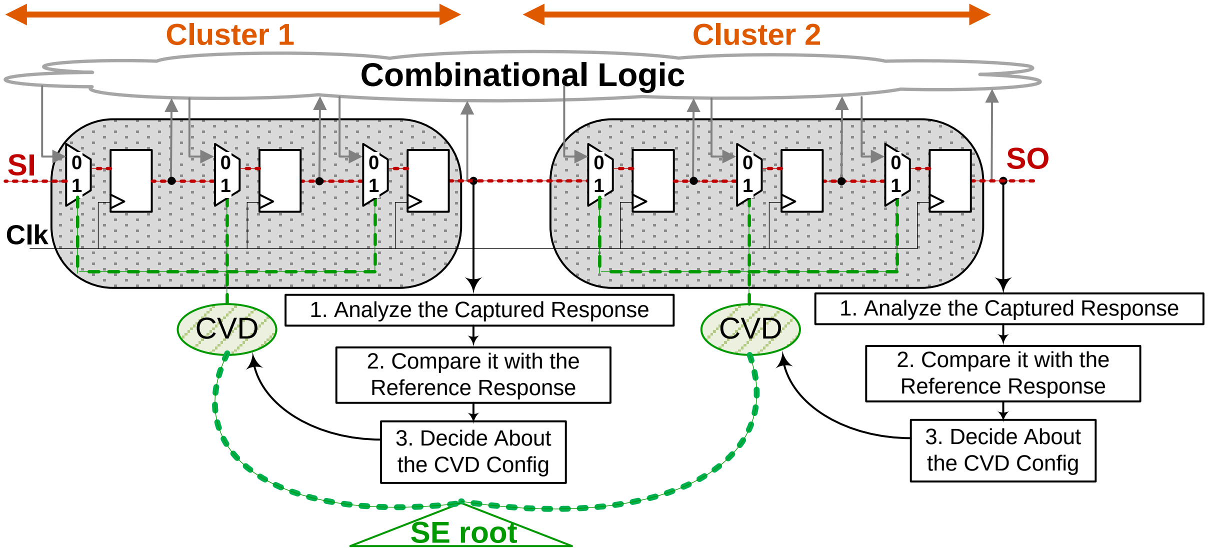

Cluster 1
Cluster 2
Combinational Logic
SI
SO
Clk
0
1
CVD
CVD
1. Analyze the Captured Response
2. Compare it with the Reference Response
3. Decide About the CVD Config
1. Analyze the Captured Response
2. Compare it with the Reference Response
3. Decide About the CVD Config
SE root

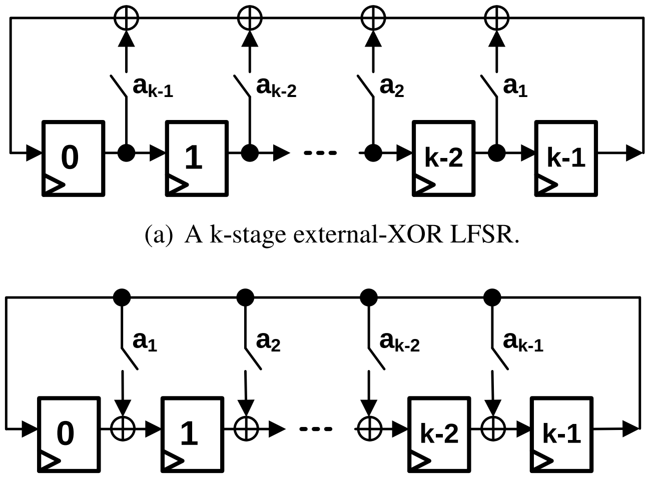
Figure 5.5: Insertion of the extra circuitry for failure detection on the SE signal and using its outcome to tune the CVD.

If the complete reference response is to be uploaded on the chip, the time for uploading the reference response for each cluster under test will be equal to the cluster size. To avoid the excessive time for uploading the reference response, *signatures* can be generated for both the reference response and the captured response. The signature for the captured response (SCR) can be generated on-chip and the signature for the reference response (SRR) can be uploaded on the chip to be compared against each other. To generate the SCR on-chip, additional circuitry, also called Response Analyzer (RA), should be provided for each cluster at pre-silicon. An RA can be a Linear Feedback Shift Register (LFSR) [73] that generates the SCR as the captured response is shifted-out from the cluster to the RA. The SRR is also uploaded as an extension to the applied calibration pattern.

The size of SCR, SRR, and RA will be elaborated next by describing LFSRs [20], [52].

LFSRs

An LFSR of size *k* consists of *k* flip-flops and a selected network of XOR gates. As shown in Figure 5.6(a), if XOR gates are placed on the external feedback path, the LFSR is called an external-XOR LFSR. Similarly, if XOR gates are placed between adjacent flip-flops, as displayed in Figure 5.6(b), the LFSR is referred to as an internal-XOR LFSR [20], [52].



(a) A k-stage external-XOR LFSR.

(b) A k-stage internal-XOR LFSR.

Figure 5.6: Instances of LFSRs [20], [52].

The structure of an LFSR can be described by a *characteristic polynomial* defined over *Galois field* GF(2) [73], called $f(x)$:

$$f(x) = 1 + a_1 x + a_2 x^2 + ... + a_{k-1} x^{k-1} + x^k$$

In the above GF(2) equation, a_1, a_2, ..., a_{k-1} are 0 or 1, the multiplication stands for the logic AND and the addition stands for the logic XOR. If $f(x)$ divides $1+x^T$ for T as the smallest positive integer, then T is called the period of $f(x)$ [52]. If $T = 2^k - 1$, then $f(x)$ is a primitive polynomial and the total number of distinct values generated by the LFSR will be maximized to $2^k - 1$. It should be noted that the all '0' value in the LFSR's k flip-flops is the only state that does not belong to the $2^k - 1$ sequence produced by the LFSR. LFSRs can be used for both stimuli generation and response analysis. For response analysis, they can have a single input or multiple parallel inputs. LFSRs that are used to generate signatures with single input are referred to as **??**Single-Input Signature Register (SISR) and an LFSR with multiple parallel inputs is called Multiple-Input Signature Register (MISR).

Based on the above discussion, in the context of our work, the response analyzer (RA) assigned to each cluster can be a SISR (for clusters with a single scan segment) or a MISR (for clusters with multiple scan segments, each scan segments should be connected to one of the MISR's inputs). To improve the area investment for RAs, we assume that the size of RAs inserted in the circuit is $\lceil log_2(m) \rceil$ for cluster size m. Therefore, for a digital circuit with n flip-flops, the total number of extra flip-flops for having one RA per cluster will be $\frac{n}{m} \times (\lceil log_2(m) \rceil)$. Reducing the RA size does increase the probability of aliasing, i.e., a fault-free response and the faulty one produce the same signature. Note, although MISR aliasing probability is a topic that was widely researched in the past three decades, a detailed analysis of the topic is beyond the scope of this thesis.

By performing clustering, CVD insertion, and inserting RAs in the circuit, the pre-silicon tasks are completed and the architectural features for finding the faulty SE are provided. In the next section, we discuss the post-silicon tasks which will speed-up finding the timing issues of the SE signal.

5.3 Post-Silicon Tasks

In order to tune the timing of the SE signal, there is a need to know what ranges of timing issues of the SE signal can be tuned by the use of CVDs.

5.3.1 Tuneable Range of Faulty SE

If the SE signal observed by some scan cells in a cluster has an early/late falling edge and it is on-time for other scan cells of the same cluster (Figure 5.7(a)), the timing issue can be fixed by adding/subtracting a delay to/from the arrival of the SE signal in order to bring it within the at-speed clock pulse. However, if some of the scan cells in a cluster observe an early falling edge and other scan cells in the same cluster observe a late falling edge of SE, CVDs cannot fix the timing of SE for all the scan cells in that cluster. The reason is as follows: adding delay to the arrival time of SE may fix early falling edges, but will worsen the late falling edges by making them arrive later. Similar reasoning can be applied while subtracting delay from the late SE signal to move it to the left within the at-speed clock pulse (Figure 5.7(b)).

The reader should note that in this chapter, the two terms of ‘moving SE to the left’ and ‘moving SE to the right’ are used to fix the timing of the late and early SE respectively (Figure 5.7(a)):

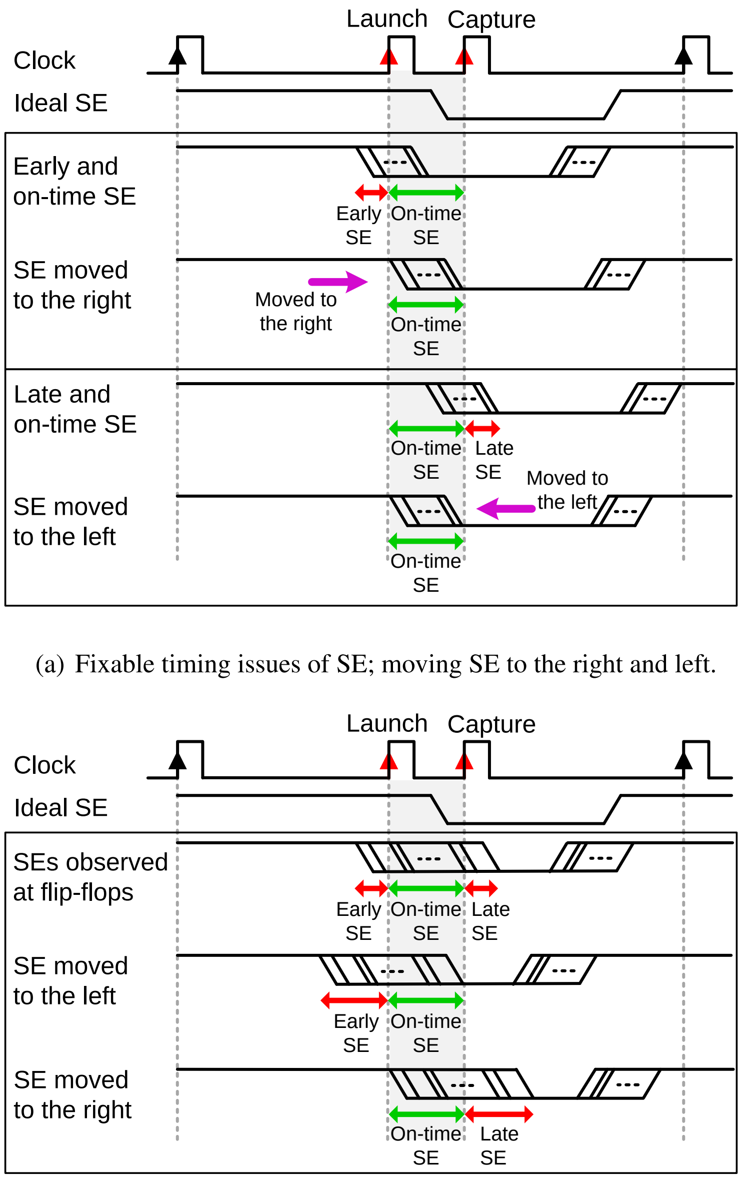
- **Moving SE to the left:** adjusting the late falling transition of SE such that it arrives earlier than its original arrival time and it falls within the at-speed clock pulse.

- **Moving SE to the right:** adjusting the early falling transition of SE such that it arrives later than its original arrival time and it falls within the at-speed clock pulse.

In order to find out if CVDs can help in calibrating the timing of SE, first, the skew of the SE signal within a cluster of scan cells should be inspected. Referring to Figure 5.7, if the skew of the SE signal within a cluster of scan cells is less than “one at-speed clock pulse”, in case of having early/late SE signal for some scan cells in that cluster, the timing of SE can be calibrated by exploiting CVDs and moving SE to the right/left. Further, if the range for falling edges of SE arriving to all flip-flops of the design is less than or equal to the “one at-speed clock pulse + maximum tuning range provided by CVDs”, then CVDs can fix all the timing issues of SE for all the clusters in the design; otherwise, the timing issue of SE cannot be fixed by CVDs, which means that the chip cannot operate correctly in the LOS test mode.

To summarize the above discussions, CVDs can be used for fixing the timing issues of the SE signal under two conditions: 1. The skew between the falling edges of the SE signal received by the scan cells of one cluster should not exceed “one at-speed clock pulse”; 2. The skew between the falling edges of SE observed by scan cells all over the chip should not be more than “one at-speed clock pulse + maximum tuning range of CVDs”.

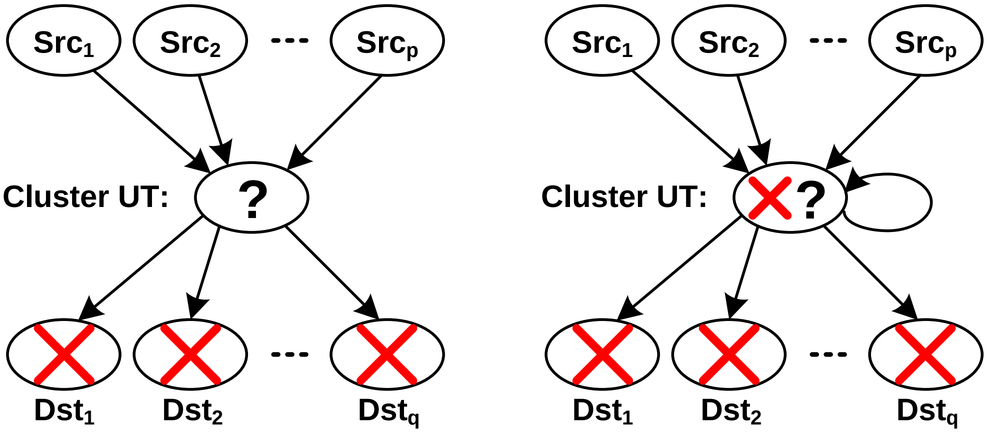
In this subsection, we studied the timing issues of SE that are fixable by exploiting CVDs. In the following subsection, we will explain how faulty SE signal at each cluster can be found. To detect the faulty SE signal, both *specialized test application strategy* and *specialized calibration patterns* are required. After the faulty SE signal is detected, our proposed CVD configuration mechanism for tuning the timing of SE will be explained.



(a) Fixable timing issues of SE; moving SE to the right and left.

(b) Non-fixable timing issues of the SE signal.

Figure 5.7: Applications of CVDs to fix the timing of the SE signal.



(a) Impact of early SE; no combinational path (self-loop) in the cluster UT.

(b) Impact of early SE; there is combinational path (self-loop) in the cluster UT.

Figure 5.8: Impact of early SE on the captured response of the cluster UT and its destination clusters.

5.3.2 Test Application Strategy

To establish a strategy for test application, initially, we have to study how timing issues of the SE signal (late/early SE) affect different clusters in a circuit.

Impact of Faulty SE on Different Clusters

The clusters generated through clustering algorithm are connected to each other through serial connections. Also, there are combinational logic paths between different scan cells from different clusters. These clusters and the combinational connections between them form a graph in which the clusters are the nodes. Also, edges in this graph represent combinational connections between scan cells of different clusters.

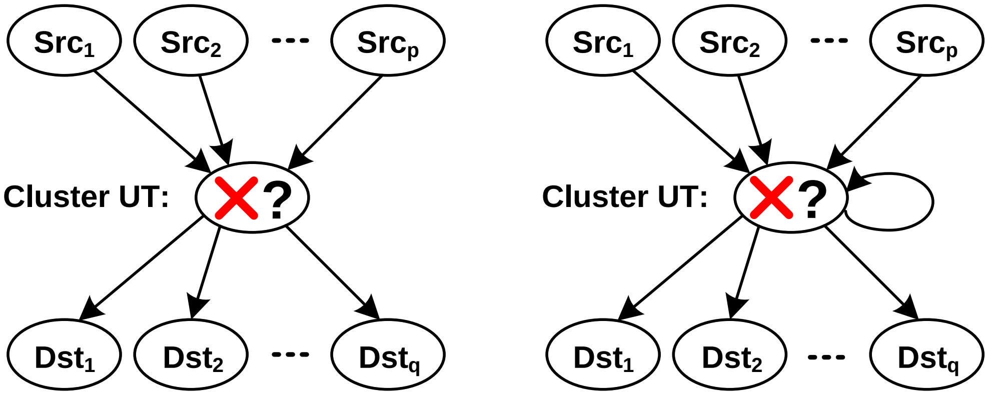
To study in detail how timing issues in the SE signal of a cluster impact other clusters of a circuit, different types of connections are studied in Figures 5.8 and 5.9; and the impact of early/late SE on each cluster is shown in the same figures with mark ‘X’.

As displayed in Figures 5.8 and 5.9, to place a cluster under test (UT), all its source clusters (Src_i), which provide test stimuli to the cluster under test, should be initialized. The cluster under test also provides test stimuli to its destination clusters (Dst_j).

In Figure 5.8(a), if there is no combinational path between the flip-flops of the cluster UT (no self-loops for the cluster UT); and, the cluster UT observes early SE, it will result in the incorrect captured response in its destination clusters. The reason is that in the cluster UT, early SE (as displayed in Figure 5.2) results in one shift less of test stimuli in cluster UT which means that the last shift of the scan-in will not be performed, and, instead of that, the cluster UT captures early response to its source clusters at the launch clock edge. This early captured response to the source clusters invalidates the value of the cluster UT at the launch clock edge and this incorrect value will be used as the test stimuli for the destination clusters. In the capture clock cycle, the cluster UT captures the correct response to the test stimuli provided by the source clusters; and the destination clusters capture wrong response to the incorrect test stimuli provided by the cluster UT (Figure 5.8(a)).

If there is a combinational path between the flip-flops of the cluster UT as shown in Figure 5.8(b) (self-loops on the cluster UT); and, the cluster UT observes early SE, the response captured at the cluster UT in the capture clock edge will be corrupted as well as the response captured at its destination clusters. The reason is that since there is a self-loop in the cluster UT, the cluster UT should also be initialized with the test stimuli and if SE is early for the cluster UT, the test stimuli at the launch clock edge will be overwritten by the early response captured by the cluster UT at the launch clock edge.

To study the impact of late SE, referring to Figure 5.9(a), if there is no self-loop on the cluster UT and it receives late SE, it will result in the incorrect captured response only in the same cluster. The reason is that flip-flops which observe late SE will not capture the



(a) Impact of late SE; no combinational path (self-loop) in the cluster UT.

(b) Impact of late SE; There is combinational path (self-loop) in the cluster UT.

Figure 5.9: Impact of late SE on the captured response of the cluster UT and its destination clusters.

response because they are in the shift mode. In fact, the value of their previous flip-flop in the scan chain will be shifted-in them at the capture clock edge. Therefore, the destination clusters of the cluster UT will not be affected. Similar reasoning can be applied for the clusters with self-loops when they observe late SE, as shown in Figure 5.9(b).

Based on the observations from Figure 5.8 and 5.9, late SE observed by a cluster UT with/without self-loop affects only the same cluster. However, early SE observed by a cluster invalidates the captured response observed by its destination clusters. Also, if the cluster UT has self-loops (Figure 5.8(b)), early SE signal affects the cluster UT as well.

To summarize, in order to detect late SE at a cluster, only the response of the cluster UT should be analyzed. To detect early SE at a cluster, the response of the cluster UT as well as its destination clusters should be analyzed by their associated RAs. It is important to note that if one or multiple destination clusters fail, the reason may not only be the early SE at the cluster UT; the reason may also be the early or late SE in any of the destination clusters because they are also capturing a response to the test stimuli provided by the cluster UT.

Diagnosis of the Faulty SE

As discussed above, detecting early SE requires that the response of more clusters to be analyzed (cluster UT and its destination clusters) when compared to the late detection of SE that only the response of the cluster UT should be analyzed. Furthermore, while a cluster is under test for early SE detection, if one or multiple failures are detected in the destination clusters of the cluster UT, it is not clear whether the source of the failures is the early SE observed in the cluster UT or early or late SE in the failing destination clusters.

Because of the extra on-chip circuitry and the extra time required for the detection of the early SE signal; and also, because early SE affects more clusters during the test application process for the detection of the early SE signal, we will focus on the detection of the late SE signal, which is more cost-effective. To cover the detection of all the faulty SE signals (both early and late), we will move the timing of the SE signal to the right such that no early SE signal exists. This is captured in Figure 5.10. As shown in this figure, when early, on-time, and late SE are observed by different flip-flops in the design, the timing of the SE signal should be moved to the right so that SE observed by flip-flops will only be on-time or late.

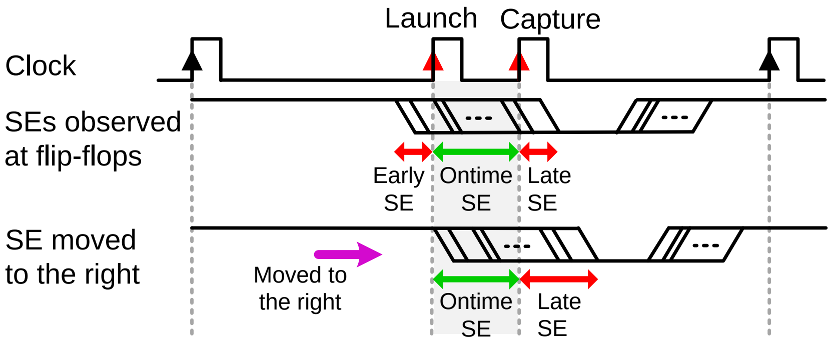


Figure 5.10: Removing all early SEs by moving SE to the right.

After moving the SE signal to the right, all the early SE signals observed by flip-flops will no longer exist and flip-flops will receive SE signal either on-time or late. At this point, to find the faulty SE signal, the clusters will only be tested for finding the late SE signals.

To determine the faulty SE signal observed by some clusters, initially we need to know in what order the clusters should be placed under test. Is it possible for all the clusters to be placed under test at the same time, or they can be under test one at a time? Another possibility is that selected group of clusters can be placed under test together.

Referring to Figure 5.9, it is known that to test whether the SE at the cluster UT is faulty or not, all the source clusters as well as the cluster UT should be initialized with the test stimuli (the cluster UT should be initialized if it has a self-loop and it provides initialization data for some bits of the cluster UT). It should be noted that not all the destination clusters can be under test at the same time because the initialization data loaded at the cluster UT may not be a valid initialization vector for all the destination clusters. As a consequence, the clusters that are connected through combinational paths will not be placed under test at the same time. An illustrative example showing which clusters are tested at the same time has been captured in Figure 5.11. In the graph displayed in this figure, the clusters that are not connected to each other can be tested at the same time. A group of clusters that are tested at the same time is called a test session. Hence, clusters *B* and *C* can be tested at the same time (test session *B* and *C*) while cluster *A* is not under test, and cluster *A* will be tested separately (test session A) while clusters *B* and *C* are not under test. The active clusters in each test session are shown in Figure 5.11.

For the large circuits with tens of clusters, the unconnected clusters to be tested at the same time cannot be found manually and an automated algorithm is required for finding the unconnected clusters. Finding the unconnected clusters is equivalent to the "graph

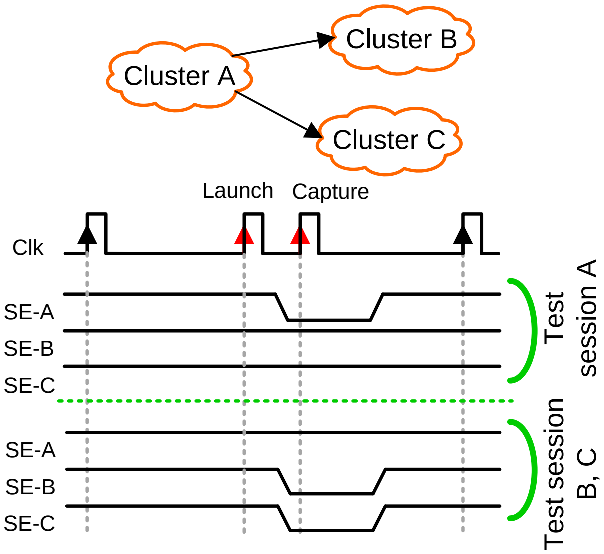


Figure 5.11: A sample of test application methodology.

coloring" problem [20] as elaborated in the following. Each cluster is equivalent to a node in the graph. If two nodes share the same color, it means that there is no connection between them and the adjacent nodes cannot share the same color. Nodes sharing the same color are equivalent to the clusters that will be placed under test in the same test session because there is no connection between them. Therefore, each color is mapped to a test session: i.e. (color 1, test session 1), (color 2, test session 2), and so on and so forth. To reduce the number of test sessions, the number of colors in the colored graph should be reduced. In order to reduce the number of colors, the number of nodes sharing the same color should be increased. This means that a maximum group of unconnected nodes should be found in the graph and all the nodes in this graph will share the same color and their equivalent clusters will be in the same test session. This process continues until all the nodes in the graph are colored (all the clusters are assigned to test sessions).

After providing the idea for test application, the next step for failure detection is to figure out what are the calibration patterns to sensitize failures caused by a faulty SE.

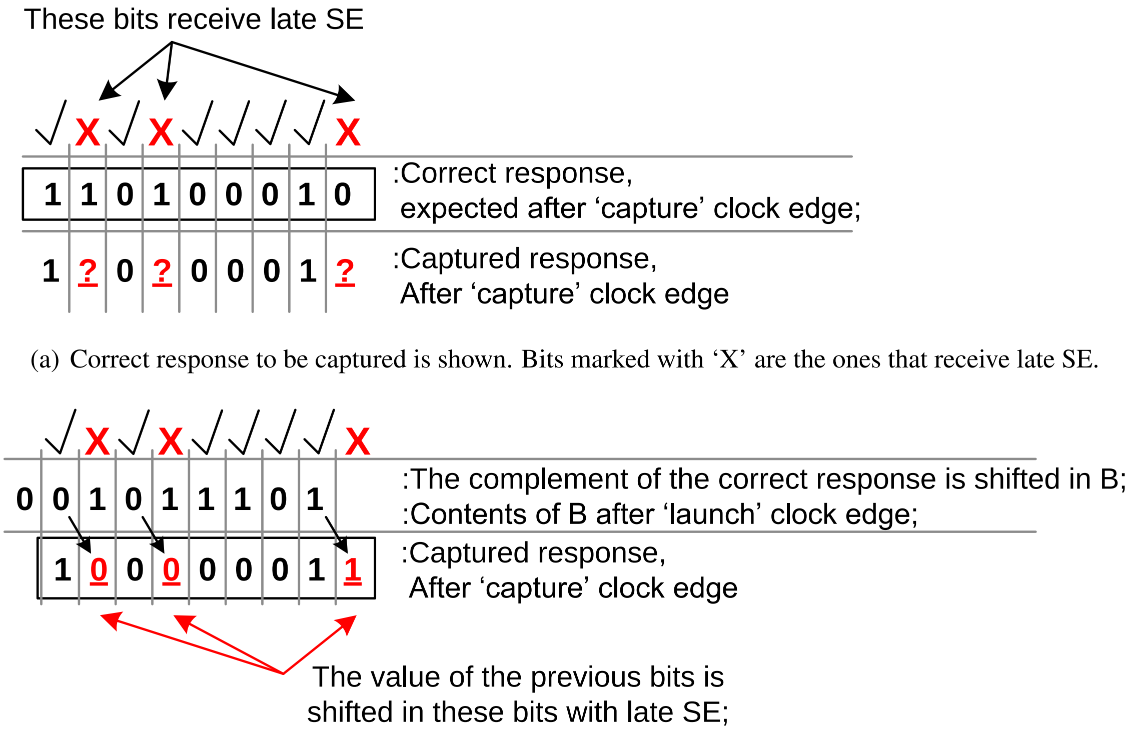
5.3.3 Diagnostic Calibration Patterns

To diagnose whether or not the SE signal observed at a cluster is late, calibration patterns should be applied to the clusters and the impact of the SE signal on the captured response will be studied to distinguish between the on-time and late SE.

In this subsection, we elaborate on the calibration patterns sensitizing a failure caused by late SE at a cluster. The exact calibration patterns will be discussed for the clusters without self-loops as displayed in Figure 5.9(a) when some of their flip-flops receive late SE. Then, the approach for finding the calibration patterns for clusters with self-loops (Figure 5.9(b)) will be explained.

- **Clusters without self-loops, to be tested for late SE:** To detect the potential late SE observed by some flip-flops of a cluster without a self-loop, the clusters that provide inputs to the cluster UT should be initialized with test stimuli and the cluster UT should capture the response. As explained before, flip-flops with late SE will not capture the response. They will shift-in the value of their previous flip-flop in the scan chain. Only the flip-flops with on-time SE will capture the response. To distinguish between the shifted value and the captured response, the cluster UT should be initialized with the complement of the correct response that is to be captured. This is explained in detail by the following example.

  Assume that cluster *B* in Figure 5.11 with 9 flip-flops is under test, and the correct response to be captured by it is ‘110100010’. If some of the flip-flops of this cluster receive late SE, they will not capture the response. They will rather receive the shifted value from their previous flip-flop. In Figure 5.12(a), assume bits marked with ‘X’ receive late SE. Therefore, in the ‘capture’ clock edge when the response is to be captured, the value of the ‘X’ bits is invalid: ‘?’. To differentiate between the invalid


These bits receive late SE
√ X √ X √ √ √ √ X
1 1 0 1 0 0 0 1 0
:Correct response,
expected after 'capture' clock edge;
1 ? 0 ? 0 0 0 1 ?
:Captured response,
After 'capture' clock edge
√ X √ X √ √ √ √ X
0 0 1 0 1 1 1 0 1
:The complement of the correct response is shifted in B;
:Contents of B after 'launch' clock edge;
1 0 0 0 0 0 0 1 1
:Captured response,
After 'capture' clock edge
The value of the previous bits is
shifted in these bits with late SE;

(a) Correct response to be captured is shown. Bits marked with 'X' are the ones that receive late SE.

(b) The captured response is shown. The bits underlined are the ones that are different from the correct response.

Figure 5.12: Finding flip-flops that receive late SE with only one calibration pattern.

value of a bit with late SE and the correct value of the same bit with on-time SE, the previous bit should be initialized with the complement of the correct response of the bit under test. Hence, if this bit does not capture the response due to late SE, the value of the previous flip-flop which is the complement of the correct response will be shifted into this bit. Similarly for clusters, if cluster *B* is initialized with the complement of the correct response, shifted to the left by one bit, after the launch clock edge as shown in Figure 5.12(b), the bits with on-time SE will capture the response after the capture clock edge, and the bits with late SE will shift-in the value of their prior bit, which is the complement of their correct value.

Based on the above explanations, flip-flops receiving late SE signal in clusters without self-loops can be diagnosed with only one calibration pattern. After diagnosing the failure, the CVD associated with this cluster will be tuned to calibrate the timing of SE for this cluster.

- **Clusters with self-loops, to be tested for late SE:** In these clusters, the flip-flops without combinational paths between them will be tested for late SE in the same way as the previous case; however, the late SE for all pairs of the flip-flops with combinational paths between them in a cluster is not guaranteed to be diagnosed with only one calibration pattern. In fact, each failure in the flip-flops with loops in a cluster is guaranteed to be detected if flip-flops with loops do not form shift registers; if they do, the response to be captured from the source of the combinational path in case of the on-time SE will be equal to the value of the prior bit shifted-in from the scan chain in case of the late SE. In this case, the complement of the response cannot be loaded in the previous bit.

  For clusters with self-loops, the maximum number of calibration patterns to detect a failure due to late SE will be equal to the number of flip-flops that are the destination flip-flops of self-loops.

In the above two subsections, we provided test application techniques to find faulty SE as well as the calibration patterns that will help find the clusters that observe late SE. After finding which clusters receive faulty SE signal, the on-chip CVD configuration flow to calibrate the timing issues of the SE signal should be performed, the details of which will be explained next.

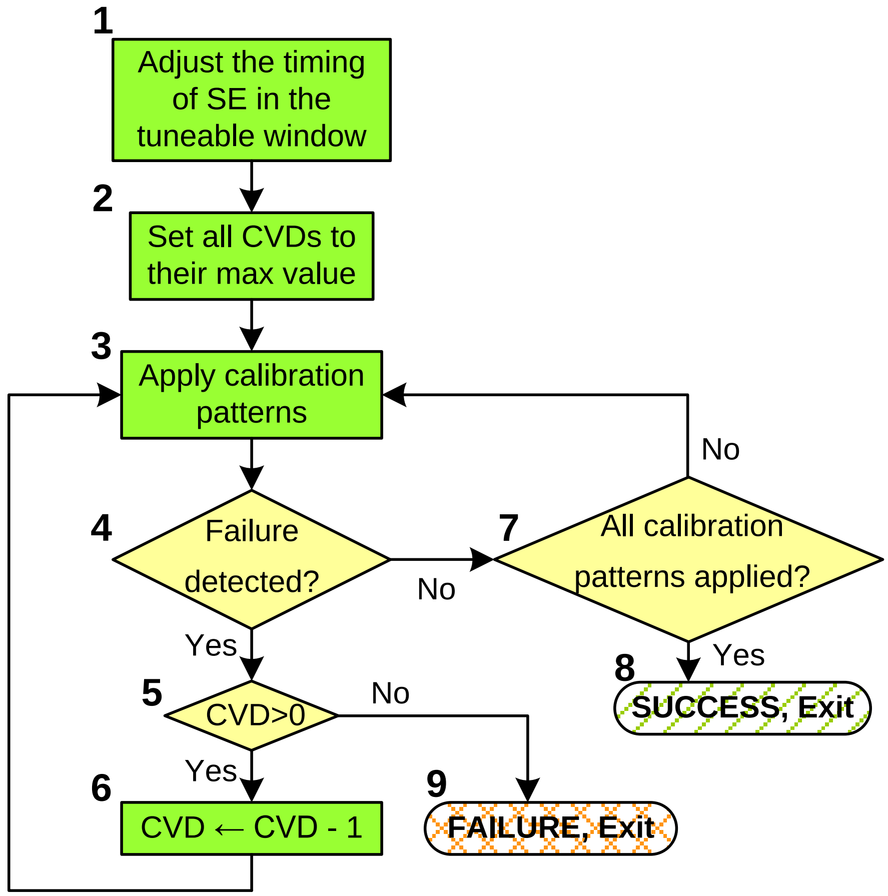

1
Adjust the timing of SE in the tuneable window
2
Set all CVDs to their max value
3
Apply calibration patterns
4
Failure detected?
No
7
All calibration patterns applied?
No
Yes
8
SUCCESS, Exit
Yes
5
CVD>0
No
Yes
6
CVD ← CVD - 1
9
FAILURE, Exit

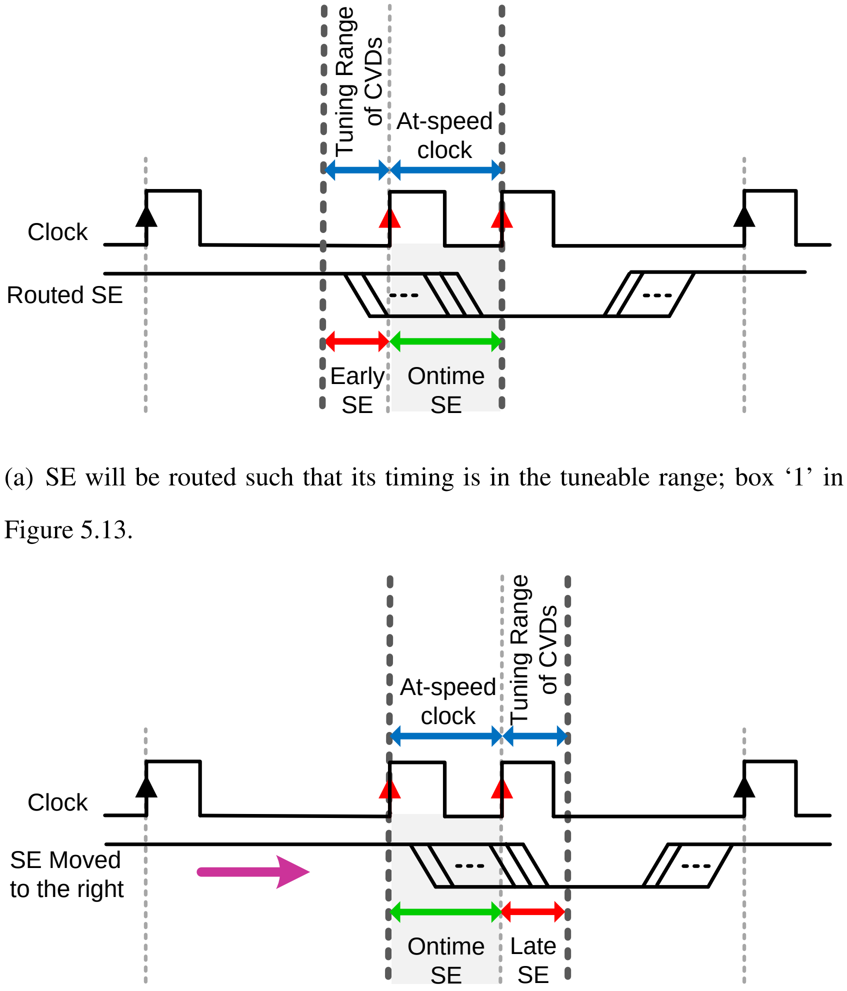
Figure 5.13: CVD configuration flow to adjust the timing of SE.

5.4 CVD Configuration Flow

By applying calibration patterns to the clusters under test, as soon as a mismatch is detected between the signature of the captured response (SCR) and the signature of the reference response (SRR) of a cluster, self adjustment circuitry for the CVD corresponding to the cluster should be activated. As explained in the previous section, the timing of the SE signal for all the clusters will be moved to the right to remove early SE signals observed by some clusters. Then, the SE signal received by clusters will either be on-time or late and from this point, detection of the late SE signals starts. After late SE signals are detected, the CVD associated with the clusters receiving late SE should be adjusted to tune the timing of the SE signal for that cluster. However, it should be noted that CVDs, based on their definition, only add delays and they do not subtract delays. To be able to subtract delay by the use of CVDs, the following flow is proposed in Figure 5.13.

As elaborated in subsection 5.3.1, the tuneable range of the faulty SE signal in a circuit is “one at-speed clock pulse + maximum tuning range of CVDs” shown in Figure 5.14(a). To satisfy this range for SE, the routing parameters for SE can be adjusted in such a way that the timing slack for SE in the design lies in the above mentioned range (Figure 5.13, box ‘1’). If the timing variation is more than the specified range, the SE will not be tuneable and the chip will fail due to the high timing variation of the SE signal.

After performing the first step in the CVD configuration flow (Figure 5.13, box ‘1’) by arranging the timing variation for the SE signal in the range provided in Figure 5.14(a), the clusters receiving early SE signal should be detected. However, as explained in detail in subsection 5.3.2, the timing of the SE signal will be moved to the right such that we deal only with late SE signal. This step is performed in box ‘2’ in Figure 5.13 by setting the value of all the CVDs to the maximum and the timing for SE signal will be moved to the right as displayed in Figure 5.14(b).



(a) SE will be routed such that its timing is in the tuneable range; box ‘1’ in Figure 5.13.

(b) The timing of SE will be delayed (moved to the right) by setting all CVDs to the maximum; box ‘2’ in Figure 5.13.

Figure 5.14: Timing of the SE signal.

After providing the maximum configuration value for all the CVDs, calibration patterns should be applied (as explained in subsection 5.3.3) to find the clusters that receive late SE (box ‘3’ in Figure 5.13). If a failure is detected in the captured response of a cluster (decision box ‘4’ in Figure 5.13), it means that the cluster is receiving late SE. To tune the timing for the late SE signals, the value of the CVD associated to the failing cluster should be decreased (box ‘6’ in Figure 5.13) if it has not already been decreased to the minimum CVD value (decision box ‘5’ in Figure 5.13). Otherwise, when a failure is detected in a cluster and the value of the corresponding CVD is ‘zero’, the configuration value of the CVD cannot be reduced anymore to compensate for the late SE and the flow reports that the timing of the SE signal cannot be tuned in box ‘9’ in Figure 5.13.

On another line of reasoning, if no failure was detected by the applied calibration pattern, the next calibration pattern should be applied to the circuit and based on the response (faulty or correct) and the CVD configuration value, either no action is required, or a CVD will be tuned. This iterative process continues until no more failures are detected in any cluster in the circuit and CVDs have been able to tune the timing of the SE signal when we report ‘SUCCESS’ in the flow, or when there are some failures that cannot be fixed with the available tuning range of CVDs and we report ‘FAILURE’.

After explaining the CVD configuration flow, in the following, we explain how the knowledge about test sessions explained in 5.3.2 can help in reducing the area assigned for RAs in the design stage before getting to the post-fabrication tasks discussed in section 5.3.

5.5 Reducing On-Chip Area Used for Faulty SE Detection

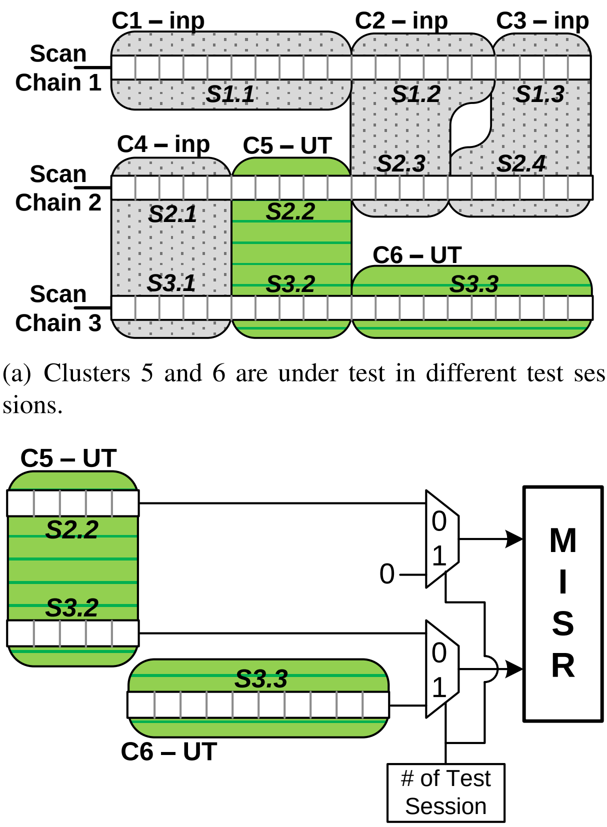
The area investment to implement the proposed method is performed at the design stage which includes the additional hardware required for inserting RAs as well as the additional hardware required for the control logic to enable or disable the clusters that are active or inactive in each test session. In the following subsection, we explain how the area allocated for RAs that are inserted per cluster (subsection 5.2.2) can be reduced. Then, the on-chip hardware required to enable or disable each test session (Figure 5.11) will be elaborated.

5.5.1 Improving the Area Allocated for RAs

As discussed in subsection 5.2.2, after the response to each calibration pattern is captured, it should be analyzed with an RA to determine whether a failure exists in the captured response. The simplest assignment of RAs to clusters is to assign one RA to each cluster, which results in overall $\frac{n}{m} \times (\lceil log_2(m) \rceil)$ extra flip-flops while n stands for the number of flip-flops of the circuit and m is the cluster size. Also, $(\lceil log_2(m) \rceil)$ is the RA size per cluster.

To reduce the additional area required for inserting RAs, multiple clusters can share RAs. In fact, the clusters that are not in the same test session and will not be tested at the same time can share RAs both in time and hardware. It should be noted that when multiple clusters share one RA, a multiplexer should be provided for each input of the RA (either in the SISR mode or in the MISR mode) to select between the values that are shifted-in the shared RA in each test session. Determining the number of clusters sharing the same RA depends on the following factors: 1. The number of clusters that are not under test in the same test session. 2. The maximum allowable multiplexer size at each input of the RA; for example, the maximum allowable size for multiplexer may be '4×1' or '8×1'. An example of sharing RAs between multiple clusters is displayed in Figure 5.15.

In this example, there are 3 scan chains and 6 clusters in total. Clusters are generated from one or two scan segments (scan segment is a division of scan chain). For example, cluster 5 is made of scan segments '$S2.2$' and '$S3.2$' which are segment 2 from chain 2, and segment 2 from chain 3 respectively. Cluster 6 also contains '$S3.3$' which is scan segment 3 from scan chain 3 (Figure 5.15(a)). Assuming that clusters 5 and 6 will be under test in two different test sessions, they can share an RA. As shown in Figure 5.15(b), in test session 0, contents of cluster 5 will be shifted-in the RA and the RA will be used in the MISR mode



(a) Clusters 5 and 6 are under test in different test sessions.

(b) Contents of cluster 5 is shifted-in MISR in test session 0, and cluster 6 is shifted-out in test session 1.

Figure 5.15: One RA shared between multiple clusters.

because cluster 5 has two scan segments and they will be shifted-in the two inputs of the RA. In test session 1, the contents of cluster 6 will be shifted-in the RA, and the RA will be used in the SISR mode for this test session. According to this example, multiple clusters in different test sessions can share the same RA so long as the size of the multiplexers at the inputs of RA can support that number of clusters. In section 5.6 the experimental results will show how the number of RAs can be reduced by multiple clusters sharing the same RA. Furthermore, the multiplexer count will be discussed.

In the following, we will discuss the extra hardware required to enable or disable the clusters that are active or inactive in each test session.

5.5.2 Area Investment To Enable or Disable A Test Session

As explained earlier in this chapter, calibration patterns are applied to the circuit through test sessions in order to find out if a late SE signal is observed by the clusters under test. To place a group of clusters under test, their source clusters should be initialized with the test stimuli. All the source clusters as well as the ones under test should be enabled in each test session and the rest of the clusters should be disabled. To enable or disable a cluster, the clock signal going to each of these clusters can be gated with the enable signal that is activated according to the test session number. The circuitry to enable or disable a test session is displayed in the example shown in Figure 5.16 that shows the area investment for the circuit shown in Figure 5.15(a).

As displayed in this Figure, an 'AND' gate is required for each cluster to enable/disable it in each test session. It is important to note that the slack in the timing of enable signal will not cause any problem because the enable signal will not need to change on a clock cycle by clock cycle basis. It actually switches while going from one test session to another.

Based on the above explanations, the total number of 'AND' gates for each circuit will be equal to the number of clusters. To calculate the required control logic to derive the enable signals for activating clusters in each test session, it should be noted which clusters, and accordingly which enable signals should be active in each test session. Hence, using a counter that counts up to the number of test sessions, we can generate all the test session numbers in the circuit. The test session number that is the output of this counter can be connected to a decoder and the enable signals for each test session can be connected to

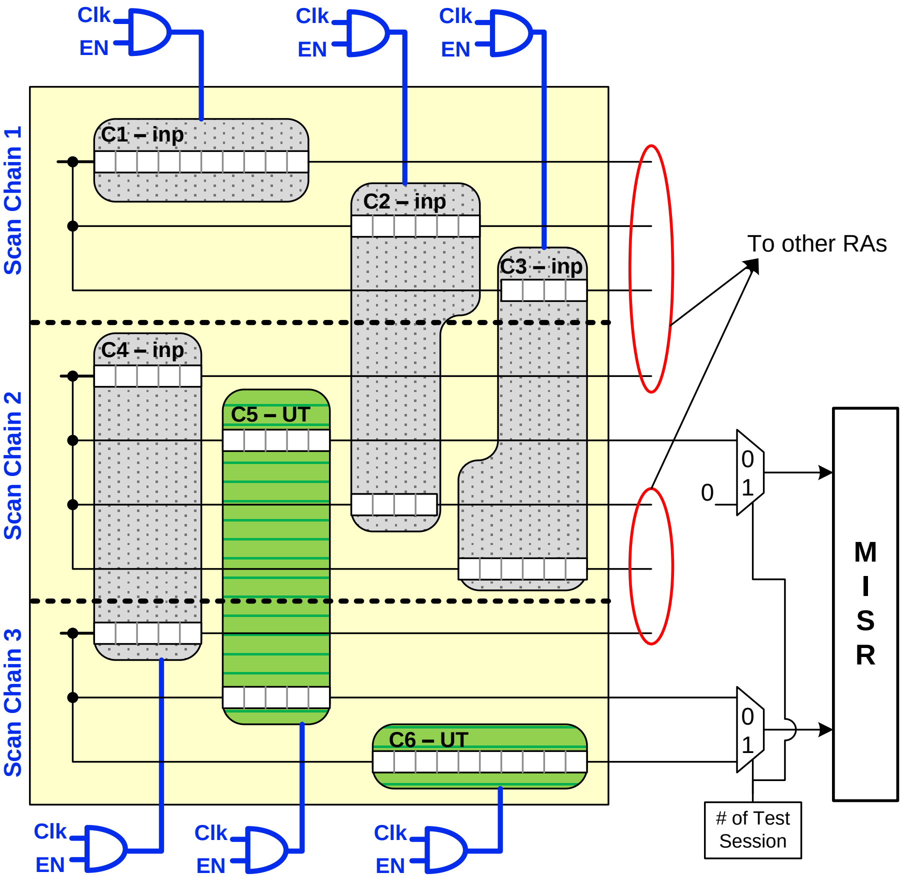


Figure 5.16: Circuitry to enable/disable test sessions.

the corresponding output of the decoder. For example, if the total number of test sessions is ‘5’, we need a counter to count from ‘0’ up to ’4’. The output of the counter will also be connected to a ‘3 × 8’ decoder to generate all five test session numbers. Assuming that we want to activate the clusters related to test session 2, the enable signals for all clusters active in test session 2 should be driven from output ‘00000100’ of the decoder. Also, the select signals for the multiplexers connected to the inputs of RAs will be driven from the output of the counter and the decoder.

In the following section, the experimental results for the test time as well as the area investment for each circuit will be discussed.

5.6 Results

To assess the proposed approach, we have synthesized ISCAS89 benchmark circuits [13] and inserted scan circuitry using a 90nm TSMC [66] standard cell library. Two categories of experiments have been run on the ISCAS89 benchmark circuits. The first category evaluates the time for applying the calibration patterns (uploading the test stimuli and the SRR, and offloading the response and the SRR) to the three largest ISCAS89 circuits. The second category of experiments evaluates the total area investment of the proposed method for each of the benchmark circuits.

In the following, initially we discuss the calibration time for the benchmark circuits and then, the additional circuitry to implement the proposed method will be elaborated.

5.6.1 Calibration Time

To measure the calibration time for the three largest ISCAS89 circuits: s38417, s38584, and s35932, we ran experiments with 1, 2, and 4 scan chains and cluster sizes from 60 up to 150 for each circuit. Each experiment shows the number of test sessions and the average time (number of clock cycles) to apply each calibration pattern (upload and offload time) for different cluster sizes and different number of scan chains. Tables 5.1, 5.2, and 5.3 summarize the time for applying calibration patterns to circuits s38417, s38584, and s35932.

To measure the number of clock cycles for applying each calibration pattern, for the clusters without self-loops for which only one calibration pattern is applied, the test time will be $T_{upload} + T_{offload} + 1$ (an extra clock cycle is needed for capture). However, it should be noted that the clusters without self-loops are less than 5% of all the clusters in each benchmark circuit. Therefore, most of the clusters to be tested for late SE are the ones with self-loops and the number of calibration patterns to detect the late SE for them is usually more than one. It is a known fact that when multiple test vectors are applied to a unit under test, $T_{offload}$ of test vector i overlaps with T_{upload} for test vector $i+1$. Hence, the test application time for each vector will be reduced to Max$\{T_{upload}, T_{offload}\}$ + 1. It should be noted nonetheless that in our work we do not need to shift the entire captured response out of the circuit. In fact, we only need to offload the captured response of each cluster and the SRR to the RA.

Table 5.1: Test application time for circuit s38417.

Circuit s38417	**No. of Chains: 1 Max. Chain length: 1564**			**No. of Chains: 2 Max. Chain length: 782**			**No. of Chains: 4 Max. Chain length: 391**		
Cluster Size	No. of Clusters	No. of Test Sessions	Avg Clock Cycle per Calibration Pattern	No. of Clusters	No. of Test Sessions	Avg Clock Cycle per Calibration Pattern	No. of Clusters	No. of Test Sessions	Avg Clock Cycle per Calibration Pattern
60	27	8	874	27	10	541	27	9	323
70	23	7	952	23	9	521	24	9	314
80	20	6	982	20	7	533	24	8	325
90	18	7	904	18	7	587	18	9	312
100	16	6	934	16	7	556	17	7	332
110	15	7	846	15	7	545	15	7	335
120	14	7	1007	14	8	594	16	8	332
130	13	8	905	13	9	567	15	8	339
140	12	7	912	12	8	646	13	7	357
150	11	7	868	11	7	613	12	8	372

Table 5.2: Test application time for circuit s38584.

Circuit s38584	**No. of Chains: 1 Max. Chain length: 1449**			**No. of Chains: 2 Max. Chain length: 725**			**No. of Chains: 4 Max. Chain length: 363**		
Cluster Size	No. of Clusters	No. of Test Sessions	Avg Clock Cycle per Calibration Pattern	No. of Clusters	No. of Test Sessions	Avg Clock Cycle per Calibration Pattern	No. of Clusters	No. of Test Sessions	Avg Clock Cycle per Calibration Pattern
60	25	13	998	25	13	629	27	15	346
70	21	12	1061	21	13	648	22	13	341
80	19	12	1063	19	14	651	19	12	331
90	17	11	1090	17	10	624	17	14	353
100	15	10	1089	15	12	652	15	14	359
110	14	11	1128	14	12	637	14	11	354
120	13	9	1129	13	10	634	13	12	357
130	12	10	1191	12	9	675	14	10	348
140	11	8	1112	11	11	683	13	10	362
150	10	8	1123	10	9	652	10	9	360

Table 5.3: Test application time for circuit s35932.

Circuit s35932	**No. of Chains: 1 Max. Chain length: 1728**			**No. of Chains: 2 Max. Chain length: 864**			**No. of Chains: 4 Max. Chain length: 432**		
Cluster Size	No. of Clusters	No. of Test Sessions	Avg Clock Cycle per Calibration Pattern	No. of Clusters	No. of Test Sessions	Avg Clock Cycle per Calibration Pattern	No. of Clusters	No. of Test Sessions	Avg Clock Cycle per Calibration Pattern
60	29	5	856	29	8	445	33	8	340
70	25	5	1008	25	7	513	28	7	319
80	22	4	829	22	5	526	22	5	346
90	20	4	950	20	5	614	20	6	315
100	18	4	1001	18	6	518	20	6	349
110	16	4	1012	16	7	477	17	6	321
120	15	4	891	15	6	574	15	6	363
130	14	3	1069	14	7	585	14	6	322
140	13	3	1053	13	3	633	13	6	358
150	13	4	882	12	5	549	13	5	339

In Tables 5.1, 5.2 and 5.3, ‘No. of Clusters’ shows the number of clusters and their corresponding CVDs with respect to the ‘Cluster size’. ‘No of Test Sessions’ shows how many test sessions are required to search for all the potential timing issues on the SE signal in all the clusters and each test session consists of the clusters that can be placed under test at the same time. ‘Avg Time per Calibration Pattern’ provides the average time required to apply a calibration pattern which is Max$\{T_{upload}, T_{offload}\}$ + 1.

As shown in Tables 5.1, 5.2 and 5.3, by increasing the cluster size, the average time to apply a calibration pattern either increases or remains in the same range. By decreasing the number of clusters, no savings can be achieved in terms of the average time for applying a calibration pattern. Similar observation can be made for different number of scan chains. Doubling the number of scan chains is expected to reduce by half the average time required per test vector in standard scan testing; however, in test application to a clustered circuit for SE calibration purpose, the average time required per calibration pattern will not reduce by half when the number of scan chains is doubled (or the length of scan chains is divided in half). The reason is that each test session requires different clusters to be activated for shifting-in the test stimuli and shifting-out the test responses (as per the test application described in subsection 5.3.2).

For the same reason as above even for a single scan chain, the average number of clock cycles per pattern is smaller than the length of the entire scan chain. Therefore, knowing the fact that having multiple scan chains results in complicating the control for test application for calibrating the SE signal, thus leading to additional on-chip hardware, the on-chip area needed to facilitate multiple scan chain tests may be better utilized for more CVDs. This will result in smaller clusters and hence increase the granularity for fine-tuning the SE signal.

In the following, additional circuitry required to implement the proposed method will be elaborated.

5.6.2 Area Investment

As explained in section 5.5, the area investment to implement the proposed method consists of the additional hardware for RAs, and the additional hardware required to enable or disable the clusters in each test session.

Table 5.4: Area investment for circuit s38417, Number of flip-flops: 1636.

		No. of Chains: 1 **Max. Chain length: 1564**			**No. of Chains: 2** **Max. Chain length: 782**			**No. of Chains: 4** **Max. Chain length: 391**		
Cluster Size	RA Size	No. of Clusters/ AND Gates	No. of RAs	MUX 2X1	No. of Clusters/ AND Gates	No. of RAs	MUX 2X1	No. of Clusters/ AND Gates	No. of RAs	MUX 2X1
60	6	27	9	36	27	6	42	27	7	42
70	7	23	8	30	23	8	30	24	6	28
80	7	20	7	28	20	6	28	24	5	32
90	7	18	7	20	18	5	26	18	5	28
100	7	16	6	20	16	4	24	17	4	26
110	7	15	6	18	15	4	22	15	5	22
120	7	14	5	20	14	4	20	16	5	24
130	8	13	4	18	13	5	16	15	3	20
140	8	12	4	18	12	4	16	12	3	18
150	8	11	4	14	11	3	16	12	2	18

Tables 5.4, 5.5, and 5.6 show how many RAs and multiplexers are required in the three large ISCAS89 circuits if the multiplexer size used for the inputs of RAs is ‘2 × 1’. We have also assumed that multiplexers can be stacked at most in four levels. For example, four levels of ‘2 × 1’ multiplexers are stacked on top of each other and they are connected to one of the inputs of an RA. It should be noted that offloading the captured response to the RAs is done with the ‘shift clock’ which is known to be slower than the at-speed clock. Hence, stacking four levels of ‘2 × 1’ multiplexers will not cause timing problems.

Table 5.5: Area investment for circuit s38584, Number of flip-flops: 1452.

		No. of Chains: 1 **Max. Chain length: 1449**			**No. of Chains: 2** **Max. Chain length: 725**			**No. of Chains: 4** **Max. Chain length: 363**		
Cluster Size	RA Size	No. of Clusters/ AND Gates	No. of RAs	MUX 2X1	No. of Clusters/ AND Gates	No. of RAs	MUX 2X1	No. of Clusters/ AND Gates	No. of RAs	MUX 2X1
60	6	25	6	38	25	4	42	27	6	40
70	7	21	5	38	21	4	34	22	5	28
80	7	19	5	36	19	3	32	19	5	28
90	7	17	5	30	17	3	28	17	3	28
100	7	15	5	20	15	3	24	15	4	26
110	7	14	3	22	14	3	22	14	4	20
120	7	13	4	18	13	3	20	13	2	22
130	8	12	3	18	12	3	18	14	2	20
140	8	11	4	14	11	3	16	13	3	12
150	8	10	3	14	10	2	16	10	2	20

An important remark that should be noted is that the area allocated for inserting RAs in the circuit can be further utilized for purposes in testing other than SE calibration, which are beyond the scope of these thesis.

In each table, we have collected results for cluster sizes from 60 up to 150, and for 1, 2, and 4 scan chains inserted for the benchmark circuits. The column "RA size" shows the size of each RA. As mentioned in subsection 5.2.2, we have assumed that for cluster size m the RA size is $\lceil log_2(m) \rceil$. "No. of Clusters/ AND Gates" shows how many clusters and 'AND' gates exist in the circuit for the specified cluster size. "No. of RAs" shows how many RAs are required for the circuit if different clusters from different test sessions can share RAs; and, "MUX 2X1" shows the total number of multiplexers of size '2×1'.

Table 5.6: Area investment for circuit s35932, Number of flip-flops: 1728.

		No. of Chains: 1 **Max. Chain length: 1728**			**No. of Chains: 2** **Max. Chain length: 864**			**No. of Chains: 4** **Max. Chain length: 432**		
Cluster Size	RA Size	No. of Clusters/ AND Gates	No. of RAs	MUX 2X1	No. of Clusters/ AND Gates	No. of RAs	MUX 2X1	No. of Clusters/ AND Gates	No. of RAs	MUX 2X1
60	6	29	11	34	29	11	36	33	13	38
70	7	25	9	32	25	9	32	28	10	34
80	7	22	8	28	22	8	28	22	7	30
90	7	20	7	26	20	6	28	20	7	26
100	7	18	6	24	18	6	24	20	5	30
110	7	16	5	22	16	5	22	17	5	24
120	7	15	6	20	15	5	20	15	5	22
130	8	14	5	18	14	5	18	14	4	22
140	8	13	5	16	13	5	18	13	4	18
150	8	13	5	20	12	4	16	13	5	16

5.7 Summary

In this chapter of the thesis, we have presented an on-chip tuning mechanism for the SE signal to compensate for the process variations in the fabrication process. While the current practice relies on pipelining the SE or local SE generation, we have introduced an alternative approach based on delay tuning elements commonly used in the clock trees of high-performance designs [47]. The proposed approach requires a set of calibration patterns to be applied before the LOS test set. As a result of this calibration process, each circuit sample will automatically configure its tuning elements placed in the SE tree. The proposed approach is complementary to the known art that can assist with at-speed LOS testing.

Chapter 6

Conclusion

Imperfect pre-silicon timing estimation tools have increased the distance between the pre-silicon timing estimates and the post-silicon real timing values, and programmable delay tuning elements are employed at post-silicon to reduce this gap. In this thesis, we have exploited the post-silicon tuning ability introduced by programmable delay tuning elements to improve a group of features in digital ICs. These features that have been improved through the work presented in this thesis include performance, testing, and reliability.

6.1 Summary of the Works Presented in this Thesis

In chapter 3, in order to provide the facilities for improving the performance of a digital IC after fabrication based on real post-silicon delay values, CVDs are inserted in the digital IC during the design stage. The main targets during the insertion process are the slow paths for which the inaccuracy in their timing estimation can lead to performance reduction at post-silicon. Before tuning the timing of these slow paths, their post-silicon delay value should be measured through the frequency stepping method.

Timing adjustment of a slow path is performed by the use of CVDs providing different clock arrival times to the source and the destination flip-flops of the slow path and delaying the clock arrival time to the destination flip-flop. However, there might be a fast path that has a common destination flip-flop with the above-mentioned slow path and delaying the timing of the clock for their common destination flip-flop may result in hold-time violation for the fast path. In order to avoid this, the fast paths of a circuit with their real delay values should be known. Hence, in our work presented in chapter 3, before we perform CVD configuration for performance maximization, we find the real delay value of fast paths. This task is carried out by a CVD configuration methodology customized to find the fast paths' real delay values. After this step is done, another CVD configuration algorithm is run to maximize the performance for the first silicon samples by the use of post-silicon fast and slow paths delay information.

An interesting observation is that, since the tuning range of CVDs is bounded, only a limited group of slow paths will affect the performance and a limited group of fast paths may fail due to hold-time violation. Knowing the fact that our proposed method generates CVD configurations based on a branch and bound search engine, using the above mentioned feature of CVDs and working on limited groups of fast and slow paths helps prune the size of the solution space effectively resulting in affordable run-times. As the final remark for the work presented in chapter 3, we have proposed a scalable method that can find the maximum performance of a circuit after fabrication with regards to the accuracy of tester measurements and the granularity of delay values added by CVDs.

On another line of thought, CVDs can be employed to improve the reliability of a circuit during its lifetime. In the work presented in chapter 4 of this thesis, we have introduced a new perspective on managing the impact of circuit aging as one of the main sources for reliability degradation. More specifically, we have targeted reliability improvement by the use of programmable delay tuning elements.

Sensor flip-flops to predict delay degradation on slow paths are inserted during the design stage. If a sensor flip-flop connected to a path is sensitized at run-time, it means that it has predicted a setup-time violation and the timing for the corresponding slow path should be adjusted. However, adjusting the timing through CVDs may result in hold-time violations for fast paths that have common destination flip-flops with slow paths. To avoid hold-time violation, we have designed a circuit structure that predicts the hold-time violation for fast paths during the normal operation of the circuit. If a setup-time violation for a slow path is predicted, and, if by the use of CVDs to adjust the timing of slow path no hold-time violation is predicted, we will be able to preserve the maximum operational performance of a circuit rather than lowering its performance. It should be noted that this process takes place within each chip during its lifetime, resulting in a unique CVD configuration for each circuit sample based on its unique delay degradation pattern.

After presenting new techniques for performance and reliability improvement, we have also investigated an alternative approach in chapter 5 for tuning the DFT logic for at-speed testing of VLSI circuits. The timing of SE signal in LOS testing may need to be calibrated to transition at-speed to compensate for the process variations induced by the manufacturing process. In order to provide the post-silicon tuning ability for the SE signal, CVDs may be used. In our approach, each individual CUT is subjected to a set of calibration patterns that can find the SE timing issues specific to the particular circuit. It also results in finding the unique CVD configuration generated for the SE tree of each circuit sample. The proposed calibration process is performed before the LOS testing to tune the timing of SE before it is used during LOS testing.

6.2 Suggestions for Future Works

In this thesis, we have been able to compensate for the lack of precise delay information at pre-silicon by using programmable delay tuning elements at post-silicon, which may result in improvement in performance, reliability, and testing of digital circuits. Although the works presented in this thesis are novel, there are further opportunities for improvement. Hence, the work presented in this thesis can be expanded and enriched by either improving the implementation, or by exploring new avenues of research in this field, examples of which are given next.

For the work presented in chapter 3 on performance improvement after fabrication, as explained, two separate CVD configuration algorithms are run for both fast paths delay measurement and performance maximization. Based on details provided in the corresponding chapter, for each of the CVD configuration algorithms, subgraphs of a full state graph related to a circuit are processed separately, one at a time, to find their CVD configuration values. However, to implement the proposed method for circuits with thousands of gates, processing subgraphs on a processing unit (such as a computer with one processor) one at a time can be time consuming.

In order to reduce the processing time for large circuits, the CVD configuration for multiple subgraphs can be distributed across multiple processing cores. Hence, the CVD configuration for multiple subgraphs can be found concurrently. The study in this field can be continued by calculating the dependencies between the number of subgraphs that can be processed at the same time to the number of processing cores. Also, there might be a need for an organizer in the case when the number of processing cores is smaller than the number of subgraphs. The organizer can select subgraphs that are assigned to each processing core in a way that the overall run-time is minimized.

CVDs may also be employed to improve the performance for low-power circuits with gated-clocks. In circuits with gated-clocks, the entire design comprises multiple partitions that are clock gated and are active or inactive in different operational modes. For example, if a low-power circuit has three partitions called A, B, and C, then partitions A and B are active in mode 1, and partitions B and C are active in mode 2 during circuit's operation. In order to improve the performance for a gated-clock low-power circuit, instead of finding the maximum overall performance of the circuit (a mode of operation in which all gated-clock partitions are active), we can find the maximum performance for each mode separately. Hence, a CVD configuration should be found for each mode, and when the circuit is switching between modes, the CVD configuration corresponding to the new mode should be loaded to CVDs in order to maximize the performance for the active mode.

Our second contribution elaborated in chapter 4 was focused on reliability improvement for digital circuits. Based on that chapter, after setup-time violations due to delay degradation are predicted, our proposed hold-time violation prediction circuit structures will be activated to predict the potential hold-time violations that may occur if CVDs are to adjust the timing of slow paths. Then, if no hold-time violation is about to occur, CVDs will tune the timing of degraded paths to preserve the maximum performance. Otherwise, techniques such as frequency adjustment should step-in to preserve the correct circuit behaviour.

Due to the slow aging process, there is no need to keep all setup-time violation prediction sensors active during the lifetime of the circuit. Hence, a monitor signal can be provided in the circuit to enable and disable the operation of setup-time violation prediction sensors. Multiple issues regarding the so-called monitor signal can be investigated, such as its routing strategy, its power consumption, and its electrical effects on the circuit (e.g., the impact of its over-shoots and under-shoots when enabling or disabling sensors).

Also, its activity and inactivity periods should be determined as a function of time and the aging model. For instance, delay degradation due to aging is more intense at the beginning of the lifetime of a circuit and it becomes less intense over time. This implies that the active durations of the monitor signal at the beginning of its operation may need to be larger than during its later phases. Another avenue to improve this work would be to study the impact of aging for the clock signal, as well as aging for both setup-time and hold-time violation prediction sensors, and how would they impact in-system CVD configuration.

Our contribution presented in chapter 5 has explored an alternative approach for improving at-speed LOS testing of scan circuits. In particular, we have investigated how CVDs can be employed to resolve timing problems for the SE signal. Since each circuit sample may have different timing issues for its SE signal, a unique CVD configuration to tune the timing of the SE specific to the CUT at hand must be determined. Our work can be extended to support a specialized test generation framework that minimizes the number of calibration patterns, which are conscious of the circuit-specific scan chain configurations. In addition, since the area investment may become prohibitive if many scan segments are employed, further studies are needed to develop sharing mechanisms for the response analyzers between the calibration phase and stuck-at and at-speed testing phases.

As a concluding remark, in this thesis we have presented several automated techniques that leverage the presence of programmable delay tuning elements for improving performance, reliability and test of digital ICs. It is anticipated that such tuning elements will become widespread in the future generations of VLSI circuits in order to tackle process variations at runtime. Therefore, the contributions presented in this thesis are a step forward toward helping the design community to better understand how to use automated techniques that will be needed to handle the large number of tuning elements.

Bibliography

[1] M. Abramovici, M. A. Breuer, and A. D. Friedman. *Digital Systems Testing and Testable Design*. Wiley-IEEE Press Publications, 1994.

[2] M. Abramovici, P. R. Menon, and D. T. Miller. Critical Path Tracing - An Alternative to Fault Simulation. *Proceedings of the IEEE/ACM Design Automation Conference*, pages 468 – 474, 1988.

[3] M. Agarwal, V. Balakrishnan, A. Bhuyan, K. Kyunglok, B. C. Paul, W. Wang, B. YangY. Cao, and S. Mitra. Optimized Circuit Failure Prediction for Aging: Practicality and Promise. *Proceedings of the IEEE International Test Conference*, pages 1 – 10, October 2008.

[4] M. Agarwal, B. C. Paul, M. Zhang, and S. Mitra. Circuit Failure Prediction and Its Application to Transistor Aging. *Proceedings of the IEEE VLSI Test Symposium*, pages 277 – 286, May 2007.

[5] N. Ahmed, C. P. Ravikumar, M. Tehranipoor, and J. Plusquellic. At-Speed Transition Fault Testing With Low Speed Scan Enable. *Proceedings of the IEEE VLSI Test Symposium*, pages 42 – 47, 2005.

[6] M. Alam. Reliability- and Process-Variation Aware Design of Integrated Circuits. *European Symposium on Reliability of Electron Devices, Failure Physics and Analysis*, 48(8):1114 – 1122, August 2008.

[7] M. A. Alam and S. Mahapatra. A comprehensive Model of PMOS NBTI Degradation. *Microelectronics Reliability*, 45(1):71 – 81, 2001.

[8] T. Asaka, S. Bhattacharya, S. Dey, and M. Yoshida. A Practical Low-Overhead RTL Design-for-Testability Technique for Industrial Designs. *Proceedings of the IEEE/ACM International Test Conference*, 7(1):265 – 274, November 1997.

[9] N. Agarwal B. Shaer, K. Aurangabadkar. Testable Sequential Circuit Design: Partitioning for Pseudoexhaustive Test. *Proceedings of the IEEE Computer Society Annual Symposium on VLSI*, pages 244 – 245, February 2003.

[10] J. Bardeen and W. H. Brattain. The Transistor, A Semiconductor Triode. *Physical Review*, 74(2):230 – 231, July 1948.

[11] P. H. Bardell, W. H. McAnney, and J. Savir. *Built In Test for VLSI: Pseudorandom Techniques*. Wiley-Interscience Publications, 1987.

[12] D. R. Bild, G. E. Bok, and R. P. Dick. Minimization of NBTI Performance Degradation Using Internal Node Control. *Proceedings of the IEEE/ACM Conference on Design, Automation and Test in Europe*, pages 148 – 153, April 2009.

[13] F. Brglez, D. Bryan, and K. Kozminski. Combinational Profiles of Sequential Benchmark Circuits. In *Proceedings of the IEEE International Symposium on Circuits and Systems*, pages 1929 – 1934, 1989.

[14] S. Brown and Z. Vranesic. *Fundamentals of Digital Logic With VHDL Design*. William C Brown Publications, 2000.

[15] M. Bushnell and V. Agrawal. *Essentials of Electronic Testing for Digital, Memory, and Mixed-Signal VLSI Circuits*. Springer Publications, 2000.

[16] T. B. Chan, J. Sartori, P. Gupta, and R. Kumar. On the Efficacy of NBTI Mitigation Techniques. *Proceedings of the IEEE/ACM Design, Automation and Test in Europe*, March 2011.

[17] C. Y. Chang and S. M. Sze. *ULSI Technology*. Mcgraw-Hill College Publications, 1996.

[18] M. Choudhury, V. Chandra, K. Mohanram, and R. Aitken. TIMBER: Time Borrowing and Error Relaying for Online Timing Error Resilience. *Proceedings of the IEEE Design, Automation and Test in Europe Conference and Exhibition*, pages 1554 – 1559, March 2010.

[19] A. Cimatti. Beyond Boolean SAT: Satisfiability Modulo Theories. *International Workshop on Discrete Event Systems*, pages 68 – 73, May 2008.

[20] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press Publications, 3rd edition, 2011.

[21] B. D. Cory, R. Kapur, and B. Underwood. Speed Binning with Path Delay Test in 150-nm Technology. *IEEE Design and Test of Computers*, 20(5):41 – 45, October 2003.

[22] A. Crouch. Debugging and Diagnosing Scan Chain. *ASM Electronic Device Failure Analysis Society*, 7(1):16 – 24, February 2005.

[23] A. Datta, S. Bhunia, J. H. Choi, S. Mukhopadhyay, , and K. Roy. Speed Binning Aware Design Methodology to Improve Profit under Parameter Variations. *Proceedings of the IEEE Asia and South Pacific Conference on Design Automation*, pages 712 – 717, January 2006.

[24] M. DeBole, R. Krishnan, V. Balakrishnan, W. Wang, H. Luo, Y. Wang, Y. Xie, Y. Cao, and N. Vijaykrishnan. New-Age: A Negative Bias Temperature Instability-Estimation Framework for Microarchitectural Components. *International Journal of Parallel Programming*, 37(4):417 – 431, August 2009.

[25] B. Doyle, P. Mahoney, E. Fetzer, and S. Naffziger. Clock Distribution on a Dual-core, Multi-threaded Itanium Family Processor. *Proceedings of the IEEE International Conference on Integrated Circuit and Technology*, February 2005.

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

[27] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, , and T. Mudge. Razor: A Low-Power Pipeline Based on Circuit-Level Timing Speculation. *Proceedings of the International Symposium on Microarchitecture*, pages 7 – 18, December 2003.

[28] P. Fang, J. Tao, J. F. Chen, and C. Hu. Design in Hot-Carrier Reliability for High Performance Logic Applications. *IEEE Custom Integrated Circuits Conference*, pages 525 – 531, 1998.

[29] J. P. Fishburn. Clock Skew Optimization. *IEEE Transactions on Computers*, 39(7):945 – 951, July 1990.

[30] H. Henderson. *Encyclopedia of Computer Science and Technology*. Facts on File, Incorporated Publications, 2008.

[31] E. R. Hnatek. *Integrated Circuit Quality and Reliability*. CRC Press Publications, 1994.

[32] Y. Huang and K. Gallie. Diagnosis of Defects on Scan Enable and Clock Trees. *Proceedings of the IEEE/ACM Conference on Design, Automation and Test in Europe*, pages 1 – 2, March 2006.

[33] Y. Huang, L. Lai, R. Guo, and W. T. Cheng. Diagnosis of Multiple Defects on Scan Enable and Clock Trees. *Proceedings of the IEEE/ACM Conference on Design, Automation and Test in Europe*, pages 1 – 2, March 2006.

[34] D. Josephson, S. Poehlman, and V. Govan. Debug Methodology for the McKinley Processor. *Proceedings of the IEEE International Test Conference*, pages 451 – 460, 2001.

[35] K. Kang, S. Gangwal, S. P. Park, and K. Roy. NBTI Induced Performance Degradation in Logic and Memory Circuits: How Effectively Can We Approach a Reliability Solution? *Proceedings of the IEEE/ACM Design Automation Conference*, pages 726 – 731, March 2008.

[36] K. Kang, H. Kufluoglu, M. A. Alam, and K. Roy. Estimation of NBTI Degradation using IDDQ Measurement. *Proceedings of the IEEE International Reliability Physics Symposium*, pages 10 – 16, April 2007.

[37] K. Killpack, C. Kashyap, and E. Chiprout. Silicon Speedpath Measurement and Feedback into EDA Flows. *Proceedings of the IEEE/ACM Design Automation Conference*, pages 390 – 395, June 2007.

[38] Z. Lak and N. Nicolici. A New Algorithm for Post-Silicon Clock Measurement and Tuning. *IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems*, pages 53 – 59, October 2011.

[39] Z. Lak and N. Nicolici. In-System and On-the-Fly Clock Tuning Mechanism to Combat Lifetime Performance Degradation. *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 434 – 441, November 2011.

[40] L. Lee, L. Wang, P. Parvathala, and T. M. Mak. On Silicon-Based Speed Path Identification. *Proceedings of the IEEE VLSI Test Symposium*, pages 35 – 41, 2005.

[41] D. Lorenz, G. Georgakos, and U. Schlichtmann. Aging Analysis of Circuit Timing Considering NBTI and HCI. *Proceedings of the IEEE International On-Line Testing Symposium*, pages 3 – 8, June 2009.

[42] E.C. Marques, G.G.S. Junior, L.A.B. Naviner, and J.F. Naviner. Effective Metrics for Reliability Analysis. *Proceedings of the IEEE International Midwest Symposium on Circuits and Systems*, pages 237 – 240, August 2010.

[43] W. H. Mcanney and J. Savir. Built-In Checking of the Correct Self-Test Signature. *Proceedings of the IEEE/ACM International Test Conference*, pages 1142 – 1145, 1986.

[44] G. De Micheli. *Synthesis and Optimization of Digital Circuits*. McGraw-Hill Publications, 1994.

[45] G. E. Moore. Cramming More Components onto Integrated Circuits. *Electronics*, pages 114 – 117, April 1965.

[46] M. Murakawa, E. Takahashi, T. Susa, and T. Higuchi. Post-Fabrication Clock Timing Adjustment for Digital LSIs with Genetic Algorithms Ensuring Timing Margins. *IEEE International Conference on Systems, Man and Cybernetics*, pages 3670 – 3674, October 2004.

[47] S. Naffziger, B. Stackhouse, T. Grutkowski, D. Josephson, J. Desai, E. Alon, and M. Horowitz. The Implementation of a 2-Core, Multi-Threaded Itanium Family Processor. *IEEE Journal of Solid-State Circuits*, 41(1):197 – 208, 2006.

[48] K. Nagaraj and S. Kundu. An Automatic Post Silicon Clock Tuning System for Improving System Performance Based on Tester Measurements. *Proceedings of the IEEE/ACM International Test Conference*, October 2008.

[49] K. Nagaraj and S. Kundu. A Study on Placement of Post Silicon Clock Tuning Buffers for Mitigating Impact of Process Variation. *Proceedings of the IEEE/ACM Conference on Design, Automation and Test in Europe*, pages 292 – 295, 2009.

[50] S. Narayanan and M. E. Levitt. Pipelined Scan Enable for Fast Scan Testing. (US Patent 5774474), June 1998.

[51] Z. Navabi. *VHDL: Analysis and Modeling of Digital Systems*. McGraw-Hill Professional Publications, 1997.

[52] Z. Navabi. *Digital System Test and Testable Design: Using HDL Models and Architectures*. Springer Publications, 1st edition, 2010.

[53] V. Nawale and T. W. Chen. Optimal Useful Clock Skew Scheduling In the Presence of Variations Using Robust ILP Formulations. *Proceedings of the IEEE International Conference on Computer Aided Design*, pages 27 – 32, November 2006.

[54] R. Nieuwenhuis, A. Oliveras, and C. Tinelli. Solving SAT and SAT Modulo Theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL. *Journal of the ACM*, 53(6), November 2006.

[55] M. Omana, D. Rossi, N. Bosio, and C. Metra. Self-Checking Monitor for NBTI Due Degradation. *Proceedings of the IEEE International Mixed-Signals, Sensors, and Systems Test Workshop*, pages 1 – 6, June 2010.

[56] B. C. Paul, K. Kang, H. Kufluoglu, M. Alam, and K. Roy. Temporal Performance Degradation under NBTI: Estimation and Design for Improved Reliability of Nanoscale Circuits. *Proceedings of the IEEE/ACM Design, Automation and Test in Europe*, pages 1 – 6, March 2006.

[57] B. C. Paul, K. Kang, H. Kufluoglu, M. A. Alam, and K. Roy. Impact of NBTI on the Temporal Performance Degradation of Digital Circuits. *IEEE Electron Device Letters*, 26(8):560 – 562, 2005.

[58] W. Qiu, J. Wang, X. Lu, Z. Li, D. M. H. Walker, and W. Shi. At-Speed Test for Path Delay Faults Using Practical Techniques. *Proceedings of the IEEE International Workshop on Current and Defect Based Testing*, pages 61 – 66, April 2004.

[59] J. M. Rabaey, A. Chandrakasan, and B. Nikolic. *Digital Integrated Circuits, Second Edition*. Prentice-Hall Publications, Second edition, 2003.

[60] P. J. Restle and A. Deutsch. Designing the Best Clock Distribution Network. *Symposium on VLSI Circuits Digest of Technical Papers*, pages 2 – 5, June 1998.

[61] S. Rusu, S. Tam, H. Muljono, D. Ayers, J. Chang, B. Cherkauer, J. Stinson, J. Benoit, R. Varada, J. Leung, R. Dilip Limaye, and S. Vora. A 65-nm Dual-Core Multithreaded Xeon Processor With 16-MB L3 Cache. *IEEE Journal of Solid-State Circuits*, 42(1):17 – 25, January 2007.

[62] K. K. Saluja, S. Vijayakumar, W. Sootkaneung, and X. Yang. NBTI Degradation: A Problem or a Scare? *Proceedings of the International Conference on VLSI Design*, pages 137 – 142, January 2008.

[63] T. Sato and Y. Kunitake. A Simple Flip-Flop Circuit for Typical-Case Designs for DFM. *Proceedings of the International Symposium on Quality Electronic Design*, pages 539 – 544, March 2007.

[64] R. S. Shelar. An Efficient Clustering Algorithm for Low Power Clock Tree Synthesis. *Proceedings of the International Symposium on Physical Design*, pages 181 – 188, March 2007.

[65] D. Tadesse, J. Grodstein, and R. I. Bahar. AutoRex: An Automated Post-Silicon Clock Tuning Tool. *Proceedings of the IEEE/ACM International Test Conference*, pages 1 – 10, October 2009.

[66] Taiwan Semiconductor Manufacturing Corporation. TSMC Website. http://www.tsmc.com.

[67] S. Tam, S. Rusu, U. N. Desai, R. Kim, J. Zhang, and I. Young. Clock Generation and Distribution for the First IA-64 Microprocessor. *IEEE Journal of Solid-State Circuits*, 35(11):1545 – 1552, November 2000.

[68] J. Tsai, D. Baik, C. C. Chen, and K. K. Saluja. A Yield Improvement Methodology Using Pre- and Post-Silicon Statistical Clock Scheduling. *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 611 – 618, 2004.

[69] J. Tsai, L. Zhang, and C. C. Chen. Statistical Timing Analysis Driven Post-Silicon-Tunable Clock-Tree Synthesis. *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 575 – 581, 2005.

[70] L.Y. Ungar and S. Davidson. Simplified Metrics for Evaluating Designs for Testability. *Proceedings of the IEEE Autotestcon*, pages 293 – 298, September 2009.

[71] G. Venkataraman, N. Jayakumar, J. Hu, P. Li, S. Khatri, A. Rajaram, P. McGuinness, and C. Alpert. Practical Techniques to Reduce Skew and Its Variations in Buffered Clock Networks. *Proceedings of the IEEE International Conference on Computer-Aided Design*, pages 592 – 596, November 2005.

[72] V. Vorisek, T. Koch, and H. Fischer. At-Speed Testing of SOC ICs. *Proceedings of the IEEE/ACM Conference on Design, Automation and Test in Europe*, pages 120 – 125, 2004.

[73] L. Wang, C. Wu, and X. Wen. *VLSI Test Principles and Architectures: Design for Testability*. Morgan Kaufmann Publications, 2006.

[74] W. Wang, V. Reddy, B. Yang, V. Balakrishnan, S. Krishnan, and Y. Cao. Statistical Prediction of Circuit Aging Under Process Variations. *Proceedings of the IEEE Custom Integrated Circuits Conference*, pages 13 – 16, September 2008.

[75] W. Wang, Z. Wei, S. Yang, and Y Cao. An Efficient Method to Identify Critical Gates Under Circuit Aging. *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 735 – 740, November 2007.

[76] W. Wang, S. Yang, S. Bhardwaj, R. Vattikonda, S. Vrudhula, F. Liu, and Y Cao. The Impact of NBTI on the Performance of Combinational and Sequential Circuits. *Proceedings of the IEEE/ACM Design Automation Conference*, pages 364 – 369, June 2007.

[77] S. Wu, L. Wang, L. Yu, H. Furukawa, X. Wen, W. Jone, N. A. Touba, F. Zhao, J. Liu, H. Chao, F. Li, and Z. Jiang. Logic BIST Architecture Using Staggered Launch-on-Shift for Testing Designs Containing Asynchronous Clock Domains. *International Symposium on Defect and Fault Tolerance in VLSI Systems*, pages 358 – 366, 2010.

[78] D. Xiang, Y. Xu, and H. Fijiwara. Non-Scan Design for Testability for Synchronous Sequential Circuits Based on Conflict Analysis. *Proceedings of the IEEE/ACM International Test Conference*, pages 520 – 529, October 2000.

[79] G. Xu and A. D. Singh. Low Cost Launch-on-Shift Delay Test with Slow Scan Enable. *Proceedings of the IEEE European Test Symposium*, pages 9 – 14, 2006.

[80] M. Zhao, X. Wei, Y. Cai, and X. Hong. Quick and Effective Buffered Legitimate Skew Clock Routing. *Proceedings of the IEEE International Symposium on Circuits And Systems*, pages 337 – 340, 2004.

[81] N. M. Zivanov. *Probabilistic Modelling and Optimization for Circuit Reliability*. ProQuest, UMI Dissertation Publications, 2011.

Index