A DFT Technique for Testing High-Speed Circuits with Arbitrarily Slow Testers

by

Muhammad A. Nummer

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2001

I hereby declare that I am the sole author of this thesis.

I authorize the University of Waterloo to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Muhammad A. Nummer

I further authorize the University of Waterloo to reproduce this thesis by photocopying or other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Muhammad A. Nummer

The University of Waterloo requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

Abstract

Rapid advances in semiconductor technology have created many challenging requirements for automatic test equipment (ATE). As a result, design-for-testability (DFT) and built-in-self-test (BIST) techniques are becoming essential parts of any high-speed VLSI design. This thesis presents a DFT technique for testing high-speed circuits with a low-speed clock in test mode. With this technique, the test mode clock frequency can be reduced with no lower limit. This technique imposes few requirements on ATEs and facilitates the testing process. A CMOS implementation capable of achieving an accuracy of 50ps is presented. The effectiveness of this technique is demonstrated using a 16-bit, 1.4GHz pipelined multiplier. Simulation results show that we are able to do performance binning and detect delay faults as small as 50ps at frequencies much lower than the rated operating frequency of the test vehicle.

Acknowledgements

All praise is due to God, Most Gracious, Most Merciful, Whose bounties and blessings are ever dominating throughout my life.

I would like to express my deepest gratitude and appreciation to my supervisor, Prof. Manoj Sachdev. Prof. Sachdev has always been an invaluable source of support, guidance, and encouragement. I would also like to thank Prof. James Barby and Prof. Anwar Hasan for reading and suggesting improvements to the presentation of this thesis.

My appreciation to all my colleagues in the VLSI research group who were of great help and support. In particular I would like to thank A. Elsayed, A. Elraey, M. Allam, M. Shashaani, A. Fahim, M. Anis, M. Elgebaly, M. Kamal, M. El Said, W. Chung, and I. Al-Mohandes.

My deep gratitude to my mother and father for their ever continuous support, encouragement, and prayers. No words of appreciation could ever reward them for all they have done for me. I would like to thank my wife, who shared with me every day throughout the course of this work. Her support, patience, and understanding played a major role in helping me finish this thesis. My daughter, Yumna, brought so much joy to my life which has been and continue to be a great source of encouragement.

This research was supported in part by Communications and Information Technology Ontario (CITO) and Gennum Corporation. This support is greatly appreciated.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The on-chip clock frequency of high-performance state-of-the-art VLSI CMOS circuits has surpassed 1.5GHz. It is expected that the speed of such circuits will continue to increase for future technology generations. The 1999 edition of the International Technology Roadmap for Semiconductors (ITRS) expects that the on-chip clock frequency will exceed 3GHz by year 2005 and 13GHz by year 2014 [3]. With smaller geometries, higher speeds, and increased interconnects, it is more likely for small imperfections in the fabrication process to cause device failure.

According to the ITRS, most of the technology problems causing yield losses and cost increases are related to the slower growth in automatic test equipment's (ATE's) capabilities versus the ever increasing device clock frequency [3]. In the past, the accuracy of ATEs used to be 4-5 times higher than the state-of-the-art ICs [18]. That is why it was easy to perform at-speed functional testing. In the last two decades, however, while the clock frequencies of VLSI circuits have improved at an average rate of 30% per year, the tester accuracy has improved only at a rate of 12%. If this trend continues, tester timing accuracy will soon approach the cycle time of high-performance devices making at-speed test almost impossible. Table 1.1 shows the ITRS

Year	1999	2001	2003	2005	2008	2011	2014
Yield (%)	87	84	79	73	64	56	50
Off-chip device period (ps)	830	700	580	500	400	340	260
Overall ATE accuracy (ps)	200	160	130	100	100	100	100

Table 1.1: ITRS Trends in yield, off-chip device speed, and tester accuracy [3].

expected trends for yield, off-chip device speed, and the overall tester timing accuracy [1]. It is clear from this data that long before the tester timing accuracy reaches the cycle time of the devices, yield loss [2] due to insufficient accuracy of the tester will become unacceptably high. As yield for future technology generations becomes a major issue, the importance of performing a test capable of ensuring acceptable quality levels becomes crucial. In the same context, if future ATEs are not able to keep up with device speed, not only the yield but the out-going quality of these devices will also be greatly affected.

The ATE cost per pin for high-performance circuits has remained approximately constant for the past 20 years at around $10-12k. Recently, this value has begun to fall below $8k/pin and is expected to continue to decrease in years to come. Nevertheless, it is expected that the demand for higher speed, greater accuracy, more time sets, and increased vector memory will offset most of the gains seen for reducing ATE cost [3]. According to the ITRS, it may cost more to test a transistor than it costs to manufacture it by 2014.

Due to the slow advances and the high cost of ATE, we might not be able to test future high-performance VLSI circuits. Therefore, it will be essential to design these circuits with design-for-testability/built-in-self-test (DFT/BIST) techniques to reduce the reliance on traditional, high-cost, full-feature testers. The requirements of ATEs designed to work with DFT/BIST techniques

[1] Overall tester timing accuracy aggregates timing error comprised of input edge placement accuracy, output edge placement accuracy, and input to output timing accuracy.

[2] Yield loss is a measure of how many "good" devices are incorrectly considered "bad" due to tester inaccuracy.

are much simpler than the traditional testers.

In this thesis, we propose a DFT technique for testing high-speed circuits with arbitrarily slow testers. Testing high-speed circuits with slow testers has several advantages. It provides the capability of detecting the subtle timing failures with relative ease resulting in improved quality. Furthermore, with these techniques, the life time of an ATE can span multiple life cycles of a product. As a result, using these techniques to test high-speed circuits is expected to reduce the cost of testing and manufacturing.

The creation of a low frequency test mode in digital circuits was first introduced by Agrawal and Chakraborty [1] in 1995. In their proposal, a quantifiable, externally controlled delay is added such that high-performance testing can be carried out with relatively slow-speed testers. They used a pulse-triggered flip-flop in which a dynamic latch is introduced inside a traditional master-slave flip-flop. In 1999, Shashaani and Sachdev proposed the controlled delay flip-flop (CDFF) [18] as an alternative to the pulse-triggered flip-flop. In this technique an additional test mode clock is used to control the delay of the flip-flop. The main advantages of the CDFF over the pulse-triggered flip-flop are the stable operation and improved performance in normal mode.

The remainder of this thesis is organized as follows. Chapter 2 gives a concise review of VLSI testing. The chapter provides details on faults and physical defects in CMOS circuits. A review of the techniques used for testing marginal ICs is provided. Chapter 3 introduces the concept of testing high-performance circuits with slow speed testers. Details of using CDFFs are given. This chapter introduces the technique proposed by this work, in which high-performance circuits can be tested with arbitrarily slow testers. Chapter 3 also illustrates how to generate clocks used for CDFFs in order to reduce the test mode clock frequency arbitrarily. Chapter 4 provides design details of the clock generation circuit. An overview of the 16-bit pipelined multiplier used as a test vehicle is given in Chapter 5. Simulation results for the clock generation

circuit, performance binning [3] for different process corners, and delay fault detection are given in Chapter 6.

[3]Performance binning is a test procedure used to segregate devices in a number of bins depending on their maximum operating frequency.

Chapter 2

Background

Advances in Very Large Scale Integration (VLSI) technology have enabled the implementation of complex digital circuits in a single chip, reducing system size and power consumption. To design such complex circuits, an array of computer aided design tools have been developed. These tools dramatically reduce the time required to design new circuits, allowing significant advances in improved system characteristics and performance. This in turn has intensified the complexity of testing such chips to verify that they function correctly. Semiconductor manufacturing processes are so complex that this verification cannot be done on a sampling basis; rather, each chip must be individually tested [7].

2.1 VLSI Testing

The purpose of testing a VLSI device is to ensure, with reasonable confidence, that the device functions according to the design specifications. This testing must be achievable within certain economic constraints to keep the cost per device as low as possible.

Following Moore's Law for the past two decades, the silicon die cost of integrated circuits

has decreased as the number of transistors per die has continued to increase. In contrast, during the same period, the cost of testing integrated circuits in high-volume manufacturing has been steadily increasing. It is predicted that the cost of testing transistors may actually surpass the cost of fabricating them within the next two decades [17]. As ICs become more highly integrated, the job of diagnosing failures becomes increasingly difficult. Marching into the deep submicron regime poses many challenges to the testing problem. With smaller geometries, higher speeds, and increased interconnects, it is more likely to have an increase in the tendency for small imperfections in the fabrication process to result in actual failures. This is why it is very essential to use advanced test techniques that enables the testing process to cope with the advances in semiconductor technology.

Fundamentally, there are two ways in which a VLSI device can fail. It can fail *parametrically* in that the technology dependent electrical parameters such as voltage, current, capacitance, speed, or gain are out of the specifications. Or, it can fail *functionally* such that the device, independent of its electrical characteristics, performs an incorrect logic function.

Generally, there are three main types of testing techniques. The first two are *parametric* testing and *functional* testing. A VLSI device may fail a parametric test and still pass a functional test and vice versa. The third is a group of tests called *accelerated life tests*. These tests simulate many years of operation by stressing the device under test. A repeated functional test pattern is applied to the device during this process so that the chip is being fully exercised during its accelerated “life-time”. These tests will be described in more detail in section 2.3.4.

2.2 Faults and Physical Defects in CMOS ICs

2.2.1 Failures, Faults, and Defects

A *failure* is said to have occurred in a circuit if it deviates from its specified behaviour [12]. A *fault*, on the other hand, is a physical defect that causes the circuit to seriously malfunction. Although it may not be feasible, the testing objective is to detect all defects which affect the circuit behaviour in any respect at the time of testing or during its lifetime. The defects that do not affect the circuit behaviour at the time of testing may cause a fault in the future, and hence become a major reliability concern.

2.2.2 Fault Models for CMOS ICs

There are many sources of faults in a circuit. Examples include breaks in signal lines and line shorts to ground, supply, or other signal lines. Other sources include design rule violations and errors in the design specifications. A fault model is a representative description of the effect the fault has on circuit operation.

Stuck-At Fault Model

In this model, it is considered that any line in a circuit may have a fault, which causes it to remain permanently either at logic 1 or at logic 0. If the logic value of a line remains at 1, the line is said to be stuck-at 1, and if the value remains at 0, it is called stuck-at 0.

The stuck-at fault model is the most commonly used logic-level fault model [16]. Nevertheless, the stuck-at fault model cannot detect many physical defects in present day VLSI, which mainly uses CMOS technology (82% of integrated circuits in 1998 [11]). Faults in CMOS circuits do not necessarily produce logical faults that can be described as stuck-at faults. Various

studies since the late 1970s suggest that the basic failure modes in VLSI circuits are physical short and open circuits [15]. These studies reported that only a small fraction of bridging and open faults can be modeled as stuck-at faults. For example a bridging fault might not cause any logical faults, rather it may cause delay or timing faults. A bridging may occur between two electrical nodes. On the other hand, any part of a diffusion, polysilicon, or metal line may have an open fault. Any contact between two layers may be open. This raised the need for comprehensive fault models to include bridging and open faults.

Bridging Fault Model

In, general, bridging faults can be classified into three groups:

- Bridging within a logic element without feedback. This is a bridging fault between internal nodes within a logic element. The most likely faults are recognized as (1) gate-to-drain bridging, (2) gate-to-source bridging, and (3) source-to-drain bridging (sometimes referred to as transistor stuck-on fault) [15].
- Bridging between two logic elements without feedback.
- Bridging faults between logical nodes with feedback. The presence of such feedback can cause the circuit to oscillate or convert it to a sequential circuit [12].

Bridging faults between transistors in CMOS circuits are very difficult to test using logic testing. Such kind of faults affect the dynamic behaviour of the circuit, reduce the noise margin, and cause the circuit to age very quickly by drawing and dissipating very large currents. The situation becomes more complex when the bridging fault has some finite resistance (resistive short). It has been observed that the short resistance can vary from few ohms to about 4.7kΩ [6]. Test methods such as I_{DDQ}, delay test, very-low-voltage testing, or burn-in are normally used to overcome the lack of deterministic testability of devices with bridging faults.

Stuck-Open and Stuck-On Fault Models

These are transistor level fault models. It is only at this level that the complete structure is known. That is why these models can give a realistic representation of CMOS faults.

A stuck-open fault implies that there is a permanent open between the drain and the source of a transistor. The drain-source resistance of a stuck-open transistor is significantly higher than the OFF resistance of a nonfaulty transistor. If the values of these two resistances are close to each other, the transistor is considered to be stuck-off. Although only about 1% of the CMOS faults are due to stuck-off/stuck-open transistors [12], considerable research has been directed at detecting these faults. This is attributed to the fact that it has been demonstrated that in the presence of these faults in a CMOS logic gate, the gate shows a memory effect under certain input conditions [16], thus turning a combinational circuit into a sequential circuit.

If a transistor is stuck-on, it operates in the conduction mode regardless of the voltages at its gate. This means that the drain-source resistance of a stuck-on transistor is always close to the ON resistance of a nonfaulty transistor. If this resistance is much smaller than the ON resistance of a nonfaulty transistor, the transistor is said to be stuck-closed. These faults can be modeled as bridging faults between the source and the drain of a transistor and again it would be easy to detect such faults using a parametric test.

Parametric and Transient Faults

A VLSI circuit might have faults that do not affect the logical behaviour of the circuit, but degrade the performance and reliability of the circuit. These are called parametric faults. These faults are considered to be major reliability threats in CMOS ICs. Parametric faults include shifts in substrate leakage current, gate-oxide leakage current, and threshold voltage. Although delay faults are also parametric faults, they are considered separately. The cause of a parametric fault may be a physical defect or a variation in a process parameter. The general procedure to sensitize

these faults is accelerated stress testing.

Another possibility is the intermittent or transient faults. This type of faults affect the circuit behaviour at random. Although, the cause of such faults can be a physical defect, most probably the cause is an environmental factor. Examples include external electromagnetic interference and ionization radiation. These faults are not repairable because they do not cause any physical damage to the hardware [16].

Delay Fault Models

The size of a defect determines whether the defect will affect the logic function of a circuit or not. Smaller defects, which are likely to cause partial shorts or opens, have a high probability of occurrence due to the statistical variation in the manufacturing process. Such defects cause the circuit to fail to meet its timing specifications without any alteration of the logic function of the circuit. These types of faults can be modeled as delay faults. Other reasons for delay faults include transistor threshold voltage shifts and increased parasitic capacitance. To test a timing fault, two popular models are used:

1. Single-gate delay fault model. A circuit has a gate delay fault if a localized timing failure causes the propagation delay of at least one path through the circuit to exceed the specified cycle time [8].

2. Path-oriented delay fault model. A circuit has a path delay fault if the propagation delay of at least one path through the circuit exceeds the specified cycle time [8].

Although a single gate may satisfy the timing specifications, the circuit may still malfunction because of the cumulative effect of delay variation. This illustrates the importance of the path-oriented delay fault models. In-general, the delay in the longest and the shortest paths are examined for delay faults. If these delays are within the clock period, the circuit is considered to be

nonfaulty; otherwise, the circuit has a delay fault.

Sometimes, the difference in timing is very small to be detected by a delay test. In such cases other test techniques that magnifies the delay fault should be used.

2.2.3 Defects in CMOS

Defects are the actual source of most of the faults taking place in CMOS. To achieve higher test accuracy, test techniques should be defect oriented rather than fault oriented. This is why it is very important to categorize defects and build test strategies based upon their electrical properties. This means that the test strategy should match the defect electrical properties rather than the fault definition.

Hawkins *et al.* [10] suggested building test strategies based upon defect classes. According to their proposal, all defects can be categorized into three groups:

1. Bridge Defect Classes. These include bridging defects at transistor node, logic gate I/O, and power lines. They can happen in a combinational or a sequential circuit. The *I-V* characteristics of these defects can be nonlinear or linear. Hawkins *et al.* showed that the detection of these defects is more efficient with a parametric test (I_{DDQ}) than with a Boolean test. Correct Boolean functionality exists for signal node bridge defects when the defect exceeds a critical resistance. It has been found that the critical resistance can be as low as 10Ω and as high as 5kΩ [10] depending on defect location, transistor size, transistor W/L ratios, CMOS technology, and input logic patterns. This defect class is further divided into three types:

    - Bridge Type-1 Combinational Defect Class. These defects occur at specific locations and include the six possible transistor node bridges, logic gate I/O node to power bus bridges, and power bus to power bus bridges.

- Bridge Type-2 Layout Defect Class. These occur at a variety of locations and include bridges between two or more logic gates or between logic gate I/O and transistor nodes.
- Bridge Type-3 Sequential Defect Class. This class includes transistor node bridges of sequential circuits.

2. Open Circuit Defect Classes. These include open contacts, metallization opens, and opens in diffusion or polysilicon. Hawkins *et al.* placed the different types of opens into six classes:

   - Open Type-1 Transistor-On Defect Class. This class causes the transistor to have a stuck-on behaviour.
   - Open Type-2 Transistor Pair-On Defect Class. This class represents an open that causes a pair of transistors to be on.
   - Open Type-3 Transistor Pair-On/Off Defect Class. In this class the open can be modeled as a stuck-at fault.
   - Open Type-4 Sequential Open Defect Class. This class represents large opens in sequential circuits which causes either degraded voltages (which might cause a Boolean fault) or strong clamping to a supply voltage.
   - Open Type-5 Transistor-Off Defect Class. This class has a memory effect in CMOS ICs and is very difficult to detect.
   - Open Type-6 Delay Defect Class. This defect class includes the delay effect in open circuits having small cracks that allow tunneling and subsequent delay errors.

   While the response of certain open circuit defects is not always predictable, the possible responses are bounded. Therefore, test strategies for open circuit defects can take account

of all six possibilities [10].

3. Parametric Delay Defect Class. This class defines the subset of defects causing delay faults that are neither in the category of bridges or opens. These defects include shifts in via resistance, threshold voltage, and transistor W/L ratios. Parametric delay faults are hard to detect either by Boolean test or by I_{DDQ} test. In order to detect such faults, delay fault testing or at speed testing seem to be promising alternatives [16].

As can be seen from the previous discussion, there is no one single test method that can detect all possible faults in CMOS ICs. Some faults are very easy to detect using a Boolean test. Others require parametric tests or a combination of the two. Among the most difficult faults to detect are faults causing what is called marginal ICs. This will be discussed in the following section.

2.3 Testing Marginal ICs

2.3.1 What are Marginal ICs

The objective of most functional and parametric test techniques is to detect chips that are not working according to design specifications. If an IC passes both functional and parametric tests, it is assumed to be functional.

However, some of the "functional" ICs may be in fact marginal ICs. Marginal ICs contain flaws; defects in a chip that do not cause failures at normal operating conditions but degrade chip performance, reduce noise immunity, or draw excess supply current [6, 9]. Marginal ICs cause problems with reliability and must be detected before they are shipped. These chips can cause intermittent failures in the sense that even if they may pass production tests, they can fail to work in the field at different operating conditions. They may cause problems if the supply voltage changes during operation due to IR drops or simultaneous switching noise. If these chips

are used for low-power applications, and if the defect inside the chip is causing abnormal static current flow, this may have serious consequences. The excess static current can cause early-life failures and accelerated chip wearout.

2.3.2 Marginal ICs and Timing Failures

The most noticeable effects of marginal ICs on chip performance are timing failures in the form of delay flaws. Timing failures occur when the delay of the manufactured component is different from the designed delay. If the timing failure is such that the circuit fails to work at the designed speed but continues to be functional at a lower speed, it is called a delay fault. On the other hand, a circuit has a delay flaw (non-operational delay failure) if there is a timing failure but the circuit continues to work at the designed speed [8].

2.3.3 Causes for Timing Failures

In this section causes for timing failures, of which delay flaws are a subset, are presented. The main source for timing failures is manufacturing defects. They can also result from design defects, such as violating layout design rules or aggressive device scaling. Normal device wearout is another source for timing failures. Listed below are the different possible causes for timing failures as identified by Chang and McClusky [5].

1. Transmission gate opens. This means that one of the transistors in a CMOS transmission gate is malfunctioning and cannot pass any signal. This causes the transmission gate to have a degraded signal at its output.

2. Threshold voltage shifts. The cause for this can be process variation or hot carrier effects. Process variation causes global threshold voltage shifts. On the other hand, hot carrier effects can cause either global or local threshold voltage shift. A higher threshold voltage

causes the transistor to have a lower transconductance. As a result, the transistor has a lower driving capability and causes an excess delay during a transition.

3. Diminished-drive gates. This situation is associated with gates whose output drives a high fan-out, long interconnection wires, or both. The reason for the diminished-drive can be improper design of the driver or weak gates having smaller gate widths caused by manufacturing defects. Some of these driving gates are designed to be parallelly connected to avoid using large devices. If one of the branches in such a gate malfunctions (for example, due to an open at its output), the other branches will be heavily loaded and might encounter diminished-drive.

4. Gate-oxide and metal shorts. These were discussed in sections 2.2.2 and 2.2.3. They can cause degraded signals and increased leakage currents.

5. Defective interconnect buffers. For deep-submicron technologies, the interconnect delay is no longer a negligible part of the total delay. This is why buffers are sometimes used to reduce the RC delay of long wires. If these buffers are defective, they might cause different failure modes including degraded signals, high leakage current, longer gate delays, or longer interconnect delays.

6. High resistance interconnects and via defects. Electromigration, via defects and stress voids, can cause the resistance of an interconnect to increase resulting in a longer RC delay.

7. Tunneling opens. These are opens due to small cracks. They allow the IC to be functional at low frequencies but fail at high frequencies.

2.3.4 Test Techniques for Marginal ICs

There are many test techniques that can be used for detecting marginal ICs. The common procedure in most of these techniques is to change the operating environment so as to provoke the flaws within the chip. Nevertheless, some other techniques perform testing at normal operating conditions.

Marginal Voltage Screening

In this technique, for each test pattern, the power supply voltage is lowered until a logic error is observed at circuit output [9]. A voltage profile of good chips can be built this way. When a marginal chip is tested it gives a different profile. The main disadvantage of this method is the need for static voltage adjustment for each test set which is very time consuming.

Cut-Off Frequency Test

For a certain value for the supply voltage, there is a cut-off frequency above which the circuit fails to function. This technique is based on searching for the cut-off frequency of good chips at different values for the supply voltage including low voltages. A chip with flaws gives out of range cut-off frequencies. This technique is also time consuming. No clear evidence is given as to what flaws could be detected and how the voltage should be chosen for optimal defect coverage [9].

Corner Testing

Corner testing is a frequently used technique in practice. In this techniques the chip is operated under various worst-case operating conditions specified by the manufacturer. This is to insure that it performs all of its designed functions under these conditions [9].

Accelerated Life Tests

These tests subject devices to higher than usual levels of stress to speed up the deterioration of the circuit under test. These stresses include voltage, temperature, humidity, corrosion, magnetic field, current, pressure, radiation, vibration, salt, and loading. Kuo *et al.* [11] identified many types of these tests. Here, some of the test that can be used to detect marginal chips are listed.

1. Burn-in. Burn-in test is the most commonly used technique for eliminating marginal chips in production [9]. It uses time, bias, current, and temperature accelerating factors to activate time-temperature-dependent failure mechanisms to the point of detection in a relatively short period of time.

2. High Temperature Storage (HTS) Test. Essentially, this test is a bake at temperatures much higher than burn-in. No bias is applied, and the device in not electrically activated. The main purpose for this test is to detect the quality of molding and wiring material used in the assembly process. Another version of this test is high temperature with bias (HTB) test. This test is useful in detecting thin film, transistor, metal, and capacitor defects in MOS devices.

3. Electrical Over-Stress (EOS). Oxide defects are not very responsive to temperature stresses. However, voltage stressing forces defective oxides to fail prematurely. Stressing may take the form of a continuous electrical over-stress combined with an elevated ambient temperature as in HTB.

4. Temperature Cycling (T/C). T/C is performed by alternatively stressing devices at hot and cold temperature extremes. It is used to monitor the reliability of metal and passivation.

Accelerated life tests are usually very expensive because special equipment and long test times are required. They are widely used for process improvement in the development phase of a chip.

In production, they are only done for particular types of chips [9].

Quiescent Power Supply Current (I_{DDQ}) Testing

Any CMOS gate consists of an NMOS pull-down network and a PMOS pull-up network. In a fault-free situation, for any given input only one part conducts, connecting the output node to either the V_{DD} or the GND node. Thus the circuit does not provide a conducting path from V_{DD} to GND. That is why in the fault-free situation, steady-state current in the circuit is very small and is on the order of nA.

In the presence of various physical defects, including defects causing delay flaws, the magnitude of the steady-state current in a CMOS IC might increase a few orders of magnitude. Thus, by monitoring this current it may be possible to determine whether or not a circuit has a defect causing a flaw or a fault.

Many researchers have investigated the ability of I_{DDQ} tests to detect different types of defects and faults. Peters and Oostdijk [14] showed that defects on serial transistors and defects between inputs, that are hard to detect by a voltage test, are easily detectable using I_{DDQ} testing. They also showed that a large number of gate-oxide shorts, that can not be detected using voltage vectors, can be detected by I_{DDQ}. Vierhaus *et al.* [21] showed that defects that have only negligible functional effects (flaws) can have measurable overcurrents in the range of 10μA. Their study was based on bridges and resistive shorts. Singh *et al.* [19] studied the detectability of different classes of opens in CMOS using I_{DDQ} testing. They indicated that a large majority of open defects in CMOS are I_{DDQ} detectable. Moreover, those that are not detected mostly display a stuck-at behaviour and can be reliably detected by Boolean testing.

Very-Low-Voltage Testing (VLV)

Very-low-voltage testing was first introduced in 1993 by Hao and McClusky [9]. This technique makes use of the voltage dependence of CMOS ICs to provoke or trigger the flaws by testing the chip at a reduced value of the supply voltage.

The propagation delay of a CMOS circuit increases monotonically as the supply voltage is reduced from the nominal value to a value close to the threshold voltage. Increments in the propagation delay due to a small change in the supply voltage are much more significant when the supply voltage is small than when it is large. Supply voltage reduction causes the delay faults to be more noticeable. Hence, these faults can be detected easily at frequencies much lower than the operating frequency.

High Performance Testing

The objective of these techniques is to detect the degradation in chip performance (timing failures) due to the presence of flaws.

A simple form of timing testing is to apply patterns to the circuit under test at system speed. This is called *at-speed* testing, and is usually only possible on fast ATE. With the advances in CMOS technology, the speed of operation is becoming very high. Due to problems such as power supply regulation, temperature variation, and electrical parasitics, tester timing inaccuracy continues to rise as a function of the shrinking clock periods of high performance designs [17].

Although, high performance testing techniques can be very efficient in detecting timing faults, they might not be able to detect delay flaws within weak ICs. This is because the difference in timing due to these flaws is usually very small and requires very high precision test equipment to perform the test at normal operating conditions.

One way that allows the use of lower speed testers to test higher speed chips is the multiplexing of tester clock pins to extend its clock frequency range [18]. This is a standard feature in

most modern testers. Other techniques are based upon the creation of a low frequency test mode in circuits. More details about these techniques are given in the following chapter.

Chapter 3

Testing High-Performance Circuit with Arbitrarily Slow Testers

The creation of a low frequency test mode in digital circuits was first introduced by Agrawal and Chakraborty [1]. In their proposal, a quantifiable, externally controlled delay is added such that high-performance testing can be carried out with relatively slow-speed testers. They used a pulse-triggered flip-flop in which a dynamic latch is introduced inside a traditional master-slave flip-flop. The resulting three-latch structure has two modes of operation; normal mode and test mode. In normal mode, the intermediate latch must hold data for most of the clock period while the other two latches remain transparent. In test mode, flip-flop delay can be modulated by changing clock's pulse width. This allows for testing combinational logic and interconnects for delay faults with a lower clock frequency. Although the concept of adding delay in test mode is elegant, this implementation has some important shortcomings as the dynamic latch makes the flip-flop operation sensitive and timing critical. Shashaani and Sachdev proposed the controlled delay flip-flop [18] as an alternative to the pulse-triggered flip-flop. In this technique an additional test mode clock is used to control the delay of the flip-flop. The main advantages of

the CDFF over the pulse-triggered flip-flop are the stable operation and improved performance in normal mode. Details of the operation of the CDFF in normal and test modes are given in the following section.

3.1 CDFF for Testing High-Performance Circuits at Low Speed

Figure 3.1 illustrates a gate level implementation of the CDFF. The transfer of data from the master latch to the slave latch is controlled through a control logic and depends on the relative timing of the clock (CLK) and the test clock (TCLK). To illustrate the operation of the CDFF, a simple model of digital VLSI circuits is depicted in Figure 3.2(a). In this model, a combinational block is sandwiched between two sequential blocks (registers, flip-flops,...etc). In normal mode, TCLK is kept high ensuring normal flip-flop operation (Figure 3.2(b)). Under this condition, the normal mode clock period (T_{NM}) is given by:

$$T_{NM} = t_{prop} + t_{comb} + t_{setup} \tag{3.1}$$

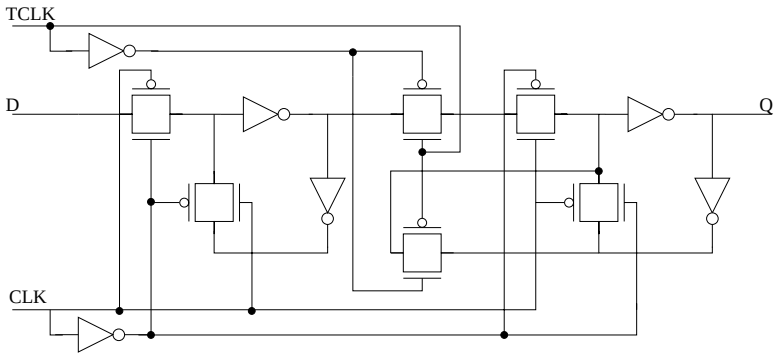


Figure 3.1: Controlled delay flip-flop [18].

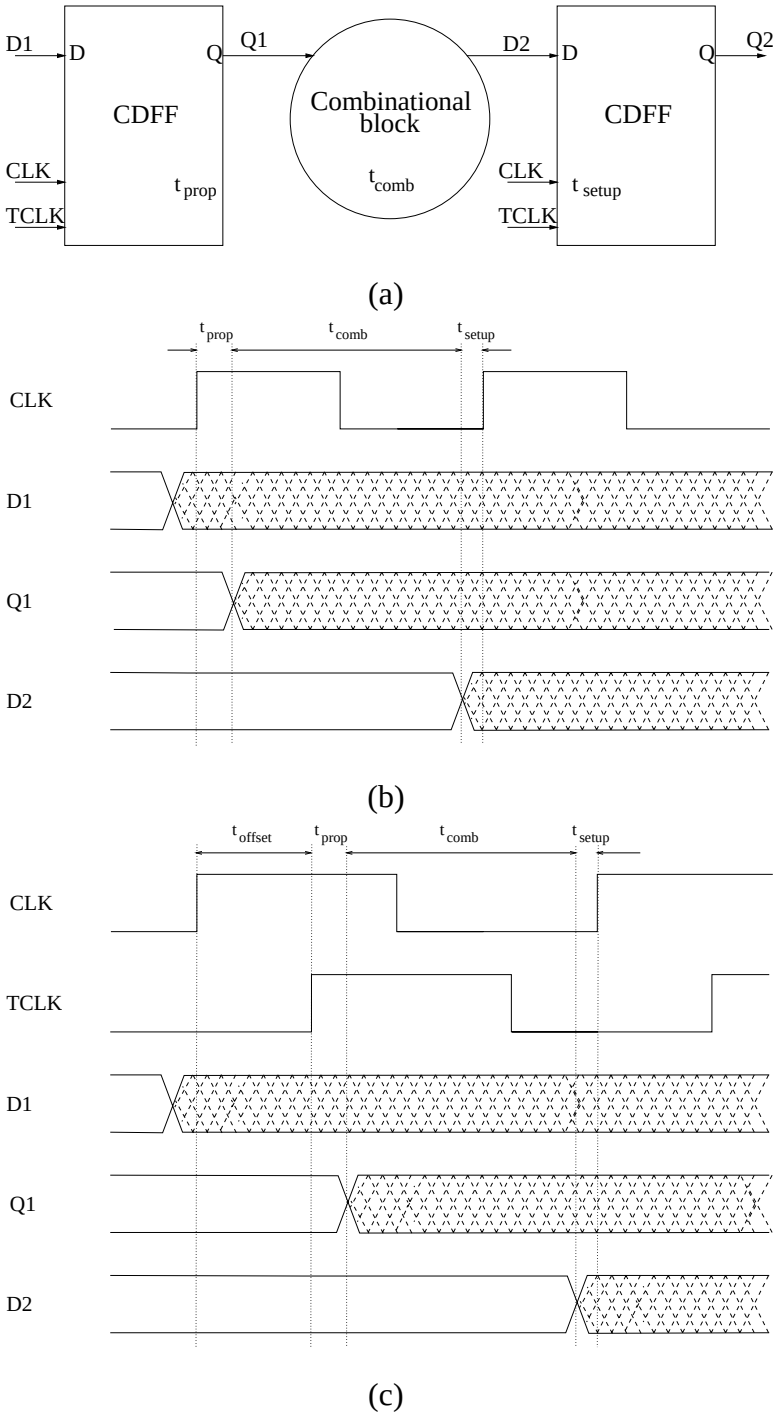


Figure 3.2: CDFF operation. (a) Circuit model. (b) Normal mode. (c) Test mode.

where t_{prop} is the propagation delay of the flip-flop, t_{comb} is the time window allowed for the combinational block to evaluate its input, Q_1, and produce the input of the next sequential block, D_2, and t_{setup} is the setup time of the flip-flop.

In test mode, a tester programmed time offset of the clock is used to generate TCLK. Consequently, flip-flop output, Q_1, appears after an additional delay equal to the time offset between the two clocks. This scenario is illustrated in Figure 3.2(c). Under this condition, the test mode clock period is given by:

$$T_{TM} = t_{prop} + t_{comb} + t_{setup} + t_{offset} \tag{3.2}$$

where t_{offset} is the time offset between the clock and the test clock. The test mode clock period should be large enough to accommodate all delay terms in Equation 3.2. It is clear from this equation that increasing t_{offset} allows the circuit to be tested at a frequency lower than the normal mode frequency. In other words, clock frequency can be reduced while the combinational circuit delays are tested with the same delay margins.

3.2 Using CDFF to Arbitrarily Reduce Test Mode Clock Frequency

In this section, we present a methodology for generating the clock and the test clock for a device using CDFFs in a way that allows the test mode clock frequency to be reduced arbitrarily. This is done through an on-chip clock generation circuit. When generating the clock and the test clock for a circuit using CDFF to improve testability, one has to take into consideration the timing requirements for correct operation. For the CDFF to function properly, the timing of the clock and the test clock must be carefully adjusted to accommodate both the setup time (t_{setup}) and the propagation delay (t_{prop}) of the flip-flop. For the combinational block, it is necessary to have the

flexibility to change the value of t_{comb} so as to determine, with reasonable accuracy, the delay through this block and test the circuit for delay faults. This is also important to enable us to do performance binning to know how well does the circuit meet its timing specifications.

3.2.1 Reducing Test Mode Clock Frequency

In test mode, reducing clock frequency while maintaining correct timing operation for all parts of the circuit means that, if the clock frequency becomes very low, t_{offset} has to be extremely large. As suggested in [18], the test clock can be generated as a delayed version of the clock with a delay of t_{offset}. The problem with this approach is that a slow tester is a low specification device. It is normally difficult for such a device to provide very large time offset with state-of-the-art timing accuracy. As an alternative, H. Speek et al. [20] suggested the use of two programmable duty-cycle controllers and a programmable delay line to generate the clock and the test clock in test mode. Using their design, reducing the test mode clock frequency to a very small value requires a large delay line to generate the required delay with appropriate timing resolution.

Careful examination of the timing diagram in Figure 3.2(c) shows that, instead of generating the test clock by delaying the clock in test mode, the clock can be generated by delaying and inverting the test clock. Generating the clock this way makes t_{offset} (which is the key factor in reducing the test mode clock frequency) independent on the relative timing of the two clocks and allows its value to be increased arbitrarily. Increasing t_{offset} while keeping all the other terms in Equation 3.2 unchanged implies a reduction in test mode clock frequency without affecting the time window allowed for the evaluation of the combinational block. It is clear that by doing this, the test mode clock frequency can be reduced with no lower limit.

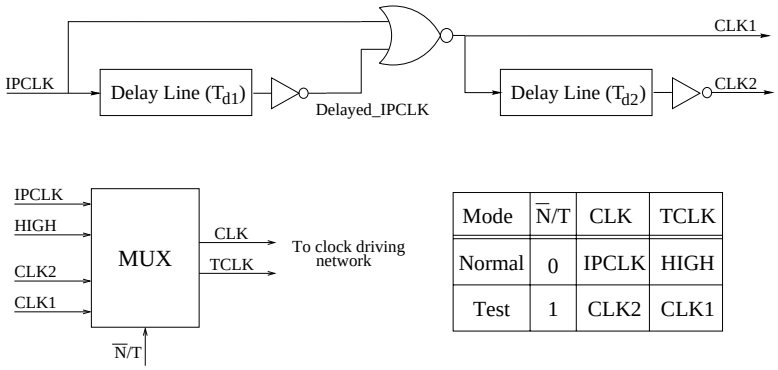
3.2.2 Clock and Test Clock Generation

Figure 3.3(a) depicts a block diagram of a system for generating the clock and the test clock. The input clock, IPCLK, is a rated frequency signal in normal mode and a low frequency, 50% duty cycle signal in test mode. A multiplexer (MUX) is used to select the mode of operation through the mode select input ($\overline{N}/T$). For normal mode operation ($\overline{N}/T$=LOW), IPCLK passes through the MUX to the CLK driving network while TCLK is kept high. In test mode ($\overline{N}/T$=HIGH), two delay lines are used to generate both CLK and TCLK. This is illustrated by the timing diagram in Figure 3.3(b). A delay line is used to generate a clock with pulse width T_{d1} (CLK1). This clock is is selected by the MUX to be the test clock, TCLK. CLK1 passes through the second delay line (with delay T_{d2}), resulting in CLK2. The MUX selects CLK2 to be CLK in test mode. Figure 3.3(b) also shows the D and Q signals of a CDFF to illustrate the relationships amongst the various timing parameters of the system in Figure 3.2(a) on one side and the delays T_{d1} and T_{d2} and the frequency of IPCLK, f, on the other side . These relationships can be expressed by the following two equations.

$$T_{d1} + T_{d2} = t_{prop} + t_{comb} + t_{setup} \tag{3.3}$$

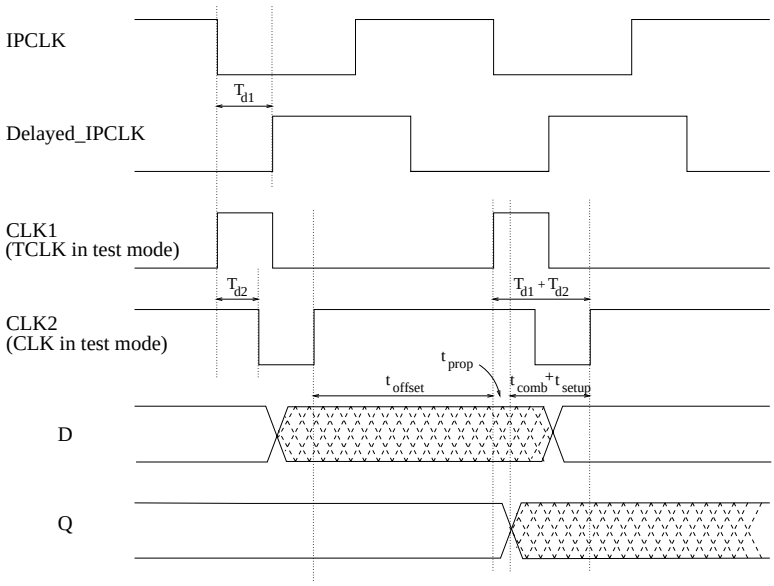
$$1/f = T_{d1} + T_{d2} + t_{offset} \tag{3.4}$$

Assuming fixed f, t_{prop}, and t_{setup}, these equations suggest that a change in either T_{d1}, T_{d2}, or both, leads to an equal change in t_{comb}. This allows the combinational block to be tested for delay faults by changing the time slot allowed for the evaluation of its inputs. For constant T_{d1} and T_{d2}, changing f causes only t_{offset} to change without affecting the operation of either the flip-flop or the combinational block. In order to ensure correct flip-flop operation with variable $T_{d1} + T_{d2}$, we characterize the CDFF to find the limiting values of T_{d1} and T_{d2}. For the flip-flop used in our study, simulations show that when T_{d2} falls below 122ps, the flip-flop ceases to function properly. This is attributed to the fact that T_{d2} has to be large enough to allow the propagation of



Mode	$\overline{N}$/T	CLK	TCLK
Normal	0	IPCLK	HIGH
Test	1	CLK2	CLK1

(a)



(b)

Figure 3.3: Generating clock and test clock. (a) Block diagram. (b) Timing diagram.

data from the master to the slave. This value of T_{d2} is equal to the worst case propagation delay of the flip-flop. The limiting value of T_{d1} is 53ps which is equal to the setup time of the flip-flop.

Chapter 4

Design of Clock Generation Circuit

The main objective of our design is to have the capability of testing high speed combinational blocks having delays as low as 400ps with a 50ps timing accuracy. Moreover, as explained before, we need to provide the ability to do performance binning in order to know how well does the DUT meet its timing specifications. To achieve these objectives, we designed the clock generation circuit such that it allows t_{comb} to be varied from 400ps to 1150ps. Referring to Equation 3.3 and considering the limiting values of T_{d1} and T_{d2} (t_{setup} and t_{prop} of the CDFF, respectively), the minimum and maximum values of $T_{d1} + T_{d2}$ are found to be 575ps and 1325ps, respectively. As stated before, T_{d1}, T_{d2}, or both can be varied to achieve these requirement. It is clear that keeping one of them constant while varying the other should save hardware required for programmable delay lines.

Two factors should be taken into consideration when choosing the values of T_{d1} and T_{d2}. Firstly, due to interconnect delays, the propagation of extremely small pulses might be difficult to achieve. Secondly, it might be difficult to maintain a very small time delay between the two clocks due to clock skew across the chip. Our implementation is designed such that T_{d1} can be varied from 275ps to 1025ps, while T_{d2} is held constant at 300ps. Although a 275ps pulse width

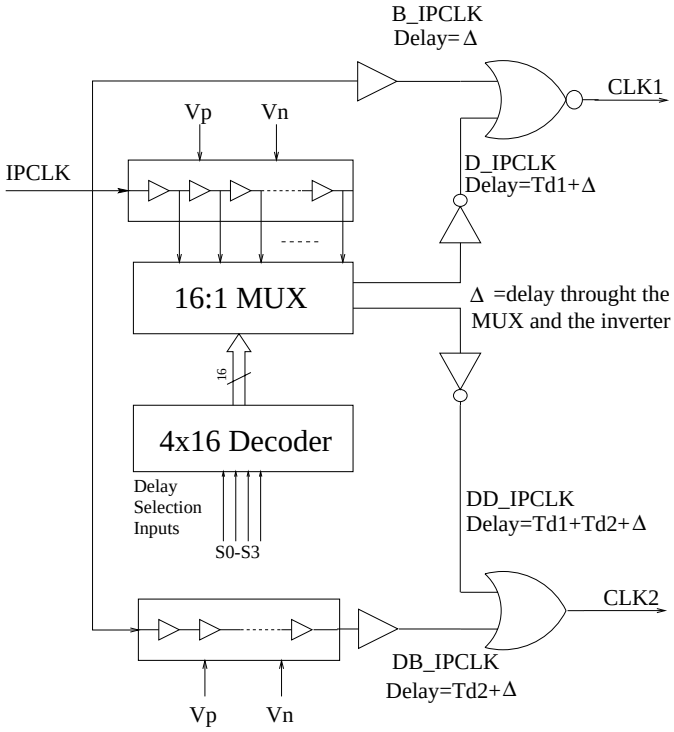
might seem very small, propagating such a small pulse is within the capability of state-of-the-art circuits. Building the TCLK driving network as a replica of the CLK driving network should help minimize the skew between the two clocks.

Two delay lines are used to generate CLK1 and CLK2 (TCLK and CLK in test mode). This is shown in Figure 4.1(a). This circuit is designed in 0.18μm CMOS technology. Figure 4.1(b) shows the signals at different points in the circuit when T_{d1} is equal to 275ps.

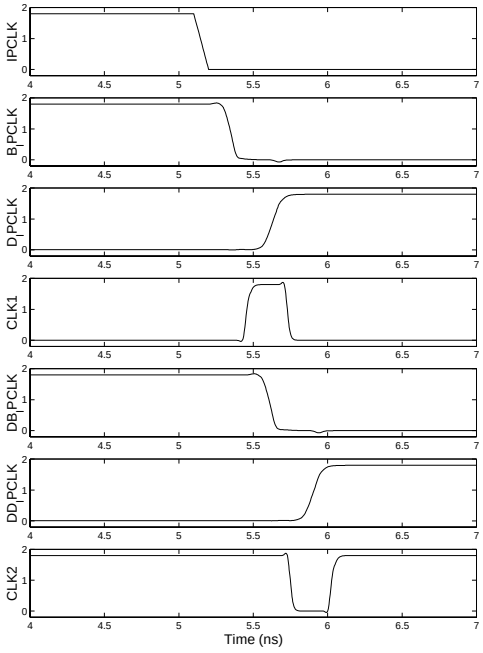
4.1 Delay Element

Each delay line consists of a chain of delay elements each having a delay of 50ps. The design of the delay element is very crucial to ensure accurate delays regardless of process, temperature, and supply voltage variations. The delay element used in our design is shown in Figure 4.2. It consists of two inverters with current control transistors M1 and M6. Referring to Figure 4.1(b), it can be shown that the delay for only the negative edge of IPCLK is critical for correct timing of CLK1 and CLK2. Therefore, the delay element is designed such that the delay is 50ps for negative going input only. This makes the sizing of transistors M3 and M4 not critical and these two transistors have close to minimum sizes. This is important to minimize the loading of the previous stage and consequently help reduce the delay for the negative going edge of the input.

The delay of the delay element is controlled by controlling the currents through transistors M1 and M6. This is done by two control voltages, V_p and V_n. If V_p and V_n are set to V_{ss} and V_{dd} respectively, currents through M1 and M6 will be maximum resulting in minimum delay for the delay element. Alternatively, if V_p and V_n are set to $V_{dd} - V_{thp}$ and $V_{ss} + V_{thn}$ respectively, where $V_{thp/n}$ is the threshold voltage of the PMOS/NMOS transistor, currents through M1 and M6 will be small resulting in a large delay. The sizes of M1 and M6 should be large enough to provide currents sufficient to achieve the required delay. Area overhead due to large control



(a)



(b)

Figure 4.1: (a) Circuit used to generate CLK1 and CLK2. (b) Signals at different points for T_{d1}=275ps.

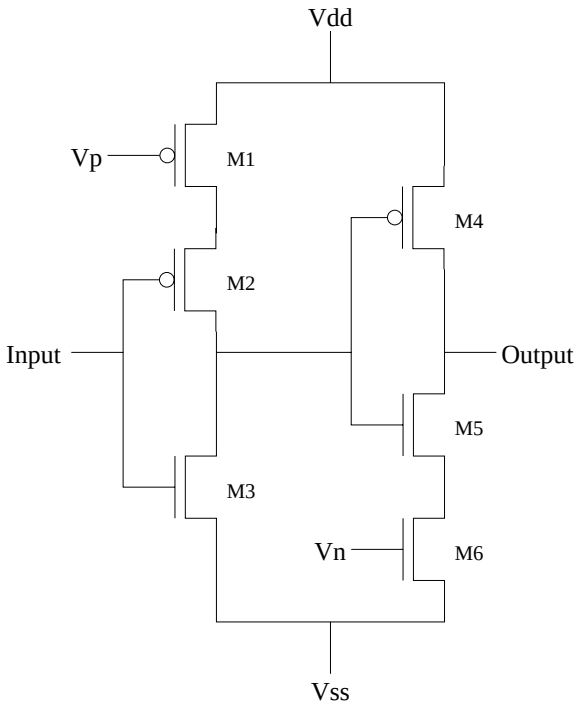


Figure 4.2: Schematic diagram of the delay element used for the delay lines.

transistors can be reduced considerably by sharing the current control transistors among multiple delay elements. In our design, only three PMOS (for M1) and three NMOS (for M6) transistors are used for all 32 delay elements used in the delay lines.

Optimum sizing of transistors M2 and M5 is important to provide sufficient currents without excessively loading the previous stage. For $V_n = V_{dd}$ and $V_p = V_{ss}$, transistor sizes are selected such that the delay element has a delay of 50ps under worst case conditions (slow-NMOS and slow-PMOS transistor models, $T = 100^oC$, and V_{dd} is 10% less than its nominal value).

4.2 Programmable Delay Line

A programmable delay line is used to generate two delayed signals, D_IPCLK and DD_IPCLK, as shown in Figure 4.1(a). The delay line consists of a chain of delay elements whose outputs are tapped and fed to a multiplexer. Two of these signals are selected by a decoder to be the outputs of the multiplexer. The delay of D_IPCLK (used to generate CLK1) is set to the required value of T_{d1}, while DD_IPCLK (used to generate CLK2) always has a fixed delay of 300ps (T_{d2}) with respect to D_IPCLK. This means that the delay chain has to be designed to have a minimum delay of 275ps ($T_{d1_{min}}$) and a maximum delay of 1325ps ($T_{d1_{max}} + T_{d2}$). Having a fixed delay between the two outputs allows the same decoder to be used to select both signals simultaneously.

Due to the propagation delay of the multiplexer, the delays of its outputs are larger than those generated from the delay chain. In order to minimize the impact it has on the timing of the delay line's outputs, the multiplexer is designed to have minimum delays. On the other hand, the design of the decoder is not timing critical. Hence, it is built with minimum size transistors.

4.3 Buffers, Gates, and Fixed-Delay Delay Line

As shown in Figure 4.1(a), IPCLK is buffered to generate B_IPCLK which is NORed with D_IPCLK to generate CLK1. A fixed-delay line is used to generate DB_IPCLK to have a delay of 300ps (T_{d2}) with respect to IPCLK. CLK2 is generated by ORing DB_IPCLK and DD_IPCLK. In order to balance and minimize their effect, the OR and NOR gates are designed to have equal delays as well as small rise and fall times. Proper sizing of the buffers is important to adjust the minimum delay of the programmable delay line as $T_{d1_{min}}$ is not a multiple of the delay element's delay ($T_{d1_{min}}$ is 275ps while the delay element's delay is 50ps). Furthermore, these buffers are crucial to compensate for the delays through the multiplexer and the inverters ($\triangle$ in Figure 4.1(a)).

Chapter 5

Test Vehicle: A 16-bit Pipelined Multiplier

Multiplication is one of the basic arithmetic operations. In fact, 8.72% of all instructions in a typical scientific program are multiplies [2]. That is why multipliers are essential building blocks in any microprocessor. Moreover, digital signal processing (DSP), image data processing, and many other applications require performing high-speed multiplications. In such cases, the multiplier determines the system performance and the operating speed. In order to verify the benefits of the technique proposed in this thesis, a 16-bit pipelined multiplier is designed and utilized as a test vehicle.

5.1 Multiplier Design

In general, parallel multiplication is done in three steps [4]. These steps are shown in Figure 5.1. In the first step, the multiplicand (M) and the multiplier (K) are used to generate partial products whose number depends on the type of encoding algorithm used. The second step is to add the partial products together in a summation network which reduces the partial products to two operands. The product is generated in the final step by adding the resulting two operands using

a carry propagate adder. For the multiplier used in this study, these steps are illustrated through the dot diagram in Figure 5.2.

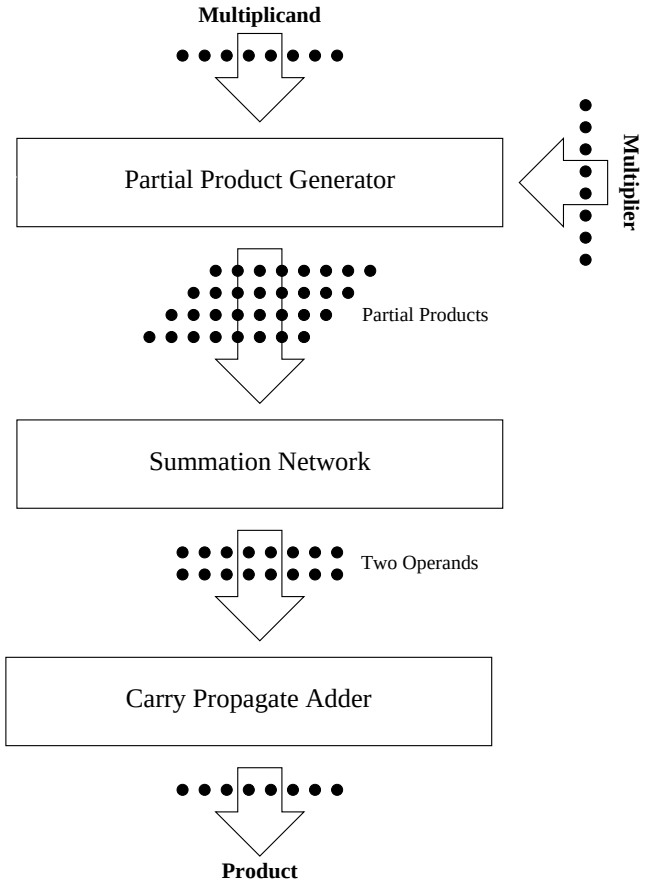


Figure 5.1: Parallel Multiplication

5.1.1 Partial Product Generation

The first step in performing multiplication is the generation of the partial products. Each partial product is a multiple of the multiplicand, based upon the value of certain bits of the multiplier. Partial product generation can be done in parallel in constant time regardless of the size of the

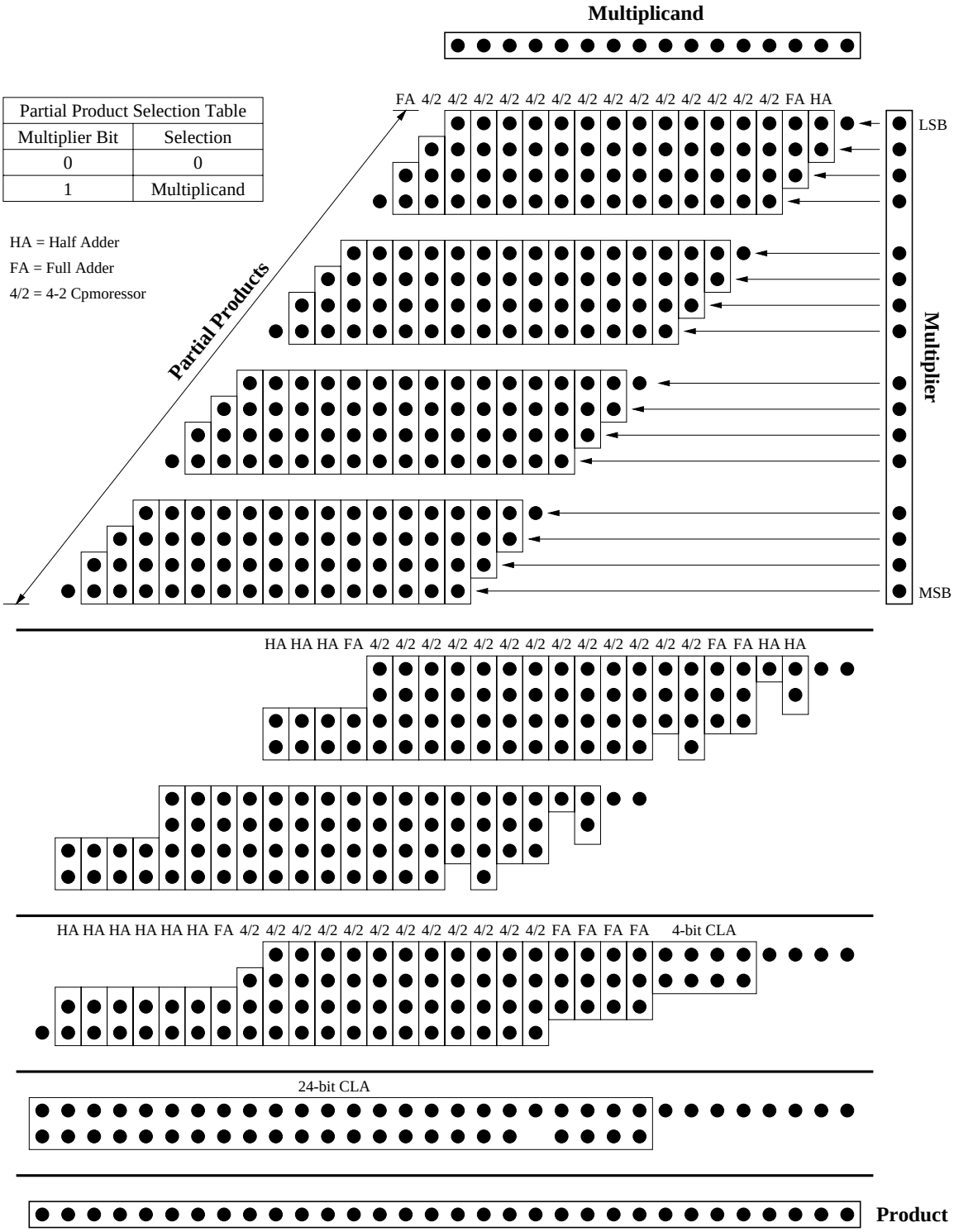


Figure 5.2: Dot diagram of the multiplier.

multiplier (assuming that the multiplicand and multiplier are available at the the inputs of the multiplier at the same time). The number of partial products depends on the encoding algorithm used for the multiplier.

In our design, no encoding is used to generate partial products. This results in a number of partial products equal to the size of the multiplier (16). This is illustrated in the dot diagram in Figure 5.2. Each partial product is represented by a row of dots. Each dot represents one bit in the partial product. As shown, the partial products are selected form the set $\{0,M\}$ depending on the corresponding bit in the multiplier. If the multiplier bit is 0, all bits in the partial product are set to zeros. Alternatively, if the multiplier bit is 1, partial product bits are set to the corresponding bits in the multiplicand. It is obvious that the partial product generation logic is a simple AND gate.

5.1.2 Summation Network

The partial products are added in the summation network using 4-2 compressors as the main component. A 4-2 compressor accepts 4 partial sums and reduces them to two [13]. In order to reduce the 16 partial products to 2 operands, three levels of 4-2 compressors are needed. This is shown in the dot diagram in Figure 5.2.

The 4-2 compressor used in our design is constructed using two full adders as shown in Figure 5.3. It has five inputs and three outputs. A 4-2 compressor accepts an input carry (C_{in}) from the cell in the lower bit position and produces an output carry (C_{out}) which is fed to the cell in the higher bit position. From the structure of the 4-2 compressor shown in Figure 5.3, it is clear that C_{out} is not a function of C_{in}. This means that the carry does not ripple though the cells of the same level in the summation network. In fact, C_{out} for all 4-2 compressors are generated in parallel. The “Sum” output of a 4-2 compressor has the same weight of the current bit position while the “Carry” output has the weight of the higher bit position.

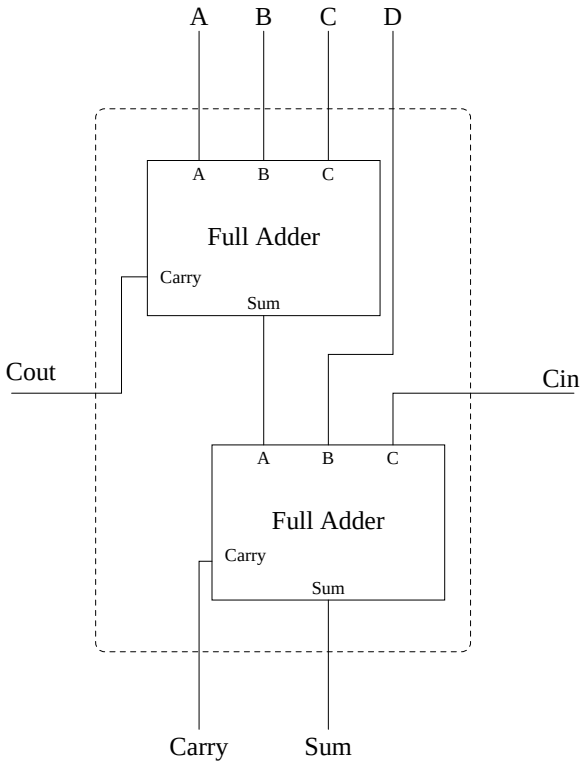


Figure 5.3: 4-2 compressor constructed with two full adders [2].

5.1.3 Carry Propagate Adder

A carry-lookahead adder with conditional sum select is used to generate the product from the two operands resulting from the summation network. The adder uses groups of four bits. For every groups, two sums are generated; one assumes a group input carry of 0, while the other assumes an input carry of 1. Three levels of carry-lookahead are used to generate all the carries. These carries are used to select the correct sum by multiplexing the two sums generated for every group.

5.2 Pipelining

The multiplier is implemented with five pipelined stages, as shown in Figure 5.4. The first stage is used to generate the 16 partial products and reduce them to 8 partial sums after the first level of the summation network. The second and third levels of the summation network are implemented in the second and third stages of the pipeline. The final addition is done in the last two stages.

Pipeline stages are separated by registers to control the timing of data flow through the multiplier. Timing analysis of the first and last stage (SN_L1 and CLA_L2 in Figure 5.4) shows that these stages can be tested for delay faults even if the input and output registers (R0 and R5) are implemented using regular flip-flops rather than CDFFs. With this arrangement, R0 is clocked using CLK in normal mode and TCLK in test mode while R5 is clocked using CLK in both modes. The other four registers are built using CDFFs. The timing diagram in Figure 5.5 shows the test mode data flow through the pipeline for two consecutive test vectors under these conditions (for simplicity, t_{setup} and t_{prop} of the flip-flops are not shown). On positive edges of TCLK, data propagates from the multiplier input through R0 to the input of SN_L1. Moreover, these edges cause data stored in registers R1, R2, R3, and R4 to propagate to the input of the next stage. Positive edges of CLK cause R1, R2, R3, and R4 to capture and hold their inputs. Same edges cause input of register R5 to propagate directly to multiplier output. Building R0 and R5 using regular flip-flops reduces the hardware associated with CDFFs without degrading the benefits of using CDFFs to enhance the testability of all pipeline stages.

5.3 Performance Characterization

Performance characterization of the multiplier is carried out in order to find its maximum operating frequency and the critical path through each stage of the pipeline. These results are shown in Table 5.1. The delays shown include the propagation delay of the register feeding the stage and

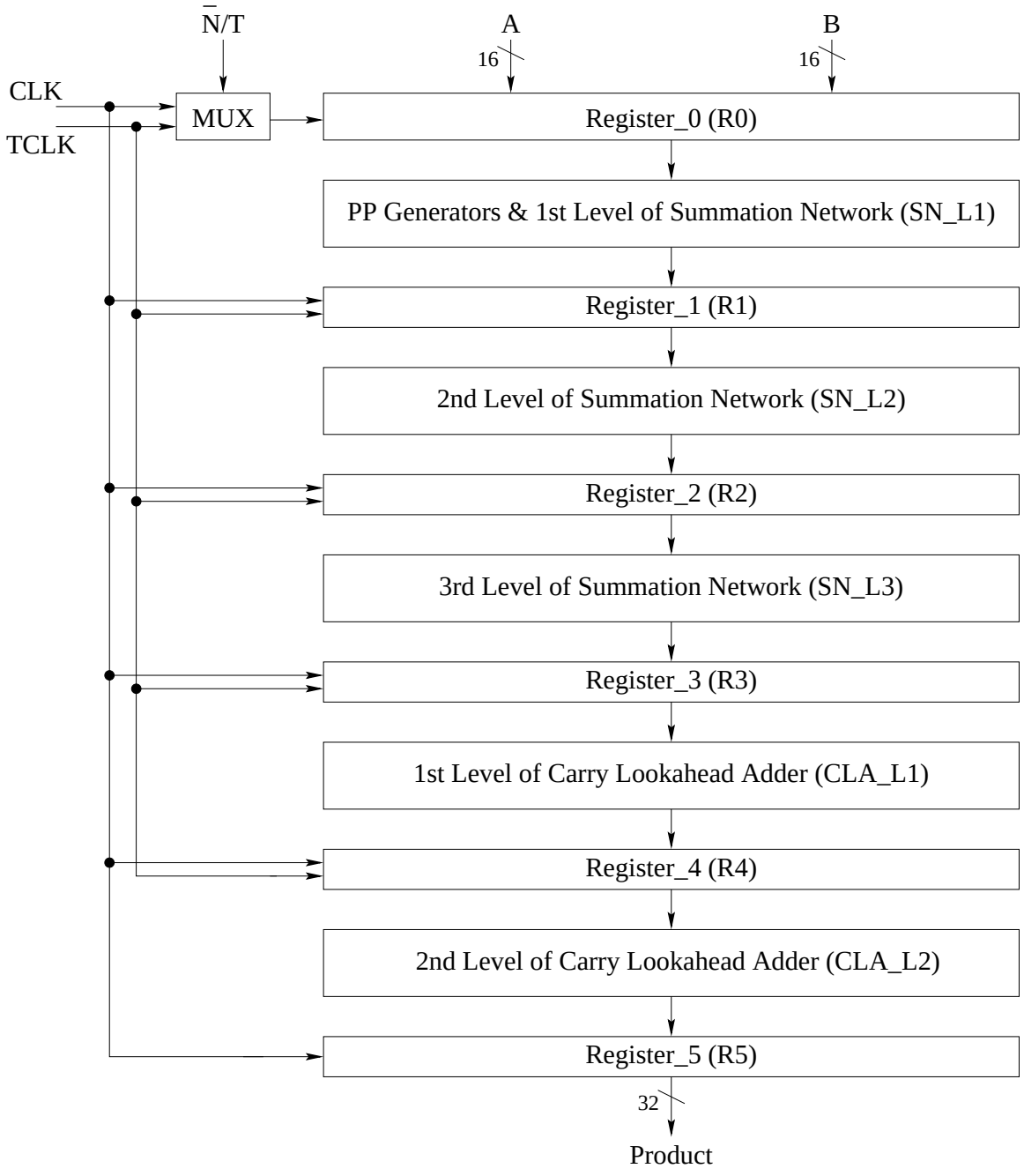


Figure 5.4: Block diagram of the pipelined multiplier used as a test vehicle.

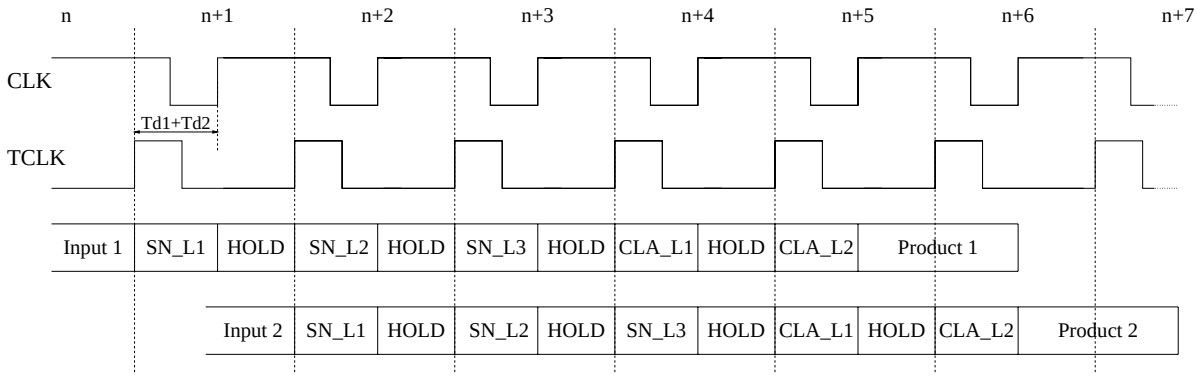


Figure 5.5: Data flow through all pipeline stages of the multiplier in test mode.

the setup time of the register accepting the output of the stage. As shown in Table 5.1, the first stage (SN_L1) has the largest delay and the operating frequency of the multiplier is determined by this stage. This delay is equal to 715ps which translates to a maximum operating frequency of 1.4GHz.

Pipeline stage	Critical path delay (ps)
SN_L1	715
SN_L2	690
SN_L3	690
CLA_L1	708
CLA_L2	645

Table 5.1: Critical path delays through multiplier stages.

Chapter 6

Simulation Results

Several simulations are carried out to test our design and to verify the possibility of reducing test mode clock frequency. Performance binning and delay fault simulations are done on the test vehicle to verify the capability of the technique to predict the maximum operating frequency of the DUT and to detect delay faults at frequencies much lower than the operating frequency.

6.1 Clock Generation Circuit

Figure 6.1 shows CLK1 for all possible combinations of decoder inputs. Simulations show that our design is capable of accurately controlling the value of T_{d1} to achieve the required design goals. The maximum deviation in the delays of the programmable delay line is $\pm$15.8% which is acceptable given the small delay values. For all values of T_{d1}, simulations show that T_{d2} is always constant at 298ps.

Signals from the clock generation circuit are used to test the CDFF at a wide range of frequencies. These results are shown in Figure 6.2 for T_{d1}=275ps and at frequencies of 100MHz, 10MHz, 1MHz, and 100kHz. These results show that any increase in the period of IPCLK is

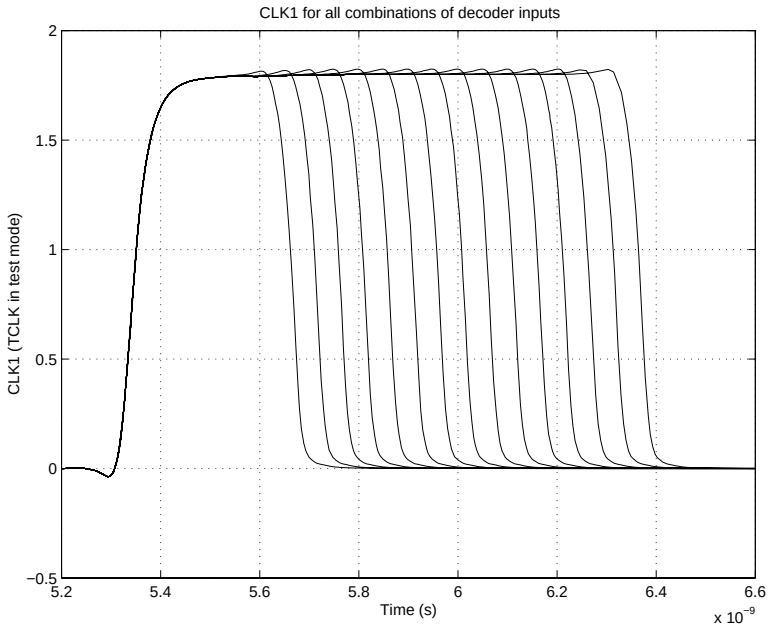
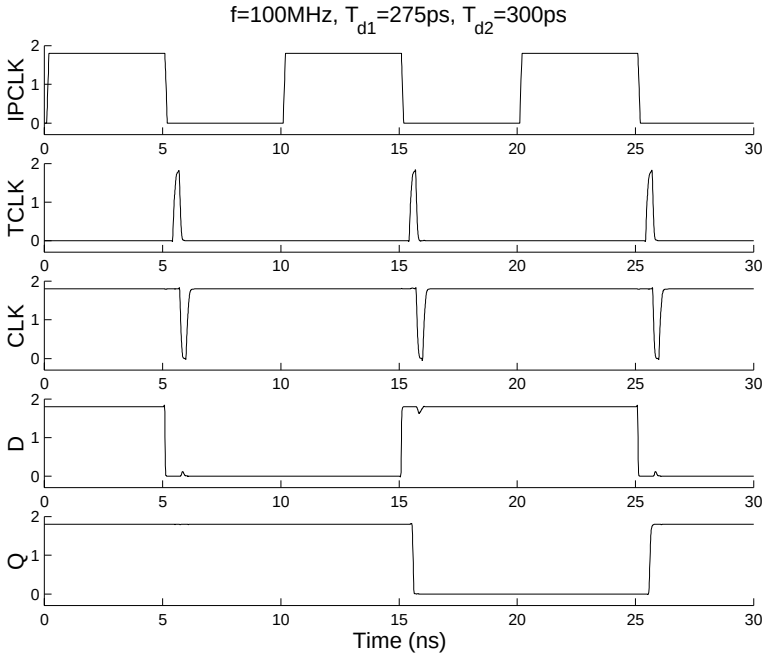


Figure 6.1: CLK1 for all possible values of S0-S3.

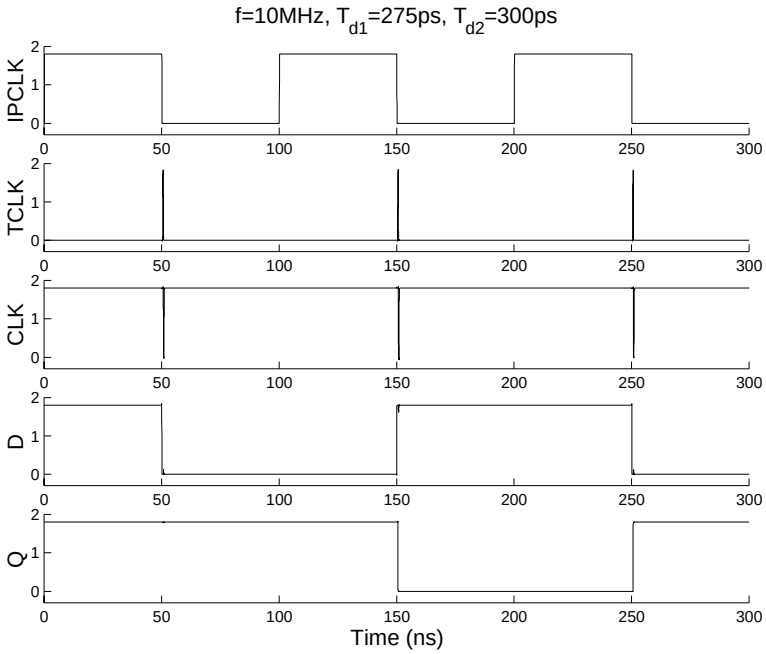
reflected as an equal increase in the CLK-Q delay of the flip-flop. This means that reducing the test mode clock frequency affects only the CLK-Q delay of the CDFF and has no effect on the time window allowed for the evaluation of the combinational block. These results show that generating CLK and TCLK using the proposed technique allows the test mode clock frequency to be reduced arbitrarily without degrading the testability of the DUT.

6.2 Performance Binning

The proposed technique can be used to do performance binning of high-speed circuits. A simple algorithm for this task is shown in Figure 6.3. As discussed before, binning can be done by changing the value of $T_{d1} + T_{d2}$. In this algorithm $T_{d1} + T_{d2}$ is varied between a minimum and a maximum value with step $(T_{d1} + T_{d2})_{step}$. DUTs are segregated in a number of bins with

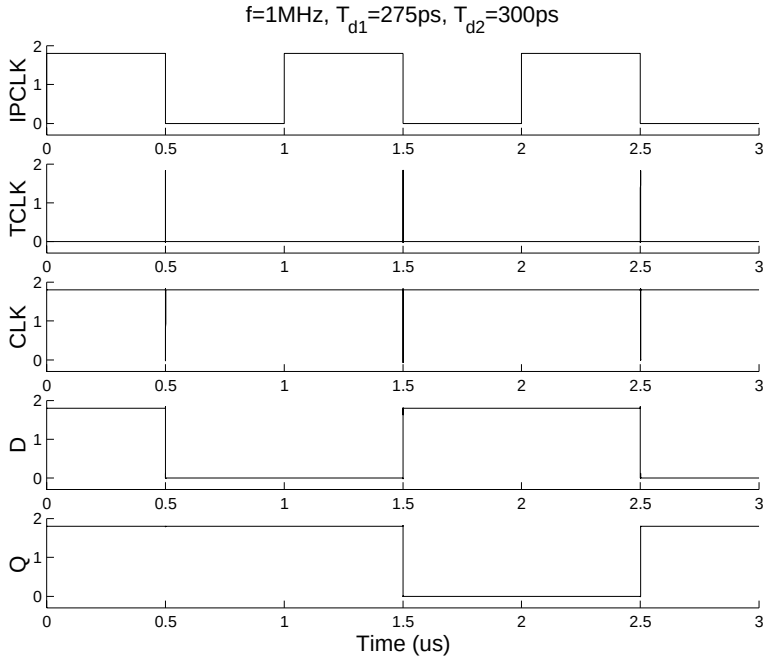


(a)

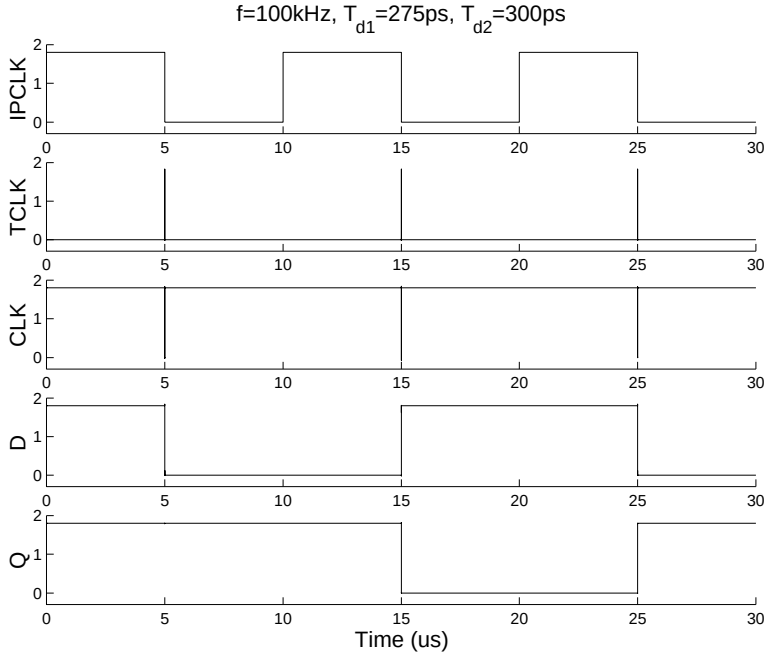


(b)

Figure 6.2: CDFF simulation results for T_{d1}=275ps. (a) f=100MHz. (b) f=10MHZ (c) f=1MHz (d) f=100kHz.



(c)



(d)

Figure 6.2: CDFF simulation results for T_{d1}=275ps. (a) f=100MHz. (b) f=10MHZ (c) f=1MHz (d) f=100kHz (*cont.*).

numbers ranging from i_{min} to $i_{max}+1$, where i_{max} is the number of steps between $(T_{d1}+T_{d2})_{min}$ and $(T_{d1}+T_{d2})_{max}$. For a given DUT, the test is first applied using the maximum value of $T_{d1}+T_{d2}$. If the device fails the test under this condition, it is placed in bin number $i_{max}+1$, which indicates that the circuit does not work at the low end of frequency range covered by the test. Otherwise, the test is repeated with a smaller value of $T_{d1}+T_{d2}$. This process continues until the DUT fails the test or the minimum value of $T_{d1}+T_{d2}$ is reached. A device passing the test with minimum value of $T_{d1}+T_{d2}$ is placed in bin i_{min}, which indicates that the DUT works at the high end of frequency range. The circuit might work at still higher frequencies. The maximum operating frequency of devices in bin i_{min} can be found through a clock generation circuit capable of generating even smaller values of $T_{d1}+T_{d2}$. This discussion shows that the choice of the minimum and maximum values of $T_{d1}+T_{d2}$ is critical. These values determine the range of operating frequencies which can be covered when doing performance binning using this technique.

As an example of applying the algorithm in Figure 6.3, performance binning is performed on the pipelined multiplier using CLK and TCLK from the clock generation network. Hence, the values of i_{min} and i_{max} are 0 and 15 respectively. Simulations are carried out under fault-free conditions. Performance binning simulations are first done using typical process models. These simulations are then repeated for all process corners. As stated in section 6, for typical process models, SN_L1 has the maximum critical path delay among all stages of the pipeline. For simplicity, we assume that this remains the case for all process corners. Hence, binning is carried out using test vectors that activate the critical path through SN_L1.

Performance binning results are given in Table 6.1. The value of T_{d1} given in the table is the minimum value allowing the multiplier to produce correct output. For typical process models, these results show that the minimum value of $T_{d1}+T_{d2}$ necessary for correct operation is 725ps. This value is slightly higher than the 715ps given in Table 5.1. This is attributed to the finite

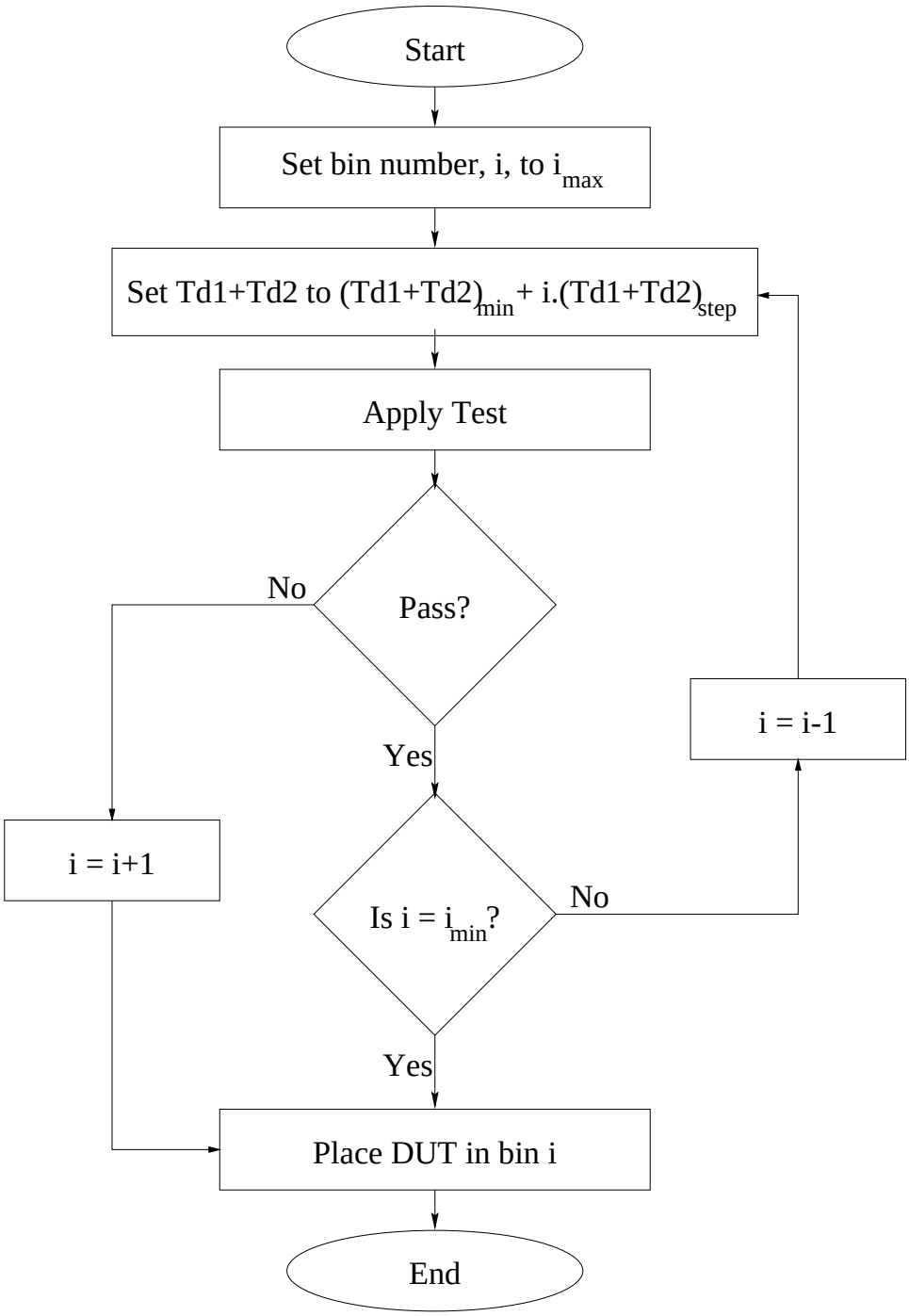


Figure 6.3: Algorithm for performance binning using the proposed technique.

Process Models	T_{d1} (ps)	$T_{d1} + T_{d2}$ (ps)	Bin #
Typical	425	725	3
Fast NMOS, fast PMOS	275	575	0
Fast NMOS, slow PMOS	425	725	3
Slow NMOS, fast PMOS	425	725	3
Slow NMOS, slow PMOS	625	925	7

Table 6.1: Performance binning results for various process models.

resolution of the clock generation circuit as it can only predict the performance to the closest 50ps. Table 6.1 also shows that for fast NMOS and fast PMOS models, the multiplier functions properly even for the minimum value of T_{d1}.

6.3 Delay Fault Simulation

Delay fault simulations are carried out using typical process models. Delay faults are inserted in the test vehicle one at a time using buffers. Using active elements to generate extra delays ensures the existence of a pre-determined delay fault without degrading the quality (rise and fall times) of the delayed signals. Using performance binning results for typical process models given in Table 6.1, T_{d1} is always set to 425ps. Figure 6.4 shows the algorithm used for fault simulation. Delay faults are inserted in 50ps increments until a value causing malfunction of the DUT is reached. Delay faults are inserted in the critical path as well as one quasi-critical path in all stages of the pipeline except SN_L3. This is because of the similarity between this stage and SN_L2 in terms of their structure and critical path characteristics.

For every target path, two vectors are used to test the circuit. The first vector initializes the DUT while the second vector activates the faulty path of the stage to be tested. Initialization

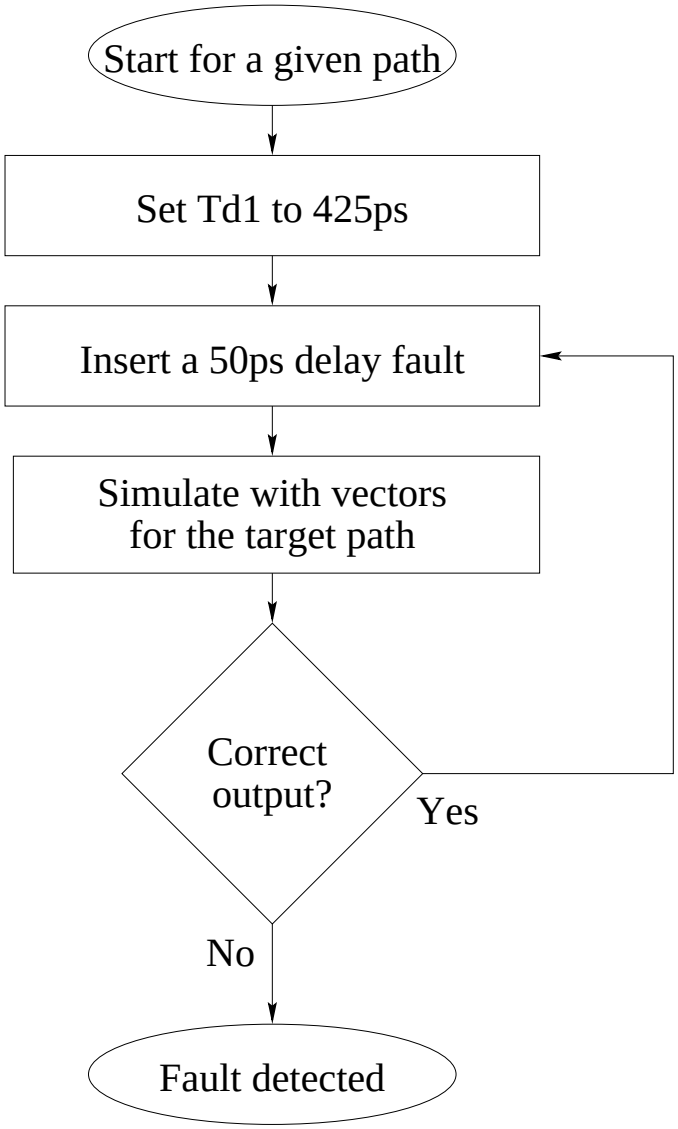
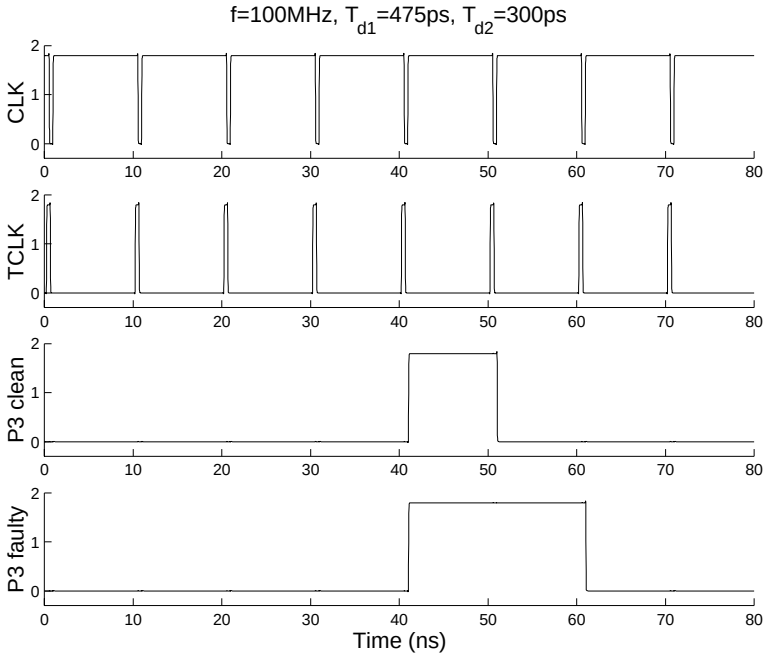


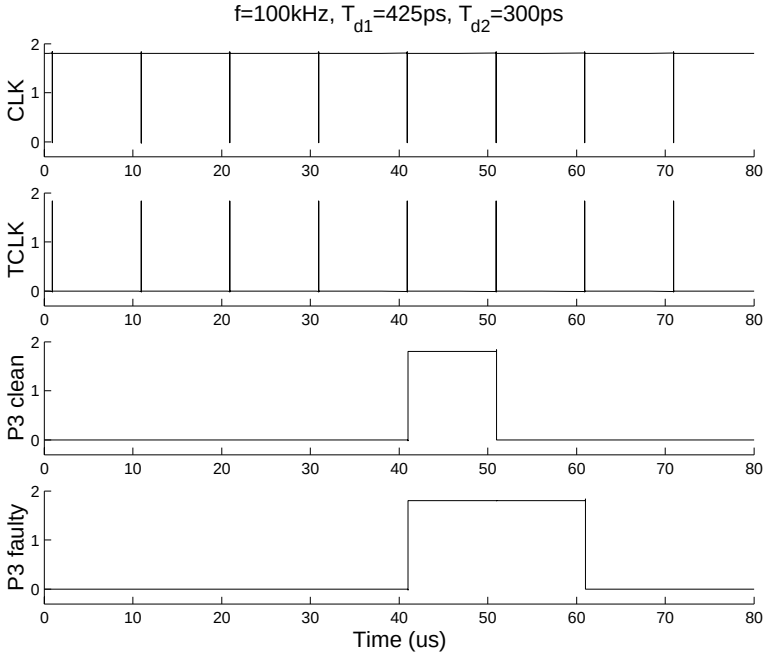
Figure 6.4: Algorithm for delay fault simulation.

vectors are chosen such that they always result in correct output even under faulty conditions. For correct multiplier operation, activation vectors should result in correct output with the sixth rising edge of CLK (refer to Figure 5.5). For a faulty circuit, failure to get the correct output at this edge of CLK means that we are able to detect the fault. The test is done at two frequencies; 100MHz and 100kHz. Figure 6.5(a) and (b) show examples of test results at these two frequencies for a 50ps delay fault in the critical path of SN_L1. These graphs show CLK, TCLK and P3 for fault-free and faulty SN_L1 operation (P3 is the only bit of the product affected by the delay fault in the critical path of SN_L1). As shown, we are able to detect the delay fault at both frequencies. Similar results are obtained for all other stages.

Table 6.2 shows delay fault simulation results for all paths tested in our study. The left half of the table gives the delays of the different paths and the amount of delay fault that has to be inserted in the target path in order to cause the DUT to malfunction. The right half of the table gives the test vectors used for each path as well as the fault-free and faulty product of the multiplier. As is apparent from column 3 and 4, the extent of delay fault that goes undetected is a function of the delay (slack) in a given path. A path with smaller delay will have a larger undetectable delay fault. For example, in the case of path # 6, the path delay is 460ps. It will take a delay fault of 300ps to cause a timing failure. On the other hand, for path # 1, it will take a delay fault of only 50ps to cause the timing failure. In general, delay fault detection is dependent on the target path delay. Most of the delay fault testing techniques have similar limitations. Balancing path delays is the most commonly used method to alleviate this problem.



(a)



(b)

Figure 6.5: Fault simulation for the critical path of SN⊥1 (a) f=100MHz. (b) f=100kHz.

Path #	Pipeline stage	Path delay (ps)	Delay fault (ps)	Vector type	Input A	Input B	Product (fault-free)	Product (faulty)
1	SN_L1	715	50	Init.	0002	FFFF	0001 FFFE	0001 FFFE
				Activ.	0000	FFFF	0000 0000	0000 0008
2	SN_L1	670	100	Init.	000C	FFFF	000B FFF4	000B FFF4
				Activ.	0008	FFFF	0007 FFF8	0007 FFF0
3	SN_L2	690	50	Init.	0001	FFFF	0000 FFFF	0000 FFFF
				Activ.	0000	FFFF	0000 0000	0000 0040
4	SN_L2	665	100	Init.	0070	FFFF	006F FF90	006F FF90
				Activ.	0040	FFFF	003F FFC0	006F FF80
5	CLA_L1	708	50	Init.	0000	FFFF	0000 0000	0000 0000
				Activ.	FFFF	FFFF	FFFE 0001	FFFD 0001
6	CLA_L1	460	300	Init.	FFFF	FFFF	FFFE 0001	FFFE 0001
				Activ.	0000	FFFF	0000 0000	0001 0000
7	CLA_L2	645	100	Init.	0000	FFFF	0000 0000	0000 0000
				Activ.	FFF0	FFFF	FFD0 0200	FFC0 0200
8	CLA_L2	450	300	Init.	FFF0	FFFF	FFD0 0200	FFD0 0200
				Activ.	0000	FFFF	0000 0000	0010 0000

Table 6.2: Delay fault simulation results.

Chapter 7

Conclusions and Future Work

We have presented a DFT technique for testing high-speed circuits with few requirements on ATEs. The technique uses CDFFs to control the delay of the DUT in order to facilitate its testing. This technique allows such circuits to be tested with arbitrarily low-frequency, 50% duty cycle input clock.

The circuit used to generate CLK and TCLK in test mode has been presented along with various design and implementation issues. Simulation results show that using this design, pulse widths as small as 275ps for TCLK can be efficiently achieved. The accuracy of the design is 50ps which is 3-4 times better than the state-of-the-art ATE.

Simulations of the CDFF with clocks from our design show the validity of this methodology. These simulations prove that the test mode clock frequency can be reduced with no lower limit while keeping the time window allowed for DUT evaluation constant.

We tested the proposed technique with a 16-bit pipelined multiplier as a test vehicle. Testing results show that performance binning of the DUT can be carried out using a simple algorithm. Fault simulation results verify the ability of the technique to detect delay faults using a clock frequency much lower than the rated frequency.

There are several areas of improvement and extension for this work. Using two clocks for the CDFF means that we have to build two clock driving networks. Redesgning the flip-flop in a way that eliminates the need for two clocks would help in reducing the circuit overhead and simplify the design process.

The current design manily targets delay faults in the combinational blocks of a sequential circuit. Extending the technique to cover the sequential blocks as well as the clock driving network(s) would be a major achievements towards providing a comprehensive delay fault coverage for all parts of the DUT.

Finally, it is essential to develop techniques to ensure the functionality and performance of the clock generation network. This is particularly important as the detectability of delay faults in the DUT depends mainly on the correct timing of the clocks generated form the clock generation network.

In conclusion, the main contributions of this research are:

- eliminating the restriction on lowering the test mode clock frequency for circuits using CDFFs to improve testability.
- developing a simple on-chip clock generation methodology that can achieve a state-of-the-art resolution regardless of the timing accuracy of the tester.

Glossary of Terms

ATE – Automatic Test Equipment.

BIST – Built-in-self-test.

CDFF – Controlled-Delay Flip-Flop.

CMOS – Complementary Metal Oxide Semiconductor.

DFT – Design-for-Testability.

DUT – Device Under Test.

ITRS – International Technology Roadmap for Semiconductors.

VLSI – Very Large Scale Integration.

References

[1] V. D. Agrawal and T. J. Chakraborty. “High-Performance Circuit Testing with Slow-Speed Testers”. *Proc. of International Test Conference*, pages 302–310, 1995.

[2] Hesham A. Al-Twaijry. *“Area and Performance Optimized CMOS Multipliers”*. PhD thesis, The Department of Electrical Engineering, Stanford University, August 1997.

[3] Semiconductor Industry Association. “International Technology Roadmap for Semiconductor, 1999 Edition”, 1999.

[4] Gary W. Bewick. *“Fast Multiplication: Algorithms and Implementation”*. PhD thesis, The Department of Electrical Engineering, Stanford University, February 1994.

[5] Jonathan T.-Y. Chang and Edward McClusky. “Detecting Delay Flaws by Very-Low-Voltage Testing”. *Proc. of 1996 International Test Conference*, pages 367–376, June 1996.

[6] Jonathan T.-Y. Chang and Edward McClusky. “Quantitative Analysis of Very-Low-Voltage Testing”. *14th VLSI Test Symposium*, pages 332–337, April 1996.

[7] Robert J. Feugate and Steven M. McIntyre. *“Introduction to VLSI Testing”*. Prentice Hall, 1988.

[8] P. Franco. “Testing Digital Circuits for Timing Failures by Output Waveform Analysis”. Technical Report 94-9, Center for Reliable Computing, Stanford Univerity, September 1994.

[9] H. Hao and E. J. McCluskey. “Very-Low-Voltage Testing for Weak CMOS Logic ICs”. *Proc. of International Test Conference*, pages 275–284, 1993.

[10] Charles F. Hawkins, Jerry M. Soden, Alan W. Righter, and F. Joel Feruson. “Defect Classes - An Overdue Paradigm for CMOS IC Testing”. *Proc. of 1994 International Test Conference*, pages 413–425, February 1994.

[11] Way Kuo, Wei-Ting K. Chien, and Taeho Kim. *“Reliability, Yield, and Stress Burn-in”*. Kluwer Academic Publishers, 1998.

[12] Parag K. Lala. *“Digital Circuit Testing and Testability”*. Academic Press, 1997.

[13] M. Mehta, V. Parmar, and Jr. E. Swartzlander. “High-Speed Multiplier Design Using Multi-Input Counter and Compressor Circuits”. *Proc. of the 10th IEEE Symposium on Computer Arithmetic*, pages 43–50, 1991.

[14] Frank Peters and Steven Oostdijk. “Realistic Defect Coverage of Voltage and Current Tests”. *Digest of papers, 1996 IEEE International Workshop on IDDQ Testing*, pages 4–8, October 1996.

[15] Rochit Rajsuman. *“Iddq Testing for CMOS VLSI”*. Artech House Inc., 1995.

[16] Manoj Sachdev. *“Defect Oriented Testing for CMOS Analog and Digital Circuits”*. Frontiers in Electronic Testing. Kluwer Academic Press, 1998.

[17] Sanjay Sengupta, Sandip Kundu, Sreejit Chakravarty, Praveen Parvathala, Rajesh Galivanche, George Kosonocky, Mike Rodgers, and TM Mak. "Defect-Based Test: A Key Enabler for Successful Migration to Structural Test". *Intel Technology Journal*, Q1, 1999.

[18] M. Shashani and M. Sachdev. "A DFT Technique for High-Performance Circuit Testing". *Proc. of International Test Conference*, pages 267–285, 1999.

[19] Adit D. Singh, Haroon Rasheed, and Walter W. Weber. "IDDQ Testing of CMOS Opens: An Experimental Study". *Proc. of 1995 International Test Conference*, pages 479–489, 1995.

[20] H. Speek, H. G. Kerkhoff, M. Sachdev, and M. Shashaani. "Bridging the Test Speed Gap: Design for Delay Testability". *Proc. of the IEEE European Test Workshop*, pages 3–8, 2000.

[21] H. T. Vierhaus, W. Meyer, and U. Glser. "CMOS Bridges and Resistive Transistor Faults: IDDQ versus Delay Effects". *Proc. of 1993 International Test Conference*, pages 83–91, October 1993.

Bibliography

[1] Dimitri Kececioglu and Feng-Bin Sun. *"Burn-In Testing, its Quantification and Optimization"*. Prentice Hall PTR, 1997.

[2] A. Keshavarzi, K. Roy, and C. F. Hawkins. "Intrinsic Leakage in Low Power Deep Submicron CMOS ICs". *Proc. of International Test Conference*, pages 146–155, 1997.

[3] G. Moyer, M. Clements, W. Lui, T. Schaffer, and R. Cavin, III. "The Delay Vernier Pattern Generation Technique". *IEEE Journal of Solid-State Circuits*, 32(4):551–562, April 1997.

[4] W. Needham, C. Prunty, and E. H. Yeoh. "High Volume Microprocessor Test Escapes, An Analysis of Defects Our Tests are Missing". *Proc. of International Test Conference*, pages 25–34, 1998.

[5] P. Nigh, W. Needham, K. Butler, P. Maxwell, R. Aitken, and W. Maly. "So What is an Optimal Test Mix? A Discussion of The SEMATECH Methods Experiment". *Proc. of International Test Conference*, pages 1037–1038, 1997.

[6] P. Nigh and A. Gattiker. "Test Method Evaluation Experiments and Data". *Proc. of International Test Conference*, pages 454–463, 2000.

[7] M. Nummer and M. Sachdev. "A Methodology for Testing High-Performance Circuits at Arbitrarily Low Test Frequency". *IEEE VLSI Test Symposium*, pages 68–74, April 2001.

[8] N. Ohkubo, M. Suzuki, T. Shinbo, T. Yamanaka, A. Shimizu, K. Sasaki, and Y. Nakagome. "A 4.4 ns CMOS 54×54-b Multiplier Using Pass-Transistor Multiplexer". *IEEE Journal of Solid-State Circuits*, 30(3):251–257, March 1995.

[9] M. Sachdev. "Deep Sub-micron IDDQ Testing: Issues and Solutions". *Proc. of European Design and Test Conference*, pages 271–278, 1997.

[10] H. Speek, H. G. Kerkhoff, M. Shashaani, and M. Sachdev. "A Low-Speed BIST Framework for High-Performance Circuit Testing". *Proc. of the 18th IEEE VLSI Test Symposium*, pages 349–355, 2000.

[11] T. W. Williams, R. H. Dennard, R. Kapur, M. R. Mercer, and M. Maly, M. . "I_{DDQ} Test: Sensitivity Analysis of Scaling". *Proc. of International Test Conference*, pages 786–792, 1996.