ATPG SIMULATION AND PATTERN GENERATION

A Thesis Submitted in Partial Fulfilment of the Requirements for the Award of the Degree of

Master of Technology

In VLSI Design

Submitted By

DAVINDER SINGH

601762005

Under Supervision of
Dr. GAGANPREET KAUR
Lecturer

ELECTRONICS AND COMMUNICATION ENGINEERING DEPARTMENT
THAPAR INSTITUTE OF ENGINEERING & TECHNOLOGY
(A DEEMED TO BE UNIVERSITY), PATIALA, PUNJAB

JULY, 2019

STMicroelectronics INDIA PVT. LTD.

Greater Noida, Uttar Pradesh 201308, India

Date: May 30, 2019

CERTIFICATE

This is to certify that **Davinder Singh (601762005),** a student of M.Tech (VLSI), Thapar Institute of Engineering & Technology, Patiala, has successfully completed one year (June 2018 – May 2019) internship program in **STMicroelectronics Pvt. Ltd., Greater Noida**. His title of the dissertation is **"ATPG Simulation and Pattern Generation".**

During the period of his internship program, he was punctual and hardworking. I wish him every success in life.

Shiv Kumar VATS

Senior Staff Engineer,

STMicroelectronics Pvt. Ltd.,

Greater Noida, India

DECLARATION

I, **Davinder Singh** hereby declare that the work presented in this thesis entitled **"ATPG Simulation and Pattern Generation"** in partial fulfillment of the requirement for the award of degree of **Master of Technology (VLSI Design)** submitted at **Electronics and Communication Department, Thapar Institute of Engineering & Technology, Patiala** is an authentic record of work carried out under supervision of **Dr. Gaganpreet Kaur (Lecturer (Ph.D.), Electronics and Communication Department. Thapar Institute of Engineering & Technology, Patiala)** from June 2018 to July 2019. The matter presented in this has not been submitted either in part or full to any other university or institute for the award of any other degree.

Date: 12/07/2019

Davinder Singh

601762005

Dr. Gaganpreet Kaur

Lecturer (Ph.D.)

Department of Electronics and Communication Engineering

Thapar Institute of Engineering & Technology

Patiala, Punjab

Date:

ACKNOWLEDGMENT

The success and final outcome of this project required a lot of guidance and assistance from many people and I am extremely privileged to have got this all along with the completion of my project. All that I have done is only due to such supervision and assistance and I would not forget to thank them.

I respect and thank **Mr. Sabyasachi Das**, for providing me an opportunity to do the project work in STMicroelectronics Pvt. Ltd. and giving me all support and guidance, which made me complete the project duly. I am extremely thankful to him for providing such a nice support and guidance, although he had a busy schedule managing the corporate affairs.

I owe my deep gratitude to the project guides **Mr. Shiv Kumar Vats, Mr. Deepen S. Talati,** and **Mr. Arjun Singh**, who took a keen interest on the project work and guided me all along, till the completion of the project work by providing all the necessary information for developing a good system.

I heartily thank my internal project guide, **Dr. Gaganpreet Kaur**, Lecturer, Dept. of ECE for her guidance and suggestions during this project work.

I am thankful for and fortunate enough to get constant encouragement, support and guidance from HOD Dr. **Alpana Agarwal** and all Teaching staff of Dept. of ECE which helped me in successfully completing my project work.

Davinder Singh
Reg. No.:601762005
Dept. of ECE
Thapar Institute of Engineering Technology
Patiala, Punjab

ABSTRACT

Because of the technological advancements in the semiconductor industry, we are now able to design ICs with as many as billion transistors while improving the performance and power. However, new challenges arrive with new technologies. The complexity increases with new technologies, and with shrinking size transistors access to internal nodes has reduced and it is now more difficult to diagnose and locate faults. The solution to this problem is designing a testable design so that they function reliably throughout their lifetime. Considering the time and money put in the development of an IC, it becomes very crucial to test it and get a maximum possible yield before handing it over to the customer. This is why around 40% of overall product cost consists of testing of chip. The field responsible for making design testable at an early stage is Design for Testability (DFT). This report explains DFT in detail, techniques involved in carrying out DFT process, like scan insertion, pattern generation and simulating those patterns on netlist.

ABOUT THE ORGANISATION

STMicroelectronics is a result of the merger of two companies namely SGS Micro electronica of Italy and Thomson Semiconductors of France in 1987. It was formerly named as SGS Thomson soon after the merger but later was renamed as STMicroelectronics in 1998. It is commonly known as ST, currently having its headquarters in Geneva, Switzerland. STMicroelectronics is one of the leading semiconductor industries in the world with a revenue of $9.66 Billion in 2018. STMicroelectronics has a broad product Portfolio ranging from sense and power technologies, automotive products, to IOT (Internet of Things) solutions. The company has a Corollas manufacturing site which originally was a common research and development center. The Grenoble site houses various company's divisions such as the silicon and Software design fabrication process R&D (research and development). STMicroelectronics has various centers all over the world. In India alone, the company has 4 operational sites.

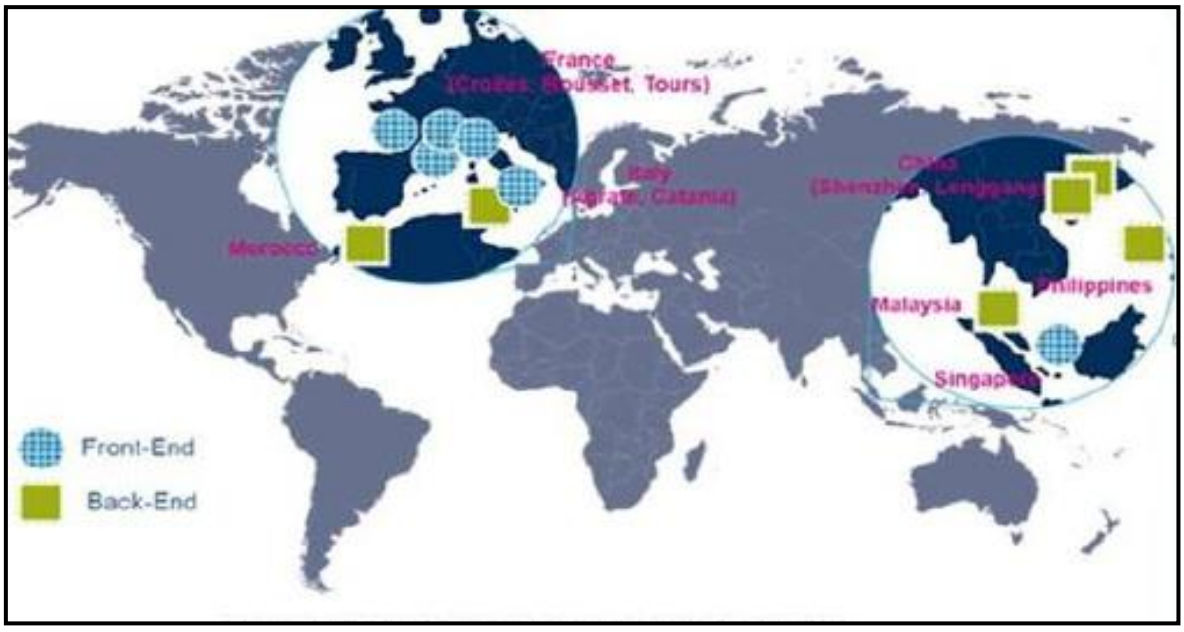


Figure 01.1-1 STMicroelectronics - About the Organization

The present CEO and president of STMicroelectronics are Jean-Marc Chery with Nicolas Dufourcq as the acting chairman of the organization. At present ST has over 46000 employees, 12 manufacturing sites, advanced R&D teams in about 10 countries and sales of

units spread across the world. ST operates front end wafer fabrication labs and back-end assembly plants, test plants and packaging plants. ST has fabrication labs located at many places in Italy and France.

STMicroelectronics has world-class work facilities in places like China, Malaysia, Philippines, India Singapore, and Morocco. STMicroelectronics brought a revolution in the semiconductor business with the introduction of FD-SOI technology in 2012 and building innovative products using the new technology. The TR&D (technology research and development) group now known as the TDP (Technology and design platform) works on intellectual properties with the aim of providing library solution and quality services to various divisions. TDP helps the company in its drive-in time to market, increasing intellectual property reuse and flawless and reliable integration.

CONTENTS

CERTIFICATE **Error! Bookmark not defined.**

DECLARATION **Error! Bookmark not defined.**

ACKNOWLEDGMENT iv

ABSTRACT v

ABOUT THE ORGANISATION vi

CONTENTS viii

LIST OF FIGURES xi

CHAPTER - 1 1

INTRODUCTION 1

1.1 INTRODUCTION TO DFT 1

1.2 SIGNIFICANCE OF DFT 1

1.2.1 Types of Testing 1

1.2.2 Test Plan 2

1.2.3 Fault and Fault Modelling 3

1.2.4 Fault Equivalence and Fault Dominance 3

1.2.5 Types of Faults 5

1.3 DFT TECHNIQUES 8

1.4 LITERATURE REVIEW 11

CHAPTER – 2 14

SCAN BASED ATPG 14

2.1 INTRODUCTION 14

2.2 SCAN CELLS 14

2.2.1 Working of Scan Cell 15

2.3 SCAN CHAINS 15

2.4 SCAN BASED ATPG v/s BIST 16

2.5 SCAN INSERTION 17

2.5.1 Partial Scan 17

2.5.2 Scan Clock 18

2.5.3 Controllability and Observability 18

2.6 SCAN OPERATION 20

CHAPTER - 3 22

ATPG 22

3.1 ATPG 22

3.1.1 Netlist 22

3.1.2 ATPG Library Files 22

3.1.3 ATPG Model 23

3.1.4 ATPG DRC Check 23

3.1.5 Generate Fault List 23

3.1.6 Generate Test Patterns 23

3.1.7 Save and Compress Test Patterns 23

3.2 ATPG APPROACH 23

3.3 ATPG PROCESS 24

3.4 PATTERN GENERATION COMMANDS 25

3.5 SAMPLE COMMAND FILE 30

3.6 Issues During Pattern Generation 31

CHAPTER - 4 33

SIMULATION RESULTS 33

4.1 INTRODUCTION TO SIMULATION 33

4.2 SIMULATION IN DFT 33

4.2.1 Compilation 33

4.2.2 Elaboration 33

4.2.3 Simulation 34

4.3 TYPES OF SIMULATIONS 34

4.3.1 Zero Delay (notim) Simulation 34

4.3.2 Best Case Simulation 34

4.3.3 Worst Case Simulation 34

4.4 SIMULATION USING REGRESSION SETUP 35

4.5 DEBUGGING SIMULATION FAILURES 35

CHAPTER – 5 39

CONCLUSION AND FUTURE SCOPE 39

5.1 CONCLUSION 39

5.2 FUTURE SCOPE 39

CHAPTER - 6 40

REFERENCES 40

6.2 JOURNALS 40

6.3 WEBSITES 40

LIST OF FIGURES

Figure 01.1-1 STMicroelectronics - About the Organization---------- vi
Figure 1.2-1Fault Equivalence and Fault Collapsing ---------- 4
Figure 1.2-2 Fault Dominance---------- 5
Figure 1.2-3 Transition Fault Example---------- 6
Figure 1.2-4 Stuck-at Fault Example ---------- 7
Figure 1.3-1 BIST Block Diagram---------- 9
Figure 1.3-2 LFSR ---------- 9
Figure 2.2-1 Comparison Between D Flip Flop and Scan Cell----------14
Figure 2.2-2 A Scan Cell ----------15
Figure 2.3-1 Scan Chain Connections Through Combinational Circuit ----------16
Figure 2.5-1 Not Controllable Node----------18
Figure 2.5-2 Node Controlled Using Scan Cell----------19
Figure 2.5-3 Not Observable Node ----------19
Figure 2.5-4 Node Observable Using Scan Cell----------20
Figure 2.6-1 Scan Chain Operation----------21
Figure 3.1-1 ATPG Flow----------22
Figure 3.4-1 STIL Procedure File Format ----------26
Figure 3.4-2 ATPG Analyzation ----------28
Figure 3.4-3 Test Coverage Results ----------29
Figure 3.4-4 Command File showing write pattern command ----------30
Figure 3.6-1 Coverage File Generated ----------32
Figure 3.6-2 Successful Pattern Generation ----------32
Figure 4.4-1 Regression Guide File ----------35
Figure 4.5-1 Log File Showing Mismatch Error ----------36
Figure 4.5-2 Unsuccessful Simulation With Mismatches ----------37
Figure 4.5-3 Log File For Successful Simulation ----------37
Figure 4.5-4 Dumped Waveform With Simulation ----------38

CHAPTER - 1

INTRODUCTION

1.1 INTRODUCTION TO DFT

Design for testability (DFT) in modern technologies is very important in VLSI industries. In this era of technology, industries focus more on the advancement of DFT technologies. It is getting very important to test a design with more accuracy before the fabrication starts. DFT has made its necessary place. The scale of electronics is more than a hundred million transistors per square millimeter. Hence, the manufacturing of these chips has become very complex and expensive. If there is any malfunctioning in the design it can cause a huge loss. So, testing is necessary at almost every level of the ASIC design flow. DFT will make sure that the design is going to work in every case.

1.2 SIGNIFICANCE OF DFT

The rapid advancement in integration technologies enables us to fabricate millions of transistors in a single integrated chip (IC) or chip. Integration of millions of transistors on one chip is not an easy task. System of systems can be implemented on a single small chip.

However, this leads us to bigger design problems and complexity. This leads to manufacturing defects and all the chips need to be physically tested by giving input signals from a pattern generator and comparing the output using a logic analyzer. This process is called testing.

Any VLSI IC can be divided into three categories

a) Analog IC
b) Digital IC
c) Mixed-signal IC

In mixed signal IC both analog and digital designs are fabricated on one single chip. Majorly ICs are mixed-signal ICs. But CAD tools are designed for the testing of digital designs.

Testing of the design before fabrication is necessary and time-saving. It can also save money, as fabrication cost is very high and if fabrication is done without testing the design then it can cause a huge loss. Once a design is fabricated on a chip then it cannot be changed. So, design for testability is a must to do the process in VLSI.

1.2.1 Types of Testing

Testing can be widely done at the system level, chip level, and board level. Along with these, it can be done at the following levels.

a) Transistor level
b) Gate level

c) RTL level
d) Functional level

Testing types can be widely divided into the following three categories

a) Characterization testing
b) Manufacturing testing
c) Acceptance testing

A. Characterization testing: In characterization testing, the design characteristics are tested in different operating conditions such as:

a) No delay
b) Best case delay
c) Worst case delay

B. Manufacturing testing: Manufacturing test covers the following types of tests:

a) Burn-in testing: In this test, the chip is operated in extreme conditions of voltage, temperature and other operating and atmospheric conditions. If chip breaks down in this test, then it is considered to be a bad chip. But sometimes it is not rejected. The chip is repaired and if it becomes capable of operating in extreme conditions than it becomes a good chip.

Other manufacturing tests are as follows:

b) Incoming inspection
c) Wafer sort or probe test

1.2.2 Test Plan

Before starting the test, a proper plan of the test is required. This test plan includes the following steps on the basis of the specifications of the chip.

- Types of test equipment to use.
- Types of tests to be run on the chip.
- Fault coverage requirement.

These plans are made on the basis of the conditions in which chip is going to work. For example, a GPS chip which is going to work in a car requires BIST in it. BIST tests the chip every time when the car starts. Also, it may be possible that the chip will work at a higher temperature, so there will be more focus on burn-in testing. It will guarantee us that chip will not break down in extreme conditions. On the other hand, if we talk about a GPS chip which is going to work in a smartphone does not need BIST in it. This chip can be tested using ATPG without adding any additional circuitry to the design. As compared to a car GPS chip it will work at a relatively low temperature. Hence, there will be no need to focus more on burn-in testing. Additional to these factors, a GPS chip in the car can cause huge damage in case of failure due to which fault

coverage is needed to be higher. These plans are made before starting scan insertion into a design. This can save so much time, money and unnecessary efforts of testing.

1.2.3 Fault and Fault Modelling

A fault is an unexpected difference between implemented hardware and expected design. A representation of a defect at a functional level is known as a fault. Taking an example of two input NAND gate, considering that inputs of NAND gate are A and B and output is Z. if A is logic 1 and B is logic 0, then our expected output will be logic 1. But in case the gate is faulty, there is a probability that output comes out to be logic 0. This is not the intended output. Hence, there can be possibly two faults or less.

- Either input B is stuck at or short to logic 1.
- Or output Z is stuck at or short to logic 0.

There can be the third possibility that both cases are happening. But that case is very rare. Here, one thing is worth noticing that if both inputs of NAND gate are set to logic 1 then there will be no difference between expected output and the actual output of the NAND gate. Hence, we will be unable to detect a fault which is actually there. That is why input patterns are generated to detect specific faults. Input patterns help in fault modelling.

Hence fault modelling is done to accurately analyze a design so that all fault sites should be covered. Following are the reasons due to which fault modelling is necessary.

- Input/output function tests inadequate for manufacturing.
- Real defects are too numerous and often not analyzable.
- A fault model identifies targets for testing and makes analysis possible.
- Effectiveness measurable by experiments.

1.2.4 Fault Equivalence and Fault Dominance

Fault equivalence and fault dominance play a major role in fault modelling.

- **Fault equivalence:** Two faults F1 and F2 are said to be equivalent if the test patterns set for F1 and F2 is the same. If faults F1 and F2 are equivalent, then the corresponding faulty functions are identical.
    - **Equivalent fault collapsing:** In a logic circuit, all single faults can be divided into disjoint equivalence subsets, where all faults in a subset are mutually equivalent. A collapsed fault set is a set that contains one fault from each equivalence subset.

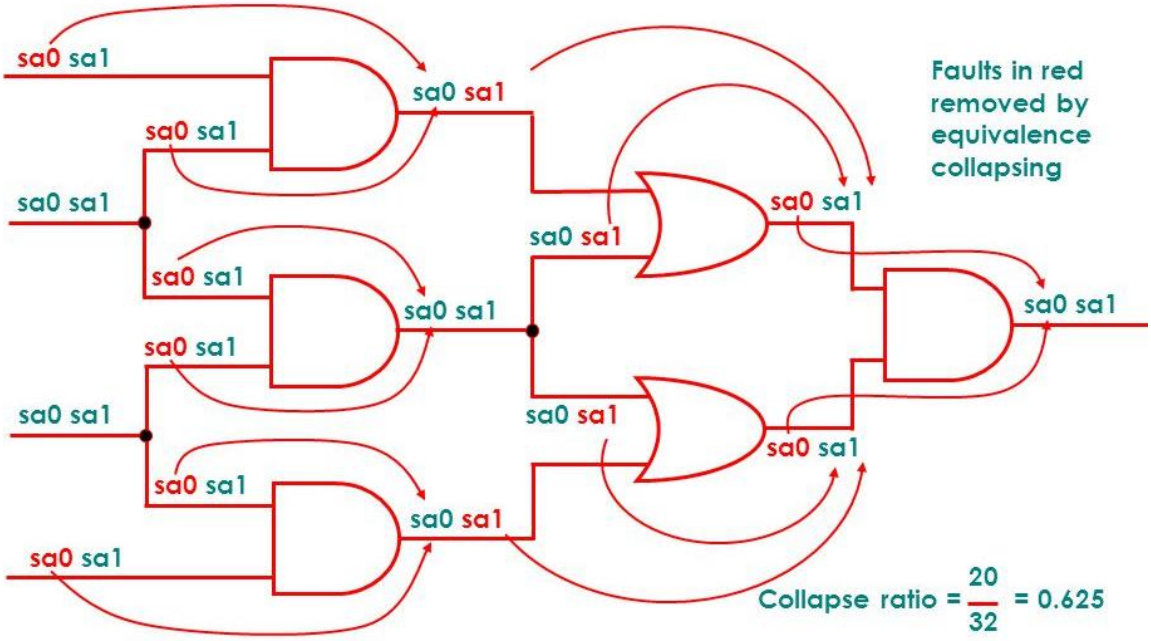


Figure 1.2-1Fault Equivalence and Fault Collapsing

- **Fault Dominance:** If all tests that were applied to detect fault F2 also detects another fault F1, then F1 is said to dominate F2.
    - **Dominance fault collapsing:** If fault F1 dominates F2, then F1 is removed from the fault list. When dominance fault collapsing is used, it is sufficient to consider only the input faults of Boolean gates. In a tree circuit, with no fan-outs, only faults at primary inputs form a dominance collapsed fault set. If two faults dominate each other then they are equivalent.

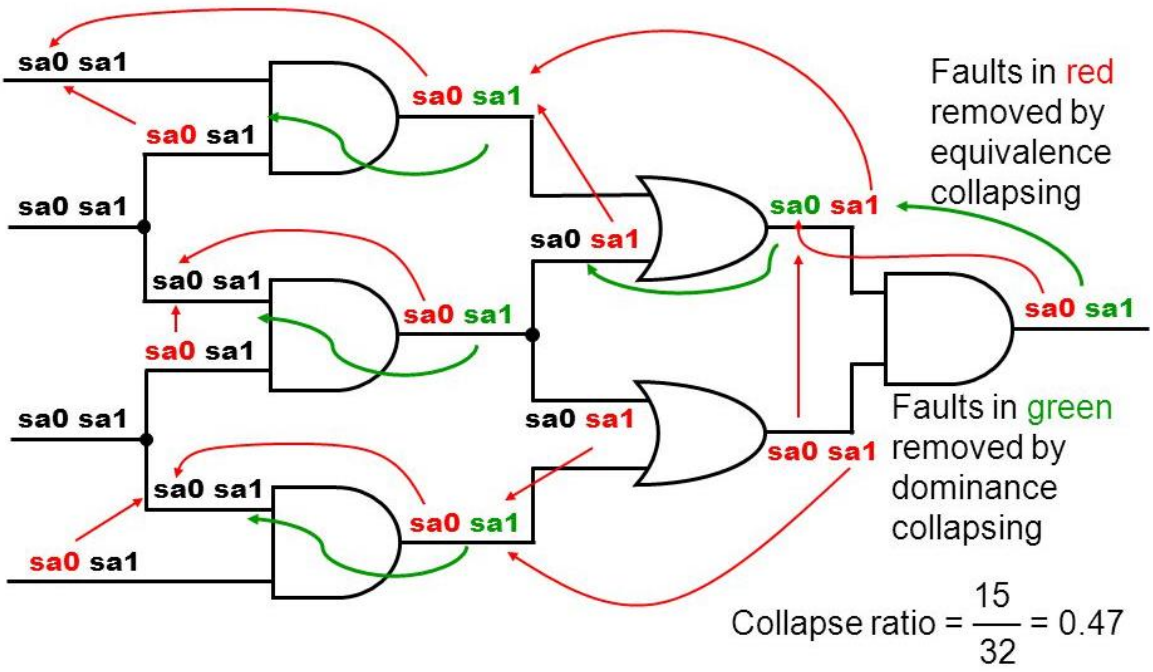


Figure 1.2-2 Fault Dominance

1.2.5 Types of Faults

- **Branch Fault:**

This fault is modeled at the behavioral level of coding. At the behavioral level, design function is defined in the code instead of using gates. A branch fault affects the branch statement. This causes a branch to connect to the incorrect destination [10].

- **Bridging Fault**:

The bridging fault is termed for a short between nets. The logic value of the shorted net is can be considered as dominant-1 (OR Bridge) or dominant-0 (AND Bridge). If a circuit is free of any stuck-at fault, there is a very high probability of a circuit to be free of bridge faults.

- **Bus Fault:**

A bus fault specifies the fault of each line present on the bus as stuck-at-1 or stuck-at-0 or fault free. Therefore, if a bus consists of n number of lines then there will be 2n stuck-at faults. A total bus fault assumes all lines of the bus to be stuck at the same 0 or 1 state.

- **Cross-point Fault**:

These faults are modeled in programmable logic arrays (PLA.) In the layout of a PLA, input and output variable lines are laid out perpendicular to the product-lines. The fault occurs when the crossing signal lines either unnecessarily get connections or remain unconnected at cross-points. Thus, cross-point faults can be classified into following types.

  - Missing cross-point
  - Extra cross-point

- **Delay Fault**:

Because of delay faults, the delay along a path is increased which causes a logic transition to fall outside the specified limit. This fault becomes more prominent at lower technology nodes. There are two types of Delay Fault Model. First one is gate-delay fault model. It assumes that delay fault is lumped at one gate in circuit. It takes into account the circuit delays and transition on inputs of the gate. The same gate may have different delay values depending on neighboring circuit elements. Another delay model is a path-delay model. It takes into account the cumulative propagation delay along the path. The problem which arises with this model is that there can be a large number of possible paths.

- **Transition Faults:**

Since MOSFET is considered as a switch. A defect with switch can be modeled as either permanently open or permanently close. This model assumes the transistor to show the behavior of stuck-open or stuck-short. Stuck-open means the transistor is open permanently. While stuck-short means transistor is shorted permanently.

Stuck-short transistor faults may be detected by monitoring the power supply current during steady state, referred to as IDDQ. This technique of monitoring the steady-state power supply current to detect transistor stuck-short faults is referred to as IDDQ testing. IDDQ testing measures quiescent power supply current rather than pin voltage, detecting device failures not easily detected by functional testing such as CMOS transistor stuck-on faults or adjacent bridging faults.

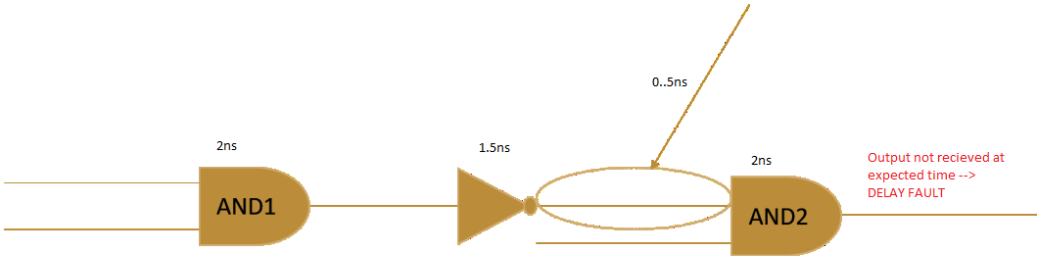


Figure 1.2-3 Transition Fault Example

In the above example, consider that AND gates have a propagation delay of 2ns and NOT gate has a propagation delay of 1.5ns. Hence, ideally, the output of AND2 should come at 5.5ns. But due to some unknown reasons, there is a delay of 0.5ns occurs between NOT gate and AND2 gate. This delay will also affect the output of AND2 gate. Now, the output will come after 6ns. But the required output should come after 5.5ns. Hence, this will result in the rise of delay faults and can also cause transition fault between NOT gate and AND2 gate.

- **Stuck-at Fault**:

The faults in this model are fixed to a value 0 or 1. It is assumed defects cause the node to be permanently stuck at a constant logic (0 or 1). The fault is called stuck-at-0 if the node is always

stuck at 0. It is called at stuck-at-1 if the node is always stuck at 1. It has also been observed that if a device after fabrication is found to be free of any stuck at faults, there is a very high probability that it will be free of any other defects. This model is further divided into two types. Single stuck-at fault model and multiple stuck-at fault model. According to a single stuck-at fault model, only one node can be faulty at a time. Multiple stuck at fault model assumes that at the same time there can be multiple stuck-at faults in the circuit. Single stuck at fault model helps in converting complexity from exponential to linear.

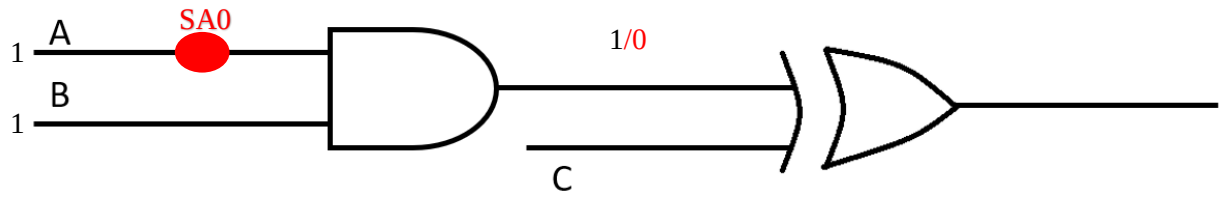


Figure 1.2-4 Stuck-at Fault Example

In the above example, consider the input A is stuck-at-0. That means, it will be impossible to put logic 1 at input A. Due to this fault, the output of AND gate will never be logic 1 whatever the value is present at input B. So, if we try to set both inputs of AND gate to logic 1 and also set input C to logic 1, the output of AND gate should be logic 1 and output of XOR gate should be logic 0. But due to stuck-at-0 at input A, the output of AND gate will be logic 0 and that of XOR gate will be logic 1. So, the actual output will be different from the expected output. Hence, due to one stuck-at fault results in the wrong output.

- **Initialization Fault**:

When memory circuits with memory elements (e.g., flip flops) are designed, they are designed in such a way that they can be initialized by applying one suitable input. This procedure is called the initialization procedure. Faults that interfere with these kinds of input values are called initialization faults.

- **Permanent Fault:**

Any faults that occur every time and does not change the behavior of the fault is called a permanent fault. On the other hand, the faults that occur at random instance of time with random behavior are called intermittent faults.

- **Quiescent Current (I_{DDQ}) Fault:**

These faults generally occur in CMOS technology. In steady state, ideally, CMOS does not conduct any current. But there is some current present that flows through CMOS. This current is called leakage current or quiescent current (I_{DDQ}). Generally, this current is only a few µA, but in some conditions, the magnitude of the current rises. This rise in current allows detecting fault by measuring current. Faults detectable by this method are called I_{DDQ} faults [10].

- **Race Fault:**

Sometimes, race conditions occur in the circuit due to stuck-at faults present in the circuit. For a certain initial state and input, the signal state of an asynchronous sequential circuit can vary depending on specific delays of its logic gates. Such a condition is known as a race condition.

- **Redundant Fault:**

Any fault that does not modify the input-output function of the circuit is called a redundant fault. These kinds of faults come under undetectable faults as there is no such pattern that can detect redundant faults.

- **Stuck-Open and Stuck-Short Faults:**

In purely digital circuits, a MOS transistor works as an ideal switch. A defect in these circuits is modeled as the switch is permanently in either the open or the shorted state. This fault model assumes just one transistor to be stuck-open or stuck-short. Stuck-short is also referred to as stuck-on or stuck-closed.

1.3 DFT TECHNIQUES

What is Design For Testability? In DFT, some additional design is added to the circuit which can be used to test the circuit. There can be many ways to test the circuit. Only two techniques are used widely in DFT.

- Built-in Self-Test (BIST)
- Scan Insertion and Automatic Test Pattern Generation (ATPG)

a) BIST:

In this technique, an additional circuit is added with the design which can perform all the testing by itself. BIST itself generates test patterns and apply them to the main design. The output is compared by the BIST. If there is any error in the outputs, then BIST will send the error signal. In addition, we can add a feature in which BIST will not let the main design to work in case of error. This can be very useful in critical designs such as when a car is starting up, BIST runs before the engine starts. This includes checking brake sensors, MP3 player, headlights of the car, etc.

Now suppose if there is any error in the brake sensors, then BIST will not let the driver to start the car. Also, there will be an indication of brake failure in front of the driver.

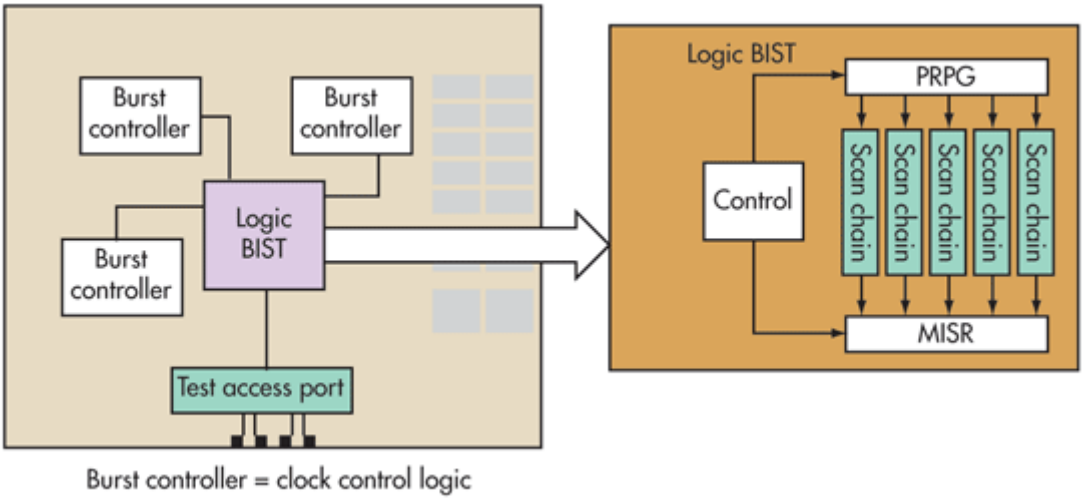


Figure 1.3-1 BIST Block Diagram

Pseudo Random Pattern Generator (PRPG) is used in BIST to generate test patterns. Once PRPG starts, it generates random patterns of any number. This does not need any memory element to store the patterns.

Linear Feedback Shift Register (LFSR) is one of many techniques to generate test patterns in BIST. In this technique, test patterns are generated in random order. These patterns are applied to the design and output is compared by the BIST.

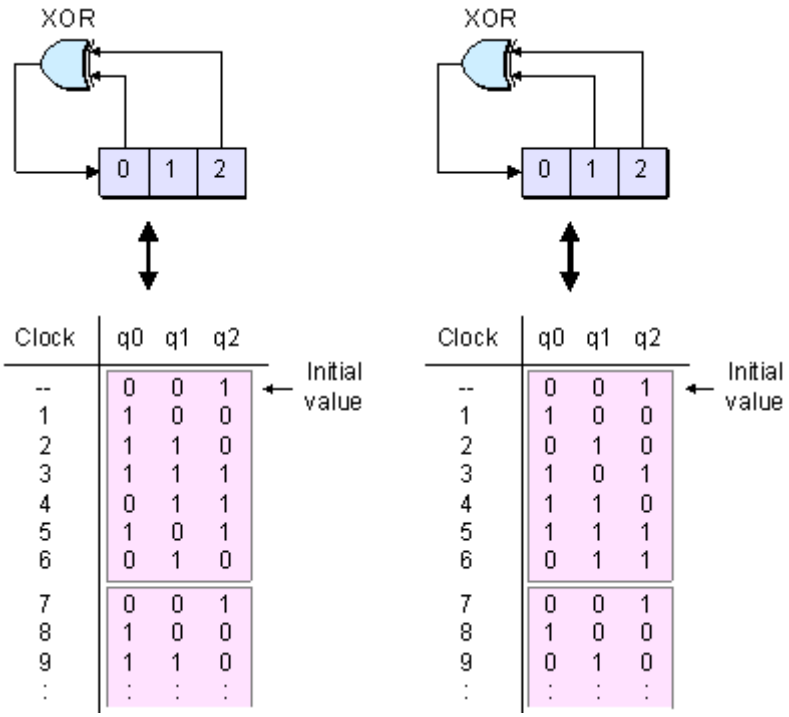


Figure 1.3-2 LFSR

b) Scan-based ATPG

In scan insertion, scan chains are added to the design. These chains consist of flip-flops called scan cells. Input test patterns are injected serially through these chains and output data is also retrieved from these chains. There can be many chains in the design. These scan cells are slightly different from the traditional flip-flops. But if we are adding something to the design, this will cost us in increasing area of the chip. To avoid this, the flip-flops which are already used in the design are converted into scan cells and then connected to each other to make scan chains. This process of converting traditional flip-flops into scan chains is called Scan Insertion. There are many advantages of scan insertion over the BIST but also have some disadvantages. These advantages and disadvantages will be discussed in the next chapter. There will be other factors about scan based ATPG that will be discussed in the next chapters. For example, the difference between scan cells and traditional flip-flops, other test techniques, and their efficiency, etc. All these questions will be answered in the following chapters.

1.4 LITERATURE REVIEW

According to Moore's law, the size of the device will continuously decrease with the increase in technology and time. With increasing technology, demands of users are also increasing. Hence, to fulfill those demands, the complexity of the designs is increasing rapidly. But the fabrication of such small designs with high complexity is a very difficult task. There can be fabrications defects in the design. In DFT, extra circuitry is added to the design which will allow the testing of design after manufacturing. The purpose of DFT is to validate that the product hardware does not contain any manufacturing defects. But adding the testing circuit is a very critical task. Before validating the circuit for manufacturing, design needs to run through a number of simulations. There are a number of things that need to be taken care of while adding scan chains into the design and achieving the required test coverage. The major problems occur when some of the faults remain undetected or untested. In fault classes, delay faults are very difficult to detect. Sometimes, a small delay can become a reason for the failure of the circuit. So, in DFT, special care is taken to detect and test those small delay faults. In this section, we shall see various research works related to the detection of small delay faults. The publications purposed various methods that help in detecting small delay faults. This can help in the increase of test coverage and minimize the failure rate of the design.

Shubhanshi Mitra et al. [1] collected delay test data from chips which are fabricated in 0.18µm technology. They suggested that maximum operating frequency in given operating conditions is influenced by three factors i.e. process speed, process shift and presence of spot defects. In their experiment, they manufactured ELF18 test chip to run an experiment on it. The results of the experiment shown the effectiveness of process monitor structure using on-chip ring oscillator. They were able to detect slow parts of the design. But they were unable to completely eliminate transition fault testing. Also, they mentioned that the experiment is effective only if the clock frequency is very well controlled.

Jing-Jia Liou et al. [2] demonstrated how the definition of the critical path can change when it comes to selecting critical paths in the deep sub-micron domain. In the experiment, the investigated that test effectiveness changes when delay assumptions change from nominal or worst-case model to a statistical model. They analyzed that traditional method of selecting the longest paths is not adequate. So, they adopted some new criteria. In the study, they considered three path selection objectives which were – statistically longest path, considering path correlation and path independence. In this way, they proposed an optimized strategy of selecting critical paths.

Manish Sharma [3] suggested a way of finding a small set of longest testable paths that can cover every gate. He used a graph traversal algorithm that can traverse all paths of a given length. He presented

an ATPG technique to automatically determines the longest path which will cover every gate or wire in the circuit without first testing the longest paths. He created an algorithm in which a target vertex is chosen and then follow longest forward as well as backward paths until output vertex and input vertex are reached respectively. This algorithm is applicable for all size of testable path set. However, the minimality of the path set size is not guaranteed.

Janak H. Patel et al. [4] proposed a segment delay fault model that represents any general delay defect. This general defect can range from a spot defect to a distributed defect. In the paper, they prepared an algorithm to generate tests for a segment delay fault. They categorize tests for segment delay faults in robust, transition and non-robust tests. In the algorithm, the generated segment delay fault model will have faults that include both rising and falling transitions, fan-out branches, and non-fan-out stems were considered for the calculations. The new segment delay faults model generated using the algorithm considered slow to rise and slow to fall defects. Also, they proposed an algorithm that is able to compute the number of segments of any possible length in the circuits.

Eun Sei Park et al. [5] proposed a method to improve delay testing quality in timing optimized designs. They proposed an algorithm to optimize the path delay values. The objective of the algorithm is to optimize the delay of all paths to the same value. They stated that path delay timing has both positive and negative effects with respect to performance and quality. Hence, the algorithm will optimize the path delay values so that silicon can be used more efficiently to give more optimal performance. These path delay timings will have a positive effect on timing slacks, detection of delay defects on the path along which the testing is done. As the delay values in all paths will be equal, there will be no need for small delay fault testing. BIST will also become more effective. ATPG for delay faults, simulation and BIST for delay faults will become less complex and more effective. The main purpose was to optimize the delay values for all paths which will result in more efficient and less difficult delay fault testing.

Yasuo Sato et al. [6] introduced a model to reflect quality, delay margin and test timing accuracy of a fabricated chip. In this model, the level of chip defects that can cause delay faults could be predicted. They experimented this model on ISCAS89 benchmark data and some other industrial designs for comparison of data. They stated that if the fabrication process quality is poor then it can cause poor yield and if design quality is poor it can cause small delay faults. Similarly, test timing accuracy and test pattern quality can cause poor chip quality and poor fault coverage respectively. In the paper, they evaluated the model from various points of view such as the effect of delay distribution of design and effect of timing on quality. The experiment revealed that transition tests alone cannot assure quality. To assure the quality, a combination of transition tests and path delay tests can be used especially when delay defects are small. This can result in saving test time and effort although test complexity may arise.

Xijiang Lin et al. [7] proposed a methodology to improve the quality of AT-Speed testing for small delay faults. They suggested a method to improve the quality of ATPG. Their motive was to generate patterns by achieving delay information from Standard Delay Format (SDF) files. SDF files contain delay timing information of the design. This information can be used to improve the quality of the test generator to detect small delay faults. In the paper, they measured the quality of test set and comparing it with the traditional test set on the basis of delay test coverage. Delay defect testing is essential to ensure the quality of deep submicron designs. Traditional transition testing does not consider timing information, whereas the timing-aware ATPG proposed in the paper considers SDF files that contain timing information. The proposed method increases the test quality of the delay test and can easily detect small delay defects.

CHAPTER – 2

SCAN BASED ATPG

2.1 INTRODUCTION

Scan-based automatic test pattern generation (ATPG) is a technique used for the testing of designs. The first step in this technique is to add scan cells into the design and connect them with combinational circuits. These scan cells are used to put logic values at the internal nodes and also to capture output values of the combinational logic of the design. These scan cells are also connected with each other and form a chain-like structure. The values captured by these cells are shifted out and compared with the expected values. The second step is to generate test patterns according to the scan chain connections with the combinational circuits. After generating patterns, they are simulated to check the reliability of the design. In this chapter, the working of scan chains and scan cells is defined.

2.2 SCAN CELLS

Scan cells or scan flops are the main part of scan insertion. These are nothing but modified flip-flops. These are slightly different from traditional flip-flops. For easier understanding, we will compare D scan cells with D flip-flops. A scan cell has two inputs and two outputs, unlike simple D flip-flop. It has input pins as test input (TI), data-in (D) and at the output, it has test output (TO) and data-out (Q), whereas a D-flip-flop has only data-in and data-out pins.

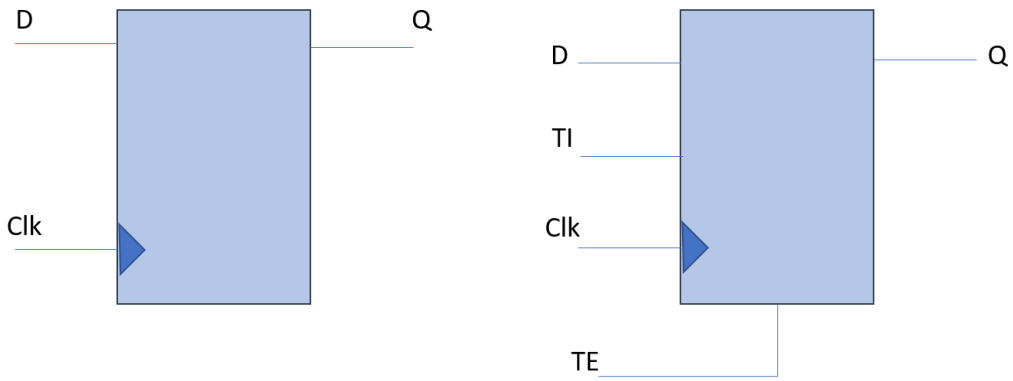


Figure 2.2-1 Comparison Between D Flip Flop and Scan Cell

A scan cell has an additional multiplexer connected to the input of a D flip-flop. The inputs of the mux are tested input and data in with a select line called test enable (TE) pin. When TE pin in 1 then TI goes to the input of flip-flop, otherwise data is flowing through the flip-flop. With this technique, a flip-flop can be used in both ways. During testing, a flip-flop works as a scan cell and during normal functioning, it can act as simple flip-flop. This helps in reducing the chip size. A detailed scan cell is shown in the figure below.

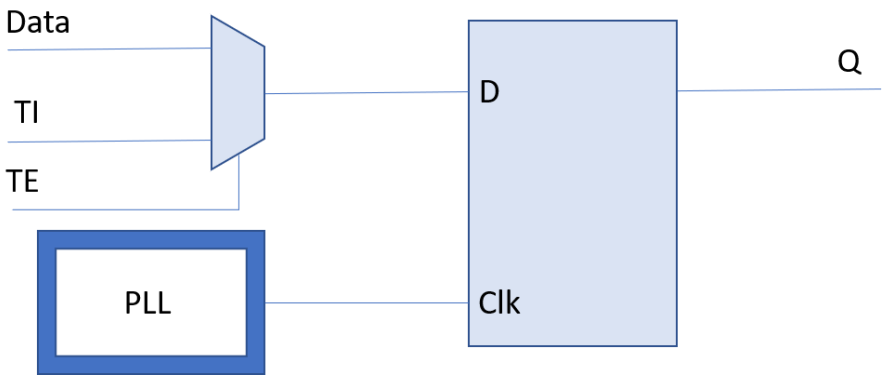


Figure 2.2-2 A Scan Cell

2.2.1 Working of Scan Cell

A scan cell can work in functional as well as testing mode. During functional mode scan cell work exactly as normal flip-flop. To set scan cell in functional mode, TE is set to logic 0. When TE is logic 0 then mux passes Data-in pin to the output and clock mux will also pass functional clock to the flip-flop. Hence, in functional mode, a scan cell will behave as a standard flip-flop. During testing, the TE pin is set to logic 1. In this case, mux will pass test input to the flip-flop. So, the test vector will reflect at the output of the combinational circuit. In case the data is applied serially in the scan chain, then the output Q of the scan cell will be connected to the TI of the next scan cell in the chain. Once the chain is filled with test vector, the data in the scan cell is parallelly applied to the combinational circuit.

2.3 SCAN CHAINS

Scan chains are consisting of scan cells connected in series. The output of every scan cell is connected to the TI of the next scan cell and also connected to the design. Data is serially entered into the scan chain. When the scan chain is fully filled with the test vector, the data stored in the scan cells are applied to the design simultaneously. When data is serially entering into the scan chain the process is called shifting. Shifting can be done at any clock frequency. After shifting, when data is applied to the combinational circuit, then the functional clock is applied so that the circuit can run at functional speed. The output of the circuit is again then stored in another scan chain which is then shifted out serially. This shifted data is then compared to the expected output. If the shifted data and the expected output is the same then the circuit is running without any faults.

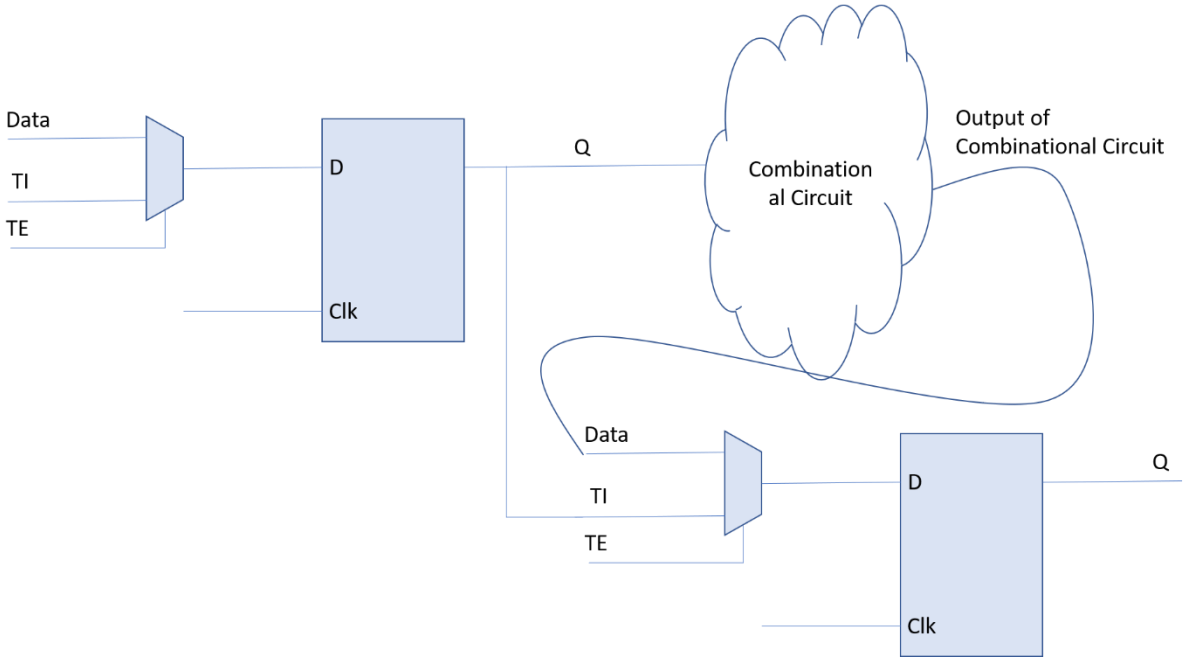


Figure 2.3-1 Scan Chain Connections Through Combinational Circuit

2.4 SCAN BASED ATPG v/s BIST

Scan-based ATPG is a one-time process whereas BIST runs every time the design starts. Hence, there is a lot of difference between these two techniques. Both have their own advantages and disadvantages. Both are equally important for different kind of designs. Some of the many differences are defined below.

1. **Fault Coverage:**

   Faults coverage in ATPG is far way better than BIST. As ATPG is run only once for a design it is very important to make sure that the design does not have any faults. For Stuck-at faults, ATPG requires fault coverage of more than 99%, whereas, for transition faults, it has coverage around 90%. In the case of BIST, it has reasonable coverage value but when compared to ATPG, it has less fault coverage.

2. **AT-Speed Testing:**

   In ATPG, delay-related defects can be removed by AT-speed testing. There are two approaches used in AT-Speed testing i.e. Launch on Shift and Launch from capture. For BIST to do AT-Speed testing needs extra hardware. Hence, it is difficult to perform AT-Speed testing.

3. **Design size:**

   ATPG does not require any extra on-chip hardware to perform testing. Whereas BIST needs extra on-chip hardware. This results in an increase in chip size, which is not acceptable.

4. **Speed of testing:**

Speed of testing does matter a lot in DFT. It is always favorable if testing is requiring minimum time. Here, BIST gets some advantages over ATPG as ATPG took longer time to perform testing of a design. One reason for ATPG to take longer time is that in ATPG more number of test patterns are required to attain more test coverage.

5. Reusability:

Once ATPG is done, it cannot be performed again very easily. BIST runs by itself every time the design runs. In the case of ATPG, reuse in board manufacturing retest is not possible due to the absence of testing circuit.

6. Memory components:

Memory components are required in ATPG. ATPG requires off-chip storage for test data so that testing can be performed on every single chip. In the case of BIST, every single chip contains its own testing circuit which itself creates test patterns to perform testing. Hence, BIST does not need any memory component on or off-chip.

Along with the above differences, ATPG and BIST have some similarities also. For example, both testing techniques use scan chains. With the advancement of technology, BIST is getting more famous. With growing technology, now BIST can have more coverage, which is almost equal to the ATPG. The leading commercially available logic BIST solution provides true at-speed testing. ATPG-based flows continue to try to provide further techniques to meet the testing challenges of today's complex designs. However, leading commercial logic BIST capabilities originally developed to address these high-end design test challenges have matured over the past several years and has become field hardened and field-proven solutions.

2.5 SCAN INSERTION

Scan insertion is the most basic and first step in DFT. Converting every flip-flop in the design into scan-cell and then connecting these scan cells with each other to make scan chains. But this process is not as easy as it looks. Good scan insertion can create a huge difference in test quality. There are some design rules that are defined and should not be violated. However, it is impossible to obey all the design rules in a single design. Following are some factors that should be known before starting the process of scan insertion.

2.5.1 Partial Scan

When every flip-flop in design is converted into a scan cell, then it is called full scan design. It uses more area and more test coverage and also more expensive. For cheaper tests, partial scan design is used. In partial scan design, not every flip-flop is converted into scan cell. It takes a lesser area and less test coverage. It is cheaper and faster than a full scan design. As fewer test patterns are needed to be generated, it also saves some memory space.

2.5.2 Scan Clock

During shifting in and shifting out of the test vectors, a different clock is used. During the launch and capture pulse, the functional clock is used. The clock used during shifting is called a scan clock. This clock is controlled by automated test equipment (ATE). Generally, the scan clock is slower than the functional clock. The ATE provides just a reference signal for clock generation on the design under test (DUT).

Power dissipation and test cost are the reasons for using the slower clock for shifting. Low clock frequency results in lower dynamic power dissipation. Along with this, every scan cell has its own setup time. Hence, with slower frequency, scan cells get enough time to avoid setup violation. Also, with high frequency, there will be higher toggling which will result in high chip temperature.

2.5.3 Controllability and Observability

- **Controllability** is defined as the ability to force value at a particular node by applying a test pattern at the primary input of the circuit. If a node is uncontrollable then no value can be forced at that node by applying any test vector on the input.

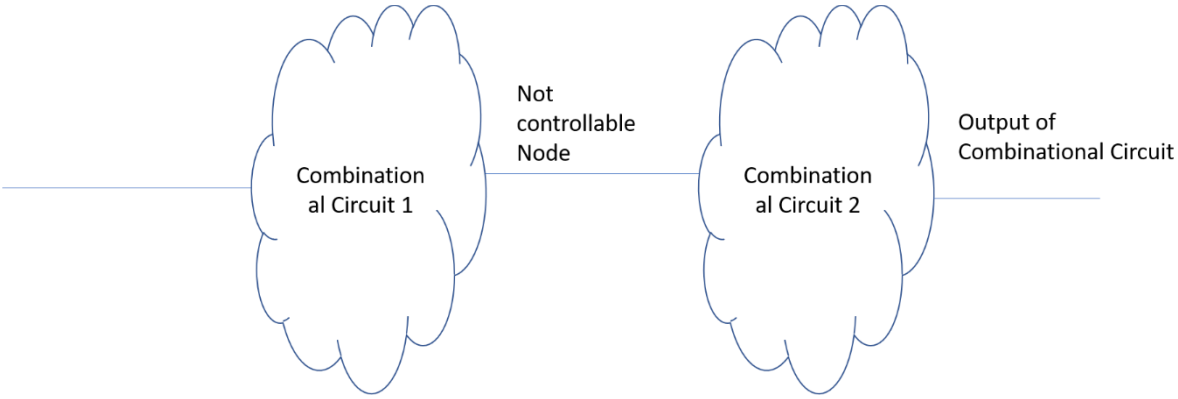


Figure 2.5-1 Not Controllable Node

For example, in the above circuit, the node between two combinational circuits is not controllable. Scan insertion helps in increasing the controllability of the design. If a scan cell is connected to that node then the value at that node can be controlled by giving the value directly to that scan cell as shown in the figure below.

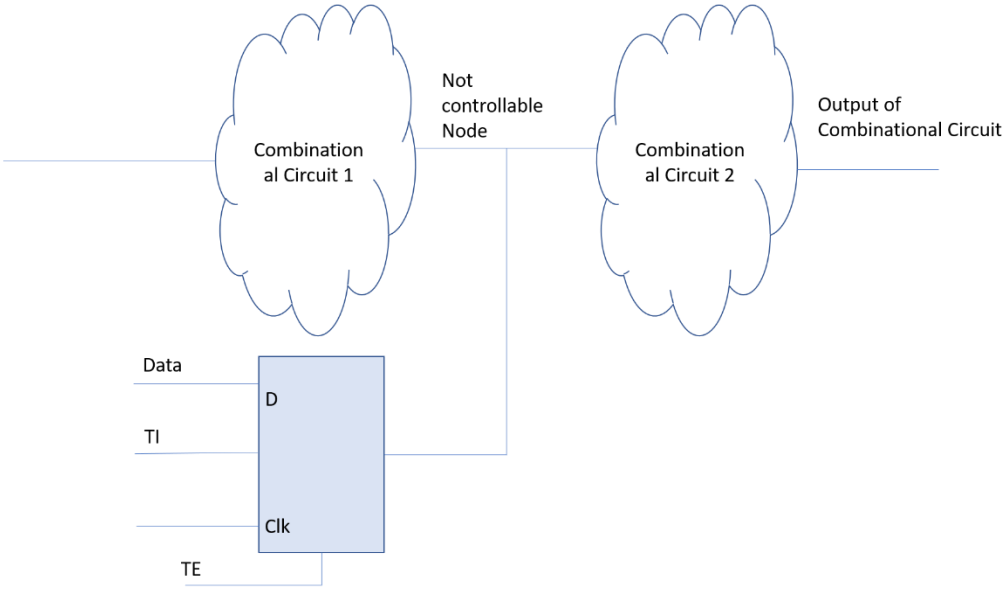


Figure 2.5-2 Node Controlled Using Scan Cell

- **Observability** is defined as the measure of how well the value at any internal node can be reflected at the output of the circuit.

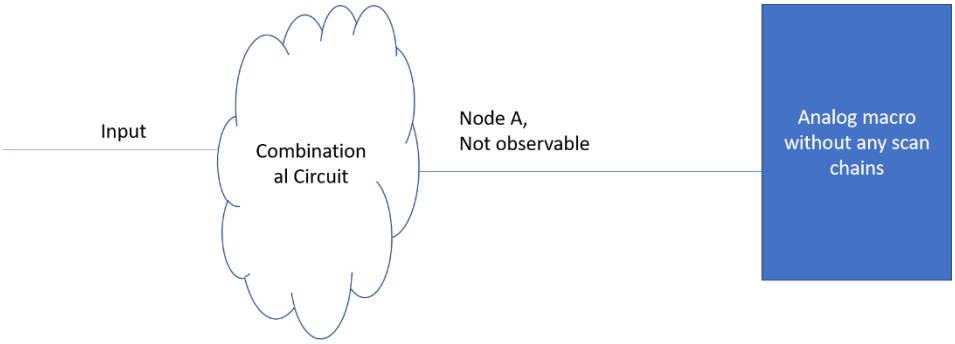


Figure 2.5-3 Not Observable Node

For example, in the above circuit, node A is not observable. It means, whatever value is available at node A cannot be observed at the primary output. If a scan cell is connected to that node then that cell will capture the value at the node. And that value can be observed at the output through that scan cell.

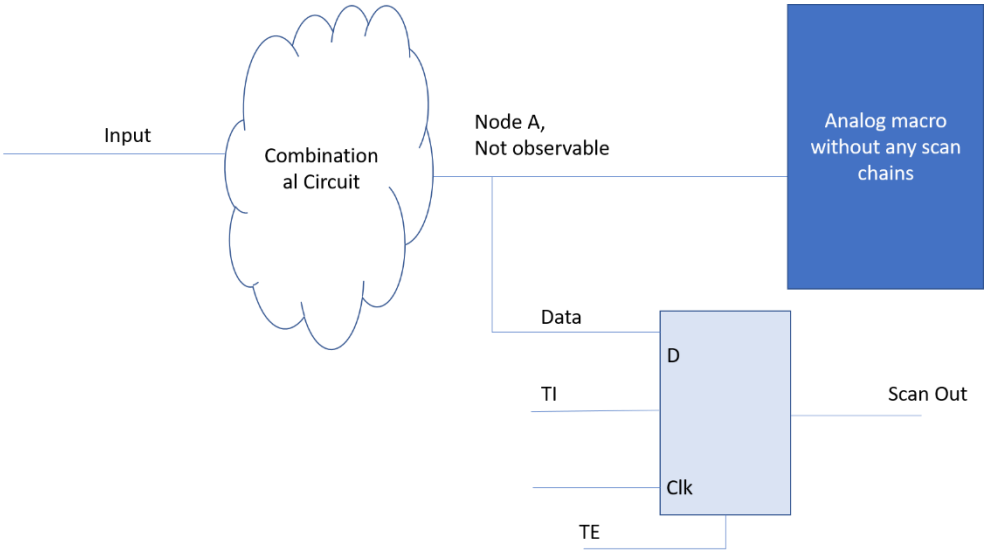


Figure 2.5-4 Node Observable Using Scan Cell

Any circuit should be highly controllable and observable to increase the test quality. Scan insertion increases the controllability and observability of the design. This because the reason scan chains are inserted in between the combinational circuits. Hence, it becomes easier to control and observe the values at internal nodes.

2.6 SCAN OPERATION

Consider a scan chain in the figure 2.5-1. The first thing is to do is to put the scan cells in the scan chains into scan mode. This can be done by setting the scan enable to logic 1. Initially, all the scan cells are in an unknown state. After putting the scan enable to logic 1, bits start shifting through each scan cell on every clock pulse. Usually, the shift clock frequency is slower than the normal functional frequency. The reason of slower shifting frequency is that if shifting is done at high frequency than all the scan cells will toggle simultaneously which leads to higher activity factor. This turns into higher dynamic power consumption and higher heat dissipation. Hence to avoid this, shifting is done at the slower clock frequency.

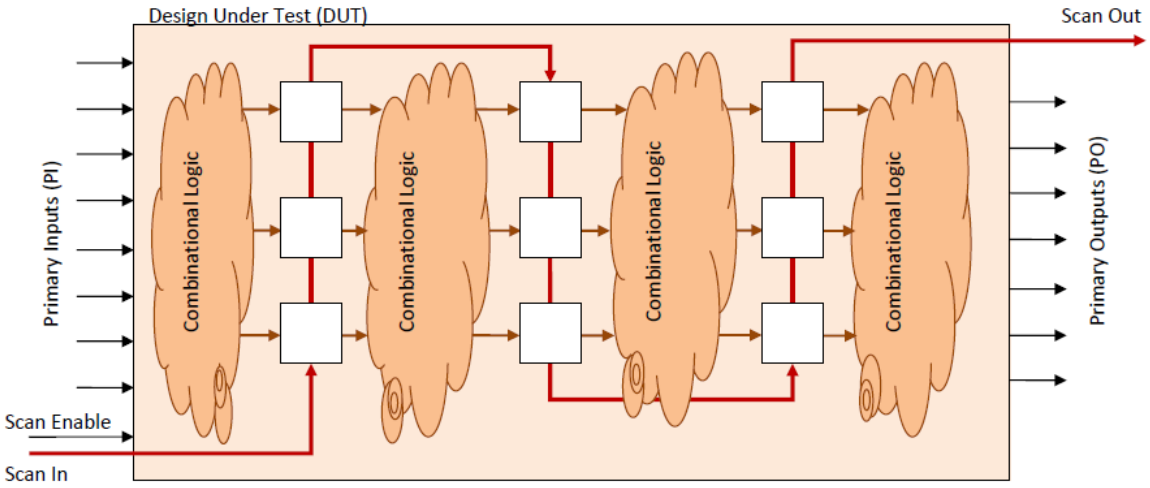


Figure 2.6-1 Scan Chain Operation

When all the shifting is done, scan enable is again set to logic 0 to disable the scan mode. Now the shifted data is applied to the combinational circuit. Once the combinational circuit generates their output, one clock pulse is applied to capture the output of the combinational circuit into the scan cells. This clock pulse is called the capture pulse. After the scan cells capture the output of the combinational circuit, once again scan enable is set to logic 1 and output is shifted out of the scan chain is serial form. A number of clock pulses required to shift the pattern into the scan chain are equal to the number of scan cells present in the scan chain.

CHAPTER - 3

ATPG

3.1 ATPG

ATPG stands for Automatic Test Pattern Generation or Generator. This technique is used to generate test patterns which are applied to the design for the purpose of testing. This technique is widely used in modern electronics for testing designs. There are number of softwares that help for the automation in generating test patterns. Before generating patterns there is a number of factors that need to be known. There is a flow to defined for ATPG process, which is shown in the figure below.

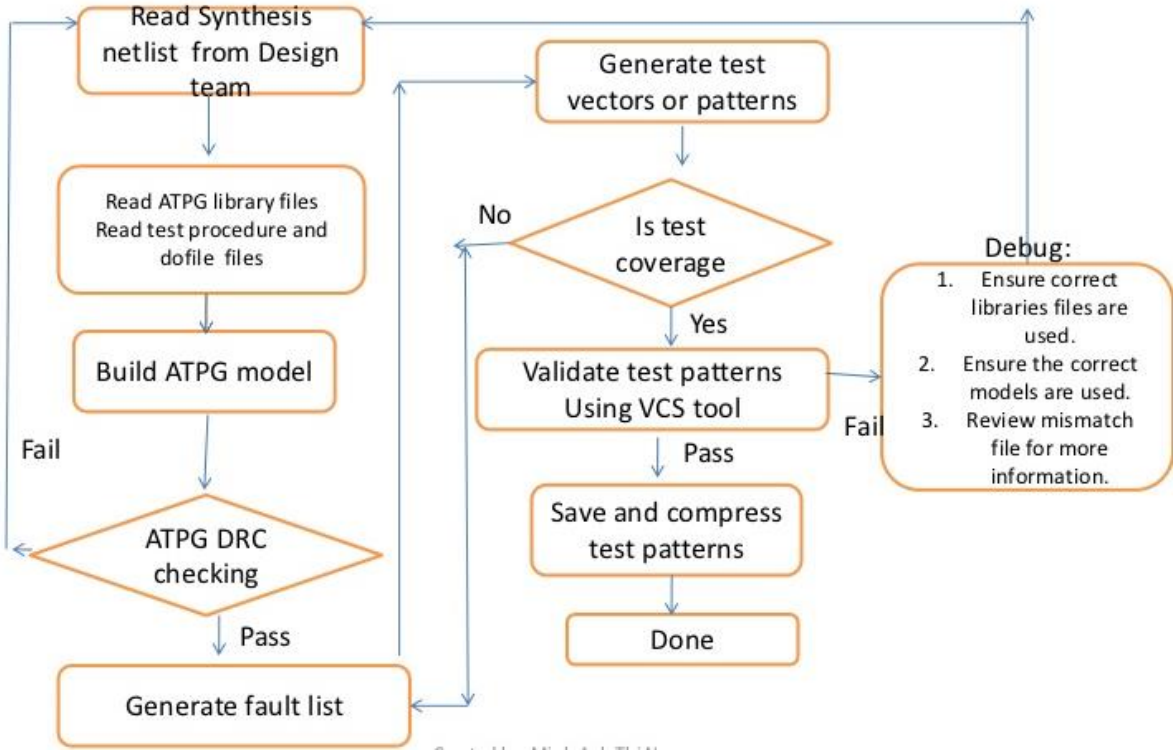


Figure 3.1-1 ATPG Flow

3.1.1 Netlist

The netlist is the Verilog code written for the design. The very first step in ATPG is reading the netlist. the scan chains are already inserted in the netlist.

3.1.2 ATPG Library Files

ATPG library files are added to add the fault types that needed to be tested in the design. These files contain information about the types of faults that can be present at a particular node.

3.1.3 ATPG Model

In this step, an ATPG model is synthesized for the full design which contains all kind of faults that can be present in the design.

3.1.4 ATPG DRC Check

In ATPG DRC check step, design rules are being checked. Sometimes, while adding scan chains into the design and distributing clock may result in a violation of design rules. To make sure, DRC is done after adding scan chains into the design. If there is any design rule violation then we have to synthesize the design netlist again.

3.1.5 Generate Fault List

It is an ASCII file which contains internal fault information. Once design rule checks are passed, the next step is to generate a fault list for the design. Those faults are added to the fault list for which we are going to generate test patterns. These fault lists are made on the basis of fault type, module type, fault class, etc. The fault list is generated so that during pattern generation, faults can be picked from the fault list and pattern can be generated to detect those faults.

3.1.6 Generate Test Patterns

When ATPG model is built, we need test patterns that can be applied to the model and test the design. The aim is to generate the minimum number of test patterns that can cover the maximum number of faults. The pattern generation tool checks the coverage while generating test patterns. If coverage is not achieved then it will keep on locating new faults from the fault list and generate patterns for those faults.

3.1.7 Save and Compress Test Patterns

Once test coverage is achieved then we move to the next step. In this step, generated patterns are saved in files. Test patterns are generated and stored in different kind of files. For ex.- STIL, Binary file, etc. A binary file is read by the tool whereas STIL files are a human-readable format of the file. The tool read patterns from these files and apply to the design and then compare the outputs.

When test patterns are generated and saved in the files then these patterns are used for fault simulation of the design with different delay values.

3.2 ATPG APPROACH

The ATPG approach consists of the following three phases. In general, these phases are called path sensitization.

- **Fault Sensitization:**

  This includes the process of finding the input values that when applied to the inputs results in value at the fault site opposite to the fault value. For ex.- if the fault is stuck-at 1 at some node then we will try to put logic 0 at that node.

- **Fault Propagation:**

  When the value is set at some internal node then it should be propagated to the output of the design. This can be done by setting the neighboring input of fault site so that the value at the fault site should propagate to the output port.

- **Justification:**

  Justification refers to finding the value at input port which will allow the neighboring inputs of fault site to be set to the value found in the fault propagation step.

It may be possible that we cannot find a set of inputs to detect the fault. These kinds of faults are classified as undetectable.

3.3 ATPG PROCESS

In ATPG, a set of test patterns are generated that achieve given test coverage. ATPG process consists of two steps:

- Pattern Generation
- Fault Simulation

In fault simulation, faults present in the design are detected by the patterns generated by the tool. There are mainly three methods used by the tool to generate patterns.

- **Random Test Pattern Generation**

  In this method, test patterns are generated in random order. Then only those patterns are identified which can detect faults. Then those patterns are stored in the pattern set. The drawback of this method is that it cannot be as precise as a deterministic test pattern generation. It cannot generate a pattern for those faults whose detection probability is very less. This method is useful in the initial testing of the design.[11]

- **Deterministic Test Pattern Generation**

  In deterministic test pattern generation, tool picks a fault from the fault list and try to create patterns that intend to detect that fault. It then checks to make sure if the pattern is able to detect the fault. This can be achieved by using the path sensitization procedure. If the tool is unable to generate a pattern for any fault then is classifies the fault as undetectable otherwise classify as a detectable fault.[11]

- **External Test Pattern Generation**

The tool can also examine the external set of patterns if there is any external set is generated before. The tool analyzes that which pattern from an external pattern set can detect faults from an active fault list. Patterns generated by this method brings efficiency in obtaining the test coverage.

3.4 PATTERN GENERATION COMMANDS

For ATPG, Synopsys Tetramax software was used. In this section, some basic commands that are given to the tool before the tool starts generating patterns will be discussed. These commands can be given to the tool directly or a .tcl file can be generated in which all commands are written in the series of their execution. This file is called command file. The command file can be sourced to the tool and tool itself take commands from the file.

Following is the design flow used for ATPG.

- **Preparing a Netlist**

The tool can accept the netlist file in VHDL, Verilog, and EDIF format. The tool runs in three different modes i.e. BUILD, DRC and TEST mode. This command can run only in build mode.

Command: `set_netlist`

- **Reading a Netlist**

The tool can read more than one netlist associated with the design. The following command can be used to read all netlists stored in "design" folder written in Verilog code.

Command: **`read_netlist /tech/*.v`**

- **Reading Library Models**

Library models are also written in Verilog code. Similar to the netlist, the tool can read more than one library models. Command to read library models is the same as that of the reading netlist. the only difference is that the path should be correct in which directory the library models are stored.

- **Setting Up and Building the ATPG Model**

At this step, tool switches from build mode to DRC mode. This command is used to set the parameters for the ATPG model.

Command: `set_build -hierarchical_delimiter .-nodelete_unused_gates`

- **Performing Test Design Rule Checking (DRC)**

In this step, design rules are checked. As described above, DRC is done before generating patterns for the design. If there is any design rule violation then first that violation should be removed before

starting the ATPG process. Standard test interface language (STIL) includes scan shift protocol, test procedure, ATPG signal, timing, and data information.

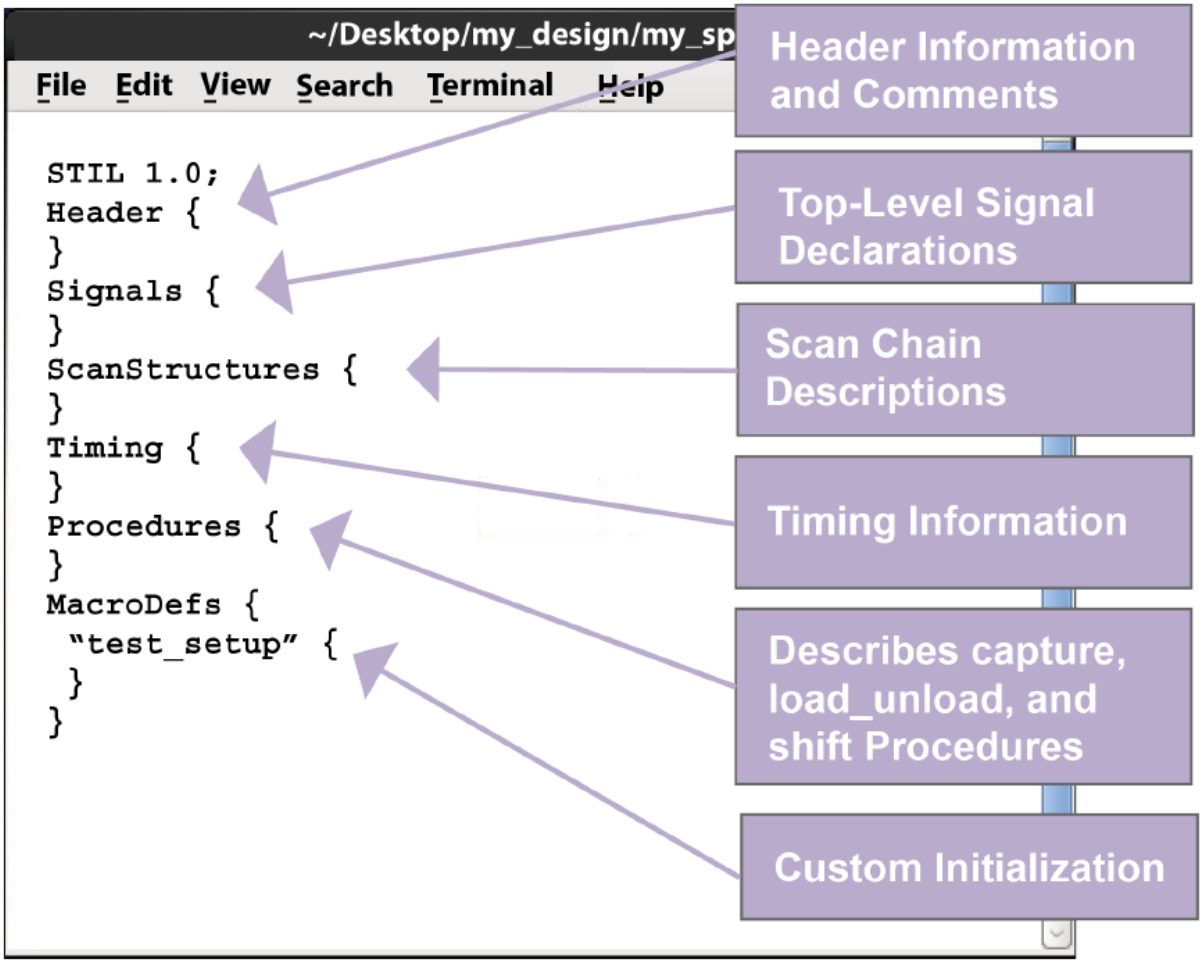


Figure 3.4-1 STIL Procedure File Format

The file in which STIL language is written is called the STIL procedure file (SPF). The SPF can be provided to the tool either by creating or by adding some previous file that is already created.

Command: `write_drc_file`

To run the DRC the SPF is read by the tool by using the following command

```
run_drc spec_stil_file.spf
```

The argument in the example, spec_stil_file.spf, is the name of the STIL procedure file.

- **Preparing for ATPG**

Before the ATPG process start, it is important to specify certain things for the tool. This includes specifying general ATPG settings, fault list, fault model, pattern source and ATPG mode. At this point, the tool changes its mode from DRC to test mode.

In ATPG settings, the tool is given the information about the maximum number of test patterns that can be created and the maximum required test coverage.

Command: `set_atpg -patterns 500 -coverage 98`

In the example, the value of a maximum number of test patterns is set to 500 and coverage is set to 98%.

Using command `add_faults -all,` the fault list is added to the tool. In fault model step, the type of faults is specified for which the tool has to create patterns. For example, if patterns are to be generated to detect transition faults then the command `set_faults -model transition` is used.

- **Running ATPG**

In this step, pattern generation takes place. The tool starts generating patterns and checking the coverage simultaneously. Basically, the tool takes one module from the design, checks the number of faults present in that module and try to create test patterns to detect those faults. Hence, the tool covers each module one by one in the design. In this step, the tool checks the coverage of each module and after creating test patterns, the tool automatically compresses patterns.

Command: **`run_atpg`**

- **Analyzing ATPG Output**

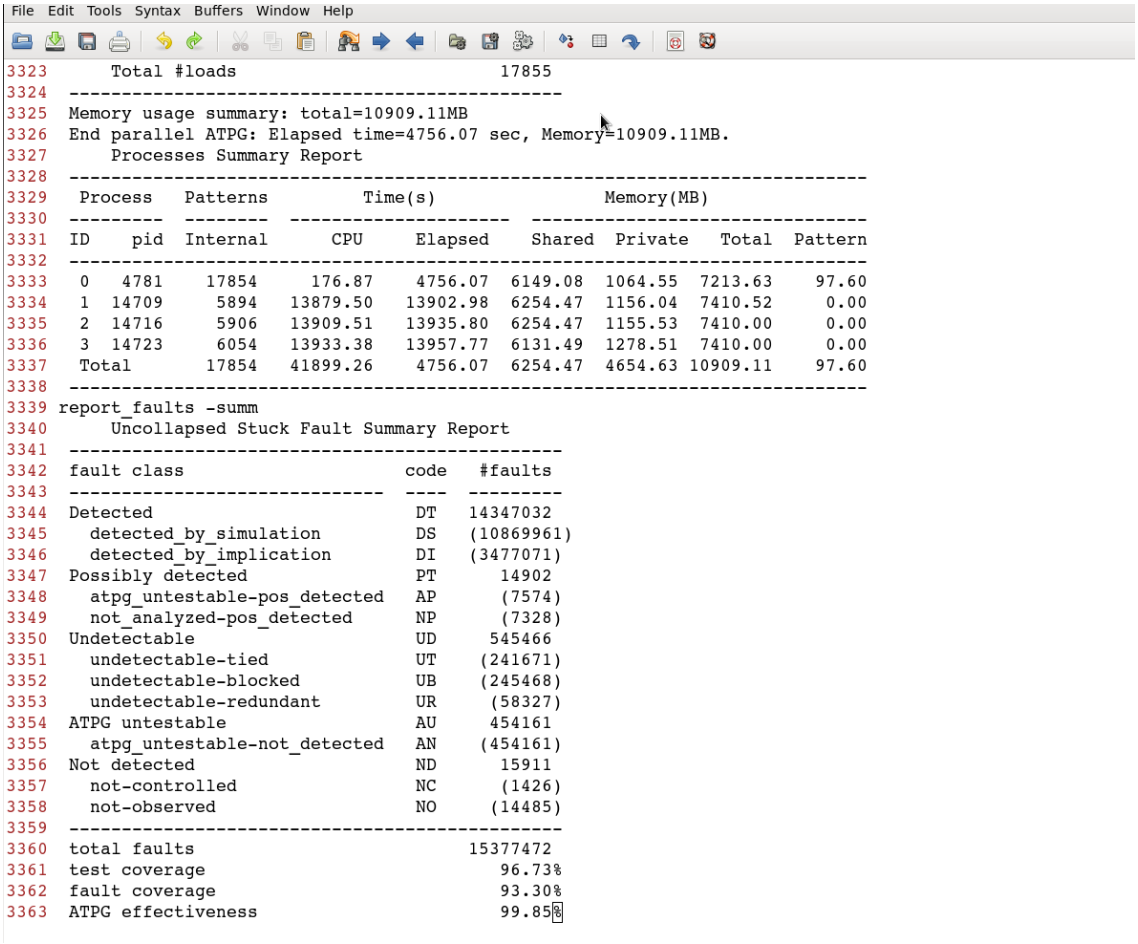
ATPG can be analyzed after the test patterns are generated. This includes the information about the number of test pattern stored, active faults, test coverage and CPU process time.

```
TEST> run_atpg
ATPG performed for 72436 faults using internal pattern source.
-----------------------------------------------------------
#patterns #faults       #ATPG faults test     process
stored    detect/active red/au/abort coverage CPU time
--------- ------------- ------------ -------- --------
Begin deterministic ATPG: abort_limit = 5...
32        49465 22971   0/0/1        70.05%   6.50
64        6808  16163 0/0/3        77.82%   10.52
96        3779  12380   1/1/4        82.13%   13.48
128       2220  10156  2/2/6        84.66%   16.02
160       1264   8890   4/2/7        86.11%   18.54
192       1415   7474   4/3/11       87.73%   20.87
224  1021      6450   6/4/13       88.89%   23.04
256        835   5610   9/6/17       89.85%   25.17
288        722   4881  13/8/19       90.68%   27.20
320        653   4223  15/11/21      91.43%   29.16
352        572   3648  16/13/26      92.08%   31.15
:          :     :     : : :          :        :
831         78    378 176/105/132    95.69%   62.35
862         73    295 184/107/142    95.78%   64.08
889         49    212 205/113/143    95.87%   65.35
```

Figure 3.4-2 ATPG Analyzation

- **Reviewing Test Coverage**

Test coverage results can be viewed by using command `report_summaries`. The tool shows the test coverage results with fault classes. This view can be changed by using command `set_faults -report collapsed` to see a collapsed view.



```
File  Edit  Tools  Syntax  Buffers  Window  Help
3323      Total #loads                          17855
3324  ------------------------------------------------
3325  Memory usage summary: total=10909.11MB
3326  End parallel ATPG: Elapsed time=4756.07 sec, Memory=10909.11MB.
3327      Processes Summary Report
3328  ------------------------------------------------------------------------------
3329   Process    Patterns          Time(s)                Memory(MB)
3330  ---------  --------  --------------------  --------------------------------
3331  ID    pid  Internal      CPU     Elapsed    Shared  Private   Total  Pattern
3332  ------------------------------------------------------------------------------
3333   0   4781    17854     176.87    4756.07  6149.08  1064.55  7213.63    97.60
3334   1  14709     5894   13879.50   13902.98  6254.47  1156.04  7410.52     0.00
3335   2  14716     5906   13909.51   13935.80  6254.47  1155.53  7410.00     0.00
3336   3  14723     6054   13933.38   13957.77  6131.49  1278.51  7410.00     0.00
3337   Total      17854   41899.26    4756.07  6254.47  4654.63 10909.11    97.60
3338  ------------------------------------------------------------------------------
3339 report_faults -summ
3340      Uncollapsed Stuck Fault Summary Report
3341  -------------------------------------------------
3342  fault class                     code   #faults
3343  ------------------------------  ----  ---------
3344  Detected                         DT   14347032
3345    detected_by_simulation         DS   (10869961)
3346    detected_by_implication        DI   (3477071)
3347  Possibly detected                PT      14902
3348    atpg_untestable-pos_detected   AP      (7574)
3349    not_analyzed-pos_detected      NP      (7328)
3350  Undetectable                     UD     545466
3351    undetectable-tied              UT    (241671)
3352    undetectable-blocked           UB    (245468)
3353    undetectable-redundant         UR     (58327)
3354  ATPG untestable                  AU     454161
3355    atpg_untestable-not_detected   AN    (454161)
3356  Not detected                     ND      15911
3357    not-controlled                 NC      (1426)
3358    not-observed                   NO     (14485)
3359  -------------------------------------------------
3360  total faults                          15377472
3361  test coverage                            96.73%
3362  fault coverage                           93.30%
3363  ATPG effectiveness                       99.85%
```

Figure 3.4-3 Test Coverage Results

- **Writing ATPG Patterns**

After generating patterns and attaining required test coverage. The patterns are now required to be stored in a file in a specific format. These stored patterns are used for further simulation and post-silicon testing.

Command: `write_patterns patterns.stil -serial -format stil`

The above command is will store patterns in STIL format. As described above, this format is human readable.

Command: `write_patterns patterns.bin -format binary -replace`

This command will store patterns in binary format. This format is read by the tool only.

```
#set_atpg -capture_cycle 4
#reset_au_faults
#run_atpg -auto


####################################
## FAULT REPORTING
####################################
report_patterns -type -all > reports/${ATPG_PASS}_patterns.rpt

report_faults -level 3 100
report_faults -level 9 100 > reports/${ATPG_PASS}_level9_coverage


####################################
## WRITE FAULTS  & COVERAGE
##
####################################
# report_faults infinium_top -summary -all -level 9 32 > REPORTS/SA_compress_pass1_debug_1000_coverage_9_32.rpt
write_faults ./faults/${ATPG_PASS}.faults.gz -replace -all -compress gzip

####################################
## WRITE PATTERNS & Image File
##
####################################
#write_image reports/${ATPG_PASS}_trans_image_final.dat -replace
if {($SAVE_PAT=="yes") && ($chain_test=="yes")} {
echo " Info :- Saving chain test patterns"
write_patterns ./patterns/BIN/${ATPG_PASS}.bin.gz -format bin -compress gzip -replace
write_pattern patterns/STIL/${ATPG_PASS}.stil.gz -format stil -rep -compress gzip -unified_stil_flow -parallel
write_pattern patterns/STIL_SPLIT/${ATPG_PASS}.stil.gz -format stil -rep -compress gzip -unified_stil_flow -parallel -split 7000
write_testbench -input ./patterns/STIL/${ATPG_PASS}.stil.gz  -output ./patterns/MAXTB/${ATPG_PASS}  -replace -config_file ./patterns/MAX_TE
    OGS/maxtb_${ATPG_PASS}_chain_stil_ver.log -verbose}
}
if {($SAVE_PAT=="yes") && ($chain_test!="yes")} {
write_patterns ./patterns/BIN/${ATPG_PASS}.bin.gz -format bin -compress gzip -replace
write_pattern patterns/STIL/${ATPG_PASS}.stil.gz -format stil -rep -compress gzip -unified_stil_flow -parallel
write_pattern patterns/STIL_SPLIT/${ATPG_PASS}.stil.gz -format stil -rep -compress gzip -unified_stil_flow -parallel -split 7000
write_testbench -input ./patterns/STIL/${ATPG_PASS}.stil.gz  -output ./patterns/MAXTB/${ATPG_PASS}  -replace -config_file ./patterns/MAX_TE
    OGS/maxtb_${ATPG_PASS}_stil_ver.log -verbose}
}
date
remove_licenses Test-Compression-ATPG

quit
```

Figure 3.4-4 Command File showing write pattern command

3.5 SAMPLE COMMAND FILE

Following is a basic example of command file to generate test patterns.

```
# --- basic ATPG command sequence
#
set_messages log last_run.log -replace
##--- read design and libraries
#
read_netlist spec_design.v -delete
read_netlist /home/vendor_A/tech_B/verilog/*.v -noabort
report_modules -summary
report_modules -error
##--- build design model
#
run_build_model spec_top_level_name
report_rules -fail
##--- define clocks and pin constraints
#
add_clocks 1 CLK MCLK SCLK
add_clocks 0 resetn iosc14m
add_pi_constraints 1 testmode
```

```
##
--- define scan chains & STIL procedures, perform DRC checks
#
run_drc spec_design.spf
report_rules -fail
report_nonscan_cells -summary
report_buses -summary
report_feedback_paths -summary
##
--- create patterns

set_atpg -abort 20 -pat 1500 -merge high
add_faults -all
run_atpg -auto_compression
report_summaries
##
--- save fault list and patterns
#
report_faults -level 5 64 -class au -collapse -verbose
write_faults faults.all -all -replace
write_patterns patterns.v -format verilog -parallel 2 -replace
#
exit
```

3.6 Issues During Pattern Generation

During pattern generation, there are a lot of things which are needed to be taken care of before starting the process. Initially, there can arise many problems in pattern generation. But with time these problems can be eliminated. During my internship time, I faced following problems during pattern generation.

- DRC is performed before the generation of test patterns. As discussed before, there can occur some design violations during the scan insertion process. During DRC, some of the violations may not be shown as errors, instead, those violations are shown as warnings. Due to which, the process continues and incorrect patterns may be generated and can only be detected during the simulation process. Hence, after generating patterns, log files should be checked very carefully. The warning was checked in the log file and report was given to the design team to eliminate that violation.
- The second problem that occurred was a failure to achieve the required test coverage. This was due to the reason that a maximum number of test patterns to be generated was set to a limit of 1000 and required coverage was 99.5%. Due to which the tool was unable to achieve the required test coverage. One solution to this problem was to increase the number of test patterns.

But that could not be applied as more test patterns will take more storage space. So, it was decided to create a coverage file in which the data is stored about the coverage achieved in a particular module. The plan was to detect some modules in which the number of fault sites are more and coverage is less and generate patterns for those modules differently. By generating patterns for these modules, the overall test coverage can be increased very easily. To detect these kinds of modules from the file, a Perl script was generated which will automatically detect module with higher fault sites and lower test coverage.

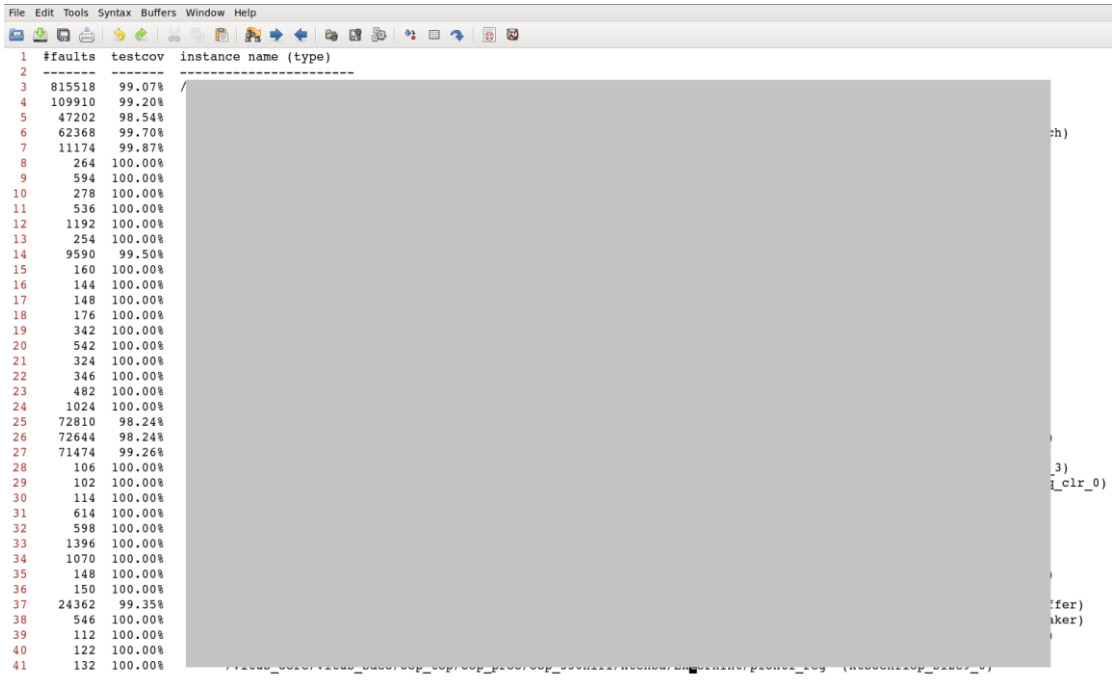


Figure 3.6-1 Coverage File Generated

The following diagram shows a log file showing a successful simulation of sample patterns.

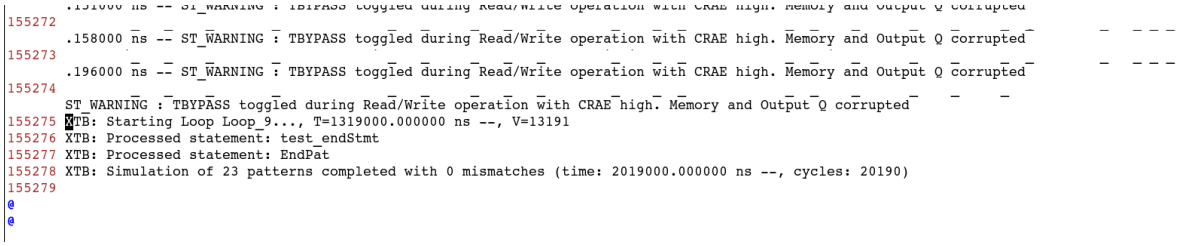


Figure 3.6-2 Successful Pattern Generation

CHAPTER - 4

SIMULATION RESULTS

4.1 INTRODUCTION TO SIMULATION

Simulation is a software modelling of hardware design, its functions, and performance. Simulation helps in understanding the working of the design in a practical environment. The patterns generated by the tool are simulated on software first. This will ensure that the patterns are right and design is working as expected. Using standard delay format (SDF) files, the timing information can be given to the simulator so that design can be simulated in a more practical way. If there is any error in the design or generated patterns that can be detected during the simulation. Hence, simulation is a very important and cost-effective step and generating patterns for the design.

4.2 SIMULATION IN DFT

For simulation purpose, Cadence NCSIM was used. To run simulations, a proper setup of files is required. If there is any file missing in the setup, it can result in the failure of the simulation. The most important files that are required for simulation are netlist, libraries, testbench and a force file. Force file contains the defaults values of some pins of the design. User can force default values on pins for required time using force file. Simulation basically is a three-step process i.e. compilation, elaboration, and simulation.

4.2.1 Compilation

Before starting the simulation, all the Verilog files including netlist, testbench, and design libraries are checked for syntax errors. If there is any error in the syntax of files, the tool will show errors and simulation process cannot proceed. After removing the syntax errors, further simulation steps can proceed which are elaboration and simulation.

The command to compile the VHDL files for the design is:

```
ncvhdl [options] vhdl_source_files
```

To compile Verilog files the command used is:

```
ncvlog [options] verilog_source_files
```

4.2.2 Elaboration

In the elaboration process, module instances are bounded one by one. This will build a model hierarchy, computes parameter values, resolves hierarchical names and prepares all for simulation. Sometimes, if there is any file missing in the setup then it can cause an error in this step as it can interrupt the hierarchy of the modules. But files SDF files will not cause any error during the simulation.

To start the elaboration process, the command used is as follows:

```
ncelab [options] top_level_design_unit
```

4.2.3 Simulation

After successfully compiling and elaborating the design model simulation is executed. The testbench provides the stimuli from the vector file. Patterns can also be there in the testbench. But for complex designs, it becomes difficult to store patterns in the testbench.

To run the simulation, the command used is as follows:

```
ncsim [options] snapshot_name
```

4.3 TYPES OF SIMULATIONS

In DFT, simulation is done in three different modes. These modes are based on the delay values provided to the design model.

4.3.1 Zero Delay (notim) Simulation

Notim simulation is done before the best case and worst case. This simulation is done without any delay values. There is no need to add SDF files in the setup to run this simulation. Notim simulation takes lesser time to run as compared to other simulations. In this simulation, the design is tested in ideal conditions. If this simulation is successful then it will be sure that the design is working properly.

4.3.2 Best Case Simulation

Best case (BC) simulation is run using the minimum possible value of delay in the design at different points of the design. Basically, this simulation is done to ensure if the design is capable to work in delay conditions. This step takes closer to the practical working condition of the design. If there is an error in simulation and very difficult to debug then the simulated values can be compared with notim simulation values at the same instant of time.

4.3.3 Worst Case Simulation

In the worst case (WC) simulation, the delay values used in the design are maximum possible delay values. In this simulation, the design model is tried to run in the worst practical conditions. This simulation is very difficult to attain success. Generally, maximum errors occur in this simulation. Once this simulation is successful, there will be surety of the design to run successfully in practical conditions.

Simulation is necessary to save manufacturing cost. Manufacturing does not start until the simulations of the design are successful. This will result in saving manufacturing cost, increase in yield, and saving time.

4.4 SIMULATION USING REGRESSION SETUP

Regression setup is used to simulate all testbenches and fault types at once. Regression setup runs all type of simulations parallelly. Hence, it takes lesser time as compared to standalone simulation. In a standalone simulation, one type of fault can be detected in one simulation run. For example, the user can either run the simulation for stuck-at fault or transition fault. But in regression setup, user can add all type of pattern files, testbenches, worst case SDF file, best case SDF file in the setup and just run the simulation. This simulation is done at the final stage of DFT, to make sure that all the errors have been debugged in stand-alone simulations.

The disadvantage of regression is that debugging becomes slightly difficult. Debugging is easy in stand-alone simulations. Secondly, memory usage is more for regression setup as many kinds of simulations are running in parallel at the same time. Each simulation needs its own memory to run. Also, strong processors are required to run simulations. Following is the README file to run regression setup. It helps to understand the regression setup and how to run it.

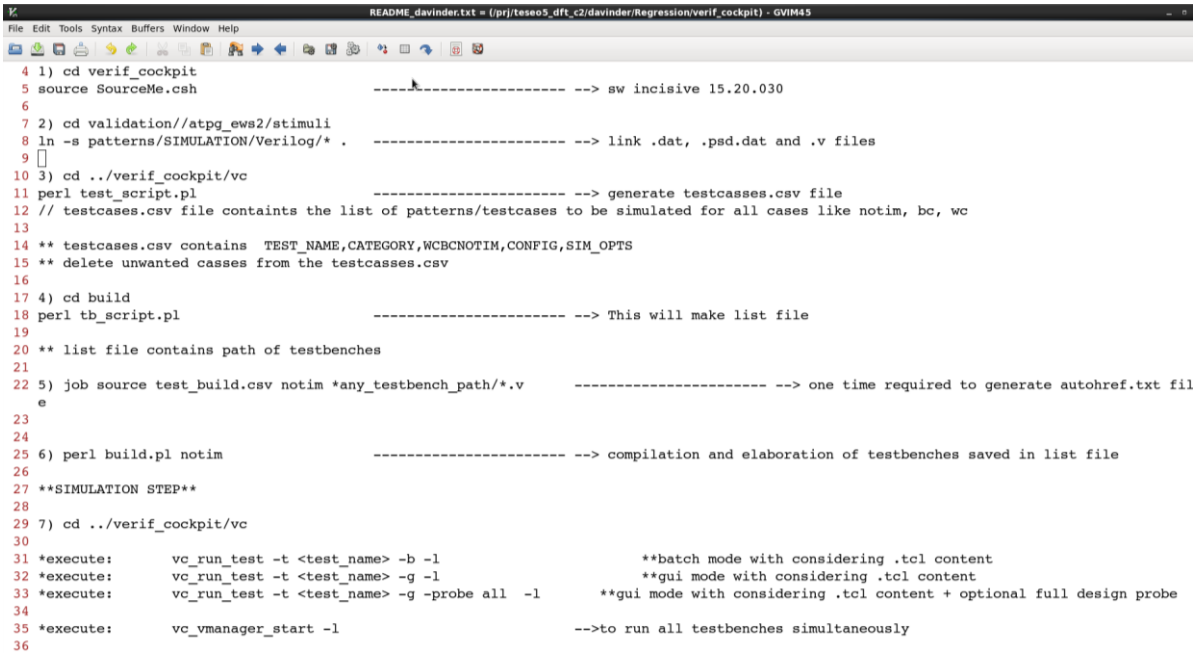


Figure 4.4-1 Regression Guide File

4.5 DEBUGGING SIMULATION FAILURES

Debugging is a very common thing to do in DFT. After running simulation, if there are errors in the simulation then those errors can be removed by debugging. At initial stages, errors in the simulation are very common. With time, those errors are debugged to reduce failures in the design. While performing simulation, the following errors occur generally.

- In the initial state of simulations, the most common error is due to missing libraries. This problem does not show until elaboration when the tool tries to make the design model for the simulation. This error shows up as follows.

ncelab: E,CUVMUR (filenamev): instance 'instance path' of design unit 'and021' is unresolved in 'module path in board directory'.

In this error, the instance 'ix913' of module 'and021' is unresolved. To resolve this problem, a thorough search of the missing module is performed. Clearly, in this problem, library module and021 is missing. The location of the corresponding library module is specified and compilation and elaboration are performed again.

- The second problem occurs due to the mismatches in the expected output and simulated output. This is a very common error during the simulation. The source of error can be anything related to the design. The problem can be due to wrong delay values defined in the SDF file or there can be other reasons. During this error, simulation is run with the waveform. The waveform is used to locate the mismatch and the instance at which the mismatch occurs. This error shows up in the log file as follows:

  At T=4421485.00 ns, V=110538, exp=0, got=x, pin GPIO22, scan cell 4

  The waveform is used to find a mismatch location and its cause. In this error, I am getting 'x' (don't care) at the output instead of '0'at GPIO22 pin. This can be solved by backtracking the schematic and finding the root of mismatch.

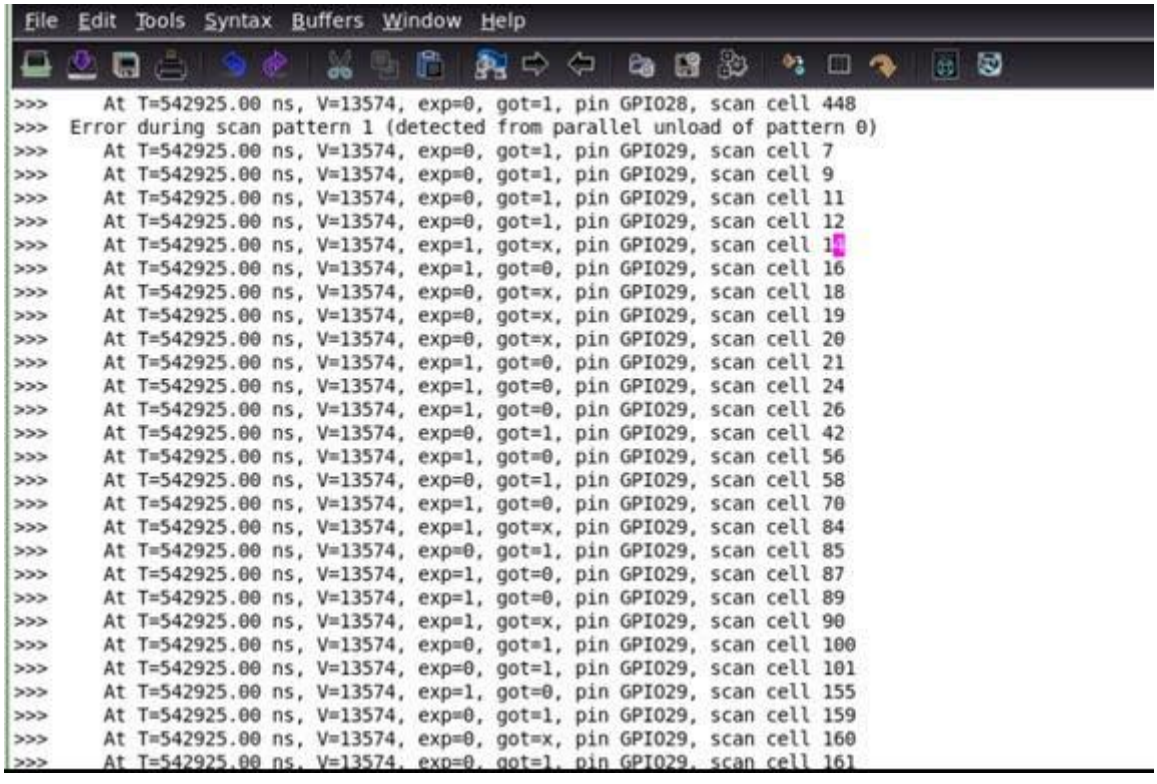


Figure 4.5-1 Log File Showing Mismatch Error

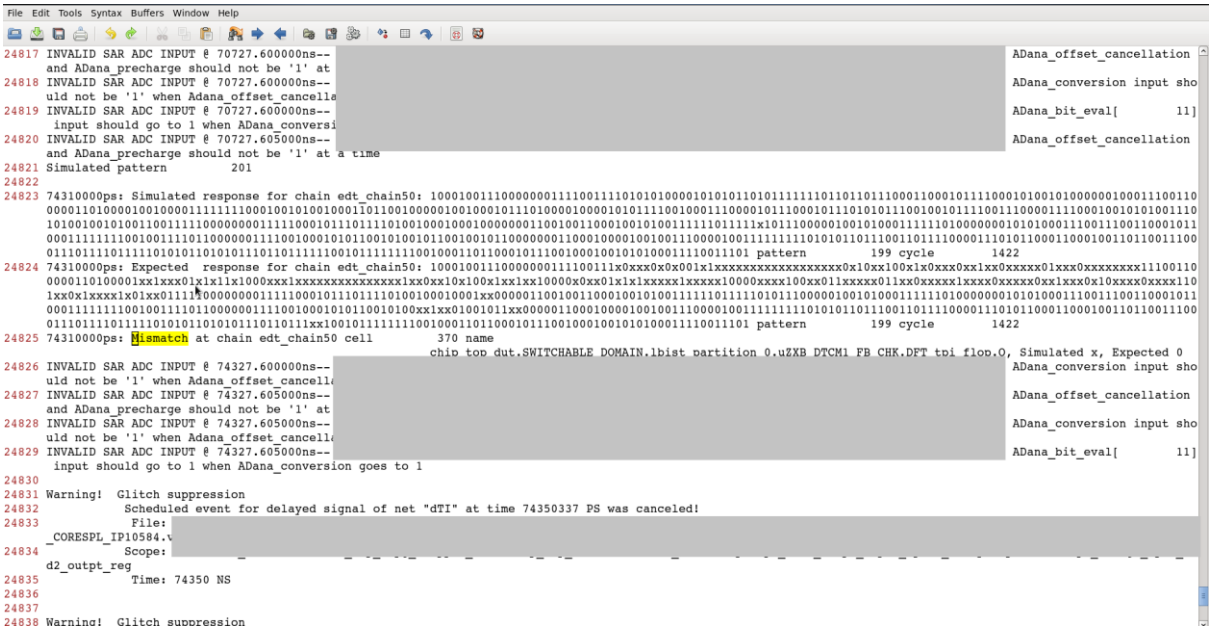


Figure 4.5-2 Unsuccessful Simulation With Mismatches

Figure 4.5-3 Log File For Successful Simulation

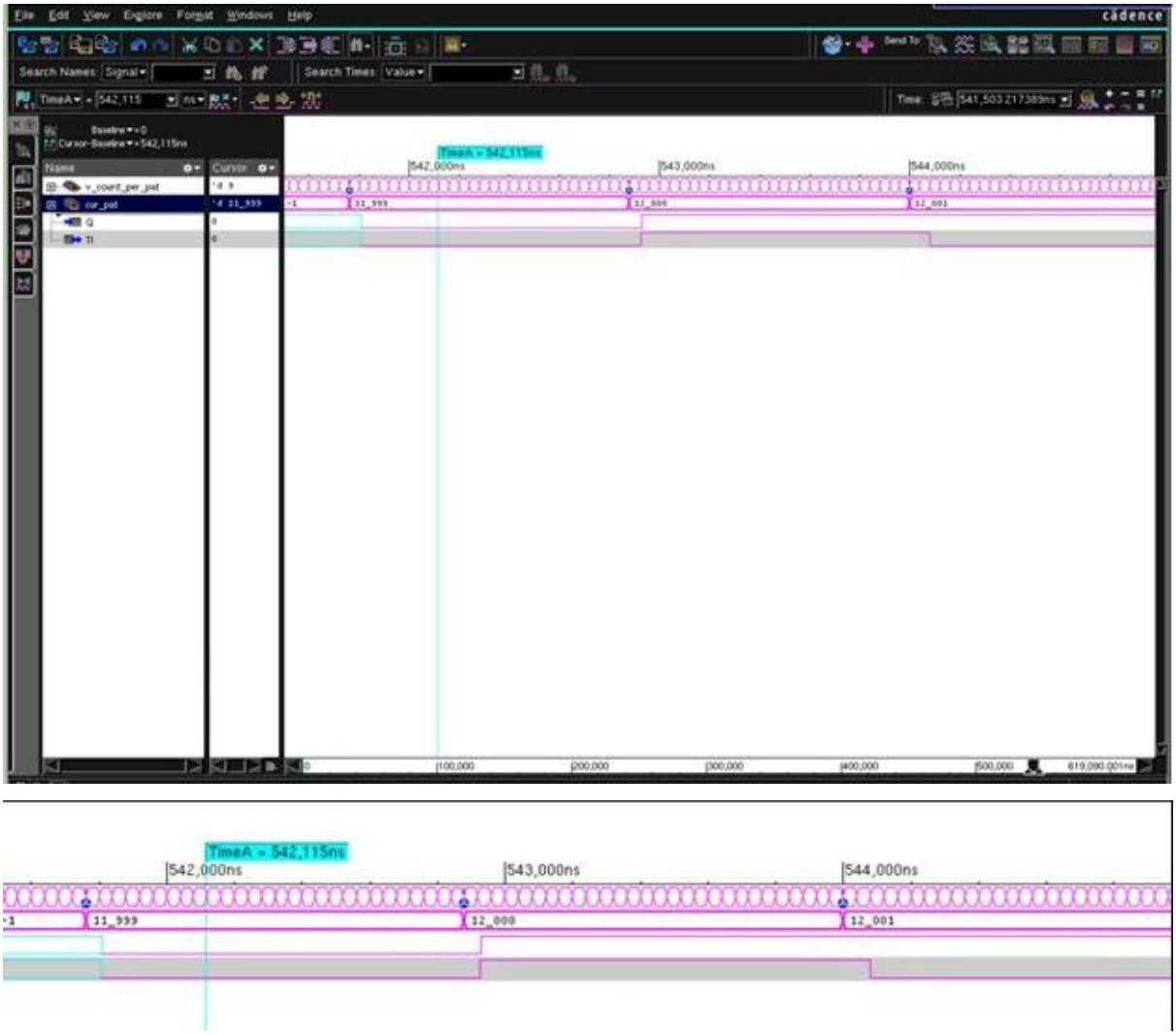


Figure 4.5-4 Dumped Waveform With Simulation

CHAPTER – 5

CONCLUSION AND FUTURE SCOPE

5.1 CONCLUSION

As the technology node is shrinking, more and more faults are coming to life. And more algorithms are being generated to detect them. So DFT has become a very important area. Around 40% of development cost is spent in DFT. Self-testing methods like LBIST and MBIST are getting popular for critical areas. The second technology that is generally used in DFT is ATPG. Both technologies are equally effective in different applications. The use of technology depends on the requirements of the application and best-suited technology is used for the testing.

During my internship period, I learned about simulating test patterns and how to debug them using Cadence NCSIM and SimVision. Also, I have learned ATPG pattern generation for transition and path delay faults using Synopsys Tetramax. Along with ATPG pattern generation, I have also learned the basics of Perl scripting and also made two scripts that help in simulation and pattern generation processes.

5.2 FUTURE SCOPE

Post-production testing is necessary to ensure that chip is not having any manufacturing defect, for which Design for Testability techniques are used from past many decades. Scan-based design for testability is highly used to increase the controllability and observability of the internal nodes of the circuit. Scan chain helps us to test the circuit but on the other hand, it gives access to the internal node of the chip to the unauthorized user which can be considered as a security risk as it can lead to information leakage which retrieves secret key used for encryption and decryption in cryptographic cores. Thus, a new security architecture that prevents unauthorized user to test the chip thus providing security to DFT structure can be proposed.

CHAPTER - 6

REFERENCES

6.2 JOURNALS

1. Mitra, Subhasish, et al. "Delay defect screening using process monitor structures." *22nd IEEE VLSI Test Symposium, 2004. Proceedings.*. IEEE, 2004.

2. Liou, Jing-Jia, et al. "Experience in critical path selection for deep sub-micron delay test and timing validation." *Proceedings of the 2003 Asia and South Pacific Design Automation Conference*. ACM, 2003.

3. Sharma, Manish, and Janak H. Patel. "Finding a small set of longest testable paths that cover every gate." *Proceedings. International Test Conference*. IEEE, 2002.

4. Heragu, Keerthi, Janak H. Patel, and Vishwani D. Agrawal. "Segment delay faults: a new fault model." *Proceedings of 14th VLSI Test Symposium*. IEEE, 1996.

5. Gupta, Puneet, and Michael S. Hsiao. "ALAPTF: A new transition fault model and the ATPG algorithm." *2004 International Conference on Test*. IEEE, 2004.

6. Park, Eun Sei, et al. "Delay testing quality in timing-optimized designs." *1991, Proceedings. International Test Conference*. IEEE, 1991.

7. Sato, Yasuo, et al. "Evaluation of the statistical delay quality model." *Proceedings of the 2005 Asia and South Pacific Design Automation Conference*. ACM, 2005.

8. Lin, Xijiang, et al. "Timing-aware ATPG for high-quality at-speed testing of small delay defects." *2006 15th Asian Test Symposium*. IEEE, 2006.

9. Sato, Yasuo, et al. "Invisible delay quality-SDQM model lights up what could not be seen." *IEEE International Conference on Test, 2005*. IEEE, 2005.

10. Michael L. Bushnell, Vishwani D. Agrawal. "Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits", Springer Science and Business Media LLC, 2002.

11. Scholar, P. Venkata Gopikumar PG. "Fault Detection By Using Random And Deterministic Test Pattern Techniques In A Mixed-Mode BIST Environment."

6.3 WEBSITES

- https://www.eetimes.com/document.asp?doc_id=1216588
- https://www.edn.com/electronics-blogs/other/4348091/Comparing-ATPG-and-BIST
- https://en.wikichip.org/wiki/scan_flip-flop
- https://www.advantest.com/documents/11348/f717e957-0326-42d1-a13c-921059722fc6
- https://anysilicon.com/overview-and-dynamics-of-scan-testing/
- https://en.wikipedia.org/wiki/Design_for_testing

Thesis Davinder Plag

ORIGINALITY REPORT

13%	8%	10%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	acknowledgementsample.com Internet Source	1%
2	Michael L. Bushnell, Vishwani D. Agrawal. "Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits", Springer Science and Business Media LLC, 2002 Publication	1%
3	Rahul Pandey, Sakshee Pandey, C.S Mohammed Shaul Hammed. "Security in Design for Testability (DFT)", 2017 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), 2017 Publication	1%
4	dspace.thapar.edu:8080 Internet Source	1%
5	www.eetimes.com Internet Source	1%
	docplayer.net	

6	Internet Source	1%
7	dejazzer.com Internet Source	<1%
8	manualzz.com Internet Source	<1%
9	Seiji Kajihara. "Evaluation of the statistical delay quality model", Proceedings of the 2005 conference on Asia South Pacific design automation - ASP-DAC 05 ASP-DAC 05, 2005 Publication	<1%
10	"Power-Aware Testing and Test Strategies for Low Power Devices", Springer Nature, 2010 Publication	<1%
11	koczorowska.net Internet Source	<1%
12	K. Heragu, J.H. Patel, V.D. Agrawal. "Segment delay faults: a new fault model", Proceedings of 14th VLSI Test Symposium, 1996 Publication	<1%
13	www.cs.uoi.gr Internet Source	<1%
14	citeseerx.ist.psu.edu Internet Source	<1%
	sina.sharif.edu	

15	Internet Source	<1%
16	people.ee.duke.edu Internet Source	<1%
17	ethesis.nitrkl.ac.in Internet Source	<1%
18	Victor Castano, Igor Schagaev. "Chapter 5 FT Models", Springer Science and Business Media LLC, 2015 Publication	<1%
19	Takashi Aikyo. "Timing-Aware ATPG for High Quality At-speed Testing of Small Delay Defects", 2006 15th Asian Test Symposium, 11/2006 Publication	<1%
20	Kwang-Ting Cheng. "Experience in critical path selection for deep sub-micron delay test and timing validation", Proceedings of the 2003 conference on Asia South Pacific design automation - ASPDAC ASPDAC, 2003 Publication	<1%
21	K CHENG. "Test Technology Trends in the Nanometer Age", VLSI Test Principles and Architectures, 2006 Publication	<1%
	staff.ustc.edu.cn	

22	Internet Source	<1%
23	Ho Fai Ko, , and N. Nicolici. "Algorithms for State Restoration and Trace-Signal Selection for Data Acquisition in Silicon Debug", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2009. Publication	<1%
24	ir.lib.nthu.edu.tw Internet Source	<1%
25	"Fault Modeling", Frontiers in Electronic Testing, 2002 Publication	<1%
26	M. Sharma, J.H. Patel. "Finding a small set of longest testable paths that cover every gate", Proceedings. International Test Conference, 2002 Publication	<1%
27	csserver.evansville.edu Internet Source	<1%
28	M KALYANAPALA. "Programmable Logic Devices", Modern Component Families and Circuit Block Design, 2000 Publication	<1%
29	acsd.ac.in Internet Source	<1%

30	Weng Fook Lee. "Chapter 5 Design for Test", Springer Nature, 2019 Publication	<1%
31	V.D. Agrawal. "Segment delay faults: a new fault model", Proceedings of 14th VLSI Test Symposium VTEST-96, 1996 Publication	<1%
32	Integrated Circuit Authentication, 2014. Publication	<1%
33	es.scribd.com Internet Source	<1%
34	Mohammad Tehranipoor, Nisar Ahmed. "Nanometer Technology Designs High-Quality Delay Tests", Springer Nature, 2008 Publication	<1%
35	sceas.csd.auth.gr Internet Source	<1%
36	"Automatic Test Pattern Generation", Electronic Design Automation for IC System Design Verification and Testing, 2016. Publication	<1%
37	grad.uprm.edu Internet Source	<1%
38	digi.lib.ttu.ee Internet Source	<1%

39	"Testing and Diagnosis of VLSI and ULSI", Springer Nature, 1988 Publication	<1%
40	www.irjet.net Internet Source	<1%
41	Jain, A.. "An improved and simplified design of cosine-modulated pseudo-QMF filterbanks", Digital Signal Processing, 200605 Publication	<1%
42	"Introduction to Hardware Security and Trust", Springer Nature, 2012 Publication	<1%
43	repositories.lib.utexas.edu Internet Source	<1%
44	search.computer.org Internet Source	<1%
45	Yoshinobu Higami. "Test Pattern Selection for Defect-Aware Test", 2011 Asian Test Symposium, 11/2011 Publication	<1%
46	L WANG. "Design for Testability", VLSI Test Principles and Architectures, 2006 Publication	<1%
47	N. Ranganathan, Raju D. Venkataramana. "Integrated Circuits", Wiley, 1999 Publication	<1%

48	Janusz Rajski. "Test Generation for Timing-Critical Transition Faults", 16th Asian Test Symposium (ATS 2007), 10/2007 Publication	<1%
49	V R Devanathan, I S Shah. "Hazard-Aware Directed Transition Fault ATPG for Effective Critical Path Test", 2011 24th Internatioal Conference on VLSI Design, 2011 Publication	<1%
50	www.iti.uni-stuttgart.de Internet Source	<1%

Exclude quotes	On	Exclude matches	< 10 words
Exclude bibliography	On		