Some Investigations on Efficient Testing & Fault Diagnosis Algorithms for VLSI Circuits

Submitted in partial fulfilment of the requirements

for the award of the degree of

Doctor of Philosophy

by

K V B V Rayudu

(Roll No: 701358)

Under the supervision of

Dr. P Sreehari Rao

Associate Professor

&

Dr D R Jahagirdar

Internal Guide, DRDO

Department of Electronics & Communication Engineering

National Institute of Technology Warangal

Telangana, India - 506004

2022

Dedicated

To

My Family,

Gurus & Friends

Approval Sheet

This thesis entitled **Some Investigations on Efficient Testing & Fault Diagnosis Algorithms for VLSI Circuits** by **K V B V Rayudu** is approved for the degree of **Doctor of Philosophy**.

Examiners

Research Supervisor

Dr. P Sreehari Rao
Department of ECE
NIT Warangal, India-506004

Co-Supervisor

Dr D R Jahagirdar
DRDO
Hyderabad, India-500058

Chairman & Head

Dr. P Sreehari Rao
Department of ECE
NIT Warangal, India-506004

Place:

Date:

Declaration

This is to certify that the work presented in this thesis entitled **Some Investigations on Efficient Testing & Fault Diagnosis Algorithms for VLSI Circuits** is a bonafied work done by me under the supervision of **Dr. P. Sreehari Rao**, **Dr D R Jahagirdar** and was not submitted elsewhere for the award of any degree.

I declare that this written submission represents my own ideas and even considered others ideas which are adequately cited and further referenced the original sources. I understand that any violation of the above will cause disciplinary action by the institute and can also evoke panel action from the sources or from whom proper permission has not been taken when needed. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea or data or fact or source in my submission.

Place:

Date:

K V B V Rayudu
Research Scholar
Roll No.: 701358

NATIONAL INSTITUTE OF TECHNOLOGY

WARANGAL, INDIA-506004

Department of Electronics & Communication Engineering

CERTIFICATE

This is to certify that the thesis work entitled **Some Investigations on Efficient Testing & Fault Diagnosis Algorithm for VLSI Circuits** is a bonafide record of work carried out by **Mr. K.V B Vasantha Rayudu (Roll No.701358)** submitted to the faculty of **Electronics & Communication Engineering** department, in partial fulfilment of the requirements for the award of the degree of **Doctor of Philosophy** in **Electronics and Communication Engineering, National Institute of Technology Warangal, India-506004**. The contributions embodied in this thesis have not been submitted to any other university or institute for the award of any degree.

Dr. P. Sreehari Rao
Research Supervisor
Associate Professor
Department of ECE
NIT Warangal, India-506 004.

Place:

Date:

Acknowledgements

I am grateful to many people who made this work possible and helped me during my Ph.D studies. I am greatly indebted to my research supervisor Prof. Krishna Prasad for his support and encouragement in choosing my research topic, who was my initial guide till hisretirement July 2017. I am greatly obliged to my research supervisor Dr. P. Sreehari Rao forgiving me excellent support during my research activity. I am very much thankful for givingresearch freedom, guidance and for the humanity shown to me.

Ever since I met him, he has been an eternal source of motivation, inspiration, encouragement and enlightenment. I specially thank Dr. D. R. Jahgirdar, Scientist-G for being my internal guide at RCI-DRDO lab, providing constant encouragement, guidance and suggestions. The thesis would not have seen the light of the day without his insistent support and cooperation.

I am also grateful to Prof. L. Anjaneyulu, Head of the Department, Dept. of Electronics and Communication Engineering, for his valuable suggestions and support that he shared during my research tenure. I take this privilege to thank all my Doctoral Scrutiny Committee members, Prof. J. V. Ramanamurthy, Department of Mathematics, Prof. C. B. Rama Rao, Professor,Department of Electronics and Communication Engineering for their detailed review,constructive suggestions and excellent advice during the progress of this research work.

I am grateful to the former Heads of the ECE department Prof. N. Bheema Rao and Prof. T. Kishore Kumar for their continuous support and encouragement. I would also appreciate the encouragement from teaching, non-teaching members and fraternity of Dept.of E.C.E. of N.I.T. Warangal. They have always been encouraging and supportive.I am also grateful to Sri. B. H. V. S. N. Murthy, DS and Director of RCI-DRDO, andteam

from RCI work centres for providing VLSI tools to carry out simulation and test set up.

I acknowledge my gratitude to all my teachers and colleagues at various places for supportingand cooperating me to complete this work. I would like to thank my family members (K. Ch. K. Mangatayaru, Ms. K. Phani Jyothi and K.S.L Prashanthi) for giving me mental support and inspiration. They have motivated and helped me to complete my thesis work successfully.

K V B V Rayudu

Abstract

Testing is very important for the diagnosis of faults at an earlier stage for any system, in particular it is critical for the integrated circuits as the replacement is highly expensive. It is very complex for scaled down the technologies due to the non-availability of the mature models. In this thesis, various test and fault diagnosis techniques are studied and modified algorithms for improvements are applied on chosen FinFET based circuits. The effectiveness of Built in Self- Test (BIST) in finding faults is analysed with respect to salient figures of merit including maximum fault coverage, speed, power dissipation, and test area overhead leading to improved design performance.

PODEM (Path Oriented Decision Making) algorithm is employed to test FinFET based combinational circuits to find out fault location efficiently at node level. GA is employed to identify the Path Delay Faults (PDF). An attempt is made to reduce the test power dissipation while testing in BRAMs by applying advanced extensible interface BIST (AXI BIST). Vedic march algorithm is applied to find corner faults in the RAM corresponding to the RISC processor. ROBDD technique is applied to find maximum number of faults optimizing area and power dissipation while testing address decoder of RAM in RISC processor.

A systolic array multiplier 94X4 is designed with reversible gates and Built-In Logic Block Observer (BILBO) logic is used for fault injection. Further, in this thesis suitability of PODEM algorithm for fault location is explored It is found that the application of AXI BIST results in significant improvement in speed (74%) and consumes power by 50%. Fault coverage ($> 95\%$) is maximized through ROBDD. A reduced set up time could be achieved (2.3 ns) through selected Vedic algorithms.

Contents

Declaration ii

Acknowledgements iv

Abstract vi

List of Figures x

List of Tables xiv

List of Abbreviations xv

1 Introduction 1

1.1 Motivation 2

1.2 Research Objectives 3

1.3 Thesis Contributions 4

1.4 Thesis Organization 4

2 LITERATURE SURVEY 6

3 Testing and Fault diagnosis in FinFET Combinational Circuits 18

3.1 Non-incremental genetic algorithm 18

3.1.1 Design of NAND gate using FinFET model 21

3.1.2 Non-incremental computation genetic algorithm for NAND gate . . 22
3.1.3 Results and discussion 24
3.1.4 Critical Path Delay Calculations 27
3.1.5 PDF of Critical path 27
3.2 PODEM (Path-Oriented Decision Making) 30
3.2.1 Functional Modules used in PODEM 31
3.2.2 Implementation using PODEM 33
3.2.3 Results and Summary 33
3.2.4 Conclusion 36

4 Testing and fault diagnosis of sequential logic circuits 39

4.1 32-bit RISC architecture 39
4.1.1 Implementation of VMA on 32-bit RISC processor 39
4.1.2 Pseudo code for VMA 40
4.1.3 Results Analysis 42
4.2 ROBBD implementation on sequential circuits 45
4.2.1 Stuck at fault (SAF) 50
4.2.2 Line at the Single stuck 51
4.2.3 Design for detection of the faults using different test vectors 51
4.2.4 Results and discussion 52
4.3 Systolic Array Multiplier (SAM) 54
4.3.1 Design and Testing of SAM with BILBO 55
4.3.2 Results and Discussion 57
4.4 Testing and fault diagnosis of ALU Blocks using Advanced BIST Algorithms 61
4.4.1 Implementation Architecture of AXI with MARCH-A Algorithm . . 61

4.4.2 Results and Summary 62

4.5 Conclusion 65

5 Conclusions and Future Scope **66**

5.1 Conclusions 66

5.2 Future Scope 67

Publications **68**

Bibliography **70**

List of Figures

2.1 a) Shorted gate b) Independent gate FinFETs [1] 7

2.2 Boolean function representation using ROBDD 8

2.3 Or-based representation using ROBDD 8

2.4 Built-In Self-Test (BIST) Block Diagram 12

2.5 write procedure in AXI protocol 14

2.6 Read procedure in AXI protocol 14

2.7 Centralized BIST architecture 15

2.8 Distributed BIST architecture 15

2.9 Reversible modified Islam gate 16

2.10 Reversible controlled operational gate 17

2.11 Reversible full adder or full Subtractor 17

3.1 NAND gate design test circuit 1 19

3.2 NAND gate design test circuit 2 20

3.3 Flow diagram for the proposed method 20

3.4 NAND gate design using FinFET library. 21

3.5 Flow chart for PDF estimation using NGA. 23

3.6 NAND GATE 24

3.7 VTC Curve Delay calculations for NAND Gate 25

3.8 Propagation Delay (tp) for NAND Gate 25

3.9 NOR GATE 25

3.10 VTC Curve Delay calculations for NOR Gate 26

3.11 Propagation Delay (tp) for NOR Gate 26

3.12 NOT GATE 26

3.13 VTC Curve Delay calculations for NOT Gate 26

3.14 Propagation Delay (tp) for NOT Gate 27

3.15 Graph for finding the fitness value in genetic algorithm 27

3.16 Critical path selected (circuit 1) 28

3.17 Weighted graph for critical path (circuit 1) 28

3.18 PDF Graph of time delay 28

3.19 CRITICAL PATH1 Selected(circuit 2) 29

3.20 CRITICAL PATH2 Selected (circuit 2) 29

3.21 Weighted graph for circuit 2 29

3.22 PODEM Algorithm Flowchart 31

3.23 Work flow of objective 34

3.24 Work flow of back trace 35

3.25 PODEM Algorithm Test Circuit 36

3.26 PODEM Algorithm 37

3.27 Propagation Delay (t_p) for NAND Gate 37

3.28 Fault value is Not detectable and detectable 37

4.1 proposed method of 32 bit RISC Architecture 40

4.2 Indication of existing and proposed march algorithm elements 40

4.3 Processor with Vedic march algorithm block 42

4.4 32-bit RISC processor with I addressing format 42

4.5 32-bit RISC processor with J addressing format 43

4.6 32-bit RISC processor with R addressing format 43

4.7 Read and write opeartion of RAM 43

4.8 Performance chart for Modern approach testing compared with existing technologies 43

4.9 Symbol of D-flip flop 45

4.10 Simplification to binary decision diagram 46

4.11 Simplification to reduced ordered binary decision diagram 47

4.12 Normal mode operation of the master-slave edge-triggered flip-flop 48

4.13 Test mode operation 1 of the master-slave edge-triggered flip-flop 48

4.14 Test mode operation 2 of the master-slave edge-triggered flip-flop 49

4.15 Identified test patterns in the design 50

4.16 Identified test patterns in the Design for detection of the faults using different test vectors 52

4.17 Normal operation of DFF 52

4.18 Stuck at '1' operation of DFF 53

4.19 Bridge fault (loss of connection) 53

4.20 Toggle fault waveform 53

4.21 Area covered using configurable logic blocks 53

4.22 Routed design in FPGA 54

4.23 Multiplier cell block 55

4.24 Proposed system with DUT and all required components 56

4.25 Full environment and testing with proposed systolic array multiplier using fault injection schemes 57

4.26 Results of reversible systolic array multiplier using pattern generator from Bilbo logic design 57

4.27 Results Systolic Array Multiplier 58

4.28 Results of SAM internal blocks COG and MIG gates 58

4.29 Results of Missing gate fault 58

4.30 Results of BILBO LFSR Mode 59

4.31 Results of BILBO MISR Mode signature comparison 59

4.32 BIST Block Diagram 61

4.33 Read and write operations of March A for 15 clock pulses 62

4.34 Flow chart for AXI BIST 63

4.35 Read and write operations of March A for (9n+1) clock pulses 63

4.36 RTL Schematic 64

4.37 Valid addressing forwrite and read 64

4.38 Serial write and read 64

List of Tables

3.1 Gate Characteristic Parameters 21

3.2 Results Comparison 30

3.3 Device ID and Node of net list 32

3.4 Results Comparison 38

4.1 Results Comparison 44

4.2 Excitation table for D-flip flop 46

4.3 Final results taken from the test mode operation 1 and 2 49

4.4 Results Comparison 54

4.5 Results Comparison of Fault Analysis 60

4.6 Results Comparison of Local Utilization 60

4.7 Results Comparison 62

List of Abbreviations

ADF	Address Decoder Fault
ALU	Arithmetic Logic Unit
AXI	Advanced Extensible Built In Self Test
ATPG	Automatic Test Pattern Algorithm
AGA	Adaptive Genetic Algorithm
BIST	Built In Self Test
BILBO	Built-In Self-Test Block Observer
BRAM	Block Random Access Memory
BDD	Binary Decision Diagram
BF	Bridging Faults
CBIST	Concurrent BIST
CUT	Circuit Under Test
CTL	Concurrent Test Latency
CMOS	Complementary Metal Oxide Semiconductor
CF	Coupling Fault
COG	Controlled Operational Gate
CLFSR	Configurable Linear Feedback Shift Register
DUT	Design Under Test
DFF	Delay Flip Flop
FINFET	Fin Field Effect Transistor
FPGA	Field Programmable Gate Array
nm	nanometer
ns	nanosecond
μm^2	Square Micrometer
LFSR	Linear Feedback Shift Register

MIG	Modified ISLAM Gate
MUX	Multiplexer
MSAF	Multiple Stuck At Fault
MISR	Multi Input Signature Register
MOSFET	Metal Oxide Semiconductor Field Effective Transistor
NGA	Non-Genatic Algorithm
NIGA	Non Incremental Genetic Algorithm
NPSF	Neighbourhood Pattern Service Faults
NCV	Non-Controlling Valve
ORA	Output Response Analyzer
PD	Propagation Delay
PODEM	Path Oriented Design Making
QC	Quantum Cost
RISC	Reduced Instruction Set Computer
ROBDD	Reduced Order Binary Decision Diagram
RTL	Resistor Transistor Logic
RTPG BIST	Transparent Random Pattern Generator Bist
RVM	Reversible Vedic Multiplier
SAM	Systolic Array Multiplier
SAF	Stuck At Faults
CF	Coupling Fault
COG	Controlled Operational Gate
CLFSR	Configurable Linear Feedback Shift Register
DUT	Design Under Test
DFF	Delay Flip Flop
FINFET	Fin Field Effect Transistor
FPGA	Field Programmable Gate Array
nm	nanometer
ns	nanosecond
μm^2	Square Micrometer
LFSR	Linear Feedback Shift Register

Chapter 1

Introduction

Now a days VLSI Devices have almost reached boundaries of Moore's law. The feature size reached almost sub 7nm. With increased complexity in design, higher frequency of operation, larger device density and more reliable performance lead to multi core device technologies posing several challenges in testing [2, 3] to find correctness of design meeting specifications and fault free operation facilitating defect free delivery of chips. Diagnosis and early detection of the faults reduces cost almost ten times that may escalate at every subsequent state of percolation as per rule of Ten [4]. Testing ensures enhanced yield with improved quality and reliability though it is cumbersome and not feasible at device level. Testing and Fault diagnosis has become very important constituent for VLSI circuit designs. Also, design for testability has become mandatory [5] to increase the controllability and observability. Towards this end, several failure mechanisms are diagnosed and testing algorithms including built in self-test (BIST) [6] are being adopted to achieve maximum fault coverage, reduced power consumption and area overhead without compromising speed.

The scaled down technologies suffer from substantial leakage current [7]. This problem is addressed through FinFET technology. The structure of the FinFET is compact, thin, and sophisticated.The fin is sandwiched between the front as well as the back of the gate to suppress the short channel effect. FinFET are considered as one of the most feasible multi-gate devices [1, 8] by adopting a simple manufacturing process and having good compatibility with planar MOSFET [9]. FinFET based circuits become more effective while performing numerous switching operations that require high speed, reduced

power even though they occupy 94% of the chip area [10]. For diagnosing critical faults, There are various types of fault models available in FinFET's [11]. On the other hand, fault modeling for single planar was investigated extensively through bridging, stuck-at, delay and stuck-open faults [12]. These CMOS fault models [13] are not adequate for covering all defects in FinFET logic gates. Mostly, the stuck-at fault model is used to detect 80% of the faults. Yet, the testing and bridging of delay faults become critical by scaling of the technology even though the presence of defects is observed in the behavior of both combinational and sequential circuits [14]. The statistical timing analysis [15] is performed to analyze and predict the efficacy of the delay test in the circuits and develops innovative progress to identify a fault in combinational logic and sequential circuits. This chapter begins with motivation, research objectives, followed by the key contributions and organisation of thesis is presented.

1.1 Motivation

The processing steps of VLSI circuits are extremely complex and costly inducing vendors to stress on more and more testability as a requirement tool to assure the reliability and the functionality of each of their designed circuits. Various testing, fault diagnostic techniques and heuristic algorithms are being employed to find out stuck at faults, delay faults, bridging faults, toggle or transition faults missing gate faults etc., to achieve max faultcoverage, less power consumption, increased speed of testing, smaller area over head. Incremental computation algorithm with max operator [16] is employed to find out target path delay fault PDF that suffers from limitation in computation time of testing. SSTA was performed using Skew Normal Canonical Model [17] to mitigate non skewness in gate delay distribution in Planar CMOS based Circuits. Non-incremental Genetic Algorithm instead of Monte carlo or Max operator was preferred to find Target Path Delay Fault PDF of FinFET based VLSI circuits to achieve fast computation time.

PODEMX algorithm [18], deductive fault simulation algorithm [19] D-Frontier algorithms are employed for fault detection (stuck at 0/1) and locations in the planar CMOS circuits. This has motivated us to implement PODEM algorithm on FinFET based Combinational circuits for fault detection (stuck at 1/0) and also faultlocation at node level.

Various BIST Algorithms viz., adaptive low power RTPG BIST [20], parallel transparent BIST [21], concurrent BIST [22] for testing memory blocks suffered from fault coverage loss, reduced speed of testing and higher power consumption. Hence to mitigate these drawbacks, AXI BIST Interface technique was launched by ARM in 2011 that used to incorporate parallel read or write operations to achieve high speed of testing, reduction in power, smaller area overhead with optimum fault coverage for testing memory blocks. Different March algorithms are used in [23] while detection of faults in RAM's was done using March C algorithms in [23] that take more time to read back data from CPU. While it offers possibility of overriding, there is no provision for testing and verify corner or primary faults.

Re-converging path delays were employed for detecting multiple faults [24] in combinational circuits. ROBDD based path delay techniques [25] and SAT based APG techniques for multiple stuck-at-faults suffer from drop in fault coverage. Reversible logic design FPGA implementation of optimised 32-bit Vedic Multiplier and Square Architecture was proposed using reversible gates for normal ALU operations [26]. For Multiplier [27] aiming area optimisation used one Vedic sutra but was constrained by controllability. Parity preserving logic-based fault tolerant reversible ALU [28] also suffers from poor controllability. The systolic array multiplier architecture circuit using reversible logic [29] and [30] demand more hardware overhead and meagre controllability. This further motivated us to improve these SAMs to be more robust with enhanced computing capability.

1.2 Research Objectives

This thesis aims to perform testing and fault diagnosis on VLSI circuits with improved speed, fault coverage, reduced power and area constraints.

- To find faults: Stuck at faults (SAF) and Multiple SAF (MSAF), Delay Faults (DF), Transition Faults (TF), Bridging Faults(BF), Fault Detection or Location and Missing Gate Faults(MGF).
- To design target circuits: FinFET based Combinational circuits and CPU sub-blocks

- To employ non-incremental computing genetic algorithm (NGA), PODEM, AXIBIST, VEDIC sutras or Algorithms (Vedic March Algorithm), ROBDD, BILBO fault injection

1.3 Thesis Contributions

The key contributions of this work are summarized as follows:

- Non-incremental genetic algorithm is presented to test and diagnose delay faults in FinFET based circuits
- PODEM algorithm was applied for detection and identification of node level faults.
- Modified AXI based BIST was applied for testing memory architectures in an attempt to reduce area over head and improve speed.
- Systolic array multiplier using fault injection, BILBO schemes was designed using reversible logic. Stuck at fault and missing gate fault is found out using fault injection model.
- Modified Vedic algorithm was applied to test memory of RISC Processor. March-C algorithm was modified to find out corner defects.
- Reduced Ordered Binary Decision Diagram (ROBDD) was applied in BIST mode by designing sequential and combinational blocks to find address decoder faults in memory block of RISC processor faults with reduced area overhead.
- Reversible logic gates are used for reducing power consumption and testing is done using for ALU controller or logic blocks

1.4 Thesis Organization

The rest of the thesis is structured as follows:

Chapter 2 Presents literature survey and a brief overview of algorithms for testing FinFET based combinational and sequential circuits.

Chapter 3 Presents algorithms for testing, fault diagnosis on FinFET based combinational circuits using non-incremental genetic algorithm and application of PODEM algorithm for fault detection.

Chapter 4 Presents application of Vedic march algorithm along with ROBDD for memory block of RISC processor, SAM and simple memory testing for ALU using advanced BIST alogrithm.

Chapter 5 Summarizes result and present conclusion along with scope of further research .

Chapter 2

LITERATURE SURVEY

The circuits designed with an advancement of the transistor structure in the shape of fins named as FinFETs [8] are perpendicular to the structure of the wafer which carries current considerably. The structure of the FinFET is compact, thin, and sophisticated [9] as its size is less than the channel length, and the existing fin is sandwiched in between the front as well as the back of the gate to suppress the short channel effect. FinFETs are considered as one of the most feasible multi-gate devices [31] by adopting a simple manufacturing process and having good compatibility with planar MOSFET. Furthermore, the existing structures in the traditional use of planar bulk equipment of sub 22nm with narrow channel behaviour developed to limit leakage current were replaced by the double gate FinFET fabrication process.

The conventional MOSFETs are used for the scaling of power which is an issue due to the short channel effect [31]. FinFET based circuits become more effective while performing numerous switching operations that require high speed, reduced power even though they occupy 94% of the chip area. For diagnosing critical faults, an innovative testing model is developed for FinFET circuits. Plenty of fault models are available for FinFETs [32] to associate faults in the design of circuits in various applications. At various levels of abstraction, fault modeling for the planar single gate CMOS is explored extensively [33]. For example, bridging the stuck-open faults and stuck at delay faults are the widely utilized fault models for CMOS helps in developing physical defects models which are performed at high abstraction levels. Mostly, the stuck-at fault model is used to detect 80% of the faults. Yet,the testing and bridging of delay faults become critical by

scaling of the technology even though the presence of defects is observed in the behavior of FinFET INV and NAND gates [33]. The dimensions and parameters of MOS transistor models are varied according to the technology. Due to the better gate control over the channel the MOSFETs are replaced with FinFETs. Moreover, the structure of FinFETs is very thin to suppress the short channel effects occurring in the device. The double gate FinFETs [34] are simple to manufacture and compatible to MOSFETs with reduced leakage current which are considered in the designing of multi-gate devices [34]. Various fault models are available for FinFET circuits to model the planar single gate CMOS is shown in Fig 2.1.

The technology advancement also leads to the process variability of manufactured circuits. The process variability makes the interconnected parameters modeled as random variables and the uncertainty is measured in the environment condition such as temperature, voltage etc.

The evaluation of critical path delay fault on planar silicon device-based circuits was proposed in [16] using non-incremental computing algorithm with Max operator. While performing variation aware test generation [16] corresponding to planar CMOS devices at gate level, the incremental computation of delay fault detection probability that utilizes larger computational time. However, its efficiency is less in terms of CPU run time and average speedup. By considering gate delay and arrival times at different nodes in planer CMOS circuits, which are limited by speed, a skew normal canonical delay model [17] was proposed for statistical static timing analysis (SSTA). An unpartitioned Level Sensitive Scan Design (LSSD) structure was presented by merging multiple test algorithms in PODEM-X [18], which is identified only at faults but not at location. A single stuck fault detection algorithm [19] was employed for testing digital circuits using

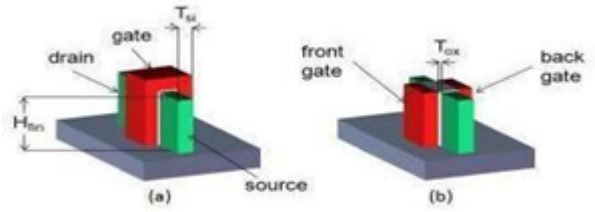


Figure 2.1 a) Shorted gate b) Independent gate FinFETs [1]

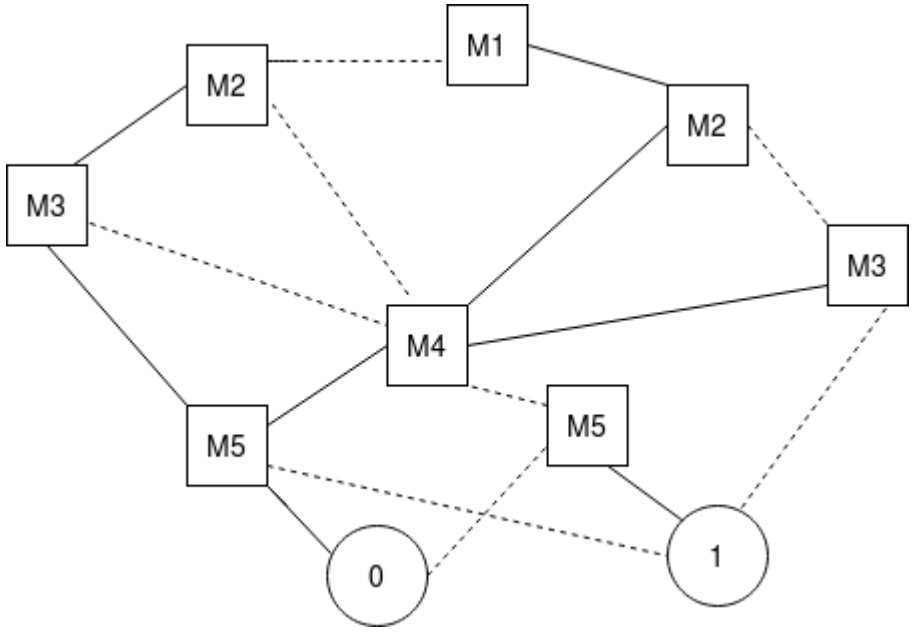


Figure 2.2 Boolean function representation using ROBDD

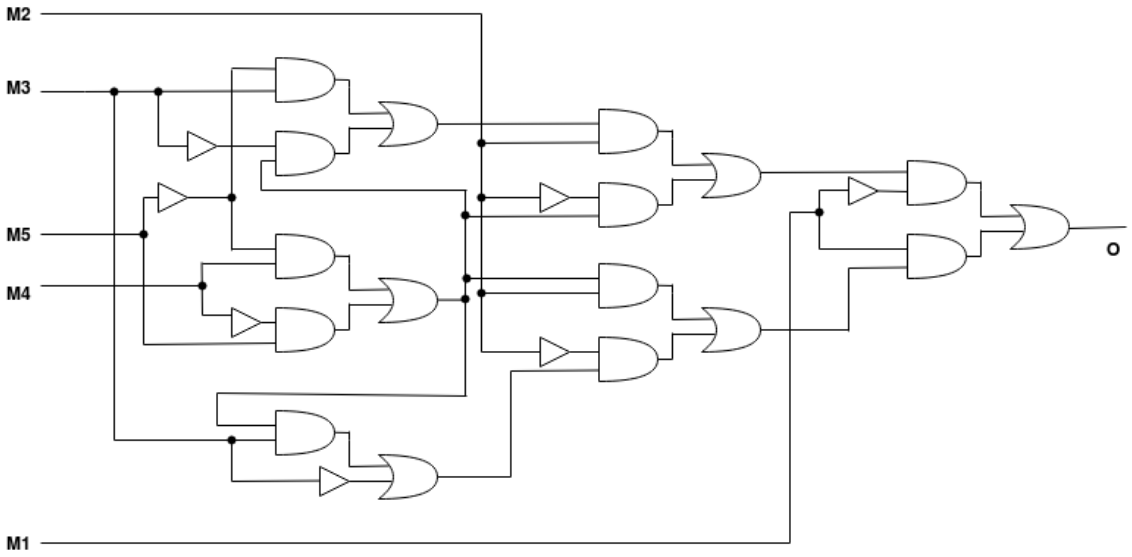


Figure 2.3 Or-based representation using ROBDD

a deductive fault simulator detecting stuck faults at circuits but unable to identify gate or node faults.

An automatic test generation algorithm: D-Frontier [19] with backtracking technique was proposed for finding stuck-at faults and coverage but fault location is not addressed. March test algorithms consist of a fixed sequence of March elements which are used to perform specific read write operation sequences, to detect the faults in memories [23]. Detection of multiple faults using converging path delays was proposed in [24] for combinational logic networks and detection of MSAF. Low Power Random Test Pattern Generator (LPRTPG), a scan-based test proposed in [35] attained a better trade-off between power reduction and test coverage loss. An efficient SAT-based ATPG Technique [36] was proposed to find Multiple Stuck At Faults (MSAF) in combinational circuits by pseudo intensive technique and it is non-synthesizable. Detection of multiple faults using converging path delays was proposed in for combinational logic networks and detection of MSAF. March test algorithms consist of a fixed sequence of March elements which are used to perform specific read write operation sequences, to detect the faults in memories [23]. For example, the representation of a sequence as $\uparrow(r_0, w_1)$, where r_0 and w_1 represents March primitives. The reading or writing representations can be performed with respect to the order like increasing ($\uparrow$), decreasing ($\downarrow$), or both ($\updownarrow$). The primitives 'r_0' represents reading '0' from a cell, 'r_1' represents reading '1' from a cell, 'w_0' represents writing '0' to a cell, 'w_1' represents writing '1' to a cell respectively. The common faults occurring in the memories are classified into the following types: Stuck at Faults (SF): The memory bits are always stuck at a logic 0 or 1.

- Transition Faults (TF): These faults occur when the memory bit fails to transit from one logic to another at the clock cycle.
- Coupling Fault (CF): The fault exists when attempting to write to a cell it changes the value of an adjacent cell.
- Address Decoder Fault (ADF): This fault arises when the cell accessing becomes corrupted. Passing an address line with an address sometimes accesses no memory or sometimes accesses multiple locations.
- Neighbor-hood Pattern Sensitive Faults (NPSF): This is a kind of coupling fault

that acts on multiple memory cells at a time.

The March-A algorithm starts with write-0 followed by read-0, write-1, and read-1 respectively which takes 15 ns to complete. Standard System-on-Chip (SoC) designs [37] include multiple cores surrounded by other cores requires a large amount of memory for executing programmes that conduct peripheral connectivity, mixed signal operations, and other tasks. Regardless of development duration, circuit testing is necessary to any architecture with N nets that contains 3N-1 defects. To improve circuit's efficiency, many configurations were proposed in [38] to withstand MSAF (Multiple Stuck At Faults). Moreover internal output and fan-out of designs that are not terminated to 2-level circuits which are fault-free [39]. The Stuck-At-Fault (SAF) algorithms were developed in [40,41] aimed at detecting single circuit defects. In addition, the Vedic march algorithm is used to test memories that are integrated with the processor, paving the path for future sophisticated system designs [42]. MSAFs suggest an improbable amount of defects that are derived from Single SAFs (SSAF) to cause long estimation durations [43]. Fan-outs eventually inherit the proposed designs in [44–46] to identify MSAFs in the circuit. When the MSAF is integrated into the Automatic Test Pattern Generator (ATPG), roughly 80% of the defects are found by testing SSAF [36]. When MSAFs are observed in various instances, it can improve the frequent identification of defects in the circuits, which leads to Circuit Satisfiability (SAT) concerns [36].

Reduced Ordered Binary Decision Diagram (ROBDD) [25] was proposed that reduced testing patterns and coverage for path delay fault in combinational circuit design. The reversible arithmetic logic unit for quantum arithmetic proposed in [27] was designed and tested using reversible technology but increased the size of ALU. Parity preserving logic-based fault tolerant reversible ALU [28] was proposed to reduce delay in fault finding using n-bit Vedic multiplier [47] design in ALU. An area-efficient multiplier design using reversible logic for Vedic multiplier [26]. was proposed to attain reduced area in ALU design. Designand implementation of optimized 32-bit FPGA Vedic multiplier and square architectures was proposed in [48] using synthesizable ALU and reversible gates. The design of speed, energy, and power efficiency for digital processors was proposed in [49] using reversible logic-based Vedic ALU, which attained speed efficiency in ALU. An efficient Vedic multiplier with high speed, less complexity, and consuming less area

was presented in [50] using an integrated Vedic MAC unit and multiplier. A multiplier using Vedic mathematics [51] was designed to reduce the area of ALU using reversible logic gates.

A sensible method of pseudo intensive testing proposed by McCluskey et.al [52] used random number generation for BIST models. The pseudo intensive testing has strong relevance to ROBDD structure contains polynomial equalized test phase which are complicated in nature at various circumstances. Moreover, it recognizes every MSAFs include ROBDD [25] based circuit designs which consists of irrelevant test age and non-terminated insufficiencies for test are shown in Fig 2.2. Different tests for stuck-at insufficiencies of the circuit are implemented from the structure as shown in the Fig 2.3. The same structure is not suitable for finding multiple faults in sequential blocks. The possible SAFs are tested in CLBs (Configurable Logic Block) inferred with shared ROBDD in the packaging of FPGA development. The design of a Systolic Array Multiplier (SAM) circuit using reversible logic [53] was designed to increase speed and efficacy in the multiplication of ALU. Built-in Logic Block Observer (BILBO) [54] was proposed to test multipliers using reversible logic. Design and implementation of VLSI SAM for DSP applications using 4-bit logic circuit was presented in [55] for the design of high speed that consume less area, and less complex multipliers. The design of self-testing and self-diagnostic systolic array cells was proposed in [56] for signal processing using one-bit logic multiplication. The low power and high-performance design of multipliers [55,56] are required to process DSP applications. So, the Systolic Array Multiplier (SAM) integrated with pipelining can use multi-dimension multiplication to achieve speed and low power dissipation. Moreover, the gate design in SAM can decrease the delay which is suitable for sorting and convolution techniques. Reversible logic proposed by Charles bennet et. al. [57,58] has an advantage that dissipation of heat is minimized for developing low power and high-speed operations. The structure of the reversible gates [59] is designed as the number of inputs are equal to the number of outputs can improve the performance of the system. A 4x4 SAM [60] is designed with reversible logic gates calculate the partial products and simulations are performed using the design tools. The faults are identified for Baugh wooley multipliers are proposed for high-speed operations and tested with BILBO logic [54]. The fault analysis for multipliers is designed by Lang and Moreno [53] with the conversion of matrix algorithms which form an array structure of the SAM. The algorithms for multiplication

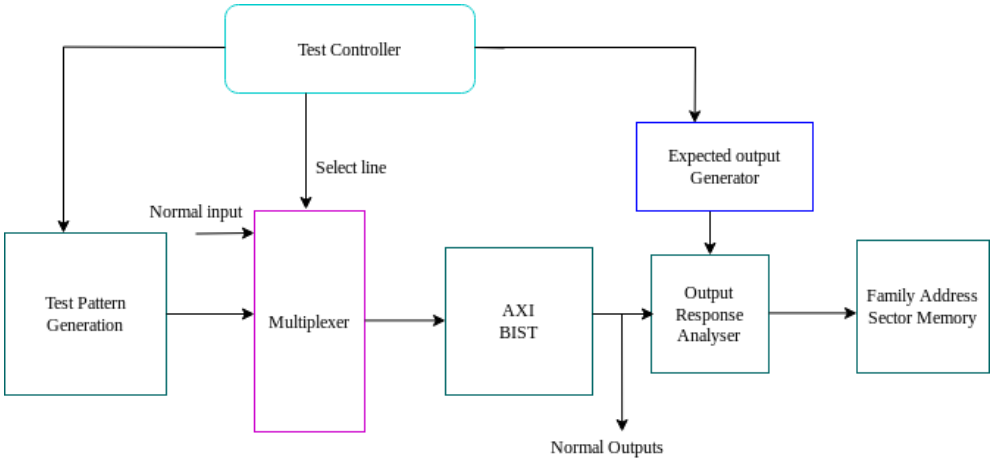


Figure 2.4 Built-In Self-Test (BIST) Block Diagram

and division are implemented and tested with LFSR [61] technique which generates random numbers that are forwarded to shift registers results increase in area of the chip. An irreversible array multiplier [62, 63] used in SAM is implemented in 90nm CMOS technology and achieved high efficiency than compared to the other multipliers. SAM design over Galois Field (GF) multiplier [64–69] generate patterns with 6-bit counter which are required to test the system. However, the delays attained in the circuits using GF multiplier designs are also reduced to some extent in the proposed design. Memory testing is very crucial for applications like aviation and military where the run time faults are leads to the vital failure of the system. The corresponding faults are managed by the Built-In Self-Test (BIST) technique [70] by detecting the faults in memory locations. The conventional BIST technique is shown in Fig 2.3. The major blocks present in the conventional BIST procedure are: Test Pattern Generator (TPG), Output Response Analyzer (ORA), and Test Controller (TC).

The TPG generates the test inputs instructed by the TC based on the range and nature of the inputs to simulate the system. The outputs of the TPG are selected using the multiplexer and given to the Circuit Under Test (CUT). The resultant simulation is matched with expected output in the ORA and analyses the faults present in the memory. Moreover, TPG consists of several test patterns stored in the RAM, a Linear Feedback Shift Register (LFSR) and a counter assisted to ensure whether the system is functioning properly or not. An efficient and Transparent-BIST (T-BIST) method to test multiple embedded memory buffers was proposed in [71], which reduced hardware overhead without

losing fault coverage. Input vector monitoring Concurrent-BIST (C-BIST) [72] to store comparative lo- cations designed with SRAM cells was presented more efficiently than BIST in terms of hardware overhead and Concurrent Test Latency (CTL). The Linear Feedback Shift Register (LFSR) reseeding algorithm to achieve low power dissipation was presented in [61] used a fewer number of transitions in the scan chains without losing fault coverage. Selective trigger scan architecture was proposed by [35] for VLSI testing to reduce switching action in the Circuit Under Test (CUT), that could avoid a large number of transitions and increased the clock frequency in the scanning process.

A high-end model processor which works for all types of instructions was presented in [23] along with a write-back memory model. However, it took more time to read and write back the same data from the CPU. A BIST controller memory technology cite23 with fault processing blocks was proposed for finding faults in the designing and testing of memory blocks. Diagnostic data detection of faults in RAM's using different march algorithms with BIST scheme [73] was proposed for testing of high memory blocks. However, only one read and one write combination is possible for test. BIST techniques based on sequential memory operations require large amount of time to complete the diagnosis. Further, the task becomes more complicated when systems with multiple memory modules of different types are involved. Such cases require multiple BIST algorithms to identify specific faults. This shortcoming can be eliminated by using Advanced eXtensible Interface (AXI) based self-test memory architecture with Block Random Access Memory (BRAM) [74] to achieve parallel read and write capability. AXI is a part of Advanced RISC Machine (ARM) bus architecture specifications and it is a one-to-one interconnection designed for high speed and high-performance microcontroller systems. AXI protocol defines how two devices can communicate with each other inside a microcontroller system. The steps involved in an AXI protocol are:

- Master and slave must handshake to confirm valid signals
- Transmission of control signal must be in separate phases
- Separate channels for transmission of signals

Continuous transfer may be accomplished through burst-type communication shown in Fig 2.5 and Fig 2.6 with read write procedure in AXI protocol. A 32-bit write address is

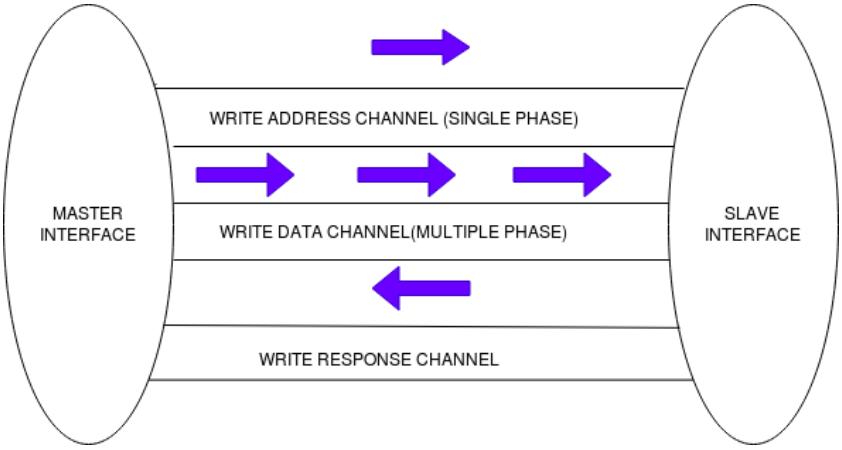


Figure 2.5 write procedure in AXI protocol

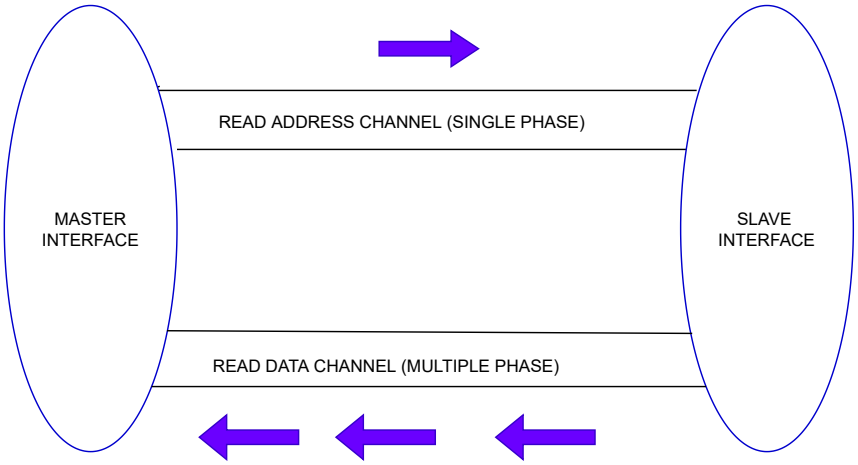


Figure 2.6 Read procedure in AXI protocol

transmitted from master to slave in a single phase. The write data channel is used to write the data in multiphase. The write response channel is used by the slave in acceptance to the write request. WDATA is a signal which is responsible for multiple data to be written. The complete write operation performed as write address, write data, and write response. The read architecture shown in Fig 2.6 can read address in single phase and read data in multiphase mode of operation so that the data can be accessed in parallel. The read operation performed in the order as read address and read response. To verify the faults in memory devices the user can create multiple test cases with known values from available centralized and distributed BIST architectures [6] which are shown in Fig 2.7 and Fig 2.8. The TPG plays an important role by generating proper test cases for testing the address and memory locations.

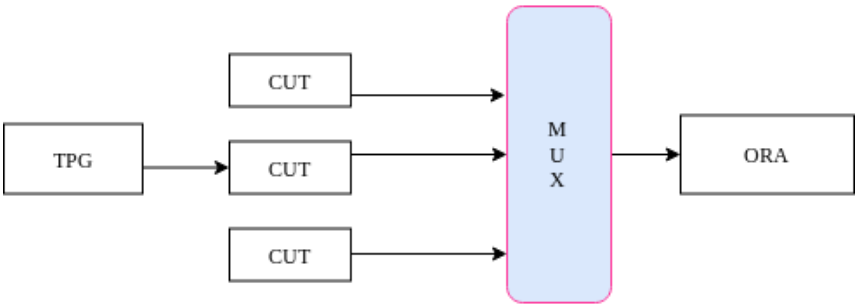


Figure 2.7 Centralized BIST architecture

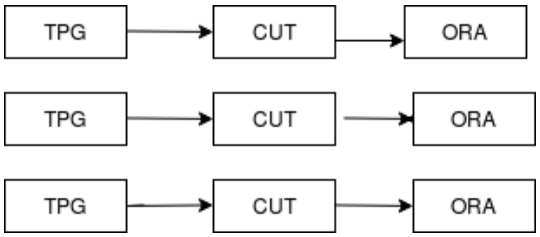


Figure 2.8 Distributed BIST architecture

In conventional BIST circuits, Linear Feedback Shift Registers (LFSR) are used for test pattern generations. The test patterns generated are categorized as deterministic, algorithmic, exhaustive and pseudo-exhaustive. The BIST controller can work in share mode when multiple processors are under test [75]. The Output Response Analyser (ORA) compact the responses of the Circuit under Test (CUT) as signature and compare them with the expected signature value to get the 'Test Result'. A flexible BIST architecture proposed by Reinaldo Silveira et al, [22] performed memory to optimize the existing basic architecture and reduce the area of circuit. The traditional memory testing suggested by Ryan Pennuccietal [76] can replace the conventional memory tests with BIST to lessen the development time. A pseudorandom generator-based Test Pattern Generator (TPG) with weighted single test proposed by Dong Xiang et al, [35] performed the polynomial selection for the TPG and the insertion of new inputs. Low power is achieved through a reseeding scheme from the clock cycles participated in generation. The concepts of periodicity and regularity is introduced in TPG by G. Harutyunyan, et al, [70] created a Fault Periodicity Table (FPT) and test them using Test Algorithm Template (TAT) and verified whether all the faults in the FTP are covered or not. Preethy K John et al, [39] proposed a fault detection algorithm for memory cores using March–C algorithm.

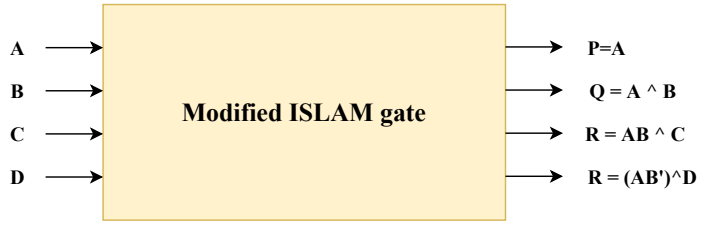


Figure 2.9 Reversible modified Islam gate

A Configurable Linear Feedback Shift Register (CLFSR) is used to generate the addresses to achieve maximum coverage. The March–C algorithm successfully identifies the stuck at faults present in the memory. A BIST based on March algorithm applied on FPGA to verify the SRAM chip proposed by Tan Li et al [77] executed 6-stage march operation followed by a series of read write operations in active mode. A novel BIST technique for FPGA memory fault detection is proposed by Mahesh kumar et.al. [78] can optimize the circuit in terms of area overhead and testing time. The coverage of STFs, CLB faults, bridge faults, wire open and delay faults are identified efficiently.

Arithmetic Logic Unit (ALU) [27] plays an important role in processors to perform mathematical operations. Modern processors integrated with cache memory can increase the speed and efficiency. ALUs are used in most of the circuits like calculators, mobile phones, computers etc. The advancements in the ALU are possible by reversible logic gates [28] shows an advantage of low power dissipation and less delay. Nevertheless, reversible logic is the promising concept in CMOS designs [79] are updated with the technology to implement nano, cryptographic and quantum computingdevices.Reversible logic is proportional to the concept of thermodynamics [57, 58] which aims to lessenthe power dissipation and area. The proposed ALU is focused and compared with the parameters like GC (Gate count), GO (Garbage output), QC (Quantum cost) and PD (propagation-Delay). Reversible gate designs use equal number of input and output variables [80] and drew equal power among fanouts which reduces the quantum cost efficiently. The Modified Islam Gate (MIG) shown in Fig.2.9 is used for full adder design designed by reversible logic gates. It consists of 4-inputs and 4-outputs whereas the output is reflected as full adder model [81].

The Controlled Operational Gate (COG) is shown in Fig.2.10 is also used to design full adder block for low power DSP applications. It consists of equal input and output

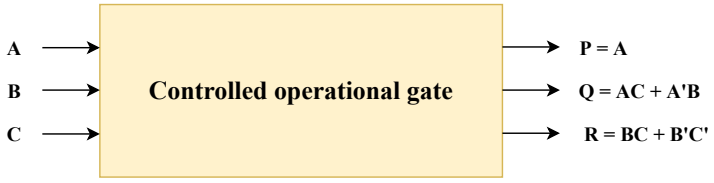


Figure 2.10 Reversible controlled operational gate

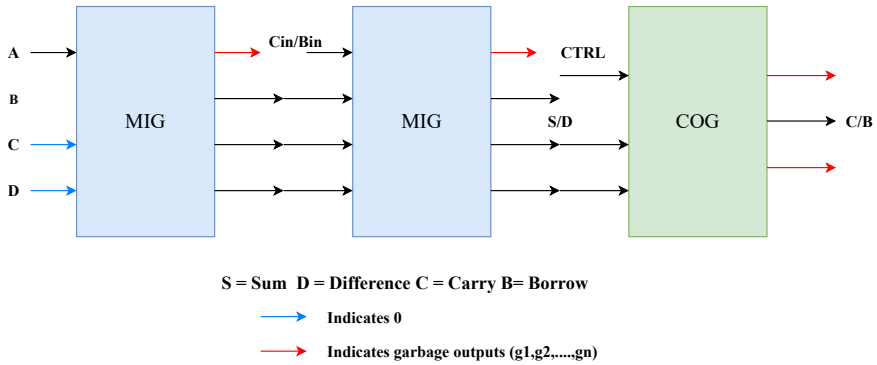


Figure 2.11 Reversible full adder or full Subtractor

variables where the logic completely depends on second and third input variables resulted to produce carrier output. A complete adder and subtractor circuits are utilized in the SAM are integratedwith MIG and COG gates which are used in applications like video, medical and many digital systems. Figure 2.11 represents reversible full adder or full subtractor. Depending upon the selection of inputs various fault models are presented in [82] use BIST as main component which is popular because of its low power and less time execution and fast computing of the complex designs. Compared to all techniques, BIST is more popular because of its low power and less time of execution. The complex designs also get testing done with BILBO because of its usage as main component with differnt operating modes.

Chapter 3

Testing and Fault diagnosis in FinFET Combinational Circuits

This chapter presents efficient test algorithms to diagnose said faults. Also, the combinational logic-based circuits that make use of multiple heterogeneous logic gates are implemented to identify the places having faults. This chapter focuses on application of non- incremental computing algorithm and PODEM (Path Oriented Decision-Making) algorithm for fault detection and location.

3.1 Non-incremental genetic algorithm

The testing and diagnosing of faults are carried out in FinFET circuits and the corresponding fault analysis is proposed using non-incremental genetic algorithm. The design of NAND, NOT, and NOR gates using PTM library in LT Spice and creating a net list for the respective circuit diagram and later imported to MATLAB for further analyzation. Non- incremental genetic algorithm is eventually designed (coded) and run for calculation of critical path delay of mean, variance, and standard deviation values and plotting the PDF graph for the critical path delay for designed circuits in alogrithm 2.1 and verified same with test circuits as shown in Fig 3.1 and Fig 3.2. Flow chart of the proposed method is stated in Fig 3.3, and iterations in Genetic Algorithm (GA) estimate the critical path delay. The critical path delay is the maximum delay between input and output and measured in μs.

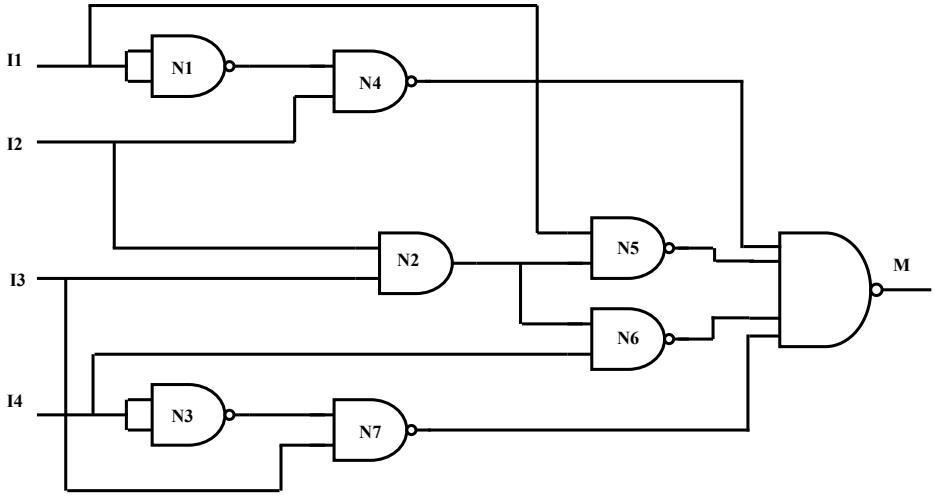


Figure 3.1 NAND gate design test circuit 1

Algorithm 3.1: Genetic Algorithm: GA(n, χ, μ)

Initialize generation $0 : K := 0$;
$PK :=$ **a population of** n **randomly-generated individuals;**
Evaluate PK**:**
compute $\Theta(i)$ **for each** $I \ \varepsilon \ PK$ **do**
Create generation $k + I$:
I**. Copy:**
Select $(l \ X \ z) \ x \ n$ **members of** Pk **and insert into** $Pk + l$;
Mutate::
Select $\mu \ X \ n$ **members of** PK**; invert a randomly – selected bit in each;**
Evaluate $Pk + l$
Compute $\theta(i)$ **for each** $i \varepsilon PK$**;**
Increment:
$K : K + 1$;**}**

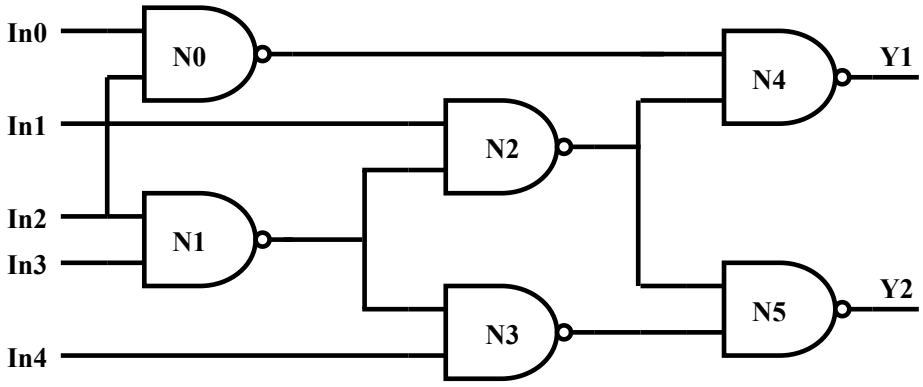


Figure 3.2 NAND gate design test circuit 2

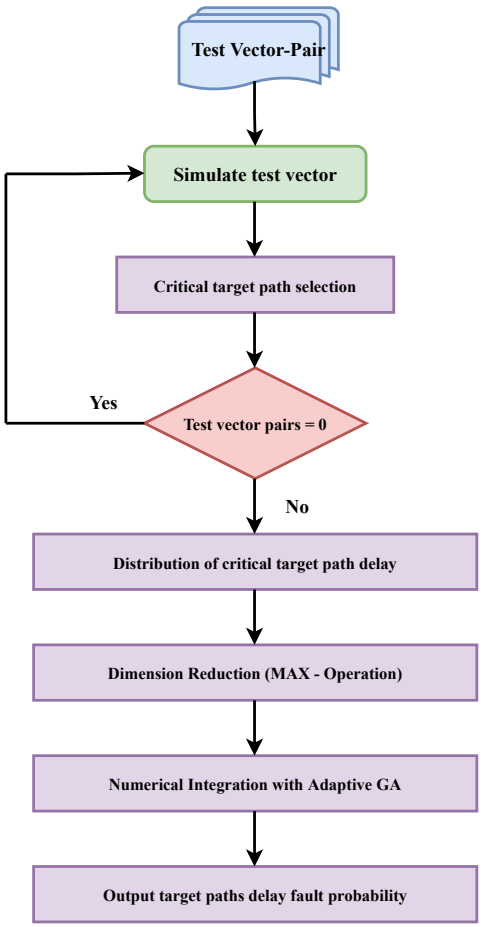


Figure 3.3 Flow diagram for the proposed method

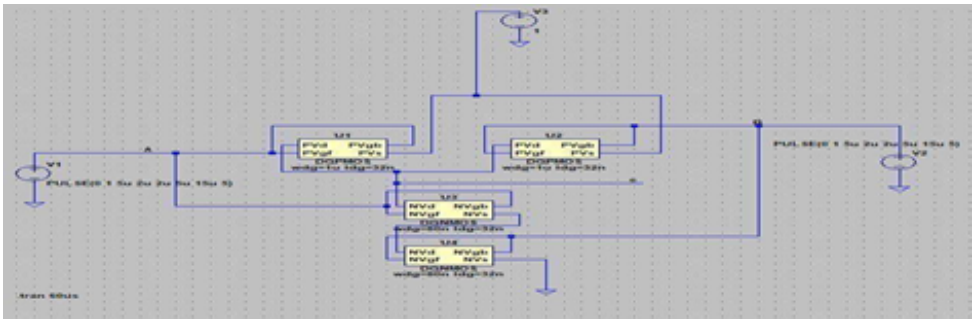


Figure 3.4 NAND gate design using FinFET library.

Table 3.1 Gate Characteristic Parameters

NAND	**NOR**	**NOT**
VOH= 0.9 V	VOH= 0.97 V	VOH= 0.8V
VOL= 0.1 V	VOL= 0.07 V	VOL= 0.66V
NMH= |VOH - VIH| = 0.02 V	NMH= |VOH - VIH| = 0.25 V	NMH= |VOH - VIH| = 0.07 V
NML = |VIL - VOL| = 0.64 V	NML= |VIL - VOL| = 0.513V	NML = |VIL -VOL| = 0.56 V
Propagation Delay (tp): (tpHL + tpLH)/2 = 0.67 μs.	Propagation Delay (tp): (tpHL + tpLH)/2= 0.04 μs.	Propagation Delay (tp): (tpHL + tpLH)/2 =0.54 μs.

3.1.1 Design of NAND gate using FinFET model

The design of NAND gate using FinFET models is shown in Fig 3.4. The supply voltages V_1 and V_2 are considered as input voltage, and V_3 is used as power supply to FinFET model. The graph of the transient and transfer curve needs to be plotted to deduce delay calculations. The simulation parameters V_{th}, V_{IH}, V_{IL}, V_{OH}, and V_{OL} along with the noise margin high (NM_H=|V_{OH}-V_{IH}|) and noise margin low (NM_L = |V_{IL} -V_{OL}|) are calculated for NOR and NOT gates are presented in Table 3.1. After the design, the gates are constructed together to find out the fault analysis and critical path analysis.

3.1.2 Non-incremental computation genetic algorithm for NAND gate

The probability in which one or more target paths are considered to capture the delays presented in the NAND gate is called path delay fault. Clock cycle time T$_c lk$ is smaller than the delay of at least one target path. In spite of finding all the possible ways to calculate the delay path, the computation of the complementary probability is effective and corresponding flow chart of the non-incremental genetic algorithm is shown in Fig 3.14.

ψ^- =P (θ ϵ θ : none of the target paths has a path delay fault) Where, ψ =1-ψ^-

- Step 1: Read the initial test vector pair
- Step 2: Simulate the test vector pair to identify the sensitized path. This is attained by the following way. The circuit instances as well as the test vector pairs are to be simulated. The transmission towards one gate is completed during simulation time, and each transmission is assigned to the gate's output during the same transition and stored as references. After simulation, if any sensitized path occurs, those are identified and is carried till the circuit input is reached. The cross check is performed when there is reference to the transition is present, for that transition then the output of the gate identifies all the output transaction corresponding to the input transaction. The propagation condition of the gate and the path gets terminated is the transition violates and also there is on any output transition.
- Step 3: After identifying the sensitized path by simulation the critical target paths are identified.
- Step 4: The condition checks for the test vector pairs. If more test vector pairs are identified the n returns to Step2. If not, the critical target path delay distribution is computed. The process is explained as below, considering 'n' as the number of critical paths in which the sensitization is performed by the test vectors within a defined subset. The random vector X is given as, X (X$_1$ . . . . X$_n$) and T, where X represent the delays of the critical paths with multi variety normal distribution.
- Step 5: The statistical operation is performed for the dimension reduction and the procedure is continued till a user-defined threshold drops numerous random

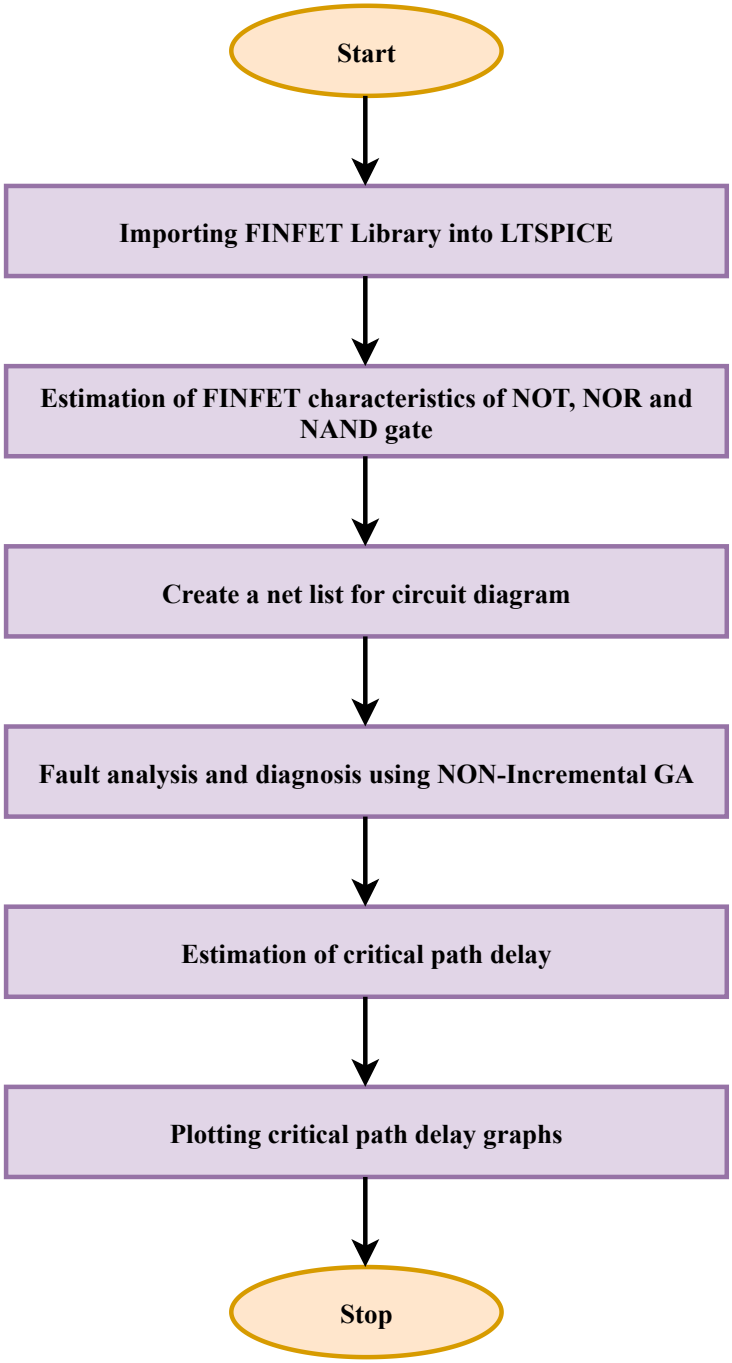


Figure 3.5 Flow chart for PDF estimation using NGA.

variables above it. An $(n-1)$ +wd dimensional normal random vector (X_1 . . . X_{n2}, Y), T approximates the distribution of the random vector is given by as (X_1 . . . ., X_{n2}, max (X_{n1}, X_n) T, with the help of a standard distribution-based MAX function application. Till the multiple variables were dropped under the value of a user defined threshold, the procedure will be continued.

- Step 6: The numerical integration is carried out and the output path delay fault probability is obtained. Let m¡n denotes the remaining random variable numbered (X_1 . . . , X_{m1}, X_m)T N_m (μ, σ) defines the m- dimensional approximation of the maximum delay X.

- Step 7: We incorporate optimization technique for path delay fault optimization. The optimization technique employed here will be adaptive genetic algorithm, which aids in reducing the probability of path delay fault. The optimized results are further processed. The non-computational genetic algorithm helps in finding the path delay by evaluating the fitness function which shows higher fitness function means better solution.

3.1.3 Results and discussion

The simulation parameters for each gate are calculated and transfer characteristics are plotted. The formulas and the calculations for particular gates are found accordingly.that is shown in Fig 3.5, Fig 3.6 and Fig 3.7. Similarly, corresponging NOR and NOT gates of the transient and transfer curve are illustrated in Fig 3.8 to Fig 3.11. From the figures, the parameters obtained are showed in Table 3.1. The Fig 3.8 shows the delay characteristics of the FinFET NAND gate.

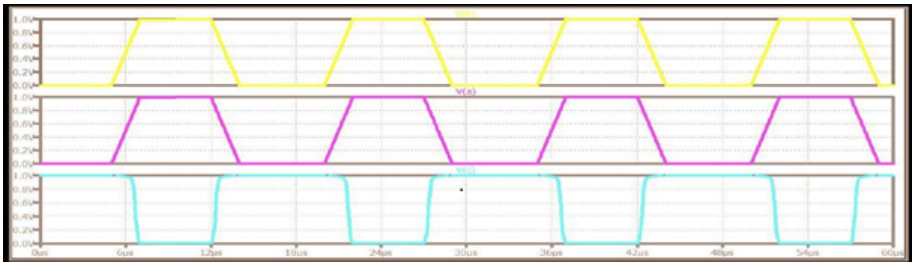


Figure 3.6 NAND GATE

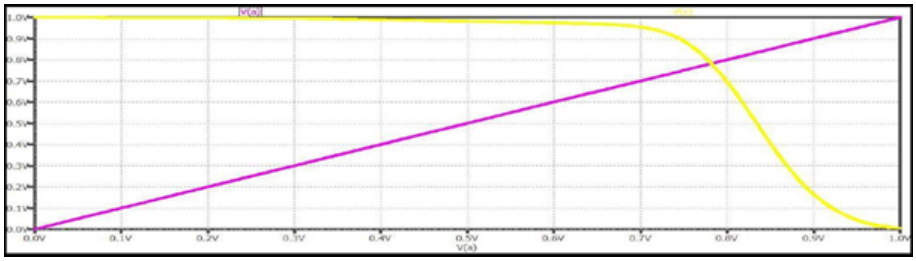


Figure 3.7 VTC Curve Delay calculations for NAND Gate

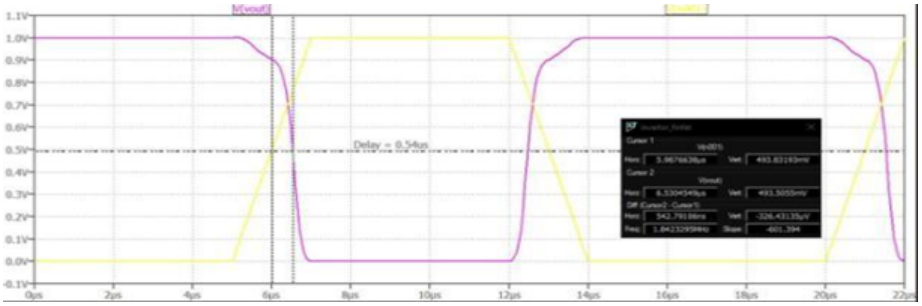
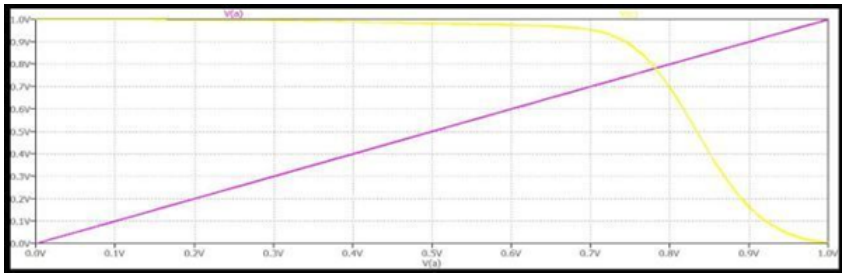


Figure 3.8 Propagation Delay (tp) for NAND Gate



The steadystate values obtained are, high to low propagation delay (t_{pHL}) is calculated which falls from V_{OH} to 50 %, lower to higher propagation delay (t_{pLH}) is calculated which takes to rise from 50% to V_{OL} Propagation delay is calculated as (t_p)= (t_{pHL} +

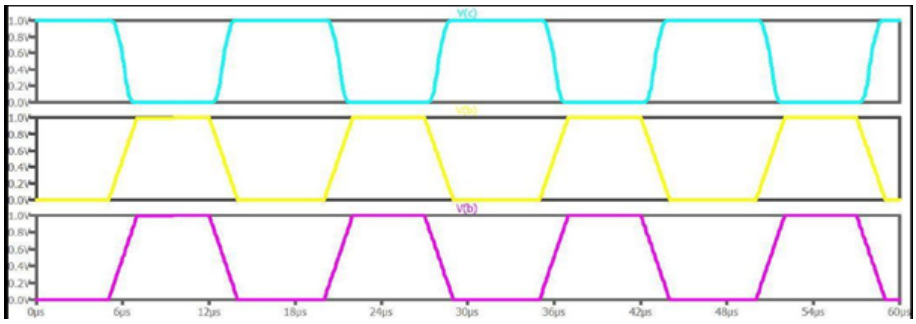


Figure 3.9 NOR GATE

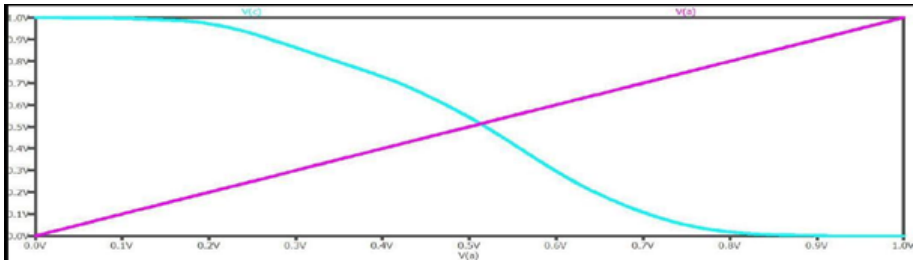


Figure 3.10 VTC Curve Delay calculations for NOR Gate

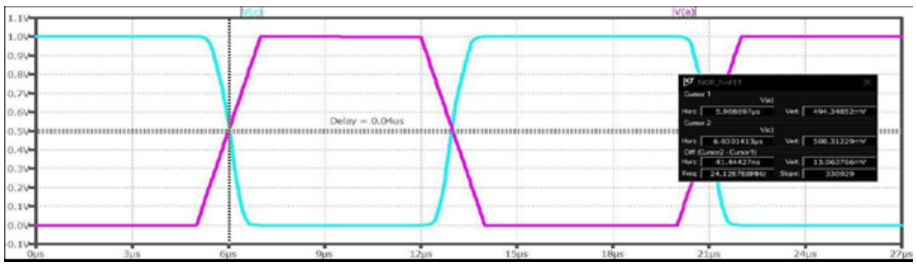


Figure 3.11 Propagation Delay (tp) for NOR Gate

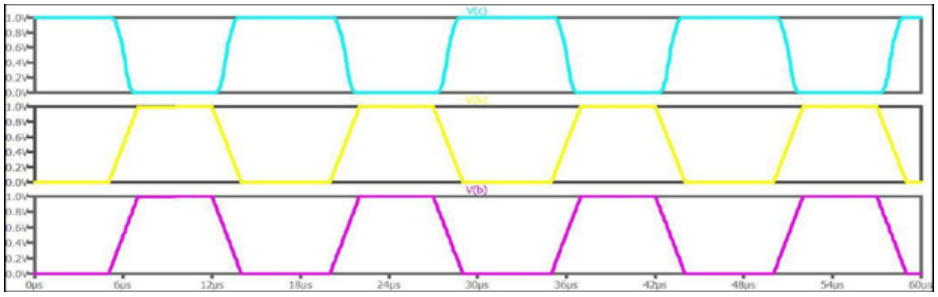


Figure 3.12 NOT GATE

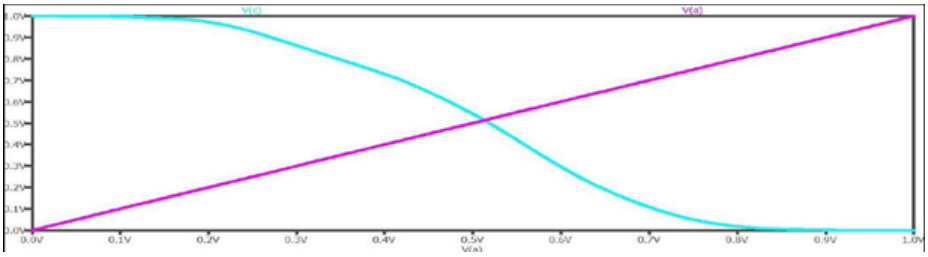


Figure 3.13 VTC Curve Delay calculations for NOT Gate

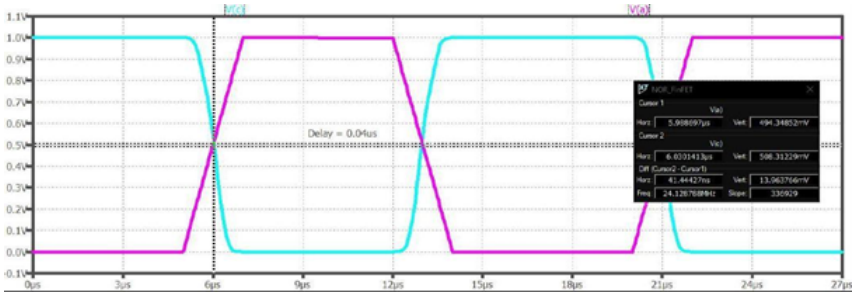


Figure 3.14 Propagation Delay (tp) for NOT Gate

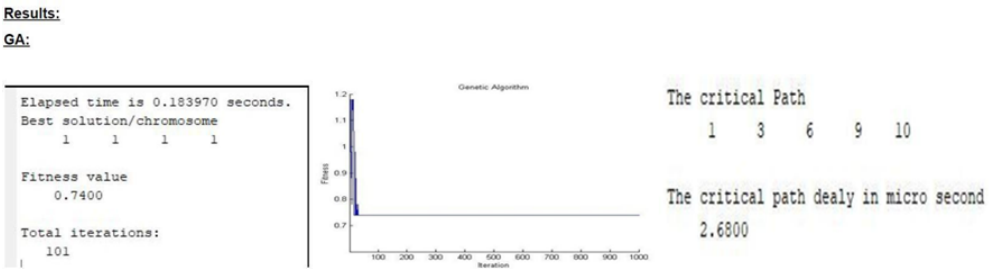


Figure 3.15 Graph for finding the fitness value in genetic algorithm

$t_{pLH})/2$ =0.67 μs, The results obtained by using genetic computation and the elapsed time are 0.183970 seconds and the best solution is for the inputs 1111 Fitness value is 0.7400. The total iterations are 101 and the respective graph is shown in Fig 3.15.

3.1.4 Critical Path Delay Calculations

The critical path is shown in Fig.3.16 and the path is highlighted in blue color (Circuit1). For example, 1-3-6-9 = 2.68 μs. The traverse of 1 to 3 is AND which is 1.34 μs and from 3to6 NAND gates it is 0.67 μs and from 6 to 9 is the path of NAND it is 0.67 μs. So, the total of this path is 2.68 μs. Now calculating the flow of input in every possible path the critical path flow is 1-3-6-9 and 1-3-8-9 which is 2.68 μs and that is the critical path value. Weighted graph for critical path is displayed in Fig.3.17.

3.1.5 PDF of Critical path

The mean is calculated by the critical path consideration for 1-3-6-9 so the mean values are added and the total mean is 2.68 μs. Standard deviation: consider 99.7% delay

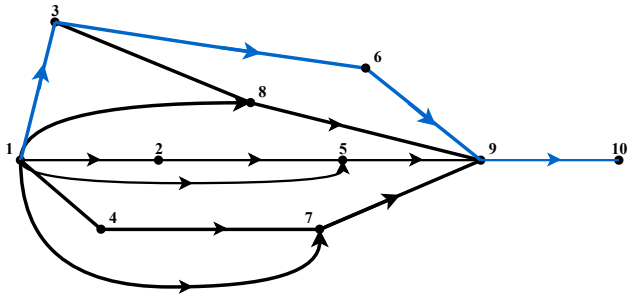


Figure 3.16 Critical path selected (circuit 1)

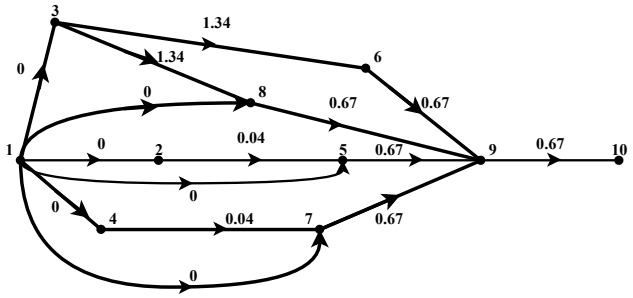


Figure 3.17 Weighted graph for critical path (circuit 1)

covers within the range of 0 to 2 μs. To cover 99.7% the standard deviation is $\sigma_t = \sigma_1 + \sigma_2 + \sigma_3 = 0.33$ μs. Total σ_t 0.33+0.33+0.33+0.33 = 1 μs.Thus, the above values which are calculated is plotted in the below graph shown in Fig 3.18. Similar to the above process, we have measured for test circuit 2 for different outputs and the corresponding results are displayed in the below sections, the critical path for test circuit 2. The critical paths for test circuit 2 for output is shown in the below Fig 3.19, Fig 3.20. The path is highlighted in blue color (1-3-4-9 and critical path delay is 2.01 sec). Weighted graph for the critical path

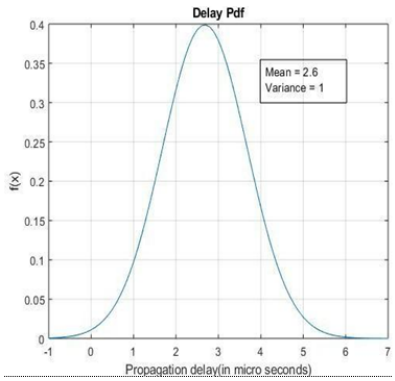


Figure 3.18 PDF Graph of time delay

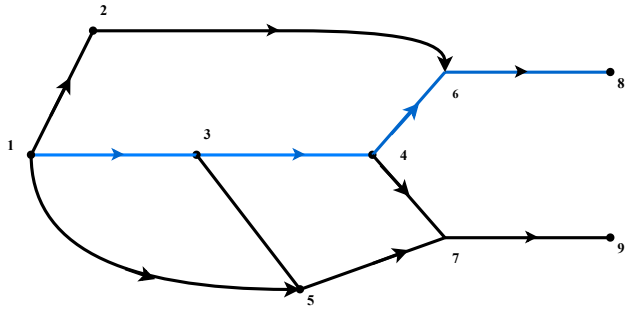


Figure 3.19 CRITICAL PATH1 Selected(circuit 2)

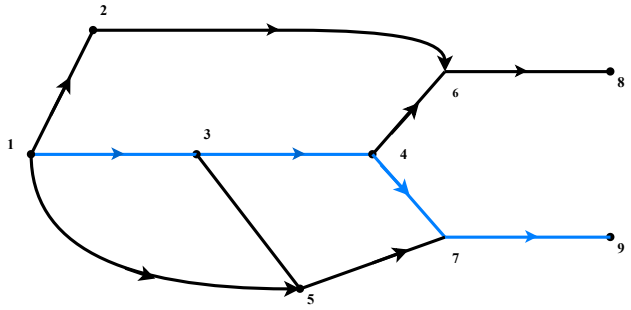


Figure 3.20 CRITICAL PATH2 Selected (circuit 2)

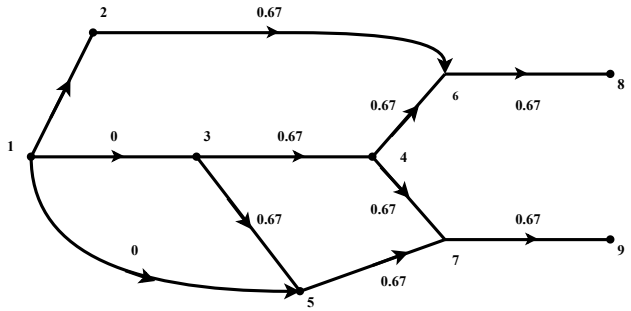


Figure 3.21 Weighted graph for circuit 2

Table 3.2 Results Comparison

S No	Algorithm	Run Time(s)	Avg. Speed Up
1	Non-IncrementalComputing (with MAX operator) [15]	0.0169-0.2502 s	5X
2	MonteCarlo Simulation (Ref basic Model) [16]	90.59-540.13 μs	X
3	SkewCanonical Model [17]	89.32550.02 μs	X
4	Non-IncrementalComputing (with Genetic Algorithm)	2.0to 5 μs	10X

is displayed in the below Fig 3.21. The table 3.2 describes the performance comparison with state-of-the-art algorithms.

3.2 PODEM (Path-Oriented Decision Making)

The generation of test vectors for circuits involving in error correction and translation may use appropriate Primary Inputs (PI). The tests are evaluated which excites the fault, that is, causes the complementary value to appear at the fault site. PODEM algorithm is used to generate test vectors for a given fault and also sensitize the faults that are changed in order to propagate the fault to a Primary Output (PO). Non-controlling values are assigned to nodes connected to the output gate in order to discover errors. Basic steps followed by PODEM algorithm are:

- Apply fault excitation conditions
- Implement the previous assignment effects
- Justify the remained unjustified lines whenever, one or more PO's are reached by the fault symptoms. In case that the justification is failed, back trace and go to step 2.
- Perform the resulting implications and go to step 2.

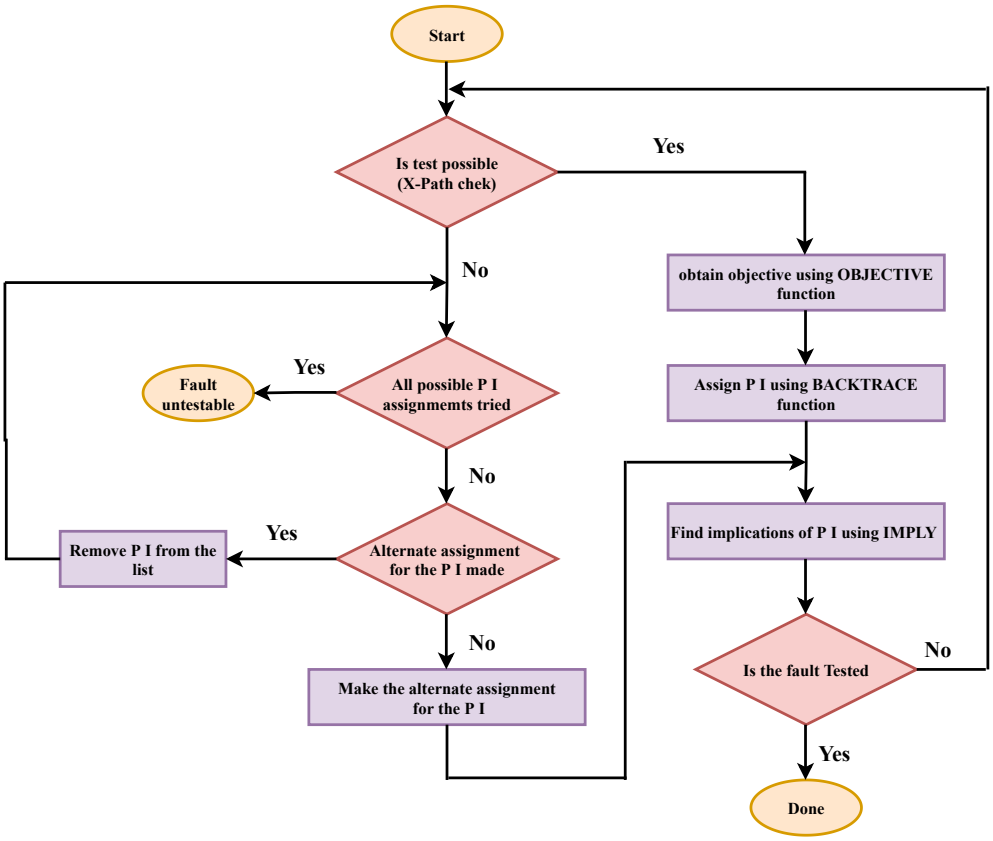


Figure 3.22 PODEM Algorithm Flowchart

- Once back tracing is completed, to propagate the fault, assign non-controlling values to nodes at output gate (by backtracking and assigning PIs appropriately)
- Always select a PI (specifically for PODEM Algorithm)

In back trace, a desired value intended for it and a path backwards is traced in the circuit until an input is found in PODEM. Moreover, it provides the most effective over the D-ALG since its search space is limited towards the circuit's PIs. However, a search space is present in a D-ALG which includes the entire nodes within the circuit as well as the PIs.

3.2.1 Functional Modules used in PODEM

The different functional modules are used in PODEM algorithm to generate test vectors for identification of faults. The global variable 'podem' used as input assignment for calling 'main podem' function and corresponding flow chat of PODEM algorithm is shown in Fig 3.22. It can create the net list and convert it to sensitize the fault.

Table 3.3 Device ID and Node of net list

Device ID	Nodes from Fig	Node of net list
1	E	n1
2	A	in1
3	G	n2
4	B	in2
5	C	in3
6	F	n3
7	D	in4
8	H	n4
9	J	n5
10	K	n6
11	L	n7
12	M	out1

Three major function's objective, back trace and imply are performed by 'main podem'. Moreover, given fault location, fault value pair are evaluated and updated at each node fault propagation. Whereas, non-Controlling Value (NCV) for each node is connected to output node. If any fault is detected and displayed along with the set of PIs. The functions Read net list and Convert net list are to store the input and output nodes in the format as Device ID, Gate Type, Output node and Input Nodes. Device ID is a unique number assigned to each gate using node map function. The Objective is a function feed by nG and nV where V is the required value on G and, the respective flow chats of objective is shown in Fig 3.23. It calls an inherited back trace function and its work flow shown in Fig 3.24. back traces this pair of (nG, nV) till a PI is achieved and asserted PI flag. After each back trace operation, value of nG, nV is updated to the next value in stack G. Every row, according to type of gate nG and nV, evaluates the values of inputs with respect to gate using "type" function. An imply function is used to perform logic simulation proceeding from the circuit depending upon the PI values. The inputs (PI and PI Value) are obtained from the objective function. It searches for the gates in netlist, where PI node is an input and stores them in device connected array. Then it calls "imply

device" function and evaluates the output node values. Moreover, it updates the node values array. For example, if 2-input NAND gate taken into consideration the two inputs and output of the row which is to be updated as 'g' = '0' for backtracking, and 'g' = '1' for implying. The netlist with respect to alphabets to facilitate the understanding of trace operation is shown in Table 3.3.

3.2.2 Implementation using PODEM

The design of NAND, NOT, and NOR gates are done in LTSPICE using FinFET models (22 nm) and then creating a netlist for circuit diagram shown in Fig 3.25 and later imported to MATLAB in which PODEM code is implemented and its corresponding flow chart is shown in Fig 3.26. The Fig 3.27 shows the delay characteristics of the FinFET NAND gate. The steady state values obtained are, High to low propagation delay (tpHL): Time is calculated which takes to fall from VOH to 50% Low to high propagation delay (tpLH): Time is calculated which takes to rise from 50% to VOL Propagation delay is calculated as (tp)= (tpHL+ tpLH)/2=0.67, similarly for NOR and NOT gates also parameters are characterized and Results are tabulated in Table 3.1.

3.2.3 Results and Summary

- Fault at node K i.e. stuck-at-1 = & gt; V = 0:

Step 1: Set objective (K, 0) and back trace (K, 0) which gives (G,1) and (D,1). D = 1 Found by back tracking (till a PI). Now call imply (D, 1) D=1implies F=0 implies L=1 while other nodes are still unassigned.

Step 2: Now check for value at fault location. It is still unassigned (value = -5) so, continue with using the second option of back trace (K, 0). i.e. (G, 1) as stored in stack earlier. Backtrace (G, 1) gives (B,1) and (C,1).Imply with (B, 1) gives G = -5, K = -5 Step 3: Now imply with C = 1 = & gt; G =1, K=0. So fault is sensitized.

Step 4: For propagation, back trace (J,1) = & gt; A=0, E =1, H=0

Step 5: Since H=0, fault cannot be propagated. So, no test exists And fault is not detectable to Fault at nodeK, stuck-at- 0 =& gt; V =1

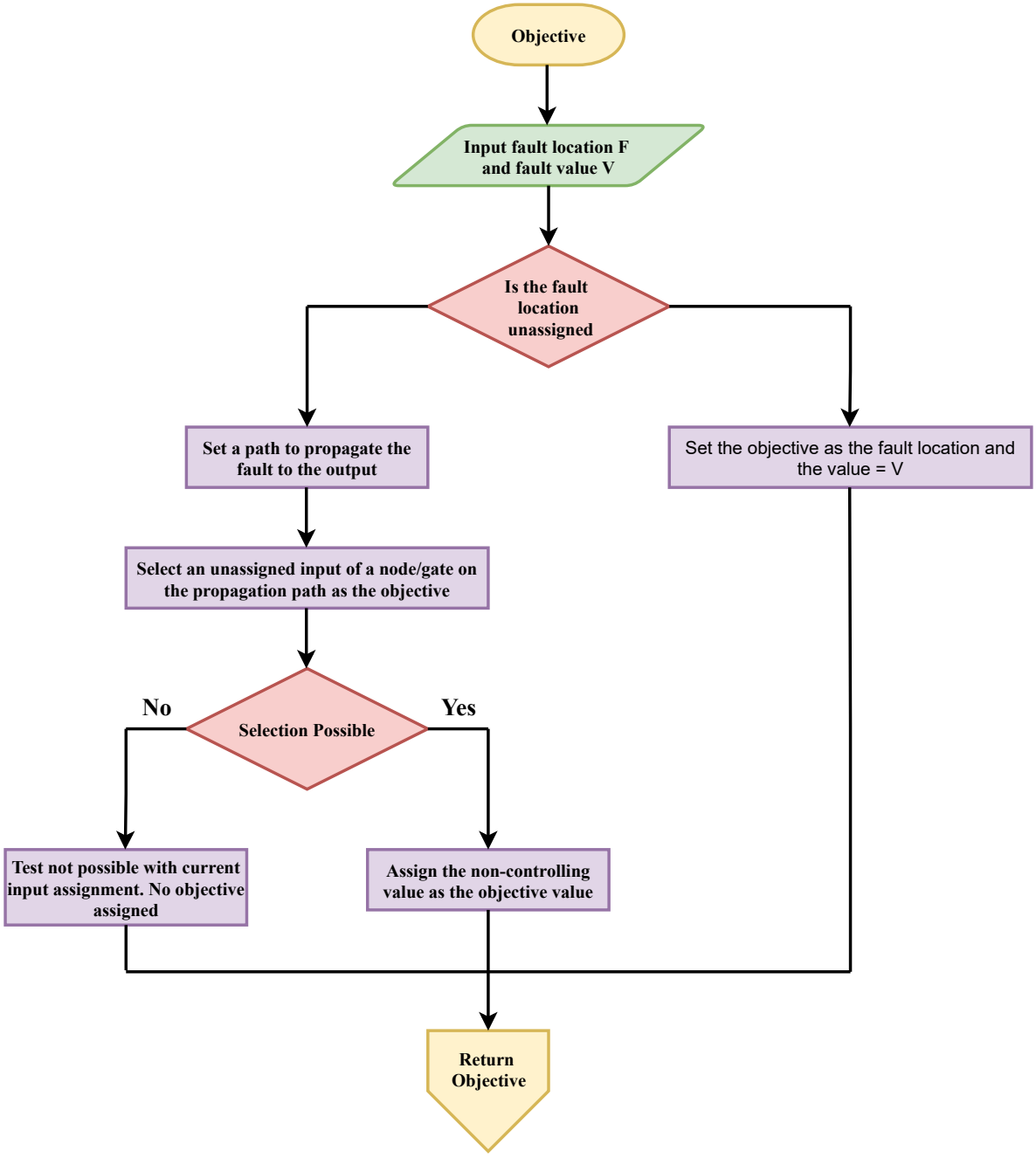


Figure 3.23 Work flow of objective

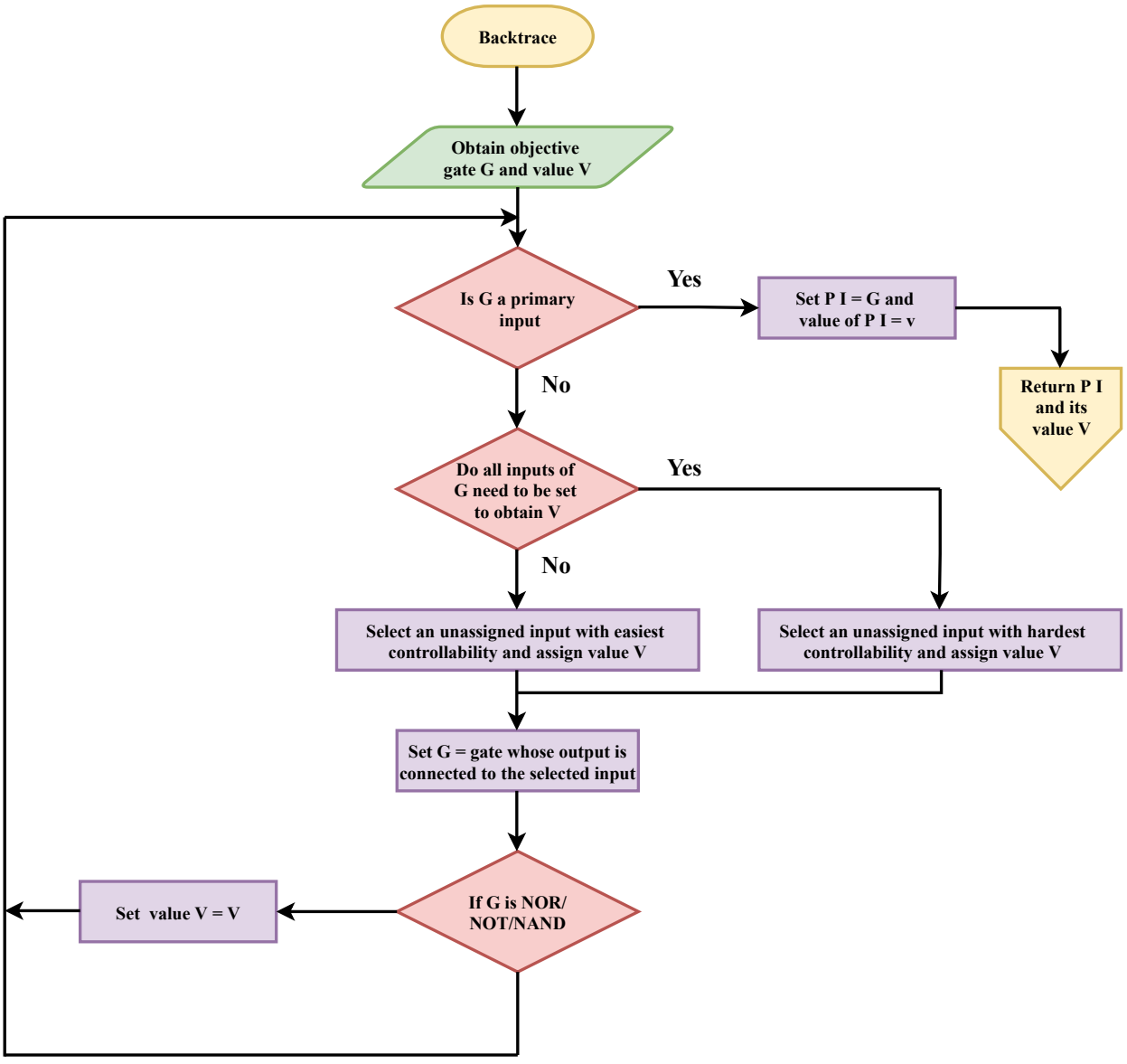


Figure 3.24 Work flow of back trace

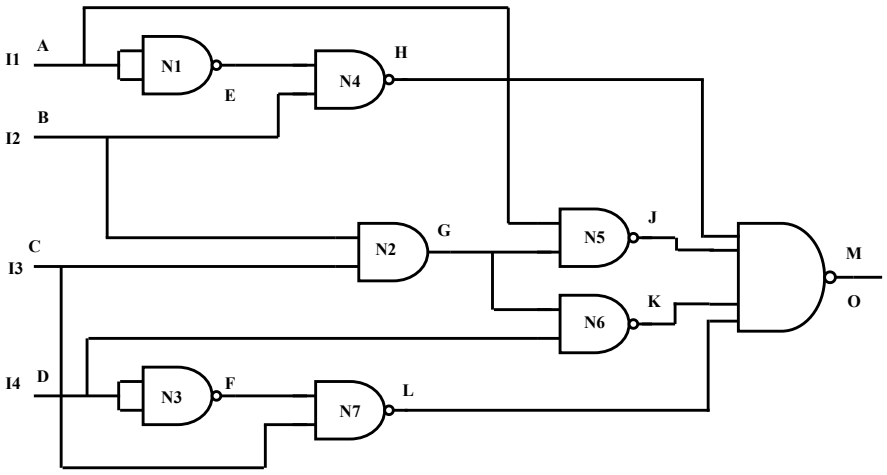


Figure 3.25 PODEM Algorithm Test Circuit

Step 6: Set objective (K, 1). Back trace (K, 1) = & gt; D=0.Imply (D,0) will give F=1, K=1. Fault at K is sensitized

Step 7: To propagate the fault, back trace (H,1) =& gt; (E, 0)= & gt; (A,1) , (B,X)Imply (A, 1) = & gt; no changes in node values.

Step 8: Back trace (J,1) =& gt; (G,0) = & gt; (C,0) imply (C,0) =& gt; (J,1),(L,1)

Step 9: Since J, L and H, all are set to non-controlling value of Nand4. Fault can be propagated. Input vector is (A, B, C, D) = (1, X, 0, 0) n Proposed test algorithm that provides better diagnosing of faults in FinFET circuits for the given test circuit.

Algorithm for the Test Circuit (Ref: Fig 3.22) the fault is detected at node n4 (Device Id H) and Fault Value is 0, similarly for any given Fin FET based Circuit faults can be detected and location can be identified with sensitive Test Vector at the inputs Ref: Fig 3.28).Table 3.4 describes the performance comparison with state-art-of various alogritms.

3.2.4 Conclusion

In this chapter, a brief overview of fundamental concepts of testing and fault diagnosys with non-incremental computation by incorporating optimization technique i.e., adaptive genetic algorithm is presented, which aids in reducing the probability of path

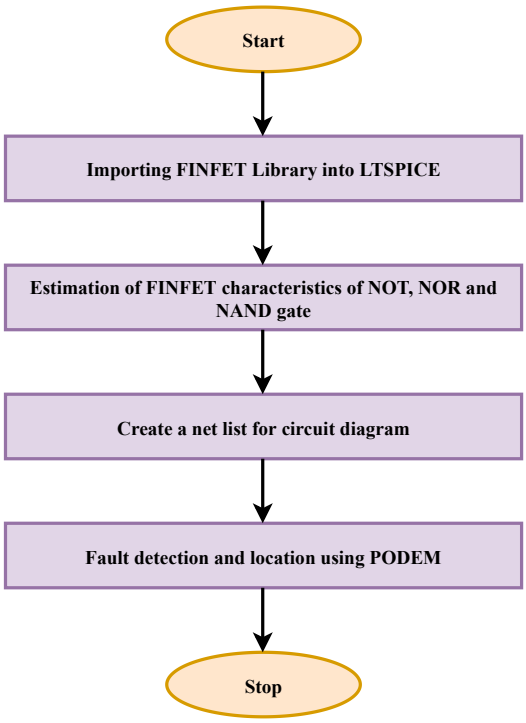


Figure 3.26 PODEM Algorithm

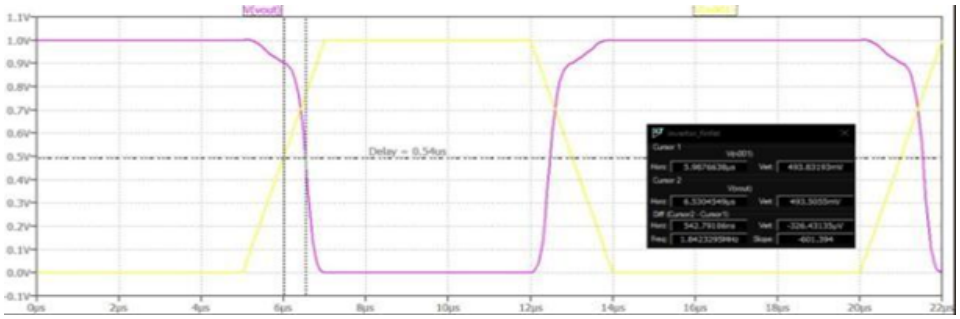


Figure 3.27 Propagation Delay (t_p) for NAND Gate

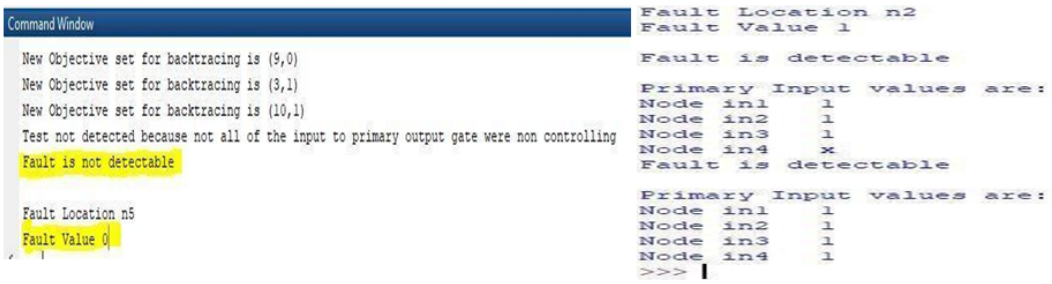


Figure 3.28 Fault value is Not detectable and detectable

Table 3.4 Results Comparison

SNO	Algorithm	Fault Detected	Fault Location (nodelevel)	Fault Value
1	PODEM X algorithm [18]	yes	no	Yes (stuck at 1/0)
2	Deductive Fault Simulator [19]	yes	no	Yes (stuck at 1/0)
3	PODEM Proposed	yes	yes	Yes (stuck at 1/0)

delayfault. The observations in the genetic algorithm and the non-incremental algorithm are useful to develop the generations of the FinFET circuits and the path propagation delays. Furthermore, best vector selection is found using PODEM algorithm to detect the fault location in advanced 22 nm FinFET based combinational circuits. The power verses delay time have been improved when the similar deployment of generations in the FinFET circuits, which are planned to optimise or converge genetic algorithm results.

Chapter 4

Testing and fault diagnosis of sequential logic circuits

This chapter presents Vedic March algorithm (VMA) to test RAM in a 32-bit RISC processor efficiently. Whereas, the Reduced Ordered Binary Decision Diagram (ROBDD) technique to design and test Multiple Stuck At Faults (MSAF) of a sequential circuit by applying few test vectors for corner cases of design.

4.1 32-bit RISC architecture

A general-purpose 32-bit RISC processor is shown in Fig 4.1 consists of 5-stage pipelining implemented by fetch, decode, execute, memory, and write back stages [25] for data read or write operations. This architecture is designed as separate data and instruction interface i.e., Harvard architecture which reduces hardware complexity. The address generation is done by RAM when connected to the BIST controller, whereas it runs with test vectors with valid reads and writes

4.1.1 Implementation of VMA on 32-bit RISC processor

VMA is implemented on RISC processor that follows concurrent testing sequences while entire memory can be grouped which are further subdivided into sub-groups. In each of the sub-group, primary faults are detected from all of the test vectors and forwarded to the groups. If more faults are detected, the group is selected individually and the last test vectors of the sub-groups are applied again till the diagnosis is finished.

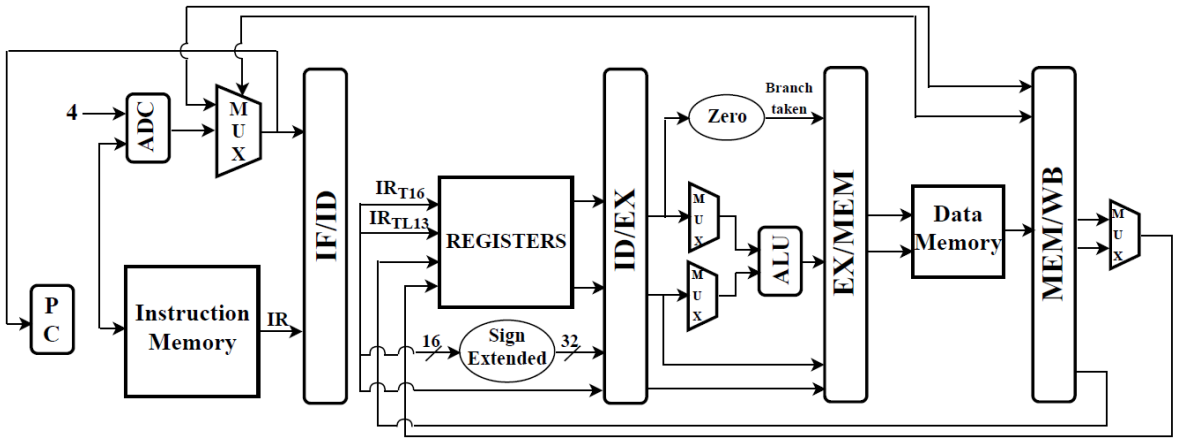


Figure 4.1 proposed method of 32 bit RISC Architecture

G1:{↑(W0); ↑(R0,W1); ↑(R1);↓(W0); ↓(R0,W1); ↓(R1)};
G2:{↑(W1); ↑(R1,W0); ↑(R0);↓(W1); ↓(R1,W0); ↓(R0)};
G11:{↑(W1); ↑(W0,R0); ↑(W0,R1,W1); ↑(R0,W1); (R1);↓(W1); ↓(W0,R0): (W0,R1,W1); ↓(R0,W1); ↓(R1)};
G22:{↑(W0); ↑(W1,R1); ↑(W1,R0,W0); ↑(R1,W0); (R0);↓(W1); ↓(W1,R1): (W1,R0,W0); ↓(R1,W0); ↓(R0)};

Figure 4.2 Indication of existing and proposed march algorithm elements

From VMA the data are divided into two sub-groups (G_1 and G_2) and used to work as concurrent flow. The sequence of march elements followed in VMA are shown in Fig 4.2. where G_1 and G_2 are the march algorithm elements, G_{11} and G_{22} are VMA elements. The number of march elements is increased to that of traditional modified March-C as '6', when concurrent flow performs, the complexity is decreased to 18 ns. Fig 4.3 shown proposed design of processor with Vedic march algorithm block.

4.1.2 Pseudo code for VMA

The pseudo code for the VMA is given below. While writing ones (1's) and zeros (0's) in block 2 and 1, it is assumed that R_1, C_1 are rows and columns of the memory design.

pseudo code: Vedic March algorithm (VMA)

1: **for**$(y = 0; y < (R-1)/2; y = y+1)$
2: **begin**
3: **for**$(z = 0; z < (C_1 - 1); z = z+1)$
4: **mem**$[y][z] = 0;$ //**write** 0 **in** $m1$
5: **end**
6: **for**$(y = (R_1 - 1)/2; y < (R_1 - 1); y = y+1)$
7: **begin**
8: **for**$(z = 0; z < (C_1 - 1); z = z+1)$
9: **mem**$[y][z] = 1;$ //**write** 1 **in** $m2$
10: **end**
11: **for**$(y = 0; y < (R_1 - 1)/2; y = y+1)$ //**writing and reading in background**
12: **begin**
13: **for**$(z = 0; z < (C_1 - 1); z = z+1)$
14: **begin**
15: **if**$(mem[y][z] == 0)$
16: **mem**$[y][z] = 1$**;**
17: **else return;**
18: **end**
19: **end**
20: **for**$(y = (R_1 - 1)/2; y < (R_1 - 1); y = y+1)$//**writing and reading in background**
21: **begin**
22: **for**$(z = 0; z < (C_1 - 1); z = z+1)$
23: **begin**
24: **if**$(mem[y][z] == 0)$
25: **mem**$[y][z] = 0;$
26: **else return;**
27: **end**
28: **end }**

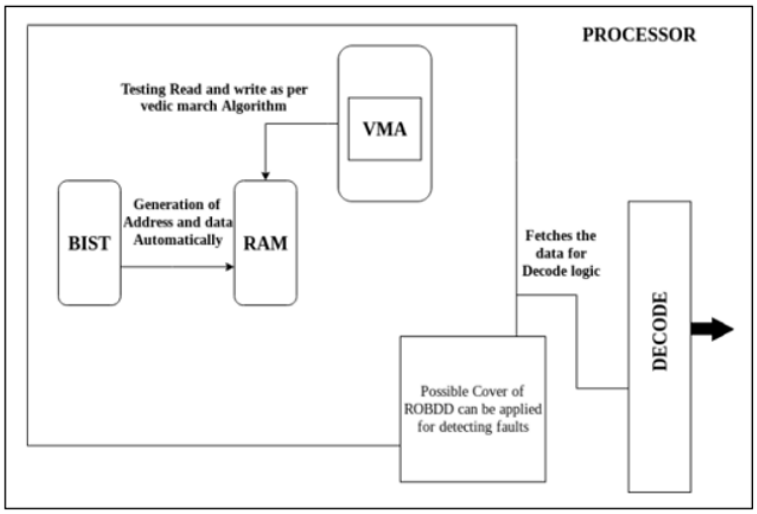


Figure 4.3 Processor with Vedic march algorithm block

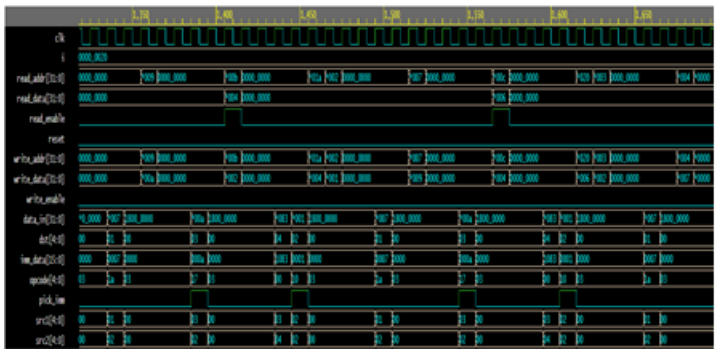


Figure 4.4 32-bit RISC processor with I addressing format

4.1.3 Results Analysis

A 32-bit RISC processor with different types of addressing formats I, J, and R is implemented shown in Fig 4.5, Fig 4.6, and Fig 4.7 respectively. They denote various datatype instruction formats in which data is loaded based on the opcode provided by machine level instruction. The data is fetched and load to memory through instructions and it appears for testing at reading or writing for load-store operations.

VMA is applied for the testing of RAM while performing read or write operations shown in Fig 4.8. The combinations of all write and read operations are tested by G_1 and G_2 blocks effectively but it is unable to test the corner cases like double reads and double writes (previous faults found from the same location for a number of times while going for the next read or write) operations. Generally, RAM cannot preserve the previous

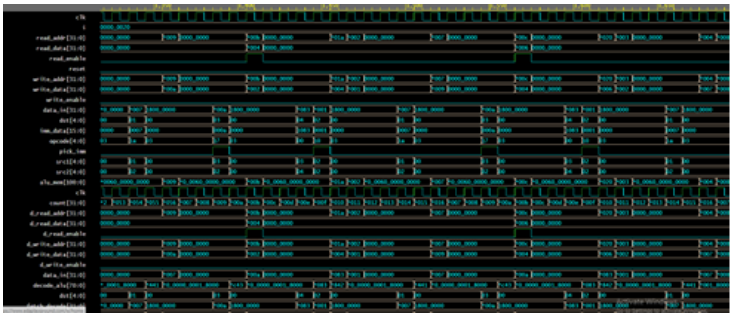


Figure 4.5 32-bit RISC processor with J addressing format

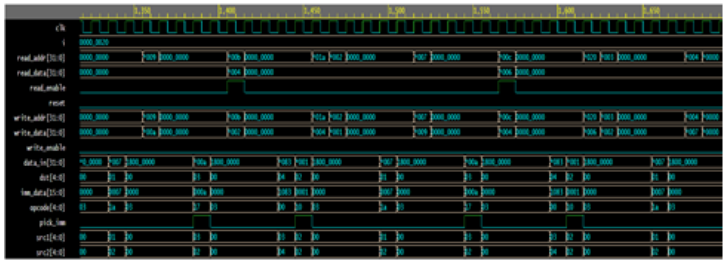


Figure 4.6 32-bit RISC processor with R addressing format

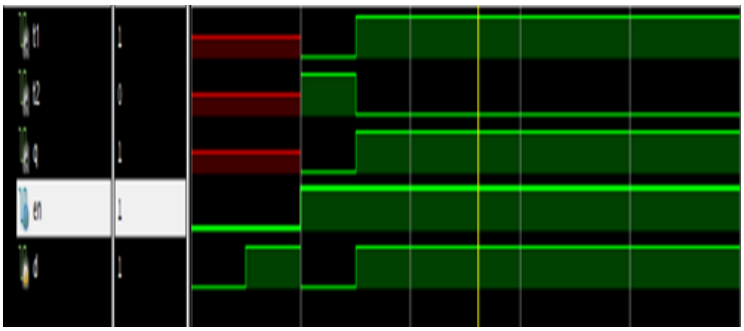


Figure 4.7 Read and write opeartion of RAM

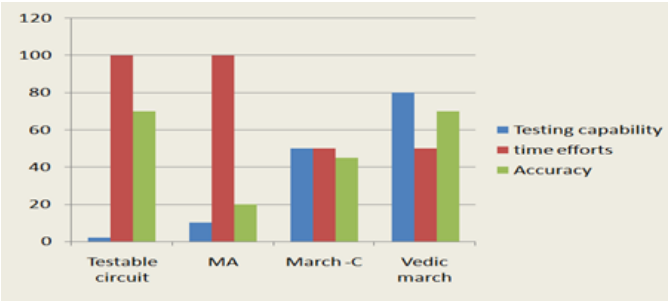


Figure 4.8 Performance chart for Modern approach testing compared with existing technologies

Table 4.1 Results Comparison

Methods	Gate count	Delay (ns)	Faults detection
Testable circuit [73]	50640	0.691 0.420	60%
March [83]	44628	0.596 0.420	65%
March-C [83]	44002	0.595 0.420	70%
Vedic March	33396	0.591 0.420	80%

data and gives undefined value when performing the next read operation. ROBBD can visualize the faults when tested across multiple corners by G_{11} and G_{22} operational blocks with reference to the Fig 4.8 for multiple read or write operations. It can detect the fault in the memory block whenever any bit or memory reached an undefined value. For example, the expected value for toggle coverage faults is shown as '000000' instead of '101010' in the process of memory detection. Nevertheless, at the end of the multiplexer block the process is repeated till it detects the faults.

The proposed technique is efficient in terms of testing capability, time efforts, and accuracy as compared to the existing approaches is shown in Fig.4.8. Hence, it can be used for high-end processor designs. The performance comparison concerning area, speed, and fault detection of VMA is given in Table 4.1. When compared to the other methods like March and March-C [83] Vedic march is faster in detecting the faults and also achieved better efficiency. Usually in testable circuits [25] more hardware and hence increased latency will result in performing maximum checks towards covering faults against multiple corners.

Even though the efficiency is good for finding stuck-at faults in single input blocks

like flip-flops but it is fails to find the faults in high-end processors. Whereas, Vedic march testing can overcome these difficulties for finding faults in the system with greater performance along with a 50% reduction in the effective area when compared to the testing methods provided in Table 4.1.

4.2 ROBBD implementation on sequential circuits

The testing and detection of faults for a 32-bit RISC processor are implemented with ROBDD sequential circuit that uses test vectors applicable for finding faults in sequential designs. A test vector is applied with sample fault i.e., a wrong address is applied and data were readout as 'y', shown in Fig 4.1. For example, a D Flipflop under test consideration will be reduced to a Binary Decision Diagram (BDD) and a multiplexer is applied at the end of each sub-block. When VMA is used on a ROBDD block that is attempting to read data from write-only places or write data to write-only locations, it generates an error signal when the fault is detected, rather than resetting and repeating the detection for multiple design failures. A ROBDD block [25] was instantiated to RAM for fault detection with reduced complexity. Also, it is used to test Multiple Stuck At Faults (MSAF). Sequential circuits' [84] implementation with ROBBD for finding faults is not only limited to high-end processors but also extends to the development of the Combinational Logic Block (CLB) of FPGAs. When single stuck-at issues are considered, the CLBs are incapable to follow the test structure due to exceptional defects that may arise with different test length. Single Event Upset (SEU) and framework imperfections

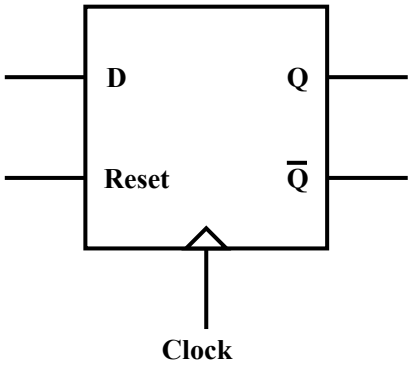


Figure 4.9 Symbol of D-flip flop

Table 4.2 Excitation table for D-flip flop

Input			**Output**
D	**Reset**	**Clock**	**Q**
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

may show different weaknesses at the CLBs. Moreover, the length of various shortages in tests is about twice the length of single stuck-at failures tests at CLB's. Consider a simple sequential circuit DFF that is shown Fig 4.9 with excitation table is presented in Table 4.2. The DFF responses in equation form is converted into BDD and the same is converted to ROBDD is shown in Fig 4.10 and Fig 4.11.

The normal mode operation of the master-slave edge-triggered flip-flop is shown in

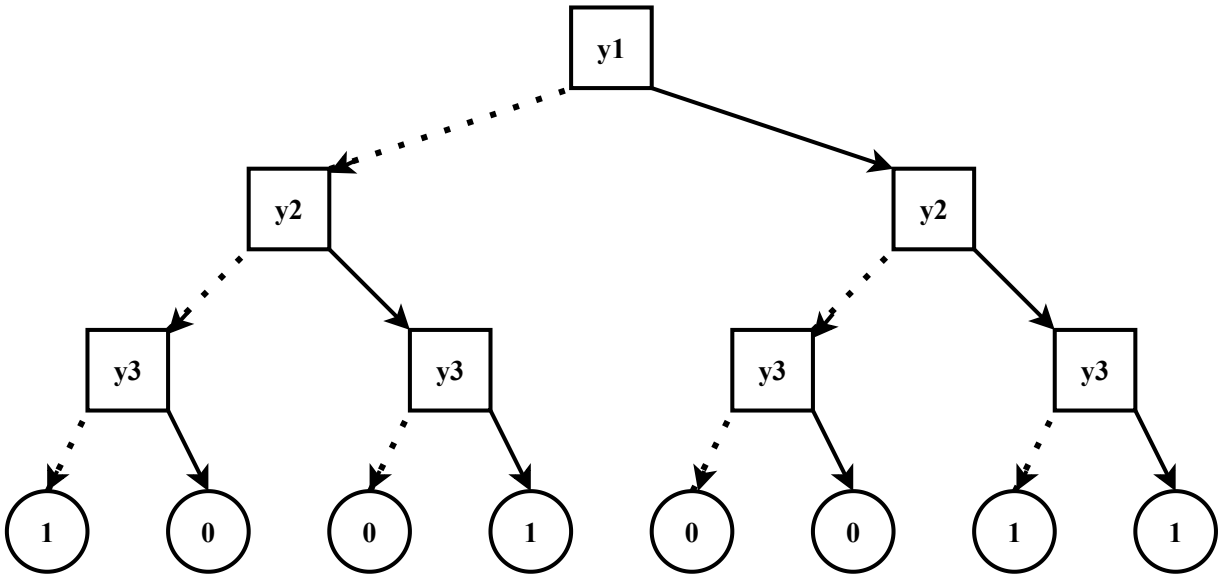


Figure 4.10 Simplification to binary decision diagram

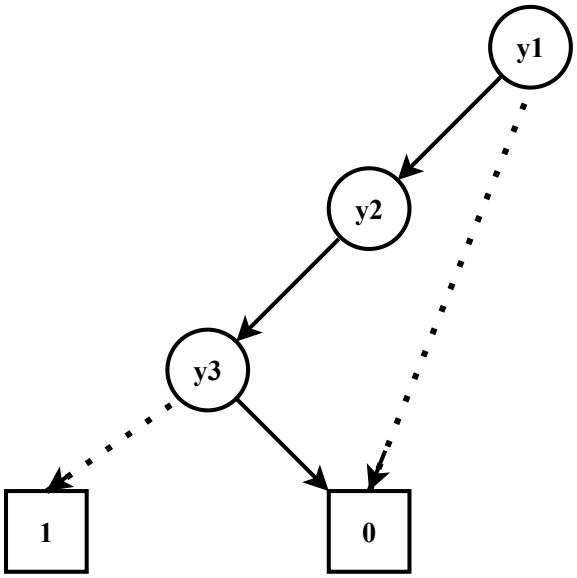


Figure 4.11 Simplification to reduced ordered binary decision diagram

Fig.4.12. However, the required control signals and settings are given below to calculate the faults in the designs.

- Normal Mode: pC1=0, pC2=1, nC1=0 and nC2=1

- pC1=0, pC2=1→ Copies the output of the positive enable D latch to avoid fan out issue

- nC1=0, nC2=1→ copies the output of the negative enable D latch to avoid fan out issue

Test Mode: The test mode 'operation 1' is shown in Fig.4.13 represents the operation based on below configurations.

All 1's are test vectors:

pC1=1,pC2=1,nC1=1 and nC2=1.

pC1=1 and pC2=1→ Breaks the feedback of the positive enable D latch

nC1=1 and nC2=1→ Breaks the feedback of the negative enable D latch

- Test Mode: The test mode 'operation 2' is shown in Fig.4.14 represents the operation based on below configurations.

All 0's are test vectors:

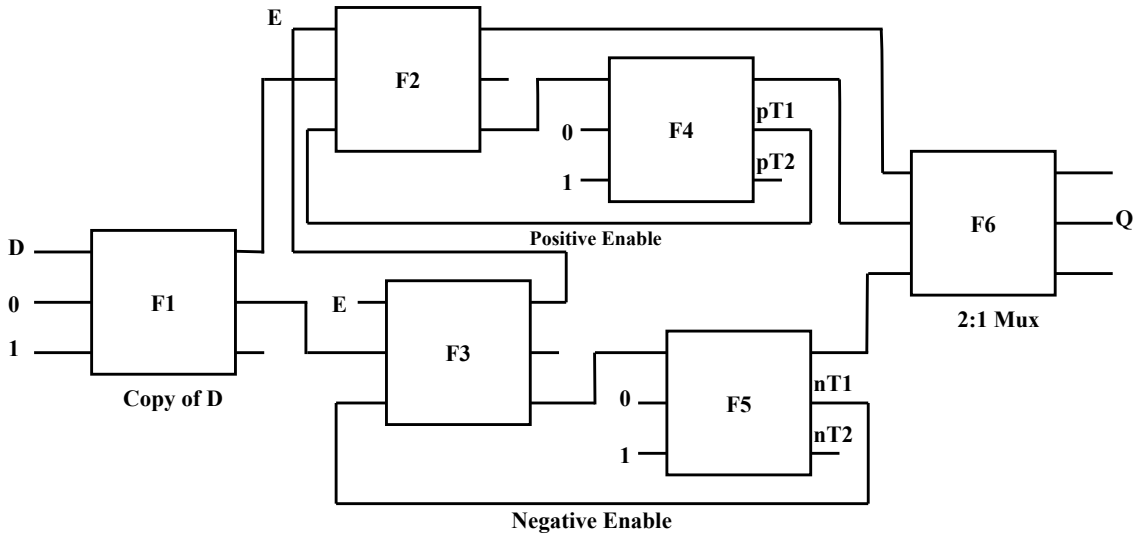


Figure 4.12 Normal mode operation of the master-slave edge-triggered flip-flop

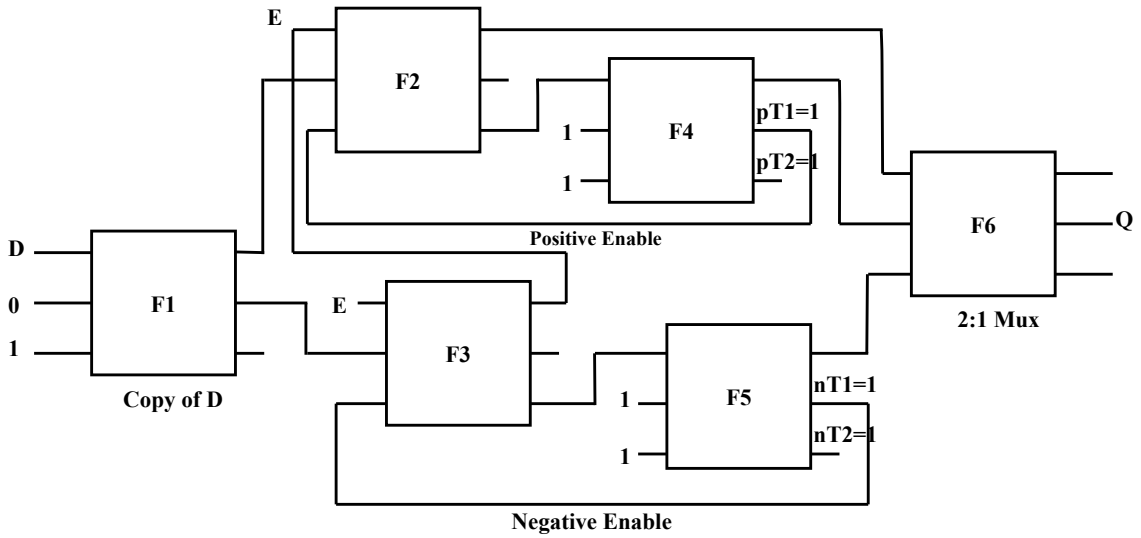


Figure 4.13 Test mode operation 1 of the master-slave edge-triggered flip-flop

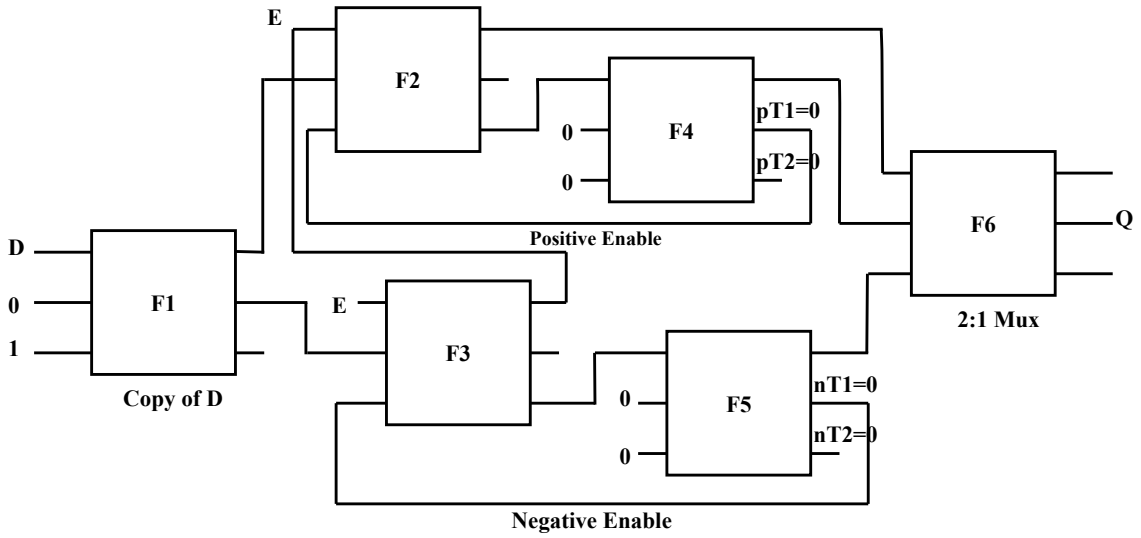


Figure 4.14 Test mode operation 2 of the master-slave edge-triggered flip-flop

Table 4.3 Final results taken from the test mode operation 1 and 2

D	E0	E1	pT1	pT2	nT1	nT2	Q
0	0	1	0	0	0	0	0
1	0	0	0	0	0	0	1
0	1	1	1	1	1	1	0
1	1	0	1	1	1	1	1

pC1=0,pC2=0,nC1=0 and nC2=0

pC1=0 and pC2=0→ Breaks the feedback of the positive enable D latch

nC1=0 and nC2=0→ Breaks the feedback of the negative enable D latch

$f1 = AB + AC, f2 = AB + AC, WhenA = 0, B = 1, f1 = 1, f2 = 1$

The control signals 1's and 0's is used to detect stuck at faults and final results based on test configurations are shown in Table 4.3. As the technology scales down, the geometry of the device shrinks. Moreover, various stuck-at faults are being introduced off late which are not considered in toto at the testing level. So, the consideration of defragmenting in the expansion of single stuck-at testing methods is to be improved to detect various stuck-at fault issues. However, the testing circuits are investigated with ROBBD included by sub-circuits i.e., DFFs. For a single test set, each sub-circuit design consists of '3N' test

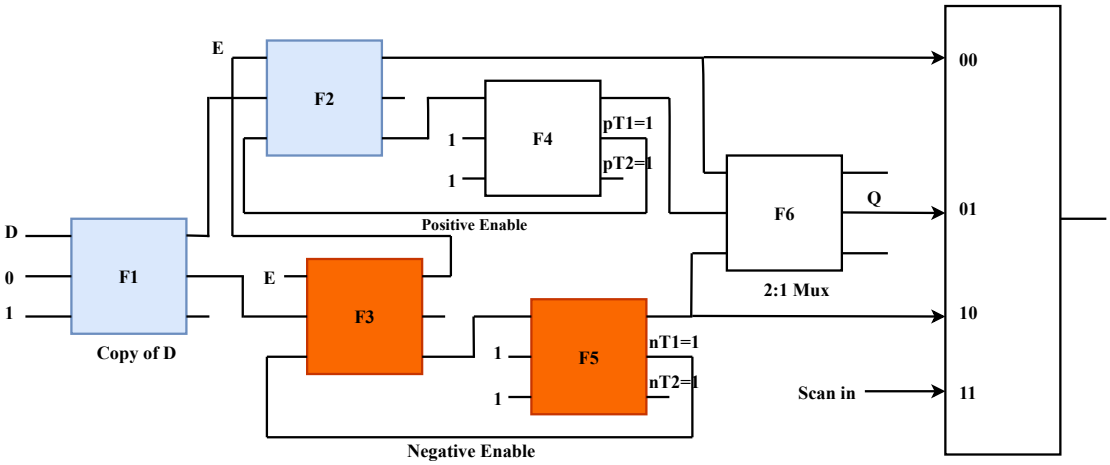


Figure 4.15 Identified test patterns in the design

vectors as upper bound and is maintained as a fragment. Where 'N' represents the exact center point for addressing the ROBBD design circuit. By partitioning, the toggling and bridge faults are also found using multiplexers and it consists of eight test vectors at the resultant polynomial-time limit. Consider '4N' as the length of the test, N represents internal nodes of the top design of ROBDD which describes circuit behavior. The usage of multiple multiplexers in sub-circuits limits the speed and turns out to be expensive. The implementation of ROBBD with a two-fold tree structure can minimize the number of multiplexers, which are preferred as a trademark for testability in the sub-circuit design as shown in Fig.4.15. Usually, the driving of basic inputs to multiplexers should provide the faults in the respective circuits whereas ROBBD can use them in an effective manner to improve the faults. Moreover, ROBBD can detect and improve the insufficient inputs to weak designs and adapt multiplexer test vectors accordingly.

4.2.1 Stuck at fault (SAF)

A particular fault weakness in design can be tested with the frameworks along with modified test configuration age gadgets to copy a falsification inside a consolidated design. So, an individual banner and sticks are believed to be stuck at logic values '0', '1', and 'x'. For example, if data is in a steady-state, the test ensures that it gathers distortion with a particular test plan. Likewise, the data could be connected directly to intelligible '0' to an affected design that cannot produce a stick because of issues observed in the accused

design. In testing of the main design, the hazards are said to be static specifically for the growing sign may lead the design untestable.

4.2.2 Line at the Single stuck

A line at the single stuck is a model that presents in automated design and most of the designs are not tested by an arrangement but follow the procedure of 'after the testing'. The modern design procedure acknowledges the single line or center point of the stuck at the design that indicates stuck happen at a line..

4.2.3 Design for detection of the faults using different test vectors

Gateway level design blocks or back-to-back design can be disconnected with the limit segments while detecting issues in the design [85]. Preferably, an entry-level design might be attempted by adding all possible information that provide the right yields. Anyway, to incorporate two 32-bit valued numbers need 264 = (0.8+1)*1019 tests and taking pretty long time. In weak modeled design, data are believed to be stuck and a test path is made to exhibit fractured design. The test path of the vectored bits is added to the design's sources of information, and a group of bits is predictable at the yield of the design. Every stick is related to the test vectors even though the circumstances are used to detect issues that the test vectors are applied for the pins which are grounded. These insufficiencies are known as a lone stuck at faults (1, 0) with respect to comparative testing modeled design and start sensibly to recognize majority CMOS privations. The designs which are failed to recognize cross weaknesses among neighbor sign lines are considered and associated with respective pins.Incidentally, single stuck-at fault imperfections are extensively preferred with few more tests empowered to a number of intricate circuits. Design for detection of the faults using different test vectors as inputs is shown in Fig.4.16. The control signals for test cases and faults as mentioned in Fig.4.13 and Fig.4.14 using ROBBD for Mux simplification indicates '1' if any fault has occurred else it is '0'. A blue rectangle box in Fig.4.16 indicates fault as an example of the design for testvectors applied as per the design requirements.

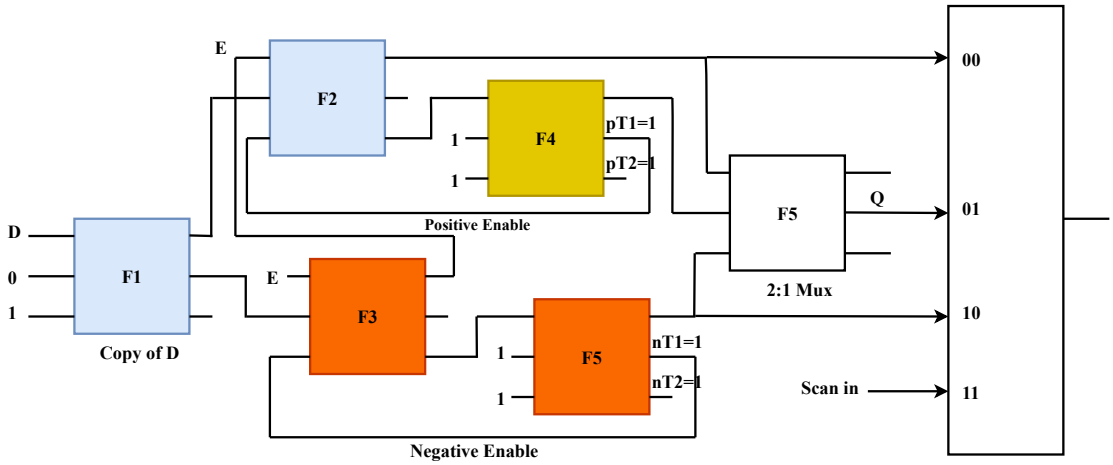


Figure 4.16 Identified test patterns in the Design for detection of the faults using different test vectors

4.2.4 Results and discussion

The normal operation of DFF without any control signal is shown in Fig.4.17 and the occurrence of stuck at faults is observed in Fig.4.18.The loss of connection between the bridges is observed in Fig.4.19 is not the outcome which was expected.The expected toggle fault with control signals from the circuit are represented in Fig.4.20.The circuit is implemented in Spartan-6 FPGA with effective area utilization is shown in Fig.4.21, which is very less compared to the existing designs under tests. The CLB part of FPGA is also shown in Fig.4.22.The implementation of ROBBD based synthesized circuit to be considered for testing single stuck-at faults with delay fault detection at an initial stage is more reliable and apt before fabrication when compared with different techniques given in Table 4.4.

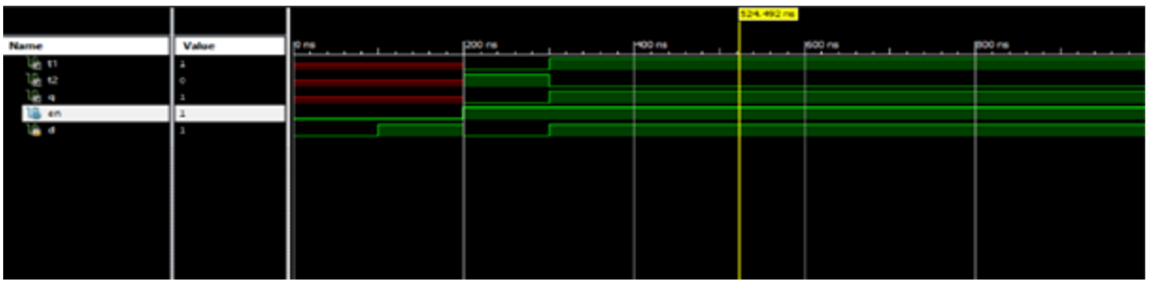


Figure 4.17 Normal operation of DFF

Figure 4.18 Stuck at '1' operation of DFF

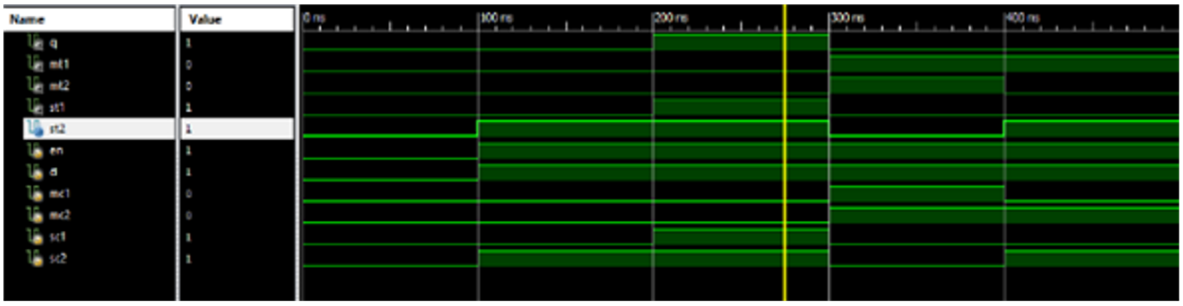


Figure 4.19 Bridge fault (loss of connection)

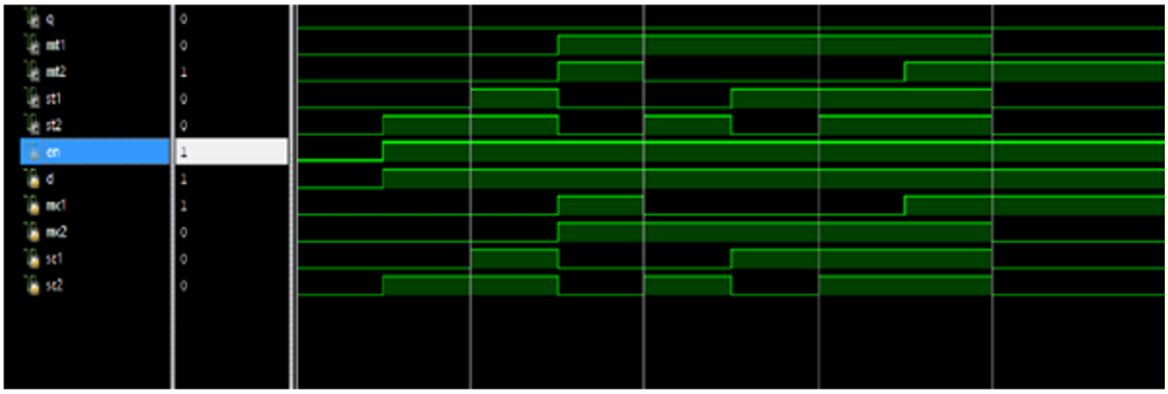


Figure 4.20 Toggle fault waveform

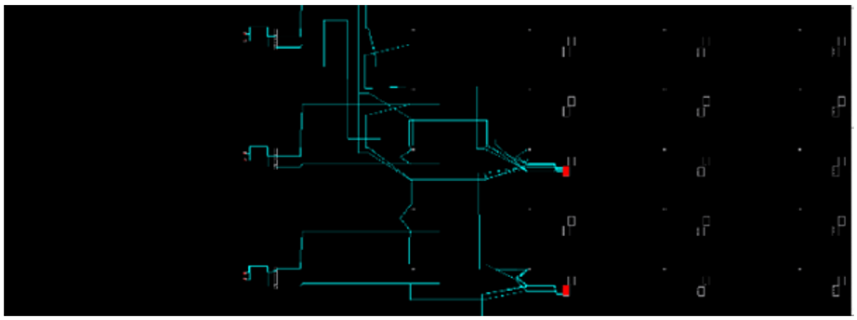


Figure 4.21 Area covered using configurable logic blocks

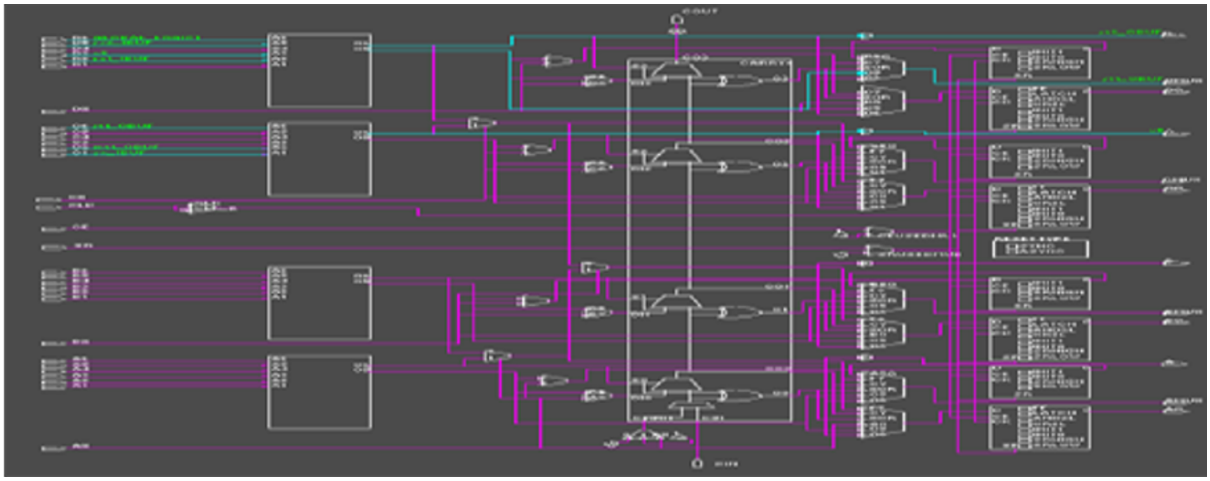


Figure 4.22 Routed design in FPGA

Table 4.4 Results Comparison

Method	Gate count	Delay(ns)	Faults	Approx.Efficiency
DET [86]	10	1.508	1	80%
MS Mode [86]	14	0.923	2	80%
Stuck at 0 [86]	6	0.832	1	90%
Stuck at 1 [86]	8	0.930	1	90%
Normal mode [86]	17	1.702	1	95%
ROBDD	15	0.859	>2	95%

4.3 Systolic Array Multiplier (SAM)

Systolic Array Multiplier (SAM) [69] is integrated with COG (Controlled Operation Gate), MIG (Modified Islam Gate) and reversible full adder to get the partial product in multiplication which is shown in Fig 4.23. An employment of toffoli gate in SAM results in optimum power consumption of the circuit. The multiplier gets bits from toffoli gates as multiplicand and generate partial product of sum and carry by following the pipelining process. For example, a 4-bit multiplication process required 16 multiplier cells to get the full product of the SAM. The multiplier result is in little endian format and the carrybit is forwarded to the next bit operation.

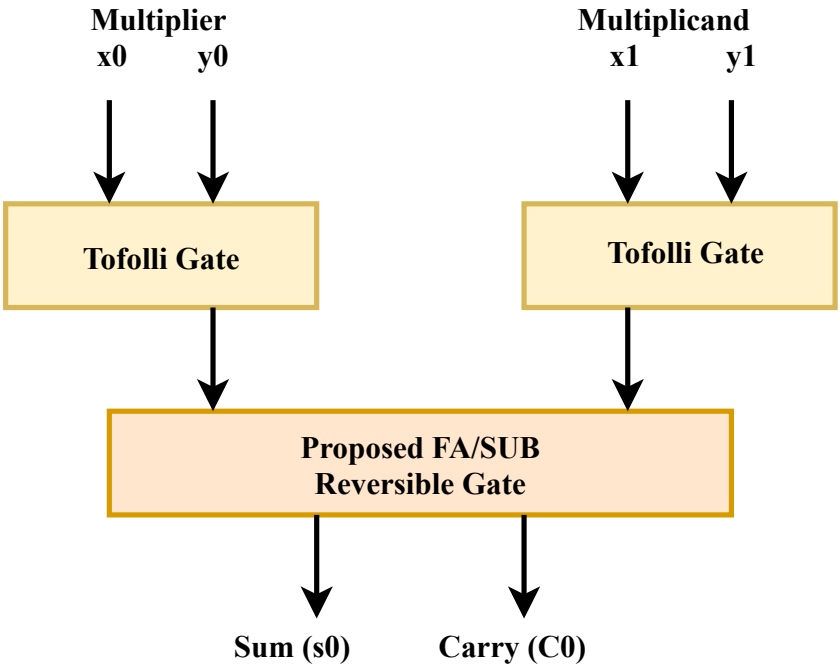


Figure 4.23 Multiplier cell block

4.3.1 Design and Testing of SAM with BILBO

The testing and fault detection for the SAM consists of Device Under Test (DUT), Golden Reference Model (GRM), Built-In Logic Block Observer (BILBO) and a checker. The faults are verified at various corner cases as shown in Fig 4.24. The proposed SAM design uses reversible gates in multiplier block to increase the speed of the operation. The multiplication process is performed in DUT block that consists of 4 stages, whereas the carryis forwarded from stage-by-stage and the end result is available at the 4 th stage. GRM is generally used for testing and verification of SOC designs and compatible for any programming language which behaves like DUT. When the faults are injected into the reference design, BILBO can generate the patterns and compare with the signature values to get the exact faults.

SAM is tested with an input signal fed in SS (Serial Scan) mode along with 4-bit multiplier and an 8-bit BILBO. If BILBO works in LFSR mode, it generates the number of required patterns to take the inputs by the multiplier. The Multi-Input Signature Register (MISR) mode perform the operations to generate signatures either good for no injection faults or bad if any faults identified. The process begins with BILBO for faults

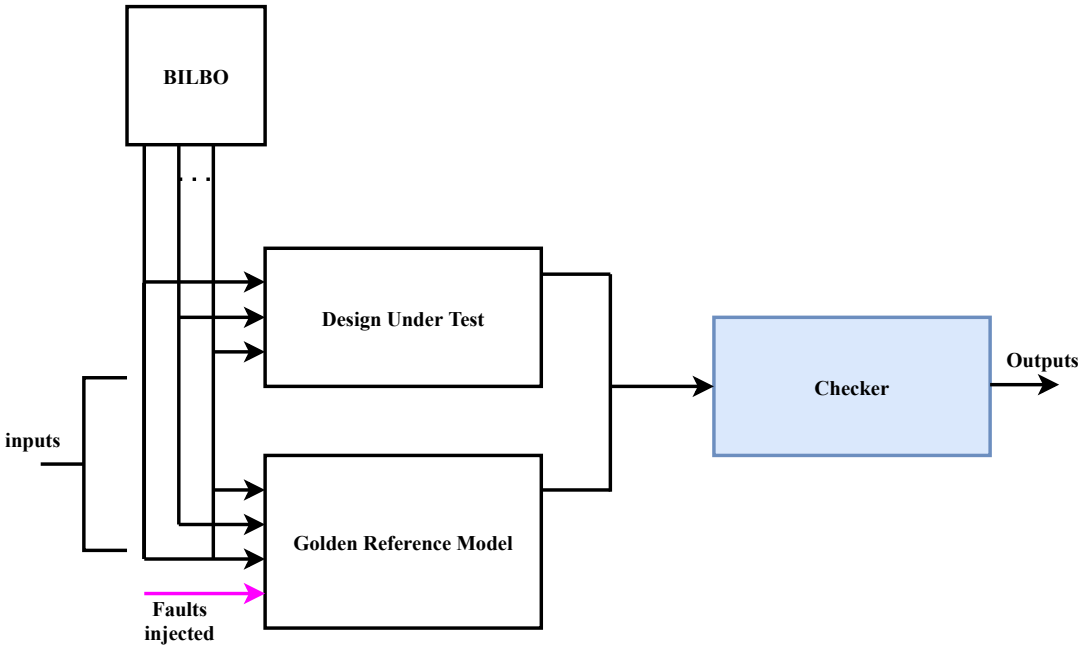


Figure 4.24 Proposed system with DUT and all required components

injected in the design while the patterns and signatures are generated by LFSR and MISR respectively the comparison takes place with existing signatures. If any match found, there is no fault identified otherwise there is a fault detected in the circuit. Checkers are named as score board logics and commonly used in verification that check the heterogeneous data received from any two blocks under DUT. SAM is tested by checkers for the detection of injecting faults and perform the comparison between GRM and DUT along with the storage of respective outputs for future usage.

With the advancement of pipelining in SAM the simultaneous processes take place in reference model even though the BILBO generating patterns with respect to fault injection scheme environment. A stuck-at-fault either '0' or '1' is injected at GRM the result may be improper whereas BILBO gives signature value as false for the same. Consecutively, the design will be modified and corrected till the BILBO pass a good signature. Like that, the process is continued to find various fault injections and compare the result with the checker.BILBO plays an important role in SAM to detect the injection faults in full environment is shown in Fig 3.2.6. The major faults of the design SAF, MSAF and

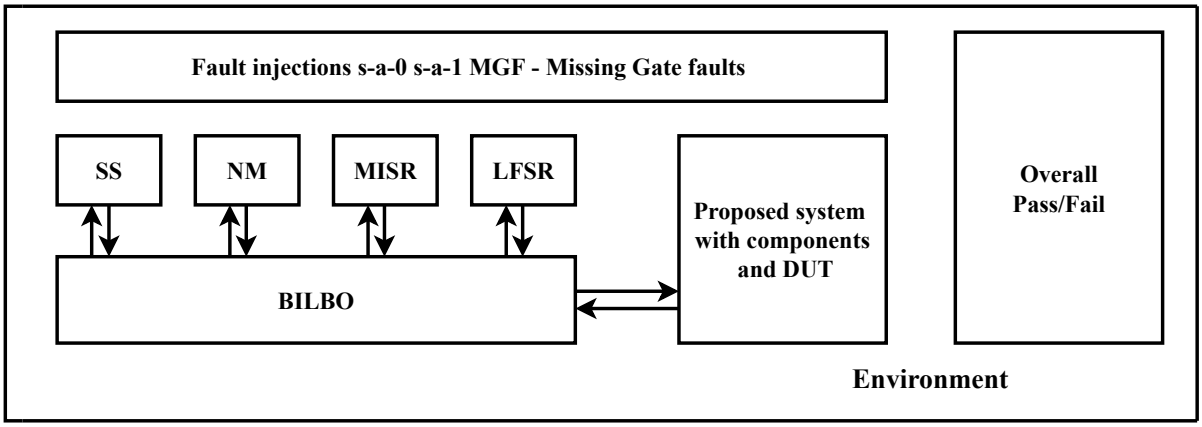


Figure 4.25 Full environment and testing with proposed systolic array multiplier using fault injection schemes

Missing Gate Fault(MGF) are found and tested by reversible BILBO efficiently in SAM.

4.3.2 Results and Discussion

SAM implemented with new modified Islam gate and its simulation results are shown in Fig 4.26 and the respective output calculated as 1111*1111=011100001.Various patterns have generated for the SAM are shown in Fig 4.27 and evaluated the result mentioned in decimal format as 14*15=210.Internal blocks of the SAM and output of the COG and MIG gates are shown in the Fig 4.28.The concept of injection logic in the design by MGF is shown in Fig 4.29 and also observe that the output does not break because of the usage of reversible logic gates. The random patterns generated from BILBO operated in LFSR

Figure 4.26 Results of reversible systolic array multiplier using pattern generator from Bilbo logic design

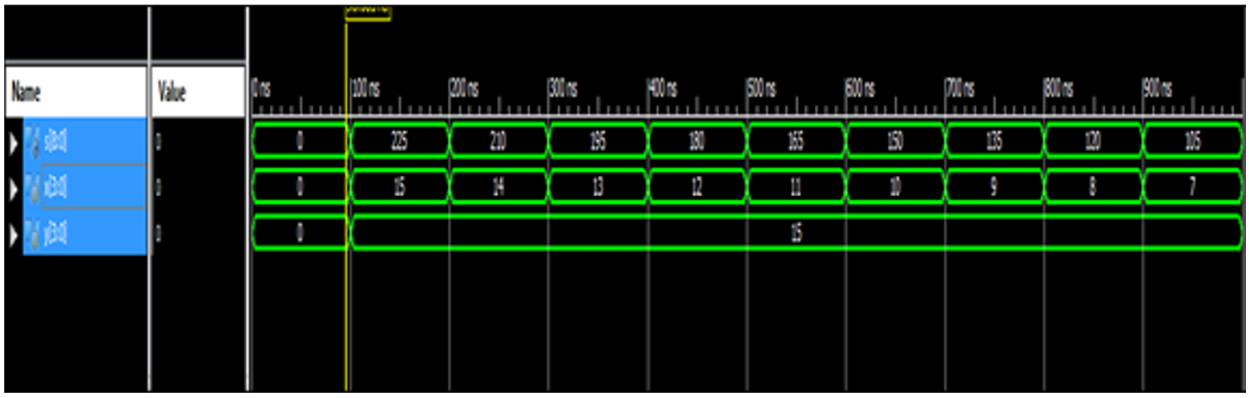


Figure 4.27 Results Systolic Array Multiplier

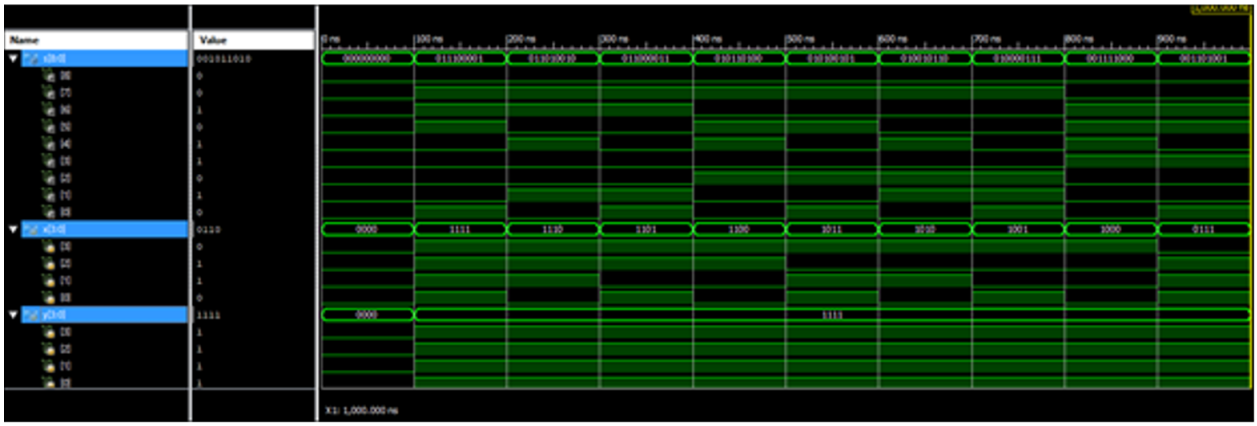


Figure 4.28 Results of SAM internal blocks COG and MIG gates

Figure 4.29 Results of Missing gate fault

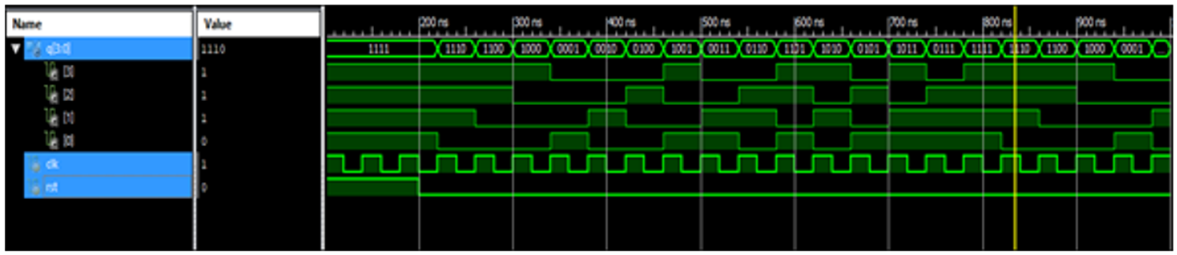


Figure 4.30 Results of BILBO LFSR Mode

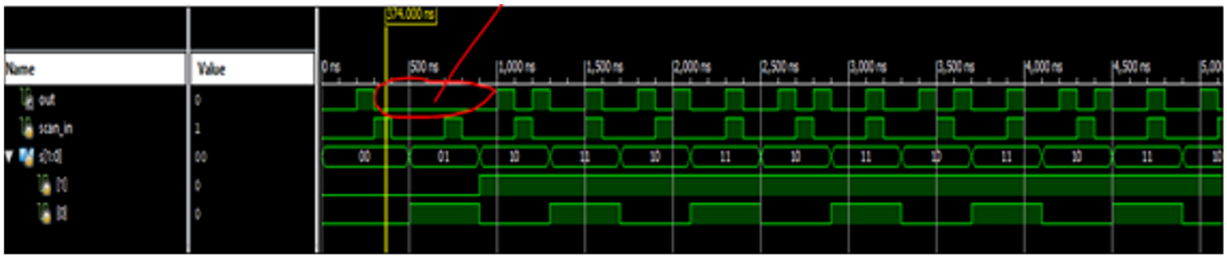


Figure 4.31 Results of BILBO MISR Mode signature comparison

mode is shown in Fig 4.30 The SAF is found from the values generated by BILBO are tested and compared with the existing signature after injecting faults are shown in Fig 4.31. Different test vector combinations are provided below for finding SAF, MSAF and MGF. Finding test vector of the resultant at stuck at 0/1 is FAILED− − −−.The output is correct atrequired places

x1=1, x2=0, x3=0, x4=1, x5=0, scan$_i n$ = 1, out=1, 3100

Finding test vector of the resultant at stuck at 0/1 is PASSED

x1=0, x2=0, x3=1, x4=0, x5=0, scan$_i n$=0, out=0, 3200

Finding test vector of the resultant at stuck at 0/1 is FAILED− − −−The output is correct at required places

x1=1, x2=0, x3=0, x4=0, x5=0, scan$_i n$=0, out=0, 3300

Finding test vector of the resultant at stuck at 0/1 is FAILED− − −−The output is correct at required places

x1=0, x2=1, x3=1, x4=0, x5=0, scan$_i n$=0, out=1, 3400

Finding test vector of the resultant at stuck at 0/1 is PASSED

x1=1, x2=0, x3=0, x4=1, x5=0, scan$_i n$=1, out=0, 3500

Finding test vector of the resultant at stuck at 0/1 is FAILED− − −−The output is correct at required places

Table 4.5 Results Comparison of Fault Analysis

Fault Analysis	**Conventional Multiplier [64]**	**Conventional Multiplier [60]**	**Proposed Multiplier**
Good Signature	200	200	200
No of faults	138	128	138
No of faults detected	130	128	134
Fault coverage	96%	96.28%	97%

Table 4.6 Results Comparison of Local Utilization

Local Utilization	**Conventional Multiplier [64]**	**Conventional Multiplier [60]**	**Proposed Multiplier**
No of Slices	76.11%	75.11%	70.28%
No of 4 input LUT's	26%	26%	25%
Time Delays(ns)	28.24	28.22	28

x1=0, x2=0, x3=1, x4=0, x5=0, scan$_i n$=0, out=1, 3600

Finding test vector of the resultant at stuck at 0/1 is PASSED

x1=1, x2=0, x3=0, x4=0, x5=0, scan$_i n$=0, out=0, 3700

Finding test vector of the resultant at stuck at 0/1 is FAILED− − −−The output is correct at required places

x1=0, x2=1, x3=1, x4=0, x5=0, scan$_i n$=0, out=0, 3800

Finding test vector of the resultant at stuck at 0/1 is FAILED− − −−The output is correct at required places

x1=1, x2=0, x3=0, x4=1, x5=0, scan$_i n$=1, out=1, 3900

Finding test vector of the resultant at stuck at 0/1 is PASSED

x1=0, x2=0, x3=1, x4=0, x5=0, scan$_i n$=0, out=0, 4000

The fault analysis of conventional multipliers is compared with the proposed SAM shown in Table 4.5. Furthermore, the local utilization of the SAM is compared with the conventional multipliers are shown in Table 4.6.

4.4 Testing and fault diagnosis of ALU Blocks using Advanced BIST Algorithms

Memory runtime faults are common, and they can result in catastrophic system failures. Built In Self-Test (BIST) can generate automatic test patterns to detect faults in memories. But it consumes large substantiate dynamic power. So, an AXI based self-test memory architecture is proposed to improve parallel read and write capability.

4.4.1 Implementation Architecture of AXI with MARCH-A Algorithm

The high-speed testing architecture with Block Random Access Memory (BRAM) in AXI based implementation which reduce the testing time is shown in Fig.4.32. The AXI based march-A algorithm presented here can execute read and write operations parallelly which is shown in Fig 4.33. Initially, (n+1) clock pulses are consumed for the execution of write-0 and read-0 operations which follow either top- down or bottom-up approach. Where as for write-1 and read-1 operations only one clock pulse is consumed. The flow chart representation of AXI-BIST is shown in Fig.4.34. Furthermore, depends on the read or write conditions the SAF and transition faults are detected for maximum memory locations. For example, if the write instruction changes suddenly from state '0' to '1' those transitions are checked and identified efficiently. The march-A with the utilization of clock pulses for specific read and write operations are represented in Fig 4.33. If the

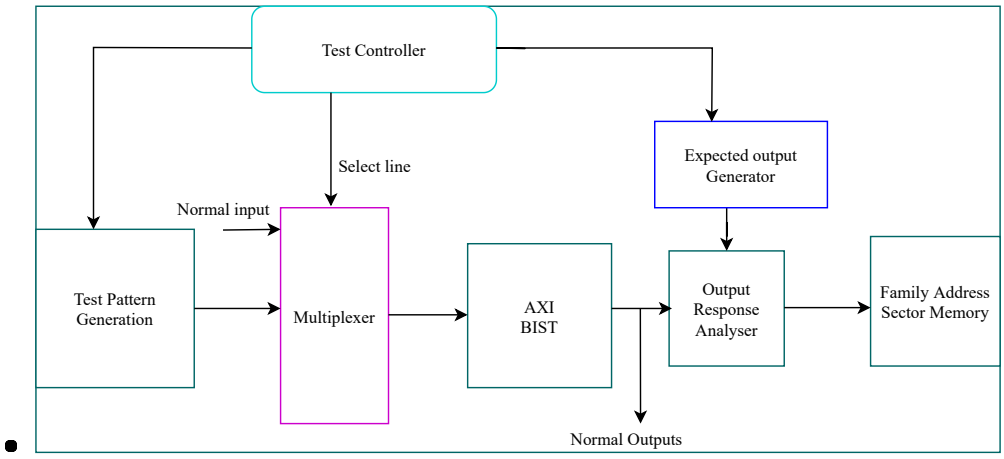


Figure 4.32 BIST Block Diagram

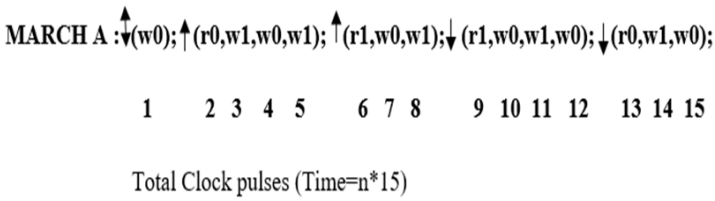


Figure 4.33 Read and write operations of March A for 15 clock pulses

Table 4.7 Results Comparison

S.No.		[39]	[86]	[78]	Proposed BIST
1	LUT count	1050	1176	1366	965
2	Dynamic Power (w)	0.58	0.61	0.96	0.32
3	Number of clock cycles (per one instruction)	10	19	27	7

algorithm is unable to detect the sudden changes in the states, then transition faults are raised. The AXI BIST consumes (9n+1) clock pulses to complete the march-A algorithm which is faster as shown in Fig 4.35 and also used a smaller number of resources than the existing algorithms.

4.4.2 Results and Summary

The simulation results of AXI based march-A are carried out in Vivado 2017 and the register transfer level schematic is shown in Fig 4.36. Generally, the existing approaches proceed to read instructions after the successful completion of write instructions for various memory locations. The AXI BIST require two clock pulses to enable the read instruction. The parallel processing of valid address signals of Axiwritevalid for write and AxiReadvalid for read are shown in Fig 4.37.The serial read and write process of 'wea' signal for conventional memory BIST is shown in Fig.3.1.14. The signal 'wea' is '1' it for write operation otherwise '0' for read operation.The AXI BIST consumes a smaller number of LUTs with reduced dynamic power when compared to the conventional BIST is shown in Table 4.7. The number of clock cycles consumed for the read and write operations are reduced to '7'.

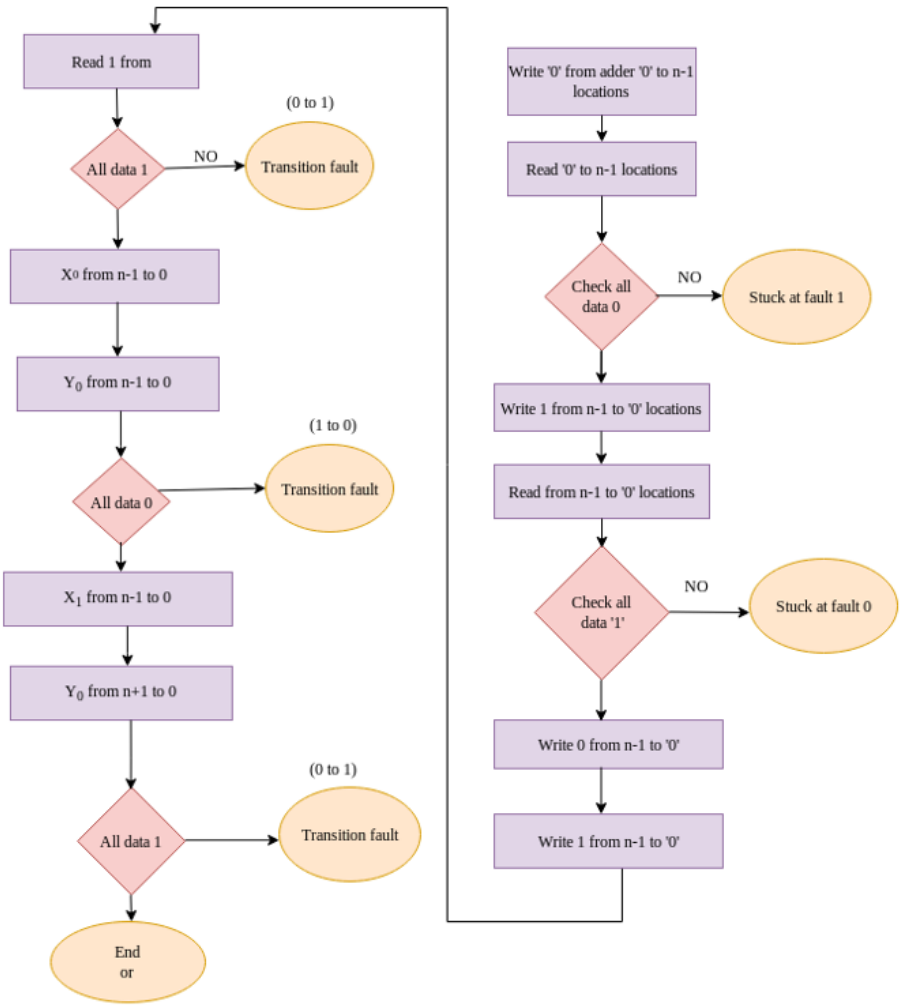


Figure 4.34 Flow chart for AXI BIST

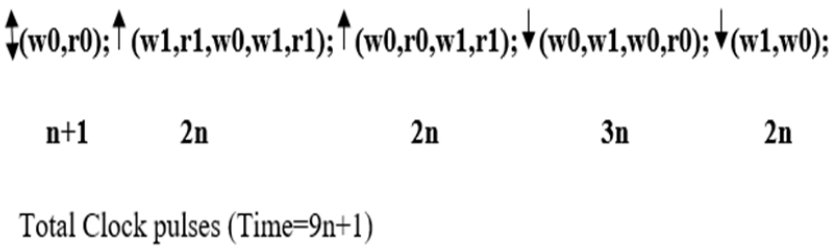


Figure 4.35 Read and write operations of March A for (9n+1) clock pulses

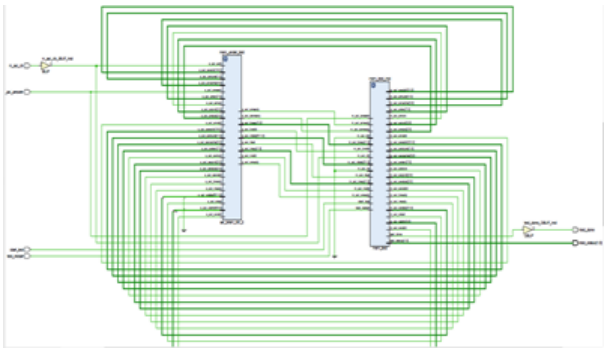


Figure 4.36 RTL Schematic

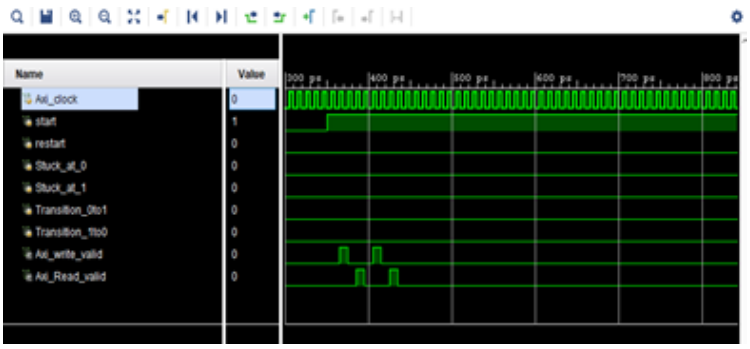


Figure 4.37 Valid addressing forwrite and read

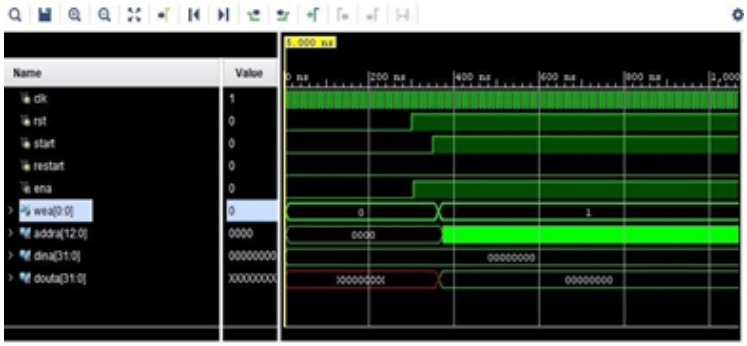


Figure 4.38 Serial write and read

4.5 Conclusion

The processor with 32-bit instruction set and RAM are tested using VMA and ROBDD designs. When compared to previous designs, the VMA and ROBDD technique was determined to have the best performance for detecting flaws in sequential architectures. Modern tests are insufficient to discover multiple stuck at faults (MSAF) such as bridge faults (BF) and toggling faults (TF). The maximum fault coverage of finding these faults using VMA and ROBDD is achieved as 80% and 95% respectively. The proposed multiplier in SAM designed with reversible logic gates performs faster as compared to the conventional multipliers used for low-power and high-speed applications. While testing SAM the faults of SAF and MGF are done efficiently with coverage of pattern generation. Future designs of SoC or sub-systems have to be integrated with BILBO scheme which has achieved 97% of fault coverage in testing system designs.AXI BIST is implemented with march algorithm attained 74% high speed, 10% area reduction and 50% power reduction in the testing of memory blocks.

Chapter 5

Conclusions and Future Scope

This chapter concludes the thesis by underlining the main contributions. It also presents the possible directions of future work.

5.1 Conclusions

Testing and fault diagnosys is viable to detect the faults in the circuits and also increase the device performance and reliability. The design of testability is applied on latest VLSI designs and implemented through BIST concepts to increase the effectiveness of finding faults like stuck at fault, fault location, delay fault, toggling fault, transition fault, bridging fault, missing gate fault etc. The proposed algorithms will take suitable actions to improve the design performance of the circuit and optimized the fault objectives like maximum fault coverage, higher speed of testing, less test power dissipation and less test area overhead.

- In chapter 3, The non-incremental computing with adaptive genetic algorithm is performed static timing analysis to compute target path delay, critical path delay and probability of target path fault delay. In this regard when Monte-carlo simulations are performed on FinFET based VLSI circuits, we achieved a higher speed i.e.10 times as compared to the conventional designs. A conventional PODEM algorithm is performed on FinFET based VLSI combinational circuits to detect node level faults and their location.

- In Chapter 4, VMA is demonstrated to test memory in RISC processors with possible corner cases, achieving a fault coverage of 80%, which is 33% greater than traditional methods. Furthermore, for corner defects verification in memory blocks, the chip area was reduced by 33% and a speed of 29% was achieved. ROBDD technique is implemented to find the faults in sequential circuits detects transition faults with fault coverage of 95%, reduced in 50% area overhead and 33% of higher speed achieved than VMA. The usage of reversible logic gates with VMA implemented in ALU achieved fast computing along with 50% power consumption and 80% efficiency. The BILBO technique detected MGFs in SAM with a fault coverage of 90%, increased speed by 28%, and reduced area overhead by 7.6%, which is better than prior approaches. AXI BIST is implemented with march algorithm attained 74% high speed, 10% area reduction and 50% power reduction in the testing of memory blocks.

5.2 Future Scope

The work proposed in this thesis can be extended for future research. Some of the possible directions in which the problems can be further pursued are:

- To compute critical path delay for FinFET based VLSI circuits in sub-nano region $\leq$ 7 nm using PODEM algorithm..
- AXI BIST technique can be applied to various memory blocks using MARCH – C, VEDIC MARCH C to achieve high speed of testing.
- VMA along with ROBDD can be configured as IP and extended to test ten or more memory blocks to detect maximum faults in the circuit.
- Vedic ALU with reversible logic gates like multiplier or squarer bit size can be increased for better performance and power reduction
- BILBO or GRM schemes can be implemented on advanced designs for fault Injection

Publications

List of International Journals:

1. K.V.B.V Rayudu, D R Jahagirdar and P Srihari Rao, "Design and Development of a Modified AXI Based BIST Technique" *International Journal of Recent Technology and Engineering,* Volume 8, Issue 4, pp.8023-8029, November, 2019. **(Scopus)**

2. K.V.B.V Rayudu, D R Jahagirdar and P Srihari Rao, "Testing and Diagnosis of Delay Faults in FinFet VLSI Circuits using Non-Incremental Genetic Algorithm" *International Journal of Innovative Technology and Exploring Engineering,*Volume -9 Issue -2, pp. 1673-1679, December, 2019.**(Scopus)**

3. K.V.B.V Rayudu, D R Jahagirdar and P Srihari Rao, "Application of PODEMn Algorithm for Fault Detection and Location in FinFET based Combinational VLSI Circuits" *International Journal of Engineering and Advanced Technology,,*Volume -9 Issue -2, pp.1-7, December, 2019. **(Scopus)**

4. K.V.B.V Rayudu, D R Jahagirdar and P Srihari Rao, "Modern Design approach of Faults (Toggling Faults, Bridge Faults and SAT) of Reduced Ordered Binary Decision Diagram based on Combo amp; Sequential Blocks" *International Journal of Reconfigurable and Embedded Systems (IJRES)* Vol. 9, No. 2, pp.158-168, July, 2020. **(SCOPUS)**

5. K.V.B.V Rayudu, D R Jahagirdar and P Srihari Rao, "Design and Testing of SAM(Systolic Array Multiplier) using Fault Injection Schemes" *International Journal of Computer Science and Information Technologies.* **((accepted for publication)) (SCOPUS)**

6. K.V.B.V Rayudu, D R Jahagirdar and P Srihari Rao, "Modern Approach of Design and Testing of Memory using Vedic march Algorithm for RISC Processor Relevance" *Microsystems System Technologies (MITE/Mamp;MS).* **(((Under review))) (SCI)**

Bibliography

[1] R. A. Thakker, C. Sathe, A. B. Sachid, M. S. Baghini, V. R. Rao, and M. B. Patil, "A novel table-based approach for design of finfet circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 7, pp. 1061–1070, 2009.

[2] M. Bushnell and V. Agrawal, *Essentials of electronic testing for digital, memory and mixed-signal VLSI circuits*. Springer Science & Business Media, 2004, vol. 17.

[3] E. J. McCluskey, "Verification testing? a pseudoexhaustive test technique," *IEEE transactions on computers*, vol. 33, no. 06, pp. 541–546, 1984.

[4] G. Sudhagar, G. Ramesh *et al.*, "Implementation of a novel architecture for vlsi testing," in *2013 International Conference on Emerging Trends in VLSI, Embedded System, Nano Electronics and Telecommunication System (ICEVENT)*. IEEE, 2013, pp. 1–4.

[5] M. Abramovici, M. A. Breuer, A. D. Friedman *et al.*, *Digital systems testing and testable design*. Computer science press New York, 1990, vol. 2.

[6] M. F. Abdulla, C. Ravikumar, and A. Kumar, "A novel bist architecture with built-in self check," in *Proceedings of 9th International Conference on VLSI Design*. IEEE, 1996, pp. 57–60.

[7] C. Beyond and M. Moore, "International technology roadmap for semiconductors."

[8] J. Gu, J. Keane, S. Sapatnekar, and C. Kim, "Width quantization aware finfet circuit design," in *IEEE Custom Integrated Circuits Conference 2006*. IEEE, 2006, pp. 337–340.

[9] G. Harutyunyan, G. Tshagharyan, and Y. Zorian, "Impact of parameter variations on finfet faults," in *2015 IEEE 33rd VLSI Test Symposium (VTS)*. IEEE, 2015, pp. 1–4.

[10] R. Hajare, C. Lakshminarayana, S. C. Sumanth, and A. Anish, "Design and evaluation of finfet based digital circuits for high speed ics," in *2015 International Conference on Emerging Research in Electronics, Computer Science and Technology (ICERECT)*. IEEE, 2015, pp. 162–167.

[11] M. O. Simsir, A. Bhoj, and N. K. Jha, "Fault modeling for finfet circuits," in *2010 IEEE/ACM International Symposium on Nanoscale Architectures*. IEEE, 2010, pp. 41–46.

[12] A. K. Jameil, "A new single stuck fault detection algorithm for digital circuits," *Int. J. Eng. Res. Gen. Sci*, vol. 3, no. 1, pp. 1050–1056, 2015.

[13] R. L. Wadsack, "Fault modeling and logic simulation of cmos and mos integrated circuits," *The Bell System Technical Journal*, vol. 57, no. 5, pp. 1449–1474, 1978.

[14] I. Kohavi and Z. Kohavi, "Detection of multiple faults in combinational logic networks," *IEEE Transactions on Computers*, vol. 100, no. 6, pp. 556–568, 1972.

[15] M. Wagner, "Efficient algorithms for fundamental statistical timing analysis problems in delay test applications of vlsi circuits," 2016.

[16] M. Wagner and H.-J. Wunderlich, "Incremental computation of delay fault detection probability for variation-aware test generation," in *2014 19th IEEE European Test Symposium (ETS)*. IEEE, 2014, pp. 1–6.

[17] M. Vijaykumar and V. Vasudevan, "Statistical static timing analysis using a skew-normal canonical delay model," in *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2014, pp. 1–6.

[18] P. Goel and B. C. Rosales, "Podem-x: An automatic test generation system for vlsi logic structures," in *18th Design Automation Conference*. IEEE, 1981, pp. 260–268.

[19] A. K. Jameil, "A new single stuck fault detection algorithm for digital circuits," *Int. J. Eng. Res. Gen. Sci*, vol. 3, no. 1, pp. 1050–1056, 2015.

[20] W. Hong, J. Choi, and H. Chang, "A programmable memory bist for embedded memory," in *2008 International SoC Design Conference*, vol. 2. IEEE, 2008, pp. II–195.

[21] L. Varghese and G. Suranya, "Test pattern generation using lfsr with reseeding scheme for bist designs," *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, vol. 3, no. 5, pp. 452–458, 2014.

[22] R. Silveira, Q. Qureshi, and R. Zeli, "Flexible architecture of memory bists," in *2018 IEEE 19th Latin-American Test Symposium (LATS)*. IEEE, 2018, pp. 1–6.

[23] T. Koshy and C. Arun, "Diagnostic data detection of faults in ram using different march algorithms with bist scheme," in *2016 International Conference on Emerging Technological Trends (ICETT)*. IEEE, 2016, pp. 1–6.

[24] A. Agrawal, A. Saldanha, L. Lavagno, and A. L. Sangiovanni-Vincentelli, "Compact and complete test set generation for multiple stuck-faults," in *Proceedings of the 1996 IEEE/ACM international conference on Computer-aided design*, 1997, pp. 212–219.

[25] T. Shah, V. Singh, and A. Matrosova, "Robdd based path delay fault testable combinational circuit synthesis," in *2016 IEEE East-West Design & Test Symposium (EWDTS)*. IEEE, 2016, pp. 1–4.

[26] Y. R. Babu and Y. Syamala, "Implementation and testing of multipliers using reversible logic," 2011.

[27] M. K. Thomsen, R. Glück, and H. B. Axelsen, "Reversible arithmetic logic unit for quantum arithmetic," *Journal of Physics A: Mathematical and Theoretical*, vol. 43, no. 38, p. 382002, 2010.

[28] T. Rakshith and R. Saligram, "Parity preserving logic based fault tolerant reversible alu," in *2013 IEEE Conference on Information & Communication Technologies*. IEEE, 2013, pp. 485–490.

[29] C. Madhulika, V. S. P. Nayak, C. Prasanth, and T. H. S. Praveen, "Design of systolic array multiplier circuit using reversible logic," in *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*. IEEE, 2017, pp. 1670–1673.

[30] S. K. Reddy, N. Reddy, and A. R. Shankar, "Design and implementation of vlsi systolic array multiplier for dsp applications," *International Journal of Computing*, vol. 2, no. 4, pp. 140–146, 2013.

[31] N. Bhardwaj, V. Mahor, and M. Pattanaik, "Robust finfet based highly noise immune power gated sram circuit design," in *2015 International Conference on Communication Networks (ICCN)*. IEEE, 2015, pp. 310–316.

[32] G. Harutyunyan, G. Tshagharyan, V. Vardanian, and Y. Zorian, "Fault modeling and test algorithm creation strategy for finfet-based memories," in *2014 IEEE 32nd VLSI Test Symposium (VTS)*. IEEE, 2014, pp. 1–6.

[33] S. Musala and A. Srinivasulu, "Self testing and fault secure xor/xnor circuit using finfets," in *2016 International Conference on Communication and Signal Processing (ICCSP)*. IEEE, 2016, pp. 1222–1226.

[34] K.-Y. Chiang, Y.-H. Ho, Y.-W. Chen, C.-S. Pan, and J. C.-M. Li, "Fault simulation and test pattern generation for cross-gate defects in finfet circuits," in *2015 IEEE 24th Asian Test Symposium (ATS)*. IEEE, 2015, pp. 181–186.

[35] D. Xiang, X. Wen, and L.-T. Wang, "Low-power scan-based built-in self-test based on weighted pseudorandom test pattern generation and reseeding," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 3, pp. 942–953, 2016.

[36] M. Fujita and A. Mishchenko, "Efficient sat-based atpg techniques for all multiple stuck-at faults," in *2014 International Test Conference*. IEEE, 2014, pp. 1–10.

[37] J.-C. Lo, C. Metra, and F. Lombardi, "Guest editors' introduction: Special section on design and test of systems-on-chip (soc)," *IEEE Transactions on Computers*, vol. 55, no. 02, pp. 97–98, 2006.

[38] A. Lodi, A. Cappelli, M. Bocchi, C. Mucci, M. Innocenti, C. De Bartolomeis, L. Ciccarelli, R. Giansante, A. Deledda, F. Campi, M. Toma, and R. Guerrieri, "Xisystem: a xirisc-based soc with reconfigurable io module," *IEEE Journal of Solid-State Circuits*, vol. 41, no. 1, pp. 85–96, 2006.

[39] P. K. John *et al.*, "Bist architecture for multiple rams in soc," *Procedia computer science*, vol. 115, pp. 159–165, 2017.

[40] E. Dupont, M. Nicolaidis, and P. Rohr, "Embedded robustness ips for transient-error-free ics," *IEEE Computer Architecture Letters*, vol. 19, no. 03, pp. 56–70, 2002.

[41] A. J. Van De Goor and I. Schanstra, "Address and data scrambling: Causes and impact on memory tests," in *Proceedings First IEEE International Workshop on Electronic Design, Test and Applications' 2002.* IEEE, 2002, pp. 128–136.

[42] Y. Bansal, C. Madhu, and P. Kaur, "High speed vedic multiplier designs-a review," in *2014 Recent Advances in Engineering and Computational Sciences (RAECS).* IEEE, 2014, pp. 1–6.

[43] H. Takahashi, K. O. Boateng, K. K. Saluja, and Y. Takamatsu, "On diagnosing multiple stuck-at faults using multiple and single fault simulation in combinational circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, no. 3, pp. 362–368, 2002.

[44] V. Agarwal and A. Fung, "Multiple fault testing of large circuits by single fault test sets," *IEEE Transactions on Circuits and Systems*, vol. 28, no. 11, pp. 1059–1069, 1981.

[45] J. L. Hughes and E. J. McCluskey, "An analysis of the multiple fault detection capabilities of single stuck-at fault test sets," in *Proceedings of the 1984 international test conference on The three faces of test: design, characterization, production*, 1984, pp. 52–58.

[46] J. L. Hughes, "Multiple fault detection using single fault test sets," *IEEE transactions on computer-aided design of integrated circuits and systems*, vol. 7, no. 1, pp. 100–108, 1988.

[47] N. Gadda and U. Eranna, "64-bit alu design using vedic mathematics," in *2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE).* IEEE, 2020, pp. 1–4.

[48] K. Morghade and P. Dakhole, "Design of fast vedic multiplier with fault diagnostic capabilities," in *2016 International Conference on Communication and Signal Processing (ICCSP).* IEEE, 2016, pp. 0416–0419.

[49] A. Gupta, U. Malviya, and V. Kapse, "Design of speed, energy and power efficient reversible logic based vedic alu for digital processors," in *2012 Nirma University International Conference on Engineering (NUiCONE)*. IEEE, 2012, pp. 1–6.

[50] B. Ravali, M. M. Priyanka, and T. Ravi, "Optimized reversible logic design for vedic multiplier," in *2015 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*. IEEE, 2015, pp. 127–133.

[51] K. Shirane, T. Yamamoto, I. Taniguchi, Y. Hara-Azumi, S. Yamashita, and H. Tomiyama, "Maximum error-aware design of approximate array multipliers," in *2019 International SoC Design Conference (ISOCC)*. IEEE, 2019, pp. 73–74.

[52] E. McCluskey and S. Bozorgui-Nesbat, "Design for autonomous test," *IEEE Transactions on Circuits and Systems*, vol. 28, no. 11, pp. 1070–1079, 1981.

[53] C.-I. Chen and R. Smith, "A self-testing and self-diagnostic systolic array cell for signal processing," in *1991 International Conference on Wafer Scale Integration*. IEEE Computer Society, 1991, pp. 75–76.

[54] V. S. P. Nayak, N. Ramchander, T. Marandi, and A. V. Krishna, "Analysis and design of low-power reversible bilbo," in *2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*. IEEE, 2016, pp. 113–117.

[55] B. K. Saptalakar, D. Kale, M. Rachannavar, and M. Pavankumar, "Design and implementation of vlsi systolic array multiplier for dsp applications," *International Journal of Scientific Engineering and Technology*, vol. 2, no. 3, pp. 156–159, 2013.

[56] L. Wang and I. Hartimo, "Systolic array for binary multiplier," in *Proceedings of ICSIPNN'94. International Conference on Speech, Image Processing and Neural Networks*. IEEE, 1994, pp. 745–748.

[57] C. H. Bennett, "Logical reversibility of computation," *IBM journal of Research and Development*, vol. 17, no. 6, pp. 525–532, 1973.

[58] ——, "Notes on the history of reversible computation," *ibm Journal of Research and Development*, vol. 32, no. 1, pp. 16–23, 1988.

[59] M. S. Islam, M. Rahman, and Z. Begum, "Synthesis of fault tolerant reversible circuits," in *IEEE International Conference on Testing and Diagnosis*, 2009, pp. 28–29.

[60] A. Vuksic and K. Fuchs, "A new bist approach for delay fault testing," in *Proceedings of European Design and Test Conference EDAC-ETC-EUROASIC.* IEEE, 1994, pp. 284–288.

[61] J. Lee and N. A. Touba, "Lfsr-reseeding scheme achieving low-power dissipation during test," *IEEE transactions on computer-aided design of integrated circuits and systems*, vol. 26, no. 2, pp. 396–401, 2007.

[62] M. Psarakis, D. Gizopoulos, A. Paschalis, and Y. Zorian, "An effective bist architecture for sequential fault testing in array multipliers," in *Proceedings 17th IEEE VLSI Test Symposium (Cat. No. PR00146).* IEEE, 1999, pp. 252–258.

[63] K. Shirane, T. Yamamoto, I. Taniguchi, Y. Hara-Azumi, S. Yamashita, and H. Tomiyama, "Maximum error-aware design of approximate array multipliers," in *2019 International SoC Design Conference (ISOCC).* IEEE, 2019, pp. 73–74.

[64] S. A. Mozhi and P. Ramya, "Efficient bit-parallel systolic multiplier over gf (2 m)," in *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT).* IEEE, 2016, pp. 4899–4902.

[65] P. K. Meher and X. Lou, "Low-latency, low-area, and scalable systolic-like modular multipliers for $gf(2\{m\})$ based on irreducible all-one polynomials," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, no. 2, pp. 399–408, 2016.

[66] S. Talapatra, H. Rahaman, and J. Mathew, "Low complexity digit serial systolic montgomery multipliers for special class of

{

\rm GF}(2^{m}) ,″ *IEEE transactions on very large scale integration (VLSI) systems*, *vol*. 18, *n*
−852, 2009.

[67] J. Xie, J. jun He, and P. K. Meher, "Low latency systolic montgomery multiplier for finite field $gf(2\{m\})$ based on pentanomials," *IEEE transactions on very large scale integration (VLSI) systems*, vol. 21, no. 2, pp. 385–389, 2012.

[68] A. Reyhani-Masoleh, "Comments on "low-latency digit-serial systolic double basis multiplier over $gf(2\{m\})$ using subquadratic toeplitz matrix-vector product approach"," *IEEE Transactions on Computers*, vol. 64, no. 4, pp. 1215–1216, 2015.

[69] H. Rahaman, J. Mathew, and D. K. Pradhan, "Test generation in systolic architecture for multiplication over $gf(2\{m\})$," *IEEE transactions on very large scale integration (VLSI) systems*, vol. 18, no. 9, pp. 1366–1371, 2009.

[70] G. Harutyunyan, S. Shoukourian, and Y. Zorian, "Fault awareness for memory bist architecture shaped by multidimensional prediction mechanism," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 3, pp. 562–575, 2018.

[71] D.-C. Huang, W.-B. Jone, and S. R. Das, "An efficient parallel transparent bist method for multiple embedded memory buffers," in *VLSI Design 2001. Fourteenth International Conference on VLSI Design.* IEEE, 2001, pp. 379–384.

[72] I. Voyiatzis and C. Efstathiou, "Input vector monitoring concurrent bist architecture using sram cells," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 7, pp. 1625–1629, 2013.

[73] S.-P. Jung, S.-W. Song, D.-H. Lee, K.-J. Kim, K.-S. Cho, and J.-S. Park, "Design & verification of 16 bit risc processor," in *Proceedings of the IEEK Conference.* The Institute of Electronics and Information Engineers, 2008, pp. 423–424.

[74] S. K. S. Hari, S. Shroff, S. N. Mahammad, and V. Kamakoti, "Efficient building blocks for reversible sequential circuit design," in *2006 49th IEEE International Midwest Symposium on Circuits and Systems*, vol. 1. IEEE, 2006, pp. 437–441.

[75] R. Laundauer, "Irreversibility and heat generation in the computational process," *IBM Journal of Research and Development*, vol. 5, no. 3, p. 83, 1961.

[76] R. Pennucci, R. Jurasek, W. Hokenmaier, L. Patrick, J. Bucci, D. Labrecque, and D. Kinney, "An analysis of an inexpensive memory test solution," in *2018 IEEE 27th North Atlantic Test Workshop (NATW).* IEEE, 2018, pp. 1–6.

[77] T. Li, H. Lee, G. Bak, and S. Baeg, "Failure signature analysis of power-opens in ddr3 sdrams," *Microelectronics Reliability*, vol. 88, pp. 277–281, 2018.

[78] M. Kumar, "An efficient fault detection of fpga and memory using built-in self test [bist]," *American Journal of Electrical and Computer Engineering*, vol. 3, no. 1, pp. 38–45, 2019.

[79] R. Laundauer, "Irreversibility and heat generation in the computational process," *IBM Journal of Research and Development*, vol. 5, no. 3, p. 83, 1961.

[80] R. Sharma, M. Kaur, and G. Singh, "Design and fpga implementation of optimized 32-bit vedic multiplier and square architectures," in *2015 International Conference on Industrial Instrumentation and Control (ICIC)*. IEEE, 2015, pp. 960–964.

[81] P. Martha, N. Kajal, P. Kumari, and R. Rahul, "An efficient way of implementing high speed 4-bit advanced multipliers in fpga," in *2018 2nd International Conference on Electronics, Materials Engineering & Nano-Technology (IEMENTech)*. IEEE, 2018, pp. 1–5.

[82] M. Esonu, A. Al-Khalili, and D. Al-Khalili, "Variations on the theme for designing fault-tolerant systolic array architectures," in *[1991] IEEE Pacific Rim Conference on Communications, Computers and Signal Processing Conference Proceedings*. IEEE, 1991, pp. 107–110.

[83] M. Parvathi, N. Vasantha, and K. Parasad, "Modified march c-algorithm for embedded memory testing," *International Journal of Electrical and Computer Engineering*, vol. 2, no. 5, p. 571, 2012.

[84] R. Drechsler, J. Shi, and G. Fey, "Synthesis of fully testable circuits from bdds," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 3, pp. 440–443, 2004.

[85] A. Banerjee and A. Pathak, "On the synthesis of sequential reversible circuit," *arXiv preprint arXiv:0707.4233*, 2007.

[86] H. Thapliyal, N. Ranganathan, and S. Kotiyal, "Design of testable reversible sequential circuits," *IEEE transactions on very large scale integration (VLSI) systems*, vol. 21, no. 7, pp. 1201–1209, 2012.