Fully Testable Circuit Synthesis for Delay and Multiple Stuck-at Faults

A Thesis
Submitted in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy
by

Toral Ankur Shah
(Roll No. 124070027)

Supervisor:
Virendra Singh

Department of Electrical Engineering
Indian Institute of Technology Bombay
Mumbai 400076 (India)

27 February 2018

Abstract

Integrated circuits in deep sub-micron technologies usually suffer from subtle additional delays due to process variations on both gates and interconnects. These variations have minimal impact on path delays of short interconnects but significant impact on path delays of very long interconnects whose resistance can be as high as a *kohm* or even greater. Additionally, aggressive fabrication techniques make the chip prone to defects that manifest as multiple stuck-at faults. The *Single Stuck-at Fault*(SSAF) model is a popular model which covers many defects but not all. *Path Delay Fault*(PDF) and *Multiple Stuck-at Fault*(MSAF) models uncover defects that are missed by the SSAF model.

Testability challenges in any synthesis approach include complete fault efficiency and ease of test generation for a fault model. The efficacy of a fault model lies in the metric of the range of the defects covered. In this work we have focussed on synthesis that gurantees complete fault efficiency for PDF and MSAF models, SSAFs are implicitly covered.

Reduced Ordered Binary Decision Diagram (ROBDD) based combinational circuit synthesis offers an increased degree of testability. This work proposes an ROBDD based circuit synthesis which has complete testability under single stuck-at, multiple stuck-at and path delay fault models. The circuit is constructed using a decision making sub-circuit (2x1 mux) for each ROBDD node. A modified mux is used for nodes connected to leaf nodes. The ROBDD based circuit can be represented by a *Disjoint Sum of Products* (DSOP) expression. Each term in this expression represents a unique path from leaf node 1 to root node of the ROBDD. The order of the variables that control each node is fixed. It is the primary input themselves that control the mux of an ROBDD node. Thus the rank of the input variables is predetermined.

To prove complete delay testability of all paths of a circuit under test, it is necessary to establish that the circuit has 100% single stuck-at fault coverage. The ROBDD based implementation provides complete single stuck-at fault testability. This is used as a basis to investigate delay testability of each path of the circuit under test.

The placement of the node in the ROBDD decides the condition whether the path that begins at it's controlling variable will be tested robustly or non-robustly. We demonstrate testability under all conditions. As a result, *all* paths of the circuit represented as DSOPs satisfy the conditions of path delay testability. The algorithm to derive the test vector pairs that generate the required signal transition is developed. The test vector pair

consists of only primary inputs, no additional inputs are required. The PDF tests are derived through DSOP manipulation.

To gain more confidence in the testable synthesis, it is analysed for multiple stuck-at fault testability. A circuit has a maximum of $3^n - 1$ multiple stuck-at fault combinations where n is the number of circuit nets. A thorough investigation of multiple stuck-at fault (MSAF) testability of ROBDD based implementation is done. Concepts of *pseudoexhaustive testing* and MSAF testability of two-level AND-OR networks are combined and applied to the 2x1 mux based implementation. First, the pseudoexhaustive test vector set for one sub-circuit (2x1 mux) is derived. This set has four vectors. The test vectors in this set are responsible for sensitizing and propagating MSAF condition to the circuit output. It is proved that, for an ROBDD with N nodes, the circuit implementation has an upper bound of $3N$ test vectors that cover MSAFs of all muxes. We systematically move towards proving that, the test vector set that detects MSAFs in all sub-circuits also detects all irredundant MSAFs of the *complete* circuit.

Two methods to derive the test vector set for MSAFs are presented. The first method uses BDD manipulations to derive pseudoexhaustive test vector set for each node. The second method uses the available *DSOP* expression. Each term of the DSOP expression is processed to derive a partial test set. All partial test sets are combined and checked for redundancies and a complete MSAF test set is derived.

Single stuck-at faults are covered by both path delay tests and MSAF tests. The test generation complexity of the PDF as well as MSAF tests is $O(m \times n)$ where m is the number of primary inputs and n is the number of ROBDD nodes.

The contribution of the thesis is as follows:

- Fully testable ROBDD based hardware synthesis approach for delay and multiple stuck-at faults
- Algorithms for test generation

Table of Contents

List of Figures vii

List of Tables xi

1 Introduction 1

1.1 Delay faults . . . 3

1.1.1 Contributors to Delay Faults . . . 4

1.2 Delay Fault Testability . . . 7

1.2.1 Delay Fault Models . . . 7

1.2.2 Path Delay Fault Tests . . . 8

1.3 Multiple Stuck-at Fault Testability . . . 12

1.4 Thesis Organisation . . . 13

2 Test Challenges and Prior Work 17

2.1 Sequential Circuit Testing . . . 17

2.1.1 Testing Sequential Circuits with Combinational ATPG . . . 18

2.1.2 Acyclic circuits . . . 21

2.2 Literature Survey . . . 24

2.2.1 Existing Approaches to Delay Testable Combinational Circuit Synthesis . . . 24

2.2.2 Existing Approaches to Multiple Stuck-at Fault Testability . . . 36

3 ROBDD Based Synthesis 45

3.1 Binary Decision Diagrams . . . 45

3.2 Synthesis Without Additional Input . . . 48

4 Path Delay Fault Testability 53

4.1 Single Stuck-at Fault Testability . . . 53

4.2 Robust and Non-Robust Testability Conditions . . . 54

4.3 Test Vector Generation . . . 58

5 Multiple Stuck-at Fault Testability Analysis **59**

5.1 MSAF testability of two-level AND-OR circuits 60

5.2 Deriving MSAF test set for each node 63

5.3 Test vector generation 64

5.4 MSAF detection with proposed approach 66

6 Algorithms to Derive Test Vectors for PDFs and MSAFs **71**

6.1 Derivation of PDF tests 71

6.2 Derivation of MSAF tests 76

6.3 Experimental Results 77

7 Conclusion and Future Scope **91**

7.1 Thesis summary 91

7.2 Future Scope 93

References **95**

Publications **100**

List of Figures

1.1 Delay trends in sub-micron technology 2
1.2 Fault modeling 2
1.3 A delay fault 4
1.4 Resistive bridges 4
1.5 Resistive opens 5
1.6 Crosstalk effect 5
1.7 Impact of coupling capacitance on victim propagation delay with same arrival times, opposite transition direction, and different load capacitances (Source:ITRS 2007) 6
1.8 Average delay increase of a gate as a result of IR-drop increase (180 nm Cadence generic standard cell library, nominal power supply voltage = 1.8V)(Source:Tehranipoor2012[51]) 6
1.9 Example:Transition Delay Fault 7
1.10 Example:Segment Delay Fault 8
1.11 Example:Path Delay Fault 8
1.12 Classification of path delay faults 9
1.13 Robustly testable path *b-d-e-g* 9
1.14 Non-robustly testable path *b-c-e* 10
1.15 FS criterion for AND gate 10
1.16 Testability of paths 11
1.17 Paths *a-c-d-g* and *b-e-f-g* are untestable for falling transitions 11
1.18 Improving path coverage 12
1.19 Multiple stuck-at faults 13
1.20 Research Overview 14

2.1 Sequential Circuit 17
2.2 Time Frame Expansion 19
2.3 Timeframe expansion example 20

2.4 Full scan design 21
2.5 Partial scan method 21
2.6 Sequential circuit classification 22
2.7 Multi-output acyclic circuit 23
2.8 Transformation to Balanced class 23
2.9 Transformation to internally balanced circuit 23
2.10 Combinational model 24
2.11 Strongly balanced circuit 24
2.12 Implementation of a function using Shannon decomposition 25
2.13 Circuit for Equation 2.1 28
2.14 EP-SOP example 29
2.15 Robust testable by extra control inputs 30
2.16 Delay testable implementation using additional control inputs 31
2.17 Mux based design(not fully delay-testable) 33
2.18 Mux based design with additional input(fully delay testable) 33
2.19 XOR based implementation of [35] 34
2.20 Shared BDD for SOP functions 35
2.21 Delay testable sequential circuit 36
2.22 A model for multiple stuck-at fault [28] 37
2.23 (a) Original circuit, (b) Circuit after normalization procedure 41
2.24 Pseudoexhaustive testing 42
2.25 Hardware partitioning 42

3.1 Binary Decision Diagram for $x_1x_2 + \bar{x}_1\bar{x}_2\bar{x}_3 + \bar{x}_1x_2x_3$ 45
3.2 Reduced Ordered Binary Decision Diagram for $x_1x_2 + \bar{x}_1\bar{x}_2\bar{x}_3 + \bar{x}_1x_2x_3$ . . 46
3.3 ROBDD example for Boolean function f 47
3.4 K-maps for function f 48
3.5 Gate implementation of the formula $f_v = \bar{x}_i \cdot f_v^{\bar{x}_i} + x_i \cdot f_v^{x_i}$ 49
3.6 Sub-circuit for nodes connected to leaf node 1 50
3.7 Sub-circuit for nodes connected to leaf node 0 50
3.8 Example 3.2.1:Improving testability by avoiding simplification 50
3.9 Circuit C corresponding to function f in Fig. 3.3 51

4.1 Stuck-at faults corresponding to delay faults 54
4.2 Robustly testable conditions 56
4.3 Non-robustly testable conditions 57

5.1 Test partitions of circuit in Figure 3.9 60
5.2 Gate implementation of the formula $f_v = \bar{x}_i \cdot f_v^{\bar{x}_i} + x_i \cdot f_v^{x_i}$ 61
5.3 a,b faults 62
5.4 Multiple fault locations at inputs of sub-circuit 63
5.5 K-maps for ROBDD nodes 64
5.6 ROBDD for x_3 65
5.7 Fault propagation on the path 67

6.1 K-maps for function f 72
6.2 Generating v_1 vectors / *b-tests* set for one path 73
6.3 v_1 *vector* / *b-test* set for one DSOP term 73
6.4 Test generation for Example 6.1.1 74
6.5 Example 6.1.2: v_1 *tests* 75
6.6 Example 6.2.2: *b-tests* 77

List of Tables

1.1 Defect coverage of Stuck-at and Delay Fault Models 3

2.1 Redundancy classes in [7] 32
2.2 State Transition Table 35
2.3 Segregation of SOPs 35
2.4 Fault values 38

5.1 Test partitions of Fig. 5.1 60
5.2 ROBDD operation for other test vectors 66
5.3 Fault propagation in presence of P_{s-a-1} 67

6.1 DSOP List 72
6.2 Combinational Circuit Test Generation 79
6.2 Combinational Circuit Test Generation 80
6.2 Combinational Circuit Test Generation 81
6.3 Sequential Circuit Test Generation-Combinational logic 82
6.4 Area, power and path delay comparisons of combinational circuit synthesis 83
6.4 Area, power and path delay comparisons of combinational circuit synthesis 84
6.4 Area, power and path delay comparisons of combinational circuit synthesis 85
6.4 Area, power and path delay comparisons of combinational circuit synthesis 86
6.5 Area, power and path delay comparisons of combinational logic synthesis of sequential benchmark circuits 87
6.5 Area, power and path delay comparisons of combinational logic synthesis of sequential benchmark circuits 88

Chapter 1

Introduction

Current VLSI technology has helped fabricate multiple systems on one chip (SoC) where all functions of a computer are integrated. It is not feasible to test the chips exhaustively for every combination of state and input. Therefore, critical hardware development requires a detailed and rigorous process to assure that the final product is correct and as reliable as possible. This in turn leads to development of hardware design approaches that guarantee complete testability of such devices.

In early days of VLSI circuits, testing was restricted to only proving logical correctness of the fabricated chips. As the technology moved towards sub-micron sizes, expectations of higher and higher speeds soared. Functional correctness of the fabricated chip tested at low frequencies was not enough anymore. New defects emerged for which new fault models were developed. Now when we are in the nanometer age, the greed for speed leads to aggressive *place and route*. This causes increase in device density and an increase in interconnect length. Hence the interconnect delay cannot be ignored as it does not scale with technology. Dominance of interconnect delay with advancement in technology is depicted in Figure 1.1. This also affects the operating speeds of the chips.

Defective chips fail to run at target speeds. They may run at speeds lower than rated speeds. If a chip is tested at the rated clock frequency and fails, it will be tested at a few lower frequencies. The chips that pass are then sorted into bins according to the speeds they were tested at. This is called *speed binning*.What prohibits the chip to run at rated speed but allows to run at lower speeds? The answer is *delay defects*.

The role of *testing* is to detect presence of defects and the role of *diagnosis* is to determine exactly where the defect is, and where the process needs to be altered. This necessitates the testing process to be correct and effective. A well thought out test strategy is crucial to economic realization of products.

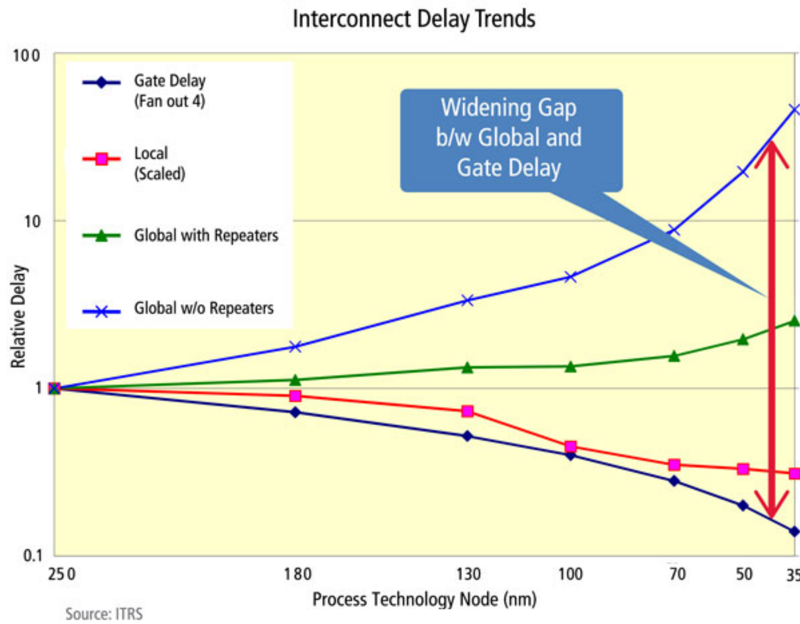


Figure 1.1: Delay trends in sub-micron technology

Real defects on a silicon chip are too numerous and often not analyzable. Fault models represent an approximation of the effects that defects produce on the behavior of the circuit. Effectiveness of the model is measurable by experiments. An ideal model should provide a high confidence of a faulty circuit being detected. The test generation for such a fault model should allow handling of large circuits with minimum computing resources. Figure 1.2 demostrates the abstraction level that a fault model provides and also represents a basic fault model known as the *Stuck-at Fault* model.

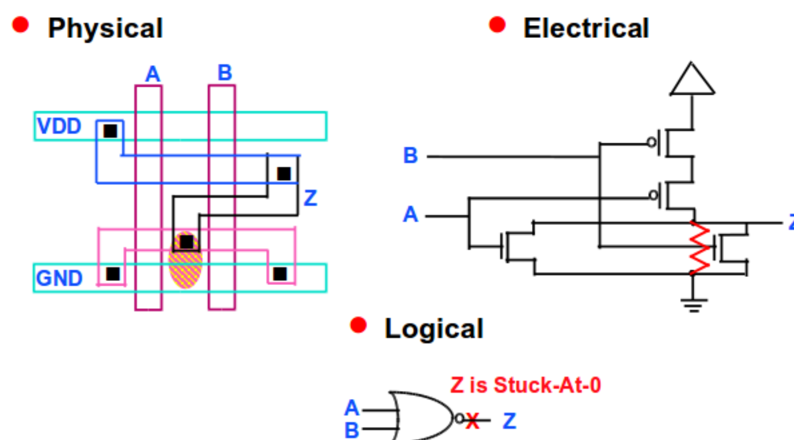


Figure 1.2: Fault modeling

Each connecting line can have two types of faults: stuck-at-1 (s-a-1) and stuck-at-0 (s-a-0). In general, several stuck-at faults can be simultaneously present in the circuit.

A circuit with n lines can have $3^n - 1$ possible stuck at combinations (multiple stuck-at faults). An n-line circuit can have at most $2n$ single stuck-at faults.

The efficacy of a fault model can be measured by the range of defects covered. In the nanometer technology regime, defects like process variations and aging dominate. These defects do not change the logic function of the chip. However, the probability of the chip being exposed to timing violations increases drastically. A single gate delay may be out of specification, but as long as the total delay remains under the specified operating period there is no fault. If the specified operating period is increased then many timing violations will be eliminated. If the specified operating period is reduced, more timing violations are exposed. Integrated circuits in deep sub-micron technologies usually suffer from subtle additional delays due to process variations on both gates and interconnects. Testing for these delay causing defects in addition to other defects is inevitable. A timing violation is commonly defined as a *delay fault*. The fault model that represents the delay defects is called the *delay fault model*. Table 1.1 gives a brief comparison of the defects that are covered by the SSAF and delay models.

Table 1.1: Defect coverage of Stuck-at and Delay Fault Models

Defects covered	**Stuck-at tests**	**Delay tests**
Hard shorts & opens	Definitely	Definitely
Resistive shorts	Maybe	Definitely
Resistive opens & coupling faults	Never	Definitely
Process variations	Never	Definitely
Aging	Never	Definitely

Table 1.1 establishes the need to cover delay faults in addition to stuck-at faults. If a device is intended for safety critical applications, it is pertinent that it should be tested using more than one fault models.

1.1 Delay faults

In a combinational logic, if there exists a logical path from a primary input PI to primary output PO on which the *total propagation time* of a 1-to-0 or 0-to-1 transition *exceeds the specified time period*, then it is a delay fault. The delay fault is graphically demonstrated in Figure 1.3.

To observe delay faults, it is necessary to generate transitions at the circuit input and propagate them to the outputs. This requires application of a pair of vectors (v_1, v_2). The

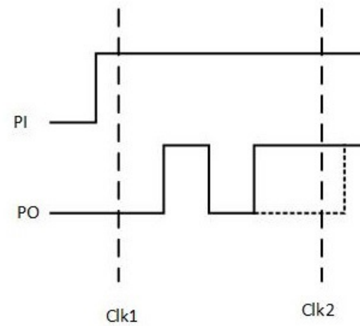


Figure 1.3: A delay fault

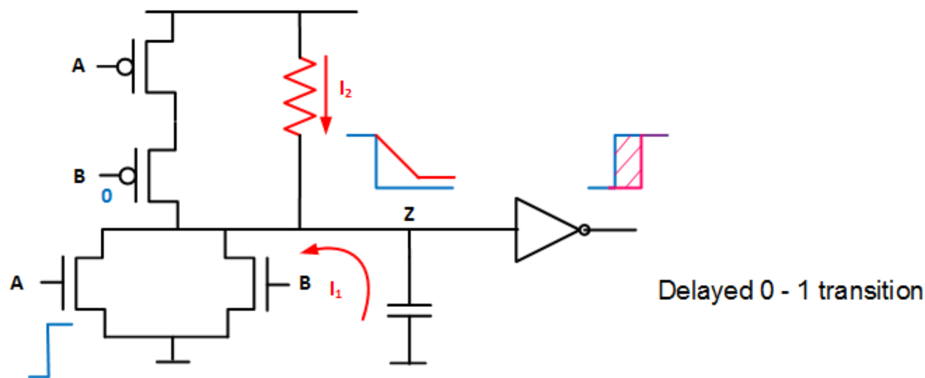


Figure 1.4: Resistive bridges

first vector v_1 stabilizes all signals in the circuit. The second vector v_2 causes the desired transition at the input of the circuit.

1.1.1 Contributors to Delay Faults

There are several contributors to delay defects. Each contributor adds a small delay and hence comprise *small delay defects* or SDDs. Some of the major contributors[31] are:

- Manufacturing Defects
  Certain manufacturing defects do not change the logic function of the chip, but can cause timing violations such as resistive bridges and opens.

  In Figure 1.4, 0-to-1 Transition on A is delayed, but 1-to-0 Transition on A is speeded up. In Figure 1.5 both 0-to-1 and 1-to-0 Transitions On A are delayed.

- Unaccounted capacitive coupling
  Aggresive design rules lead to aggresive *place and route*. Any unaccounted coupling capacitance leads to crosstalk effects that can increase or decrease delays of victim or aggressor nets or both depending on transition direction, transition arrival time etc. Figure 1.6 demonstrates how coupling capacitance has a direct impact on

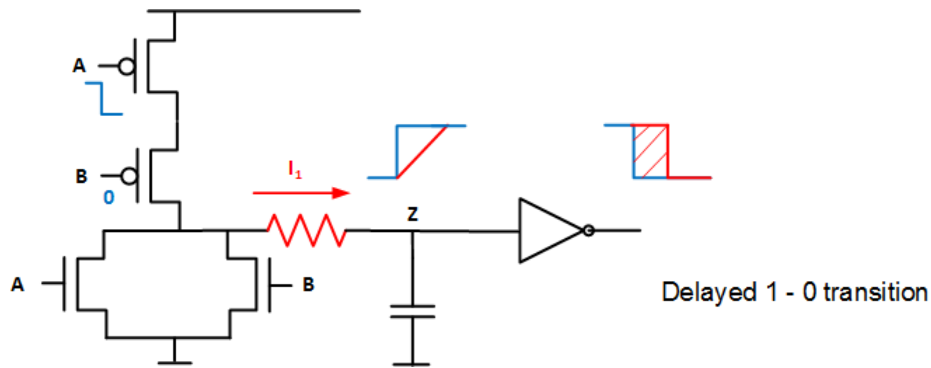


Figure 1.5: Resistive opens

the victim′s delay. It can be seen from Figure 1.7 that for different load capacitance cases, the propagation delay on the victim net increases linearly with the coupling capacitance. Variable $d_{coupling_{arrival}}$ denotes the victim net delay considering the impact of coupling capacitance size and C_{a-v} is the coupling capacitance between the aggressor and victim nets. For the same transition direction case, the crosstalk delay decreases linearly. These delays dynamically add to the chip delays. The effect of coupling capacitors is discussed in Miller[13].

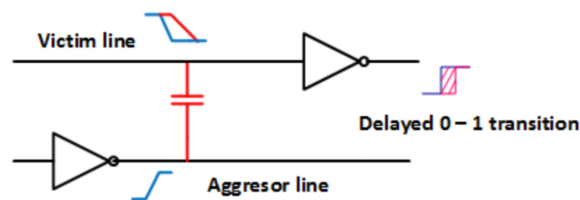


Figure 1.6: Crosstalk effect

- Power supply variations

  The increase in frequency and decrease in the rise/fall transition times in latest designs causes more simultaneous switching activity within a small time interval and further cause increase in current density and voltage drop along power supply nets. This leads to slower transitions.

  The voltage drop on a gate will directly impact its performance. The reduced voltage results in performance degradation or functional failures of the circuit. Figure 1.8 presents the simulation results of an AND gate with different power supply voltages. The output load capacitance of the gate is 0.1 pF. It is seen that with 20% IR-drop (0.36 V), the average gate delay decrease can be approximately 21%. This experiment is based on 180 nm Cadence Generic Standard Cell Library with nominal $V_{dd} = 1.8V$[51],[43].

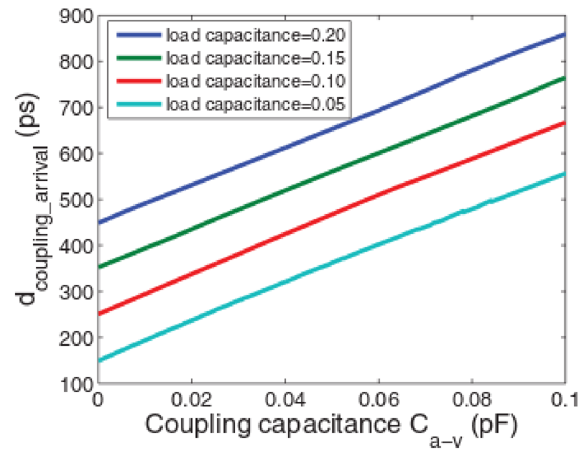


Figure 1.7: Impact of coupling capacitance on victim propagation delay with same arrival times, opposite transition direction, and different load capacitances (Source:ITRS 2007)

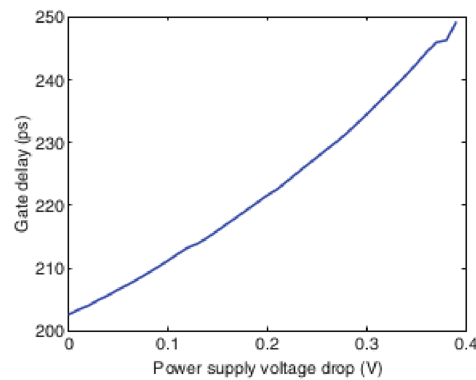


Figure 1.8: Average delay increase of a gate as a result of IR-drop increase (180 nm Cadence generic standard cell library, nominal power supply voltage = 1.8V)(Source:Tehranipoor2012[51])

- Process variations
  Process variations are statistical abnormal variations in geometry after fabrication. For example, if line spacing is affected, *coupling* faults occur. Similarly, if line thickness is affected,*delay* increases and if there are variations in gate threshold level, there is an *increase* in transition time.

- Aging
  After manufacturing, the circuit performance degrades over time due to several aging effects that cause a drift of device and interconnect parameters. These effects must be considered during a timing analysis of an aged circuit. Conservative design that would account for aging is not preferred as it eliminates the advantages of

moving to a smaller technology node. Some of the causes of performance degradation due to aging are: 1) **Metal migration** which causes line thinning leading to increase in interconnect resistance, 2) **Gate oxide degradation** over time which causes *Hot Carrier Injection* which in turn leads to increased device threshold voltages, consequently gate transition time increases, 3)**Negative Bias Temperature Instability**(NBTI), a condition that affects the transconductance of a *pmos* transistor causing a decrease in drain current. This delays low to high transitions.

1.2 Delay Fault Testability

1.2.1 Delay Fault Models

- Transition delay model(TDF):

  The TDF model assumes that a large delay defect is lumped at one logical node, such that any signal transition passing through the node will be delayed past the clock period. In Figure 1.9, the delay is assumed to be lumped at NAND gate

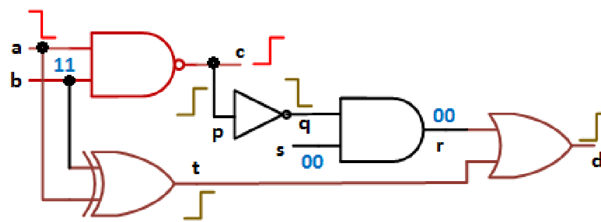


Figure 1.9: Example:Transition Delay Fault

between *a* and *c*. A large delay defect on line a will be detected at pin *c* if delay on path *a*-*c* exceeds specifications. A small delay defect on line a may not make delay of path *a*-*c* large enough to be detected. It should be tested through long path a-d to be detected.

Test generation for TDFs is based SSAF tests. If SSAF tests are available, TDF test generation requires little additional effort. However, TDFs do not detect small delay defects.

- Segment delay fault model:

  The segment delay fault model assumes a distributed delay along a small segment of a long path.

  In Figure 1.10, path *a*-*p*-*q*-*r*-*d* is untestable for falling transition launched at *a*, but part of its segment, namely segment *p*-*q*-*r*-*d* is testable. All the paths that pass

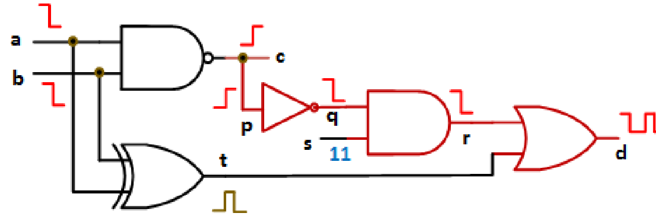


Figure 1.10: Example:Segment Delay Fault

through segment *p-q-r-d* are considered for testing. This indicates that the longest path may not be covered.

- Path delay fault model(PDF):
  The PDF model takes into account distributed delay of the entire path[49]. In Fig-

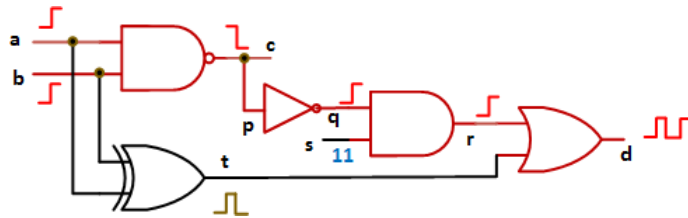


Figure 1.11: Example:Path Delay Fault

  ure 1.11, path *a-p-q-r-d* is tested for rising transition at pin *a*. Small delay defects distributed along the path will be tested if the cumulative delay exceeds specification. For PDFs, test generation is difficult as number of paths may explode for a complex circuit, however, it detects small delay defects.

The *Transition delay* tests may detect many shorts and opens missed by stuck-at tests, may also detect some capacitive coupling faults. However distributed and small delay defects do not get covered. Thus *Transition delay* tests cannot be used to explicitly target critical paths. The *Path Delay Fault (PDF) Model* is best suited for targeting all paths; critical paths included since since all distributed delays are covered.

1.2.2 Path Delay Fault Tests

Path delay faults can be classified according to testability characteristics based on specific fault detection conditions[31][33]. Figure 1.12 illustrates the classification. The robust, non-robust and validatable non-robust, and functional sensitizable faults can affect the performance of the circuit hence they are termed as functional irredundant faults . Functional unsensitizable faults, also called functional redundant faults, can never independently determine the performance and they cannot be tested.

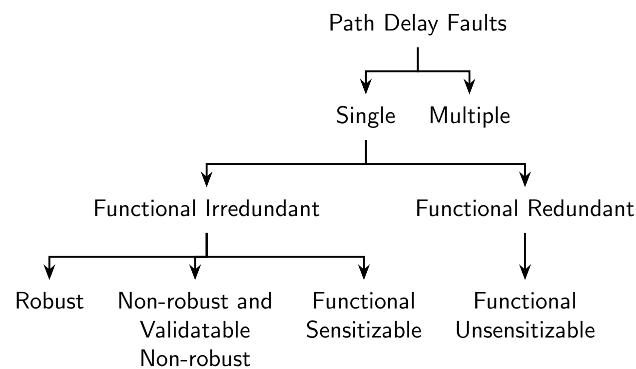


Figure 1.12: Classification of path delay faults

Delays of falling transition and rising transition along the same path from a primary input to a primary output may be different. In the general case, a pair of vectors (v_1, v_2) for each kind of transitions of a path is necessary. In accordance with the conditions of fault manifestation, singly testable PDFs are divided into robust testable faults, non-robust testable faults and functional sensitizable faults:

- *A PDF is robust testable if there is a test pair on which a fault manifestation does not depend on delays of other circuit paths.*

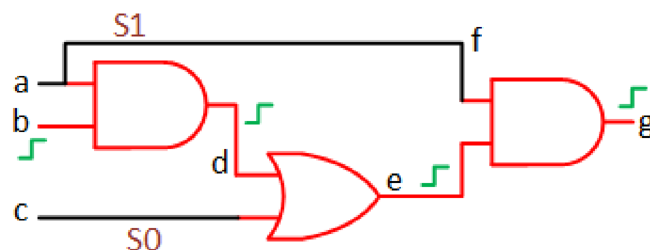


Figure 1.13: Robustly testable path *b-d-e-g*

In Figure 1.13, paths *a-f* and *c* can be independently driven to static values under v_1 and v_2. This allows the transition on b to be propagated over path *b-d-e-g*. Thus path b-d-e-g is a robustly testable path, independent of delay on paths a-f and c. In most robust testable cases, *OFF* paths have stable non-controlling values under v_1 and v_2. If only robust tests are considered then the PDF coverage is low due to these strict requirements.

- *A PDF is non-robust testable if fault detection is possible only when all OFF paths of the circuit are fault-free.*

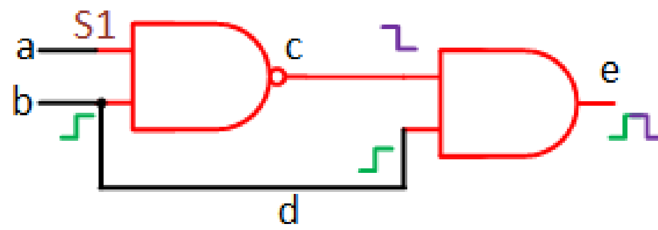


Figure 1.14: Non-robustly testable path *b-c-e*

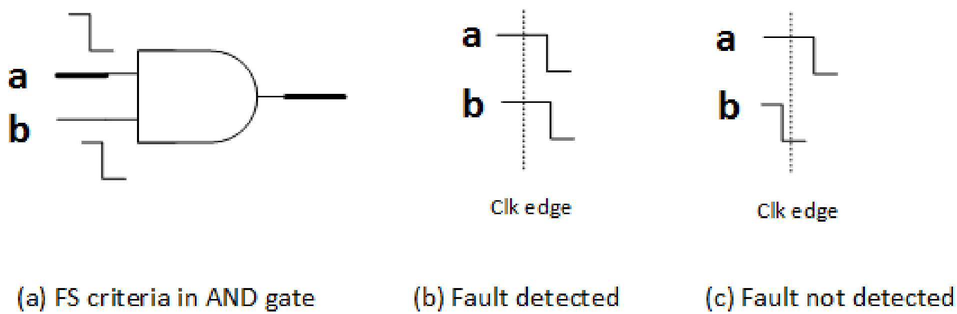


Figure 1.15: FS criterion for AND gate

In Figure 1.14, a rising transition at location b travels via paths *b-c* and *b-d*. If path *b-c-e* is to be tested, delays on path *b-d* cannot be ignored. If path *b-d* is fault free, then the rising transition arrives at *e* first. If path *b-c-e* is faulty then the falling transition will arrive at e delayed. Since the polarities of the transitions are opposite, a glitch-like signal appears at *e*. If this signal is detected at the clock-edge then it will be detected as a faulty output and will be marked as a delay fault. Thus *b-c-e* will be detected as faulty only if path *b-d* is fault free. The example given here leads to a conclusion that if the stringent conditions of the *OFF* paths are relaxed, it is still possible to detect a fault on a path, provided that the *OFF* paths settle at non-controlling values under v_2.

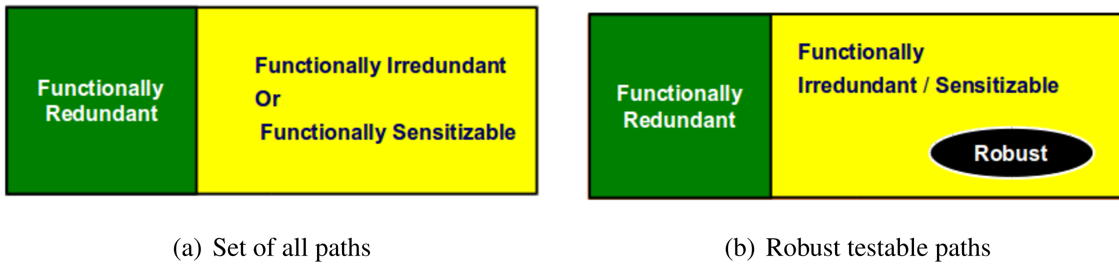
- *The functional sensitization criterion requires that there exists more than one faulty path in the circuit in order for the target fault to be detected.*
  For functional sensitizable PDFs, the detection of faults depends on the delays on signals outside the target path. Figure 1.15 illustrates the functional sensitizable criterion for an AND gate. When the ON-input *a* and OFF-input *b* both have falling transitions, for the fault to be detected the output of the AND gate both transitions have to be late. This is because the arrival time of the signal at the output is determined by the earlier of the two falling transitions. If the OFF path *b* is not delayed then the fault on path a will not be detected. Thus for a functional sensitizable fault

detection on a path, both ON and OFF paths have to transition from non-controlling value to controlling value and both have to be delayed.

In this work, we consider robustly and non-robustly testable PDFs.

While testing a path non-robustly, if all the *OFF* paths are robustly tested and found fault-free then the non-robust test of the *ON* path will not be invalidated. In this case the target path delay fault is *validatable non-robust path delay fault*. It is as good as a robust test.

Let us consider a set of paths for a circuit represented in Figure 1.16(a). Each circuit has functionally redundant(functionally sensitizable) and functionally irredundant(functionally unsensitizable) paths. Given the stringent conditions of robust testability (Figure 1.13), the ratio of paths that can be tested robustly is very low as shown in Figure 1.16(b)[25].



(a) Set of all paths (b) Robust testable paths

Figure 1.16: Testability of paths

The number of paths in a circuit are in general very large. In the worst case scenario, ATPG may overtest sequential redundant paths or may abort in case of an irredundant path. Consider a NAND implementation of XOR gate shown in Figure 1.17.

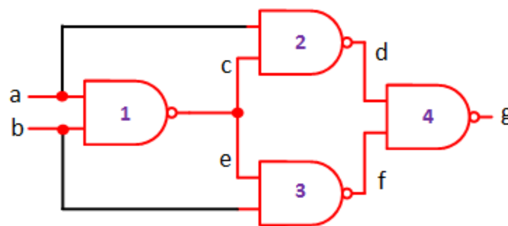


Figure 1.17: Paths *a-c-d-g* and *b-e-f-g* are untestable for falling transitions

Paths *a-c-d-g* and *b-e-f-g* can never be tested for falling transitions, are redundant. So this implementation of the XOR gate is to be avoided.

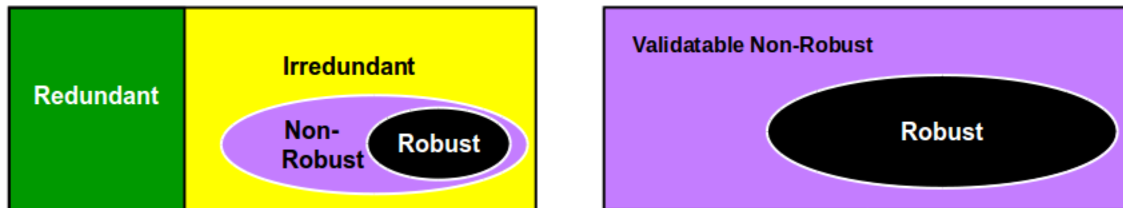
If non-robust tests are accomodated then there is marginal increment in the path coverage as depicted in Figure 1.18(a). Redundant paths still exist and ATPG may waste time deriving tests for them. To overcome the redundant path issue, addition of muxes

[41] or gates[48] on the dependent OFF paths have been proposed. This allows better control of OFF paths. The disadvantages of these approaches are:

- ATPG may take long time to declare that no tests exist for a particular path.
- This is a post design-phase procedure, hence any circuit changes will have to be followed by timing analysis.
- Additional control inputs are needed as in [8],[41]-[15]. This would result in additional input pins.

This brings us to the motive of our research. A synthesis which demonstrates

- **No redundant paths**
- **All irredundant paths testable using either robust tests or non-robust tests, Figure 1.18(b)**
- **Ease of test generation**



(a) Including non-robust tests

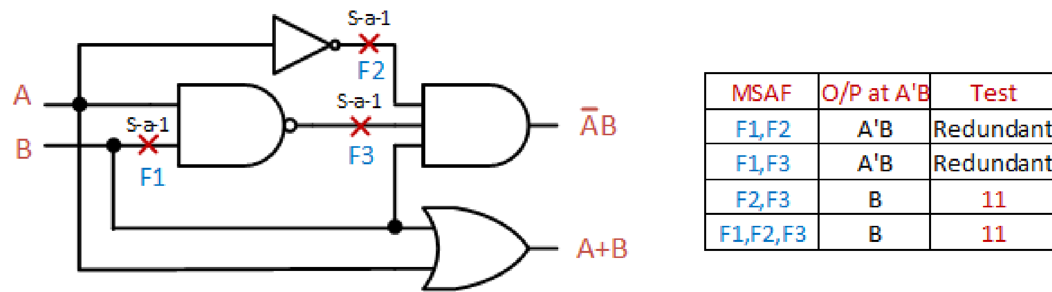
(b) Paths in a testable synthesis

Figure 1.18: Improving path coverage

1.3 Multiple Stuck-at Fault Testability

Detection of Multiple stuck-at faults(MSAFs) in a circuit gives increased confidence of defect coverage. The multiple stuck-at fault(MSAF) model covers many stuck-open and bridging defects. Many locations simultaneously are to be considered having stuck-at faults. A circuit with n-nets can have $3^n - 1$ multiple stuck-at faults. Single stuck-at faults test set do not cover the entire MSAF list since simultaneous presence of faults at multiple locations may mask each other.

Consider the circuit in Fig.1.19. If F_1 and F_2 exist simultaneously, no test exists that can detect F_1 in presence of F_2 and vice versa, indicating that they both mask each other. Similar is the case for F_1 and F_3 when existing simultaneously. But multiple fault combinations (F_2, F_3) or (F_1, F_2, F_3) are detectable using $AB = 11$.



MSAF	O/P at A'B	Test
F1,F2	A'B	Redundant
F1,F3	A'B	Redundant
F2,F3	B	11
F1,F2,F3	B	11

Figure 1.19: Multiple stuck-at faults

Many approaches exist that consider MSAF detection using varied angles. The guarantee of irredundant paths in the circuit synthesis motivates to consider multiple fault detection of smaller sections of the circuit since all sections are reachable.

1.4 Thesis Organisation

The thesis is organised as follows:

Chapter 2 discusses challenges in testing. The test generation complexities of sequential circuits are discssed. The motivation to consider testablity of combinational circuits is established. The need to cover PDFs and MSAFs in logic circuits is already discussed in the current chapter. Chapter 2 presents two approaches available in literature that increase the testability in logic circuits. One is developing algorithms for test generation and other is hardware modification.

The ROBDD based hardware synthesis approach proposed by us is explained in Chapter 3. Circuit transformations and steps for hardware implementation are described.

The proposed ROBDD based synthesis is analysed for path delay faults in Chapter 4. Beginning from single stuck-at fault testability, conditions for robust and validatable non-robust path delay tests are established.

Chapter 5 demonstrates the Multiple Stuck-at Fault testability of the synthesis approach. Pseudoexhaustive test methodolgy is applied on one sub-circuit and conditions for complete MSAF testability are established.

The algorithms to derive test vectors for PDFs and MSAFs are developed in Chapter 6. Test vector calculations for a few combinational and sequential benchmark circuits are listed in this chapter. Area, power and maximum path delays before and after BDD based synthesis are comapred.

Chapter 7 concludes the thesis and gives direction for future work.

Thesis Contribution

The main contribution of this thesis is an ROBDD based circuit synthesis. Circuit transformations suggested for each node ensure that no additional control inputs are required and there are no redundant paths in the circuit. We have shown that all paths are testable for distributed path delays with either robust tests or validatable non-robust tests. Since the synthesis is structured, we have been able to prove complete multiple stuck-at fault testability for the circuit.

Other contribution of the thesis is the development of test generation algorithms for PDF and MSAF faults. The advantage of these algorithms is that they facilitate test vector generation at design time and have polynomial test generation complexity. The test vector pairs for PDFs and tests for MSAFs comprise of primary inputs only. Figure 1.20 depicts the research overview of the thesis.

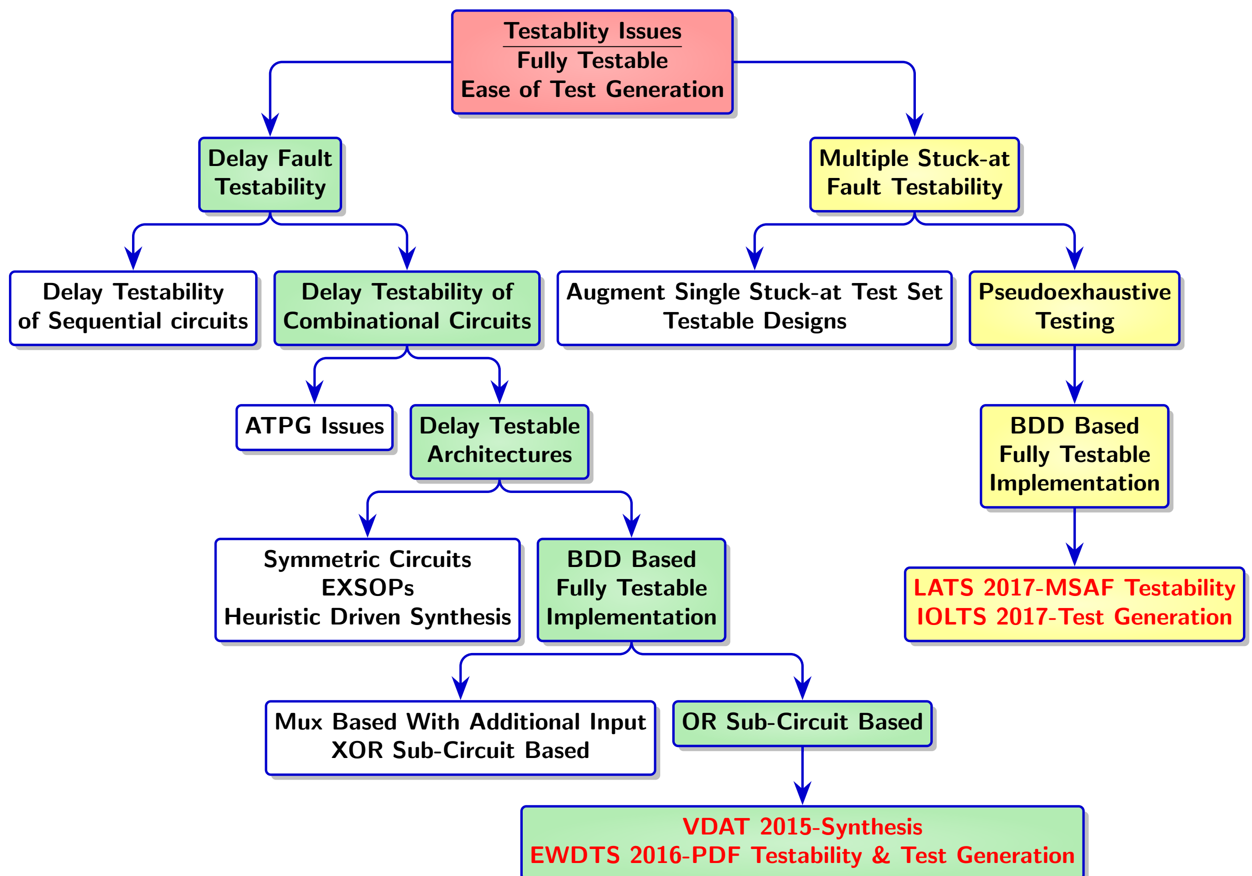


Figure 1.20: Research Overview

Chapter summary

This chapter presents the motivation for research on testable synthesis. The reasons for considering path delay and multiple stuck-at fault models are discussed. The next chapter presents challenges in sequential and combinational circuit testing. Existing designs having varied degrees of testability under PDF and MSAF models are discussed.

Chapter 2

Test Challenges and Prior Work

Most of the VLSI chips used for computational purposes encompass the digital domain wherein sequential circuits dominate. These sequential circuits comprise of sequential components i.e., flip-flops and combinational logic. As discussed in the last chapter, each chip has to pass through the test process. This is required to maintain the quality and economy.

Testing sequential circuits has its challenges. In the upcoming section we discuss the issues with sequential circuit testing and then moving to testable combinational circuits. Existing approaches to testable synthesis are examined in the next section.

2.1 Sequential Circuit Testing

A sequential circuit comprises of memory elements (flip-flops) and logic to generate output and next state. Figure 2.1 denotes the general sequential circuit construction.

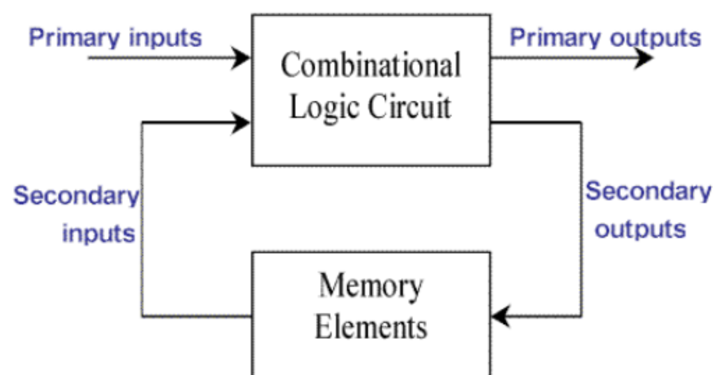


Figure 2.1: Sequential Circuit

The following issues increase the difficulty level of sequential circuit testing

1. Unknown initial values of FFs
2. Propagation of FF values to POs
3. Long ordered test sequences for 1 fault
   (a) Initialize FFs
   (b) Apply combinational test
   (c) Propagate value of FFs to POs

Thus the test for a fault may be a sequence of several vectors that must be applied in the specified order. In comparison, a single vector can detect a fault in combinational circuit. The computational complexity of combinational ATPG is NP-complete (NP-complete means that no polynomial expression for the compute time function is found and the problem is presumed to have exponential complexity). But ATPG algorithms continuously evolve due to improving heuristic algorithms and procedures. This allows the complexity of combinational fault simulation to be $O(n^r)$(n is the size of the circuit and r is a constant > 2) is allowing combinational ATPGs to find test vectors in polynomial time. The worst case ATPG computational complexity of a sequential circuit is $O(n \times 2^{no.pi} \times 4^{no.ff})$; n is the size of the circuit, pi are the primary inputs and ff are the flip-flops present in the circuit. If the faulty and fault-free circuits are not initialized, then the complexity is $O(n \times 2^{no.pi} \times 9^{no.ff})$, for 9 valued logic. This necessitates design approaches where sequential circuits have combinational test generation complexity. Here are some methods that help transform sequential circuits for the purpose of achieveing combinational test generation complexity.

2.1.1 Testing Sequential Circuits with Combinational ATPG

1. **Time Frame Expansion**: In this approach, the sequential circuit is "unrolled" into a larger combinational circuit. If the test sequence for a single stuck-at fault contains n vectors,
   - Replicate combinational block n times
   - Place fault in each block
   - Generate a test for the multiple stuck-at fault using combinational ATPG with 9-valued logic.

   Thus the test vector has to detect multiple faults. The expansion is done till the fault is propagated to any of the POs. The presence of fault in all time-frames adds

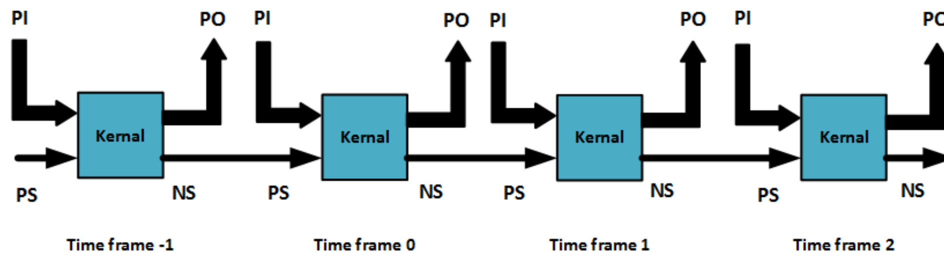


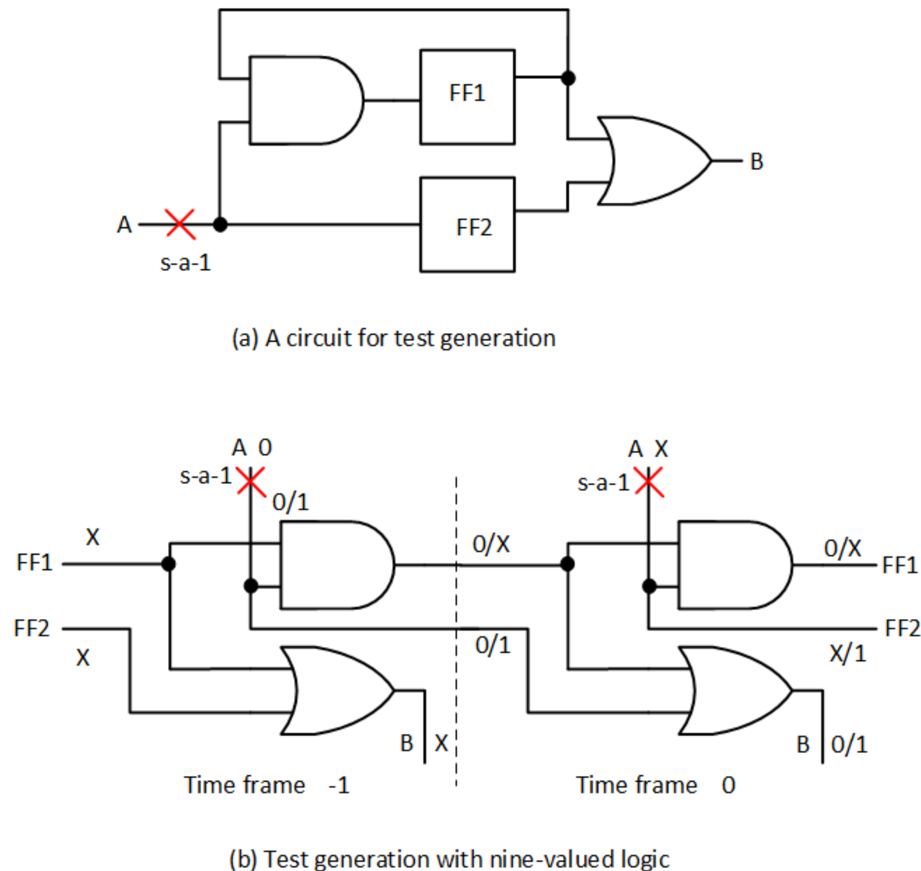
Figure 2.2: Time Frame Expansion

tremendous complexity to the test generation process. An approximate method is described in Bushnell et. al.[11]. Here, only the time-frame of fault activation is assumed to have the fault. The circuit states in all time-frames, previous to the time frame in which fault is detected, have to be justified for both fault-free and faulty circuits. This is achieved by incorporating a *universal reset* primary input which initializes all flip-flops correctly in both fault-free and faulty circuits. Thus, a combinational test can be generated in a single time frame. If the fault can only be detected at a flip-flop input, then "future" time-frames are added without the fault and the fault effect is propagated through them to a primary output. This is shown in the example of Figure 2.3. There is a limit of PIs and POs that a combinational ATPG can handle thus the maximum length of this expansion is limited. This method is inapplicable to circuits with large number of FFs in the feedback path.

2. **Full scan method**: The aim is to have a circuit with combinational test generation complexity. This can be achieved by scanning all the flip-flops in the feedback path such as shown in Figure 2.4. A combinational ATPG is used to generate the test vectors for the combinational part. The test vectors are applied by combination of PI values and scanned values in the FFs. The responses are latched in the scan FFs to be scanned out.

   This method seems to be most appealing but if the FFs are in millions the scan chain would grow larger. It would take millions of cycles to apply just one vector. Also one test cycle would test only certain stuck at faults. Delay test vectors are applied in pairs and at rated speed. To incorporate delay testing, additional FFs/latches need to be added at each location in the scan chain[11]. This additional hardware increases the critical path length. Application of one test pair requires two scan cycles causing test time to increase tremendously.

3. **Partial scan**: In the partial scan method only a partial number of FFs are scanned as shown in Figure 2.5. This has two advantages viz. :



(a) A circuit for test generation

(b) Test generation with nine-valued logic

Figure 2.3: Timeframe expansion example

(a) The number of FFs in the scan chain are reduced

(b) The complexity of the sequential circuit is reduced allowing test generation to be done by a sequential (Time frame expansion based) ATPG

Despite the advantages, the test length and ATPG run time for such circuits can be quite large. A sequential ATPG program can achieve fault coverage in excess of 95% when about 25 to 50% of the FFs are scanned.

In either of the methods discussed earlier, all or some FFs have to be modified to incorporate scan design. Increase in FF delay may increase delay of the critical path. A non-scan design offers the following advantages :

- Reduction in area overhead of FFs
- Reduction in FF delay
- Use of Combinational ATPG
- Complete Fault Efficiency

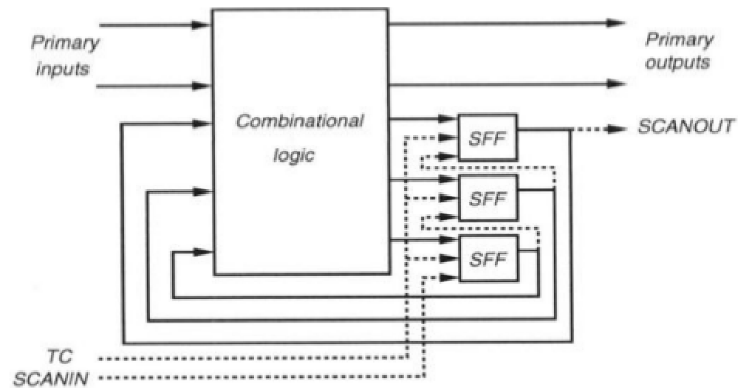


Figure 2.4: Full scan design

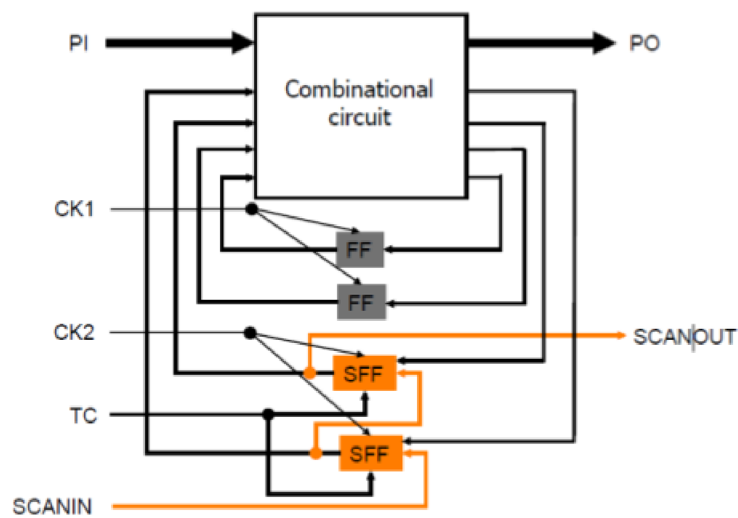


Figure 2.5: Partial scan method

- Shorter test application time
- At speed testing

One of the non-scan methods for testing sequential circuits is to identify a class of sequential circuits that have combinational test generation complexity. Certain classes of acyclic circuits fall into this category.

2.1.2 Acyclic circuits

A clocked synchronous circuit is called acyclic or cycle-free if the flip-flops contain no feedback. An acyclic circuit is known to be initializable and the test sequence length of any detectable single fault has an upper bound of $d_{max} + 1$, where d_{max} is the sequential depth defined as the number of FFs on the longest path. A cyclic circuit can be converted to acyclic by scanning a minimum set of FFs, known as minimum feedback

vertex set (MVFS). A combinational ATPG model can be obtained by replacing all FFs by wires. Literature [11] shows that for a SAF in this model, the test vector found by the combinational ATPG, if repeated d_{max} times, will detect the fault. Many faults are still undetected and sequential ATPG is required to cover the faults. Additionally, the issue of detecting multiple faults lingers even if the logic is selectively duplicated. (Multiple faults may mask each other and the fault may not propagate to the output at all). Not all acyclic circuits have combinational test generation complexity. Acyclic sequential circuits having combinational test generation complexity can be classified as:

1. Balanced circuits - Gupta et.al [21]

2. Internally balanced circuits - Fujiwara [20],[19]

3. Strongly balanced circuits - Balakrishnan and Chakradhar [6]

The classification of acyclic sequentiall circuits is shown graphically in Figure 2.6.

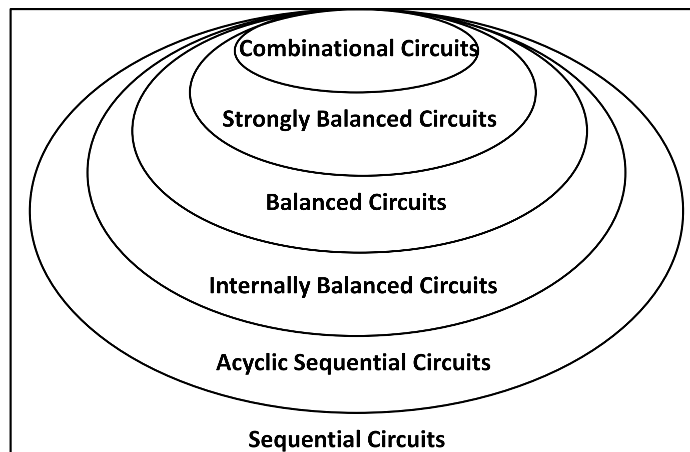


Figure 2.6: Sequential circuit classification

Consider a multi-output acyclic circuit[27] shown in Fig 2.7. The classes of the sequential circuits with combinational test generation complexity will be described with this circuit as an example.

1. **Balanced sequential circuits**: In a balanced circuit, all signal paths between any two nodes (inputs, outputs, gates and flip-flops) have the same number of FFs. Any acyclic circuit can be converted into a balanced circuit by removing some FFs via partial-scan. Replacing FFs with buffers or wires gives a combinational model that can produce tests for all detectable faults.

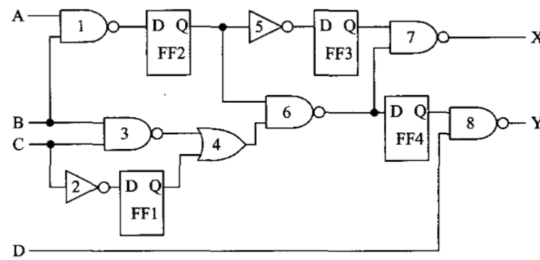


Figure 2.7: Multi-output acyclic circuit

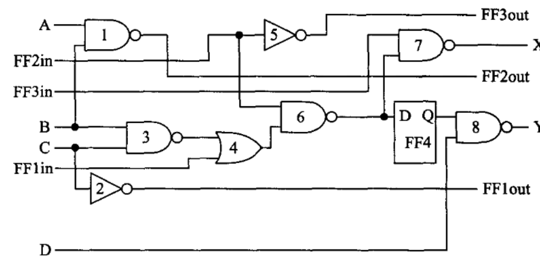


Figure 2.8: Transformation to Balanced class

Advantage: Though it has time expansion to $d_{max} + 1$, similar to acyclic, there are no duplicates in the time expanded circuit. In Fig. 2.7 we see that the paths from A,B & C to output X have different number of FFs. To convert to a balanced circuit described earlier FF1, FF2 and FF3 have been scanned. Thus we see in Fig.2.8 that from any input to one output, the number of FFs is the same.

2. **Internally balanced circuits**: In this case all node pairs excluding those involving the PIs are balanced, ref. Figure 2.9. PIs with unbalance are split as additional PIs. A combinational model can now be derived by replacing FFs with wires or buffers, ref. Figure 2.10. A combinational ATPG is now used to generate the test vectors. This test vector is converted to a test seq. of length $d_{max} + 1$. The bits split from original PIs are appropriately placed in the sequence for application to the corresponding original PI. Bits of unsplit PI are duplicated in each vector. This method requires lower partial scan overhead than *balanced* circuits.

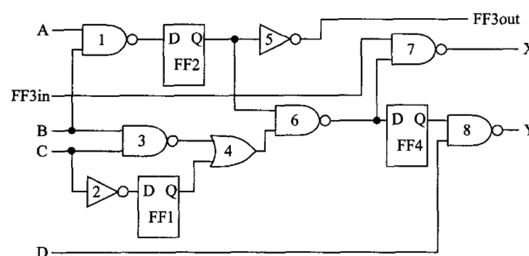


Figure 2.9: Transformation to internally balanced circuit

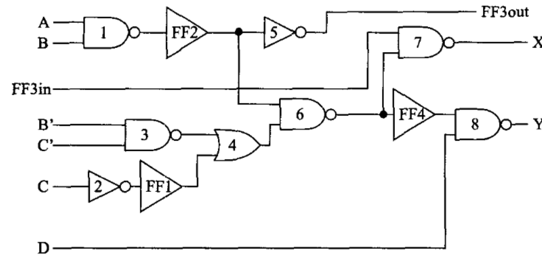


Figure 2.10: Combinational model

3. **Strongly balanced circuits**: A strongly balanced circuit is balanced and in addition, all paths between a node and all reachable PIs in its fan-in cone have the same sequential depth, ref. Figure 2.11. This allows the combinational test vectors to be pipelined through the sequential circuit without repeating any vector except the last one. The last one is repeated d_{max} + 1 times. The advantage of this approach is that the test vectors are compact. Adversely more number of FFs needs to be scanned as compared to internally balanced and balanced structures. Many algorithms are available in literature that aid in the transformations discussed earlier. But in any case it boils down to the testability of the resulting combinational circuit. Thus it is important to discuss design of delay testable combinational logic.

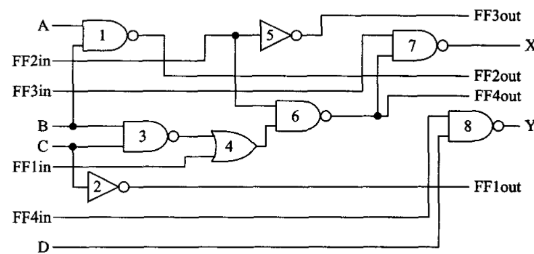


Figure 2.11: Strongly balanced circuit

2.2 Literature Survey

2.2.1 Existing Approaches to Delay Testable Combinational Circuit Synthesis

A path is *functionally testable* if a functional test exists for it, else it is a *functionally untestable* path. A *functionally untestable* path is not responsible for the performance of the circuit but a *functionally testable* path is. Path delay faults on functionally testable paths degrade the performance. In this scenario and those mentioned in Section 1.2.1

and 2.1.2, the most appropriate synthesis is the one that guarantees that all paths are *functionally testable*.

If the circuit is designed in such a way that the delay testability of all paths is guaranteed then the disadvantages discussed in Section 1.2.1 can be overcome. Here are some approaches to combinational circuit synthesis that offer full PDF coverage.

Shannon Expansion

The first work to ensure complete hazard-free robust fault coverage of all path delay faults in a combinational circuit was based on Shannon′s decomposition theorem[30] where any function f can be represented as

$$f(x_1, x_2, \cdots x_n) = x_i \cdot f_{x_i} + \bar{x}_i \cdot f_{\bar{x}_i}$$

Where f_{x_i} and $f_{\bar{x}_i}$ are co-factors of the function f with respect to variable x_i and are obtained by making $x_i = 1$ and $x_i = 0$, respectively, in f. The corresponding decomposed circuit is shown in Figure 2.12.

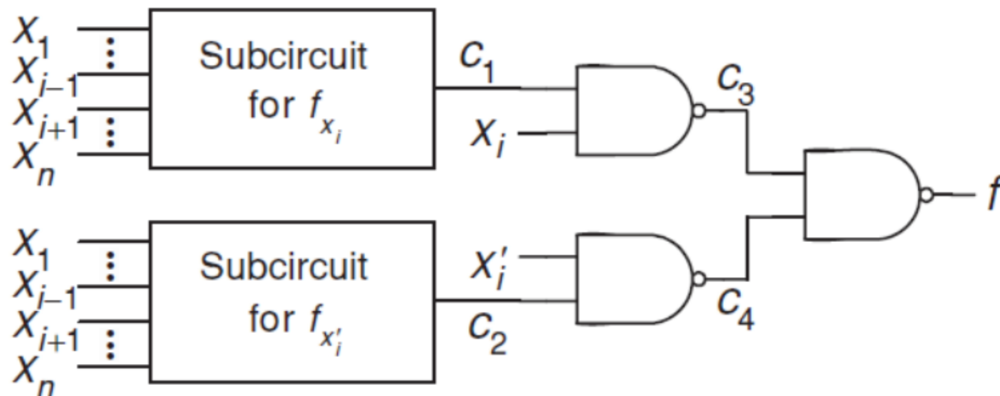


Figure 2.12: Implementation of a function using Shannon decomposition

This type of implementation is appreciable only if the function f is binate in x_i . If this is the case and if the sub-circuits of the two co-factors are robustly testable then the decomposed circuit will be robustly testable. This type of decomposition is fruitful since one can always find at least one vector which, when applied to $x_1, x_2, \cdots, x_{i1}, x_{i+1} \cdots x_n$, results in $f_{x_i} = 1$ and $f_{\bar{x}_i} = 0$ ($f_{x_i} = 0$ and $f_{\bar{x}_i} = 1$) allowing the path $x_i C_3 f (\bar{x}_i C_4 f)$ to be robustly tested. By making $x_i = 1 (x_i = 0)$ the sub-circuit for $f_{x_i} (f_{\bar{x}_i})$ can be fully tested by applying the robust tests to the corresponding sub-circuits. It may be possible that the sub-circuit is not robustly testable after one level of decomposition. The above given method can be applied to the co-factors recursively until a robustly testable design is achieved. After max $n - 2$ Shannon decompositions, two variable co-factors are obtained which are

guaranteed to be robustly testable. Thus this method guarantees termination in a robustly testable circuit.

Another way of terminating is when a co-factor is unate in all its variables since robust tests can be found for each path in such a sub-circuit in which the initialization and test vectors differ in just the literal being tested. After the process of decomposition, if the sub-circuit is found to be robustly testable, even if some of the variables are binate, further decomposition can be avoided. Furthermore, the sharing of logic among the cofactor sub-circuits does not compromise the robust testability property. Suggested heuristics for determination of variables to be chosen first:

- Variables appearing most of the times in complemented or un-complemented form
- Variables causing robust untestability in maximum number of gates

Example 2.2.1 *: Consider the two-level circuit based on the expression*

$$f_2 = x_1x_3 + x_1x_2 + \bar{x}_1\bar{x}_2 + x_3x_4 + \bar{x}_3\bar{x}_4$$

The only robustly untestable literals are x_1 *and* x_3 *in the first AND gate. Using one of the above two heuristics, we choose either* x_1 *or* x_3 *. If we choose* x_1 *, we obtain the decomposition*

$$f_2 = x_1(x_2 + x_3 + \bar{x}_4) + \bar{x}_1(\bar{x}_2 + x_3x_4 + \bar{x}_3\bar{x}_4)$$

Since the two cofactors are robustly testable, the decomposed circuit for f_2 *is also robustly testable.*

Issues and possible solutions:

- This method would be fruitful for constructing multilevel circuits if the variables chosen for decomposition occur frequently and the SOPs and are binate otherwise redundancy will increase.
- It is mentioned earlier that recursive decomposition is applied to increase the levels. This would reduce the area but there is no guarantee that the sub-circuit will continue to be robustly testable. As mentioned earlier two level implementations of irredundant functions may not be robustly testable.

Symmetric circuits

A *symmetric* function is defined as a switching function of *n* variables which is invariant under any permutation of the variables. It is given in Menon et. al [26] that that a two-level irredundant circuit realizing a minimum cover of a consecutive symmetric function

$S^n(a_l - a_r)$, is robustly testable for all path-delay faults if and only if 1)$a_l = 0$, or $a_r = n$, or 2) $\binom{n}{a_l} = \binom{n}{a_r}$. If the above condition is not satisfied, the number of untestable paths in the circuit becomes at least $\left[\left\{\binom{n}{a_l} \sim \binom{n}{a_r}\right\} + 1\right] \min(a_l, n - a_r)$.

The proposed method of synthesizing $S^n(a_l - a_r)$ shown in Chakraborty et. al[12], Rahaman et. al[42], for full (100%) robust path-delay fault testability is based on the following three key observations:

- $S^n(a_l - a_r)$ is unate if $a_l = 0$ or $a_r = n$;
- A two-level irredundant AND-OR realization of a unate symmetric function is fully robustly delay testable. If $a_l = a_r$, then each minterm will be a prime implicant by itself, and a minimum two-level realization will be 100% robust testable since $\binom{n}{a_l} = \binom{n}{a_r}$.
- Every consecutive symmetric function $S^n(a_l - a_r)$, $a_l \neq a_r$ where $a_l \neq 0, a_r \neq n$, can be expressed as a logical composition (e.g., AND, NOR) of two consecutive symmetric functions which are unate, and the resulting composite circuit realizing $S^n(a_l - a_r)$ is 100% robustly delay fault testable.

The following theorem represents the proposed composition technique. Theorem : $S^n(a_l - a_r), a_l \neq a_r, l < r$, can be expressed as a composition of two unate and consecutive symmetric functions as follows:

1. $S^n(a_l - a_r) = S^n(a_l - a_n) \cdot \overline{S^n(a_{r+1} - a_n)}$
2. $S^n(a_l - a_r) = \overline{S^n(a_0 - a_{l-1}) + S^n(a_{r+1} - a_n)}$
3. $S^n(a_l - a_r) = S^n(0 - a_r) \cdot \overline{S^n(0 - a_{l-1})}$
4. $S^n(a_l - a_r) = S^n(0 - a_r) \cdot S^n(a_l - a_n)$

Example 2.2.2 : *An irredundant two-level realization of $S^6(4,5)$ is not robustly delay testable. If this is implemented as per case 1 of the proposed technique of synthesis-for-testability, it is given as*

$$S^6(4,5) = S^6(4,5,6)\overline{S^6(6)}$$

. The minimum s-o-p expression of $S^6(4,5,6)$ and $S^6(6)$ are given by

$$\begin{aligned} S^6(4,5,6) = & x_1x_2x_4x_5 + x_1x_3x_4x_5 + x_1x_2x_4x_6 + x_1x_3x_4x_6 \\ & + x_1x_2x_3x_5 + x_1x_2x_3x_4 + x_2x_4x_5x_6 + x_2x_3x_4x_5 + x_2x_3x_4x_6 \\ & + x_1x_2x_5x_6 + x_1x_3x_5x_6 + x_1x_4x_5x_6 + x_3x_4x_5x_6 + x_1x_2x_3x_6 + x_2x_3x_5x_6 \end{aligned} \tag{2.1}$$

And $S^6(6) = x_1x_2x_3x_4x_5x_6$.

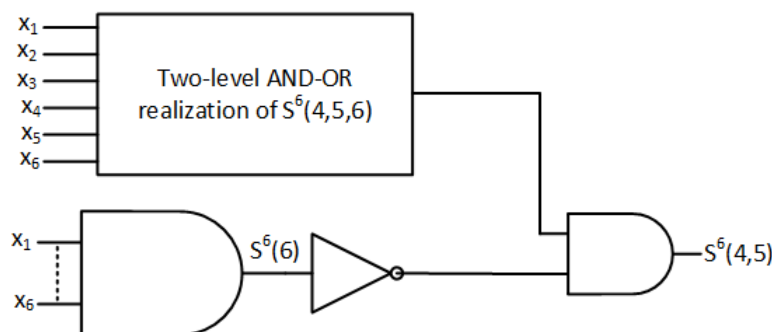


Figure 2.13: Circuit for Equation 2.1

The circuit in Figure 2.13 will now be robustly testable and can be implemented using basic two level circuits as described in Chakraborty et.al[12]. Rahman et al.[42] have proposed modified DTSL and shuffle exchange based approach to demonstrate complete path delay fault testability for symmetric circuits. A simple universal cellular module that admits a recursive structure is designed for synthesizing unate symmetric functions. General symmetric functions are then realized following a unate decomposition method. This design guarantees complete and robust path delay testability for *Unate Symmetric* functions.

The challange here is that all general circuits cannot be transformed to symmetric circuits. Also symmetric circuits constitute a small fraction of all digital circuits.

EXOR Projected Sum-of-products

Shannon decomposition is the basis of synthesis approach of BDD circuits. A Boolean function f is represented in terms of its two sub-functions derived by substituting the value 0 and the value 1 to one of the input variables, say x_i. The Boolean function f is projected onto two complementary subspaces: the space where $x_i = 0$, and the space where $x_i = 1$. This decomposition eliminates one input variable. But the Hamming distances among the cofactors of f do not change; thus, if f is represented as a minimal SOP, its projections onto the two smaller subspaces cannot be further minimized. Alternatively a different kind of Shannon projections, based on the use of EXOR gates is called the projected sum-of-products (PSOP) approach proposed by Bernasconi et al. in [8],[9]. These projections, in addition to eliminating one variable, due to the presence of EXOR gates, reduce the Hamming distances among the factors appearing in each subspace, so that further merges are possible, even when the initial projected SOP for f is minimal. The EP-SOP expressions can be derived starting from a SOP representation ϕ of a Boolean function f in two steps.

Step 1: Project ϕ onto the two subspaces $(x_i \oplus x_j)$ and $(x_i \oplus x_{\bar{j}})$, and we obtain the following expression:

$$\xi_{ij} = (x_i \oplus x_j)\phi_{\oplus} + (x_i \oplus x_{\bar{j}})\phi_{\bar{\oplus}}$$

After the projection minimize the two SOPs $\phi_{\oplus}$ and $\phi_{\bar{\oplus}}$ in order to minimize the EP-SOP ξ_{ij}.

Step 2: The Minimal (i,j)-EP-SOP of f is the expression

$$\xi_{ij} = (x_i \oplus x_j)\phi_{\oplus}^{(min)} + (x_i \oplus x_{\bar{j}})\phi_{\bar{\oplus}}^{(min)}$$

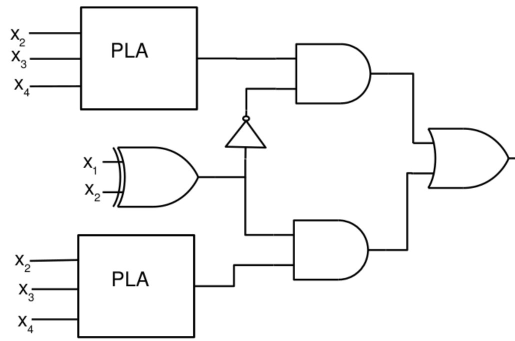


Figure 2.14: EP-SOP example

Example 2.2.3 *: Let a Boolean function ϕ be:*

$$\phi = \bar{x}_1 x_2 \bar{x}_3 + x_1 \bar{x}_2 \bar{x}_3 + x_1 x_2 x_3 + \bar{x}_1 \bar{x}_2 x_3 + x_3 \bar{x}_4$$

Let ϕ be projected onto the spaces $(x_1 \oplus x_2)$ and $(x_1 \oplus x_{\bar{2}})$ The first product p in ϕ contains both x_1 and x_2, since x_1 is complemented and x_2 is not complemented we project p onto the space $(x_1 \oplus x_2)$. The projected product is $x_2 x_3$. The unique crossing product of ϕ is $x_3 \bar{x}_4$ since it does not contain x_1 and x_2. This product will be inserted in both the spaces without any literal removal. The overall projection will return the form:

$$(x_1 \oplus x_2)(\bar{x}_2 x_3 + x_2 x_3 + x_3 \bar{x}_4) + (x_1 \oplus x_{\bar{2}})(x_2 \bar{x}_3 + \bar{x}_2 \bar{x}_3 + x_3 \bar{x}_4)$$

The implementation is shown in Figure 2.14.

In this approach if the original SOPs are PDF testable then the transformed circuit is also PDF testable. Thus while performing minimization, approaches that give a PDF testable design should be considered. Example 2.2.3 is 1EP-SOP since only one variable is considered for projection. The design as well as testing complexity increases when more variables are considered for projection[9]. Every additional varaible chosen for projection leads to insertion of extra test control variables.

Delay Testable Design Using Funtional properties

A delay testable synthesis method is proposed in Mitra et. al[39] which taps into functional properties of Boolean expressions and suggests testability preserving transformations.

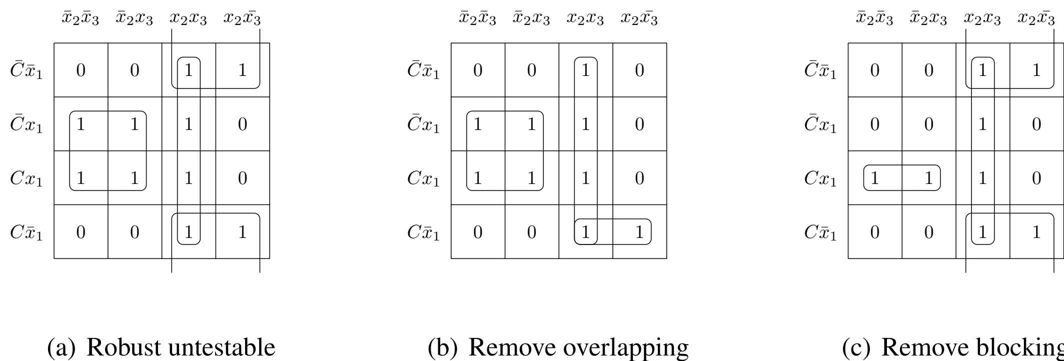
Given the boolean cubes of a function, first, two-level robust PDF testable circuit is designed by properly grouping the cubes. Then some testability-preserving algebraic factorization techniques are used design to multilevel circuits.

This synthesis approach targets improving robust path delay testability. The first step is to find which paths are not robustly testable. This information is obtained from the K-map. Implicants which have variables overlapping may inhibit robust path delay testability. After indentification, extra control variables are introduced in such a way that they do not hinder the function and at he same time aid in removing the overlap of the variables, thus allowing robust path delay testability. Following is an example of the approach.

Example 2.2.4 *: For a function* $f(x_1, x_2, x_3) = \sum(0, 2, 3, 4, 5, 7)$*, the cube* x_2x_3 *is a blocked cube with respect to literal* x_2*, as it has an overlapped vertex (011) with another cube* $\bar{x}_1x_2$*, and its relatively essential vertex(rev)(111) is blocked by vertex (101) of cube* $x_1\bar{x}_2$*. Here none of the paths would be robustly testable.*

Consider a 4-variable function $f = (C, x_1, x_2, x_3)$ *with one additional control input C as shown in Figure 2.15(a). The original function is restored when C is set to logic 0. The cube* x_2x_3 *is still blocked with respect to literal* x_2*, as it has overlapping vertices* $(C, x_1, x_2, x_3) = (0011, 1011)$ *with cube* $\bar{x}_1x_2$*, and its revs (0111, 1111) are blocked by vertices (0101, 1101) of cube* $x_1\bar{x}_2$*. Thus, the circuit is still robust untestable.*

However, if the overlapping cube $\bar{x}_1x_2$ *is replaced by* $C\bar{x}_1x_2$*, the K-map becomes as shown in Figure 2.15(b). In this case, the normal function is restored by setting C to logic*



(a) Robust untestable (b) Remove overlapping (c) Remove blocking

Figure 2.15: Robust testable by extra control inputs

1. The vertex (0011) now becomes a rev of x_2x_3, that has no blocking vertex with respect to the literal x_2 . Hence, the circuit becomes robust testable.

The main idea behind the above technique is to remove the overlapping cube. However, robust testability can also be achieved by removing the blocking cube. The vertices (0101, 1101) of the cube $x_1\bar{x}_2$ block (0111, 1111) of the cube x_2x_3 in Figure 2.15(a). By ANDing the blocking cube with C, we obtain a free rev (0111) of the cube x_2x_3 which was blocked earlier. Therefore, the circuit of Figure 2.15(c) now becomes robust testable.

An algorithm is demonstrated in the proposal wherein, control inputs are added wherever necessary. Figure 2.16 shows the delay testable circuit. The drawbacks of this approach are: i) Additional control inputs are required ii) In multilevel implementation, the network is required to be unate.

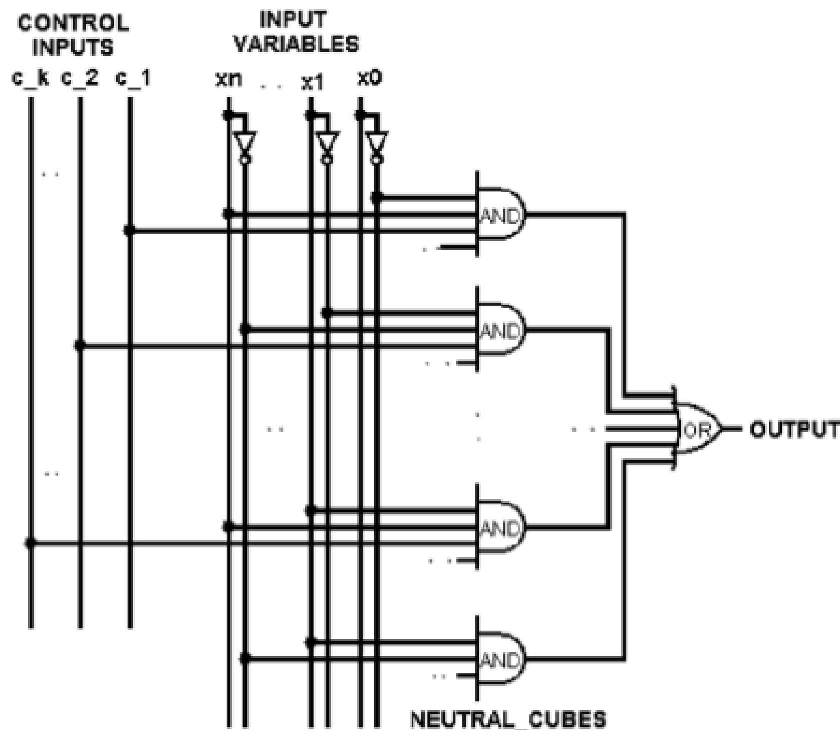


Figure 2.16: Delay testable implementation using additional control inputs

BDD approach

One of the methods to represent Boolean functions are BDDs or *Binary Decision Diagrams*[10]. They are based on binary trees and are discussed in detail in Chapter 3.

Because of the modular structure, ROBDDs (*Reduced Ordered Binary Decision Diagrams*) can be easily implemented with multiplexers. Not all mux based implementations are fully delay testable [7],[4]-[5]. One of the earliest works is discussed here.

BDD based testable synthesis was proposed by Becker in [7]. In this work most of the nodes are replaced by 2x1 multiplexers. Depending upon what combinations can be applied to the mux inputs, six redundancy classes are listed. The testability depends on whether 01 or 10 or both are applicable to the mux inputs or not.

Red.class	Possible mux inputs			
1	00	01	10	11
2		01	10	11
3	00	01	10	
4		01	10	
5	00	01		11
6	00		10	11

Table 2.1: Redundancy classes in [7]

We see from Table 2.1 that in classes 5 and 6, values 10 and 01 are not applicable to the mux inputs. This creates redundant paths in the circuit. In order to eliminate this redundancy, circuit transformations are proposed for nodes under classes 5 and 6. It is proved that for nodes that fall under classes 1-4, all paths are robustly testable and for those nodes only, complete single stuck-at fault coverage implies complete multiple stuck-at fault coverage. XOR gates are used for nodes under class 4. The challenge here is that internal circuit transformations have the probability of creating redundancies in the rest of the circuit. This work concentrates on robust testability of paths. Thus all paths in the circuit are not testable due to existence of nodes under class 5 and 6. The circuit transformations aim at removal of redundancies may have a large probalility of creating secondary redundancies. This drawback is overcome in the work by Dreschler[15].

Consider the circuit in Figure 2.17. The AND gate with one of the input tied to 0 can never be tested. An additional switch *t* and an inverter is added in [15] as shown in Figure 2.18. The input *t* is flipped to generate robust testable conditions for nodes that fall under class 5 and 6. If *t* is set to constant 0, the circuit computes the original function. If *t* is set to 1, the complement is computed. By changing the value of *t* , all *internal* signals, i.e., signals corresponding to edges in the ROBDD, change their value. This can be seen

as follows:

$$\begin{aligned}
\bar{f} &= \overline{\bar{x}_i g + x_i h} \\
&= (\overline{\bar{x}_i g})(\overline{x_i h}) \\
&= (x_i + \bar{g})(\bar{x}_i + \bar{h}) \\
&= \bar{x}_i \bar{g} + x_i \bar{h}
\end{aligned}$$

No circuit transformations are done for any of the internal nodes. Addition of the additional control input t to the mux based circuit guarantees 100% testability for both single stuck-at fault (SAF) and PDF models. The size of a circuit is proportional to the given ROBDD size. The major disadvantage of the approach in [15] is the use of additional input (Figure 2.18) which may lead to an increase in the number of the chip pins. Also when t is flipped, the output is complemented. Test generation procedure has to take this into account.

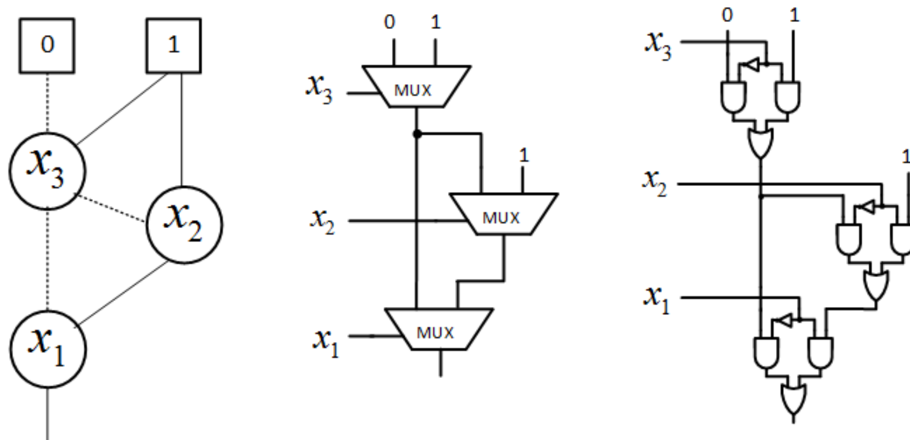


Figure 2.17: Mux based design(not fully delay-testable)

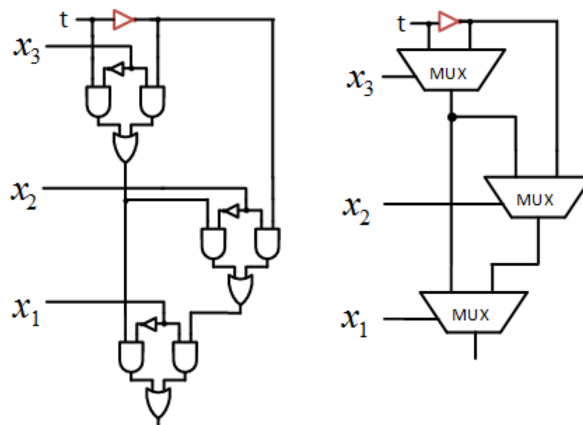


Figure 2.18: Mux based design with additional input(fully delay testable)

In an alternate approach suggested by Matrosova et al. in [35], circuits are constructed from ROBDDs by special sub-circuit implementation using Shannon expansion:

$$f_v = \bar{x}_i \cdot f_v^{x_i=0} \oplus x_i \cdot f_v^{x_i=1} \tag{2.2}$$

Equation (2.2) represents an internal node v of the ROBDD. In this formula the operation $\oplus$ is realized by XOR gate. It is proved that each path delay fault of the resulted circuit manifests itself as robust testable fault. When applying the test pairs in the definite order we can detect any PDF of the circuit. Thus this synthesis technique guarantees 100% testability of the corresponding circuits for PDFs without an additional input, Figure 2.19. The disadvantage here is that the path lengths increase considerably due to presence of XOR gates which are is non-standard gates. The worst case delay through the XOR gate will be a three-gate-delay. The path length is dependent on the number of inputs as they decide the maximum ROBDD node depth. As the number of inputs increase, the path length increases by number of inputs x 3 gate delays.

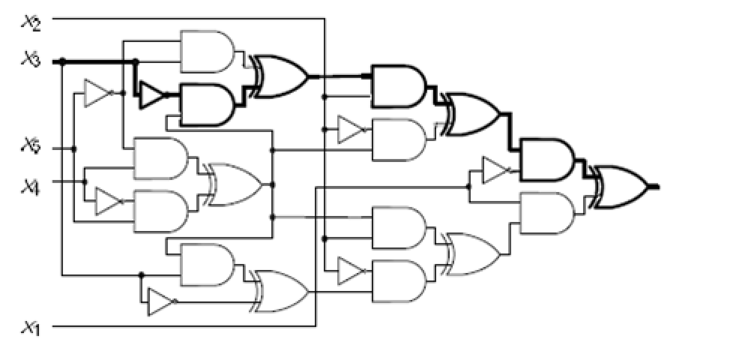


Figure 2.19: XOR based implementation of [35]

As an extension of this work, the authors of [35] have combined the ROBDD based implementation of the combinational part of a sequential circuit with its next state logic to construct a delay testable implementation for the *Finite State Machine* output [34]. Here the FSM states are encoded with (2,4) codes. Table 2.2 indicates a state transition diagram with state variables and output variables.

Delay fault of a path is detected on the outputs y_1, y_2, y_3, y_4. The circuit of one output y_1 is constructed as follows.

$$y_1 = z_1z_2(\bar{x}_1x_3 + x_1x_2x_3) + z_2z_3(x_2\bar{x}_3 + \bar{x}_1x_3) + z_3z_4(x_1\bar{x}_2x_3 + x_2x_3) + z_1z_4(x_2x_3)$$

Individual input variable SOPs are segregated and a shared ROBDD is made for them, ref. Figure 2.20. The complete delay testable circuit for y_1 is given in Figure 2.21.

Table 2.2: State Transition Table

Primary Inputs			Present State				Next State				Outputs				
x_1	x_2	x_3	z_1	z_2	z_3	z_4	z_1	z_2	z_3	z_4	y_1	y_2	y_3	y_4	y_5
0	-	1	1	1	-	-	1	1	0	0	1	0	0	1	0
1	0	-	1	1	-	-	1	1	0	0	0	0	0	1	0
1	1	1	1	1	-	-	0	1	1	0	1	0	0	1	0
-	1	0	-	1	1	-	0	1	1	0	1	0	1	1	0
0	-	1	-	1	1	-	0	0	1	1	1	0	1	1	0
1	0	1	-	-	1	1	0	0	1	1	1	1	0	0	0
0	-	0	-	-	1	1	1	0	0	1	0	1	0	0	0
-	1	1	-	-	1	1	1	0	0	1	1	1	0	0	0
1	-	0	1	-	-	1	1	0	0	1	0	1	0	0	1
-	1	1	1	-	-	1	1	1	0	0	1	1	0	0	1

Table 2.3: Segregation of SOPs

q	Input variable SOPs
1	$\bar{x}_1 x_3 + x_1 x_2 x_3$
2	$x_2 \bar{x}_3 + \bar{x}_1 x_3$
3	$x_1 \bar{x}_2 x_3 + x_2 x_3$
4	$x_2 x_3$

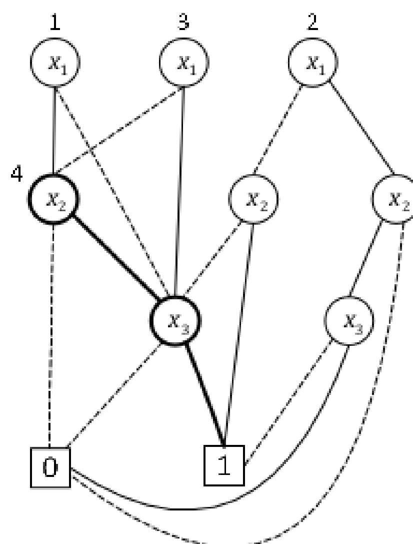


Figure 2.20: Shared BDD for SOP functions

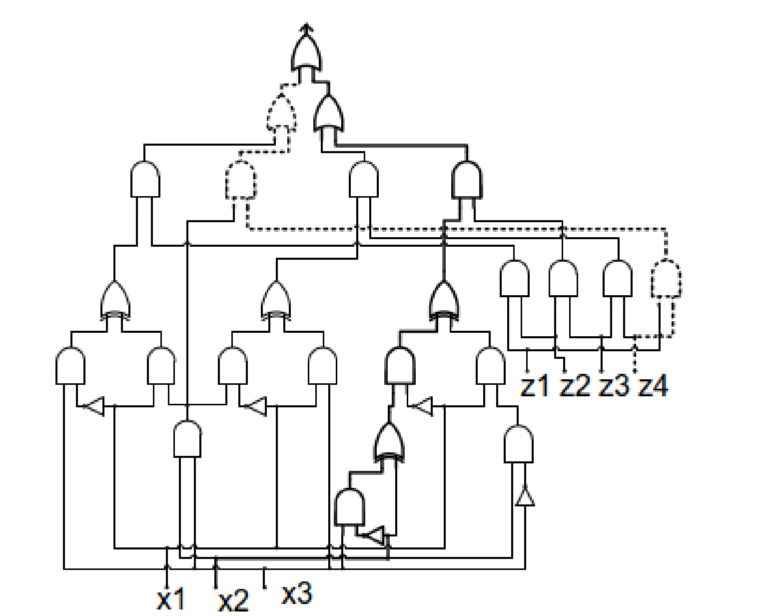


Figure 2.21: Delay testable sequential circuit

In the current work, circuits are constructed from ROBDDs by covering each internal node v with the sub-circuit implementing the simpler Shannon expansion:

$$f_v = \bar{x}_i \cdot f_v^{x_i=0} + x_i \cdot f_v^{x_i=1} \qquad (2.3)$$

In this formula the operation + is realized by OR gate. The path delay faults manifest themselves either as robust testable or validatable non-robust testable ones. It means that this synthesis approach too guarantees 100% testability for essential PDFs without an additional input. Long path lengths due to XOR gates are avoided. Another advantage the OR gate based sub-circuit has over XOR gate based sub-circuits is that most of the nodes can be implemented using 2x1 muxes. For the *gscl45nm* library, a mux would occupy an area of $3.754\mu m^2$ where as the XOR based sub-circuit would occupy $8.446\mu m^2$ of area.

2.2.2 Existing Approaches to Multiple Stuck-at Fault Testability

The single stuck-at fault model is widely used, so the SSAF test set is readily available. Some of the works, [50]-[23] focus on mapping SSAF tests to evaluate MSAF tests. This takes minimum test generation effort but may not cover all MSAFs.

Alternatively, MSAF testability issue has been approached in two ways:

1. MSAF testable synthesis

2. Develop efficient algorithms for test generation

Here is a list of approaches that work on test generation to improve MSAF coverage:

- **Model MSAFs as SSAFs**- Kim et al.[28]. In this work, procedure is provided to model any multiple-stuck at fault as a single stuck-at fault. The method requires insertion of $n + 3$ modeling gates where n represents the multiplicity of the targeted fault. The modeling approach is demonstrated in Figure 2.22. This allows multiple faults detection by a single stuck-at fault ATPG. As n increases, complexity of test generation increases. The disadvantage here is that the fault coverage for multiple faults is very less.

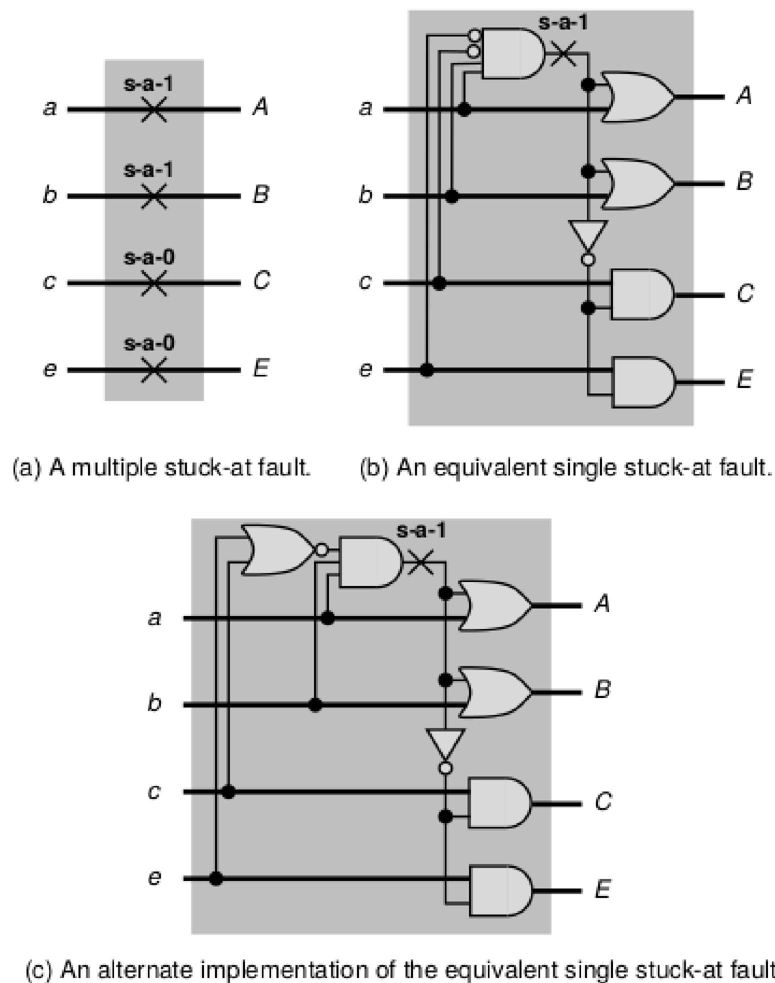


Figure 2.22: A model for multiple stuck-at fault [28]

- **SAT based approach**-Fujita et al.[17]: In this work ATPG steps are combined with fault simulation steps as a set of pure incremental SAT problems. For n faulty locations, stuck-at 0 and stuck-at 1 faults are modeled using two additional variables (x, y) for each faulty location. The fault values are indicated in Table 2.4. Now a formula for ATPG is defined. *in* is the set of primary inputs and *out* is the set of primary outputs. $NoFault(in)$ and $Faulty(in, xy)$ are the logic functions realised at

Table 2.4: Fault values

x_i	y_i	Fault
0	0	Fault free
0/1	1	S-a-1
1	0	S-a-0

the outputs by the circuit without and with additional circuits for stuck-at faulty behaviour. The ATPG process for a fault is formulated as the following SAT problem.

$$\exists in, xy \cdot Faulty(in, xy) \neq NoFault(in) \tag{2.4}$$

This is a normal SAT problem and indicates that some fault can be detected by some input vector as under that input vector, the two circuits behave differently. The solution values of variables, (in, xy) are marked as (in_1, xy_1) respectively. This means that the fault corresponding to xy_1 can be detected by in_1. Both xy_1 and in_1 are explicit representations of the fault and its corresponding test vector.

In traditional ATPG process, fault simulators are used for the input vector in_1, in order to eliminate the detectable faults from the target remaining faults. This approach is not appropriate for detecting multiple faults as the possible fault combinations can never be manipulated explicitly. In order to eliminate detected faults *implicitly* rather than explicitly, another SAT problem is formulated.

$$\exists xy \cdot Faulty(xy, in_1) \neq NoFault(in_1) \tag{2.5}$$

Where in_1 is one of the solutions for Equation (2.4). All the faults corresponding to the values of xy, whhich are the solution of SAT problem Equation (2.5), can be detected by the test vector in_1, as under that test vector, $Faulty$ and $NoFault$ behave differently. In order to eliminate the detected faults by the test vector in_1, the following constrant is added to Equation (2.4):

$$Faulty(xy, in_1) = NoFault(in_1)$$

This constrains xy to be the ones which behave correctly with test vector in_1, i.e., undectable under in_1. This works as a way to eliminate all detectable faults from the target faults in the next iteration.

The next step of the ATPG is to solve the following SAT problem:

$$\begin{aligned} \exists xy, in \cdot (Faulty(in, xy) \neq NoFault(in)) \\ \wedge (Faulty(xy, in_1) = NoFault(in_1)) \end{aligned} \tag{2.6}$$

where in_1 is the solution of Equation (2.4). If the solution values of variables (xy, in) for Equation (2.6) are (xy_2, in_2) respectively then in_2 becomes the second test vector. It detects some faults which are not detected by in_1. This is repeated till there is no more solution. It is assumed that the following SAT problem has a solution

$$\begin{aligned}
&\exists xy, in \cdot (Faulty(in, xy) \neq NoFault(in)) \\
&\quad \wedge (Faulty(xy, in_1) = NoFault(in_1)) \\
&\quad \wedge (Faulty(xy, in_2) = NoFault(in_2)) \wedge \cdots \\
&\quad \wedge (Faulty(xy, in_{n-1}) = NoFault(in_{n-1}))
\end{aligned} \tag{2.7}$$

But the following SAT problem has no solution

$$\begin{aligned}
&\exists xy, in \cdot (Faulty(in, xy) \neq NoFault(in)) \\
&\quad \wedge (Faulty(xy, in_1) = NoFault(in_1)) \\
&\quad \wedge (Faulty(xy, in_2) = NoFault(in_2)) \wedge \cdots \\
&\quad \wedge (Faulty(xy, in_{n-1}) = NoFault(in_{n-1})) \\
&\quad \wedge (Faulty(xy, in_n) = NoFault(in_n))
\end{aligned} \tag{2.8}$$

As Equation (2.7) has a solution and Equation (2.8) does not have a solution, the test vectors $in_1, in_2, \cdots in_n$ can detect all of the detectable faults, as the unsatisfiablity of Equation (2.8) guarantees that there is no more detectable fault. So they become a set of complete test vectors for all combinations of multiple stuck-at faults which are detectable. Advancement of this algorithm that works on transformed circuits too is described in Fujita et.al[18]. Another work that targets multiple stuck-at faults via detection of double stuck-at faults is presented in Moore et.al[40]. The premise is that, given some double stuck-at(DSA) fault $\{f_1, f_2\}$, as long as f_2 does not interfere with the propagation of f_1, f_1 will be detected, and therefore $\{f_1, f_2\}$ will also be detected. The method focuses on putting path constraints on the circuit such that a fault is guaranteed to propagate to a primary output. Once a test pattern is found for some fault and the constraints are in place, any other fault which may interfere with the constraints can easily be identified and dealt with.

- **Augment SSAF tests to cover MSAFs**-Agrawal et al.[2]: This work proposes an ATPG method wherein all MSAF tests of a logic circuit are generated using two complementary algorithms. The first algorithm finds pairs of input vectors to detect the occurrence of target single stuck-at faults independent of the occurrence of other faults. The second uses a sophisticated branch and bound procedure to complete the test set generation on the faults undetected by the first algorithm.

The disadvantage of both these proposals is that they have exponential test generation complexity.

Jacob et al.[24] have suggested a measure to give lower bound on multiple faults that can be detected using single stuck-at test sets based on *Gauranteed To Be Detected* or GTBD faults. A fault is GTBD if its detection is not masked by the presence of any other fault or faults. If one of the faults of a multiple fault combination is GTBD then the MSAF is also GTBD. In a circuit with n lines, if the number of GTBD faults are k, then the lower bound on MSAF coverage is derived as:

$$C_{comb} \geq 1 - \frac{1}{3^k} \tag{2.9}$$

This requires classifying single stuck-at faults into GTBD and non-GTBD faults which in turn would require MSAF simulation. It is not possible to simulate all combinations of MSAFs. Additionally, since the coverage depends only on k and not on n, the coverage indicated by this measure can be very misleading. Nevertheless, the concept of GTBD faults helps in easier test generation.

Next we look at existing MSAF testable design approaches:

- **Fan-out free circuits**-Hayes et al.[22]: This work suggests a network transformation called “*Normalization*" wherein superflous lines and gates are eliminated and only NAND gates are used (Ref. Figure 2.23). This method allows detection of all multiple faults of the entire circuit.

- **Irredundant two-level circuits**-Kohavi et al.[29]: This seminal work proves that an experiment designed to detect all multiple faults on external inputs of an irredundant two-level AND-OR network, detects all the faults within the network as well. Thus an irredundant two-level AND-OR implementation is testable for multiple stuck-at faults. A compact test generation method is also suggested.

- **Internal fan-out free circuits**-Schertz et al.[44]: This work suggests a fault class structure which results from application of fault collapsing techniques which results in a division of circuit into fan-out free segments. The detection of all single faults within a restricted connected set implies detection of all multiple faults within that connected set.

- **BDD implementations**: Mux based BDD implementations of [7] indicate complete MSAF testability for nodes that can be tested robustly for path delay faults. Mux implementation of [15] is also testable for all MSAFs but with the help of an additional control signal. XOR based BDD implementation of [35] is also testable for MSAFs but presence of XOR gates creates long path lengths.

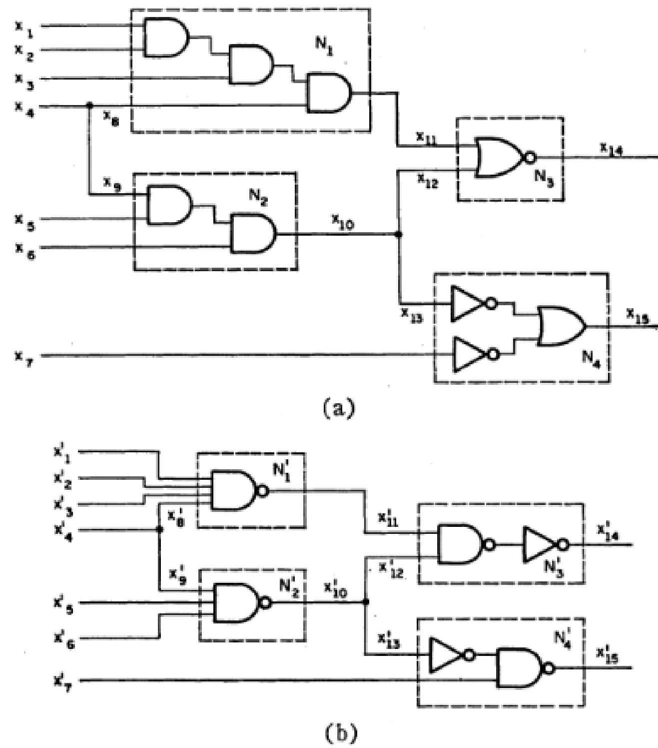


Figure 2.23: (a) Original circuit, (b) Circuit after normalization procedure

In this work we investigate the MSAF testability of mux based BDD implementation which does not have an additional control signal. Concepts of *Pseudoexhaustive testing* and MSAF testability of two-level AND-OR circuits are used for the analysis.

Exhaustive testing of a circuit-under-test would cover complex faults including MSAFs and bridging faults, however it is impractical. A practical approach called *Pseudoexhaustive testing* was proposed by McCluskey et al. in [37],[36]. Their proposal was aimed at *Built-In Self-Test* architectures. The premise of pseudoexhaustive testing is as follows.

If it is possible to apply all 2^n patterns to a circuit with n inputs, test generation is eliminated. It is unnecessary to calculate fault coverage since the entire truth table is verified. No faults would remain undetected. The major hurdle is this approach is that most of the circuits have large number of inputs causing long test times. The work proposes an approach where each circuit output is traced back to determine actual number of inputs in its fan-in cone and apply only those inputs exhaustively. This is represented in Figure 2.24. The work also suggests partition methods so that each partition output function depends on a sufficiently small number of input variables.

Two methods to perform partitioning are proposed: *Hardware partitioning* and *sensitized partitioning*. Access to the embedded inputs and outputs of the sub-circuit under test can be achieved by inserting multiplexers and connecting the embedded inputs and

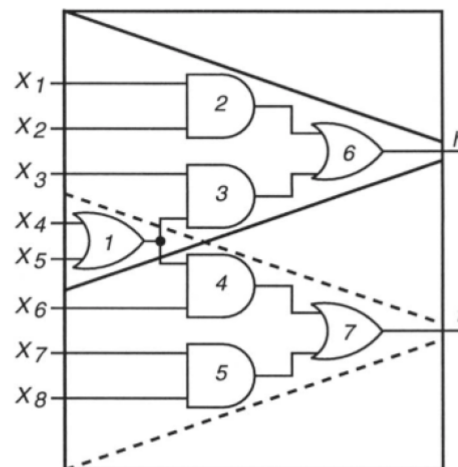


Figure 2.24: Pseudoexhaustive testing

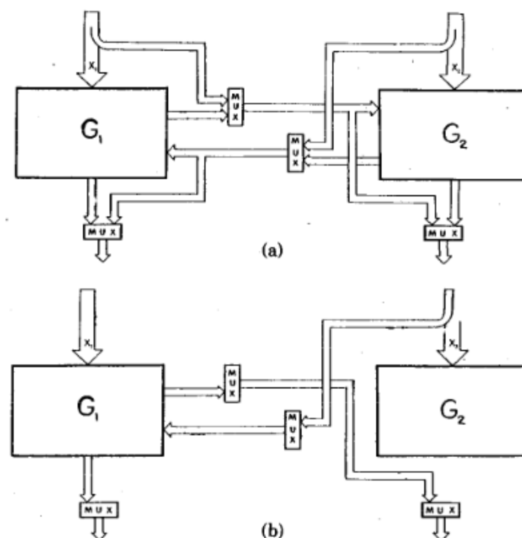


Figure 2.25: Hardware partitioning

outputs of each sub-circuit to those primary inputs and outputs that are not used by the sub-circuit under test. Figure 2.25(a) depicts this hardware partitioning scheme. By controlling the multiplexers, all the inputs and outputs of each sub-circuit can be accessed using primary input and output lines. For example, to test sub-circuit Gi the multiplexers can be controlled as depicted in Figure 2.25(b) to completely access all the inputs and outputs (including the embedded intermodule lines).

For the purpose of hardware partitioning, a considerable amont of diagnostic circuitry consists of routing multiplexers. Insertion of these muxes hinder the speed of operation and are expensive.

In sensitized partitioning, same testing discipline is achieved without insertion of any multiplxers. Circuit partitioning and sub-circuit isolation can be achieved by applying the

appropriate input pattern to some of the input lines. The effect achieved is similar to that of hardware partitioning. Paths from the primary inputs to the sub-circuit inputs and paths from the sub-circuit output to the primary output can be sensitized. Using these paths each sub-circuit can be tested exhaustively. The challenge in sensitized partitioning is whether all possible combinations of sub-circuit inputs can be delivered to the sub-circuit. Combining harware partitioning with sensitized partitioning would give best accessability.

Pseudoexhautive testing has potential application to ROBDD based implementation since the synthesis is modular and, sensitising and propagation paths are readily available.

Chapter summary

: In this chapter it is demonstrated that it is better to work with combinational circuits than sequential circuits as combinational circuits have polynomial test generation complexity. Algorithmic as well as hardware approaches that facilitate ease of test generation under PDF and MSAF models are depicted. A case for a BDD based synthesis approach is developed. The upcoming chapter expalins the BDD prelimaries and provides steps for the proposed synthesis.

Chapter 3

ROBDD Based Synthesis

Logic synthesis is a process by which an abstract form of desired circuit behavior, typically at register transfer level (RTL), is turned into a design implementation in terms of logic gates, typically by a synthesis tool. The synthesis tool is usually guided by constraints provided by the user for the purpose of area and timing optimization. As seen in earlier chapters, it is advisable to use different synthesis approaches to improve the testability of digital circuit designs. Binary Decision Diagram(BDD) based synthesis is known to have higer degree of testability as compared to other techniques. In this chapter we discuss BDD based circuit transformations that serve our motive of complete PDF and MSAF testabilty.

3.1 Binary Decision Diagrams

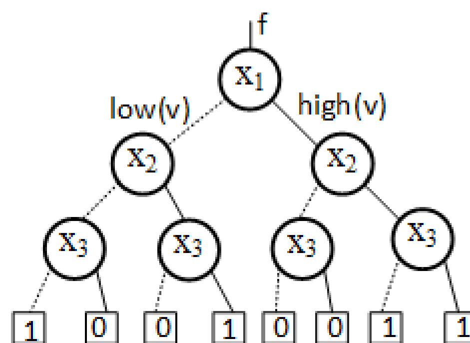


Figure 3.1: Binary Decision Diagram for $x_1x_2 + \bar{x}_1\bar{x}_2\bar{x}_3 + \bar{x}_1x_2x_3$

A *Binary Decision Diagram* or BDD[10] is a rooted acyclic graph $G = (V, E)$ with vertex set V containing two types of vertices, *non-terminal* and *terminal* vertices. Consider Figure 3.1, nodes marked by x_1, x_2 andx_3 are non-terminal nodes. A terminal vertex(node) v has attribute $value(v) \in \{0, 1\}$. A non-terminal vertex v has attribute

index(v) $\in \{1, 2, \cdots, n\}$ and two children, *low*(v), *high*(v) (marked by dotted and straight lines respectively, in Figure 3.1). The attribute *index*(v) specifies a linear ordering of the variables in the support of the BDD; i.e., it satisfies the property such that for any non-terminal node v, $index(v) < \{index(low(v)), index(high(v))\}$; For example: $x_1 < x_2$.

Reduced Ordered Binary Decision Diagram or ROBDD extends the above definitions of BDD with two additional constraints:

- For any non-terminal node v, if *low*(v) is also non-terminal then $index(v) < index(low(v))$, and similarly, if *high*(v) is also non-terminal, then $index(v) < index(high(v))$.
- There exists no $v \in V$ with $low(v) = high(v)$ and there are no two nodes v and v' such that the sub-BDDs rooted by v and v' are **isomorphic**.

The ROBDD for Figure 3.1 is shown in Figure 3.2.

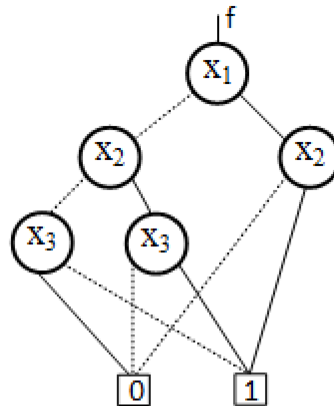


Figure 3.2: Reduced Ordered Binary Decision Diagram for $x_1x_2 + \bar{x}_1\bar{x}_2\bar{x}_3 + \bar{x}_1x_2x_3$

In an ROBDD, every sub-graph that is rooted at vertex v represents a unique function f_v. The directed acyclic graph rooted at root r of the entire ROBDD represents the main Boolean function $f = f_v$. A function f_v is stored in the data structure of the node v and is defined recursively as follows:

- If v is a terminal node such that $value(v) = 0$, then $f_v = 0$, else if v is a terminal node such that $value(v) = 1$, then $f_v = 1$
- If v is a non-terminal node such that $index(v) = i$ then, $f_v = x_i f_{high(v)} + \bar{x}_i f_{low(v)}$, where the variable order is $x_1 < x_2 < \cdots < x_n$ and, $f_{high(v)}$ and $f_{low(v)}$ represent the functions rooted at *high*(v) and *low*(v) respectively

Thus each ROBDD node v can be represented by the Shannon expansion in Equation 3.1.

$$f_v = \bar{x}_i \cdot f_v^{\bar{x}_i} + x_i \cdot f_v^{x_i} \tag{3.1}$$

$f_v^{x_i}$ and $f_v^{\bar{x}_i}$ are evaluated at $x_i = 1$ and $x_i = 0$ respectively. This theorem allows to construct a unique Boolean function at each node of the ROBDD $G = (V, E)$, starting from the root which represents the main function of the entire ROBDD. The construction is performed as follows; given a node v representing f_v, and v with attribute $index(v) = i$, the functions represented by it's two children, $low(v)$ and $high(v)$ are $f_{low(v)} = f^{\bar{x}_i}$ and $high(v) = f^{x_i}$, respectively. $f^{\bar{x}_i}$ and f^{x_i} are known as the co-factors of f with respect to $\bar{x}_i$ and x_i respectively.

Since the variable ordering is fixed for the ROBDD, the representation of the Boolean function is canonical. A circuit with m outputs will have a ROBDD with m root nodes and two terminal or leaf nodes: leaf node-0 and leaf node-1. This would be a shared ROBDD. For the current work we consider a single output function, hence a single output ROBDD as shown in Fig.3.3.

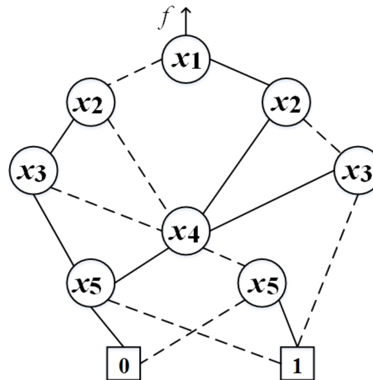
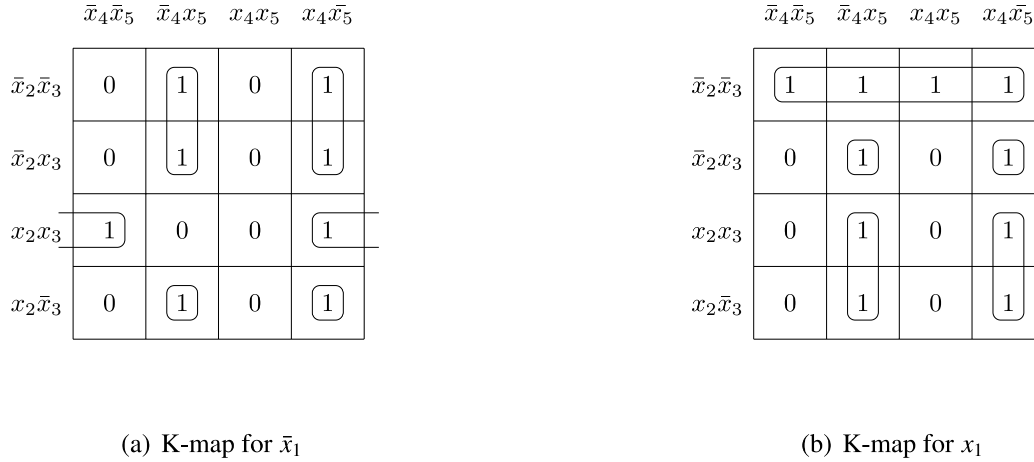


Figure 3.3: ROBDD example for Boolean function f

For the purpose of understanding, the K-maps for the function indicated in Figure 3.3 are shown in Figure 3.4.

It can deduced from the K-maps, that the minterms resulting from the ROBDD construction are *irredundant* and *canonical*. It can also be seen that, the minterms are *disjoint* making each minterm a prime implicant. Thus the function f can be represented by a *Disjoint Sum of Products*(DSOP) expression [14].

For a logic function represented by ROBDD, all the individual paths starting from a root node to leaf node 1 constitute the DSOP. The DSOP of function f of Figure 3.3 is as



(a) K-map for $\bar{x}_1$ (b) K-map for x_1

Figure 3.4: K-maps for function f

follows:

$$
\begin{aligned}
f = x_1\bar{x}_2\bar{x}_3 + x_1\bar{x}_2x_3\bar{x}_4x_5 + x_1\bar{x}_2x_3x_4\bar{x}_5 \\
+x_1x_2\bar{x}_4x_5 + x_1x_2x_4\bar{x}_5 + \bar{x}_1\bar{x}_2\bar{x}_4x_5 \\
+\bar{x}_1\bar{x}_2x_4\bar{x}_5 + \bar{x}_1x_2\bar{x}_3\bar{x}_4x_5 + \bar{x}_1x_2\bar{x}_3x_4\bar{x}_5 \\
+\bar{x}_1x_2x_3\bar{x}_5
\end{aligned}
\tag{3.2}
$$

The DSOP pays a crucial role in test generation for *Path Delay Faults*(PDFs) as well as *Multiple Stuck At Faults*(MSAFs).

3.2 Synthesis Without Additional Input

ROBDD based synthesis involves implementing each node with a decision making circuit. Intuitively a 2x1 mux can be used to replace each ROBDD node. Alternatively, an XOR based circuit can also be used. This work has been published in [47].

According to the placement of the ROBDD nodes, they can be classified into

- Nodes with both edges (*high*(v) and *low*(v)) connected to other nodes
- Nodes with both edges connected to leaf nodes
- Nodes with one of the edges connected to leaf nodes

Nodes that have both edges connected to other nodes are implemented using multiplexers where the node variable x_i acts as a control signal and the co-factors $f_v^{\bar{x}_i}$ and $f_v^{x_i}$ are the mux inputs; Example nodes x_1, x_2, x_4 in Figure 3.3. Nodes that have both edges connected to leaf nodes do not require a dedicated mux. Consider node x_5 in Figure 3.3 whose *high*(v) is connected to 0 and *low*(v) is connected to 1. This is equivalent to $f_v^{x_4} = \bar{x}_5$. So

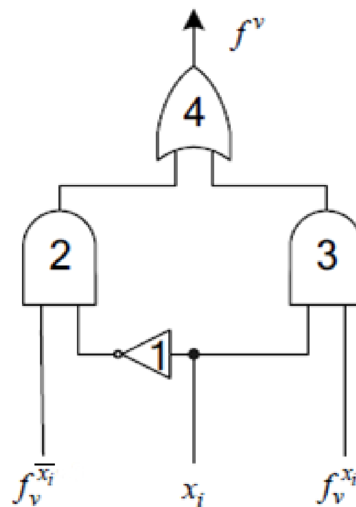


Figure 3.5: Gate implementation of the formula $f_v = \bar{x}_i \cdot f_v^{\bar{x}_i} + x_i \cdot f_v^{x_i}$

x_5 can directly be connected to the mux inputs controlled by x_4. Some transformation is required for nodes with one of the edges connected to leaf nodes. This is discussed as follows.

In order to eliminate the additional control input of [15], the node connected to the leaf nodes are replaced by a slightly modified circuit as compared to the 2x1 mux. All intermediate nodes are made up of Invert-AND-OR sub-circuits(2x1 muxes) shown in Fig.3.5.

Each node that connects to a leaf node can have either of four conditions:

- High successor connected to leaf node 0
- High successor connected to leaf node 1
- Low successor connected to leaf node 0
- Low successor connected to leaf node 1

If any of the successors is connected to 0, then the sub-circuit can be simplified to an AND gate without disturbing the testability. If one of the successors is connected to leaf node 1 then the Invert-AND-OR sub-circuit simplifies to an OR gate. This will cause a transition on the control variable to be masked by a 1 on the other input of the OR gate[15]. Instead of the OR gate, following Boolean algebra rule is used.

$$\bar{x}_i + f_{x_j} = \bar{x}_i + x_i \cdot f_{x_j} \tag{3.3}$$

$$x_i + f_{x_j} = x_i + \bar{x}_i \cdot f_{x_j} \tag{3.4}$$

The resultant transformation for circuits connected to the leaf nodes is shown in Figures 3.6 and 3.7. This ensures that any OR gate input that lies on the OFF path always has a non controlling value.

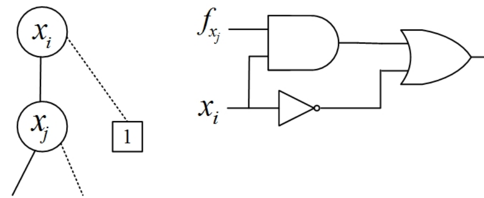


Figure 3.6: Sub-circuit for nodes connected to leaf node 1

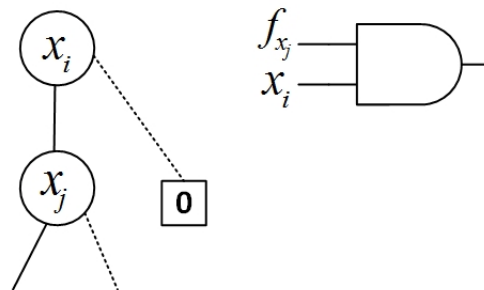


Figure 3.7: Sub-circuit for nodes connected to leaf node 0

The premise of using this transformation can be deduced from the following example.

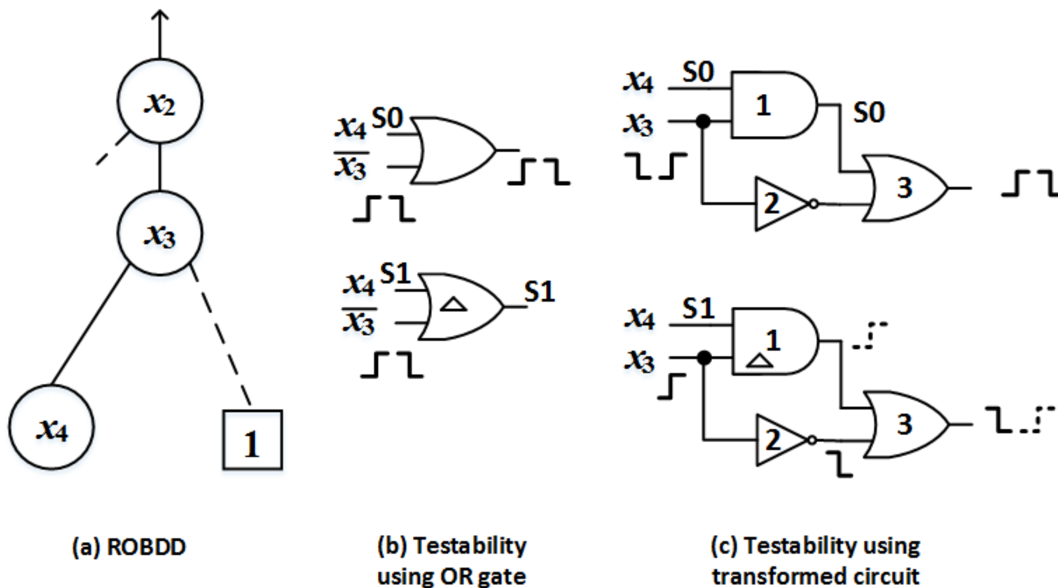


Figure 3.8: Example 3.2.1:Improving testability by avoiding simplification

Example 3.2.1 *Consider a section of an ROBDD in Figure 3.8(a). The DSOP terms would be* $x_2\bar{x}_3 + x_2x_3x_4$*. This would reduce to* $x_2(\bar{x}_3 + x_4)$*. The implementation using OR gate is shown in Figure 3.8(b). If* $x_4 = 0$*, any transition on* $\bar{x}_3$ *can be robustly propagated. If subpath* x_2x_3 *from path* $x_2x_3x_4$ *is to be tested via a transition at* x_3*, the transition will be masked by a 1 at* x_4*. But if implementation of Figure 3.6 is followed, the untestability via* x_3 *can be overcome. Consider Figure 3.8(c). If* $x_4 = 0$*, any transition on* x_3 *can be propagated to the next stage via gates 2 and 3. Let us assume that path* x_3 *- gate 2 - gate 3 is tested and is fault free. When* $x_4 = 1$*, and a rising transition is applied at* x_3*, paths* x_3 *- gate 1 - gate 3 and* x_3 *- gate 2 - gate 3 will be activated. Since path* x_3 *- gate 2 - gate 3 is fault free, the corresponding transition arrives in time. If path* x_3 *- gate 1 - gate 3 has a delay fault, the corresponding transition will be delayed. This results in a glitch-like signal at the output. If this signal is captured at the clock edge, then the delay fault in path* x_3 *- gate 1 - gate 3 is detected. Thus this transformation increases testability as compared to OR gate implementation.*

The composite circuit for the ROBDD in Fig.3.3 is shown in Fig.3.9.

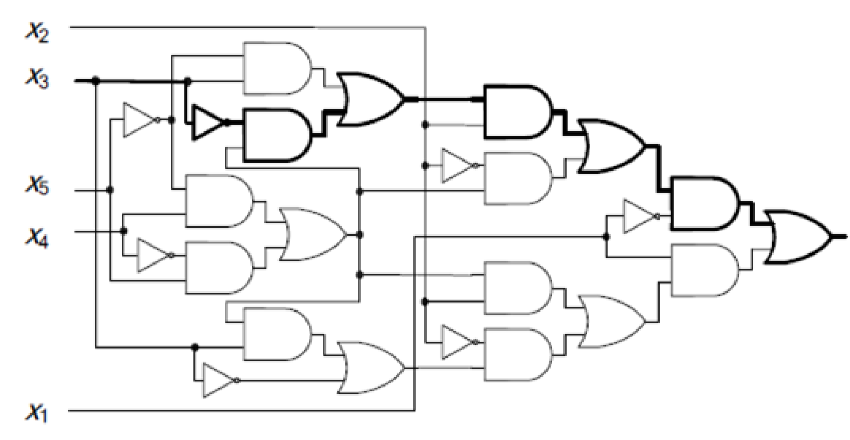


Figure 3.9: Circuit *C* corresponding to function *f* in Fig. 3.3

Equation (3.2) indicates the DSOP equation for circuit C (Figure 3.9).

It should be noted that in an ROBDD based implementation, the ordering of variables is fixed. For ease of understanding, the variables are ranked as per their position in the ROBDD. The root node variable x_1 is ranked the highest and the node variable farthest form the root node is ranked the lowest. In Equation 3.2, x_1 has the highest rank and x_5 has the lowest rank.

Limitations of ROBDD based synthesis

ROBDD based designs have two major disadvantages:

- The number of paths may increase exponentially
- Since the levels are not bounded, large path delays could exist

Chapter summary

: This chapter provides circuit transformation steps for an ROBDD based synthesis. It is shown that each path can be represented by a DSOP. All DSOP terms combined represent all paths of the circuit. No paths in the synthesized circuit are redundant. No additional control signal is required. Analysis of the synthesized circuit for PDF testability is discussed in the next chapter.

Chapter 4

Path Delay Fault Testability

For any path in an ROBDD based circuit to be fully testable for delays, it is required that all single stuck-at faults of the path are testable. This chapter aims to establish complete PDF testability of ROBDD based synthesized circuit.

4.1 Single Stuck-at Fault Testability

Since the circuit is constructed using AND-OR implementation, all single stuck-at-0 faults on the path would be tested by the product term that represents the path.

Using fault equivalence, any single stuck-at-1 fault at any location on a path can be traced to the nearest input variable. This variable is used to sensitize the fault.

Single stuck-at faults can occur at primary inputs or on the gates which fall on a path of circuit C. Let this path be represented by product K from the DSOP which includes primary input x_i; $K = f(x_1, \cdots, x_i, \cdots, x_{n-1}, x_n)$. Let $K^{'}$ be a product term obtained by complementing only x_i to $\bar{x}_i$, all other variables remaining as they are in K; $K^{'} = f(x_1, \cdots, \bar{x}_i, \cdots, x_{n-1}, x_n)$.

Lemma 1 *If $K^{'}$ is not an implicant of DSOP then the fault $x_{i(s-a-1)}$ is detectable*

Proof *In the terms K and $K^{'}$, only the value of variable x_i differs. Since K is one of the DSOP terms, $K^{'}$ will not be a DSOP term. In an ROBDD, K and $K^{'}$ both cannot exist since isomorphic sub-graphs are removed. A test vector tv_1 that turns $K^{'}$ into 1 will turn K to 0 in a fault free condition. None of the paths are functionally active and output is 0. If a stuck-at-1 fault exists at x_i, application of tv_1 causes the output to be 1 and $x_{i(s-a-1)}$ detected.*

Lemma 2 *A test vector tv_0 that turns K to 1 detects $x_{i(s-a-0)}$*

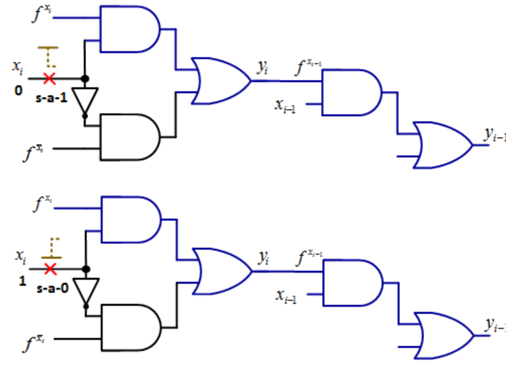


Figure 4.1: Stuck-at faults corresponding to delay faults

Proof *If test vector tv_0 turns K to 1, it will turn K' to 0 in fault free case. If application of tv_0 causes the output to be 0, a stuck-at-0 fault exists at x_i. This is true for all input variables included in K. Thus tv_0 will detect all single stuck-at-0 faults of the input variables that constitute the path K.*

4.2 Robust and Non-Robust Testability Conditions

A stuck-at-0 fault can also be considered as *slow-to-rise* delay fault and a stuck-at-1 fault as *slow-to-fall*, demonstared in Figure 4.1. In [33] the conditions of robust path delay fault manifestation for test pairs are formulated and are as follows:

1. The stuck-at fault corresponding to the delay fault of the literal should be detectable. The corresponding test vector constitutes v_2 of the test pair (v_1, v_2).

2. The variable x_i from where the path begins takes complementary values for v_1, v_2 vectors.

It can be inferred from lemmas 1 and 2 that all single stuck-at faults of the path under consideration are detectable. Also the test vectors have complementary values of literal-under-consideration for stuck-at faults of opposite polarity. Thus both the conditions stated above are satisfied. To adopt this to the DSOP for the purpose of generating test pairs for the PDF manifestation[33], additional notions are required.

Let x_1 represent the root node, x_n indicate the last literal (as per ROBDD rank of variables) and x_i mark the node where the path begins. A path that begins at x_i ends at the root node x_1 and constitues a sub-product of the DSOP term of that path. If the node x_i where the path begins is not the last node in the corresponding DSOP term then there may be several prolongations and consequently several products containing the literal x_i i.e. there could be many paths from x_i to leaf node 1. Now the product term representation can

be split into three parts, viz. sub-product including x_1 to $x_{i-1}(K_\varepsilon)$, x_i, set of sub-products from x_{i+1} to $x_n(\{K_\alpha\})$. This composite product term be represented as $\{K_\varepsilon, x_i, \{K_\alpha\}\}$ [33].

Considering one 2x1 mux of the circuit, the paths controlled by x_i and $\bar{x}_i$ in the same sub-circuit form *companion* paths. The other inputs to the mux are $f_v^{\bar{x}_i}$ and $f_v^{x_i}$ Gates 1-2-4 and 3-4 in Figure 3.5 form companion paths. They have the same sub-product K_ε, i.e. the propagation path to circuit output via root node. Let $K(u)$ be the minimal cube covering vectors v_1, v_2 of a test pair. Let the path under test be β (gates 3-4 in Fig.3.5).

Theorem 3 *If $K_\gamma \in \{K_\alpha\}$ is a sub-product on which the condition $f_v^{\bar{x}_i} \neq f_v^{x_i}$ occurs, especially $f_v^{\bar{x}_i} = 0$ and $f_v^{x_i} = 1$ and the product $K_\varepsilon K_\gamma$ (excluding $x_i, \bar{x}_i$) is represented by K^*, then this product K^* represents a test pair which can robustly test path β for rising and falling transitions in x_i.*

Proof *Since $K_\gamma \in \{K_\alpha\}$, the product term $x_i K_\varepsilon K_\gamma = x_i K^*$ represents a valid DSOP term which makes the output 1. The product $x_i K^*$ represents the Boolean vector v_2 that turns the product $K_\varepsilon K_\gamma$ to 1. It means v_2 is a test pattern for the fault. The product $\bar{x}_i K^*$ represents vector v_1 of this test pair. The product v_1 turns DSOP into 0. The test pair detects rising transition of the path β. Actually v_1 is orthogonal to all products of the DSOP that does not contain sub-product k_ε and $\bar{x}_i$ as k_γ is orthogonal to $f_v^{\bar{x}_i}$. To test for falling transition v_1 must be taken as v_2 and vice versa for the same path β. $K(u) = K^*$ is orthogonal to*

- *All products of the DSOP except products of $\{K_\alpha\}$*
- *All products of the DSOP that does not contain sub-product k_ε*
- *Products of the DSOP containing sub-product k_ε and $\bar{x}_i$ as k_γ is orthogonal to $f_v^{\bar{x}_i}$*

Also none of the DSOP products contain repeated literals. Thus all conditions of robust testable manifestation for rising and falling transition of the path β are fulfilled. Hence the theorem is proved.

Corollary 4 *If there exists a path γ for which $f_v^{x_i}(\gamma) = 1, f_v^{\bar{x}_i}(\gamma) = 0$ and a path δ for which $f_v^{x_i}(\delta) = 0, f_v^{\bar{x}_i}(\delta) = 1$ then for both paths that begin at the same input and marked with the literals x_i and $\bar{x}_i$, PDFs manifest themselves as robust testable for rising and falling transitions.*

Figure 4.2 illustrates the above mentioned corollary.

Example 4.2.1 *Consider a path that begins from x_3 to root node x_1 in Figure 3.9. ($k_\varepsilon = \bar{x}_1 x_2 \bar{x}_3$). This sub path exists in the 8th term and the 9th term of the DSoP.*

$$K_\alpha = \{\bar{x}_1 x_2 \bar{x}_3 \bar{x}_4 x_5, \bar{x}_1 x_2 \bar{x}_3 x_4 \bar{x}_5\}$$

We can use either of the terms to generate the test pair (v_1, v_2). *Selecting* 8th *term* $\bar{x}_1 x_2 \bar{x}_3 \bar{x}_4 x_5$ *gives* $v_2 = 01001$. *To test the path via* x_3*, the value of* x_3 *is complemented to get* $v_1 = \bar{x}_1 x_2 x_3 \bar{x}_4 x_5 = 01101$. *It should be noted that* v_1 *is not an implicant of the DSOP. Thus* $(v_1, v_2) = (01\mathbf{1}01, 01\mathbf{0}01)$. *This sequence will test the path with rising transition via* x_3. *Interchanging* v_1 *&* v_2 *will test the path with falling transition. Hence to test for falling transition via* x_3, $(v_1, v_2) = (01\mathbf{0}01, 01\mathbf{1}01)$.

The values for $f_v^{\bar{x}_i}$ & $f_v^{x_i}$ are generated by combinations of input varaibles x_{i+1} to x_n . x_i is the primary input which can be driven directly. For each ROBDD node only one of the following conditions hold good. These conditions arise because of the ROBDD structure itself. The conditions are as follows:

1. $f_v^{\bar{x}_i} = 0, f_v^{x_i} = 1$ as well as $f_v^{\bar{x}_i} = 1, f_v^{x_i} = 0$ can be delivered to the node or
2. Only $f_v^{\bar{x}_i} = 0, f_v^{x_i} = 1$ can be delivered to the node or
3. Only $f_v^{\bar{x}_i} = 1, f_v^{x_i} = 0$ can be delivered to the node

Condition 1 exists if $f_v^{x_i} \not\subset f_v^{\bar{x}_i}$, $f_v^{\bar{x}_i} \not\subset f_v^{x_i}$ and $f_v^{x_i} \cap f_v^{\bar{x}_i} = 0$. Condition 2 exists if $f_v^{\bar{x}_i} \subset f_v^{x_i}$ and condition 3 exists if $f_v^{x_i} \subset f_v^{\bar{x}_i}$ for a node v.

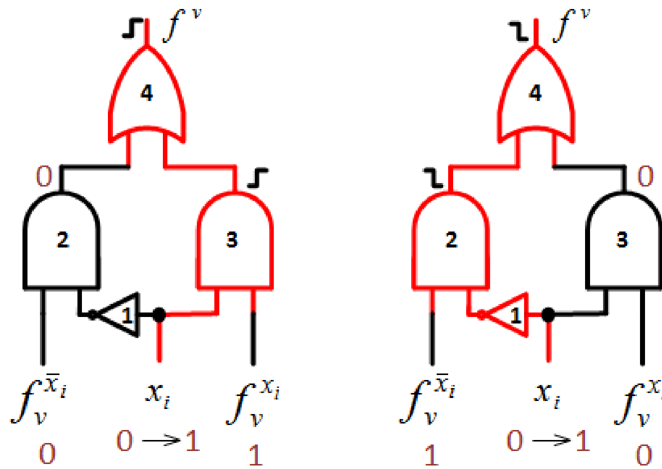


Figure 4.2: Robustly testable conditions

In case 1, both paths of the mux (1-2-4, 3-4) are robustly testable. This is demonstrated in Fig. 4.2. In case 2, path 3-4 is robustly testable and in case 3, path 1-2-4 is robustly testable. It is possible to test the companion paths in cases 2 & 3 for PDFs non-robustly.

Theorem 5 *Let there be a ROBDD node, wherein the condition* $f_v^{x_i} = 0, f_v^{\bar{x}_i} = 1$ *does not exist, however let* K_γ *be a sub-product on which the condition* $f_v^{\bar{x}_i} = 1$ *and* $f_v^{x_i} = 1$ *exists*

and the product $K_\varepsilon K_\gamma$ (excluding $x_i, \bar{x}_i$) is represented by K^*. The product K^* represents a test pair which can non-robustly test path α(1-2-4) for rising transition.

Proof *The product $x_i K^*$ represents the Boolean vector v_2 that turns the product K_γ from $\{K_\alpha\}$ to 1. It means v_2 is a test pattern for the fault $x_{i(s-a-0)}$. The product $\bar{x}_i K^*$ represents vector v_1 of this test pair. The test pair detects rising transition of the path α. The product v_1 turns K^* to 1. Moreover as the condition $f_v^{x_i} = 0, f_v^{\bar{x}_i} = 1$ is not feasible, the product K^* is an implicant of DSOP and consequently v_1 is not a test pattern for the fault. Besides it $K(u)$ is also implicant of the DSOP. Hence the theorem is proved.*

If path 3-4, which is robustly testable in this case, is tested and found fault free then the non-robust test for path 1-2-4 will not be invalidated. Thus for cases 2 and 3 mentioned earlier, the companion paths of the robustly testable paths can be tested using validatable non-robust tests. This condition is demonstrated in Fig.4.3.

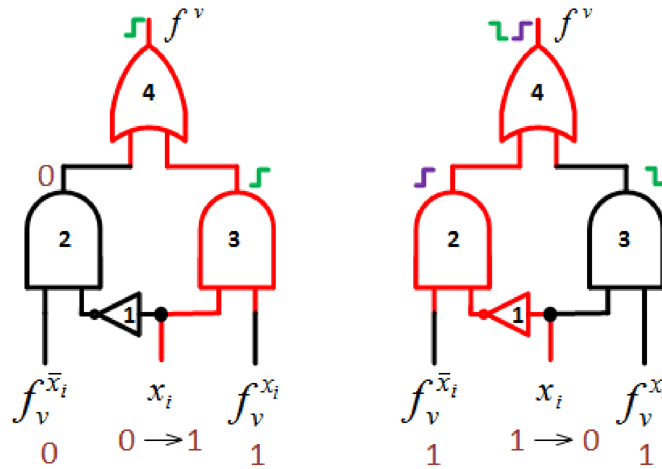


Figure 4.3: Non-robustly testable conditions

In order to detect validatable non-robust PDF of the path and its rising transition, the test pair to detect robust testable PDF for the companion path has to be delivered first. If the companion path is fault free, validatable non-robust test will detect PDF of rising/falling transition of the considered path.

- If only $f_v^{\bar{x}_i} = 0, f_v^{x_i} = 1$ can be delivered to the node, path 3-4 is robustly testable, path 1-2-4 has a validatable non-robust testable PDF in falling transition at x_i.
- If only $f_v^{\bar{x}_i} = 1, f_v^{x_i} = 0$ can be delivered to the node, path 1-2-4 is robustly testable, path 3-4 has a validatable non-robust testable PDF in rising transition at x_i.

4.3 Test Vector Generation

Each ROBDD node is controlled by an input variable. Thus each node becomes a potential beginning of a path that ends in root node x_1. Given a DSOP term, the test vector pair of the path $\{x_1, ...x_i\}$ consists of

- A product term K_α consisting of variables from x_{i+1} to x_n, this product term sensitises the path.
- A product term K_ϵ consisting of variables from x_1 to x_{i-1}, this product term is responsible for fault propagation.
- The primary input x_i where the path begins.

From Section 4.2 we infer that the test vector pair of a path corresponds to the DSOP term it represents, where the literal of the path origin takes complementary values. $K_\alpha(\bar{x}_i)x_iK_\epsilon$ will be the test vector pair (v_1, v_2).

DSOP manipulation is used to derive the PDF test sets. Since initial steps of the process are common with MSAF test vector derivation, the topic is revisited in Chapter 6.

Chapter summary

: In this chapter, it is proved that each path in the synthesis satisfies conditions of path delay testability. Conditions that lead to classification of paths as robustly testable and non-robustly testable for delays, are discussed. It is shown that the non-robust tests would not be invalidated, hence the test quality is as good as robust tests. Test vector pairs can be generated from the primary input variables, no additional inputs are required. MSAF testability of the synthesis approach is covered in the upcoming chapter.

Chapter 5

Multiple Stuck-at Fault Testability Analysis

Testing a circuit exhaustively has two benefits, viz.: Fault models including multiple stuck-at faults, bridging faults are covered and test generation effort is negligible. This would be highly infeasible as the circuit complexity increases. In the current example of circuit C, the output is a function of 5 variables. The number of input variables being very small, exhaustive test set that has $2^5 = 32$ vectors would detect all MSAFs. As the number of inputs increase for larger circuits, the test vector volume explodes.

Alternatively, a larger circuit can be segmented into partitions having few inputs so that they can be tested exhaustively for those inputs,i.e., Pesudoexhaustive testing (proposed by McCluskey[37][36][52]). This test strategy is also described in [**?**] and [1]. Each node in an ROBDD is implemented using a sub-circuit. This makes ROBDD based circuits inherently segmented/partitioned.

Each ROBDD node has a fixed order of nodes under its hierarchy. The nodes are marked by primary inputs. The number of inputs in the fanin cone of a node would depend on order of the node. Depending on how many inputs can be considered for exhaustive testing, a corresponding node and sub-circuit output is chosen for observation. Here additional logic would be required to observe the segment output directly. A multiplexer with increased size is used before the scan flop. A default selection will select the circuit output and during testing the multiplexer can choose between the segment outputs. Figure 5.1 shows an example of segmentation of circuit C shown in Figure 3.9. Table 1 shows segment representation.

It may not always be possible to implement large multiplexers as they hinder the speed of operation and are expensive. As ROBDD based implementation represents a binary tree structure, additional multiplexers can be avoided. Each node is implemented

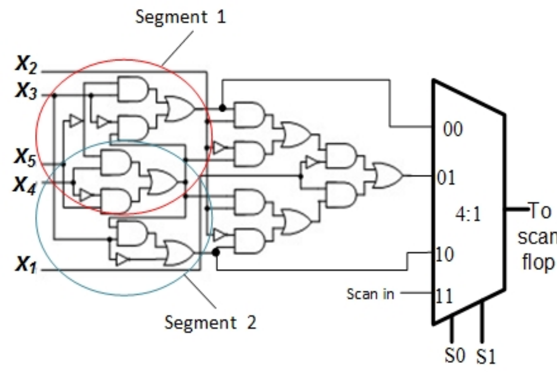


Figure 5.1: Test partitions of circuit in Figure 3.9

Table 5.1: Test partitions of Fig. 5.1

Mux select	Mux output
0	$f(x_3, x_4, x_5)$
1	$f(x_1, x_2, x_3, x_4, x_5)$ (Normal output)
2	$f(x_3, x_4, x_5)$
3	Scan input

using a sub-circuit redrawn here as Figure 5.2. Since the entire circuit is composed of the same sub-circuit, it is a natural candidate for the partition. Paths from the primary inputs to the sub-circuit inputs and from sub-circuit outputs to the circuit output need to be sensitized [37]. These paths will allow the sub-circuits to be tested exhaustively. Since the ROBDD based implementation has all the paths testable under the *path delay fault model*, sensitization and propagation is ensured for each path. This work has been published in [45],[46]. Test vectors derived for each sub-circuit drive the primary inputs so as to sensitize the faults of partition-under-test and propagate the partition output to the circuit output.

5.1 MSAF testability of two-level AND-OR circuits

It has been proved in Section 4.2 that the circuit C is fully path delay testable. This indicates that fault sensitizing and propagating paths are readily available. Test vectors derived for each mux drive the primary inputs so as to sensitize the faults of partition-under-test(one 2x1 mux corresponding to one node) and propagate the partition output to the circuit output.

The expression for each ROBDD node v is

$$f_v = \bar{x}_i \cdot f_v^{\bar{x}_i} + x_i \cdot f_v^{x_i}$$

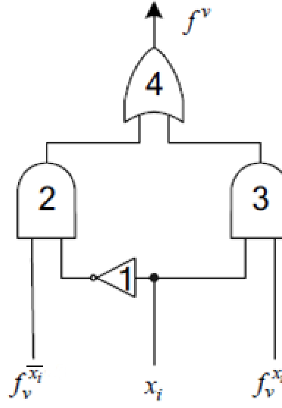


Figure 5.2: Gate implementation of the formula $f_v = \bar{x}_i \cdot f_v^{\bar{x}_i} + x_i \cdot f_v^{x_i}$

Since the inputs are $x_i, f_v^{\bar{x}_i}$ and $f_v^{x_i}$, eight combinations are required to test the mux exhaustively.

The values for $f_v^{\bar{x}_i}$ & $f_v^{x_i}$ are generated by combinations of input variables x_{i+1} to x_n. x_i is the primary input which can be driven directly. For each ROBDD node only one of the following conditions hold good. These conditions arise because of the ROBDD structure itself. The conditions are as follows:

1. $f_v^{\bar{x}_i} = 0, f_v^{x_i} = 1$ as well as $f_v^{\bar{x}_i} = 1, f_v^{x_i} = 0$ can be delivered to the node or

2. Only $f_v^{\bar{x}_i} = 0, f_v^{x_i} = 1$ can be delivered to the node or

3. Only $f_v^{\bar{x}_i} = 1, f_v^{x_i} = 0$ can be delivered to the node

Condition 1 exists if $f_v^{x_i} \not\subset f_v^{\bar{x}_i}, f_v^{\bar{x}_i} \not\subset f_v^{x_i}$ and $f_v^{x_i} \cap f_v^{\bar{x}_i} = 0$. Condition 2 exists if $f_v^{\bar{x}_i} \subset f_v^{x_i}$ and condition 3 exists if $f_v^{x_i} \subset f_v^{\bar{x}_i}$ for a node v.

This means all 8 values of input combinations can not be applied. Thus any mux in the circuit cannot be tested pseudoexhaustively. Though exhaustive values cannot be applied to the node, four values can definitely be applied to each node.

We need to apply a different approach to detect all multiple faults in the mux. The 2x1 mux circuit is a two-level irredundant AND-OR circuit. Any test set which detects every single fault in a two-level single-output combinational circuit also detects every multiple fault in the circuit[44]. Further, for irredundant two-level AND-OR networks, a test vector set that detects all multiple stuck-at faults at the inputs of the network will also detect all MSAF combinations of the network [29]. Thus, if the four values that can be applied to the node detect all multiple faults at the mux inputs $x_i, f_v^{\bar{x}_i}$ and $f_v^{x_i}$, all MSAFs of the mux will be tested.

Since the circuit in Fig. 3.5 is a two-level irredundant AND-OR circuit, each product term is a prime implicant. If one of the variables is stuck-at-0 then the entire term disap-

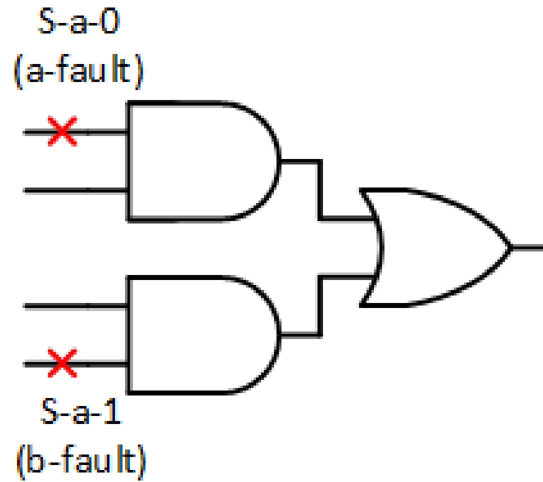


Figure 5.3: a,b faults

pears creating *disappearing* fault or *a-fault* [29]. If a variable is stuck-at-1, it disappears from the product term creating *shrinkage* or *b-fault* [29].

Lemma 6 *Single stuck-at fault at a gate input connected to x_i is equivalent to multiple a,b-faults of the 2×1 mux sub-circuit*

Proof *For the 2x1 mux (shown in Figure 3.5), stuck-at-1 fault at input x_i corresponds to the multiple a,b-fault because of the following:*

1. *First product term of Equation 1 disappears due to literal $\bar{x}_i$ turning into constant 0 (a-fault).*

2. *Literal x_i disappears in the second product of the formula since it is constant 1 (b-fault).*

Similarly, stuck-at-0 fault at input x_i also corresponds to the multiple a,b-fault as:

1. *Literal $\bar{x}_i$ disappears in the first product of the formula.*

2. *Second product disappears due to x_i being forced to be constant 0 (a-fault).*

Hence proved.

Lemma 7 *A test vector set that detects all a,b faults at sub-circuit inputs (x_i, $f_v^{\bar{x}_i}$, $f_v^{x_i}$) will detect all multiple faults of the sub-circuit.*

Proof *From Lemma 1, we see that a single stuck-at fault causes disappearance of either a literal (b-fault) or a product term (a-fault). Since Equation 1 represents an irredundant*

SOP, all the terms are prime implicants. To test whether a product term disappears, it is needed to test only that product term which itself is a prime implicant. Thus the two terms of the Equation 1 will detect all s-a-0 faults at the AND gate inputs (called a-tests). If a literal in the product term disappears due to a s-a-1 fault then the product term is realized without the literal in question. The sub-cube next to this product term having a complemented value of the literal will detect this fault (b-test). The set of a,b-tests will detect all MSAFs at sub-circuit inputs. Therefore all MSAFs of the sub-circuit are detected [29].

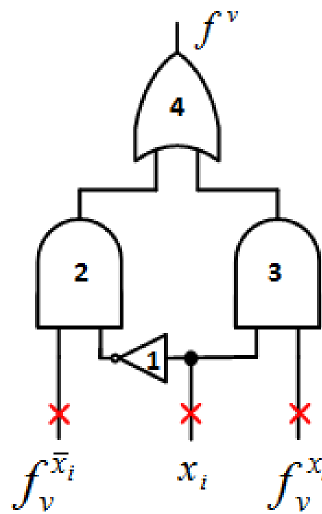


Figure 5.4: Multiple fault locations at inputs of sub-circuit

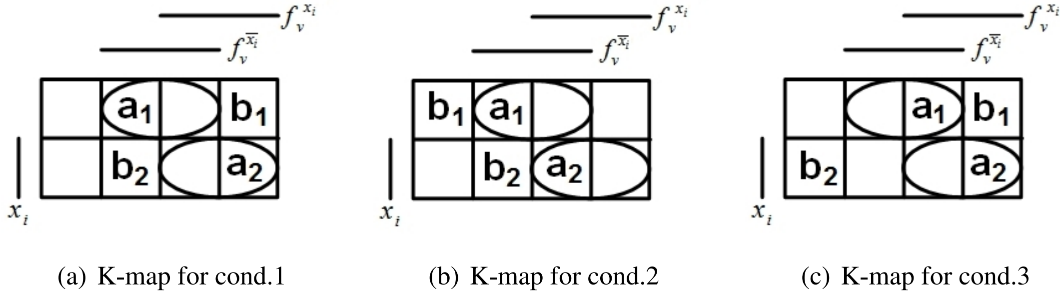
5.2 Deriving MSAF test set for each node

Depending on the conditions discussed in Section 5.1 *a,b-test* set for each case is as follows:

1. When $f_v^{\bar{x}_i} = 0, f_v^{x_i} = 1$ as well as the condition $f_v^{\bar{x}_i} = 1, f_v^{x_i} = 0$ is true, *a,b-test* set is $(x_i, f_v^{x_i}, f_v^{\bar{x}_i}) = \{001, 010, 101, 110\}$ as shown in Figure 5.5(a).

2. Only when the condition $f_v^{\bar{x}_i} = 0, f_v^{x_i} = 1$ is true, *a,b-test* set is $(x_i, f_v^{x_i}, f_v^{\bar{x}_i}) = \{000, 001, 101, 111\}$ as shown in Figure 5.5(b).

3. Only when the condition $f_v^{\bar{x}_i} = 1, f_v^{x_i} = 0$ is true, *a,b-test* set is $(x_i, f_v^{x_i}, f_v^{\bar{x}_i}) = \{010, 011, 100, 110\}$ as shown in Figure 5.5(c).



(a) K-map for cond.1 (b) K-map for cond.2 (c) K-map for cond.3

Figure 5.5: K-maps for ROBDD nodes

There are two methods that can be used to derive MSAF test vectors. The first method that uses BDD manipulation is dicussed here. Alternatively, a DSOP manipulations based method can also be used to derive the MSAF tests. Since DSOP manipulations are used to derive PDF tests too, the approach is dealt with in a separate chapter.

5.3 Test vector generation

BDD manipulation method

Each ROBDD node maps to one partition that comprises of one 2x1 mux. A valid test vector K must comprise of

- A product term K_α that sensitizes the partition inputs.
- A product term K_ϵ that propagates the partition output to circuit output via path ϵ.
- The primary input x_i that represents the partition

Let $K_\alpha x_i(\bar{x}_i)K_\epsilon$ be the composite test pattern on which the multiple faults at inputs of the mux manifest on circuit C output. The Boolean vector which turns this product into 1 must be included in the partition tests for the mux.

Following is the procedure to find partition test vectors when $f_v^{\bar{x}_i} \neq f_v^{x_i}$ is always true.

Let there be two vectors γ and δ such that $f_v^{\bar{x}_i}(\gamma) = 0, f_v^{x_i}(\gamma) = 1$ and $f_v^{\bar{x}_i}(\delta) = 1, f_v^{x_i}(\delta) = 0$, also $\{\gamma, \delta\} \subseteq K_\alpha$.

There may exist several paths from the mux output function to the circuit output, i.e., from the node representing the partition to root node. K_ϵ is a set of all such paths. Any one path from this set can be selected.

The algorithm for obtaining γ required to generate test vectors a_2 and b_1 from Figure 5.5 is as follows:

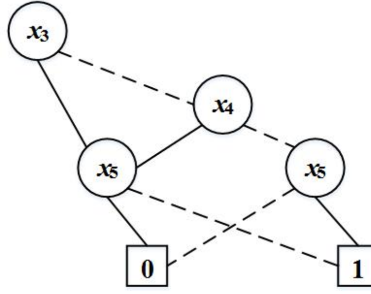


Figure 5.6: ROBDD for x_3

1. ROBDD($f_v^{x_i}$) denotes an ROBDD for function $f_v^{x_i}$ for given internal node v. It's root is the internal node in which the bold edge($x_i = 1$) runs from v. The ROBDD($f_v^{\bar{x}_i}$) root is the internal node in which the dashed edge($\bar{x}_i$) from v runs. The terminal nodes of ROBDD($f_v^{x_i}$) and ROBDD($f_v^{\bar{x}_i}$) coincide with the terminal nodes of the ROBDD of the function f on the corresponding output of circuit C.

2. To get ROBDD($\bar{f}_v^{\bar{x}_i}$), exchange the terminal nodes of the ROBDD($f_v^{\bar{x}_i}$) i.e. replace leaf node 0 with leaf node 1 and vice versa

3. Multiply ROBDD($f_v^{x_i}$) and ROBDD($\bar{f}_v^{\bar{x}_i}$) and let ROBDD R* represent the result.

4. If ROBDD R* is not empty, a path from R* root till its terminal node 1 is represented by the product γ.

Note that the vector δ which satisfies conditions $f_v^{\bar{x}_i}(\delta) = 1$ and $f_v^{x_i}(\delta) = 0$ is computed in the similar way, i.e., by multiplication of ROBDD($f_v^{\bar{x}_i}$) and ROBDD($\bar{f}_v^{x_i}$). Thus all the test patterns shown in Figure 5.5 for the conditions $f_v^{\bar{x}_i}(\gamma) = 0, f_v^{x_i}(\gamma) = 1$ and $f_v^{\bar{x}_i}(\delta) = 1, f_v^{x_i}(\delta) = 0$ can be derived.

Example 5.3.1 *Let us consider a node marked by x_3 on the left side of Fig. 3.3. The ROBDD of x_3 is shown in Fig. 5.6. For this node condition 1 is true. The test vector set derived using the above stated algorithm, for the node marked with x_3 (with respect to Figure 5.5) is as follows:*

$$\begin{aligned}\{a_2, b_1, a_1, b_2\} = &\{\underline{\bar{x}_1 x_2 x_3 \bar{x}_4 \bar{x}_5}, \bar{x}_1 x_2 \bar{x}_3 \bar{x}_4 \bar{x}_5, \\ &\underline{\bar{x}_1 x_2 \bar{x}_3 \bar{x}_4 x_5}, \bar{x}_1 x_2 x_3 \bar{x}_4 x_5\}\end{aligned} \tag{5.1}$$

Table 5.3 shows which ROBDDs are required to calculate the remaining test vectors that encompass the conditions $f_v^{x_i} \subset f_v^{\bar{x}_i}$ and $f_v^{\bar{x}_i} \subset f_v^{x_i}$. For each entry, the path from root node to terminal 1 node represents the corresponding test vector.

Table 5.2: ROBDD operation for other test vectors

Test vector	Figure no.	ROBDD required
a_2	Fig. 5.5(a),Fig. 5.5(c)	ROBDD($f_v^{x_i}$)
a_2	Fig. 5.5(b)	ROBDD($f_v^{\bar{x}_i}$)
a_1	Fig. 5.5(a), Fig. 5.5(b)	ROBDD($\bar{f}_v^{\bar{x}_i}$) * ROBDD($\bar{f}_v^{x_i}$)
b_2	Fig. 5.5(c)	ROBDD($\bar{f}_v^{\bar{x}_i}$) * ROBDD($\bar{f}_v^{x_i}$)

The test set $\{a_1, a_2, b_1, b_2\}$ is the *partition (pseudoexhaustive) test set*. This set is enough to detect all irredundant multiple faults of the partition under test (2x1 mux). The composite test set constitutes of *partition test sets* all nodes.

ROBDD multiplication can be done in time $O(|C|^2)$ [7]; $C = V + E$, where V is the number of ROBDD nodes and E is the number of edges. Time required to find the set of applicable test inputs is proportional to C. Thus the complete test set for all the ROBDD nodes can be computed in time $O(|C|^3)$ [7]. As ROBDD size increases, this method becomes computationally cumbersome. Thus we look at an alternative method for generating test vectors.

DSOP manipulation method

As initial steps of the DSOP manipulation test generation for MSAFs are common with test generation of PDFs, this method is discussed in detail in Chapter 6.

5.4 MSAF detection with proposed approach

It has been shown in Section 5.1 that each sub-circuit of the ROBDD based implementation is considered as a partition and it is testable for multiple faults using *sensitized partitioning* method [37]. The *ab-test* set for each partition is derived either using method described in Section 5.3 or that described in Section 6.2.

In an ROBDD, the rank of literals is known and fixed. Since each node corresponds to a primary input, the rank of primary inputs is known. Without loss of generality, the rank of input variables is taken as $x_1, ..., x_{n-1}, x_n$, where x_1 is the root node and is ranked the highest.

Theorem 8 *A b-test vector for a mux controlled by a variable of rank i detects all stuck-at-1 faults on the OFF path of partitions controlled by variables of rank greater than i.*

Proof *Let us consider a general circuit shown in Figure 5.7. Let h be the path which traverses through the muxes controlled by* x_i *and* x_{i-1} *respectively. We assume that the circuit is tested according to lower to higher ranked variables. Muxes controlled by* x_i *will be tested only if muxes at* x_{i+1} *are found fault free. Hence to generalize it, all muxes controlled by variables of rank lesser than i are assumed to be fault free. Any test vector*

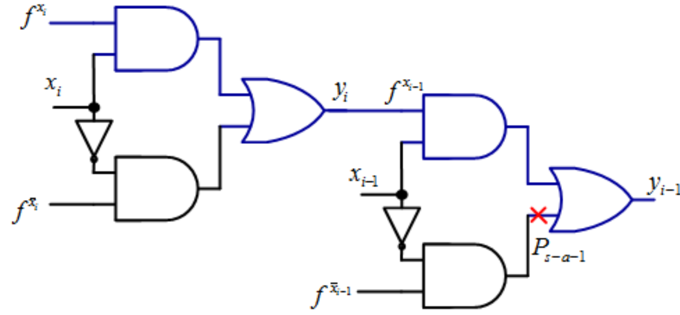


Figure 5.7: Fault propagation on the path

from the partition test set derived for the mux controlled by x_i *will generate* D *or* $\bar{D}$ *at the mux output* y_i*. The test vector also ensures that* y_i *is propagated to the next circuit (as per rank of path h) by applying necessary value to* x_{i-1} *(1 in this case). If mux controlled by* x_{i-1} *is fault free then* D *or* $\bar{D}$ *will be propagated to* y_{i-1}*. Location P lies on the OFF path. A stuck-at-1 fault may exist here due to an actual stuck-at fault or due to multiple faults in the circuit representing* $\bar{x}_{i-1}.f^{\bar{x}_{i-1}}$*. The situations that can occur when the value of* y_i *is* D *or* $\bar{D}$ *are shown in Table 5.3.*

Table 5.3: Fault propagation in presence of P_{s-a-1}

y_i	y_{i-1}	Fault propagated	Mask
D	1	P_{s-a-1}	No
$\bar{D}$	1	No	Yes

For the circuit marked by variable x_i*, when a-test is applied, the output at* y_i *would be* $\bar{D}$*. This would be masked by the* P_{s-a-1} *fault. However, a b-test would generate D. In this case* P_{s-a-1} *would be propagated at the output by one of the tests and circuit would be detected as faulty. Thus a test vector for a sub-circuit controlled by variable* x_i *will detect any stuck-at-1 fault on the OFF path of the muxes controlled by variables having rank grater than i lying on path h. This is applicable to all OFF path s-a-1 faults of path h. Hence it is prerogative to test all muxes as per the rank of the variables beginning from the lowest. This test strategy will detect all the faults in the circuit. Hence proved.*

If net P is stuck-at-0, it acts as a non-controlling value for the OR gate. D or $\bar{D}$ value from the previous circuit will be propagated to the next circuit in incrementing rank of variables. This P_{s-a-0} fault will be detected when *a-test* is applied which has been derived specifically for the partition where it exists.

Lemma 9 *For an ROBDD based implementation, if partition tests are delivered in the ROBDD variable order $x_n, x_{n-1}, ...x_1$, all irredundant multiple faults of the circuit-under-test are detected.*

Proof *Let the partition currently under test be controlled by x_i where $n > i > 1$. It is possible that faults of muxes of variable ranked j ($i > j > 1$) affect the fault propagation of muxes at variables ranked i to the circuit output. Either a faulty output is produced or fault propagation is masked and a fault free output is produced. In case of a faulty output, testing will be stopped. If fault propagation of a mux at variable rank i is masked by faults of a mux at variable rank j testing will continue. Since tests are delivered for variables ranked lower to higher, the faults at variable rank j will eventually be tested for by their corresponding a,b-tests and detected. Thus the tests applied in ROBDD variable order $x_n, x_{n-1}, \cdots, x_1$ will detect all irredundant multiple faults of the circuit.*

An ROBDD with N nodes will have N partitions and hence it will require $4N$ test vectors to test the entire circuit. ROBDD or DSOP manipulations that are required to derive the test vectors can be done in polynomial time. Note that the actual number of test vectors required are less than $4N$.

Given the test set for each sub-circuit be $\{a_1, a_2, b_1, b_2\}$, test vectors a_1, a_2 represent $\bar{x}_i \cdot f_v^{\bar{x}_i} + x_i \cdot f_v^{x_i}$. These terms are sub-sets of the *Disjoint Sum of Products* (DSoP). A DSoP term is unique for an individual path and will comprise either of the *a-tests* for all nodes of the said path.

Let path p represent one of the DSoP terms. Let the number of nodes encountered on the path p be m. One *a-test* is common to all nodes in p. Each node resides on two paths, one controlled by x_i and the other by $\bar{x}_i$. This means that for each node, another DSoP term (the other *a-test*) is also available. So for path p, number of test vectors would be $2m + m = 3m$ where $2m$ would be number of *b-tests* and m are the number of *a-tests*. This is true for all the nodes in the circuit. Thus, a maximum of $3N$ test vectors need to be applied to detect all MSAFs, N being the ROBDD node count.

Chapter summary

: In this chapter, full MSAF testability of each sub-circuit is proved using approaches of pseudoexhaustive testing and MSAF testability of two-level AND-OR circuits. Addition-

ally it is demonstrated that the combined test set of the sub-circuits detects all MSAFs of the complete circuit. In the upcoming chapter, test generation algorithms for PDFs and MSAFs are demonstrated and experimental results of some combinational and sequential benchmark circuits are listed.

Chapter 6

Algorithms to Derive Test Vectors for PDFs and MSAFs

This chapter discusses development of test generation algorithms for PDFs and MSAFs. Test vector calculations are provided for some combinational and sequential benchmark circuits.

From Chapter 4 we see that for detecting PDFs the test vector pair of a path corresponds to the DSOP term it represents and the literal of the path origin takes complementary values in (v_1, v_2). Also from Chapter 5 we infer that the *b-tests* are one bit complement of *a-tests*. This similarity allows us to have a common approach to find initial v_1 vectors/*b-tests*.

Each path form the root node to terminal node 1 constitutes one product term of the *Disjoint Sum of Products* or DSOP expression of the ROBDD. Table 6.1 lists the DSOP terms for ROBDD in Fig. 3.3 and corresponding circuit in Fig. 3.9. These terms can be graphically represented by K-maps as shown in Fig. 6.1(a) and Fig. 6.1(b).

To obtain the v_1 vectors/*b-tests*, we need to complement the bits in each DSOP terms one by one. This can be done using matrices. For a circuit with n literals, we create a *test generator* matrix G of size $n \times n$ as shown in Fig. 6.2. This in an *identity* matrix. The rows have a complemented bit in incremental positions. D is a column matrix consisting of one DSOP term. A bitwise EX-OR operation of G with D will give initial values of v_1 vectors/*b-tests* corresponding to each literal as shown in Fig. 6.3.

6.1 Derivation of PDF tests

Each DSOP term represents a circuit path from terminal node to root node. If a circuit is represented by d DSOPs then it will have d number of paths. Let the paths be represented

Table 6.1: DSOP List

Number	Product Term
1	$x_1\bar{x}_2\bar{x}_3$
2	$x_1\bar{x}_2x_3\bar{x}_4x_5$
3	$x_1\bar{x}_2x_3x_4\bar{x}_5$
4	$x_1x_2\bar{x}_4x_5$
5	$x_1x_2x_4\bar{x}_5$
6	$\bar{x}_1\bar{x}_2\bar{x}_4x_5$
7	$\bar{x}_1\bar{x}_2x_4\bar{x}_5$
8	$\bar{x}_1x_2\bar{x}_3\bar{x}_4x_5$
9	$\bar{x}_1x_2\bar{x}_3x_4\bar{x}_5$
10	$\bar{x}_1x_2x_3\bar{x}_5$

	$\bar{x}_4\bar{x}_5$	$\bar{x}_4x_5$	x_4x_5	$x_4\bar{x}_5$
$\bar{x}_2\bar{x}_3$	0	1	0	1
$\bar{x}_2x_3$	0	1	0	1
x_2x_3	1	0	0	1
$x_2\bar{x}_3$	0	1	0	1

(a) K-map for $\bar{x}_1$

	$\bar{x}_4\bar{x}_5$	$\bar{x}_4x_5$	x_4x_5	$x_4\bar{x}_5$
$\bar{x}_2\bar{x}_3$	1	1	1	1
$\bar{x}_2x_3$	0	1	0	1
x_2x_3	0	1	0	1
$x_2\bar{x}_3$	0	1	0	1

(b) K-map for x_1

Figure 6.1: K-maps for function f

by P_1 to P_d. Each path $P_i = f(x_1, ...x_n)$ where $x_1, ...x_n$ are input variables. Each path P_i has sub-paths.

$P_{i_2} = f(x_1, x_2)$ path begins at x_2, ends at x_1

$P_{i_3} = f(x_1, x_2, x_3)$ path begins at x_3, ends at x_1

$\vdots$

$P_{i_n} = f(x_1, x_2, \cdots, x_n)$ path begins at x_n, ends at x_1

For a rising transition, the vector v_2 should force the output to 1. This can be achieved by considering the DSOP term P_i as v_2. The term P_i will act as v_2 for all sub-paths P_{i_2} to P_{i_n}. The v_1 vector is supposed to drive the output to 0. This means v_1 has to be orthogonal

Let one DSoP term D be $f\{x_1, x_2, \cdots, x_n\} = \{10\cdots111\}$

$$\mathrm{G}(n \times n) = \begin{bmatrix} 0000\cdots0001 \\ 0000\cdots0010 \\ \vdots \\ 0001\cdots0000 \\ 0010\cdots0000 \\ 0100\cdots0000 \\ 1000\cdots0000 \end{bmatrix} \quad D = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$G \oplus D = v_1 \text{ vectors} / \textit{b-tests}. \tag{6.1}$$

Figure 6.2: Generating v_1 vectors / *b-tests* set for one path

$$v_1\text{vectors/b-tests} = \begin{bmatrix} 10\cdots110 \\ 10\cdots101 \\ 10\cdots011 \\ \vdots \\ 01\cdots111 \end{bmatrix} \quad \begin{matrix} \leftarrow v_1 \textit{vector} / \textit{b-test}(x_n) \\ \leftarrow v_1 \textit{vector} / \textit{b-test}(x_{n-1}) \\ \leftarrow v_1 \textit{vector} / \textit{b-test}(x_{n-2}) \\ \vdots \\ \leftarrow v_1 \textit{vector} / \textit{b-test}(x_1) \end{matrix}$$

Figure 6.3: v_1 *vector / b-test* set for one DSOP term

to all v_2 vectors (DSOP terms). v_1 vectors for sub-paths P_{i_2} to P_{i_n} can be derived by complementing the value of the variable that originates the path. This is shown in Figures 6.2 and 6.3.

Example 6.1.1 *Let us consider the* 8th *DSOP term from Table 6.1 i.e.,* $\bar{x}_1x_2\bar{x}_3\bar{x}_4x_5$. *The* v_1 *tests are generated in Fig. 6.4. We see that some of the generated vectors are not orthogonal to some* v_2 *vectors, i.e., they will not be valid* v_1 *tests and have to be discarded. This DSOP term cannot originate* v_1 *vector for* $\bar{x}_1x_2$. *The term* $\bar{x}_1x_2$ *exists in the* 9th *and the* 10th *terms, valid* v_1*-tests can be created from them.*

This example indicates that some amount of redundancy exists in the approach of generating v_1 vectors. This also means that if a sub-path exists in more than one paths, it can have more than one v_1 test vectors. As the ranking of the variable that originates a sub-path, goes higher, the probability of having one than one v_1 vectors is higher. This is due to ROBDD construction.

$$\text{G (nxn)} = \begin{bmatrix} 00001 \\ 00010 \\ 00100 \\ 01000 \\ 10000 \end{bmatrix} \qquad D = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\text{b-tests} = \begin{bmatrix} 01000 \\ 01011 \\ 01101 \\ 00001 \\ 11001 \end{bmatrix} \quad \begin{matrix} \leftarrow v_1 \textit{vector / b-test}(x_5) \\ \leftarrow v_1 \textit{vector / b-test}(x_4) \\ \leftarrow v_1 \textit{vector / b-test}(x_3) \\ \leftarrow \textit{another } v_2 \textit{ vector / a-test(DSOP term 6)} \\ \leftarrow \textit{another } v_2 \textit{ vector / a-test(DSOP term 4)} \end{matrix}$$

Figure 6.4: Test generation for Example 6.1.1

Let P represent the set of DSOP terms. The procedure to derive one set of v_1 vectors for one term P_i in the ROBDD based design is as follows:

1. For P derive $V^*_{jk} \forall j = 1, 2, \cdots, m$ and $k = 1, \cdots, r_j$, where m is the number of DSOP terms and r_j is the number of literals in each term. This is done by performing EX-OR operation of P_i and $n \times n$ unity matrix, n is the number of input literals of the circuit. (Note: The last vector corresponding to x_1 needs to be ignored since P_{i_2} is the last and smallest path).

2. For every $V^*_{ab} \supseteq V^*_{jt}$, replace V^*_{ab} with V^*_{jt} from the list. This will minimise the number of don't care terms.

3. In V^*_{jk}, fill don't care terms with all combinations resulting in V^*_x.

4. For every $V^*_y \subset P$, delete V^*_y from V^*_x. This eliminates any term in V^*_x that matches any of the DSOP terms.

5. For every term in V^*_{jk} retain one term from V^*_x such that $V^*_{jk} \supset V^*_x$. This will be the list of fully specified valid v_1 vectors for P_i.

Example 6.1.2 *: Reconsider term 10 from DSOP list i.e., $\bar{x}_1 x_2 x_3 \bar{x}_5$ i.e.,011-0. The v_1 tests for this term are derived in Fig. 6.5. v_2 can be* 01100 *or* 01110*. For product term 10 the*

test vector pairs for each sub-path are

$$(v_1, v_2) = (01101, 01100)\ \textit{for path}\ P_{10_5}$$

$$(v_1, v_2) = (01000, 01100)\ \textit{for path}\ P_{10_3}$$

$$(v_1, v_2) = (00100, 01100)\ \textit{for path}\ P_{10_2}$$

$$V_{jk} = \begin{bmatrix} 011-1 \\ 010-0 \\ 001-0 \end{bmatrix} \quad V_x = \begin{bmatrix} 01101 \\ 01111 \\ 01000 \\ \mathbf{01010} \\ 00100 \\ \mathbf{00110} \end{bmatrix} \quad V_x(updated) = \begin{bmatrix} \mathit{01101} \\ \mathit{01111} \\ 01000 \\ 00100 \end{bmatrix} \quad V_x(final) = \begin{bmatrix} 01101 \\ 01000 \\ 00100 \end{bmatrix}$$

Figure 6.5: Example 6.1.2: v_1 *tests*

If path begins at a node x_i that has robust tests then the test vector pair (v_1, v_2) can be derived as follows:

- Select the product term in the DSOP that includes path-to-be-tested.

  The test vector v_2 is the selected term itself.

- Test vector v_1 is the selected product term with the variable x_i complemented. This would be the test pair for rising transition. For falling transition, values of v_1 and v_2 are interchanged.

If the path-to-be-tested begins at a node x_i that has one non-robust test then:

- First derive the test vector pair using the method described above for the companion path that can be robustly tested.

  For the non-robust testable path, select the product term in DSOP that includes the path. This is v_2.

- Application of v_2 will detect if a PDF exists on the path.

Although v_1 can be found by complementing the value of x_i in the product term, it will a part of the DSOP and does not have contribution in testing the path. Since in this case v_1 does not exist, falling transition cannot be tested for.

If the nodes have to be classified as robust testable or non-robust testable, test generation complexity increases. To simplify the test generation process we consider the steps

for test generation given for robust testable nodes for all types of nodes. This allows us to use DSOP terms derived from ROBDD to generate the test patterns directly. Thus the test patterns can be generated at design time.

The test generation complexity of the DSOP based method is $O(m \times n)$; m is the number of DSOP terms, n is the number of inputs.

6.2 Derivation of MSAF tests

Valid *b-tests* need to be orthogonal to all *a-tests* i.e., all DSOP terms of the function under consideration. The initial *b-tests* are derived as per matrix operations shown in Figures 6.2 and 6.3.

Total *ab-tests* = all *a-tests* i.e., all DSOP terms + all *b-tests* i.e., sum of *b-tests* created by each DSOP term.

Example 6.2.1 *Let us consider the 8th DSOP term from Table 6.1, i.e., $\bar{x}_1 x_2 \bar{x}_3 \bar{x}_4 x_5$. The b-test generation is shown in Fig. 6.4. The last two entries in the b-test list correspond to some other a-tests. This can be verified from the K-map of Fig. 6.1(a). This means that this DSOP term cannot originate b-tests for $\bar{x}_1 x_2$. Since the term $\bar{x}_1 x_2$ exists in the 9th and the 10th terms, valid b-tests can be created from them.*

This example indicates that redundancy has to be checked everytime a set of *b-tests* are generated.

A method to derive minimal *b-tests* for *irredundant sum of products* was presented in Kohavi et. al.[29]. The *b-tests* derived for ROBDD based DSOP represented circuits are not minimal since each variable represents a sub-circuit. Additional steps are required in ROBDD based system to eliminate redundancy.

Let P represent the set of DSOP terms. The procedure to derive *b-tests* B for the ROBDD based design is as follows:

1. For every term in P derive $B^*_{jk} \forall j = 1, 2, \cdots, m$ and $k = 1, 2, \cdots, r_j$, where m is the number of DSOP terms and r_j is the number of literals in each term. This is done by performing EX-OR operation of each DSOP term and $n \times n$ unity matrix, n is the number of input literals of the circuit.

2. For every $B^*_{ab} \supseteq B^*_{jt}$, remove B^*_{ab} from the list. Every test for B^*_{jt} is a test for B^*_{ab}, while the converse is not true. This gives the number of valid *b-tests*.

3. In the updated B^*_{jk}, fill don't care terms with all combinations resulting in B^*_x.

4. For every $B_y^* \subset P$, delete B_y^* from B_x^*. This eliminates any term in B_x^* that matches any of the DSOP terms.

5. For every term in B_{jk}^* retain one term from B_x^* such that $B_{jk}^* \supset B_x^*$. This will be the list of fully specified valid *b-tests*.

This can be done quite early in the design cycle since the DSOP terms are available at design time. It should be noted that this test generation strategy holds true only for ROBDD based designs.

Example 6.2.2 *Consider term 10 from DSOP list i.e., $\bar{x}_1 x_2 x_3 \bar{x}_5$ i.e.,011-0. The b-tests for this term are derived in Fig. 6.6. In B_x matrix, the terms highlighted in bold are ones that match other DSOP terms and are eliminated. In B_x(updated) matrix, terms highlighted in italics are tests corresponding to the same variable so either of them can be retained. The DSOP term itself is the a-test* = 01100 *or* 01110*. The ab-test set for this DSOP term is* $\{01100, 01101, 01000, 00100, 11100\}$.

$$B_{jk} = \begin{bmatrix} 011-1 \\ 010-0 \\ 001-0 \\ 111-0 \end{bmatrix} \quad B_x = \begin{bmatrix} 01101 \\ 01111 \\ 01000 \\ \mathbf{01010} \\ 00100 \\ \mathbf{00110} \\ 11100 \\ \mathbf{11110} \end{bmatrix} \quad B_x(updated) = \begin{bmatrix} \mathit{01101} \\ \mathit{01111} \\ 01000 \\ 00100 \\ 11100 \end{bmatrix} \quad B_x(final) = \begin{bmatrix} 01101 \\ 01000 \\ 00100 \\ 11100 \end{bmatrix}$$

Figure 6.6: Example 6.2.2: *b-tests*

The complete test set derived by combining all the partial test sets and eliminating redundancies is capable of detecting all multiple stuck-at faults. Single stuck-at faults are part of the MSAFs and are automatically covered.

6.3 Experimental Results

For ROBDD based implementation the synthesis tool is to be directed to avoid optimization so as to maintain the structure. We have used *ABC* tool[38] to generate verilog representation of BDD based circuits. *Synopsys Design Compiler* is used to estimate area, power consumption and maximum path delays. The implementation is done using

gscl45nm library. The *Design Compiler* is guided to perform unconstrained syhthesis for original benchmark circuits. For the BDD based synthesis *Design Compiler* is forced to use multiplexers, inverters and AND,OR gates.

Python programs are written to implement the PDF and MSAF test generation algorithms. SSAF simulation of ROBDD designs is done using *ABC*.

For a few combinational benchmark circuits, data regarding test vectors and logic gates count is given in Table 6.2. Column I indicates the benchmark circuit under consideration, column II indicates the primary inputs(PIs) and outputs(POs) of the circuit. ROBDD nodes and corresponding DSOP terms are listed in columns III and IV respectively. Number of SSAF tests are listed in column V. These tests have been generated for full SSAF testability using fault simulator of *ABC*. MSAF tests and PDF tests which guarantee complete testability are listed in column VI and VII respectively. These test vectors have been generated using the DSOP based algorithms.

Sequential circuits have pseudo-primary inputs(PPIs) and pseudo-primary outputs(PPOs) in addition to the primary inputs and outputs due to presence of flip flops. ROBDD construction in sequential circuits includes the PPIs and PPOs in addition to the primary inputs and outputs. Table 6.3 indicates test vectors for combinational section of some sequential benchmark circuits. Column III in Table 6.3 lists all primary and pseudoprimary, inputs and outputs. Column IV and V list ROBDD nodes and corresponding DSOP terms respectively. Columns VI, VII, and VIII list the SSAF,MSAF and PDF test sets respectively.

It should be noted that the circuits have multiple outputs. All the outputs do not have all the inputs in their fan-in cones. If all the inputs are considered for all outputs, the x-filling process of step 3 in section 6.2 may increase exponentially, especially for sequential circuits. Hence while deriving *b-tests* only one output and corresponding inputs are taken into consideration at a time. The process is repeated for all outputs.

Since fault simulation is not available for MSAFs, test compaction is not possible. As a result the test data volume of MSAFs is larger for most of the circuits as compared to the SSAF test data volume. Nevertheless, if unoptimized synthesis for ROBDD based designs is done, the tests generated at design time are valid MSAF tests for the synthesized circuit.

While deriving the PDF tests, all sub-paths of one path(root node to leaf node) are considered. Thus the PDF test volume is large. Since the PDF tests have to be applied in a sequence, it is not possible to reduce the test data volume. If one decides to test only longest paths, the test data volume may be clipped.

Table 6.4 lists the area comparisons of the original benchmark circuit description and BDD based synthesis using proposed method, for combinational benchmark circuits. Columns II and III indicate area considerations before and after BDD based synthesis. Column IV indicates the amount of increase/decrease in area. The amount of power consumtion before and after BDD based synthesis is listed in columns V and VI. Increase/decrease in power is indicated in column VII. The maximum path delays are compared in columns VIII and IX. Column X indicates the amount of increase/decrease in path delays after BDD based synthesis. Similar comparisons for sequential benchmark circuits are depicted in Table 6.5

It should be noted that the test methodology presented in this work is restricted to circuits which can be represented by ROBDDs. In circuits, where the number of paths are very large and generating ROBDDs is increasingly difficult, the test generation approach of this work does not hold good.

Table 6.2: Combinational Circuit Test Generation

I Combinational benchmarks	II PI/PO	III Robdd nodes	IV DSOP terms	V SSAF tests[a]	VI MSAF tests[b]	VII PDF tests[c]
b1	3/ 4	6	6	4	10	6
C17	5/ 2	9	7	6	6	7
decod	5/ 16	80	16	17	48	64
rd53	5/ 3	21	35	12	34	42
xor5	5/ 1	5	16	10	32	16
cm138a	6/ 8	48	48	10	7	24
5xp1	7/ 10	83	74	21	81	154
con1	7/ 2	16	9	11	15	15
rd73	7/ 3	35	147	28	132	211
z4ml	7/ 4	28	59	15	55	112
f51m	8/ 8	58	78	40	94	163
misex1	8/ 7	61	33	18	28	73
mm4a	8/ 4	439	508	184	1203	2473
rd84	8/ 4	50	294	28	273	473
sqrt8	8/ 4	39	42	22	49	90
9sym	9/ 1	24	148	29	138	356

[a] Test vectors for full SSAF testability using commercial ATPG

[b] Test vectors for full MSAF testability in proposed synthesis using DSOP based algorithm

[c] Test vectors for full PDF testability in proposed synthesis using DSOP based algorithm

Table 6.2: Combinational Circuit Test Generation

I Combinational benchmarks	II PI/PO	III Robdd nodes	IV DSOP terms	V SSAF tests[a]	VI MSAF tests[b]	VII PDF tests[c]
clip	9/ 5	141	150	66	191	409
ldd	9/ 19	142	61	27	38	133
alu2	10/ 6	177	156	131	322	558
x2	10/ 7	46	28	21	28	45
sao2	10/ 4	116	75	43	173	274
cm85a	11/ 3	50	48	19	118	160
cm151a	12/ 2	36	17	24	43	57
alu4	14/ 8	760	635	480	1425	2995
cu	14/ 11	78	22	32	64	92
misex3	14/ 14	871	1306	272	1972	6694
cm163a	16/ 5	35	27	21	29	50
cmb	16/ 4	48	26	18	30	34
pdc	16/ 40	4706	6062	230	2026	26198
pm1	16/ 13	62	37	17	45	52
t481	16/ 1	30	481	51	630	2587
table5	17/ 15	1819	551	308	1247	4289
tcon	17/ 16	24	24	17	21	16
vda	17/ 39	1161	573	125	322	2028
pcle	19/ 9	107	45	29	90	129
sct	19/ 15	124	73	35	92	125
cc	21/ 20	88	45	26	50	79
cm150a	21/ 1	32	17	24	52	65
duke2	22/ 29	642	201	143	357	1206
cordic	23/ 2	90	1180	57	6604	8217
ttt2	24/ 21	243	138	59	227	348
i1	25/ 13	59	27	26	44	61
vg2	25/ 8	269	110	155	293	511
lal	26/ 19	146	102	60	115	172
pcler8	27/ 17	153	61	51	100	181

[a] Test vectors for full SSAF testability using commercial ATPG

[b] Test vectors for full MSAF testability in proposed synthesis using DSOP based algorithm

[c] Test vectors for full PDF testability in proposed synthesis using DSOP based algorithm

Table 6.2: Combinational Circuit Test Generation

I Combinational benchmarks	II PI/PO	III Robdd nodes	IV DSOP terms	V SSAF tests[a]	VI MSAF tests[b]	VII PDF tests[c]
c8	28/ 18	116	79	43	102	149
frg1	28/ 3	103	119	96	172	532
term1	34/ 10	227	255	122	780	1129
count	35/ 16	200	184	73	209	379
unreg	36/ 16	96	48	48	68	112
b9	41/ 21	220	138	84	176	295
seq	41/ 45	2536	1467	424	2472	9986
cht	47/ 36	181	81	71	110	164
x1	51/ 35	671	304	337	817	1098
e64	65/ 65	2144	65	97	2082	2080
example2	85/ 66	746	367	237	796	1281
x4	94/ 74	628	526	270	1008	1387
o64	130/ 1	130	65	69	81	98
x3	135/ 99	1070	657	393	1737	2377
i6	138/ 67	613	238	220	214	515

[a] Test vectors for full SSAF testability using commercial ATPG

[b] Test vectors for full MSAF testability in proposed synthesis using DSOP based algorithm

[c] Test vectors for full PDF testability in proposed synthesis using DSOP based algorithm

Table 6.3: Sequential Circuit Test Generation-Combinational logic

I Sequential benchmarks	II PI/PO	III PI+PPI/ PO+PPO	IV Robdd nodes	V DSOPs	VI SSAF tests[a]	VII MSAF tests[b]	VIII PDF tests[c]
s298	3/6	17/20	114	70	48	48	140
s382	3/6	24/27	213	167	64	191	507
s400	3/6	24/27	213	167	67	191	507
s526	3/6	24/27	195	144	79	128	344
s27	4/1	7/4	22	18	11	9	18
s444	4/6	25/27	220	167	75	190	506
s386	7/7	13/13	198	51	51	114	242
s1488	8/19	92/79	632	339	155	217	1006
s1494	8/19	14/25	632	283	153	215	1006
s344	9/11	24/26	135	249	49	276	701
s208	11/2	19/10	76	30	53	68	117
s1196	15/14	33/32	1149	1136	378	1906	5576
s1238	15/14	33/32	1200	1136	382	1928	5576
s953	16/23	45/52	653	235	165	331	1174
s420.1	18/1	34/17	206	169	114	252	425
s820	18/19	23/24	377	127	107	195	519
s510	19/7	25/13	229	110	85	85	298
s1512	29/21	86/78	980	1948	446	17040	20953
s641	35/24	54/43	1157	925	421	1246	4107
s713	35/23	54/42	1130	912	387	1208	4074
s838	35/ 2	67/34	352	114	209	567	903

[a] Test vectors for full SSAF testability using commercial ATPG

[b] Test vectors for full MSAF testability in proposed synthesis using DSOP based algorithm

[c] Test vectors for full PDF testability in proposed synthesis using DSOP based algorithm

Table 6.4: Area, power and path delay comparisons of combinational circuit synthesis

I Comb. BM Circuits	II Original Area μm^2	III Proposed Synth. Area μm^2	IV % Increase	V Original Power mW	VI Proposed Synth. Power mW	VII % Increase	VIII Original Path delay ns	IX Proposed Synth. Path Delay ns	X % Increase
5xp1	219.163095	174.580	-20.34	0.066	0.054	-18.35	0.39	0.4	2.56
9sym	107.939	161.439	49.57	0.070	0.072	2.98	0.5	0.7	40
alu2	699.726	725.538	3.69	0.211	0.216	2.23	1.52	0.82	-46.05
alu4	2072.898	2545.483	22.8	0.485	0.782	61.27	0.95	1.26	32.63
b1	13.140	22.057	67.86	0.004	0.005	31.02	0.09	0.14	55.56
b9	210.716	283.457	34.52	0.043	0.061	42.94	0.28	0.61	117.86
C17	14.548	15.487	6.45	0.003	0.003	15.54	0.09	0.15	66.67
c8	207.900	229.018	10.16	0.049	0.053	8.59	0.44	0.53	20.45
cc	105.123	106.062	0.89	0.019	0.019	2.59	0.29	0.3	3.45
cht	299.413	288.619	-3.61	0.068	0.061	-10.96	0.28	0.31	10.71
clip	275.948	374.501	35.71	0.079	0.123	56.77	0.49	0.56	14.29
cm138a	35.197	50.215	42.67	0.004	0.007	69.02	0.21	0.27	28.57
cm150a	82.127	92.921	13.14	0.023	0.026	11.93	0.31	0.35	12.9
cm151a	30.035	47.869	59.37	0.010	0.014	43.52	0.19	0.22	15.79
cm163a	66.171	72.741	9.93	0.015	0.018	17.09	0.31	0.42	35.48

Table 6.4: Area, power and path delay comparisons of combinational circuit synthesis

I	II	III	IV	V	VI	VII	VIII	IX	X
Comb. BM Circuits	Original Area μm^2	Proposed Synth. Area μm^2	% Increase	Original Power mW	Proposed Synth. Power mW	% Increase	Original Path delay ns	Proposed Synth. Path Delay ns	% Increase
cm85a	94.799	104.654	10.4	0.025	0.028	11.75	0.31	0.48	54.84
cmb	56.785	63.825	12.4	0.005	0.007	49.92	0.29	0.25	-13.79
con1	31.912	37.544	17.65	0.007	0.010	29.93	0.12	0.2	66.67
cordic	217.755	154.869	-28.88	0.063	0.046	-26.04	0.47	0.44	-6.38
count	234.181	232.773	-0.6	0.051	0.050	-2.22	0.91	1.16	27.47
cu	89.636	94.799	5.76	0.016	0.016	3.22	0.21	0.38	80.95
decod	79.312	75.088	-5.33	0.011	0.012	12.15	0.16	0.24	50
duke2	723.191	771.529	6.68	0.088	0.129	46.89	0.6	0.71	18.33
e64	411.576	459.445	11.63	0.020	0.031	55.52	2.05	1.51	-26.34
example2	536.410	712.397	32.81	0.088	0.139	57.97	0.49	0.58	18.37
f51m	250.606	202.738	-19.1	0.074	0.063	-15.86	0.42	0.47	11.9
frg1	267.501	121.079	-54.74	0.066	0.028	-57.22	0.53	0.63	18.87
i1	80.720	81.189	0.58	0.013	0.013	0.23	0.27	0.3	11.11
i6	671.568	1817.130	170.58	0.184	0.401	118.14	0.84	1.08	28.57
lal	156.277	211.654	35.44	0.034	0.043	27.83	0.46	0.42	-8.7

Table 6.4: Area, power and path delay comparisons of combinational circuit synthesis

I Comb. BM Circuits	II Original Area μm^2	III Proposed Synth. Area μm^2	IV % Increase	V Original Power mW	VI Proposed Synth. Power mW	VII % Increase	VIII Original Path delay ns	IX Proposed Synth. Path Delay ns	X % Increase
ldd	165.194	193.821	17.33	0.034	0.048	42.6	0.3	0.37	23.33
misex1	109.816	118.264	7.69	0.022	0.027	24.21	0.3	0.35	16.67
misex3	2237.622	1372.702	-38.65	0.419	0.292	-30.37	0.97	0.96	-1.03
mm4a	460.383	1448.729	214.68	0.060	0.222	268.59	1.31	1.24	-5.34
o64	273.602	302.698	10.63	0.076	0.072	-5.68	0.3	0.34	13.33
pcle	113.571	202.738	78.51	0.025	0.027	7.54	0.41	0.61	48.78
pcler8	183.027	150.175995	-17.95	0.024	0.032	31.44	0.43	0.49	13.95
pdc	56.785	63.825	12.4	0.005	0.007	50.46	0.29	0.25	-13.79
pm1	81.189	91.044	12.14	0.013	0.015	14.56	0.2	0.25	25
rd53	40.829	65.702	60.92	0.020	0.023	16.87	0.25	0.41	64
rd73	69.456	107.000	54.05	0.038	0.043	15.02	0.32	0.48	50
rd84	92.921	137.505	47.98	0.054	0.057	6.49	0.39	0.56	43.59
sao2	292.374	293.782	0.48	0.064	0.063	-1.84	0.45	0.6	33.33
sct	109.816	135.628	23.5	0.021	0.027	29.85	0.39	0.51	30.77
seq	4203.520	3892.374	-7.4	0.448	0.896	99.82	1.12	1.3	16.07

Table 6.4: Area, power and path delay comparisons of combinational circuit synthesis

I	II	III	IV	V	VI	VII	VIII	IX	X
Comb. BM Circuits	Original Area μm^2	Proposed Synth. Area μm^2	% Increase	Original Power mW	Proposed Synth. Power mW	% Increase	Original Path delay ns	Proposed Synth. Path Delay ns	% Increase
sqrt8	91.044	102.307	12.37	0.026	0.029	10.64	0.28	0.41	46.43
t481	71.334	91.044	27.63	0.019	0.015	-24.08	0.22	0.25	13.64
table5	1406.492	2743.528	95.06	0.142	0.539	279.24	0.92	1.26	36.96
tcon	41.298	41.298	0	0.009	0.009	0	0.06	0.06	0
term1	313.492	309.269	-1.35	0.078	0.077	-1.6	0.49	0.56	14.29
ttt2	341.650	356.199	4.26	0.072	0.081	12.3	0.44	0.62	40.91
unreg	167.540	223.856	33.61	0.034	0.047	38.86	0.37	0.37	0
vda	1496.128	1826.516	22.08	0.297	0.432	45.67	0.76	0.74	-2.63
vg2	188.189	308.330	63.84	0.039	0.075	90.3	0.5	0.67	34
x1	626.515	678.608	8.31	0.119	0.144	21.33	0.52	0.49	-5.77
x2	80.720	107.000	32.56	0.015	0.025	62.54	0.21	0.3	42.86
x3	1442.628	1887.525	30.84	0.329	0.432	31.12	0.49	0.94	91.84
x4	647.165	686.117	6.02	0.135	0.146	8.28	0.62	0.58	-6.45
xor5	17.833	36.605	105.26	0.009	0.013	39.28	0.17	0.32	88.24
z4ml	50.215	80.250	59.81	0.020	0.024	21.42	0.27	0.39	44.44

Table 6.5: Area, power and path delay comparisons of combinational logic synthesis of sequential benchmark circuits

I	II	III	IV	V	VI	VII	VIII	IX	X
Seq. BM Circuits	Original Area [a] μm^2	Proposed Synth. Area[a] μm^2	% Increase	Original Power[a] mW	Proposed Synth. Power[a] mW	% Increase	Original Path delay[a] ns	Proposed Synth. Path Delay[a] ns	% Increase
s1196	1118.81	346.34	-69.04	0.206	0.015	-92.63	1.03	0.1	-90.29
s1238	1097.69	346.34	-68.45	0.213	0.015	-92.88	0.88	0.1	-88.64
s1488	272.19	354.32	30.17	0.014	0.016	10.06	0.1	0.1	0
s1494	272.19	339.30	24.66	0.014	0.015	6.61	0.1	0.1	0
s1512	198.51	300.35	51.3	0.011	0.012	13.34	0.76	0.83	9.21
s208	244.97	346.34	41.38	0.015	0.015	-0.2	0.23	0.1	-56.52
s27	362.30	344.94	-4.79	0.015	0.016	5	0.1	0.1	0
s298	672.51	1746.73	159.73	0.173	0.392	126.6	0.75	0.82	9.33
s344	252.95	346.34	36.92	0.015	0.015	-2.11	0.24	0.1	-58.33
s382	316.78	344.94	8.89	0.019	0.016	-15.89	0.1	0.1	0
s386	272.19	347.28	27.59	0.014	0.014	-2.51	0.1	0.1	0
s400	286.74	344.94	20.29	0.016	0.016	-1.29	0.1	0.1	0
s420.1	198.51	366.52	84.63	0.011	0.020	87.92	0.76	0.74	-2.63
s444	272.66	348.69	27.88	0.014	0.014	-1.38	0.1	0.1	0

[a] Combinational logic

Table 6.5: Area, power and path delay comparisons of combinational logic synthesis of sequential benchmark circuits

I	II	III	IV	V	VI	VII	VIII	IX	X
Seq. BM Circuits	Original Area [a] μm^2	Proposed Synth. Area[a] μm^2	% Increase	Original Power[a] mW	Proposed Synth. Power[a] mW	% Increase	Original Path delay[a] ns	Proposed Synth. Path Delay[a] ns	% Increase
s510	198.51	305.98	54.14	0.011	0.013	17.77	0.76	0.81	6.58
s526	359.95	344.94	-4.17	0.015	0.016	5.86	0.1	0.1	0
s641	341.18	302.70	-11.28	0.040	0.012	-69.39	1.14	0.83	-27.19
s713	341.18	302.70	-11.28	0.040	0.012	-69.39	1.14	0.83	-27.19
s820	198.51	321.47	61.94	0.011	0.012	11.72	0.76	0.71	-6.58
s838	548.61	594.13	8.3	0.012	0.015	18.35	1.34	0.92	-31.34
s953	817.05	346.34	-57.61	0.078	0.015	-80.59	0.1	0.1	0

[a] Combinational logic

Chapter Summary

DSOP based algorithms to derive PDF and MSAF test vectors are described in this chapter. The derived test vectors are listed for some combinational benchmark circuits and combinational logic of sequential benchmark circuits. Area, power and maximum path delays of these circuits are compared for before and after BDD based synthesis. The next section concludes the thesis and gives direction for future work.

Chapter 7

Conclusion and Future Scope

It has become necessary to bring *delay* testability and *multiple stuck-at* fault testability to the forefront in VLSI testing as *single stuck-at* fault tests are not enough to cover all defects. Single stuck-at fault tests cover hard short and open defects. The *path delay fault* model covers these defects and additionally covers resistive opens, shorts, coupling faults and aging. Multiple stuck-at fault model covers coupling and bridging faults.

Testing is the most expensive process in chip manufacturing. There is always a need for test methodology that guarantees fault coverage and demonstrates ease of test generation. There are two approaches to pursue such a test methodology. One can either improve test generation algorithms or can develop synthesis approaches that guarantee testability and have ease of test generation, or both.

In this thesis we have investigated testability issues in combinational logic circuits. The main contributions of our work is to propose a *ROBDD* based testable synthesis.

The motivation of the work in this thesis is a combinational circuit synthesis which demonstrates:

- No redundant paths, all irredundant paths should be testable for path delays using either robust or validatable non-robust tests
- Entire circuit is testable for multiple stuck-at faults
- Ease of test generation for PDFs and MSAFs, no additional control signals required

7.1 Thesis summary

Binary Decision Diagram based implementations have higher degree of testability than other circuits. Since the variable ordering is fixed for the *Reduced Ordered Binary Decision Diagram*(ROBDD), the representation of the Boolean function is *canonical* and the

product terms are *irredundant*. In this thesis, internal nodes are replaced by 2x1 multiplexers. Transformation is applied only to nodes that have one edge connected to leaf nodes. The transformation maintains testability and aids in test generation without the requirement of additional control signals.

Each path is now represented by a *Disjoint Sum of Products* term. Any term which is orthogonal to the DSOP term will force the output to 0. Since the DSOP terms are irredundant, no redundant paths exist in the circuit.

We establish complete delay testability of each path by proving the following:

- All single stuck-at faults of the path under test are detectable
- The stuck-at fault corresponding to the delay fault of the literal(where the path begins) is detectable. The corresponding test vector constitutes v_2 of the test pair (v_1, v_2).
- The variable x_i from where the path begins takes complementary values for v_1, v_2 vectors.

Due to structure of the ROBDD, paths encompassing certain nodes are not robustly testable. In such cases we demonstrate how these paths can be tested non-robustly. Thus we have established conditions for robust and non-robust testabilty. Additionally we demonstrate that for our BDD based synthesis, the non-robust tests would not be invalidated if the robust testable paths are covered first. Validatable non-robust tests are as good as robust tests. Thus our synthesis method guarantees complete path delay testability using robust and validatable non-robust tests. It is also possible to know the path lengths at design time since the muxes in a path are directly proportional to the number of inputs that constitute the path.

Good testability property of a synthesis method depends on the fault models it covers. Our testable synthesis covers the path delay fault model which inherently includes the stuck-at fault model and transition fault model. We now investigate testablity of the synthesis for the multiple stuck-at fault model.

Using the foundations of pseudoexhaustive testing, we investigate the MSAF testability of one 2x1 mux. Pseudoexhaustive testing involves partitioning a circuit into smaller sections and testing those sections exhaustively. We apply this technique to exhaustively test one sub-circuit(one 2x1 mux). Since the circuit is testable for path delay fauts, sensitizing and propagation paths already exist. However, due to the contitions existing on the nodes described earlier, it is not possible to deliver exhaustive input conditions to the sub-circuit. So we augment this process with another approach.

We apply the MSAF testability property of 2 level irredundant AND-OR implementations to our sub-circuit. We prove that the deliverable test vectors to each node detect all MSAF combinations of the input nets. Since our implementation is irredundant, internal MSAFs are covered too. Thus all MSAFs of the sub-circuit are covered. The classical BDD manipulation method to derive test vectors for each node is found to be computationally expensive; The test generation complexity is $O(|C|^3)$, where C is the size of the ROBDD. Our other contribution is that we propose an original method that uses *Disjoint Sum of Products*(DSOPs) to generate the test vectors which has test generation complexity of $O(m \times n)$; m is the number of DSOPs and n is the number of primary inputs. This contribution also shows that the test generation method for MSAF can also be used to derive tests for PDFs. The PDF test set is a superset of MSAF tests.

One of our major contributions is to systematically prove that, when MSAF tests derived for all nodes are applied to the circuit, MSAF combinations of the *entire* circuit are covered. Thus our synthesis is fully testable for MSAFs.

We have performed experiments on combinational circuits and combinational sections of sequential circuits. Synthesis tool is guided to perform unoptimized synthesis with constraints to use only 2x1 muxes and 2 input AND/OR gates. It can be deduced from the results that synthesis for complete testability may lead to increase in area, power and delay. However, this is not the case for all circuits. If the logic is more decision based then area, power and delay increase marginally. Single output circuits have larger ratio of increased area as compared to multi-output circuits due to absence of shared ROBDDs. The advantage of this synthesis is the direct use of multiplexer cells which are readily available in submicron technologies which eventually aid in restricting the area. Another conclusion that can be derived from the experiments is that unavailablity of MSAF simulators prohibits test compaction leading to a large MSAF test volume as compared to SSAF test volume. The test generation methodology is scalable as long as BDDs are constructed. Tests can be generated for larger circuits if tools that can create BDDs for larger circuits, are developed.

7.2 Future Scope

The synthesis method proposed by us is complete but it throws up scalability issues. The method is for BDD based synthesis only. BDD tools have limitations regarding the size of the circuits. As the circuit size increases, the BDD tool times out. Other point of argument is the sacrifice of operating speed for testability. As ROBDD node depth increases, path lengths increase, which is an undesired outcome. In order to overcome both these issues

an approach that considers testable two level synthesis with BDD based synthesis can be investigated. A synthesis method which combines *Sum of Pseudo Products*(SPPs) and BDD implementation was proposed in [16]. The aim of the synthesis is to reduce the node depth. The resultant circuit is fully testable for SAFM. The concept of partitioning the design into 2-level implementation and BDD implementation can be induced from this work.

One of the challenges with delay testability is the delivery of test vector pairs. Test vectors are delivered to the logic via serial scan flops. To perform a PDF test, first, the combinational circuit is initialized with vector v_1 to generate an output vector o_1. The transition vector v_2 is then applied to induce logic transition on the path to be tested. Finally, the result output o_2 from the combinational circuit is latched into flip-flops. The time interval between the application of v_2 and the capture of o_2 must be kept exactly the same as in a normal operation, the expected delay of the circuit. For delivery of such (v_1, v_2) pairs, the scan flop hardware is modified to enhanced scan[11]. In the thesis we have shown that the test vector pair (v_1, v_2) differ in one bit only. Once v_1 is scanned in, only one bit has to be flipped to get v_2. This can be achieved using Random Access Scan(RAS)[3]. Le et al.[32] have combined enhanced serial scan with RAS for reduction in test application time. This method has a potential application in scheduling test delivery for synthesis proposed in this thesis.

References

[1] M. Abramovici, M. A. Breuer, and A. D. Friedman. *Digital Systems Testing and Testable Design*. Wiley-IEEE Press, 1990.

[2] A. Agrawal, A. Saldanha, L. Lavagno, and A. L. Sangiovanni-Vincentelli. Compact and complete test set generation for multiple stuck-faults. In *Proceedings of International Conference on Computer Aided Design*, pages 212–219, Nov 1996.

[3] H. Ando. Testing vlsi with random access scan. In *COMPCON*, pages 50–52, Feb 1980.

[4] P. Ashar, S. Devadas, and K. Keutzer. Gate-delay-fault testability properties of multiplexor-based networks. In *Proceedings. International Test Conference*, pages 887–, Oct 1991.

[5] P. Ashar, S. Devadas, and K. Keutzer. Path-delay-fault testability properties of multiplexor-based networks. *Integration, the VLSI Journal*, 15(1):1 – 23, 1993.

[6] A. Balakrishnan and S. T. Chakradhar. Sequential circuits with combinational test generation complexity. In *Proceedings of 9th International Conference on VLSI Design*, pages 111–117, Jan 1996.

[7] B. Becker. Synthesis for testability: Binary decision diagrams. In *Finkel A., Jantzen M., STACS 1992. Lecture Notes in Computer Science*, volume 577, pages 849–855, 1992.

[8] A. Bernasconi, V. Ciriani, and R. Cordone. Exor projected sum of products. In *2006 IFIP International Conference on Very Large Scale Integration*, pages 284–289, Oct 2006.

[9] A. Bernasconi, V. Ciriani, and R. Cordone. An approximation algorithm for fully testable kep-sop networks. In *17th ACM Great Lakes symposium on VLSI (GSVLSI)*, pages 417–422, Oct 2007.

[10] R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, Aug 1986.

[11] M. Bushnell and V. Agrawal. *Essentials of Electronic Testing for Digital, Memory and Mixed-signal VLSI Circuits*. Kluwer Academic Publishers, 2000.

[12] S. Chakrabarti, S. Das, D. K. Das, and B. B. Bhattacharya. Synthesis of symmetric functions for path-delay fault testability. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 19(9):1076–1081, Sep 2000.

[13] P. Chen, D. A. Kirkpatrick, and K. Keutzer. Miller factor for gate-level coupling delay calculation. In *IEEE/ACM International Conference on Computer Aided Design. ICCAD - 2000. IEEE/ACM Digest of Technical Papers (Cat. No.00CH37140)*, pages 68–74, Nov 2000.

[14] N. Drechsler, M. Hilgemeier, G. Fey, and R. Drechsler. Disjoint sum of product minimization by evolutionary algorithms. In *Raidl G.R. et al. (eds) Applications of Evolutionary Computing. EvoWorkshops 2004. Lecture Notes in Computer Science*, volume 3005, 2004.

[15] 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*, 23(3):440–443, March 2004.

[16] G. Fey, A. Bernasconi, V. Ciriani, and R. Drechsler. On the construction of small fully testable circuits with low depth. In *10th Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD 2007)*, pages 563–569, Aug 2007.

[17] M. Fujita and A. Mishchenko. Efficient sat-based atpg techniques for all multiple stuck-at faults. In *2014 International Test Conference*, pages 1–10, Oct 2014.

[18] M. Fujita, N. Taguchi, K. Iwata, and A. Mishchenko. Incremental atpg methods for multiple faults under multiple fault models. In *Sixteenth International Symposium on Quality Electronic Design*, pages 177–180, March 2015.

[19] H. Fujiwara. *Logic Testing and Design for Testability*. MIT Press, 1985.

[20] H. Fujiwara. A new class of sequential circuits with combinational test generation complexity. *IEEE Transactions on Computers*, 49(9):895–905, Sep 2000.

[21] R. Gupta, R. Gupta, and M. A. Breuer. The ballast methodology for structured partial scan design. *IEEE Transactions on Computers*, 39(4):538–544, Apr 1990.

[22] J. P. Hayes. A nand model ror fault diagnosis in combinational logic networks. *IEEE Transactions on Computers*, C-20(12):1496–1506, Dec 1971.

[23] J. L. A. Hughes. Multiple fault detection using single fault test sets. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 7(1):100–108, Jan 1988.

[24] J. Jacob and N. N. Biswas. Gtbd faults and lower bounds on multiple fault coverage of single fault test sets. In *International Test Conference(ITC)*, pages 849–855, 1987.

[25] N. K. Jha and S. Gupta. *Testing of Digital Systems*. Cambridge University Publication, 2003.

[26] W. Ke and P. R. Menon. Delay-testable implementations of symmetric functions. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 14(6):772–775, Jun 1995.

[27] Y. C. Kim, V. D. Agrawal, and K. K. Saluja. Combinational test generation for various classes of acyclic sequential circuits. In *Proceedings International Test Conference 2001 (Cat. No.01CH37260)*, pages 1078–1087, 2001.

[28] Y. C. Kim, V. D. Agrawal, and K. K. Saluja. Multiple faults: modeling, simulation and test. In *Proceedings of ASP-DAC/VLSI Design 2002. 7th Asia and South Pacific Design Automation Conference and 15th International Conference on VLSI Design*, pages 592–597, 2002.

[29] I. Kohavi and Z. Kohavi. Detection of multiple faults in combinational logic networks. *IEEE Transactions on Computers*, C-21(6):556–568, June 1972.

[30] Z. Kohavi and N. Jha. *Switching and Finite Automata Theory*. Cambridge University Publication, 3 edition, 2010.

[31] A. Krstic and K.-T. T. Cheng. *Delay Fault Testing for VLSI Circuits*, volume 14. Springer US, 1 edition, 1998.

[32] K. T. Le, D. H. Baik, and K. K. Saluja. Test time reduction to test for path-delay faults using enhanced random-access scan. In *20th International Conference on VLSI Design held jointly with 6th International Conference on Embedded Systems (VLSID'07)*, pages 769–774, Jan 2007.

[33] A. Matrosova, V. Lipsky, A. Melnikov, and V. Singh. Path delay faults and enf. In *2010 East-West Design Test Symposium (EWDTS)*, pages 164–167, Sept 2010.

[34] A. Matrosova, E. Mitrofanov, and V. Singh. Delay testable sequential circuit designs. In *East-West Design Test Symposium (EWDTS 2013)*, pages 1–4, Sept 2013.

[35] A. Matrosova, E. Nikolaeva, D. Kudin, and V. Singh. Pdf testability of the circuits derived by special covering robdds with gates. In *East-West Design Test Symposium (EWDTS 2013)*, pages 1–5, Sept 2013.

[36] E. J. McCluskey. Verification testing : A pseudoexhaustive test technique. *IEEE Transactions on Computers*, C-33(6):541–546, June 1984.

[37] E. J. McCluskey and S. Bozorgui-Nesbat. Design for autonomous test. *IEEE Transactions on Computers*, C-30(11):866–875, Nov 1981.

[38] A. Mischenko. Berkeley logic synthesis and verification group, abc: A system for sequential synthesis and verification.

[39] R. Mitra, D. K. Das, and B. B. Bhattacharya. On designing robust path-delay fault testable combinational circuits based on functional properties. In *2014 IEEE Computer Society Annual Symposium on VLSI*, pages 202–207, July 2014.

[40] C. J. Moore, P. Wang, A. M. Gharehbaghi, and M. Fujita. Test pattern generation for multiple stuck-at faults not covered by test patterns for single faults. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–4, May 2017.

[41] I. Pomeranz and S. M. Reddy. Design-for-testability for improved path delay fault coverage of critical paths. In *21st International Conference on VLSI Design (VLSID 2008)*, pages 175–180, Jan 2008.

[42] H. Rahaman and D. K. Das. A simple delay testable synthesis of symmetric functions. *Manandhar S., Austin J., Desai U., Oyanagi Y., Talukder A.K. (eds) Applied Computing. AACC 2004. Lecture Notes in Computer Science*, 3285, 2004.

[43] S. K. Rao, C. Sathyanarayana, A. Kallianpur, R. Robucci, and C. Patel. Estimating power supply noise and its impact on path delay. In *2012 IEEE 30th VLSI Test Symposium (VTS)*, pages 276–281, 2012.

[44] D. R. Schertz and G. Metze. A new representation for faults in combinational digital circuits. *IEEE Transactions on Computers*, C-21(8):858–866, Aug 1972.

[45] T. Shah, A. Matrosova, B. Kumar, M. Fujita, and V. Singh. Testing multiple stuck-at faults of robdd based combinational circuit design. In *2017 18th IEEE Latin American Test Symposium (LATS)*, pages 1–6, March 2017.

[46] T. Shah, A. Matrosova, and V. Singh. Test pattern generation to detect multiple faults in robdd based combinational circuits. In *2017 IEEE 23rd International Symposium on On-Line Testing and Robust System Design (IOLTS)*, pages 211–212, July 2017.

[47] 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)*, pages 1–4, Oct 2016.

[48] M. Siebert and E. GramatovÃą. Delay fault coverage increasing in digital circuits. In *2013 Euromicro Conference on Digital System Design*, pages 475–478, Sept 2013.

[49] G. L. Smith. Estimating power supply noise and its impact on path delay. In *International Test Conference(ITC)*, pages 342–349, 1985.

[50] 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*, 21(3):362–368, Mar 2002.

[51] M. Tehranipoor, K. Peng, and K. Chakraborty. *Test and Diagnosis for Small Delay Defects*, volume 4. Springer-Verlag New York, 1 edition, 2012.

[52] J. G. Udell and E. J. McCluskey. Partial hardware partitioning: a new pseudo-exhaustive test implementation. In *International Test Conference 1988 Proceedings:New Frontiers in Testing*, pages 1000–, Sep 1988.

List of Publications

- Journal Publication

  1. **Toral Shah**, Anzhela Matrosova, Masahiro Fujita, and Virendra Singh, "*Multiple Stuck-at Fault Testability Analysis of ROBDD Based Combinational Circuit Design*", Journal of Electronic Testing(JETTA), (Volume 34, Number 1),February 2018, DOI: 10.1007/s10836-018-5703-3

- Conference Publications

  1. **Toral Shah**, Anzhela Matrosova,Virendra Singh, "*Test Pattern Generation to Detect Multiple Faults in ROBDD based combinational Circuits*", 23rd IEEE International Symposium on On-Line Testing and Robust System Design(IOLTS) 2017, Greece, http://ieeexplore.ieee.org/document/8046223/
  2. **Toral Shah**, Anzhela Matrosova, Binod Kumar, Masahiro Fujita, Virendra Singh, "*Testing multiple stuck-at faults of ROBDD based combinational circuit design*", 18th IEEE Latin American Test Symposium(LATS) 2017, Bogota, Colombia, http://ieeexplore.ieee.org/document/7906753/
  3. **Toral Shah**, Virendra Singh, Anzhela Matrosova, "*ROBDD based Path Delay Fault testable combinational circuit design*", 14th IEEE East-West Design and Test Symposium(EWDTS) 2016, Yerevan, Armenia, http://ieeexplore.ieee.org/document/7807682/
  4. **Toral Shah**, Anzhela Matrosova, Virendra Singh, "*BDD based PDF testable combinational circuit design*", 14th IEEE Workshop on RTL and High Level Testing(WRTLT) 2015, Mumbai, India
  5. **Toral Shah**, Anzhela Matrosova, Virendra Singh, "*PDF testability of a combinational circuit derived by covering ROBDD nodes using Invert-And-Or circuits*", 8th VLSI Design and Test(VDAT) 2015, Ahmedabad, India, http://ieeexplore.ieee.org/document/7208130/
  6. Anzhela Matrosova, E.Mitrofanov, **Toral Shah**, "*Simplification of Fully Delay Testable Combinational circuits*", 21st IEEE International Symposium on On-Line Testing and Robust System Design(IOLTS) 2015, Halkidiki, Greece, http://ieeexplore.ieee.org/document/7229829/
  7. Anzhela Matrosova, E.Mitrofanov,**Toral Shah**, "*Multiple Stuck-at Fault Testability of a Combinational Circuit Derived by Covering ROBDD Nodes*

by Invert-And-Or Sub-circuits", 13th IEEE East-West Design and Test Symposium(EWDTS) 2015, http://ieeexplore.ieee.org/document/7493099/

8. Prashant Singh, **Toral Shah**, Virendra Singh, "*An Improved Single-Input-Change(SIC) Based Built-In-Self-Test for Delay Testing*", 13th IEEE Workshop on RTL and High Level Testing(WRTLT) 2014, Hangzhou, China (Not included in thesis)

9. **Toral Shah**, Mrudula Modi, "*Spirituality as Science: Experience is Proof*, 8th All India Students Conference on Science and Spiritual Quest(AISSQ) 2014, India", http://www.academia.edu/4908966/Spirituality_as_Science_Experience_is_Proof(Not included in thesis)