

TEST PATTERN GENERATION FOR VLSI CIRCUITS

A THESIS

*submitted in fulfilment of the
requirements for the award of the degree*

of

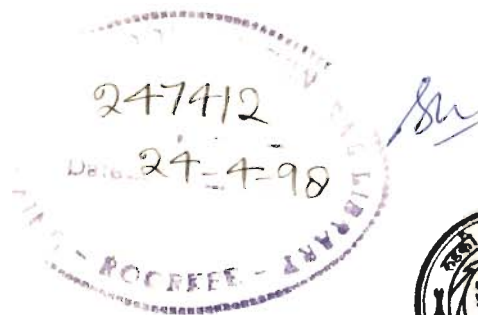
DOCTOR OF PHILOSOPHY

in

ELECTRONICS AND COMPUTER ENGINEERING

By

GOPAL KRISHAN SHARMA



DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING
UNIVERSITY OF ROORKEE
ROORKEE-247 667 (INDIA)


JANUARY, 1997

Gratis


CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis entitled **Test Pattern Generation for VLSI Circuits** in fulfilment of the requirement for the award of the Degree of **Doctor of Philosophy** and submitted in the Department of Electronics and Computer Engineering of the University is an authentic record of my own work carried out during a period from September 1993 to January 1997 under the supervision of Dr. R.P. Agarwal, Dr. J.P. Gupta, Dr. S. Sarkar and Dr. S. Ahmad.

The matter presented in this thesis has not been submitted by me for the award of any other degree of this or any other University.



(GOPAL K. SHARMA)

This is to certify that the above statement made by the candidate is correct to the best of our knowledge.


(Dr. S. Sarkar)


Professor

Dept. of Electronics & Computer Engg.
University of Roorkee, Roorkee - 247 667


(Dr. R.P. Agarwal)


Professor

Dept. of Electronics & Computer Engg.
University of Roorkee, Roorkee - 247 667


(Dr. S. Ahmad)

Deputy Director

Central Electronics Engg. Research
Institute, Pilani - 333 031



(Dr. J.P. Gupta)


Member Secretary

All India Council for Technical
Education, New Delhi - 110 002

Date: 15th January 1997

The Ph.D. Viva-Voce examination of Mr. Gopal K. Sharma, Research Scholar, has been held on 13/9/97 at 10:30 AM


1. 


2. 

3. 

4. 

(Signature of
Supervisors)


(Signature of H.O.D.)


(Signature of External
Examiner)

Abstract

Test pattern generation problem is known to be NP -complete and conventionally, automatic generation of test patterns for circuits with N number of primary inputs is characterized as a search of N -dimensional 0-1 state space. Several Automatic Test Pattern Generation (ATPG) algorithms have been developed in the past. But, in spite of considerable progress in the development of these algorithms, the computational resources required for ATPG are still enormous because the efficiency of these algorithms has not kept pace with the increasing complexity of VLSI circuits which is of the order of several tens of thousands of gates. Furthermore, these algorithms are targeted for serial computers. However, massively parallel computers and distributed network of powerful workstations, available in most of the VLSI-CAD environments, present a promising paradigm for solving such compute-intensive ATPG problems for combinational as well as sequential circuits. With the availability of these paradigms, new approaches and cost-effective methods, are therefore, imperative in order to efficiently solve the ATPG problem. Efforts have been made to investigate new test generation algorithms so that they could be easily extended to parallel and distributed computing paradigms.

Recently, two different ATPG approaches have been developed in which the problem of test pattern generation is formulated either as a Boolean satisfiability (SAT) or an optimization problem. These approaches are radically different from the conventional methods of generating test patterns for circuits from their gate level description. In these approaches, the generation of test patterns is done in two steps. In the first step, a formula is constructed which will be either an energy function or a Boolean truth (false) function representing *Boolean difference* between the fault-free and faulty circuits. In the second step, energy minimization techniques are applied to minimize

the energy function or the Boolean false function and SAT algorithms are applied to satisfy the Boolean truth function. Although energy minimization and SAT problems are as hard as ATPG problem itself, these approaches have two significant advantages. First, since the function of the circuit is mathematically expressed, several new techniques may be investigated to solve the ATPG problem. Second, the non-causal form of the model would allow the use of parallel processing.

New techniques for the ATPG approaches as described above have been attempted in this thesis. In the optimization based approach, a new quadratic 0-1 programming technique to minimize the energy function, which is of the form of pseudo-Boolean quadratic function: $E(\mathbf{x}) = \mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{c}^T \mathbf{x}$, has been developed and applied to obtain test patterns. Other test generation methods developed are based on Genetic Algorithms (GAs). In these methods, an objective function is derived from either an energy function or a Boolean false function. In order to apply GAs, the constrained ATPG problem has been transformed into the unconstrained one by modifying the objective function with the help of a penalty method which is developed by using schema design. The objective function so obtained has been mapped into fitness form which is then maximized to generate test patterns. The above ATPG process has been further accelerated by incorporating a transitive closure algorithm to reduce the Boolean false function to be minimized by GAs.

Finally, the proposed ATPG methods have been implemented by developing a CAD tool to run on a Sun Sparc 10 UNIX workstation. This tool has been exercised by generating test patterns for a set of stuck-at faults in several practical example circuits. Experimental results in terms of average CPU time per fault for the example circuits are reported in the thesis. The results obtained by the proposed ATPG methods have also been compared to the results published in the literature for the same example circuits which shows the efficiency and effectiveness of the proposed methods.

Acknowledgements

I wish to express my heartfelt gratitude to my supervisors **Dr. R.P. Agarwal, Dr. J.P. Gupta, Dr. S. Sarkar** and **Dr. S. Ahmad** for offering guidance, encouragement, help and useful suggestions throughout. I consider myself very fortunate for having been associated with them.

The cooperation and help extended by the Head of the Department of Electronics and Computer Engineering is gratefully acknowledged.

Thanks are due to the authorities of Central Electronics Engineering Research Institute (CEERI), Pilani (Rajasthan) and Council of Scientific & Industrial Research (CSIR), New Delhi for granting necessary leave in this regard. My special thanks are due to **Prof. R.N. Biswas**, Director, CEERI and **Dr. Chandra Shekhar**, Group Leader, IC Design Group, CEERI for by extending the facilities available in the VLSI Design Center.

I am grateful to **Dr. Vishwani D. Agarwal**, Distinguished Member of Technical Staff at the AT&T Bell Laboratories in Murray Hill, NJ (USA) and **Dr. Kewal K. Saluja**, Professor, Department of Electrical & Computer Engineering, University of Wisconsin-Madison (USA) for bringing me in this important area of research and in fact, they are my main motivator.

I am highly grateful to **Prof. M.P. Kapoor**, Director, Thapar Institute of Engineering & Technology, Patiala for his constant encouragement, valuable suggestions and moral support.

It is difficult to express my gratitude to my wife and children as anything said would be inadequate to compensate for the neglect that is sadly and inevitably the outcome of undertaking doctoral research. Finally, I thank all my relatives and friends for their valuable help and co-operation during my thesis work.

Contents

Candidate's Declaration	
Abstract	i
Acknowledgements	iii
1 Introduction	8
2 Test Generation Algorithms: A Review	17
2.1 Algebraic Algorithms	19
2.2 Structural Algorithms	20
2.2.1 Combinational Algorithms	21
2.2.2 Sequential Algorithms	38
2.3 Parallel Processing Techniques	43
2.3.1 Fault Partitioning	44
2.3.2 Heuristic Parallelization	46
2.3.3 Search-Space Partitioning	47
2.3.4 Algorithmic Partitioning	49

2.3.5	Topological Partitioning	50
2.4	Recent Approaches	52
2.4.1	ATPG Constraint Network	53
2.4.2	Optimization Methods for Test Generation	54
2.4.3	Test Generation Using Boolean Satisfiability	59
2.5	Problem Formulation	64
3	A New Quadratic 0-1 Programming for Test Generation	67
3.1	Quadratic 0-1 Programming Technique	69
3.2	A New Quadratic 0-1 Programming Technique	74
3.2.1	Stability Race Condition	76
3.2.2	Criteria for Fixation of Variables	78
3.2.3	Backtracking Procedure	83
3.3	Examples	83
3.4	Computational Algorithm	95
3.5	Complexity Analysis	98
4	Genetic Algorithm Based Test Generation	100
4.1	Genetic Algorithms in Search and Optimization	103
4.1.1	A Simple Genetic Algorithm	103
4.1.2	Adaptive Genetic Algorithms	109
4.2	Genetic Algorithms for Test Generation Problem	111
4.3	Energy Minimization Approach	113

4.3.1	Schema Design	114
4.3.2	Penalty Method	115
4.3.3	Mapping Objective Function to Fitness Form	116
4.3.4	Example	116
4.4	Boolean Satisfiability Approach	119
4.5	Test Generation Algorithm	121
4.6	Automatic Test Pattern Generation CAD Tool	127
4.6.1	Circuit Modeling	127
4.6.2	Objective Function Evaluation	129
4.6.3	Penalty Inclusion	130
4.6.4	Global Optimization	130
5	Implementation Details and Experimental Results	132
5.1	Translator Module	134
5.2	ATPG Network Module	136
5.3	ATPG Program Module	138
5.3.1	QUADTEST	139
5.3.2	GATEST	141
6	Conclusion and Future Scope of Work	151
6.1	Conclusion	151
6.2	Future Scope of Work	155
	References	157

List of Figures

1.1	Design and test development cycle of VLSI chip realization.	9
2.1	AND gate D-cubes.	22
2.2	An example combinational circuit	24
2.3	D-algorithm example.	24
2.4	A generic combinational circuit.	26
2.5	Podem search-space graph.	28
2.6	A Fan example.	30
2.7	A block schematic of the ATPG tool.	33
2.8	Abstract problem representation of deterministic test generation.	35
2.9	(a) Canonical structure of a synchronous sequential circuit, and (b) its combinational iterative array model.	40
2.10	An example circuit for sequential test generation.	41
2.11	Time frame expansion of the example circuit for sequential test generation.	42
2.12	Search-space partitioning	48
2.13	ATPG Constraint Network.	53

2.14	(a) 2-input AND gate and (b) its neural network model.	55
2.15	(a) 2-input XOR gate and (b) its neural network representation.	56
2.16	CNF formulas for the basic gates.	60
2.17	(a) An example circuit and (b) its implication graph.	61
2.18	Reduced implication graph.	62
3.1	(a) An AND gate and (b) its neural network model.	70
3.2	(a) An example circuit (b) Neural network representation of the example circuit for d stuck-at-0 fault.	72
3.3	(a) Implication graph for the example circuit and (b) its condensed form.	74
3.4	$[E(\mathbf{x})]_{x_i}$ as function of x_i	76
3.5	(a) An ISCAS'85 circuit: c17.isc (b) Modified ISCAS circuit for the stuck-at-1 fault on line x_6	87
3.6	ATPG neural graph for the ISCAS circuit.	88
4.1	A plot for the energy function.	101
4.2	A plot for the Boolean false function.	102
4.3	The ISCAS '85 circuit with XOR as an output interface	120
4.4	Genetic Algorithms Based Test Generation Method	122
4.5	Implication graph of the example ISCAS circuit with $x_6 = 0$, $x_{13} = 0$, and $x_{14} = 1$	124
4.6	(a) Reduced version of the implication graph shown in Figure 4.4 after fixing the variables, $x_1 = 1$, $x_3 = 1$, $x_{10} = 1$, $x_{12} = 0$, and $x_8 = 1$, (b) Condensed form of the implication graph (a)	125

4.7	A CAD Tool for Automatic Test Pattern Generation.	128
5.1	Abstract view of the CAD tool.	133
5.2	<i>c17.isc</i> : An ISCAS'85 circuit	136
5.3	Graph representation for the <i>c17.isc</i> circuit	137
5.4	ATPG constraint network of the ISCAS circuit <i>c17.isc</i> for stuck-at-1 fault on line x_6	138
5.5	Graph representation for the ATPG constraint network of <i>c17.isc</i> circuit	139

List of Tables

5.1	Experimental Results: QUADTEST	141
5.2	Experimental Results: GATEST	148

Chapter 1

Introduction

Recent advances in integrated circuit (IC) technology have made it possible to fabricate digital circuits with a very large number of devices on a single chip. **Very Large Scale Integration (VLSI)** is the fabrication of millions of components and interconnections on a chip by a common set of manufacturing steps. In the era of **Ultra Large Scale Integration (ULSI)**, it is expected to have as many as 100 million transistors or even more on a single chip by the end of this century. Because of this dramatic increase in the circuit size, the design and test development of the VLSI chips becomes a costly affair with a very large turn around time. The complete design and test development cycle of the VLSI chip realization is depicted in Figure 1.1. The design and test planning of VLSI chips begins with specifications. In VLSI design, architectural design is the first step which consists of partitioning of a VLSI chip into realizable functional blocks. Logic design comes after the architectural design step and includes several test activities. In the logic design, either the logic is synthesized in a testable form or the synthesized logic is analyzed for testability. After logic synthesis, test patterns are generated and combined with the test plan to develop a program for the **Automatic Test Equipment (ATE)**. The actual testing of VLSI chips takes place after the phys-

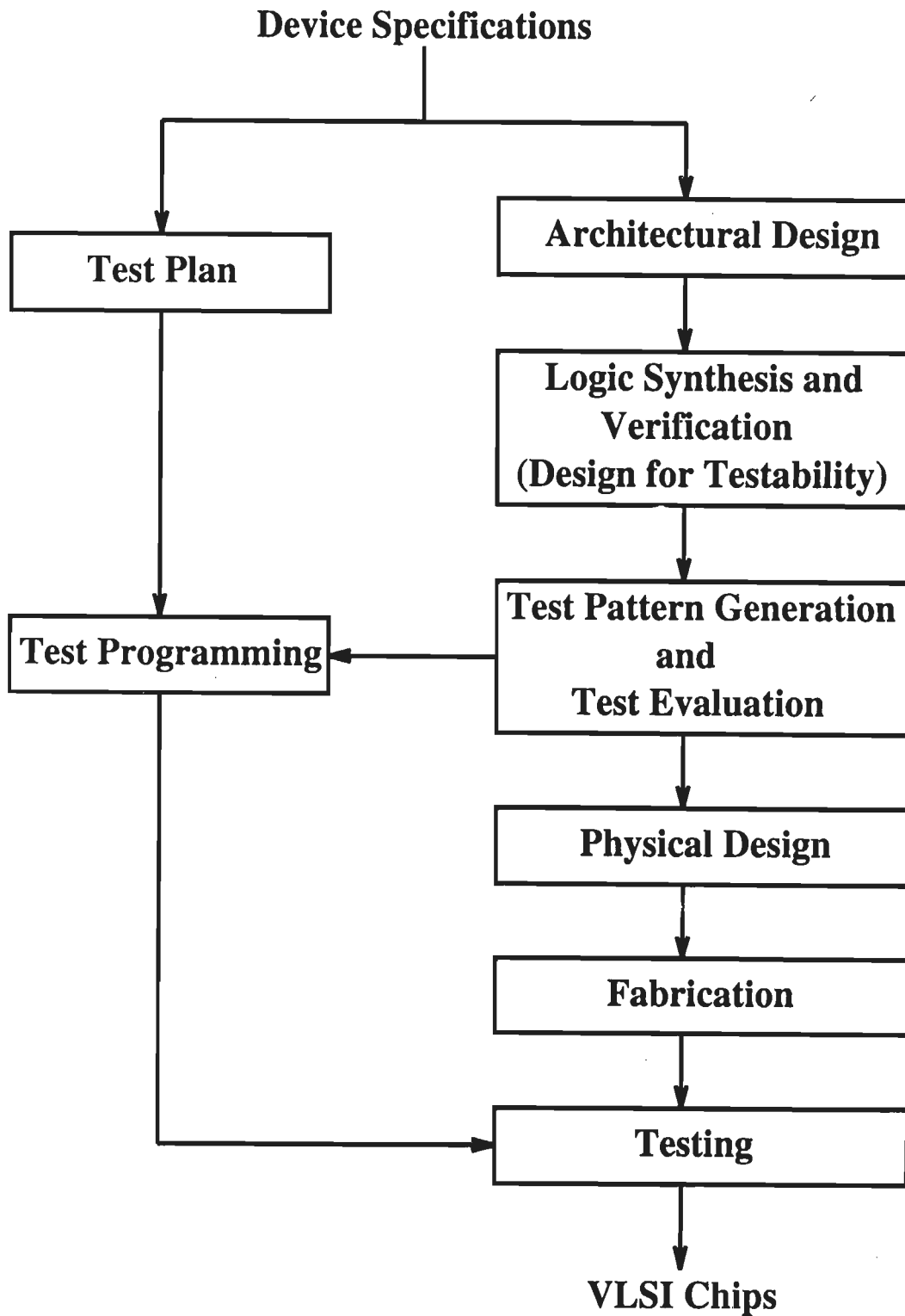


Figure 1.1: Design and test development cycle of VLSI chip realization.

ical design (layout, timing verification and mask generation) and fabrication (wafer processing).

Testing of VLSI circuits, because of their increased complexities and reduced geometries, becomes much more difficult and expensive as it costs more than 50 per cent of the total chip cost. The advantages of VLSI, *i.e.* reduced system cost, better performance, greater reliability, would be lost unless the chips can be tested economically. The testing process detects the physical defects produced during the fabrication of the systems. A sequence of input stimuli, known as *test patterns* or *tests*, is required to test a VLSI chip. These tests are supposed to determine correctness of each component (transistor *etc.*) and each interconnection fabricated on the chip. The test patterns are developed so that they should thoroughly check every node of the circuit. Normally, these tests are required to detect a very high fraction (*i.e.* close to 100 percent) of the *modeled faults* in the circuit which means that the device should be defect-free. The detected fraction of faults is called the *fault coverage* of the tests.

Test patterns for a circuit are developed in two phases. In the first phase, known as *design verification*, tests are generated to verify logical correctness and timing behavior of the circuit through simulation. The second phase of test generation consists of generation of *manufacturing tests* in order to ensure that circuit is fabricated correctly. However, these tests do not determine whether the circuit has been designed correctly. These test vectors are generated (either manually or automatically) after a complete design by driving the effect of faults within the circuit to a circuit output where the fault can be detected. **Automatic Test Pattern Generation (ATPG)** involves the generation of test vectors automatically in order to detect failures in the VLSI chip. The ATPG process is driven by the single stuck-at fault model which assumes that all faults can be modeled by a line being either permanently 0 or 1. After test pattern generation, the vectors and their expected responses must be moved to a

piece of ATE, which applies the vectors to the input pins of the VLSI chip and compares the output pin responses with the expected responses to determine whether a fault is present within the circuit or not. Since the same set of vectors will be applied to a large number (a million or more in many cases) of copies of the chip, short test sequences are desirable.

Generating test patterns for digital integrated circuits is a very hard problem. Sequential circuits, because of their memory elements, might require many vectors to test one fault. The time required to generate these vectors is too large for VLSI chips. To solve this problem, circuit designers use *design-for-testability* techniques. One such technique is scan design which converts sequential circuits into combinational circuits by chaining all the flip-flops in the circuit into one or several shift registers. This allows using faster combinational test generation algorithms for vector generation. Another technique known as built-in-self-test adds IC hardware to generate random patterns that can be applied to the chip's combinational logic. This technique has the advantages of applying long vector sequences at circuit speed and there is little or no need of test generation. However, the disadvantage of this technique is that the chip becomes more complex leading to performance penalties. Both the techniques have gained wide and growing acceptance as devices become more complex. Further discussion on these techniques is beyond the scope of this thesis as the focus is on the test pattern generation.

The problem of Automatic Test Pattern Generation is known to be *NP*-complete [55], [32] and conventionally, the generation of tests is characterized as a search of N -dimensional 0-1 state space, where N is the number of primary inputs in the circuit to be tested. A wide range of ATPG algorithms have been developed in the past and focussed uniprocessors machines as their hardware platforms. Major emphasis has been on the increase in the efficiency of these serial algorithms through better

heuristics and data structures. However, the overall gains achieved through these developments have not kept pace with the increasing complexity of the VLSI circuits as sometimes hours and days are required in generating tests for the practical circuits of even moderate complexity. The complexity of the circuits on which test generation tools are used is growing faster than the speed of the computers on which they run. Therefore, new algorithms and techniques for test generation are required for both the circuits of today and tomorrow.

The availability of affordable parallel machines and distributed network of idle workstations in most of the VLSI-CAD environments has opened a new front for the development of efficient parallel/distributed algorithms for the compute-intensive test generation problem. In order to harness the computational power of these machines, several parallelization techniques have been investigated in the recent past. These techniques have tried to parallelize some of the portions of the conventional uniprocessor algorithms and executed them in parallel. Although, these techniques have shown some promising results, but much work remains since no effective parallel algorithms have been found.

Most recently altogether different approaches are developed in which the testing problem has been reformulated so that the algorithms for test generation can be easily parallelized to run on massively parallel and distributed computing platforms. In these approaches, the test generation problem is transformed into an optimization or Boolean satisfiability problem. Although both the optimization and satisfiability problems are as hard as test generation itself, but they have two significant advantages. First, several operation research and graph-theoretic techniques may become applicable and second, the non-causal form of the model makes parallel processing possible for such compute-intensive test generation problem.

The optimization based test generation radically differs from the conventional methods that generate test vectors for circuits from their gate level description. The circuit is modeled as a network of idealized computing elements, known as *neurons*, connected through bidirectional links and the neuron is a binary (0-1) element. The relationship between the input and output signal states of a logic gate is expressed by an *energy function* such that the zero energy (also the minimum energy) states correspond to the gate's logic function. Using this neural network modeling technique, the test generation problem has been transformed into an energy minimization problem for which various techniques like directed search augmented by probabilistic relaxation have been attempted to determine its global minimum. A quadratic 0-1 programming technique has also been devised for test pattern generation since the energy function is of the form of pseudo-Boolean quadratic function $E(\mathbf{x}) = \mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{c}^T \mathbf{x}$ with $\mathbf{x} \in \{0, 1\}^n$, where \mathbf{Q} is an $n \times n$ symmetric matrix of constants, \mathbf{c} is a vector of constants, \mathbf{x} is a vector of n Boolean variables, and \mathbf{x}^T is the transpose of \mathbf{x} .

Test pattern generation method using Boolean satisfiability constructs a formula expressing the *Boolean difference* between fault-free and its corresponding faulty circuit. This formula may be derived in the form of a Boolean truth function or a false function. In order to generate a test pattern for a given fault, one needs to satisfy the formula using the Boolean satisfiability algorithm when the formula is in the form of a truth function. As far as the false function is concerned, it is to be minimized to 0 and quite similar to the energy minimization approach. Test generation methods using both kinds of functions has been developed and reported in the literature.

The thesis is mainly focused on the development of new algorithms and tools for test pattern generation based on optimization and Boolean satisfiability approaches because these approaches have significant advantages as discussed above. In the optimization based test generation approach, the ATPG problem is formulated as energy

minimization for which several algorithms have been proposed. In this kind of ATPG formulation, the function to be minimized is of the form of pseudo-Boolean quadratic function for which a new quadratic 0-1 programming technique has been developed to find its global minimum. Although this is a mathematical technique, it has several advantages. One important advantage is that the exploitation of the special structure of the quadratic function arising from the test generation problem has become possible as the proposed technique uses the circuit specific knowledge with ATPG constraints. Other significant advantage is that efficient heuristics can be easily incorporated which further accelerate the minimization process. Some of the heuristics are described in detail by illustrating specific examples. A computational algorithm has been developed which efficiently minimizes the energy function with moderate number of variables for small practical example circuits [104]. However, its efficiency decreases for the large number of variables due to branch-and-bound process used in the algorithm.

Other methods based on Genetic Algorithms (GAs) for generating test patterns have been developed which do not incorporate branch-and-bound process at any stage of the ATPG. In these methods, GAs are applied to both kinds of ATPG problem formulation viz. optimization as well as Boolean satisfiability. In the optimization based approach, GAs are used to find the global minimum of the energy function for generating test patterns for a given set of faults. In the Boolean satisfiability approach, instead of deriving a Boolean truth function in Conjunctive Normal Form (CNF), a Boolean false function is derived which is then minimized to find test patterns. This is an advantage as both kinds of problem formulations ultimately lead to minimization problem and thus can be solved by using GAs. The behaviour of both the energy and Boolean false functions derived for the test generation of stuck-at faults in combinational circuits has been studied and it is found that these functions are of multimodal type. The global optimum solution of the problem represented by these functions can

be located in a multimodal landscape which further confirms the suitability of GAs known to be the most effective search and optimization algorithms in such problem spaces. Furthermore, GAs have a proven track record of being one of the most robust search and optimization algorithms applied in a number of combinatorial optimization problems. Other important features of GAs are that they are innovative and have inherent amenability to be processed in parallel.

In order to apply GAs for minimization of energy function or Boolean false function, an objective function is derived which is essentially required to guide GA-based search. This objective function is then modified to transform the constrained ATPG problem into an unconstrained one by using a penalty method. This penalty method has been developed with the help of schema design. The objective function so obtained is then mapped into fitness form (maximization form) by using a mapping procedure. The fitness function is then maximized using GAs and test patterns are obtained for all the detectable faults. A GA-based CAD tool for the ATPG problem has been developed and experimental results have been reported in the thesis.

Organization of the Thesis

The research contribution made in this thesis is organized in six chapters as given below.

- **Chapter 2: *Test Generation Algorithms: A Review.*** This chapter reviews the work carried out in the area of test pattern generation for digital logic circuits. Most important test generation algorithms for combinational as well as sequential circuits are reported in this chapter. Various parallelization techniques are discussed. Most recently developed optimization and Boolean satisfiability test

generation approaches are also presented in this chapter. The chapter concludes with a brief description of the problem formulation.

- **Chapter 3:** *A New Quadratic 0-1 Programming Technique.* A new quadratic 0-1 programming technique applicable to the optimization based test generation approach has been proposed in this chapter. The applicability of this technique has been demonstrated by solving the test generation problem for some example circuits. Various heuristics and their role in the energy minimization process are discussed in brief. A computational algorithm based on the proposed technique is also given in this chapter. This chapter concludes with a brief discussion on this new technique.
- **Chapter 4:** *Genetic Algorithm Based Test Generation.* This chapter proposes new test generation methods using Genetic Algorithms. It starts with an introduction to Genetic Algorithms (GA) and describes their applications to the test generation problem. Issues involved in the proposed GA-based methods are discussed and a test generation algorithm has been proposed. Finally, a CAD tool based on GAs has been proposed.
- **Chapter 5:** *Implementation Details and Experimental Results.* The implementation details of the test generation algorithms proposed in Chapter 3 and 4 are given in this chapter. A prototype CAD tool proposed in Chapter 4 has been developed. Experimental results in terms of CPU time are also presented by running the ATPG CAD tool for generating test patterns for practical example circuits. These results are also compared to the results reported in the literature.
- **Chapter 6:** *Conclusion and Future Scope of Work.* Finally, conclusions are drawn and future scope of work in the area of the thesis is discussed in this chapter.

Chapter 2

Test Generation Algorithms: A Review

Intensive research efforts have been made in the development of efficient test generation algorithms for combinational as well as sequential logic circuits. Test pattern generation is usually more difficult for sequential circuits than the combinational ones mainly due to the poor controllability and observability of sequential logic. Putting storage elements of a sequential logic in a desired state would require a sequence of external inputs applied over a set of clock cycles and similar is the case for observing their values through the external outputs. This implies that the sequential test generation is not only more difficult (complex) than the combinational test generation but the resulting test length is also usually much larger. However, the popular *Design for Testability* (DFT) techniques, particularly *scan designs* [31], [37], [108], increase the controllability and the observability of the storage elements in the feedback path of sequential circuits. This would reduce the complexity of sequential test generation to that of combinational test generation by transforming the sequential circuit to a combinational logic in the test mode. Hence, for sequential circuits implemented with

the use of scan techniques, it is sufficient to develop efficient test generation algorithms only for combinational circuits. Due to this reason, there has been a heavy emphasis on test pattern generation algorithms for combinational circuits.

A wide range of techniques has been proposed for combinational circuit test generation. At one end of the spectrum are the exhaustive and random techniques. Exhaustive test generation means the generation of all possible input patterns and may be an obvious choice for circuits with small number of primary inputs. **R**andom **T**est **G**eneration (RTG) [4] is another simple technique that generates input vectors probabilistically and simulates whether these vectors detect fault(s) in the circuit. If a random vector detects a fault, then it is retained as a test. In RTG, a large set of random tests is needed in order to achieve high-quality tests usually determined by a *fault simulation method* in terms of *fault coverage*. Various fault simulation methods and their role in test generation are described later in this chapter. In RTG, though the test generation process itself is simple, determining the test quality by fault simulation may be an expensive process. Moreover, a longer test set costs more to apply because it increases the testing time and the memory requirements of the tester.

At the other end of the spectrum, there are deterministic techniques that can be divided into two groups: one consisting of *algebraic algorithms* and the other of *structural algorithms*. These techniques generate tests by processing a model of the circuit in contrast to RTG which do not consider the structure of the circuit-under-test. Compared to RTG, deterministic techniques are more expensive in terms of CPU time, but they produce shorter and high quality tests. Algebraic algorithms use the Boolean difference formulation (Section 2.1) to solve the problem symbolically [34], [90] and structural algorithms perform a topological search of the circuit-under-test. Algebraic algorithms have not been very practical for two reasons: (1) symbolic solutions require excessive storage and (2) heuristics required to solve practical test generation problems

symbolically are not available. Structural algorithms use a data structure representing the circuit-under-test instead of deriving a Boolean expression. These algorithms systematically enumerate the search space using branch-and-bound method and employ heuristics to guide the search process. The most successful structural algorithms are described in Section 2.2.

2.1 Algebraic Algorithms

The most popular algebraic method is the *Boolean difference method* developed by Sellers *et al.* [90]. In this approach, a Boolean formula is constructed for a combinational circuit as its output realizes a logic (Boolean) function. Let $F(x_1, x_2, \dots, x_n)$ be a logic function of the input Boolean variables x_1, x_2, \dots, x_n . The Boolean difference of $F(\mathbf{x})$ with respect to its input variable x_i is defined as

$$F(x_1, \dots, x_i, \dots, x_n) \oplus F(x_1, \dots, \bar{x}_i, \dots, x_n)$$

and is denoted by $\frac{dF(\mathbf{x})}{dx_i}$, where \oplus denotes the logical exclusive-OR (XOR) operation.

It can also be represented as

$$\frac{dF(\mathbf{x})}{dx_i} = F_i(0) \oplus F_i(1)$$

where,

$$F_i(0) = F(x_1, \dots, x_{i-1}, 0, x_{i+1}, \dots, x_n),$$

$$F_i(1) = F(x_1, \dots, x_{i-1}, 1, x_{i+1}, \dots, x_n),$$

The set of tests for x_i stuck-at 0 is

$$X_i \cdot \frac{dF(\mathbf{x})}{dx_i}.$$

where, X_i is the function representing the output of the sub-circuit with output at x_i .

Similarly, the set of tests for x_i stuck-at 1 is given by

$$\bar{X}_i \cdot \frac{dF(\mathbf{x})}{dx_i}.$$

The effect of two faults at the input of a logic circuit on its output can be analyzed by defining the double Boolean difference as

$$\frac{dF(\mathbf{x})}{d(x_i x_j)} = F(x_1, \dots, x_i, \dots, x_j, \dots, x_n) \oplus F(x_1, \dots, \bar{x}_i, \dots, \bar{x}_j, \dots, x_n)$$

Thus, test generation for multiple stuck-at faults can be generalized by using multiple Boolean differences [62].

Test generation using Boolean difference approach can be characterized as algebraic in a sense that it manipulates circuit equations to generate test patterns. Other algebraic test generation methods developed are the propositional method of Poage [81], the equivalent normal form [8], the cause-effect equation [17], the SPOOF procedure [28], and the structure description function [57]. All these methods derive equations for a fault-free circuit and manipulate the equations to generate test patterns. However, it is an arduous task to manipulate algebraic equations for large circuits and therefore, can not be generalized. Furthermore, the tedious nature of the algebraic manipulations of these methods lead to be impractical for test generation.

2.2 Structural Algorithms

A number of structural algorithms have been developed for test generation of combinational and sequential circuits. The test generation algorithms for combinational and sequential circuits are known as combinational and sequential algorithms respectively. These algorithms are briefly discussed in this section.

2.2.1 Combinational Algorithms

In combinational test generation algorithms, a test pattern is generated in two steps: (1) assign values to create a change (*i.e.* fault-effect) at the fault-site, and (2) search for consistent values on all signal lines in the circuit such that the fault-effect is successfully propagated to at least one of its primary output (PO). The most successful test generation systems are based on structural algorithms. Some of the notable ones are D-algorithm [85], Podem [45], Fan [33], [35], Tops [58], Socrates [88], [89], Cont [102], [103], and Est [41]. All these algorithms are briefly described in the following subsections.

The D-Algorithm

The D-algorithm is the first complete test pattern generation algorithm developed by Roth in 1966 [85] which introduces a D notation and a five-valued calculus in order to activate a single stuck-at fault and to propagate its fault-effect to at least one PO of the circuit. The five-valued calculus consists of logic values 0, 1, X (an unknown value), and two additional values D and \bar{D} . A D value signifies a logic value of 1 in the fault-free (good) circuit and 0 in the faulty circuit. A \bar{D} value signifies a logic value of 0 in the fault-free (good) circuit and 1 in the faulty circuit. The D-algorithm defines two D-cubes associated with each gate in the circuit: a *primitive D-cube* of a fault (pdcf) and a *propagation D-cube* (pdc). A pdcf is a set of inputs that produces an error signal on the output of that gate if it contains a fault. A pdc specifies the input values necessary to propagate an error signal on an input of a gate to its output. The pdcf and pdc of a two input AND gate are shown in Figure 2.1.

The basic operation of the D-algorithm is the repeated intersection of the D-cubes necessary to perform the tasks required to generate a test for a specific fault. The test generation is accomplished in three steps: fault sensitization, fault propagation

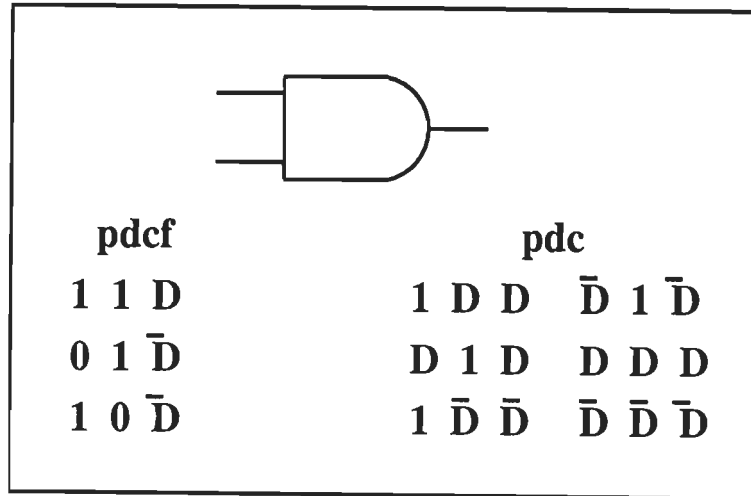


Figure 2.1: AND gate D-cubes.

and line justification. Fault sensitization is the process by which a circuit node is made to produce an erroneous value as a result of the fault. Sensitization is done by specifying an input combination for the circuit element containing the fault, using the pdcf that cause the output to take on the appropriate D value. Fault propagation is the process of propagating the D values to the primary outputs so that they can be monitored. The list of all gates whose output value is currently logical X but have one or more error signals (either D or \bar{D} s) on their inputs is called the D -frontier. Fault propagation process selects one gate from the D -frontier and assigns values to the unspecified gate inputs so that the gate output becomes D or \bar{D} . This process sensitizes all possible paths from the fault site to the primary outputs. This feature, referred to as *multiple-path sensitization* is necessary for the D -algorithm to guarantee its completeness. During fault sensitization and fault propagation, certain nodes of the circuit require to have specific values. Assigning values on the primary inputs to achieve the required value on the node is called line justification. The primary inputs to be used to justify a goal are usually determined by backtracing through circuit topology from

the node in question to the primary inputs. A value is assigned to one of these primary inputs and a simulation-like process, called forward implication, is performed to see whether the current input assignment while satisfying the goal is consistent or not. If it satisfies the goal, a test is generated which means that the fault is sensitized and the fault-effect (error) is successfully propagated to the primary outputs. However, if it does not, a different value is selected and the process is repeated until either the test is generated or in case of any inconsistency, the fault under consideration is declared as redundant fault.

To illustrate the D-algorithm, let us consider a combinational circuit as shown in Figure 2.2 which is to be tested for a stuck-at-1 fault on signal line j . According to the first step of the D-algorithm, the fault is to be sensitized by producing an error value, *i.e.* \bar{D} , on the line j and it is shown by test cube, TC0 in Figure 2.3. The fault is sensitized using a pdcf of the NOR gate (TC1), *i.e.* by setting $g = 1$. Using the backward implication procedure, both the inputs, a and b , of the AND gate g must have 1 value which means $a = b = 1$ and TC2 shows these assignments. In order to advance the D -frontier, a 0 value is required on k node which in turn implies that both the inputs, i and f , of the OR gate k must have 0 values (TC3). The 0 value on the output of the OR gate i implies $e = h = 0$ (TC4). In the justification step, h has to be justified to 0, which can be done by assigning either input of the AND gate h to a 0 value. By setting c to a value 0 (TC5), a test is generated by the D-algorithm.

Now consider the test cubes (propagation first) as given in Figure 2.3 for pdcf selection. When selecting the test cube TCb for the initial fault, its forward implication determines that the value on the signal lines k and l to be 1. Since error is not visible on the primary output l , no test is generated (TCc). In this case, the algorithm would backtrack to the last decision made and select the alternate choice to proceed further. In the D-algorithm, choices are available at many internal nodes in the circuit and

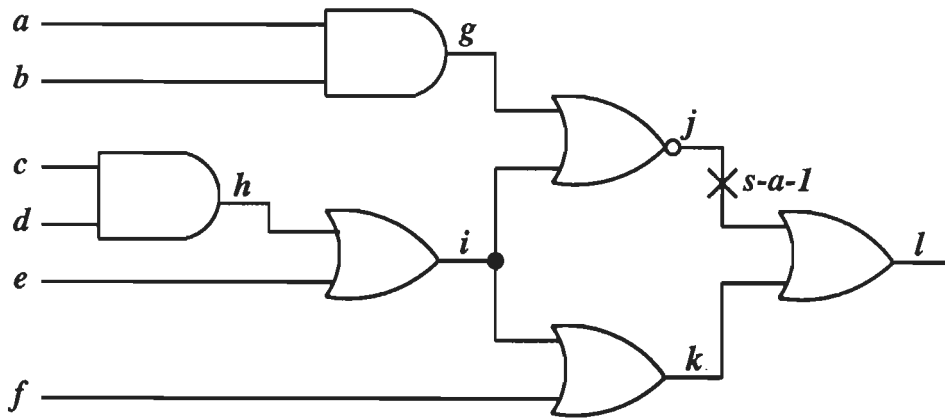


Figure 2.2: An example combinational circuit

	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>	<i>g</i>	<i>h</i>	<i>i</i>	<i>j</i>	<i>k</i>	<i>l</i>	
TC0										\bar{D}			← initial test cube
TC1							1	X	\bar{D}				← pick pdcf
TC2	1	1					1	X	\bar{D}				← imply from <i>g</i> to <i>a</i> and <i>b</i>
TC3	1	1				0	1	0	\bar{D}	0			← set objective <i>k</i> = 0 and imply <i>i</i> and <i>f</i>
TC4	1	1			0	0	1	0	0	\bar{D}	0	\bar{D}	← imply from <i>i</i> to <i>e</i> and <i>h</i>
TC5	1	1	0	X	0	0	1	0	0	\bar{D}	0	\bar{D}	← justify <i>h</i>

Test cubes (justification first)

	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>	<i>g</i>	<i>h</i>	<i>i</i>	<i>j</i>	<i>k</i>	<i>l</i>	
TCa										\bar{D}			← initial test cube
TCb							X	1	\bar{D}				← pick pdcf
TCc							X	1	\bar{D}	1	1		← imply <i>k</i> and <i>l</i> from <i>i</i> ; no test

Test cubes (propagation first)

Figure 2.3: D-algorithm example.

more than two choices may be present in the case of more than two-input gates in the circuit. Due to this fact, the size of the algorithm's search space increases and makes backtracking more complex.

In fact, the original D-algorithm does not mention about the order of the

processes – fault sensitization, fault propagation and justification – to be performed. However, the efficiency of the test generation depends heavily on the order of these operations and the most efficient order is determined by the circuit topology. As an example, in test generation for j stuck-at-1 fault in the circuit of Figure 2.2, if fault propagation is done before selecting a pdcf, the unique 0 value required on node i will be discovered and the pdcf for faulty gate will be fixed. Then the test generation can proceed without any possibility of backtracking. Subsequently developed algorithms attempted to reduce the size of the solution space that must be searched. Goel made an attempt in the same direction and developed Podem (**P**ath-**O**riented **D**Ecision **M**aking) algorithm [45] as D-algorithm turned out to be extremely inefficient in generating tests for the class of combinational circuits that implement error-correction and translation (ECAT) type functions [45].

The Podem Algorithm

Podem is an implicit enumeration algorithm in which all possible primary input patterns are implicitly, but exhaustively, examined as tests for a given fault. In Podem, the test generation problem is formulated as a search of the N -dimensional 0-1 state space constrained by a set of Boolean equations, where N is the number of primary inputs of a combinational circuit. To view the test generation as a finite space search problem, let us consider the combinational circuit shown in Figure 2.4, where g is an internal net and the objective is to generate a test pattern for the g stuck-at-0 fault. The state on g can be expressed as a Boolean function of the primary inputs, x_1, x_2, \dots, x_N . Similarly, each primary output ($y_j, j = 1, 2, \dots, M$) can be expressed as a Boolean function of the state on net g as well as the primary inputs x_1, x_2, \dots, x_N . Let

$$g = G(x_1, x_2, \dots, x_N)$$

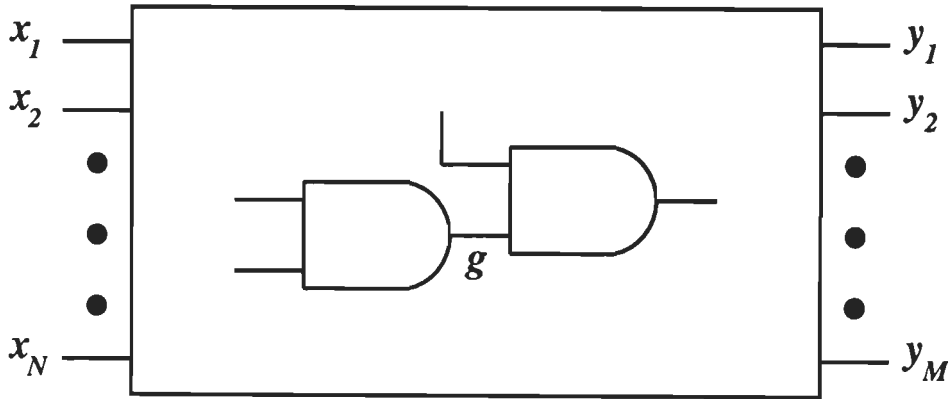


Figure 2.4: A generic combinational circuit.

and

$$y_j = Y_j(g, x_1, x_2, \dots, x_N)$$

where $1 \leq j \leq M$ and $x_i = 0$ or 1 for $1 \leq i \leq N$

The problem of test generation for g stuck-at-0 can be stated as one of solving the following set of Boolean equations:

$$G(x_1, x_2, \dots, x_N) = 1$$

$$Y_j(1, x_1, x_2, \dots, x_N) \oplus Y_j(0, x_1, x_2, \dots, x_N) = 1$$

for at least one j , $1 \leq j \leq M$ and $x_i = 0$ or 1 for $1 \leq i \leq N$

In order to generate test pattern for g stuck-at-1 fault, the same set of equations are to be solved except that G is set equal to 0. Hence, test generation can be viewed as a search of N -dimensional 0-1 state space subject to satisfying the above set of equations.

The Podem algorithm begins by trying to justify D or \bar{D} at the node under test, similar to the D-algorithm. In Podem, this justification is done by assigning values to primary inputs that affect the node in question. These primary inputs may

be found by backtracing through the circuit topology. When an input assignment is made, *i.e.* *branching* in the context of *branch-and-bound algorithms*, forward implication (a simulation-like process) is carried out to evaluate all node values implied by the assignment. In case of any inconsistency found due to this assignment in order to achieve the goal, the complementary value is tried. When both assignment choices at a node are rejected, the algorithm backtracks efficiently to the previous input assignment. The rejection of a primary input assignment results in a *bounding* of the decision tree, in the context of the branch-and-bound algorithms. This process finally results in an orderly search methodology that will implicitly search the entire solution space.

The search methodology of Podem is represented by a decision tree structure illustrated in Figure 2.5. After the value at the faulty node is justified, subsequent objectives are set up to propagate the *D*-frontier along a path or paths to the primary output(s). The exact order in which this process occurs is again implementation dependent. The important characteristics of Podem is that the strategy of assigning values only to primary inputs orders the search space. This procedure lets the search methodology prune the search space implicitly and increase efficiency.

To illustrate the test generation using Podem, let us consider the combinational circuit shown in Figure 2.2 with a slight modification that fault at node *j* is stuck-at-0 instead of stuck-at-1 fault. In order to generate test pattern for the fault under consideration, Podem orders the binary search space as shown in the form of a decision tree (Figure 2.5). A simple heuristic is used to construct this search space that always prefers to assign a logical 1 value on a primary input. Initially, all primary inputs of the example circuit are at logical *X* value, *i.e.* unknown value. Using this simple heuristic, the node *a* is assigned to a logical 1 value which also provides consistency in the circuit. Now the assignment of a logical value 1 on node *b* in the circuit leads to inconsistency, therefore, all assignments with $b = 1$ in the solution space should be

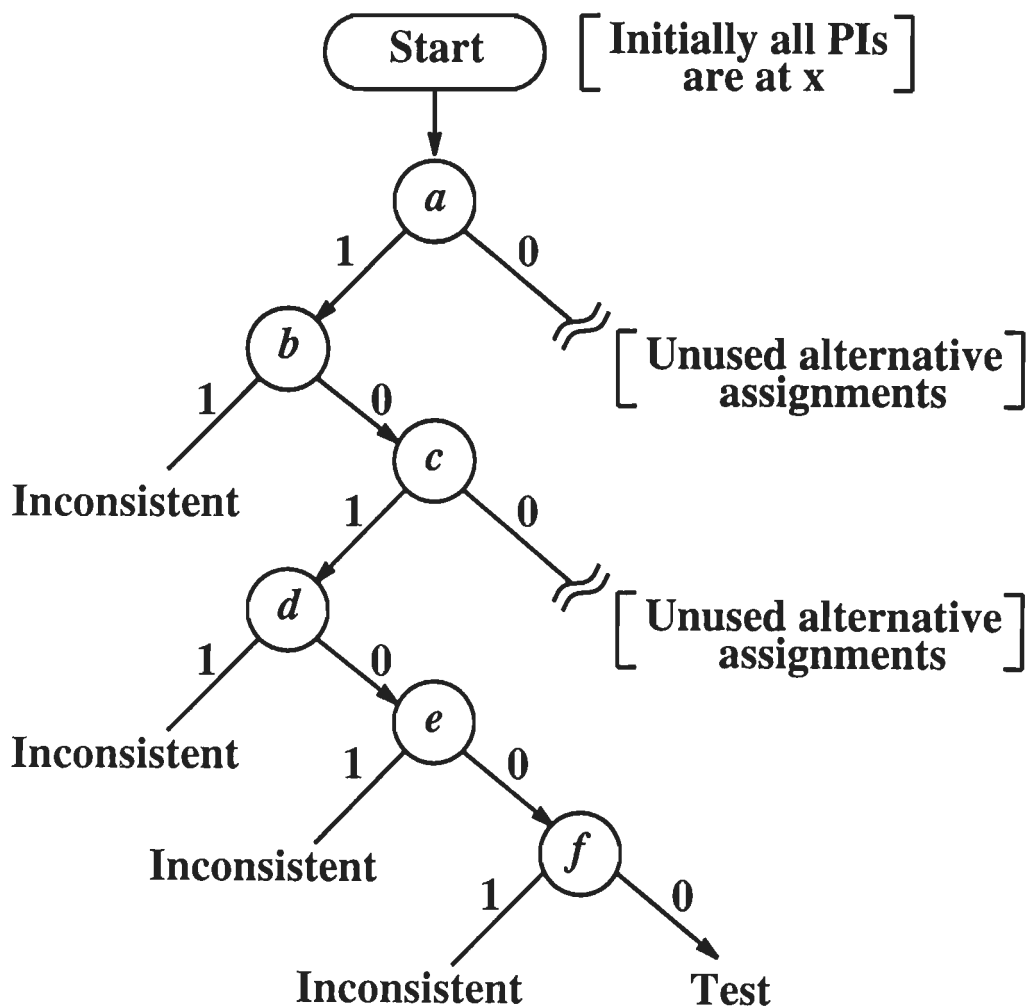


Figure 2.5: Podem search-space graph.

pruned as shown in the decision tree. Hence, the remaining alternate value on *b*, *i.e.* $b = 0$ is tried and the decision tree is further explored. The entire process is iteratively continued till a test pattern is found. For the example circuit, the test pattern for the given fault *j* stuck-at-0 is thus obtained by assigning $a = 1, b = 0, c = 1, d = 0, e = 0$ and $f = 0$ as shown in Figure 2.5. Podem is a complete test generation algorithm in that a test pattern will be found if one exists.

The Fan Algorithm

In Podem, the test generation may become time-consuming for the circuits with large number of fan-outs and therefore, needs improvements in order to handle such circuits. The Fan (**F**an-out oriented test generation algorithm) [33], [35] is an improved version of the Podem algorithm in the sense that the test generation process is accelerated by incorporating several techniques. The important goals of Fan are to reduce the number of backtracks in the binary decision tree and shorten the process time between backtracks. The Fan algorithm introduces the following two major extensions to the backtracing concept of Podem.

- Termination of backtracing in Fan at *internal lines* than stopping at primary inputs.
- Usage of a *multiple-backtrace* procedure to simultaneously satisfy a set of objectives rather than trying to satisfy one objective at a time.

Fan defines internal lines as bound, free and head line where it stops backtracing. A *bound line* is a signal line reachable from some fan-out point, *i.e.* there exists a path from some fan-out point to a signal line. A signal line that is not bound is said to be a *free line* and a *head line* is a free line that is adjacent to some bound line.

In the circuit shown in Figure 2.6, signal lines i_1 , i_2 , j , k , and l are bound lines. Lines a , b , c , d , e , f , g , h , and i are all free lines. Among the free lines, g , i , and f are head lines of the circuit since these lines are adjacent to the bound lines j , i_1 or i_2 , and k respectively. As far as the value assignments on these lines are concerned, free lines may have a uniquely assigned value whereas bound lines can not have unique (independent) values assigned to them. By definition, head lines can also be assigned values arbitrarily because they are free lines and can always be independently justified.

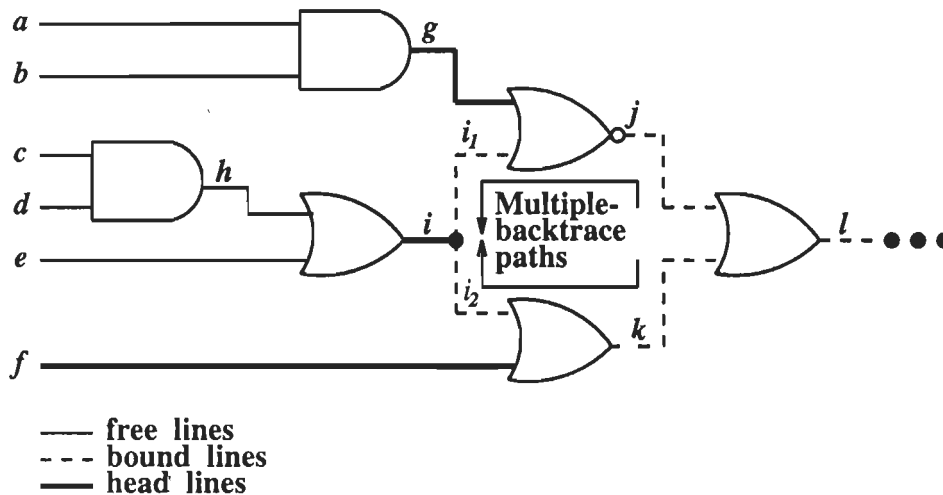


Figure 2.6: A Fan example.

Therefore, they can be treated as primary inputs in the justification process. Once a test is found by treating headlines as primary inputs, the values on them can be justified at the end of the test generation process.

Multiple-backtrace procedure is an important characteristic of the Fan algorithm which handles re-convergent fan-out branches buried in the circuit. This procedure reduces the number of backtracks that must be made in the search. In the multiple-backtrace of Fan, an objective is defined by a triplet

$$(s, n_0(s), n_1(s))$$

where s is an objective line, $n_0(s)$ is the number of times the object value 0 is required to be set on line s , and $n_1(s)$ is the number of times the object value 1 is required to be set on line s . The multiple-backtrace procedure starts with a set of *initial objectives* and defines different set of objectives. Starting from the set of initial objectives, a set of objectives that appear during the procedure is called a set of *current objectives*. A set of objectives that is obtained at head lines is called a set of *head objectives*. A set

of objectives on fanout points is called a set of *fanout-point objectives*.

Multiple-backtrace procedure of Fan is illustrated here by considering the circuit shown in Figure 2.6 as an example. Assume that this circuit is a part of some larger circuit and during the test generation process, certain value is necessary at node l in the circuit. In Podem, a single backtrace could be made along the path $l \rightarrow j \rightarrow g \rightarrow a, b$. The values for inputs a and b could be chosen so that the goal is satisfied with a unique value on nodes i and k . Suppose the value on k can not be achieved with the value chosen for i . Then, a significant amount of backtracking could result in the search of binary decision tree. In Fan, a multiple-backtrace procedure would backtrace both the $l \rightarrow j \rightarrow i$ and $l \rightarrow k \rightarrow i$ paths (Figure 2.6) and determine the value needed at i to satisfy the goal. This value would then set a requirement for the justification of the value at node l . This in turn would increase the efficiency of the Fan algorithm significantly for the combinational circuits with numerous buried re-convergent fan-outs.

A CAD tool [92] has been developed to generate test patterns automatically for a given set of faults in a combinational logic circuit. The block schematic of the CAD tool for Automatic Test Pattern Generation (ATPG) is shown in Figure 2.7. The Fan algorithm has been implemented as a core of the CAD tool for ATPG. Fan algorithm was chosen because it is significantly faster than Podem algorithm as the search-space ordered by the Fan is relatively less than the Podem. The search-space ordered by the Fan may be further pruned with the use of efficient heuristics. These heuristics basically provide help in accelerating ATPG process by reducing the number of backtracks or bad decision. Heuristics based on controllability/observability (C/O) measures, such as Scoap [47], [48], Savir's cutting algorithm [86], Predict [91] *etc.* have been considered in order to couple within the ATPG framework.

In the heuristic development phase of the CAD tool, it is found that the popular Scoap measures lack essentially in the assumption of signal independence of fan-out signals [5]. It gives rise to errors when re-convergence is encountered. Moreover, they are also unable to provide a way to calibrate the circuit controllability/observability information in an absolute way. A new efficient heuristic based on probabilistic estimation has been proposed which considers the correlation of the inputs in a re-convergent fan-out path and provides more accurate calculations of node controllabilities than Predict.

The quality of test patterns generated by the CAD tool is determined by a fault simulator which has been incorporated with in the ATPG framework and depicted in Figure 2.7. Among the various fault simulation methods [68], [82], [106] developed in the past, a **Parallel Pattern Single Fault Propagation** (PPSFP) algorithm [106] has been implemented in this CAD tool. The fault simulator provides the information about the quality of the test patterns in terms of fault coverage, undetected and undetectable faults. Moreover, this simulator may be used for automatic fault dictionary generation for diagnosis of complex digital systems [93].

The Socrates Algorithm

Schulz *et al.* [88] developed an **Automatic Test Pattern Generation** (ATPG) system called Socrates based on the Fan algorithm. This ATPG system incorporates several distinct techniques to accelerate the ATPG process for combinational or scan-based circuits. The application of these techniques results in an earlier recognition of conflicts and redundancies and in a reduction of the number of backtracks. In particular, these techniques provide improvements in implication, unique sensitization, and multiple backtrace procedures. Among these techniques, the most important one is a *learning*

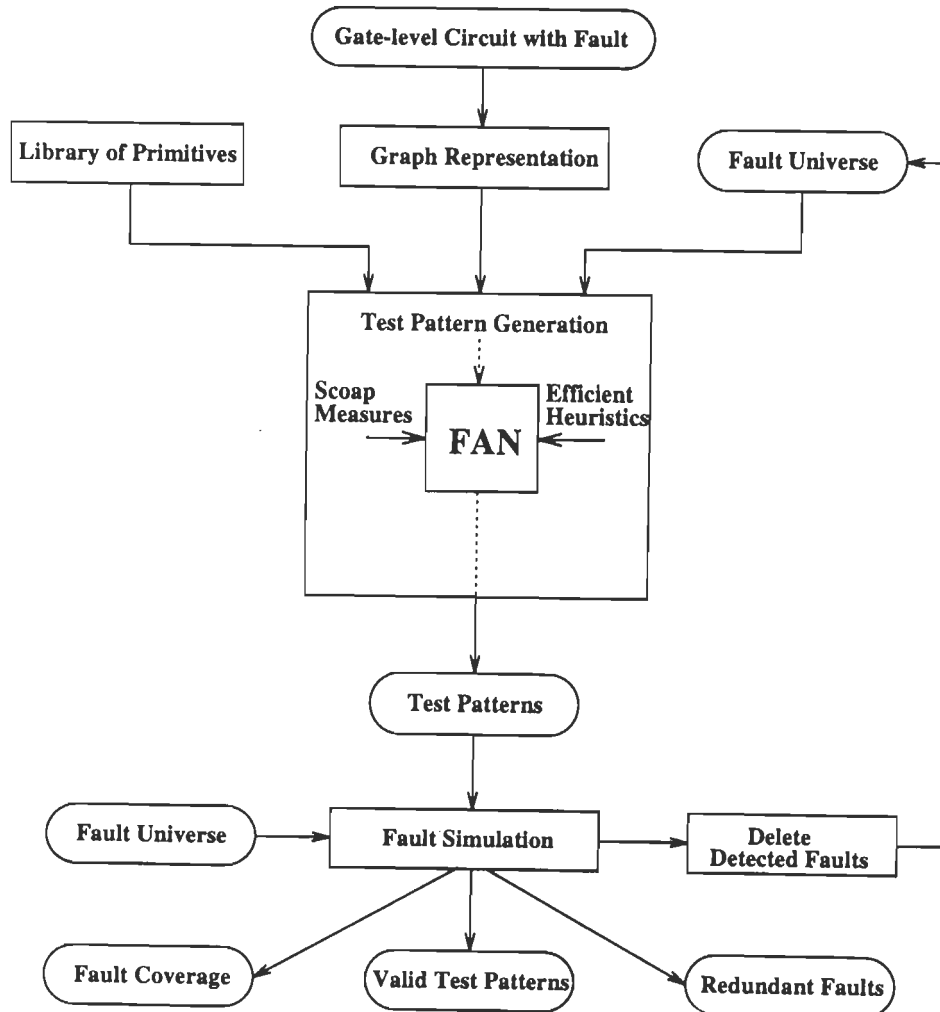


Figure 2.7: A block schematic of the ATPG tool.

procedure which is performed during the pre-processing phase of Socrates. The overall strategy of the learning procedure is to assign a logic value to a certain signal of the circuit, to perform all implications from that assignment and to learn from the results of the implications. The implications are performed by using a *logical identity* called *contrapositive*, *i.e.*

$$(P \Rightarrow Q) \Leftrightarrow (\neg Q \Rightarrow \neg P)$$

where P and Q are arbitrary assertion variables. In order to decide efficiently whether an implication is worthwhile learning or not, the following *learning criterion* is used.

- Let i be the signal at which the current learning step is initialized by the assignment $i = v_i$ and let j be a signal at which a fixed logic value v_j (0 or 1) is assigned during the implication procedure, *i.e.*

$$(i = v_i) \Rightarrow (j = v_j)$$

- Furthermore, let j be the output of a gate, say g . If v_j requires all inputs of g to have non-controlling values and a forward implication has contributed the assignment $j = v_j$, then the implication

$$(j = \bar{v}_j) \Rightarrow (i = \bar{v}_i)$$

is considered to be worthwhile learning.

In Socrates, the test generation problem is also viewed as a search problem for which algorithms usually build a decision tree and apply a backtracking procedure [33], [35], [45] in order to find a solution of the problem. However, the decision tree can be represented as a triangle shown in Figure 2.8 and can be divided into a *solution area* and several *non-solution areas* as shown in the triangle. It is important to note that whenever a search procedure enters into one of the non-solution areas, no solution can be obtained by making additional assignments on the primary inputs of the circuit and only possibility to leave the non-solution area is through backtracking.

Majority of the deterministic test generation algorithms are not able to identify the entire non-solution areas and identify only parts of them. During the search procedure, whenever a test generation algorithm enters into an identified non-solution area, immediately backtracking will take place as soon as it is recognized that a solution can not be found with the current assignment. Contrary to this, the unidentified

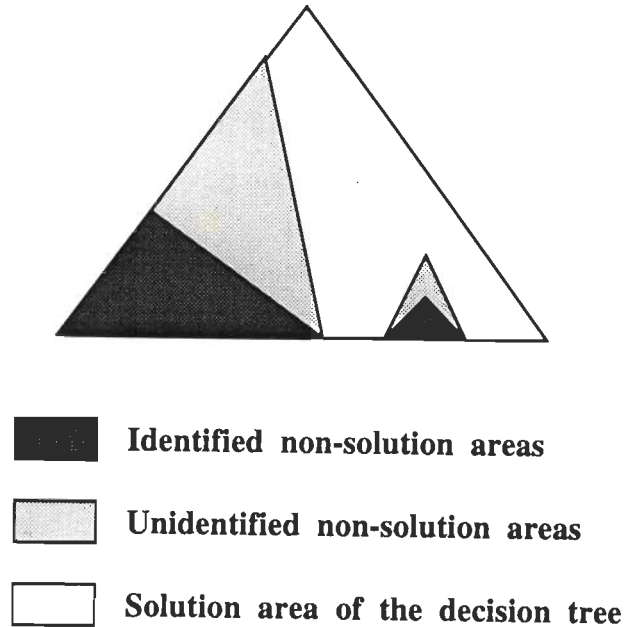


Figure 2.8: Abstract problem representation of deterministic test generation.

non-solution area causes the most serious problem to the test generation algorithms as it will require a lot of backtracking in order to leave the non-solution area. Hence, the following two global goals are aimed in the development of Socrates:

1. Minimization of the unidentified non-solution areas, and
2. Avoidance of entire non-solution areas during the search process.

The Est Algorithm

The Est algorithm [41] is an acronym for **E**quivalent **S**Tate Hashing algorithm which accelerates any combinational circuit test generation algorithm. Est detects *equivalent search states*, which are saved for all faults during test pattern generation. The search space is reduced by using **B**inary **D**ecision **D**iagram (BDD) further than other algorithms can [33], [35], [45], [58], [85]. This algorithm made the following contributions:

- It characterizes the search state of the test generation algorithm using the *E-frontier*, a cut-set of the circuit induced by a set of primary input assignments. The frontier is a partition between the circuit part labeled with 0, 1, D , and \bar{D} signals and the part labeled with X signals.
- It uses the matching of E-frontiers to cause early backup from infeasible searches or early search termination with a test pattern.
- It is the first test pattern generation algorithm which uses knowledge about the search space for a prior fault to accelerate search for a test of the current fault.
- It also uses first time the knowledge of opportunistically-discovered redundant faults in the circuit to further reduce the search space for later faults.

Est accelerated the TOPS [58] test generation algorithm 328 times and enabled it to handle all faults in the ISCAS '85 combinational benchmark circuits. Est also accelerated the Socrates algorithm 5.81 times on the same benchmark circuits. Extensions to the Est algorithm are reported in [42].

Other ATPG Algorithms

The TOPS algorithm [58] uses the Fan's concept of the independent justification of headlines a step further by introducing a new type of node called a *basis node*. A basis node is the absolute dominator of all nodes that precede in the circuit graph. The concept of dominator nodes has come from graph theory. A node dominates another node if all paths from that node to the root pass through the dominator. All free lines and head lines in a circuit are basis nodes as they are all points of total re-convergence. In the circuit shown in Figure 2.6, the free lines a through i are basis nodes. Since node l is a point of total re-convergence, it is also a basis node. As a basis node, node

l can have any value required to generate a test in the remainder of the circuit, and the justification of that value at the total-re-convergence node l can be postponed in the same way as the justification of a head line.

A *backtrace-stop line* used in Fast (**F**ault-oriented **A**lgorithm for **S**ensitized-path **T**esting) algorithm [2] represents another generalization of the head-line concept, based on an analysis that is both topological and functional. In Fast, a line l is a backtrace-stop line for value v , if the assignment $l = v$ can be justified without conflicts. For example, in Figure 2.6, l is a backtrace-stop line for value 0, because $l = 0$ can be justified by $a = b = 1$ and $f = 0$ that is without assigning any line with re-convergent fanout.

The Cont algorithm [102] implements a different approach in which if the target fault is not detected by the current input vector, Cont switches over to a new target fault. Candidate faults for target switching are identified by interleaving fault simulation steps with incremental test generation steps. In the second version [103] of the Cont algorithm, all unspecified inputs are randomly assigned prior to fault simulation.

Other test generation methods that combine features of deterministic algorithms with those of RTG are Raps [44] and Smart [2]. The Raps (**R**andom **P**ath **S**ensitization) algorithm attempts to create random critical paths between PIs and POs. Initially all values are at X . Raps starts by randomly selecting a PO, say z and a binary value v . Then the objective (z, v) is mapped into a PI assignment (i, v_i) by a *random backtrace* procedure *Rbacktrace*. Rbacktrace is similar to the backtrace procedure used by Podem except that the selection of a gate input is random. In Podem, this selection is guided by heuristics. The assignment $i = v_i$ is then simulated using 3-value simulation and the process is repeated until the value of PO z becomes binary.

After all the POs have binary values, a second phase of Raps assigns the PIs (if any) with X values. The entire procedure is repeated to generate tests as needed. As a result, test sets generated by Raps are smaller and achieve a higher fault coverage than those obtained by RTG.

Smart (Sensitizing Method for Algorithmic Random Testing) is also a combined deterministic and random test generation method which corrects the problems encountered by Raps. Smart generates a vector incrementally and relies on a close interaction with fault simulation. The partial vector generated at each step is simulated using *critical-path tracing* [1] which guides the test generation process. This interaction is based on two by-products of the critical-path tracing: *stop lines* and *restart gates*. Stop lines delimit areas of the circuit where additional fault coverage can not be obtained, while restart gates point to areas where new faults are likely to be detected with little effort. Experimental results reported in [2] show that Smart achieves higher fault coverage with smaller test sets and requires less CPU time as compared to Raps.

2.2.2 Sequential Algorithms

Test generation for sequential circuits is recognized to be a more complex problem than for the combinational circuits. Primarily, most sequential test generation methods have been devised on the basis of the fundamental combinational algorithms like D-algorithm [85] and Podem [45]. The extended D-algorithm of Kubo [63] and Putzolu and Roth [83] and the nine-valued model of Muth [75] are some basic procedures that are good for use in the test generation for sequential circuits. However, these algorithms are found difficult to implement and highly inefficient for large practical sequential circuits.

Considerable research has been done in the development of sequential test generation methods and a number of sequential test generators have been reported in

the literature. The taxonomy of the sequential test generation methods is given below:

1. Time frame expansion based,
2. Simulation based, and
3. Rule based or expert systems.

In time frame expansion based test generation methods, a synchronous sequential circuit, S is transformed into a combinational iterative array by cutting the feedback loops of the clocked flip-flops as shown in Figure 2.9(b). Each cell $C(i)$ of the array is identical to the combinational circuit C of Figure 2.9(a). In this transformation, the clocked flip-flops (FF) of S are modeled as combinational elements $F(i)$ which are referred to as *pseudo flip-flops*. In Figure 2.9(b), $C(i)$ and $F(i)$ correspond to time frame i . Suppose that an input sequence $x(0) x(1) \dots x(k)$ is applied to the sequential circuit S in initial state $y(0)$ and S generates the output sequence $z(0) z(1) \dots z(k)$ with the state sequence $y(1) y(2) \dots y(k+1)$. Then the iterative array will generate the output $z(i)$ from cell i , in response to the input $x(i)$ to cell i ($1 \leq i \leq k$). This modeling technique maps the time domain response of the sequential circuit into a space domain response of the iterative array. A time frame expansion of the example sequential circuit given in Figure 2.10 is illustrated in Figure 2.11. Due to this transformation, the test generation methods developed for combinational circuits could be extended to the sequential circuits. A similar technique exists for asynchronous sequential circuits.

The time frame expansion based test generation can be further subdivided into algorithms that do test generation in

- forward time,
- reverse time, and

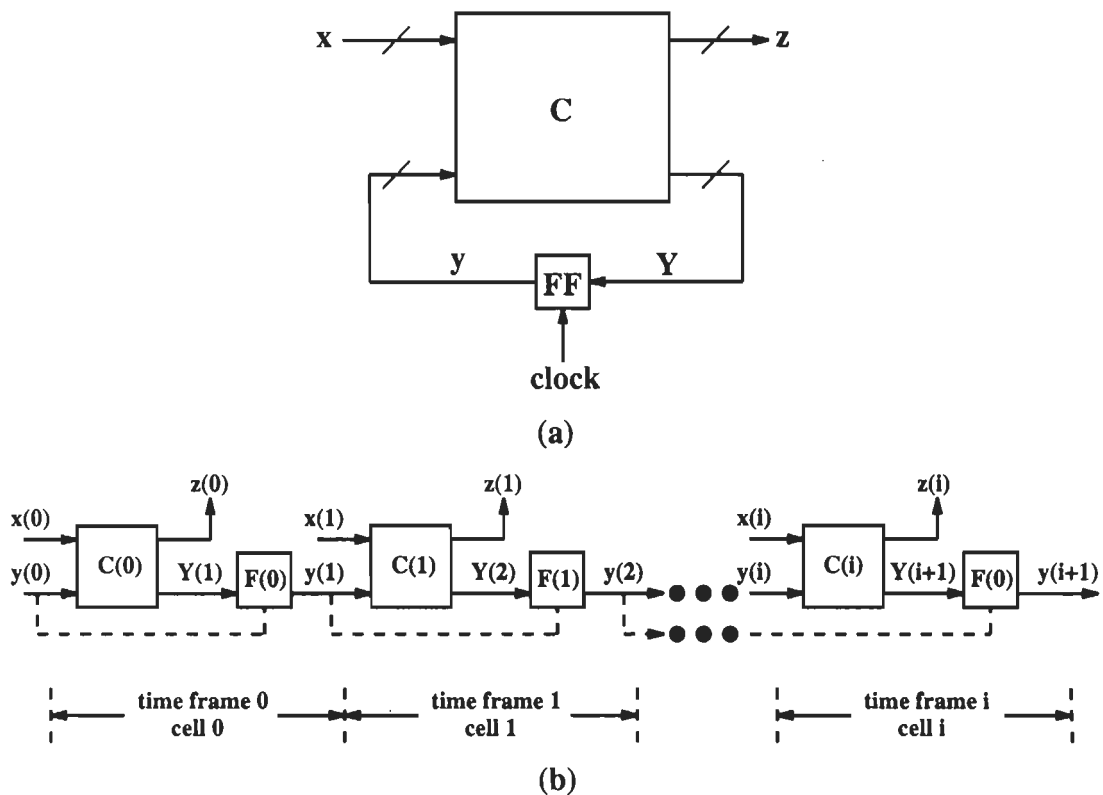


Figure 2.9: (a) Canonical structure of a synchronous sequential circuit, and (b) its combinational iterative array model.

- a combination of forward time and reverse time

The extended D-algorithm and Muth's method both perform test generation in forward time. Sequential test generation algorithms based on path sensitization methods in a multiple time frame environment are reported in [12], [25], [26], [49], [72], [73]. The methods proposed in [25], [26], [72], [73] rely on reverse-time processing of the circuit. In reverse-time processing, an output for the target fault is determined first and then the algorithm works backward in time from the primary output to the fault site and finally to an uninitialized state. STG1 [72], STG2 [25], STG3 [72], and Dust [49], use multiple path sensitization based on D-algorithm framework, while Marlett [73]

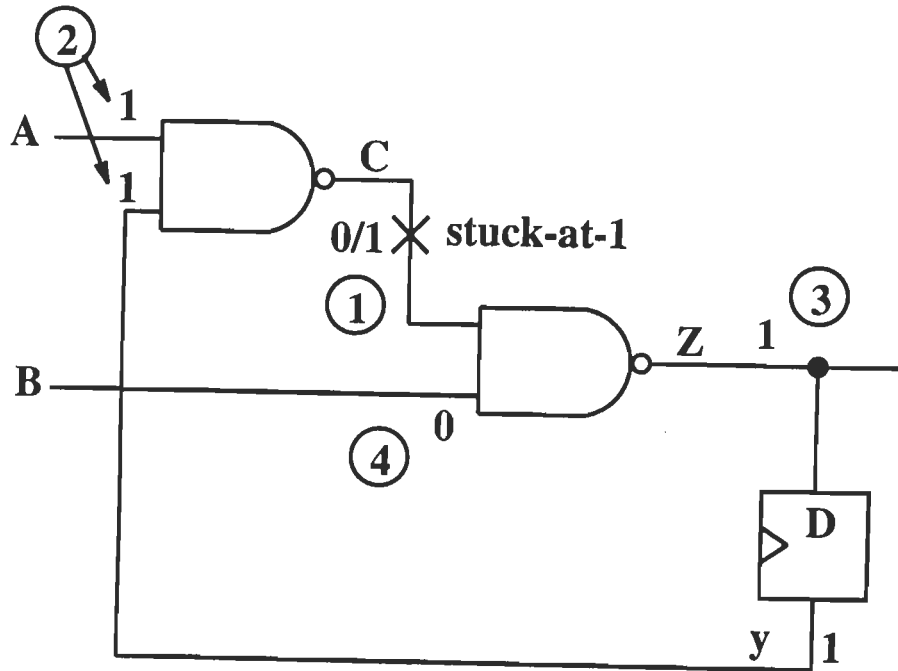


Figure 2.10: An example circuit for sequential test generation.

developed the algorithm based on a single path-sensitization technique combined with a 16-valued logic model and a reevaluation method. The advantage of the reverse-time processing methods is that they are memory efficient since only the current timeframe and previous timeframe need to be in memory at a time. The disadvantages of these methods are that they are unnecessarily complex and inefficient because they work backward in time and are D-algorithm based. Some of the disadvantages are overcome by Essential [12] which uses static and dynamic information during test generation and information from the preprocessing phase to keep the search space as small as possible.

Hitest [15], Stallion [71], Steed [40] and Fastest [56] are the only known sequential circuit test generators that make use of Podem which has been much ignored as the basis for sequential circuit test generation. Hitest is a knowledge-based interactive test generation system and uses Podem for one timeframe, treating flip-flop outputs as

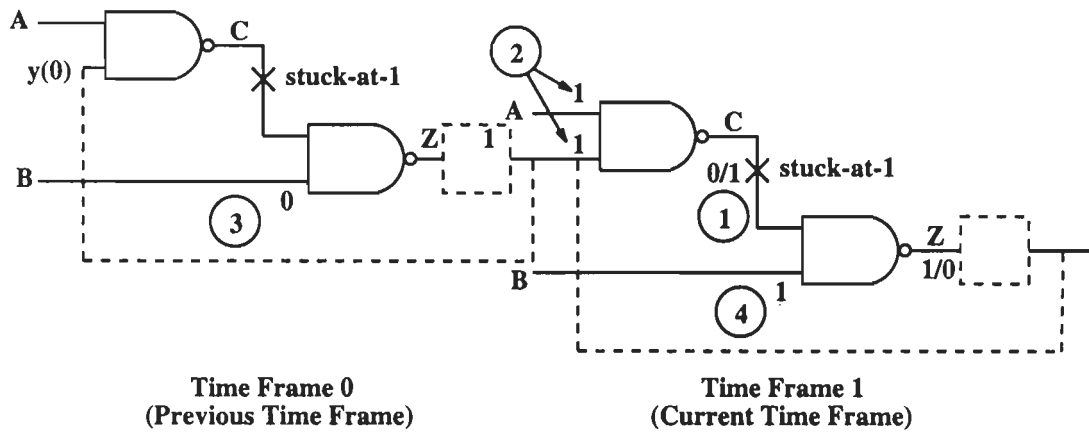


Figure 2.11: Time frame expansion of the example circuit for sequential test generation.

pseudo inputs and flip-flop inputs as pseudo outputs of a single timeframe. In Stallion, Podem is used for a forward enumeration process to excite a fault and propagate it to a primary output. The results of this phase are an initial state vector and a test sequence. Stallion also uses Podem as a backward justification algorithm, which is used to obtain a transfer sequence to bring the circuit from a reset state to the initial state, but assumes the circuit to be fault-free. Steed is an improved and substantially modified version of Stallion and obtains a transfer sequence in a more efficient manner by constructing ON and OFF arrays for each flip-flop input using Podem. Stallion and Steed are very efficient in the forward enumeration process, but the backward justification can be quite inefficient. Further, the assumption of a fault-free circuit during justification leads to the previous state information (PSI) problem which is discussed in [56]. Fastest exclusively operates in forward time and correctly solves the PSI problem since all state information concerning the faulty circuit is considered unknown at the start of test generation. A key factor in Fastest's performance was found to be an initial timeframe algorithm which determines the number of timeframes to begin test generation and the timeframe to attempt to excite the fault. The fastest has been found

successful in reducing the length of test sequences and CPU time from test generation.

In the simulation approach to sequential circuit test generation, Contest algorithm [27], [6] based on a concurrent fault simulator has been developed. The concurrent fault simulator allows Contest to generate tests for a group of faults rather than a single target fault in the early stages of test generation. It is then able to switch phases to single-fault test generation as the fault list gets reduced. The main advantage of a simulation-based test generator such as Contest is that there is no restriction on the number of timeframes allowed during test generation. However, there can be very large memory demands early in the test generation process, since the effect of every fault in the fault list must be stored for every node in the circuit. Moreover, memory demand decreases as the fault list is reduced, but the inefficiency of Contest lying in early stages affects its average performance rather adversely.

2.3 Parallel Processing Techniques

All the ATPG algorithms described in the previous sections share one common theme that they are developed to run on conventional uniprocessor computers and therefore, referred as serial ATPG algorithms. Although, much effort has gone into increasing the efficiency of these serial ATPG algorithms, the overall gains achieved through these developments have not kept pace with increasing circuit size, and computation times are still excessive. However, the computation time can be reduced simply by using a faster machine. The availability of affordable parallel machines and distributed network of idle workstations in most of the VLSI-CAD environments has opened a new front for the development of efficient parallel/distributed ATPG algorithms in order to harness the computational power of these machines. In the recent past, several techniques [60] have been developed to parallelize the compute-intensive ATPG process. These techniques

fall into five major categories as given below:

- Fault Partitioning,
- Heuristic Parallelization,
- Search-space Partitioning,
- Algorithmic Partitioning, and
- Topological Partitioning.

2.3.1 Fault Partitioning

Fault partitioning is the most basic parallelization method in which the fault list is partitioned for distribution among processors. In this technique, each processor independently generates tests for each fault on its portion of the fault list until all tests have been generated. So, if the fault list is divided carefully, each processor will have roughly the same amount of work and will finish in about the same time. However, in practice, optimal partitioning of the fault list is not easy to do a priori and therefore, the task scheduling can be done dynamically with each processor requesting a new fault from a master scheduler whenever it is idle. Dynamic scheduling requires increased communications overhead because of the requests from idle processors for new faults to process.

Patil and Banerjee [78] analyze fault-partitioning issues in an integrated parallel test generation/fault simulation environment. In their system, fault simulation is performed every time after a processor generates a test for a specific fault, say f_i . Fault simulation determines other faults covered by the generated test vector. If another fault, say f_j , is covered by the test vector, it must be removed from the fault

list of all the processors to which f_j has been assigned and for that a message must be sent to all the processors instructing them to remove f_j from their fault list. This increases the communication overhead and reduces the possible speedup. Patil and Banerjee presented results from several methods of partitioning faults across processors. These methods include random partitioning, partitioning by input and output cones and mandatory constraint propagation. A combined static and dynamic load-balancing technique was also reported by Patil and Banerjee [78] in which initially, faults are allocated to a processor using one of the methods mentioned above. If a processor succeeds in generating tests for all of the faults in its list, it requests work from the processor with the largest remaining fault list. This processor sends half of its list to the idle processor. This load-balancing scheme results in high message traffic only in the end of the test generation process. This combined technique resulted in near linear speedup for up to 16 processors on the largest benchmarks circuits.

Fujiwara and Inoue [36] developed an analytical method to calculate the optimal granularity and speedup ratio of a fault partitioning system. Here, optimal granularity refers to the size of the fault list allocated to each processor. The fault-partitioning system was implemented on a network of Sun workstations and obtained experimental results for optimal granularity and maximum speedup which showed the same trends as the analytical results, but the actual performance was less than predicted.

Major disadvantages of fault-partitioning is the long setup time for a message-passing system. The whole ATPG program and circuit database must be loaded into the memory of each processor across the message fabric. If the total amount of work that can be divided among the processors is large, *i.e.* when the fault list is long, then the percentage of time spent on setup can be kept small and this scheme has promise. Moreover, the method performs poorly if most of the processing time is spent only for a

few hard-to-detect faults. The experimental results published for systems based on this technique show that linear speedup is possible only for a small number of processors, usually less than 10 [36], [78].

2.3.2 Heuristic Parallelization

Heuristics are used to guide ATPG process and it has been found that many heuristics will produce a test for a given fault within some computation time limit while other heuristics fail to do so [24]. These complementary heuristics can be used in a multiprocessor system to aid ATPG. Chandra and Patel [23] used two basic strategies: a variation of the fault-partitioning scheme and *concurrent parallel heuristics*. The variation of the fault-partitioning method is termed as uniform partitioning in which the fault list is divided among the processors and each generates tests for the faults on its own portion of the list. However, multiple heuristics are used in sequential order to generate the tests, and if a heuristic fails to generate a test within a time limit, that heuristic is discontinued and the next one on the list is begun. This scheme is slightly better than the fault-partitioning scheme described above because the multiple heuristics shorten the test generation time for hard-to-detect faults. Chandra and Patel [23] found that this strategy produces almost linear speedup for a system of five distributed-memory processors.

In the concurrent parallel heuristic method, the ATPG system is required to have $k \times h$ processors, where h is the number of different heuristics available. If k equals 1, each processor computes a test for the same fault using one of the h heuristics. In the case of the successful generation of a test for the fault by a processor, it sends a “*stopwork*” message to the other processors in the cluster and they stop processing that fault. A new fault is selected from the list and the process begins again. When

k is greater than 1, the processors are clustered into groups of h and each cluster works on a separate fault. Hence, the system is actually using a combination of the fault-partitioning and heuristic parallelization schemes.

The major drawback of the concurrent heuristic method is that for each fault, the work of the $h - x$ processors is wasted, where x is the number of heuristics used by the uniform method for generating a test. Chandra and Patel [23] found that for most of the benchmark circuits the concurrent heuristic method generally does not perform well as compared to the uniform partitioning method. In some cases, the concurrent heuristic method, on a system of five distributed-memory processors, produced no speedup over a uniprocessor system.

2.3.3 Search-Space Partitioning

Search-space partitioning is a method which parallelizes the work on a single fault by dividing the search space into independent parts and evaluating them concurrently. This approach implements the popular branch-and-bound method in parallel and involves concurrent evaluation of subproblems. The search-space partitioning technique is also referred to as OR parallelism. Patil and Banerjee [77], [79] have developed a parallel branch-and-bound method for the ATPG problem using OR parallelism. Their method is based on Podem which orders the search space that can be divided easily. The process of dividing a search tree is illustrated in Figure 2.12. Initially, the search space belongs to a processor X that is divided into two parts for processors X and Y. It may be noted that the processors are in fact working on disjoint search spaces. During test generation, if processor X finds a conflict, it backtracks and tries an alternate value for input a . In case processor Y finds a conflict, the processor Y backtracks and tries an alternate value for input c . Hence, each processor will backtrack to a different place

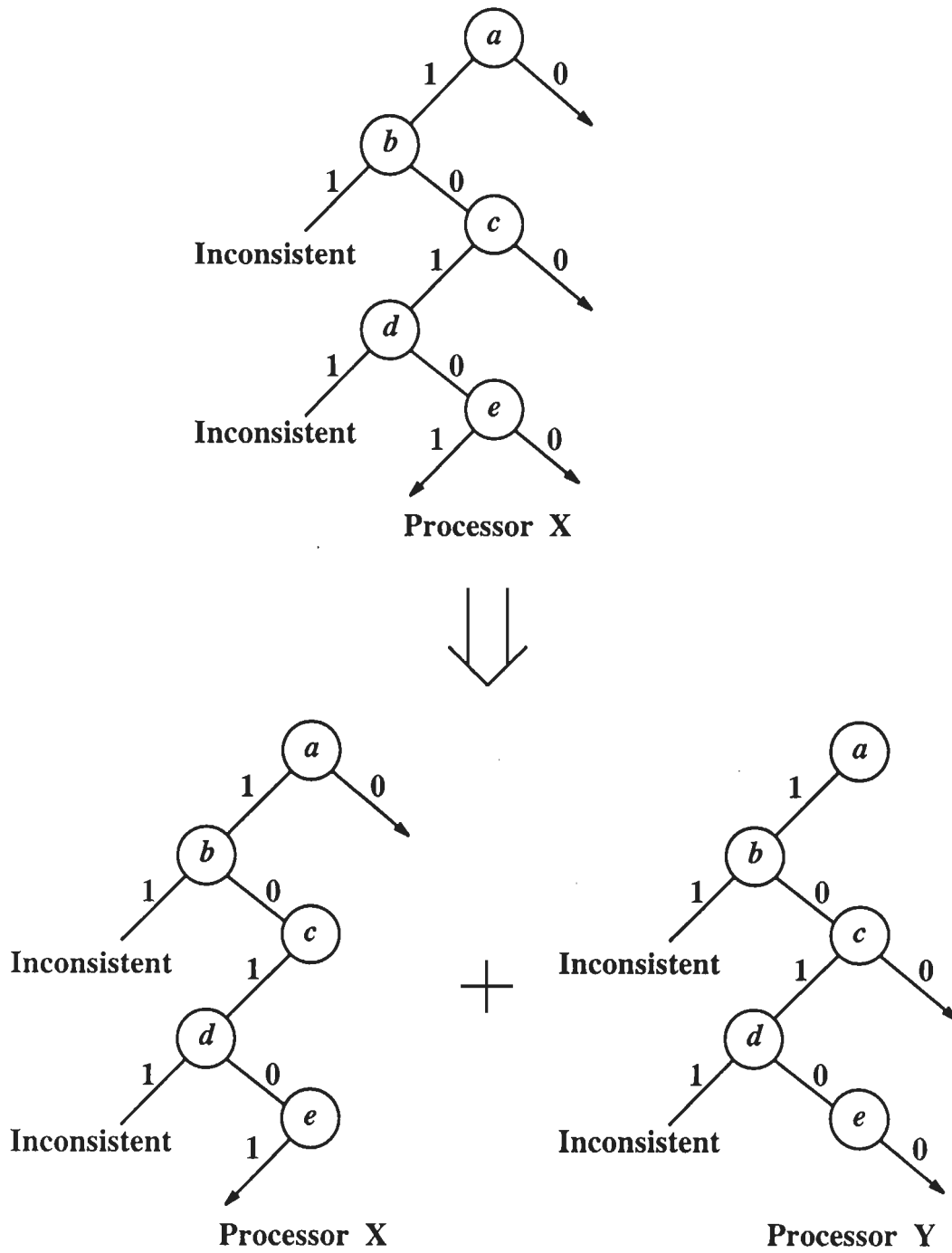


Figure 2.12: Search-space partitioning

in the search space. This approach keeps the current search space as large as possible making the search more efficient.

Patil and Banerjee [77] implemented the search-space partitioning scheme on a 16 node Intel iPSC/2, *i.e.* a hypercube message-passing machine. The results presented by them showed that the parallel algorithm runs much faster than the uniprocessor implementation and exhibits nearly linear speedup for up to 16 processors. The major problem with this scheme is that the long setup time is required. The entire circuit database and ATPG program must be made to reside into each processor's memory. Since processors are dedicated to only one independent task that does not change, there is very low communications overhead and greater efficiency in this scheme. The search-space partitioning technique is found most appropriate for circuits that contain a small fraction of hard-to-detect faults, since test generation of these faults is a compute-intensive task. This technique is ideally suited to message-passing systems because of its course-grained parallelism. Motohara *et al.* [74] also presented a search-space-division method for hard-to-detect faults. They have implemented their technique on a Links-1 multimicroprocessor system. Using the search-space-division method, linear speedup for up to 50 processors is reported to have achieved on several benchmark circuits with up to 1000 gates.

2.3.4 Algorithmic Partitioning

Algorithmic partitioning allows more than one processor to work simultaneously on finding a test for a single stuck-at fault. It uses the divide-and-conquer approach, dividing the process into smaller independent subtasks that can be assigned to separate processors to be executed in parallel. This method of parallelization is also known as AND parallelism or *functional partitioning*. Most of the serial ATPG algorithms

discussed in the previous sections are difficult to parallelize functionally. The few subtasks that can be identified, such as fault sensitization and path sensitization, are not independent. Motohara *et al.* [74] presented a parallel scheme which uses a type of functional partitioning to remove the easy-to-detect faults from the fault list. In this scheme, the procedure of removing the easy-to-detect faults is carried out before the parallel method for hard-to-detect faults discussed in the previous subsection is run. A linear speedup for up to 10 processors is achieved during algorithmic phase. Lover (**L**ogic **v**erification) system [70] developed by Ma *et al.* is also based on a functional-partitioning approach. Although logic verification is a different problem than ATPG, with different objectives and constraints, some of the steps required for both these problems are similar. The novelty about the Lover system is that it includes a dynamic scheduling scheme that keeps all processors busy.

2.3.5 Topological Partitioning

The requirement of all the parallel techniques discussed so far is that each processor must have access to the entire circuit database. This may pose a severe problem for large circuits because each processor may not have enough memory to hold the entire circuit database. Furthermore, in a message-passing system loading large database into memory takes time. Topological partitioning overcomes this problem by partitioning the circuit into separate sub-circuits and instantiating each on a different processor. Hirose *et al.* [54] used a topological partitioning for parallel logic simulation and generated test patterns for combinational logic circuits. The major disadvantage of their approach is that it has been tailored to one specific-purpose machine and is not suited to the many readily available general-purpose multiprocessors. Smith *et al.* [98] discussed six different partitioning schemes for circuit partitioning. These schemes are:



297412

random partitioning, natural partitioning, partitioning at gate level, partitioning by element strings, and partitioning by fan-in and fan-out cones. All these partitioning schemes are described in [98]. The results presented by Smith *et al.* indicate that for simulation, random partitioning scores best in concurrency but worst in interprocessor communication. Kramer presented a system based on circuit partitioning for the Connection Machine [61]. This system goes to the extreme of instantiating each gate on a single Connection Machine processor or a group of processors. This approach is viable only because the Connection Machine has very fast interprocessor communication and the ATPG algorithm used finds test patterns for all of the faults in the circuit simultaneously. The algorithm is communication bound and the approach is tailored specifically for the Connection Machine. The disadvantage of this approach is that runtime increases exponentially for circuits with more than 15–18 inputs and therefore, unsuitable for most of the practical circuits.

Although applications of parallel and distributed network of machines to the ATPG problem has shown some promising results, but much work remains. Each of the parallelization techniques discussed above has some drawbacks. No technique, except search-space partitioning, has demonstrated the capacity to produce linear speedup for more than 16 processors. Search-space partitioning technique shows the greatest promise for scalability to large number of processors. However, this technique does not answer the problem of large databases created by increasing VLSI circuit sizes and it is applicable only to hard-to-detect faults. Furthermore, it does not address acceleration of the ATPG process for easy-to-detect faults which constitute the majority of the fault list for most practical circuits. Therefore, it becomes necessary to investigate new techniques for solving the compute-intensive ATPG problem, so that they can be easily extended to run on massively parallel and distributed computing platforms. The following section describes some of the recently developed approaches to solve ATPG

problem that can easily exploit fine-grain parallel computing and in general allows effective use of parallel processing.

2.4 Recent Approaches

Optimization methods and Boolean satisfiability are the most recent approaches attempted to find the solution of the ATPG problem. Although these problems are also as hard as the test generation itself, but they have two significant advantages. First, several operations research technique like linear and non-linear programming and graph-theoretic algorithms may be made applicable to the test generation problem. Second, the non-causal form of the model allows the use of parallel processing for the compute-intensive ATPG problem. These approaches formulated the test generation problem as optimization or Boolean satisfiability problem. This problem formulation has three *necessary* and *sufficient* conditions that any set of signal values must satisfy to be a test.

1. The signal in the fault-free and faulty circuits at the fault site must assume complementary values, *e.g.* 1 and 0 respectively for a stuck-at-0 fault.
2. The fault-free and faulty circuits should produce different output values for the same test vector.
3. The set of values must not violate the functionality of any gate of the circuits.

Based on these conditions a test generation network, known as *ATPG constraint network*, is constructed which incorporates the ATPG constraints.

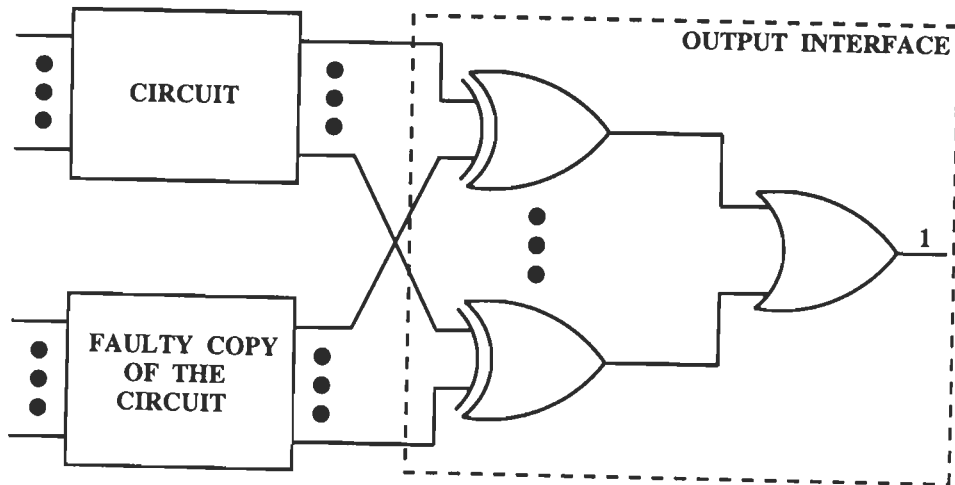


Figure 2.13: ATPG Constraint Network.

2.4.1 ATPG Constraint Network

The ATPG constraint network is constructed by joining the circuit-under-test and its *faulty image* that is a copy of the sub-circuit affected by the fault. The primary output(s) of the fault-free and faulty circuits are connected through an *output interface* to ensure that at least one primary output (PO) of the faulty circuit will be different from the corresponding fault-free PO. The output interface will consist of n two-input exclusive-OR (XOR) gates and one n -input OR gate, where n is the number of POs in the faulty image of the circuit. The inputs to an XOR are the corresponding POs of the circuit-under-test and the faulty image. The output of the OR gate is assigned a fixed value of 1 throughout test generation. For a single-output of the faulty circuit, the output interface will consist only a single XOR and whose output will also be constrained to a value 1. The circuit-under-test, its faulty image, and the output interface constitute the *ATPG constraint network* as shown in Figure 2.13.

2.4.2 Optimization Methods for Test Generation

Chakradhar *et al.* [20] formulated the ATPG problem as an optimization problem by using an unconventional digital circuit modeling technique which constructs a neural network for a given digital logic circuit. This modeling technique relates the input and output signal states of a logic gate through an *energy function*. The energy function is defined over a network of neurons such that the minimum-energy states correspond to the gate's function. Similarly, the function of an entire digital logic circuit can be expressed by a single energy function. This technique models every signal (net), represented by a neuron in the circuit, and the value on the net is its activation value, *i.e.* either a logical 0 or 1. The neurons corresponding to the primary inputs (outputs) of the circuit are called *primary input (output) neurons*. Neurons corresponding to the input (output) signals of a gate are called *input (output) neurons*. Neural networks for 2-input AND, OR, NAND, NOR, XOR and XNOR gates and 1-input NOT gate constitute the *basis set* and gates with more than two inputs are constructed from this basis set.

The neural network for a digital circuit is represented by combining the neural networks for the individual gates of the circuit and characterized by an energy function E that has global minima only at the neural states consistent with the function of all gates in the circuit. This energy function E is uniquely specified by the weights on the links connecting the neurons and the thresholds of the neurons. The energy function of the neural network will be of the form:

$$E(\mathbf{x}) = -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n Q_{ij} x_i x_j - \sum_{i=1}^n c_i x_i + K \quad (2.1)$$

where n is the number of neurons in the neural network, Q_{ij} is the weight associated with the link between neurons i and j such that $Q_{ii} = 0$, x_i is the activation value of neuron i , c_i is the threshold of neuron i and K is a constant.

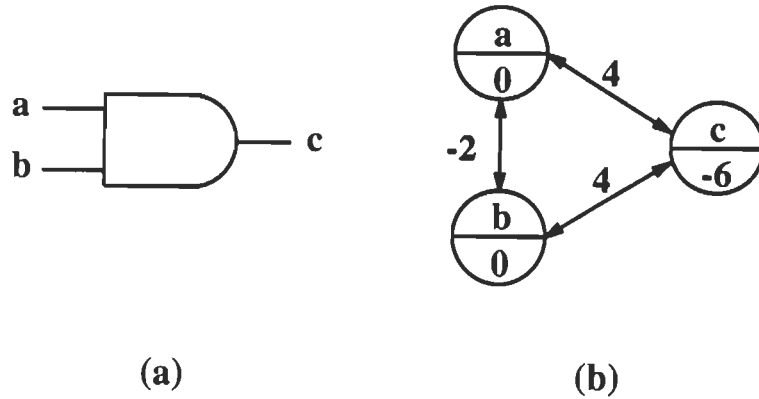


Figure 2.14: (a) 2-input AND gate and (b) its neural network model.

It is important to note that the energy function for a network in the basis set assumes a global minimum value 0 at all consistent network states and greater than 0 for all other network states. Neural networks for 2-input AND, OR, NAND, and NOR gates consist of three neurons that correspond to the input and output nets of the gate. As an illustration, 2-input AND and XOR gates and their corresponding neural networks are shown in Figure 2.14 and 2.15 respectively. It may be noted that the XOR neural network has an additional neuron that does not correspond to any external net associated with the gate. This neuron has an activation value 1 when both inputs to the gate are 1.

In this optimization based approach, first the ATPG neural net is constructed for the test generation network by using the digital circuit modeling technique based on neural network. A fault is injected into the ATPG neural net by assigning fixed activation values to the fault-free and faulty neurons. After fault injection, the output interface incorporates a constraint by assigning a fixed value of 1 to its output in order to ensure that at least one primary output of the faulty circuit will be different from the corresponding fault-free circuit output. Therefore, if the fault is testable, there

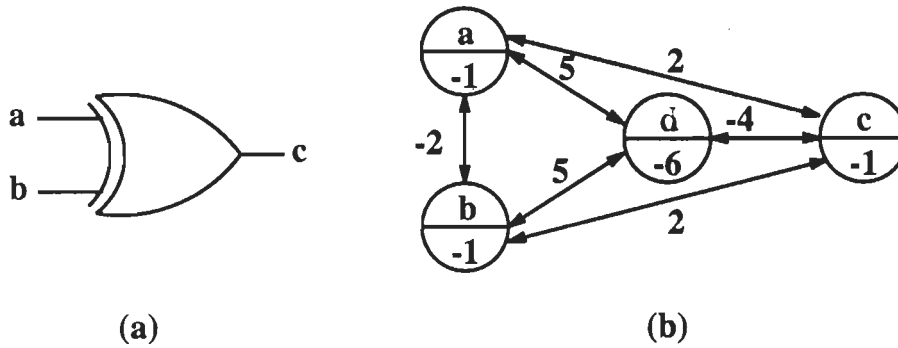


Figure 2.15: (a) 2-input XOR gate and (b) its neural network representation.

exists a *consistent labeling* of the neurons in the ATPG neural net with values from the set $\{0, 1\}$ that does not violate the functionality of any gate. A test pattern can be found for a given fault by finding a minimum energy state of the ATPG neural net by using an appropriate optimization technique. The activation values of the primary input neurons in the consistent labeling of the ATPG net form the required test pattern for the given fault.

Several optimization methods have been applied to find the global optimal for generating the test patterns for some target fault. These methods are either using an actual neural network or based on the use of serial or parallel computers to simulate the neural network. Due to non-availability of large scale implementations of neural networks, only simulation-based approaches have been explored and a continuous optimization technique for test generation is reported [21] which simulates an analog neural network on a commercial neurocomputer. Gradient-descent and probabilistic relaxation methods [20] and graph-theoretic techniques [19], [22] have been attempted for the energy minimization formulation of the test generation.

Iterative Relaxation Methods for Energy Minimization

A fast gradient-descent search algorithm has been used to find the global minimum of the energy function derived for the test generation problem. In this problem formulation, all the global minima of the energy function will be *zero*. If the search terminates at a local minimum for which the final energy of the network is *non-zero*, a probabilistic relaxation technique is used to determine the global minimum. The gradient-descent algorithm is a greedy algorithm and therefore can terminate at a local minimum due to the fact that it only accepts moves which reduce the energy of the neural network. In order to avoid sticking at the local minimum, probabilistic algorithms are devised which also accept some moves that increase the energy of the neural network. Since the final objective is to find the global minimum of the energy, higher-energy moves are only accepted in a probabilistic sense. Thus, higher-energy moves are allowed with a probability of entering into the global minimum area but lower-energy moves are statistically favored. These probabilistic search algorithms are also referred to as simulated annealing methods [10], [76], [84], [105].

The probabilistic methods use *acceptance probability* p_k which is the probability that the state of the neural network with $x_k = 1$ is preferred over the state with $x_k = 0$. The acceptance probability is given by

$$p_k = \frac{1}{1 + e^{-\Delta E_k/T}}. \quad (2.2)$$

where ΔE_k is the energy difference between the two states of the neural network S_1 and S_2 and given by $\Delta E_k = E(S_2) - E(S_1)$. S_1 and S_2 be the state of the neural network in which the k th neuron has activation value 1 and 0 respectively. A random number r between 0 and 1 is generated and if $r \leq p_k$, state S_1 is accepted as the next state. Otherwise, state S_2 is accepted. The probabilistic method minimizes the energy function E by starting at an initial value of the parameter T with some (possibly

random) initial state of the neural network. At the initial value of the parameter T , a sequence of states (constituting a single Markov chain) is generated and the value of T is successively lowered to generate subsequent Markov chains. Eventually as T approaches 0, state transitions become more and more infrequent and finally, the neural network stabilizes in a state which provides the final solution. Consequently, it is an approximation algorithm. The initial value of T , the number of states in the Markov chain at each value of T , and the rate of decrease of T are all important parameters that affect the speed of the algorithm and the quality of the final solution. However, the choice of these parameters for a specific problem is not straightforward and therefore, other methods based on quadratic 0-1 programming technique have been devised for energy minimization.

Graph-Theoretic Methods for Energy Minimization

The aim in the optimization-based test generation approach is to minimize the energy function $E(\mathbf{x})$ given by (2.1). This energy function can be written in the form of a pseudo-Boolean quadratic function [52]: $E(\mathbf{x}) = \mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{c} \mathbf{x}$, where \mathbf{Q} is a symmetric $n \times n$ weight matrix with null diagonal elements, \mathbf{c} is a vector of n real numbers (thresholds of the neurons), \mathbf{x} is a vector of n binary variables, and \mathbf{x}^T is a transpose of \mathbf{x} . Finding the minimum of the pseudo-Boolean quadratic function is known as the *quadratic 0-1 programming* problem. The representation of the energy function in the pseudo-Boolean quadratic function form has a significant advantage that a wide range of techniques available for quadratic 0-1 programming may become applicable for solving the test generation problem.

Although, several algorithms [13], [14], [18], [43], [51], [53], [67], [80], [100], [101], [107] have been developed in the past to find the minimum of the pseudo-Boolean

quadratic function, but the disadvantage of these algorithms is twofold: first, they are not suitable to provide the global minima for the functions with large number of variables. Second, these algorithms have not considered the special structure of the pseudo-Boolean quadratic function derived for the test generation problem. Chakradhar *et al.* [19] developed a quadratic 0-1 programming technique based on graph-theoretic methods to find the global minimum of the energy function derived for the test generation problem. This technique has shown some promising results for solving the test generation problem and therefore, deserves further attention.

2.4.3 Test Generation Using Boolean Satisfiability

The test generation method using Boolean satisfiability developed by Larrabee is neither a purely structural method nor an algebraic one [64], [65], [66]. In this method, a formula expressing the *Boolean difference* between a fault-free circuit and its corresponding faulty sub-circuit is constructed and represented in conjunctive normal form (CNF), also known as product of sums. This formula is then satisfied by applying a SAT algorithm in order to obtain the test patterns for a given set of faults. However, the problem of satisfying a CNF formula (SAT problem) is an *NP*-complete problem [29]. But, the transformation of the test generation problem into a SAT problem has an advantage that SAT algorithms can be applied to generate test patterns.

Interestingly, the class of formulas generated by combinational circuits consists at least two thirds of the clauses that have only two disjuncts (are in 2CNF). This is true because each 2-input unate gate contributes two binary (2CNF) clauses and one ternary clause as shown in Figure 2.16. Unate gates with more than two inputs contribute more than two thirds binary clauses, and fan-out points, buffers, and inverters contribute only binary clauses. In practice, 80% to 90% of the clauses are found in 2CNF. However,

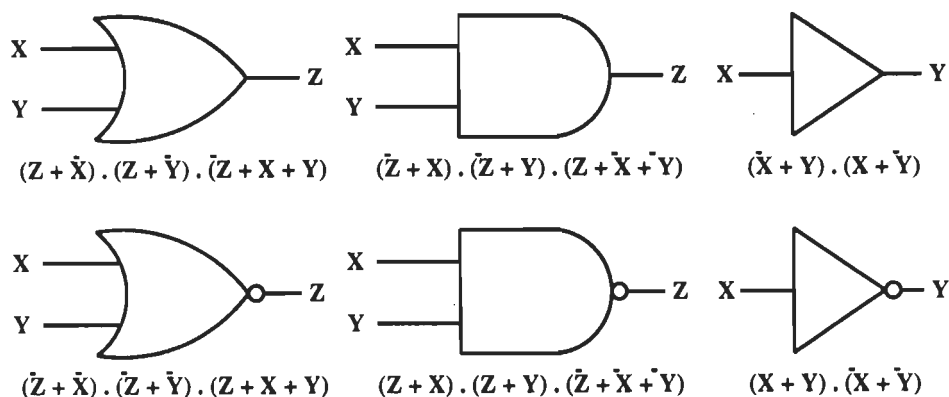


Figure 2.16: CNF formulas for the basic gates.

the problem of satisfying a 2CNF formula, 2SAT, is satisfiable in time linear in the number of clauses plus the number of variables [11]. As there may be an exponential number of 2SAT solutions, the ternary clauses can be used to guide the iteration through the 2SAT assignments.

Larrabee used the linear-time algorithm for satisfying a 2CNF formula [11] to solve the SAT problem by constructing an *implication graph*. In the implication graph, there are two vertices labeled X and \bar{X} for each variable X occurring in the 2CNF clauses and for every 2CNF clause $(X + Y)$ there are two directed edges in the graph: one from \bar{X} to Y and another from \bar{Y} to X . As an example, consider a circuit shown in Figure 2.17(a). Boolean expression in CNF form for the circuit is given below.

$$(\bar{A} + A_1) \cdot (A + \bar{A}_1) \cdot (\bar{A} + A_2) \cdot (A + \bar{A}_2) \cdot (\bar{A}_1 + B) \cdot (\bar{A}_1 + \bar{B}) \cdot (\bar{C} + A_2) \cdot (\bar{C} + B) \cdot (\bar{A}_2 + \bar{B} + C)$$

The implication graph of the 2CNF portion of this formula is shown in Figure 2.17(b). Signal contradictions are determined by finding strongly connected components in this implication graph which has two strongly connected components: $\{\bar{A}_2, \bar{A}, \bar{A}_1, B\}$ and its complement $\{A_2, A, A_1, \bar{B}\}$. The reduced implication graph shown in Figure 2.18 clearly shows that C implies \bar{C} and therefore C must be bound to 0. In the example

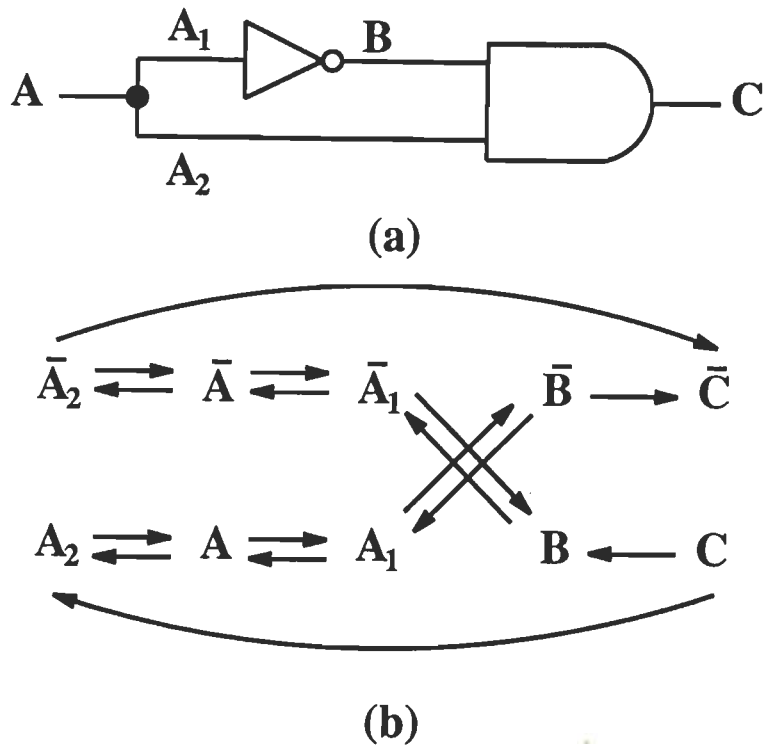


Figure 2.17: (a) An example circuit and (b) its implication graph.

circuit also, C is equal to $A \cdot \bar{A}$ which reassures that C must be bound to 0. Given the binding of C , only one unbound node in the graph remains, and it can assume either Boolean value and remain consistent with the ternary clause.

Larrabee developed an ATPG system based on the Boolean satisfiability method in which test generation is performed in two phases: random and algorithmic. In the first phase of Random Test Generation (RTG), 32 pseudo-random test patterns are simulated by using a PPSFP simulator reported by Waicukauski *et al.* [106]. In this phase, test patterns for the easily tested faults (generally 80% to 99% of total faults) can be generated. Second phase performs algorithmic test generation is performed that is based on the Boolean satisfiability and determines the remaining faults of the circuit.

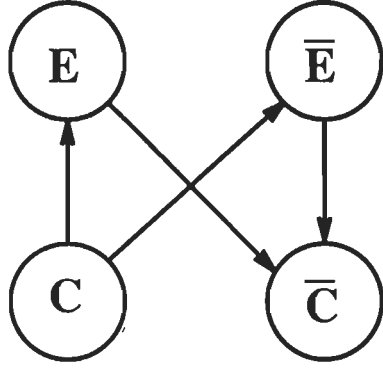


Figure 2.18: Reduced implication graph.

A similar approach has been attempted by Chakradhar *et al.* [22] in which a formula is constructed with a difference that *Boolean false function* is derived instead of the analogous *Boolean truth function*. Consider an example of 2-input OR gate with a, b as inputs and c its output. Since $c = a + b$, it is easy to see that the equation:

$$F_{OR} = c \oplus (a + b) = 0 \quad (2.3)$$

is satisfied only by those values of a, b and c that satisfy the OR gate function. Here, \oplus denotes the logical exclusive-OR (XOR) operation. F_{OR} is referred as the Boolean false function for the OR gate and can be written as

$$F_{OR} = a\bar{c} + b\bar{c} + \bar{a}\bar{b}c \quad (2.4)$$

where $+$ denotes the logical OR operation. A similar formulation is possible for representing the OR gate as an energy function where a, b and c were treated as arithmetic variables. The Boolean false functions of all the primitive gates can be computed in the same way. The false function for a digital circuit is the logical OR of the false functions for all gates in the circuit and consists of a set of binary and ternary relations. The problem of determining a signal assignment that satisfies the false function is equivalent to minimizing the energy function of the digital circuit.

A transitive closure based algorithm [22] has been proposed for test generation by determining signal values that satisfy the Boolean equation derived for the ATPG constraint network which incorporates necessary conditions for fault activation and path sensitization. This algorithm is a sequence of two main steps that are repeatedly executed: transitive closure computation and decision making. Since the implication graph only includes local pairwise (or binary) relations, it is a partial representation of the netlist. Higher-order signal relationships are represented as additional ternary relations. The transitive closure of the implication graph determines global pairwise logical relationships among all signal pairs. These relationships either force values on some signals or indicate contradictory requirements caused by a redundant fault. A test is found if signals thus determined satisfy the Boolean equation. Otherwise, this algorithm enters into the decision-making phase in which an unassigned signal is fixed and the transitive closure is updated to determine logical consequences of this decision.

Transitive closure determines four basic conclusions: *contradiction* – an impossible signal relation, *fixation* – a 0 or 1 value for a signal, *identification* – two signals must assume identical value, and *exclusion* – two signals must not assume certain states. A key feature of the algorithm is that dependencies derived from the transitive closure are used to reduce ternary relations to binary relations that in turn dynamically update the transitive closure. The signals are either determined from the transitive closure or enumerated until the Boolean equation is satisfied.

The ATPG program implementing the transitive closure based test generation algorithm also includes a random test generator and a fault simulator. In the first phase, the system randomly generates test patterns and perform fault simulation. The random test generation is followed by deterministic test generation based on transitive closure computations. Fault simulation is performed after every test vector to eliminate other detected faults. Experimental results appeared in [22] show that most of the faults

are detected using random test generation and only the fraction of the total number of faults are detected by transitive closure algorithm.

2.5 Problem Formulation

The ATPG problem is known to be *NP*-complete problem and conventionally, the generation of test patterns is characterized as a search of N -dimensional 0-1 state space, where N is the number of primary inputs in a combinational circuit, and the search process is guided by heuristics. In spite of considerable research progress, the ATPG algorithms developed so far still require enormous computational resources for generating the test patterns. Moreover, conventional ATPG algorithms are targeted to run on serial (uniprocessor) computers. Although attempts have already been made to solve the ATPG problem on parallel and distributed network of machines, but limited success has been achieved as the parallelization techniques developed so far could not produce linear speedup for parallel machines with large number of processors. Hence, new concepts and cost-effective approaches are, therefore, imperative in order to efficiently solve the compute-intensive ATPG problem as the complexity of the circuits is several tens of thousands of logic gates.

The availability of massively parallel computers presents a new promising paradigm for compute-intensive VLSI-CAD applications like ATPG problem. Efforts have been made towards the investigations of new test generation approaches that can easily be extended to massively parallel computing. A major outcome of these efforts are optimization and Boolean satisfiability based test generation approaches. The significant advantage of these approaches is twofold: first, several operations research techniques and satisfiability (SAT) algorithms are applicable to solve the ATPG problem and second, they are easy to parallelize. Hence, special attention is required to

focus optimization and Boolean satisfiability approaches to solve the ATPG problem. In the optimization approach, the ATPG problem is transformed into an energy minimization problem by deriving the energy function for the circuit-under-test using the unconventional digital circuit modeling technique. So in order to obtain the required test patterns for some target faults, efficient algorithms are to be developed to find the global minima which in turn provide the test patterns. As far as the Boolean satisfiability approach is concerned, Boolean false function expressing the Boolean difference between the fault free and faulty circuit may be derived instead of the Boolean truth function. The advantage of doing so is that the problem of determining a signal assignment can be made by minimizing the Boolean false function to 0 which is equivalent to minimizing the energy function of the optimization.

In energy minimization formulation of the ATPG problem, the energy function obtained is a pseudo-Boolean quadratic function, *i.e.* of the form: $E(\mathbf{x}) = \mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{c} \mathbf{x}$, where \mathbf{Q} is a symmetric $n \times n$ weight matrix with null diagonal elements, \mathbf{c} is a vector of real numbers (thresholds of the neurons), \mathbf{x} is a vector of n variables, and \mathbf{x}^T is a transpose of \mathbf{x} . Quadratic 0-1 programming technique based on graph-theoretic methods has been developed to find the global minimum of the energy function derived for the ATPG problem. However, new algorithms may be investigated in order to exploit the special structure of the pseudo-Boolean quadratic function obtained from the test generation problem. Some traditional search and optimization methods like gradient-descent and probabilistic methods have also been attempted for the energy minimization. But, these methods have their limitations and are not robust. Genetic Algorithms (GAs), on the other hand, are robust, innovative and highly reliable global optimization and search algorithms and, surprisingly, not applied to this kind of optimization problem.

Keeping all the above points into consideration, this thesis focuses to investi-

gate new quadratic 0-1 programming technique and Genetic Algorithms based methods for solving the ATPG problem that is formulated as an optimization or a Boolean satisfiability problem. The development of a CAD tool based on these algorithms for generating test patterns automatically is also a part of the thesis.

Chapter 3

A New Quadratic 0-1

Programming for Test Generation

This chapter mainly focuses on the energy minimization formulation of the test generation problem which has the similarities to the minimization of the Boolean false function in the satisfiability based approach. Optimization algorithms like directed search technique augmented by probabilistic relaxation and simulated annealing [10], [20], [59], [76], [84], [105] have been initially considered for the energy minimization. The disadvantage of these algorithms is twofold: first, it becomes too time-consuming due to their complex annealing schedule [9] and second, the mapping of the domain-specific (*i.e.* test generation) problem into the simulated annealing paradigm is not straightforward. Therefore, there is a need to develop efficient algorithms for the energy minimization problem.

Since the energy function for the test generation problem is of the form of a pseudo-Boolean quadratic function $E : \{0, 1\}^n \mapsto \mathcal{R}$, a whole suite of techniques available for the quadratic 0-1 programming problem can be applied to find a solution.

The pseudo-Boolean quadratic function $E(\mathbf{x})$ is defined as

$$E(\mathbf{x}) = \mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{c}^T \mathbf{x}, \quad (3.1)$$

where

$$\mathbf{x} = (x_1, x_2, \dots, x_n)^T;$$

$$\mathbf{Q} = [Q_{ij}] \text{ is a real symmetric matrix of order } n, \forall i, j \in \{1, 2, \dots, n\};$$

$$\mathbf{c} = (c_1, c_2, \dots, c_n)^T; \text{ and}$$

$$x_i = 0 \text{ or } 1, \forall i = 1, 2, \dots, n.$$

The quadratic 0-1 programming problem is to find the minimum of the quadratic function $E(\mathbf{x})$. The problem of finding the minimum of $E(\mathbf{x})$ is an *NP*-complete problem [38], for which various algorithms have been proposed in the past and reported in the review chapter. Out of these algorithms very few have been considered for the VLSI-CAD applications. A linearization mechanism proposed by Glover and Woosley [43] transforms the quadratic function $E(\mathbf{x})$ and the constraints into a mixed linear program by means of the introduction of a set of linearization variables, $y_{ij} = x_i x_j$ and the following set of constraints on the y -variables:

$$x_i + x_j - y_{ij} \leq 1,$$

$$x_i - y_{ij} \geq 0,$$

$$\text{and } x_j - y_{ij} \geq 0.$$

The advantage of using linearization is that the algorithms developed for linear programming problems can be effectively applied. Other algorithms for finding the minimum of pseudo-Boolean quadratic function $E(\mathbf{x})$ are either based on the branch-and-bound method [18] or using the graph-theoretical approach [13]. Using these traditional techniques, there is little hope to find exact solutions to even modest-sized problems,

i.e. functions with less than 150 variables. As far as the test generation problem is concerned, the number of variables in the energy function which is linear in terms of the number of signals in the combinational logic circuits, is very high. Therefore, it is quite unlikely that these techniques can be used to find exact solutions to the test generation problem for the circuits having more than about a hundred signals. In addition, the special structure of the pseudo-Boolean quadratic function arising from the test generation problem could not be much exploited. Chakradhar *et al.* [19] initially proposed a technique known as quadratic 0-1 programming technique for the test generation problem for finding exact minima of the energy function. This technique is detailed below.

3.1 Quadratic 0-1 Programming Technique

In this technique, the energy function of the form of the pseudo-Boolean quadratic function is first written in a *positive pseudo-Boolean form* called *posiform*. A posiform is any sum of *monomials* with positive coefficients and a monomial is a product aT where T is a term (*i.e.* a finite product of distinct literals) and a is a real number called the *coefficient* of the monomial. A posiform without a constant term is *homogeneous* and a posiform with a constant term is *inhomogeneous*. The energy function \mathbf{E} in posiform can be divided into two sub-functions – a homogeneous posiform and an inhomogeneous posiform. This technique could exploit the following result to efficiently minimize the energy function:

A minimizing point of a quadratic homogeneous posiform, known to have a minimum value 0, can be obtained in linear time complexity in the number of variables and terms in the function [16].

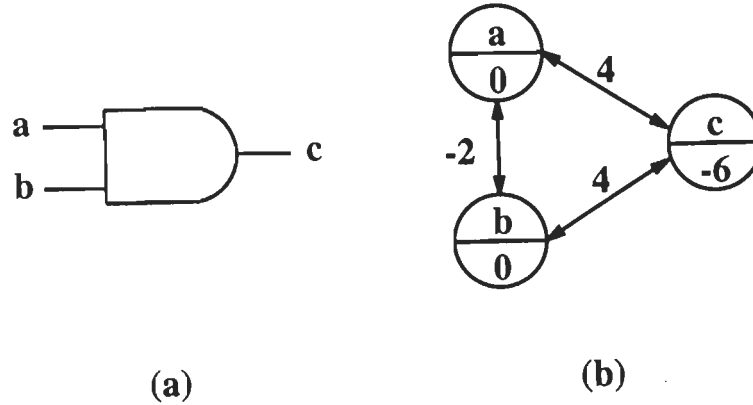


Figure 3.1: (a) An AND gate and (b) its neural network model.

To explain this technique, let us consider the AND gate and its neural network model as shown in Figure 3.1. Its energy function is derived from the test generation problem formulation described in the previous chapter and given by:

$$E_{AND}(a, b, c) = [2c(1 - a)] + [2c(1 - b)] + [2(c - ac - bc + ab)] \quad (3.2)$$

It may be observed that all the three terms in this function are non-negative and hence, $E_{AND} = 0$ requires each term should simultaneously become 0. The first two terms can be written in posiform as

$$E_{AND}(a, b, c) = [2c\bar{a}] + [2c\bar{b}] + [2(c - ac - bc + ab)] \quad (3.3)$$

The sub-function comprising the first two terms is a homogeneous posiform with a minimum value of 0. The third term can be rewritten only as an inhomogeneous posiform: $2\bar{a}c + 2\bar{b}c + 2ab + 2\bar{c} - 2$. Similar models and energy functions can be derived for other logic gates [20]. Hence, the energy function for the entire circuit, E_{CKT} , can be expressed as two sub-functions, a homogeneous posiform E_H and an inhomogeneous posiform E_I , each having a minimum value of 0. Furthermore, all circuit signals appear in the homogeneous posiform which suggests the following method to minimize the energy function:

1. Find a new minimizing point α of E_H . The time complexity of this step is linear in the number of variables and terms in E_H [16].
2. Check if α is also a minimizing point of E_I . If yes, then α is a minimizing point of E_{CKT} and a test vector is obtained. Otherwise go to Step 1.

Although the minimizing point α of E_H can be obtained in linear time, but the homogeneous posiform may have exponentially many minimizing points and therefore, a branch-and-bound technique is needed to systematically enumerate all minimizing points of E_H to find a minimizing point of E_{CKT} . A linear-time algorithm by Aspvall *et al.* [11] was used to find all the minimizing point of the homogeneous posiform [19]. This algorithm constructs a *directed implication graph* and finds its strongly-connected components to construct the *condensation graph*. Larrabee also used a similar approach but formulated the test generation problem as a Boolean satisfiability problem [66].

Let us consider the test generation problem for the stuck-at-0 fault at the primary output (PO) d of the circuit shown in Figure 3.2(a). Since the fault is at PO, there is no need to create a faulty circuit copy. The energy functions for the individual gates can be derived following the technique of Chakradhar *et al.* [20] and are given as below:

$$E_{NOT}(b, a) = [2ab] + [2\bar{a}\bar{b}] \quad (3.4)$$

$$E_{AND}(a, b, c) = [2c\bar{a}] + [2c\bar{b}] + [2(c - ac - bc + ab)] \quad (3.5)$$

$$E_{OR}(a, c, d) = [2a\bar{d}] + [2c\bar{d}] + [2(d - cd - ad + ac)] \quad (3.6)$$

The energy function for the circuit is:

$$E_{CKT} = E_{NOT}(b, a) + E_{AND}(a, b, c) + E_{OR}(a, c, d) \quad (3.7)$$

The homogeneous posiform is

$$E_H = [2ab] + [2\bar{a}\bar{b}] + [2c\bar{a}] + [2c\bar{b}] + [2a\bar{d}] + [2c\bar{d}] \quad (3.8)$$

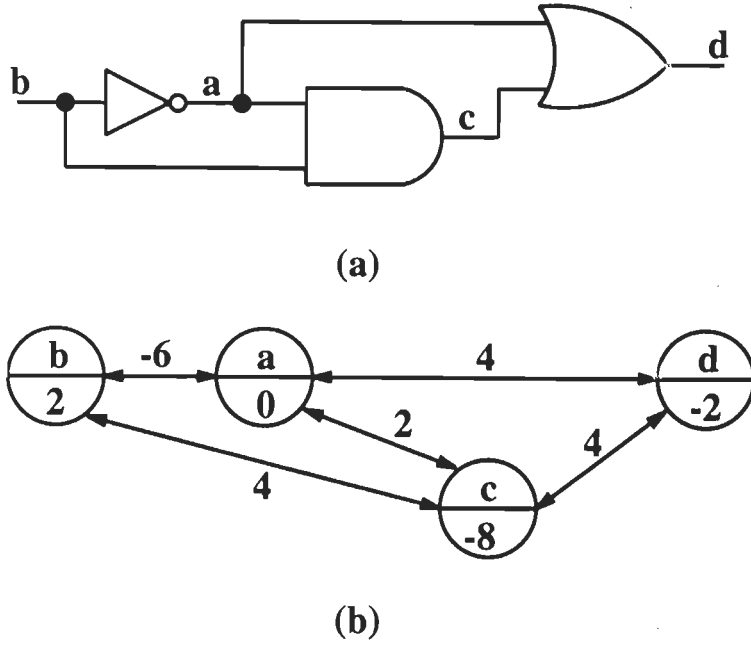


Figure 3.2: (a) An example circuit (b) Neural network representation of the example circuit for d stuck-at-0 fault.

and the inhomogeneous posiform E_I consists the remaining terms of E_{CKT} which comes out to be

$$E_I = [2(c - ac - bc + ab)] + [2(d - cd - ad + ac)] \quad (3.9)$$

In order to sensitize the stuck-at-0 fault at d , the value at d should be 1 and therefore, E_H can be reduced to:

$$E_H = [2ab] + [2a\bar{b}] + [2c\bar{a}] + [2c\bar{b}] \quad (3.10)$$

Chakradhar *et al.* [19] construct an implication graph for the homogeneous posiform, E_H . The implication graph corresponding to (3.10) and its condensed form is shown in Figure 3.3.

Now, a minimizing point of the E_H is found by using a linear-time algorithm of Aspvall *et al.* [11]. Let the components c and its dual \bar{c} be marked first. By setting

$c = 0$, \bar{c} implies the strongly connected component $\{\bar{a}, b\}$. This strongly connected component of the implication graph should be assigned to 1 as it can not be assigned to 0 because of the illegal assignment. Therefore, $\{\bar{a}, b\}$ should be assigned to 1. Now mark the component $\{a, \bar{b}\}$ and its dual $\{\bar{a}, b\}$ and set $a = 0$, $\bar{b} = 0$ and $\bar{a} = 1$ and $b = 1$. This leads to a minimizing point of E_H for the assignments $a = c = 0$ and $b = 1$. These values of a , b and c evaluates $E_H = 0$, but E_I does not evaluate to 0 and hence, a new minimizing point of E_H is required.

Using a branch-and-bound search method on the variables c and a , all the minimizing points of E_H can be systematically generated. So, if $c = 1$, then literals in components $\{a, \bar{b}\}$ and $\{\bar{a}, b\}$ should be assigned 1 which causes a conflict because a variable and its complement are being assigned the same value. Thus, $c = 1$ does not lead to the minimizing point of E_H . Now, backtrack and set $c = 0$ which does not imply any other variable. Assign $a = 0$ and set all literals in the component containing \bar{a} to 1. This results in $b = 1$ and the minimizing point of E_H is $a = c = 0$, $b = 1$. Unfortunately, this minimizing point of E_H also is not a minimizing point of E_I . Again backtrack and re-assign $a = 1$ which results in $b = 0$ and the minimizing point $a = 1$, $b = c = 0$ which is also a minimizing point of E_I . Hence, $b = 0$ is the test pattern for the fault d stuck-at-0.

This simple example shows that in the worst case, one may have to examine all 2^n combinations of n variables to systematically enumerate all minimizing points of E_H in order to find a minimizing point of E_{CKT} . Several ideas like *transitive closure* and incorporation of additional constraints using problem-specific knowledge are suggested to accelerate the minimization process [22]. These ideas can be easily incorporated in other graph-theoretic methods for test generation [66]. The advantage of the transitive closure method is that it identifies the variables that must assume constant values at all minimizing points of E_H . The additional constraints may be derived from the path

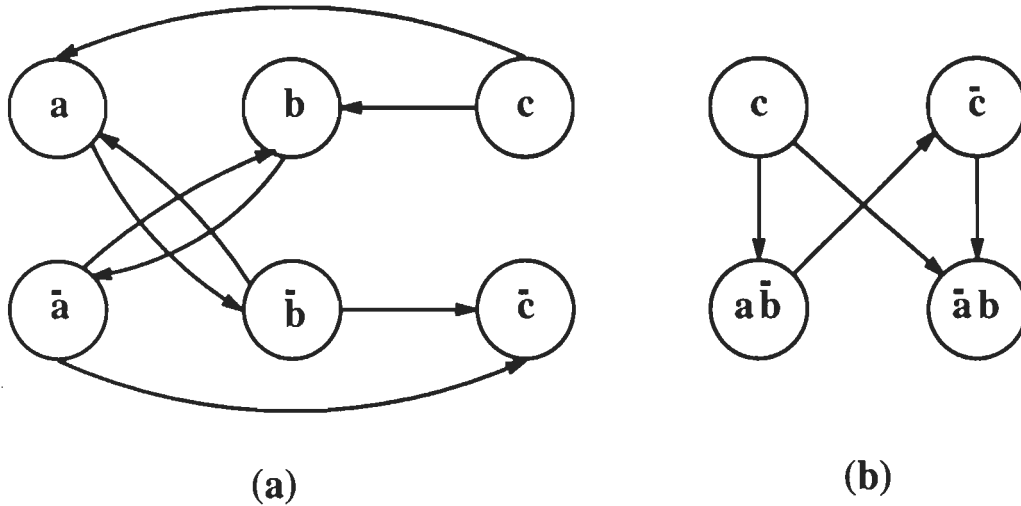


Figure 3.3: (a) Implication graph for the example circuit and (b) its condensed form.

sensitization information and can be included into the energy function. Keeping all these points into consideration, a new quadratic 0-1 programming technique [96] given below has been proposed.

3.2 A New Quadratic 0-1 Programming Technique

In the test generation method using quadratic 0-1 programming technique [97], the aim is to minimize the energy function which is of the form of the pseudo-Boolean quadratic function $E(\mathbf{x}) = \mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{c}^T \mathbf{x}$ as defined in (3.1). In order to find the minimum of $E(\mathbf{x})$, first of all the partial derivatives of $E(\mathbf{x})$ with respect to each of the x_i 's are calculated. There is no loss of generality if it is assumed that $E(\mathbf{x})$ is differentiable on $[0, 1]$, i.e. $\frac{\partial E}{\partial x_i}$ exist $\forall x_i \in [0, 1]$ with the following remarks:

- The objective function E consists of a finite number of terms r .

- $\frac{\partial E}{\partial x_i}$ will be linear functions of x_i 's as E is at the most quadratic function in terms of x_i 's.

Therefore, all $\frac{\partial E}{\partial x_i}$ must also consists of finite number of terms and consequently $\frac{\partial E}{\partial x_i}$ must lie in a finite interval. Also,

$$\frac{\partial E}{\partial x_i} = g(x_1, x_2, \dots, x_n), \forall i = 1, 2, \dots, n. \quad (3.11)$$

Here g is linear in x_i . Hence, it is possible to calculate the minimum and maximum of all possible values of $\frac{\partial E}{\partial x_i}$, $\forall i = 1, 2, \dots, n$.

Let $minp$ and $maxp$ denote the minimum and maximum value of $\frac{\partial E}{\partial x_i}$ respectively. These values are calculated over the entire domain and taking into consideration all the constraints. In the calculation of $minp$ and $maxp$, the following steps are to be considered:

Step 1. If for any x_i , $maxp$ is found to be ≤ 0 , $minp$ will always be ≤ 0 . It means that over the entire domain and considering all the constraints, $\frac{\partial E}{\partial x_i}$ takes values which are ≤ 0 irrespective of the values taken by other variables. Therefore, one can say that the function $E(x)$ is monotonically decreasing in terms of the variable x_i as shown in Figure 3.4(a). Thus, it is concluded that

$$[E(\mathbf{x})]_{x_i=1} \leq [E(\mathbf{x})]_{x_i}, \forall x_i \in [0, 1] \quad (3.12)$$

irrespective of the values taken by other variables. Hence, it is advisable to pivot x_i at value 1 in order to yield a better value as the main objective is to minimize E .

Step 2. If for any x_i , $minp$ is found to be ≥ 0 , $maxp$ will always be ≥ 0 . It means that over the entire domain and taking all the constraints into consideration, $\frac{\partial E}{\partial x_i}$ takes values which are ≥ 0 irrespective of the values taken by other variables.

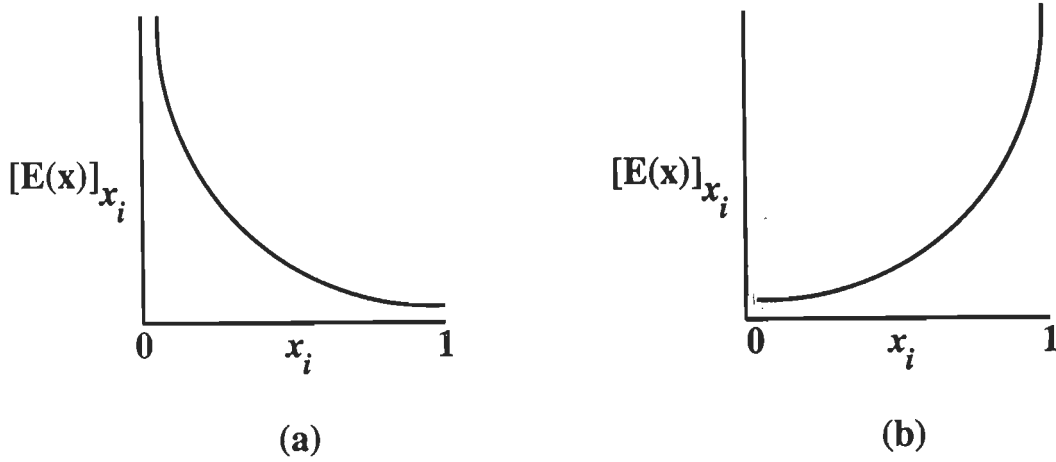


Figure 3.4: $[E(\mathbf{x})]_{x_i}$ as function of x_i

Therefore, one can say that the function $E(x)$ is monotonically increasing in terms of the variable x_i as shown in Figure 3.4(b). Thus, it is concluded that

$$[E(\mathbf{x})]_{x_i=0} \leq [E(\mathbf{x})]_{x_i}, \forall x_i \in [0, 1] \quad (3.13)$$

irrespective of the values taken by other variables. Hence, it is advisable to pivot x_i at value 0 in order to yield a better value in terms of the minimization of E .

Step 3. In steps 1 and 2, some of the x_i variables are pivoted at 0 or 1. Because of this fixation, the *minp* and the *maxp* values may get altered. So, evaluate them accordingly and repeat steps 1 and 2 until no further variable gets fixed assignment or a condition called *Stability Race Condition* given below is encountered.

3.2.1 Stability Race Condition

Consider a 2-input AND gate with x_1, x_2 as the inputs and x_3 as the output. By the aid of digital circuit modeling technique using neural networks, the energy function of

the AND gate can be derived as

$$E(\mathbf{x}) = 2x_1x_2 - 4x_1x_3 - 4x_2x_3 + 6x_3 \quad (3.14)$$

Assuming that this AND gate is a part of a circuit and x_1 is fixed at a value 1 during the process of the function minimization, the energy function reduces to:

$$E(\mathbf{x}) = 2x_2 - 4x_2x_3 + 2x_3 \quad (3.15)$$

As the aim is to find out the values of x_2 and x_3 for which the minimum value of the energy function is 0, the partial derivatives of $E(\mathbf{x})$ with respect to x_2 and x_3 are calculated and given below:

$$\frac{\partial E}{\partial x_2} = -4x_3 + 2 \quad (3.16)$$

$$\frac{\partial E}{\partial x_3} = -4x_2 + 2 \quad (3.17)$$

and their *maxp* and *minp* are computed as:

$$\text{maxp} \left(\frac{\partial E}{\partial x_2} \right) = 2, \text{minp} \left(\frac{\partial E}{\partial x_2} \right) = -2$$

$$\text{maxp} \left(\frac{\partial E}{\partial x_3} \right) = 2, \text{minp} \left(\frac{\partial E}{\partial x_3} \right) = -2$$

So, no decision can be taken to pivot either variable. Let us take the decision to pivot x_3 to a value 1. Then it is found that

$$x_3 = 1 \Rightarrow \frac{\partial E}{\partial x_2} = -2 \Rightarrow \underbrace{x_2 = 1}_{\text{(pivoted)}} \Rightarrow \frac{\partial E}{\partial x_3} = -2 \Rightarrow \underbrace{x_3 = 1}_{\text{(pivoted)}}.$$

It means that

$$x_3 = 1 \Leftrightarrow x_2 = 1.$$

Such a system is said to be *stable*. Let us now fix x_3 to a value 0, then

$$x_3 = 0 \Rightarrow \frac{\partial E}{\partial x_2} = 2 \Rightarrow \underbrace{x_2 = 0}_{\text{(pivoted)}} \Rightarrow \frac{\partial E}{\partial x_3} = 2 \Rightarrow \underbrace{x_3 = 0}_{\text{(pivoted)}}$$

i.e.

$$x_3 = 0 \Leftrightarrow x_2 = 0.$$

This concludes that the system becomes stable with $x_2 = x_3 = 1$ or $x_2 = x_3 = 0$ and confirms the validity of the concept as the variable values provide the consistent labeling of the signal values of the AND gate. Hence, more than one valid solution may exist for such problems.

3.2.2 Criteria for Fixation of Variables

Various criteria for fixation of variables based on the theoretical concepts given above have been discussed here. Although due to the functionality of the gates, most of the variables in the energy function $E(\mathbf{x})$ are inter-related but it is assumed that such relationships among the variables is taken care by the neural network itself, and therefore, only the test generation constraints to partition the variable vector \mathbf{x} are considered. These constraints are basically derived from the test generation concepts: fault excitation and fault-effect propagation. The variable vector \mathbf{x} is partitioned into *free variables* and *related or bound variables*. The free variables are the variables which do not have any constraints. On the other hand, the bound variables are inter-related by means of $x_i + x_j = 1$ and x_i, x_j are not connected with any other variable, *i.e.* only the test generation constraints are considered. The *maxp*, *minp* and *Curr_Ones* for each free and related variables are calculated, where *maxp* and *minp* are as defined previously and *Curr_Ones* is the value taken by $\frac{\partial E}{\partial x_i}$, assuming all the unpivoted vector variables of value 1. The fixation criteria for free and related variables are given below.

Fixation Criteria for Free Variables

The free variables are the variables not related with any other variable due to the test generation constraints. Let x_i be a free variable in the function $E(\mathbf{x})$. The criteria for fixation of the free variables are as follows:

S-I $\max p \geq 0$ and $\min p \geq 0$.

It means that the value of the partial derivative of $E(\mathbf{x})$ with respect to x_i will always be ≥ 0 because it lies in the range of 0 to some positive value. It implies that the function is monotonically increasing with respect to x_i . Therefore, it is better to fix that particular x_i at 0 as the goal is to minimize the function $E(\mathbf{x})$.

S-II $\max p \leq 0$ and $\min p \leq 0$.

This means that the value of the partial derivative of $E(\mathbf{x})$ with respect to x_i will always be ≤ 0 because it lies in the range 0 to some negative value. It implies that the function is monotonically decreasing with respect to x_i . As the problem is of minimization, this implies that it is better to fix that particular x_i at 1.

The above two are the very strong criteria, *i.e.* the decision taken is absolutely deterministic. Unfortunately, sometimes one might not be able to decide about the fixation on the basis of the observations S-I and S-II. For example, consider the case when $\max p(\frac{\partial E}{\partial x_1}) = 2$, $\min p(\frac{\partial E}{\partial x_1}) = -2$, and $\max p(\frac{\partial E}{\partial x_2}) = 2$, $\min p(\frac{\partial E}{\partial x_2}) = -2$. In this case, nothing can be said about the actual values of the partial derivatives and hence no variable can be fixed. So, in order to handle such situations, the following two more criteria are added which can be applied only in the case of a particular variable

which does not satisfy the criteria S-I and S-II.

S-III $Curr_Ones \leq minp$.

Here, the value of $Curr_Ones$ for a particular x_i is less than or equal to the minimum possible value taken by a partial derivative of E with respect to x_i . Thus, it is better to fix x_i at value 1 in order to yield a minimum value of the energy function. This is a locally best decision. The criteria is somewhat weaker than S-I and S-II. But still it is purely deterministic.

S-IV $Curr_Ones \geq maxp$.

Here, the value of $Curr_Ones$ for a particular x_i is greater than or equal to the maximum possible value taken by a partial derivative of E with respect to x_i . Thus, it is better to fix x_i at value 0 in order to yield a minimum value of the energy function. This is a locally best decision. This criteria is also somewhat weaker than S-I and S-II, but the approach is still purely deterministic.

Fixation Criteria for Related Variables

The related variables are the bound variables due to the test generation constraints. Assume x_i and x_j to be the related variables, *i.e.* the variables x_i and x_j are related by the constraints such that $x_i + x_j = 1$. Here, the decision taken for x_i is to be influenced by the decision taken for x_j which is dependent on partial derivative value of x_i as well as x_j . The criteria for fixing the related variables are as follows:

R-I Unique minimum for x_i .

In the $Curr_Ones$ vector of all unpivoted related variables, suppose there exists a unique minimum, say for x_i . Then, if the constraint is $x_i + x_j = 1$,

this implies that it is better to fix x_i at 1 and x_j at 0 in order to yield a minimum value of energy function E . The decision is taken because if the criteria is satisfied by x_i and x_j , then it indicates that the value taken by partial derivative of E with respect to x_i is less than the value taken by partial derivative of E with respect to x_j . This in turn implies that growth rate of the energy function E with respect to x_i is less than growth rate of the energy function E with respect to x_j . Hence, it is advisable to fix x_i at 1 and x_j at 0. This is a locally best decision. If the unique minimum does not exist, then this criterion fails and no decision can be made on the value of x_i . In that case, the following criterion may be considered:

R-II Unique maximum for x_i .

In the *Curr_Ones* vector of all unpivoted related variables, let there exist a unique maximum, say for x_i . Then, if the constraint is $x_i + x_j = 1$, this implies that it is better to fix x_i at 0 and x_j at 1 in order to yield a minimum value of energy function E . The decision is taken because if the criteria is satisfied by x_i and x_j , then it indicates that the value taken by partial derivative of E with respect to x_i is greater than the value taken by partial derivative of E with respect to x_j . This in turn implies that growth rate of energy function E with respect to x_i is greater than growth rate of the energy function E with respect to x_j . Hence, it is advisable to fix x_i at 0 and x_j at 1. This is a locally best decision. However, if the unique maximum also does not exist, then this criterion fails.

When both the above criteria fail, any one variable is selected for which the maximum occurs among the *Curr_Ones* vector of unpivoted related variables. Let it be x_i and its corresponding constraint variable be x_j .

Then the following two more criteria for fixation of the related variables are added:

R-III When $Curr_Ones$ of $x_i \neq Curr_Ones$ of x_j .

This implies that $Curr_Ones$ of x_i is greater than $Curr_Ones$ of x_j . Hence, growth rate of the energy function with respect to x_i is greater than that with respect to x_j . Hence, it is advisable to fix x_i at 0 and x_j at 1 in order to yield a minimum value of the energy function. Now, suppose a situation is encountered where $Curr_Ones$ of x_i is equal to $Curr_Ones$ of x_j , then this criterion also fails. In that case the following final criterion may be considered:

R-IV When $Curr_Ones$ of $x_i = Curr_Ones$ of x_j .

In this case, one simply cannot say whether to assign x_i to 0 and x_j to 1 or vice versa. So, a decision can be made randomly *i.e.* say fix x_i at 0 and x_j at 1 or vice versa. However, this is the weakest criterion.

So, using the criteria R-I to R-IV, the solution space is continuously explored till all the variables are fixed. Finally the value of the energy function $E(\mathbf{x})$, which is termed as *energy*, is computed. The value of $E(\mathbf{x})$ should be 0 value at its minimum and if the energy obtained is greater than 0, it means that a wrong decision has been taken somewhere in the fixation of the related variables. The wrong decision could be possible only due to the weak criteria chosen. So if such a situation arises, then backtracking has to be made *i.e.* the latest decision taken on the basis of the weak criterion has to be reversed and the solution space needs to be further explored. This process continues till a zero value for the function $E(\mathbf{x})$ is obtained or till a point where all decisions made are exhausted by backtracking. That will

be the worst case for the algorithm. In case a non-zero value is obtained as the minimum *energy* value, the fault under consideration is declared redundant.

3.2.3 Backtracking Procedure

The proposed technique uses the popular backtracking procedure only in the case where an inconsistency is found during the fixation of the related variables. For backtracking, one needs a backtracking stack for storing the information of various parameters: *maxp*, *minp*, *Curr_Ones*, *Count* and *Decision* where *Count* gives the number of variables that are fixed before the last decision is made and *Decision* gives the value of the variable fixed at that particular moment. These entries in the stack will be made during the fixation of related variables based on the weakest criterion R-IV. So, whenever one decides to backtrack, the system is restarted with the help of these entries as follows:

- First, all the variables that were fixed after count are declared as unpivoted.
- The decision is popped out and reversed.
- System is loaded with stored *maxp*, *minp* and *Curr_Ones*.

Once a backtracking is made, the normal procedure is continued for fixation of the free and the related variables as explained above, *i.e.* S-I to S-IV and R-I to R-IV.

3.3 Examples

This section illustrates the proposed technique by generating test patterns for stuck-at faults in the combinational circuits. Consider the test generation problem for a stuck-at-0 fault on signal line *d* in a simple circuit shown in Figure 3.2(a). In this case,

there is no need to create a faulty copy of the circuit because the fault is at primary output (PO). The neural network representation for the example circuit is given in Figure 3.2(b).

Since the energy function for a circuit is derived by using its neural network representation [20], the energy E for the example circuit is obtained by adding Equations (3.4), (3.5) and (3.6) and substituting all the complement variables using the following expression:

$$\bar{x}_i = 1 - x_i$$

where \bar{x}_i is the complement of x_i . Finally, the energy E comes out to be:

$$E = 6ab - 4bc - 2ac - 4ad - 4cd - 2b + 8c + 2d + 2 \quad (3.18)$$

Since d is stuck-at 0, the signal d has to be assigned a fixed value of 1 in order to sensitize the fault. By substituting $d = 1$ in (3.18), E gets reduced to:

$$E = 6ab - 4bc - 2ac - 4a - 2b + 4c + 4 \quad (3.19)$$

In this technique, one needs to calculate the partial derivatives of the energy function E with respect to the remaining variables in (3.19) and their corresponding maximum, minimum and the value assuming all the unpivoted vector variables have the value 1, *i.e.* $maxp$, $minp$ and $Curr_Ones$ respectively. The partial derivatives of the energy function E as obtained from (3.19) are:

$$\frac{\partial E}{\partial a} = 6b - 2c - 4 \quad (3.20)$$

$$\frac{\partial E}{\partial b} = 6a - 4c - 2 \quad (3.21)$$

$$\frac{\partial E}{\partial c} = -4b - 2a + 4 \quad (3.22)$$

and their corresponding $maxp$, $minp$ and $Curr_Ones$ are computed as:

$$maxp\left(\frac{\partial E}{\partial a}\right) = 2, \quad minp\left(\frac{\partial E}{\partial a}\right) = -6, \quad \text{and} \quad Curr_Ones\left(\frac{\partial E}{\partial a}\right) = 0;$$

$$\begin{aligned} \max_p \left(\frac{\partial E}{\partial b} \right) &= 4, \quad \min_p \left(\frac{\partial E}{\partial b} \right) = -6, \quad \text{and } Curr_Ones \left(\frac{\partial E}{\partial b} \right) = 0; \\ \max_p \left(\frac{\partial E}{\partial c} \right) &= 4, \quad \min_p \left(\frac{\partial E}{\partial c} \right) = -2, \quad \text{and } Curr_Ones \left(\frac{\partial E}{\partial c} \right) = -2 \end{aligned}$$

In this example, all the variables are free variables as the fault site is a primary output which does not affect other part of the circuit. So, the criteria for fixation described in the subsection 2.2.1 gets applied. Unfortunately, it is not possible to decide the fixation on the basis of the observations S-I and S-II which holds very strong criteria. Although, the criterion S-III is somewhat weaker than S-I and S-II, it is applicable to fix the variable c at a value 1 since $Curr_Ones$ is equal to \min_p . By substituting $c = 1$ in (3.20) and (3.21), the partial derivatives of the energy function reduce to:

$$\frac{\partial E}{\partial a} = 6b - 6 \quad (3.23)$$

$$\frac{\partial E}{\partial b} = 6a - 6 \quad (3.24)$$

and their corresponding \max_p , \min_p and $Curr_Ones$ become:

$$\max_p \left(\frac{\partial E}{\partial a} \right) = 0, \quad \min_p \left(\frac{\partial E}{\partial a} \right) = -6, \quad \text{and } Curr_Ones \left(\frac{\partial E}{\partial a} \right) = 0,$$

$$\max_p \left(\frac{\partial E}{\partial b} \right) = 0, \quad \min_p \left(\frac{\partial E}{\partial b} \right) = -6, \quad \text{and } Curr_Ones \left(\frac{\partial E}{\partial b} \right) = 0,$$

Now, the criterion S-II becomes applicable to fix both a and b variables at value 1. But, by substituting $a = 1$, $b = 1$ and $c = 1$ in (3.18), E becomes nonzero which means that solution is not found. It means that the decision of fixing the variable c at a value 1 could not lead to globally best solution. So, backtracking is made by reversing the decision of fixing the variable c . Now fix the c variable at a value 0. The partial derivatives of the energy function comes out to be:

$$\frac{\partial E}{\partial a} = 6b - 4 \quad (3.25)$$

$$\frac{\partial E}{\partial b} = 6a - 2 \quad (3.26)$$

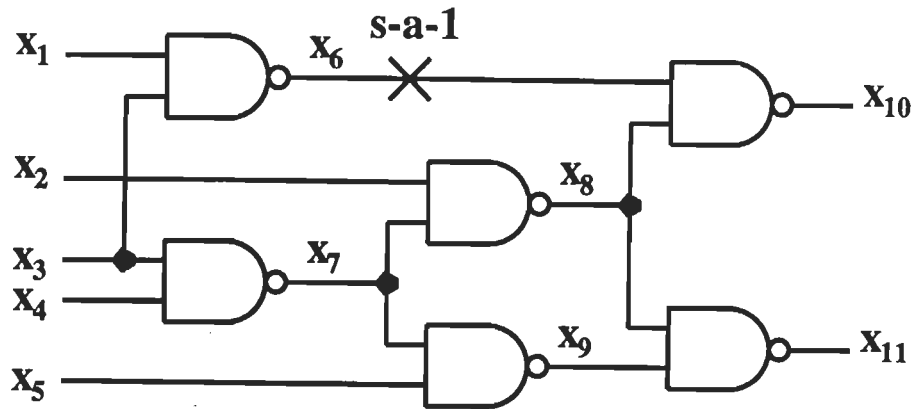
and their corresponding $maxp$, $minp$ and $Curr_Ones$ becomes:

$$maxp \left(\frac{\partial E}{\partial a} \right) = 2, \quad minp \left(\frac{\partial E}{\partial a} \right) = -4, \quad \text{and} \quad Curr_Ones \left(\frac{\partial E}{\partial a} \right) = 2,$$

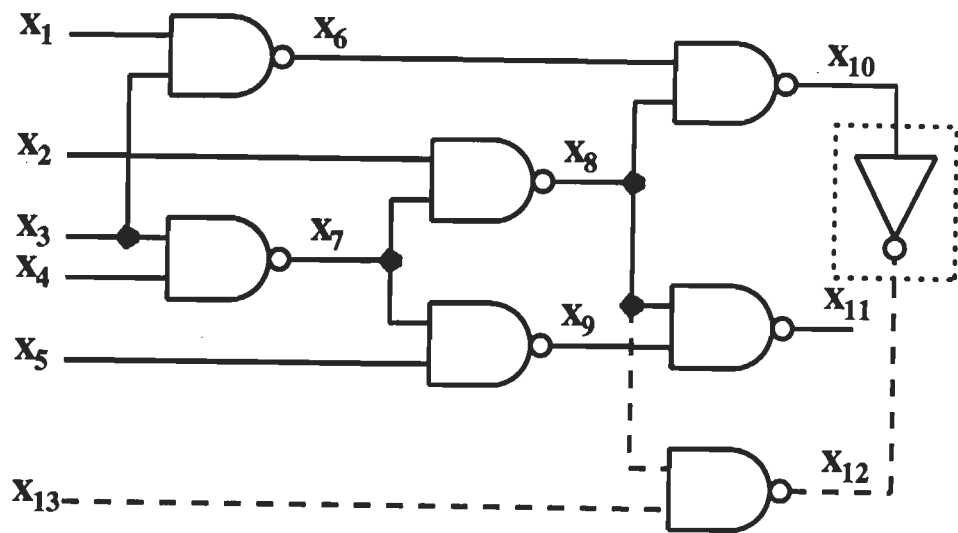
$$maxp \left(\frac{\partial E}{\partial b} \right) = 4, \quad minp \left(\frac{\partial E}{\partial b} \right) = -2, \quad \text{and} \quad Curr_Ones \left(\frac{\partial E}{\partial b} \right) = 4,$$

Again, looking at the values of $maxp$ and $minp$, both of the strong criteria S-I and S-II fail and S-IV becomes applicable in fixing variables a and b since $Curr_Ones$ becomes equal to $maxp$. Since the criterion S-IV provides only the locally best solution and not the global one, the decision of fixing one variable at a time should be taken. Now, which variable to be assigned first, is determined by using heuristics. One simple heuristic is to give priority to the primary input variable if it is available for the assignment. So, with the help of this heuristic and the criterion S-IV, the variable b may be assigned to a value 0. By substituting $b = 0$, the partial derivative of E with respect to a comes out to be -4 , which means that both $maxp$ and $minp$ are less than 0 and by applying the criterion S-II, a can be fixed at a value 1. After assigning values to all the variables, *i.e.* $a = 1, b = 0$ and $c = 0$ in (3.19), E comes out to be 0 which is the required solution.

To see the effectiveness of the quadratic 0-1 programming technique just described, let us consider an ISCAS'85 combinational benchmark circuit, *c17.isc* for generating a test pattern for the stuck-at-1 fault on line x_6 as shown in Figure 3.5(a). The sub-circuit copy, affected by the fault, with an output interface is shown in Figure 3.5(b). The ATPG neural graph for the given fault in the ISCAS circuit is given in Figure 3.6. The weight matrix \mathbf{Q} , the threshold vector \mathbf{c} and the transpose of the variable vector, \mathbf{x}^T to calculate the energy function, $E(\mathbf{x})$ of the neural network are given by:



(a)



(b)

Figure 3.5: (a) An ISCAS'85 circuit: c17.isc (b) Modified ISCAS circuit for the stuck-at-1 fault on line x_6 .

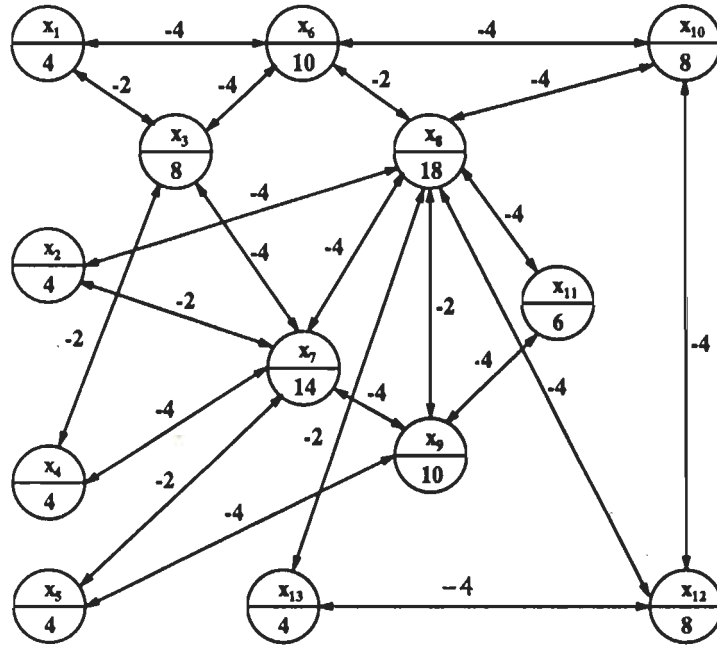


Figure 3.6: ATPG neural graph for the ISCAS circuit.

$$Q = \begin{bmatrix}
 0 & 0 & -2 & 0 & 0 & -4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & -2 & -4 & 0 & 0 & 0 & 0 & 0 \\
 -2 & 0 & 0 & -2 & 0 & -4 & -4 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & -2 & 0 & 0 & 0 & -4 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & -4 & 0 & 0 & 0 & 0 \\
 -4 & 0 & -4 & 0 & 0 & 0 & 0 & -2 & 0 & -4 & 0 & 0 & 0 \\
 0 & -2 & -4 & -4 & -2 & 0 & 0 & -4 & -4 & 0 & 0 & 0 & 0 \\
 0 & -4 & 0 & 0 & 0 & -2 & -4 & 0 & -2 & -4 & -4 & -4 & -2 \\
 0 & 0 & 0 & 0 & -4 & 0 & -4 & -2 & 0 & 0 & -4 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & -4 & 0 & -4 & 0 & 0 & 0 & -4 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & -4 & -4 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & -4 & 0 & -4 & 0 & 0 & -4 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & -4 & 0
 \end{bmatrix} \quad (3.27)$$

$$\mathbf{c}^T = [4 \ 4 \ 8 \ 4 \ 4 \ 10 \ 14 \ 18 \ 10 \ 8 \ 6 \ 8 \ 4]; \quad (3.28)$$

$$\text{and } \mathbf{x}^T = [x_1 \ x_2 \ x_3 \ x_4 \ x_5 \ x_6 \ x_7 \ x_8 \ x_9 \ x_{10} \ x_{11} \ x_{12} \ x_{13}]; \quad (3.29)$$

Since the test generation constraints are basically derived from the sensitization of a fault and the fault-effect propagation to the PO. Therefore, in this example first, fix the variables x_6 and x_{13} at values 0 and 1 respectively in order to sensitize the stuck-at-1 fault at line x_6 which is the primary requirement in the test generation process. By substituting these values of x_6 and x_{13} , the energy function comes out to be:

$$\begin{aligned} E(\mathbf{x}) = & 2x_1x_3 + 2x_2x_7 + 4x_2x_8 + 2x_3x_4 + 4x_3x_7 + 4x_4x_7 + 2x_5x_7 + 4x_5x_9 + 4x_7x_8 + \\ & 4x_7x_9 + 2x_8x_9 + 4x_8x_{10} + 4x_8x_{11} + 4x_8x_{12} + 4x_9x_{11} + 4x_{10}x_{12} - 4x_1 - 4x_2 - \\ & 8x_3 - 4x_4 - 4x_5 - 14x_7 - 16x_8 - 10x_9 - 8x_{10} - 6x_{11} - 4x_{12} + 40 \end{aligned} \quad (3.30)$$

In this test generation example, the variable x_{10} appearing in the fault-free circuit is related to its corresponding variable in the faulty copy, *i.e.* x_{12} . Both the variables are bound variables and their relation is given by $x_{10} + x_{12} = 1$. This relationship of bound variables can be generalized using the constraint $x_i + x_j = 1$, where x_i and x_j are the bound variables.

The partial derivatives of $E(\mathbf{x})$ with respect to each free variable, x_i and their corresponding minimum and maximum values, *i.e.* $minp$ and $maxp$ respectively are calculated as follows:

$$\frac{\partial E}{\partial x_1} = 2x_3 - 4; \quad minp = -4, \quad maxp = -2; \quad (3.31)$$

$$\frac{\partial E}{\partial x_2} = 2x_7 + 4x_8 - 4; \quad minp = -4, \quad maxp = 2; \quad (3.32)$$

$$\frac{\partial E}{\partial x_3} = 2x_1 + 2x_4 + 4x_7 - 8; \quad minp = -8, \quad maxp = 0; \quad (3.33)$$

$$\frac{\partial E}{\partial x_4} = 2x_3 + 4x_7 - 4; \quad minp = -4, \quad maxp = 2; \quad (3.34)$$

$$\frac{\partial E}{\partial x_5} = 2x_7 + 4x_9 - 4; \text{ minp} = -4, \text{ maxp} = 2; \quad (3.35)$$

$$\frac{\partial E}{\partial x_7} = 2x_2 + 4x_3 + 4x_4 + 2x_5 + 4x_8 + 4x_9 - 14; \text{ minp} = -14, \text{ maxp} = 6 \quad (3.36)$$

$$\frac{\partial E}{\partial x_8} = 4x_2 + 4x_7 + 2x_9 + 4x_{10} + 4x_{11} + 4x_{12} - 16; \text{ minp} = -16, \text{ maxp} = 4 \quad (3.37)$$

$$\frac{\partial E}{\partial x_9} = 4x_5 + 4x_7 + 2x_8 + 4x_{11} - 10; \text{ minp} = -10, \text{ maxp} = 4; \quad (3.38)$$

$$\frac{\partial E}{\partial x_{11}} = 4x_8 + 4x_9 - 6; \text{ minp} = -6, \text{ maxp} = 2; \quad (3.39)$$

Using the criteria for fixing the free variables, it is easily seen that the criterion S-II is satisfied by (3.31) and (3.33) and therefore, the variables x_1 and x_3 can be fixed to a value 1. Since the variable x_{10} and x_{12} are related variables, the partial derivatives of $E(\mathbf{x})$ with respect to each related variable x_i and their corresponding *Curr_Ones* values are calculated and are given by:

$$\frac{\partial E}{\partial x_{10}} = 4x_8 + 4x_{12} - 8; \text{ Curr_Ones} = 0; \quad (3.40)$$

$$\frac{\partial E}{\partial x_{12}} = 4x_8 + 4x_{10} - 4; \text{ Curr_Ones} = 4; \quad (3.41)$$

According to the criteria for fixing the related variables, *i.e.* R-I and R-II, one must know, whether there exists a unique maximum and/or minimum value among the *Curr_Ones* values. In this particular case, both the unique maximum and the unique minimum exist for x_{12} and x_{10} respectively. And, therefore, using the criteria R-I and R-II, x_{12} and x_{10} values can be fixed at 0 and 1 respectively. These values of x_{10} and x_{12} also satisfy the test generation constraint, *i.e.* $x_{10} + x_{12} = 1$.

Up to this point, to confirm the validity of the present approach, one can verify these steps by using the conventional test generation method. In the conventional method first, one needs to activate the stuck-at-1 fault at the fault site (output of the NAND gate x_6 in Figure 3.5) by setting its fault-free value at 0 which can only be done when both the inputs of the NAND gate are to be set to 1, *i.e.* x_1 and x_3 both

have to be 1. And, when the input to the NAND gate x_{10} , i.e. x_6 is assigned a value 0, its output x_{10} will remain at value 1 since the 0 value is the controlling value of the NAND gate. The second step in the test generation method is to propagate the fault effect to the primary output. In that case, a PO through which the fault effect is propagated, that PO should have different logical value in the fault-free and the faulty circuit. Since x_{10} is 1, x_{12} will be 0. In this approach, the same values of the variables, x_1 , x_3 , x_{10} and x_{12} have been fixed to 1, 1, 1 and 0 respectively which confirms the validity of the new approach up to this point.

Substituting the pivoted values of the variables x_1 , x_3 , x_{10} and x_{12} in (3.31) through (3.39), the partial derivatives of E and their $minp$ and $maxp$ values come out to be:

$$\frac{\partial E}{\partial x_2} = 2x_7 + 4x_8 - 4; \min p = -4, \max p = 2; \quad (3.42)$$

$$\frac{\partial E}{\partial x_4} = 4x_7 - 2; \min p = -2, \max p = 2; \quad (3.43)$$

$$\frac{\partial E}{\partial x_5} = 2x_7 + 4x_9 - 4; \min p = -4, \max p = 2; \quad (3.44)$$

$$\frac{\partial E}{\partial x_7} = 2x_2 + 4x_4 + 2x_5 + 4x_8 + 4x_9 - 10; \min p = -10, \max p = 6; \quad (3.45)$$

$$\frac{\partial E}{\partial x_8} = 4x_2 + 4x_7 + 2x_9 + 4x_{11} - 12; \min p = -12, \max p = 2; \quad (3.46)$$

$$\frac{\partial E}{\partial x_9} = 4x_5 + 4x_7 + 2x_8 + 4x_{11} - 10; \min p = -10, \max p = 4; \quad (3.47)$$

$$\frac{\partial E}{\partial x_{11}} = 4x_8 + 4x_9 - 6; \min p = -6, \max p = 2; \quad (3.48)$$

It may be noted that the remaining variables are only the free variables as all the related variables have been fixed. Unfortunately, at this moment, no decision can be taken to fix any of the remaining free variables as none of them satisfies the criteria S-I and S-II. So, *Curr_Ones* has to be calculated to make a locally best decision and in this particular case all *Curr_Ones* values for the remaining variables have the same value as their corresponding *maxp* values since the coefficients of all the variable terms

appeared in the partial derivative are positive. According to the fixation criterion S-IV, *i.e.* if $Curr_Ones \geq maxp$ for a given variable, say x_i , it may be assigned a value 0 and hence, all the remaining variables deserve to be fixed to 0. Since the criterion S-IV provides a locally best decision, the decision should be taken to fix a single variable at a time and should not be enforced to all the variables. But the problem is that which variable to be chosen first for fixation. To have a choice, one can use heuristics and there are two possible heuristics based on the number of occurrences of a variable in the partial derivatives as given below:

Heuristic 1. One possible heuristic is that the preference may be given to a variable which has more number of its occurrence in the partial derivatives. In the above example, x_7 will be preferred over x_8 because number of occurrence of x_7 is five, while x_8 occurs only in the four partial derivatives. The idea in this heuristic is that the priority should be given to a variable which reduces as many number of terms in the partial derivatives as possible.

Heuristic 2. The other possible heuristic is that the preference may be given to a variable which has less number of its occurrence in the partial derivatives. In the above example, x_2 and x_{11} have equal number of occurrences, *i.e.* 2 then the sum of their coefficients will decide the preference. As the aim is to minimize the energy function, the priority may be given to a variable which has the larger sum since the variable is going to be fixed at 0. In that case, x_{11} will be preferred over x_2 . The idea is that the priority should be given to a variable which is less dependent on other variables.

These heuristics can be modified in the same fashion for a case when $Curr_Ones \leq minp$ for a variable, say x_i and according to the criterion S-III, x_i may be fixed at value

1. Using the first heuristic, fix x_7 at a value 0 and now (3.42)–(3.48) reduce to

$$\frac{\partial E}{\partial x_2} = 4x_8 - 4; \text{ minp} = -4, \text{ maxp} = 0; \quad (3.49)$$

$$\frac{\partial E}{\partial x_4} = -2; \text{ minp} = -2, \text{ maxp} = -2; \quad (3.50)$$

$$\frac{\partial E}{\partial x_5} = 4x_9 - 4; \text{ minp} = -4, \text{ maxp} = 0; \quad (3.51)$$

$$\frac{\partial E}{\partial x_8} = 4x_2 + 2x_9 + 4x_{11} - 12; \text{ minp} = -12, \text{ maxp} = -2; \quad (3.52)$$

$$\frac{\partial E}{\partial x_9} = 4x_5 + 2x_8 + 4x_{11} - 10; \text{ minp} = -10, \text{ maxp} = 0; \quad (3.53)$$

$$\frac{\partial E}{\partial x_{11}} = 4x_8 + 4x_9 - 6; \text{ minp} = -6, \text{ maxp} = 2; \quad (3.54)$$

The partial derivatives given by (3.49)–(3.53) satisfy the strong criterion S-II and therefore, now it is possible to fix x_2, x_4, x_5, x_8 and x_9 at value 1 and by substituting the values of x_8 and x_9 in (3.54), the partial derivative of E with respect to x_{11} comes out to be a positive constant, *i.e.* 2. Hence, x_{11} can be fixed at a value 0 and finally the solution is obtained which in turn provides the consistent labeling of the logical assignments in the circuit and can be verified accordingly. The logical values of the primary input signal variables, *i.e.* $x_1 = 1, x_2 = 1, x_3 = 1, x_4 = 1$ and $x_5 = 1$ form the required test pattern for the x_6 stuck-at-1 fault in the example circuit.

Now, let us consider the second heuristic and see its effectiveness. According to this heuristic, one can fix x_{11} at value 0 and due to this fixation, the partial derivatives given by (3.46) and (3.47) are reduced. Hence, the partial derivatives given by (3.42)–(3.48) and their *minp* and *maxp* values can be derived as

$$\frac{\partial E}{\partial x_2} = 2x_7 + 4x_8 - 4; \text{ minp} = -4, \text{ maxp} = 2; \quad (3.55)$$

$$\frac{\partial E}{\partial x_4} = 4x_7 - 2; \text{ minp} = -2, \text{ maxp} = 2; \quad (3.56)$$

$$\frac{\partial E}{\partial x_5} = 2x_7 + 4x_9 - 4; \text{ minp} = -4, \text{ maxp} = 2; \quad (3.57)$$

$$\frac{\partial E}{\partial x_7} = 2x_2 + 4x_4 + 2x_5 + 4x_8 + 4x_9 - 10; \text{ minp} = -10, \text{ maxp} = 6; \quad (3.58)$$

$$\frac{\partial E}{\partial x_8} = 4x_2 + 4x_7 + 2x_9 - 12; \text{ minp} = -12, \text{ maxp} = -2; \quad (3.59)$$

$$\frac{\partial E}{\partial x_9} = 4x_5 + 4x_7 + 2x_8 - 10; \text{ minp} = -10, \text{ maxp} = 0; \quad (3.60)$$

The partial derivatives given by (3.58) and (3.59) satisfy by the criterion S-II and therefore, both the x_8 and x_9 can be fixed at 1, which again reduces the above set of partial derivatives to

$$\frac{\partial E}{\partial x_2} = 2x_7; \text{ minp} = 0, \text{ maxp} = 2; \quad (3.61)$$

$$\frac{\partial E}{\partial x_4} = 4x_7 - 2; \text{ minp} = -2, \text{ maxp} = 2; \quad (3.62)$$

$$\frac{\partial E}{\partial x_5} = 2x_7; \text{ minp} = 0, \text{ maxp} = 2; \quad (3.63)$$

$$\frac{\partial E}{\partial x_7} = 2x_2 + 4x_4 + 2x_5 - 2; \text{ minp} = -2, \text{ maxp} = 6; \quad (3.64)$$

Applying the criterion S-I to (3.61) and (3.63), one is able to fix x_2 and x_5 at 0 and only two unpivoted variables x_4 and x_7 are remaining for which the partial derivatives are

$$\frac{\partial E}{\partial x_4} = 4x_7 - 2; \text{ minp} = -2, \text{ maxp} = 2; \quad (3.65)$$

$$\frac{\partial E}{\partial x_7} = 4x_4 - 2; \text{ minp} = -2, \text{ maxp} = 2; \quad (3.66)$$

At this juncture, no decision can be taken to pivot either of the two variables and now suppose x_7 variable is assigned a value 1 then it may be seen that

$$x_7 = 1 \Rightarrow \frac{\partial E}{\partial x_4} = 2 \Rightarrow \underbrace{x_4 = 0}_{\text{(pivoted)}} \Rightarrow \frac{\partial E}{\partial x_7} = -2 \Rightarrow \underbrace{x_7 = 1}_{\text{(pivoted)}}.$$

which means

$$x_7 = 1 \Leftrightarrow x_4 = 0.$$

and the system becomes *stable*. Now, let us see what happens when x_7 is fixed at value 0.

$$x_7 = 0 \Rightarrow \frac{\partial E}{\partial x_4} = -2 \Rightarrow \underbrace{x_4 = 1}_{\text{(pivoted)}} \Rightarrow \frac{\partial E}{\partial x_7} = 2 \Rightarrow \underbrace{x_7 = 0}_{\text{(pivoted)}}$$

i.e.

$$x_7 = 0 \Leftrightarrow x_4 = 1.$$

Hence, it is clear that the system becomes stable with $x_4 = 0$ and $x_7 = 1$ or $x_4 = 1$ and $x_7 = 0$. Interestingly, from the circuit shown in Figure 3.5(a), it can be easily seen that the output of the NAND gate, *i.e.* x_7 has two inputs x_3 and x_4 and x_3 got assigned to the value 1 during the fault excitation. The output of the NAND gate x_7 will be 0 when signal line x_4 is assigned the logic value 1 and the output will be the logic value 1 when x_4 is assigned to a 0 value. So, in this case, there will be two test patterns to detect the stuck-at-1 fault on the signal line x_6 in the example circuit.

3.4 Computational Algorithm

In the pseudo-Boolean quadratic function $E(\mathbf{x})$, the number of variables are n and let m be the number of test generation constraints defined in the previous section. The partial derivatives of the objective function with respect to the variable x_i is denoted by $PD(x_i)$. The *maxp*, *minp* and *Curr_Ones* are also defined previously. The following algorithm in C-like language finds the minimum of $E(\mathbf{x})$:

1. **for** $i \leftarrow$ **to** n **do** /* Calculates Partial Derivatives $PD(x_i)$. */
 - if** x_i is *free_variable*
 - Calculate $PD(x_i)$;
 - Compute *maxp*, *minp* and *Curr_Ones* values;
 - end if**
 - if** x_i is *related_variable*
 - Calculate $PD(x_i)$;
 - Compute *Curr_Ones* values;

```

    end if

end for

2. Check_Ones(). /* This function fixes the free and related variables. */

2.1 for  $i \leftarrow$  to  $p$  do /*  $p$  be the number of free variables. */

    2.1.1 if  $maxp \geq 0 \ \&\& \ minp \geq 0$  /* Monotonically increasing. */
        pivot the variable  $x_i$  at value 0;
    2.1.2 if  $maxp \leq 0 \ \&\& \ minp \leq 0$  /* Monotonically decreasing. */
        pivot the variable  $x_i$  at value 1;
    2.1.3 if  $x_i$  is not_pivoted && ( $Curr\_Ones \leq minp \ || \ Curr\_Ones \geq$ 
         $maxp$ )
        decide a variable  $x$  to be pivoted; /* Apply heuristics
        */
    2.1.4 if  $Curr\_Ones \leq minp$ 
        pivot the variable  $x$  at value 1;
        go to Step 3;
    end if
    2.1.5 if  $Curr\_Ones \geq maxp$ 
        pivot the variable  $x$  at value 0;
        go to Step 3;
    end if

end for /* It fixes the free variables. */

2.2 for  $i \leftarrow$  to  $q$  do /*  $q$  be the number of related variables. */

    Calculate  $maxp$  and  $minp$  among the  $Curr\_Ones$  vector;

```

```

2.2.1 if maxp exists for unique  $x_i$ 
    Fix  $x_i$  variable at value 0;
    Fix constraint variable of  $x_i$  at 1;
end if

2.2.2 if minp exists for unique  $x_i$ 
    Fix  $x_i$  variable at value 1;
    Fix constraint variable of  $x_i$  at 0;
end if

2.2.3 if maxp and minp does not exist for unique  $x_i$ 
    select any variable  $x_i$ ; /* Make a heuristic choice. */

2.2.4 if Curr_Ones of  $x_i \neq$  Curr_Ones of the constraints of  $x_i$ 
    Fix corresponding  $x_i$  variable at value 0;
    Fix constraint variable of  $x_i$  at 1;
end if

if Curr_Ones of  $x_i ==$  Curr_Ones of the constraints of  $x_i$ 
    Fix arbitrary  $x_i$  variable at value 1;
    Fix constraint variable of  $x_i$  at 0;
    Store maxp, minp, Curr_Ones;
    Push decision into a stack; /* for backtracking. */
end if

end for /* It fixes the related variables. */

3. if all variables pivoted (i.e.  $n = r$ ) /*  $r$  be the number of pivoted variables. */

    go to Step 4;

for  $i \leftarrow$  to  $r$  do

```

```

    Compute maxp, minp, and Curr_Ones values;
    go to Step 2; /* Check for fixation of the  $x_i$  variables.*/

end for

4. Evaluate the energy of the function  $E(x)$ ; /* The energy is the function value.
*/

5. while energy  $\neq$  energy_required & all variables not_pivoted do

    if stack is empty

        go to Step 6;

    Backtrack; /* Reverse the decision.*/
    go to Step 3;

6. end /* the optimum solution */

```

3.5 Complexity Analysis

The complexity of the algorithm described in the previous section lies mainly in calculating the partial derivatives and in the evaluation of the pseudo-Boolean quadratic function $E(\mathbf{x})$. The calculation of the partial derivatives and the corresponding functional value can be done in linear time. It can be observed from the description of the algorithm that single *for* loops, *if – else* statements and *while* loop are mainly used. The running time of a *for* loop is at the most the running time of the statements inside the *for* loop (including tests) times the number of iterations. The number of iterations in the algorithm is n , *i.e.* the number of variables in the pseudo-Boolean quadratic function $E(\mathbf{x})$. The statements inside the *for* loop are the calculations of

the partial derivatives, *maxp*, *minp* and *Curr_Ones* which can be performed in linear time. So the time-complexity of the *for* loops used in our program comes out to be $\mathcal{O}(n^2)$. As far as the running time of an *if - else* statement is concerned, it is never more than the running time of the test plus the larger of the running times of the statements under this condition. The rest of the algorithm is only the linear time complex, *i.e.* $\mathcal{O}(n)$. Thus the overall run-time complexity of the algorithm turns out to be $\mathcal{O}(n^2)$.

The run-time complexity of the proposed algorithm may be further decreased by using an efficient data structure for the **Q** matrix representation and an appropriate algorithm for the computation of the partial derivatives with their maximum and minimum values. It is also important to note that in this polynomial time algorithm, heuristics have been employed in order to accelerate the minimization process. Furthermore, due to the backtracking (branch-and-bound) procedure although incorporated in the last stage of the algorithm, it may become too time consuming in the case of the test pattern generation for the large complex circuits. However, the complexity of this branch-and-bound step in the proposed algorithm is further reduced with the use of efficient heuristics. In order to overcome from the complexity of the branch-and-bound step of this quadratic 0-1 programming algorithm, new test generation methods have been proposed [94], [95] and described in the next chapter.

Chapter 4

Genetic Algorithm Based Test Generation

Once the test generation problem has been formulated as a search or an optimization problem, the well known optimization techniques like gradient descent, simulated annealing [20], quadratic 0-1 programming [19], transitive closure [22], *etc.* have been attempted for finding the global optimum solution of the problem. A new quadratic 0-1 programming technique has been proposed for the test generation problem and described in the Chapter 3, though seems to be quite efficient, but it may not be that much effective in case of large practical circuits due to its branch-and-bound step which becomes essential when no criteria gets satisfied during the dynamic fixation of the variables. The transitive closure based test generation algorithm on the other hand derives a Boolean false function from a formula expressing Boolean difference between the fault-free circuit and its faulty image, *i.e.* the copy of a sub-circuit affected by the fault in question. In this approach, the test generation problem is transformed into the problem of determining signal assignments that satisfy the false function. This approach is equivalent to the minimization of the energy function, which is of the form

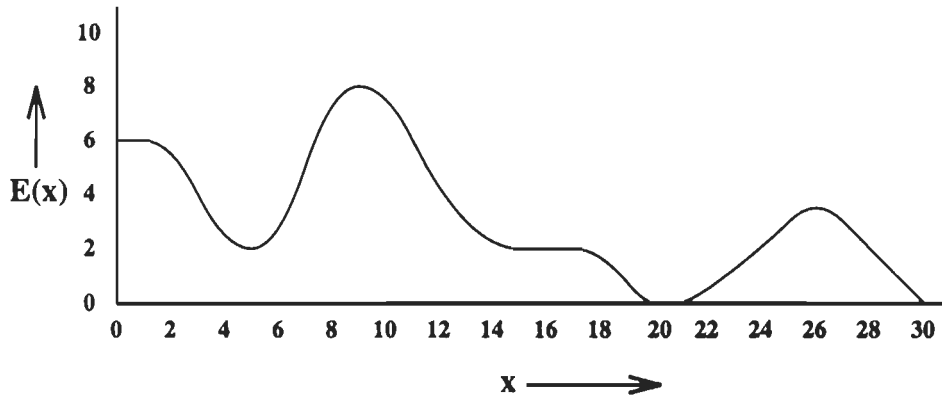


Figure 4.1: A plot for the energy function.

of the pseudo-Boolean quadratic function.

The nature of the minimization function plays an important role in the selection of the appropriate optimization algorithms. A study has been made to know the behavior of the energy function and the Boolean false function. Both the functions are derived for the test generation of stuck-at faults in a simple ISCAS'85 combinational benchmark circuit and are of multimodal type. These functions are depicted via plots shown in Figure 4.1 and 4.2. The global optimum solution of the problem represented by these functions can be located in a multimodal landscape.

Genetic algorithms, being one of the most effective search and optimization algorithms in a multimodal search space, lend themselves readily for the test generation problem as it can be formulated either as a search of the N dimensional 0-1 state space, where N is the number of primary inputs or an optimization problem. Since in test generation, the crux of the problem is either search or optimization, the choice of Genetic Algorithms (GAs) is more impressive considering its proven track record as one of the most robust search and global optimization algorithms applied in a number of combinatorial optimization problems. The robustness of GAs is due to their capacity

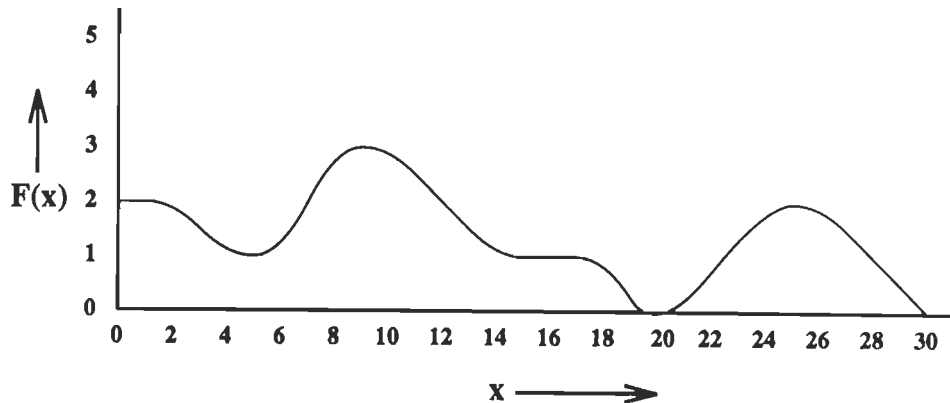


Figure 4.2: A plot for the Boolean false function.

to locate the global optimum solution in a multimodal landscape. Furthermore, GAs are innovative and have inherent amenability to be processed in parallel.

This chapter starts with a gentle introduction to genetic algorithms and describes new methods for test generation using these algorithms. Emphasis is given to derive an objective function which is essentially required to guide the search to be performed by GAs. Schema design and a penalty method to transform the constrained test generation problem into an unconstrained one are discussed. In the application of GAs, the objective function must be in fitness form, *i.e.* in the maximization form. A procedure is described to map the objective function to a fitness form so that it can be directly utilized by GAs for test generation. Finally, a CAD tool based on GAs has been developed for test generation and described in detail.

4.1 Genetic Algorithms in Search and Optimization

Genetic algorithms are search algorithms based on the principles of natural selection and genetics and draw inspiration from the natural evolution process and simulate the Darwinian principle of survival of the fittest. In the early 1970s, Holland proposed Genetic Algorithms as computer programs that mimic the evolutionary processes in nature. These algorithms manipulate a population of potential solutions of a search or an optimization problem. GAs work specifically with encoded representations of the solutions that is equivalent to the genetic material of individuals in nature and not directly on the solutions themselves. The solutions are encoded as *strings* of bits from a binary alphabet. Each solution is associated with a *fitness value* that provides the information about the goodness of the solution as compared to other solutions in the population. Higher the fitness value of an individual, the higher is its chances of survival and reproduction, and the larger is its representation in the subsequent generation. GAs rely on two basic operators, namely *crossover* and *mutation*. Crossover operator is a mechanism of probabilistic and useful exchange of information among solutions to locate better solutions and mutation operator causes sporadic and random alteration of the bits of the strings. Both these operators have direct analogy from nature as crossover recombines the genetic material while mutation plays the role of regenerating lost genetic material.

4.1.1 A Simple Genetic Algorithm

A Simple Genetic Algorithm (SGA) works essentially with a population of binary strings, where each string consists of 0s and 1s is the encoded representation of a

solution to the search or optimization problem. Genetic operators – crossover and mutation, allow the SGA to create the subsequent generation from the binary strings of the current population. This cycle is repeated until a desired optimum solution is found or a termination criterion is satisfied, by predefining a number of generations to be processed. The basic structure of SGA is summarized as follows:

```
Simple_Genetic_Algorithm()  
{  
    Initialize_Population;  
    Evaluate_Population;  
    While termination_criterion not satisfied  
    {  
        Select solutions for next population;  
        Perform crossover and mutation;  
        Evaluate_Population;  
    }  
}
```

In principle, an SGA is characterized by the components given below:

- Encoding Mechanism
- Population
- Fitness Function
- Selection Mechanism
- Crossover and Mutation
- Control Parameters

These components are explained below in the context of the test generation problem.

Encoding Mechanism

The encoding mechanism represents the problem variables in binary form so that the solution should be mapped to a binary string. Fortunately, for the test generation problem, the variables are the digital signals of the circuit-under-test and have only the binary values. Since the test generation problem has been formulated as a search or optimization problem, its solution, if exists, will also be a binary string. The individual bits of the string correspond to the signal values of the test generation network and there will a consistent labeling of the signal values with no violation in the functionality of any gate.

Population

Let us consider an m -tuple (x_1, x_2, \dots, x_m) of the signal values of the test generation network. An m -bit string obtained from this m -tuple can be defined as:

$$\mathbf{x} = x_1x_2\dots x_m$$

where each $x_i \in \{0, 1\}$. An initial population is generated by n such strings given as follows:

$$\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(n)}$$

where each $\mathbf{x}^{(i)} = x_1x_2\dots x_m$; $x_j \in \{0, 1\}$, $1 \leq j \leq m$.

The initial population of strings $\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(n)}\}$ can be generated by the toss of a coin for each x_j of each $\mathbf{x}^{(i)}$. Starting with an initial population of strings, the SGA creates the subsequent generation by applying the crossover and the mutation operator. Essentially, it is a process of putting together the bits and pieces of the fittest

of the old and is achieved from the knowledge of the fitness values (evaluated from the fitness function) associated with each individual string.

Fitness Function

In the application of GAs, the objective function, *i.e.* the function to be optimized, is essentially required for evaluating each string. Since the objective function values may vary from problem to problem, the uniformity should be maintained over various problem domains. The *fitness function* is used to normalize the objective function values to a range between 0 and 1. This normalized value of the objective function is termed as *fitness* of the string and is used in the selection mechanism to evaluate the strings of the population.

Selection Mechanism

In selection process, individual strings are copied according to their objective function values based on the nature's Darwinian principle of survival of the fittest. 'Good' solutions are *selected* for reproduction while 'bad' solutions are eliminated and the 'goodness' of the solution is determined by its fitness value. The higher the fitness value of an individual, the higher its chances of survival and reproduction and the larger its representation in the subsequent generation.

The selection procedure may be implemented in algorithmic form in a number of ways. However, the most popular one is a *proportionate selection scheme* in which a string with fitness value f_i is allocated f_i/\bar{f} offspring, where \bar{f} is the average fitness value of the population. The number f_i/\bar{f} represents the string's expected number of offspring. A string with a fitness value higher than the average is allocated more than one offspring, while a string with a fitness value less than the average is allocated

less than one offspring. Allocation methods include some randomization to remove methodical allocation toward any particular set of strings so that the actual allocation of offspring to strings matches the expected number of offspring f_i/\bar{f} .

The SGA implements the proportionate selection using a *roulette wheel selection scheme* [46], where each current string in the population has a roulette wheel slot sized in proportion to its fitness. The angle subtended by the sector at the center of the wheel equals to $2\pi f_i/\bar{f}$. A string is allocated an offspring if a randomly generated number in the range 0 to 2π falls in the sector corresponding to the string. The entire population can be generated for the next generation by selecting the strings with the help of this scheme. In this scheme, there could be large sampling errors in the sense that the final number of offspring allocated to a string might vary significantly from the expected number.

Crossover and Mutation

Crossover is a crucial operator of the SGA and comes after selection. It proceeds in two steps. First, members of the newly reproduced strings in the mating pool are mated at random. Second, each pair of strings undergoes for crossover. The SGA uses a very simple approach of single-point crossover. Let l be the string length. An integer position k along the string is selected uniformly at random in the range 1 to $l - 1$. The portions of the two strings beyond this integer position (crossover point) are exchanged to form two new strings. The crossover point may assume any of the $l - 1$ possible values with equal probability. After choosing a pair of strings, the crossover is invoked by the algorithm only if a randomly generated number in the range 0 to 1 is greater than the crossover probability, p_c . Otherwise, the strings remain unmodified. In a large population, p_c gives the fraction of strings actually crossed.

Mutation plays a secondary role of restoring lost genetic material in the SGA and comes after crossover. Mutation of a bit involves flipping it, *i.e.* changing a 0 to 1 or vice-versa. Similar to p_c , which controls the probability of crossover, another control parameter p_m (the mutation probability), gives the probability that a bit will be flipped. The bits of a string are independently mutated, *i.e.* the mutation of a bit does not affect the probability of mutation of other bits.

Control Parameters

The optimum performance of the SGA depends on the choice of control parameters, namely the crossover and mutation probabilities and the population size. The trade-offs that arise in this regard are mentioned below:

- Increasing the crossover probability increases recombination of building blocks, but it also increases the disruption of good strings.
- Increasing the mutation probability tends to transform the genetic search into a random search, but it also helps restoring the lost genetic material.
- Increasing the population size increases its diversity and reduces the probability that the GA will prematurely converge to a local optimum, but it also increases the time required for the population to converge to the optimal regions in the search space.

Although the choice of the control parameters is an open research problem, but two distinct parameter sets have emerged. One has a small population size and relatively large crossover and mutation probabilities, while the other has a large population size, but much smaller crossover and mutation probabilities. Typically, these two categories are:

- population size 30, crossover probability (p_c) 0.9, and mutation probability (p_m) 0.01 [50].
- population size 100, crossover probability (p_c) 0.6, and mutation probability (p_m) 0.001 [30].

Apart from selection, crossover and mutation, various other operators are common in the working of GAs. One such operator is *fitness scaling* [50] and is widely used. Fitness scaling readjusts fitness values of the strings in order to prevent the premature convergence of the population to suboptimal solutions.

In optimizing multimodal functions, it is important that GAs should be able to locate the region in which the global optimum exists and then to converge to the optimum. Although GAs possess hill-climbing properties essential for multimodal function optimization, they may also stuck at a local optimum, especially when the population size is small. To improve the overall GA performance in multimodal function optimization, adaptive probabilities of crossover and mutation have been used [99]. The GA based on the adaptive crossover and mutation probabilities is called as Adaptive Genetic Algorithm (AGA).

4.1.2 Adaptive Genetic Algorithms

In multimodal function optimization, GAs must have two characteristics – its capacity to converge to an optimum (local or global) after locating the region containing the optimum and the capacity to explore new regions of the solution space in search of the global optimum. These two characteristics could be achieved by varying the probability of crossover (p_c) and the probability of mutation (p_m) adaptively depending upon the fitness value of the solutions. These probabilities are increased when the popula-

tion tends to get stuck at a local optimum and are decreased when the population is scattered in the solution space.

It has been concluded, on the basis of extensive experiments on a wide range of problems in [99], that the AGA performs very well for the problems that are highly multimodal which means that the number of local optima in the search space is large. As the function to be optimized in the test generation problem is highly multimodal in nature, AGA should be preferred over the SGA because it outperforms the SGA. The comparative study of AGA with SGA has already been done extensively in [99] and hence not discussed in this chapter. The conclusions reported in [99] are directly exploited here for the optimization based approach for the test generation problem.

The most important contribution in the application of the AGA is the design of adaptive crossover and mutation probability, *i.e.* p_c and p_m respectively. Expressions for p_c and p_m are given as follows:

$$p_c = k_1(f_{max} - f')/(f_{max} - \bar{f}), f' \geq \bar{f} \quad (4.1)$$

$$p_c = k_3, f' < \bar{f} \quad (4.2)$$

and

$$p_m = k_2(f_{max} - f)/(f_{max} - \bar{f}), f \geq \bar{f} \quad (4.3)$$

$$p_m = k_4, f < \bar{f} \quad (4.4)$$

where,

\bar{f} : average fitness value of the population,
 f_{max} : maximum fitness value of the population,
 f : fitness value of the solution, and
 f' : larger fitness values of the solutions to be crossed.
 and $k_1, k_2, k_3, k_4 \leq 1.0$.

Since moderately large values of p_c ($0.5 < p_c < 1.0$) and the small values of p_m ($0.001 < p_m < 0.05$) are essential for the successful working of GAs [46], k_1, k_2, k_3, k_4 values must be assigned such that p_c and p_m remain in the prescribed range. Keeping these values of p_c and p_m in mind, for the successful working of AGA, the values of the k_1, k_2, k_3 , and k_4 coefficients are proposed to be 1.0, 0.5, 1.0 and 0.5 respectively.

4.2 Genetic Algorithms for Test Generation Problem

The application of GAs to the test generation problem has been recently attempted by few groups. Most of them use Random Test Generation (RTG) followed by the conventional fault simulation and the task for the GAs is only to minimize a cost function to find a test. These GA-based techniques are no more different than the simulation-based methods for the test generation except that the cost minimization is done using GAs. Major bottleneck in parallelization of the existing GA-based algorithms is the three stages in sequence, *i.e.* RTG, fault simulation and cost minimization using GAs which limits the overall throughput. Taking these points into consideration, new GA-based test generation methods have been proposed here. These methods are not using RTG followed by the conventional fault simulation at any stage.

In this new test generation approach, first the test generation problem is trans-

formed into either an optimization or a Boolean satisfiability problem and then GAs are applied to find the global optimum solution of the problem which in turn provides the test patterns for a given set of faults in the circuit-under-test. The problem representation and objective function (also called evaluation function) derivation are the key issues in the application of GAs. Problem representation is done in a very simple and straightforward manner but emphasis is given to derive the objective function. This is an important requirement in the application of GAs as it is used to guide the search by evaluating the optimality of the solution. For the test generation problem, the objective function can be derived using either of the following two approaches:

1. Energy minimization approach which provides a mathematically expressed circuit function based on an unconventional digital circuit modeling technique using neural networks [20].
2. Boolean satisfiability approach in which the test generation problem is transformed into satisfying a Boolean false function analogous to a Boolean truth function used in [66].

In these approaches, a test generation network for a circuit-under-test is constructed by creating an image (copy) of the sub-circuit affected by the given fault and connecting their corresponding primary outputs through an *output interface* in the same manner as described in Section 2.4.1. After construction of the test generation network, an objective function is derived based on the above-mentioned approaches. Both the approaches are different and discussed in the following Section.

4.3 Energy Minimization Approach

In this approach, the test generation network is transformed into a neural network using the unconventional digital circuit modeling technique discussed in Section 2.4.2. The energy function of the test generation neural network is given by

$$E(\mathbf{x}) = -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n Q_{ij} x_i x_j - \sum_{i=1}^n c_i x_i + K \quad (4.5)$$

where n is the number of neurons in the neural network, Q_{ij} is the weight associated with the link between neurons i and j such that $Q_{ii} = 0$, x_i is the activation value of neuron i , c_i is the threshold of neuron i and K is a constant.

In this approach, for all the testable faults, there exists a *consistent labeling* of the neurons in the neural network with their activation values from the set $\{0, 1\}$ that does not violate the functionality of any gate. For a given fault, a test pattern can be obtained by finding a global minimum of the energy function $E(\mathbf{x})$ given by the Equation (2.1). The global minimum of $E(\mathbf{x})$ will be 0 at which the neural network is labeled consistently. At the global minimum, the activation values of the corresponding primary input neurons form the test vector for the given fault. Hence, this approach transforms the test generation problem into an energy minimization problem.

The energy minimization problem for test generation is a constrained one because of the test generation constraints. In order to solve this problem with the help of GAs, there is a need to transform the constrained problem into an unconstrained one. This transformation is carried out with the help of schema design and a penalty method. Since in the application of GAs, the objective function must be in fitness form, *i.e.* in the maximization form, a mapping is done to obtain objective function into fitness form. The following subsections describe these procedures in detail.

4.3.1 Schema Design

The search by GAs is greatly simplified with the concept of schema, which is a powerful tool in the technique of GAs. A schema (plural: schemata) is a string that indicates the similarities at certain positions in a subset of strings and is generally represented by the symbols 0, 1, * (don't care). A schema and a string match each other if at certain specific locations either 1 or 0 exists in both and the schema has * in the remaining positions. The schema design is carried out by the schema theorem which states that short, low order, above average schemata receive exponentially increasing trials in subsequent generations. This is the essence of the schema theorem, first proposed by Holland as the "fundamental theorem of genetic algorithms". A formal statement of the schema theorem can be given by the following equation:

$$N(\mathbf{h}, t + 1) \geq N(\mathbf{h}, t) \frac{f(\mathbf{h}, t)}{\bar{f}(t)} \left[1 - p_c \frac{\delta(\mathbf{h})}{l - 1} - p_m o(\mathbf{h}) \right] \quad (4.6)$$

where

- $f(\mathbf{h}, t)$: average fitness value of schema \mathbf{h} in generation t ,
- $\bar{f}(t)$: average fitness value of the population in generation t ,
- p_c : crossover probability
- p_m : mutation probability
- $\delta(\mathbf{h})$: length of the schema \mathbf{h}
- $o(\mathbf{h})$: order of the schema \mathbf{h}
- $N(\mathbf{h}, t)$: expected number of instances of schema \mathbf{h} in generation t , and
- l : number of bit positions in a string or string length.

The factor:

$$p_c \frac{\delta(\mathbf{h})}{l - 1}$$

gives the probability that an instance of the schema \mathbf{h} is disrupted by crossover, and $p_m o(\mathbf{h})$ gives the probability that an instance is disrupted by mutation [46].

It has been proved that in a single generational cycle, the GA processes only n strings (n is the population size), but it leads to the processing of $\mathcal{O}(n^3)$ schemata. This tremendous increase in the computational power achieved by GAs to simultaneously process a large number of schemata is termed as *implicit parallelism*. This capacity of GAs arises from the fact that a string simultaneously represents 2^l different schemata.

Using the test generation constraints, the schema has been designed by defining a string where its alternate positions may represent the logical values of the original and duplicate signals. Thus, a schema can be obtained in which two consecutive bits are the inversion of one another. These two consecutive bits represent the logical values of the fault-free and faulty signals. For fault detection at least one of the primary output of the fault-free and the faulty states should be the complements of each other. Thus, for the m -bit string of logical values, the schema is defined as

$$\mathbf{h} = * \dots * x_j x_{j+1} * \dots * \quad (4.7)$$

where $x_j = \overline{x_{j+1}}$, for some j with in the limits given by $0 \leq j \leq (m-1)$. As the two bits of the schema are now fixed, a constraint is introduced into the energy minimization process using the penalty method.

4.3.2 Penalty Method

In order to discourage the strings violating the constraints, a penalty may be introduced. It should be such that those strings which tend to violate the constraints are reduced in fitness. The energy function given by (2.1) is considered as a measure of fitness and the constrained problem is stated as:

$$\text{Minimize } E(\mathbf{x}) \quad \text{subject to } h_i(\mathbf{x}) \geq 0$$

where \mathbf{x} is an m -tuple of the form discussed previously and h_i s are the constraints. The statement of the constrained problem can be transformed into an unconstrained problem as:

$$\text{Minimize } E'(\mathbf{x}) = E(\mathbf{x}) + \sum_i r_i \Phi[h_i(\mathbf{x})] \quad (4.8)$$

where Φ is a penalty function and r_i is the penalty coefficients or the weights associated with the penalty. The r_i values should be such that even moderate violation of the constraints invite significant penalty. The coefficients r_i may be sized separately to suit each constraint. The function Φ is defined by

$$\Phi[h_i(\mathbf{x})] = \begin{cases} 0 & \text{if } h_i(\mathbf{x}) \geq 0 \\ h_i^2(\mathbf{x}) & \text{otherwise} \end{cases} \quad (4.9)$$

4.3.3 Mapping Objective Function to Fitness Form

The objective in genetic algorithms is naturally stated as the maximization of the fitness function in contrast to the minimization problem at hand. Thus, the energy function given in (4.8) can not be directly treated as the fitness function. To map the energy minimization problem into fitness form, we use the following transformation:

$$f(\mathbf{x}) = \begin{cases} C_{max} - E'(\mathbf{x}) & E'(\mathbf{x}) \leq C_{max} \\ 0 & \text{otherwise} \end{cases} \quad (4.10)$$

where C_{max} is an input coefficient which may be either largest value of the function $E'(\mathbf{x})$ observed in a process or the largest value of $E'(\mathbf{x})$ in the last k -generations.

4.3.4 Example

Let us consider the test generation example for the circuit shown in Figure 3.5(a) with a s-a-1 fault at line x_6 . Figure 3.5(b) shows the test generation network after the

fault injection, *i.e.* by duplicating the fault-affected portion of the circuit. Figure 3.6 illustrates the corresponding ATPG neural graph where each signal is represented by a neuron having a threshold. Hopfield neural network model [69] is used in which the links are bidirectional and their labels show their weights. The schema just after the fault injection stage, *i.e.* due to fault activation, is

$$\mathbf{h}_1 = * * * * * 0 1 * * * * *$$

where each bit corresponds to the activation values of the $x_1, x_2, x_3, x_4, x_5, x_6, x_{13}, x_7, x_8, x_9, x_{10}, x_{12}$ and x_{11} neurons respectively.

For fault detection, at least one Primary Output (PO) of the fault-free circuit must have different value from its corresponding PO of the faulty circuit. In the current example, x_{10} and x_{12} are the POs of the fault-free circuit and its faulty image respectively. Since the task of the GAs is to search a string which has the different x_{10} and x_{12} values, two more schemata may be derived as given below:

$$\mathbf{h}_2 = * * * * * * * * * 1 0 *$$

$$\mathbf{h}_3 = * * * * * * * * * 0 1 *$$

Using the schema \mathbf{h}_1 , the constraints $h_{11}(\mathbf{x})$, $h_{12}(\mathbf{x})$ and $h_{13}(\mathbf{x})$ may be derived and given below with their conditions for violations:

$$h_{11}(\mathbf{x}) = (x_6 + x_{13} - 1) < 0 \quad \text{for } x_6 = x_{13} = 0; \quad (4.11)$$

$$h_{12}(\mathbf{x}) = (-x_6 + x_{13}) < 0 \quad \text{for } x_6 = 1, x_{13} = 0; \quad (4.12)$$

$$h_{13}(\mathbf{x}) = (-x_6 - x_{13} + 1) < 0 \quad \text{for } x_6 = x_{13} = 1; \quad (4.13)$$

Similarly, other constraints are derived using the schemata \mathbf{h}_2 and \mathbf{h}_3 with the conditions for their violation given as below:

$$h_2(\mathbf{x}) = (x_{10} + x_{12} - 1) < 0 \quad x_{10} = x_{12} = 0; \quad (4.14)$$

$$h_3(\mathbf{x}) = (-x_{10} - x_{12} + 1) \geq 0 \quad x_{10} = x_{12} = 1; \quad (4.15)$$

Although a number of alternatives exist to derive the penalty function Φ , but usually the square of the constraint, *i.e.* $\Phi[h_i(\mathbf{x})] = h_i^2(\mathbf{x})$ is used for all violated constraints as given in (4.9). Hence, the following penalty functions are derived for the above mentioned constraints:

$$\Phi[h_{11}(\mathbf{x})] = \begin{cases} (x_6 + x_{13} - 1)^2 & x_6 x_{13} = 00 \\ 0 & \text{otherwise} \end{cases} \quad (4.16)$$

$$\Phi[h_{12}(\mathbf{x})] = \begin{cases} (-x_6 + x_{13})^2 & x_6 x_{13} = 10 \\ 0 & \text{otherwise} \end{cases} \quad (4.17)$$

$$\Phi[h_{13}(\mathbf{x})] = \begin{cases} (-x_6 - x_{13} + 1)^2 & x_6 x_{13} = 11 \\ 0 & \text{otherwise} \end{cases} \quad (4.18)$$

$$\Phi[h_2(\mathbf{x})] = \begin{cases} (x_{10} + x_{12} - 1)^2 & x_{10} x_{12} = 00 \\ 0 & \text{otherwise} \end{cases} \quad (4.19)$$

$$\Phi[h_3(\mathbf{x})] = \begin{cases} (-x_{10} - x_{12} + 1)^2 & x_{10} x_{12} = 11 \\ 0 & \text{otherwise} \end{cases}$$

Reflecting the fact that the string containing the bits $x_6 x_{13} = 10$ is the most undesirable than any other combination of these specific bits, the coefficient weights can be assigned as $r_1 = r_3 = r_4 = r_5 = 4$ and $r_2 = 8$. Thus the problem reduces to *Minimize* $E'(\mathbf{x})$ where

$$E'(\mathbf{x}) = E(\mathbf{x}) + r_1 \Phi[h_{11}(\mathbf{x})] + r_2 \Phi[h_{12}(\mathbf{x})] + r_3 \Phi[h_{13}(\mathbf{x})] + r_4 \Phi[h_2(\mathbf{x})] + r_5 \Phi[h_3(\mathbf{x})]$$

which is an unconstrained energy function. A test pattern may be found for the given fault by finding the global minimum of this unconstrained energy function. It may be

noted that the global minimum of the $E'(\mathbf{x})$ shall be zero which may be obtained with the help of GAs.

4.4 Boolean Satisfiability Approach

In the test generation method using Boolean satisfiability (SAT) [66], a formula is constructed in conjunctive normal form (CNF), expressing *Boolean difference* between fault-free circuit and its faulty copy. An important advantage of this approach is that well studied SAT algorithms can be considered for satisfying the formula which provides a test pattern for a given fault. Instead of constructing the formula in CNF form, a *Boolean false function* is derived analogous to the *Boolean truth function*. To see the construction of the Boolean false function, consider a simple 2-input OR gate with a , b as input and c as an output of the gate. Since $c = a + b$, it is easy to see that the equation

$$F_{OR} = c \oplus (a + b) = 0 \quad (4.20)$$

is satisfied only by those values of a , b and c that satisfy the OR gate function. Here, $+$ and \oplus denote the logical OR and exclusive-OR (XOR) operation respectively. F_{OR} is referred as the Boolean false function for the OR gate and can be written as

$$F_{OR} = a\bar{c} + b\bar{c} + \bar{a}\bar{b}c \quad (4.21)$$

A similar formulation is possible for representing OR gate as an energy function where a , b and c were treated as arithmetic variables. The Boolean false functions of all the primitive gates can be computed in the same way. The false function for a digital circuit is the logical OR of the false functions for all gates in the circuit. As an example, the false function for the circuit shown in Figure 4.3 is

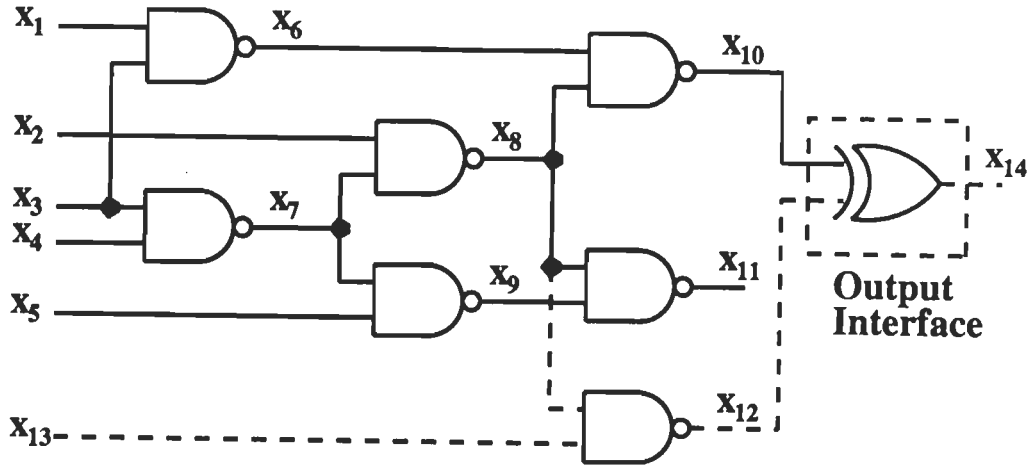


Figure 4.3: The ISCAS '85 circuit with XOR as an output interface

$$\begin{aligned}
 F_{CKT} = & \bar{x}_1\bar{x}_6 + \bar{x}_3\bar{x}_6 + x_1x_3x_6 + \bar{x}_3\bar{x}_7 + \bar{x}_4\bar{x}_7 + x_3x_4x_7 + \bar{x}_2\bar{x}_8 + \bar{x}_7\bar{x}_8 + x_2x_7x_8 + \\
 & \bar{x}_7\bar{x}_9 + \bar{x}_5\bar{x}_9 + x_7x_5x_9 + \bar{x}_6\bar{x}_{10} + \bar{x}_8\bar{x}_{10} + x_6x_8x_{10} + \bar{x}_8\bar{x}_{11} + \bar{x}_9\bar{x}_{11} + \\
 & x_8x_9x_{11} + \bar{x}_8\bar{x}_{12} + \bar{x}_{13}\bar{x}_{12} + x_8x_{13}x_{12} + \bar{x}_{10}x_{12}\bar{x}_{14} + x_{10}\bar{x}_{12}\bar{x}_{14} + \\
 & \bar{x}_{10}\bar{x}_{12}x_{14} + x_{10}x_{12}x_{14}
 \end{aligned} \tag{4.22}$$

It may be noted that the Boolean false function for the digital circuit consists of a set of binary and ternary relations. The problem of determining a signal assignment that minimizes this false function to zero is equivalent to the energy minimization problem which is already discussed. Chakradhar *et al.* [22] used the same Boolean false function and proposed a transitive closure algorithm for test generation. In the transitive closure based approach, binary relations are represented as an *implication graph* and standard graph-theoretic techniques are used to determine the transitive closure. Signal assignments and contradictions are determined by finding strongly connected components in the implication graph. Larrabee also used the same algorithm

proposed by Aspvall *et al.* and determined the signal assignments and contradictions for satisfying the Boolean truth function in CNF form. We have proposed a GA-based test generation method in which the Boolean false function is minimized using Genetic Algorithms and given as below.

4.5 Test Generation Algorithm

Figure 4.4 shows the flow chart of the test generation algorithm. The following are the test generation steps:

1. Derive Boolean false function for the gate-level circuit and reduce it by adding test generation constraints, *i.e.* substitute the fault-free and faulty value of the corresponding signal variables in order to activate the fault and constrain the output of the output interface to the logical value 1 for the successful fault effect propagation to at least one of the primary output (PO).
2. Construct the implication graph from the binary relations of the reduced false function obtained in Step 1.
3. Compute the transitive closure of the implication graph and identify the global dependencies. In case of contradiction, the fault under consideration is declared redundant and if the signals got fixed evaluate the Boolean false function to 0, a test pattern is found.
4. The signal assignments made in Step 3 may reduce the ternary relations of the false function to binary relations and if yes, then update the implication graph by adding these new binary relations and go to Step 3.

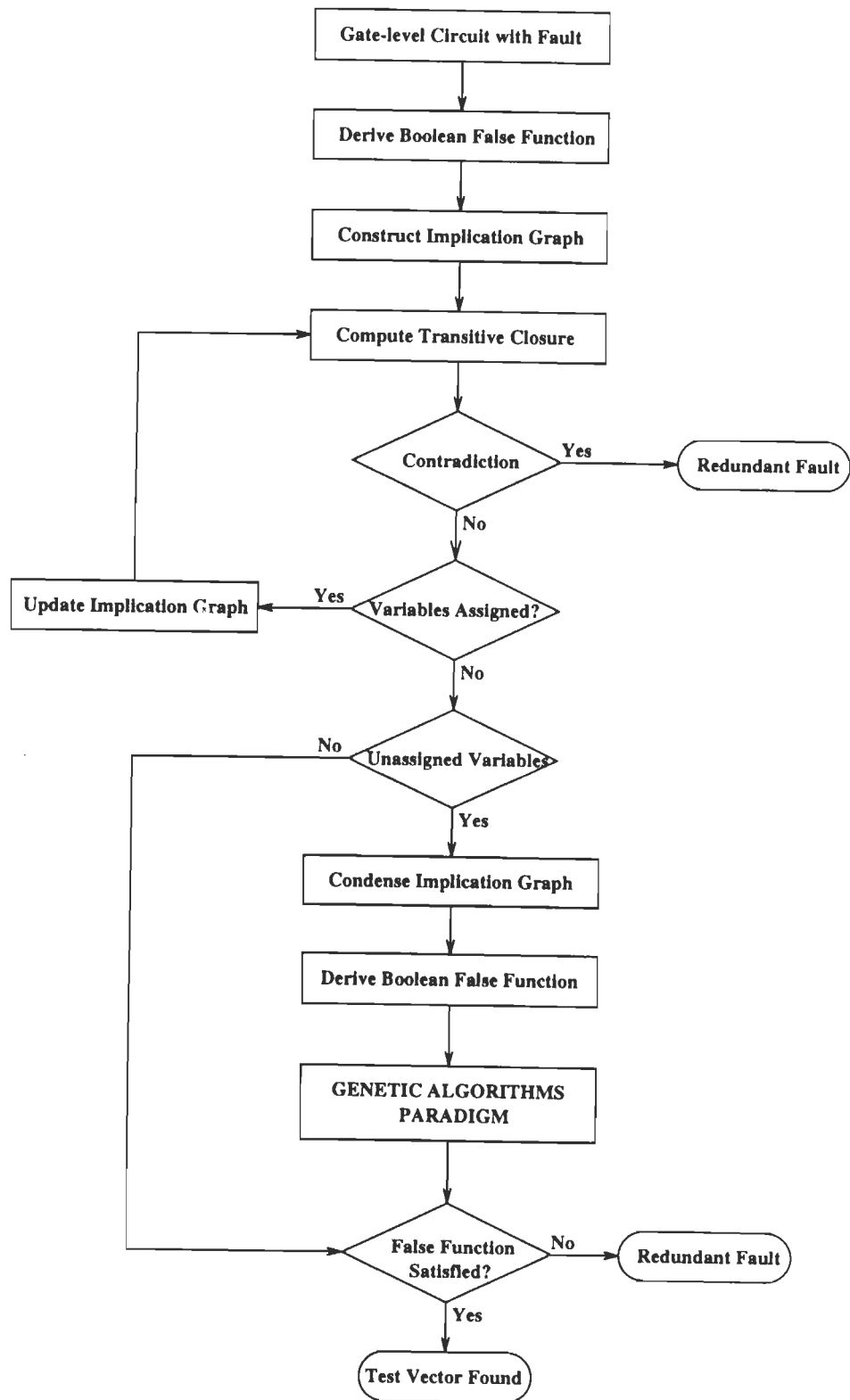


Figure 4.4: Genetic Algorithms Based Test Generation Method

5. For the remaining unassigned variables, reduce the Boolean false function and update its corresponding implication graph.
6. Construct the condensation graph of the implication graph obtained in Step 4 by finding the strongly connected components [7].
7. Derive objective function using the reduced false function as obtained in Step 4.
8. Apply SGA or AGA to find the global optimum solution for which the determined values of the signals (variables) evaluate the false function to 0.
9. If the global optimum is found in Step 8, the fault is detected and a test pattern is reported. Otherwise, fault is declared redundant.

To illustrate the test generation method using Boolean false function, consider the same example circuit as shown in Figure 4.3. The Boolean false function of the circuit is given by the Equation (4.22). In test generation, there are two basic steps: Fault activation and the fault-effect propagation to at least one of the primary output (PO). The fault activation is usually done by creating a change in the signal value at fault site. So to create a change at the fault site x_6 at which stuck-at-1 fault is considered, the fault-free value 0 should be assigned and simultaneously, its corresponding faulty line, *i.e.* x_{13} should be fixed to the value 1. In order to observe the fault effect at the PO(s) of the circuit, the fault-free value of the PO(s) should have complementary value at its corresponding faulty PO(s). Hence, one can conclude that in case of successful fault-effect propagation, the output the the XOR must have logical value 1. Therefore, the variable x_{14} must have a 1 value. Substituting these values of x_6 , x_{13} , and x_{14} , Equation (4.22) reduces to

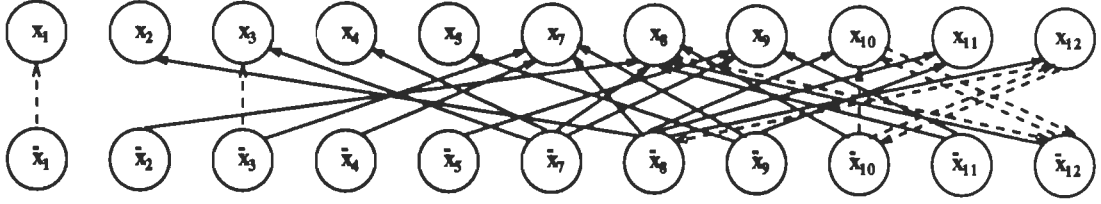
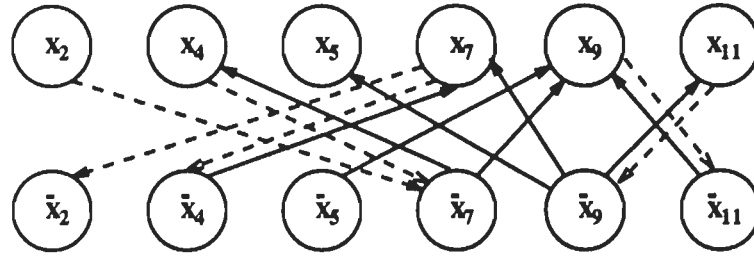


Figure 4.5: Implication graph of the example ISCAS circuit with $x_6 = 0$, $x_{13} = 0$, and $x_{14} = 1$.

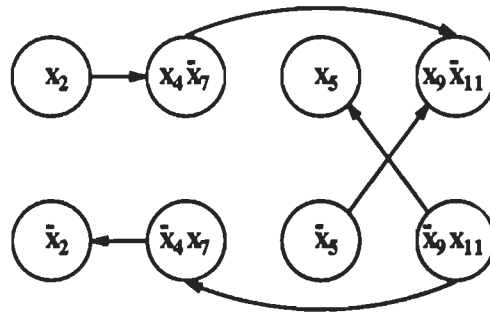
$$\begin{aligned}
 F_{CKT} = & \bar{x}_1 + \bar{x}_3 + \bar{x}_3\bar{x}_7 + \bar{x}_4\bar{x}_7 + x_3x_4x_7 + \bar{x}_2\bar{x}_8 + \bar{x}_7\bar{x}_8 + x_2x_7x_8 + \bar{x}_7\bar{x}_9 + \\
 & \bar{x}_5\bar{x}_9 + x_7x_5x_9 + \bar{x}_{10} + \bar{x}_8\bar{x}_{10} + \bar{x}_8x_{11} + \bar{x}_9\bar{x}_{11} + x_8x_9x_{11} + \bar{x}_8x_{12} + \\
 & x_8x_{12} + x_{10}\bar{x}_{12} + x_{10}x_{12}
 \end{aligned} \tag{4.23}$$

The implication graph from the binary relations of the reduced false function given by Equation 4.23 is shown in Figure 4.5. The dotted arrows represent the new relations derived from the binary and the ternary relations.

In order to determine contradiction and the fixation of the variables, the transitive closure of the implication graph is computed from which it can be seen that there is no contradiction found because there is no variable x for which x implies \bar{x} and \bar{x} implies x . However the variables x_1 , x_3 , and x_{10} can be assigned a fixed value of 1 because in the implication graph shown in Figure 4.5, \bar{x}_1 implies x_1 , \bar{x}_3 implies x_3 , and \bar{x}_{10} implies x_{10} . Now, it may be observed from the implication graph (Figure 4.5) that the signal variables x_{10} , \bar{x}_{12} and x_8 and its dual consisting of \bar{x}_{10} , x_{12} and \bar{x}_8 are strongly related. Therefore, the variables x_{10} , x_{12} and x_8 can be directly assigned the logical value 1, 0 and 1 respectively. By substituting these newly assigned values, the



(a)



(b)

Figure 4.6: (a) Reduced version of the implication graph shown in Figure 4.4 after fixing the variables, $x_1 = 1$, $x_3 = 1$, $x_{10} = 1$, $x_{12} = 0$, and $x_8 = 1$, (b) Condensed form of the implication graph (a)

F_{CKT} function finally reduces to

$$F_{CKT} = \bar{x}_4\bar{x}_7 + x_4x_7 + x_2x_7 + \bar{x}_7\bar{x}_9 + \bar{x}_5\bar{x}_9 + x_7x_5x_9 + \bar{x}_9x_{11} + x_9x_{11} \quad (4.24)$$

and its corresponding implication graph is shown in Figure 4.6(a). It may be noted in the implication graph that the remaining variables are x_2 , x_4 , x_5 , x_7 , x_9 and x_{11} and no fixed assignment can be made on any of these variables by using the transitive closure method. However, in the original transitive closure based test generation approach, a branch-and-bound process has been invoked for making further assignments on the remaining variables. But, the worst case complexity of the branch-and-bound method is $\mathcal{O}(2^n)$, where n is the problem size that is the number of variables in the current

example. Keeping this in view, we have proposed the further minimization of the reduced false function by using GAs.

In order to apply GAs, first the false function has to be represented as an objective function in fitness form. The derivation of the objective function is exactly done in the same way as mentioned for the energy minimization approach. The only difference is that in place of using the energy function, $E(\mathbf{x})$, the Boolean false function, F_{CKT} has to be used. As far as the mapping of the objective function is concerned, the largest value of the input coefficient, C_{max} can be taken as the total number of terms in the false function for the circuit. In this method, the string length defines the number of variables appeared in the objective function and their values has to be determined by using GAs. However, the string length can be minimized by finding the strongly connected components in the implication graph as the members of a strongly connected component can be simultaneously assigned the same logical value. This minimization of the string length is important as it will help in reducing the search space.

In order to find the strongly connected components the implication graph shown in Figure 4.6(a), a condensation form of the graph is constructed and shown in Figure 4.6(b). The string constitutes of the variable values of x_2 , $x_4\bar{x}_7$, x_5 and $x_9\bar{x}_{11}$, *i.e.* the string length of only four bits is required. Now initial population is generated by randomly generating 4-bit strings and GAs are applied to select a string for which the false function evaluates to 0. There may be more than one such strings which satisfy the false function. The string found by the test generation program is 0111 which provides the signal assignments: $x_2 = 0$, $x_4 = 1$, $x_5 = 1$ and $x_9 = 1$ and therefore, $x_7 = 0$ and $x_{11} = 0$. The signal values assigned on the primary input variables form the test pattern which is 10111 for the stuck-at-1 fault at the signal line x_6 . It may be noted that six variable string has been reduced to four variable string and thus a saving in search space has been obtained for the above example. This saving can be

very substantial for actual problem sizes for practical large circuits.

4.6 Automatic Test Pattern Generation CAD Tool

This section presents a CAD tool developed for Automatic Test Pattern Generation (ATPG) for the combinational digital circuits. The block schematic of the CAD tool is shown in Figure 4.7. Core of the tool is based on the genetic algorithms which finds the global optimum solution of the problem as the test generation problem is transformed into the search or optimization problem. The global optimum solution provides the test pattern for a fault to be detected in the circuit. The CAD tool for the ATPG problem mainly consists of the following four stages:

- Circuit Modeling
- Objective Function Evaluation
- Penalty Inclusion
- Global Optimization

The details of all these four stages are described in the following subsections.

4.6.1 Circuit Modeling

In this stage, the gate-level circuit is transformed into a test generation network by adding the image (copy) of the sub-circuit affected by the fault to the fault-free circuit and connecting their corresponding primary output(s) via an output interface. The output interface is an inverter when the faulty image has only the single primary output (PO) and the ports of the inverter are connected to the PO of the faulty copy

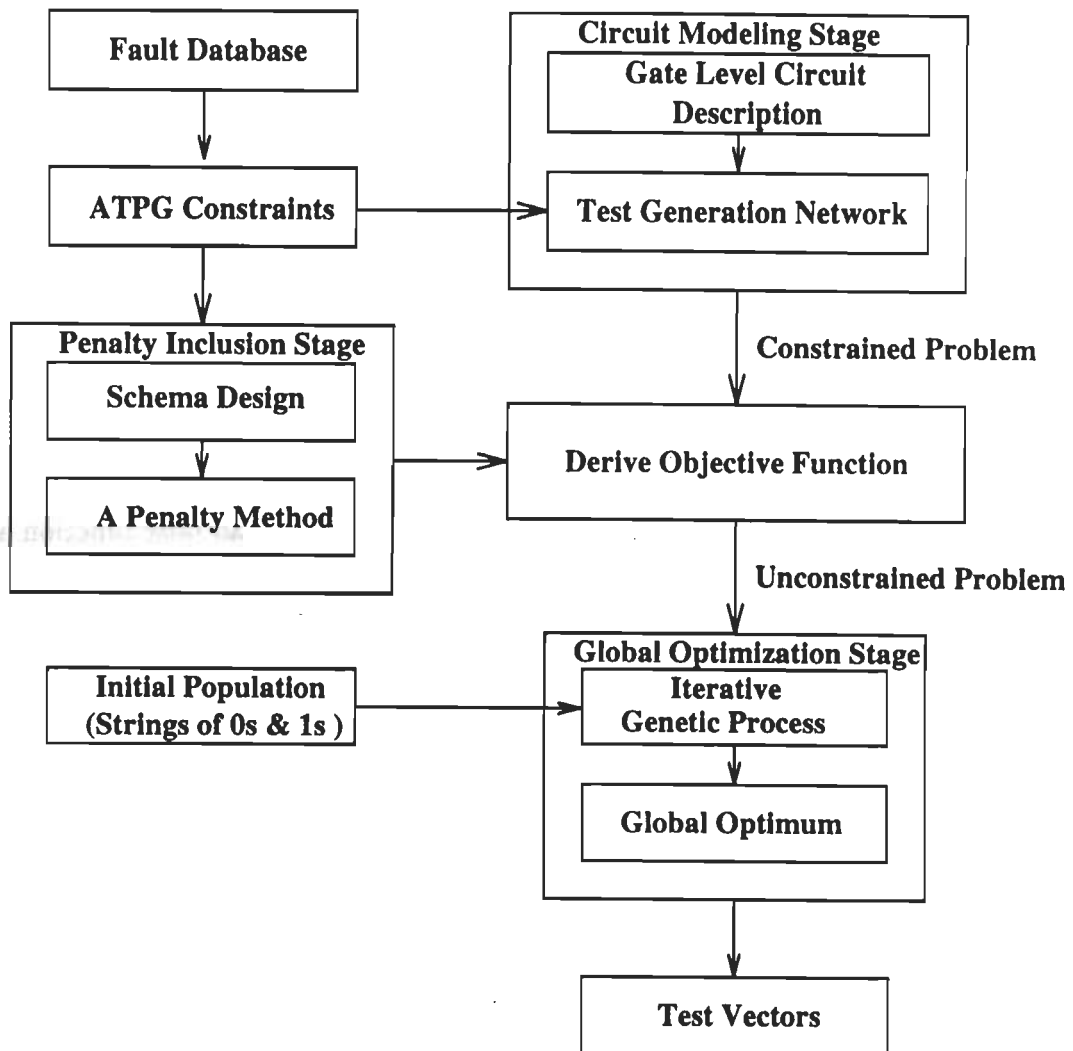


Figure 4.7: A CAD Tool for Automatic Test Pattern Generation.

to its corresponding PO of the fault-free circuit. This inverter may be replaced with an 2-input XOR gate, the inputs to which are the faulty and fault-free PO and the output is constrained to the logical value 1 for the fault detection. When there are n number of POs in the faulty circuit, the output interface may consist of n number of 2-input XOR gates and one n -input OR gate. Again the inputs to the XOR gate are the faulty PO and its corresponding fault-free PO and their outputs are connected as inputs to

the OR gate, the output of which is constrained to the logical 1 value. The resulting network, *i.e.* consisting of fault-free circuit, its faulty image and the output interface is termed as *test generation network*. The test generation network is further modeled using either of the two approaches previously described, *i.e.* energy minimization or the Boolean satisfiability approach.

In the energy minimization, the test generation network is modeled into a neural network using the unconventional digital circuit modeling technique [20] and the test generation problem is transformed into the minimization of the energy function derived for the test generation neural network. Similarly in the Boolean satisfiability approach, the test generation network is represented by the Boolean false function and the global minimum of this function will be 0 at which a test is generated.

4.6.2 Objective Function Evaluation

The test generation network constructed in the circuit modeling stage acts as the input to this stage. Once the test generation problem is transformed into the minimization problem for which the function is obtained using either of the energy minimization or the Boolean satisfiability approach, the aim is to find the global minimum of the problem. At this stage, the minimization function is transformed into an objective function so that GAs can be directly used this function to find its global optimum. This objective function must be mapped into a fitness form, *i.e.* in the maximization form, for which the mapping procedure is explained in the Section 4.3.

As mentioned earlier that the test generation problem is a constrained problem due to the test generation constraints, it is better first to transform this constrained problem into an unconstrained one before applying GAs. This transformation is done with the help of the penalty method based on schema design. The schema design also

plays an important role in the application of GAs. The objective function is modified at this stage by using the penalty method as described in Section 4.2.

4.6.3 Penalty Inclusion

This stage introduces a penalty using the penalty method in order to transform the constrained optimization problem into an unconstrained one. During this transformation, a penalty is introduced only in the case of the violation of certain ATPG constraints. These constraints are usually generated for a given fault from the data base for the circuit-under-test. Using the ATPG constraints, the schema (or schemata) has to be designed. The schema represents the type of strings not violating certain ATPG constraints – strings that are desirable and may lead to feasible solutions.

Once the schema is designed, the penalty functions and their corresponding penalty coefficients may be derived depending upon the the type of constraint violation. Example discussed in the Section 4.4 illustrates the procedure of defining penalty coefficients and functions. The penalty introduced must ensure the degrading of the function (*i.e.* decrease in the objective function value or increase in the energy function value) for the undesirable strings that lead to non-feasible solutions hence to be discouraged.

4.6.4 Global Optimization

This is the final stage of the CAD tool for the test pattern generation. At this stage, GAs are applied to find the global optimum solution for the unconstrained problem for which the objective function is derived in the previous stages. The GAs start with the initial random population of strings of 0 and 1s and maximize the fitness function by carrying out the iterative genetic process. This genetic process may be based on the

SGA or the AGA. The only difference between these two is that in SGA, the crossover probability, p_c and the mutation probability, p_m are to be fixed, and in the AGA, p_c and p_m vary adaptively in response to the fitness values of the strings. At the end of the genetic process, the global optimum is attained at which the energy function or the false function reduces to 0. This global optimum solution provides the desired test pattern for the given fault in the circuit.

The above CAD tool has been developed and reported in the next chapter.

Chapter 5

Implementation Details and Experimental Results

New quadratic 0-1 programming and Genetic Algorithms based ATPG methods as described in Chapter 3 and 4 respectively have been implemented for the development of an ATPG CAD tool. The tool consists of two C programs separately written for the energy minimization using the quadratic 0-1 programming technique and the GA-based minimization of the Boolean false function. The first ATPG program is called QUADTEST that is exclusively written for the new quadratic 0-1 programming algorithm which has been implemented for the energy minimization. In the second program called GATEST, the GA-based test generation algorithm has been implemented as per the flow-chart shown in Figure 4.4. A strictly modular approach has been followed in the CAD tool development which ensures that the separate modules of the software package could be compiled and debugged independently. This approach also facilitates program modification and expansion at later stages. Various abstract-level modules of the tool have been depicted in its block schematic shown in Figure 5.1. The CAD tool is mainly consists of the following three components:

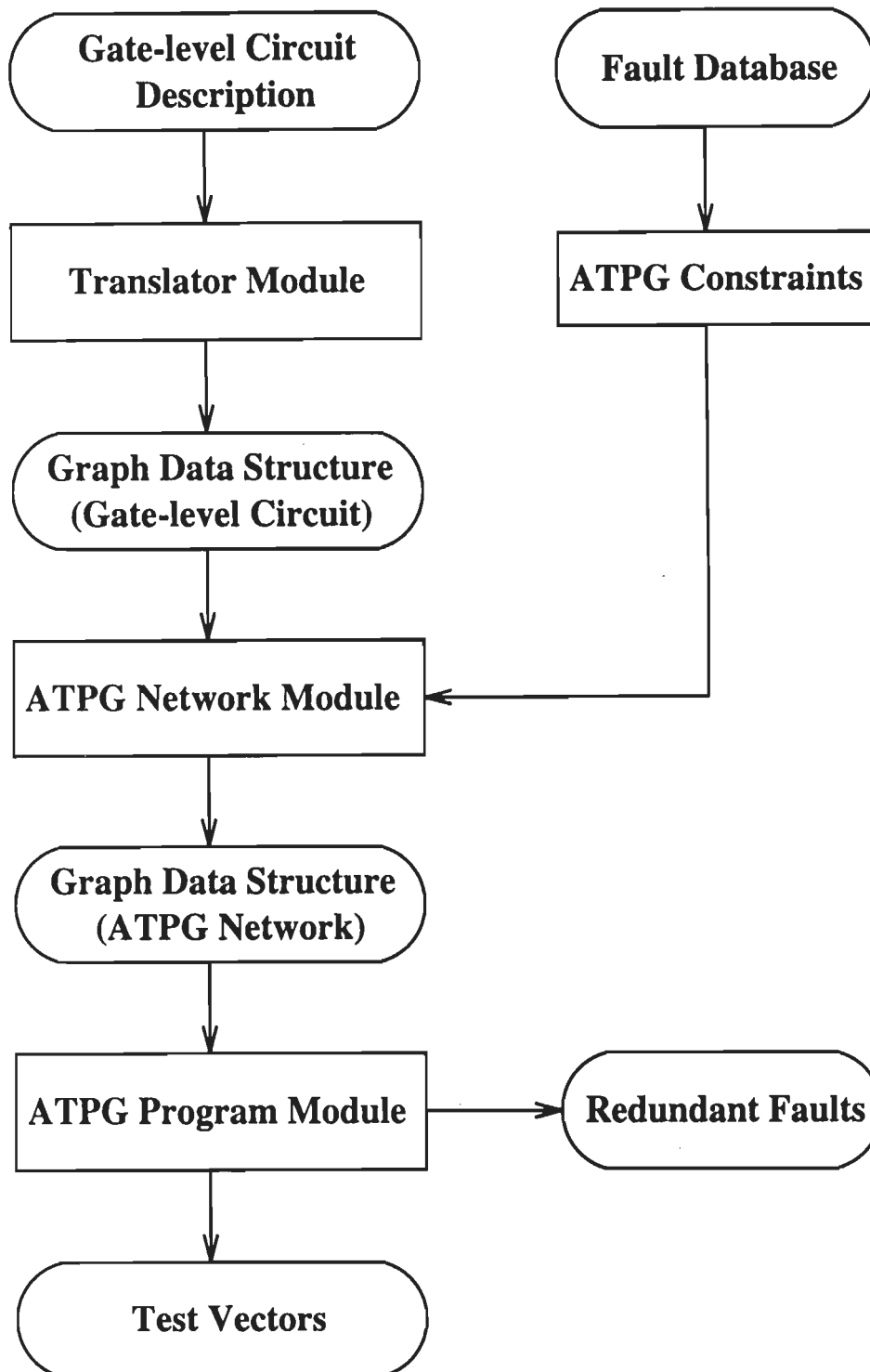


Figure 5.1: Abstract view of the CAD tool.

- Translator Module
- ATPG Network Module
- ATPG Program Module

5.1 Translator Module

The implementation of the proposed ATPG method based on quadratic 0-1 programming technique and genetic algorithms starts with the gate-level design description of a circuit and the fault data base. A *connectivity language* is required to describe the gate-level design of the circuit. This connectivity language should be either user-friendly or support ISCAS'85 netlist format. The ISCAS'85 benchmarks are ten combinational circuits provided at the 1985 International Symposium on Circuits and Systems. The ISCAS'85 netlist format was never formally documented, rather it was distributed on a magnetic tape and no formal language description such as Backus Naur Form (BNF) exists for this format. Due to non-availability of the netlist of the ISCAS'85 circuits, a user-friendly connectivity language has been designed for describing the gate-level circuit. The translator module consists of a language compiler developed using the UNIX operating system tools: YACC and LEX.

The output of the translator module is the structural model of the circuit represented in the form of a directed graph, $G(V, E)$, where V is a finite set of vertices, *i.e.* functional elements or logic gates and E is a finite set of edges which represent the interconnections between the components of the circuit. An edge $e \in E$ is an ordered pair (u, v) , where $u, v \in V$ and it represents a signal of the circuit *i.e.* input or output of a functional element. The set of vertices for the structural model of a circuit is given

by

$$V = V_{PI} \cup V_{PO} \cup V_{FE} \cup V_{FO}$$

where,

$$V_{PI} = \{\text{Primary Inputs}\}$$

$$V_{PO} = \{\text{Primary Outputs}\}$$

$$V_{FE} = \{\text{Functional Elements}\}$$

$$V_{FO} = \{\text{Fan Outs}\}$$

Since the complexity of VLSI circuits is normally of the order of tens of thousands of logic gates, the data structure used to build the circuit graph should be memory efficient. Considering this fact, an efficient data structure has been used to represent a circuit as a directed graph. In addition to this, the data structure can optimally be extended to both types of the proposed ATPG methods. The graph structure for the ISCAS'85 circuit with Stuck-at-1 fault as depicted in Figure 5.2 is shown in Figure 5.3 and the set of vertices are as follows:

$$V_{PI} = \{x_1, x_2, x_3, x_4, x_5\}$$

$$V_{PO} = \{x_{10}, x_{11}\}$$

$$V_{FE} = \{x_6, x_7, x_8, x_9, x_{10}, x_{11}\}$$

$$V_{FO} = \{FO_1, FO_2, FO_3\}$$

After building the graph data structure, an ATPG constraint network is to be constructed before the actual implementation of the ATPG program module.

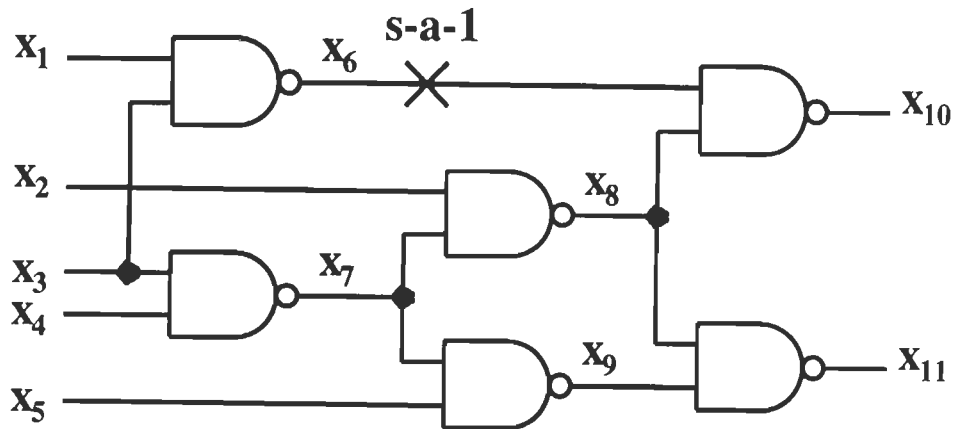
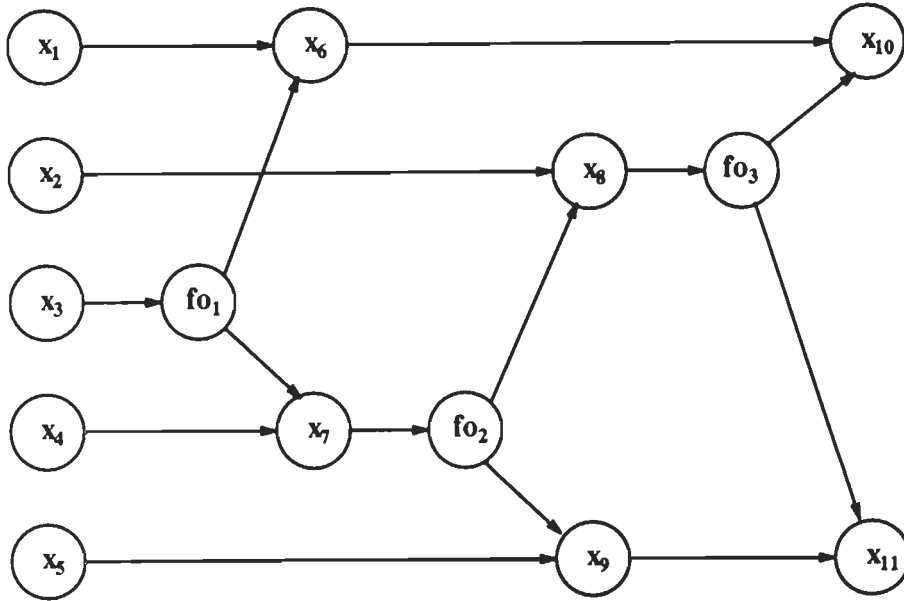


Figure 5.2: c17.isc: An ISCAS'85 circuit

5.2 ATPG Network Module

Once the circuit-under-test is represented in terms of graph with the help of data structure, the fault data base is accessed in order to fetch the information about the faults to be considered. For each fault in the fault data base, the ATPG constraint network is constructed in two steps. First, the circuit graph is upgraded by concatenating the fault affected sub-circuit graph. To accomplish this, a standard graph traversal algorithm has been used since one has to traverse the directed graph from the edge corresponding to the fault site of the fault-free circuit to the primary output node(s). Second, if there exists a single PO in the sub-circuit affected by the fault, a single XOR node will be added to the upgraded graph such that the input edges of the XOR node will be connected to the faulty and fault-free PO nodes. The output node of this XOR will be constrained to the value 1 in order to ensure the fault detection for which the PO of the faulty circuit must be different from its corresponding fault-free PO. In case there appears more than one PO in the fault-affected sub-circuit graph, a subgraph consisting of n two-input XOR gates and one n -input OR gate will be added



(a)

Figure 5.3: Graph representation for the *c17.isc* circuit

to the resulting graph, where n is the number of POs in the fault-affected sub-graph. In the subgraph, the inputs to each XOR gate will correspond to each pair of fault-free and faulty PO and the outputs of these XOR gates will be fed to the inputs of the OR gate. Since for fault detection, at least one PO of the faulty sub-circuit must be different from its corresponding fault-free PO, the output of the OR gate will be constrained to a value 1. The final graph obtained will be the ATPG constraint network represented in the graph data structure. The ATPG constraint network of the ISCAS'85 circuit (*c17.isc*) for the stuck-at-1 fault on line x_6 is shown in Figure 5.4 and its graph structure is shown in Figure 5.5. The set of vertices in the graph structure of the ATPG constraint network are as given below:

$$V_{PI} = \{x_1, x_2, x_3, x_4, x_5, x_{13}\}$$

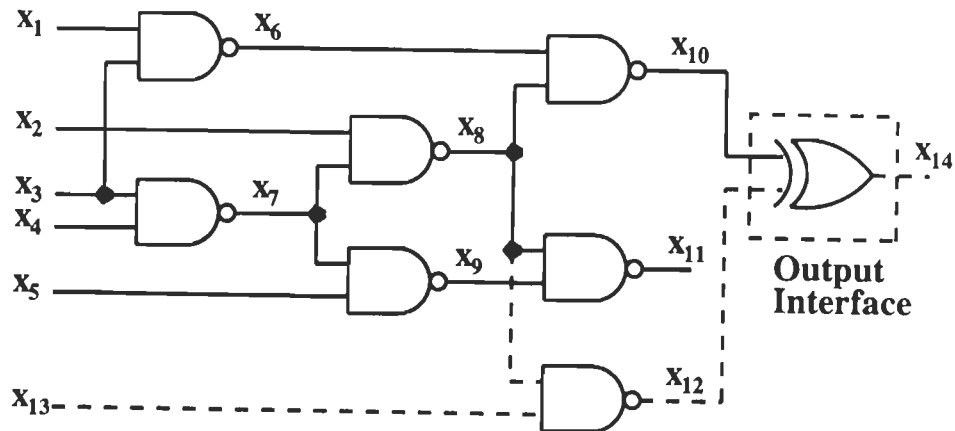


Figure 5.4: ATPG constraint network of the ISCAS circuit *c17.isc* for stuck-at-1 fault on line x_6

$$V_{PO} = \{x_{14}\}$$

$$V_{FE} = \{x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{14}, x_{12}\}$$

$$V_{FO} = \{FO_1, FO_2, FO_3\}$$

5.3 ATPG Program Module

The ATPG program module consists of two separately written C programs: QUADTEST is for new quadratic 0-1 programming algorithm based ATPG method and the GATEST for GA-based test generation method. Both of these methods start with the graph data structure for the ATPG constraint network. Although both the methods derive either an energy function or a Boolean false function which is to be minimized in order to find a test, the approach to derive these functions and their minimization is entirely different. The implementation details of each method with experimental results obtained for some example combinational circuits are described in the following

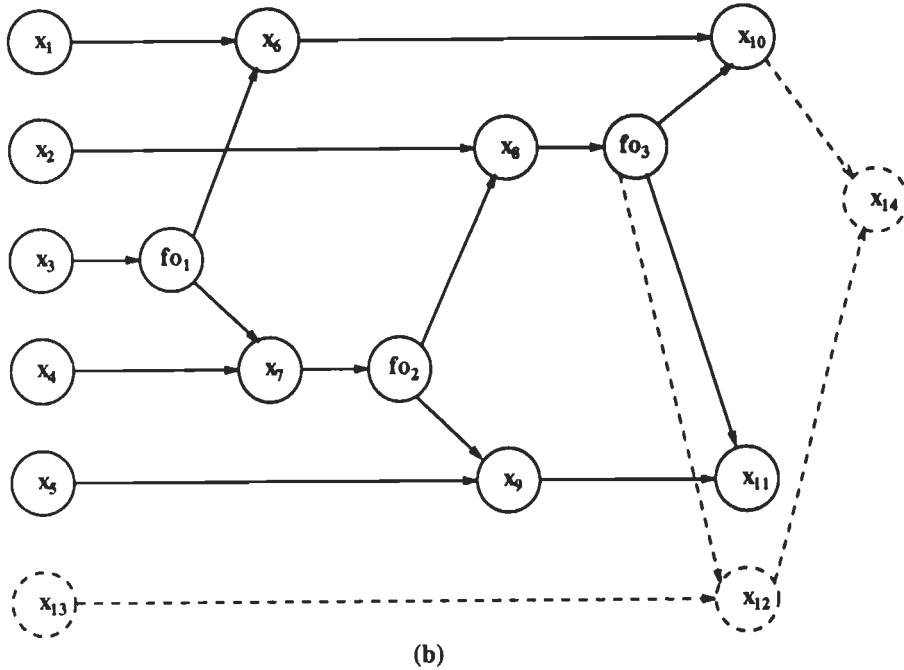


Figure 5.5: Graph representation for the ATPG constraint network of *c17.isc* circuit

subsections:

5.3.1 QUADTEST

The proposed ATPG method based on the new quadratic 0-1 programming technique has been implemented in a prototype C language program called QUADTEST and run on a Sun Sparc 10 UNIX workstation to generate tests for example combinational circuits. In QUADTEST, first of all, a neural graph is constructed for the ATPG constraint network by using the same digital circuit modeling technique as described in the subsection 2.4.2. In order to accomplish this, a neural compiler is used to transform the ATPG constraint network in to the ATPG neural net. This neural compiler uses the neural database which consists of neural models for all the basic gates. Since the neural net is also a kind of graph, an enhanced version of the graph data structure

is used to represent it. Finally, in QUADTEST, an energy function is derived for the ATPG neural net that is resulted from the basic energy minimization formulation of the test generation problem. In order to find a test for a given fault, this energy function is to be minimized to zero. The QUADTEST minimizes the energy function which is of the type of pseudo-Boolean quadratic function.

The QUADTEST is purely deterministic because Random Test Generation is not done at any stage of the ATPG process. The energy function, obtained from the previous stages of the ATPG system, is minimized strictly using the new quadratic 0-1 programming technique. Efficient data structure is used to provide all the necessary information required during the test generation process. The algorithm provides the test vector for a fault under consideration if it is testable. After every test vector, the energy function for each of the undetected faults is evaluated and as soon as energy evaluates to 0 for a fault, the same is deleted from the set of undetected faults. This procedure is repeated for all the undetected faults. This way not only the fault set is get reduced but an optimal test set is obtained.

The QUADTEST not only generates the test patterns for all the testable faults but also identifies redundancies in the circuit. Test patterns have been generated for some example combinational circuits which confirms the feasibility and efficiency of the QUADTEST. The results obtained by the proposed methods for the c17.isc (ISCAS'85 circuit), the Schneider's circuit [87], the ECAT (Error Correction and Translation) circuit, and the SN5483A four-bit carry look-ahead binary full adder are tabulated here. Average ATPG time per fault for the example circuits is shown in Table 5.1. The experimental ATPG system based on QUADTEST does not have a fault simulator and no circuit partitioning and fault collapsing techniques are used. The average CPU time per fault obtained by the proposed method is also compared with the results reported earlier. The comparative results given in Table 5.1 show the effectiveness of

Table 5.1: Experimental Results: QUADTEST

<i>Circuit</i>	<i>c17.isc</i>	<i>Schneider</i>	<i>ECAT</i>	<i>SN5483A</i>
Number of Signals	17	28	23	104
Number of Inputs	5	4	6	9
Number of Outputs	2	3	1	5
Number of gates	6	8	9	36
Total Faults Considered	34	56	46	208
Redundant Faults	0	2	8	0
Faults Detected	34	54	38	208
Fault Coverage	100%	100%	100%	100%
New Quadratic 0-1 Programming:				
Average CPU Time per Fault (sec.)	0.01	0.02	0.04	0.24
Reported in the Literature:				
Average CPU Time per Fault (sec.)	–	0.12	0.43	3.3

the proposed method.

5.3.2 GATEST

Genetic Algorithms based test generation method has been implemented in another prototype C program called GATEST. This prototype which generates a test for a given fault or classifies it as redundant if it is undetectable, executes the following steps:

1. First of all, the Boolean false function is constructed for the ATPG constraint network with fault. In order to activate the fault, the values 1(0) and 0(1) are to

be assigned to signals at the fault site in the fault-free and faulty portions of the circuit, respectively, for a s-a-0(1) fault. For successful fault effect propagation to at least one of the PO, the output of the ATPG network is to be constrained to a logical value 1. The false function will get reduced by substituting these fixed signal values.

2. An implication graph is constructed from the binary relations of the reduced false function obtained in Step 1.
3. The global dependencies are derived from the transitive closure of the implication graph which is computed by using an efficient computational algorithm described later in this Section. If a contradiction occurs, the fault under consideration is declared redundant. In case all the signal variables are get assigned and the false function evaluates to 0, then test pattern is found. Otherwise, the newly assigned variables may reduce some of the ternary relations of the false function to binary relations. The implication graph may be updated for these additional binary relations. Again, transitive closure is computed for this updated implication graph and this process continues until no more fixation is made or any contradiction occurs.
4. In case the test pattern is not found and fault is not declared redundant, a reduced version of the false function will be obtained for minimization as no further assignments can be made using the transitive closure computation. The condensed form of the implication graph corresponding to the reduced false function is derived by finding its strongly connected components.
5. Genetic Algorithms are applied to find the minimum of the reduced false function which will be 0 at its global minimum. To guide GA-based search, objective function in fitness form is derived by using the reduced false function. If the global

minimum is found, the fault is detected and a test pattern is found. Otherwise, the fault is declared redundant.

The transitive closure computation and the false function minimization using GAs are the two major steps in GATEST and therefore, require special attention. The following subsections are devoted to describe each in detail.

Efficient Transitive Closure Algorithm

The worst-case space and time complexities of conventional transitive closure algorithms [7] for the graph with n number of vertices are $\mathcal{O}(n^2)$ and $\mathcal{O}(n^3)$ respectively. The time complexity could be reduced to $\mathcal{O}(n^{2.49})$ by exploiting the equivalence between matrix multiplication and transitive closure and using fast matrix multiplication methods. An efficient algorithm has been implemented for computation of the transitive closure by exploiting the special structure of the implication graphs. The implication graphs corresponding to the false function are sparse which means that the graph contains a very small number of edges compared to fully connected graph and also have the *duality* property [11], *i.e.* if the graph contains an arc from x to y then it will also contain an arc from \bar{y} to \bar{x} . Using this duality property of the implication graphs, *strongly connected components* [7] are computed using a simple depth-first search, where a strongly connected component is a set of vertices that are reachable from each other. This transitive closure algorithm appears to run in linear expected time complexity and has $\mathcal{O}(n)$ space complexity.

In the context of test generation, the transitive closure computation means that only the necessary logical conclusions like fixation of the variables, contradictions *etc.* are derived by finding the strongly connected components and *condensation* of the implication graph [7]. The procedure for deriving logical conclusions has been

implemented only in three steps as given below.

1. Compute strongly connected components of the implication graph which correspond to the false function. All contradictions and identifications are determined from the strongly connected components.
2. Construct the condensation graph.
3. Determine the transitive closure of the condensation graph. All fixations are derived from the transitive closure.

Minimization of Boolean False Function Using Genetic Algorithms

The reduced false function that can not be further pruned by the transitive closure method used to be further minimized by the branch-and-bound procedure which led to a too costly solution to find its minimum. We apply Genetic Algorithms to minimize the reduced false function by implementing the following steps:

1. The implementation of a sequential GA starts with the initial population of binary strings which represent the potential solutions and the initial population is generated by using a random number generator. The number of signal variables in the false function decides the length of the string which is termed as chromosome length (*lchrom*). The population size (*pop_size*) is the number of such strings in a given population.
2. The next step is reproduction which evaluates and selects pairs of strings for mating according to their relative strengths. The individual string is evaluated by a fitness function, *i.e.* an objective function in fitness form. The fitness function is obtained by subtracting the false function, say $F(\mathbf{x})$ from its largest value. Let

C_{max} be the largest value of the false function. Then the fitness function can be written as

$$\text{fitness function} = C_{max} - \text{false function} \quad (5.1)$$

where, C_{max} can be taken as the total number of terms in the false function.

In selection, strings are selected by using a weighted roulette wheel method so that strings with higher fitness values have a higher number of offspring in the succeeding generation. These strings are then entered into a mating pool, *i.e.* a tentative new population.

3. After reproduction, next comes the crossover which determines whether crossover is to occur on a pair of strings by using a flip function: tossing a biased coin (with probability p_{cross}). If the result is head (true), the strings are swapped and the *crossover_point* is determined by a random number generator. However if the result is tail (false), the strings are simply copied. A new population will be obtained as a result of this crossover step.
4. After crossover, the mutation operator is applied to the new population which may be an alteration of a random bit in a given string. The mutation function uses the biased coin toss (flip) with probability $p_{mutation}$ to determine whether to change a bit or not.
5. Finally, termination criterion is included in which if the false function reduces to 0 during evaluation of the strings, the fault is detected and the values of the primary input signal variables will form the required test pattern. Otherwise, the fault under consideration is declared redundant.

In the initial phase of GATEST, the transitive closure of the implication graph is computed which contains global pairwise logical relationships among all signal vari-

ables. Signal dependencies like fixation are determined from the transitive closure. Other dependencies may result in conflicts, and, hence able to identify redundancies in the circuit. The key feature of this particular step is that the dependencies derived from the transitive closure are used to reduce ternary relations of the false function to binary relations which dynamically update the transitive closure. For GA-based minimization of the false function, values of the different input parameters like *pop-size*, *pcross*, *pmutation* are required as they play a major role in finding the optimum solution efficiently. Although the crossover and mutation probabilities are adaptively changed in the GATEST implementation, initially the following two sets of control parameters have been employed.

- *pop-size*: 100, *pcross*: 0.67, and *pmutation*: 0.001.
- *pop-size*: 32, *pcross*: 0.87, and *pmutation*: 0.01.

The selection of these control parameters depends on the number of variables in the false function. If the variables are small in number, then small population size parameters are used and if large number of variables are there, then large population size parameters are taken as input to the GA-process. The chromosome length, *lchrom* is just the total number of variables in the the false function. In GATEST, the crossover and mutation probabilities, *i.e.* *pcross* and *pmutation* respectively, are modified adaptively. These probabilities are computed using the following expressions:

$$pcross = k_1(f_{max} - f')/(f_{max} - \bar{f}), f' \geq \bar{f} \quad (5.2)$$

$$= k_3, f' < \bar{f} \quad (5.3)$$

$$pmutation = k_2(f_{max} - f)/(f_{max} - \bar{f}), f \geq \bar{f} \quad (5.4)$$

$$= k_4, f < \bar{f} \quad (5.5)$$

where,

\bar{f} : average fitness value of the population,

f_{max} : maximum fitness value of the population,

f : fitness value of the solution, and

f' : larger fitness values of the solutions to be crossed.

and $k_1, k_2, k_3, k_4 \leq 1.0$. The coefficients k_1, k_2, k_3 , and k_4 values are taken to be 1.0, 0.5, 1.0 and 0.5 respectively, because the moderately large values of p_{cross} ($0.5 < p_{cross} < 1.0$) and the small values of $p_{mutation}$ ($0.001 < p_{mutation} < 0.05$) are essential for the successful working of GAs.

GATEST has been implemented in C to run on a SUN SPARC 10 UNIX workstation. It generates test patterns for all the detectable faults and also identifies redundant faults in the circuit. In the case of redundant faults, redundancies are identified either in the phase of transitive closure computation or during minimization of the false function using GAs. So whenever, any redundancy is reported in the transitive closure computation phase, GATEST simply exits by declaring the fault as redundant. In case no redundancy is reported during the transitive closure computation, GAs will be run for the maximum number of generations. In case GAs are also unable to detect the fault then it will be known only after the maximum number of generations. Such cases, although less in number, increase the average CPU time per fault for ATPG and can not be avoided. The performance of the GATEST has been tested by generating test patterns for the same set of example combinational circuits as considered for QUADTEST. The performance of both the ATPG program has been compared and found that for large circuits GATEST performs better than QUADTEST as it can be seen from the average CPU time per fault for the 4-bit carry look-ahead binary full adder (SN5483A). The QUADTEST performance is found better for the smaller

Table 5.2: Experimental Results: GATEST

<i>Circuit</i>	<i>c17.isc</i>	<i>Schneider</i>	<i>ECAT</i>	<i>SN5483A</i>
Number of Signals	17	28	23	104
Number of Inputs	5	4	6	9
Number of Outputs	2	3	1	5
Number of gates	6	8	9	36
Total Faults Considered	34	56	46	208
Redundant Faults	0	2	8	0
Faults Detected	34	54	38	208
Fault Coverage	100%	100%	100%	100%
GA-Based Approach:				
Average CPU Time per Fault (sec.)	0.02	0.04	0.07	0.20
Reported in the Literature:				
Average CPU Time per Fault (sec.)	–	0.12	0.43	3.3

circuits. The test generation results in terms of average CPU time per fault, number of faults detected and found redundant are given in Table 5.2. Like QUADTEST, the GATEST also does not have any fault simulator and no circuit partitioning and fault collapsing techniques are used in the ATPG program. The experimental results obtained by GATEST shown in Table 5.2 are compared with the results reported in the literature. These results show the effectiveness of the proposed GA-based method.

Although both the ATPG programs QUADTEST and GATEST have been successfully tested by generating test patterns for the example combinational circuits, but their performance could not be tested on the ISCAS'85 benchmark circuits because

the required netlists of these circuits are not available. However, the feasibility of the proposed ATPG methods and their effectiveness have been shown by comparing the results obtained for the example circuits to the published results. Though the approaches in the development of these programs have been finalized keeping in view of their future scope of parallelization. But, since the programs have been implemented on serial machines, efforts have been made to make them faster to run on these machines. Furthermore, approaches followed in both the programs share a common goal that is minimization of a function, where the function is either energy function or Boolean false function. Hence, further discussions are required to elaborate the ATPG program development.

The new quadratic 0-1 programming technique has been investigated due to the fact that the energy function derived from the test generation problem formulation has the special structure and is of the form of pseudo-Boolean quadratic function. This special structure has been exploited in QUADTEST and the minimization process has been accelerated by incorporating the heuristics. However, the minimization of the energy function can also be done by using the GA-based approach in which the objective function is derived using the energy function in place of the false function. As far as the mapping of the objective function resulted from the energy function is concerned, the value of the input coefficient C_{max} is required which is difficult to know a priori for the energy function and therefore, it may be taken as the largest value of the energy function observed in a process or in the last k -generations. But, in the minimization of the false function using GAs, the total number of terms in the function is taken as the C_{max} value. Furthermore, the GA-based approach for test generation problem is accelerated by incorporating the transitive closure method in the initial phase of the process. The transitive closure computation helps in reducing the false function by making the fixed assignments on the signal variables and identifies early

conflicts. Since the implementation of both the ATPG programs has been focussed to run on serial machines, the false function minimization using the GA-based test generation approach is preferred over the energy function minimization.

Chapter 6

Conclusion and Future Scope of Work

6.1 Conclusion

Test pattern generation problem for VLSI circuits has been studied for which a number of combinational as well as sequential ATPG algorithms have been developed in the past. Most of these algorithms have been reviewed in the thesis. Although investigations of sequential ATPG algorithms is an important area of research, the thesis is mainly focussed on the development of combinational ATPG algorithms keeping in view of their vast applications as discussed in the Chapter 2. The ATPG algorithms reported in the literature are found mostly serial in nature which means that they are developed to run on the conventional single processor machines and the entire *bag of tricks* have been used to speed up these algorithms. Despite putting lots of efforts, the gains achieved through these developments have not kept pace with the increasing size of VLSI circuits resulting in huge computation times for ATPG. With the

availability of parallel machines and distributed network of idle workstations in most of the VLSI-CAD environments, it has become possible to harness the computational power available for solving the compute-intensive ATPG task. Various parallelization techniques have been reported in the literature but there are some drawbacks with every technique as discussed again in Chapter 2 and it has been recommended that altogether new techniques have to be investigated for solving the compute-intensive VLSI test problem. Recently two different approaches have been developed for solving such problem so that they can be easily extended to run on massively parallel and distributed computing platforms.

The work carried out in this thesis has been in the same direction of the development of new ATPG methods easily extendable to run in a parallel/ distributed computing paradigm. The proposed methods are based on the recently developed approaches in which the ATPG problem is formulated either as an optimization or a Boolean satisfiability approach. Both the approaches are radically different from the conventional methods of generating test patterns for circuits from their gate level descriptions. Since, in the optimization based approach, the ATPG problem is transformed into the minimization of the energy function, which is of the form of pseudo-Boolean quadratic function, a new quadratic 0-1 programming technique has been proposed in order to find the global minimum of the energy function that in turn provides the test pattern for a given fault.

Once the ATPG problem is formulated as an optimization problem, a whole suite of optimization algorithms can be applied to find a solution. Genetic Algorithms (GA), being most effective and more efficient than other traditional optimization algorithms, have been studied and found very much suitable to solve the ATPG problem. It is also noted that GAs can be the basis for ATPG since they are robust and have inherent amenability to be processed in parallel. Considering all these advantages of

genetic algorithms, new GA-based test generation methods have been proposed and described in this thesis. In the GA-based test generation method, schema design and a penalty method has been described, which transforms the constrained ATPG problem into an unconstrained one by modifying an objective function. The objective function is essentially required to guide the GA-based search. This function is derived by using the energy function for the optimization based ATPG approach. In the Boolean satisfiability method, the objective function is based on the Boolean false function instead of a truth function. Finally, a mapping has been done to transform the objective function into fitness form.

A prototype ATPG system based on the proposed algorithms has been developed which consists of two programs: QUADTEST and GATEST. Both the programs have been demonstrated by generating test patterns for single stuck-at faults in several practical combinational circuits. The new quadratic 0-1 programming technique has shown good results for some example circuits and its performance has to be tested out for large practical circuits. However, this technique may become too time consuming for the large circuits due to its branch-and-bound process and the large number of variables in the energy function. This fact can be easily observed from the results obtained for the comparatively bigger example circuit of 4-bit carry look-ahead binary full adder (SN5483A) using the QUADTEST (Table 5.1). On the other hand, minimization of the Boolean false function by the GA-based method has been found very effective as it can be seen from the results obtained for the same example circuit with fairly a large set of faults (Table 5.2). This is because of the fact that the GA-based method does not incorporate branch-and-bound process at any stage and both the transitive closure computation and the false function minimization using GAs has been used in an efficient manner. The efficiency of the GA-based method has been shown by running the ATPG program, GATEST, for the same example practical circuits as of the

QUADTEST. The results obtained by this new method are also compared to the results reported in the literature. It can be observed from Table 5.1 and 5.2 that the average CPU time per fault for example circuits obtained by QUADTEST and GATEST is relatively less as compared to directed search technique augmented by probabilistic relaxation using the similar optimization based ATPG method by Chakradhar *et al.* [20]. However, these results have not been compared to the results obtained from other existing methods which use Random Test Generation (RTG) in their initial phase of the ATPG process due to several reasons. One important reason is that RTG produces a longer tests set which costs more to apply because it increases the testing time and memory requirements of the tester. Another significant reason is that fault simulation will be more expensive for longer tests set. Due to these reasons RTG could not be preferred over the deterministic test generation techniques which are usually much more expensive in terms of computational time but produce shorter and high quality tests. The comparative study shows the effectiveness of the GA-based method for solving the ATPG problem.

The GA-based test generation method can be easily parallelized by parallelizing its transitive closure computation and GA-based false function minimization steps. The computation of transitive closure can be accelerated through parallel processing since it belongs to class NC that means a hierarchy of problems solvable by deterministic algorithms in polylog time using a number of processors which are polynomial bound. The false function minimization using GAs can be parallelized by assigning the iterative genetic process on separate processors to be executed in parallel on a network prototype. In this case more than one processor can be simultaneously assigned a job of the false function minimization in order to obtain the required test patterns for a given set of faults.

6.2 Future Scope of Work

The future scope of the work carried out in this thesis will be on parallelization of the proposed ATPG methods. The parallelization of these methods, though not attempted here, may provide good speedup for solving the ATPG problem on parallel machines with a large number of processors. Recently, mathematical techniques have been reported in the literature for solving the quadratic 0-1 programming problem on hypercube architectures but they have not yet applied to solve the ATPG problem. The work carried out on the GA-based CAD tool development for the ATPG problem can be easily updated for an efficient implementation on the parallel/distributed computing paradigms.

It is observed from the results reported in the thesis that the proposed quadratic 0-1 programming technique and GA-based methods have been found very effective for solving a difficult problem of test pattern generation for combinational logic circuits. Since the ATPG problem has been formulated as an optimization or a Boolean satisfiability problem, other compute-intensive real world problems can also be solved using the same methodology of their reformulation into an optimization or a Boolean satisfiability problem. For example, the VLSI layout problem, the Boolean satisfiability problem, the traveling salesman problem and many other combinatorial optimization problems can be solved via quadratic 0-1 programming. Most recently, quadratic 0-1 programming technique has been applied for synthesis of application specific data paths in VLSI-CAD [39]. Hence, the new technique proposed in the thesis to find the minimum of $E(\mathbf{x}) = \mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{c}^T \mathbf{x}$ with $\mathbf{x} \in \{0, 1\}^n$ is directly become applicable to these problems. Solving graph problems is another important problem domain of various engineering problem for which the proposed quadratic 0-1 programming technique finds its application. It may be observed that the transformation of graph into logic

is possible through an expression which is of the form of an energy function and thus can be solved by using the same quadratic 0-1 programming technique.

The GA-based methods can also be applied to solve the above mentioned real world problems. As an illustration, the GA-based ATPG method finds the global minimum of the Boolean false function which is zero and the solution obtained for the problem is in terms of the values for the signal variables involved in the function. Since this false function can be transformed into its corresponding Boolean truth function which is usually represented in the conjunctive normal form (CNF), the signal values of the variables determined by GA-based method also satisfy the CNF formula. So the GA-based method can be applied to the Boolean satisfiability problem for which choosing an objective function is far from obvious. Other real world problems may also be solved using the similar GA-based method in which the problem is first transformed into an optimization or a search problem and then GAs can be applied to find the solutions.

References

- [1] Abramovici M., Menon P.R., and Miller D.T., "Critical Path Tracing: An Alternative to Fault Simulation," *IEEE Design & Test of Computers*, Vol. 1, No. 1, pp. 83-93, February 1984.
- [2] Abramovici M., Kulikowski J.J., Menon P.R., and Miller D.T., "SMART and FAST: Test Generation for VLSI Scan-Design Circuits," *IEEE Design & Test of Computers*, Vol. 3, No. 4, pp. 43-54, August 1986.
- [3] Abramovici M., Breuer M. A., and Friedman A.D., *Digital Systems Testing and Testable Design*, Computer Science Press, New York, NY, 1990.
- [4] Agrawal V.D., "When to Use Random Testing," *IEEE Transactions on Computer-Aided Design*, Vol. C-27, No. 11, pp. 1054-1055, November 1978.
- [5] Agrawal V.D., and Seth S.C., *Test Generation for VLSI Chips*, IEEE Computer Society Press, Los Alamitos, CA, 1988.
- [6] Agrawal V.D., Cheng K.T., and Agrawal P., "A Directed-Search Method for Test Generation Using a Concurrent Simulator," *IEEE Transactions on Computer-Aided Design*, Vol. 8, No. 2, pp. 131-138, February 1989.
- [7] Aho A.V., Hopcroft J.E., and Ullman J.D., *The Design and Analysis of Computer Algorithms*, Addison-Wesley Publishing Company, Reading, MA, 1974.

- [8] Armstrong D.B., "On Finding a Nearly Minimal Set of Fault Detection Tests for Combinatorial Nets," *IEEE Transactions on Electronic Computers*, Vol. EC-15, No. 2, pp. 63-73, February 1966.
- [9] Aarts E.H.L., and Laarhoven P.J.M. van, "Statistical cooling: A general approach to combinatorial optimization problems," *Philips Journal of Research*, Vol. 20, No. 4, pp. 193-226, September 1985.
- [10] Aarts E.H.L., and Korst J.H., *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing*, New York: Wiley, 1989.
- [11] Aspvall B., Plass M.F., and Tarjan R.E., "A linear-time algorithm for testing the truth of certain quantified Boolean formulas," *Information Processing Letters*, Vol. 3, No. 8, pp. 121-123, March 1979.
- [12] Auth E., and Schulz M.H., "Test Pattern Generation for Sequential Circuits," *IEEE Design & Test of Computers*, Vol. 3, pp. 72-86, June 1991.
- [13] Bahahona F., "A solvable class of quadratic 0-1 programming," *Discrete Applied Mathematics*, Vol. 13, No. 1, pp. 23-26, 1986.
- [14] Balas E., "An additive algorithm for solving linear programs with zero-one variables," *Operations Research*, Vol. 13, pp. 517-546, 1965.
- [15] Bending M.I., "Hitest - A Knowledge-Based Test Generation System," *IEEE Design & Test of Computers*, Vol. 1, pp. 83-93, February 1989.
- [16] Billionnet A., and Jaumard B., "A decomposition method for minimizing quadratic pseudo-Boolean functions," *Operations Research Letters*, Vol. 8, No. 3, pp. 161-163, 1989.

- [17] Bossen D.C., and Hong S.J., "Cause and Effect Analysis for Multiple Fault Detection in Combinational Networks," *IEEE Transactions on Computers*, Vol. C-20, No. 11, pp. 1252-1257, November 1971.
- [18] Carter M.W., "The Indefinite Zero One Quadratic Problem," *Discrete Applied Mathematics*, Vol. 7, No. 1, pp. 23-44, January 1984.
- [19] Chakradhar S.T., Agrawal V.D., and Bushnell M.L., "Automatic test generation using quadratic 0-1 programming," *27th ACM/IEEE Design Automation Conference*, pp. 654-659, 1990.
- [20] Chakradhar S.T., Bushnell M.L., and Agrawal V.D., "Towards massively parallel automatic test generation," *IEEE Transactions on Computer-Aided Design*, Vol. 9, No. 9, pp. 2235-2258, September 1990.
- [21] Chakradhar S.T., Agrawal V.D., and Bushnell M.L., *Neural Models and Algorithms for Digital Testing*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1991.
- [22] Chakradhar S.T., Agrawal V.D., and Rothweiler S.G., "A transitive closure algorithm for test generation," *IEEE Transactions on Computer-Aided Design*, Vol. 12, No. 7, pp. 1015-1028, July 1993.
- [23] Chandra S.J., and Patel J.H., "Test Generation in a Parallel Processing Environment," *Proceedings of the IEEE International Conference on Computer Design: VLSI in Computers and Processors*, pp. 11-14, 1988.
- [24] Chandra S.J., and Patel J.H., "Experimental Evaluation of Testability Measures for Test Generation," *IEEE Transactions on Computer-Aided Design*, Vol. 8, No. 1, pp. 93-97, January 1989.

- [25] Cheng W.T., and Chakraborty T.J., "Gentest: An Automatic Test Generation System for Sequential Circuits," *IEEE Computer*, Vol. 22, No. 4, pp. 43–49, April 1989.
- [26] Cheng W.T., and Davidson S., "Sequential Circuit Test Generator (STG) Benchmark Results," *Proceedings International Symposium on Circuits and Systems*, pp. 1938–1941, April 1989.
- [27] Cheng K.T., Agrawal V.D., and Kuh E.S., "A Simulation-Based Method for Generating Tests for Sequential Circuits," *IEEE Transactions on Computers*, Vol. C-39, No. 12, pp. 1456–1463, December 1990.
- [28] Clegg F.W., "Use of SPOOFs in the Analysis of Faulty Logic Networks," *IEEE Transactions on Computers*, Vol. C-22, No. 3, pp. 229–234, March 1973.
- [29] Cook S.A., "The complexity of theorem proving procedures," *Proc. Third Annual ACM Symposium on Theory of Computing*, 1971.
- [30] DeJong K.A., and Spears W.M., "An Analysis of the Interacting Roles of Population Size and Crossover in Genetic Algorithms," *Proc. First Workshop Parallel Problem Solving from Nature*, Springer-Verlag, Berlin, pp. 38–47, 1990.
- [31] Eichelberger E.B., and Williams T.W., "A Logic Design Structure for LSI Testability," *Journal Design Automation & Fault-Tolerant Computing*, Vol. 2, No. 2, pp. 165–178, May 1978.
- [32] Fujiwara H., and Toida S., "The Complexity of Fault Detection Problem for Combinational Circuits," *IEEE Transactions on Computers*, Vol. C-31, No. 6, pp. 555–560, June 1982.

- [33] Fujiwara H., and Shimono T., "On the Acceleration of Test Generation Algorithms," *IEEE Transactions on Computers*, Vol. C-32, No. 12, pp. 1137-1144, December 1983.
- [34] Fujiwara H., *Logic Testing and Design for Testability*, MIT Press, Cambridge, Massachusetts, 1985.
- [35] Fujiwara H., "FAN: A Fanout-Oriented Test Pattern Generation Algorithm," *Proceedings of the IEEE International Symposium on Circuits and Systems*, pp. 671-674, July 1985.
- [36] Fujiwara H., and Inoue T., "Optimal Granularity of Test Generation in a Distributed System," *IEEE Transactions on Computer-Aided Design*, Vol. 9, No. 8, pp. 885-892, August 1990.
- [37] Funatsu S., Wakatsuki N., and Yamada A., "Designing Digital Circuits with Easily Testable Consideration," *Proceedings of the IEEE International Test Conference*, pp. 98-102, September 1978.
- [38] Garey M.R., and Johnson D.S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman & Company, San Francisco, 1979.
- [39] Geurts W., Catthore F., and Man Hugo De, "Quadratic Zero-One Programming-Based Synthesis of Application Specific Data Paths," *IEEE Transactions on Computer-Aided Design*, Vol. 14, No. 1, pp. 696-702, January 1995.
- [40] Ghosh A., Devadas S., and Newton A.R., "Test Generation and Verification for Highly Sequential Circuits," *IEEE Transactions on Computer-Aided Design*, Vol. 10, No. 5, pp. 652-667, May 1991.

- [41] Giraldi J., and Bushnell M.L., "EST: The New Frontier in Automatic Test Pattern Generation," *Proceedings of the 27th ACM/IEEE Design Automation Conference*, pp. 667-672, June 1990.
- [42] Giraldi J., and Bushnell M.L., "Search State Equivalence for Redundancy Identification and Test Generation," *Proceedings of the IEEE International Test Conference*, 1991.
- [43] Glover F., and Woosley E., "Converting the 0-1 polynomial programming problem to a 0-1 linear program," *Operations Research*, Vol. 212, 180-182, 1974.
- [44] Goel P., "RAPS Test Pattern Generator," *IBM Technical Disclosure Bulletin*, Vol. 21, No. 7, pp. 2787-2791, December 1978.
- [45] Goel P., "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits," *IEEE Transactions on Computers*, Vol. C-30, No. 3, pp. 215-222, March 1981.
- [46] Goldberg D.E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, Mass., 1989.
- [47] Goldstein L.H., "Controllability/Observability Analysis of Digital Circuits," *IEEE Transactions on Circuits and Systems*, Vol. CAS-26, pp. 685-693, September 1979.
- [48] Goldstein L.H., and Thigpen E.L., "SCOAP: Sandia Controllability/Observability Analysis Program," *Proceedings 17th Design Automation Conference*, Minneapolis, MN, pp. 190-196, June 1980.
- [49] Gouders N., and Kaibel R., "Advanced Techniques for Sequential Test Generation," *Proceedings 2nd European Test Conference*, pp. 293-300, 1991.

- [50] Grefenstette J.J., "Optimization of Control Parameters for Genetic Algorithms," *IEEE Trans. on Systems, Man and Cybernetics*, Vol. SMC-16, No. 1, pp. 122-128, January/February 1986.
- [51] Hammer P.L., and Rudeanu S., "Pseudo-Boolean programming," *Operations Research*, Vol. 17, 231-261, 1969.
- [52] Hammer P.L., and Simeone B., *Quadratic Functions of Binary Variables*, Technical Report RRR # 20-87, Rutgers Center for Operations Research (RUTCOR), Rutgers University, NJ 08903, June 1987.
- [53] Hansen P., "Note sur l'extension a la méthode d' énumération implicite aux programmes nonlinéaires en variables zéro-un," *Cahiers du Centre de Recherche Opérationnelle*, Vol. 11, 162-166, 1969.
- [54] Hirose F., Takayama K., and Kamato N., "A Method to Generate Tests for Combinational Logic Circuits Using an Ultra High Speed Logic Simulator," *Proceedings of the 1988 International Test Conference*, pp. 102-107, 1988.
- [55] Ibarra O.H., and Sahni S.K., "Polynomially Complete Fault Detection Problems," *IEEE Transactions on Computers*, Vol. C-24, No. 3, pp. 242-249, March 1975.
- [56] Kelsey T.P., Saluja K.K., and Lee S.Y., "An Efficient Algorithm for Sequential Circuit Test Generation," *IEEE Transactions on Computers*, Vol. C-42, No. 12, pp. 242-249, December 1993.
- [57] Kinoshita K., Takamatsu Y., and Shibata M., "Test Generation for Combinational Circuits by Structure Description Functions," *Proceedings of the 10th International Symposium on Fault-Tolerant Computing*, pp. 152-154, 1980.

- [58] Kirkland T., and Mercer M.R., "A Topological Search Algorithm for ATPG," *Proceedings of the 24th ACM/IEEE Design Automation Conference*, pp. 502-508, June 1987.
- [59] Kirkpatrick S., Gelat Jr. C.D., and Vecchi M.P., "Optimization by simulated annealing," *Science*, Vol. 220, 671-680, 1983.
- [60] Klenke R.H., Williams R.D., and Aylor J.H., "Parallel-Processing Techniques for Automatic Test Pattern Generation," *IEEE Computer*, Vol. 25, No. 1, pp. 71-84, January 1992.
- [61] Kramer G., "Employing Massive Parallelism in Digital ATPG Algorithms," *Proceedings of the IEEE International Test Conference*, pp. 108-121, 1983.
- [62] Ku C.T., and Masson G.M., "The Boolean Difference and Multiple Fault Analysis," *IEEE Transactions on Computers*, Vol. C-24, No. 1, pp. 62-71, January 1975.
- [63] Kubo H., "A Procedure for Generating Test Sequences to Detect Sequential Circuit Failures," *NEC Research and Development*, Vol. 12, 1968.
- [64] Larrabee T., "Efficient Generation of Test Patterns Using Boolean Difference," *Proceedings of the IEEE International Test Conference*, pp. 795-801, August 1989.
- [65] Larrabee T., "A framework for evaluating test pattern generation strategies," *Proceedings of the International Conference on Computer Design*, October 1989.
- [66] Larrabee T., "Test Generation Using Boolean Satisfiability," *IEEE Transactions on Computer-Aided Design*, Vol. 11, No. 1, pp. 4-15, January 1992.

- [67] Laughunn D., "Quadratic binary programming with applications to capital budgeting problems," *Operations Research*, Vol. 18, pp. 454-461, 1970.
- [68] Levendel Y.H., Menon P.R., and Miller C.E., "Fault Simulation Methods - Extensions and Comparison," *Bell System Technical Journal*, Vol. 60, No. 9, pp. 2235-2258, November 1981.
- [69] Lipmann R., "An Introduction to Computing with Neural Nets," *IEEE ASSP Magazine*, Vol. 12, No. 2, pp. 102-110, April 1987.
- [70] Ma H.K.T. *et al.* "Logic Verification Algorithms and Their Parallel Implementation," *Proceedings of the 24th ACM/IEEE Design Automation Conference*, pp. 283-290, June 1987.
- [71] Ma H.K.T., Devadas S., Newton A.R., and Sangiovanni-Vincentelli A., "Test Generation for Sequential Circuits," *IEEE Transactions on Computer-Aided Design*, Vol. 7, No. 10, October 1988.
- [72] Mallela S., and Wu S., "A Sequential Circuit Test Generation System," *Proceedings of the IEEE International Test Conference*, pp. 57-61, November 1985.
- [73] Marlett R., "An Effective Test Generation System for Sequential Circuits," *Proceedings of the 23rd ACM/IEEE Design Automation Conference*, pp. 250-256, June 1986.
- [74] Motohara A. *et al.* "A Parallel Scheme for Test Pattern Generation," *Proceedings IEEE International Conference on Computer-Aided Design*, pp. 156-159, 1986.
- [75] Muth P., "A Nine-Valued Circuit Model for Test Generation," *IEEE Transactions on Electronic Computers*, Vol. C-25, No. 6, pp. 630-636, June 1976.

- [76] Nahar S., Sahni S., and Shragowitz E., "Simulated annealing and combinatorial optimization," *Proceedings 23rd Design Automation Conference*, Las Vegas, 293-299, June 1986.
- [77] Patil S., and Banerjee P., "A Parallel Branch-and-Bound Algorithm for Test Generation," *Proceedings of the 26th ACM/IEEE Design Automation Conference*, pp. 339-344, June 1989.
- [78] Patil S., and Banerjee P., "Fault Partitioning Issues in an Integrated Parallel Test Generation/Fault Simulation Environment," *Proceedings of the IEEE International Test Conference*, pp. 718-726, August 1989.
- [79] Patil S., and Banerjee P., "A Parallel Branch and Bound Algorithm for Test Generation," *IEEE Transactions on Computer-Aided Design*, Vol. 9, No. 3, pp. 313-322, March 1990.
- [80] Peterson C., "Computational experience with variants of the Balas' algorithm applied to the selection of R&D projects," *Management Science*, Vol. 13, 736-784, 1967.
- [81] Poage J.F., "Derivation of Optimum Tests to Detect Faults in Combinational Circuits," *Mathematical Theory and Automation*, New York: Polytechnic Press, 1963.
- [82] Pradhan D.K., *Fault-Tolerant Computing - Theory and Techniques*, Volume-I, Prentice-Hall, 1986.
- [83] Putzolu G.R., and Roth J.P., "A Heuristic Algorithm for Testing of Asynchronous Circuits," *IEEE Transactions on Electronic Computers*, Vol. C-20, No. 6, pp. 639-647, June 1971.

- [84] Randelman R.E., and Grest G.S., "N-city traveling salesman problem: optimization by simulated annealings," *Journal of Statistical Physics*, Vol. 45, 885–890, 1986.
- [85] Roth J.P., "Diagnosis of Automata Failures: A Calculus and a Method," *IBM Journal of Research and Development*, Vol. 10, No. 7, pp. 278–291, July 1966.
- [86] Savir J., Ditlow G.S., and Bardell P.H., "Random Pattern Testability," *IEEE Transaction on Computers*, Vol. C-33, pp. 79–90, January 1984.
- [87] Schneider P.R., "On the Necessity to Examine D-Chains in Diagnostic Test Generation," *IBM Journal of Research and Development*, Vol. 11, No. 1, page 114, January 1967.
- [88] Schulz M.H., Trischler E., and Sarfert T.M., "SOCRATES: A Highly Efficient Automatic Test Pattern Generation System," *IEEE Transactions on Computer-Aided Design*, Vol. 7, No. 1, pp. 126–136, January 1988.
- [89] Schulz M.H., and Auth E., "Advanced Automatic Test Pattern Generation and Redundancy Identification Techniques," *Proceedings of the 18th Symposium on Fault-Tolerant Computing*, pp. 30–35, June 1988.
- [90] Sellers E.F., Hsiao M.Y., and Bearnson L.W., "Error Detecting Logic for Digital Computers," *IEEE Transactions on Computers*, Vol. C-17, No. 7, pp. 676–683, July 1968.
- [91] Seth S.C., Pan L., and Agrawal V.D., "PREDICT – Probabilistic Estimation of Digital Circuit Testability," *Fault-Tolerant Computing Symposium (FTCS-15) Digest of Papers*, Ann Arbor, MI, pp 220–225, June 1985.

- [92] Sharma G.K., Sarkar S., Gupta J.P., and Agarwal R.P., "Efficient Heuristic Based CAD Tool for VLSI Test Generation," *Ninth National Convention of Electronics and Telecommunication Engineers*, University of Roorkee, pp. 270–274, March 1994.
- [93] Sharma G.K., Agarwal R.P., Sarkar S., and Gupta J.P., "An Efficient Tool for Automated Fault Diagnosis of Complex Digital Systems," *18th International Spring Seminar on Electronics Technology*, Temešvár-Czech Republic, pp. 60–64, June 1995.
- [94] Sharma G.K., Agarwal R.P., Sarkar S., and Gupta J.P., "Automatic Test Generation Using Genetic Algorithms," *7th International Conference on Microelectronics (ICM'95)*, Kuala Lumpur, Malaysia, December 19–21, 1995.
- [95] Sharma G.K., Misra P., Sarkar S., and Gupta J.P., "Genetic Algorithms Based CAD Tool for VLSI Test Generation Problem," *International Conference on Robotics, Vision and Parallel Processing for Industrial Automation*, Ipoh, Perak, Malaysia, November 28–30, 1996.
- [96] Sharma G.K., Kasana H.S., and Gupta J.P., "A Computational Algorithm for Solving a Quadratic 0-1 Programming Problem," *Communicated to Journal of Experimental Algorithms*, USA.
- [97] Sharma G.K., Agarwal R.P., Sarkar S., and Gupta J.P., "A New Quadratic 0-1 Programming Algorithm for Automatic Test Pattern Generation," *Communicated to Computers & Electrical Engineering (An International Journal)*, USA.
- [98] Smith S.P., Underwood B., and Mercer M.R., "An Analysis of Several Approaches to Circuit Partitioning for Parallel Logic Simulation," *Proceedings of*

- the IEEE International Conference on Computer Design: VLSI in Computers and Processors*, pp. 664–667, 1987.
- [99] Srinivas M., and Patnaik L.M., “Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms,” *IEEE Trans. on Systems, Man and Cybernetics*, Vol. 24, No. 4, pp. 656–666, July/August 1994.
- [100] Taha H.A., “Sequencing by implicit ranking and zero-one polynomial programming,” *Research Report No. 70-3*, Department of Industrial Engineering, University of Arkansas, September 1970.
- [101] Taha H.A., “A Balasian-based algorithm for zero-one polynomial programming,” *Management Science*, Vol. 18, pp. 328–343, 1972.
- [102] Takamatsu Y., and Kinoshita K., “CONT: A Concurrent Test Generation System,” *IEEE Transactions on Computer-Aided Design*, Vol. 8, No. 9, pp. 966–972, September 1989.
- [103] Takamatsu Y., and Kinoshita K., “Extended Selection of Switching Target Faults in CONT Algorithm for Test Generation,” *Journal Electronic Testing: Theory and Applications*, Vol. 1, No. 3, pp. 183–189, October 1990.
- [104] *The TTL Data Book for Design Engineers, Second edition*, Texas Instruments, 1973.
- [105] Laarhoven P.J.M. van, and Aarts E.H.L., *Simulated Annealing: Theory and Applications*, Dordrecht, The Netherlands: Kluwer Academic, 1987.
- [106] Waicukauski J.A., Eichelberger E.B., Forlenza D.O., Lindbloom E., and McCarthy T., “Fault Simulation for Structured VLSI,” *VLSI Design*, Vol. VI, pp. 20–32, 1985.

- [107] Watters L.J., "Reduction of integer polynomial problems to zero-one linear programming problems," *Operations Research*, Vol. 15, pp. 1171-1174, 1976.
- [108] Williams M.J.Y., and Angell J.B., "Enhancing Testability of Large Scale Integrated Circuits Via Test Points and Additional Logic," *IEEE Transactions on Computers*, Vol. C-22, No. 1, pp. 46-60, January 1973.