Delay Fault Testing and Statistical Detectability Analysis in Scan-Based Logic Circuits

by

Zaifu Zhang

A Thesis

Submitted to the Faculty of Graduate Studies

in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

Department of

Electrical and Computer Engineering

The University of Manitoba

Winnipeg, Manitoba

National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file *Votre référence*

Our file *Notre référence*

ISBN 0-612-16391-1

Canada

Name _____

Dissertation Abstracts International is arranged by broad, general subject categories. Please select the one subject which most nearly describes the content of your dissertation. Enter the corresponding four-digit code in the spaces provided.

ELECTRONICS AND ELECTRICAL

SUBJECT TERM

0544 U·M·I

SUBJECT CODE

Subject Categories

THE HUMANITIES AND SOCIAL SCIENCES

COMMUNICATIONS AND THE ARTS

Architecture 0729
Art History 0377
Cinema 0900
Dance 0378
Fine Arts 0357
Information Science 0723
Journalism 0391
Library Science 0399
Mass Communications 0708
Music 0413
Speech Communication 0459
Theater 0465

EDUCATION

General 0515
Administration 0514
Adult and Continuing 0516
Agricultural 0517
Art 0273
Bilingual and Multicultural 0282
Business 0688
Community College 0275
Curriculum and Instruction 0727
Early Childhood 0518
Elementary 0524
Finance 0277
Guidance and Counseling 0519
Health 0680
Higher 0745
History of 0520
Home Economics 0278
Industrial 0521
Language and Literature 0279
Mathematics 0280
Music 0522
Philosophy of 0998
Physical 0523
Psychology 0525
Reading 0535
Religious 0527
Sciences 0714
Secondary 0533
Social Sciences 0534
Sociology of 0340
Special 0529
Teacher Training 0530
Technology 0710
Tests and Measurements 0288
Vocational 0747

LANGUAGE, LITERATURE AND LINGUISTICS

Language
- General 0679
- Ancient 0289
- Linguistics 0290
- Modern 0291

Literature
- General 0401
- Classical 0294
- Comparative 0295
- Medieval 0297
- Modern 0298
- African 0316
- American 0591
- Asian 0305
- Canadian (English) 0352
- Canadian (French) 0355
- English 0593
- Germanic 0311
- Latin American 0312
- Middle Eastern 0315
- Romance 0313
- Slavic and East European 0314

PHILOSOPHY, RELIGION AND THEOLOGY

Philosophy 0422
Religion
- General 0318
- Biblical Studies 0321
- Clergy 0319
- History of 0320
- Philosophy of 0322

Theology 0469

SOCIAL SCIENCES

American Studies 0323
Anthropology
- Archaeology 0324
- Cultural 0326
- Physical 0327

Business Administration
- General 0310
- Accounting 0272
- Banking 0770
- Management 0454
- Marketing 0338

Canadian Studies 0385
Economics
- General 0501
- Agricultural 0503
- Commerce-Business 0505
- Finance 0508
- History 0509
- Labor 0510
- Theory 0511

Folklore 0358
Geography 0366
Gerontology 0351
History
- General 0578
- Ancient 0579
- Medieval 0581
- Modern 0582
- Black 0328
- African 0331
- Asia, Australia and Oceania 0332
- Canadian 0334
- European 0335
- Latin American 0336
- Middle Eastern 0333
- United States 0337

History of Science 0585
Law 0398
Political Science
- General 0615
- International Law and Relations 0616
- Public Administration 0617

Recreation 0814
Social Work 0452
Sociology
- General 0626
- Criminology and Penology 0627
- Demography 0938
- Ethnic and Racial Studies 0631
- Individual and Family Studies 0628
- Industrial and Labor Relations 0629
- Public and Social Welfare 0630
- Social Structure and Development 0700
- Theory and Methods 0344

Transportation 0709
Urban and Regional Planning 0999
Women's Studies 0453

THE SCIENCES AND ENGINEERING

BIOLOGICAL SCIENCES

Agriculture
- General 0473
- Agronomy 0285
- Animal Culture and Nutrition 0475
- Animal Pathology 0476
- Food Science and Technology 0359
- Forestry and Wildlife 0478
- Plant Culture 0479
- Plant Pathology 0480
- Plant Physiology 0817
- Range Management 0777
- Wood Technology 0746

Biology
- General 0306
- Anatomy 0287
- Biostatistics 0308
- Botany 0309
- Cell 0379
- Ecology 0329
- Entomology 0353
- Genetics 0369
- Limnology 0793
- Microbiology 0410
- Molecular 0307
- Neuroscience 0317
- Oceanography 0416
- Physiology 0433
- Radiation 0821
- Veterinary Science 0778
- Zoology 0472

Biophysics
- General 0786
- Medical 0760

EARTH SCIENCES

Biogeochemistry 0425
Geochemistry 0996
Geodesy 0370
Geology 0372
Geophysics 0373
Hydrology 0388
Mineralogy 0411
Paleobotany 0345
Paleoecology 0426
Paleontology 0418
Paleozoology 0985
Palynology 0427
Physical Geography 0368
Physical Oceanography 0415

HEALTH AND ENVIRONMENTAL SCIENCES

Environmental Sciences 0768
Health Sciences
- General 0566
- Audiology 0300
- Chemotherapy 0992
- Dentistry 0567
- Education 0350
- Hospital Management 0769
- Human Development 0758
- Immunology 0982
- Medicine and Surgery 0564
- Mental Health 0347
- Nursing 0569
- Nutrition 0570
- Obstetrics and Gynecology 0380
- Occupational Health and Therapy 0354
- Ophthalmology 0381
- Pathology 0571
- Pharmacology 0419
- Pharmacy 0572
- Physical Therapy 0382
- Public Health 0573
- Radiology 0574
- Recreation 0575
- Speech Pathology 0460
- Toxicology 0383

Home Economics 0386

PHYSICAL SCIENCES

Pure Sciences

Chemistry
- General 0485
- Agricultural 0749
- Analytical 0486
- Biochemistry 0487
- Inorganic 0488
- Nuclear 0738
- Organic 0490
- Pharmaceutical 0491
- Physical 0494
- Polymer 0495
- Radiation 0754

Mathematics 0405
Physics
- General 0605
- Acoustics 0986
- Astronomy and Astrophysics 0606
- Atmospheric Science 0608
- Atomic 0748
- Electronics and Electricity 0607
- Elementary Particles and High Energy 0798
- Fluid and Plasma 0759
- Molecular 0609
- Nuclear 0610
- Optics 0752
- Radiation 0756
- Solid State 0611

Statistics 0463

Applied Sciences

Applied Mechanics 0346
Computer Science 0984
Engineering
- General 0537
- Aerospace 0538
- Agricultural 0539
- Automotive 0540
- Biomedical 0541
- Chemical 0542
- Civil 0543
- Electronics and Electrical 0544
- Heat and Thermodynamics 0348
- Hydraulic 0545
- Industrial 0546
- Marine 0547
- Materials Science 0794
- Mechanical 0548
- Metallurgy 0743
- Mining 0551
- Nuclear 0552
- Packaging 0549
- Petroleum 0765
- Sanitary and Municipal System Science 0554 0790

Geotechnology 0428
Operations Research 0796
Plastics Technology 0795
Textile Technology 0994

PSYCHOLOGY

General 0621
Behavioral 0384
Clinical 0622
Developmental 0620
Experimental 0623
Industrial 0624
Personality 0625
Physiological 0989
Psychobiology 0349
Psychometrics 0632
Social 0451

DELAY FAULT TESTING AND STATISTICAL DETECTABILITY ANALYSIS IN SCAN-BASED LOGIC CIRCUITS

BY

ZAIFU ZHANG

A Thesis submitted to the Faculty of Graduate Studies of the University of Manitoba in partial fulfillment of the requirements of the degree of

DOCTOR OF PHILOSOPHY

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

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

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

Abstract

With the ever increasing complexity of digital logic systems, testing of circuits and systems has become a major effort in design and production. Test pattern generation, design for test, and testability analysis issues have become more and more important. In the past, the most common model used to describe faults in circuits has been the static stuck-at fault model. More recently, attention has focused on dynamic delay fault models. The work reported here concentrates on these models. First, an automatic test pattern generation technique for stuck-open and delay faults using neural network models is presented. The transition gate level model of stuck-open faults in CMOS circuits and their relation to transition gate delay faults (e.g. slow-to-rise (SR) and slow-to-fall (SF) transition gate delay faults) in logic circuits are discussed. A neural network computation technique for generating robust test patterns for stuck-open and delay faults is proposed. Second, a technique augmenting scan path shift register latches (SRLs) in a multiple scan chain skewed-load delay fault testing scheme with a one-level XOR network is proposed. This technique aims to reduce *structure dependency*, *linear dependency*, and *shift dependency* in multiple scan chain skewed-load delay fault testing. In addition, the proposed multiple scan chain scheme fed by selected taps from a limited stage cellular automata (CA) generator can be applied to achieve exhaustive testing of logic circuits when incorporating it with a logic partition technique. Simulation experiments on the ISCAS85 benchmark circuits show a substantial improvement in the transition gate delay fault coverage with the proposed schemes. Third, a statistical transition gate delay and path delay fault detectability estimation technique is presented. The definitions for

transition probability and transition sensitization probability for random input patterns are given. Calculating line transition sensitization considering line correlation within each gate and each fanout free region is presented, and a strategy to compute the transition observabilities of different types of fanout stems is proposed. The detectability of a path delay fault can be evaluated as the product of the observabilities of the input line to its *head gate* within each fanout free logic cone on the path, multiplied by the transition controllability of the path. This estimation technique has been used to grade both transition gate delay and path delay faults, with comparison to accurate fault simulation results. Finally, future work is discussed.

Acknowledgements

I would like to thank my supervisor Dr. Robert D. McLeod for his constant and patient supervision and encouragement with my research. His very kind and generous personality has influenced me greatly.

In addition, I would like to thank Dr. Witold Pedrycz for helping me to understand neural-net computation mechanisms, Dr. Howard C. Card for the numerous discussions and teaching me how to initiate real research, and Dr. D. Michael Miller for discussions and help with my research.

Also, I would like to thank my good friends: Dave Blight, Ken Ferens, Gordon McGonigal, Dean McNeill, Richard Wieler, T.T. Chau, Hongwei Kong, and Ming Zhang for their availability and assistance. I have been fortunate to make these nice friends, and I have learned much from them in both life and academia.

Financial support for this work from the Natural Sciences and Engineering Research Council of Canada, the Federal Government of Canada's MICRONET Centers of Excellence Program, and the Canadian Microelectronics Corporation is gratefully acknowledged.

Finally, I would like to thank my wife, Zhonghui Yao, and my son, Yao Zhang, for their constant support and tolerance with me when I was busy with this thesis. I could not have completed this thesis without their understanding and support.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

The testing of a digital system, or a single integrated circuit is a major portion of the effort in the overall design process. Logic circuit testing plays an extremely vital role with the ultimate goal of achieving high reliability, availability, safety, maintainability, fault tolerance, and/or meeting design requirements. A digital system will be tested and diagnosed on numerous occasions during the development cycle as well as during the product lifetime. In order for a system to perform its intended mission in a reliable manner, testing and diagnosis must be quick and effective.

The testing of a digital system attempts to answer this question: does the digital system work correctly? That is, the testing of a digital system attempts to ensure the correctness of the system in that the hardware is defect-free.

For an external test, it is necessary that the device under test be removed from its operational environment and be subjected to various tests using test equipment. However, with continuously increasing digital circuit density and limited access to

the internal circuitry of the device being tested, it is difficult to perform a complete test externally. Performing a complete test on increasingly complex devices requires increasingly expensive automatic test equipment.

Alternately, testing can be specified as one of the system functions, that is, self-test. Self-test can be implemented in software, hardware, or both. A software test alone has the following disadvantages: poor diagnostic resolution and a long test period. Therefore, built-in self-test (BIST) implemented in the hardware itself is often adopted in a digital system. BIST can perform efficient tests on the system itself in less time and without overly expensive automatic test equipment.

Research in BIST schemes for static stuck-at fault testing has been extensively investigated in the past. However, a defect due to random variation in process parameters may cause a logic circuit to pass static stuck-at fault testing but cause the logic circuit to fail when operated at system speeds. Research in BIST schemes for dynamic delay fault testing becomes more important, especially for digital systems which require high speed performance features. Delay fault testing can verify both the functional behavior and correct operation at system speed.

In this thesis, test pattern generation techniques for stuck-open and delay fault testing using neural network computation machines, multiple scan chain delay fault test schemes, and a statistical transition delay and path delay fault analysis technique are explored.

1.2 Scope

Chapter 2 surveys various built-in self-test methodologies. Scan based design, logic fault models (used to model various hardware faults), test pattern generation, and test response compaction techniques in BIST schemes are explained. Several BIST architectures are reviewed and explained. Novel techniques for evaluating BIST designs, especially hardware emulation techniques based on field programmable gate arrays (FPGA), are reviewed.

In Chapter 3, a test pattern generation technique for stuck-open and delay faults with neural network computation is presented. A robust test pattern generation procedure is proposed. The procedure to generate test patterns for path delay faults is also presented.

In Chapter 4, a multiple scan chain skewed-load delay fault test scheme augmented by a one-level XOR network is presented. A shift register latch re-arranging heuristic with the proposed multiple scan chain test scheme to enhance delay fault testing is proposed. The arrangeability of the SRLs in a scan chain and various *dependencies* existing in the multiple scan chains is proposed. It is indicated that the proposed multiple scan chain test scheme fed by selected taps from a limited stage CA pattern generator can be used to arrange the SRLs for exhaustive stuck-at fault testing when incorporated with a testability-driven logic partitioning technique.

In Chapter 5, a statistical transition delay and path delay fault analysis technique to estimate delay fault coverage and grade transition delay faults for combinational logic circuits is proposed. The accuracy of the estimated transition sensitization controllabilities of $0 \rightharpoonup 1$ and $1 \rightharpoonup 0$ transitions of the lines in each gate is improved by

considering the line correlation in each fanout free logic cone. A strategy to estimate the transition observabilities of fanout stems with various topologies is presented. The detectability of a path delay fault is estimated as the product of the observabilities of the input line to its *head gate* within each fanout free logic cone on the path, multiplied by the transition controllability of the path.

Finally, Chapter 6 concludes by summarizing the major points proposed in the thesis. A brief discussion of future work is also presented.

Chapter 2

Literature Survey

With the growing complexity of modern VLSI circuits, the ability to perform cost effective testing has become increasingly difficult. Since each integrated circuit (IC) device is of very high density and the ratio of pins to gates in each IC device is quite low there is limited accessibility to the internal nodes of a circuit. In order to reduce the cost of testing and improve the quality of test, inherent testability for either deterministic or random tests is a desirable feature of circuits in modern logic design. Design for test and built-in test strategies can improve controllability and observability of internal circuit logic blocks, provide better test coverage in less time, and eliminate the need for expensive specialized test equipment to diagnose faults[1][2][3][4].

Since testing sequential logic circuits is much more difficult than testing combinational logic circuits, scan based IC design in which sequential logic circuits are transformed into combinational logic circuits is often adopted. A variety of partial and full scan design schemes have been proposed[5]–[9] to enhance IC testing. In general, full scan design can significantly reduce the cost of test generation and lead

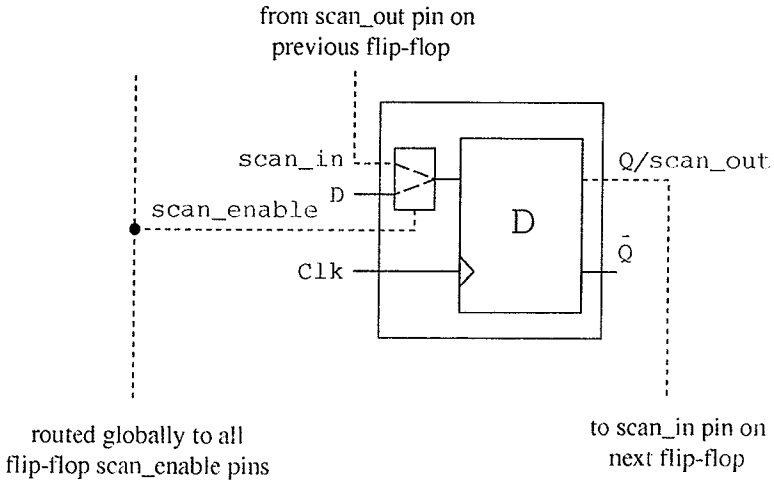


Figure 2.1: A multiplexed D flip-flop scan element

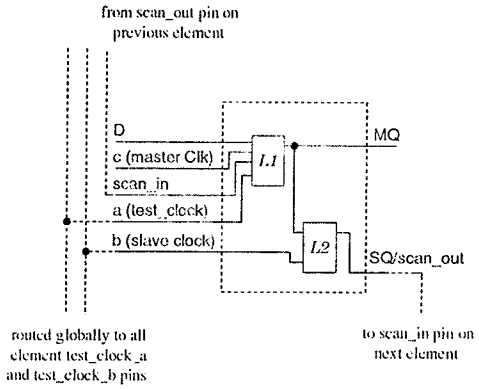
to higher fault coverage, but storage elements like flip-flops and/or register latches in the full scan design are more complex and more silicon overhead is needed. Compared to the full scan design, the partial scan design is more economical since only selected storage elements are made scannable[3]. However, in the partial scan design, an optimal selection of a minimal number of storage elements forming a scan path such that the resulting circuit has a balanced structure[1] is an *NP-complete* problem[10].

2.1 Scan Based Design

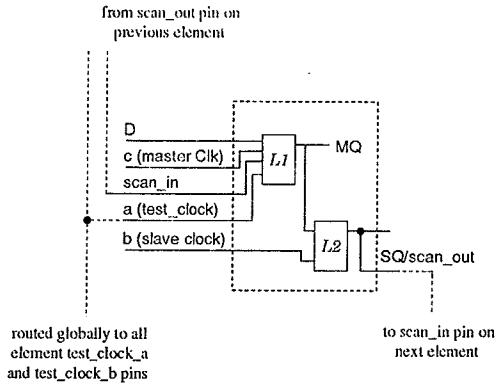
In scan based design of an IC, the memory elements (flip-flops or shift register latches) in the circuit are threaded together into a serial shift register chain such that the values of these memory elements can be both controlled and observed[1].

There are several different scan implementation styles which are commonly used in

[1]A synchronous sequential circuit S is said to be balanced, denoted as a balance-structure, if for any two clouds v_1 and v_2 in S, all signal paths (if any) between v_1 and v_2 go through the same number of registers.



(a) A single latch LSSD implementation scan cell



(b) A double latch LSSD implementation scan cell

Figure 2.2: LSSD scan cell implementation styles

scan based design. As shown in figure 2.1, a multiplexed flip-flop scan implementation includes a flip-flop and a multiplexer that selects the input data. In the normal operation mode, the *scan_enable* control signal selects the data from the system input (D). During the serial shifting mode, the multiplexer selects data from the *scan_in* input. The multiplexed flip-flop scan implementation requires relatively low area overhead and three additional I/O pins.

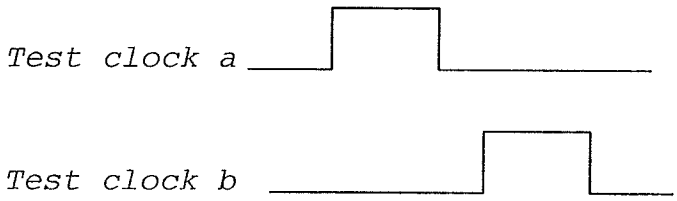


Figure 2.3: Non-overlapping test clocks a and b

Researchers at IBM[8][9][1] have developed a serial integrated scan chain architecture: level-sensitive scan design (LSSD). Figure 2.2 illustrates the LSSD single latch scan cell implementation and the LSSD double latch scan cell implementation.

In the single latch LSSD implementation as shown in figure 2.2(a), two non-overlapping test clocks a and b as shown in figure 2.3 are added, to shift data from the scan input through both the master and slave stages to the slave output SQ. System data D is enabled by the level-sensitive clock c to the output of the master latch MQ.

In the double latch LSSD implementation as shown in figure 2.2(b), the clock b (slave_clock) is used in both normal and serial shift (test) operation. The system data D is clocked through the latch pair using master/slave clocking on the master clock c and the slave clock b. Data are clocked out to the slave output SQ. In the serial shift operation mode, two-phase, non-overlapping master/slave test clocks are applied to the a (test_clock) and the b (slave_clock) clock inputs to shift data from the scan input through both the master and slave stages to the slave output SQ.

The LSSD implementations have negligible performance impact. The LSSD implementations are well suited to both full and partial scan design. However, the single latch LSSD implementations have slightly higher area overhead.

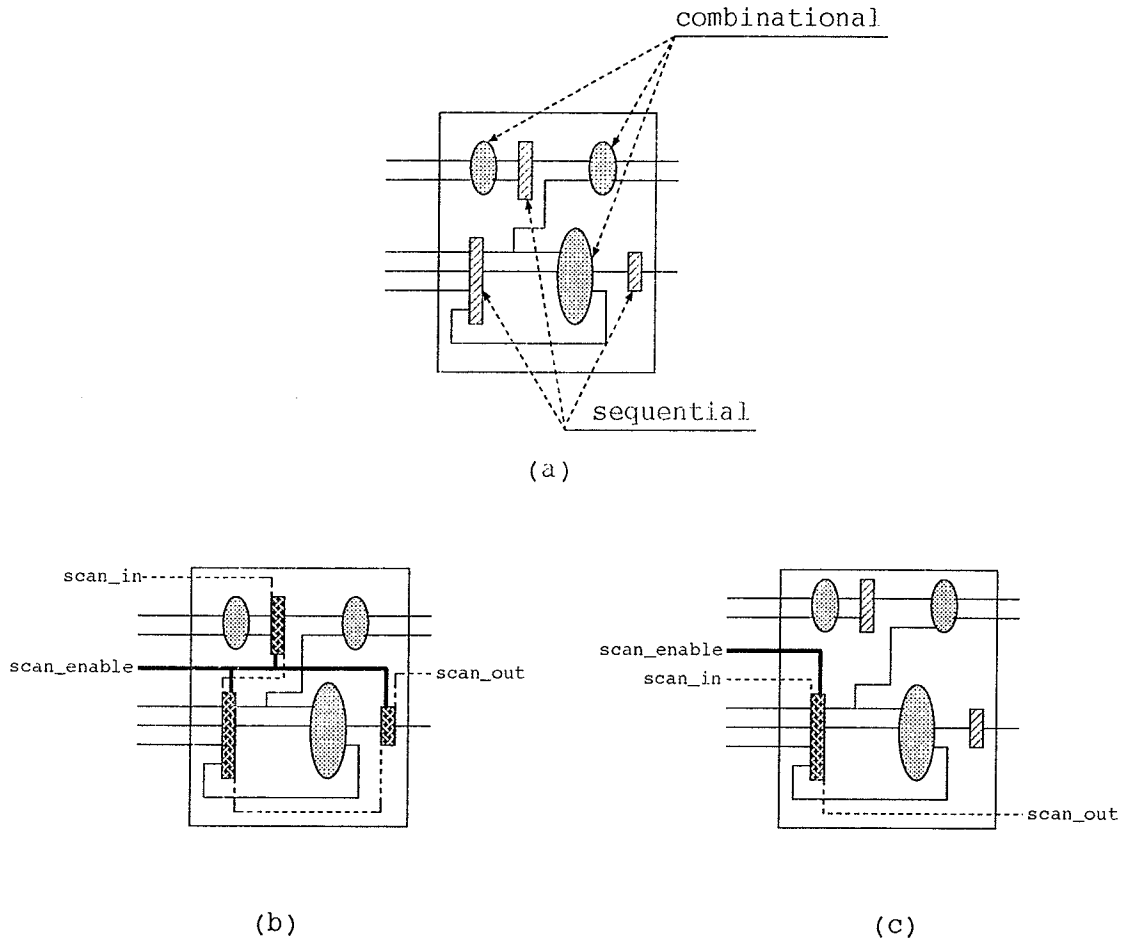


Figure 2.4: Block diagram of the multiplexed flip-flop based full scan and partial scan designs

Figure 2.4(a) shows a block diagram of an IC design without scan circuitry. Figure 2.4(b) shows a general structure of the IC with the multiplexed flip-flop based full scan design in which all sequential flip-flops are modified. Figure 2.4(c) shows a diagram of the IC with partial scan. In partial scan design, not all of the sequential flip-flops are modified. In general, when a design is modified with insertion of scan circuitry, a number of additional physical I/O pins (e.g. *scan_in*, *scan_enable*, and

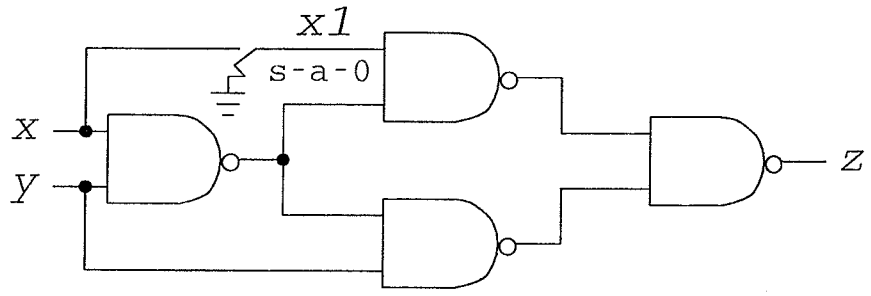


Figure 2.5: A circuit with a s-a-0 fault in line X_1

scan_out, etc.) are required for test purposes. However, the testability of a design after inserting the scan circuitry is dramatically improved because the internal states of the IC can be both observed and controlled.

2.2 Fault Models and Fault Collapsing

Failures of breaks in lines and shorts between lines may occur during fabrication. The types of failures vary with the technology. A physical failure will change the function of a logic circuit. In order to determine the existence of (or absence of) any physical failures in a logic circuit, an appropriate sequence of test inputs are applied to the logic circuit. Because many different physical failures may very well cause the same error under a test, modeling physical failures as logic faults is used to simplify the complexity of fault analysis with various physical failures.

Fault analysis requires modeling. Fault models aid in the generation of test patterns and in the evaluation of test quality defined in terms of coverage of modeled faults[11]. Fault modeling is strongly related to circuit modeling. For logic circuits, there are static and dynamic fault models.

2.2.1 Stuck-At Fault Model

Stuck-At Fault

Stuck-at faults are assumed to affect only the interconnections (or lines) between gates in a logic circuit. Each line can have two types of stuck-at faults: stuck-at-0 and stuck-at-1. Stuck-at faults permanently stay at logic 0 or 1 levels, hence they are static fault models. As shown in figure 2.5 there exists a s-a-0 fault in line X_1 in the circuit. This s-a-0 fault will make the logic value of line X_1 stay at logic 0 whatever logic values are applied to the input X. One of the attributes of stuck-at faults is that many different physical faults can be represented by them.

The stuck-at fault model is technology independent, and many non-classical faults can be detected during stuck-at fault testing[3].

Single Stuck-At Fault Model

There are $3^p - 1$ possible stuck-at faults for a logic circuit containing p lines. Fault analysis on all possible faults becomes intractable for a large number of lines n, since the number of faults increase exponentially with the value of p. Usually, a single stuck-at fault model is adopted in fault analysis. This assumption is justified by the *frequent testing strategy*, which states that an IC should be tested often enough such that the probability of more than one fault developing between two consecutive testing experiments is sufficiently small[3]. It has been reported that, in most cases, a multiple fault can be detected by tests designed for single stuck-at faults that compose the multiple one[12][3].

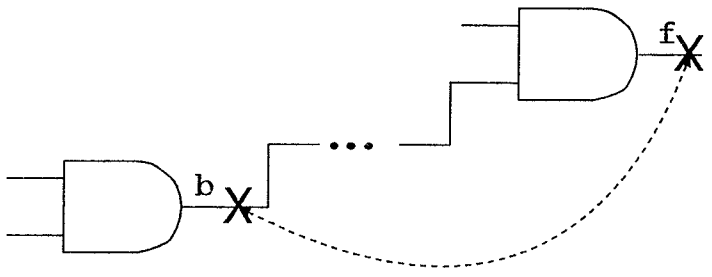


Figure 2.6: A circuit with a feedback AND bridging fault $(b.f)$ between line b and f

2.2.2 Bridging Fault Model

A bridging fault models a short which is formed by connecting points not intended to be connected in a logic circuit. A bridging fault can be classified as an AND bridging fault or an OR bridging fault according to the function introduced by the short[3]. When there exists at least one path between the two shorted lines, a feedback AND (or OR) bridging fault, as shown in figure 2.6, occurs. Feedback bridging faults may cause combinational circuits to become sequential.

In general, most bridging faults can be detected by only one test. However, some feedback bridging faults may require a test sequence to detect them[3].

2.2.3 Delay Fault Model

Some defects do not affect the logic function, but do affect the speed of a gate. A delay fault models a failure caused by a defect in a clocked logic circuit in which the combinational logic circuit fails to propagate data in time for clocking into next stage of latches or primary outputs[13][14]. There are two types of delay fault models: path

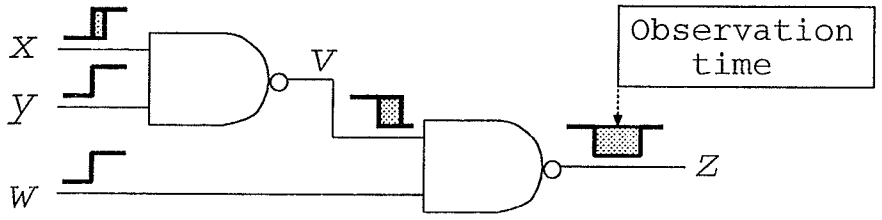


Figure 2.7: A circuit with a slow to rise transition path fault along a path $X - V - Z$

delay faults and gate delay faults.

Path Delay Fault Model

A path delay fault is a path of the combinational logic circuit between input and output latches in which a transition in the specified direction initiated by the setting of an input latch does not arrive at the path output in time for proper setting into an output latch. An example of a slow to rise transition path fault along a path $X - V - Z$ is shown in figure 2.7.

The path delay fault model aims to account for the cumulative, or additive effect of delays along a path. It is global and able to model a delay defect that may be distributed over a region of a circuit[16].

Gate Delay Fault Model

Every logic gate introduces a delay to the signals propagating through it; therefore, the nominal delay in propagating these signals are lumped and associated with each logic gate[2][15][3]. A gate delay fault is defined as a gate defect that results in a slow transition fault on at least one path from the site of the gate to a latch or an output.

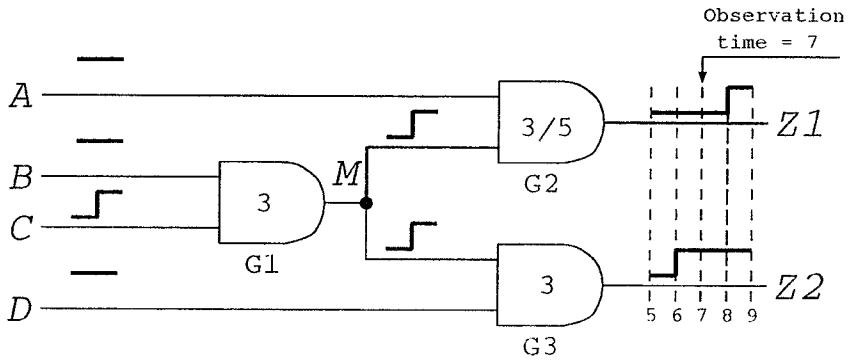


Figure 2.8: A circuit with a gate delay fault in a gate G_2

As shown in figure 2.8, there exists an extra nominal delay of 2 units in gate G_2 in the circuit. This extra delay will cause a slow transition fault along the path through $M - Z_1$, when the measurement time occurs at 7 units.

A second gate delay fault model is the transition gate delay fault model or gross gate delay fault model[4][17]. A transition gate delay fault (or gross gate delay fault) models a gate delay in which the magnitude is such that it causes a slow transition fault of all paths passing through the gate. Hence, the transition gate delay fault model is less accurate than the gate delay fault model.

The gate delay fault model is a local fault model. It is less accurate than the path delay fault model. However, the number of path delay faults may grow exponentially with the circuit depth such that test generation for the entire path delay fault set may become intractable[22]. Also, since test generation and fault simulation for gate delay faults need quantitative timing analysis for the gate delay faults with various sizes[18]–[21], they are computationally expensive. Usually, the transition gate delay fault model is used for evaluating the detection of the delay faults.

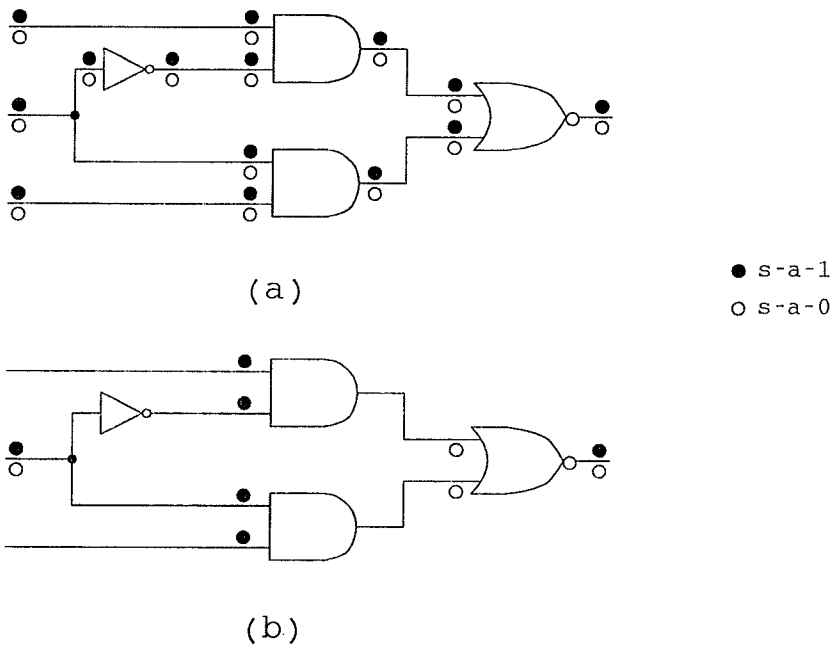


Figure 2.9: (a) All possible single stuck-at faults in a circuit, (b) Collapsed single stuck-at faults based on the structure equivalence relations

2.2.4 Fault Collapsing in a Single Fault Set

In general, in a logic circuit with p lines there are $2 \times p$ single stuck-at faults (or gate delay faults) in the complete single fault set. Since there are some faults in which the effect is indistinguishable at the output of the circuit, these faults are equivalent and can be represented by one of them. The time required for fault simulation and test pattern generation on the collapsed fault set can be reduced without any loss of information.

The stuck-at faults which are functionally or structurally equivalent can be collapsed[23]. For example, any input s-a-0 fault is equivalent to the output s-a-0 fault in an *AND* gate, and the output s-a-1 fault of an *AND* gate is equivalent to all input s-a-1 faults of the gate. The collapsed single stuck-at fault set in an *AND* gate includes

s-a-1 faults on the all inputs of the gate and a s-a-0 fault on the output of the gate. Figure 2.9 illustrates the process of fault collapsing based on the structure equivalence relations. Generally, structure equivalence fault collapsing reduces the initial fault set by 50%[3]. The stuck-at fault collapsing procedure based on functional equivalence relations can be used to further reduce the collapsed fault number[24][25].

Similarly, transition gate delay faults also can be collapsed by the structural equivalence to obtain a reduced fault set[101][97][102].

2.2.5 Redundant Faults

Faults in a logic circuit are said to be redundant if no test vectors exist to detect the faults. Lines with redundant faults are called redundant lines. In general, if a line has a stuck-at redundant fault, it also has a redundant transition delay fault. The existence of a redundant fault can invalidate the testing of a detectable fault if both are present simultaneously[26]. Redundancy also affects the test pattern generation time since identifying redundancy is a *NP-complete* problem[26][3]. Therefore, the incorporation of redundant lines should be avoided during the design phase of a logic circuit.

2.3 Built-In Self-Test

Built-in self-test (BIST) is the capability of a circuit (chip, board, or system) to test itself. When testing is built into the hardware, it has the potential of being fast and cost-efficient as well as hierarchical[27]. Growing demands for high product quality in manufacturing and throughout the life cycle of a product can often only

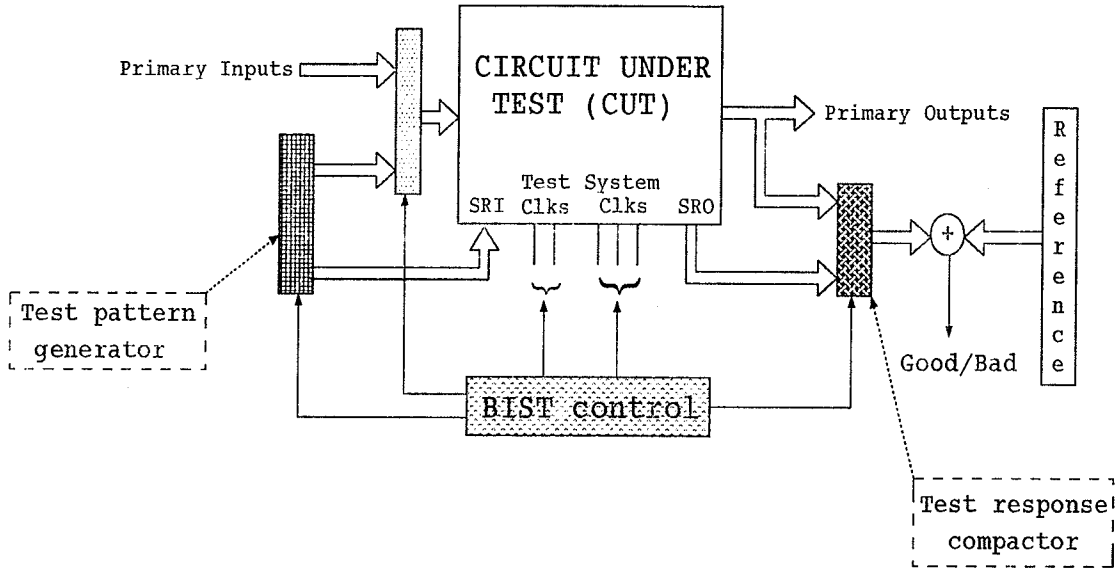


Figure 2.10: A general BIST configuration

be met at reasonable expense using BIST techniques[29]. Figure 2.10 shows a general off-line BIST configuration. During the test session, test patterns generated by the test pattern generator are loaded into the primary inputs and the scannable shift register inputs (SRI). The test responses are captured by the primary outputs and shift register outputs (SRO), and compacted by the test response compactor. A Good/Bad indication is obtained after the last test by comparing the binary values remaining in the compactor, to the reference values[2][28].

Two key components of BIST configuration are test pattern generation and test response compaction[2][27]. These two components influence the effectiveness of a BIST scheme.

2.3.1 Test Pattern Generation for BIST

Test pattern generation is crucial for stimulating and propagating faulty circuit behavior to primary outputs. Test patterns can be generated with the traditional structural automatic test pattern generation (ATPG) techniques with the help of the static and dynamic learning strategies[31]–[35] or Boolean satisfiability algorithms[36][37]. The generated test patterns are first compacted to obtain a minimum test set, and then stored in a read-only memory (ROM) for use during testing. Since the stored-pattern BIST scheme has a high area overhead, pseudo-random pattern generation is often used. a pseudo-random test eliminates the difficulty with test pattern generation and can have very high fault coverage. Two types of pseudo-random pattern generators are considered: linear feedback shift registers (LFSR)[38][2] and one dimensional linear cellular automata (CA)[39][40]. It has been illustrated that maximum length LFSR and CA sequences have good randomness properties[2][39]. Since the phase shift of adjacent bits of an n stage maximum length CA is much larger than that of adjacent bits of an n-stage maximum length LFSR[41], CAs can generate better parallel random test patterns than LFSRs for delay and stuck-open fault tests[42][43][44]. The problem with pseudo-random testing is that some circuits contain random-pattern-resistant faults and thus require long test lengths to ensure high fault coverage[3]. Weighted pseudo-random pattern generation techniques[61][45][4][48], test point insertion techniques[2][3][4][46], and techniques specifying test patterns by reprogramming multiple polynomial LFSR and re-seeding[49][50] can be used to improve the testability of these random-pattern-resistant faults.

2.3.2 Test Response Compaction

Test response compaction is crucial for capturing and retaining the faulty behavior of circuits. There are several different compaction schemes including transition counting, syndrome checking, and signature analysis[47][2][3].

In the transition counting technique, the total number of 0-to-1 and 1-to-0 transitions in the response stream are counted, and compared with the number in the fault-free circuit response stream.

In the syndrome checking technique, the total number of 1's in the response sequence of a circuit under test are counted. This total number of 1's is called the syndrome of the circuit under test. Generally, the syndromes of fault-free and faulty circuits are different.

Signature analysis is a compaction technique based on the concept of cyclic redundancy checking (CRC) and is realized in hardware using LFSRs or CAs[2][3][51]. In signature analysis compaction, the response sequence is fed into a LFSR or a CA and results in a compacted sequence. This compacted sequence is of the same length as that of the LFSR or the CA, and is called the signature of the circuit under test in response to the applied test sequence.

It has been proven that none of these compaction techniques are perfect, that is, aliasing or error masking[2] happens in all these compaction schemes[27]. The signature analysis compaction scheme is the most popular and used in BIST since it can be easily realized by means of a LFSR, or CA[51]. It has been pointed out that a CA

[2]When the signatures (total number of 0-to-1 and 1-to-0 transitions or syndromes) of the faulty and fault-free output responses are same, it is referred to as *error masking*, and the erroneous output response is said to be an *alias* of the correct output response.

and a LFSR which are based on the same irreducible (or primitive) polynomial have the same aliasing probabilities[52]. Also, for an n stage LFSR or CA based signature analyzer fed with a test bit stream of length m, the aliasing (or error masking) probability will be

$$\frac{2^{m-n}-1}{2^m-1} \cong \frac{1}{2^n} \tag{2.1}$$

when $m \gg n$[47][2][3].

2.4 Several BIST Architectures

A circuit which uses a BIST technique is often a scan-based design[8]. Usually, a LFSR or CA test pattern generator, signature analyzer, and an appropriate BIST controller are included in the circuit as shown in figure 2.10. In general, BIST techniques can be classified into two categories: test-per-clock and test-per-scan schemes[2][53][28]. In the test-per-clock BIST scheme, a multiple input signature register (MISR) is configured to capture a response for a test pattern in each clock period. In the test-per-scan scheme, a signature analyzer captures a response for a test pattern in each scan cycle. A scan cycle is the number of clock cycles required to shift the test pattern into a serial scan path or to shift the response out of the serial scan chain plus one or more normal mode clocks.

2.4.1 Built-In Logic Block Observation Technique

The built-in logic block observation (BILBO) technique was proposed by Koenemann et al[54]. In a BILBO design, each storage element (e.g. flip-flop or shift register latch) is configured as a controllable shift register (CSR) cell. By specifying the

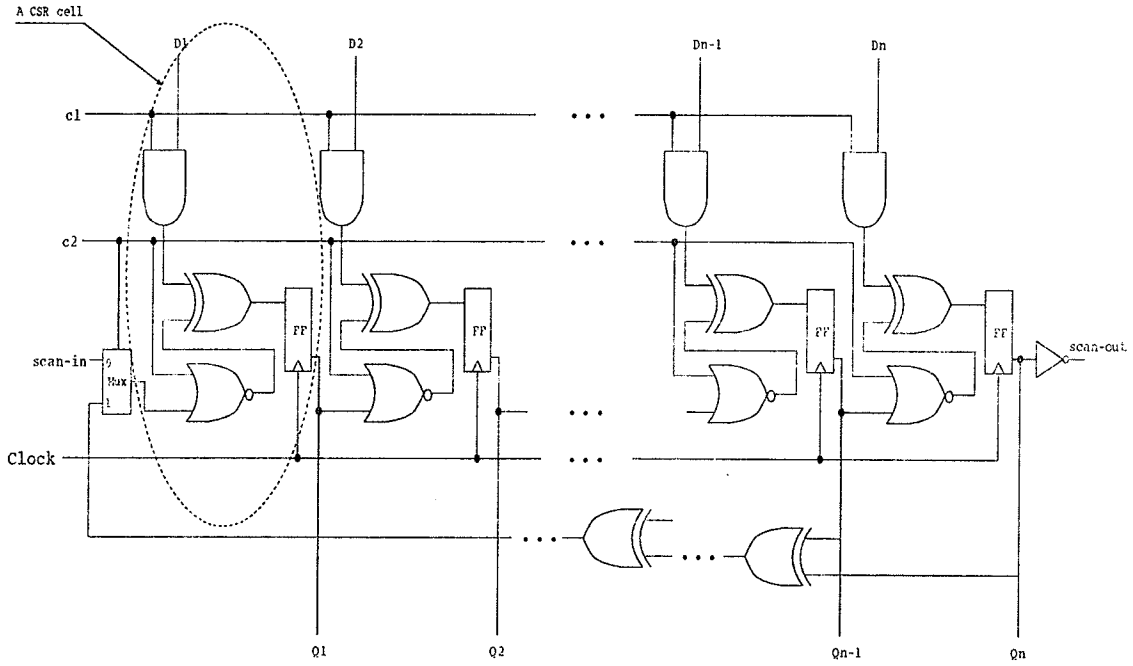


Figure 2.11: n-bit controllable shift registers functioning in any one of four modes specified by the control signals c_1 and c_2

control signals c_1 and c_2, the CSRs as shown in figure 2.11 can be placed in any one of four modes including normal operation, an n stage LSFR, an n stage multiple input signature register, and resetting all flip-flops (or shift register latches) to a logic 0 value.

As shown in figure 2.12, for a sequential circuit design with the BILBO technique, two BILBO CSR A and CSR B blocks are required. When both CSR A and CSR B blocks work in normal operating mode, the non-overlapping clocks *Clk A* and *Clk B* can be used to create a master-slave flip-flop. In the test session, the CSR A are configured as a LFSR pseudo-random pattern generator and the CSR B are configured as a multiple input signature register to capture and compact the responses of the combinational logic circuit.

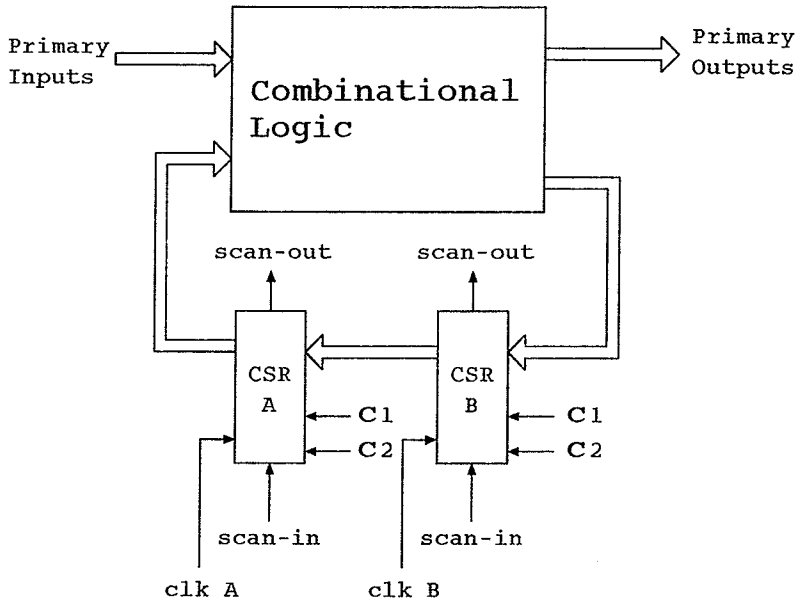


Figure 2.12: A sequential circuit design with the BILBO technique

The advantage of the BILBO technique is that it is an efficient test-per-clock test. A disadvantage is that there is more delay between the outputs of the CSR A and the inputs of the CSR B since each input of the CSR B must pass through an AND gate and an XOR gate before being applied to the flip-flop input. Also, a significant area overhead is required to implement the CSR cells.

2.4.2 Simultaneous Self-Test Technique

The simultaneous self-test (SST) scheme is a built-in test approach which can also significantly reduce test time[2]. During the SST test session, the scan path is first loaded with seed data, and test data need not be continuously shifted into the scan path. In the double-latch design SST scheme as shown in figure 2.13, the shift register latch chain in the feedback path simultaneously performs both test pattern generation

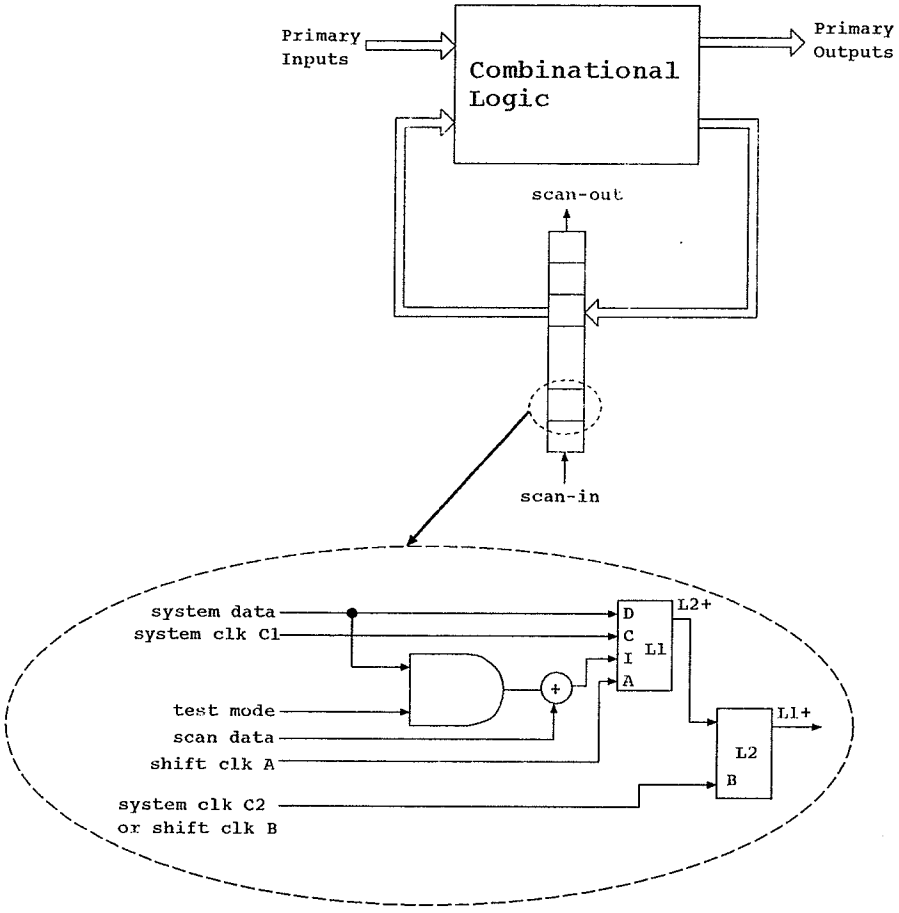


Figure 2.13: Double-latch design simultaneous self-test BIST architecture

and response data compaction in the test session. Since each A/B clock cycle is a test, the SST test is a test-per-clock scheme. Separate pattern generators and signature analyzers are not explicitly required in the SST scheme as each shift register latch simultaneously collects test results from the logic driving its system data port and supplies test patterns through its $L2$ output during the test session. At the end of the test, *test mode* is turned off and the contents of the SRLs in the scan path are shifted out as the signature to be compared with the fault-free signature values.

The area overhead of the SST scheme is low. However, evaluating the test quality

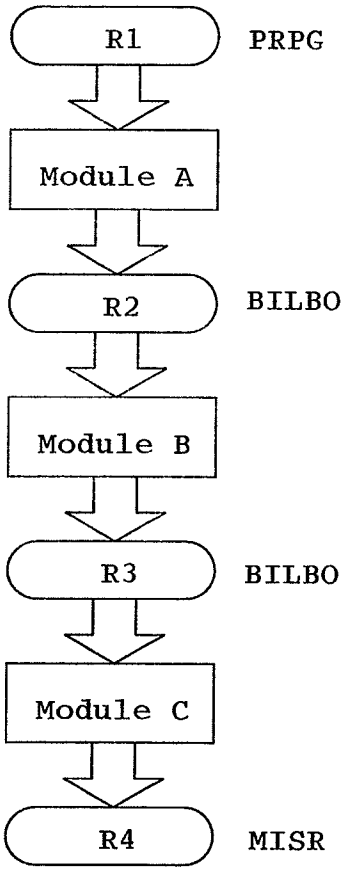


Figure 2.14: Pattern generator driven BIST architecture

of the SST scheme becomes difficult since fault simulation of a sequential logic circuit is required. Error cancellation and masking can occur at the shift register latches in the SST scheme[3], comparable to that of other BIST schemes.

2.4.3 Pattern Generator Driven Scheme

In the pattern generator driven (PGD) test scheme as shown in figure 2.14, a circuit is partitioned into a collection of combinational logic (CL) modules such that each CL module is separated from other modules by register latches[55]. Those register

latches (R_1) which are not at the output of any CL module are configured as complete pseudo-random pattern generators (PRPG) such as LFSRs or CAs, and all other register latches that act as output register latches (R_2, R_3, and R_4) for some CL module are configured as MISRs. Some MISRs (R_2 and R_3) may simultaneously act as test pattern generators for other CL modules. All the MISRs are assumed to implemented using primitive polynomials in the PGD test design.

In the PGD test design, the MISRs can be fed from primary outputs back to the inputs of some CL modules and primary inputs. That is, a circular path test can be implemented.

Kim et al[55] pointed out that the probabilities of input patterns to a MISR (e.g. R_2 or R_3) do not affect the number of different patterns generated by the MISR (e.g. R_2 or R_3). They also conjecture that the average number of different patterns among m patterns generated by an n stage MISR (e.g. R_2 or R_3) is

$$2^n[1-(1-\frac{1}{2^n})^m] \tag{2.2}$$

That is, the properties pertaining to the randomness of the patterns generated by an MISR are true even if the inputs are non-equiprobable. The PGD test scheme is an efficient test-per-clock BIST scheme.

2.4.4 Circular Self-Test Path Technique

The circular self-test path (CSTP)[56][57] test scheme is similar to SST[2]. Figure 2.15 illustrates a general circular self-test path BIST architecture. In the CSTP test, a set of register latches (e.g. R1, R2, R3, R5, R6, and R7 as shown in figure 2.15) are selected and connected as a feedback shift register implementing the polynomial

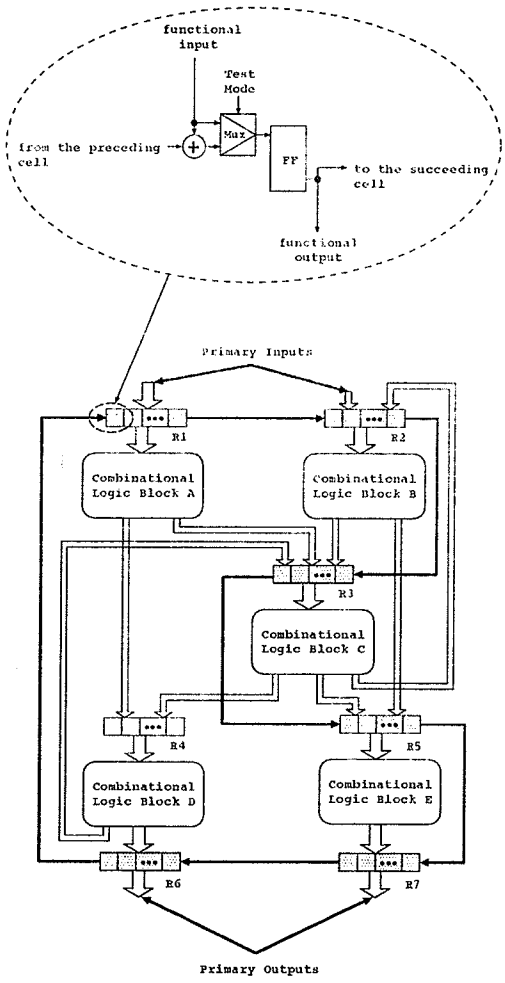


Figure 2.15: Circular self-test path BIST architecture

$1 + X^k + f(X)$, where k is the number of selected register latches in the circuit, and $f(X)$ is the nonlinear feedback function that is decided by the combinational logic blocks (e.g. Block A, Block B, ..., Block E). CSTP may be employed as a partial scan technique, that is, not all register latches (e.g. the register block R4 as shown in figure 2.15 is a conventional register block) need to be configured as self-test cells. In CSTP, explicit test pattern generators and test response compactors are not needed since the register latches in the CSTP path will generate test patterns and also perform test

response compaction. Compared with the conventional BIST schemes, CSTP is a low area overhead test scheme, and can span significant portions of a chip. Also, very high fault coverage can be achieved in the CSTP test scheme[58]. In CSTP, the aliasing probabilities can be set very small by reading more bits of the register latch contents in the CSTP path as the signature. The necessary features of a CSTP scheme is that the registers fed by primary inputs and the registers driven by primary outputs should be included in the CSTP path, and all storage cells must be initializable to a known state before testing.

2.4.5 STUMPS Architecture

In 1982, Bardell and McAnney[59] proposed a scan-path BIST architecture called STUMPS (self-test using MISR and parallel sequences) for a LSSD-based multiple chip system. The STUMPS structure as shown in figure 2.16 consists of two additional register blocks, a parallel random pattern generator (PRPG) and a MISR. The PRPG is a linear feedback shift register. Through an XOR gate network, the PRPG can provide the proper outputs to each shift register latch scan chain. The STUMPS configuration allows for on-module test application and result collection with control and clocking from an external source (test system). The STUMPS scheme can perform tests on all the chips simultaneously, or on one particular chip by selectively de-gating all chains but one into the MISR[60].

The STUMPS scheme has resulted in excellent test coverage, diagnostic, and test throughput for testing the IBM ES/9000 System[60]. STUMPS requires only minimal additional I/O for the test mode selection. As the test chip is inserted into the scan path of the combinational logic under test, there is no adverse effect on critical paths

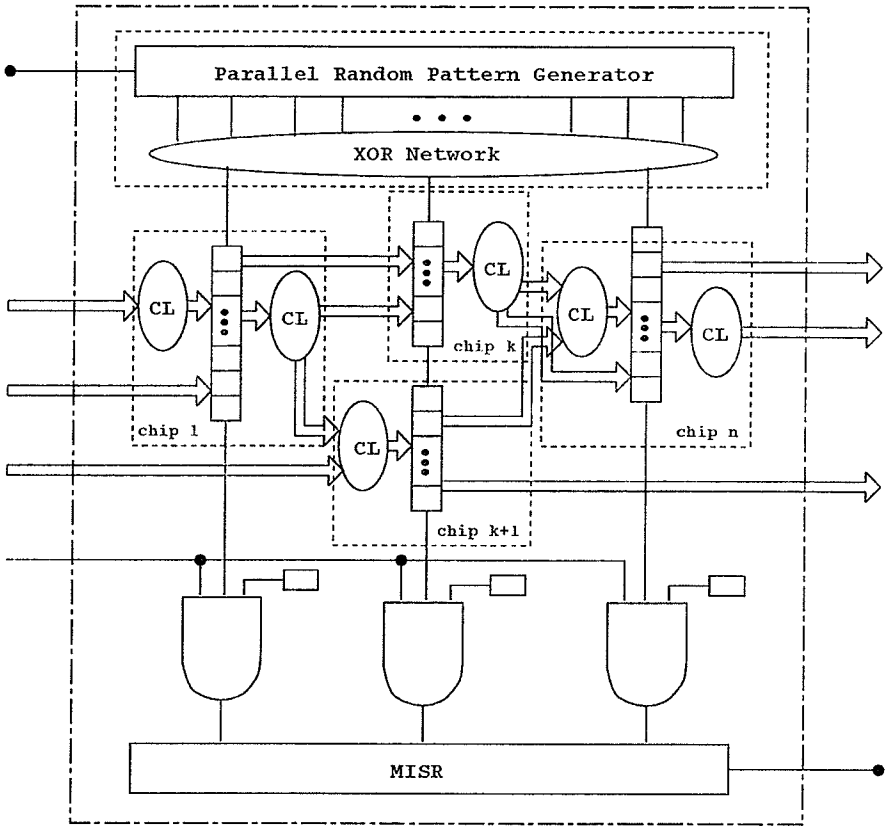


Figure 2.16: STUMPS BIST architecture

of the system[2].

2.4.6 Programmable Weighted Pattern Test Design

In general, BIST is random-pattern testing combined with compacting output responses in signature registers. The off-chip procedure of test pattern generation is eliminated in a BIST scheme. However, random-pattern testing is often unable to consistently achieve the required fault coverage with a reasonable number of test patterns. This deficiency is due to a lack of control of the random patterns to be directed toward the desired test coverage goal[4]. The use of weighted random-pattern test-

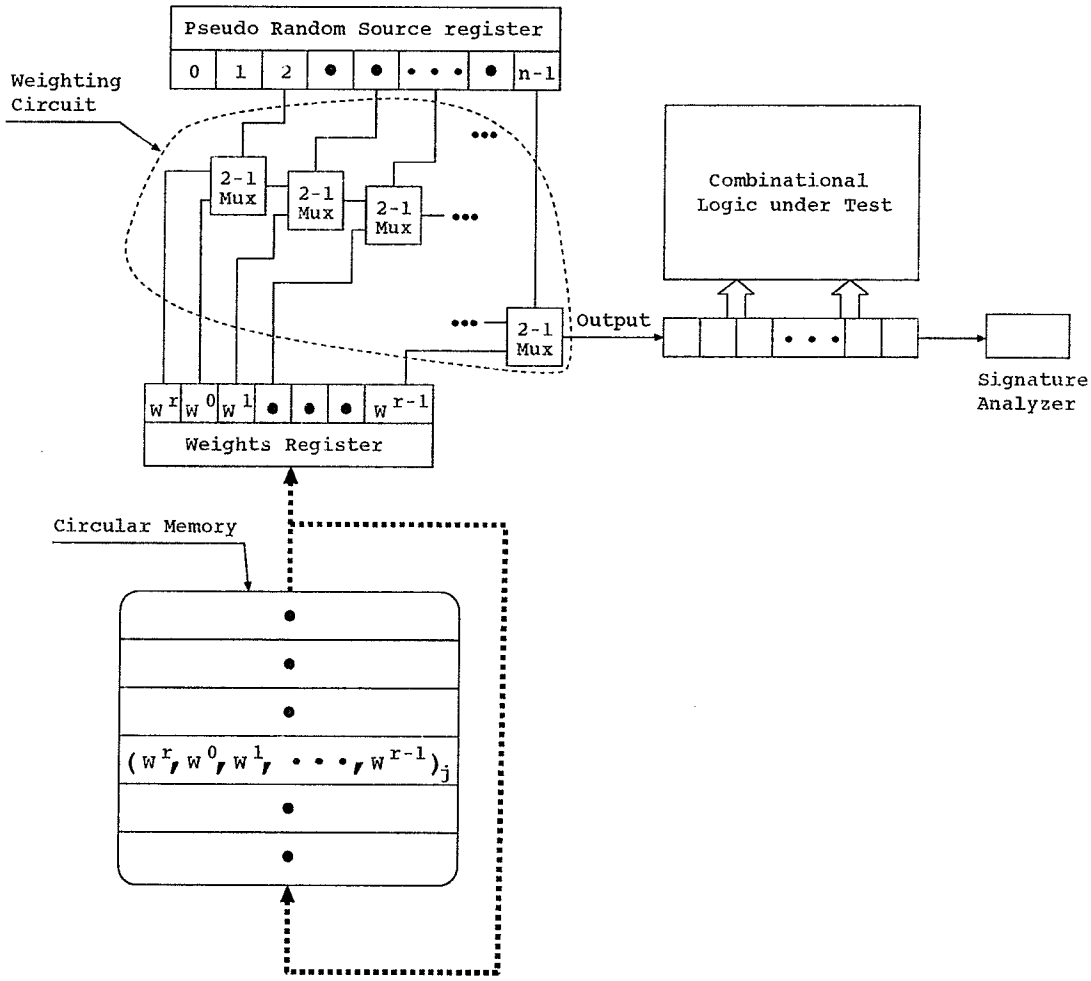


Figure 2.17: Programmable weighted pattern test scheme

ing is a means to control a random test to achieve maximum test coverage with a relatively small number of patterns.

The concept of weighted random patterns was introduced in the early 1970s and has undergone extensive investigations[61][2][4][45].

Figure 2.17 illustrates an efficient weighted pattern generation hardware system[45]. It consists of a circular memory which supplies multiple sets of weights, a weight reg-

ister which controls the weight or signal probability of the output, a source register that can be an n stage LFSR or CA, and a weighting circuit that generates various required weights. As reported in [45] the weighted pattern generation hardware system illustrated in figure 2.17 is efficient in testing random resistant faults, and can quantize weights uniformly to any desired precision.

However, calculating optimal multiple sets of weights for combinational logic circuits is very difficult. It has been determined that it is *NP-hard*[62]. Usually, approximation heuristics are used to obtain an acceptable set of the multiple weights[63][2][62][64].

2.4.7 Re-seeding and Re-programming LFSR Test Scheme

Although random resistant faults can be efficiently tested with a weighted pattern generator, the hardware cost in configuring the weighting circuitry and storing multiple sets of weights could be very high. It was pointed in [49] that many designers are unwilling to pay more than an additional 1% in chip area for better test coverage after having paid a price for implementing scan design and boundary scan to support system level self-test.

In 1991, Konemann proposed a re-seeding LFSR pseudo-random generator scheme as shown in figure 2.18 to achieve a better test coverage[49]. In his proposed scheme, the "programmability" of LFSR pseudo-random generator hardware is utilized. In the first stage, 100K-200K pseudo-random patterns are applied to the circuit without re-seeding the LFSR to achieve an initial test coverage of 90% - 95%. Then, precalculated seeds are loaded into the LFSR between test patterns to influence the bit

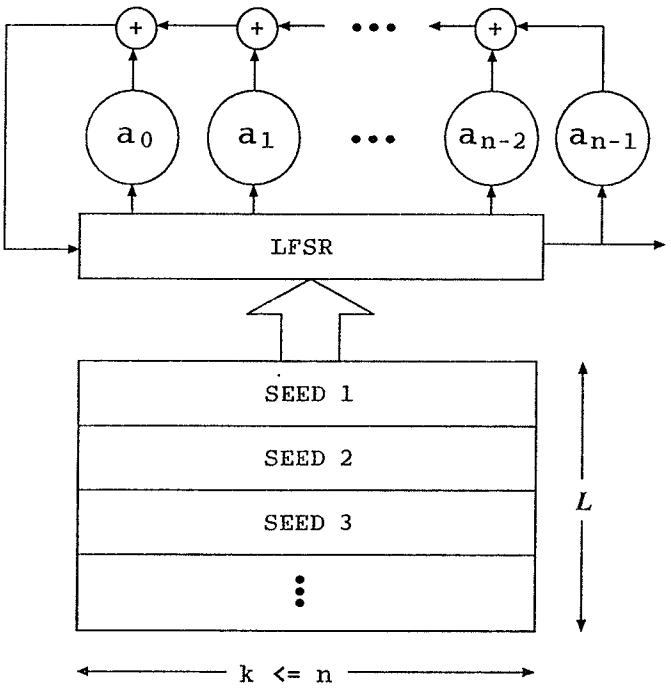


Figure 2.18: Re-seeding an n stage LFSR pseudo-random generator test scheme

values in the generated test patterns for better fault detection.

Typically, not all the bits in a test vector for a fault are "care" bits that determine a test. By utilizing linearity in the LFSR pseudo-random generator function, the preferred seed values can be calculated to result in loading the proper "care" bit values into the shift register latches in the scan chains. Konemann has given an upper bound on the probability finding no proper seeds for a fault test vector with m "care" bits and an n stage LFSR pseudo-random generator as follows:

$$P(m,n) = \prod_{i=0}^{m=1} \frac{(2^n - 1) - (2^i - 1)}{(2^n - 1) - 1} \tag{2.3}$$

Based on the above equation, Konemann estimated that the LFSR generator

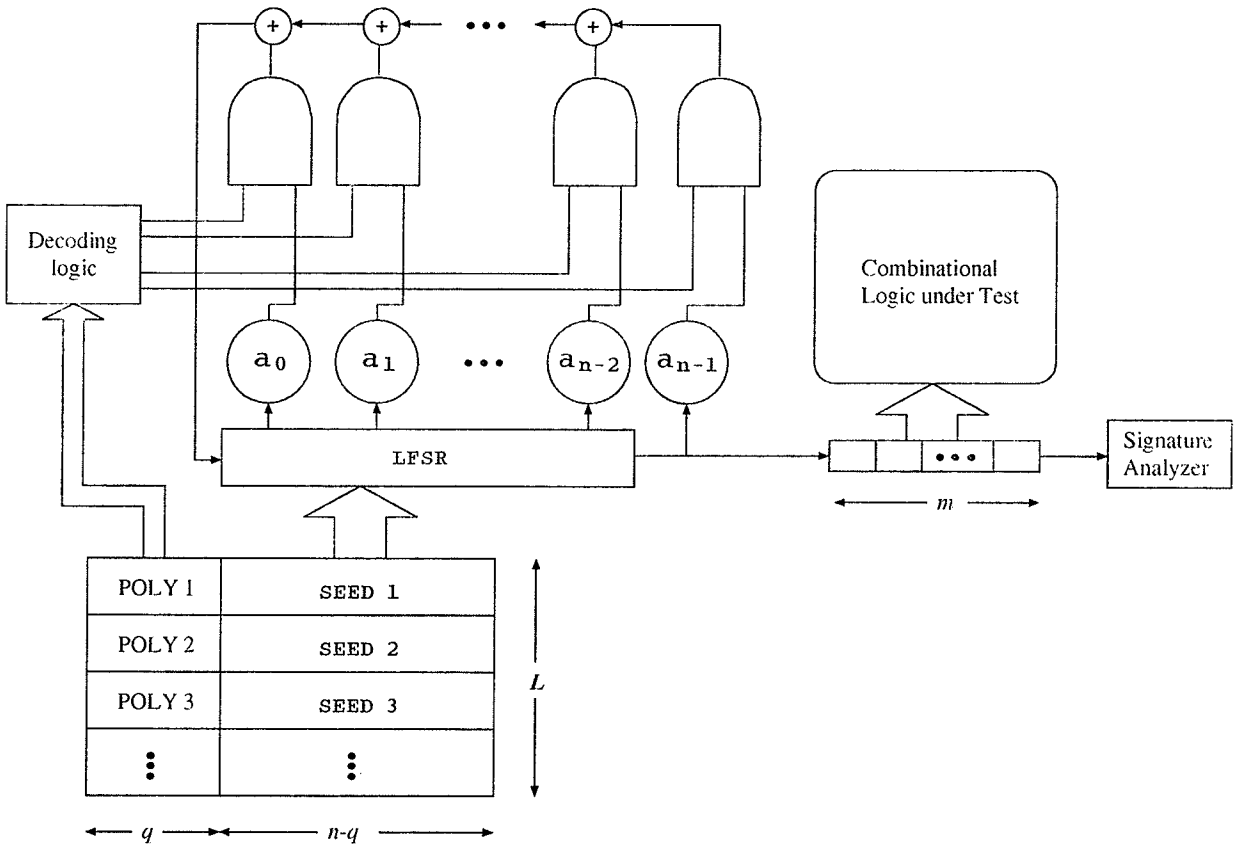


Figure 2.19: Re-seeding and multiple programming in an n stage LFSR pseudo-random generator test scheme

should have $s + 20$ stages in order to reduce the probability finding no proper seeds for a test vector with s "care" bits to less than 10^{-6}.

In 1992, Hellebrand et al significantly extended Konemann's re-seeding of an n stage LFSR generator test scheme to the re-seeding and re-programming of an n stage LFSR generator as shown in figure 2.19[50]. In their scheme, only a $s + 4$ stage LFSR is required to reduce the probability of finding no proper seeds for a test vector with s "care" bits to less than 10^{-6}, where s is the number of "care" bits in a test vector used for re-seeding, and 4 bits are used to identify one of 16 polynomial

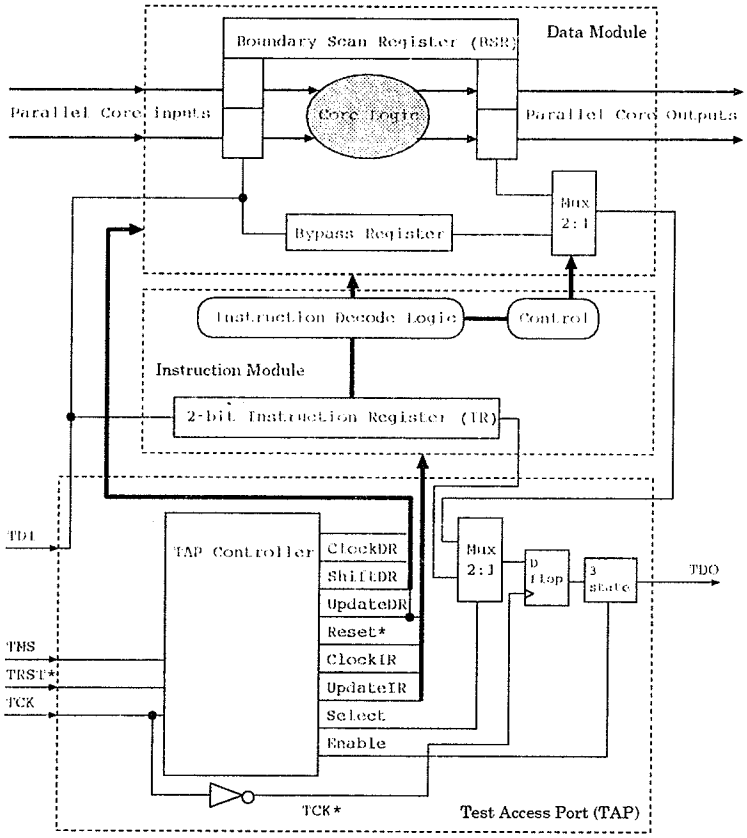


Figure 2.20: IEEE 1149.1 boundary scan architecture

implementations[50].

2.5 Boundary Scan Test Design

Scan based design techniques can also be used at the printed circuit board (PCB) level[66][67][68][70] for simplifying the test of the PCB and for system level integration. For multiple chip module design, the use of a central controller is advised rather than implementing a controller in each chip to simultaneously activate self-test on all chips to detect and isolate faulty chips on the module[72]. For these purposes, the

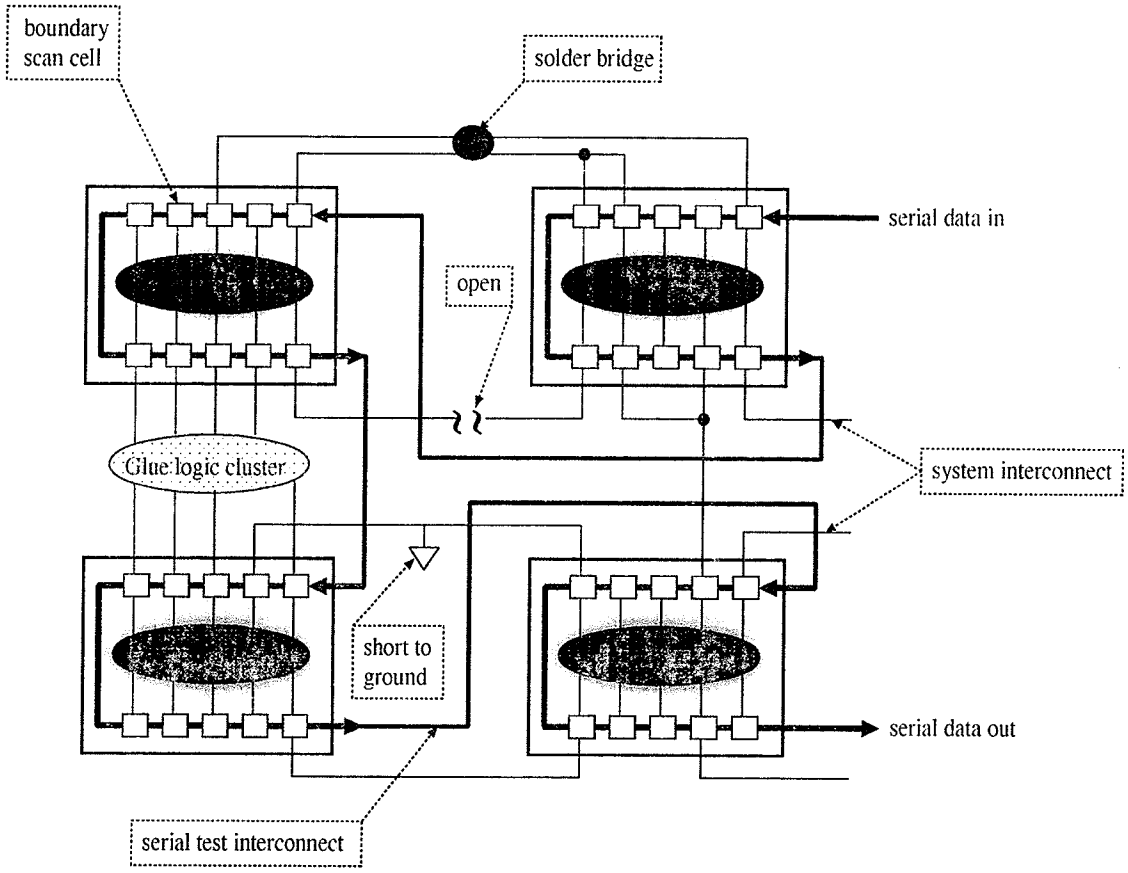


Figure 2.21: Board level fault testing with the boundary scan design

boundary scan test has been proposed[155][67][68]. The IEEE 1149 boundary scan standard architecture as shown in figure 2.20 has been established[69]. The boundary scan template has three principal tasks. It allows the circuit to function normally, allows data to be shifted in or results to be shifted out, and it can conduct several circuit tests. The boundary scan template supports several models, external test (PCB interconnection test), internal test (internal logic circuit test), bypass mode, and BIST within each chip[66]. Users can define their own instructions to provide other testing functions within the boundary scan architecture[68][72].

Figure 2.21 illustrates the PCB board-level failures (e.g. solder bridges, open faults, and shorts to ground, etc.) that can be tested with the boundary scan architecture.

2.6 IDDQ Testing

The IDDQ testing technique is a parametric test that can be used to detect bridging, stuck-on, and gate-oxide faults in CMOS designs[73][74][75]. By measuring the quiescent current (I_{ddq}) consumed in a circuit after applying a test pattern, bridging, stuck-on, and gate-oxide faults in the circuit can be detected. Since the quiescent current will be very low in a good fully complementary CMOS circuit, the quiescent current of the circuit will be abnormally high if bridging, stuck-on, and gate-oxide faults exist in the circuit[74].

In built-in IDDQ testing, built-in circuitry monitoring the quiescent current is added to the design, and a test pattern generation procedure is required in order to stimulate the faults[74][75][76]. IDDQ testing consists of applying the test vectors, allowing the signals to settle, and then measuring the quiescent current. As current measuring is slow, the IDDQ test must be run slower than normal operations, thus increasing the test time.

The defect (or fault) coverage of IDDQ testing is often reduced by the use of dynamic logic. In dynamic logic, the output node of a gate is not always driven by a path to V_{DD} (the source) or V_{SS} (the ground); instead, the capacitance of the node is used to store a value. This can make it difficult to detect bridging faults since a bridge is detected by setting the two nodes to opposite values, thus creating a path

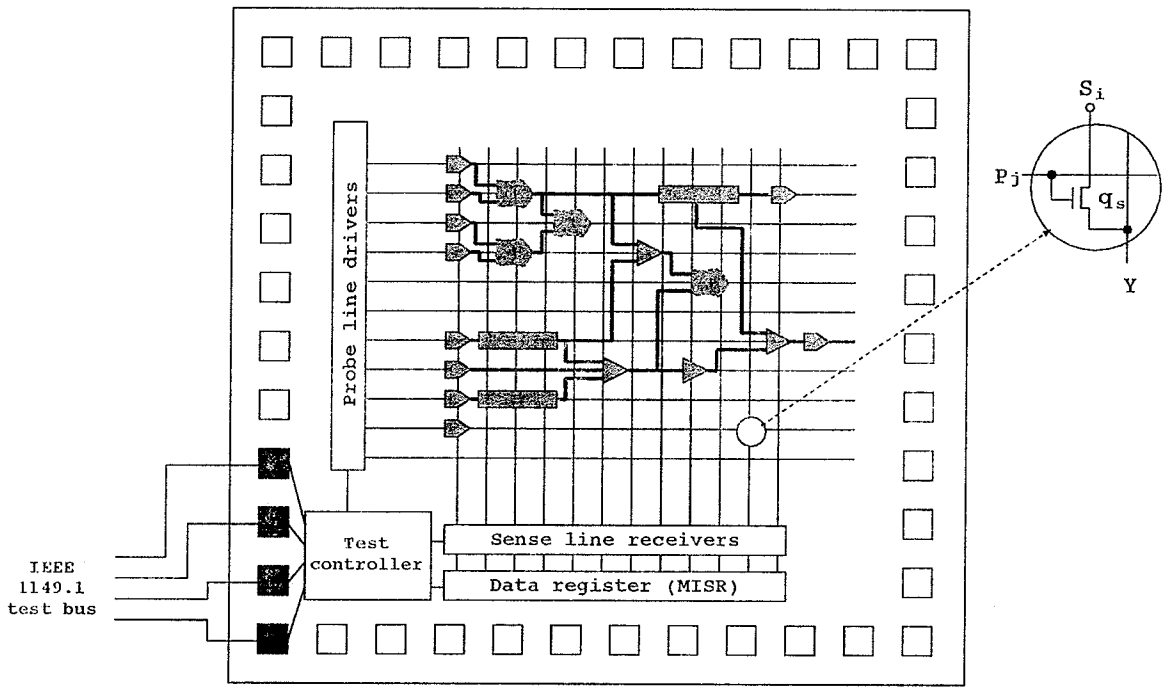


Figure 2.22: CrossCheck test architecture

from V_{DD} to V_{SS} which causes a current drain. A bridge between two nodes driven by dynamic logic will never create such a path, and cannot be detected as an IDDQ fault. Also, IDDQ testing can not be used to detecting stuck-open faults[74].

2.7 CrossCheck Test Design

In the crosscheck design[77], the test structures are embedded into the logic circuit. These test structures add massive observability to the design, and provide controllability to specialized storage cells. As shown in figure 2.22, the crosscheck test architecture consists of a test point array, a test controller, and specialized storage elements called the cross-controlled latches (CCL).

The test point array includes a probe and sense line grid embedded in the logic

circuit structure, with the probe and sense lines orthogonal to one another. At the intersection of each probe and sense line is a very small sense transistor (e.g. q_s). The probe line connects to the gate of the sense transistor q_s, and the sense line connects to the drain of the sense transistor. One terminal of the sense transistor can connect to internal signal points for observing internal logic values as well as injecting signals back into the logic circuit. The test controller receives instructions over a four-pin test bus, controls the activation of probe lines, and sends/receives test data over the sense lines. The CCL cells are connected to the sense transistors, and their values can be observed and injected through the sense transistors.

In the control mode, the crosscheck structures provide write access to all the internal storage elements. In the observation mode, all the internal circuit values can be observed by addressing the probe lines. The probe lines activate row sense transistors, allowing transfer of the internal logic values from the macrocells onto sense lines. From there, the values are received by the data register (MISR) for signature compaction.

The crosscheck test technique incurs a very small performance overhead at the expense of high area overhead. Due to the massive observability, the crosscheck test technique can achieve high fault coverage for various transistor defects[77].

2.8 Evaluating BIST Design

Since the quality of a BIST design depends on the test pattern generation and the test response compaction. It is crucial to have tools to assess test pattern generators and test response compactors efficiently.

Fault simulation is an important tool in evaluating a test set or in assisting in the generation of test patterns. The effectiveness of a BIST design can be reported by a fault simulator, that is, a fault simulator could be used to evaluate fault coverage for a given test set and determine if aliasing occurs during compaction. However, fault simulation is quite expensive for large scale designs or large pattern sets in reporting fault coverage or aliasing of detectable faults. This is particularly true for the SST, PGD, and CSTP test schemes since sequential fault simulation is required.

2.8.1 Fault Simulation

Fault simulation involves fault specification, fault insertion, fault-effect generation, fault detection, and fault dropping[3]. Fault specification defines the fault model and performs fault collapsing. Fault insertion determines which faults in the circuit are to be simulated. Traditional stuck-at fault simulation techniques are classified as parallel, deductive, or concurrent fault simulation[71][78][79]. These methods suffer from wasted computing during the simulation process, and severe fault list-management problems that may degrade performance and require excessive memory utilization.

A more efficient and fast fault simulation technique[80] for combinational logic circuits was proposed in 1985. It is called the parallel-pattern single-fault propagation (PPSFP) method. The PPSFP technique takes maximum advantage of parallelism without wasted calculations. The PPSFP technique performs a fault-free simulation with machine-word length parallel patterns and fault simulation from faults remaining on the fault list in each pass. As more faults are detected, fewer fault simulations are required in each pass. The PPSFP technique is much faster than traditional parallel, deductive, or concurrent fault simulation methods. The PPSFP method has been

further improved and sped up by the techniques reported in [81][82][83].

The PPSFP technique has relieved the difficulty of evaluating fault coverage in the conventional BIST designs. However, difficulties still exist in evaluating aliasing occurring in a BIST design and fault coverage analysis in the SST, PGD, and CSTP schemes since extremely arduous computation effort is needed to simulate aliasing problems for all faults considered. Also, the PPSFP method is limited to combinational circuit fault simulation. Fault simulation for circular path or feedback sequential circuits is still computationally expensive.

2.8.2 Fault Emulation

Contrary to software fault simulation, field-programmable gate arrays (FPGA)[84] [85] based hardware emulation is very promising in evaluating fault coverage and aliasing due to faults for both combinational and sequential logic circuit designs[86]–[89]. Hardware emulation can run at very high speeds – almost in real time, and emulation time is not as affected by the complexity of circuits. Hardware emulation[86] can be used to efficiently and experimentally analyze the effectiveness of the SST, PGD, and CSTP test schemes. A brief overview of our early work using FPGAs used for hardware emulation follows.

Configurable Array Logic: Algotronix CAL1024

The CAL1024 (Configurable Array Logic) from Algotronix[84] is a regular programmable array. This architecture contains 1024 identical logic cells arranged in a 32-by-32 matrix. At the boundary of the chip, 128 programmable I/O pins allow for

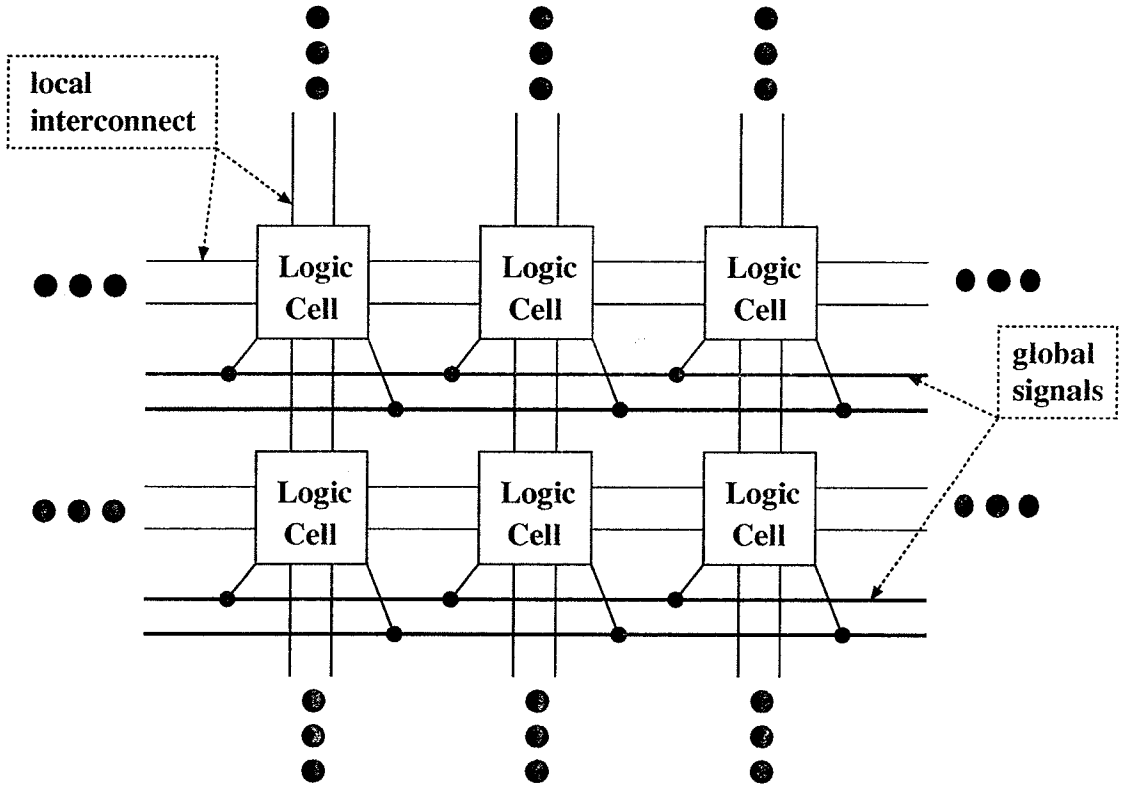


Figure 2.23: Algotronix FPGA chip architecture

cascading the chips in even larger arrays. Figure 2.23 shows the cell interconnection in the CAL1024 architecture. Each cell in CAL1024 is connected to the east, south, west, and north neighbor, and to two global-interconnect signals. Also, each cell receives row select lines and bit lines that are used to program the static RAM bits within the logic cells.

The functions of each logic cell in the CAL1024 chip can be logic 0, logic 1, *BUFFER*, *NOT*, *AND*, *OR*, *NAND*, *NOR*, *XOR*, *XNOR*, D-latch, etc.[84]. Also, each logic cell in the CAL1024 chip allows the user to read back the output of its function block. This allows for continuous monitoring of a cell internally and

for I/O to be freed for other purposes, especially when cascading chips in an array configuration.

CAL1024 Based Hardware Emulator

The hardware emulator is derived from a CAL1024 based computer. The computer is made up of an array of Algotronix CAL1024 chips. The computer is on a board that fits into an AT compatible bus. It enables the user to down-load the design into the FPGA array and run the circuit in almost real time. The user can manipulate the control functions of the chip and the clocking by way of the PC interface[84].

As the emulation is actually occurring in hardware, it reduces the time it takes to perform a fault simulation or generate a signature. Because of the re-programability of the emulator, it is very easy to inject faults in a logic circuit. The CAL1024 allows the user to change individual cells during write cycles. This feature has been manipulated to further expedite the fault injection process, and to enhance test point insertion. The fault injection procedure could be further improved by simultaneously injecting faults which are in independent blocks with the hardware emulator. This is not possible with serial machine simulation.

Stuck-At Fault Emulation Schemes

Figure 2.24 illustrates a basic FPGA based fault emulation scheme. In this scheme, a copy of the design is down-loaded into the CAL1024 chips. The fault-free circuit is emulated and the final signature is stored after a given number of patterns. By comparing the signature of the good circuit and the one with the fault injected,

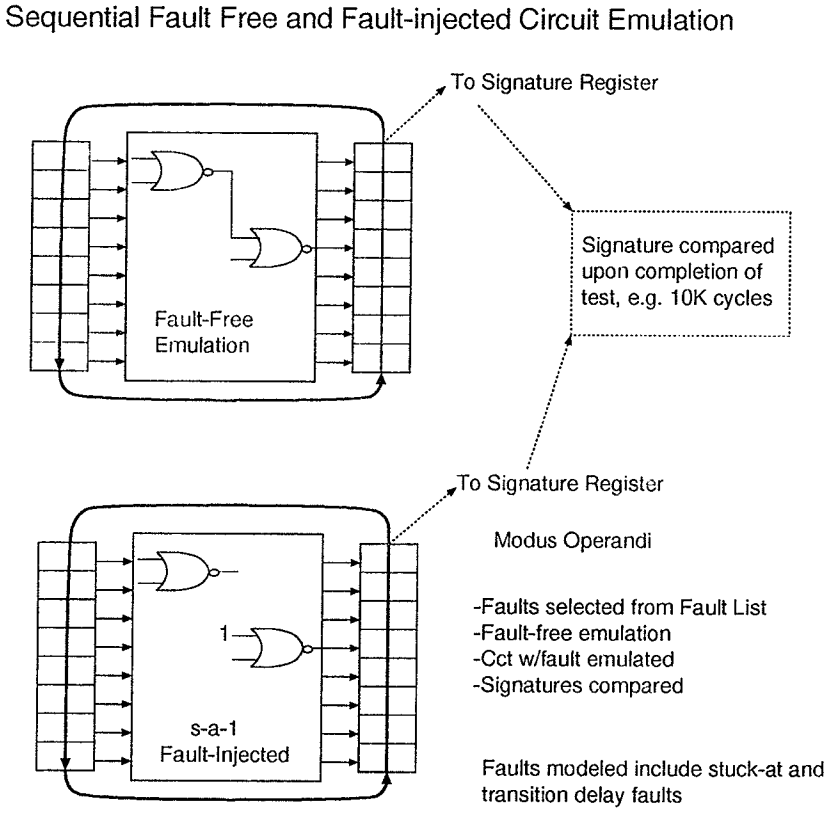


Figure 2.24: FPGA based fault emulation scheme 1

the detection of the fault can be determined. This emulation scheme takes into consideration aliasing, and reports real fault detection results of the test scheme adopted in the design.

Figure 2.25 shows an improved emulation scheme which still allows for maximum sized circuits but involves the recording of multiple signatures. These multiple signatures may be at various iterations of the test cycle. Signatures generated during the emulation of the faulty circuit are compared to the multiple signatures stored. This allows for the fault to be dropped early in the test cycle and reduces the overall test time.

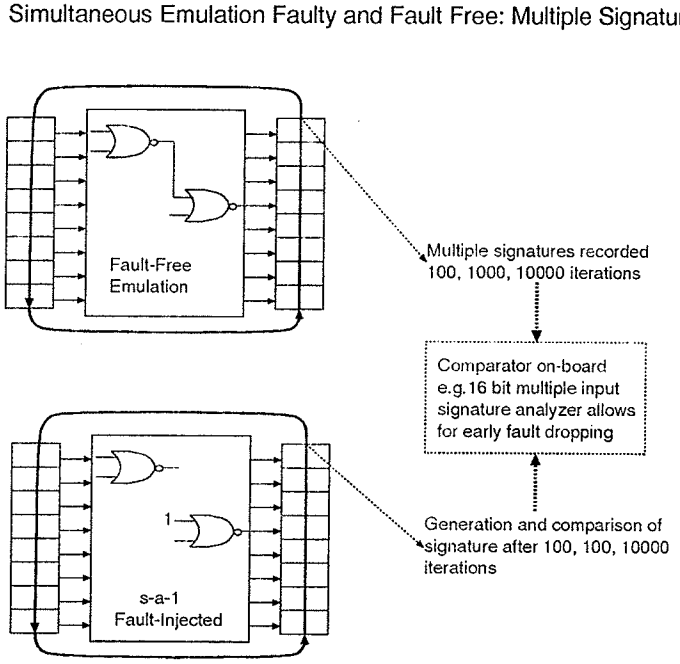


Figure 2.25: FPGA based fault emulation scheme 2

The emulation scheme shown in figure 2.26 allows for immediate fault dropping upon detection of a difference in output between the fault free and faulty circuit. This is accomplished by simultaneously emulating the both the fault free circuit and fault injected circuit. The comparison circuit consists of a logic ORing a number of XOR gates fed by the outputs from both the fault free and faulty circuit. Upon the receipt of a difference, the fault in question can be removed from consideration.

Transition Gate Delay Fault Emulation Scheme

The hardware emulator can also be used for transition delay fault emulation[88]. Figure 2.27 shows an emulation scheme for a slow-to-rise (SR) transition delay fault. The emulation scheme consist of three similar circuits. One circuit is the golden circuit,

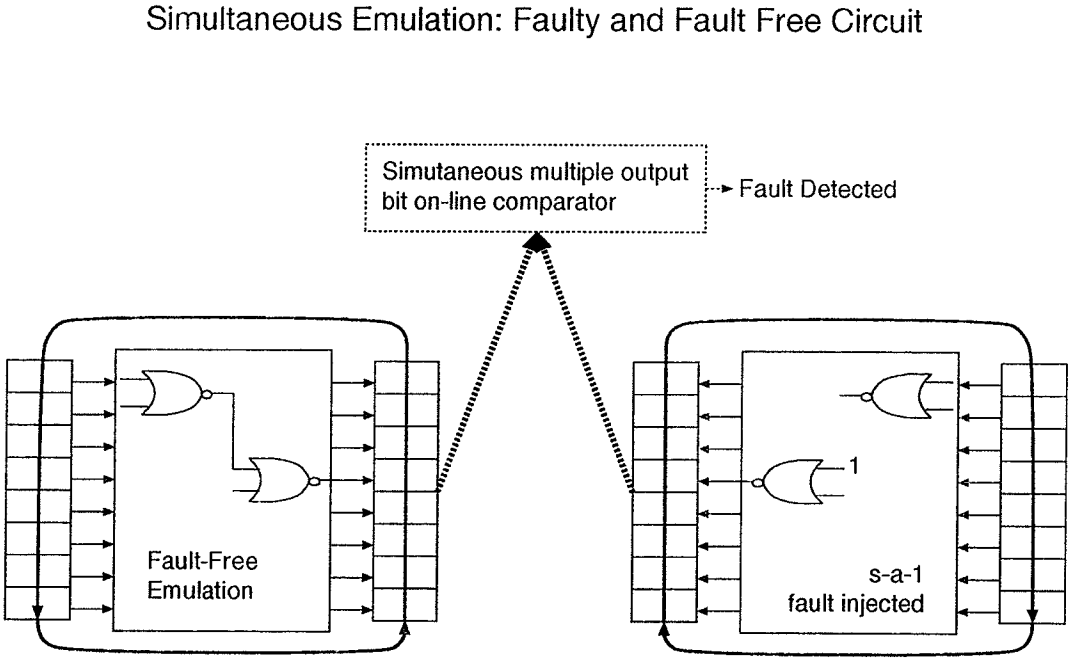


Figure 2.26: FPGA based fault emulation scheme 3

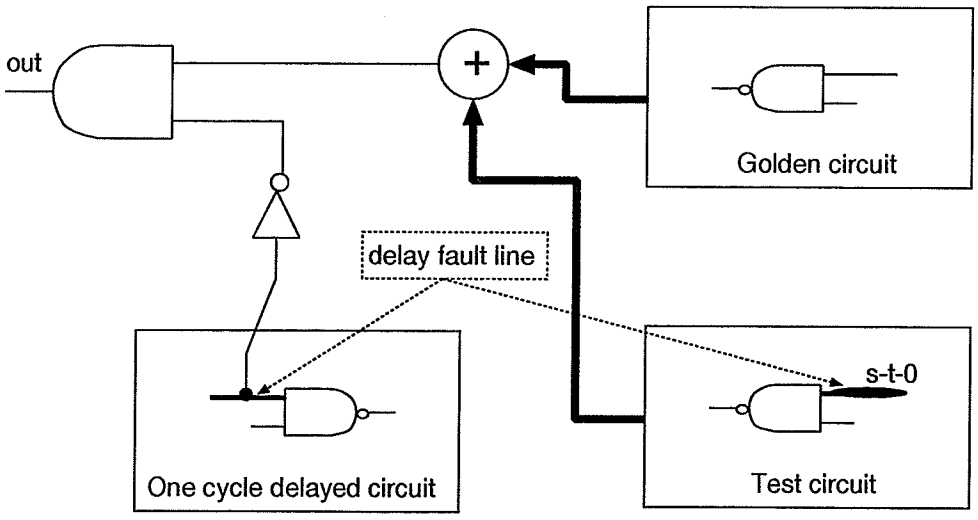


Figure 2.27: Emulation scheme for a slow-to-rise (SR) transition delay fault

another is the circuit onto which the proper stuck-at faults would be implemented. The third circuit is also a golden circuit but is delayed by one clock cycle. When a fault is detected from the test circuit, the one clock cycle delayed golden circuit is observed to find the delayed value of the line under test. In this way if the line under test in the delayed circuit, and the line under test in the test circuit have the same value, the transition delay fault on that line will be detected.

Recently, a hardware emulator configured with multiple chip arrays has been developed[89]. It is based on an Aptix field programmable circuit board (FPCB) G2 on which four Xilinx 4010 chips are mounted.

2.9 Summary

This section has briefly overviewed IC testing with emphasis on BIST schemes. Although not exhaustive, it provides the context and motivation for the following sections.

Chapter 3

Stuck-Open and Delay Fault Test Generation

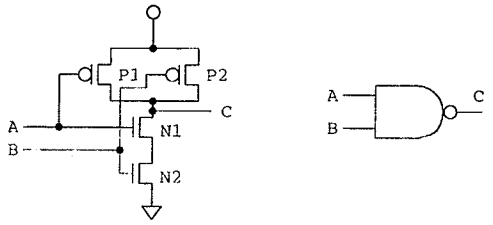
3.1 Introduction

In this chapter, a test pattern generation technique based upon neural network computational models for stuck-open faults and delay faults is presented. The research is an extension of the work of Chakradhar et al[96].

Since the inception of scan path design, testing integrated circuits has been, to great extent, transformed into testing combinational circuits[2]. At present, the most widely accepted fault model is the static stuck-at fault model. However, there are defects which may not be covered by a simple stuck-at test. For example, a transistor stuck-open fault in a CMOS combinational circuit may force the circuit to exhibit sequential behavior[90]. In addition, delays beyond specification may arise due to process variations, stray capacitances, or physical defects[3][16]. These effects in a circuit may cause unacceptable delays in $0 \rightharpoonup 1$ or $1 \rightharpoonup 0$ transition along paths from primary inputs to primary outputs. These faults, referred to as stuck-open

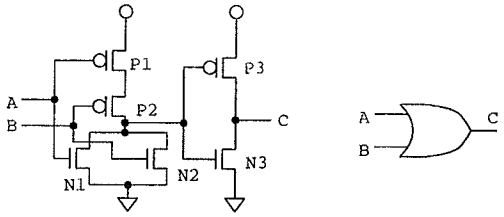
faults and delay faults, require two pattern tests for their detection. The first pattern initializes the transition gate delay fault (or stuck-open fault) at the fault site, and the second provokes the fault and propagates the transition to an output. It has been reported that in high speed CMOS circuits, transistor stuck-on faults can be detected as delay faults[91]. The demand to ensure that a circuit can operate correctly at the desired clock speed has produced a growing interest in stuck-open fault and delay fault testing[16]. In these cases the static stuck-at fault model does not adequately meet the requirements. Since stuck-open faults and delay faults are realistic faults in logic circuits, it is worthwhile considering algorithms and techniques that may be useful in generating test patterns for these non-traditional fault models.

The test pattern generation problem for stuck-at faults is known to be *NP-complete*[92][93], and it is thus unlikely that a test generation algorithm with polynomial time complexity could be found[94]. Test generation for stuck-open faults will be more difficult than for stuck-at faults since pairs of test patterns are required. Hence, test pattern generation for stuck-open faults is at least in the same complexity class, and as such, may lend itself to non-traditional computational models[95]. Neural network models have been discussed recently by Chakradhar et al[96] as a means of determining logical network satisfaction and as a means of determining test vectors for stuck-at faults. These neural networks do not have a learning phase; rather, the dynamics of the neural network is used as a means of computing inputs and outputs which produce minimum energy configurations. Here, a method to extend these test pattern generation techniques to stuck-open and delay faults is discussed. It should be noted however, that in order to be efficient, these types of techniques and applications will rely on neural network hardware or parallel computation as opposed to



stuck-open P1: SF(A), stuck-open N1: SR(A),
stuck-open P2: SF(B), stuck-open N2: SR(B).

(a)



stuck-open P1: SF(A), stuck-open N1: SR(A).
stuck-open P2: SF(B), stuck-open N2: SR(B).
stuck-open P3: SR(C), stuck-open N3: SF(C).

(b)

Figure 3.1: CMOS *NAND* and *OR* gates and their equivalent transition gate delay fault models

simulations on serial machines.

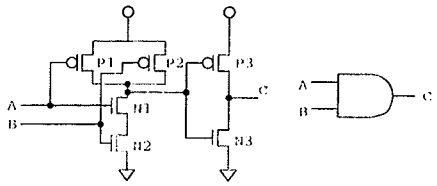
3.2 Stuck-Open Fault Modeling

As testing both transistor stuck-open faults and delay faults requires a pair of patterns, there is a relation between these faults. Only transistor stuck-open faults in full static CMOS gates are considered here. A transition gate delay fault of a gate is a logical model for a defect that delays either a rising or falling transition[13][14].

For each primitive gate (NOT, $NAND$, AND, NOR, OR), slow-to-rise (SR) and slow-to-fall (SF) faults on lines can be used to model CMOS transistor stuck-open faults. Since pairs of test patterns are required to detect a SR or SF transition gate delay fault of a line, test patterns which detect a SR or SF transition gate delay fault of a line will detect the stuck-open fault corresponding to it[97][98][99].

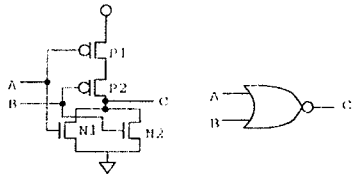
As an example, consider the two input $NAND$ gate and its transistor level model given in figure 3.1(a). We can easily verify that the line SR and SF transition gate delay faults are equivalent to the stuck-open faults indicated. A pair of test patterns *A=10, B=11* which detect transistor $P1$ stuck-open will detect the SF transition gate delay fault of line A. In the $NAND$ gate, SR and SF faults of line C can be represented as broken connection faults with power or ground. Similarly, primitive CMOS AND, NOR, and NOT gates and their equivalent transistor level models are shown in figure 3.2. For primitive CMOS gates, if all SF and SR transition gate delay faults on lines are detected, their stuck-open faults are also detected.

For complex CMOS gates, the gate level modeling technique of Reddy et al[100] is used. The *nFETs* of a complex gate are used to describe the equivalent gate level circuit. Again, SR and SF transition gate delay faults of the lines are used to model the CMOS transistor stuck-open faults. Consider a CMOS XOR gate and its transistor level circuit as shown in figure 3.3(a). The SR and SF transition gate delay faults of the equivalent gate lines are shown in figure 3.3(b). These SR and SF transition gate delay faults correspond to stuck-open faults at the transistor level. If all these SR and SF transition gate delay faults are detectable, all the transistor stuck-open faults will also be detectable. In the gate level circuit shown in figure 3.3(a), lines A and B can be modeled as SR and SF transition gate delay faults



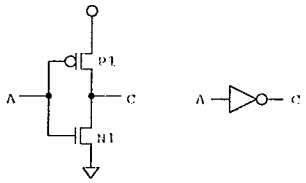
stuck-open P1: SF(A), stuck-open P2: SF(B), stuck-open P3: SR(C).
stuck-open N1: SR(A), stuck-open N2: SR(B), stuck-open N3: SF(C).

(a)



stuck-open P1: SF(A), stuck-open P2: SF(B),
stuck-open N1: SR(A), stuck-open N2: SR(B).

(b)



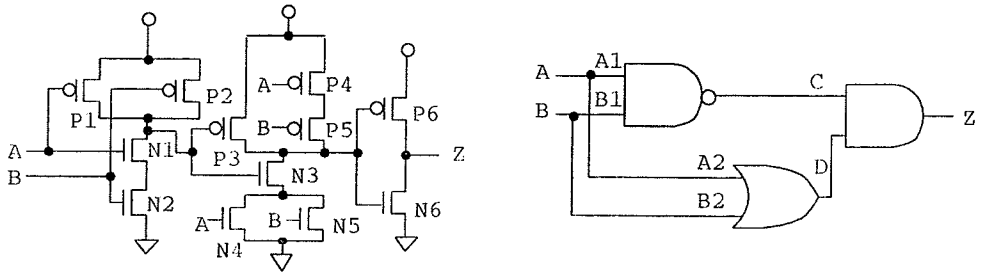
stuck-open P1: SF(A), stuck-open N1: SR(B).

(c)

Figure 3.2: CMOS AND, NOR and NOT gates and their equivalent transition gate delay fault models

from previous level gates, SR and SF transition gate delay faults on line D can represent either $P4$ or $P5$ and either $N4$ or $N5$ stuck-open faults, and SR and SF transition gate delay faults of line Z can model broken connections between the output and power or ground.

As discussed above, when a CMOS transistor circuit is transformed into its gate level circuit, the SR and SF transition gate delay fault models of equivalent gate lines are used to model stuck-open faults. For primitive gates, the SR and SF transition



(a)

stuck-open P1: SF(A1), stuck-open N1: SR(A1),

stuck-open P2: SF(B2), stuck-open N2: SR(A2),

stuck-open P3: SF(C), stuck-open N3: SR(C),

stuck-open P4: SF(A2), stuck-open N4: SR(A2),

stuck-open P5: SF(B2), stuck-open N5: SR(B2),

stuck-open P6: SR(Z), stuck-open N6: SF(Z).

(b)

Figure 3.3: A CMOS XOR gate, its equivalent gate level circuit and equivalent SR & SF transition gate delay faults

gate delay faults of gate lines as shown in figures 3.1 and 3.2 are used to model transistor stuck-open faults, and for complex CMOS gates, a similar procedure for modeling the XOR gate is used to obtain an equivalent gate level circuit and the SR and SF transition gate delay faults of its lines represent transistor stuck-open faults in the complex CMOS gate.

As CMOS transistor stuck-open faults are modelled with SR and SF equivalent transition gate delay fault lines, only SR and SF transition gate delay faults of the lines that correspond to stuck-open faults of CMOS transistors are considered. If

two SR (SF) transition gate delay faults are equivalent, the stuck-open faults corresponding to them are also equivalent. As such, a fault collapsing procedure[101][102] can be used to reduce the number of faults to be considered.

The only difference between testing line SR (SF) transition gate delay faults and testing stuck-open faults is that all line SR (SF) transition gate delay faults have to be considered in line SR (SF) transition gate delay fault testing, and only line SR (SF) transition gate delay faults which correspond to transistor stuck-open faults are required in stuck-open fault testing.

3.3 Neural Models of Gate Level Circuits

3.3.1 Neurons and the Hopfield Neural Network

A simplified neuron model (i.e. artificial neuron) is a processing element (PE) as shown in figure 3.4[103][104]. A neuron may have many inputs from the other neurons, but produces only one output. Connections between neurons are known as synapses. Each synapse has associated with it a weight which represents the contribution that a particular input has towards the neuron output. The neuron performs a simple weight summation of its inputs and may activate its output with neural activation function $f(\cdot)$ which could be a sigmoid function or sign function as shown in figure 3.4. Mathematically, the output of neuron i in figure 3.4 is given by

$$X_i = f(\sum_{k=1}^{N} T_{ji} X_j + I_i) \tag{3.1}$$

A Hopfield neural network is a symmetric fully connected set of neurons, in which the connection weight from neuron i to neuron j is same as the connection weight

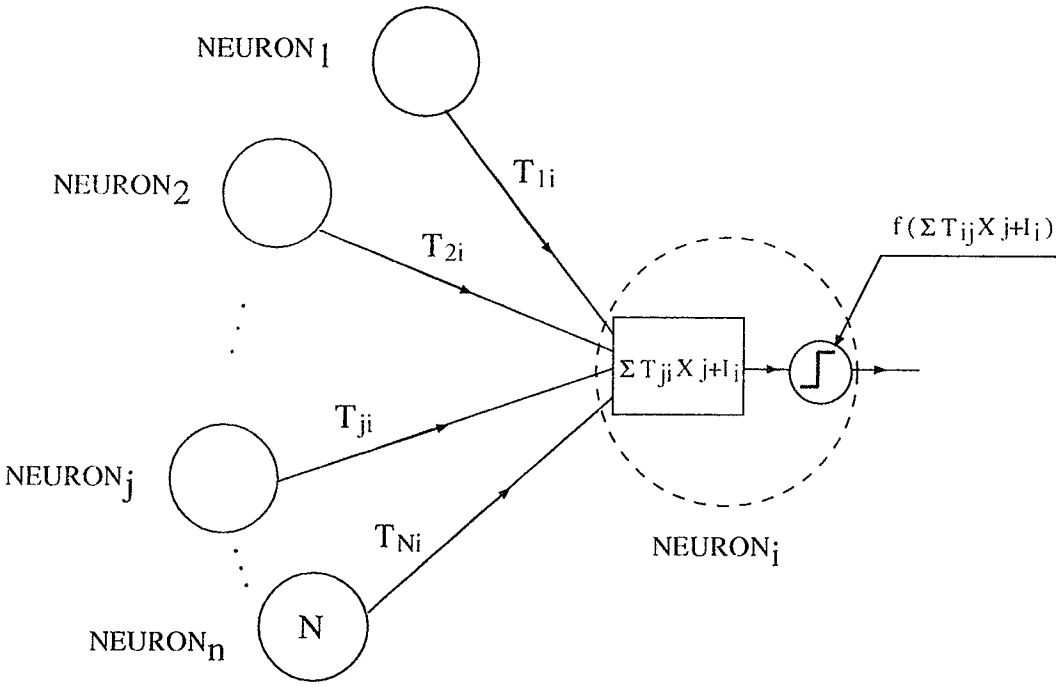


Figure 3.4: An artificial neuron diagram

from neuron j to neuron i, that is, $T_{ij} = T_{ji}$ [105][103][104]. As shown in figure 3.5, a Hopfield neural network has only one layer and symmetric weights for any connection between neuron i and neuron j, and it is a fully connected neural network.

3.3.2 Neural Models of Gate Level Circuits

Neural networks have been used to solve problems from many different fields that are difficult to solve using sequential computer systems[103][104]. Chakradhar et al[96] proposed a technique for automatic test generation for stuck-at faults using a Hopfield neural network[105]. The test generation problem is transformed into a combinatorial optimization problem which can be solved by a Hopfield binary neural network[1]. The

[1]In a Hopfield binary neural network, each neuron has two states (0 or 1).

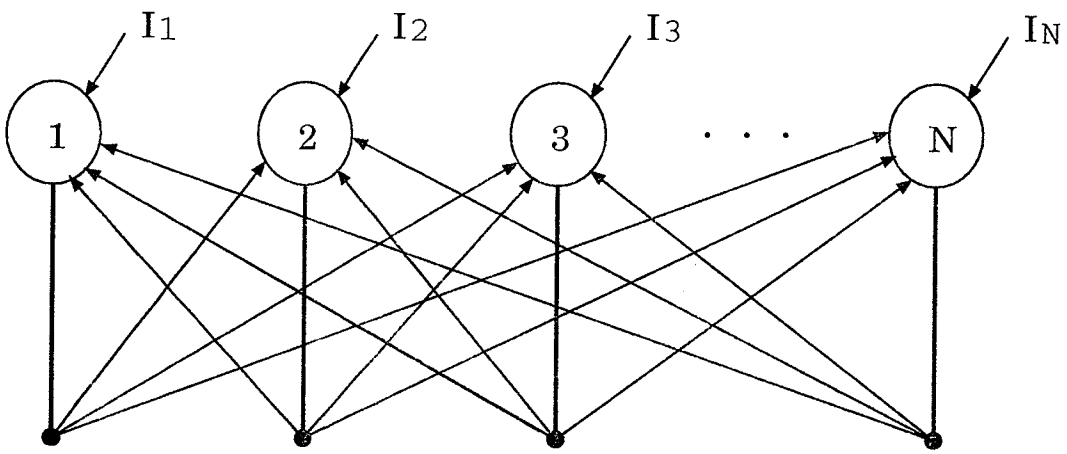


Figure 3.5: A Hopfield neural network

technique is quite different from conventional test pattern generation methods[11].

In the Chakradhar et al's method, primary input (output) neurons correspond to primary inputs (outputs) of the circuit. Each gate is independently mapped onto a neural network and the interconnections between the gates are used to produce a neural network representing the complete circuit. The neural network for a digital circuit is characterized by an energy function E defined as follows:

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} T_{ij}V_iV_j - \sum_{i=1}^{N} I_iV_i + K \tag{3.2}$$

In equation (3.2), N is the number of neurons in the neural network, T_{ij} is the weight between neurons i and j, V_i is the activation value of neuron i, I_i is the threshold of neuron i, K is a constant, and $T_{ii} = 0$. The energy function E will have minimal value only at the neuron states consistent with the satisfaction of all gates in the circuit.

Neural networks for 2-input primitive AND, OR, $NAND$, NOR, and NOT gates constitute the basis set, and the parameters of these neural networks can be deter-

Gate	*T*	*I*	*K*
AND	$\begin{pmatrix} 0 & A+B & A+B \\ A+B & 0 & -B \\ A+B & -B & 0 \end{pmatrix}$	$\begin{pmatrix} -2A-B & 0 & 0 \end{pmatrix}$	K=0
OR	$\begin{pmatrix} 0 & A+B & A+B \\ A+B & 0 & -B \\ A+B & -B & 0 \end{pmatrix}$	$\begin{pmatrix} -B & -A & -A \end{pmatrix}$	K=0
NAND	$\begin{pmatrix} 0 & -A-B & -A-B \\ -A-B & 0 & -B \\ -A-B & -B & 0 \end{pmatrix}$	$\begin{pmatrix} 2A+B & A+B & A+B \end{pmatrix}$	K=2A+B
NAND	$\begin{pmatrix} 0 & -A-B & -A-B \\ -A-B & 0 & -B \\ -A-B & -B & 0 \end{pmatrix}$	$\begin{pmatrix} B & B & B \end{pmatrix}$	K=B
NOT	$\begin{pmatrix} 0 & -2A \\ -2A & 0 \end{pmatrix}$	$\begin{pmatrix} A & A \end{pmatrix}$	K=A

Table 3.1: Parameters of neural network for basis gates

mined by solving the constraint equations derived from the equation given in (3.2). These are summarized in table 3.1, where A and B are positive constants. Neural networks for gates with more than two inputs can be constructed from the basis set. For example, the construction of a neural network for a 4-input AND gate is shown in figure 3.6. A neural network is constructed from the basis set by replacing neurons with identical labels by a single neuron with a threshold equal to the sum of the original neuron threshold and merging identical edges into a single weight equal to the sum of the original weights[96].

Associated with each neuron i in a neural network, is a decision hyperplane which can be defined as

$$I_i + \sum_{j=1}^{N} T_{ij} V_j = 0 \tag{3.3}$$

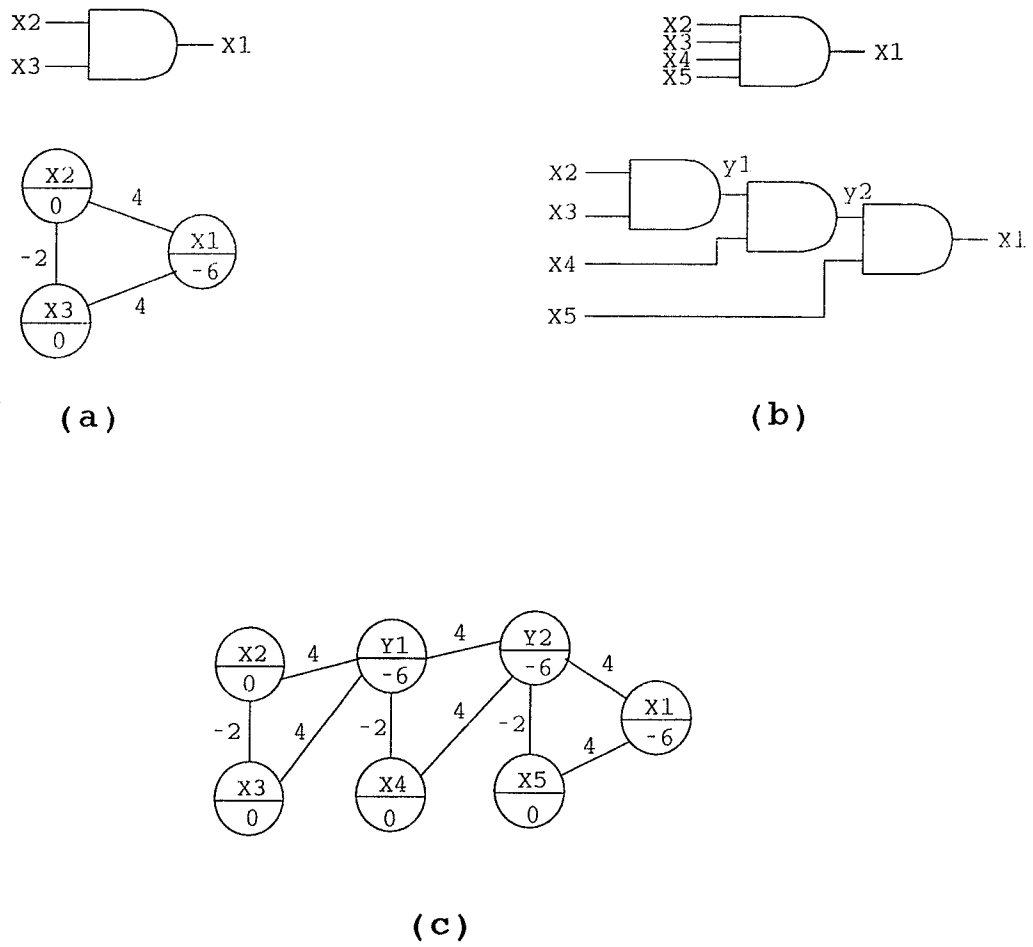


Figure 3.6: A neural network for 4-input AND gate

in an *N-1* dimensional space[96]. There are three sets, P_{i_on}, P_{i_off}, and P_{i_other}, whose elements are the consistent states of the network for each neuron i. A state belongs to $P_{i\_on}(P_{i\_off})$ if it is a consistent state and the neuron has an activation value 1 (0). P_{i_other} consists of all the consistent states which are not in the sets P_{i_on} or P_{i_off}. The decision hyperplane of a neuron i should divide the elements in P_{i_on} and P_{i_off} into different sides of the decision hyperplane, and the elements in P_{i_other} should lie on the decision hyperplane. A theorem[96] is stated below:

Theorem 1 [96]: A necessary condition for the existence of a network of N neurons and an energy function E defined in (3.2) to model a device with N terminals is the existence of a decision hyperplane for each of the neurons.

From equation (3.2) we can calculate the difference between the global energy of a neural network when a neuron i is off and when it is on as

$$E(V_i = 0) - E(V_i = 1) = I_i + \sum_{j=1}^{N} T_{ji} V_j \tag{3.4}$$

For an arbitrary point $s \in P_{i_{on}}(P_{i_{off}})$, neuron i has an activation value 1 (0) in the consistent state S_1 and 0 (1) in the inconsistent state S_2. The E in equation (3.2) should have a lower energy for the consistent state S_1. As such the energy difference given in equation (3.4) will be positive (negative) and the decision hyperplane $I_i + \sum_{j=1}^{N} T_{ji} V_j = 0$ will divide the $N-1$ dimensional space into two regions, i.e. one in which consistent states lie and one in which inconsistent states lie. However, for an arbitrary point $s \in P_{i_{other}}$, the energy difference given in equation (3.4) will not change since the two consistent states lie on the hyperplane defined by $I_i + \sum_{j=1}^{N} T_{ji} V_j = 0$.

3.4 Test Pattern Generation for Stuck-Open or Transition Delay Faults

The stuck-open (or transition gate delay fault) fault test generation procedure is similar to the stuck-at test generation formulation[96]. After a neural network is constructed for a logic circuit, a stuck-open fault is inserted into the circuit. The neural network for the fault injected circuit and the neural network for the fault-free logic circuit are combined through an output interface. The interface facilitates

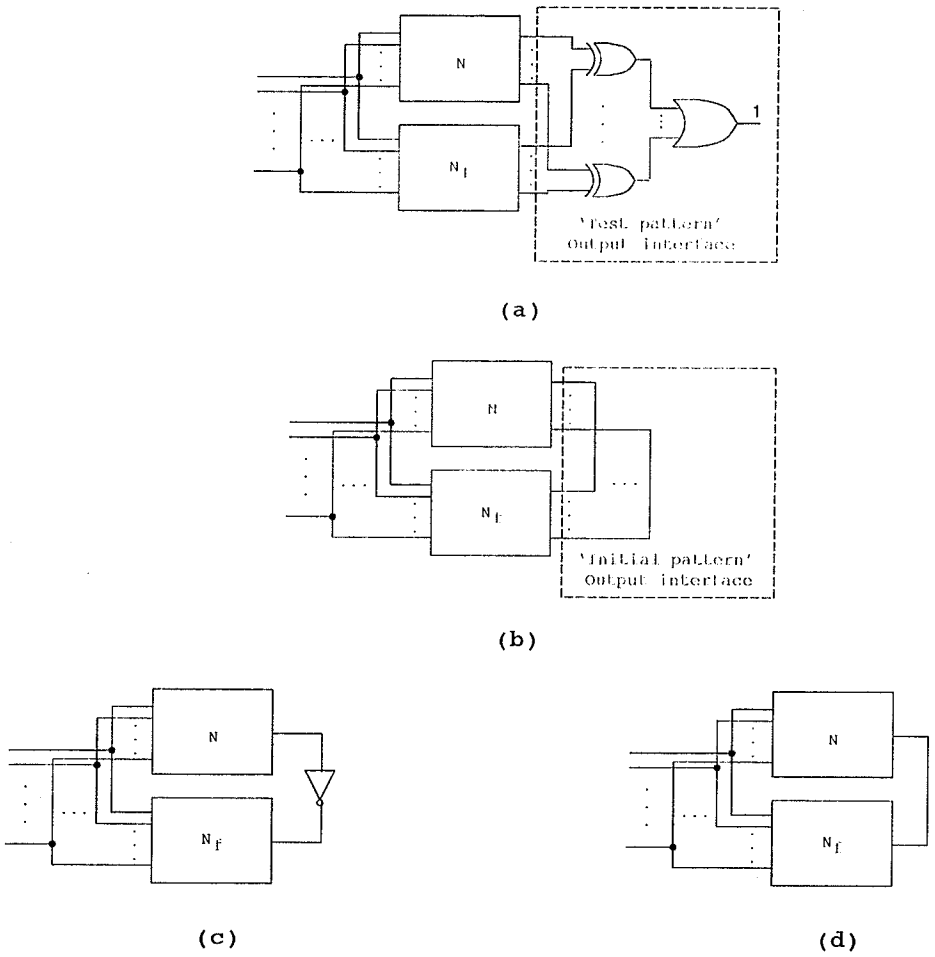


Figure 3.7: Networks for test pattern generation

discrimination between primary outputs of the fault injected circuit and primary outputs of the fault-free logic circuit for the 'test pattern'. In addition, the same value can be forced on the primary outputs of the fault injected circuit and primary outputs of the fault-free logic circuit for the 'initial pattern'. Figures 3.7(a) and (b) illustrate the 'test pattern' and 'initial pattern' interfaces for a stuck-open fault in a logic circuit with multiple primary outputs. Figure 3.7 (c) and (d) illustrate the interfaces for a single primary output circuit.

Detection of a *SR (SF)* transition gate delay fault at a gate level circuit requires two patterns in sequence. One is the 'initial pattern' which has value *0 (1)* for a *SR (SF)* transition gate delay fault at the line site to initiate the test, and the other is the 'test pattern' which activates the fault and propagates the fault effect to the primary outputs. Since 'test pattern' generation for a *SR (SF)* transition gate delay fault is similar to test pattern generation for a line stuck-at fault, detection of the stuck-at-0 (stuck-at-1) fault of a line is a necessary condition to detect a *SR (SF)* transition gate delay fault of the line. When generating the 'initial pattern' for a circuit, the fault site line should be fixed at 0(1) for a *SR (SF)* transition gate delay fault in order to initialize the test. In the actual two pattern test generation procedure for *SR (SF)* transition gate delay faults of gate lines, a switch-able output interface could be used. When a 'test pattern' has been generated for a fault, the output interface should be switched to the interface connection shown in figure 3.7(b) or (d) to generate an 'initial pattern' for the fault.

When generating the 'initial pattern' and the 'test pattern' for a *SR (SF)* transition gate delay fault with a serial or parallel computer, a gradient descent algorithm can be used to find consistent labeling of the neurons in the neural networks which correspond to the networks in figure 3.7(a) or (c) and in figure 3.7(b) or (d). When the global minimum energy E given in equation (3.2) is found, the activation values of the primary input neurons in the network form the 'initial pattern' or the 'test pattern'.

If a line of a circuit is both stuck-at-1 and stuck-at-0 fault detectable, the *SR* and *SF* transition gate delay faults of the line are also detectable. Combining test pattern T_0, which detects the stuck-at-0 fault and test pattern T_1, which detects the

stuck-at-1 fault in the sequences such as T_0, T_1, T_0, or T_1, T_0, T_1, will detect the *SR* and *SF* transition gate delay faults on the line.

For all gate lines in a circuit which are both stuck-at-1 and stuck-at-0 fault detectable for an irredundant circuit (i.e. there are no redundant stuck-at faults in the circuit), the networks in figure 3.7(a) or (c) can be used to generate test patterns based on the above property. However, when robust test patterns are needed, test patterns may not necessarily be found with the networks in figure 3.7(a) or (c), since even when a line is both stuck-at-1 and stuck-at-0 fault detectable, robust test patterns for the *SR (SF)* transition gate delay fault may require additional constraints as discussed below.

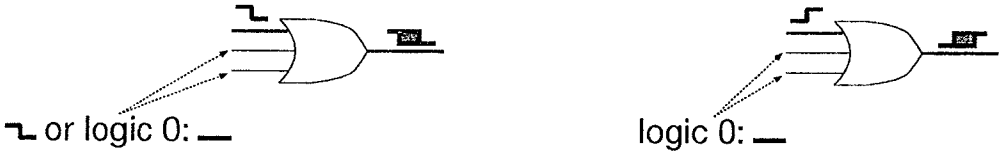
3.5 Robust Test Generation For Stuck-Open or Transition Delay Faults

As detecting stuck-open faults requires a pair of patterns, possible gate delays and timing skews in the network may invalidate the test and the faults may not be detected[22][97][98][99]. Robust test patterns are required to ensure that gate delays and timing skews in the network do not affect the detection of the fault.

In a robust test, transitions from 0 to 1 (or 1 to 0) on the inputs of a gate do not produce a static hazard on its output when the two patterns are applied in sequence. Such static 0-hazards or 1-hazards may invalidate the test. Figure 3.8 shows the required transitions on the off-lines in an *AND* gate and an *OR* gate for robust delay fault test pattern generation.

In test pattern generation for a robust transition gate delay fault test, the network

(a)



(b)

Figure 3.8: The required transitions in the off-lines of an AND gate and an OR gate for a robust test on a transition gate delay fault

in figure 3.7(a) or (c) is used to generate a 'test pattern', then the network in figure 3.7(b) or (d) is switched to generate an 'initial pattern' with an extra constraint term added to prevent hazards. This can be realized with an extra term added to the energy function E defined in equation (3.2) as follows:

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} T_{ij}V_iV_j - \sum_{i=1}^{N} I_iV_i + K$$

$$+ C_r[\sum_{i}^{G}(\sum_{(j,k)}^{g_i} |V_j^{gate_i} - V_j^{gate_i(T_2)}||V_k^{gate_i} - V_k^{gate_i(T_2)}|)] \qquad (3.5)$$

Where $V_j^{gate_i(T_2)}$ is the j-th input value of the 'test pattern' T_2 in gate i, G is the number of gates, g_i is number of inputs at gate i, $V_j^{gate_i} \in \{V_1, V_2, ..., V_N\}$, and C_r is an appropriate positive weight constant for the extra term.

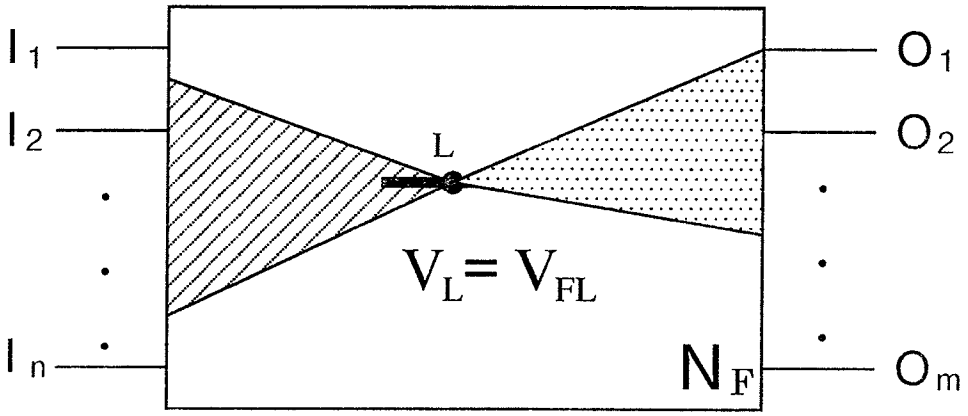


Figure 3.9: Regions required the extra constraint term in robust test pattern generation

After the 'test pattern' T_2 is generated, $V_j^{gate_i(T_2)}$ is known for any $j \in g_i$ and $i \in G$. The parity of $|V_j^{gate_i} - V_j^{gate_i(T_2)}|$ can be decided as

$$|V_j^{gate_i} - V_j^{gate_i(T_2)}| = \begin{cases} V_j^{gate_i} & \text{if } V_j^{gate_i(T_2)} = 0 \\ 1 - V_j^{gate_i} & \text{if } V_j^{gate_i(T_2)} = 1 \end{cases} \tag{3.6}$$

The extra constraint term in equation (3.5) will only change weights, threshold values, and the constant K of a neural network. It will not increase the number of neurons in the network. The extra constraint term is used to restrict the number of transitions from 0 to 1 (or 1 to 0) of inputs of each gate in a circuit to be not more than one. This extra constraint term will always be larger than zero if changes of inputs of each gate in the circuit are greater than one. This ensures that when E in equation (3.2) is globally minimal, the number of input transitions for each gate is not greater than one. Using equation (3.2) and starting at 'test pattern' values or random values, an 'initial pattern' can be found and as such, a robust test pattern can be attained.

In general, the extra constraint term in equation (3.5) gives a sufficient constraint. Since changes of weights, threshold values, and the constant K of a neural network

depend on the 'test pattern', a less strict constraint than the extra term in equation (3.5) can be found for each gate type and specific values of the 'test pattern'.

For example, for a three input AND gate, if all input neuron values of the 'test pattern' are control values, that is, 0, then no extra constraint is needed in that gate. Also, if two input neuron values of a 'test pattern' are control values and the other is a non-control value, the maximum transition number can be two at the three inputs, and if all input neuron values of a 'test pattern' are non-control values, no extra constraint is needed. For other types of gates, similar less strict constraints can be found for specific values of the 'test pattern' for each gate.

When robust test patterns for a circuit with a fault are generated, only neurons corresponding to input gate lines in the network with the fault are required to contain the extra constraint term. Furthermore, for the network with the fault, only neurons corresponding to lines of gates in the affected fan-in and the affected fanout regions of that fault line L as shown in figure 3.9 are required to contain the extra constraint term. Hence, the number of gates G in equation (3.5) can be reduced to the number of gates within the fan-in and fanout regions corresponding to a fault. Furthermore the constraint associated with these gates may be relaxed as discussed above. One further relaxation is made possibly by considering gates along the critical path[106][107]. Since a *SR (SF)* transition gate delay fault that is testable may not be robust testable and since the generation of the 'initial pattern' is related to the 'test pattern', the network energy may not always reach a global minimal. The potential of using a non-zero local minima has not been explored further although it may have some utility in other applications such as test point insertion.

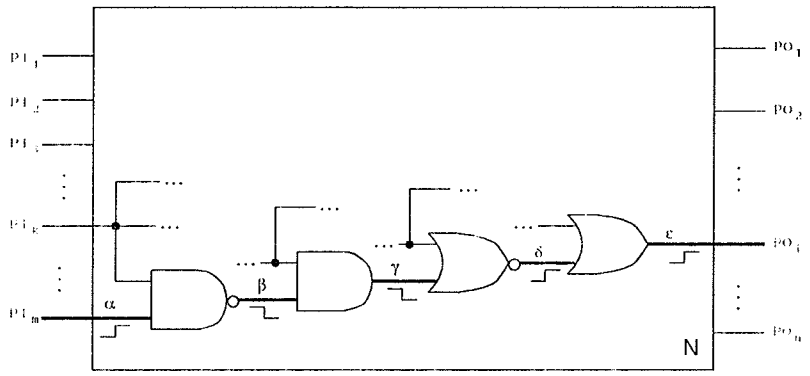
Figure 3.7(b) and (d) illustrate the networks for the generation of an 'initial pattern' after a 'test pattern' has been generated. The structural constraint is still conservative and as such additional relaxation can be implemented. When generating the 'initial pattern', if the neurons in faulty network are satisfied, the neurons in fault-free network will also be satisfied. The neural network can be pruned by considering only the neurons in faulty network while keeping the fault initialized, that is, the neuron corresponding to a SR transition gate delay fault is 0 and the neuron corresponding to a SF transition gate delay fault is 1.

Figure 3.9 shows the network representing the fault that can be used to generate an 'initial pattern'. This can be realized by switching off the 'test pattern' output interface as well as connections between the networks representing the fault-free and faulty circuits.

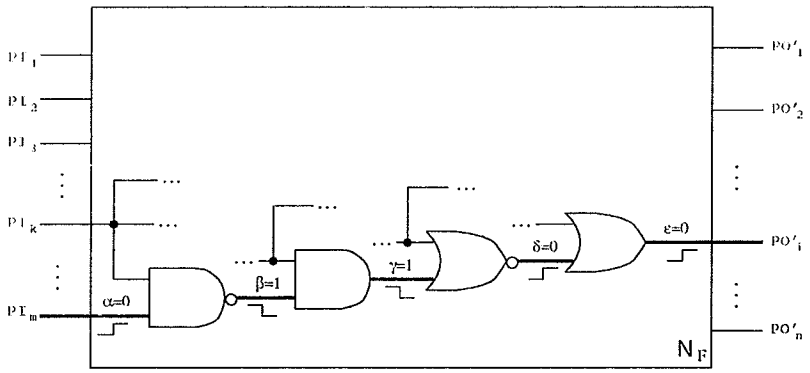
3.6 Path Delay Test Generation

A path delay is the accumulated delay along a path from a primary input line to a primary output line. When a path propagation delay along a circuit path exceeds its specification, we say that a delay fault exists in this path[13][14]. In principle, generating a path delay test is the same as generating a gate delay test since a pair of patterns are required in both cases. As in the transition gate delay fault test generation discussed in the previous section, when a path delay fault is injected in a circuit the values of gate lines along the considered path are required to be set properly in generating both the 'initial pattern' and 'test pattern'.

Hence, when generating a 'test pattern' for a path delay fault of a circuit with its



(a)



(b)

Figure 3.10: An illustration of a path delay fault test generation in a circuit

corresponding neural network as shown in figure 3.7(a) and (c), the values of neurons which correspond to the lines along the considered path in the faulty network are required to be set to appropriate fixed values. A similar pruning procedure and setting of these neuron values is implemented in generating the 'initial pattern' for the path delay fault for both non-robust and robust test generation.

The main difference between generating a transition gate delay fault test and

generating a path delay fault test is that, when generating the 'test pattern' and 'initial pattern' for a transition gate delay fault test, only the neuron corresponding to the transition gate delay fault considered is required to be set to an appropriate fixed value in the faulty network as shown in figure 3.7. However, when generating the 'test pattern' and 'initial pattern' for a path delay fault test, the neurons corresponding to the lines of the considered path are required to be set to appropriate fixed values. The procedure to generate robust test patterns can be applied to delay test generation in both cases.

Consider the example in figure 3.10(a), in which a test for a path α–β–γ–δ–ϵ delay fault is generated with the Hopfield neural network. During the generation of the 'test pattern' and 'initial pattern' for the path α–β–γ–δ–ϵ delay fault, the neurons corresponding to the lines of α, β, γ, δ, and ϵ are required to be set to the appropriate values in the faulty circuit as shown in figure 3.10(b).

3.7 Implementation and Results

The test pattern generation scheme discussed above has been simulated on a serial computer, and a gradient descent algorithm was adopted. As the gradient descent algorithm frequently terminates at a local minimum, the gradient descent algorithm was combined with probabilistic "hill-climbing". When the state of a neuron changes to another state, it will more likely be accepted if the energy function of the neural network is reduced. Otherwise, the acceptance probability p_k is calculated by [96]

$$p_k = \frac{1}{1 + e^{-\Delta E_k / T}} \tag{3.7}$$

Circuit	*Gate*	*I/O*	*Faults*	*Stuck-Open Fault Coverage*	
				Non-robust	*Robust*
Circuit 1	6	5/2	24	100%	100%
Circuit 2	9	6/1	44	95.5%	88.6%
Circuit 3	10	4/2	44	100%	100%
Circuit 4	14	3/2	34	100%	100%
Circuit 5	28	5/3	68	100%	96.0%

Table 3.2: Fault coverage of example circuits

A random number p_{ran} between 0 and 1 is generated, and if $p_k > p_{ran}$ the new state is accepted as the next state. The parameter T in the above equation is given by

$$T = T_j = T_0(1 - \frac{j}{N})^3 \tag{3.8}$$

for j from 0 to N, where T_0 is initial temperature and N is the number of steps taken in decreasing the temperature. As in the closely related simulated annealing problem, there are several adjustable parameters associated with the "Boltzmann" machine[108]. The temperature schedule selected was one based on preliminary simulations which appeared to allow the algorithm to spend a large portion of its time in the region of greatest energy reduction, or in terms of simulated annealing in the region of a phase transition.

The actual test generation procedure is described as follows. Initial values of the primary neurons in the network are randomly chosen and a fault is injected to the network. The initial values of other neurons are obtained by logic simulation. The gradient descent algorithm or the gradient descent algorithm combined with

the probabilistic method is used to search for consistent states of neurons in the network while generating a 'test pattern'. After finding a 'test pattern', a network pruning procedure is required to switch the 'test pattern' output interface to the 'initial pattern' output interface. The search continues with the gradient descent algorithm or the gradient descent algorithm combined with the probabilistic method starting at the 'test pattern' states of the neurons to find an 'initial pattern'. For generating robust test patterns, the weights of the neurons corresponding to input lines of gates in the affected fan-in and fanout regions corresponding to the fault line are modified according to the extra constraint term in equation (3.5). Then a similar search starts at 'test pattern' states of the neurons to find a robust 'initial pattern'.

Consider the circuit[109] shown in figure 3.11(a). Suppose that the line *c2* of the circuit has a *SR* fault which corresponds to the transistor associated with the *nFET* part of gate *G2* as being stuck-open. The corresponding neural network is given in figure 3.11(b), in figure 3.11(c) the fault is injected, and figure 3.11(d) the interface for generating a 'test pattern' for the given fault. The neural network in figure 3.12 (a) is used to generate a 'test pattern' for the fault and the neural network in figure 3.12 (b) is used to generate an 'initial pattern'. Simulation has been performed to generate test patterns and robust test patterns for the fault in question. The pairs of patterns for the *c2* fault are found to be { *A=10, B=11, C=01, D=11, E=00* } and { *A=00, B=11, C=01, D=11, E=00* } for a robust test. When searching for an 'initial pattern' based on the 'test pattern' for a robust test, weights and threshold values of some neurons in the network with the fault are required to be updated according to equation (3.5). As shown in figure 3.13, the weight and threshold values of neurons associated with input lines of gates *G4*, *G5*, and *G6* are updated. The extra term in

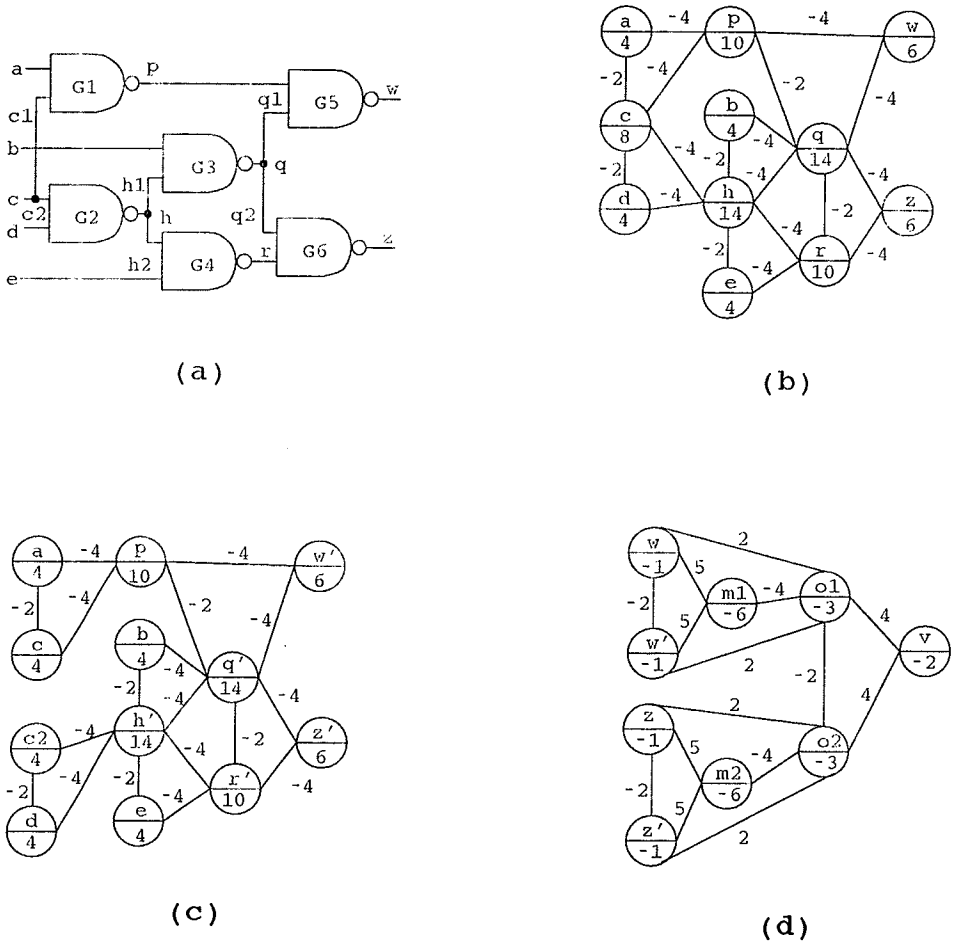


Figure 3.11: Example circuit and its correspondent neural networks for generating test patterns

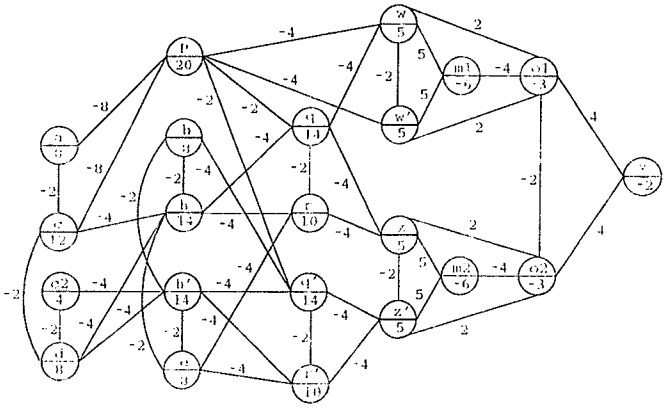
the energy function is

$$C_r[V_e(1 - V_{h'}) + V_{q'}(1 - V_p) + V_{q'}(1 - V_{r'})]$$

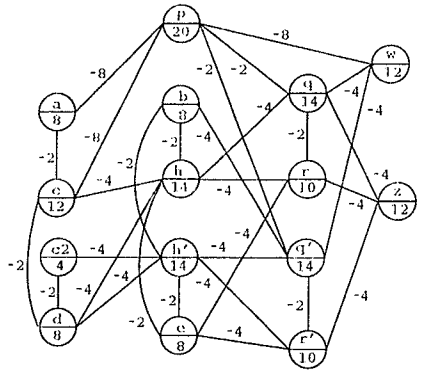
or

$$(V_e + 2V_{q'} - V_eV_{h'} - V_pV_{q'} - V_{q'}V_{r'}),$$

where $C_r = 1$. The gate *G3* does not need an extra constraint term since all



(a)

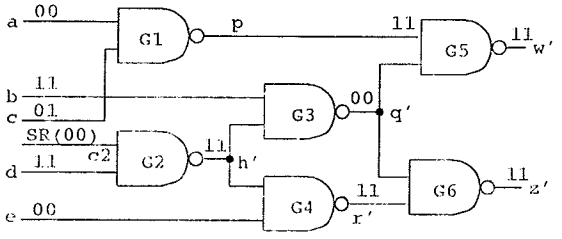


(b)

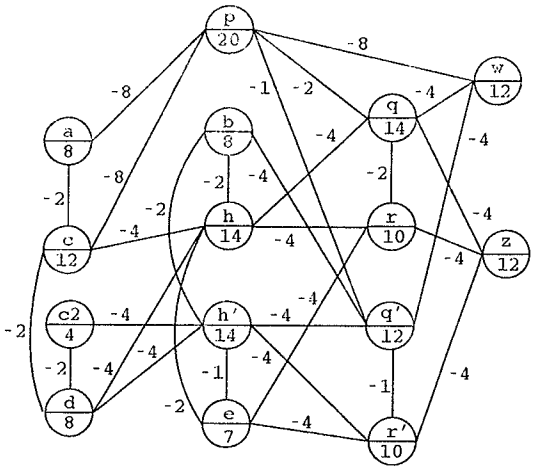
Figure 3.12: Neural networks for generating a 'test pattern' and an 'initial pattern'

input neuron values of the 'test pattern' at gate *G3* are non-control values. Figure 3.13 illustrates the neural network to generate an 'initial pattern' based on the 'test pattern' for a robust test. Figure 3.14(a) and (b) illustrate that only a neural network representing the faulty circuit is required to generate the 'initial patterns' for a non-robust test and a robust test.

A number of small scale circuits[109][110][111][112] (Circuit 1 is in figure 2 in [109], Circuit 2 is figure 13 in [110], Circuit 3 is figure 5 in [111], and Circuit 4 and Circuit 5



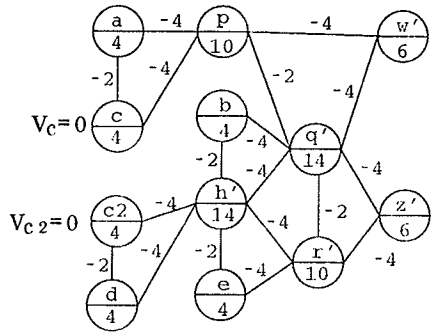
(a)



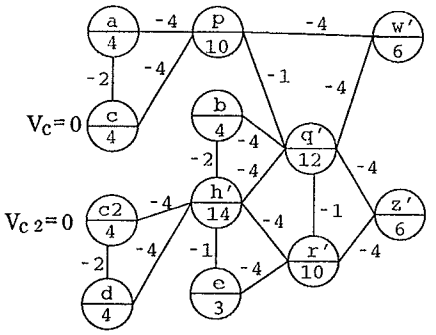
(b)

Figure 3.13: Circuit inserted with a fault and its neural network for generating robust test patterns

are figure 5.39 and figure 5.38 in [112]) have been simulated. The fault coverage results are shown in table 3.2. In circuit 2, there are two redundant faults and two faults which are not robustly testable. Circuit 4 and Circuit 5 are a one bit adder and a two bit parallel adder, respectively. In Circuits 1, 2, and 3, the transition gate delay faults in all lines were considered while for Circuits 4 and 5, only the transition gate delay faults in the lines modeling transistor faults were considered. The simulation started



(a)



(b)

Figure 3.14: Neural networks representing the faulty circuit to generate 'initial patterns' for non-robust (a) and a robust test (b)

at either random values or all zeros (or all ones) on the neurons in each constructed neural network for a 'test pattern', and started at the 'test pattern' values for the 'initial pattern'. Robust test patterns were not found in these cases for Circuit 5. This implies that robust test patterns depend on the 'test pattern', and that perhaps some additional constraints should be added to this phase of the generation process. This has not been investigated further. Through experimenting with a number of

different 'test pattern' values, 100% robust fault coverage for Circuit 5 was attained.

3.8 Summary

A technique for automatic test pattern generation with neural networks for delay fault and stuck-open faults in CMOS combinational circuits was investigated. Also, a scheme for generating a robust test pattern for a stuck-open fault and delay fault was proposed. The simulation results of a number of small scale circuits are satisfactory and illustrate the techniques. These methods have a number of drawbacks when considering extensions to larger circuits, and will likely be of greater utility as parallel machines dedicated to neural network simulation evolve. For instance, extensive simulation experiments have been implemented on an ANZA neuron-computer in generating test patterns for stuck-at test faults by Chakradhar et al[113].

The computational efficiency of the automatic test pattern generation with neural networks could also be improved if the 3-valued neural networks proposed in [114] were adopted.

Recently, Cooper et al[115] derived neural network models for n and $pFET$ transistors for use in switch-level test generation to improve on the unidirectional gate level model of the above work which cannot represent the high impedance state. In their models, almost twice the number of neurons is required[115].

Chapter 4

Design for Skewed-Load Delay Test

4.1 Introduction

In this chapter, a CA driven multiple scan chain skewed-load delay fault testing scheme is presented. A SRL (Shift Register Latch) re-arranging heuristic to enhance delay fault testing is also presented.

The purpose of a delay test is to verify that propagation paths in a logic circuit are within the specific design tolerances. As discussed in previous chapters, delay faults could be modeled as gate delay faults or path delay faults[13][14]. However, the fault set for a path delay fault model is very large for large circuits making it difficult to enumerate all faults in a fault evaluation within a reasonable time. An alternative to enumerating all path delay faults is the transition gate delay fault. Transistor stuck-open faults and transistor stuck-on faults in high speed CMOS circuits also can be modeled by the transition gate delay fault model[100][98][99][91].

Delay fault testing can verify both the functional behavior and correct operation

of a VLSI device at system speeds[16]. Hence, the transition gate delay fault model is usually adopted as a metric by which circuits are evaluated although this metric is incomplete.

Built-in self-test schemes (BIST)[53][2][28] are commonly used to test circuits for stuck-at faults. In order to facilitate delay fault testing at machine speed some modifications to the SRLs in the scan path may be needed[125][126]. In 1983, Barzilai et al[116] proposed shifting alternate SRLs with a flipping strategy to avoid *shift dependency.* In [117], Craig and Kime proposed a pseudo-exhaustive adjacency testing (PEAT) scheme to achieve $n \cdot 2^n$ adjacent pattern pairs for stuck-open fault testing. In [122], Mao and Ciletti presented a heuristic based SRL arrangement algorithm to attack the *shift dependency* problem. Zhang, Byrne, and Miller[118] have investigated BIST parallel generators for sequential faults based on specific instances such as XLFSRs and XLHCAs of XLFSMs[1] to remove *shift dependency.* Savir et al[16] presented a heuristic to re-arrange SRLs in the scan path to enhance transition fault test quality. Leestra et al[119] also presented a reconfigurable scan path design approach to facilitate stuck-open and delay fault detection. These methods all have merit. However, at this time there is no general consensus on the most suitable BIST scheme for delay based fault testing.

[1]Let s = (s_1, s_2, ..., s_n be the state of an n-cell LFSM (Linear Finite State Machine). The state of the corresponding XLFSM is given by (s_1, s_3, s_5, ..., s_2, s_4 s_6, ...). The XLFSRs and XLHCAs are specific instances of XLFSMs.

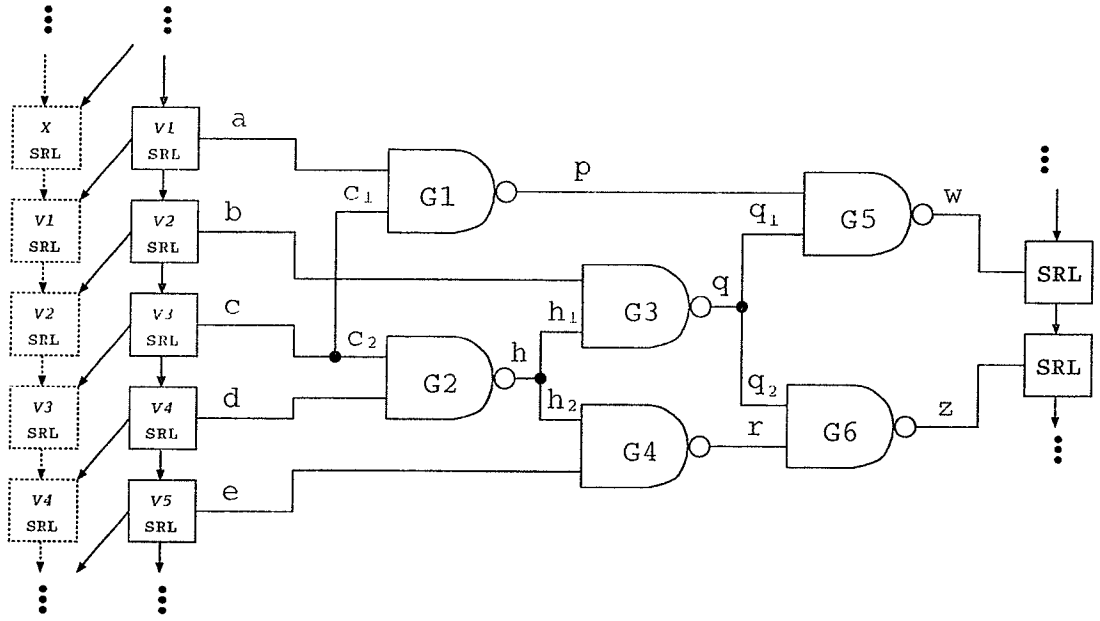


Figure 4.1: Skewed-load delay test scheme

4.2 Skewed-Load Delay Test

As shown in figure 4.1, a skewed-load delay test is a delay test where the second vector of a delay test pair is a one bit shift of the first vector in the pair. This situation occurs when testing combinational logic residing between scan chains. Hence, in a single scan chain shifting skewed-load testing scheme, the number of distinct pattern pairs (or transitions) that can be applied to a circuit is only $2 \cdot (2^n - 1)$ for n inputs[16]. Some necessary transitions may be missing because of the *shift dependency*. Figure 4.2 illustrates state transition diagrams for two and three adjacent connected SRLs. Each state has only two possible transitions.

Savir and Berry[128][125] introduced an AC strength metric to measure the effectiveness of a test scheme. It is defined as a fraction of the exhaustive two pattern

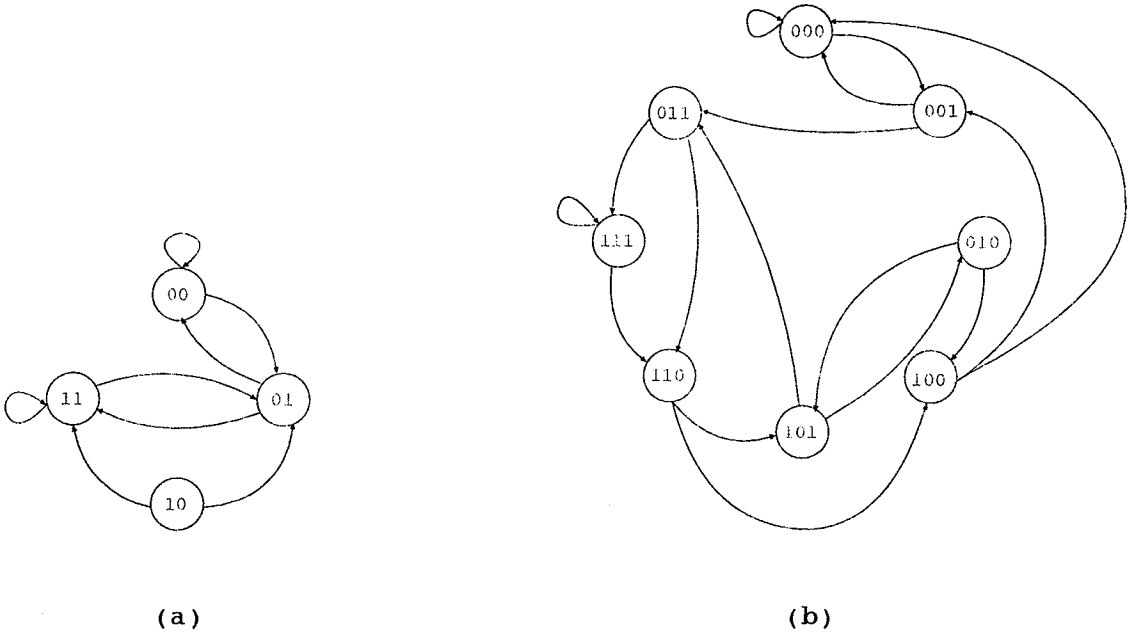


Figure 4.2: State transition diagram of two (a) and three (b) adjacent connected SRLs

pairs that can be applied to a logic circuit in a reasonable time to give an indication of how effective a particular scheme may be in applying AC test patterns to a circuit. Ideally, there are $2^n \cdot (2^n - 1)$ pairs of patterns for n bits[128], that is, if all the transition pairs appear in the SRLs and/or primary inputs of a circuit with a test scheme, the AC strength of the test scheme will be 1.

In [128], Savir and Berry pointed out that the AC strength is 2^{-n+1} for a single scan chain one bit shift test scheme, the AC strength is 2^{-n} for the test-per-clock test scheme using a LFSR as a parallel pattern generator, and the AC strength is 1 for the test-per-clock test scheme using two parallel LFSRs with n and m stages, respectively. They also proved that the AC strength is 2^{-n+1} in the scheme proposed in [116]. Also, Savir and Patil[16][125] have analyzed how *shift dependency* affects

the transition gate delay fault detection in skewed-load testing, and have developed a transition test calculus and a transition gate delay detection analysis scheme.

As the test pattern is strongly related to its initial pattern in a skewed-load delay test, there may not exist test pattern pairs for certain delay faults in a gate with more than two inputs from adjacent shift register latches. For example, as shown in figure 4.1, the SR transition gate delay fault in line c_2 of gate $G2$ is not detectable in the skewed-load test scheme by applying any number of patterns used in the test. This is because the test pattern pairs do not exist in this shift scheme. As a consequence of the *shift dependency* in a skewed-load test scheme, some transition gate delay faults will escape testing, and the gate delay fault coverage will drop.

Savir and Patil have given a lower bound for the gate delay fault coverage in the skewed-load test scheme[126]. Their evaluation is quite conservative, but illustrates how *shift dependency* can affect the delay fault coverage in the worst case *shift dependency* situation. In [127], Vida-Torku reported that the loss of delay test coverage in empirical studies due to latch adjacency (or *shift dependency*) in a boundary-scan architecture is typically between 1% to 5%.

As most test pattern generators in BIST schemes are LFSRs or CAs[2][39][40], there exist *linear dependencies* in the bit streams[2]. When using contiguous cells in a LFSR to drive multiple scan chains, there will be serious *structural dependency* as the same value appears in adjacent scan chains offset by one shift[2].

In order to improve test efficiency and reduce various *dependencies* that occur in scan based delay test, two schemes for re-arranging the SRLs combined with augmenting the scan path SRLs with a one-level XOR network and adopting improved

multiple scan chains fed by the selected taps of a CA generator are proposed.

4.3 Reduced Bit Stream Correlated Pseudo-Random Pattern Generators

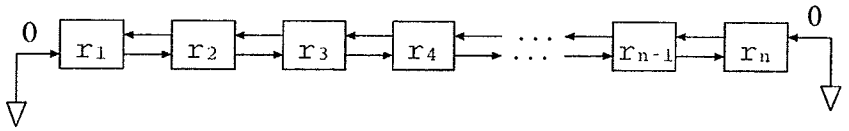
4.3.1 One-Dimensional Cellular Automata

The pseudo-random nature of individual bit streams from certain regular (single rule) one-dimensional cellular automata was first discussed by Wolfram[129]. Hybrid CAs (multiple rules) were later discussed as an alternative source of parallel test stimuli in BIST schemes in [39][40][48]. A cellular automaton evolves in discrete steps with the next value of one cell being determined by its previous value and that of a local set of cells known as neighbor cells. The extent of the neighborhood is a variable parameter, but the number of neighbors generally increases with the dimensionality of the CA under consideration.

Generally, CAs are characterized by the geometry, the neighborhood specification, the number of states per cell, and the rules by which the CA cells calculate their next state. Pries et al proposed *Rule 90/Rule 150* hybrid CAs which yield maximal length non-repeating binary sequences[130]. Serra et al[52] later described a practical algorithm to construct a linear hybrid CA from a selected irreducible polynomial. Recently, Cattell and Muzio proposed a more efficient algorithm for the synthesis of a one-dimensional linear hybrid cellular automaton[134].

Rules 90/150 of hybrid CAs are defined as follows:

$$Rule\ 90: \quad a_i(t+1) = a_{i-1}(t) \oplus a_{i+1}(t)$$



(a) n bit one-dimensional CA

$r_i = 0$ for Rule 90 $r_i = 1$ for Rule 150

(b) cells with Rule 90 and Rule 150

Figure 4.3: An n stage one-dimensional hybrid cellular automaton with null boundary conditions

$$Rule\ 150: \quad a_i(t+1) = a_{i-1}(t) \oplus a_i(t) \oplus a_{i+1}(t)$$

Figure 4.3 illustrates an n stage one-dimensional hybrid cellular automaton with null boundary conditions, where the next value of a cell depends on the values of the left and right neighbors and its present value.

4.3.2 Phaseshift of Conventional LFSRs

In BIST schemes[2][60][132], a pseudo-random test pattern generator is used to ideally deliver uncorrelated random sequences to several scan chains in parallel. If adjacent scan chains are fed from contiguous shift register cells in a LFSR, the same values appear in the adjacent scan chains offset by one bit. Consequently, *structure dependency* occurs[2]. Basically, *structure dependency* is *linear dependency* caused by the contiguous shift register cells feeding adjacent scan channels. In figure 4.4, the shift register

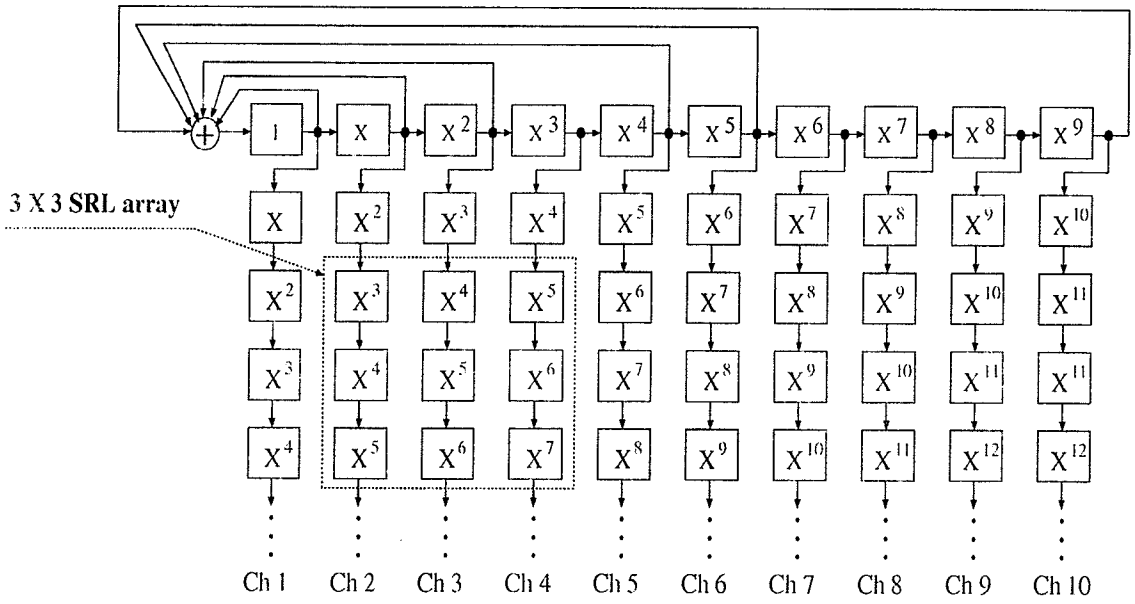


Figure 4.4: Phaseshift of the 10 channels fed by a 10 stage LFSR

cells in a 10 stage LFSR with primitive polynomial $X^{10}+X^6+X^5+X^3+X^2+X+1$ feed 10 scan channels. It can be easily seen that the SRLs on diagonals have the same phaseshifts, that is, the same values will appear in these cells shifted.

Lemma 1: For any $r \times m$ shift register latch array driven by the contiguous cells of an n stage LFSR, there are at most $2^{(r+m-1)}$ (2^n–1, if $r+m \geq n$) different patterns in the array during the whole period of the n stage m-sequence LFSR.

For example, consider the 3 × 3 SRL array formed by the three contiguous channels from the 10 stage LFSR is shown in figure 4.4. There exist at most 5 linearly independent sampling residues, X^i mod $p(X)$ (i = 3, 4, ..., 7), from the 3 × 3 SRL array. Hence, there are at most 2^5 different patterns in the array during the whole period of the LFSR.

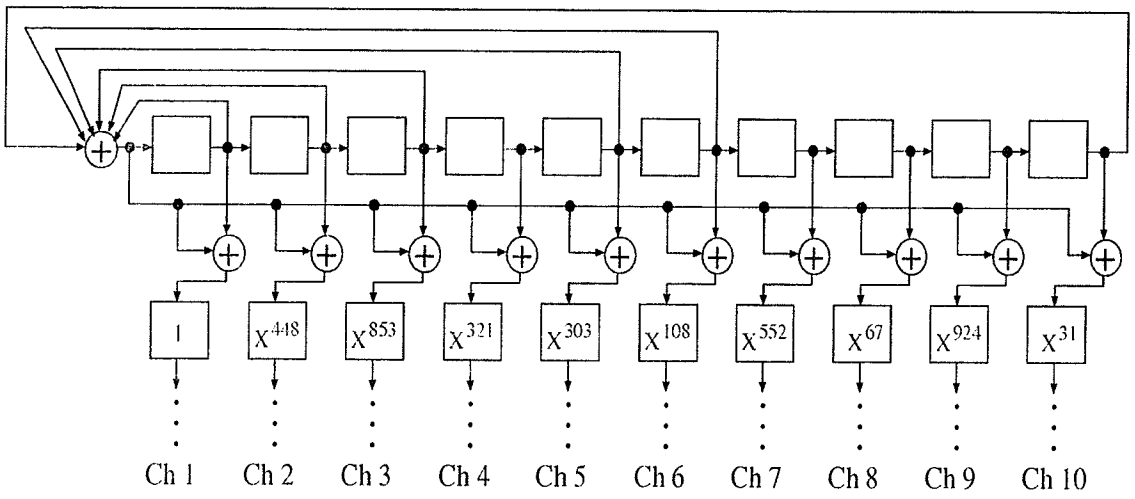


Figure 4.5: Phaseshift of the SRLs driven by a 10 stage LFSR in Bardell's scheme

4.3.3 Phaseshift of Bardell's Scheme

The phaseshift among the different scan channels fed by the contiguous shift register cells in an n stage LFSR is small. An extra XOR network can be introduced to reduce the *structure dependency* by generating a large phaseshift among these channels. In 1990, Bardell[131] proposed a scheme to generate large phaseshifts among the different scan channels as shown in figure 4.5. In this scheme, connections which span the length of a LFSR generator are required.

4.3.4 Phaseshift of Hybrid Cellular Automata

One-dimensional 90/150 hybrid CAs with null boundary conditions are only connected locally. They have the advantages of reduced load capacitance and a regular structure, which results in efficient layouts and potentially improved operating frequencies. The correlation among CA cells is less than that among LFSR cells, and provides effective parallel pseudo-random bit streams[39][40]. Bardell[41] has alge-

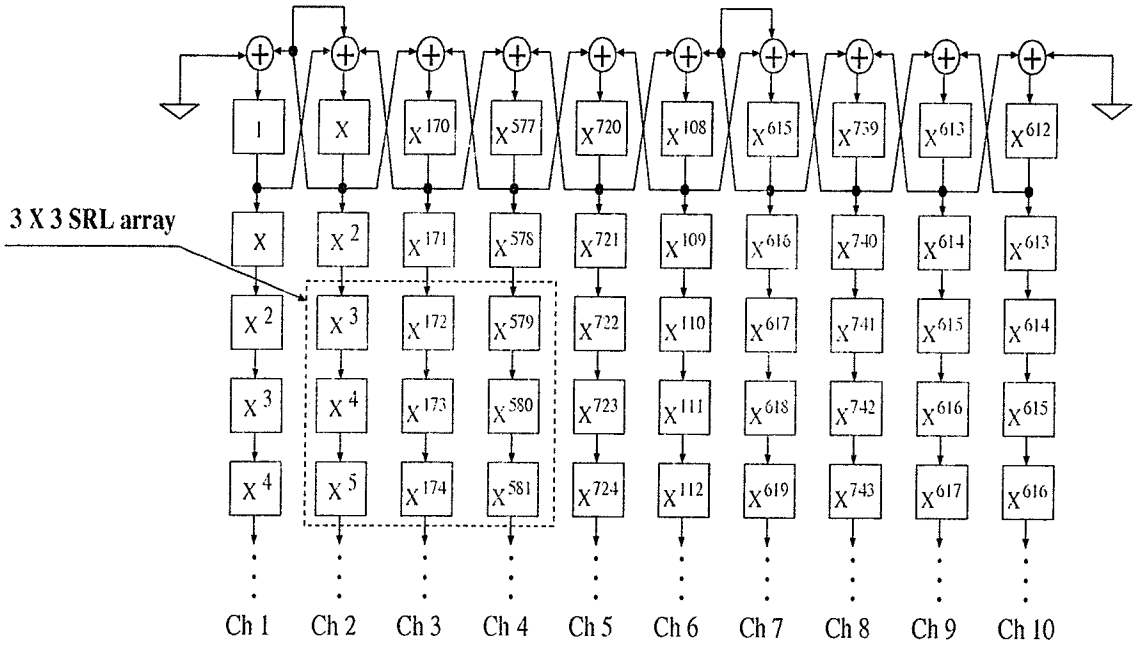


Figure 4.6: Phaseshift of the contiguous cells in a 10 stage 90/150 hybrid CA with a characteristic polynomial $X^{10} + X^6 + X^5 + X^3 + X^2 + X + 1$

braically analyzed the phaseshift between outputs of cells in linear CAs. Typically, one-dimensional 90/150 hybrid CAs generate large phaseshifts among contiguous cells. Figure 4.6 illustrates the phaseshift of the channels fed by the contiguous cells in a minimum hardware cost[2] one-dimensional 90/150 hybrid CA[133]. As the phaseshift among the channels varies quite differently, the possibility of different patterns appearing in any $r \times m$ $(r+m \leq n)$ shift register latch array fed by an n stage hybrid CA is greater than in those fed by an n stage LFSR. A 3×3 SRL array formed by the three contiguous channels from the 10 stage hybrid CA with the rules {*90, 150, 90, ..., 90, 150, 90, 90, 90* } as shown in figure 4.6 can generate 2^6 different patterns in

[2]A *rule* 90 cell in a linear one-dimensional hybrid CA requires two 2-input XOR gates while a *rule* 150 cell requires three. Minimum hardware cost means using the minimal number of *rule* 150 cells in synthesizing a linear one-dimensional hybrid m-sequence CA.

the array during the whole period of the CA. Also, all 2^4 different patterns appear in any 2×2 SRL sub-arrays of the 3×3 SRL array. The possible number of patterns appearing in any array driven by an n stage hybrid CA. is more than that appearing in any array driven by an n stage LFSR.

4.3.5 Phaseshift of 5-neighbor CA derived from 90/150 CA

In matrix notation, the next state X(t+1)=$[x_1(t+1), x_2(t+1), \ldots\ x_n(t+1)]^T$ of an n stage one dimensional hybrid CA as shown in figure 4.3 can be expressed as

$$X(t+1) = \begin{pmatrix} r_1 & 1 & 0 & 0 & \ldots & 0 & 0 \\ 1 & r_2 & 1 & 0 & \ldots & 0 & 0 \\ 0 & 1 & r_3 & 1 & \ldots & 0 & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & 0 & \ldots & 1 & r_n \end{pmatrix} X(t)$$

$$= A \cdot X(t) \tag{4.1}$$

where r_i=0 for *rule 90*, r_i=1 for *rule 150*, i=1,2, ... n. A is the companion matrix of an n stage one dimensional hybrid CA with null boundary conditions.

From equation (4.1), we can easily obtain A^2 as follows:

$$A^2 = \begin{pmatrix} 1+r_1 & r_1+r_2 & 1 & 0 & 0 & \ldots & 0 & 0 & 0 \\ r_1+r_2 & r_2 & r_2+r_3 & 1 & 0 & \ldots & 0 & 0 & 0 \\ 1 & r_2+r_3 & r_3 & r_3+r_4 & 1 & \ldots & 0 & 0 & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & 0 & 0 & \ldots & 1 & r_{n-1}+r_n & 1+r_n \end{pmatrix} \tag{4.2}$$

where + is defined as a *mod* 2 addition. A^2 corresponds to the companion matrix of an n stage 5-neighbor CA derived from the n stage one dimensional hybrid CA. The

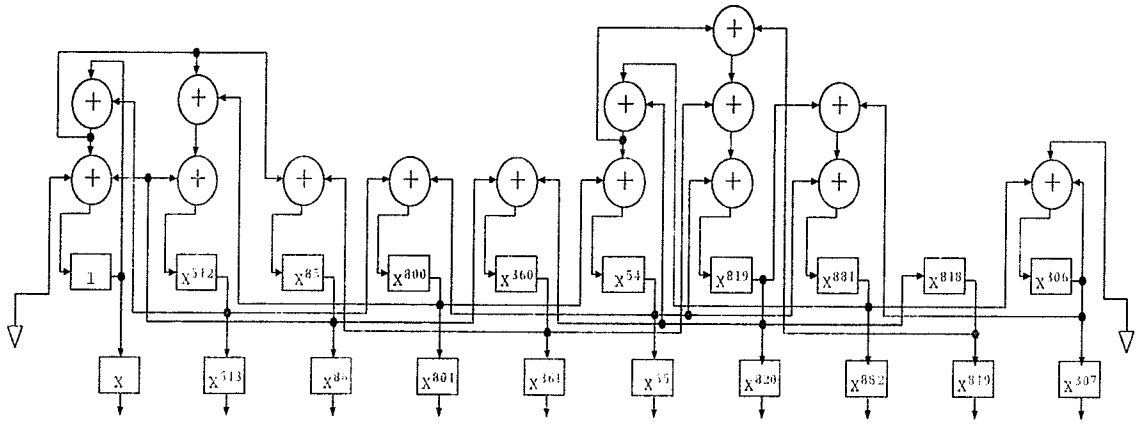


Figure 4.7: A 10 stage 5-neighbor CA with null boundary conditions

derived 5-neighbor CA has the following property:

Lemma 2: If A is a companion matrix which corresponds to an n stage m-sequence hybrid CA, A^2 is a companion matrix of an n stage 5-neighbor m-sequence CA configuration[38].

Hence, for a given n stage m-sequence hybrid CA, we can easily obtain its n stage 5-neighbor m-sequence CA. The implementation of the derived 5-neighbor CA will cost slightly more since longer connections and extra XOR gates are required among the cells. However, since r_i=0 for most of the cells in an n stage minimum hardware cost hybrid CA, the majority of the cells in the 5-neighbor CA will be fed by second left and second right neighbors. Also, by considering common terms in the companion matrix A^2 of the 5-neighbor CA, we can obtain a lower cost implementation. For a cell i and its left and right neighbors with rules of r_{i-1}=0, r_i=1, and r_{i+1}=0 in an n stage minimum hardware cost hybrid CA, only the left and right neighbors of the cell i in the 5-neighbor CA will be affected. By carefully selecting common terms, we will only need three extra 2-input XOR gates for cell i, and its left and right neighbors

in an implementation. Figure 4.7 shows a 10 stage 5-neighbor CA derived from a 10 stage minimum hardware cost hybrid CA[133]. Five extra 2-input XOR gates are required in the 5-neighbor CA.

As most cells in an n stage 5-neighbor CA is fed by its second left and second right cells, or all the 5-neighbor cells, the next state of a subspace spanned by any m ($m < n$) contiguous cells depends on the present states in these m contiguous cells and the first two neighbors in both right and left boundaries of the m contiguous cells. Hence, the possible transition number in the subspace spanned by the m contiguous cells has been increased to $2^4 \cdot (2^m - 1)$, and the quality of transition test pairs as a parallel test pattern generator will be improved compared to the hybrid CA. Also, the phaseshift among the cells of 5-neighbor CAs is comparable to that of hybrid CAs. In addition, the 5-neighbor CAs eliminate adjacent cells with a phaseshift of 1. Tables 4.1 and 4.2 illustrate the phaseshift between cell 1 and the cells 2, 3, ..., n for minimum hardware cost hybrid CAs and their derived 5-neighbor CAs for n=24, 26, 28, 30, 32. From the tables, it can be seen that one-dimensional minimum hardware cost hybrid CAs and their derived 5-neighbor counterparts generate large phaseshifts among the cells.

4.4 Configuring and Feeding Multiple Scan Chains

In a single scan chain based BIST scheme, test application time is proportional to the length of the scan chain and a long test application time is needed during the test session[2]. While configuring multiple scan chains, less time is required to scan data in from a pseudo-random generator and scan out test results to a multiple input

K	n=24	n=26	n=28	n=30	n=32
2	1	33619967	1	536870943	3385127866
3	13071425	34603007	120329699	538968063	3554484478
4	3	33562623	50790212	537133055	1328824908
5	9133058	33554559	120976160	536872959	2584054956
6	9365636	33554463	166606520	536887295	1262063224
7	5782652	41943039	237819585	536871423	452170550
8	7	33556479	110800327	671088639	795729287
9	2392538	33554943	182893482	536936447	1422682354
10	2109376	33555455	153043466	570425343	1137119236
11	11059516	33554447	73558095	553648127	1340760185
12	13438449	33558527	22444504	603979775	3596784209
13	13816968	33587199	232080396	536879103	2133189983
14	415564	50331647	60475329	537001983	1449602619
15	13351812	34078719	166805775	536870927	492379102
16	9527578	33554495	41987312	805306367	1591458575
17	11862917	37748735	176528270	537919487	3041061177
18	868347	33554687	100732255	536871935	337552441
19	4451331	33554439	207486931	536871167	4061393500
20	4218753	33570815	18601510	536903679	3945402262
21	11862913	33816575	232080388	545259519	57313028
22	8157120	35651583	65225350	536875007	1270220315
23	11862911	33554435	204304325	536870919	79129895
24	11862910	33685503	85565928	537395199	3914203779
25	–	33554433	232080384	536871039	3041061169
26	–	33554432	83974625	541065215	1403743683
27	–	–	232080382	536870915	3696670742
28	–	–	232080381	536870975	1560095528
29	–	–	–	536870913	3041061165
30	–	–	–	536870912	1221382304
31	–	–	–	–	3041061163
32	–	–	–	–	3041061162

Table 4.1: Phaseshifts relative to cell 1 of 24, 26, 28, 30, 32 cell hybrid CAs. Arrangements of Rule 90/150 as in [133].

K	n=24	n=26	n=28	n=30	n=32
2	8388608	50364415	134217728	805306383	1692563933
3	14924320	50855935	194382577	806354943	1777242239
4	8388609	50335743	25395106	805437439	664412454
5	4566529	50331711	60488080	805307391	1292027478
6	4682818	50331663	83303260	805314559	631031612
7	2891326	54525951	253127520	805306623	226085275
8	8388611	50332671	189617891	872415231	2545348291
9	1196269	50331903	91446741	805339135	711341177
10	1054688	50332159	76521733	822083583	568559618
11	5529758	50331655	170996775	813694975	2817863740
12	15107832	50333695	11222252	838860799	724650280
13	6908484	50348031	116040198	805310463	1066594991
14	207782	58720255	164455392	805371903	2872284957
15	6675906	50593791	217620615	805306375	246189551
16	4763789	50331679	20993656	939524095	795729287
17	14320066	52428799	88264135	805830655	3668014236
18	8822781	50331775	184583855	805306879	2316259868
19	10614273	50331651	237961193	805306495	2030696750
20	10497984	50339839	9300755	805322751	1972701131
21	14320064	50462719	116040194	809500671	28656514
22	4078560	51380223	32612675	805308415	2782593805
23	14320063	50331649	236369890	805306371	2187048595
24	5931455	50397183	42782964	805568511	4104585537
25	–	50331648	116040192	805306431	3668014232
26	–	16777216	176205040	807403519	2849355489
27	–	–	116040191	805306369	1848335371
28	–	–	250257918	805306399	780047764
29	–	–	–	805306368	446788758
30	–	–	–	268435456	610691152
31	–	–	–	–	3668014229
32	–	–	–	–	1520530581

Table 4.2: Phaseshifts relative to cell 1 of 24, 26, 28, 30, 32 derived 5-neighbor CAs.

signature register (MISR). Subsequently, the whole test application time is reduced. However, for any scan based test, *structure dependency* and *linear dependency*[2] may degrade test efficiency.

Configuring multiple scan chains from an n stage m-sequence CA will improve the AC strength. Taking m ($m \leq \frac{1}{2}n$) scan channels from the odd or even numbered cells of the n stage CA, that is, cells 1, 3, 5, ..., $2m$-1 (or cells 2, 4, 6, ..., $2m$) as 'visible variables'[135], the following representation equation can be derived from (4.1):

$$\begin{pmatrix} x_1(t+1) \\ x_3(t+1) \\ x_5(t+1) \\ \cdots \\ x_{2m-1}(t+1) \end{pmatrix} = \begin{pmatrix} r_1 & 0 & 0 & 0 & \ldots & 0 & 0 \\ 0 & r_3 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 0 & r_5 & 0 & \ldots & 0 & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & 0 & \ldots & 0 & r_{2m-1} \end{pmatrix} \begin{pmatrix} x_1(t) \\ x_3(t) \\ x_5(t) \\ \cdots \\ x_{2m-1}(t) \end{pmatrix} +$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & \ldots & 0 & 0 \\ 1 & 1 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 1 & 1 & 0 & \ldots & 0 & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & 0 & \ldots & 1 & 1 \end{pmatrix} \begin{pmatrix} x_2(t) \\ x_4(t) \\ x_6(t) \\ \cdots \\ x_{2m}(t) \end{pmatrix}$$

$$= A_v X_v(t) + A_u X_u(t) \tag{4.3}$$

The rank of the matrix A_u is m, and $d = (n-m) - rank(A_u)$. Therefore, all the 2^m distinct transitions in the space spanned by these m odd (or even) numbered taps occur, and each of them is traced $2^d = 2^{n-2m}$ times during the complete period of the n stage CA (see theorem 1 in [135]).

Lemma 3: In a skewed-load testing scheme with m ($m \leq \frac{1}{2}n$) channels feeding scan chains from the odd or even numbered cells of an n stage m-sequence CA pattern

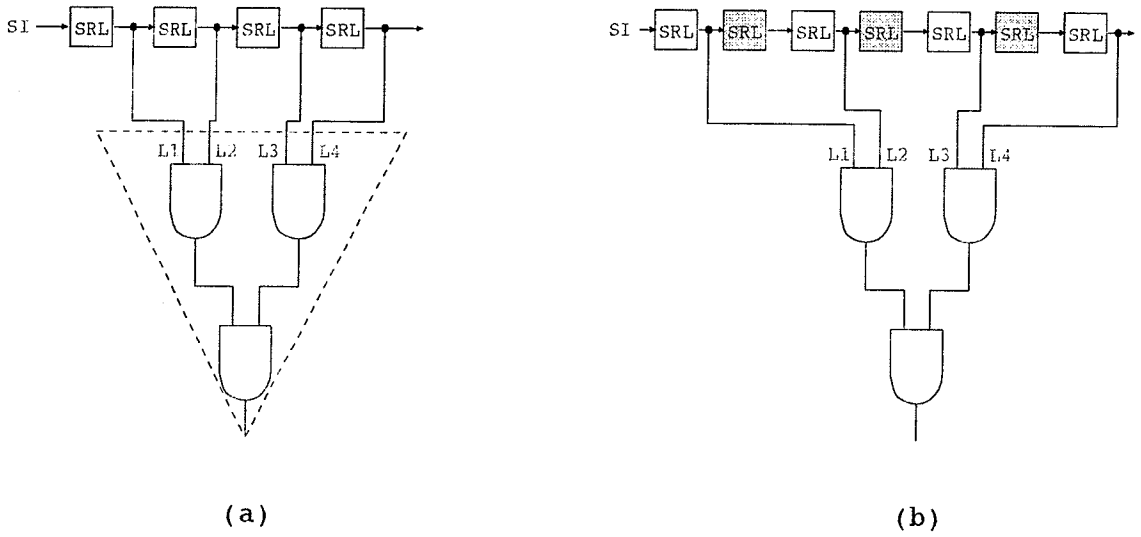


Figure 4.8: Adding dummy SRLs to reduce *shift dependency*

generator, all 2^m distinct transitions in the space spanned by these m odd or even numbered taps will occur. Also, the AC strength has been increased from 2^{-n+1} in a serial scan chain to 2^{-n+m} in the m multiple scan chains from the CA generator.

As the subspace spanned by the m odd (or even) numbered taps of an n stage m-sequence CA pattern generator is exhaustively covered, and all the transitions of each state in the subspace occur, the lemma can be easily proved with the extension of theorem 1 in [135] and the formula which is used to calculate the AC strength proposed in [128].

4.5 Adding a One-Level *XOR* Network to Improve Transition Pair Quality

In [19][128][126][16], heuristic methods are used to re-arrange connections of shift register latches in the scan-path and insert dummy SRLs to eliminate *shift dependency*

in logic sub-cones to enhance fault coverage of the skewed-load test. Sometimes a prohibitive number of dummy SRLs are needed in order to achieve a satisfactory gate delay fault coverage. For example, three dummy SRLs (shaded ones) are required to generate all the possible test pattern pairs in figure 4.8(b). As there are $N!$ arrangements for N inputs, it is not practical to search exhaustively the optimal scan chain arrangement, even for a moderate size of N. In [19], Mao and Ciletti pointed out that the arrangement problem is *NP-hard*.

Lemma 4: Suppose a *shift dependency* exists with a maximum of m SRLs among the n SRLs. In a serial scan chain skewed-load testing, at least $2m - n - 1$ dummy SRLs are required (if $2m - n - 1 \leq 0$, no dummy SRLs are required) to eliminate the effect of *shift dependency* in the SRLs by reordering the scan chain.

This lemma can be easily proven. Assume there exists only one *shift dependency* cone among the m SRLs in an n SRL serial scan chain. The other $n - m$ SRLs can be inserted among these m SRLs to reduce the *shift dependency*. However, to eliminate the *shift dependency* among those m SRLs, $m-1$ dummy SRLs are required. Hence, the total number of dummy SRLs required is $2m - n - 1$. When overlapped *shift dependency* cones exist with maximum m SRLs among the n SRLs, more dummy SRLs are required to eliminate the *shift dependency* in every cone.

Lemma 4 only gives a lower bound on the number of dummy SRLs required to absolutely eliminate the *shift dependency* among the SRLs in the skewed-load delay fault testing.

Adding a prohibitive number of dummy SRLs increases the length of the scan path, costing more area and requiring longer test application times. This is due to

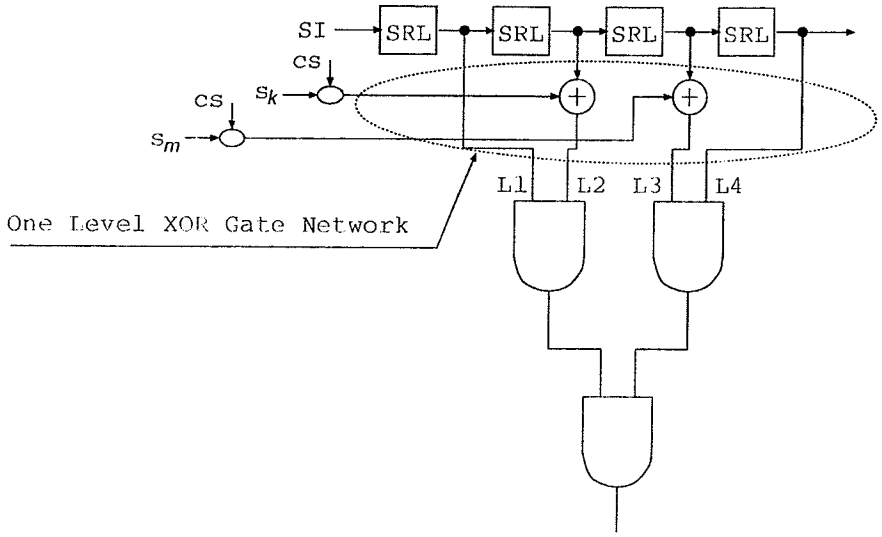


Figure 4.9: Adding one-level XOR network to reduce *shift dependency*

loading long scan strings and shifting outputs to a signature analyzer. As such, it is worthwhile to consider alternative techniques. For example, the scan path can be augmented with simple logical elements to further reduce *shift dependency.* It is well known that when two independent equiprobable signals feed an XOR gate, the output signal will still have the same probability as the inputs. In BIST, pseudo-random test schemes, the scan-in signals generated by LFSR or CA generators[2][39] are considered to be random signals with equal probability. When two signals from the LFSR or CA pseudo-random generator have relatively large phase-shifts and are linearly independent, they can be taken as independent signals. Therefore, the addition of a limited number of one level XOR gates to modify the signals from the SRLs which feed correlated logic cones is proposed. These XOR gates are driven by lines from these SRLs and lines from less correlated SRLs or taps from the generator during a test session. In system mode, the XOR function is disabled by a control signal *CS.* The control signal *CS* can be the scan enable signal in a multiplexor based

flip-flop scan style.

As an example, consider the circuit fed by four consecutive connected SRLs, as shown in figure 4.8(a). Necessary transitions of test patterns from {0xxx, x0xx, xx0x} to {1111} never occur, leaving several delay faults undetected. In figure 4.9, after adding two XOR gates driven by signals S_k, S_m, SRL_2, and SRL_3, the transitions from {0010, 0011, 0100, 0101, 1000, 1001, 1110, 1111} to {1111} are possible. This can be easily verified if signals from S_k and S_m are linearly independent with a relatively large phaseshift, and the taps which span the subspace satisfy lemma 3.

Barzilai et al[136] pointed out that the m-sequence generated by a LFSR corresponding to a primitive polynomial $p(X)$ exhaustively covers the k-subspace $\{s_1, s_2, ..., s_k\}$ spanned by k noncontiguous SRLs if, and only if, the k residue class $[X^{s_i} \bmod p(X)]$ are linearly independent. It can be easily seen that for any k-subspace spanned by the concerned k SRLs, their linear independence property will not change after adding an XOR network driven by signals which are not linearly dependent on these k SRLs. Therefore, the m-sequence still exhaustively covers the subspace. That is, no patterns are lost with the one level XOR network but the number of possible transitions are increased.

4.6 Re-arranging SRLs for Multiple Scan Chain Delay Fault Testing

When logic cones are driven from contiguous SRLs, only a small fraction of all possible pairs of test patterns which may be required for detection of delay faults in those logic cones can be applied[128]. However, separating the inputs that belong to the same

logic cone to avoid contiguous connections of these SRLs will increase the possible pairs of test patterns during test. As loading long scan strings and shifting the results into the signature analyzer makes long test application times, breaking the scan chain into several shorter ones will accelerate the test[2][60][132]. Also, as discussed in the previous section, multiple scan chains will increase transition pairs of test patterns if uncorrelated scan-in signals from a random generator are used. Subsequently, delay fault testing is enhanced. In this section, a configurable two scan chain re-arrangement heuristic is described.

Two configurable scan chains fed by two independent scan-in signals are adopted during SRL re-arrangement. More than two scan chains can be considered. These two scan-in signals are obtained by taping two next to adjacent bits from a 24 stage minimum hardware-cost one dimensional CA generator[133]. Since correlation between them in CAs is much smaller than between them in LFSRs[39][40], the phase-shift between them in CAs can be very large, and possible transition pairs generated by the SRLs are increased.

In addition, n ($\geq$24) stage CAs can be used as parallel pseudo-random generators to feed multiple scan chains for testing various combinational logic units in a chip or multiple chips simultaneously. For example, the n ($\geq$24) stage CAs can be used as parallel pseudo-random generators in the schemes proposed in [2][60][132]. An additional XOR network is not required to generate a satisfactory phaseshift as would be required for a conventional LFSR[2][60].

For a SRL to be arranged, its next connection is considered from among a group of SRLs whose physical layout is local to it since global arrangement of SRLs could

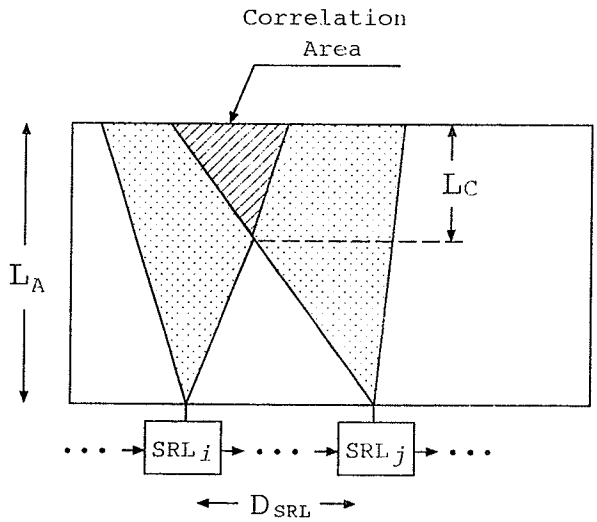


Figure 4.10: Correlation between SRL_i and SRL_j

require substantively long wires. Local adjustment is preferred to avoid a long wire routing penalty. However, a trade-off can be achieved by expanding the allowable range of interconnection.

The selection of the next connected SRL in a given range is made by choosing the SRL to which it is least correlated. Correlation here is defined as the gate overlap as affected by the concerned SRLs. The deeper the level at which the gates are affected, the lesser the correlation. The correlation among SRLs is calculated by flagging the gates affected by a particular SRL, then comparing it with the gates affected by the SRLs within a given range.

Savir and Patil have pointed out that most of the undetectable faults which occur due to *shift dependency* in skewed-load testing appear to be at levels closer to the circuit inputs than the outputs[16]. It can be argued the dependence in deeper logic levels are largely taken care of by the circuit itself. The dependency at deeper levels

is mainly related to the dependency at levels closer to the circuit inputs.

Correlation levels of gates L_C, within potential levels of logic L_A, and range of SRLs D_{SRL}, for arranging SRLs in multiple scan chains are considered, as shown in figure 4.10. Currently, L_A ranges from 1 to 40 levels, and L_C up to 6 levels. The SRL interconnection range D_{SRL} can range from 1 to 20 or more for circuits with a large number of primary inputs. If two SRLs are not correlated in these levels, correlation between them is neglected, and these two can be adjacent. Otherwise, the least correlated SRL within the D_{SRL} is chosen to be connected adjacently. The re-ordering of SRL connections is carried out alternatively for the two (or more than two) scan chains.

After the local adjustment, fault simulation is performed on the circuit with an efficient PPSFP fault simulation algorithm[97]. By comparing the undetected fault lists of the circuit obtained in fault simulation with independent patterns and one bit shift patterns, nearly all of the undetectable faults due to *shift dependency* are obtained. For each of these undetected faults we mark the SRLs that affect the gate at which the fault lies, then analyze connections of these SRLs to obtain *shift dependency* information. Some SRLs are simultaneously marked by a number of undetectable faults. These SRLs are grouped into clusters, and a one level XOR network driven by the outputs of the SRLs and "independent signals" from another tap of a generator or the outputs of other uncorrelated SRLs are added to improve transition pair coverage and to reduce *shift dependency.* These "independent signals" are appropriately chosen from the SRLs not belonging to the dependent SRL clusters to ensure the possible different patterns on the concerned SRLs will not be decreased.

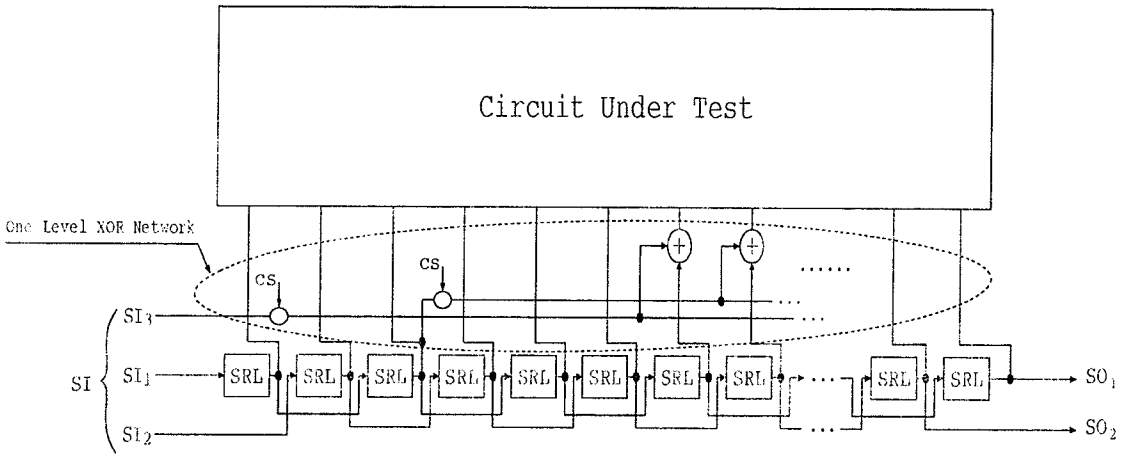


Figure 4.11: Local re-arrangement and additional one-level XOR network scheme

The procedure of adding an XOR network controlled by another tap of a generator or the outputs of the uncorrelated SRLs is carried out iteratively with the help of fault simulation. At each step, the undetectable faults due to *shift dependency* are analyzed, then necessary XOR gates controlled by "independent" signals are added. Fault simulation is again performed on the circuit. If the fault coverage is acceptable, the arrangement is done; otherwise the procedure is repeated. If the fault coverage is still not satisfactory after a preset maximum number of XOR gates are used, D_{SRL} is re-adjusted to a larger range, and then the procedure of adding XOR gates is repeated to further reduce *shift dependency.*

A circuit with locally re-arranged scan chains and a one level XOR network is shown in figure 4.11.

As an example, consider a circuit where *shift dependency* exists as shown in figure 4.12(a). Figure 4.12(b) and figure 4.12(c) illustrate local re-arrangements with SRLs with a range of 2 and 1, respectively. An XOR gate is required to further reduce

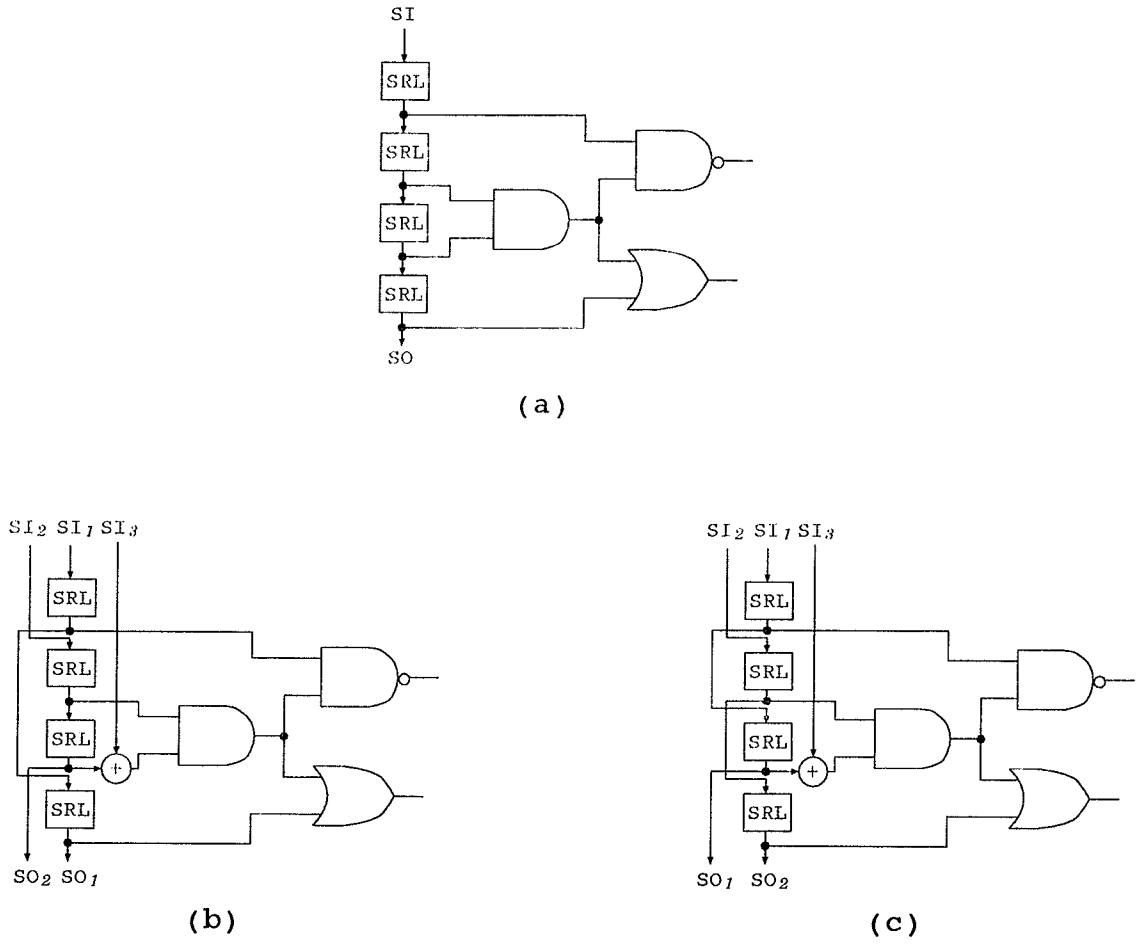


Figure 4.12: A SRL re-arrangement example

shift dependency in both cases.

4.7 Selected Tap Fed Multiple Scan Chains for Pseudo-Exhaustive Testing

In several pseudo-exhaustive test techniques, an exhaustive test for each primary output logic cone is provided to ensure 100% testable stuck-at fault coverage[136]–[141]. In order to ensure exhaustive testing of each logic cone, the test patterns

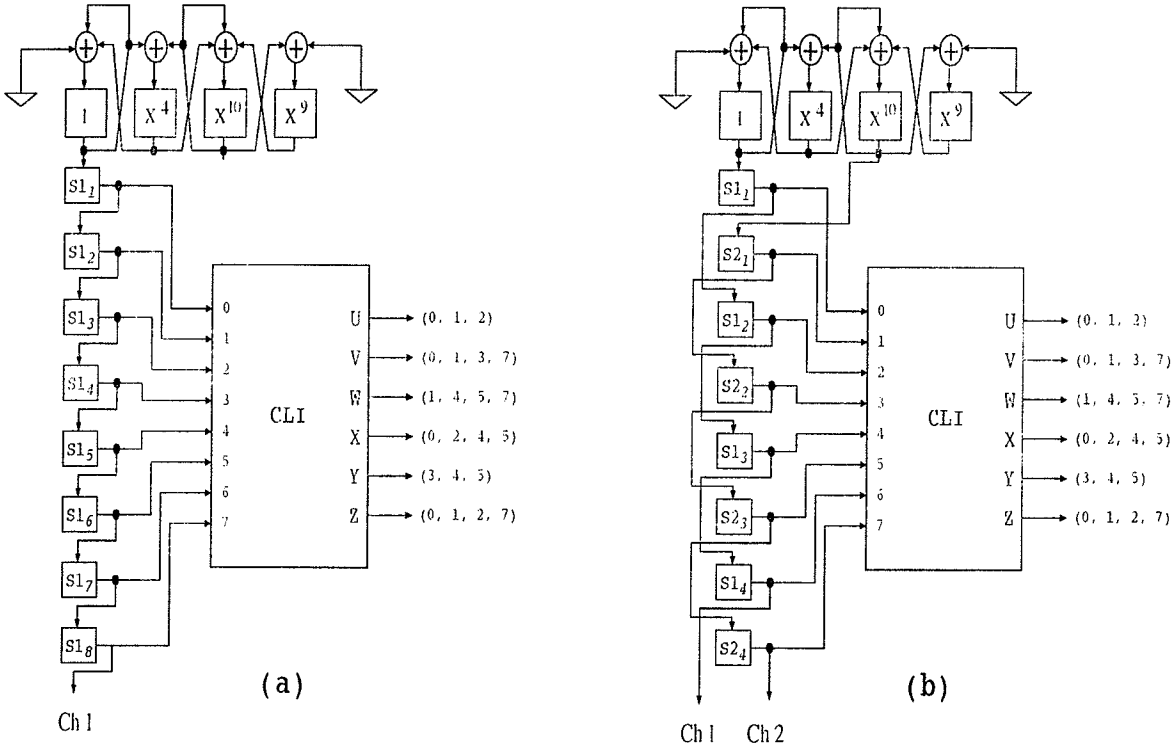


Figure 4.13: An example of arranging multiple scan chains for exhaustive testing

generated by an n stage linear generator should cover the subspace spanned by the primary inputs of the concerned logic cone. That is, the residues with regard to the characteristic polynomial of the LFSR or CA generator on these input positions should be linearly independent[136]. Usually a serial scan chain cannot ensure that all patterns in all subspaces, spanned by the inputs of all output logic cones, are exhaustively generated. As an example, consider a serial scan chain testing of a combinational circuit CL_I as shown in figure 4.13(a). The circuit has 8 inputs and 6 outputs. As each output cone includes at most 4 inputs, a 4 stage CA generator is used. It is obvious that the output logic cones V=(0, 1, 3, 7) and X=(0, 2, 4, 5) are not tested exhaustively. In order to ensure exhaustive testing of every logic cone,

extra XOR gates may be required to generate independent residues for each logic cone[138][140]. Figure 4.13(b) shows two scan chain testing from the cells 1 and 3 of the 4 stage CA generator. It can easily be verified that every output logic cone can be exhaustively tested.

Configuring multiple chains fed by the large phaseshift cells of CAs can ease the difficulty in arranging SRLs for exhaustive testing. Hence, a SRL arrangement procedure for pseudo-exhaustively testing similar to that proposed in [140] for multiple chains can be developed.

4.7.1 Multiple Scan Chain Testing

Although adopting multiple scan chains driven by the selected cells with large phaseshifts from an m-sequence CA can largely avoid *structure dependency*, the *linear dependency* still exists in the SRLs among these multiple chains[2]. Bardell[142] has calculated the *linear dependency* in SRLs among multiple scan chains.

Table 4.3 shows the residues of the first 10 SRLs in channels 1, 2, 3, and 4 fed by a 4 stage minimum cost 150/90 CA generator with the characteristic polynomial X^4+X+1. Assume d_{ij} is the residue of the SRL j in the scan chain i with regard to the characteristic polynomial $p(X)$ of the CA generator, and $S_{i,j}$ is the phaseshift of SRL_j in channel i with regard to the first cell in the CA generator. If the SRLs 1 (d_{11}=X, S_{11}=X) and 5 (d_{15}=X^2+X, S_{15}=X^5) of channel 1, and the SRL 7 (d_{37}=X^2, S_{37}=X^{17}) of channel 3 in table 4.3 drive a 3 input logic cone, not all patterns will appear at the input of the cone since the residues on these SRLs are not linearly independent. That is, $d_{11}+d_{15}+d_{37}=0$, similarly for SRLs 1 and 3 of channel 2, and

Scan Cell No.	Channel 1	Channel 2	Channel 3	Channel 4
1	X (X)	X^2+X (X^5)	X^3+X^2+X (X^{11})	X^2+X+1 (X^{10})
2	X^2 (X^2)	X^3+X^2 (X^6)	X^3+X^2+X+1 (X^{12})	X^3+X^2+X (X^{11})
3	X^3 (X^3)	X^3+X+1 (X^7)	X^3+X^2+1 (X^{13})	X^3+X^2+X+1 (X^{12})
4	X+1 (X^4)	X^2+1 (X^8)	X^3+1 (X^{14})	X^3+X^2+1 (X^{13})
5	X^2+X (X^5)	X^3+X (X^9)	1 (X^{15})	X^3+1 (X^{14})
6	X^3+X^2 (X^6)	X^2+X+1 (X^{10})	X (X^{16})	1 (X^{15})
7	X^3+X+1 (X^7)	X^3+X^2+X (X^{11})	X^2 (X^{17})	X (X^{16})
8	X^2+1 (X^8)	X^3+X^2+X+1 (X^{12})	X^3 (X^{18})	X^2 (X^{17})
9	X^3+X (X^9)	X^3+X^2+1 (X^{13})	X+1 (X^{19})	X^3 (X^{18})
10	X^2+X+1 (X^{10})	X^3+1 (X^{14})	X^2+X (X^{20})	X+1 (X^{19})

Table 4.3: The residues (and phaseshift) of the first 10 SRLs in channels 1, 2, 3, and 4 fed by a 4 stage minimum cost 150/90 CA generator with characteristic polynomial X^4+X+1.

the SRL 4 of channel 4 in table 4.3. This happens because the residues on these SRLs are linearly dependent.

However, the *linear dependency* in the SRLs among channel 1 and channel 3 is different from that in the SRLs among channel 2 and channel 4 because the phaseshifts in these channels are quite different. The residues on the SRLs 1 and 5 of channel 2, and the SRL 7 of channel 4 can be verified as being linearly independent, and so are the residues on the SRLs 1 and 3 of channel 1, and the SRL 4 of channel 3. Therefore the following lemma can be stated:

Lemma 5: Assume $S_i(X)$, $S_j(X)$, $S_k(X)$, and $S_l(X)$ represent phaseshift polynomials of the SRLs in scan chains i, j, k, and l fed by an n stage m-sequence CA or LFSR generator with characteristic polynomial $p(X)$, and $S_{i,j}(X)=S_i(X)+X^{s_{i,j}}S_j(X)$ and

$S_{k,l}(X)=S_k(X) + X^{s_{k,l}}S_l(X)$. If $X^{s_{i,j}} \neq X^{s_{k,l}} \bmod p(X)$, the *linear dependency* in the SRLs among the two-scan chains i and j will be different from that among the SRLs in the two-scan chains k and l.

Lemma 5 provides a criterion to determine whether or not there is the same *linear dependency* in the SRLs among two different two-scan chains. Although *linear dependency* occurs in the SRLs among multiple scan chains fed by an n stage CA or LFSR generator, *Linear dependency* in fixed positions of the SRLs among multiple scan chains can be avoided by reconfiguring different scan chains using the criterion given in Lemma 5. Scan based stuck-at fault tests and skewed-load delay fault tests for combinational circuits can be enhanced in this manner.

As an example, assume that the correlated inputs of output logic cones V and W are changed to V=(0, 2, 3, 5) and W=(1, 2, 4) in figure 4.13(b). It can be verified that the output logic cone V=(0, 2, 3, 5) cannot be exhaustively tested with the SRLs in channel 1 and 3 feeding two-scan chains. Also, the output logic cone W=(1, 2, 4) can not be exhaustively tested with the SRLs in channel 2 and 4 feeding two-scan chains as shown in 4.13(b). Conversely, if implementing two-scan chains from channels 1 & 3, and 2 & 4 as shown in figure 4.14, respectively, in every second pass during the test session, both output cones can be exhaustively tested.

The above observation forms the basis for improved tap selection in the multiple scan chain skewed-load testing scheme. With the multiple scan chain testing scheme as shown in figure 4.15, it is less difficult to arrange the SRLs to achieve pseudo-exhaustive stuck-at fault testing.

An improved tap selection for the multiple scan chain scheme (figure 4.15) is

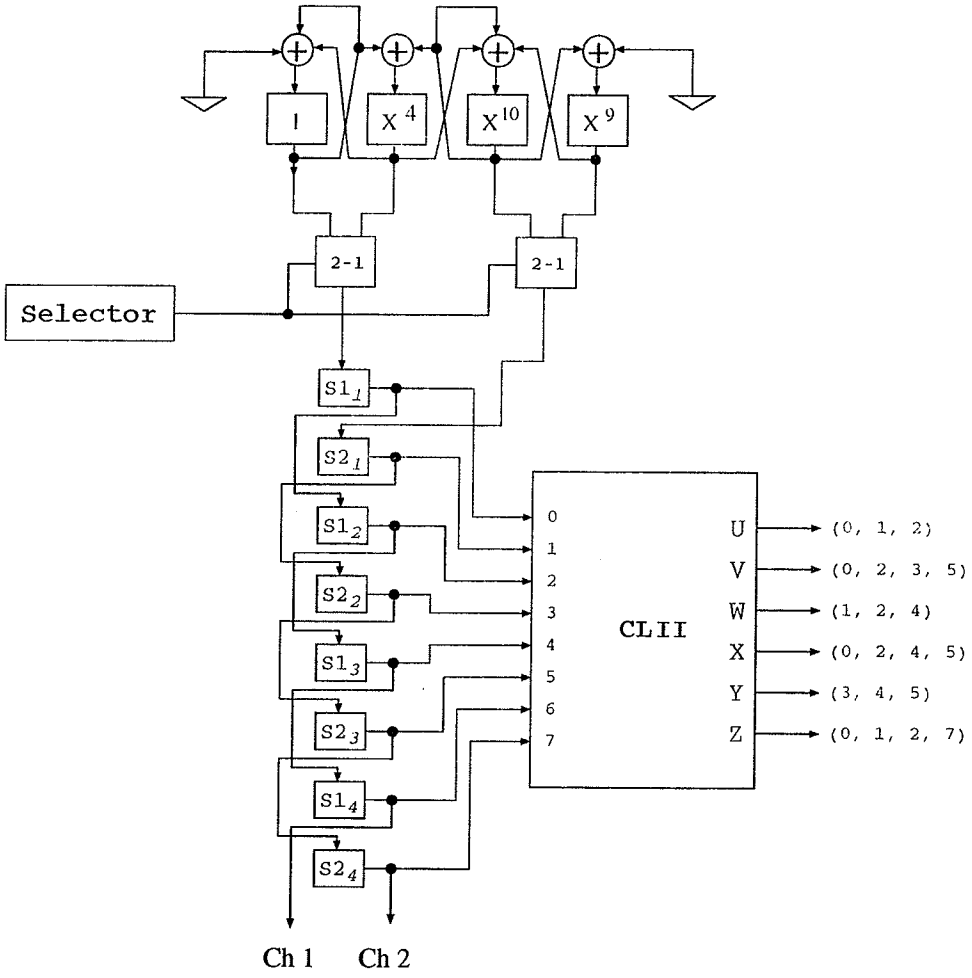


Figure 4.14: The selected two taps fed multiple scan chain for exhaustive testing

proposed to enhance both static and dynamic fault testing. This is done by avoiding fixed *linear dependency* in the SRLs among the multiple scan chains and improving the AC strength. The control signal for the selection of m multiplexers is supplied by a selector with the length of the scan chains plus one.

A SRL re-arrangement heuristic similar to the one presented in the previous section could be used to improve skewed-load delay fault testing in conjunction with the

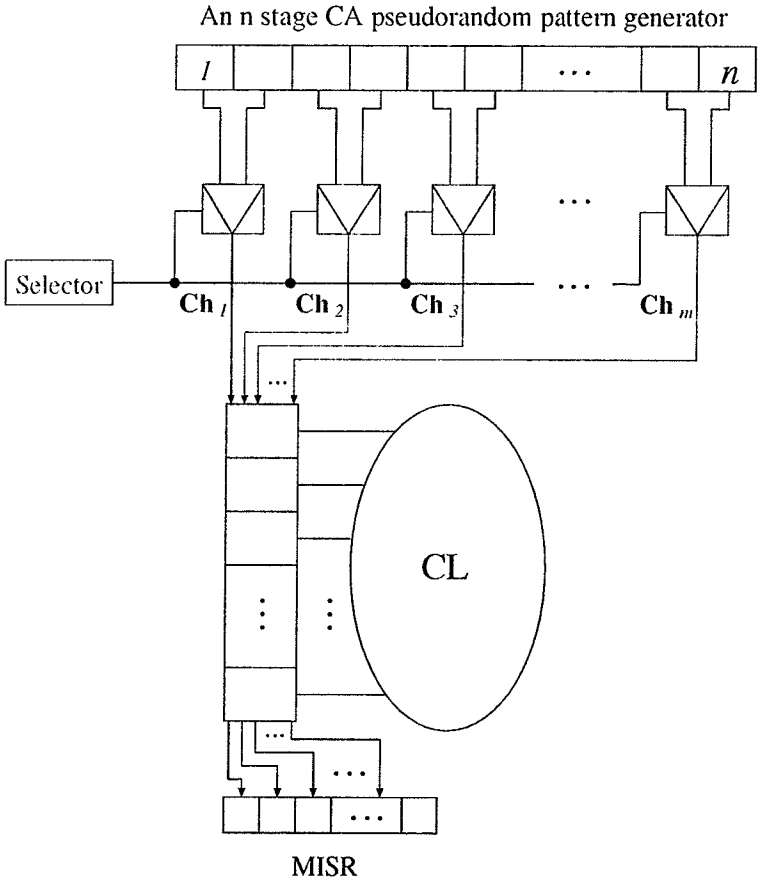


Figure 4.15: The proposed selected taps fed multiple scan chain testing scheme proposed tap selection method.

4.8 Simulation Experiments

Based on the schemes proposed above, the widely accepted ISCAS85 combinational benchmark circuits[144] have been simulated for transition gate delay fault coverage. Table 4.4 gives the characteristics of ISCAS85 combinational benchmark circuits. Transition gate delay simulation methods similar to those presented by [97][98] are

Circuit Name	Circuit Function	Total Gates	Input Lines	Output Lines	Redundant Faults[1]
C432	Priority Decoder	160 (18 *EXOR*)	36	7	4
C499[2]	ECAT	202 (104 *EXOR*)	41	32	8
C880	ALU and Control	383	60	26	0
C1355[2]	ECAT	546	41	32	62
C1908	ECAT	880	33	25	21
C2670	ALU and Control	1193	233	140	117
C3540	ALU and Control	1669	50	22	137
C5315	ALU and Selector	2307	178	123	59
C6288	16-bit Multiplier	2406	32	32	34
C7552	ALU and Control	3512	207	108	131

[1]These are stuck-at type redundant faults. [2]Circuits C499 and C1355 are functionally equivalent. All *XOR* gates of C499 have been expanded into their 4-*NAND* gate equivalents in C1355.

Table 4.4: ISCAS '85 benchmark circuit characteristics

used. The transition gate delay fault set includes slow-to-rise and slow-to-fall faults on primary input lines with over two fanout branches, all *AND, NOR, NAND, OR, INV, BUFF, XOR* gate input slow-to-rise and slow-to-fall faults, and *AND, NOR* gate output line slow-to-fall faults, *NAND, OR* gate output line slow-to-rise faults as well as *XOR* output line slow-to-rise and slow-to-fall faults when their gate outputs have over two fanout branches. In benchmark circuits C1908, C2670, and C3540, two

input gates with common inputs are replaced by single input gates.

A 24 stage CA pseudo-random number generator[39][133] is used as the random bit stream generator engine since these are practical and can generate a sufficient number of patterns. They also offer highly uncorrelated pseudo-random patterns with very large phaseshifts when used as parallel pattern generators. Taps 24 and 22 are selected as SI_1 and SI_2, and tap 20 as SI_3 for the test scheme shown in figure 4.11. The primary input order in the combinational benchmark circuit netlists are assumed as the order in which they would appear in a real design.

Table 4.5 illustrates the transition gate delay fault coverage of the benchmark circuits with a one-level XOR network incorporated with the proposed arrangement scheme with 100,000 patterns. In table 4.5, the fault coverage of all benchmark circuits has been improved with the proposed scheme shown in figure 4.11, compared to a conventional serial skewed-load scheme. Also, the fault coverage on most of the benchmark circuits with the one-level XOR network added is even better than the fault coverage using the parallel CA generator, with the exception of circuit C2670.

Table 4.6 gives the number of XOR gates and control gates used by the re-arrangement heuristic for the test scheme augmented by a one-level XOR network. These results are comparable to the scheme in [126] which employ dummy SRLs to reduce *shift dependency*. Less area overhead is likely to be incurred with the addition of a one-level XOR network than strictly employing dummy SRLs because implementing an XOR gate in CMOS technology requires fewer transistors than implementing a dummy SRL[143].

Table 4.7 illustrates the transition gate delay fault coverage of the ISACS85 bench-

mark circuits for the selected tap fed multiple scan chain testing scheme with 100,000 patterns. In table 4.7, the fault coverage of all benchmark circuits has been improved with the proposed scheme. Also, the fault coverage with the 3 chains is generally better than that with the 2 chains for the benchmark circuits as the AC strength has been improved in the 3 chain test scheme. At present, the insertion of dummy latches to further improve the fault coveragehas not been investigated. A higher degree of fault coverage may reasonably achieved when the proposed multiple chain test scheme is combined with the heuristic presented by Savir and Patil[16].

4.9 Summary

An efficient multiple scan chain test scheme augmented by a one-level XOR network and a selected tap fed multiple chain testing scheme for skewed-load delay fault testing has been proposed and demonstrated. A re-arranging SRL heuristic for multiple chain testing scheme is also presented.

Specifically, the selected tap fed multiple chain testing scheme is suitable for exhaustive testing for stuck-at fault model since the scheme not only ensures 100% pattern coverage for each primary output cone, but also improves the transition quality of these patterns. The selected tap fed multiple chain testing scheme eases the difficulty of routing SRLs in order to ensure 100% pattern coverage for each primary output logic cone.

Extensive experiments on the ISCAS85 benchmark circuits for delay fault testing have been implemented. The fault coverage of all benchmark circuits has been improved with the proposed schemes. The improvement of the transition gate delay

fault coverage in the benchmark circuits is between 0.1% to 6%. This is consistent with the improvement ratio reported by Vida-Torku[127] and is comparable to the results reported by Savir and Patil[16]. Hence, the proposed schemes are effective for improving skewed-load delay fault test efficiency.

Circuit Name	One dimensional Cellular Automata(CA)			
	Faults No.	*Default*	*After Arrangement*	*Parallel CA*
C432	796	96.98	98.74	98.61
C499	988	99.09	99.19	99.09
C880	1617	95.98	99.94	99.57
C1355	2420	96.78	98.97	98.55
C1908	3226	97.49	99.69	98.67
C2670	4412	80.03	84.16	90.82
C3540	6248	89.68	95.90	93.41
C5315	9392	96.76	99.33	99.33
C6288	11104	99.08	99.23	99.23
C7552	13112	94.97	95.36	95.07

Table 4.5: Transition gate delay fault coverage using 24 stage one dimensional cellular automata(CA) and parallel CA with the initial value 10000...0, *Default* refers to a simple scan chain, *After Arrangement* refers to the method augmenting one-level XOR network presented here, *Parallel CA* refers to a parallel cellular automata pattern generator

Circuit Name	*XOR Gate No.*	*Control Gate No.*
C432	10	1
C499	1	1
C880	13	1
C1355	8	1
C1908	6	1
C2670	40	19
C3540	15	4
C5315	11	1
C6288	4	1
C7552	42	20

Table 4.6: XOR gates and control gates used in the proposed scheme

Circuit	One dimensional Cellular Automata (CA)	
	2 chains	*3 chains*
C432	97.24	98.62
C499	99.19	99.19
C880	99.51	99.57
C1355	98.76	98.80
C1908	99.32	99.41
C2670	85.72	86.76
C3540	94.97	95.45
C5315	98.84	99.29
C6288	99.22	99.21
C7552	94.98	95.08

Table 4.7: Transition gate delay fault coverage using 24-bit one dimensional cellular automata(CA) with the initial value 10000...0, *2 chains* and *3 chains* refer to the number of selected taps which feed multiple scan chains

Chapter 5

Statistical Analysis of Delay Faults

5.1 Introduction

In this chapter, a statistical analysis technique for delay faults is presented. The research contribution is an extension of stuck-at fault analysis[166] to delay fault analysis.

A built-in test is often a pseudo-random test, and the quality of the test results depends heavily on the random testability of the circuits. Testability analysis of circuits is used to: *(i)* guide logic circuit modifications in order to improve testability; *(ii)* improve test pattern selection; and *(iii)* evaluate fault coverage affected by signature analysis compaction in BIST schemes[2][145][146][147].

5.1.1 Review of Previous Work

Analytical and approximate testability analysis of circuits for static stuck-at type fault models has been widely discussed[148]–[152][109][153]–[165]. These analytical and approximation methods estimate probabilities and detectabilities for non-redundant

combinational circuits as well as general combinational circuits. It was pointed out that the fault detection problem is *NP-Complete*[163].

Testability measures, such as SCOAP, presented in [150][151], are primarily related to deterministic test pattern generation. They are used to compute deterministic controllability and observability values of a circuit to help identify gross circuit areas where test generation will be difficult and where design for testability should be used[2]. The testability information provided by SCOAP is not immediately comparable to that provided by fault simulation[171].

In [148], an analytical algorithm to calculate the signal probabilities of a combinational circuit was presented. This algorithm can accurately compute the signal probability of any line in a circuit, but exponential execution time is needed in general[2].

In [109] and [153], a fast approximate testability analysis method called COP was proposed. The technique computes the controllability of faults by assigning probabilities to the primary inputs and then propagating them forward through the circuit. The observability is computed by assigning probabilities to the primary outputs and then propagating backwards to the primary inputs using the controllability data in the process. Finally, the testability of a fault is computed by multiplying the controllability and observability. COP will introduce computation errors when calculating the observabilities of re-convergent fanout stems[145].

In [152], a cutting algorithm to calculate signal probabilities and line detectabilities of a combinational circuit was proposed. In this algorithm, the lower and upper bounds of signal probabilities are estimated. With wisely chosen lower and upper

bounds of signal probabilities at fanout branches, better signal probabilities could be estimated. Detectabilities for the fault on a line could be evaluated by calculating the output signal probability of the original circuit compared to the fault injected circuit. The output is obtained by ORing the outputs of EXCLUSIVE ORed primary outputs from both original and fault injected circuits. This technique is computationally expensive in estimating the detectabilities of each line in a circuit. It is reported that the cutting algorithm tends to misidentify many easy-to-test faults as hard[172].

In [154] and [160], a combinational logic circuit is partitioned into a number of "supergates" according to signal correlation relations. The signal correlation is considered in each "supergate" during the calculation of the gate line probabilities. Both the exact and approximate analysis of the testability of a circuit can be implemented by this method.

In [165], an algorithm based on reduced ordered binary decision diagrams (ROBDD) was proposed. This method is integrated in a divide-and-conquer approach based on an extension of the Shannon decomposition of combinational circuits. When the solution of a problem exceeds a given space limit, the problem is divided into two subproblems.

In 1984, Jain and Agrawal presented a statistical fault analysis technique for stuck-at type fault models[166]. The detectabilities were evaluated by comparisons to results obtained by fault simulation. As discussed in [169] the signal correlation between the inputs of the same gate had been taken into account. This statistical fault analysis technique can accurately estimate fault coverage with respect to any set of N input patterns if certain adjustable parameters are chosen properly. Grouping

detection probabilities in roughly low (0.0–0.1), medium (0.1–0.9), and high (0.9–1.0) ranges was experimented with by Jain and Singer[170]. It has been observed[171] that this method can provide some information about the testability of single faults or the entire design, even though the information may not be very accurate.

Another statistical fault detectability estimation technique using PPSFP was presented by Waicukauski et al[167], in which the incorporation of fault simulation is utilized in evaluating fault detectabilities. In [102] a detectability measuring technique which is similar to the approaches presented in [109][153] for transition gate delay faults was presented. Also, an efficient statistical estimation of fault coverage with a test vector sampling technique for delay faults has been proposed in [173]–[175].

5.1.2 Motivation

As indicated, there has been considerable research in determining the detectabilities of faults and fault grading. The main difficulty in the development of useful techniques arises as a consequence of the requisite accuracy and competition from progress made in fast fault simulation[80]–[83][97][98]. Accuracy fault detectability analysis is the most pressing issue in that accurate and fast fault grading provides additional information not directly available from fault simulation. For example, observability, controllability and transition probability information can be used to improve testability through guided test point insertion as well as to obtain knowledge about expected test pattern lengths[2][168].

Also, a variety of high speed integrated circuit test instrumentation techniques is being developed currently, such as scanning electron beam probing[178], high

speed electro-optic probing[179], and several high resolution scanned probe microscopes[180][181], which can all benefit from accurate fault grading. Often a limited number of internal test points are accessible with these instrumentation techniques due to signal acquisition time and because usually only the top level interconnect of a multi-level fabrication process can be accurately monitored. In order for these methods to be effective, fault grading information is required.

In this chapter, a statistical testability analysis technique based on the transition gate delay and path delay fault models presented in [13][14] is discussed. Using machine-word length parallel fault free simulation of combinational circuits, the transition probabilities and transition sensitization probabilities of $0 \rightharpoonup 1$ and $1 \rightharpoonup 0$ transitions of the lines in each gate or each fanout free logic cone are evaluated. A strategy to estimate transition observabilities of fanout stems is proposed. With this method, transition observabilities for any kind of fanout stem can be approximated, although it tends to give a slightly conservative estimate for stems with a large number fanout branches. Finally, the detectability estimation results of transition gate delay faults and path delay faults for the ISCAS85 benchmark circuits[144] are evaluated and compared to results obtained by fault simulation to validate the effectiveness of the estimation method.

5.2 Line Sensitization and Transition Sensitization

The transition gate delay fault detectability of a line in a circuit consists of two parts: transition probability and observability. A gate input line is said to be "sensitive" if

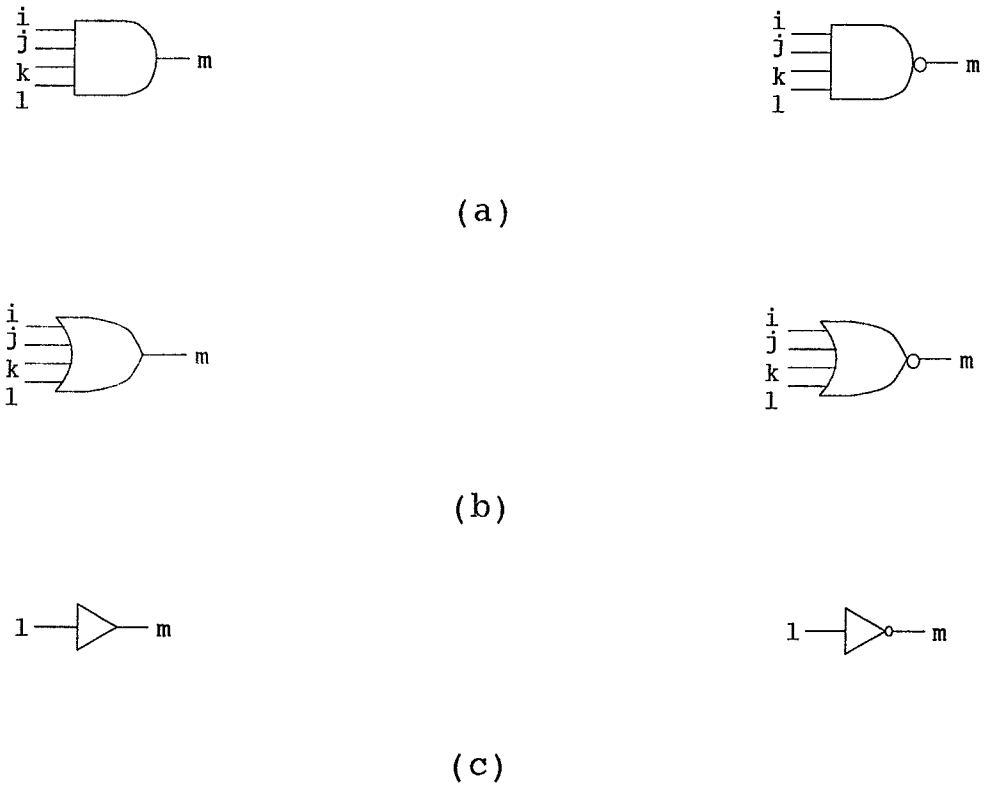


Figure 5.1: Primitive *AND*, *NAND*, *OR*, *NOR*, *BUFF*, and *NOT* gates.

complementing its value alters its gate output[106]. Similarly, a $0 \rightarrow 1$ (or $1 \rightarrow 0$) transition of a gate input line is defined as "sensitive" if this transition will cause a $0 \rightarrow 1$ or $1 \rightarrow 0$ transition of the output. This is a *local sensitization* of a gate. *Global sensitization* of a line is defined as complementing the value of the line so that a change in at least one of the primary output lines occurs. *Global transition sensitization* of a line is defined as a $0 \rightarrow 1$ ($1 \rightarrow 0$) transition of the line which will cause a $0 \rightarrow 1$ (or $1 \rightarrow 0$) transition to occur in at least one of the primary output lines. Clearly, primary output lines and their $0 \rightarrow 1$ (or $1 \rightarrow 0$) transitions are always globally sensitized.

5.2.1 Calculating Line Sensitization and Transition Sensitization within a Gate

$S0^{(T)}(l)$ and $S1^{(T)}(l)$ are used to represent 0 and 1 sensitization values of a line l in a pattern T, and $S_{0\rightarrow 1}{}^{(T-1,T)}(l)$ and $S_{1\rightarrow 0}{}^{(T-1,T)}(l)$ are used to represent $0 \rightarrow 1$ and $1 \rightarrow 0$ transition sensitization values of a line l in a pair of patterns *(T-1,T)*. When a line l is 0 (or 1) sensitized or its $0 \rightarrow 1$ (or $1 \rightarrow 0$) transition is sensitized in a pattern T or a pair of patterns *(T-1,T)*, the values of $S0^{(T)}(l)$ (or $S1^{(T)}(l)$) or $S_{0\rightarrow 1}{}^{(T-1,T)}(l)$ (or $S_{1\rightarrow 0}{}^{(T-1,T)}(l)$) will be 1, otherwise 0. With respect to the primitive gates as shown in figure 5.1, the sensitization and transition sensitization of a input line in a pattern T and a pair of patterns *(T-1, T)* can be defined and calculated as

- for an *AND (NAND)* gate, $S1^{(T)}(l) = S1^{(T)}(i) = S1^{(T)}(j) = S1^{(T)}(k) = Value^{(T)}(m)$ ($S1^{(T)}(l) = S1^{(T)}(i) = S1^{(T)}(j) = S1^{(T)}(k) = \overline{Value^{(T)}(m)}$) and $S0^{(T)}(l) = 1$, if $Value^{(T)}(1) = 0$, and $Value^{(T)}(i) = Value^{(T)}(j) = Value^{(T)}(k) = 1$, etc., where Value($l$) is the logic value of line l.

- for an *OR (NOR)* gate, $S0^{(T)}(l) = S0^{(T)}(i) = S0^{(T)}(j) = S0^{(T)}(k) = \overline{Value^{(T)}(m)}$ ($S0^{(T)}(l) = S0^{(T)}(i) = S0^{(T)}(j) = S0^{(T)}(k) = Value^{(T)}(m)$) and $S1^{(T)}(l) = 1$, if $Value^{(T)}(l) = 1$, and $Value^{(T)}(i) = Value^{(T)}(j) = Value^{(T)}(k) = 0$, etc..

- for a line l, its $0 \rightarrow 1$ and $1 \rightarrow 0$ transition in a pair of patterns *(T-1,T)*, can be calculated as

$$TC_{0\rightarrow 1}{}^{(T-1,T)}(l) = \overline{Value^{(T-1)}(l)} \times Value^{(T)}(l) \tag{5.1}$$

and

$$TC_{1\rightarrow 0}{}^{(T-1,T)}(l) = Value^{(T-1)}(l) \times \overline{Value^{(T)}(l)} \tag{5.2}$$

respectively,

- for a line l, we calculate its sensitizations ${S_{0\rightarrow 1}}^{(T-1,T)}(l)$ and ${S_{1\rightarrow 0}}^{(T-1,T)}(l)$ in a pair of patterns $(T-1,T)$ by

$$ {S_{0\rightarrow 1}}^{(T-1,T)}(l) = S1^{(T)}(l) \times \overline{Value^{(T-1)}(l)} \tag{5.3} $$

and

$$ {S_{1\rightarrow 0}}^{(T-1,T)}(l) = S0^{(T)}(l) \times Value^{(T-1)}(l) \tag{5.4} $$

respectively, where $Value^{(T)}(l)$ represents logic value of line l at a pattern T.

Obviously, for a $BUFF$ (or NOT) gate, the sensitization and transition sensitization values of the input line will be same as (or inverse of) its output.

5.2.2 Calculating Transition Sensitization within a Fanout Free Logic Cone

In calculating the transition sensitization of an input line to a gate, the signal correlation among the input lines to the gate are taken into account[166][169]. However, if the signal correlation among the lines in a fanout free logic cone are considered (which entails a little extra computation), the accuracy of the calculated transition sensitization of the lines of a fanout free logic cone will be improved. The definition of a fanout free logic cone is the same as that used in [106][3]. A fanout free logic cone can include a primary output gate or a fanout stem gate as its *head gate* and (or) all predecessor gates with a fanout of one. A combinational logic circuit can be partitioned into non-overlapping fanout free logic cones. Figure 5.2(a) illustrates an example partitioning of a 1-bit full adder into five non-overlapping fanout free logic cones.

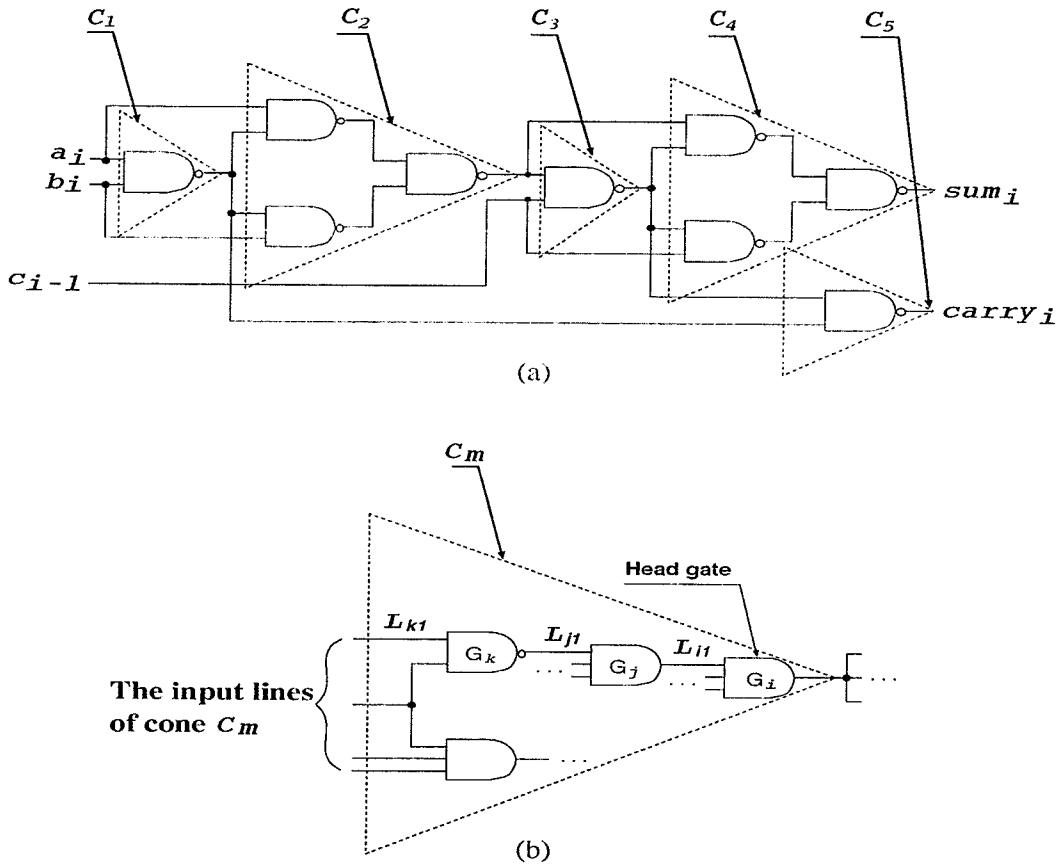


Figure 5.2: (a) Five non-overlapping fanout free logic cones in a 1-bit full adder, (b) a fanout free logic cone C_m

The transition sensitization of an input line to a *head gate* of a fanout free logic cone can be calculated using formulas (5.3) and (5.4). The transition sensitization of an input line to a gate within a fanout free logic cone can also be calculated by an appropriate transition sensitization value of the gate output line to the *head gate* of the fanout free logic cone (this depends on the parity of the concerned gate). As an example, for a fanout free logic cone C_m shown in figure 5.2(b), ${S_{0\rightarrow 1}}^{(T-1,T)}(L_{i1})_{(head\ gate)}$ and ${S_{1\rightarrow 0}}^{(T-1,T)}(L_{i1})_{(head\ gate)}$ are the $0 \rightharpoonup 1$ and $1 \rightharpoonup 0$ transition sensitization values of the input line L_{i1} to the *head gate* G_i in the fanout free logic cone C_m. The $0 \rightharpoonup 1$

and $1 \rightharpoonup 0$ transition sensitization values of input line L_{j1} in G_j can be evaluated as

$$S_{0\rightharpoonup 1}{}^{(T-1,T)}(L_{j1})_{(cone)} = S_{0\rightharpoonup 1}{}^{(T-1,T)}(L_{j1})_{(gate)} \times S_{0\rightharpoonup 1}{}^{(T-1,T)}(L_{i1})_{(head\ gate)}$$

and

$$S_{1\rightharpoonup 0}{}^{(T-1,T)}(L_{j1})_{(cone)} = S_{1\rightharpoonup 0}{}^{(T-1,T)}(L_{j1})_{(gate)} \times S_{1\rightharpoonup 0}{}^{(T-1,T)}(L_{i1})_{(head\ gate)}$$

Similarly, the $0 \rightharpoonup 1$ and $1 \rightharpoonup 0$ transition sensitization values of input line L_{k1} in G_k is evaluated as

$$S_{0\rightharpoonup 1}{}^{(T-1,T)}(L_{k1})_{(cone)} = S_{0\rightharpoonup 1}{}^{(T-1,T)}(L_{k1})_{(gate)} \times S_{1\rightharpoonup 0}{}^{(T-1,T)}(L_{j1})_{(cone)}$$

and

$$S_{1\rightharpoonup 0}{}^{(T-1,T)}(L_{k1})_{(cone)} = S_{1\rightharpoonup 0}{}^{(T-1,T)}(L_{k1})_{(gate)} \times S_{0\rightharpoonup 1}{}^{(T-1,T)}(L_{j1})_{(cone)}$$

respectively, where L_{k1} is one of the input lines of the cone C_m.

This computation procedure begins at primary outputs (or *head gates*) and proceeds backward through each fan-out free region.

5.2.3 Calculating Robust Transition Sensitization

A robust delay fault test is a test that cannot be invalidated (or masked) by delay characteristics in the gates which are not on the path under test[110][22]. Hence, a robust transition sensitization on a path under test will ensure the robust delay fault test.

For *AND* and *NAND* gates with n inputs, the robust transition sensitization values of an input line l can be calculated as:

$$S_{0\rightharpoonup 1}{}^{(T-1,T)}(l)^{(robust)} = \prod_{i=1, i\neq l}^{n} [(\overline{Value^{(T-1)}(i)} \times Value^{(T)}(i) \text{ or}$$

$$Value^{(T-1)}(i) \times Value^{(T)}(i))] \times S_{0\rightarrow 1}{}^{(T-1,T)}(l), \quad (5.5)$$

and

$$S_{1\rightarrow 0}{}^{(T-1,T)}(l)^{(robust)} = \prod_{i=1,i\neq l}^{n} [Value^{(T-1)}(i) \times Value^{(T)}(i)] \times S_{1\rightarrow 0}{}^{(T-1,T)}(l) \quad (5.6)$$

Similarly, the formulae to calculate robust transition sensitization values for OR and NOR gates with n inputs can be calculated as:

$$S_{0\rightarrow 1}{}^{(T-1,T)}(l)^{(robust)} = \prod_{i=1,i\neq l}^{n} [\overline{Value^{(T-1)}(i) \times Value^{(T)}(i)}] \times S_{0\rightarrow 1}{}^{(T-1,T)}(l) \quad (5.7)$$

and

$$S_{1\rightarrow 0}{}^{(T-1,T)}(l)^{(robust)} = \prod_{i=1,i\neq l}^{n} [(Value^{(T-1)}(i) \times \overline{Value^{(T)}(i)} \text{ or } \overline{Value^{(T-1)}(i)} \times \overline{Value^{(T)}(i)})] \times S_{1\rightarrow 0}{}^{(T-1,T)}(l) \quad (5.8)$$

By calculating robust transition sensitization values of lines in the gates or fanout free logic cones of a logic circuit, one can estimate robust delay testability of the logic circuit.

5.3 Evaluation of Line Transition and Transition Sensitization Probability

With the definition of sensitization and transition sensitization of a line l, the sensitization count, transition count, and transition sensitization count of a line l are defined as follows:

- $S0(l)$ is the zero sensitization count of line l.

- $S1(l)$ is the one sensitization count of line l.
- $TC_{0\to1}(l)$ is the 0 to 1 transition count of line l.
- $TC_{1\to0}(l)$ is the 1 to 0 transition count of line l.
- $S_{0\to1}(l)$ is the 0 to 1 transition sensitization count of line l.
- $S_{1\to0}(l)$ is the 1 to 0 transition sensitization count of line l.

The definitions of $S0(l)$ and $S1(l)$ for a line l are the same as in [166].

The zero (or one) sensitization count for N patterns can be calculated as

$$\begin{aligned} S0(l) &= \sum_{T=1}^{N} S0^{(T)}(l) \\ S1(l) &= \sum_{T=1}^{N} S1^{(T)}(l) \end{aligned} \tag{5.9}$$

For N patterns, the transition count and sensitization count of a line l can be calculated as

$$\begin{aligned} TC_{0\to1}(l) &= \sum_{T=2}^{N} TC_{0\to1}{}^{(T-1,T)}(l) \\ TC_{1\to0}(l) &= \sum_{T=2}^{N} TC_{1\to0}{}^{(T-1,T)}(l) \end{aligned} \tag{5.10}$$

and

$$\begin{aligned} S_{0\to1}(l) &= \sum_{T=2}^{N} S_{0\to1}{}^{(T-1,T)}(l) \\ S_{1\to0}(l) &= \sum_{T=2}^{N} S_{1\to0}{}^{(T-1,T)}(l) \end{aligned} \tag{5.11}$$

It is obvious that the calculation of the transition count and transition sensitization count of $0 \rightarrow 1$ and $1 \rightarrow 0$ can be carried out in parallel using the machine-word length for calculation efficiency. Since the difference between $0 \rightarrow 1$ and $1 \rightarrow 0$ transitions of a line l is at most one for N patterns, only calculation of one transition count (either $TC_{0\rightarrow 1}(l)$ or $TC_{1\rightarrow 0}(l)$) for a line l is required for each iteration.

The transition count probabilities (ptc) and sensitization count probabilities(ps) of a line l with N patterns can be evaluated as

$$\begin{aligned} ptc_{0\rightarrow 1}(l) &= \frac{TC_{0\rightarrow 1}(l)}{N-1} \\ ptc_{1\rightarrow 0}(l) &= \frac{TC_{1\rightarrow 0}(l)}{N-1} \end{aligned} \tag{5.12}$$

and

$$\begin{aligned} ps_{0\rightarrow 1}(l) &= \frac{S_{0\rightarrow 1}(l)}{N-1} \\ ps_{1\rightarrow 0}(l) &= \frac{S_{1\rightarrow 0}(l)}{N-1} \end{aligned} \tag{5.13}$$

since there are *N-1* transitions for N patterns.

5.4 Evaluation of Transition Observabilities

After calculating gate line transition probabilities and transition sensitization probabilities in a circuit, the transition observabilities (OBs) of the primitive gate input lines as shown in figure 5.1 (for XOR and $XNOR$ gates, equivalent circuits can be constructed from those primitive gates) and fanout stem lines can be calculated.

5.4.1 Evaluating Primitive Gate Line Transition Observabilities (OB)

For an AND gate as shown in figure 5.1(a), $OB_{0\to1}(l)$ and $OB_{1\to0}(l)$ of its input line l are computed with the gate output line m transition observabilities as

$$\begin{aligned} OB_{0\to1}(l) &= OB_{0\to1}(m) \times Prob[i_{x\to1}, j_{x\to1}, k_{x\to1}|l_{0\to1}] \\ &= OB_{0\to1}(m) \times \frac{Prob[i_{x\to1}, j_{x\to1}, k_{x\to1}, l_{0\to1}]}{ptc_{0\to1}(l)} \\ &= OB_{0\to1}(m) \times \frac{ps_{0\to1}(l)}{ptc_{0\to1}(l)} \end{aligned} \tag{5.14}$$

and

$$\begin{aligned} OB_{1\to0}(l) &= OB_{1\to0}(m) \times Prob[i_{x\to1}, j_{x\to1}, k_{x\to1}|l_{1\to0}] \\ &= OB_{1\to0}(m) \times \frac{Prob[i_{x\to1}, j_{x\to1}, k_{x\to1}, l_{1\to0}]}{ptc_{1\to0}(l)} \\ &= OB_{1\to0}(m) \times \frac{ps_{1\to0}(l)}{ptc_{1\to0}(l)} \end{aligned} \tag{5.15}$$

where x is the "don't-care" value and $x \in \{0, 1\}$. The transition observabilities of an OR gate have the same form as the ones of an AND gate.

Similarly for $NAND$ and NOR gates, the transition observabilities of input lines can be calculated as

$$OB_{0\to1}(l) = OB_{1\to0}(m) \times \frac{ps_{0\to1}(l)}{ptc_{0\to1}(l)} \tag{5.16}$$

and

$$OB_{1\to0}(l) = OB_{0\to1}(m) \times \frac{ps_{1\to0}(l)}{ptc_{1\to0}(l)} \tag{5.17}$$

For NOT and $BUFF$ gates, the transition observabilities of the input lines l are

$$OB_{0\to 1}(l) = OB_{1\to 0}(m) \tag{5.18}$$
$$OB_{1\to 0}(l) = OB_{0\to 1}(m)$$

and

$$OB_{0\to 1}(l) = OB_{0\to 1}(m) \tag{5.19}$$
$$OB_{1\to 0}(l) = OB_{1\to 0}(m)$$

respectively.

5.4.2 Evaluating Fanout Stem Line Transition Observabilities

In general, the difficulty encountered with statistical approximations is due to correlations at fanout stems. Figure 5.3 illustrates several possible fanout stem topologies and a discussion as to how they are handled follows.

Case I

For a fanout stem as shown in figure 5.3(a), in which all fanout branches fan into AND, $NAND$, OR, and NOR gates but not NOT or $BUFF$ gates, each fanout branch's single path transition sensitization probability is calculated with all the other branches not sensitized. With these single path sensitization probabilities, single path sensitized transition observabilities for each fanout branch are obtained. The stem transition observabilities can be estimated as a summation of the single path sensitized transition observabilities of all fanout branches augmented or compensated

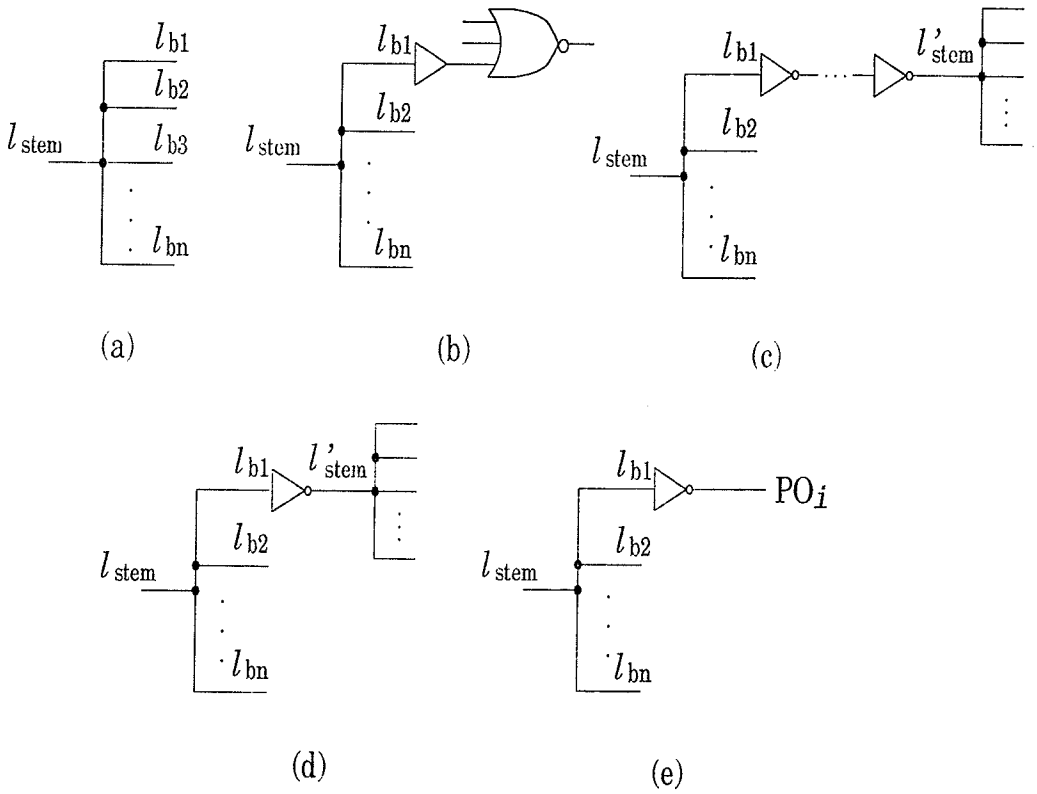


Figure 5.3: Various fanout stem topologies with n fanout branches

by the maximum difference in each fanout branch's transition observability and its single path sensitized transition observability. As such, the single path transition count of fanout branches are calculated as

$$S_{0\to1}{}^{single}(l_{b_i}) = \sum_{T=2}^{N} S_{0\to1}{}^{(T-1,T)}(l_{b_i}) \times \overline{\biguplus_{j\neq i}^{n} S_{0\to1}{}^{(T-1,T)}(l_{b_j})} \tag{5.20}$$

and

$$S_{1\to0}{}^{single}(l_{b_i}) = \sum_{T=2}^{N} S_{1\to0}{}^{(T-1,T)}(l_{b_i}) \times \overline{\biguplus_{j\neq i}^{n} S_{1\to0}{}^{(T-1,T)}(l_{b_j})} \tag{5.21}$$

where, $\biguplus_{j\neq i}^{n}$ is a logic OR summation.

With these single path transition counts of fanout branches, single path transition observabilities of all fanout branches, $OB_{0\to1}{}^{single}(l_{b_i})$ and $OB_{1\to0}{}^{single}(l_{b_i})$ can be

calculated with the formulae (5.14)–(5.19). Fanout stem line transition observabilities can then be evaluated as

$$OB_{0\to 1}(l_{stem}) = \sum_{j=1}^{n} OB_{0\to 1}{}^{single}(l_{b_j}) + \max_j[OB_{0\to 1}(l_{b_j}) - OB_{0\to 1}{}^{single}(l_{b_j})] \quad (5.22)$$

and

$$OB_{1\to 0}(l_{stem}) = \sum_{j=1}^{n} OB_{1\to 0}{}^{single}(l_{b_j}) + \max_j[OB_{1\to 0}(l_{b_j}) - OB_{1\to 0}{}^{single}(l_{b_j})] \quad (5.23)$$

The second term in the above equations (5.22) – (5.23) is used to compensate for transition observabilities from multiple branch sensitization. $OB_{0\to 1}(l_{b_j})$ and $OB_{1\to 0}(l_{b_j})$ are the transition observabilities of a branch line l_{bj} at a gate without constraints from other fanout branches. $OB_{0\to 1}{}^{single}(l_{b_j})$ and $OB_{1\to 0}{}^{single}(l_{b_j})$ are the transition observabilities of a branch line l_{bj} at a gate with the constraint that other fanout branches are not sensitized. Hence, $OB_{0\to 1}{}^{single}(l_{b_j})$ and $OB_{1\to 0}{}^{single}(l_{b_j})$ are the transition observabilities with single branch sensitization, and $OB_{0\to 1}(l_{b_j})$ and $OB_{1\to 0}(l_{b_j})$ are the transition observabilities with single and multiple branch sensitization. The difference between them represents the transition observabilities from multiple branch sensitization, and the maximum value among the all differences is taken as the transition observabilities from multiple branch sensitization.

Case II

For a fanout stem as shown in figure 5.3(b), some fanout branches fan into *BUFF* (or *NOT*) gates, but not *BUFF* (or *NOT*) gate chains. Each of these *BUFF* (or *NOT*) gates only fan into one gate. For this case each fanout branch's single path transition sensitization probabilities are calculated with the other branches not sensitized. The

sensitization probabilities of the fanout branches like l_{b1} which fan into *BUFF* (or *NOT*) gates are properly taken as those of the lines which fan into their immediate successor gates.

Case III

For a fanout stem as shown in figure 5.3(c), (d), or (e), some fanout branches fan into *BUFF* (or *NOT*) gate chains, their immediate successor *BUFF* (or *NOT*) gates have a fanout of two or more, or some fanout branches fan into *BUFF* (or *NOT*) gates whose outputs are primary outputs. The transition observabilities of the stem are calculated using

$$OB_{0\to1}(l_{stem}) = \sum_{i\in S_1} {OB_{0\to1}}^{single}(l_{b_i}) + \sum_{j\in S_2} {OB_{0\to1}}^{single}(l_{b_j}) \tag{5.24}$$

and

$$OB_{1\to0}(l_{stem}) = \sum_{i\in S_1} {OB_{1\to0}}^{single}(l_{b_i}) + \sum_{j\in S_2} {OB_{1\to0}}^{single}(l_{b_j}) \tag{5.25}$$

Here, S_1 is the set which includes branches like l_{b1} in figure 5.3(c), (d), and (e), with the set S_2 containing the reminder of the fanout branches. The second term in the above equations (5.24)-(5.25) represents the transition observabilities from both single and multiple branch sensitizations since the branches in set S_1 will not be included when calculating ${OB_{0\to1}}^{single}(l_{b_j})$ and ${OB_{1\to0}}^{single}(l_{b_j})$ which may include the transition observabilities with multiple branch sensitizations from branch l_{bj} and branches in the set S_1. In a practical evaluation of transition observabilities for a stem using the above equations, the summations of the right hand terms may result in a value larger than 1.0 due to over-compensation from multiple branch sensitization or from a stem that has a high observability value itself. In these instances a value

close to 1.0 (e.g. 0.9) is taken as the observability value of the stem. The results are found to be insensitive to the values between 0.8 to 0.95. One reason for this is that very few fanout stems are overcompensated. This is rarely the case, and it seems to appear only in stems which have the form shown in figure 5.3(c) or (d), based on experimental experience.

In figure 5.3(e), when calculating the single sensitization of branches like l_{b1}, ${S_{0\rightarrow 1}}^{single}(l_{b_1})$ and ${S_{1\rightarrow 0}}^{single}(l_{b_1})$, constraint terms such as $\overline{\uplus_{j\neq i}^{n}{S_{0\rightarrow 1}}^{(T-1,T)}(l_{b_j})}$ and $\overline{\uplus_{j\neq i}^{n}{S_{1\rightarrow 0}}^{(T-1,T)}(l_{b_j})}$ are not needed since fanout branches of this type are functionally equivalent to an inverted primary output or a primary output.

5.4.3 Primary Output Transition Observabilities

For a primary output line l, the $0 \rightarrow 1$ and $1 \rightarrow 0$ transition observabilities are taken as 1.0 since any $0 \rightarrow 1$ and $1 \rightarrow 0$ transitions in a primary output line will be always sensitized and observed.

5.5 Estimating Detectability and Fault Coverage

5.5.1 Estimating Transition Delay Fault Detectability and Fault Coverage

With the estimated transition probabilities and transition observabilities of a line l, the gate delay fault detectabilities for each line can be calculated by

$$d_{sr}(l) = ptc_{0\rightarrow 1}(l) \times OB_{0\rightarrow 1}(l) \qquad (5.26)$$
$$d_{sf}(l) = ptc_{1\rightarrow 0}(l) \times OB_{1\rightarrow 0}(l)$$

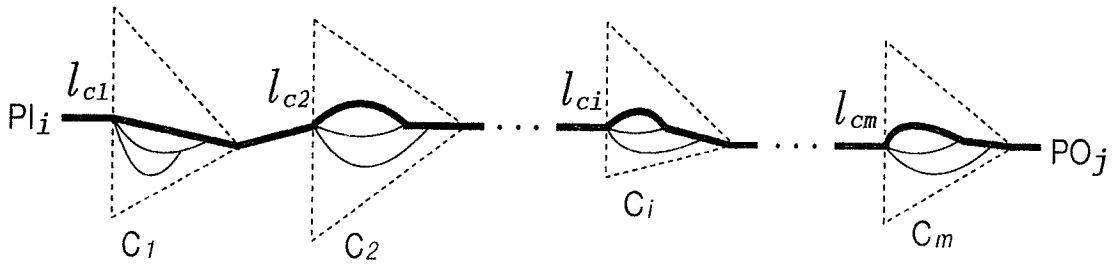


Figure 5.4: The fanout free logic cones and a path from PI_i to PO_j

for slow-to-rise (SR) and slow-to-fall (SF) transition faults, respectively.

For N random patterns and a fault set with F faults, the estimation of transition gate delay fault coverage(EFC) can be calculated as

$$EFC = \frac{1}{|F|} \sum_{f \in F} [1 - \prod_{i} (1 - d_{f_{N_i}})^{N_i}]$$

or

$$EFC = 1 - \frac{1}{|F|} \sum_{f \in F} [\prod_{i} (1 - d_{f_{N_i}})^{N_i}] \tag{5.27}$$

Here $d_{f_{N_i}}$ is the incremented statistical detectability probability of a gate delay fault at pattern N_i, and N_1, N_2, ... N_m are sampling points of *N-1* random patterns, N_m=*N-1*, with N being the entire sequence.

5.5.2 Estimating Path Delay Fault Detectability and Fault Coverage

Path delay fault detectability is estimated as the product of the observabilities of the input line to its *head gate* within the fanout free logic cone on the path multiplied by the transition controllability of the path.

As an example, consider figure 5.4, a path from PI_i to PO_j passes through the fanout free logic cones C_1, C_2, ..., C_m through their input lines l_{C_1}, l_{C_2}, ..., l_{C_m}. The detectability of the path can be calculated as

$$d_{sr}(PI_i - PO_j) = tc_{0\to 1}(l_{C_1}) \times [\prod_{i=1}^{m} \frac{tsc_{0\to 1}(l_{C_i})}{tc_{0\to 1}(l_{C_i})} \text{ or } \frac{tsc_{1\to 0}(l_{C_i})}{tc_{1\to 0}(l_{C_i})}]$$

and

$$d_{sf}(PI_i - PO_j) = tc_{1\to 0}(l_{C_1}) \times [\prod_{i=1}^{m} \frac{tsc_{0\to 1}(l_{C_i})}{tc_{0\to 1}(l_{C_i})} \text{ or } \frac{tsc_{1\to 0}(l_{C_i})}{tc_{1\to 0}(l_{C_i})}] \tag{5.28}$$

respectively, where the choice of $tsc_{0\to 1}(l_{C_i})$ and $tc_{0\to 1}(l_{C_i})$ or $tsc_{1\to 0}(l_{C_i})$ and $tc_{1\to 0}(l_{C_i})$ depends on the transition parity of the input line l_{C_i} of the fanout free logic cone C_i with regard to the given transition test.

For N given test patterns, the estimated path delay fault coverage (EFC) without using the incremented statistical detectability of a delay fault at pattern N_i ($i \in \{1, 2, ..., m\}$), is calculated by

$$EFC = 1 - \frac{1}{|F|} \sum_{f \in F} (1 - d_f)^N \tag{5.29}$$

5.6 Simulation Experiment Results

For the studies undertaken here, the ISCAS85 benchmark circuits[144] and their new non-redundancy version[176] were used. Comparisons between estimated fault coverage (EFC) and estimated grading of hard-to-detect and easy-to-detect faults and those obtained by fault simulation with same random sequence are presented. The results are illustrated in table 5.1, table 5.2, and table 5.3. Table 5.1 compares

Circuit Name	1024			3008			6048			10016		
	EFC	SIM	Δ	EFC	SIM	Δ	EFC	SIM	Δ	EFC	SIM	Δ
C880	96.40	94.74	1.66	99.11	98.27	0.84	99.75	99.07	0.68	99.75	99.44	0.31
C1355	94.27	95.17	-0.90	97.79	97.93	-0.14	98.72	98.39	0.33	98.80	98.47	0.33
C1908	89.32	87.79	1.53	96.03	96.78	-0.75	99.20	98.98	0.22	99.60	99.41	0.19
C2670	65.56	80.55	-14.99	65.73	81.28	-15.55	66.23	81.41	-15.18	66.25	81.41	-15.16
C3540	86.91	88.64	-1.73	89.18	94.80	-5.62	94.81	95.79	-0.98	95.36	95.89	-0.53
C5315	98.85	98.40	0.45	99.24	99.25	-0.01	99.42	99.34	0.08	99.43	99.34	0.09
C6288	98.14	99.23	-1.09	98.14	99.23	-1.09	98.14	99.23	-1.09	98.14	99.23	-1.09
C7552	90.93	90.95	-0.02	93.77	92.53	1.24	94.71	93.07	1.64	95.22	93.56	1.66
c1355_nr	94.20	95.42	-1.22	97.77	98.25	-0.48	98.71	98.71	0.00	98.79	98.79	0.00
c1908_nr	91.68	88.24	3.44	97.59	97.32	0.37	99.38	99.38	0.00	99.72	99.78	-0.06
c2670_nr	82.32	81.82	0.50	82.41	82.28	0.13	82.45	82.36	0.09	82.45	82.45	0.00
c3540_nr	95.81	92.52	3.29	97.17	98.88	-1.71	99.37	99.71	-0.34	99.55	99.80	-0.25
c5315_nr	99.08	99.00	0.08	99.46	99.90	-0.44	99.63	100.00	-0.37	99.63	100.00	-0.37
c6288_nr	100.00	100.00	0.00	100.00	100.00	0.00	100.00	100.00	0.00	100.00	100.00	0.00
c7552_nr	93.53	92.98	0.55	94.69	94.51	0.18	95.39	94.99	0.40	95.82	95.34	0.52

Table 5.1: Estimation and fault simulation of transition gate delay fault coverage (percentage) for the ISCAS85 benchmark circuits

Circuit Name	$DET < 0.1$	$DET < 0.01$	$DET < 0.001$	$DET < 0.0001$	$DET < 0.00001$
C880	1125/1128	311/322	99/99	3/3	0/0
C1355	2164/2164	801/801	157/165	32/40	21/29
C1908	2290/2291	1395/1396	544/590	6/19	0/9
C2670	3515/3519	1301/1389	831/888	766/824	735/801
C3540	5166/5210	2413/2560	826/857	190/254	145/243
C5315	7708/7710	1128/1211	90/146	5/62	5/62
C6288	3715/4273	112/112	85/85	85/85	85/85
C7552	9505/9538	2258/2446	952/1233	596/938	268/608
c1355_nr	2148/2148	781/781	157/157	32/32	21/21
c1908_nr	2276/2277	1383/1385	524/579	0/2	0/0
c2670_nr	2649/2657	846/913	640/651	617/617	597/597
c3540_nr	4832/4864	2160/2244	580/583	11/11	0/0
c5315_nr	7585/7587	1053/1122	82/91	0/0	0/0
c6288_nr	3546/4193	27/27	0/0	0/0	0/0
c7552_nr	9056/9060	2070/2145	747/924	469/668	250/347

Table 5.2: Hard-to-detect transition gate delay fault grading at different given detectability thresholds with the estimation technique for the ISCAS85 benchmark circuits

Circuit Name	$DET > 0.1$	$DET > 0.01$	$DET > 0.001$	$DET > 0.0001$	$DET > 0.00001$
C880	236/489	1208/1295	1459/1518	1612/1614	1617/1617
C1355	256/256	330/1619	2168/2255	2380/2380	2381/2391
C1908	406/935	1587/1830	2208/2636	3185/3207	3205/3217
C2670	414/893	2329/3023	2733/3524	2847/3588	2852/3611
C3540	431/1038	2872/3688	4453/5391	5549/5994	5824/6005
C5315	780/1682	5863/8181	8909/9246	9277/9330	9281/9330
C6288	5839/6831	10849/10992	10897/11019	10897/11019	10897/11019
C7552	1022/3574	8179/10666	11631/11879	12100/12174	12405/12504
c1355_nr	256/256	261/1623	2058/2247	2372/2372	2373/2383
c1908_nr	403/930	1592/1822	2320/2628	3196/3205	3198/3207
c2670_nr	428/853	2338/2597	2838/2859	2893/2893	2894/2913
c3540_nr	491/1046	3135/3666	4657/5327	5730/5899	5877/5910
c5315_nr	761/1648	5837/8113	8868/9144	9197/9235	9201/9235
c6288_nr	6037/6771	10920/10937	10964/10964	10964/10964	10964/10964
c7552_nr	1022/3485	7798/10400	11403/11621	11790/11877	12085/12198

Table 5.3: Easy-to-detect transition gate delay fault grading at different given detectability thresholds with the estimation technique for the ISCAS85 benchmark circuits

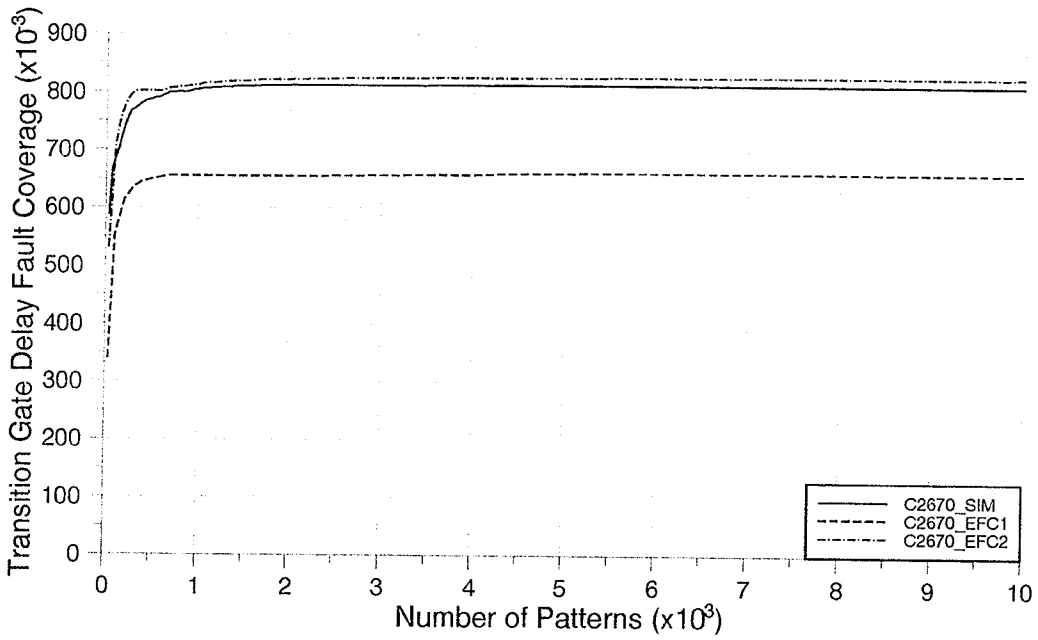


Figure 5.5: The estimated and fault simulated transition gate delay fault coverage of C2670 benchmark circuit

the estimated fault coverage (EFC) considering line correlation within each gate to that obtained by fault simulation for 1024, 3008, 6048, and 10016 pseudo-random input patterns. The gate fault set used is same as the one used in chapter 4. The $\_nr$ indicates the non-redundant version of the circuit.

In general, for all cases except C2670, the estimate using the proposed method considering line correlation within each gate is within 1.7% of the fault coverage determined by fault simulation. For all non-redundant benchmark circuits the estimate is within 0.6% for a reasonable number of patterns (e.g. 10,000).

Circuit C2670, on which the estimate is quite pessimistic, is interesting in that

it is the circuit that is random pattern resistant and contains 117 redundant stuck-at fault lines[182]. These lines are redundant for transition gate delay fault test as well, and they affect the accuracy of the proposed estimation method when only line correlation within each gate are considered.

Figure 5.5 gives fault coverage curves obtained using the proposed estimation method considering line correlation within each gate (C2670_EFC1) and each fanout free region (C2670_EFC2) respectively, and by fault simulation for ISCAS85 benchmark circuit C2670. The estimation curve C2670_EFC1, considering line correlation within each gate, is quite pessimistic when compared with the fault simulation result. However, the estimation curve C2670_EFC2 after considering line correlation within each fanout free region, gives a very accurate estimation, and the difference between them is less than 0.1% at 10,000 patterns.

Fault detectability grading experiments on the ISCAS85 benchmark circuits using the estimation technique have also been performed. Table 5.2 and table 5.3 give the fault grading results obtained with the estimation technique considering line correlation within each gate and by fault simulation after *100,000* patterns without fault dropping. The tables indicate the benchmark circuit, number of the gate delay faults, % of faults with estimated detectabilities less than (or greater than) a given threshold as compared to fault simulated detectabilities, as well as actual numbers of faults at a given detectability. The comparison is done as follows: if a simulated detectability of a line is less (or greater) than the given threshold, then it is compared to the estimated detectability. For example, for C3540 there are 6248 delay faults considered. For a detectability less than 0.001, 826 faults were obtained from the estimation method as compared to 857 faults obtained from fault simulation, producing a yield of 96.38%.

Compared with the results for stuck-at fault grading reported by Huisman[171], the grading accuracy for hard-to-detect faults using the proposed estimation technique has been improved in all the cases.

Grading accuracy for hard-to-detect transition gate delay faults have been compared in both ISCAS85 redundant and non-redundant benchmark circuits. The results indicate that the accuracy for non-redundant benchmark circuits is better than that for redundant benchmark circuits. It seems that redundant lines in circuits affect the accuracy of the estimation method since observability, transition probability, and transition sensitization probability of each redundant line will not be zero in most cases, but their conjunction set will be null[177][160]. The evaluation accuracy of observability, transition probability, and transition sensitization probability of the predecessor lines will be thus affected by redundant lines. The estimation method has no ability to calculate accurately the transition sensitization probabilities and observabilities on the lines that need a larger region to identify redundancy than the fanout free logic cone. From table 5.2 and table 5.3, we can see that the estimation method produces reasonably good fault grading results in almost all cases.

In figures 5.6 and 5.7, the fault grading curves for C2670 are given. They are estimated by the proposed technique when considering line correlation within each gate (C2670_LESS I and C2670_LARGE I) and each fanout free region (C2670_LESS II and C2670_LARGE II), respectively. When the cutoff detectability threshold is at 10^{-5}, about 91.76% of hard-to-detect faults are graded. The estimation technique considering line correlation within each gate mistakes about 20% of easy-to-detect faults as hard-to-detect faults for all given detectability thresholds. On the other hand, the estimation technique considering line correlation within each fanout free

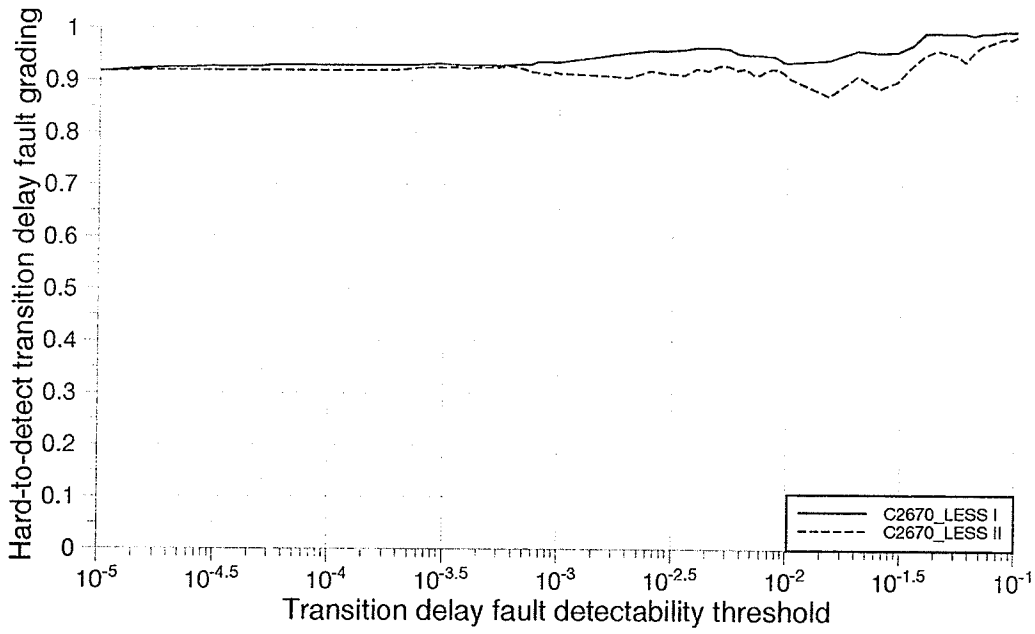


Figure 5.6: The hard-to-detect transition gate delay fault grading curve diagram of C2670 benchmark circuit

region successfully grades nearly all easy-to-detect faults for all given detectability thresholds. The estimation technique considering line correlation within each fanout free region gives very good fault grading results.

Extensive experiments on both redundant and non-redundant versions of the benchmark circuits C2670 and C7552 with the estimation method considering line correlation within each fanout free region have been implemented for a large number of random patterns. Table 5.4 illustrates the transition fault coverage of these benchmark circuits obtained by fault simulation and the estimation technique considering line correlation within each fanout free region without sampling with up to *1,000,000*

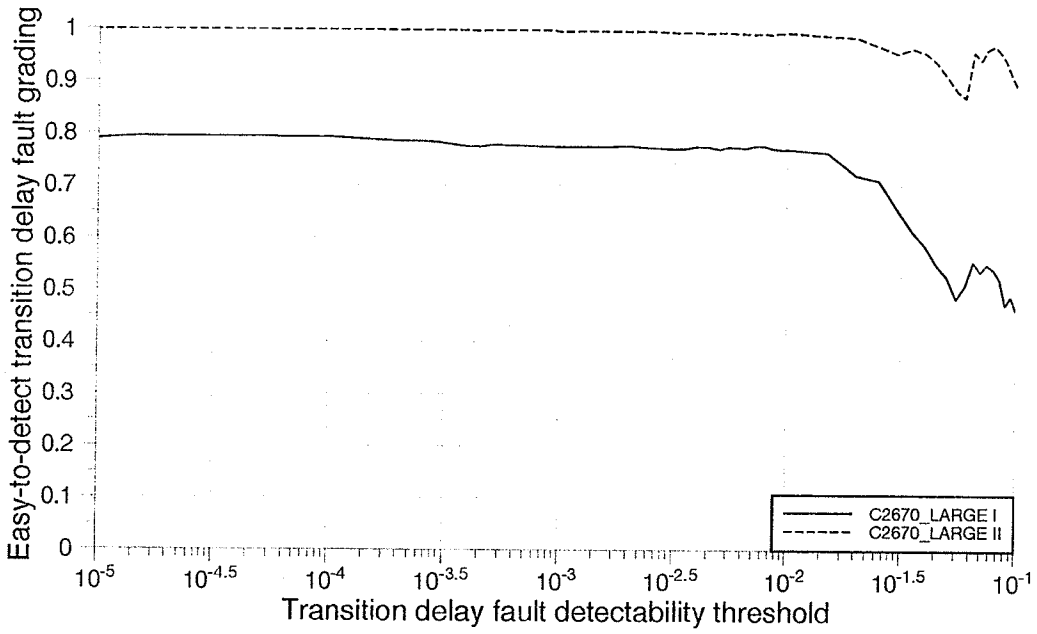


Figure 5.7: The easy-to-detect transition gate delay fault grading curve diagram of C2670 benchmark circuit

random patterns. The maximum error between the estimation and accurate fault simulation is within 2.3% throughout the whole range. Table 5.5 presents undetected fault grading results after a number of random patterns. The majority of undetected faults can be identified, and nearly all undetected faults can be identified for the non-redundant version of the circuits with the estimation technique considering line correlation within each fanout free region. Table 5.6 illustrates the graded detectabilities of the undetected faults after 1 million random patterns with the estimation technique considering line correlation within each fanout free region. Most of them are correctly graded.

Circuit Name	10016		100000		500000		1000000	
	EFC	SIM	EFC	SIM	EFC	SIM	EFC	SIM
C2670	82.88	81.41	83.24	81.84	87.94	88.17	89.60	89.82
C7552	95.83	93.56	97.45	95.36	97.80	96.23	97.98	96.71
c2670_nr	82.40	82.45	82.85	82.99	88.73	90.91	90.82	92.99
c7552_nr	95.14	95.34	97.09	97.23	97.87	98.06	98.33	98.50

Table 5.4: Estimated fault coverage considering line correlation within each fanout free region compared to fault simulation

Circuit Name	FLT NO.	10016	100000	500000	1000000
C2670	4412	756/820	735/801	449/522	368/449
C7552	13112	561/845	314/608	280/494	259/432
c2670_nr	3510	616/616	597/597	312/319	231/246
c7552_nr	12545	541/585	326/348	228/244	181/188

Table 5.5: Grading undetected faults for various numbers of random patterns with the estimation technique considering line correlation within each fanout free region

Experiments have also been implemented on estimating path delay fault coverage on the ISCAS85 benchmark circuits with the estimation method considering line correlation within each fanout free region. Table 5.7 shows the path delay fault coverage results for the ISCAS85 benchmark circuits obtained by accurate fault simulation[183] and by the proposed technique. The estimated path delay fault coverages of the various benchmark circuits are comparable to those obtained by accurate fault simulation. For most of the circuits, the estimation technique gives a reasonably good estimation,

Circuit Name	Detectability			
	10^{-2} *to* 0.06	10^{-4} *to* 10^{-2}	10^{-6} *to* 10^{-4}	$< 10^{-6}$
C2670	6	58	17	368
C7552	11	90	72	259
c2670_nr	0	0	15	231
c7552_nr	0	0	7	181

Table 5.6: Number of undetected faults with estimated detectabilities in range indicated

and for the circuits with relative small path delay fault set the estimated path delay fault coverages are very accurate.

Table 5.8 illustrates the estimated path delay fault coverage of a sampled fault subset of the benchmark circuits. The fault coverages in table 5.8 are consistent with those in table 5.7. The proposed estimation method is statistically reliable and stable.

In general, the estimation method can supply very useful fault grading information. Its performance for fault coverage estimation and fault grading on non-redundant circuits is better than that of redundant circuits. Also, the estimation method considering line correlation within each fanout free region gives more accurate information on the fault grading than that considering line correlation within each gate.

Fault grading for the purposes of test point insertion or logic modification for enhanced detectability can be achieved by considering fault sets at a gate or a line. Information about any hard-to-detect faults in a line or a gate obtained by the estimation grading method is sufficient for guiding logic modification such as test point

Circuit Name	*Fault Number*	10016		20000		50000	
		SIM	EFC	SIM	EFC	SIM	EFC
C880	17284	40.18	36.69	48.84	48.62	60.74	61.63
C1355	8346432	3.85	3.11	6.16	5.64	9.53	11.30
C1908	1458114	2.59	1.54	4.34	2.68	8.09	5.44
C2670	1359920	3.81	1.69	4.69	1.99	5.87	2.38
C5315	2682610	5.53	3.64	7.47	5.68	9.85	10.06
C7552	1452988	10.84	5.25	11.67	8.48	12.32	15.65
c1355_nr	6756672	4.60	3.29	7.29	5.88	11.20	11.52
c1908_nr	1458050	2.57	1.53	4.31	2.66	8.04	5.40
c2670_nr	34384	63.95	61.28	71.47	70.48	81.89	82.26
c5315_nr	2507188	5.56	3.79	7.47	5.87	9.78	10.25
c7552_nr	1310178	10.38	5.35	11.19	8.60	11.84	15.99

Table 5.7: Fault simulation and estimation of non-robust path delay fault coverage for the ISCAS85 benchmark circuits

insertion.

The CPU time used in estimating the detectabilities of the benchmark circuits is measured on a SUN 3/260 and compared to the CPU time required by accurate gate delay fault simulation[98] on a Micro-VAX workstation. As shown in table 5.9, CPU time used by both the estimating method and accurate transition gate delay fault simulation[98] on the ISCAS85 benchmark circuits with 20,000 patterns is given. The computation time of the estimation method is comparable to accurate

Circuit Name	*Fault Number*	*Number of Patterns*		
		10016	20000	50000
C1355	259826	3.09	5.61	11.24
C1908	45384	1.56	2.72	5.53
C2670	84596	1.64	1.93	2.32
C5315	83772	4.05	6.31	11.04
C7552	45234	5.35	8.62	15.75
c1355_nr	210868	3.30	5.88	11.52
c1908_nr	45384	1.54	2.69	5.49
c5315_nr	78250	4.23	6.56	11.32
c7552_nr	40832	5.37	8.63	15.99

Table 5.8: Estimation of path delay fault coverage with sampling fault subsets for the ISCAS85 benchmark circuits

fault simulation and can be further improved by optimized program code.

5.7 Summary

In this chapter, an efficient method to estimate both transition gate delay fault and path delay fault detectabilities are proposed. The transition probability, transition sensitization probability and a strategy to handle different kinds of fan-out stems in calculating observation probabilities were given. Motivation for this work was presented as augmenting existing techniques for improving testability using fault simulation and related design for test methods. Results of estimated fault coverage were

Circuit Name	Estimation Method	Accurate Simulation Method
C880	29.22sec.	55.5sec.
C1355	43.47sec.	116.3sec.
C1908	62.27sec.	150.3sec.
C2670	90.98sec.	463.8sec.
C3540	122.72sec.	358.6sec.
C5315	193.57sec.	322.6sec.
C6288	241.15sec.	445.9sec.
C7552	271.58sec.	769.6sec.

Table 5.9: CPU - time used in the estimation method and the accurate simulation method for 20,000 random patterns

presented and compared to results obtained from direct fault simulation. These results are comparable for a reasonable number of random test vectors(e.g. 10,000, 100,000, 500,000, and 1,000,000) and in the majority of cases within 2.3% of the fault coverage obtained by fault simulation for redundant circuits and within 0.6% of fault coverage from fault simulation for non-redundant circuits. Based on the experimental results on benchmark circuits, the performance of the estimation method for fault coverage and fault grading for non-redundant circuits is better than that for redundant circuits. The estimation method can supply very useful fault grading information for logic modification in order to enhance the testability of circuits.

Chapter 6

Summary and Future Work

6.1 Summary

This thesis investigates issues relating to stuck-open fault and delay fault test generation with Hopfield neural networks, efficient multiple scan chain skewed-load delay test schemes and statistical delay fault detectability analysis. These issues are important in various BIST aspects.

The primary contributions of this thesis are:

- An automatic test pattern generation technique for stuck-open and delay faults using neural network models was proposed and demonstrated.
- A technique augmenting scan path shift register latches (SRLs) in a multiple scan chain skewed-load delay fault testing scheme with a one-level XOR network was proposed and demonstrated.
- A statistical transition gate delay and path delay fault detectability estimation technique was proposed and demonstrated.

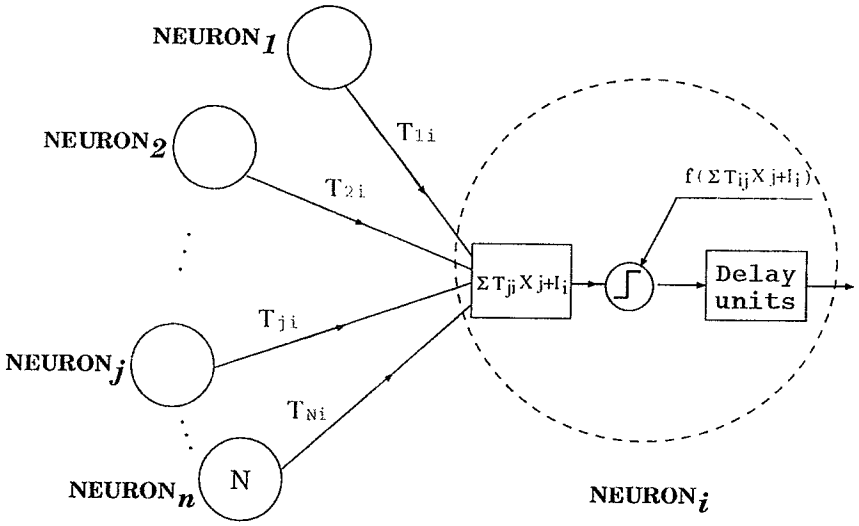


Figure 6.1: A modified neural model with nominal delay units

6.2 Future Work

6.2.1 Test Pattern Generation for Small Gate Delay Faults

Delay fault testing addresses defects which cause logic circuits to fail when operating at the system speeds. The extra delays caused by these defects can be considered quantitatively with the help of the timing analysis in the logic circuits in the delay fault test generation procedure. As shown in figure 6.1, a modified neural model with nominal delay units could be used for quantitative small gate delay fault test pattern generation. It can be implemented by adding nominal timing constraints in the equation (3.2) or (3.5) during test generation with Hopfield neural networks.

6.2.2 Statistically and Quantitatively Small Gate Delay Fault Testability Analysis

The proposed statistical delay fault detectability analysis technique could be extended to analyze quantitatively small delay fault detectability of logic circuits by using multiple-value logic to describe various relevant possible transitions. This research will be useful to achieve more accurately delay fault detectabilities.

6.2.3 Statistical Testability Analysis Guided Test Point Insertion

The problem of how to efficiently and optimally select test point insertion sites with minimum cost is still to be investigated. Since the proposed statistical delay fault detectability analysis technique can accurately grade faults in general, a technique using statistical testability analysis to guide test point insertion for combinational circuits to improve testability can be formulated as follows:

(1): Calculate controllabilities and sensitization probabilities of lines in the circuit based on fault-free simulation of the circuit with pseudo-random CA or LFSR patterns. Stop when $\frac{|P1(0)_{N1}(l_i)-P1(0)_{N2}(l_i)|}{P1_{N2}(l_i)} \leq \varepsilon$ is satisfied for two different sampling patterns N_1 and N_2 ($N_2 > N_1$ and ε is a given relative error threshold value) or maximum iteration number N is reached.

(2): Calculate observabilities and detectabilities of lines based on estimation of controllabilities and sensitization probabilities of lines obtained in **(1)**, and set up the list of lines in which detectabilities are lower than the threshold value.

(3): Select a line from the list. If its controllability is less than the threshold value, specify it as a control point (if its observability is less than the threshold value specify it as an observation point, or if both are less than the threshold value, specify it as a control point and an observation point in turn). Then estimate the updated detectabilities of all lines and count the number of lines in which detectabilities are lower than the threshold value after selecting this test point. Compare the number with one obtained in previous iteration. If it is less than that obtained in previous iteration, record this site, otherwise discard it.

The updated detectabilities of all lines can be estimated by

for an AND gate

$$P1^{new}(m) = \prod_{i=1}^{N} P1^{old}(m) * \frac{P1^{new}(g_i)}{P1^{old}(g_i)} \tag{6.1}$$

for a NOR gate

$$P1^{new}(m) = \prod_{i=1}^{N} P1^{old}(m) * \frac{1 - P1^{new}(g_i)}{1 - P1^{old}(g_i)} \tag{6.2}$$

for an OR gate

$$P1^{new}(m) = 1 - \prod_{i=1}^{N}(1 - P1^{old}(m)) * \frac{1 - P1^{new}(g_i)}{1 - P1^{old}(g_i)} \tag{6.3}$$

for a $NAND$ gate

$$P1^{new}(m) = 1 - \prod_{i=1}^{N}(1 - P1^{old}(m)) * \frac{P1^{new}(g_i)}{P1^{old}(g_i)} \tag{6.4}$$

the sensitization of g_j for AND and $NAND$ gates

$$S^{new}(g_j) = \prod_{i=1, i\neq j}^{N} S^{old}(m) * \frac{P1^{new}(g_i)}{P1^{old}(g_i)} \tag{6.5}$$

and sensitization of g_j for OR and NOR gates

$$S^{new}(g_j) = \prod_{i=1, i \neq j}^{N} S^{old}(m) * \frac{1 - P1^{new}(g_i)}{1 - P1^{old}(g_i)} \tag{6.6}$$

(4): After trying all candidate lines in the list at (3), select one which gives the fewest number of lines in which detectabilities are lower than the threshold value at the present trial. Insert a test point in this site, and calculate updated controllabilities, observabilities, and detectabilities for the modified circuit.

(5): If the list is empty or the maximum number of test points are reached, the procedure stops, otherwise go to (3).

The above test point insertion procedure can be incorporated with timing analysis to set the constraints; it specifies that no controllable test points are allowed to be inserted on a critical path to reduce the effect on system performance.

6.2.4 Generating Optimal Test Pattern Pairs for Enhancing Delay Fault Testing

In addition, research focusing on weighted random pattern or re-seeding and reprograming LFSRs for delay fault test can be initiated in future. Savir[184] proposes a bidirectional double latch design as shown in figure 6.2, where an extra control signal SH is added. The control signal SH controls shift direction of the SRLs. When SH=0, each SRL accepts the shift data from its predecessor SRL, and when SH=1, each SRL accepts the shift data from its successor SRL. A small adjustment can be made on Savir's design of the SRLs to facilitate efficient delay fault testing. The control signal, SH, can be set such that when SH=1, each SRL accepts the shift data from its $-L2$ output.

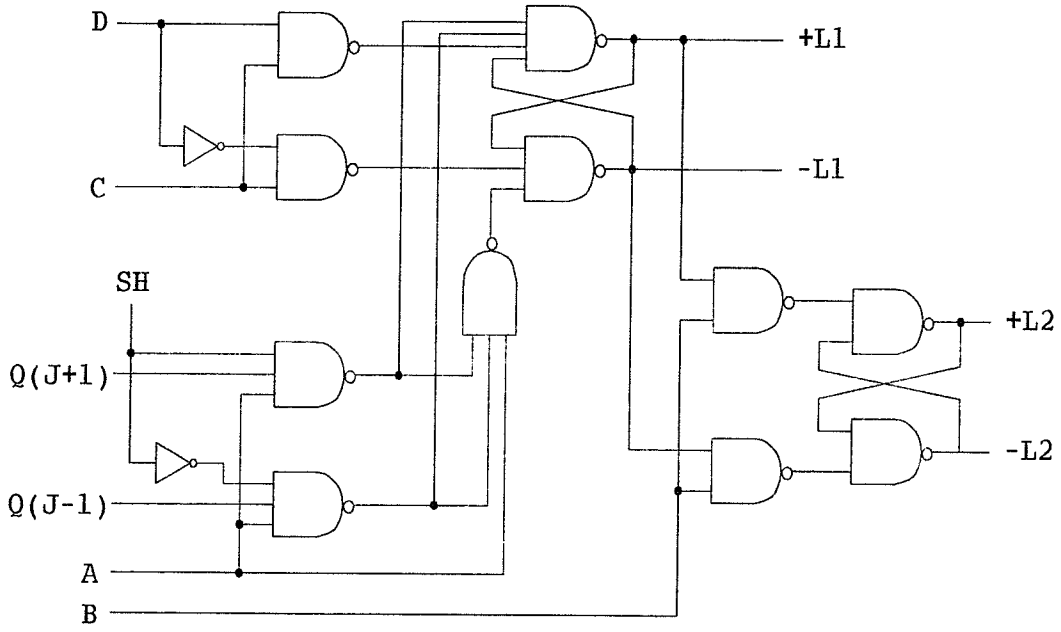


Figure 6.2: Savir's $NAND$ design of the bidirectional double latch

A delay test procedure can be formulated as follows:

- Set SH=0, and shift inverse test data from a weighted random pattern generator or a re-seeded and reprogramed LFSR to the m SRLs by clocking clocks A and B m cycles as an initial pattern of a delay test.
- Set SH=1, and shift data from its $-L2$ output for each SRL by one cycle. This will be the test pattern of the delay test.
- Set SH=0 again, clock the system clock C to capture the responses, and shift the responses out to the MISR for compaction.

Based on the above test procedure, the initial pattern of a delay test is obtained by the inverse of the optimized test pattern generated by either a weighted random

pattern generator or a re-seeding re-programable LFSR which is calculated to activate the hard-to-detect stuck-at faults. Hence, the inversed pattern will set these faults. Subsequently, the two patterns will make an optimal delay fault testing pair.

6.2.5 Synthesis Testability Features in High Level Design

With the proliferation of high-level synthesis software[185] for digital systems, it is becoming common practice to design circuits using register-level components. Research on incorporating testability at the stage of high-level synthesis should be the focus since high-level synthesis programs explore a large design space. It is also necessary to consider testability on register-level components to reject inferior designs at the early stages of design or to insert testability features hierarchically.

Bibliography

[1] T.W. Williams and K.P. Parker, "Design for Testability — A Survey", *Proc. of IEEE,* Vol.71, No.1, pp.-98-112, Jan., 1983.

[2] P.H. Bardell, W.H. McAnney, and J. Savir, "Built-In Test for VLSI: Pseudorandom Techniques", *John Wilcy & Sons,* 1987.

[3] M. Abramovici, M.A. Breuer, and A.D. Friedman, "Digital Systems Testing and Testable Design", *Computer Science Press,* New York 1990.

[4] E.B. Eichelberger, E. Lindbloom, J.A. Waicukauski, and T.W. Williams, "Structured Logic Testing", *Prentice Hall, Englewood Cliffs,* 1991.

[5] E. Trischer, "Incomplete Scan Path with an Automatic Test Generation Methodology", *Int'l Test Conference,* pp.152-158, 1980.

[6] V.D. Agrawal, K.-T. Cheng, D.D. Johnson, and T. Lin, "A Complete Solution to the Partial Scan Problem", *Int'l Test Conference,* pp.44-51, 1987.

[7] K.-T. Cheng and V.D. Agrawal, "An Economical Scan Design for Sequential Logic Test Generation", *Proc. of Int'l Fault Tolerant Computing Symp.,* pp.28-35, 1989.

[8] E.B. Eichelberger and T.W. Williams, "A Logic Design Structure for LSI Testing", *Proc. of Design Automation Conference,* pp.462-468, 1977.

[9] S. DasGupta, R.G. Walther, and T.W. Williams, "An Enhancement to LSSD and Some Applications Of LSSD in Reliability", *Proc. of Int'l Fault Tolerant Computing Symp.,* pp.32-34, 1981.

[10] R. Gupta and M.A. Breuer, "BALLAST: A Methodology for Partial Scan Design", *Proc. of Int'l Fault Tolerant Computing Symp.*, pp.118-125, 1989.

[11] V.D. Agrawal and S.C. Seth, Test Generation for VLSI Chips, *Computer Society Press*, Washington, D.C., 1988.

[12] J.L.A. Hughes, "Multiple Fault Detection Using Single Fault Test Sets", *IEEE Trans. on Computer Aided Design*, Vol.7, pp.100-108, 1988.

[13] E.P. Hsieh, R.A. Rasmussen, L.J. Vidunas, and W.T. Davis, "Delay Test Generation", *Proc. of Design Automation Conference*, pp.486-491, 1977.

[14] G.L. Smith, "Model for Delay Faults", *Proc. Int. Test Conference*, pp.342-349, 1985.

[15] A.K. Pramanick and S.M. Reddy, "On the Detection of Delay Faults", *Int'l Test Conference*, pp.845-856, 1988.

[16] J. Savir and S. Patil, "Scan-Based Transition Test", *IEEE Trans. on Computer Aided Design*, Vol.12, pp.1232-1241, 1993.

[17] J. Savir and W.H. McAnney, "Random Pattern Testability of Delay Faults", *Proc. Int'l Test Conference*, pp.263-273, 1986.

[18] V.S. Iyengear, B.K. Rosen, and J.A. Waicukauski, "On Computing the Sizes of Detected Delay Faults", *IEEE Trans. on Computer Aided Design*, Vol.9, pp.299-312, 1990.

[19] W.W. Mao and M.D.Ciletti, "A Variable Observation Time Method for Testing Delay Faults", *Proc. of Design Automation Conference*, pp.728-731, 1990.

[20] F. Fink, K. Fuchs, and M.H. Schulz, "An Efficient Parallel Pattern Gate Delay Fault Simulator with Accelerated Detected Fault Size Determination Capabilities", *Proc. of European Test Conference*, pp.171-180, 1991.

[21] D. Dumas, P. Girard, C. Landrault, and S. Pravossoudovich, "An Implicit Delay Fault Simulation Method with Approximate Detection Threshold Calculation", *Proc. Int'l Test Conference*, pp.705-713, 1986.

[22] K. Fuchs, F. Fink, and M.H. Schulz, "DYNAMITE: An Efficient Automatic Test Pattern Generation System for Path Delay Faults", *IEEE Trans. on CAD,* Vol.10, No.10, pp.1323-1335, October 1991.

[23] D.R. Schertz and G. Metze, "A New Representation for Faults in Combinational Digital Circuits", *IEEE Trans. on Computers,* Vol.21, No.8, pp.858-866, August 1972.

[24] A. Lioy, "Advanced Fault Collapsing", *IEEE Design & Test of Computers,* pp.64-71, 1992.

[25] A. Lioy, "On the Equivalence of Fanout-Point Faults", *IEEE Trans. on Computers,* Vol.42, No.3, pp.268-271, March 1993.

[26] *Fault Tolerant Computing: Theory and Application*, edited by D.K.Pradhan, Prentice-Hall, A Division of Simon & Schuster, Inc. Englewood Cliffs, New Jersey, 1986.

[27] V.D. Agrawal, C.R. Kime, and K.K. Saluja, "A Tutorial on Built-In Self-Test, Part I: Principles", *IEEE Design & Test of Computers,* pp.73-82, March 1993.

[28] V.D. Agrawal, C.R. Kime, and K.K. Saluja, "A Tutorial on Built-In Self-Test, Part II: Applications", *IEEE Design & Test of Computers,* pp.69-77, June 1993.

[29] Y. Zorian and A. Ivannov, "Programmable Space Compaction for BIST", *Proc. of Int'l Fault Tolerant Computing Symp.,* pp.340-349, 1993.

[30] E.J. McCluskey, "Built-In Self-Test Techniques", *IEEE Design & Test of Computers,* pp.21-36, June 1985.

[31] J.P. Roth, "Diagnosis of Automata Failures: A Calculus and a Method", *IBM J. Research and Development,* Vol.10, pp.278-291, 1966.

[32] P. Goel, "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits", *IEEE Trans. on Computer,* Vol. C-30, pp.215-222, March 1981.

[33] H. Fujiwara and T. Shimono, "On the Acceleration of Test Generation Generation Algorithms", *IEEE Trans. on Computer,* Vol. C-32, pp.1137-1144, Dec. 1983.

[34] M.H. Schulz, E. Trischler, and T.M. Sarfert, "SOCRATES: A Highly Efficient Automatic Test Pattern Generation System", *IEEE Trans. on CAD,* Vol.7, No.1, pp.126-137, Jan. 1988.

[35] W. Kuntz and D.K. Pradhan, "Accelerated Dynamic Learning for Test Pattern Generation in Combinational Circuits", *IEEE Trans. on CAD,* Vol.12, No.5, pp.684-694, Jan. 1993.

[36] T. Larrabee, "Test Pattern Generation Using Boolean Satisfiability", *IEEE Trans. on CAD,* Vol.7, No.1, pp.4-15, Jan. 1992.

[37] S.T. Chakradhar, V.D. Agrawal, and S.G. Rothweiler, "A Transitive Closure Algorithm for Test Generation", *IEEE Trans. on CAD,* Vol.12, No.7, pp.1015-1028, July 1993.

[38] S.W. Golomb, *Shift Register Sequences,* Aegean Park Press, Laguna Hills, California, 1982.

[39] P.D. Hortensius, R.D. McLeod, W.Pries, D.M. Miller, and H.C. Card, "Cellular Automata-Based Pseudorandom Number Generators for Built-In Self-Test", *IEEE Trans. on Computer Aided Design,* Vol.8, pp.842-859, 1989.

[40] P.D. Hortensius, R.D. McLeod, and H.C. Card, "Parallel Random Number Generations for VLSI Systems Using Cellular Automata", *IEEE Trans. on Computer,* Vol.38 C-10, pp.1466-1472, Oct. 1989.

[41] P.H. Bardell, "Analysis of Cellular Automata Used as Pseudorandom Pattern Generators", *Proc. of Int. Test Conference,* pp.762-768, 1990.

[42] S. Zhang and D.M. Miller, "A Comparison of LFSR amd Cellular Automata BIST", *Proc. of Canadian Conference on VLSI,* pp.8.4.1-8.4.9, 1990.

[43] Z. Zhang and R.D. McLeod, "Estimating Stuck-Open Fault Coverage for CMOS Combinational Circuits", *Proc. of Canadian Conference on VLSI,* pp.8.3.1-8.3.8, 1990.

[44] Z. Zhang, R.D. McLeod, and W. Kinsner, "Simulation Oriented Diagnosis and Analysis of Stuck-Open Faults in CMOS Combinational Circuits", *Proc. of 5th Test Workshop, New Directions for IC Testing,* Ottawa, Canada, 1991.

[45] F. Brglez, C. Gloster, and G. Kedem, "Hardware-Based Weighted Random Pattern Generation for Boundary Scan", *Proc. of Int. Test Conference,* pp.264-274, 1989.

[46] B.H. Seiss, P.M. Trouborst, and M.H. Schulz, "Test Point Insertion for Scan-Based BIST", *Proc. of Euro. Test Conference,* pp.253-262.

[47] T.W Williams, W. Daehn, M. Gruetzner, and C.W. Starke, "Aliasing Errors in Signature Analysis Registers", *IEEE Design & Test of Computers,* pp.39-45, April 1987.

[48] P.D. Hortensius, R.D. McLeod, and B.W. Podaima, "Cellular Automata Circuits for Built-In Self-Test", *IBM J. Research and Development,* Vol.34, pp.389-405, 1990.

[49] B. Konemann, "LFSR-Coded Test Patterns for Scan Designs", *Proc. of Euro. Test Conference,* pp.237-242 1991.

[50] S. Hellebrand, S. Tarnick, J. Rajski, and B. Courtois, "Generation of Vector Patterns Through Reseeding of Multiple-Polynomial Linear Feedback Shift Registers", *Proc. of Int. Test Conference,* pp.120-129, 1992.

[51] P.D. Hortensius, R.D. McLeod, and H.C. Card, "Cellular Automata-Based Signature Analysis for Built-In Self Test", *IEEE Trans. on Computer,* Vol.39, pp.1273-1483, Oct. 1990.

[52] M. Serra, T. Slater, J. Muzio, and D.M. Miller, "The Analysis of One-Dimensional Linear Cellular Automata and Their Aliasing Properties", *IEEE Trans. on Computer Aided Design,* Vol.9, pp.768-778, 1990.

[53] J.J. LeBlanc, "LOCST: A Built-In Self-Test Technique", *IEEE Design & Test of Computers,* pp.234-241, Nov. 1984.

[54] B. Konemann, J. Mucha, and G. Zwiehoff, "Built-In Logic Block Techniques", *Proc. Int. Test Conference,* pp.37-41, 1979.

[55] K. Kim, D.S. Ha, and J.G. Tront, "On Using Signature Registers as Pseudorandom Pattern Generators in Built-In Self-Testing", *IEEE Trans. on Computer Aided Design,* Vol.7, pp.919-928, 1988.

[56] A. Krasniewski and A. Pilarski, "Circular Self-Test Path: A Low-Cost BIST Technique for VLSI Circuits", *IEEE Trans. on Computer Aided Design,* Vol.8, pp.46-55, 1989.

[57] A. Pilarski, A. Krasniewski, and T. Kameda, "Estimating Testing Effectiveness of the Circular Self-Test Path Technique", *IEEE Trans. on Computer Aided Design,* Vol.10, pp.1301-1317, 1992.

[58] A. Pierzynska, K. Gaj, and A. Pilarski, "Estimating Testing Effectiveness of the Circular Self-Test Path Technique", *Technical Report,* CSS/LCCR TR93-08, Simon Fraser University, Burnaby, B.C., Canada, July 1993.

[59] P.H. Bardell and W.H. McAnney, "Self-Testing of Multichip Logic Modules", *Proc. of Int'l Test Conference,* pp.200-204, 1982.

[60] P.H. Bardell and M.J. Lapointe, "Production Experience with Built-In Self-Test in the IBM ES/9000 System", *Proc. of Int'l Test Conference,* pp.28-36, 1991.

[61] H.D. Schnurmann, E. Lindbloom, and R.G. Carpenter, "The Weighted Random Test-Pattern Generator", *IEEE Trans. on Computers,* C-24, pp.695-700, 1975.

[62] H.-J. Wunderlich "Multiple Distributions for Biased Random Test Patterns", *Proc. of Int'l Test Conference,* pp.236-244, 1988.

[63] R. Lisanke, F. Brglez, A.J. Degeus, and D. Gregory, "Testability-Driven Random Test-Pattern Generation", *IEEE Trans. on Computer Aided Design,* Vol.6, pp.1082-1087, 1987.

[64] F. Brglez, C. Gloster, and G. Kedem, "Built-In Self-Test with Weighted Random Pattern Hardware", *Proc. Int'l Conference on Computer Design,* pp.161-166, 1990.

[65] P.T. Wagner, "Interconnect Testing with Boundary Scan", *Int'l Test Conference,* pp.52-57, 1987.

[66] C.S. Gloster and F. Brglez, "Boundary Scan with Built-In Self-Test", *IEEE Design & Test of Computers,* pp.266-274, Feb. 1989.

[67] J.-C. Lien and M.A. Breuer, "A Universal Test and Maintenance Controller for Modules and Boards", *IEEE Trans. on Industrial Electronics,* Vol.36, pp.231-240, 1989.

[68] K.P. Parker, "The Impact of Boundary Scan on Board Test", *IEEE Design & Test of Computers,* pp.18-30, Aug. 1989.

[69] C.M. Maunder and R.E. Tulloss, *The Test Access Port and Boundary Scan Architecture,* IEEE Computer Society Press, Los Alamitos, California, 1990.

[70] A.S. Hassan, V.K. Agarwal, B. Nadeau-Dostie, and J. Rajski, "BIST of PCB Interconnects Using Boundary-Scan Architecture", *IEEE Trans. on Computer Aided Design,* Vol.11, pp.1278-1288, 1992.

[71] S. Sehu, "On an Improved Diagnosis Program", *IEEE Trans. on Electronic Computer,* Vol. EC-12, pp.76-79, Feb. 1965.

[72] N.H. Weste and K. Eshraghian, *Principles of CMOS VLSI Design: A Systems Perspective (2nd Edition),* Addison-Wesley Publishing Company, 1993.

[73] H. Hao and E.J. McCluskey, "On the Modelling and Testing of Gate Oxide Shorts In CMOS Logic Gates", *IEEE Int'l Workshop on DFT on VLSI Systems,* pp.161-174, 1991.

[74] D.M. Miller, J.C. Muzio, and M. Serra, *Study on Design for Testability: Literature Survey,* Dept. of Computer Science, University of Victoria, 1993.

[75] B.D. Brown and R.D. McLeod, "Built-In Current Mode Circuits for I_{ddq} Monitoring", *IEEE Int'l Custom Integrated Circuits Conference,* pp.30.6.1-30.6.4, 1993.

[76] F.J. Ferguson and T. Larrabee, "Test Pattern Generation for Realistic Bridge Faults in CMOS ICs", *IEEE Int'l Test Conference,* pp.492-499, 1991.

[77] S. Chandra, K. Pierce, G. Srinath, H.R. Sucar, and V. Kulkarni, "CrossCheck: An Innovative Testability Solution", *IEEE Design and Test of Computers,* pp.56-68, 1993.

[78] D.B. Armstrong, "A Deductive Method of Simulating Faults in Logic Circuits", *IEEE Trans. on Computer,* Vol. C-21, pp.464-471, May 1972.

[79] E.G. Ulrich and T.G. Baker, "Concurrent Simulation of Nearly Identical Digital Networks", *Computer* Vol.7, No.4, pp.39-44, April 1974.

[80] J.A. Waicukauski, E.B. Eichelberger, D.O. Forlenza, E. Lindbloom, and T. McCarthy, "Fault Simulation for Structured VLSI", *VLSI Systems Design,* pp.20-32, Dec., 1985.

[81] K.J. Antreich and M.H. Schulz, "Accelerated Fault Simulation and Fault Grading in Combinational Circuits", *IEEE Trans. on Computer-Aided Design,* pp.704-711, Sept., 1987.

[82] F. Maamari and J. Rajski, "A Method of Fault Simulation Based on Stem Regions", *IEEE Trans. on Computer-Aided Design,* pp.212-220, Feb., 1990.

[83] H.K. Lee and D.S. Ha, "An Efficient, Forward Fault Simulation Algorithm Based on the Parallel Pattern Single Fault Propagation", *Int'l Test Conference,* pp.946-955, 1991.

[84] Algotronix Ltd., *Configurable Array Logic User Manual,* Edinburgh, U.K., 1991.

[85] Xilinx, *The Programmable Logic Data Book,* San Jose, U.S.A., 1994.

[86] R.W. Wieler, Z. Zhang, and R.D. McLeod, "Using an FPGA Based Computer as a Hardware Emulator for Built-In Self-Test Structures", *IEEE Int'l Workshop on Rapid System Prototyping,* pp.16-21, 1994.

[87] R.W. Wieler, Z. Zhang, and R.D. McLeod, "An Alternative Fault Simulation Tool: FPGA Based Hardware Emulator for Built-In Self-Test Structures", *Canadian Workshop on Field-Programmable Devices,* pp.3.3.1-3.3.7, 1994.

[88] R.W. Wieler, Z. Zhang, and R.D. McLeod, "Simulating Static and Dynamic Faults in Built-In Self-Test Structures with a FPGA Based Emulator", *Field-Programmable Logic: Architecture, Synthesis and Application,* Eds. by R.W. Hartenstein and M.Z. Servit, Lecture Notes in Computer Science, Springer-Verlag, Vol. 849, pp.240-250, 1994.

[89] R.W. Wieler, Z. Zhang, and R.D. McLeod, "Emulating Static Faults Using a Xilinx Based Emulator", *IEEE Symposium on FPGA for Custom Computing Machine,* 1995.

[90] R.L. Wadsack, "Fault Modeling and Logic Simulation of CMOS and MOS Integrated Circuits", *Bell System Technical Journal,* pp. 1449-1474, 1978.

[91] D. Baschiera and B. Coutois, "Advances in Fault Modelling and Test Pattern Generation for CMOS", *Int'l Conference on Computer Design,* pp.82-85, 1986.

[92] O.H. Ibarra and S.K. Sahni, "Polynomially Complete Fault Detection Problems", *IEEE Trans. on Computer,* Vol. C-24, pp.242-249, March 1975.

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

[94] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness,* W.H. Freeman & Company, New York, 1979.

[95] F.N. Najm and I.N. Hajj, "The Complexity of Fault Detection in MOS VLSI Circuits", *IEEE Trans. on CAD,* Vol.9, pp.995-1001, September 1990.

[96] S.T. Chakradhar, M.L. Bushnell, and V.D. Agrawal, "Toward Massively Parallel Automatic Test Generation", *IEEE Trans. on CAD,* Vol.9, pp.981-994, September 1990.

[97] J.A. Waicukauski, E. Lindbloom, B.K. Rosen, and V.S. Iyengar, "Transition Fault Simulation", *IEEE Design & Test of Computers,* pp.32-38, 1987.

[98] M.H. Schulz and F. Brglez, "Accelerated Transition Fault Simulation", *Proc. of Design Automation Conference,* pp.237-243, 1987.

[99] H. Cox and J. Rajski, "Stuck-Open and Transition Fault Testing in CMOS Complex Gates", *Proc. Int. Test Conference,* pp.688-694, 1988.

[100] S.M. Reddy, V.D. Agrawal, and S.K. Jain, "A Gate Level Model for CMOS Combinational Logic Circuits with Application to Fault Detection", *Proc. of Design Automation Conference,* pp.504-509, June 1984.

[101] H.-C. Shih and J.A. Abraham, "Fault Collapsing Techniques for MOS VLSI Circuits", *Proc. Int. Test Conference,* pp.370-375, 1986.

[102] W. Wu and C.L. Lee, "A Probabilistic Testability Measure for Delay Faults", *Proc. of Design Automation Conference,* pp.440-445, 1991.

[103] R. Palmer, A. Krough, and J. Hertz, *Introduction to the Theory of Neural Computation,* Addison-Wesley Publishing Company, 350 Bridge Parkey, Redwood, CA 9405, 1991.

[104] J.A. Freeman and D.M. Skapura, Neural Networks: Algorithms, Applications, and Programming Techniques, *Reading, MA: Addison-Wesley Publishing Company,* 1991.

[105] J.J. Hopfield, "Neurons with Graded Response Have Collective Computational Properties Like Those of Two State Neurons", *Proc. of Nat'l Academy of Sciences,* pp.3088-3092, May 1984.

[106] M. Abramovici, P.R. Menon, and D.T. Mill, "Critical Path Tracing: An Alternative to Fault Simulation", *IEEE Design & Test of Computers,* Vol.1 pp.83-93, Feb. 1984.

[107] P. Agrawal, V.D. Agrawal, and S.C. Seth, "DynaTaPP: Dynamic Timing Analysis With Partial Path Activation in Sequential Circuits", *Proc. of European Design Automation Conference,* pp.138-141, 1992.

[108] E. Arts and J. Korst, Simulated Annealing and Boltzmann Machine: A Stochastic Approach to Combinational Optimization and Neural Computing, *John Wiley & Sons,* 1990.

[109] F. Brglez, "On Testability Analysis of Combinational Networks", *Proc. of International Symposium on CAS,* pp.221-225,1984.

[110] C.J. Lin and S.M. Reddy, "On Delay Fault Testing in Logic Circuits", *IEEE Trans. on CAD,* Vol.6, pp.694-703, 1987.

[111] M. Serra and J.C. Muzio, "Space Compaction for Multiple-Output Circuits", *IEEE Trans. on CAD,* Vol.7, pp.1105-1113, 1988.

[112] D.A. Pucknell, Fundamentals of Digital Logic Design: with VLSI Circuit Applications, *Englewood Cliffs, NJ: Prentice Hall,* 1990.

[113] S.T. Chakradhar, V.D. Agrawal, and M.L. Bushnell, *Neural Models and Algorithms for Digital Testing,* Kluwer Academic Publishers, 1991.

[114] H. Fujiwara, "Three-Valued Neural Networks for Test Generation", *IEEE Int'l Conference on Fault Tolerant Computing,* pp.64-71, 1990.

[115] C.L.C. Cooper and M.L. Bushnell, "Neural Models for Transistor and Mixed-Level Test Generation", *IEEE Test Symposium,* pp.208-213, 1994.

[116] Z. Barzilai and B.K. Rosen, "Comparison of AC Self-Testing Procedures", *Int'l Test Conference,* pp.89-94, 1983.

[117] G.L. Craig and C.R. Kime, Pseudo-Exhaustive Adjacency Testing: A BIST Approach for Stuck-Open Faults", *IEEE Int'l Test Conference,* pp.126-137, 1985.

[118] S. Zhang, R. Byrne, and D.M. Miller, "BIST Generators for Sequential Faults", *Int'l Conference on Computer Design,* pp.260-263, 1992.

[119] J. Leestra, M. Koch, and T. Schwederski, "On Scan Path Design for Stuck-Open and Delay Fault Detection", *IEEE European Test Conference,* pp.201-210, 1993.

[120] J.A. Waicukauski and E. Lindbloom, "Transition Fault Simulation by Parallel Pattern Single Fault Propagation", *Int'l Test Conference,* pp.542-549, 1986.

[121] V.S. Iyengar, B.K. Rosen, and I. Spillinger, "Delay Test Generation 2 – Algebra and Algorithms", *Int'l Test Conference,* pp.867-876, 1988.

[122] W.W. Mao and M.D. Ciletti, "Arrangement of Latches in Scan-Path Design to Improve Delay Fault Coverage", *Int'l Test Conference,* pp.387-393, 1990.

[123] R. Gupta and M.A. Breuer, "Ordering Storage Elements in a Single Scan Chain", *Int'l Conference on Computer Aided Design,* pp.408-411, 1991.

[124] S. Narayanan, C. Njinda, M.A. Breuer, "Optimal Sequencing of Scan Registers", *Int'l Test Conference,* pp.293-303, 1992.

[125] J. Savir, "Skewed-Load Transition Test: Part I, Calculus", *Int'l Test Conference,* pp.705-713, 1992.

[126] S. Patil and J. Savir, "Skewed-Load Transition Test: Part II, Coverage", *Int'l Test Conference,* pp.714-722, 1992.

[127] E. K. Vida-Torku, "Impact of Boundary-Scan Design on Delay Test", *Int'l Test Conference,* pp.96-105, 1992.

[128] J. Savir and R. Berry, "At-Speed Test is Not Necessarily an AC Test", *Int'l Test Conference,* pp.722-728, 1991.

[129] S. Wolfram, "Statistical Mechanics of Cellular Automata", *Rev. Mod. Physics,* Vol. 55, No. 3, pp. 601-644, 1983.

[130] W. Pries, A. Thanailakis, and H.C. Card, "Group Properties of Cellular Automata and VLSI Applications", *IEEE Trans. on Computers,* Vol. 35, pp. 1013-1024, 1986.

[131] P. Bardell, "Design Considerations for Parallel Pseudorandom Pattern Generators", *Journal of Electronic Testing: Theory and Applications,* Vol.1, pp.73-87, 1990.

[132] B. Nadeau-Dostie, D. Burek, and A.S.M. Hassan, "ScanBist: A Multifrequency Scan-Based BIST Method", *IEEE Design & Test Computer,* pp.7-17, Jan. 1994.

[133] S. Zhang, D.M. Miller, and J.C. Muzio, "Determination of Minimum Cost One-Dimensional Linear Hybrid Cellular Automata", *IEE Electronics Letters,* Vol. 27, No. 18, pp. 1625-1627, 1991.

[134] K. Cattell and J.C. Muzio, "Synthesis of One-Dimensional Linear Hybrid Cellular Automata", *Dept. of Computer Science, University of Victoria,* 1994.

[135] K. Furuya and E.J. McCluskey, "Two-Pattern Test Capabilities of Automonous TPG Circuits", *IEEE Int'l Test Conference,* pp.704-711, 1991.

[136] Z. Barzilai, D. Coppersmith, and A. Rosenberg, "Exhaustive Bit Pattern Generation in Discontiguous Positions with Applications to VLSI Testing", *IEEE Trans. on Computers,* C-32, pp.190-194, 1983.

[137] E.J. McCluskey, "Verification Testing — A Pseudoexhaustive Test Technique", *IEEE Trans. on Computers,* C-33, pp.541-546, 1984.

[138] S.B. Akers, "On the Use of Linear Sums in Exhaustive Testing", *IEEE Int'l Symp. on Fault Tolerant Computing,* pp.148-153, 1985.

[139] L.T. Wang and E.J. McCluskey, "Condensed Linear Feedback Shift Register (LFSR) Testing — A Pseudoexhaustive Test Technique", *IEEE Trans. on Computers,* C-35, pp.367-370, 1986.

[140] R. Srinivasan, S.K. Gupta, and M.A. Breuer, "Novel Test Pattern Generator for Pseudo-Exhaustive Testing", *IEEE Int'l Test Conference,* pp.1050-1041, 1993.

[141] R. Srinivasan, S.K. Gupta, and M.A. Breuer, "An Efficient Partitioning Strategy for Pseudo-Exhaustive Testing", *IEEE Int'l Conference on DAC,* pp.242-248, 1993.

[142] P.H. Bardell, "Calculating the Effects of Linear Dependencies in m-Sequences Used as Test Stimuli", *IEEE Trans. on CAD,* Vol.11, pp.83-86, 1992.

[143] J. Wang, S. Fang, and W. Feng, "New Efficient Designs for XOR and XNOR Functions on the Transistor Level", *IEEE Journal of Solid-State Circuits,* Vol.29, No.7, pp.780-786, 1994.

[144] F. Brglez and H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortran", *Proc. of Int'l Symp. on CAS,* Kyoto, Japan, 1985.

[145] R. Kapur, J. Ferguson, and M. Abadir, "Trade-off Analysis of the Effectiveness of Testability Estimators", *IEEE European Test Conference,* pp.341-349, 1991.

[146] A. Majumdar and S. Sastry, "On the Distribution of Fault Coverage and Test Length in Random Testing of Combinational Circuits", *Proc. of Int'l Design Automation Conference,* pp.341-346, 1992.

[147] M. Damiani, P. Olivo, M. Favalli, and B. Riccò, "An Analytical Model for the Aliasing Probability in Signature Analysis Testing", *IEEE Trans. on Computer Aided Design,* Vol.8, pp.1133-1144, 1989.

[148] K.P. Parker and E.J. McCluskey, "Probabilistic Treatment of General Combinational Networks", *IEEE Trans. on Computers,* Vol. C-24, No.6, pp.668-670, 1975.

[149] P. Agrawal and V.D. Agrawal, "Probabilistic Analysis of Random Test Generation Method for Irredundant Combinational Logic Networks", *IEEE Trans. on Computer,* Vol. C-24, No.7, pp.691-695, 1975.

[150] L.H. Goldstein, "Controllability/Observability Analysis of Digital Circuits", *IEEE Trans. on Circuits and Systems,* Vol. CAS-26, No.9, pp.685-693, 1979.

[151] L.H. Goldstein and E. Thigpen, "SCOAP: Sandia Controllability/Observability Analysis Program", *Proc. of Int'l Design Automation Conference,* pp.190-196, 1980.

[152] J. Savir, G.S. Ditlow, and P.H. Bardell, "Random Pattern Testability", *IEEE Trans. on Computers,* Vol. C-33, No.1, pp.79-90, 1984.

[153] F. Brglez, P. Pownall, and R. Hum, "Applications of Testability Analysis: From ATPG to Critical Delay Path Tracing", *Int'l Test Conference,* pp.705-712, 1984.

[154] S.C. Seth, L. Pan, and V.D. Agrawal, "PREDICT-Probabilistic Estimation of Digital Circuit Testability", *Proc. of Int'l Fault Tolerant Computing Symp.,* pp.220-225, 1985.

[155] K.D. Wagner, C.K. Chen, and E.J. McCluskey, "Pseudorandom Testing", *IEEE Trans. on Computers,* Vol.36, pp.332-343, March, 1987.

[156] H-J. Wunderlich, "PROTEST: A Tool for Probabilistic Testability Analysis", *Proc. of Design Automation Conference,* pp.204-211, 1985.

[157] S.C. Seth, B.B. Bhattachaya, and V.D. Agrawal, "An Exact Analysis for Efficient Computation of Random-Pattern Testability in Combinational Circuits", *Proc. of Int'l Fault Tolerant Computing Symp.,* pp.318-323, 1986.

[158] S. Chakravarty and H.B. Hunt, "On the Computation of Detection Probability for Multiple Faults", *Int'l Test Conference,* pp.252-262, 1986.

[159] B. Krishnamurthy and I.G. Tollis, "Improved Techniques for Estimating Signal Probabilities", *IEEE Trans. on Computers,* pp.1041-1045, July, 1989.

[160] S.C. Seth and V.D. Agrawal, "A New Method for Computation of Probabilistic Testability in Combinational Circuits", *INTEGRATION, the VLSI Journal,* No. 7, pp.49-75, 1989.

[161] S. Ercolani, M. Favalli, M. Damiani, P. Olivo, and B. Ricco, "Estimation of Signal Probability in Combinational Network", *IEEE European Test Conference,* pp.132-138, 1989.

[162] K.M. Butler and M.R. Mercer, "The Influences of Fault Type and Topology on Fault Model Performance and the Implications to Test and Testable Design", *Proc. of Design Automation Conference,* pp.673-678, 1990.

[163] H. Fujiwara, "Computational Complexity of Controllability/Observability Problems for Combinational Circuits", *Proc. of Int'l Fault Tolerant Computing Symp.,* pp.64-69, 1988.

[164] P. Camurati, P. Prinetto, and M.S. Reorda, "Exact Probabilistic Testability Measures for Multi-Output Circuits", *Journal of Electronic Testing: Theory and Applications,* pp.229-234, 1990.

[165] R. Krieger, B. Becker, and R. Sinković, "A BDD – Based Algorithm for Computation of Exact Fault Detection Probabilities", *Proc. of Int'l Fault Tolerant Computing Symp.,* pp.186-195, 1993.

[166] S.K. Jain and V.D. Agrawal, "STAFAN: An Alternative to Fault Simulation", *Proc. of Design Automation Conference,* pp.18-23, June 1984.

[167] J.A. Waicukauski, E.B. Eichelberger, D.O. Forlenza, E. Lindbloom, and T. McCarthy, "A Statistical Calculation of Fault Detection Probabilities by Fast Fault Simulation", *Int'l Test Conference,* pp.779-784, 1985.

[168] S. Sastry, and M. Breuer, "Detectability of CMOS Stuck-Open Faults Using Random and Pseudorandom Test Sequences", *IEEE Trans. on Computer-Aided Design,* pp.933-946, Sept., 1988.

[169] V.D. Agrawal, "Statistical Testing", *Testing and Diagnosis of VLSI and ULSI, Kluwer Academic Publishers,* pp.33-47, 1988.

[170] S.K. Jain and D.M. Singer, "Characteristics of Statistical Fault Analysis", *Int'l Conference on Computer Design,* pp.24-30, 1986.

[171] L.M. Huisman, "The Reliability of Approximate Testability Measures", *IEEE Design & Test of Computer,* pp.59-67, 1988.

[172] R.K. Gaede, M.R. Mercer, and B. Underwood, "Calculation of Greatest Lower Bounds Obtainable by the Cutting Algorithm", *IEEE Int'l Test Conference,* pp.498-505, 1986.

[173] K. Heragu, V.D. Agrawal, and M.L. Bushnell, "FACTS: Fault Coverage Estimation by Test Vector Sampling", *IEEE Test Symp.,* pp.266-271, 1994.

[174] K. Heragu, V.D. Agrawal, and M.L. Bushnell, "An Efficient Path Delay Fault Coverage Estimator", *IEEE Design Automation Conference,* pp.516-521, 1994.

[175] K. Heragu, V.D. Agrawal, and M.L. Bushnell, "Statistical Methods for Delay Fault Coverage Analysis", *Int'l Conference on VLSI,* pp.166-170, 1994.

[176] G-J. Tromp and A.J. van de Goor, "Logic Synthesis of 100-percent Testable Logic Networks", *Int'l Conference on Computer Designs,* pp.428-431, 1991.

[177] J. Savir, "Good Controllability and Observability Do Not Guarantee Good Testability", *IEEE Trans. on Computers,* pp.1198-1200, Dec., 1983.

[178] J.M. Halbout, P.G. May, and G.L. Chiu, "Noncontact High-speed Waveform Measurements with the Picosend Photoelectron Scanning Electron Microscope", *IEEE J. of Quantum Electronics,* Vol.24, pp.234, 1988.

[179] J.M. Wiesenfeld, "Electro-optic Sampling of High-speed Device and Integrated Circuits", *IBM J. of Research and Development,* Vol.34, pp.141, 1990.

[180] G.E. Bridges and D.J. Thomson "High-Frequency Circuit Characterization using the AFM as a Reactive Near-Field Probe", *J. Ultramicroscopy,* Vol. 42, pp.321-328, 1992.

[181] G.E. Bridges, R.A. Said, and D.J. Thomson *Heterodyne Electrostratic Force Microscopy for Non-Contact High Frequency Integrated Circuit Measurement,* Electronics Letters, Vol.29, pp.1448-1449, 1993.

[182] M.H. Schulz and E. Auth, "Improved Deterministic Test Pattern Generation with Application to Redundancy Identification", *IEEE Trans. on CAD,* Vol.8, pp.811-816, 1989.

[183] S. Zhang and D.M. Miller, "A Fault Simulation Program for Combinational Circuits", *Dept. of Computer Science, University of Victoria,* Oct. 1991.

[184] J. Savir, "The Bidirectional Double Latch (BDDL)", *IEEE Trans. on Computers,* Vol.35, pp.65-66, 1986.

[185] Z. Zhang, R. Wieler, G. Jonatschick, and H. Poskar "Synthesizing Testability Features into a Design with the Synopsys Test Compiler", *IEEE WESCANEX, Communications, Power, and Computing,* pp.462-467, May 1995