EXPERIMENTAL RESULTS ON ALIASING ERRORS IN CIRCULAR BIST DESIGN

by

Rajiv D. Kothari

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Electrical Engineering

APPROVED:

Dr. D. S. Ha, Chairman

Dr. J. R. Armstrong

Dr. S.F. Midkiff

September 1991

Blacksburg, Virginia

ABSTRACT

The circular BIST design is a technique in which the existing circuit is modified, so that the processes of test generation and response compaction are carried out by the circuit being tested itself. Most response compaction techniques suffer from loss of information, known as aliasing. Aliasing is said to occur in a response compaction technique when the response generated by the circuit, under the presence of a fault, is different from its fault-free response, but this information is later lost during compaction, and the faulty compacted response at the end of the test session is identical to the fault-free compacted response.

A program to synthesize circular BIST hardware on general sequential circuits has been developed. A parallel fault simulator has been developed to detect aliasing errors in circular BIST design. Experimental results on aliasing probability in circular BIST design are reported for twenty-three sequential benchmark circuits.

ACKNOWLEDGMENTS

First and foremost, I wish to thank the chairman of my committee, Dr. D. S. Ha, without whose guidance, inspiration, and assistance, this project would not have been completed. Thanks also go to Dr. J. R. Armstrong and Dr. S. F. Midkiff for agreeing to serve on my committee and providing valuable comments.

No words can express my profound gratitude and love towards my parents for supporting me throughout this trying period.

Table of Contents

List of Illustrations

List of Tables

1.0 Introduction

The complexity of digital circuits has been increasing rapidly since the invention of transistors. The past decade in particular has seen astonishing growth in chip density. It is no longer surprising to hear of chips with a transistor count exceeding several million. It is well known that the complexity of testing is directly related to the complexity of the circuit being tested. This means that as the circuit complexity grows, the testing complexity grows with it. Testing today's VLSI circuits is thus a formidable challenge.

Conventional testing techniques involve applying a set of test patterns to the circuit and comparing the response of the circuit with it's pre-computed fault-free response. The main problems associated with this form of testing are listed below.

- The test pattern derivation process is complex.
- Large volumes of data (fault-free response and input test patterns) are required to be stored in the tester.
- The procedure of applying test patterns and comparing faulty and fault-free response takes time, and may not be able to run at the operating frequency of the circuit.
- Cost of test equipment is high.

The ideas of Design for Testability (DFT) techniques and built-in self-test (BIST) were developed to address these problems.

One of the most commonly used design for testability techniques for sequential circuits is the so called scan design technique [1,2]. This technique eases the test pattern generation process in addition to providing improved diagnosability. It requires the modification of existing flip-flops or registers in a circuit to form one (or several) scan chain(s), so that test patterns can be shifted into and out from these modified flip-flops through some primary input(s) and some primary output(s), respectively. If all flip-flops are included in the scan chain(s), the method is called full-scan, and if a subset of flip-flops is included in the chain(s), the method is called partial-scan. This technique increases the testability of the circuit, but still requires test pattern generation. The speed of testing is slow since each test pattern has to be shifted in and each response shifted out one bit at a time.

Built-in self-test techniques overcome some or all of the disadvantages associated with the testing techniques mentioned above. The basic idea of built-in self-test (BIST) is to place a test pattern generator(s) and a test response evaluator(s) on the same chip. During test mode, test patterns are generated and applied to the circuit under test. The test response is collected,

compacted, and analyzed internally. Built-in self-test is also known by other names such as built-in test (BIT), self-test, self-verification, and autonomous test. The main advantages of BIST are the elimination of the procedures for test pattern generation and test response evaluation (and hence the test equipment). In addition, BIST can be run at the same speed as the circuit under test, an important consideration for high speed circuits. These advantages are achieved at the cost of extra testing circuitry and/or speed degradation. Various methods have been proposed to reduce the test circuit overhead; for example Built-In Logic Block Observer (BILBO) [3,4], Simultaneous Self-Test (SST) [5], a technique based on LSSD (level sensitive scan design) [6], etc. The common idea in all these methods is the modification of existing circuitry, so that the modified circuit behaves as a test generator and/or a response evaluator during test mode. The problem of speed degradation can be alleviated by adding test circuitry at only non-critical paths in the circuit.

The circular BIST technique proposed by Krasniewski and Pilarski [7] is one such technique which modifies the existing circuit to enable the processes of test generation and response evaluation to be carried out on the chip itself. The basic idea behind circular BIST is the conversion of selected existing flip-flops in the circuit to multifunction modules (similar to BILBO), which are connected together to form a circular chain or a circular path. The circular chain formed by the modified flip-flops acts as a pseudorandom test pattern generator during test mode. Response compaction is carried out by the same circular chain simultaneously.

Most data compaction techniques suffer from a problem known as aliasing. Aliasing is said to occur in a signature analysis register or any other data compaction technique when the response generated by the circuit under the presence of a fault is different from its fault-free response, but during response compaction, this information is lost, and the signature at the end of the test session is identical to the fault-free signature. Low aliasing probability is clearly desirable in all response compaction techniques. Many methods for estimating aliasing

probability have been suggested under certain assumptions. These assumptions may or may not be reasonable. No definite experimental results have been published yet. The primary goal of this thesis is to conduct experiments to find the aliasing probability of the circular BIST design technique.

Chapter 2 gives background on general BIST techniques as well as a literature survey of papers published in the area of circular BIST design techniques. The test structure of circular BIST hardware is also explained. Aliasing errors are explained in more detail. Chapter 3 deals with a number of different tools developed for the purpose of conducting simulations. The two primary tools developed for this thesis, CBIST and PSIM, are examined in detail. A brief explanation of other ancillary tools is also given. Experimental results on aliasing probabilities are presented for twenty three benchmark circuits in Chapter 4. Chapter 5 concludes the thesis with a summary and a few suggestions for future work in the area of circular BIST.

2.0 Background

Chapter 2 begins with a general discussion of built-in self-test. Section 2.2 deals with various aspects of circular built-in self-test. Circular BIST is a new technique and not much research has been done in this specific area. Only a handful of papers have been published which deal specifically with this subject. Section 2.2.1 gives a survey of techniques and results presented in three of these papers. Section 2.2.2 examines the test circuit structure implemented by the CBIST tool developed for this research. Section 2.3 gives a brief explanation of aliasing errors. The motivation for the work done for this thesis is given in Section 2.4.

2.1 Background on Built-in Self-test Techniques

When test pattern generation and test response evaluation are carried out on the circuit under test itself, the method of testing is called built-in self-test. It has several advantages over conventional forms of testing, some of which are listed below.

- Time and effort needed to derive test patterns is saved since patterns are generated on the chip automatically.
- The need to store high volumes of test response data and fault-free response is eliminated due to response compaction.
- The cost of external test equipment is drastically reduced.
- Field testing of the circuit is made easier.
- Speed of testing is much higher (the same as the chip operating speed) than possible with external testers.
- Since testing is conducted at the chip operating speed, some timing hazards which would not be detected at lower test frequency, may be detected.

The primary disadvantage of BIST is the area overhead associated with the additional test circuitry. The speed of the circuit may also be degraded during normal operation.

Many techniques for on-chip test pattern generation have been developed over the years. A ROM on the CUT (Circuit Under Test) can be used to store externally generated test patterns. It is practical for only a small number of test patterns and is not used commonly for current VLSI circuits. Instead of storing the patterns directly in the ROM, it is possible to store a program in the ROM to generate the patterns. This is only suitable for circuits with the capability of executing programs, such as a microprocessor. By far the most attractive and

prevalent technique for on-chip test pattern generation, at present, is the use of a Linear Feedback Shift Register (LFSR) to generate pseudorandom patterns. An LFSR [8] is a series connection of D-type flip-flops with certain values of flip-flops feeding back to the first flip-flop through an EXOR (Exclusive OR) gate. The output of each flip-flop generates pseudorandom patterns. The structure of an LFSR is usually derived from a primitive polynomial since it generates 2^n-1 distinct patterns for an n-bit LFSR, before repeating a previously generated pattern.

A number of different on-chip response compaction techniques have been studied and applied. The important ones are parity techniques, counting techniques, and signature analysis. Among these, signature analysis has proven to be the most effective. The structure of a signature analysis register (SAR) [9,10] is similar to the structure of an LFSR, except that the SAR has an input connected to the output signal of the circuit under test whose responses are to be compacted. This input is fed to the first flip-flop in the SAR through an exclusive or gate. A MISR (Multiple Input Signature Register) is an SAR in which each flip-flop has an input connected through an EXOR gate. It can be used for response compaction in circuits with more than one output.

A number of techniques have been developed in which the process of test generation and response compaction is achieved by the same module or entity. One such technique, called Built-In Logic Block Observer (BILBO) was proposed by Konemann, et al. [3,4]. In this technique, pattern generation and response compaction functions are provided by a multifunction register called BILBO. A similar technique was described by Bardell and McAnney [5], called Simultaneous Self-Test (SST). Circular BIST, which is used in this thesis, is another such method of integrating test pattern generation and test response compaction. A detailed discussion on the circular BIST technique is given in the following section. Some other BIST techniques can be found in references [6,11,12]. An overview of various BIST techniques and

structures can be found in references [13,14].

2.2 Circular BIST

2.2.1 Literature Survey

The circular BIST technique was first suggested by Krasniewski and Pilarski [7], although the basis of this technique has been known for a number of years, going back to BILBO. Their design applies to circuits which consist of combinational logic blocks and registers. A set of registers are selected and modified into so called self-test path registers. All modified registers are put into a scan chain, and the output of the last register is fed back into the first register to form a circular chain. An example of circular BIST design and a self-test path register presented in [7] is shown in Figure 1. The self-test path register behaves like an LFSR during test mode and is responsible for generating test patterns and compacting responses. If a global reset signal is present in the original circuit, there are no restrictions on the selection criteria used to choose the registers to be modified. If a global reset signal is not present, the circular path must include, as a minimum, all primary input and primary output registers, in addition to a set of internal registers such that each closed signal path in the circuit includes at least one modified register. This requirement is needed to ensure that the whole circuit can be initialized.

Extensive simulations were performed by Krasniewski and Pilarski [7] to determine the effectiveness of circular BIST. Most of these simulations were performed to determine the efficiency of test pattern generation, and more specifically, to find the state coverage produced by the scan chain. State coverage is a measure of the number of different patterns produced by

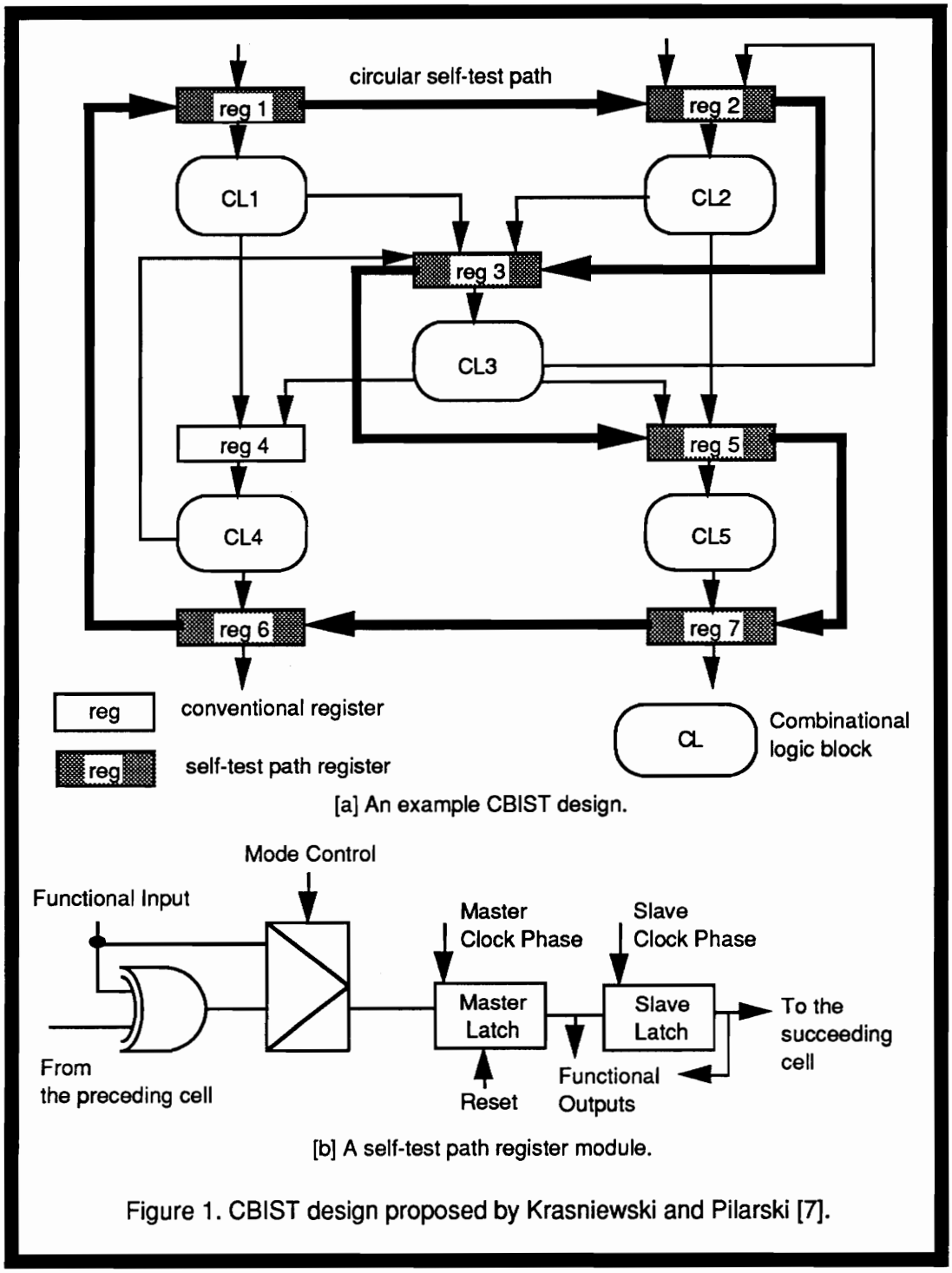


Figure 1. CBIST design proposed by Krasniewski and Pilarski [7].

the chain. A 10-bit chain producing 512 distinct patterns out of 1024 possible patterns is said to have a state coverage of 50%. A summary of results reported in [7] is given below.

- The effect of the initial state of the chain on the state coverage is negligible.
- State coverage increases as the length of the chain increases.
- When the length of a test session is a few times (4 to 8) that of its corresponding exhaustive test session, the state coverage is close to 100%.

The conclusion reached is that the test pattern generation achieved by the circular path is almost independent of the circuit under test and is close to being pseudorandom. It should be noted here that simulations were performed on models of circular path (not actual circuits made up of gates and flip-flops) and not on actual circuits.

A different CBIST design was proposed by Stroud which is applicable to general sequential circuits with flip-flops [15,16]. A set of flip-flops (instead of registers as in the original CBIST design proposed by Krasniewski and Pilarski) is selected and modified into multifunction modules called BIST flip-flops. The BIST flip-flops are connected in a scan chain to form a circular path. In order to enhance system level diagnosis, an SAR is included as a part of the scan chain.

Pradhan, et al. [17] modified Stroud's design in two aspects. First, scan cells were added at the primary inputs and the primary outputs of the circuit under test to implement the so called boundary scan structure. Boundary scan is useful for board level and system level testing. It also enables the application of supplemental deterministic testing to achieve higher fault coverage than that achieved by only BIST testing. Secondly, the SAR was eliminated from the scan chain.

The structure of a BIST flip-flop is shown in Figure 2. Figure 3 shows a general sequential circuit which is partitioned into flip-flops and combinational logic blocks. The architecture of the CBIST design proposed by Pradhan, et. al. is shown in Figure 4. This CBIST

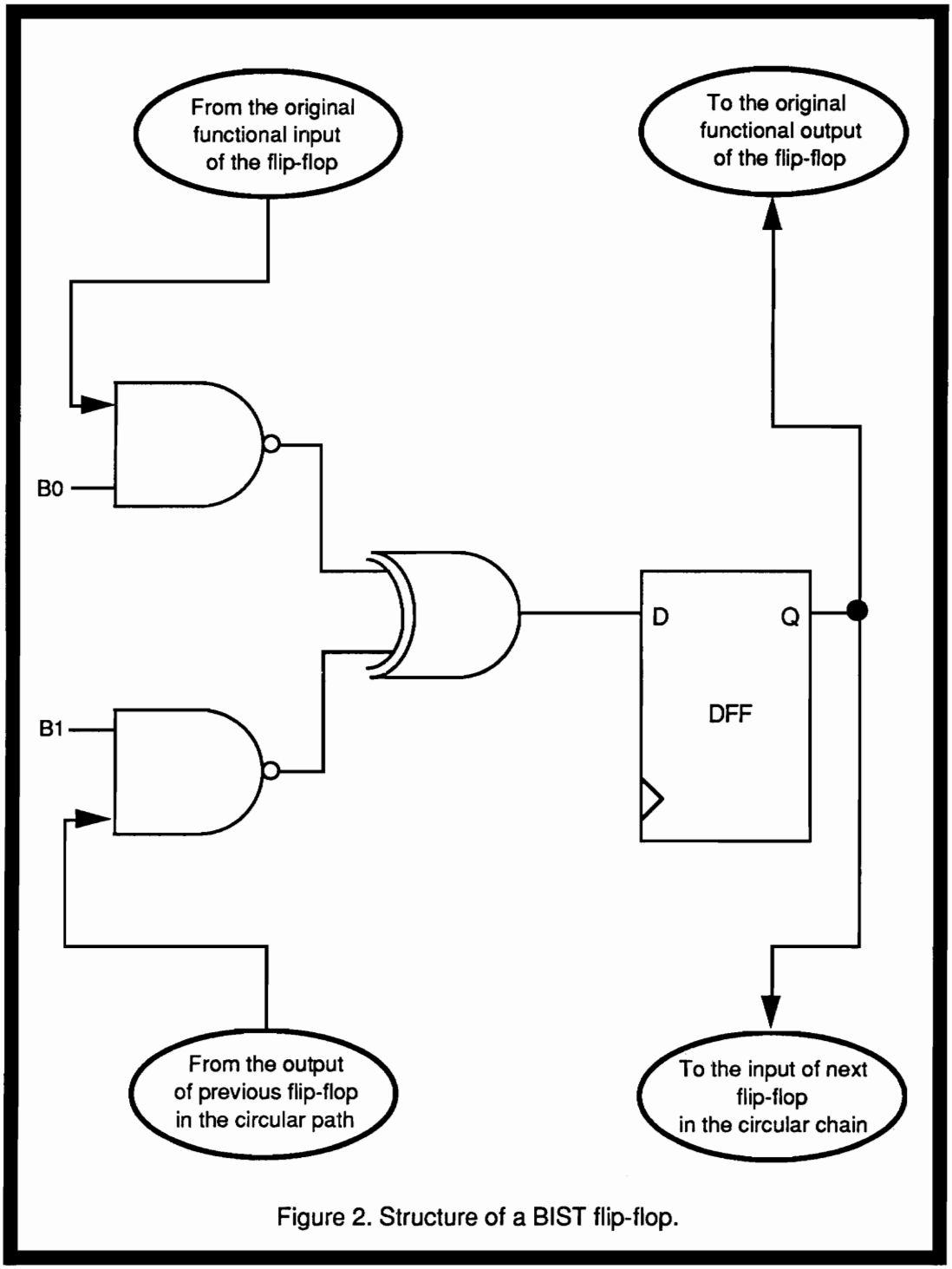


Figure 2. Structure of a BIST flip-flop.

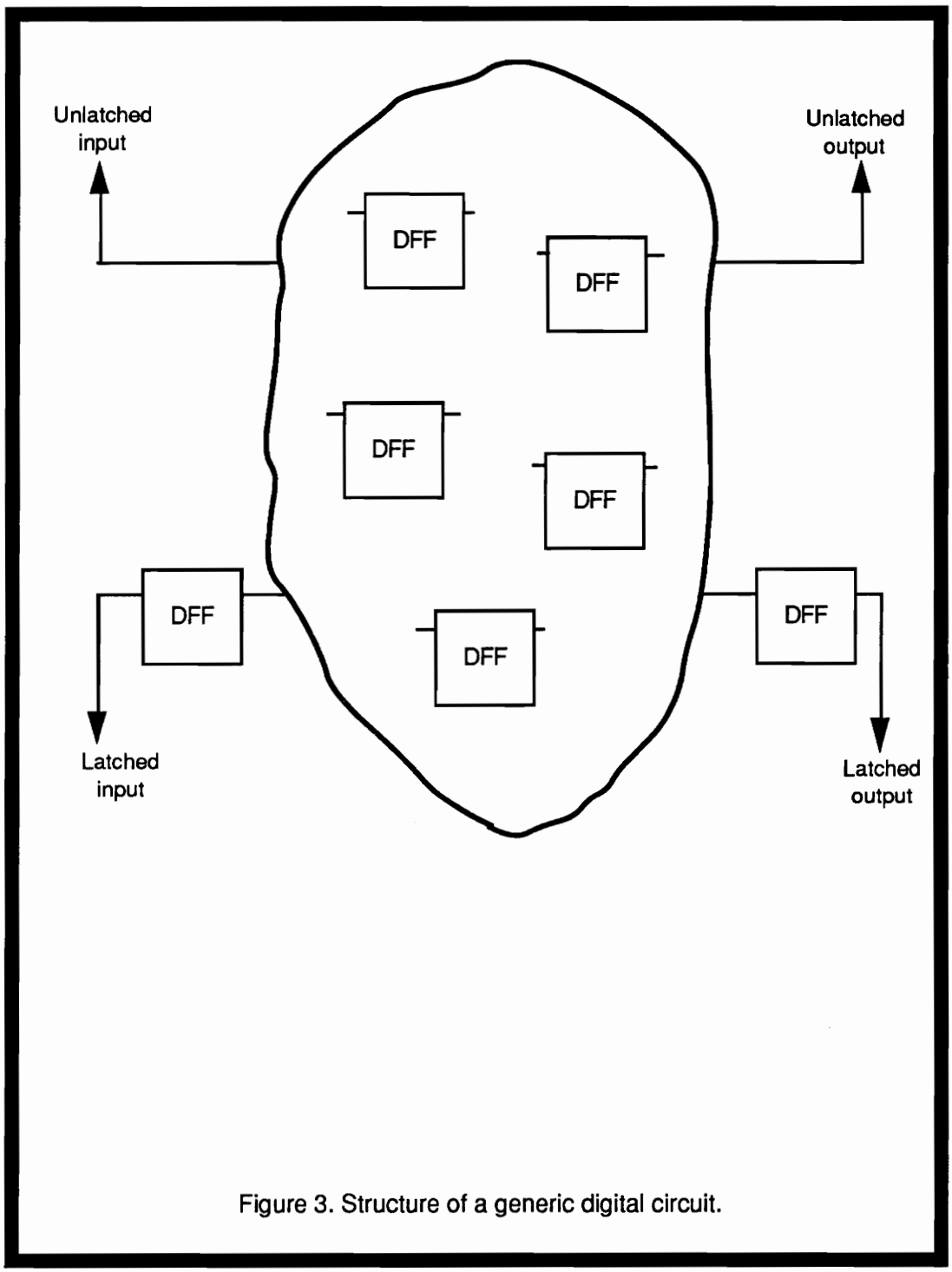


Figure 3. Structure of a generic digital circuit.

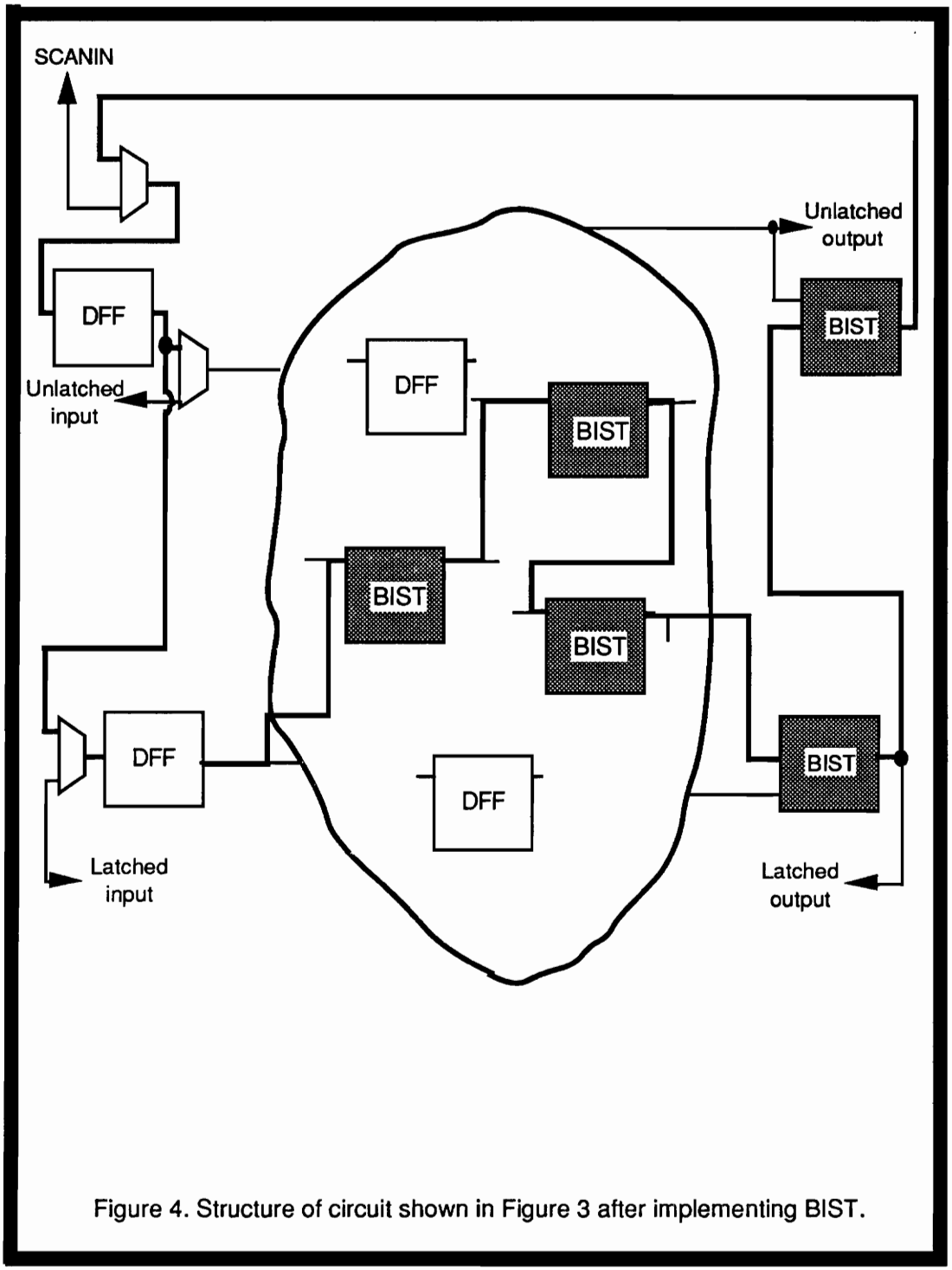


Figure 4. Structure of circuit shown in Figure 3 after implementing BIST.

design is implemented on the circuit of Figure 3. The scan cells at the inputs form a shift register during testing. All flip-flops at primary outputs are BIST flip-flops which compact test response. This CBIST design is used in this thesis for studying aliasing probabilities. A detailed explanation of this design is presented in the following section.

Stroud [15,16] and Pradhan, et al. [17] conducted investigations to find an efficient method of selecting flip-flops to be modified and included in the circular path. The primary goal of their investigations was to select the least number of flip-flops while maintaining high fault coverage. As a result of their efforts, a tool called CKT (Circuit Know Thyself) [17] was developed. CKT automates the flip-flop selection process based on the criteria listed below.

- The number of inputs to the logic cone feeding the flip-flop.
- Minimization of reconvergent fanout.
- Observability.
- Controllability.

Evaluation of the above four criteria revealed that selections based on reconvergent fanout produced the highest fault coverage [17]. A strong correlation between the sequential depth of the circuit and fault coverage was also observed.

Analytical results on circular BIST are hard to obtain, since the patterns generated depend on the functionality of the circuit under test. A probabilistic analysis in [7], performed under the simplistic assumption that the stream of bits applied to different inputs of the path are independent of each other, has proved that some time after initialization (settling time) the probability of a one at any bit is equal to the probability of a zero. No experimental results on the aliasing probability of the circular BIST technique have yet been reported. Hence the focus of this thesis.

2.2.2 Circular BIST Test Structure

As mentioned earlier, circular BIST is implemented by modifying existing flip-flops in the circuit and connecting these modified flip-flops (BIST flip-flops) in a circular path. The circular path is capable of generating test patterns and simultaneously compacting response to these patterns. In fact, the result of compaction is used as the test pattern for the next clock period.

There are four modes of operation for the BIST flip-flop shown in Figure 2. They are Reset, System, Scan, and BIST. During reset mode, the BIST flip-flop is reset to zero, during system mode it behaves as a normal D-type flip-flop, and during scan mode it behaves as a shift register. Testing is conducted in the BIST mode. Two control pins, B0 and B1, are used to choose the mode of operation of the BIST flip-flops. The circuit operation mode depends on the BIST flip-flop operation mode plus the values of SCAN and SCANIN lines. The SCAN and SCANIN lines are added to the original circuit to enable scanning in test vectors into the chain and scanning out signature from the chain. The different modes of operation of the circular BIST design are shown in Table 1.

The overall test circuit structure can be explained with the help of Figures 3 and 4. Figure 3 depicts a general sequential circuit. Figure 4 shows the structure of the same circuit after implementing circular BIST on it. The process of test circuit synthesis can be explained by considering the following cases.

- Circuit to be added at existing internal flip-flops.
- Circuit to be added at latched or unlatched primary outputs.
- Circuit to be added at latched or unlatched primary inputs.
- Circuit to be added for scanning in patterns serially in scan mode.

Table 1. Modes of Operation of a Circuit with BIST Implemented on it

B0	B1	SCAN	SCANIN	MODE OF OPERATION
0	0	X	X	Reset all BIST flip-flops
0	1	0	X	Scan mode: shift signature out serially
0	1	1	Test Data	Scan mode: Shift in initializing vector or test data serially
1	0	X	X	Normal mode
1	1	0	X	Circular BIST test mode

Internal flip-flops are those which are not directly connected to any primary output or primary input. Some of the internal flip-flops are selected to be modified into BIST flip-flops. A proper criteria for selection of flip-flops is important if an optimum balance between low test circuit overhead and high fault coverage is desired. When only a subset of all existing flip-flops are selected and modified, the method of testing is called partial scan. If all existing flip-flops are selected and modified, the method is called full scan. This is the method used in this thesis. That is, all existing internal flip-flops are selected and modified into BIST flip-flops.

All outputs of a circular BIST circuit have BIST flip-flops connected to them. These BIST flip-flops are essential for compacting test responses. All flip-flops at latched outputs are converted into BIST flip-flops. Two NAND gates and an EXOR (Exclusive OR) gate are added to the flip-flop to make it reconfigurable. In the case of an unlatched output, the whole BIST flip-flop module is added at the output.

The contents of the flip-flops in the scan chain are observed at a latched output under the scan mode. If a latched output is not present in the original circuit, the addition of a 2x1 multiplexer at any convenient output is a possible solution. This is shown in Figure 5. The contents of the flip-flops in the scan chain can now be observed at this output by running the circuit in scan mode for a predetermined number of clock cycles.

A shift register is provided for all inputs of a circular BIST circuit. Several reason for adding a shift register at all inputs are given below.

- It provides isolation of the circuit under test from the circuit driving its inputs during normal operation.
- Along with the other flip-flops in the circular path, it enables deterministic testing. That is, a test pattern shifted into the scan chain can be applied to the rest of the circuit and the test response can be latched into the BIST flip-flops.

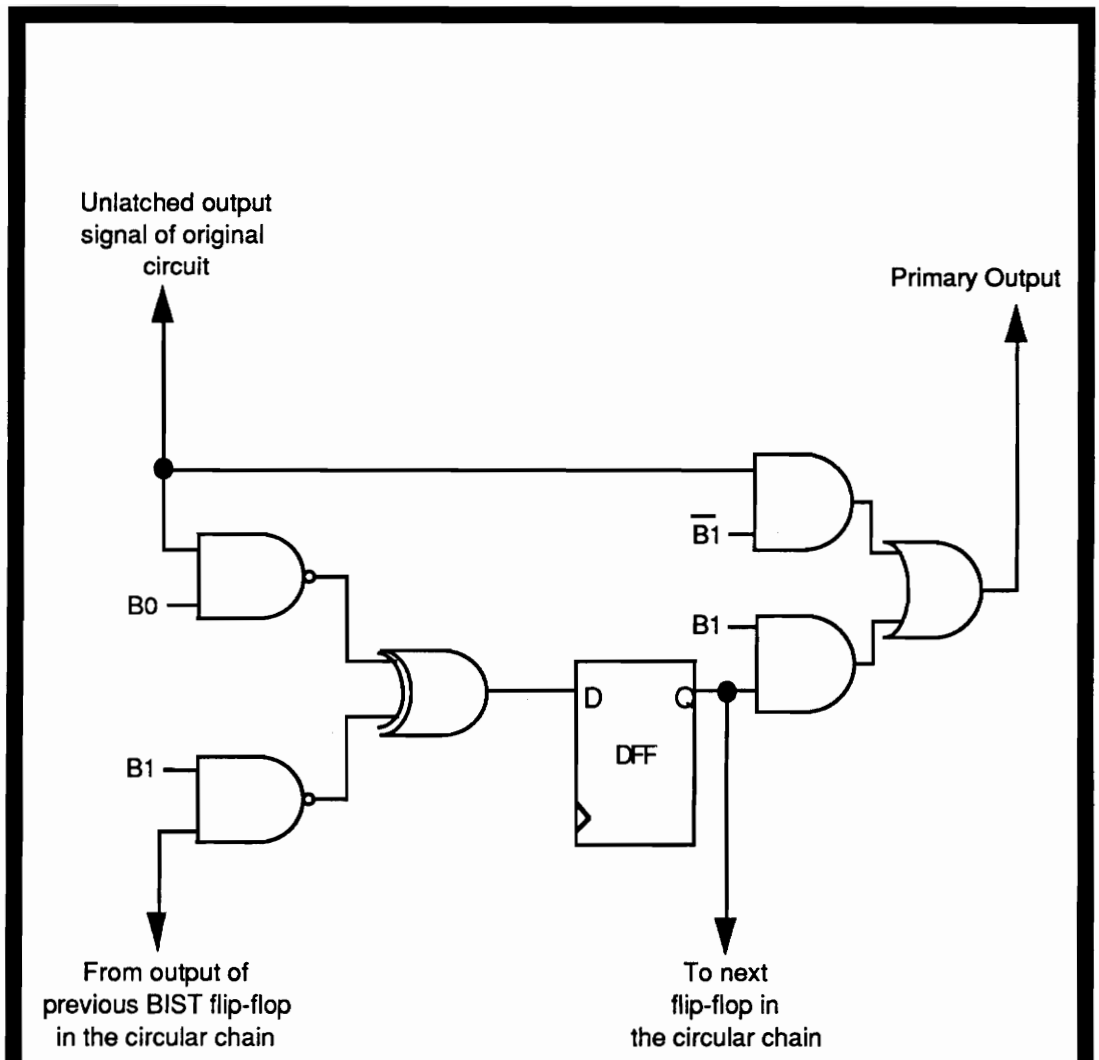


Figure 5. Structure of circuit added at unlatched output to enable observation of signature through that output.

For a latched input, a 2x1 multiplexer is added at the input of the flip-flop. During reset and normal mode, the signal from the primary input is routed to the flip-flop, whereas during scan and BIST mode, the signal from the previous flip-flop in the circular chain is routed to the flip-flop. The mode control signal B1 is used as the select signal for the multiplexer.

For an unlatched input, a 2x1 multiplexer along with a flip-flop needs to be added at that input. The two inputs to the multiplexer are the primary input and the output of the previous flip-flop in the circular path. B1 is used again as the select signal for the multiplexer. The modification of latched and unlatched inputs is shown in Figure 4.

A note about reseting the circuit flip-flops is in order here. All BIST flip-flops can be reset to zero by applying a single clock pulse when the circuit is in reset mode (B0 = 0, B1 = 0). The flip-flops at the primary inputs are not converted to BIST flip-flops and hence cannot be reset in this way. Unless a global reset signal is present in the original circuit, initializing values should be shifted into these flip-flops through the SCANIN pin.

Figure 6 shows the original gate and register level circuit of the ISCAS89 benchmark circuit S27. The ISCAS89 benchmark circuits [18] are the circuits on which simulation experiments were performed. Figure 7 shows circuit S27 after it was modified by the circular BIST test synthesis program CBIST developed as a part of this research. The flip-flops with shaded interior are BIST flip-flops and the trapezoidal forms represent 2x1 multiplexers. The mode control signals B0 and B1 are not explicitly shown in the circuit.

2.3 Explanation of Aliasing Errors

Aliasing, also known as fault masking, can be defined as the loss of information due to data compaction. In the specific context of fault simulation or testing, fault masking is said to occur when, at the time of application of a test pattern, a fault is detected, but this information

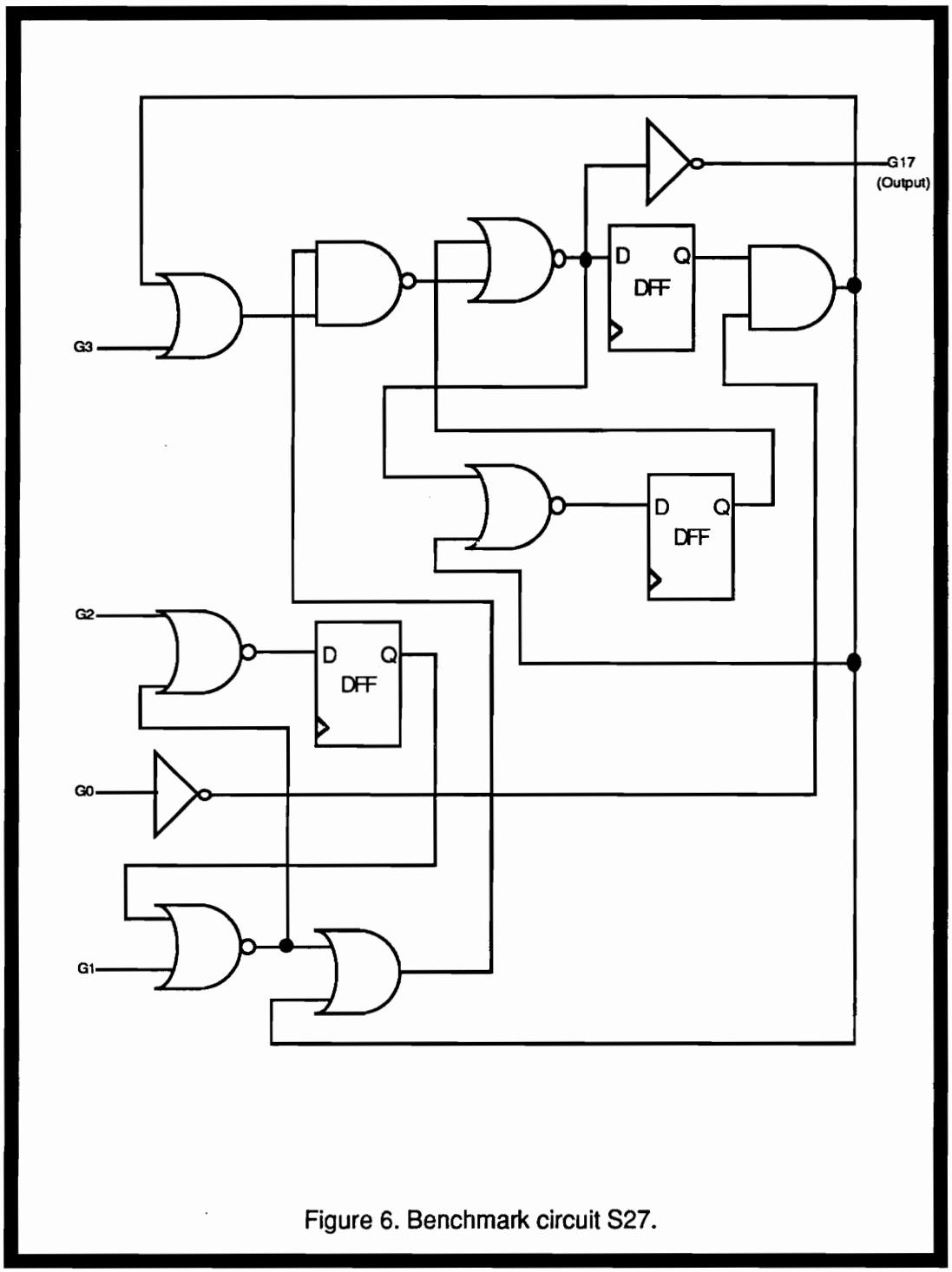


Figure 6. Benchmark circuit S27.

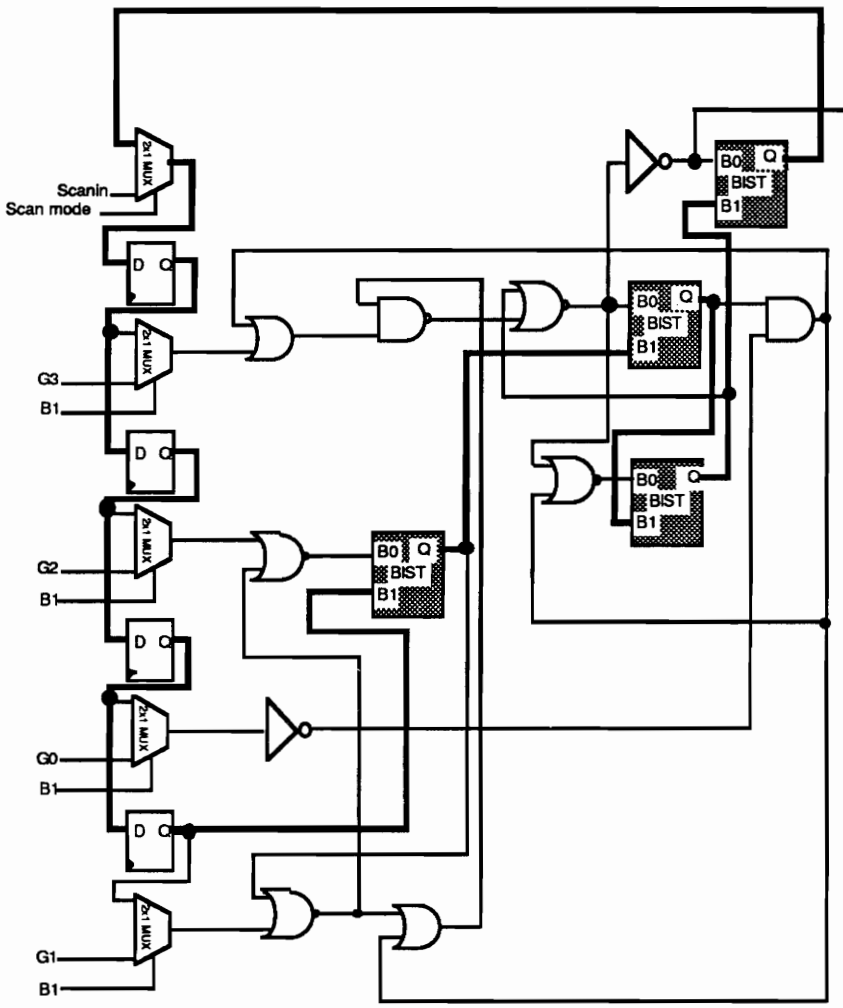


Figure 7. Circuit S27 with BIST implemented on it.

is later lost due to response compaction. In the case of circular BIST, the contents of the flip-flops in the circular path are identical to their corresponding fault-free values at the end of the test, but they were different at least once during the test.

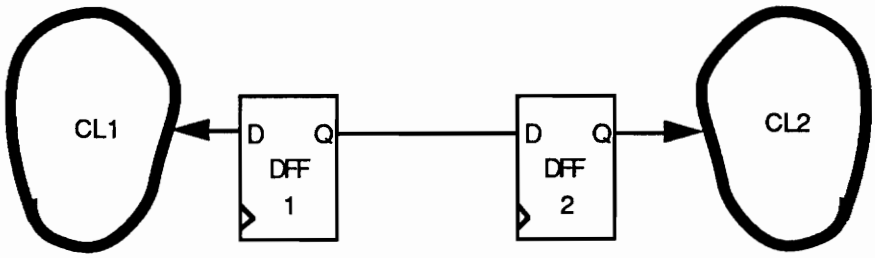
2.3.1 Aliasing Errors Due to Register Adjacency

An important source of fault masking in circular BIST is register adjacency. Register adjacency occurs when two flip-flops connected to each other directly in the original circuit are connected to each other in the same order when they are converted to BIST flip-flops in the modified circuit. Figure 8(a) shows the connection of two flip-flops in the original circuit. Figure 8(b) shows the two flip-flops in Figure 8(a) converted to BIST flip-flops, and connected to each other in the same order as in the original circuit. In the BIST mode, the inputs to the EXOR gate are at the same level. This means that the flip-flop input will always remain at zero, no matter what the values of its input are. This leads to fault masking. There can be other subtle forms of fault masking besides the simple illustration shown here. No attempt to detect or reduce register adjacency is made by our circular BIST test synthesis tool.

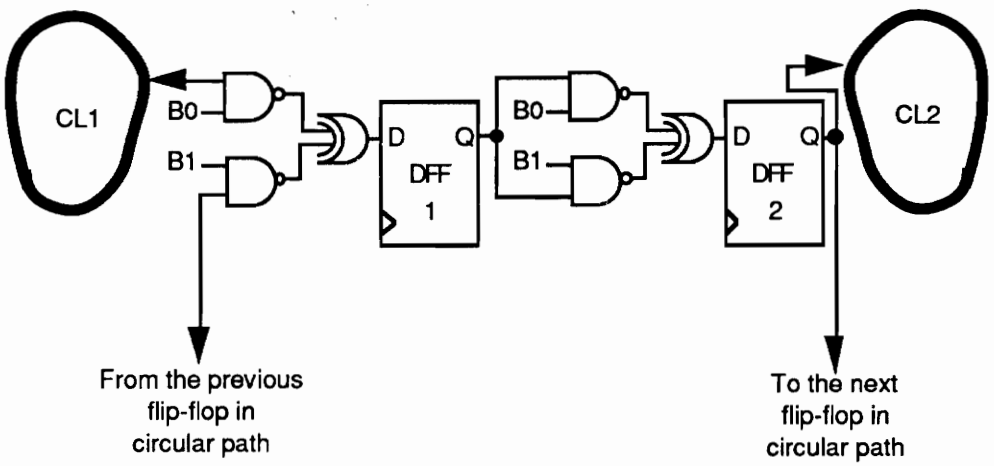
2.3.2 Experimental Measurement of Aliasing Probability

Before we discuss the computation of aliasing probability, two terms, "true fault detect" and "optimistic fault detect" are explained first.

True fault coverage is the fault coverage obtained when aliasing errors are considered in calculating the coverage. This would be the coverage realizable in an actual implementation of the design. Each fault detected in this manner is called a true fault detect. The method of



[a] Original Circuit.



[b] Modified Circuit.

Figure 8. Example of register adjacency.

obtaining the number of true fault detects and the true fault coverage for a circuit is described below.

1 Inject a fault into the circuit.

2 Apply n clocks (i.e. n test patterns).

3 Observe the signature in the circular chain. If it is different from the corresponding fault-free signature, the fault is detected.

4 Repeat from step 1 if more faults are to be simulated.

5 True fault coverage = number of true fault detects / total number of collapsed faults.

The optimistic fault coverage is the fault coverage obtained when aliasing errors are ignored. If a fault is detected during the course of the test session it is assumed to be detectable at the end of the test session. Each fault detected in this manner is called an optimistic fault detect. Optimistic coverage is always higher than true fault coverage. The method of obtaining the number of optimistic fault detects and the optimistic fault coverage for a circuit is described below.

1 Inject a fault into the circuit.

2 Apply a single clock pulse (one test pattern).

3 If the effect of the fault reaches any of the flip-flops in the circular chain, the fault is detected.

4 Mark the fault as detected.

5 If more patterns are to be applied, go to step 2.

6 Repeat from step 1 if more faults are to be simulated.

7 Optimistic fault coverage = number of optimistic faults detects / total number of faults.

After the number of true and optimistic fault detects for a circuit are known, the aliasing probability for that circuit can be found from the expression shown below.

$$AP = (OFD - TFD) / OFD$$

where AP is the Aliasing Probability, OFD is the number of Optimistic Fault Detects, and TFD is the number of True Fault Detects. The fault coverage most often used in practice is the optimistic fault coverage since it takes much less time to compute than the actual fault coverage. Once the aliasing probability for a circular BIST circuit is known, the true fault coverage can be obtained from the optimistic fault coverage using the following expression.

$$TFC = OFC (1 - AP)$$

where TFC is the True Fault Coverage and OFC is the Optimistic Fault Coverage.

2.4 Motivation

Too high an aliasing probability for a compression technique can render that method of testing ineffective. Low aliasing probability should thus be one of the factors considered when evaluating the effectiveness of any testing technique utilizing response compaction. Many methods for estimating the aliasing probability of various response compaction techniques have been published in the open literature. Most of these methods use some type of assumption to derive the estimate for that method. Estimation of aliasing probability in the circular BIST method is especially hard to obtain since the test patterns generated depend on the functionality of the circuit under test. No experimental results on aliasing probability in this method of testing have yet been published. This was the main motivation for the work done in this thesis.

Obtaining experimental results on aliasing probability requires performing computationally intensive fault simulation. Existing fault simulators were found to be either too slow or too cumbersome to use for our purpose. This led to the development of a new fault simulator which is especially suitable for obtaining results on aliasing errors. The cost of developing a new fault simulator is justified by the fact that if the aliasing probability for a

particular testing technique is known, the true fault coverage can be found from the value of optimistic fault coverage, which is much easier and faster to obtain.

Many testing techniques use SARs for detecting faults. Results on aliasing errors in SARs are included for the purpose of comparison with the circular BIST method.

3.0 Simulation Tools

In order to measure aliasing probability of the CBIST design, a number of tools have been developed for this research. A brief discussion describing each of them form the contents of this chapter. The relevant details are given for all the tools. Examples of formats of input and output files are given where relevant. The overall organization of the tools is shown in Figure 9. All simulations were performed on the ISCAS89 benchmark circuits. CBIST, the tool that implements circular BIST hardware on ISCAS89 sequential circuits is described in Section 3.1. Section 3.2 deals with ISCTOHILO (ISCAS89 To HILO), a tool developed to convert circuit files from the ISCAS89 format to the HILO format. Another netlist format conversion tool, ISCTOPSIM (ISCAS89 To PSIM), which converts file formats from ISCAS89 to a format readable by PSIM, is described in Section 3.3. The tool used to create a collapsed fault file for a circuit is separate from the simulator itself. This tool, FAULTDIR, is described in Section 3.4. Finally, PSIM, the parallel fault simulator developed for this thesis, is described in Section 3.5. All programs mentioned here are written in the C language and run under the Unix operating system.

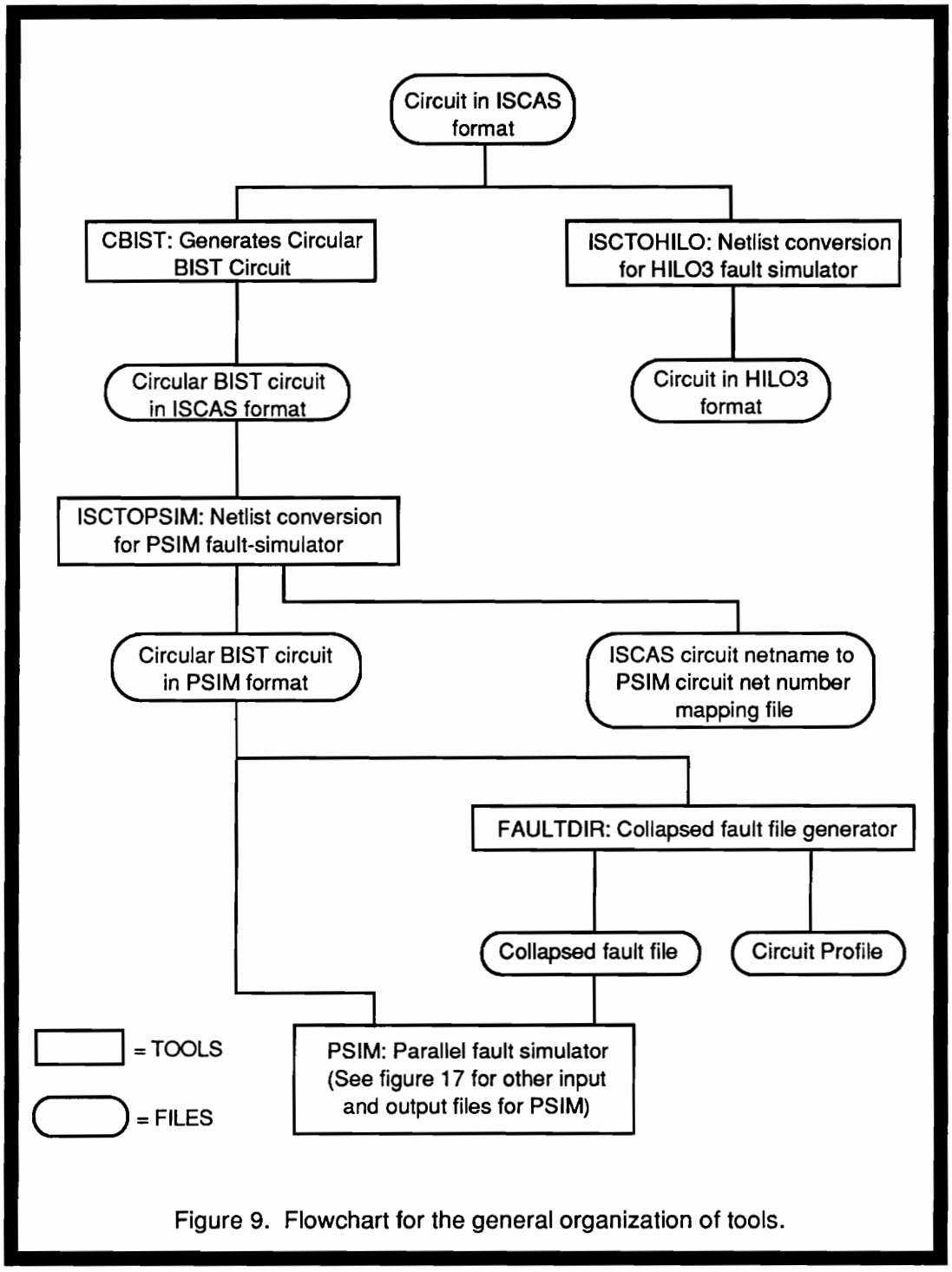


Figure 9. Flowchart for the general organization of tools.

3.1 CBIST

CBIST is a program that implements circular built-in self-test on general sequential circuits. It reads a circuit in the ISCAS format and creates a CBIST circuit in the same format.

The structure of test circuit added by CBIST is described in detail in Section 2.2.2. The names of the test control pins added is not, however, identical to those described previously. The circuit mode control pins B0 and B1 are respectively B$0 and B$1 and the lines SCAN and SCANIN are S$ and S$IN. This is to avoid the possibility that the signal names B0, B1, SCAN, and SCANIN might already be present in the original circuit file. Signal names with the dollar sign in them are less likely to occur in a circuit description file. For the same reason, all other signals added as a part of the circular BIST test circuit overhead have names with dollar sign inserted somewhere in them.

A flowchart describing the overall structure of the program CBIST is shown in Figure 10. The program reads in the circuit information and creates different arrays for its input, output, and flip-flop elements. As was stated in Section 2.2.2, CBIST handles the task of test circuit synthesis separately for the cases of internal flip-flops, unlatched inputs, latched inputs, unlatched outputs, and latched outputs. A separate array for each of the above cases is formed. The internal flip-flops are then modified to BIST flip-flops and connected together in a chain. Next, existing flip-flops at primary outputs are modified to BIST flip-flops, and BIST flip-flop modules are added at unlatched outputs. The BIST flip-flops at the outputs are then connected together to form a chain and one end of this chain is connected to the chain of internal flip-flops. Similarly, test circuit is added at primary inputs. Ultimately, the two ends of the chain are connected together so that a circular path is formed. Test circuit to add the scan capability is inserted somewhere in this chain.

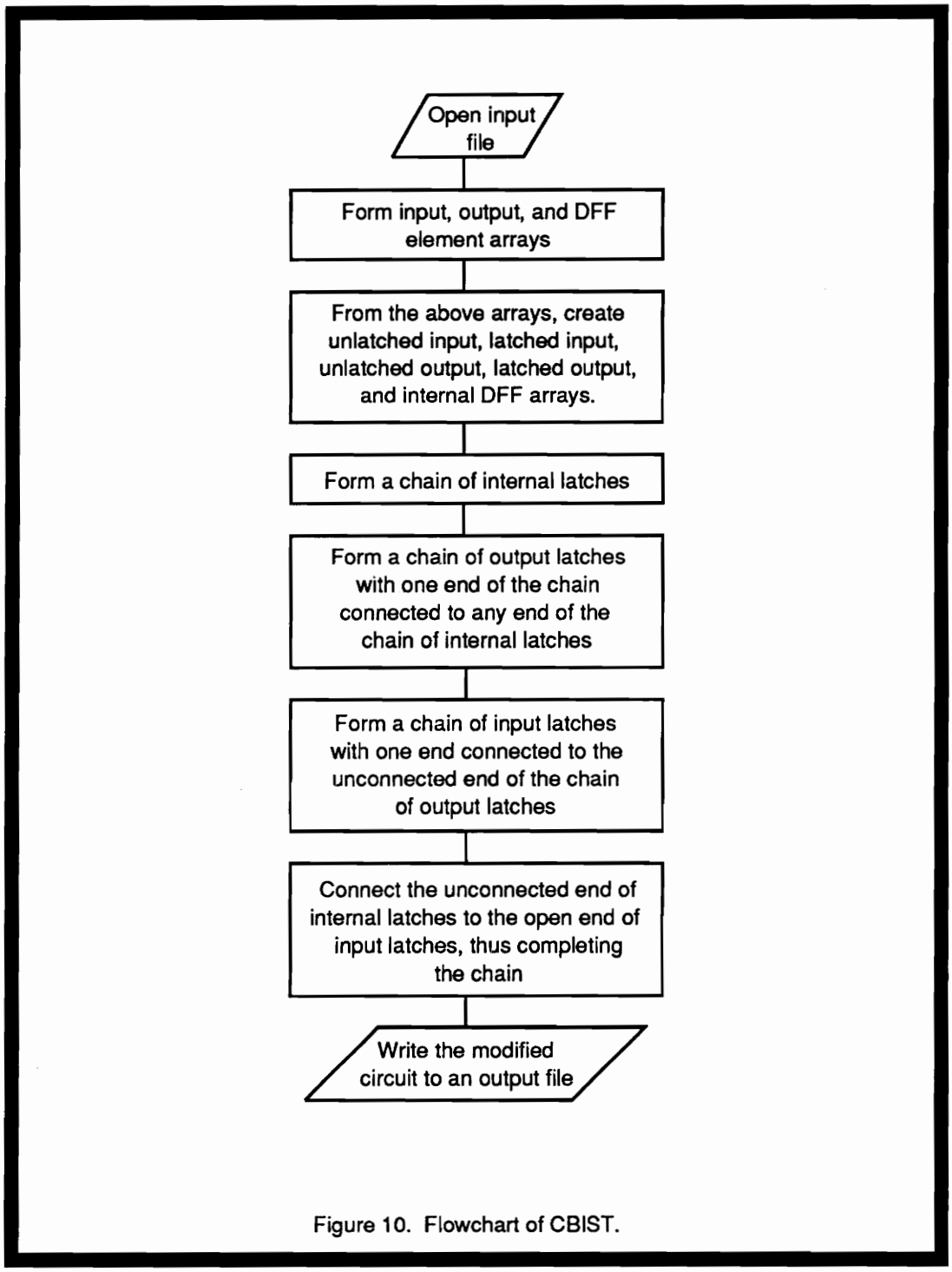


Figure 10. Flowchart of CBIST.

There are some restrictions on the format of the input file aside from following the format of ISCAS89 bench circuits. They are listed below.

- All comments (lines beginning with a '#') have to be at the beginning of a circuit file.
- The type of elements have to be described in the order shown below.

  1. INPUT 2. OUTPUT 3. DFF 4. All other elements.
- Elements of the same type have to be described in a single block and elements of different types have to be described in different blocks, with each block corresponding to each element and any two blocks being separated by at least one line.

An example of the input circuit file S27 (in ISCAS89 format) is shown in Figure 11. Figure 12 shows the modified circuit file S27.bist which is the output of program CBIST. Gate level diagrams of circuits S27 and S27.bist are shown in Figure 6 and Figure 7 respectively. The modified circuit in Figure 12 is shown in 2 columns to fit in one page. In a circuit file, the circuit description is written in a single column. Note that in the output file, the elements of a particular type are not necessarily listed in one continuous block. Also note that the comments at the beginning of the modified circuit are copied as they are from the original file and do not reflect the added test hardware. Information about the number of elements and the number of collapsed faults in a modified circuit is provided by the program FAULTDIR.

3.2 ISCTOHILO

ISCTOHILO is a netlist conversion tool which converts a circuit in ISCAS89 format to a circuit format readable by the HILO3 fault simulator. Figure 13 shows the circuit S27 in HILO3 format. The purpose of converting a circuit in ISCAS89 format to the HILO3 format was to use the commercially available simulator to verify tools developed for this thesis.

All gates in the HILO3 file are assumed to have zero gate delays. Exclusive OR gates

```
# s27
# 4 inputs
# 1 outputs
# 3 D-type flipflops
# 2 inverters
# 8 gates (1 ANDs + 1 NANDs + 2 ORs + 4 NORs)

INPUT(G0)
INPUT(G1)
INPUT(G2)
INPUT(G3)

OUTPUT(G17)

G5 = DFF(G10)
G6 = DFF(G11)
G7 = DFF(G13)

G14 = NOT(G0)
G17 = NOT(G11)

G8 = AND(G14, G6)

G15 = OR(G12, G8)
G16 = OR(G3, G8)

G9 = NAND(G16, G15)

G10 = NOR(G14, G11)
G11 = NOR(G5, G9)
G12 = NOR(G1, G7)
G13 = NOR(G2, G12)
```

Figure 11. Circuit S27 in ISCAS89 form.

```
# s27
# 4 inputs
# 1 outputs
# 3 D-type flipflops
# 2 inverters
# 8 gates (1 ANDs + 1 NANDs + 2 ORs + 4 NORs)

INPUT(G0)
INPUT(G1)
INPUT(G2)
INPUT(G3)

OUTPUT(G17)

G14 = NOT(O$G0)
G17 = NOT(G11)

G8 = AND (G14, G6)

G15 = OR(G12, G8)
G16 = OR(O$G3, G8)

G9 = NAND(G16, G15)

G10 = NOR(G14, G11)
G11 = NOR(G5, G9)
G12 = NOR(O$G1, G7)
G13 = NOR(O$G2, G12)

INPUT(B$0)
INPUT(B$1)
INPUT(S$IN)
INPUT(S$)

S$NT = NOT(S$)
B$1INV = NOT(B$1)

G6 = DFF(X$G11)
G7 = DFF(X$G13)
D$G17 = DFF(X$G17)
D$G2 = DFF(O$S$IN)
D$G3 = DFF(D$G2)
D$G0 = DFF(D$G3)
D$G1 = DFF(D$G0)
G5 = DFF(X$G10)

M$SIN = AND(S$, S$IN)
M$D$G17 = AND(S$NT, D$G17)
M$G2 = AND(B$1INV, G2)
M$D$G2 = AND(B$1, D$G2)
M$G3 = AND(B$1INV, G3)
M$D$G3 = AND(B$1, D$G3)
M$G0 = AND(B$1INV, G0)
M$D$G0 = AND(B$1, D$G0)
M$G1 = AND(B$1INV, G1)
M$D$G1 = AND(B$1, D$G1)

O$S$IN = OR(M$SIN, M$D$G17)
O$G2 = OR(M$G2, M$D$G2)
O$G3 = OR(M$G3, M$D$G3)
O$G0 = OR(M$G0, M$D$G0)
O$G1 = OR(M$G1, M$D$G1)

N$G11 = NAND(B$0, G11)
N$G5 = NAND(B$1, G5)
N$G13 = NAND(B$0, G13)
N$G6 = NAND(B$1, G6)
N$G17 = NAND(B$0, G17)
N$G7 = NAND(B$1, G7)
N$G10 = NAND(B$0, G10)
N$D$G1 = NAND(B$1, D$G1)

X$G11 = XOR(N$G11, N$G5)
X$G13 = XOR(N$G13, N$G5)
X$G17 = XOR(N$G17, N$G7)
X$G10 = XOR(N$G10, N$D$G1)
```

Figure 12. S27.BIST in ISCAS89 form.

```
CCT s27.hilo(G0, G1, G2, G3, G17, P$ST, C$LR)

CLOCK1 (100, 25, 25)
 C$1(C$K);

AND
 G$2(G8, G14, G6);
OR
 G$3(G15, G12, G8)
 G$4(G16, G3, G8);

NAND
 G$5(G9, G16, G15);

NOR
 G$6(G10, G14, G11)
 G$7(G11, G5, G9)
 G$8(G12, G1, G7)
 G$9(G13, G2, G12)

NOT
 G$0(G14, G0)
 G$1(G17, G11)

TOY
 R$0(G5, G10, C$K, C$LR, P$ST)
 R$1(G6, G11, C$K, C$LR, P$ST)
 R$2(G7, G13, C$K, C$LR, P$ST);

WIRE G17;

INPUT G0, G1, G2, G3, P$ST, C$LR.
```

Figure 13. Circuit S27 in HILO3 format.

are described as instances of a predefined structural subcircuit called XON. D type flip-flops are declared as instances of a predefined functional subcircuit called TOY. These two subcircuits must be in the HILO3 scratch area before any simulation can be performed. The primary outputs in the original circuit are referred to as WIREs in the HILO3 circuit. Refer to the Appendix and the HILO3 users manual [19] for details.

3.3 ISCTOPSIM

ISCTOPSIM is another netlist format conversion tool which generates an intermediate file to be read by the PSIM fault simulator. The input of PSIM is a circuit in ISCAS format. The main purpose of creating an intermediate file is to speed up PSIM which is computationally intensive.

ISCTOPSIM converts netlist names into netlist numbers so that simulation can progress faster. Since this conversion is done prior to simulation, it also avoids having to go through the same process every time the same circuit is simulated.

The general format of an element specification in the output file created by ISCTOPSIM is shown below.

<input 1> <input 2> ... <input n> <element type> ; <output number> <output name>.

Input 1, input 2 ... input n are the signal numbers of the inputs to the gate, *output number* is the signal number of the output of the gate, and *output name* is the name of the output signal. *Element type* can be INPUT, OUTPUT, NOT, BUF, AND, OR, NAND, NOR, or DFF.

In addition to the output file generated in PSIM format, ISCTOPSIM creates another file which maps signal numbers to their corresponding signal names. Although this information is available in the circuit file in PSIM format, the map file is easier to use. A buffer element called "BUF" is added at all output signals to facilitate inserting faults at the

output pins of the circuit. Aside from the "BUF" elements at the outputs, the circuit output file in PSIM format has a one to one correspondence with its corresponding file in ISCAS format.

Figure 14[a] shows the circuit S27 in PSIM format. The map file is shown in Figure 14[b]. In the map file, lines beginning with a "#" indicate an output signal and those beginning with a "*" indicate the output of a flip-flop.

3.4 FAULTDIR

FAULTDIR is a tool to generate an equivalent fault list for a circuit file. Keeping the fault generating function separate from the simulator has the advantage of not having to generate faults every time a circuit is simulated. The fault file generated by FAULTDIR is an ASCII file and can easily be edited to exclude any fault not required to be simulated or add a specific fault to be simulated.

The program is invoked by typing FAULTDIR at the command line. After it is invoked, the user is asked for the name of the circuit file and the name of the fault file which is to be created. The circuit file has to be in the PSIM format. The format of the faults listed in the output file is shown below.

gate/pin/type/1

Gate is the signal number of the output of the gate at which the fault is to be inserted, *pin* is the number of the signal (any of the input signals of the gate or the output signal) at which the fault is to be inserted, and *type* could be either "0" or "1" indicating a stuck-at-0 or a stuck-at-1 fault, respectively. Ignore the *"1"* at the end. The collapsed fault file for ISCAS89 benchmark circuit S27 is shown in Figure 15.

As mentioned earlier, FAULTDIR creates a collapsed fault file, which means that only one fault out of a set of equivalent faults is included in the fault file to be simulated. For

```
# s27
# 4 inputs
# 1 outputs
# 3 D-type flipflops
# 8 gates (1 ANDs + 1 NANDs + 2 ORs + 4 NORs)

INPUT ; 1 G0
INPUT ; 2 G1
INPUT ; 3 G2
INPUT ; 4 G3

1 NOT ; 72 G14
69 NOT ; 65 G17

72 68 AND ; 73 G8

75 73 OR ; 74 G15
4 73 OR ; 76 G16

76 74 NAND ; 77 G9

72 69 NOR ; 67 G10
66 77 NOR ; 69 G11
2 70 NOR ; 75 G12
3 75 NOR ; 71 G13

67 DFF ; 66 G5
69 DFF ; 68 G6
71 DFF ; 70 G7

OUTPUT ; 65 G17

65 BUF ; 79 O_BUF_G17
```

[a]

```
      G0    1
      G1    2
      G2    3
      G3    4
#     G17   65
*     G5    66
      G10   67
*     G6    68
      G11   69
*     G7    70
      G13   71
      G14   72
      G8    73
      G15   74
      G12   75
      G16   76
      G9    77
```

[b]

Figure 14. [a] Circuit S27 in PSIM format.
[b] Map file MP_S27.

```
2/2/0/1
3/3/0/1
4/4/0/1
65/65/0/1
65/65/1/1
66/66/0/1
67/67/0/1
67/67/1/1
67/72/0/1
67/69/0/1
68/68/1/1
68/69/0/1
68/69/1/1
69/69/0/1
69/69/1/1
70/70/0/1
71/71/0/1
71/71/1/1
71/75/0/1
72/72/0/1
72/72/1/1
73/73/0/1
73/73/1/1
73/72/1/1
74/74/1/1
74/75/0/1
74/73/0/1
75/75/0/1
75/75/1/1
76/76/1/1
76/73/0/1
77/77/0/1
```

Figure 15. Collapsed Fault file for circuit S27.

example, for a 2-input NAND gate, the output stuck-at-1 is equivalent to any input stuck-at-0. Therefore, only the output stuck-at-1, output stuck-at-0 and all inputs stuck-at-1 faults are included in the fault file. Besides fault equivalence arising due to gate functionality, fault equivalence due to interconnection of elements is also considered. For example, if the output of an AND gate is connected to an input of an OR gate, the faults at the output of the AND gate do not have to be considered if they are equivalent to the faults at the input of the OR gate, which have already been considered. The collapsed faults to be included in the fault file are summarized below.

- AND/NAND: Stuck-at-1 faults on all input signals of the gate.
- OR/NOR: Stuck-at-0 faults on all input signals of the gate.
- XOR/D-type flip-flop: Stuck-at-0 and stuck-at-1 faults on input signals.
- Fanout Stems: Stuck-at-0 and stuck-at-1.
- NOT/BUF: None
- OUTPUT: Stuck-at-0 and stuck-at-1.

The buffers (BUF) are not present in the original ISCAS circuits. They are added by the netlist format conversion tool ISCTOPSIM in order to facilitate insertion of faults at primary outputs.

3.5 PSIM

PSIM is an interactive two level event driven parallel-fault simulator developed to measure aliasing probability of the circular BIST design described in the previous chapter. Most of the existing fault simulators are not suitable for this purpose. They are versatile, in that they can handle multiple logic values and gate delays, but are too slow for measuring aliasing probability, which requires repeated brute force simulation of the circuit for each fault or a set of faults.

3.5.1 Implementation Details

PSIM takes advantage of the inherent word parallelism of computers [20] and simulates either 15 or 31 faults (depending on the word size of the computer) in a single pass. This means that if there are more than 15 or 31 faults in a fault file, the simulator has to go through more than one pass to simulate all the faults present in the fault file. For example, if a circuit has 100 faults, the simulator will have to go through four passes to simulate all 100 faults, assuming it is executed on a 32 bit machine.

The fault model used in PSIM is the stuck-at x (stuck-at 0 and stuck-at 1) model. A single fault is injected for every faulty machine simulated. Faults are not dropped in PSIM since the signature has to be observed at the end of the test session in the presence of the faults. Simulation for optimistic fault coverage and true fault coverage proceeds simultaneously since fault-free and faulty machines are simulated concurrently. The fault detection mechanism used during simulation depends on the option PSIM is run under. If the normal fault simulation option is chosen, primary output values are observed. If the BIST option is chosen, the flip-flops in the circular path are observed for detection of faults.

At the beginning of simulation, all flip-flops and primary inputs are assigned the logic value zero unless a different initial value is assigned by the user. Initialization is carried out by turning off the selective trace mechanism. After initialization of the circuit, simulation is continued for the specified number of clock periods, and the final signature is observed and compared with the fault-free signature. The process of initialization and simulation is repeated for each additional set of faults.

3.5.2 Functions Available in PSIM

The primary function of PSIM is the detection of aliasing errors in the circular BIST design technique. To find the aliasing probability, we need to know the optimistic fault coverage and the true fault coverage. As explained in Chapter 2, optimistic fault coverage is obtained by ignoring aliasing errors and the true fault coverage takes into account any aliasing errors that might affect the final signature. PSIM reports both these values, enabling the calculation of aliasing probability.

Aside from the main function mentioned above, a few other functions are available in PSIM. They are listed below.

- Given a set of input test patterns, PSIM can perform NORMAL fault simulation. NORMAL fault simulation means that instead of checking the signature of the circuit in the circular path registers to determine whether a fault is detected or not, the primary outputs of the circuit are checked. It also generates a file which reports the number of faults detected by each test pattern.
- A TRACE file which lists the signal values for a user defined set of nodes can be obtained for fault-free simulation of a circuit. Generating a TRACE file for fault simulation makes sense only if the number of faults to be simulated is fewer than 31. If this is not so, the TRACE file would be overwritten during every additional pass the simulator goes through.
- Aside from the NORMAL and BIST methods of testing, PSIM has three built-in Signature Analysis Registers (SARs) of sizes 8, 16, and 32-bits, which are used to compact test responses in conjunction with the circular BIST technique. The output of one flip-flop in the scan chain is tapped to drive the SARs as illustrated in Figure 16. The output which drives the SARs is chosen arbitrarily. The result of this technique is that the compacted response

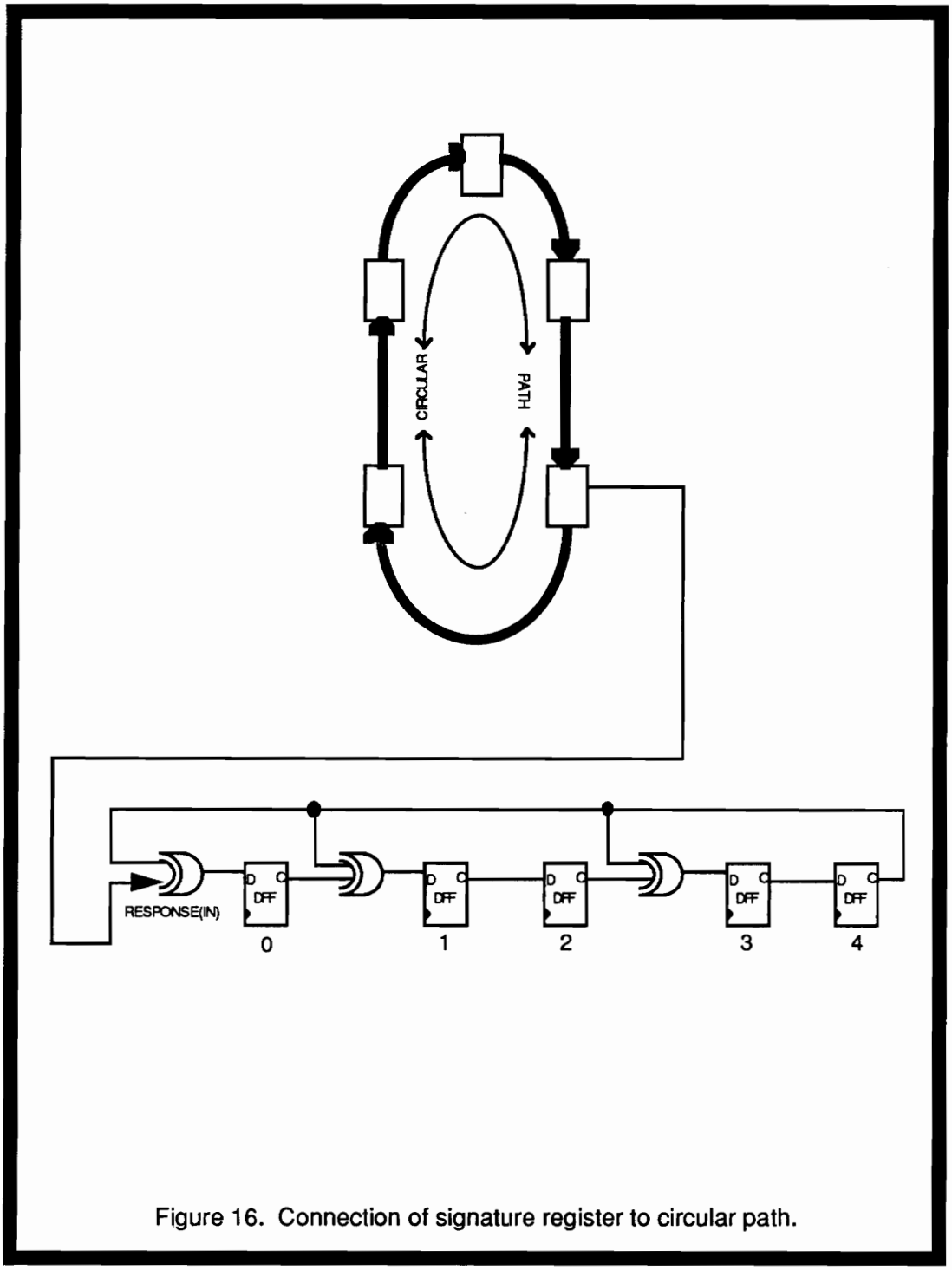


Figure 16. Connection of signature register to circular path.

of the circuit in the circular path flip-flops undergoes further compaction in the SARs. Only the signature in the SAR is checked when the SAR fault detection option is chosen. Results are reported in separate files (to be explained in the following section). The primitive polynomials of the three SARs are shown below.

- 8-bit: $1 + x^2 + x^3 + x^7 + x^8$
- 16-bit: $1 + x^{11} + x^{13} + x^{14} + x^{16}$
- 32-bit: $1 + x^4 + x^5 + x^{31} + x^{32}$

The SARs are can be either chosen individually or all at the same time.

3.5.3 Capabilities of PSIM

Currently PSIM can handle upto 30000 gates and 9 fan-ins per gate but this limit can be extended easily. The following elements are defined in the simulator.

- AND (can be any size from 2-input to 9-input)
- OR (can be any size from 2-input to 9-input)
- NAND (can be any size from 2-input to 9-input)
- NOR (can be any size from 2-input to 9-input)
- XOR (can be any size from 2-input to 9-input)
- NOT (Inverter)
- BUF (Buffer)
- DFF (D-type flip-flop)
- INPUT (Primary input)
- OUTPUT (Primary output)

Additional element types can be easily defined by making only a few addition to the PSIM source code.

Six kinds of files are involved when using PSIM, depending on the options chosen. They are listed below.

Input Files:

- Circuit description file
- Fault file
- Input signal file

Output Files:

- TRACE file
- PATTERNS file
- BATOUT files

The circuit description file is a circuit file in PSIM format. It is needed irrespective of the option under which the simulator is run. A fault file (in the format of the output of FAULTDIR) can be any file listing the set of collapsed faults for the circuit being simulated. A fault file is obviously not needed when performing fault-free simulation. The input signal file provides signal values to primary inputs at different clock periods. The format of this file is shown in the Appendix. As mentioned earlier, the TRACE file lists the signal values of user specified nets during the course of simulation. The nodes whose values are to be traced are defined by the user in the input signal file using the TRACE directive. Refer to the Appendix for the format of this declaration. The PATTERNS file is generated when the NORMAL mode of fault detection is used. It lists the number of faults detected by each test pattern. Finally, four different BATOUT files are generated depending on the option chosen at the command line. The BATOUT file is generated irrespective of the option chosen. In addition to this, the -8, -16, and the -32 options generate the BATOUT8, the BATOUT16, and the BATOUT32 files, respectively. The -a option generates all four files. The format of output in these files is explained in the Appendix. Figure 17 shows the options available in PSIM and the input and

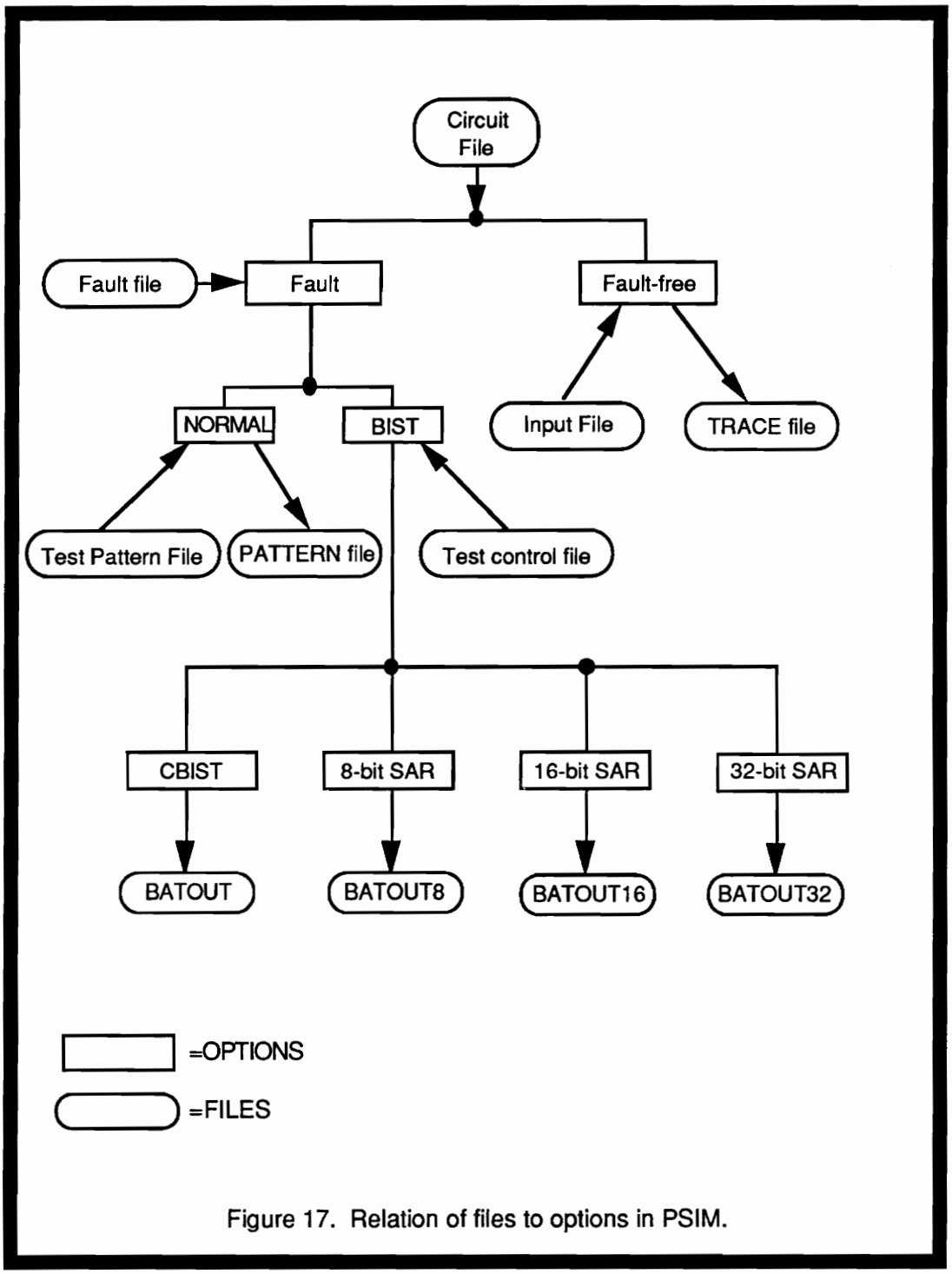


Figure 17. Relation of files to options in PSIM.

output files associated with each option.

A broad flowchart of the structure of PSIM is shown in Figure 18 which is divided into two pages. The first page shows the user interaction part of the simulator. The user has two options for entering the clock values at which results will be reported in the BATOUT files. Manual entry of clock periods makes sense when the aliasing errors are to be found at only a few specific clock periods. If a statistical analysis of aliasing errors is to be performed, reporting aliasing errors at specific clock intervals makes sense. The heart of the simulation algorithm is shown on the second part of the flowchart. Note that the circuit needs to be initialized at the beginning of every pass of simulation because insertion of different faults in the circuit may produce different initial net values.

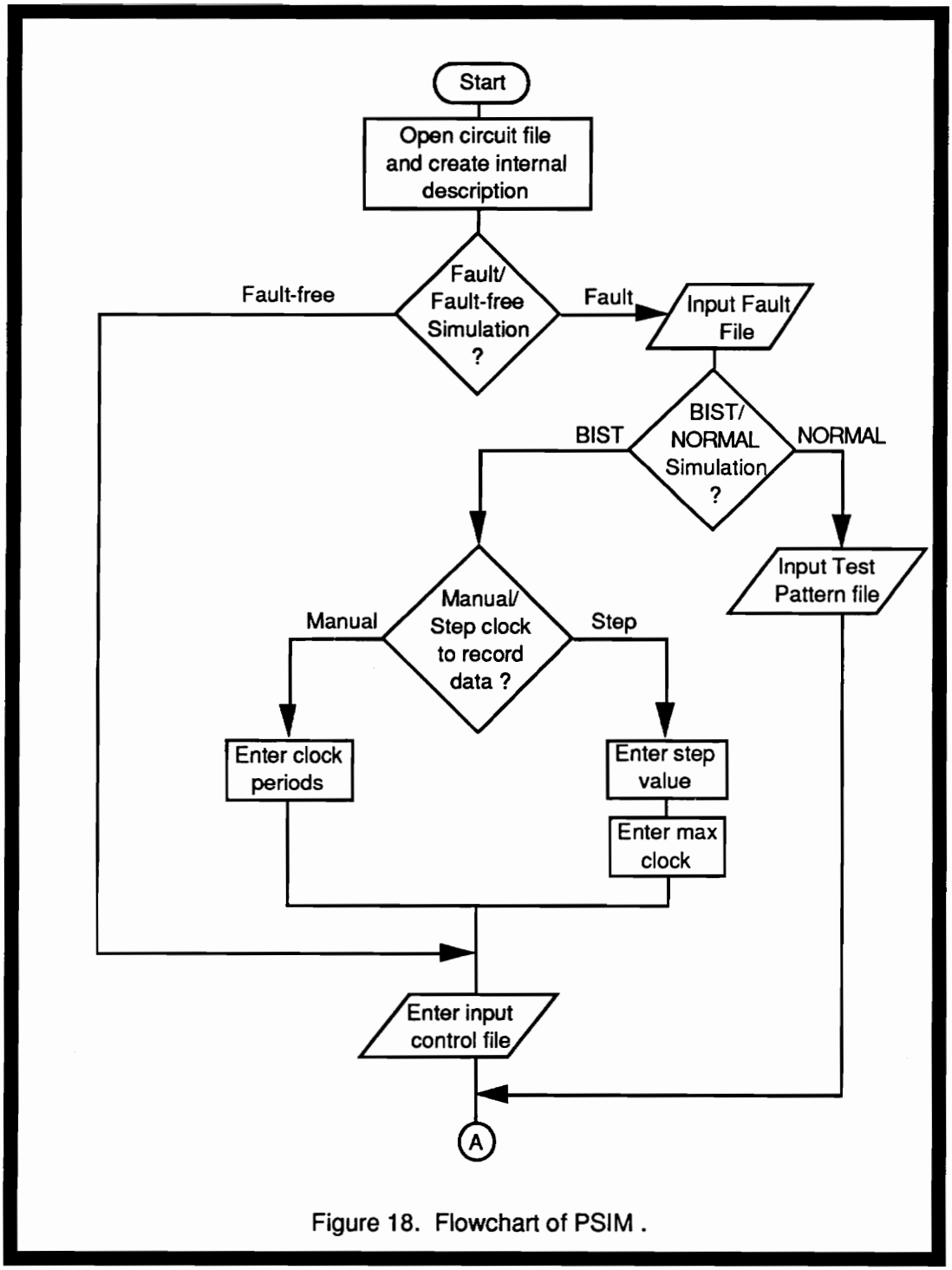


Figure 18. Flowchart of PSIM .

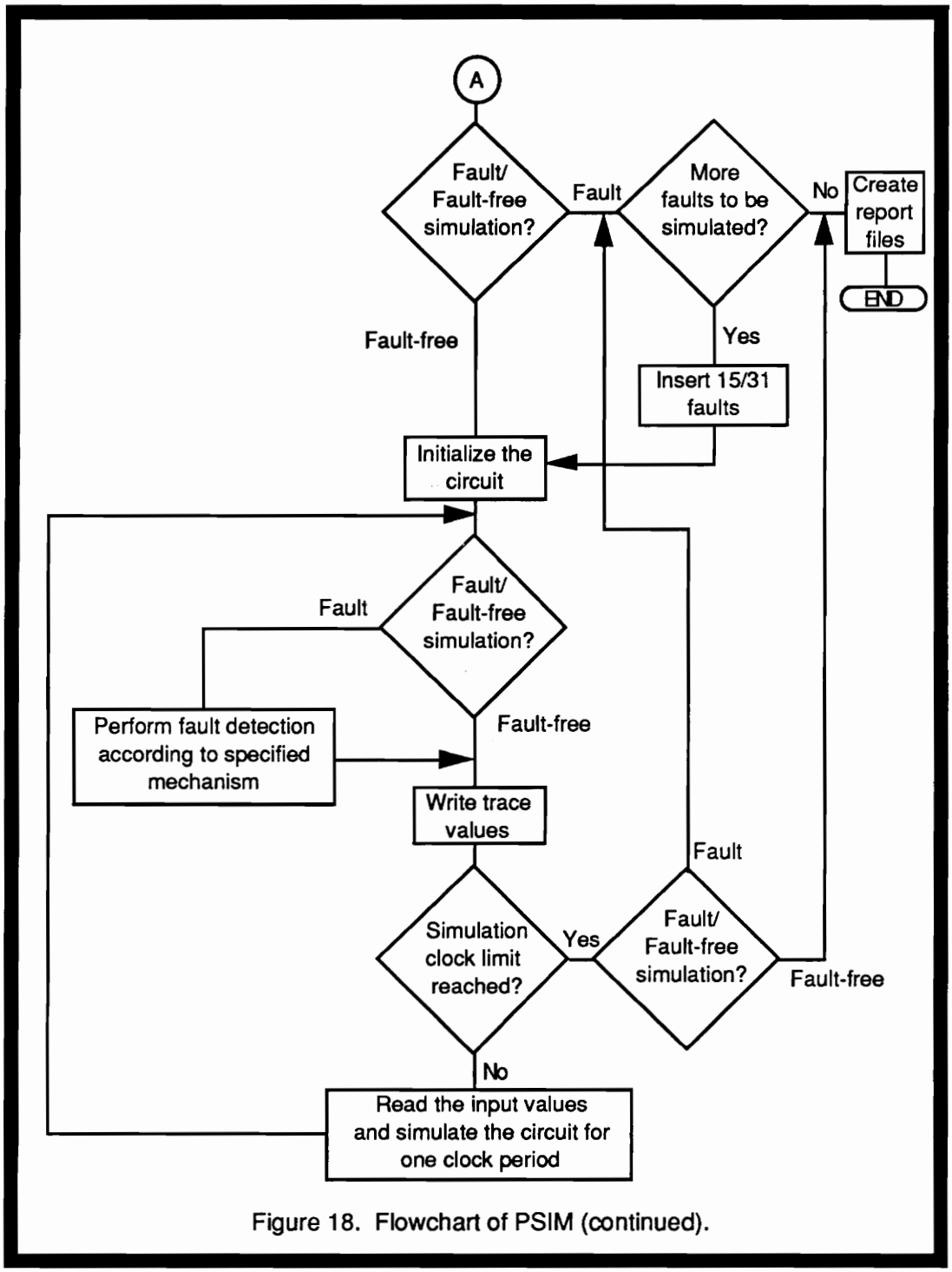


Figure 18. Flowchart of PSIM (continued).

4.0 Experimental Results

A number of experiments were performed on twenty-three ISCAS89 sequential benchmark circuits to obtain a variety of results on aliasing probability. General information on these circuits and on assumptions made during simulations is presented in Section 4.1. The initial set of experiments were conducted to verify the functions of the CBIST and the PSIM tools. Results of these experiments are briefly mentioned in Section 4.2. The second series of simulations were performed to obtain results on aliasing probability for the circular BIST design. Various results of these simulations are reported in Section 4.3. As mentioned earlier, three SARs are built into the PSIM fault simulator. Results on aliasing errors detected in these SARs are reported in Section 4.4.

4.1 Experiment Environment

All experiments were performed on the ISCAS89 benchmark circuits [18]. These circuits are made up of the following primitive elements; AND, NAND, OR, NOR, NOT, INPUT, OUTPUT, and D-type flip-flop. The circular BIST hardware was added to these circuits by the CBIST tool described in the previous chapter. The modified circuit (output of CBIST) is made up of EXOR gates in addition to the primitive elements of the original circuit. All existing flip-flops are included in the scan chain for the purpose of this thesis. The ordering of flip-flops in the chain is arbitrary.

Simulations were performed under the single stuck-at fault model. The FAULTDIR tool was used to generate a set of collapsed faults (including faults in the added BIST circuitry) to be included in the fault file. Results for aliasing errors in the circular BIST design and the three SARs were obtained in the same pass.

For all experiments, initial states of all flip-flops in the circuit were set to zero. The test length in all cases was 2000 test patterns, i.e. 2000 clock periods. These patterns may not all be distinct (more will be said about this in the explanation for limit cycling later in this chapter).

Table 2 gives a profile of the twenty-three ISCAS89 circuits on which simulations were performed. The number of inputs, outputs, gates, flip-flops, and faults for each of the modified circuits is shown in the table.

4.2 Simulations to Verify CBIST and PSIM

Simulations to verify the correct implementation of circular BIST on general sequential circuits by CBIST were performed using the HILO3 fault simulator [19]. The first series of

Table 2. ISCAS89 Benchmark Circuit Profiles

NAME	# of inputs	# of outputs	# of gates	# of flip-flops	# of faults
S208	15	2	164	21	427
S298	7	6	193	23	538
S344	13	11	270	35	696
S349	13	11	271	35	704
S382	7	6	253	30	695
S386	11	7	224	20	600
S400	7	6	260	30	722
S420	23	2	312	37	786
S444	7	6	276	30	770
S510	23	7	312	32	900
S526	7	6	288	30	841
S526n	7	6	289	30	839
S641	39	24	616	78	1311
S713	39	23	629	77	1411
S820	22	19	420	42	1312
S832	22	19	418	42	1332
S838	39	2	602	69	1501
S953	20	23	535	45	1531
S1196	18	14	669	45	1720
S1238	18	14	648	45	1831
S1423	21	5	950	96	2375
S1488	12	19	757	33	1856
S1494	12	19	751	33	1876

simulations were performed to verify that the behavior of the modified circuit does not deviate from that of the original circuit when the former is run in NORMAL mode. A large number of random patterns were applied to the inputs of both the circuits and their outputs were compared. For all circuits on which simulations were performed, the outputs of the original and the modified circuits were observed to be identical.

Simulations were performed on the modified circuits to test the proper functioning of the four different circuit modes. For example, the modified circuits were run in SCAN mode to verify that values at the SCANIN input were correctly shifted in. Similar experiments were performed to test the working of the other three modes. In each case the circuits were found to behave according to expectation.

The last set of simulations were performed to verify the validity of results reported by PSIM. Specifically, the results produced by PSIM were compared with those produced by HILO3. They were found to be identical. These simulations were performed for only the fault-free case. The working of the fault simulation mechanism in PSIM was verified by comparing the number of faults detected by a given test pattern as reported by PSIM with that reported by another fault simulator, HOPE. It was found that each test pattern detected the same number of faults in each case.

4.3 Aliasing Errors in Circular BIST Design

The primary results of simulations to find aliasing errors in circular BIST design are reported in the BATOUT files. The value of aliasing probability at the end of 2000 clock periods is immediately available from these results. Values for the the maximum and the average aliasing probability can be derived from these results.

Results on aliasing probability for the twenty-three ISCAS89 circuits are shown in

Table 3. These values are the probability values found at the end of the test session (after the application of 2000 clock periods). From the table we notice that the aliasing probability is zero for all but two circuits, S386 and S1494. Even for these circuits, only one aliasing error was found at the end of 2000 clock periods. The final values of the optimistic and the true fault coverage are the same for all circuits, except for the two circuits in which aliasing errors were detected.

One observation made from the results obtained from these simulations was that the true fault coverage for a circuit does not necessarily increase monotonically. The optimistic fault coverage always increases monotonically. This can be seen from the graph of the true and the optimistic fault coverage versus the number of clock periods for benchmark circuit S27 shown in Figure 19.

The final fault coverage (true or optimistic) for all circuits simulated is relatively low. The reason for this is that some of the faults in the circuit are never sensitized during the test session. For example, since the control line B1 is held at one during test mode, inverted B1 is always at zero during the test mode. B1 is used as the select line for all multiplexers at primary inputs, therefore, some faults in this multiplexer will never be sensitized. These and other faults which are undetectable during BIST testing may be detected later through deterministic testing.

Some of the circuits, such as S208, S386, show particularly low fault coverage. This is due to a phenomenon called limit cycling. Simply put, limit cycling is said to occur when the same set of patterns generated in the circular chain keeps occurring over and over. This means that only a subset of all possible test patterns are generated by the circular path, no matter how many test patterns (clock periods) are applied. Limit cycling would not be a problem if it occurs near the end of the test session or if the set of patterns repeated is relatively large. It can be a problem if this is not the case. An example of limit cycling in circuit S386 is shown in Figure 20.

Table 3. Aliasing Probability in Circular BIST Design After 2000 Clock Periods

NAME	# of flip-flops	# of faults	Optimistic Fault Coverage (%)	True Fault Coverage (%)	Aliasing Probability ($x10^{-3}$)
S208	21	427	23.653	23.653	0
S298	23	538	87.175	87.175	0
S344	35	696	85.057	85.057	0
S349	35	704	84.943	84.943	0
S382	30	695	88.633	88.633	0
S386	20	600	59.500	59.333	2.801
S400	30	722	87.950	87.950	0
S420	37	786	23.028	23.028	0
S444	30	770	87.922	87.922	0
S510	32	900	86.778	86.778	0
S526	30	841	88.585	88.585	0
S526n	30	839	87.604	87.604	0
S641	78	1311	77.803	77.803	0
S713	77	1411	77.817	77.817	0
S820	42	1312	83.308	83.308	0
S832	42	1332	82.207	82.207	0
S838	69	1501	22.718	22.718	0
S953	45	1531	71.783	71.783	0
S1196	45	1720	88.081	88.081	0
S1238	45	1831	82.687	82.687	0
S1423	96	2375	88.505	88.505	0
S1488	33	1856	92.403	92.403	0
S1494	33	1876	92.004	91.951	0.579

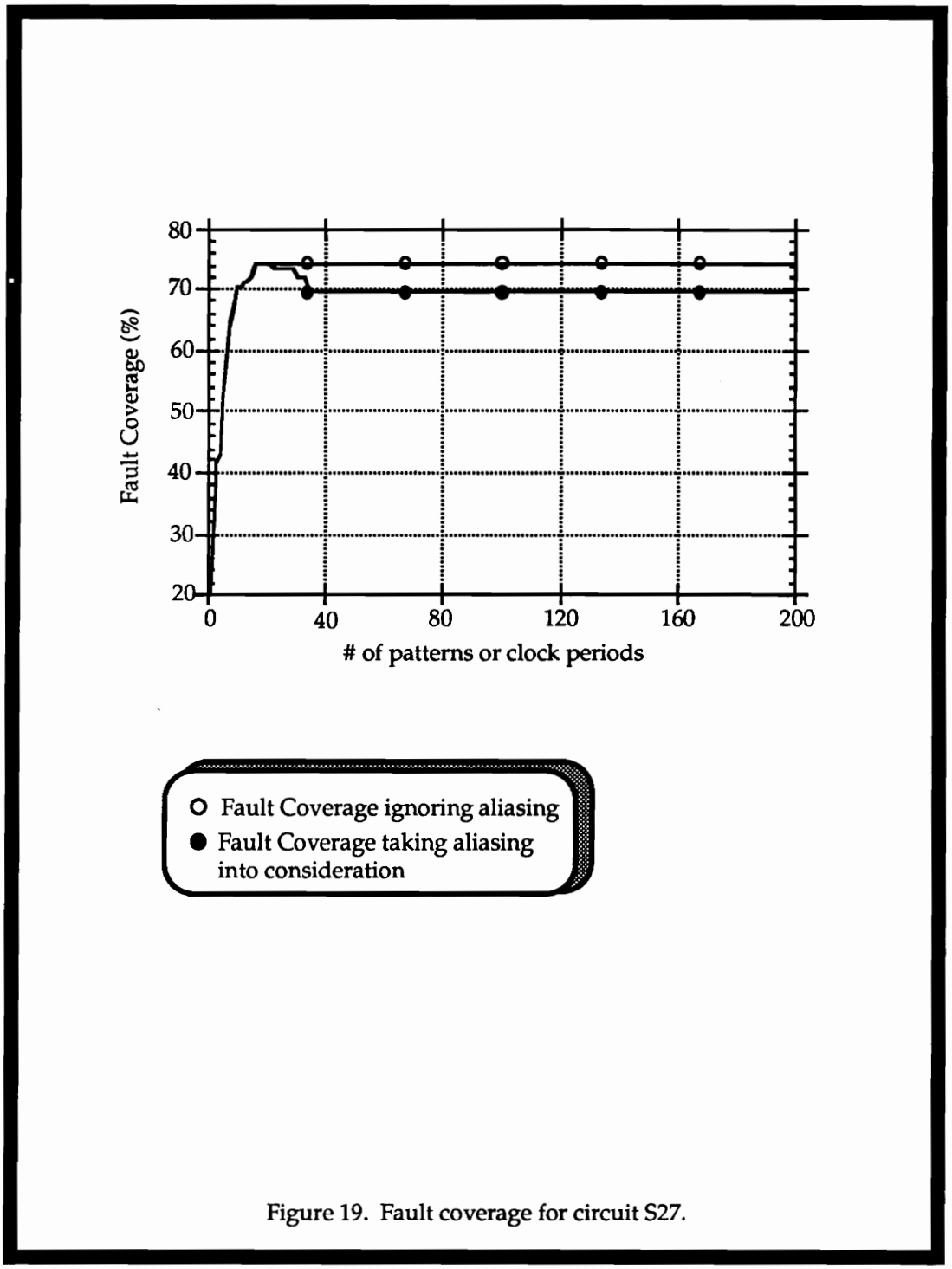


Figure 19. Fault coverage for circuit S27.

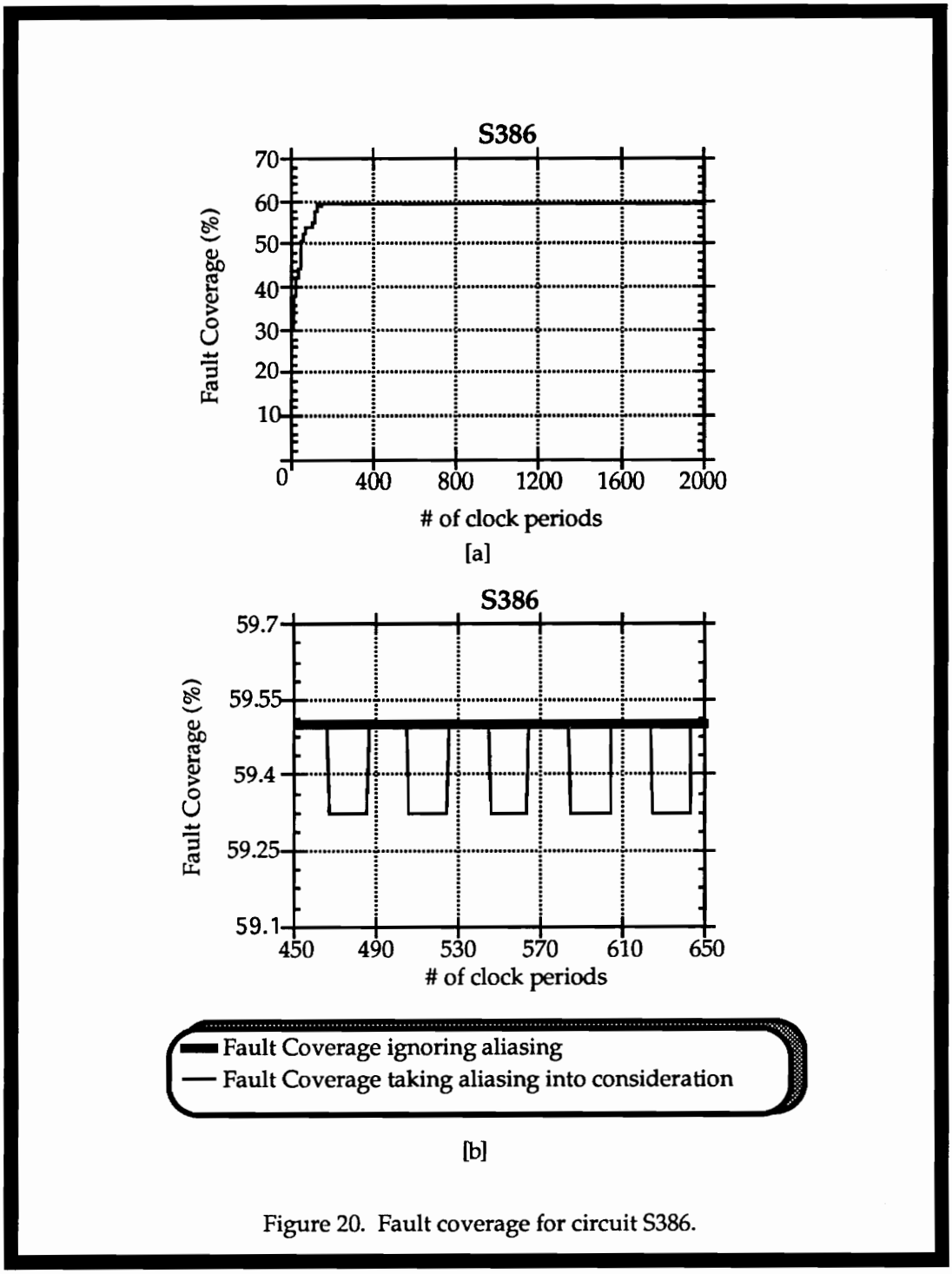


Figure 20. Fault coverage for circuit S386.

Figure 20[a] shows the true fault coverage for the whole test span of 2000 clock cycles. It can be seen that the true fault coverage reaches a maximum of 59.333% early on during the test session. When a portion of this graph is zoomed into (Figure 20[b]), we see that the fault coverage actually oscillates between the values of 59.333% and 59.5%. As expected, the optimistic fault coverage (shown as a bold line) stays constant at 59.5% once it reaches that value. Since the same patterns and the same response are repeated for each cycle (about 40 clock periods), the same aliasing error keeps repeating. Hence the true fault coverage oscillates.

The maximum aliasing probability for a given test length is the highest aliasing probability encountered for the test session at any time during the test. If the test session was terminated any time before it reaches the end, the aliasing probability at that point is guaranteed to be lower than or equal to the maximum aliasing probability.

If the test time (the number of clock periods for which the circuit is run in test mode) is known prior to the design of a testing technique, the exact aliasing probability at the end of the given number of clock periods is an ideal measure of aliasing. The average aliasing probability for a circular BIST design may be a better measure of aliasing if the quality of the testing technique is to be evaluated. The expression for average aliasing probability is shown below.

$$\text{Average Aliasing Probability} = \frac{\text{Number of aliasing errors detected during the test session}}{\text{Number of optimistic fault detects} \times \text{Test time}}$$

Results for the average aliasing probability and the maximum aliasing probability for the ISCAS circuits are shown in Table 4. Aliasing errors were detected in eleven of the twenty three circuits simulated. Note that aliasing errors occur in only two circuits when the test response is observed at the end of 2000 clock periods.

High average aliasing probability could occur in two cases. The first case is in the presence of limit cycling. Since the same patterns and response keep repeating for every cycle,

Table 4. Average and Maximum Aliasing Probability in Circular BIST

NAME	Maximum aliasing probability ($x10^{-3}$)	Average aliasing probability ($x10^{-3}$)
S208	9.9	0.02
S298	0	0
S344	0	0
S349	0	0
S382	0	0
S386	11.204	1.686
S400	0	0
S420	0	0
S444	0	0
S510	10.243	0.958
S526	14.765	0.066
S526n	10.884	0.407
S641	0	0
S713	0	0
S820	0.915	0.012
S832	0	0
S838	0	0
S953	2.73	0.786
S1196	0	0
S1238	0.66	0.003
S1423	0.951	0.04
S1488	0.538	0.324
S1494	0.579	0.39

the aliasing error occurring during this cycle also keeps repeating. The total number of aliasing errors would be very high in this case, which in turn would increase the average aliasing probability. This was the case with circuit S386. The second case in which the total number of aliasing errors would be high is when an aliasing error corresponding to a particular fault is encountered, and that particular fault is not detected again for a large number of clock periods.

4.4 Aliasing Errors in SARs

Results were obtained for the average aliasing probability and the maximum aliasing probability for the 8-bit, 16-bit and the 32-bit SARs connected to one of the flip-flops in the circular path as shown in Figure 17. The results for the 8-bit and the 16-bit SARs are given in Table 5. No aliasing errors were found in the 32-bit SARs. That is, the average and maximum aliasing probabilities in the 32-bit SAR for all circuits were found to be zero. The values of aliasing probabilities in the 8-bit SAR are the highest among the four methods of compaction (circular BIST, 8-bit SAR, 16-bit SAR, 32-bit SAR). This is expected since the aliasing probability in SARs is estimated to be $1/(2^n)$ where n is the length of the SAR. The probability values for the 16-bit SAR are between those for the 8-bit and the 32-bit SARs but substantially lower than that for the 8-bit SAR.

Finally a note about the fault coverage achieved in the SARs is in order. The fault coverage obtained using the SARs for fault detection can never be greater than the optimistic fault coverage observed in the circular path, since the input of the SARs is derived from one of the circular BIST flip-flops. It is possible, though, for the true fault coverage observed in the SARs to be higher than that observed in the circular path. This would be the case when the circular chain containing the fault detection information passes it to the SARs before losing the information itself due to aliasing. This was observed in the case of circuit S386, where the

Table 5. Average and Maximum Aliasing Probability in Signature Registers

NAME	SAR-8 Maximum Probability ($x10^{-3}$)	SAR-8 Average Probability ($x10^{-3}$)	SAR-16 Maximum Probability ($x10^{-3}$)	SAR-16 Average Probability ($x10^{-3}$)
S208	59.406	3.158	19.802	0.069
S298	21.322	3.679	4.264	0.016
S344	21.959	3.866	3.378	0.013
S349	18.395	3.908	3.344	0.02
S382	17.857	3.991	3.247	0.03
S386	42.027	3.893	5.602	0.013
S400	20.472	3.935	3.15	0.005
S420	60.773	4.5	5.525	0.008
S444	14.771	3.728	4.431	0.013
S510	43.534	3.753	2.561	0.019
S526	18.792	3.686	1.342	0.003
S526n	14.966	3.987	2.721	0.013
S641	20.588	3.846	1.961	0.02
S713	24.59	4.014	1.821	0.016
S820	22.873	3.647	1.83	0.009
S832	24.657	3.664	1.826	0.007
S838	32.258	3.654	5.865	0.015
S953	34.577	3.579	2.73	0.04
S1196	13.861	3.561	1.98	0.015
S1238	13.21	3.796	1.321	0.011
S1423	11.418	3.931	1.903	0.021
S1488	13.442	3.663	1.753	0.015
S1494	11.594	3.725	1.739	0.013

coverage at the end of the test session in the circular path was observed to be lower than in the SARs. The reverse situation, fault coverage in circular path being greater than that in the SARs, is also possible if the SAR suffers from a high number of aliasing errors. Such is the case with the 8-bit SAR.

5.0 Conclusion

5.1 Summary

The primary goal of this thesis was to obtain experimental results on aliasing probability in the circular BIST design technique. A number of tools were developed to accomplish this goal. A circular BIST test synthesis tool (CBIST) was developed to automate the process of implementing circular BIST design for general sequential circuits consisting of flip-flops and primitive gates. Existing fault simulators were found to be too cumbersome to use for detecting aliasing errors in the CBIST design. A parallel-fault simulator (PSIM) suitable for this purpose was developed to solve this problem. A number of other ancillary tools were developed to support the experiments done for deriving results obtained for this thesis.

The CBIST tool was used to implement the circular BIST design for 23 ISCAS89 sequential benchmark circuits. These circuits were simulated using the PSIM fault simulator to detect aliasing errors. The circuits were operated in the BIST (test) mode for 2000 clock periods. The first series of results obtained were the aliasing probability observed at the end of the test session for the 23 circuits. Aliasing errors were detected in only 2 of the 23 circuits at the end of the application of 2000 clock periods. The second series of results reported were the maximum and the average aliasing probability encountered in the 23 circuits over a period of 2000 clock periods. Aliasing errors were detected in 11 of the 23 circuits simulated. The maximum values of the average aliasing probability and the maximum aliasing probability for any of the 23

circuits was 0.0147 and 0.0016, respectively. The last series of results reported were the average and maximum aliasing probability observed in an 8,16, and 32-bit signature analysis register connected to the circular paths of the 23 circuits as shown in figure 16. As expected, the 8-bit SAR exhibited the maximum number of aliasing errors. No aliasing errors were detected in the 32-bit SAR for any circuit. Substantially lower number of aliasing errors were encountered in the 16-bit SAR compared to the 8-bit SAR.

5.2 Suggestions for Future Work

A number of possibilities exist for continuing investigations on the circular BIST design technique and aliasing errors in circular BIST design. Some of them are listed below.

- An obvious area of further research would be to find the aliasing probability in circular BIST design implemented using partial scan (instead of full scan used in this thesis). The effect of the length of the circular path on aliasing probability could be investigated.
- Experiments could be carried out in which the effect of reordering the flip-flops in a circular path could be observed.
- As mentioned in Chapter 4, a number of faults are never sensitized during BIST testing. The test coverage obtained for all circuits was rather low due to this reason. Some way of detecting these undetected faults should be found, such as supplementing the BIST method with some other form of testing.
- Limit cycling was discovered to be a detriment to the circular BIST design. Effects of such variables as the initial value of flip-flops in the chain and the ordering of the flip-flop in the chain on occurrence of limit cycling could be investigated.

In summary, more knowledge about important issues such as limit cycling is necessary before this design technique can be adopted in practice.

References

[1] E. J. McCluskey, "A survey of design for testability scan techniques", *VLSI Design*, Vol. V, No. 12, Dec. 1984, pp.38-61.

[2] E. B. Eichelberger and T. W. Williams, "A logic design structure for LSI Testability", *Proc. 14th Design Automation Conf.*, June 1977, pp. 462-468.

[3] B. Konemann, J. Mucha, and G. Zwiehoff, "Built-in logic block observation techniques", *Digest 1979 Test Conference*, Oct. 1979, pp. 37-41.

[4] B. Konemann, J. Mucha, and G. Zwiehoff, "Built-in test for complex digital integrated circuits", *IEEE J. Solid-State Circuits*, vol. SC-15, No. 3, June 1980, pp. 315-318.

[5] P. H. Bardell and W. H. McAnney, "Self-testing of multichip logic modules", *Digest 1982 International Test Conference*, Nov. 1982, pp. 200-204.

[6] D. Komonytsky, "LSI self-test using level sensitive scan design and signature analysis", *Digest 1982 International Test Conference*, Nov. 1982, pp. 414-424.

[7] A. Krasniewski and S. Pilarski, "Circular self-test path: a low-cost BIST technique for VLSI circuits", *IEEE Transactions on Computer-Aided Design*, Vol. 8, No. 1, Jan 1989, pp. 46-55.

[8] N. Benowitz, D. F. Calhoun, G. E. Alderson, J. E. Bauer, and C. T. Joeckel, "An advanced fault isolation system for digital logic", *IEEE Transactions on Computers*, vol. C-24, No. 5, May 1975, pp. 489-497.

[9] R. A. Frohwerk, "Signature analysis: A new digital field service method", *Hewlett Packard Journal*, May 1977, pp. 2-8.

[10] A. Y. Chan, "Easy-to-use signature analyzer troubleshoots complex logic circuits", *Hewlett Packard Journal*, May 1977, pp. 9-14.

[11] P. P. Fasang, "BIDCO, built-in digital circuit observer", *Digest 1980 Test Conference*, Nov. 1980, pp. 261-266.

[12] R. M. Sedmak, "Implementation techniques for self-verification", *Digest 1980 Test Conference*, Nov. 1980, pp. 267-278.

[13] E. J. McCluskey, "Built-In Self-Test Techniques", *IEEE Design & Test of Computers*, April 1985, pp. 21-28.

[14] E. J. McCluskey, "Built-In Self-Test Structures", *IEEE Design & Test of Computers*, April 1985, pp. 29-36.

[15] C. E. Stroud, "Automated BIST for sequential logic synthesis", *IEEE Design & Test of Computers*, Dec. 1988, pp. 22-32.

[16] C. E. Stroud, "An automated BIST approach for general sequential logic synthesis", *Proc. 25th ACM/IEEE Design Automation Conference*, June 1988, pp. 3-8.

[17] M. M. Pradhan, E. J. O'Brien, S. L. Lam, and J. Beausang, "Circular BIST with partial scan", *Digest 1988 International Test Conference*, pp. 719-729.

[18] F. Brglez, D. Bryan, K. Kozminski, "Combinational Profiles of Sequential Benchmark Circuits" , *1989 International Symposium on Circuits and Systems*, May 1989.

[19] HILO3 Users Training Manual.

[20] D. K. Pradhan (editor), *Fault-tolerant computing - theory and techniques*, volume1, chapter 3, pp. 184-265.

Appendix.

The usage and other relevant details for tools developed for this thesis form the contents of this appendix. Some information is repeated from chapter 3 for the purpose of completeness.

A.1 CBIST

CBIST is used to synthesize circular BIST hardware on general sequential circuits consisting of flip-flops and combinational logic blocks. The usage is

CBIST <*input file*>

where *"input file"* is the file name of a circuit in ISCAS format. Each element in an ISCAS circuit description file is described in the format shown below.

outnode = gate_type(innode1, innode2, ... , innoden)

where *outnode* is the name of the output node of the element, *innodex* are the names of the fan-in signals of the element, and *gate_type* is the type of the element. The name of the output file created by CBIST, where the modified circuit description in ISCAS format is stored, is *"input file"*.bist.

A few restrictions on *input file* are mentioned below.

- All comments (lines beginning with a '#') must be at the beginning of the circuit file.

- The order of description of elements in the files should be as follows: 1. INPUT 2. OUTPUT 3. DFF 4. All other element types.
- Elements of different types must be described in different blocks with each block being separated from any other block by at least one line.

Comments in the modified circuit are directly copied from the original circuit file and do not reflect the added test hardware.

A.2 ISCTOHILO

ISCTOHILO is used to convert the net list format of a circuit in ISCAS format to a circuit in HILO format. The usage is

ISCTOHILO *<input file> <output file>*

where *input file* is the circuit description file in ISCAS format and *output file* is the circuit description file in HILO format. Refer to the HILO3 user's manual for the format of HILO circuit description file. Delays for all elements in the HILO description file are assumed to be zero. Outputs are referred to as WIREs. Since no primitive elements for exclusive or gates and D-type flip-flops are defined in HILO3, these elements are called XON and TOY respectively in the HILO circuit file created by ISCTOHILO. The definitions for these modules must be present in the HILO3 SCRATCH area before any simulation can take place. The user must create subcircuit modules for these elements (with names XON and TOY). These modules must then be copied into the SCRATCH area before simulating the circuit. Refer to the HILO3 user's manual for details.

A.3 ISCTOPSIM

ISCTOPSIM is a net list format conversion tool to convert circuit description files in ISCAS format to PSIM format. The net names in the ISCAS circuit file are converted to net numbers. The correspondence between the net names and the net numbers is described in a mapping file created by ISCTOPSIM. The usage for the tool is

ISCTOPSIM *<input file>*

where *input file* is a circuit description file in ISCAS format. The name of the circuit description file in PSIM format, created by ISCTOPSIM, is "f.a". The format of element description in PSIM format is as shown below.

<input1> <input2> ... <inputn> <element_type> ; <output_number> <output_name>

Input1, input2, ... inputn are the signal numbers for the inputs to the element, *output_number* is the signal number for the output of the element, and *output_name* is the name of the output signal as defined in the ISCAS file. *Element_type* can be INPUT, OUTPUT, NOT, BUF, AND, NAND, OR, NOR, XOR, or DFF. The name of the mapping file created by ISCTOPSIM is "MP_*<input file>*".

A.4 FAULTDIR

FAULTDIR is used to create a collapsed fault file for a circuit. The program is invoked

by typing FAULTDIR at the command line. The user is prompted for the name of a circuit description file in PSIM format and the name of the fault file to be created by PSIM. Faults are listed in the following format in the fault file.

gate/pin/type/1

gate is the output signal number of the gate at which a fault is to be injected, *pin* is the number of the signal of the gate at which the fault is to be injected. *type* can be either 0 or 1 indicating a stuck-at-0 or a stuck-at-1 type fault respectively. Ignore the 1 at the end.

A.5 PSIM

PSIM is an interactive event-driven parallel-fault simulator, developed to detect aliasing errors in circular BIST design. The usage for PSIM is

PSIM <-*x*>

The command line parameter *x* chooses the type of fault detection mechanism used by PSIM to calculate results. The valid values for *x* are 8, 16, 32, or a. The 8, 16, and 32 options choose the 8-bit, 16-bit, and 32-bit SARs to be used for fault detection. The default is only the circular BIST fault detection mechanism (if no command line parameter is given), unless the NORMAL fault mechanism is chosen by the user. If the -a option is chosen, results for all four fault-detection mechanism are reported.

After the simulator is invoked, the user is prompted for a variety of information. An example of a typical session is given below. The italicized words are responses given by the user and the normal text is what is printed by PSIM on the terminal.

Enter the name of the circuit file. *s27.bist*

Enter the fault file name.

If fault-free simulation is to be performed,

hit return. *s27.bflt*

BIST/NORMAL FAULT SIMULATION? (1/2) *1*

MANUAL INPUT/STEP (1/2) ? *2*

ENTER THE CLOCK STEP VALUE *1*

ENTER THE MAXIMUM CLOCK PERIOD *200*

Enter the input file name.

If no input file is to be supplied hit return.

If no input file is given, the circuit will be

initialized with all inputs and flip-flops at zero. *s27.200*

If NORMAL fault simulation option is chosen instead of BIST, a file called PATTERNS is created by PSIM, which lists the number of undetected faults detected by the pattern applied at each clock period. If BIST option is chosen, depending on the command line parameter, a BATOUTx (BATOUT, BATOUT8, BATOUT16, or BATOUT32) file is generated. The BATOUT file is generated regardless of the option chosen. The -8, -16, and -32 options generate BATOUT8, BATOUT16, BATOUT32 file in addition to the BATOUT file, and the -a option generates all four files. The BATOUTx files report results in the following format.

CKT CLOCK TOTAL FBFLT TBFLT FFLTC TFLTC

where CKT is the circuit name, CLOCK is the number of clock period, TOTAL is the total number of faults in the circuit, FBFLT is the number of optimistic fault detects, TBFLT is the number of true fault detects, FFLTC is the optimistic fault coverage, and TFLTC is the true fault coverage.

The input signal file has two different directives. The first is TRACE. It is used to trace signal values during simulation. The values of signals specified in the directive are

output to a file called TRACE. This directive, if it appears in the input signal file, should be the first line in the file. The syntax is as shown below.

TRACE: {*signal1, signal2, ... signaln*}

where *signal1, signal2 ... signaln* are the signal numbers whose logic values are to be written to the TRACE file. The signal numbers must be separated by one comma and at least one space. The logic values in the TRACE file are written in hexadecimal form. A maximum of 12 signals can be traced at a time. Using TRACE makes sense only when fault-free simulation is to be performed or the number of faults to be simulated is less than 32 (the TRACE file would be overwritten for each pass during fault simulation if more than 31 faults are to be simulated).

The second directive is END. It is used to indicate the end of simulation. It must appear as the last line in an input signal file. The syntax is

END: {*last_clock*}

where *last_clock* is the clock period till which simulation is to be performed.

The syntax for specifying changes in signal values is shown below.

clock_num: {*signal1 = value, signal2 = value, ... signaln = value*}

where *clock_num* is the clock period at which the specified changes are to take place, *signal1, signal2, ... signaln* are the signal numbers whose logic values are to be changed, and *value* is the logic level (0 or 1) to be applied to the signal. An example of an input signal file is shown below.

TRACE: {5, 6, 7, 8, 88, 90, 91, 92}

1: {5 = 0, 6 = 1, 7 = 0}

8: {7 = 1)

19: {7 = 0}

END: {30}

VITA

Rajiv D. Kothari was born in Bombay, India in 1964. He received a B.Sc. degree in Physics from the University of Bombay in 1985. He was enrolled in the B.S. program in Electrical Engineering at Virginia Tech from 1986 to 1988. He then transferred to the M.S. program in Electrical Engineering at Virginia Tech. He hopes to continue his research in the areas of CAD Engineering and VLSI testing at Intel Corporation in Santa Clara, California.