MINIMISATION OF TEST APPLICATION TIME IN THE SCAN TECHNIQUE

BY

KWASI ADU-BOAHEN OPARE

BSC. ELECTRICAL/ELECTRONIC ENGINEERING (HONS.)

A THESIS SUBMITTED TO THE DEPARTMENT OF TELECOMMUNICATION

ENGINEERING,

KWAME NKRUMAH UNIVERSITY OF SCIENCE AND TECHNOLOGY

IN PARTIAL FUL LMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

FACULTY OF ELECTRICAL AND COMPUTER ENGINEERING

COLLEGE OF ENGINEERING

JANUARY, 2010

DECLARATION

I hereby declare that this submission is my own work towards the MSc and that, to the best of my knowledge, it contains no material previously published by another person nor material which has been accepted for the award of any other degree of the University, except where due acknowledgement has been made in the text.

....

Student Name & ID Signature Date

Certi ed by:

....

Supervisor(s) Name Signature Date

Certi ed by:

....

Head of Dept. Name Signature Date

ABSTRACT

The focus of this study is on how test application time in the scan technique can be minimized. A novel method of how this can be achieved, called the Vector Match Approach, is presented. It takes advantage of matching patterns in test vectors, by rearranging them such that those with matching patterns are closer to each other. Flow charts describing the logic of the algorithm and an example, illustrating how it works are also shown. In addition, the circuit architecture for the Vector Match Approach is presented. It incorporates a scan register and a multiple-input signature register (MISR) together with the circuit under test.

TABLE OF CONTENTS

TABLE OF CONTENTS

LIST OF TABLES

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENT

My sincere appreciation goes to my supervisor, Dr K. O. Boateng, for his guidance, comments, contributions and constructive criticisms, throughout the period of this study. I also express my gratitude to the other members of the Faculty of Electrical and Computer Engineering.

Special appreciation goes to my parents, Professor and Mrs J. A. Opare for their love and support throughout my years of education, my wife, Mrs Dina Opare, for her continuing support, and to the Lord God Almighty who has been my
constant source of inspiration throughout the period of this study.

Finally I would like to express my sincere appreciation to all others who in diverse ways assisted me in the completion of this thesis.

CHAPTER 1

INTRODUCTION

Digital circuits have become indispensable in modern times. They can be found in many computerised and digital systems used in critical, as well as non-critical applications. The critical systems are needed in such elds as Medicine and Communication. In less critical applications, digital circuits are found in devices such as the mobile phone and the digital camera. They are also found in some domestic appliances, including the washing machine and the television. Considering the varied applications in which they are employed, it is essential that digital circuits conform exactly to desired requirements.

To ensure that digital devices perform as required, designs of digital circuits are tested to verify that they correctly implement given speci cations. Multiple manufactured copies of a design are also tested after fabrication. This is to ensure
that the manufacturing process has not introduced aws [1]. In some cases, devices being used on the eld are also tested to ensure that they still conform to functional requirements after a period of usage. The basis of all testing techniques is to apply prede ned sets of inputs, called tests, to a circuit and compare the outputs observed with the patterns that a correctly functioning circuit is supposed to produce [1]. In test application therefore, circuit lines and pins should be accessible for the test vectors to be applied. In combinational circuits, accessing circuit lines and pins is not di cult. Accessing them on sequential circuits on the other hand is very di cult. This is however overcome by making changes to the given circuit design to help decrease its overall di culty of testing. This process is termed Design for Testability (DFT) [2].

1.1. PROBLEM STATEMENT

One DFT method used for sequential circuits is the Scan Technique. In this method, ip- ops (or latches) within the circuit are designed and connected in a manner that enables the circuit to operate in two modes; normal and test modes [3, 4]. With such an arrangement, tests can be applied to inaccessible lines in sequential circuits by

shifting them into the ip- ops. In normal mode, the logic behaviour of the system is identical to that of the original circuit design without the scan device. The response to the applied tests is then captured in the ip- ops. In test mode, the values captured in the ip- ops are observed. Simultaneously, the next test values are applied by shifting (scanning) them into
the ip- ops.

1.1 Problem Statement

The use of scan techniques in circuits is useful in test application. This is because; scan increases the controllability and observability of sequential circuits. In other words, otherwise inaccessible lines in sequential circuits become accessible when scan techniques are introduced. According to [2], when the rst batch of chips (rst silicon) for a new design is fabricated, the use of scan helps in locating design errors and weaknesses during debugging. In operational systems, the use of scan also helps in locating failing components, thereby reducing the time required to repair the system.

One of the low sides of the scan technique is that, during test application, every bit in the test vectors is scanned in one clock cycle at a time. According to [2], for a sequential circuit with *m* state inputs, if *n* vectors are needed to test for certain faults, the number of clock cycles taken to scan them in is shown in Equation 1.1.[1] In addition, it takes the number of clock cycles shown in Equation 1.2 to scan out the last response.

1.2. OBJECTIVES

Putting all together, the total number of clock cycles required to apply *n* test vectors to a circuit with *m* state inputs is shown in Equation 1.3.

$$C_{in} = n(m + 1) \tag{1.1}$$

$$C_{out} = m - 1 \tag{1.2}$$

$$C_{clk} = n(m + 1) + m - 1 \tag{1.3}$$

[1] The term in brackets contains the value 1, to take care of the additional clock cycle needed to apply the scanned-in vector.

It can be deduced from the equation that, as the number of test vectors (*n*) and/or bits (*m*) in them increases, the number of clock cycles needed to scan in all of the bits also increases, thereby, increasing test application time correspondingly.

It is the desire of every manufacturer, that the time spent in producing a nished product is reduced. In the case of the integrated circuit (IC) manufacturing industry, circuits produced must be tested before they are sent out, adding to the production time of circuits. Therefore, reducing the test application time of circuits reduces the production time which translates into shorter time to market.

1.2 Objectives

1. To review the architecture of the Scan Design for Testability (DFT) Technique used in CMOS circuits.
2. To develop an algorithm to reduce test application time in scan-based circuits.

1.3 Scope of Work

In this work, a modi cation of the scan architecture by incorporating a MultipleInput Signature Register (MISR) has been proposed. Based on the modi ed architecture, an algorithm has been developed to exploit pattern repetition in test stimuli in order to reduce test application time.

Chapter 2

Literature Review

2.1 Faults

According to the Chambers 21st Century Dictionary, a fault can be de ned as a aw or defect in an object or structure. In digital circuits, a fault can bring about errors which can lead to system failure. In other words, when a fault is present in a digital circuit, it can lead to errors which will prevent that circuit from doing what it is supposed to do.

Faults in digital circuits come about as a result of physical defects. Some of these physical defects are located in the semiconductor wafer. Wafer defects are found in clusters which are randomly distributed over the whole wafer. Every part of the wafer has an equal probability of having a defect cluster. Furthermore, any part of a di usion, polysilicon, or metal line may also have an open fault. Open faults can also occur between contacts of any two layers. Bridging is another type of fault that may also occur in the wafer. It occurs between any two electrical nodes, whether they belong to one layer or di erent layers. Bridging among multiple nodes is also equally likely. Defects can also be brought about by misalignment, dust and other particles, pinholes in dielectrics, mask scratches and thickness
variations [5].

Faults that occur in circuits can be classi ed as either permanent or non-permanent, according to the way they manifest themselves in time [2]. Functional design errors, manufacturing defects, broken components or parts of components and breaks and shorts caused by design faults or solder splashes are examples of permanent faults [2]. Non-permanent faults can either be transient or intermittent.

2.2. ABSTRACTION LEVELS IN DIGITAL CIRCUITS

Environmental conditions such as humidity, temperature, pressure, vibration and electromagnetic interference cause transient faults. Intermittent faults on the other hand are caused by non-environmental conditions such as loose connections or ageing components.

2.2 Abstraction Levels in Digital Circuits

Dealing with faults at the chip level is complex in most cases. This is because numerous defects exist at that level [1]. Moreover, such faults are di cult to analyse. To deal with the complexities involved, actual defects that have the capability of

occurring in chips are modelled with fault models at higher levels of abstraction. By so doing, the need to derive tests for every possible fault is eliminated. The reason is that, many defects that occur at the chip level manifest themselves as a single fault at higher levels. Moreover, modelling faults at higher levels of abstraction enables the fault model to be technologically independent
[2].

Traditionally, various levels of abstraction at which modelling is done are, behavioural, functional, structural, switch-level and geometrical [2]. These levels of abstraction are the various levels at which an electronic circuit can be described.
(See Figure 2.1).

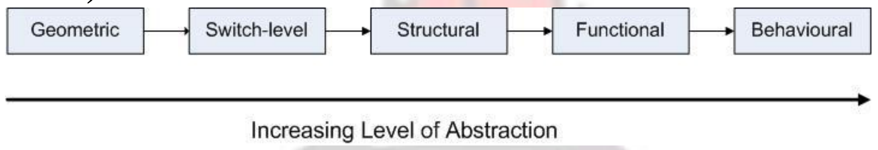


Figure 2.1: Levels of Abstraction in Digital Circuits

At the behavioural level, a digital system is described using a hardware descriptive language, such as Verilog or VHDL. It is at this level that the ow of data and control signals in the system is described.

2.3. FAULT MODELS

A digital system can also be described at the register-transfer level (RTL). This method of description is referred to as the functional description. The description at this level may contain registers, adders, multipliers, multiplexers, decoders and other modules.

The structural level description of a digital system is given at the logic level. At this level, logic gates, such as *AND*, *OR*, *NOT*, *NAND*, *NOR*, *XOR* and their interconnections are used in the description.

The transistor-level details of a circuit are seen in the switch-level description of that circuit. At this level, the various transistors that make up the nMOS and pMOS networks in the CMOS technology implementation of a circuit can be seen.

Finally, the geometric description of a circuit depicts the layout of that circuit. This description enables one to determine device geometries. Line widths, interline and inter-component distances can also be determined from this description.

2.3 Fault Models

The process of modelling defects at higher levels of abstraction in the design hierarchy is referred to as fault modelling [6, 7]. In testing CMOS circuits, they can be modelled at either the gate or switch level. The level a circuit is modelled depends on the kind of fault being tested. For example, in modelling the stuck-at fault (SAF), a circuit is modelled at the structural or gate level. Stuck-on and stuck-open fault levels are however modelled at switch level.

2.3.1 Stuck-at Fault Model

In this model, a fault manifests itself as an interconnection or line (input or output of a gate) being permanently stuck at logic 0 or 1. A line, *w*, stuck-at-0 (SA0) is denoted, w/0. It is denoted w/1 if the value is stuck-at-1 (SA1) [1]. Consider the circuit in Figure 2.2. Line *w* is assumed to have a stuck-at 1 fault. It can be realised that, applying a test

vector (*abc*) = (001) to the inputs of the circuit results in logic 0 on line *w* instead of logic 1. This error is propagated right through to the output *Z*. At the output, a logic value 1 is detected instead of 0. From this illustration, it can be seen that stuck-at faults, being logical faults
have the capability of changing the intended logical function of a digital circuit.

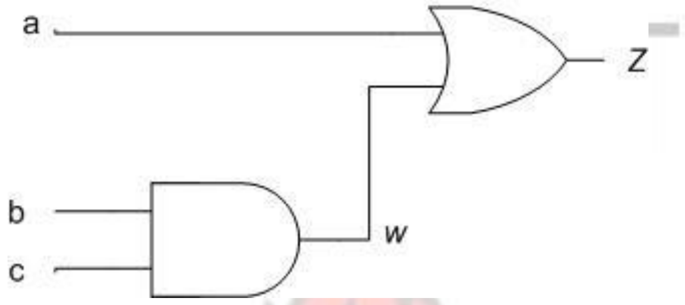


Figure 2.2: A stuck-at-1 fault on line *w*

According to [8], the stuck-at fault model is an abstract model. In other words, it does not model a physical defect. This, however, does not mean that faults caused by physical defects cannot be detected. On the contrary, a physical defect may manifest itself as an error in the logic of a circuit. In that case, such an error will be detected at the gate level. By staying away from physical details the stuck-at fault model remains e ective with changing technologies and design
styles [8].

2.3.2 Stuck-open and Stuck-on Fault Models

Stuck-open and stuck-on faults, denoted, SOpF and SOnF respectively, a ect the condition of transistors in a circuit. A stuck-open fault refers to a transistor that
has become permanently non-conducting. A stuck-on fault on the other hand refers to a transistor that has become permanently conducting. Consider the circuit in Figure 2.3(a). It shows a CMOS implementation of the *NOR* gate.

The pull-down network is made up of nMOS transistors Q_1 and Q_2 connected in parallel. The pull-up network is also made up of pMOS transistors Q_3 and Q_4 connected in series. Now, supposing a defect, *d* exists which causes an open that prevents Q_1

from conducting. In such a condition, Q_1 is said to have a stuck-open fault (SOpF). Let the set of input vectors in the following sequence
(w_1,w_2) = (0,0),(0,1),(1,0),(1,1) be applied. When (0,0) is applied, Q_3 and Q_4 conduct, whilst the output *F* becomes logic 1. When the vector (0,1) is applied next, *F* becomes logic 0 because Q_2 conducts. Upon the application of
(1,0), the SOpF in Q_1 causes the path from *F* to *V ss* to be non-conducting. The output *F*, therefore retains the previous logic value, which is 0 in this case. Finally, applying vector (1,1) causes Q_2 to conduct thereby bringing output *F* to logic 0. Now, with the scenario described above, there can be a false conclusion that no fault exists even though Q_1 is stuck-open. To be able to detect the SOpF, a two-pattern test, consisting of an initialisation vector and a test vector are employed. The initialisation vector sets the output of the circuit to a logic value which is the complement of the logic value expected in a fault-free circuit. In our example in Figure 2.3(a), the vector (1,0) caused *F* to be logic 0. The initialisation vector is therefore supposed to cause *F* to be logic 1. Vector (0,0) is capable of achieving the expected results. The vectors are applied to the circuit in the following sequence; (0,0),(1,0). In the presence of SOpF at Q_1, the output *F* becomes logic 1 when such a sequence is applied. The fault is therefore detected.

Figure 2.3(b) shows a two-input *NAND* gate with an assumed defect *d* resulting in an SOnF in transistor Q_4. To test for the fault, let the vector (1,1) be applied. If the fault exists, transistors Q_1, Q_2 and Q_4 conduct. An intermediate voltage therefore results at the output. Depending on the resistances of the nMOS and pMOS transistors, the exact voltage may map unto either logic 0 or 1. The SOnF is detected if the voltage maps unto logic 1 at the output, otherwise the fault is not detected. Again, let us assume that an SOnF exists in the transistor Q_2 this time around. Let us apply vector (1,0) to test for this fault. If the fault exists,

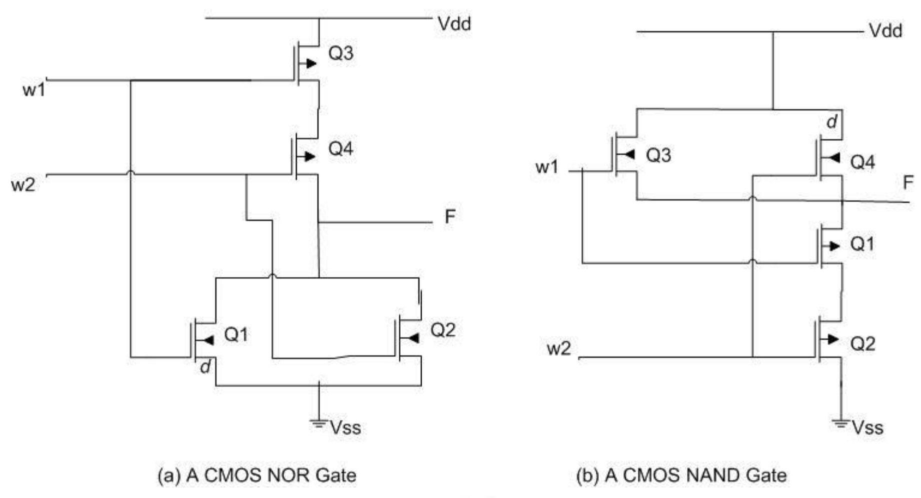


Figure 2.3: Two-input CMOS Gates

transistors Q_1, Q_2 and Q_4 conduct, resulting in an intermediate voltage at the output. This is similar to the previous scenario, however, this time around the intermediate voltage must map unto logic 0 at the output for the fault to be
detected. There is therefore a contradiction in the requirements for the detection of SOnFs in Q_2 and Q_4. This shows that logic monitoring, that is, monitoring the logic value at the output of the gate is not adequate for detecting all SOnFs in it [2]. Current monitoring is a solution to this inadequacy. In this method, the current drawn by the circuit is measured to ensure the detection of all SOnFs in
the circuit. [9, 10, 11, 12].

2.3.3 Bridge Fault Model

Bridge fault models deal with faults that come about as a result of defects caused by shorts. There are several ways in which bridge faults can be modelled. One of them is the Wired-Logic model. In this model, it is assumed that bridges cause wired-*AND* and wired-*OR*. In other words, the bridge fault between the two lines causes the lines to either result in an *AND* or an *OR* function respectively.

In Figure 2.4, the points connected with the dotted lines may be modelled as either an *AND* or an *OR*. This model, however, is only applicable in non-

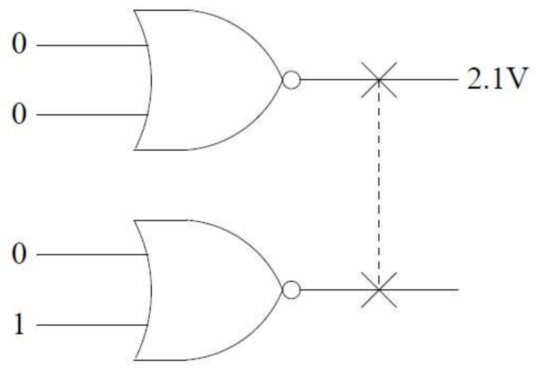


Figure 2.4: Bridge fault model

CMOS technology [13]. Another bridge fault model which can be used in CMOS technology as well, is the Voting model. In this model, a con ict between the nMOS and pMOS networks is assumed. The reason for this assumption is that, the fault occurs between a node which has logic 1 and another with logic 0. The model therefore tries to nd the correct operating point by examining the transistor con gurations of both the nMOS and pMOS networks [14].

2.3.4 Delay Fault Model

Delay faults occur when the time taken for a signal to propagate falls outside the bounds required for normal operation. It is possible for such a condition to occur in a circuit which is structurally correct. These faults are modelled as either gate delay faults (GDF) or path delay faults (PDF). A GDF manifests as a slow 0 to 1 or 1 to 0 transition at a gate [15]. The PDFs on the other hand manifest when the propagation of a 0 to 1 or 1 to 0 transition from a primary input to a primary output is slow [16].

2.4 Testing of Faults in Digital Circuits

Fabricated copies of a circuit or at least samples of them are tested to nd out whether there are some faults present. This is done by applying test signals to

2.4. TESTING OF FAULTS IN DIGITAL CIRCUITS

the inputs of a fabricated copy, also called, circuit under test (CUT).The results are then captured at the outputs and analysed.

According to [2] Due to its continued popularity for test development, a wide range of fault simulators use the single SAF model . This is because; the model is able to detect many other faults apart from stuck-at faults. This is also supported by [8]. However, in many cases, a combination of fault models is used.

2.4.1 Testing of Faults: Combinational versus Sequential Circuits

Testing of combinational circuits is easy. The reason is that, all the inputs and outputs can be accessed. It is therefore easy to apply test vectors to them and to observe the response at the output. The response can then be compared to the truth table of the circuit to determine if there are errors.

In sequential circuits however, apart from inputs and outputs that may be accessed directly, there is the existence of memory elements. These elements can cause the circuit to be in several states. Therefore, the response observed at the output depends not only on the tests applied, but on the state of the circuit as well. It would have been easier if the response could be compared with the functionality speci ed in the state table. However, it is very di cult to know the state in which the circuit is, since the state variables are not observable externally. This problem associated with sequential circuits is solved by designing them such that they can be easily tested. This is achieved by using Design for Testability Techniques.

2.4.2 How the Automated Test Equipment (ATE) works

An automated (or automatic) test equipment (ATE) basically consists of a tester and a handler. The tester performs the actual testing on the device under test (DUT) itself. The handler on the other hand transfers the DUT to the test site and positions it properly for testing to be done. It also takes it o the test site after testing is done.

The tester is usually controlled by the test program, written in a high level programming language. The test program consists of several blocks of test, directed towards a certain parameter of the DUT. It controls the DUT xtures during the testing process. It also determines the electrical excitation the tester needs to apply and the correct time to apply them.

During the testing process, the ATE sends the test vectors to the DUT in the form of electrical excitation. The response is then compared to the expected response. The expected response is usually within a range of lower and upper limits. A DUT with a response outside this range is deemed to have failed the
test [1].

2.5 Design for Testability (DFT)

Design for Testability or Design for Test (DFT) refers to techniques of making changes to a given circuit design to help decrease its overall di culty of testing. The changes made may involve some addition of features to the circuit or the modi cation of it. Built-in Self Test (BIST) and Scan Techniques are two of the widely used Design for Testability Methods.

2.5.1 Built-in Self-test (BIST)

Built-in self-test (BIST) is a technique that makes a circuit capable of testing itself. A generalised BIST architecture is shown in Figure 2.5. It is made up of a pattern generator and a response compressor. The pattern generator generates test vectors, (x_{n-1},..., x_2, x_1, x_0) and applies them to the circuit under

test (CUT). The response from the CUT, (p_{n-1},..., p_2, p_1, p_0) is then compressed into a single pattern by the response compressor. This compressed pattern is referred to as the signature. The signature of the CUT is compared with that of
a fault-free circuit to determine whether a fault exists or not.

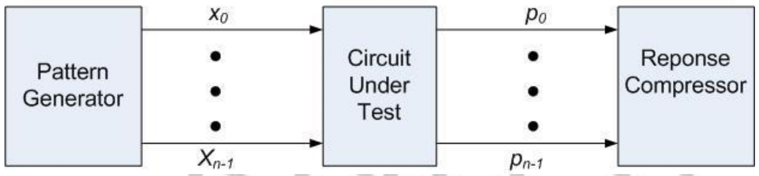


Figure 2.5: A Generalised BIST Architecture

2.5.1.1 Linear Feedback Shift Register (LFSR)

A linear feedback shift register (LFSR) is made up of a shift register with added linear feedback. It is often used as the pattern generator in the BIST architecture in Figure 2.5. An n-stage LFSR is capable of generating a sequence of $2^n - 1$ pseudorandom patterns, containing the vectors needed to be applied to the CUT. Figure 2.6 shows an example of an LFSR implemented with a 4-bit shift register. In this example, the feedback is connected to the rst stage by means of an XOR gate. The binary sequence, (x_3, x_2, x_1, x_0) form the PRBS.

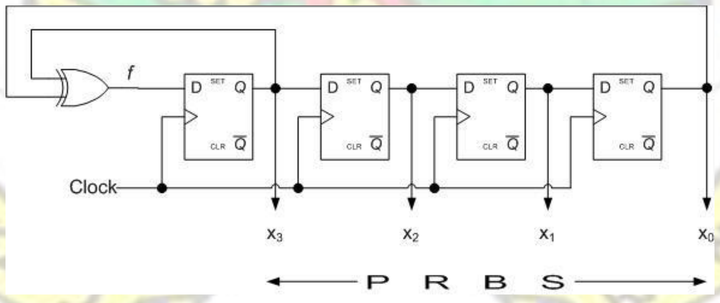


Figure 2.6: A 4-Bit Linear Feedback Shift Register (LFSR)

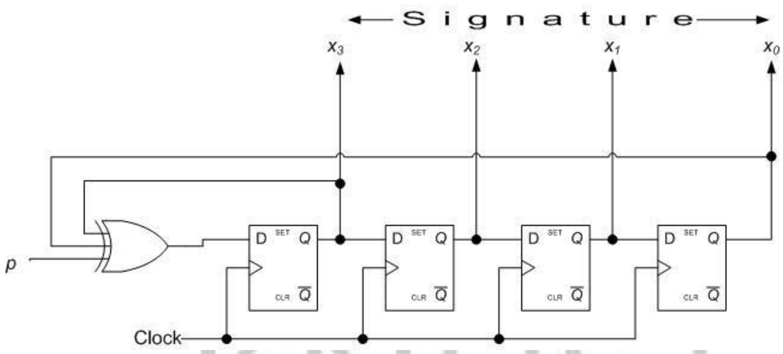


Figure 2.7: A 4-Bit Single-input Compressor Circuit (SIC)

2.5.1.2 Multiple-input Signature Register (MISR)

The LSFR can be used to implement single-input compressor circuits (SIC). This can be achieved by connecting the output of the CUT through line *p*, in addition to the feedback signals as shown in Figure 2.7. The sequence, (x_3, x_2, x_1, x_0) form the signature. This idea is extended to implement a multiple-input signature register (MISR), by combining the response from the CUT with the feedback of the LFSR, through XOR gates. Figure 2.8 shows an example of a 4-Bit MISR with four outputs. The output from the CUT is represented by the binary sequence,
(p_3, p_2, p_1, p_0), whilst that of the signature is represented by (x_3, x_2, x_1, x_0). Figure 2.9 also shows the same circuit with serial output. The MISR is used as
the response compressor of the BIST architecture in 2.5.

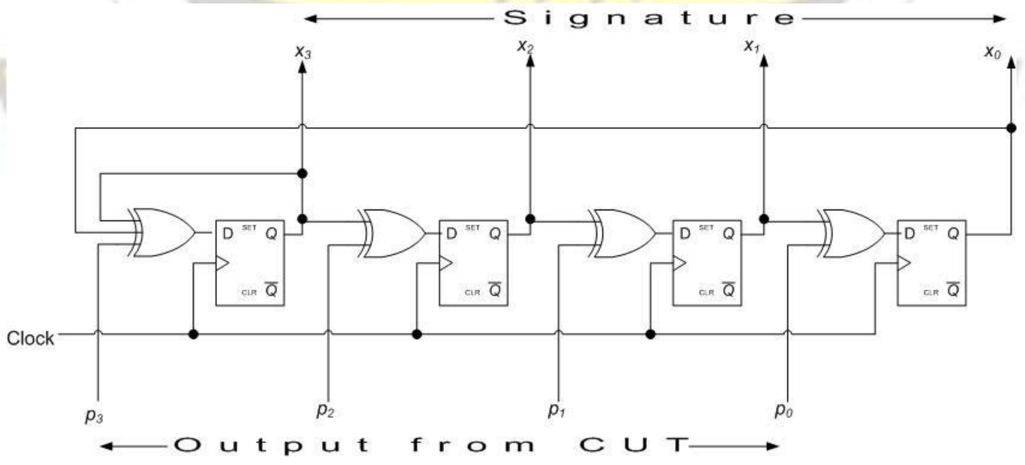


Figure 2.8: A Multiple-output 4-Bit MISR

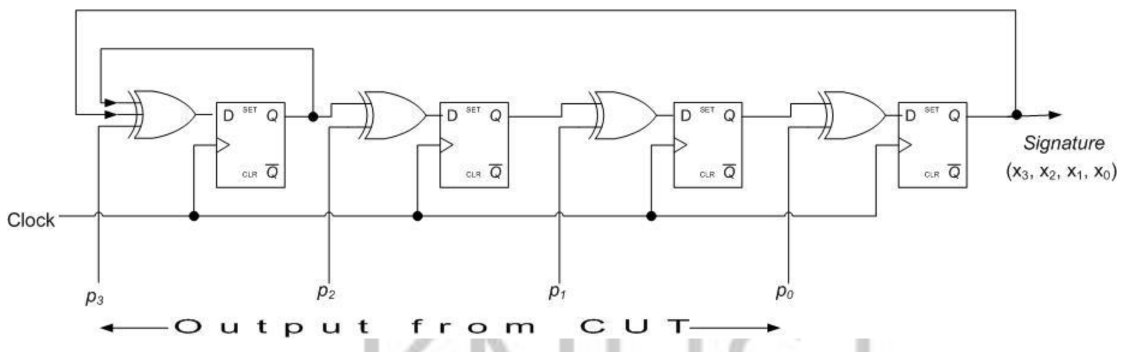


Figure 2.9: A Serial-output 4-Bit MISR

2.5.2 Scan Design

A diagram of a general model of a sequential circuit is given in Figure 2.10. It consists of a block of combinational circuits and a set of ip- ops. The circuit has x_1,x_2, ...x_n as the primary inputs and z_1,z_2, ...z_n as the primary outputs. The state input and output variables are y_1,y_2, ...y_k and Y_1,Y_2, ...Y_k respectively.

The inputs and outputs of ip- op FF_l are Y_l and y_l respectively.

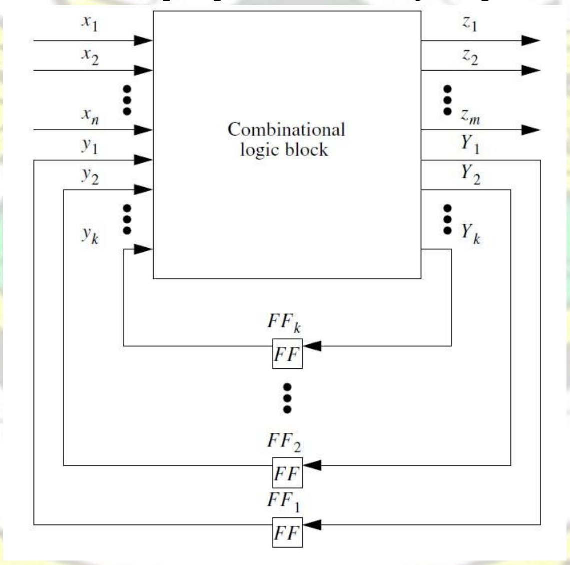


Figure 2.10: A General Model of a Sequential Circuit (Jha and Gupta,©200

In a normal circuit, accessing the state inputs and outputs are di cult. Scan elements are therefore introduced to allow for easy accessibility. This can be realised with multiplexed-input scan ip- ops or level-sensitive scan design.

2.5.2.1 Multiplexed-input scan ip- ops

One approach to the implementation of the scan technique is by connecting a two-way multiplexer to the input of each ip- op and connecting them as a shift register through a serial path, as shown in Figure 2.11(a). With this approach, a value that forms part of a test vector and a value at the state output can be multiplexed to the input of the combinational network, through the ip- ops, as
shown in Figures 2.11(b) and (c).

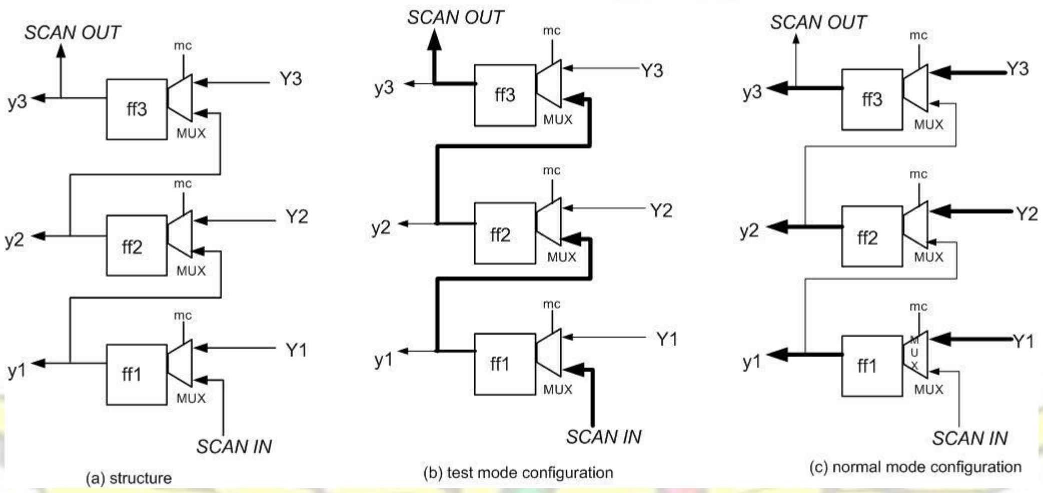


Figure 2.11: The scan method based on multiplexed input ip- ops

2.5.2.2 Level-sensitive scan design

The latches used in the level-sensitive scan design are generally con gured in pairs as shown in Figure 2.12.

D and *C* are the normal data input and clock lines respectively. Lines D_S, *A* and *B* also form the shift part of the latch. D_S is used for shifting in data, whilst *A* and *B* form the two phase non-overlapping shift clocks. Line Q is the data output. In the normal mode, clock A is held low whilst non-overlapping clocks C and B are used. In that mode, the value on line *D* is input to the ip- op. In the test mode, non-overlapping clocks *A* and *B* are used, and data is scanned in through D_S whilst clock *C* is kept low.

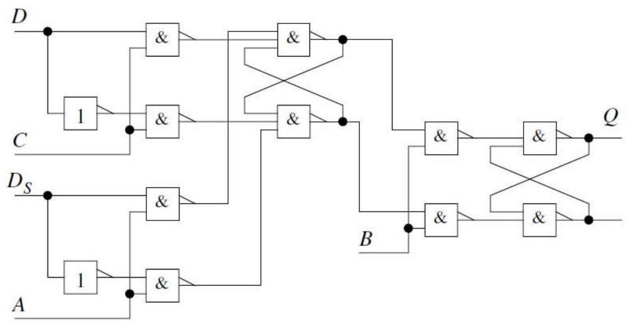


Figure 2.12: A Level Sensitive scan element for a double-latch design (Eichelberger and Williams, 1977)(© 1977 IEEE)

2.5.2.3 Test Application using Scan

In the scan methodology, the circuit operates either in the normal or test mode. During the normal mode, the inputs to the ip- ops are driven by the next state variables, Y_1,Y_2, ...Y_k. In other words, it functionally operates as the circuit was designed to operate. When being used in test application, the circuit is made to operate in the test mode. In that mode, test vectors, forming the present state variables are scanned into the ip- ops through the scan-in pin, as shown in Figure 2.11(b). The circuit is then switched to normal operation and the results captured into the ip- ops, also shown in Figure 2.11(c). The circuit is then switched back into test mode for the results captured in the ip- ops to be scanned out through the scan-out pin.

2.5.2.4 Bene ts of the Scan Technique

One main bene t of adding scan circuitry in designs is that, they make sequential circuits easily testable. Scan increases the controllability and observability of circuits. As a result, acceptable fault coverage is achieved.

Aside test generation and application, scan techniques can be used to nd out design errors and weaknesses when debugging the rst batch of chips fabricated for a new design. This rst batch of chips is called rst silicon.

In an operational environment, the use of scan helps in locating failing components. As a result, the time required to repair such devices is reduced. This bene t of scan ensures system availability.

2.5.2.5 Costs Associated with the Scan Technique

The scan technique solves the problem of controllability and observability di culties. It therefore makes sequential circuits easily testable. This however, comes with a few penalties. These penalties include area overhead, performance penalty, the need for extra pins and an increase in test application time.

Area Overhead

The use of scan requires additional logic and routing. This increases the area layout of the circuit, thereby increasing the manufacturing cost of the circuit. This increase in layout area is referred to as the area overhead. The manufacturing costs associated with area overhead come about in two main ways. The rst one is that, as the layout area increases, fewer copies of a chip can be manufactured on a wafer of semiconductor. The other one is that, as the area increases, there is a decrease in yield, as such the percentage of manufactured copies that work
decreases.

Performance Penalty

The additional circuitry needed to implement scan increases delay. The decrease in speed during normal operation is referred to as the performance penalty. It is signi cant for high speed circuits.

2.6. EXISTING APPROACHES TO REDUCING TEST APPLICATION TIME IN THE SCAN TECHNIQUE

The Need for Extra Pins

Introducing scan circuitry comes with it the need to introduce extra pins for signals. The scan-in and scan-out signals are examples of those that require extra pins. Too many extra pins is not desirable.

Increase in Test Application Time

The introduction of scan DFT techniques increases test application time. The reason is that several clock cycles are required to apply all the test vectors by scan. Consider a set of *n* test vector. Assuming that these test vectors are needed to test a sequential circuit with *m* state inputs, it will take $n(m+1)$ clock cycles to scan in all the tests and apply them. It will take additional m-1 clock cycles to scan out. The total number of clock cycles required to apply n test vectors to the circuit is therefore $n(m + 1) + m -$ 1, as seen in Equation 1.3. It can be deduced from the equation that, as the number of test vectors (*n*) and/or bits (*m*) in them increases, the number of clock cycles needed to scan in all of the bits also increases, thereby, increasing test application time correspondingly.

2.6 Existing Approaches to Reducing Test Application Time in the Scan Technique

Increase in test application time is a major cost associated with scan. Several techniques have been developed to deal with this problem. Examples of these techniques include the use of multiple scan chains [17, 18], re-ordering of scan chains [2] to obtain an optimal arrangement and Abramovici's Approach [19].

2.6. EXISTING APPROACHES TO REDUCING TEST APPLICATION TIME IN THE SCAN TECHNIQUE

2.6.1 Multiple Scan Chains

The use of multiple scan chains helps in reducing test application time. In this approach, the design of a scan-based circuit is done with multiple chains of ipops (latches), instead of a single chain. This enables portions of a test vector to be scanned into the ip- ops simultaneously, thus, reducing the time needed to
scan-in that test vector.

2.6.2 Re-ordering of scan chains

This approach enhances the use of multiple scan chains. It seeks to obtain an optimal arrangement of scan registers which can reduce test application for a particular circuit. Although this is computationally complex [20], it has been shown to be possible [21]. However, it must be noted that having an optimal order of registers depends on four main issues [2], namely, test application scheme, shift policy, the characteristics of the design and the number of test vectors required for each kernel.

2.6.3 Abramovici's Approach

This technique takes advantage of the fact that a group of faults may have the state input parts of their test vectors being similar. If such faults can be detected by only observing the primary outputs, then they can be tested by rst scanning in the desired state input values. The scan ip- ops are then put on hold for their contents to be held. Next, the primary input parts of each vector are applied and the response observed at the primary output. This process is repeated for the
other sets of test vectors.

Chapter 3

Architecture Modi cation and

Algorithm Development

One of the problems associated with the scan technique is the increase in test application time. This comes about because every bit in the test vectors is scanned in one clock cycle at a time. Many approaches to solving this problem have been proposed. This study presents a novel approach, called the Vector Match Approach, to solving the problem with minimal overhead. The new approach takes advantage of the possibility of having matching patterns in some vectors. If a vector, B, has a pattern that matches another pattern in a previously scanned in vector, A, then it may be possible to scan in only the un-matching part of, *B*. This helps cut down the number of clock cycles needed to scan in all the vectors. To illustrate this, consider the vectors *A* and *B*;

$$A=\begin{bmatrix}0\\1\\1\\0\\0\\1\\0\\1\end{bmatrix} \qquad B=\begin{bmatrix}1\\0\\0\\1\\0\\1\\1\\1\end{bmatrix}$$

If the vectors *A* and *B* are the state-input portions of some test vectors, it will take 8 clock cycles for *A* to be scanned in. It will take another 8 clock cycles for vector *B* also to be scanned in. In all, it will take 16 clock cycles for both to be

3.1. THE ARCHITECTURE

scanned in. In this example, the 3rd to the 8th elements of vector *A* match the rst six elements of *B*. Therefore, in the proposed solution, if *B* is to be scanned in after *A* only the last two elements of *B* need to be scanned in. It therefore takes 10 clock cycles for vectors *A* and *B* to be scanned in. See Equation 1.1.

From Equation 1.1, the total number of clock cycles required to apply *n* test vectors, each having *m* elements, to a circuit in the scan technique is; $C_{in} = n(m + 1)$. It can be deduced from the equation that, as the number of test
vectors (*n*) and/or bits (*m*) in them increases, the number of clock cycles also increases, thereby, increasing test application time correspondingly. If the number of clock cycles needed to scan in a vector can be reduced from *m* to $\tilde{m}$, the total number of clock cycles can be reduced correspondingly to $C_{in} = n(\tilde{m} + 1)$. That is what the Vector Match approach seeks to achieve.

An algorithm based on the Vector Match approach for minimising test application time in the scan technique, is proposed. This algorithm works by pre-processing the vectors to allow those with matching patterns to follow consecutively. The rearranged set of vectors can then be scanned in. The test vectors are represented as an *m* x *n* matrix.

3.1 The Architecture

In the scan architecture, the ip- ops are arranged such that test vectors are scanned into them during the test mode. During normal mode, the scanned in vectors, together with those from the primary inputs are applied to the circuit under test (CUT). The response is then captured in the ip- ops. In this process, the ip- ops that hold the scanned in vectors are the same ones that capture the response. Therefore, the scanned in vectors are cleared by the response from the CUT. This attribute of the scan architecture does not allow the re-use of patterns in a test vector that is already scanned in. As a result, the whole of the next test

vector must be scanned in during the test mode. Moreover, any reduction in scan in time requires the same reduction in scan out time. This also makes it di cult to develop algorithms for minimising test application time.

A new architecture, shown in Figure 3.1, is therefore being proposed. It consists of the circuit under test (CUT) with scan circuitry connected to its input. Its output is also connected with a multiple-input signature register (MISR) that has a serial output. The MISR is used to compress the response from the CUT. The signature is then scanned out at the end of the testing process through the serial output. It can be observed from the gure that only the MISR found

in the traditional BIST architecture is incorporated. The pseudorandom binary sequence generator (PRBSG) is not used. Therefore, instead of pseudorandom patterns, deterministic vectors, (in the case of this study, the test vectors from the rearranged matrix, $\tilde{A}_{(p)}$, de ned in Section 3.2), are scanned into the CUT.

The addition of the MISR in the proposed architecture enables the separation of the response vectors from the scanned in vectors. Whilst a response vector is captured by the MISR, the scanned in vector stays in the scan ip- ops to be reused. This arrangement makes the development of algorithms for minimising test application time easier. This is demonstrated in the development of the Vector Match Algorithm.

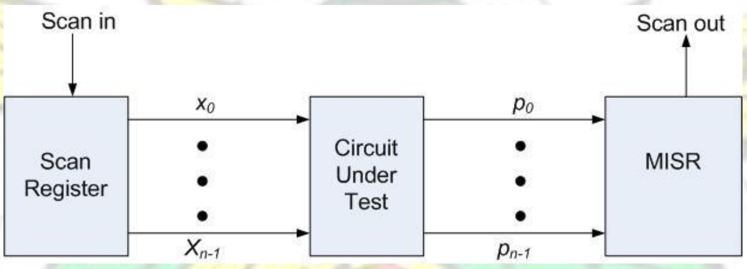


Figure 3.1: Architecture for the Vector Match Algorithm

3.2. TERMS AND VARIABLES USED IN THE VECTOR MATCH ALGORITHM

3.2 Terms and Variables Used in the Vector Match Algorithm

The algorithm depends on some terms and variables. These terms and variables are de ned below.

Terms

Initialization vector: The test pattern used to set the output of a logic gate to the complement of the logic value expected when SOpF or a delay fault is being tested. It is usually the rst pattern in a test pair.

Dependent vector[2]: The test pattern that follows an initialization vector in a test pair. It is actually the essential pattern in the test pair for testing an SOpF or a delay fault.

Independent vector[1]: For a particular fault, this vector is a complete test on its own and so does not need an initialization vector.

Pivot vector[1]: A vector that serves as the basis of comparison for other vectors. Starting

Pivot Vector[1]: The rst test vector to be used as a pivot vector.

Pivot Cycle[1]: This marks the beginning of usage of a pivot vector till when a new one is selected.

Row Shift[1], r: This is the number of elements of the pivot vector shifted before a matching pattern is found.

Skip[1], **S**: This signi es the number of elements in a vector that matched a pattern in the pivot vector. It sends a signal to the ATE, regarding the number of bits to ignore in a particular vector, when it is being scanned in.

It is calculated thus; $s = m - r$.

3.2. TERMS AND VARIABLES USED IN THE VECTOR MATCH ALGORITHM

Sum of Skips, ${}^{P}S$: See Section 3.3.

Variables

A: Original (Given) Matrix

$A_{(p)}$: Matrix with the vector at position p as the (prospective) starting pivot vector

[2] These terms were coined for the purposes of the algorithm.

$\tilde{A}_{(p)}$: The resultant matrix, or the optimum arrangement for minimising test application time when the vector at position p is used as the starting pivot vector k: The variable used to mark the position of the current pivot

vector

x: This variable shows the relative position of a vector to the position of a pivot vector. That vector may be the one being compared with the pivot vector or the one that matches it.

y: This variable represents the number of dependent vectors directly following an initialising vector. It is also the same as the number of initialising vectors directly following a pivot vector which is initialising, including the pivot vector itself.

Other De nitions

Let $A \equiv [T]$, an mxn matrix made up of binary elements $[t_{ij}]$ such that;

$[T] \equiv [T_j]$, where $1 \le j \le n$ and T_j is a column vector at the j_{th} column

$T_k \equiv [t_{ik}]$, where $1 \le i \le m$, for $j = k$

The test vectors are the column vectors within the matrix, ie $[T_j]$.

Let $T_k^{(r)} \subset T_k$, where r = number of rows (elements) that T_k has been shifted up.

3.3. THE SUM OF SKIPS, ^{P}S AND ITS SIGNIFICANCE

In other words,

$$T_k^{(r)} \subset T_k \text{ where } T_k \equiv [t_{ik}], \text{ and } 1 + r \le i \le m, \text{ for } j = k$$

Therefore, the full vector with no shift will be;

$$T_k \equiv T_k^{(0)}$$

Also,

$$\text{Let } T_k^{(-r)} \subset T_k \text{ where } T_k \equiv [t_{ik}], \text{ and } 1 \le i \le m - r, \text{ for } j = k$$

3.3 The Sum of Skips, ^{P}S and its Signi cance

The Sum of Skips is de ned as;

$$\sum S = \sum_{j=1}^{n} S_j$$

,where S_j is the *skip, S,* of the vector at the j^{th} position of $\tilde{A}_{(p)}$.

A resultant matrix, $\tilde{A}_{(p)}$, having the highest sum of skips signi es that the pivot vectors perform fewer comparisons before nding a match. As a result, there are more patterns that match. This therefore translates into savings in clock cycles when the test vectors are being scanned in. The reason is that the ATE bypasses (skips) more vectors during the scanning in process.[3] Therefore, the higher the sum of skips, ^{P}S, the more savings in clock cycles are achieved.

3.4 The Algorithm

The algorithm is made up of the Initialising Pivot procedure (Section 3.4.3) and the Pivot Cycle procedure (Section 3.4.4). It also has the Compare function, described in Section 3.4.1. All these modules are used in the Main procedure
described in Section 3.4.2.

[3] The skip value of a vector signi es the number of elements of that vector the ATE must bypass.

3.4. THE ALGORITHM

In the algorithm, the following are assumed;

- All initialising and dependent vectors are tagged as such.
- The vectors are arranged such that all dependent vectors are immediately preceded by their initialising vectors.

3.4.1 The Compare Function

The Compare function takes two arguments[3], $T_k^{(r)}$ and $T_{k+x}^{(-r)}$, and returns true if a match exists. Otherwise, a false is returned. A match exists if,

for all t_{ik} in $T_k^{(r)}$, where $1 + r \leq i \leq m$, and for all

t_{ik+x} in $T_{k+x}^{(-r)}$, where $1 \leq i \leq m - r$,

$t_{ik} = t_{ik+x}$ or t_{ik+x} is a don't-care.

3.4.2 The Main Procedure

The Main Procedure, as the name suggests, is the main algorithm that the Vector Match Approach is based on. Its owchart is shown in Figure 3.2.The following steps describe the ow of the algorithm.

1. START Main Procedure.
2. Set $p = 1$, where p is the position of a prospective starting pivot vector, T_p. The number of rows and columns are also m and n respectively.
3. Is $p \leq n$? If NO, select the resultant matrix, $\tilde{A}_{(p)}$ having the highest ${}^{P}S$. Where two or more exist, select $\tilde{A}_{(p)}$ with the lowest p. The selected $\tilde{A}_{(p)}$ is the optimum matrix for reducing test application time for the test vectors in matrix A, the original matrix. Go to step 10.

3.4. THE ALGORITHM

[3]The two arguments are $T^{(r)}_{k+x+y-1}$ and $T^{(-r)}_{k+x+y}$ when the Compare function is called in an
Initialising Pivot procedure

4. Is T_p a dependent vector? If YES ^PS is not valid, so increment p and go to step 3.

5. Select vector T_p. Associate skip, $S = 0$ with T_p and set *pivotFound* = 1. Initialise variables as shown; $k = p$, $r = 0$, $x = 0$ and $y = 0$.

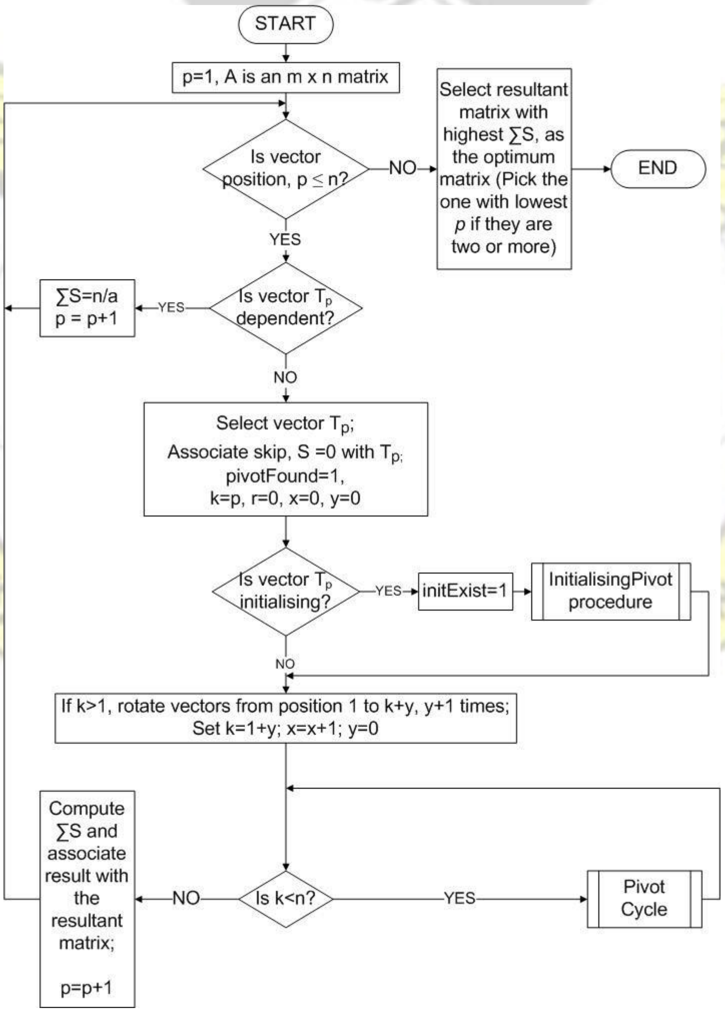
6. Is T_p an initialising vector? If YES set the variable *initExist* = 1 and perform the Initialising Pivot Procedure.

7. If $k > 1$, rotate vectors from position 1 to $k + y$, $y + 1$ times. Set $k = 1+y$. Increment x and set $y = 0$.

8. Is $k < n$? If YES perform the Pivot Cycle Procedure and repeat step 8.

9. Compute ^PS and associate result with the resultant matrix, $\tilde{A}_{(p)}$. Increment p and go to step 3.

10. END Main Procedure.


START
p=1, A is an m x n matrix
Is vector position, p ≤ n?
NO
Select resultant matrix with highest ∑S, as the optimum matrix (Pick the one with lowest p if they are two or more)
END
YES
∑S=n/a
p = p+1
YES
Is vector T_p dependent?
NO
Select vector T_p; Associate skip, S =0 with T_p; pivotFound=1, k=p, r=0, x=0, y=0
Is vector T_p initialising?
YES
initExist=1
InitialisingPivot procedure
NO
If k>1, rotate vectors from position 1 to k+y, y+1 times; Set k=1+y; x=x+1; y=0
Compute ∑S and associate result with the resultant matrix; p=p+1
NO
Is k<n?
YES
Pivot Cycle

3.4. THE ALGORITHM

Figure 3.2: The owchart of the Main Procedure

3.4.3 The Initialising Pivot Procedure

The Initialising Pivot procedure is found in the Main procedure and the Pivot Cycle procedure. It is called when the selected starting pivot vector is an initialising vector, and also, when the result of the Compare function executed using

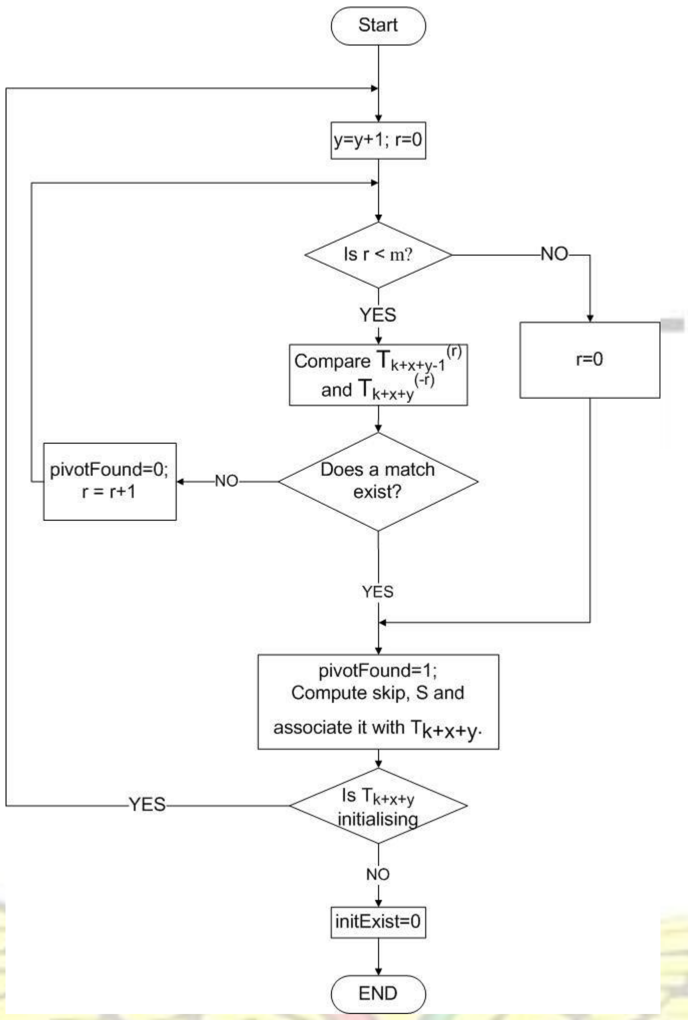
the two arguments[3] $T_k^{(r)}$ and $T_{k+x}^{(-r)}$ in the Pivot Cycle Procedure returns true and $T_{k+x}^{(0)}$ is found to be an initialising vector. When a vector is tested and found to be initialising, the variable *initExist* is set to 1. The owchart is shown in Figure 3.3. The following steps are executed.

1. START Initialising Pivot Procedure.
2. Increment *y* and set *r* = 0.
3. Is *r* < *m*? (In other words, if shifting has been done, are there still some elements in the initialising pivot vector?). If NO set *r* = 0 and go to step 6.
4. Perform the Compare function with $T_{k+x+y-1}^{(r)}$ and $T_{k+x+y}^{(-r)}$ as the arguments.
5. Does a match exist? If NO set *pivotFound* = 0, increment *r* and go to step 3.
6. Set *pivotFound* = 1. Compute skip, *S*, and associate with T_{k+x+y}.
7. Is T_{k+x+y} an initialising vector? If YES go to step 2.
8. Set *initExist* = 0.
9. END Initialising Pivot Procedure.

3.4.4 The Pivot Cycle Procedure

The Pivot Cycle procedure is found in the Main procedure. It is called when a non-initialising pivot vector is found. Its owchart is shown in Figure 3.4. The following steps are performed in the procedure.

3.4. THE ALGORITHM



Start
y=y+1; r=0
Is r < m?
NO
YES
Compare $T_{k+x+y-1}^{(r)}$ and $T_{k+x+y}^{(-r)}$
r=0
pivotFound=0; r = r+1
NO
Does a match exist?
YES
pivotFound=1; Compute skip, S and associate it with T_{k+x+y}.
YES
Is T_{k+x+y} initialising
NO
initExist=0
END

Figure 3.3: The Initialising Pivot Procedure

1. START Pivot Cycle Procedure.
2. Set $r = 0$ and $y = 0$.
3. Is $r < m$? If NO set $x = 1$ and $r = 0$. Then go to step 8.
4. Is $k+x > n$? (In other words, has the position of the vector being compared with the pivot vector exceeded the number of test vectors (n)?). If YES set x=1, increment r and go to step 3.
5. Is T_{k+x} (the vector being compared with the pivot vector) a dependent vector? If YES set *pivotFound* = 0, increment x and go to step 4.
6. Perform the Compare function with $T_k^{(r)}$ and $T_{k+x}^{(-r)}$ as the arguments.
7. Did the execution of the Compare function in step 6 return true? If NO set *pivotFound* = 0, increment x and go to step 4.
8. Set *pivotFound* = 1. Compute skip, S and associate it with T_{k+x}.

9. Is T_{k+x} an initialising vector? If NO go to step 11.

10. Set *initExist* = 1 and perform the Initialising Pivot Procedure.
11. Rotate vectors from *k*+1 to *k*+*x*+*y*, *y*+1 times (if *x* > 1). Set *k* = *k*+*y*+1, *x* = 1 and *y* = 0.
12. END Pivot Cycle Procedure.

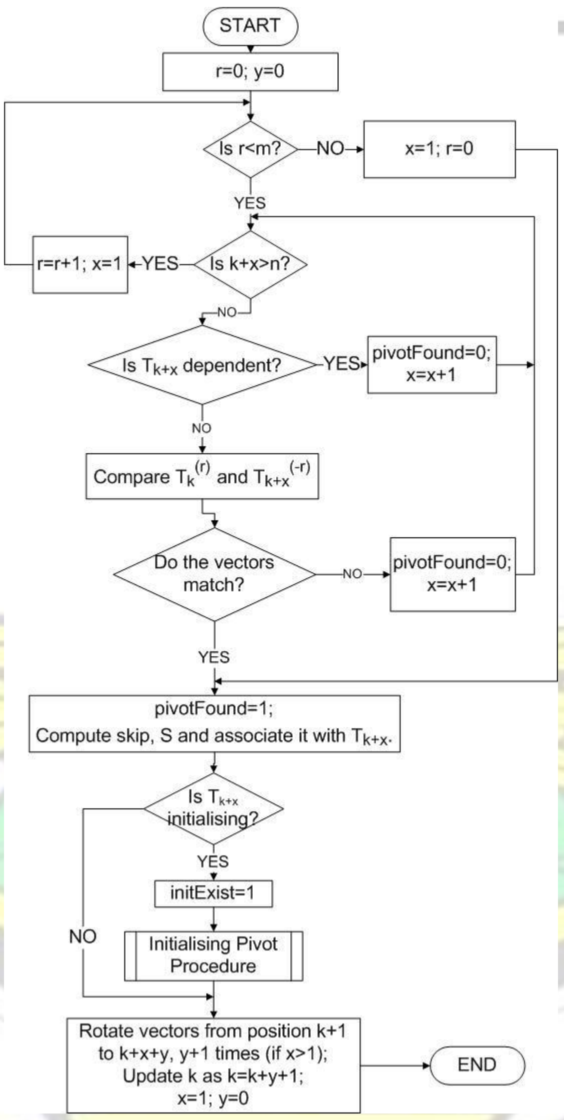

START
r=0; y=0
Is r<m?
NO
x=1; r=0
YES
Is k+x>n?
YES
r=r+1; x=1
NO
Is T_{k+x} dependent?
YES
pivotFound=0; x=x+1
NO
Compare $T_k^{(r)}$ and $T_{k+x}^{(-r)}$
Do the vectors match?
NO
pivotFound=0; x=x+1
YES
pivotFound=1; Compute skip, S and associate it with T_{k+x}.
Is T_{k+x} initialising?
YES
initExist=1
Initialising Pivot Procedure
NO
Rotate vectors from position k+1 to k+x+y, y+1 times (if x>1); Update k as k=k+y+1; x=1; y=0
END

Figure 3.4: The Pivot Cycle Procedure

3.5 The Test Application Process

When a test vector is about to be scanned in, the Automated Test Equipment (ATE) reads its skip value to determine the number of rows that must be skipped.

3.5. THE TEST APPLICATION PROCESS

A skip value of *s*, tells the ATE to skip the elements of the rst *s* rows of the vector and scan in the rest. Simultaneously, the binary data within the scan register is shifted up *s* times to accommodate the incoming binary data. The new test vector within the scan register is then applied to the CUT for the MISR to capture the response. After all the test vectors have been applied, the compressed response (signature) is scanned out of the MISR and compared with the signature of a
fault-free circuit.

Chapter 4

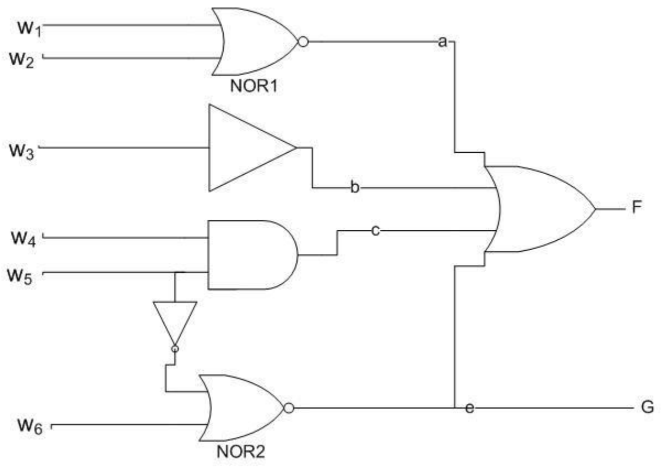
Sample Implementation and Testing

Consider the circuit in Figure 4.1(a). Let the following stuck-at faults, a/0, b/0, c/0, e/1 and F/1 be targeted. In addition, let the stuck-open faults (SOpFs) at Q_3, Q_4, Q_7 and Q_8 be also targeted. Using standard automatic test pattern generation (ATPG) algorithms, the test vectors in Table 4.1 can be obtained for the stuck-at faults. The test pairs in Table 4.2 can also be used to test for Stuck Open faults at Q_3 and Q_4 in Figure 4.1(b) and Q_7 and Q_8 in Figure 4.1(c) respectively. Putting all the vectors together, the matrix in Table 4.3 is one arrangement that can be used to test for the targeted faults. It is therefore our given matrix, *A*. The vectors labelled *i* and *d* are the initializing and dependent
vectors respectively.

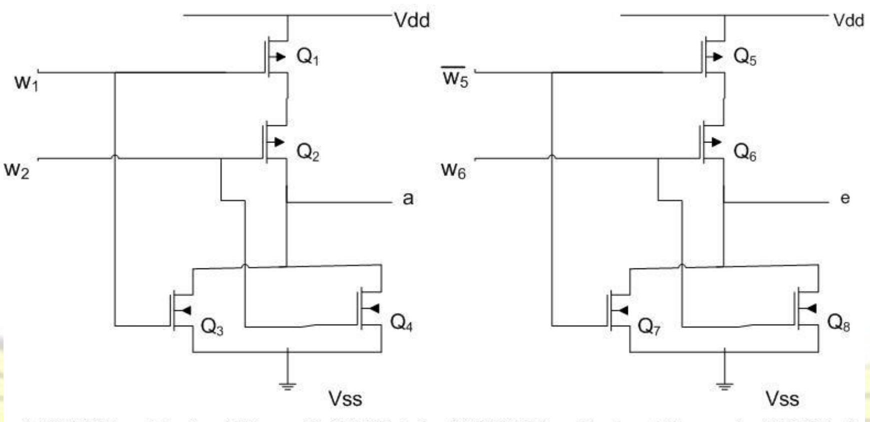
In this example, the number of rows (representing the number of elements in a test vector), *m* is 6. The number of columns (representing the number of test vectors), *n* is 12.

Table 4.1: Test Vectors for the Stuck-at Faults

Faults	a/0	b/0	c/o	e/1	F/1
Test Vectors	0	0	1	1	0
	0	1	0	1	1
	0	1	0	0	0
	1	0	1	1	0
	0	0	1	1	0
	1	1	0	1	0



(a) An Example Circuit



(b) CMOS Transistor Level Diagram for NOR Gate 1 (c) CMOS Transistor Level Diagram for NOR Gate 2

Figure 4.1: Example Circuits to Illustrate How the Algorithm Works

Executing the Main Procedure

The Main Procedure starts with the rst vector in the given matrix, A as a prospective starting pivot vector. Therefore, p = 1. The variable $k = p$ so k = 1. The variables x, y and r are also initialised as x = 0, y = 0 and r = 6. Since the vector T_1 is not a dependent vector, it is a valid starting pivot vector. A skip, S = 0 is therefore assigned to it, as shown in Table 4.4. The vector T_1 is

also not an initialising vector so the Initialising Pivot procedure is not executed. Moreover, the condition $x > 1$ is not true at this stage; therefore, no rotation is done. The variable k remains unchanged since y = 0. Variable x is incremented. At this stage, $k < 12$ (i.e. the number of vectors in the matrix) and vector T_k is Table 4.2: Vector Pairs for the Stuck-open Faults

Faults	SOpF Q3		SOpF Q4		SOpF Q7		SOpF Q8	
	0	0	1	0	1	1	1	1
	1	0	0	0	1	0	0	0
	1	0	0	0	0	0	1	0
Vector Pairs	0	1	0	1	1	0	0	0
	1	0	1	0	1	1	0	1
	0	0	0	0	1	0	0	0

Table 4.3: Test Vectors for all Targeted Faults, Matrix A

Labels		i	d/i	d				i	d		i	d
Skips												
	0	1	1	0	1	1	0	1	1	0	0	0
	1	0	0	0	1	0	1	1	0	0	1	0
	1	1	0	0	0	0	0	0	0	0	1	0
Vectors	0	0	0	1	1	1	0	1	0	1	0	1
	0	0	1	0	1	1	0	1	1	0	1	0
	1	0	0	0	1	0	0	1	0	1	0	0

a (starting) pivot vector so the Pivot Cycle procedure is initiated.

Executing the Pivot Cycle Procedure with T_1 as the Pivot Vector

In the Pivot Cycle procedure, the variables r and y remain (or are initialised as) $r = 0$ and $y = 0$. The condition $r < 6$ is therefore true. The position of the vector T_{k+x} is less than the number of vectors (i.e. the condition $k + x > n$ is false) and

T_{k+x} (T_2) is not a dependent vector so the Compare function is executed with $T_1^{(r)}$ and $T_2^{(-r)}$ as arguments, where $r = 0$. There is no match so *pivotFound* = 0 and x is incremented to 2. The execution of the Compare function, with $T_1$and T_3 as arguments also returns false. The process of incrementing x and executing the Compare function continues, ignoring all dependent vectors. When x is incremented and T_{k+x} is found to be a dependent vector, it is not used as an argument. The vectors, T_3, T_4, T_9 and T_{12} are therefore not used when encountered. In such cases, *pivotFound* = 0 and x is incremented. After T_{12}, Table 4.4: Matrix $A_{(1)}$

Labels		i	d/i	d				i	d		i	d
Skips	0											

Vectors	0	1	1	0	1	1	0	1	1	0	0	0
	1	0	0	0	1	0	1	1	0	0	1	0
	1	1	0	0	0	0	0	0	0	0	1	0
	0	0	0	1	1	1	0	1	0	1	0	1
	0	0	1	0	1	1	0	1	1	0	1	0
	1	0	0	0	1	0	0	1	0	1	0	0

there is no other vector to be compared so the condition $k + x > n$ is true. T_1 is therefore shifted up by one row, thereby incrementing r to 1, as shown on Table 4.5. The variable x is also set to 1. The Compare function is executed with the arguments $T_k^{(1)}$ and $T_{k+x}^{(-1)}$, where the latter argument takes values from T_2 to T_{12}. This also does not produce a match so r is incremented to 2 and x is reset to 1. With $r = 2$ and T_6 as the second argument, the Compare function returns true when $x = 5$ (see Table 4.6). The *pivotFound* variable is set to 1 and a skip, $S = 4$ (see Section 3.2) is assigned to T_6. Since T_6 is not an initialising vector, the Initialising Pivot procedure is not executed. Vectors T_2 up to T_6 are rotated once. This brings the vector T_6 to position 2, as shown in Table 4.7. Variable k is updated to $k = 2$. Variables x and y are also reset to $x = 1$ and $y = 0$. The Pivot Cycle procedure ends.

Table 4.5: Starting pivot row shifted up by 1

Labels		i	d/i	d				i	d		i	d
Skips	0											
Vectors	1	1	1	0	1	1	0	1	1	0	0	0
	1	0	0	0	1	0	1	1	0	0	1	0
	0	1	0	0	0	0	0	0	0	0	1	0
	0	0	0	1	1	1	0	1	0	1	0	1
	1	0	1	0	1	1	0	1	1	0	1	0
	■	0	0	0	1	0	0	1	0	1	0	0

Table 4.6: Starting pivot row shifted up by 2

Labels		i	d/i	d				i	d		i	d
Skips	0											
Vectors	1	1	1	0	1	1	0	1	1	0	0	0
	0	0	0	0	1	0	1	1	0	0	1	0
	0	1	0	0	0	0	0	0	0	0	1	0
	1	0	0	1	1	1	0	1	0	1	0	1
	■	0	1	0	1	1	0	1	1	0	1	0
	■	0	0	0	1	0	0	1	0	1	0	0

Table 4.7: The second pivot vector

Labels			i	d/i	d			i	d		i	d
Skips	0	4									4	
Vectors	0	1	1	1	0	1	1	1	1	0	0	0
	1	0	0	0	0	1	0	1	0	0	1	0
	1	0	1	0	0	0	0	0	0	0	1	0
	0	1	0	0	1	1	1	1	0	1	0	1
	0	1	0	1	0	1	1	1	1	0	1	0
	1	0	0	0	0	1	0	1	0	1	0	0

Back to the Main Procedure

Since $k < n$, the Pivot Cycle procedure is executed again with T_2 as the pivot vector.

Executing the Pivot Cycle Procedure with T_2 as the Pivot Vector

With $T_2^{(r)}$ and $T_{2+x}^{(-r)}$ as arguments for the Compare function, no match is found when $r = 0$ and $r = 1$. A match is however found at T_{11} when $r = 2$ and $x = 9$ (see Table 4.8). A skip, $S = 4$ is assigned to T_{11} and *pivotFound* is set to 1. Since T_{11} is an initialising vector, *initExist* is set to 1 and the Initialising Pivot procedure is initiated.

Table 4.8: A match found after shifting two rows up

Labels			i	d/i	d			i	d		i	d
Skips	0	4									4	
Vectors	0	0	1	1	0	1	0	1	1	0	0	0
	1	1	0	0	0	1	1	1	0	0	1	0
	1	1	1	0	0	0	0	0	0	0	1	0
	0	0	0	0	1	1	0	1	0	1	0	1
	0	■	0	1	0	1	0	1	1	0	1	0
	1	■	0	0	0	1	0	1	0	1	0	0

Executing the Initialising Pivot Procedure

Since *initExist* = 1, the Initialising Pivot procedure starts by incrementing y to 1 and setting r to 0. Since $r < m$, the Compare function is executed with $T_{k+x+y-1}^{(r)}$ and $T_{k+x+y}^{(-r)}$ (i.e $T_{11}^{(r)}$ and $T_{12}^{(-r)}$ where T_{12} is the dependent vector of

T_{11}) as arguments. No match is found so *pivotFound* = 0. Variable r is also incremented to 1, by shifting up the pivot vector by 1. The Compare function is executed again. This also does not produce a match. A match is however found when $r = 5$ (Table 4.9) so *pivotFound* is set to 1. The dependent vector, T_{12}, is

assigned a skip, $S = 1$. Since it is not an initialising vector, the Initialising Pivot procedure ends with *initExist* = 0.

Table 4.9: Comparing initialising and dependent vectors

Labels			i	d/i	d			i	d		i	d
Skips	0	4									4	1
Vectors	0	0	1	1	0	1	0	1	1	0	0	0
	1	1	0	0	0	1	1	1	0	0	■	0
	1	1	1	0	0	0	0	0	0	0	■	0
	0	0	0	0	1	1	0	1	0	1	■	1
	0	■	0	1	0	1	0	1	1	0	■	0
	1	■	0	0	0	1	0	1	0	1	■	0

Back to the Pivot Cycle Procedure

Vectors T_3 to T_{12} are rotated $y + 1$ (i.e. 2) times. This brings T_{11} to the third position, followed by T_{12} in the fourth position, as shown in Table 4.10. Variable k is updated to 4 whilst x and y are reset to $x = 1$ and $y = 0$. The Pivot Cycle

procedure ends.

Table 4.10: T_4 as pivot vector

Labels			i	d	i	d/i	d			i	d	
Skips	0	4	4	1								
Vectors	0	1	0	0	1	1	0	1	0	1	1	0
	1	0	1	0	0	0	0	1	1	1	0	0
	1	0	1	0	1	0	0	0	0	0	0	0
	0	1	0	1	0	0	1	1	0	1	0	1
	0	1	1	0	0	1	0	1	0	1	1	0
	1	0	0	0	0	0	0	1	0	1	0	1

Back to the Main Procedure

Since $k < n$, the Pivot Cycle procedure is executed again with T_4 as the pivot vector.

Executing the Pivot Cycle Procedure with T_4 as the Pivot Vector

Ignoring all dependent vectors, for example T_7, the Compare function with $T_4^{(r)}$ and $T_{4+x}^{(-r)}$ as arguments returns true at T_9 when r = 2 and x = 5 (see Table 4.11). The *pivotFound* variable is set to 1 and a skip, S = 4 is assigned to T_9. Vectors T_5 up to T_9 are rotated once, bringing T_9 to the fth position, as shown in Table
4.12. Variable k is updated as k = 5 whilst x and y are reset to x = 1 and y = 0.
The Pivot Cycle procedure ends.

Table 4.11: Comparison with T_4

Labels			i	d	i	d/i	d			i	d	
Skips	0	4	4	1					4			
Vectors	0	1	0	0	1	1	0	1	0	1	1	0
	1	1	1	1	0	0	0	1	1	1	0	0
	1	0	1	0	1	0	0	0	0	0	0	0
	0	1	0	0	0	0	1	1	0	1	0	1
	0	1	1	■	0	1	0	1	0	1	1	0
	1	0	0	■	0	0	0	1	0	1	0	1

Table 4.12: T_5 as pivot vector

Labels			i	d		i	d/i	d		i	d	
Skips	0	4	4	1	4							
Vectors	0	1	0	0	0	1	1	0	1	1	1	0
	1	1	1	0	1	0	0	0	1	1	0	0
	1	0	1	0	0	1	0	0	0	0	0	0
	0	1	0	1	0	0	0	1	1	1	0	1
	0	1	1	0	0	0	1	0	1	1	1	0
	1	0	0	0	0	0	0	0	1	1	0	1

Back to the Main Procedure

Since $k < n$, the Pivot Cycle procedure is executed again with T_5 as the pivot vector.

Executing the Pivot Cycle Procedure with T_5 as the Pivot Vector

Ignoring all dependent vectors, the Compare function with $T_5^{(r)}$ and $T_{5+x}^{(-r)}$ as arguments returns true at T_{12} when r = 3 and x = 7. The *pivotFound* variable is set to 1 and a skip, S = 3 is assigned to T_{12} (see Table 4.13). Since it is not an initialising vector, *initExist* = 0 and the Initialising Pivot procedure is not executed. Vectors T_6 to T_{12} are rotated once, bringing T_{12} to the sixth position (Table 4.14). Variable k is updated as k = 6 whilst x and y are reset to x = 1 and y = 0. The Pivot Cycle procedure ends.

Table 4.13: Comparison with T_5

Labels			i	d		i	d/i	d		i	d	
Skips	0	4	4	1	4							3
Vectors	0	1	0	0	0	1	1	0	1	1	1	0
	1	1	1	0	0	0	0	0	1	1	0	0
	1	0	1	0	0	1	0	0	0	0	0	0
	0	1	0	1	■	0	0	1	1	1	0	1
	0	1	1	0	■	0	1	0	1	1	1	0
	1	0	0	0	■	0	0	0	1	1	0	1

Table 4.14: T_6 as pivot vector

Labels			i	d			i	d/i	d		i	d
Skips	0	4	4	1	4	3						
Vectors	0	1	0	0	0	0	1	1	0	1	1	1
	1	1	1	0	1	0	0	0	0	1	1	0
	1	0	1	0	0	0	1	0	0	0	0	0
	0	1	0	1	0	1	0	0	1	1	1	0
	0	1	1	0	0	0	0	1	0	1	1	1
	1	0	0	0	0	1	0	0	0	1	1	0

Back to the Main Procedure

Variable $k < n$, so the Pivot Cycle procedure is executed again with the T_6 as the pivot vector.

Executing the Pivot Cycle Procedure with T_6 as the Pivot Vector

Ignoring all dependent vectors, the Compare function with $T_6^{(r)}$ and $T_{6+x}^{(-r)}$ as arguments returns true at T_7 when r = 3 and x = 1. The *pivotFound* variable is set to 1 and a skip, S = 3 is assigned to T_7 (see Table 4.15). Since it is an initialising vector, *initExist* = 1 and the Initialising Pivot procedure is executed.

Table 4.15: Comparison with T_6

Labels			i	d			i	d/i	d		i	d
Skips	0	4	4	1	4	3	3					
Vectors	0	1	0	0	0	1	1	1	0	1	1	1
	1	1	1	0	1	0	0	0	0	1	1	0
	1	0	1	0	0	1	1	0	0	0	0	0
	0	1	0	1	0	■	0	0	1	1	1	0
	0	1	1	0	0	■	0	1	0	1	1	1
	1	0	0	0	0	■	0	0	0	1	1	0

Executing the Initialising Pivot Procedure

The Initialising Pivot procedure starts by incrementing y to 1 and setting r to 0. Since $r < m$, the Compare function is executed with $T^{(r)}_{k+x+y-1}$ and $T^{(-r)}_{k+x+y}$ (i.e $T^{(r)}_{7}$ and $T^{(-r)}_{8}$ where T_8 is the dependent vector of T_7) as arguments. A match is found when r = 2. The dependent vector, T_8, is therefore assigned a skip, S = 4 (see Table 4.16). It is also found to be an initialising vector so y is incremented to 2 and r set to 0. The Compare function is executed again with $T^{(r)}_{8}$ and $T^{(-r)}_{9}$ as arguments. A match is found when r = 1. The dependent vector, T_9, is assigned a skip, S = 5 (see Table 4.17). Since vector T_9 is not an initialising vector, the Initialising Pivot procedure ends.

Table 4.16: Comparing vectors T_7 and T_8

Labels			i	d			i	d/i	d		i	d
Skips	0	4	4	1	4	3	3	4				
Vectors	0	1	0	0	0	0	1	1	0	1	1	1
	1	1	1	0	1	0	0	0	0	1	1	0
	1	0	1	0	0	0	0	0	0	0	0	0
	0	1	0	1	0	1	0	0	1	1	1	0
	0	1	1	0	0	0	■	1	0	1	1	1
	1	0	0	0	0	1	■	0	0	1	1	0

Table 4.17: Comparing vectors T_8 and T_9

Labels			i	d			i	d/i	d		i	d

Skips	0	4	4	1	4	3	3	4	5			
Vectors	0	1	0	0	0	0	1	0	0	1	1	1
	1	1	1	0	1	0	0	0	0	1	1	0
	1	0	1	0	0	0	1	0	0	0	0	0
	0	1	0	1	0	1	0	1	1	1	1	0
	0	1	1	0	0	0	0	0	0	1	1	1
	1	0	0	0	0	1	0	■	0	1	1	0

Back to the Pivot Cycle Procedure

Since the condition $x > 1$ is not true, no rotation is done. T_9 becomes the new pivot vector and k is updated to 9 (i.e. $k = k+y+1$). The Pivot Cycle procedure ends with $x = 1$ and $y = 0$.

Back to the Main Procedure

Variable $k < n$, so the Pivot Cycle procedure is executed again with T_9 as the pivot vector.

Executing the Pivot Cycle Procedure with T_9 as the Pivot Vector

With $T_9^{(r)}$ and $T_{9+x}^{(-r)}$ as arguments for the Compare function, no match is found for $r = 0$ up to $r = 5$ (i.e. shifting up the pivot vector 5 times), ignoring dependent vectors. Shifting up the pivot vector again by 1 exhausts all of its elements and $r = 6$ (see Table 4.18). Since the condition $r < m$ is false, x and r are set to 1 and 0, respectively. The *pivotFound* variable is also set to 1 and a skip, $S = 0$, is assigned to T_{k+x} (T_{10}). Furthermore, since $x = 1$, the condition $x > 1$ is not true, so no rotation is done. Variable k is updated to $k = 10$ ($k = k + y + 1$). Variables x and y are reset to $x = 1$ and $y = 0$, if not already so. The Pivot

Cycle procedure ends.

Table 4.18: T_9 as pivot vector

Labels			i	d			i	d/i	d		i	d
Skips	0	4	4	1	4	3	3	4	[redacted]	0		
Vectors	0	1	0	0	0	0	1	1	[redacted]	1	1	1
	1	1	1	0	1	0	0	0	[redacted]	1	1	0
	1	0	1	0	0	0	1	0	[redacted]	0	0	0
	0	1	0	1	0	1	0	0	[redacted]	1	1	0
	0	1	1	0	0	0	0	1	[redacted]	1	1	1
	1	0	0	0	0	1	0	0	[redacted]	1	1	0

Back to the Main Procedure

Variable $k < n$ so the Pivot Cycle procedure is executed again with T_{10} as the pivot vector.

Executing the Pivot Cycle Procedure with T_{10} as the Pivot Vector

The Compare function with $T_{10}^{(r)}$ and $T_{10+x}^{(-r)}$ as arguments returns true at T_{11} when $r = 0$ (no shifting is done) and $x = 1$. T_{11} is therefore assigned a skip, $S = 6$ (see Table 4.19). Since T_{11} is an initialising vector, *initExist* = 1 and the Initialising Pivot procedure is executed.

Table 4.19: A match found at T_{11} without shifting

Labels			i	d			i	d/i	d		i	d
Skips	0	4	4	1	4	3	3	4	5	0	6	
Vectors	0	1	0	0	0	0	1	1	0	1	1	1
	1	1	1	0	1	0	0	0	0	1	1	0
	1	0	1	0	0	0	1	0	0	0	0	0
	0	1	0	1	0	1	0	0	1	1	1	0
	0	1	1	0	0	0	0	1	0	1	1	1
	1	0	0	0	0	1	0	0	0	1	1	0

Executing the Initialising Pivot Procedure

Since *initExist* = 1, *y* is incremented to 1 and *r* is set to 0. With $T_{11}^{(r)}$ and $T_{12}^{(-r)}$ as arguments to the Compare function, a match is found when *r* = 5. The dependent vector, T_{12}, is therefore assigned a skip, *S* = 1 (see Table 4.20). T_{12} is not an initialising vector so the Initialising Pivot procedure ends.

Table 4.20: T_{11} as pivot vector

Labels			i	d			i	d/i	d		i	d
Skips	0	4	4	1	4	3	3	4	5	0	6	1
Vectors	0	1	0	0	0	0	1	1	0	1	1	1
	1	1	1	0	1	0	0	0	0	1	[redacted]	0
	1	0	1	0	0	0	1	0	0	0	[redacted]	0
	0	1	0	1	0	1	0	0	1	1	[redacted]	0
	0	1	1	0	0	0	0	1	0	1	[redacted]	1
	1	0	0	0	0	1	0	0	0	1	[redacted]	0

Back to the Pivot Cycle Procedure

Since the condition *x* > 1 is not true, no rotation is done. Variable *k* is updated to *k* = 12 (*k* = *k*+*y*+1). The Pivot Cycle procedure ends with *x* = 1 and *y* = 0.

Back to the Main Procedure

Since *k* = 12 the condition, *k* < *n* is false. The sum of all the skip values, ^{P}S, is therefore computed and assigned to the resultant matrix, $\tilde{A}_{(1)}$, in Table 4.21.

The Main Procedure continues by incrementing *p* to 2. In other words, T_2, of matrix *A* is selected as the next prospective starting pivot vector, with an assigned skip, *S* = 0, as shown in Table 4.22. Since T_2 is not a dependent vector it is a valid starting pivot vector. The execution of the Main procedure therefore continues till ^{P}S of $\tilde{A}_{(2)}$ is computed and assigned to it. The same goes for all the other prospective starting pivot vectors of matrix *A*, from T_3 to T_{12}. Note however,

4.1. RESULTS OBTAINED

Table 4.21: The resultant matrix $\tilde{A}_{(1)}$

Labels			i	d			i	d/i	d		i	d
Skips	0	4	4	1	4	3	3	4	5	0	6	1
Vectors	0	1	0	0	0	0	1	1	0	1	1	1
	1	1	1	0	1	0	0	0	0	1	1	0
	1	0	1	0	0	0	1	0	0	0	0	0
	0	1	0	1	0	1	0	0	1	1	1	0
	0	1	1	0	0	0	0	1	0	1	1	1
	1	0	0	0	0	1	0	0	0	1	1	0

The sum of skips ${}^{P}S$ = 35 for matrix $\tilde{A}_{(1)}$.

that vectors T_3, T_4, T_9 and T_{12} are dependent vectors and so they are not valid starting pivot vectors. Their sums of skips, ${}^{P}S$ are therefore not valid.

Table 4.22: Selecting T_2 as prospective starting pivot vector for Matrix A

Labels		i	d/i	d				i	d		i	d
Skips		0										
Vectors	0	1	1	0	1	1	0	1	1	0	0	0
	1	0	0	0	1	0	1	1	0	0	1	0
	1	1	0	0	0	0	0	0	0	0	1	0
	0	0	0	1	1	1	0	1	0	1	0	1
	0	0	1	0	1	1	0	1	1	0	1	0
	1	0	0	0	1	0	0	1	0	1	0	0

4.1 Results Obtained

When the Vector Match Algorithm is applied on the example matrix, A, a summary of the results obtained is shown in Table 4.23. The resultant matrices, $\tilde{A}_{(1)}$ and $\tilde{A}_{(5)}$, have the highest ${}^{P}S$ of 35.

Since $\tilde{A}_{(1)}$ has the least p, it is chosen as the optimum arrangement for minimising test application time. From Equations 1.1, 1.2 and 1.3, the pure scan technique without the Vector Match method takes 84 clock cycles to scan in all the vectors and additional 5 clock cycles to complete the scanning out process. In all, it takes

4.2. THE VECTOR MATCH APPROACH VERSUS OTHER APPROACHES

Table 4.23: The ${}^{P}S$ values of the various resultant matrices, $\tilde{A}_{(p)}$

$\tilde{A}(p)$	PS
$\tilde{A}(1)$	35

$\tilde{A}(2)$	34
$\tilde{A}(3)$	n/a
$\tilde{A}(4)$	n/a
$\tilde{A}(5)$	35
$\tilde{A}(6)$	27
$\tilde{A}(7)$	29
$\tilde{A}(8)$	29
$\tilde{A}(9)$	n/a
$\tilde{A}(10)$	30
$\tilde{A}(11)$	32
$\tilde{A}(12)$	n/a

a total of 89 clock cycles to complete the testing process. With the new method, it takes 49 $(84 -^{\mathrm{P}}S)$ clock cycles to scan in the test vectors and additional 5 clock cycles to scan out the test signature. This brings the whole testing process to 54 clock cycles. The new approach therefore gives better results compared to the pure scan technique.

4.2 The Vector Match Approach versus other Approaches

The Vector Match Approach can be used to enhance the e ectiveness of other existing approaches used in reducing test application time in the scan technique. In the case of using multiple scan chains, the Vector Match Approach may be applied to each of the scan chains simultaneously. By so doing, reduction obtained for each of the scan chains will contribute to the overall reduction in test application time. Similarly, this new approach may also be applied in the situation where scan chains are ordered. It may also be possible to combine this new approach and that of Abramovici. In the case of the latter, instead of scanning in

the whole of the state inputs, it may be possible for the Vector Match Approach to be applied to make it more e cient.

Chapter 5

Conclusion and Recommendations

The use of digital circuits has become pervasive in modern times. They can be found in many devices, ranging from common household appliances to critical systems such as those used in medicine and space exploration. It is therefore imperative that they perform their speci ed functions correctly. This is ensured by applying tests to them at the design stage. Manufactured copies of the circuit are also tested to isolate those that developed aws introduced by the manufacturing process. One method used for testing digital circuits is the Scan Technique.

5.1 Conclusion

In this work, the architecture of scan-based circuits and the modes of operation of the scan technique were studied. After reviewing the scan architecture, it was realised that it does not allow the re-use of patterns in already scanned-in test vectors. This is because the same ip- ops that hold the scanned-in vectors are the ones that capture the response during the test application process. As a result, an already scanned-in vector is cleared by the response. Consequently, every bit in all the test vectors must be scanned in. This makes test application time very long. To solve this problem, a new architecture was proposed.

The new architecture consists of the circuit under test (CUT) with scan circuitry connected to its input. The output is also connected to a multiple-input signature register (MISR) having a serial output. The MISR is used for capturing and compressing the response at the output of the circuit under test during the test application process. The use of additional circuitry (MISR) for capturing the

5.2. RECOMMENDATIONS

response allows the scanned-in vector to be held in the ip- ops. As a result, repeating patterns in that vector can be re-used, at the cost of an incurred modest addition to hardware overhead. An algorithm, called the Vector Match Algorithm (VMA), was developed to take advantage of repetitive patterns in test vectors by re-using them. By so doing, it avoids a complete scan-in of whole vectors. Time savings is achieved as a result.

The Algorithm works by rearranging test vectors to achieve the optimum arrangement for reducing test application time. In the rearrangement process, patterns in test vectors are compared with that of a vector T_k, called the pivot vector. If a vector T_j, is found to have the longest matching pattern, it is made to follow T_k directly. The vector T_j, now becomes the new pivot vector for the process to continue. The rearrangement process also appends skip values to the test vectors. The purpose of these skip values is to send signals to the automated test equipment, concerning the number of bits to ignore in a particular vector when it is being scanned in.

5.2 Recommendations

The Vector Match Approach may be used together with some of the already existing approaches. Therefore, I recommend research work exploring the possibility of reducing clock cycles further by combining the proposed approach with others.

Although this work provided a new perspective to minimising test application time in the scan technique, further work must be done to ascertain its usefulness as an alternative test application method. A suggestion is therefore being made, that a program based on the Vector Match Algorithm be written and run on test vectors of some benchmark circuits for a more rigorous evaluation. Layouts of benchmark circuits, based on the new scan architecture, should also be developed
for fabrication.

References

[1] S. Brown and Z. Vranesic, Fundamentals of Digital Logic with VHDL Design. McGraw-Hill, 2005.

[2] N. Jha and S. Gupta, Testing of Digital Systems. Cambridge University Press, 2003.

[3] M. Williams and J. Angell, Enhancing testability of large-scale integrated circuits via test points and additional logic, in IEEE Trans. on Computers, 1973.

[4] E. B. Eichelberger and T. W. Williams, A logic design structure for design for testability, in Proc. Design Automation Conference, 1977.

[5] R. Rajsuman, Iddq testing for cmos vlsi, Proceedings of the IEEE, vol. 88, April 2000.

[6] J. P. Hayes, Fault modeling, in IEEE Design & Test of Computers, vol. 2, 1985.

[7] J. Abraham and W. Fuchs, Fault and error models for vlsi, in Proceedings of the IEEE, 1986.

[8] J. H. Patel, Stuck-at fault: A fault model for the next millennium, in Proceedings of the Proceedings International Test Conference, 1998.

[9] Y. Malaiya and S. Su, A new fault model and testing technique for cmos devices, in Proc. Int. Test Conference, 1982.

[10] S. Reddy, M. Reddy, and V. Agrawal, Robust tests for stuck-open faults in cmos combinational circuits, in Proc. Int. Symposium Fault-Tolerant Computing, 1984.
References

[11] Y. Malaiya, Testing stuck-on faults in cmos integrated circuits, in Proc. Int. Conference on Computer-Aided Design, 1984.

[12] J. M. Soden, R. K. Treece, M. R. Taylor, and C. F. Hawkins, Cmos ic stuckopen fault electrical e ects and design consideration, in International Test Conference, 1989.

[13] M. Abramovici and P. R. Menon, A practical approach to fault simulation and test generation for bridging faults, in IEEE Transactions on Computers, 1985.

[14] J. M. Acken and S. D. Millman, Accurate modelling and simulation of bridging faults, in Proceedings of the Custom Integrated Circuits Conference, 1991.

[15] E. Hsieh, R. Rasmussen, L. Vidunas, and W. Davis, Delay test generation, in Proc. Design Automation Conference, 1977.

[16] J. Lesser and J. Shedletsky, An experimental delay test generator for lsi logic, in IEEE Transactions on Computers, 1980.

[17] S. Narayanan and M. A. Breuer, Asynchronous multiple scan chains, in Proceedings of VLSI Test Symposium, 1995.

[18] S. Narayanan, R. Gupta, and M. A. Breuer, Optimal con guring of multiple scan chains, in IEEE Trans. on Computers, 1993.

[19] M. Abramovici, K. B. Rajan, and D. T. Miller, Freeze: A new approach for testing sequential circuits, in Proceedings of the Design Automation Con-
ference, 1992.

[20] S. Narayanan, C. Njinda, and M. A. Breuer, Optimal sequencing of scan registers, in Proceedings of International Test Conference, 1992.
References

[21] S. Narayanan, Scan Chaining and Test Scheduling in an Integrated Scan Design System. PhD thesis, University of Southern California, Los Angeles,
CA., 1994.