Doctor's Thesis

Studies on Hierarchical Two-Pattern Testability of Controller-Data Path Circuits

Md. Altaf-Ul-Amin

November 11, 2002

Department of Information Processing
Graduate School of Information Science
Nara Institute of Science and Technology

Doctor's Thesis
Submitted to Graduate School of Information Science
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
DOCTOR of ENGINEERING

Md. Altaf-Ul-Amin

Thesis Committee

Hideo Fujiwara, Professor
Katsumasa Watanabe, Professor
Michiko Inoue, Associate professor

Studies on Hierarchical Two-Pattern Testability of Controller-Data Path Circuits

Md. Altaf-Ul-Amin

Abstract

Two-pattern test is required to identify delay faults in a circuit. The importance of delay fault testing is increasing gradually because of the fact that traditional stuck-at fault testing is failing to guarantee an acceptable quality level for today's high-speed chips. Some defects and/or random process variation do not change the steady state behavior of a circuit but affect the at speed performance. Any degradation in at speed performance is detected by delay testing and it is likely to become industrially accepted in near future. A straightforward solution to two-pattern testability is the enhanced-scan design. But this incorporates very high area overhead and long test application time. In this thesis we present a hierarchical testability technique for delay faults. There are a number of delay fault models. Among these, the path delay fault model is more general and can overcome the limitations of other models. Our approach is developed on path delay fault model. The design hierarchy we consider is (Register Transfer Level) RTL, where the number of primitive elements in the circuit is greatly reduced. At RTL a circuit can be divided into two parts: a controller and a data path.

Firstly, we consider the data path as a separate entity. We introduce the concept of RTL paths in a data path. Based on this we develop the definition of hierarchically two-pattern testable (HTPT) data path. The advantages of an HTPT data path are (i) the data path can be tested using any delay fault model, (ii) combinational (Automatic Test Pattern Generation) ATPG can be used and (iii) the same fault coverage can be obtained as with the enhanced scan approach. We also point out some necessary and sufficient conditions to support the propagation of two-pattern vectors via two or more control paths in a data path.

Secondly, a design for testability (DFT) method is formulated that can be applied to augment a data path to an HTPT data path. We propose a special type of DFT element called rotating enhanced flip-flop (REFF) that consists of two latches and a multiplexer. An REFF can hold two-bit data for as many cycles as it is required. Other DFT elements we use are multiplexers and thru functions. Our DFT method incorporates a graph-based analysis of an HTPT data path and makes use of some graph algorithms such as breadth first search and finding shortest paths between nodes etc. We have conducted experiments by applying our method to several benchmark circuits and a RISC processor provided by a semiconductor industry. Through our experiments we have showed that our method can achieve similar advantages to the enhanced scan approach at a much lower hardware overhead cost and test application time.

Thirdly, we extend our approach for a complete circuit consists of both controller and the data path. We propose a DFT scheme for two-pattern testability of a controller-data path circuit. The scheme makes use of both scan and non-scan techniques. The data path is first transformed into an HTPT data path. Then an enhanced scan chain is inserted on the control lines and the status lines. The enhanced scan chain is extended via the state register of the controller. In some cases, the data path is further modified by adding test multiplexers. Then a test controller is designed and integrated to the circuit. In this scheme, we test multiplexer select lines and register load lines as RTL segments. To test these RTL segments we propose some techniques that are compatible to overall operation of a controller data path circuit. For a given circuit, the area overhead incurred by our scheme decreases reasonably with the increase in bit-width of the data path of the circuit. The proposed scheme substantially lowers the test application time, supports hierarchical test generation and can achieve fault coverage similar to that of the enhanced scan approach.

CONTENTS

List of Figures

List of Tables

CHAPTER 1

Introduction

Electronic equipment is now being used in every aspect of human life such as medical, military, space, communication, industry and commercial applications. The key components in all types of electronic equipment are integrated circuits (ICs). Extremely high reliability is required for some ICs because an IC failure may become life critical in applications like medical and aerospace. The expectation of zero failure may be met if all manufacturing defects are eliminated. In IC manufacturing line, different physical defects occur during the numerous physical, chemical and thermal processes. Obviously the occurrence of physical defects during manufacturing cannot be avoided. Therefore to separate good ICs from the bad ones, testing is performed after they are manufactured and before they are shipped out to the customer or to the assembly line. However, testing every physical defect is not practically feasible and some sort of fault modeling is needed. The most popular fault model is stuck-at fault model, which assumes that because of some physical defect some line/lines in the IC remains stuck-at logic '1' or logic '0'. However test based on stuck-at fault model is failing to guarantee an acceptable quality level for modern day high-speed high-density chips. As a result, some other test methodologies based on other fault models has emerged. Out of these the delay fault testing [1] and the IDDQ testing [2] are most well known and important. In this thesis we are dealing with delay fault testing.

Continuing advances in design techniques and fabrication process technology have made it possible to design and manufacture very high speed chips. These high-speed chips are designed under very tight timing constraints. For the proper operation of these high-speed circuits, the propagation delay of each sensitizable path in a circuit should be less than a specified limit (usually the clock interval). As the clock interval is gradually decreasing the need and significance for delay fault testing is gradually increasing. Like all other faults, delay faults are also originated from manufacturing defects. Now, for the sake of

testing propagation delays in a circuit, it is not feasible to test every manufacturing defect and some sort of fault modeling is required. Several fault models for delay testing are proposed such as transition delay, gate delay, line delay, path delay and segment delay fault models. Transition, gate and line delay models are used for representing delay defects lumped at a gate. Test generation for these models is simpler but they have some disadvantages. For example while testing a faulty gate its delay fault may be compensated by other faster gates. Again, a delay defect in a real circuit is usually not localized to a single gate but distributed over several gates. Therefore path and segment delay fault models are proposed to cope with these problems. Test generation algorithms for path delay faults are more complex.

In the context of a combinational circuit, a path is concatenation of signal lines and gates starting from a primary input to a primary output. Each path corresponds to two faults (rising and falling transition). The number of paths even in a circuit of reasonable size is very large and in the worst case can be exponential with the circuit size. The large number of paths presents a long fault list. The complexity of test generation and duration of test application time increases as the length of the fault list increases. However, some faults do not affect the circuit performance (functional unsensitizable faults) and need not to be tested. All the faults that can affect the circuit performance are commonly referred to as primitive faults and includes robust, non-robust and functional sensitizable faults. Of all primitive faults, only robust faults can be independently tested and their test generation is also easier. Test generation for non-robust and functional sensitizable faults is difficult and again they are not good quality tests as robust tests. Therefore, the path delay fault testability of the combinational circuits can be improved (i) By reducing the number of paths in the circuit and (ii) By increasing the percentage of robust paths in the circuit. Research has already been conducted targeting the above issues [3] [4] [5][6].

However, many practical circuits are sequential circuits and their path delay fault testing incorporates further difficulties. A sequential path is defined as the concatenation of several combinational paths in several consecutive time frames. The number of sequential paths in a circuit with feedback might be infinite. To make the number of paths finite, the

idea of path segments has been introduced. A path segment starts from the primary input or the output of a flip-flop and ends at the primary output or the input of a flip-flop. The idea of path segment allows considering the sequential circuits as combinational circuits and again a list of primitive faults can be prepared. The scenario of delay fault testing in a sequential circuit is as follows: First a sequence of input vectors is applied to initialize the circuit to some predetermined state. Then an input vector, which is supposed to activate the delay fault, is applied and the circuit is clocked at speed to latch the fault effect. Finally another sequence of input vectors is applied to propagate the fault effect to a primary output. Delay fault coverage of sequential circuits tested in this way has been shown to be very poor [7] [8] [9]. Even by inserting standard scan this problem cannot be properly handled. Because, instead of a single vector (unlike to stuck at fault) it is necessary to apply a vector pair to test a single path delay fault. The straightforward solution to this problem is the enhanced scan design. The flip-flops in enhanced scan design can store two bits instead of just one and thus facilitate the application of a vector pair. But the problem with enhanced scan design is very high area overhead and very long test application time. As an alternative to the enhanced scan approach, in this thesis we present a hierarchical testability technique for delay faults. In the following we discuss the overview of the rest of this thesis.

Chapter 2 contains general discussion on delay testing. Topics like existing delay testing schemes, delay fault models and the classification of path delay faults are discussed.

Chapter 3 contains an analysis on hierarchical two-pattern testability of a data path. We introduce the concept of RTL paths in a data path. Based on this we develop the definition of hierarchically two-pattern testable (HTPT) data path. We point out some necessary and sufficient conditions to support the propagation of two-pattern vectors via two or more control paths in a data path. We present these conditions as theorems together with proofs. A graph based analysis of the HTPT data path is also presented. Then a design for testability (DFT) method is discussed that can be applied to augment a data path to an HTPT data path. We propose a special type of DFT element called rotating enhanced flip-flop (REFF) that consists of two latches and a multiplexer. An

REFF can hold two-bit data for as many cycles as it is required. Other DFT elements we use are multiplexers and thru functions. Our DFT method makes use of some graph algorithms such as breadth first search and finding shortest paths between nodes etc. We have conducted experiments by applying our method to several benchmark circuits and a RISC processor provided by a semiconductor industry. Through our experiments we have showed that our method can achieve similar advantages to the enhanced scan approach at a much lower hardware overhead cost and test application time.

In Chapter 4, we propose a DFT scheme for two-pattern testability of a controller-data path circuit. The scheme makes use of both scan and non-scan techniques. The data path is first transformed into an HTPT data path. Then an enhanced scan chain is inserted on the control lines and the status lines. The enhanced scan chain is extended via the state register of the controller. In some cases, the data path is further modified by adding test multiplexers. Then a test controller is designed and integrated to the circuit. In this scheme, we test multiplexer select lines and register load lines as RTL segments. To test these control segments we propose some techniques that are compatible to overall operation of a controller data path circuit. For a given circuit, the area overhead incurred by our scheme decreases reasonably with the increase in bit-width of the data path of the circuit. The proposed scheme substantially lowers the test application time, supports hierarchical test generation and can achieve fault coverage similar to that of the enhanced scan approach.

Chapter 5 contains conclusive remarks and suggestions for future work.

CHAPTER 2

Two-Pattern Testing

2.1 Introduction

Two-pattern testing is required for post-fabrication timing verification of integrated circuits. Such testing is widely known as delay testing and is becoming increasingly important for high-speed circuits. In this chapter, we discuss existing delay testing schemes, delay fault models and classification of path delay faults.

2.2 Delay Testing Schemes

A test for a delay fault consists of two successive test patterns. The first pattern is called the initialization vector and the second pattern is called the propagation vector. To detect delay faults even if the test sequence is applied at low frequency, the circuit response must be captured at-speed after the application of each test pair. In the following, we discuss existing delay testing schemes for combinational and sequential circuits.

2.2.1 Combinational Circuits

Test application scheme for a combinational circuit is shown in Figure 2.1. In normal operation, only one clock is used to control the input and output latches and its period is T_c. In test mode, the input and output latches are controlled by two different clocks of same period (T_s). T_s is assumed to be greater than T_c. The first vector is applied at t_0 and the second vector is applied at t_1 where $t_1 - t_0 = T_s$. Time T_s is assumed to be long enough to stabilize the circuit signals under the first vector. The test response is captured at t_2 where $t_2 - t_1 = T_c$.

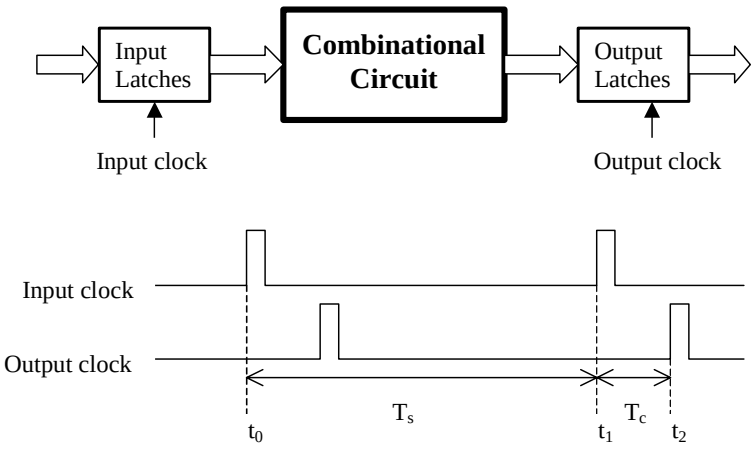


Figure 2.1 Delay testing scheme for combinational circuits

2.2.2 Sequential Circuits

Testing delay faults in a sequential circuit is more difficult than testing delay faults in a combinational circuits. This is because application of arbitrary vector pair is not possible to general or even standard scan sequential circuits. The enhanced scan design can guarantee the application of arbitrary vector pairs at the cost of very high area overhead and long test application time. Enhanced scan flip-flops allows the pre-scan of any two vectors and can apply them in consecutive clocks. The scenario of delay fault testing in a non-scan sequential circuit is as follows: First a sequence of input vectors is applied to initialize the circuit to some predetermined state. Then an input vector, which is supposed to activate the delay fault, is applied and the circuit is clocked at speed to latch the fault effect. Finally another sequence of input vectors is applied to propagate the fault effect to a primary output. Delay fault coverage of sequential circuits tested in this way has been shown to be very poor [7] [8] [9].

The coverage of delay faults can be somewhat improved in standard scan sequential circuits. Two time frame test generation technique is used to generate tests for delay faults for standard scan sequential circuits. In the first time frame all the primary inputs and the flip-flops are fully controllable but in the second time frame only the primary inputs are fully controllable. The second vector is obtained either by functional

justification [10] or by scan shifting [11]. In functional justification the second vector is obtained by using the normal mode of the test pin and it is just the next state values obtained by application of the first vector. In scan shifting, the second vector is produced by shifting the contents of the scan chain by one bit. Figure 2.2 illustrate the concept of functional justification and scan shifting concepts. The order of flip-flops in the scan chain cannot affect the fault coverage when functional justification is used. But this is not true for the case of scan shifting.

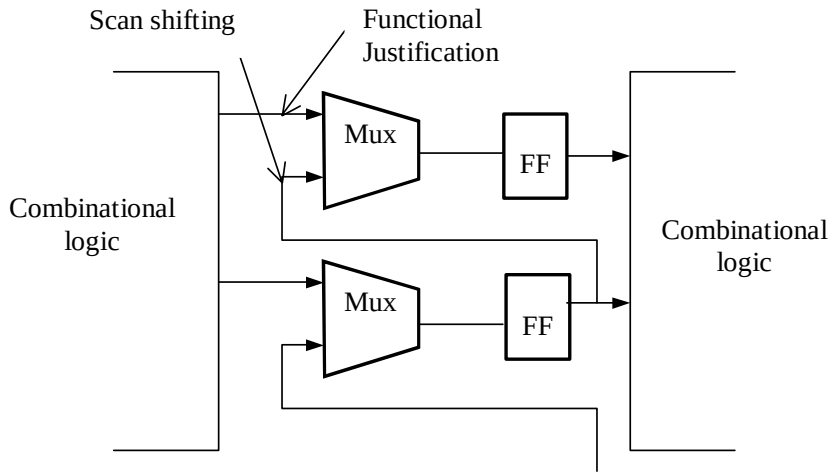


Figure 2.2: Delay testing schemes in standard scan design

2.3 Delay Fault Models

Like all other faults, delay faults are also originated from manufacturing defects. Now, for the sake of testing propagation delays in a circuit, it is not feasible to test every manufacturing defect and some sort of fault modeling is required. Several fault models for delay testing are proposed such as transition delay, gate delay, line delay, path delay and segment delay fault models. We discuss each model briefly in the following.

Transition delay fault model: This model [12] assumes that delay fault affect only one gate in the circuit. Each gate may have two faults: slow to rise and slow to fall i.e. the fault results in an increase of the nominal delay of the gate. Under the transition fault model, this increase in delay is large enough to prevent the transition from reaching any

primary output within specified time. However, this assumption is not realistic and a disadvantage of this fault model. The advantages of this model are (i) the number of faults in the circuit is linear in terms of the number of gates and (ii) the stuck-at fault test generation and fault simulation tools can be easily modified to handle transition faults.

Gate delay fault model: This fault model [13] [14] is based on the assumption that the delay fault is lumped at one gate in the circuit. Unlike the transition model, gate delay model does not assume that the fault affects the circuit performance independent of the propagation path through the fault site. It is assumed that only long paths through the circuit might cause performance degradation. The advantages and disadvantages of gate delay fault model are similar to those of the transition delay fault model.

Line delay fault Model: Line delay fault model [15] is a variation of gate delay fault model. This model tests rising or falling delay faults on a given line in the circuit. The fault is propagated through the longest sensitizable path passing through the line. The number of faults equals twice the number of lines in the circuit because each line is tested for rising and falling delay faults. Sensitizing the longest path through the target line allows the detection of the smallest delay fault on the target line.

Path delay fault model : This delay fault model is based on distributed delay in a circuit [16]. A path is concatenation of signal lines and gates starting from a primary input to a primary output. A path is considered faulty if its propagation delay exceeds a specified limit. Each path corresponds to two faults (rising and falling transition). The delay of a path represents sum of the delays of the gates and interconnects on that path and this is a realistic approach. However, the major limitation of this model is that the number of faults in a circuit can be very large.

Segment delay fault model: A segment delay fault model [17] is something in between the transition delay and path delay fault model. A segment is a partial path and this model assumes that the delay fault of a segment is large enough to cause a delay fault on all paths that include the segment. The number of segments of some given length is much lower than the number of paths in a circuit. This helps to reduce the number of faults while considering some sort of distributed delay, which is more realistic. The length of a segment can be decided based on available statistics about manufacturing defects.

2.4 Classification of Path Delay Faults

Path delay faults can be classified as robust, non-robust, functional sensitizable and functional unsensitizable based on sensitization criteria [1]. The robust, non-robust, functional sensitizable faults can affect circuit performance and they have to be tested. On the other hand functional unsensitizable faults can never independently affect the circuit performance and they do not have to be tested. Before explaining each of these fault types, we discuss several terminologies. An input to a gate is said to have a *controlling value* if it determines the value of the gate output regardless of the values on the other inputs of the gate. The complement of the controlling value is known as the *non-controlling value*. For example, for an OR gate, controlling value is 1 and non-controlling value is 0. An input to a gate is called *on-input* with respect to a path P if it is on P. An input to a gate is called *off-input* with respect to a path P if the gate is on P and the input is not an on-input.

2.4.1 Robust Testable Path Delay Faults

A path P is called robust testable if there exist at least one vector pair (v1, v2) such that
-(v1, v2) launches the appropriate transition at the beginning of P.
-At each gate along P, if the on-input transitions to a controlling value, the off-inputs of the gate remain stable at non-controlling values or if the on-input transitions to a non-controlling value, the off-inputs of the gate either remain stable at non-controlling values or transition to a non-controlling value.
Figure 2.3 shows these transition conditions for AND and OR gates. Figure 2.4 shows a robust testable path. The bold lines indicate the target paths.

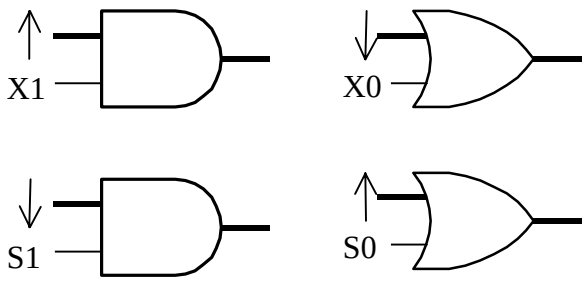


Figure 2.3: Robust transition conditions for AND and OR gates

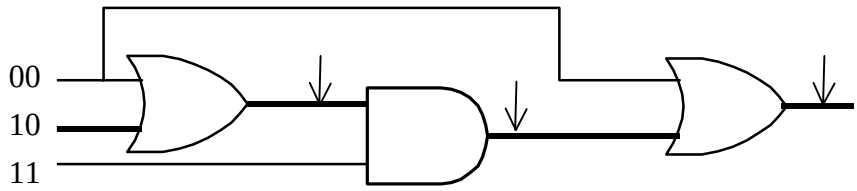


Figure 2.4: A Robust testable path

The robust testability condition ensures that if the path is faulty then it is guaranteed to fail independent of the delays in other paths of the circuit. Therefore, for testing purposes, robust tests are the highest quality tests for path delay faults and should be applied when they exist.

2.4.2 Non-Robust Testable Path Delay Faults

A robust untestable path P is non-robust testable if there exist at least one vector pair (v1, v2) such that,

-(v1, v2) launches the appropriate transition at the beginning of P.

-At least at one gate along P, if the on-input transitions to a controlling value, the off-inputs of the gate transition to a non-controlling value and the rest of the gates satisfy robust testability condition.

Figure 2.5 shows non-robust transition conditions for AND and OR gates. Figure 2.6 shows a non-robust testable path.

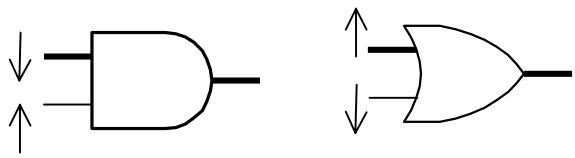


Figure 2.5: Non-robust transition conditions for AND and OR gates

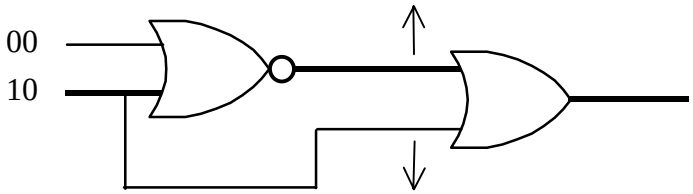


Figure 2.6: A Non-robust testable path

A non-robust test to detect a delay fault along a path may be invalidated by the presence of delay faults along one or more side paths.

2.4.3 Functional Sensitizable Path Delay Faults

A non-robust untestable path P is functional sensitizable if there exist at least one vector pair (v1, v2) such that

-(v1, v2) launches the appropriate transition at the beginning of P.

-At least at one gate along P, if the on-input transitions to a controlling value, the off-input/inputs of the gate also transitions to a controlling value and the rest of the gates satisfy either robust or non-robust testability condition.

Figure 2.7 shows the functional sensitization for AND and OR gate and Figure 2.8 shows a functional sensitizable path.

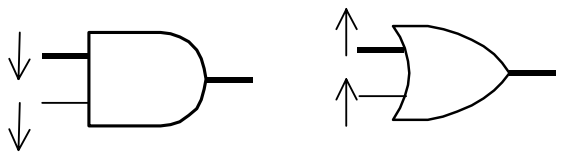


Figure 2.7: Functional sensitization conditions for AND and OR gates

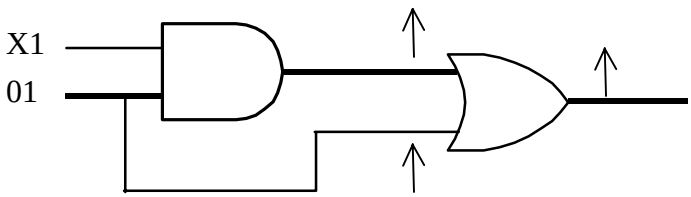


Figure 2.8: A functional sensitizable path

Functional sensitization criterion requires the presence of more than one faulty path in the circuit in order for the target fault to be detected.

2.4.4 Functional Unsensitizable Path Delay Faults

Functional unsensitizable path delay faults are defined as the faults for which under all possible vector pairs at least at one gate if the on-input transitions to a controlling value, the off-inputs of the gate remain stable at controlling values or if the on-input transitions to a non-controlling value, the off-inputs of the gate either remain stable at controlling values or transitions to a controlling value.

These transition conditions are shown in Figure 2.9 and Figure 2.10 shows a functional unsensitizable path. In figure 2.10 the transition on the target path is stopped at gate 2 because the off-input assumes a final controlling value while the on-input assumes a final non-controlling value.

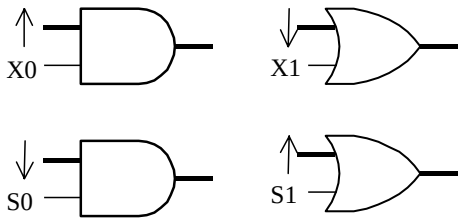


Figure 2.9: Functional unsensitization conditions for AND and OR gates

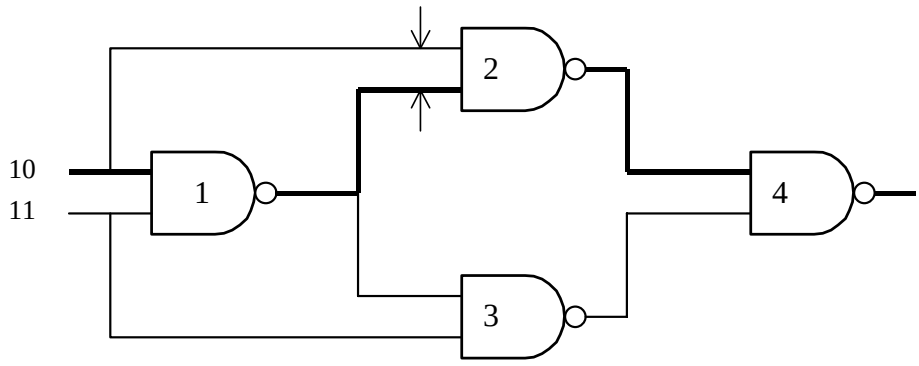


Figure 2.10: A functional unsensitizable path

Functional unsensitizable paths can never affect the circuit performance and need not to be tested.

CHAPTER 3

Design for Hierarchical Two-Pattern Testability of Data Paths

3.1 Introduction

In this chapter, we introduce the concept of hierarchical testability of data paths for delay faults. A definition of hierarchically two-pattern testable (HTPT) data path is developed. Also, a design for testability (DFT) method is presented to augment a data path to become an HTPT one. The DFT method incorporates a graph-based analysis of an HTPT data path and makes use of some graph algorithms. The proposed method can provide similar advantages to the enhanced scan approach at a much lower hardware overhead cost and test application time.

Two-pattern tests are necessary to detect delay defects in a circuit. The importance of detecting delay defects has soared in recent years to keep pace with the rapid increase in the speed of integrated circuits. A straightforward solution to two-pattern testability is the enhanced scan [18] [19]. But this incorporates very high area overhead and test application time. In the present work, we introduce hierarchical two-pattern testability of data paths. Hierarchical testability targeting stuck-at faults has been explored in a number of research works [20] [21]. These works address testability at register transfer level (RTL) and thus exploit the advantages of higher-level design hierarchy where the number of primitive elements in the design is greatly reduced. In these works it is shown that hierarchical testability is better than the gate-level based full-scan method in the context of test generation time, test application time and sometimes even area overhead. Our approach is hierarchical testability for delay faults. The design hierarchy we consider is RTL. At RTL a circuit can be divided into two parts: a controller and a data path. In this chapter, we consider the data path only. We assume that all control inputs and status outputs of a data path are directly controllable and directly observable, respectively.

There are a number of delay fault models. Among these, the path delay fault model is more general and can overcome the limitations of other models. We develop HTPT data path targeting all detectable path delay faults. However, in the present work, we do not consider paths involving control inputs. The advantages of an HTPT data path are as follows:

(i) The data path can be tested using any delay fault model,

(ii) Combinational ATPG can be used and

(iii) The same fault coverage can be obtained as with the enhanced scan approach.

Analyzing the functionality of a circuit, some paths in the data path might be proven to be multiple clock tolerant paths. A k-cycle tolerant path can affect the normal operation of the circuit if the delay along the path exceeds k cycles. Such a long delay is most likely to be caught by test for transition faults and need not be targeted for delay testing [22]. In this paper we developed our approach assuming that all paths of unity sequential depth are single clock tolerant. However, if some multiple clock tolerant paths exist and are discarded from the target fault list, our algorithm is still applicable and may result in less hardware overhead.

3. 2 The Concept of RTL Paths

A data path consists of hardware elements and buses. We assume that the buses in a data path are of the same bit width. However, we will relax this assumption afterwards. The RTL paths are the paths in an RTL circuit of a data path that, (i) start at a primary input (PI) and end at a register or (ii) start at a register and end at another/same (in case of feedback) register or (iii) start at a register and end at a primary output (PO) or (iv) start at a PI and end at a PO. It is obvious that the sequential depth of an RTL path is one. “PI1-R1” and “R1-ADD-MUX6-R4” are two examples of RTL paths in the arbitrary and simple data path of Figure 3.1. We exclude the paths that start at a constant register. The data path of Figure 3.1 has total eighteen RTL paths and they are listed in Table 3.1. In the lower hierarchy, each RTL path consists of a number of 1-bit wide paths. These individual 1-bit wide paths may be classified as robust, non-robust, functional sensitizable and functional unsensitizable paths. We discuss this classification in detail in

chapter 2. To guarantee the timing performance, it is necessary to test the robust, non-robust and functional sensitizable (FS) paths [6]. To test a robust path, it is

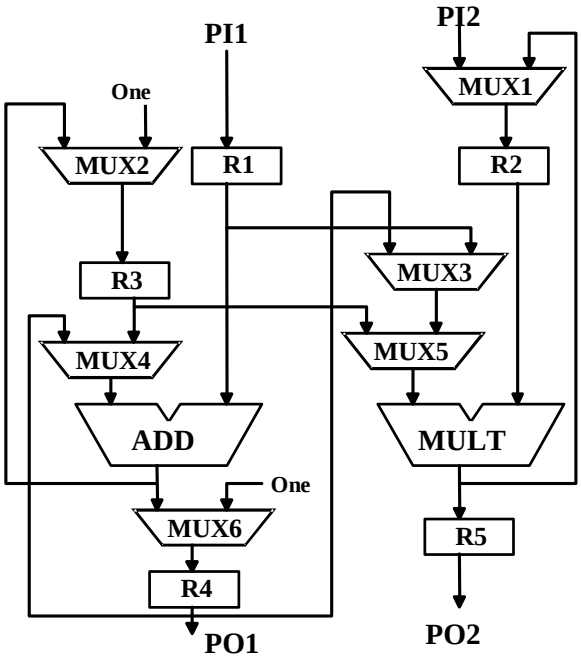


Figure 3.1: An arbitrary data path

Table 3.1: List of RTL paths in the data path of Figure 3.1

1	PI1-R1
2	R1-ADD-MUX2-R3
3	R1-ADD-MUX6-R4
4	R3-MUX4-ADD-MUX2-R3
5	R3-MUX4-ADD- MUX6-R4
6	R4-MUX4-ADD-MUX6-R4
7	R4-MUX4-ADD- MUX2-R3
8	PI2-MUX1-R2
9	R2-MULT-R5
10	R2-MULT-MUX1-R2
11	R1-MUX3-MUX5-MULT-R5
12	R1-MUX3-MUX5-MULT-MUX1-R2
13	R3-MUX5-MULT-R5
14	R3-MUX5-MULT- MUX1-R2
15	R4- MUX3-MUX5-MULT-R5
16	R4- MUX3-MUX5-MULT- MUX1-R2
17	R4-PO1
18	R5-PO2

possible to find two vectors (a vector pair) that differ from each other by only at single bit [7]. An FS path needs to be tested in a group simultaneously with one or more other FS paths.

Hence the testing of a group of FS paths requires two vectors, which may differ from each other at multiple bits [1]. From now on the testing of an RTL path will mean the testing of the robust, non-robust and functional sensitizable paths in it.

Path delay fault testing requires launching a transition at the start of a path by applying a pair of vectors, propagating the transition along the path and allowing fault effect observation from the end of the path [1]. However, many of the RTL paths in the data path neither start at a PI nor end at a PO. Therefore, some paths are necessary to ensure the flow of test data (test vectors and test responses) from PIs to appropriate registers and from appropriate registers to POs. Paths used for the flow of test vectors are referred to as *control paths* and paths used for test responses are referred to as *observation paths*. Any logic value can be propagated along the control paths and the observation paths. An RTL path may cross one or more multiplexers (MUXs) and operational modules. If an input of a MUX or an operational module is on an RTL path then this input is an *on-input*. Other input/inputs, which are not on the path, are called *off-inputs*.

3. 2.1 Paths Through a MUX

In a data path, MUXs are very common elements and are used as interconnecting units. Let us consider a 2 to 1 MUX as shown in Figure 3.2. Both A and B are n-bit wide. C is the control input. If C selects A then,

(i) propagation of signals from A(A_1,……A_n) to O(O_1,…….O_n) is robust (off-inputs remain stable at non-controlling value) and independent of the signals at B and

(ii) there is no merging gate among the paths (A_1 to O_1), (A_2 to O_2),……..(A_n to O_n) i.e. there are only n mutually independent (1-bit wide) paths from A to O.

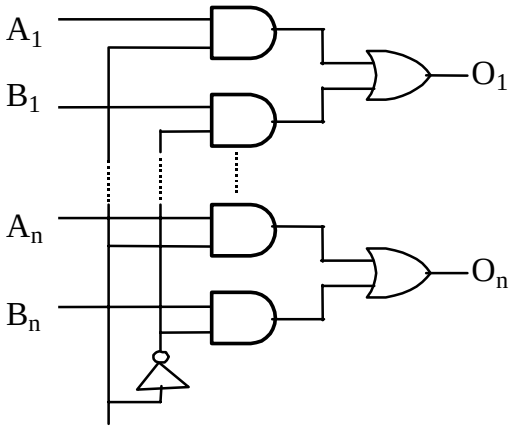


Figure 3.2: n-bit wide 2 to 1 MUX

The case for the paths from B to O is similar. Therefore, while testing any RTL path crossing one or more MUXs the select input/inputs should select the on-input/inputs of the MUX/MUXs and the off-input/inputs of the MUX/MUXs can be don't care. For example, to test the path "PI2-MUX1-R2" (Figure 3.1), two-pattern vectors should be applied at *PI2* and test responses should be captured at *R2*. The off-input of *MUX1* may be don't care.

3.2.2 Paths Through an Operational Module

Many RTL paths cross not only MUXs but also operational modules. In the following example, we discuss such a path.

Example 3.1: The path "R1-MUX3-MUX5-MULT-R5" in Figure 3.1 crosses the operational module *MULT*. The segment "R1-MUX3-MUX5-" of this path is like a wire in a sense that the signal values at *R1* appear unchanged at the output of the *MUX5* if the control inputs of the MUXs select the on-inputs. Again the segment "-R5" on the output side of *MULT* is obviously like a wire. The core segment of this path is the part of the path inside the *MULT*. In other words the test vector set required to test the part of the path inside the *MULT* is the same as to test the whole path "R1-MUX3-MUX5-MULT-R5". These test vectors can be generated by separately considering the gate level circuit

structure of the *MULT*. Obviously the bit width of these test vectors spans both inputs of the *MULT*. Suppose bits of the test vectors to be applied to the off-input of the *MULT* are not all don't cares. Hence to test the path "R1-MUX3-MUX5-MULT-R5", test vectors should be applied not only at *R1* but also at the off-input of the *MULT*. Test vectors can be applied at the off-input of the *MULT* from register *R2*. Therefore, we say that the RTL path " R1-MUX3-MUX5-MULT-R5" is an HTPT path if, (i) there exist two control paths from PI/PIs to *R1* and *R2* that support the application of two-pattern vectors and (ii) there exists an observation path to propagate the test responses from *R5* to a PO.

In Figure 3.1, we can see that disjoint control paths "PI1-R1" and "PI2-MUX1-R2" are sufficient to apply two-pattern vectors and the test response can be observed using the observation path "R5-PO2". It is noticeable that these control and observation paths can also test the RTL path "R2-MULT-R5".

Though most of the operational modules commonly used in data paths have two inputs, there might be cases of chaining as shown in Figure 3.3 (a). This type of module can be regarded as a 3-input operational module. Based on the similar reasons explained in *Example 1* it can be realized that three control paths and only one observation path would be sufficient to test any RTL path that crosses such a module. Some modules in a data path may have their inputs connected to constant registers. If x inputs of an m-input operational module (x<m) are directly connected to constant registers then we regard such a module as an (m-x)-input operational module. The module M of Figure 3.3(b) is considered as a 1-input module.

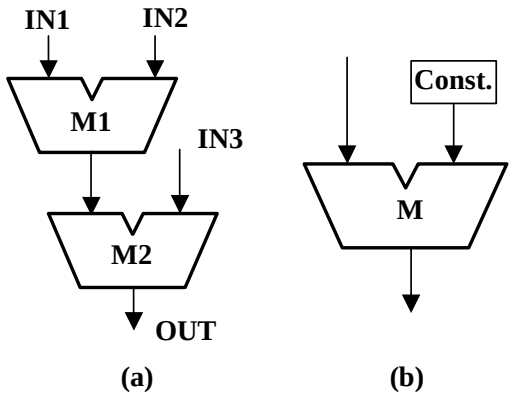


Figure 3.3: (a) Chaining of modules, (b) A module connected to a constant register

3.3 The HTPT Data Path

In this section we define the HTPT data path and some other related terms.

Difinition 3.1: The ***degree of an RTL path*** is n, if it crosses an n-input operational module.

For example, the path "R1-MUX3-MUX5-MULT-R5" of *Example 1* is an RTL path of degree 2. There might be paths in a data path that crosse no operational module. In Fig. 1, the path “PI2-MUX1-R2” crosses only a MUX and the path “PI1-R1” is simply a wire. These paths are considered as RTL paths of degree 1. The degree of an RTL path implies the number of control paths that are sufficient to ensure its hierarchical two-pattern testability.

Definition3.2: An RTL path of degree n, which crosses an n-input operational module say M, is an ***HTPT path*** if there exist n control paths C_1, C_2…C_n and an observation path P_1 such that, (i) C_1 is from a PI to the starting register of the path (in the case that it starts at a PI, C_1 is an empty path) and (ii) C_2….C_n is from PI/PIs to a register/registers which are either directly or through MUX/MUXs connected to the n-1 off-input/inputs of M (In the case that an off-input of M is connected to a PI, the corresponding control path is an empty path) and (iii) C_1, C_2…C_n support the application of two-pattern test and (iv) P_1 is from the ending register of the path to a PO (in the case that the path ends at a PO, P_1 is an empty path).

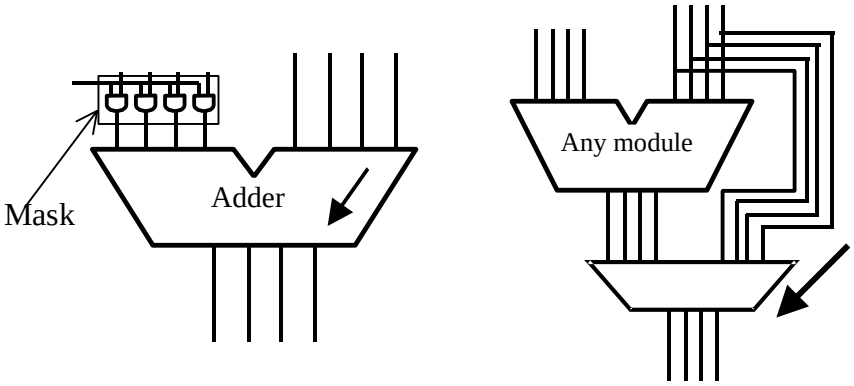
Definition 3: The set of control paths and an observation path that are sufficient to ensure the hierarchical two-pattern testability of an RTL path is referred to as the ***test plan*** of the path.

Definition 4: A data path is an ***HTPT data path*** if each of its RTL paths has a test plan.

3.4 Conditions for Control Paths to Support Two-Pattern Test

Here, we first briefly discuss the *thru* function [20]. The *thru* function allows the propagation of a logic value from an input to the output of an operational module without any change. For common two-input operational modules (e.g. adder, multiplier, etc.), a *thru* function from an input to the output can be realized by providing a constant at the other input. The necessary constant can be provided either by means of a support path or

by adding a mask. For other modules *thru* functions can be realized by a MUX. Figure 3.4 shows the realization of thru functions. Initially, we assume that a *thru* function exists



(a) By a mask element (b) By a multiplexer

Figure 3.4: Realization of thru function

from any input to the output of any operational module. Under this assumption, a control path can be represented by the lines and the registers it crosses. However, such an assumption is not necessary for an HTPT data path and will be relaxed later on.

The general condition: Let C_1, C_2…. C_n are n control paths starting at PI/PIs and ending at n different points EP_1, EP_2,……EP_n (these ending points may be either registers or PIs). Let, a vector pair (v_1, v_2) span over n control paths. Hence each of v_1 and v_2 can be divided into n partial vectors. Each of the partial vectors is associated with a control path. Based on this, we represent v_1 and v_2 as $v_1=v_{11}\&v_{12}\&……v_{1n}$ and $v_2 = v_{21}\&v_{22}\&……..v_{2n}$ (the symbol '&' merely links the partial vectors). Bit-width of each of v_{11}, v_{12},……v_{1n}, v_{21}, v_{22},……..v_{2n} is equal to the bit-width of the data path. Let v_{11}, v_{12},……v_{1n}, v_{21}, v_{22},……..v_{2n} can be fed from PI/PIs according to a schedule such that v_{11}, v_{12},……v_{1n} simultaneously appears at EP_1, EP_2,……EP_n and in the next clock cycle v_{21}, v_{22},……..v_{2n} simultaneously appear at EP_1, EP_2,……EP_n then we say that C_1, C_2…. C_n support the application of two-pattern test.

The sequential depth of a control path is the number of registers that appear on the path. To denote the sequential depth of any control path say, C_1 we use the notation $SD(C_1)$. In the following theorem we mention necessary and sufficient conditions for two control paths to support two-pattern test.

Theorem 3.1: Two control paths C_1 and C_2 from PI/PIs to two different points EP_1 and EP_2 support the application of two-pattern test if and only if one of the following conditions is satisfied:

Condition 1: $C_1 \cap C_2 = \phi$ i.e. C_1 and C_2 are disjoint.

Condition 2: $|SD(C_1') - SD(C_2')| \geq 2$, where C_1' and C_2' are disjoint parts of C_1 and C_2 from EP_1 and EP_2 to the nearest merging point of C_1 and C_2, respectively.

Condition 3: Either C_1'or C_2' crosses at least two hold registers.

Condition 4: When $|SD(C_1') - SD(C_2')| = 1$, the disjoint part (i.e. one of C_1'or C_2') with lower sequential depth crosses a hold register.

Proof: An arbitrary vector pair (v_1, v_2) involving two control paths can be represented as $v_1 = v_{11}\&v_{12}$ and $v_2 = v_{21}\&v_{22}$. In the following the sufficiency and the necessity of the above four conditions are discussed.

Sufficiency:

Condition 1: In Figure 3.5(a), m and n are two arbitrary numbers. Hence it represents a general topology that matches with condition 1. Here, v_{11} and v_{21} can be fed from PI_1 in consecutive clocks and similarly v_{12} and v_{22} can be fed from PI_2. The general condition can be met by differing the feeding from PI1 and PI2 by |m-n| clocks.

Condition 2: In Figure 3.5(b), m and n are two arbitrary numbers such that $|n-m| \geq 2$. Hence it represents a general topology that matches with condition 2. All partial vectors should enter C_1' and C_2' clock by clock through the merging point MP. Hence any other merging between C_1 and C_2 before MP has no effect. Let n>m and n-m = r. Therefore according to condition 2, $r \geq 2$. The general condition can be met by first allowing v_{11} and v_{21} to enter C_1' in consecutive clocks and then after r-2 clocks by allowing v_{12} and v_{22} to enter C_2' in consecutive clocks.

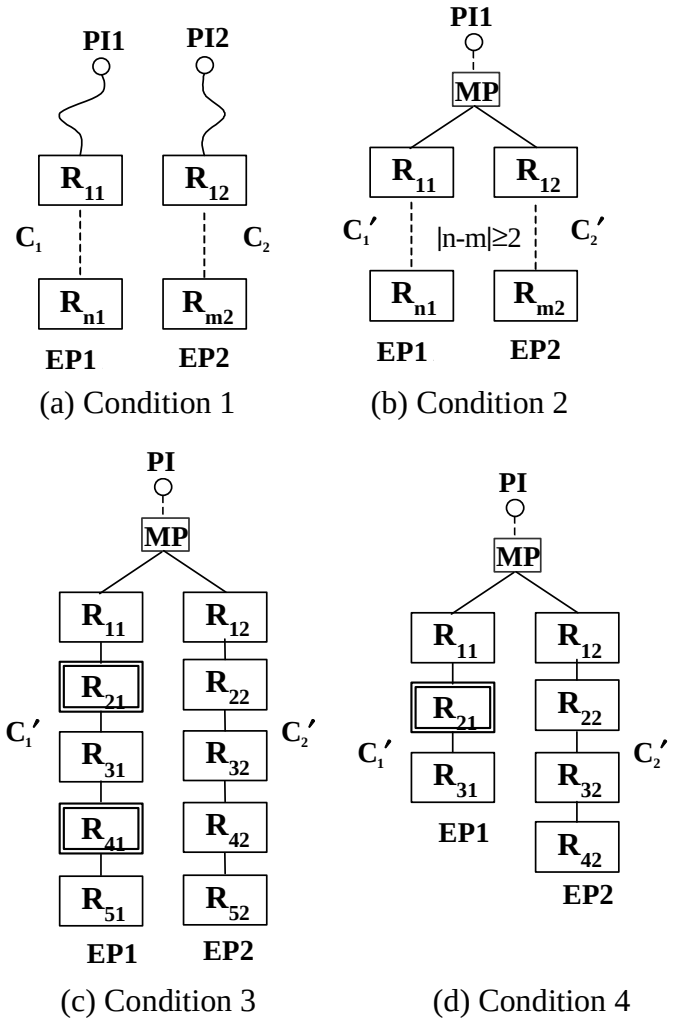


Figure 3.5: Control paths depicting the conditions of *Theorem 1*

Condition 3: Any of $C_1'$and C_2', say C_1' crosses two hold registers. First allowing v_{11} and v_{21} to enter $C_1'$and then allowing v_{12} and v_{22} to enter C_2' the general condition can be met. Here, v_{11} and v_{21} might be required to hold for one or more clock cycles in the hold registers of C_1'depending on the position of the hold registers and the number of registers in $C_1'$and C_2'. An example that matches with this condition is shown in Figure 3.5(c) and the corresponding scheduling of the application of the partial vectors is shown in Table 3.2. In Figure 3.5, registers with double line border are hold registers. Referring to Table 3.2, v_{11} and v_{21} are fed to C_1' in the 1st and 2nd clock and v_{12} and v_{22} are fed to C_2' in the 3rd and 4th clock. Notice that v_{11} and v_{12} simultaneously appear at EP1 and EP2 in the 7th

clock and v_{21} and v_{22} do the same in the following clock. The contents of the registers for certain clocks that are not important to this scheduling are not shown in the table.

Table 3.2: Schedule of partial vectors corresponding to Figure 3.5(c)

Clk	MP	C_1'					C_2'				
		R_{11}	R_{21} (H)	R_{31}	R_{41} (H)	R_{51}/ EP1	R_{12}	R_{22}	R_{32}	R_{42}	R_{52}/ EP2
0	v_{11}										
1	v_{21}	v_{11}									
2	v_{12}	v_{21}	v_{11}								
3	v_{22}		v_{21}	v_{11}			v_{12}				
4			v_{21}		v_{11}		v_{22}	v_{12}			
5			v_{21}		v_{11}			v_{22}	v_{12}		
6				v_{21}	v_{11}				v_{22}	v_{12}	
7					v_{21}	v_{11}				v_{22}	v_{12}
8						v_{21}					v_{22}

Condition 4: Any of $C_1'$and C_2' say C_1'crosses a hold register and so according to condition 4, $SD(C_2') - SD(C_1')=1$. Now, first allowing v_{11} to enter $C_1'$and then immediately allowing v_{12} and v_{22} to C_2' and then again allowing v_{21} to C_1' the general condition can be met. An example that matches with this condition is shown in Figure 5(d) and the corresponding scheduling of the application of the partial vectors is shown in Table 3.3. The explanation of Table 3.3 is similar to that of the Table 3.1.

Table 3.3: Schedule of partial vectors corresponding to Figure 3.5(d)

Clk	MP	C_1'			C_2'			
		R_{11}	R_{21} (H)	R_{31}/ EP1	R_{12}	R_{22}	R_{32}	R_{42}/ EP2
0	v_{11}							
1	v_{12}	v_{11}						
2	v_{22}		v_{11}		v_{12}			
3	v_{21}		v_{11}		v_{22}	v_{12}		
4		v_{21}	v_{11}			v_{22}	v_{12}	
5			v_{21}	v_{11}			v_{22}	v_{12}
6				v_{21}				v_{22}

Necessity:

Two control paths C_1 and C_2 that do not fulfill any of the above four conditions should satisfy all the following properties.

1. $C_1$and C_2 are not disjoint.
2. $|SD(C_1') - SD(C_2')|= 0$ or 1.
3. None of C_1'or C_2' crosses two hold registers.
4. When $|SD(C_1') - SD(C_2',)|=1$, the disjoint part (i.e. one of C_1'or C_2') with lower sequential depth does not cross a hold register.

All possible cases fulfilling the above properties are as follows.

1. C_1 and C_2 are not disjoint and $|SD(C_1') - SD(C_2')| =1$ or 0 but none of $C_1'$and C_2' crosses any hold register.
2. C_1 and C_2 are not disjoint and $|SD(C_1') - SD(C_2')| = 0$ and any one or both of $C_1'$and C_2' crosses a hold register.
3. C_1 and C_2 are not disjoint and $|SD(C_1') - SD(C_2')| =1$ and the disjoint part with higher sequential depth crosses a hold register.

However, no scheduling can obviously be found for these cases to fulfill the general condition. Hence condition 1, condition 2, condition 3 and condition 4 are the only conditions for two control paths to support the two-pattern test. □

In general case, any number, say n control paths can merge in numerous fashions. Each of these fashions may be of different nature depending on the number and position of hold registers in each control path. Hence to find out necessary and sufficient conditions for n control paths to support two-pattern test is a complex task. However in the following theorem some sufficient conditions are mentioned

Theorem 3.2: n control paths support the application of two-pattern test if one of the following conditions is satisfied:

1. All n paths are disjoint

2. Mutually disjoint parts of n-1 paths from their end points cross two hold registers.

3. The merging fashion of n paths after MP is a tree with MP as the root and the difference of sequential depths of any two partial paths from MP to their respective end points is two or more, where MP is the nearest merging point of all n paths from their end points.

Proof: The proof of this theorem is similar to that of *Theorem 3.1*. □

Here, one thing is noteworthy. Let an RTL path P start at a register R and crosses an operational module M. For P to be an HTPT path, a control path from a PI to R is necessary. Now, if R is a feedback register (FR) to M, the control path must not incorporate a *thru* function to M. In Figure 3.1, *R3* is an FR to *ADD*. Only one control path from a PI to *R3* is "PI1-R1-ADD-MUX2-R3". This control path cannot be used to test the path "R3-MUX4-ADD-MUX6-R4". It cannot be used because the first vector of a vector pair is loaded at *R3* to settle the signal lines throughout the path including the part of the path in *ADD*. In the next clock, the second vector is loaded at *R3* to launch the desired transition. However, the first vector cannot do its job if the second vector propagates using a *thru* function to *ADD*. Therefore, we should always carefully exclude such control paths.

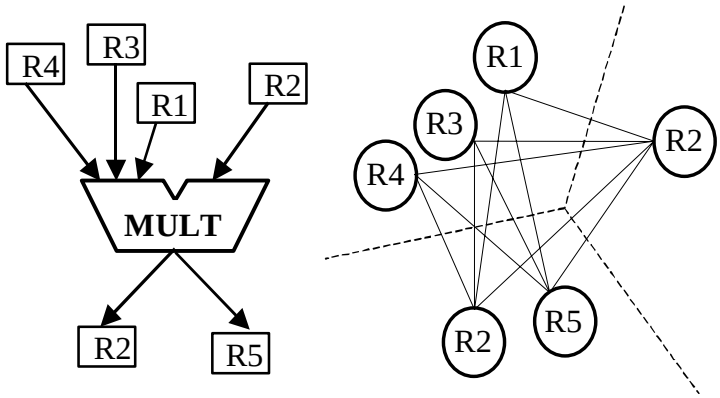
3.5 Graph Based Analysis of HTPT Data Paths

All RTL paths that cross an operational module having two or more inputs can be represented as a graph. We call this graph the *structural connectivity graph* (SCG) of the module. The nodes of the SCG consist of the module and the registers, PIs and POs that are at the starting and at the ending of the RTL paths through the module. The edges of the SCG represent the connections of the module with the other nodes. Let the set of nodes connected to each of the inputs of an n-input module be R_{i1}, $R_{i2,}$ .....R_{in} and the set of nodes connected to the output of the module is R_o. Let $R_{i1} = \{r_{11}, r_{21}, r_{31}\ldots..\}$, $R_{i2} = \{r_{12}, r_{22}, r_{32}\ldots.\}$,……. $R_{in} = \{r_{1n}, r_{2n}, r_{3n}\ldots..\}$ and $R_o=\{r_1, r_2, r_3\ldots.\}$. From the SCG of an n-input module an (n+1)-partite graph can be generated. We call this the *register compatibility graph* (RCG) of the module. R_{i1}, $R_{i2,}$ .....R_{in} and R_o are the n+1 set of nodes. The edges of the RCG are determined by the following rules.

Rule1: If there exist n control paths C_1, C_2…..C_n from PI/PIs to any node $r_{j1} \in R_{i1}$, $r_{k2} \in R_{i2,}$……..$r_{ln} \in R_{in}$, that support the application of two-pattern test, there exist edges between any two nodes of r_{j1}, r_{k2}, ……and r_{ln}.

Rule2: If there exists an observation path from any node $r_m \in R_o$ to a primary output, there exist edges between r_m to any node in R_{i1} and $R_{i2,}$ and.....R_{in}.

Example 3.2: The SCG of *MULT* of Figure 3.1 is shown in Figure 3.6(a). Figure 3.6(b) shows the RCG of *MULT* generated under the assumption that a *thru* function exists from any input to the output of any operational module. Table 3.4 shows that which edge/edges in Figure 3.6(b) are related to which control or observation path/paths in the arbitrary data path of Figure 3.1.



(a) (b)

Figure 3.6: (a) SCG and (b) RCG of *MULT* of Figure 3.1

Table 3.4: List of edges of the graph in Figure 5(b) and corresponding paths

R1-R2	PI1-R1 and PI2-MUX1-R2
R3-R2	PI1-R1-ADD-MUX2-R3 and PI2-MUX1-R2
R4-R2	PI1-R1-ADD-MUX6-R4 and PI2-MUX1-R2
R1-R5, R2-R5, R3-R5, R4-R5	R5-PO2
R1-R2, R2-R2, R3-R2, R4-R2	R2-MULT-R5-PO2

Definition 3.5: An edge in the RCG of a module from any node in R_o to any node in R_{i1}or $R_{i2,}$ …….or R_{in}, actually represents an RTL path through the module and hence we refer to such an edge as a ***path edge***.

Definition 3.6: An edge in the RCG between any two nodes of R_{i1},$R_{i2,}$ ……..and R_{in} is a ***control edge***.

Definition 3.7: If a sub-graph of the RCG of a module with only one node from each of R_{i1}, $R_{i2,}$ .....R_{in} and R_o is a clique, then this is referred to as a ***test clique***.

A test clique implies the existence of n control paths and one observation path, which are sufficient to test n RTL paths passing through n inputs of the module, i.e. a test clique in the RCG of an n-input module incorporates test plan for n paths.

Theorem 3.3: All RTL paths through a module are two-pattern testable if the path edges corresponding to all RTL paths through the module exist in its RCG and each path edge is part of some test clique.

Proof: If a path edge is part of a test clique, then this test clique is sufficient to guarantee the two-pattern testability of the corresponding path. Now if each path edge is part of some test clique, then each RTL path passing through the module is two-pattern testable. □

3.6.The DFT Elements

This section discusses the DFT elements we utilize in our approach. We consider three types of DFT elements. They are MUXs, *thru* functions and rotating enhanced flip-flops (REFFs). The operation of a MUX is well known. We briefly explained about the *thru* function in Section 3.4. An REFF consists of two flip-flops and a MUX as shown in Figure 3.7. The control input of the MUX is used as the mode selector. In normal mode the REFF behaves like a normal flip-flop. Using normal mode two bits can be loaded to the REFF. Just after loading two bits, the mode can be changed to test mode. In test mode the bits exchange their position at every clock but remain stored in the REFF. Hence an REFF can be regarded as a 2-bit hold register. Of a two-pattern vector, the bit of which vector should be loaded first to the REFF depends on the time of loading and the time of applying the vectors.

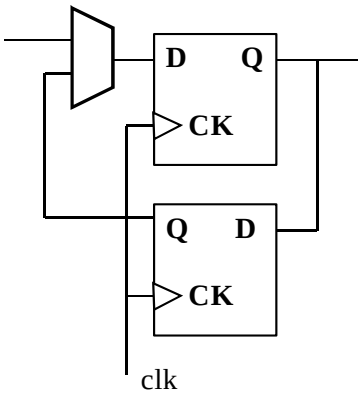


Figure 3.7: Rotating enhanced flip-flop

3.7 Algorithm for Adding DFT Elements

Figure 3.8 illustrates the flowchart of our algorithm. In the following sub-sections we describe in detail the techniques and heuristics we use for different steps of the flowchart. For simplicity we describe our algorithm assuming that operational modules in the data path has a maximum two inputs and there is no chaining of modules. This means there is no RTL path of degree 3 or more in the data path. Most of the benchmarks satisfy this condition. Also, our approach can be extended for data paths with modules having more than two inputs.

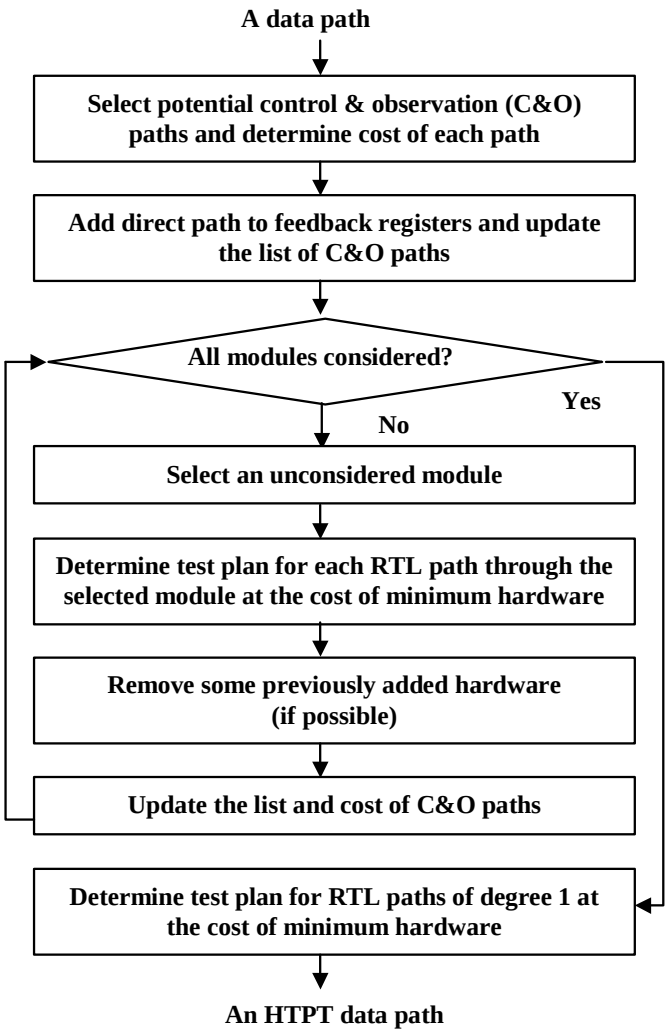


Figure 3.8: Flow-chart of the algorithm

3.7.1 Selecting Potential Control and Observation Paths

Every register (other than the constant registers) of a data path is at the starting of some RTL path and also at the ending of some RTL path. Control paths are therefore necessary from PIs to all registers and so are the observation paths from all registers to POs. However the number of paths in the data path can be very large and we use some heuristics to select some potential control and observation paths. We make a set CP of potential control paths and a set OP of potential observation paths for each register R. The first members of CP are the shortest depth paths from each PI to R. We favor the shortest depth paths because they help to reduce the test application time. To determine

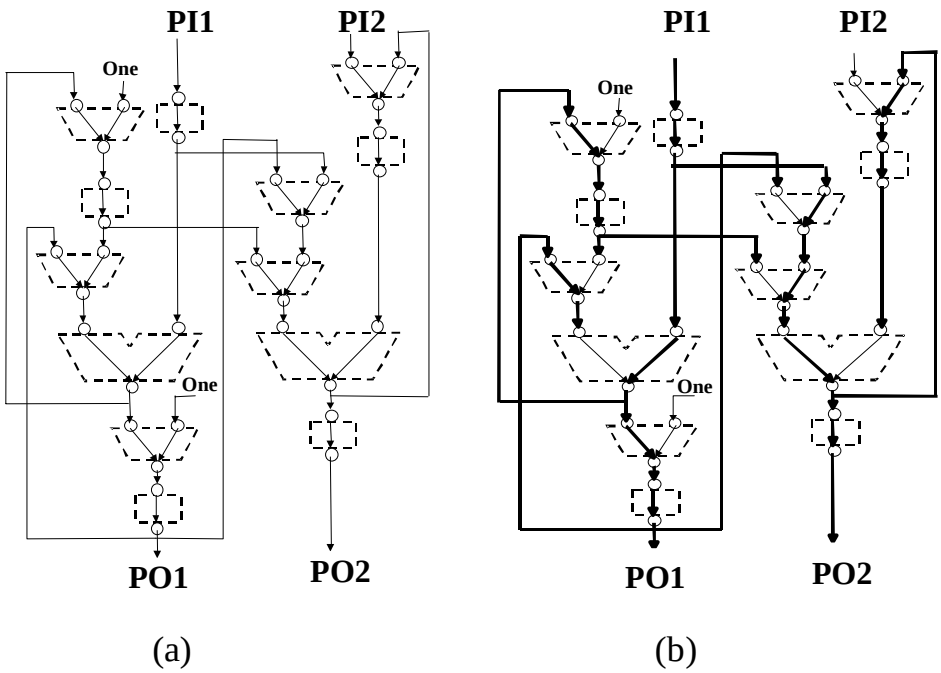


Figure 3.9: (a) Port digraph of the data path of Figure 3.1 and (b) Breadth first tree of the digraph of Figure 3.9(a)

the shortest depth control paths, we represent the data path as a port digraph [23]. The digraph of the data path of Figure 3.1 is shown in Figure 3.9(a). A node represents a port in the data path and any edge say, (u,v) implies that either a metal line connects u to v or u and v are input port and output port, respectively of the same element. The shortest depth control paths can be selected by performing a modified *breadth first search* (BFS) on the digraph. The BFS is performed n times where n is the number of PIs of the data

path. Each of n PIs are once considered as the source node. Figure 3.9(b) shows the breadth first tree (thick lines) of the digraph of Figure 3.9(a) considering PI1 as the source node. This tree contains the shortest depth paths from PI1 to each register reachable from PI1. Similarly the members of OP of any register R are the shortest depth paths from R to each PO. The shortest depth observation paths can be determined by reversing the edges of the digraph and performing similar BFS m times where m is the number of POs. For each member of CP and OP of each register we maintain a variable, which we refer to as the "cost" of the path. The cost of the path is nothing but the number of *thru* functions associated with the path, i.e. the cost of a *thru* function is 1.

3.7.2 Adding Direct Paths to Feedback Registers

We first consider the (feedback registers) FRs of all modules. Suppose a register R_f appears on both input and output sides of the SCG of a module M. This implies that R_f is an FR to M. Now, if a control path from a PI to R_f incorporates a *thru* function to M, then this control path cannot be used to test any RTL path that crosses M (explained in Section 3.4). So we first find in the CP of R_f for a control path without a *thru* function to M. If we succeed, we switch to another FR of M or FR of other module. If no such control path is found in the CP of R_f we search the entire data path. For this search, we remove the edges from inputs to output of M in the digraph and then find shortest path from each PI to R_f until a path is found. If a path is found, this path is added to the CP of R_f. However the failure of such a search implies that there is no control path from any PI to R_f without a *thru* function to M. Under such a situation a direct path is added to R_f using a test MUX from a PI. This PI is chosen based on the control paths in CPs of the registers connected to the other input of M (input to which R_f is not connected). If the majority of the nodes connected to the other input of M have control path from a PI, we choose other PI for R_f. This direct path is included to the CP of R_f. In case the data path has single input, it might be difficult to find a suitable PI. In such a situation we augment each flip-flop of R_f to REFF. This information is added to the paths in the CP of R_f. We then switch to another FR of M or FR of another module until all FRs in the data path are considered. We start to consider the FRs of the modules nearer to PIs first. Sometimes a single MUX can be

used for more than one FR of a module. In Figure 3.10, the test MUX is providing control paths from *PI2* to both *R3* and *R4*. In the case of a single input data path, if it becomes necessary to augment flip-flops of more than one register to REFF, we use a global REFF register connected to the only PI of the data path. Such an REFF register can be used as a PI for two-pattern testing. This is because an REFF can store two bits for as long as it is necessary. Direct paths then can be added from the global REFF register to FRs using test MUXs. We use global REFF because the hardware overhead incurred by registers is very high. Whenever we add any DFT element, we update the CPs of registers affected by the addition. Addition of test MUXs and REFF registers creates some more paths of degree 1 in the data path. We should also consider the two-pattern testability of these paths. Because of adding DFT elements in this step, the SCGs of modules in the data path remain unchanged.

3.7.3 Determining Test Plans

We discussed our algorithm for adding some DFT elements in the previous section. In this section, we add more DFT elements to ensure the test plans for all RTL paths. First we address the RTL paths of degree 2 and then of degree 1.

RTL paths of degree 2: We consider all RTL paths passing through a module at a time. The following four steps are performed for each 2-input operational module.

Generating RCG from SCG (step 1): Let M be a 2-input module and R_{i1}, R_{i2} and R_o are the set of nodes connected to the left input, right input and output of the module. Each member of R_{i1} and R_{i2} has a set of control paths CP and each member of R_o has a set of observation path OP. Let $r_{j1} \in R_{i1}$ and $r_{k2} \in R_{i2}$. Now we look for two control paths, one from CP of r_{j1} and other from CP of r_{k2}, such that they satisfy any one of the four conditions of *Theorem 1*. In case we find more than one pair of control paths to satisfy any condition of *Theorem 1*, we choose the pair having lowest cost. We add a control edge between r_{j1} and r_{k2} in the RCG and keep a record of the lowest cost control paths that support this edge. We should keep in mind that in the case that r_{j1} and r_{k2} is a FR to M, we cannot consider a control path for r_{j1} or r_{k2} incorporating a *thru* function to M unless r_{il} or r_{jr} is augmented to an REFF register. If we fail to find a control edge between r_{j1} and r_{k2} we switch to another pair of nodes. We search for control edges between every

possible pair of nodes with one node from R_{i1} and one node from R_{i2}. For each member of R_o, we choose the lowest cost observation path from its OP and add path edges to the RCG as described in *rule 2* in Section 3.2.

Adding DFT elements to Augment RCG (step 2): In the RCG, some node/nodes of R_{i1} or R_{i2} may not be connected to any control edge. For any such node we add DFT elements to create a control edge in the RCG incorporating this node. First we try by adding a direct path to such a node from some PI by means of a test MUX. In the worst case we augment such a node to an REFF register. However, if it becomes necessary to augment more than one register, we create a global REFF register as mentioned in Section 4.2.2. Direct paths can then be created from the global REFF register to the required nodes by means of test MUXs. The RCG of M is now sufficient for two-pattern testability of any RTL path through it. However, when a direct path to a node is created, we check whether this path can be utilized to remove some DFT elements added for the previously processed modules. A direct path to a node might be used instead of some other necessary control path/paths to the same node, which incorporates a *thru* function.

Minimizing control edges in RCG (step 3): The RCG may have some excess control edges. Of all the control edges we select the minimum number of them that fulfill the condition that each node in R_{i1} or R_{i2} is connected to at least one control edge. The minimum number of control edges will create the minimum number of test cliques in the RCG that are sufficient for test plans of RTL paths through M.

Adding DFT elements for thru Functions (step 4): At this step, we relax the assumption that a *thru* function exists from any input to the output of any operational module. We first check whether a *thru* function that we need really exists (can be implemented by a support path) or whether we should add a DFT element (mask or others) for it. To do this we consider one test clique at a time. A test clique in the RCG of a two-input operational module incorporates two control paths and one observation path. We search for support paths for the *thru* functions associated to these three paths using some manual calculation. A support path may have a timing conflict with control paths or other support paths. Timing conflict means it becomes necessary to provide different values at the same PI at the same time. We add mask elements to realize the *thru* functions for which any support path cannot be found without timing conflict. The cost of the *thru* function

realized by a mask element becomes zero and we update the cost of all control and observation paths, which includes this *thru* function. We consider all the test cliques in the RCG of M in a similar way.

The above four steps are performed for all 2-input modules considering the modules nearer to the PIs first. When we finish all the 2-input modules, the testability of all the RTL paths of degree 2 is ensured.

RTL paths of degree 1: We now consider the testability of RTL paths of degree 1. Let R_s and R_e be the start and end registers of an RTL path of degree 1. We choose the lowest cost path from the CP of R_s as the control path and the lowest cost path from the OP of R_e as the observation path. If any *thru* function of cost 1 is associated with these paths, we try to realize it using a support path. In case we cannot find any support path without timing conflict, we add a mask element or a multiplexer to realize them. We consider all RTL paths of degree 1 in a similar way. Figure 3.10 shows the HTPT equivalent of the data path of Figure 3.1 augmented by following the algorithm described above.

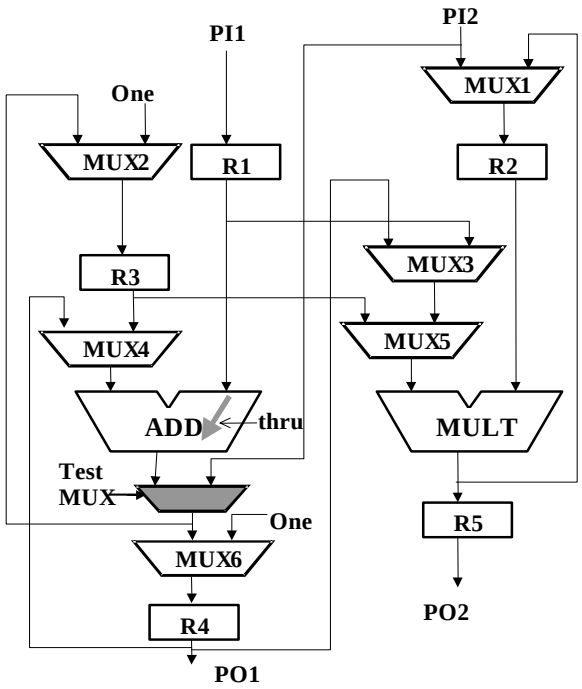


Figure 3.10: The HTPT equivalent of the data path of Figure 3.1

3.7.4 Relaxing the Assumption of Bit Width

In practice, a data path may not be of consistent bit width. It is not a problem, if a control path becomes gradually narrower or an observation path becomes gradually wider in their respective downstream. However, in the case of a control path, it is a problem if it is narrower than its end point anywhere in the upstream. Such a problem can be tackled by means of a MUX or an REFF and a MUX. A direct path is added from some PI or REFF to the end point of the control path using a MUX. In test mode, this direct path controls the bits that cannot be controlled because of some narrower bit-width in the upstream In the case of an observation path, it is a problem if it becomes narrower than its start point anywhere in the downstream. Such a problem can be tackled by using one or more MUXs. Direct path/paths are added from the start point of the observation path to the PO/POs using MUX/MUXs. In test mode, these direct paths are used to observe extra bits that cannot be observed because of some narrower bit-width in the downstream.

3.8 Experimental Results

In this section we present experimental results to compare our method with the enhanced scan approach. For comparing area overhead we applied our method to data paths of three benchmark circuits and a RISC processor provided by industry. The characteristics of these data paths are shown in Table 3.5. In this table PIs and POs denote the number of primary inputs and primary outputs of the data path respectively. REGs, MUXs and OPs are the numbers of registers, multiplexers and operational modules in the data path. The last column of Table 3.5 shows the areas of the data paths generated by the logic synthesis tool Design Compiler (Synopsys). The areas depend on the libraries we use for design synthesis.

Table 3.5: Circuit characteristics

Circuit	Bit width	PIs	POs	REGs	MUXs	OPs	Area
Paulin	16	2	2	7	11	4	10528
LWF	16	1	1	5	5	3	3512
Tseng	16	3	2	6	7	7	7802
RISC	32	1	3	40	84	19	94357

Table 3.6 shows the results regarding area overhead. In both our method and the enhanced scan approach, the percentage of area overhead decreases with the increase in bit width of the data paths. However, the area overhead incurred by our method is always much lower. For our DFT method, the columns MUX, *thru* and REFF show the number of added MUXs, *thru* functions and REFF registers.

Table 3.6: Comparison of area overhead

Circuit	Bit width	Enhanced scan	Our method			
		Area Overhead (%)	Area Overhead (%)	MUX	*thru*	REFF
Paulin	8	42.77	8.52	3	2	0
	16	27.66	5.53			
	32	16.23	3.36			
LWF	8	65.99	12.75	0	0	1
	16	59.23	11.28			
	32	56.55	10.88			
Tseng	8	42.39	4.76	1	2	0
	16	31.99	3.93			
	32	21.65	2.62			
RISC	32	35.27	2.04	2	2	1

The tool we used for the experiments to compare test application time are DelayPath and TestGen from Synopsys. DelayPath can generate path list for designs of up to 5000 gates. And TestGen can generate tests for only robust and nonrobust testable paths. Because of these limitations we conducted our experiments for the data path of Paulin, LWF and Tseng considering their bit width only 8. Also, test vectors that cover only robust and non-robust testable paths are considered. 100% coverage is obtained for robust and non-robust testable paths in both our method and the enhanced scan approach. Identifying and generating tests for functional sensitizable paths in a large multi-level circuit is a hard task [1]. However, if test can be generated for such paths, both our method and the enhanced scan approach allow their application.

Table 3.7 shows the results of our experiments regarding test application time. For all three data paths the test application time of our method is much shorter compared to that of the enhanced scan approach. This is because, in the enhanced scan approach, the test vectors and test responses are shifted in to and shifted out from the enhanced scan chain serially. Contrary to this our method supports the parallel propagation of test vectors and test responses via data path lines.

Table 3.7: Test application time for 100% coverage of robust and non-robust testable paths (cycles)

Circuit	Bit width	Enhanced scan	Our method
Paulin	8	442888	5950
LWF	8	61580	7510
Tseng	8	106720	2936

3.9 Conclusions

The concept of hierarchical testability for delay faults is introduced in this chapter. Precomputed vector pairs are propagated via control paths from primary inputs to appropriate locations and are applied in a two-pattern fashion to test paths of unity sequential depth. Test responses are propagated via observation paths from the end points of paths under test to primary outputs. A DFT method is also presented that can be applied to augment a data path to become a hierarchically two-pattern testable one. Priority has been given to use the existing paths in the data path as control paths and observation paths. Experimental results show that the area overhead and test application time of our DFT method is smaller compared to those of the enhanced scan approach for some benchmark data paths.

Design for Two-Pattern Testability of Controller-Data Path Circuits

4.1 Introduction

In this chapter, we introduce a design for testability (DFT) scheme for delay faults of a controller-data path circuit. The scheme makes use of both scan and non-scan techniques. Firstly, the data path is transformed into a hierarchically two-pattern testable (HTPT) data path based on the method described in Chapter 3. Then an enhanced scan chain is inserted on the control lines and the status lines. The enhanced scan chain is extended via the state register of the controller. If necessary, the data path is further modified. Then a test controller is designed and appropriately integrated to the circuit. Our approach is mostly based on path delay fault model. However the multiplexer (MUX) select lines and register load lines are tested as Register Transfer Level (RTL) segments. For a given circuit, the area overhead incurred by our scheme decreases reasonably with the increase in bit-width of the data path of the circuit. The proposed scheme support hierarchical test generation and can achieve fault coverage similar to that of the enhanced scan approach.

Testing delay faults in sequential circuits is significantly more difficult than testing delay faults in combinational circuits. This is because application of an arbitrary two-pattern test is not possible to non-scan or standard-scan sequential circuits [1]. Functional justification [10] and scan shifting [11] techniques are used to apply two-pattern test in circuits with standard-scan. However, these techniques cannot guarantee the application of arbitrary two-pattern test. The scheme that can apply arbitrary two-pattern test to a sequential circuit is the enhanced scan design [18] [19]. Enhanced scan flip-flops can store two bits and can apply them in consecutive clocks. However the area overhead of the enhanced scan approach is very high. Another disadvantage of the enhanced scan approach is that the test application time is very long because of scan operation.

A controller-data path circuit is a sequential circuit. A controller is usually a finite state machine. It consists of a state registers and a combinational logic block. Behaviorally a data path is represented by a data flow graph, in which nodes represent operations (addition, multiplication etc.) and edges represent data variables. Structurally, the data path consists of operational modules, registers, MUXs and buses. The controller applies control signals to the data path in each time step and thus guides the data path in making computations. In some circuits, the controller receives status signal from the data path in some time steps. In chapter 3, we presented a method to transform a data path to an HTPT one. As an extension to that, in this chapter we develop a scheme for two-pattern testability of a controller-data path circuit. The model we consider in our approach for a controller-data path circuit is shown in Figure 4.1. In this model, registers separate the controller and the data path. The register on the control lines is referred to as control line register (CLR) and the register on the status lines is referred to as the status line register (SLR). We also assume that the circuit is a synchronous sequential circuit.

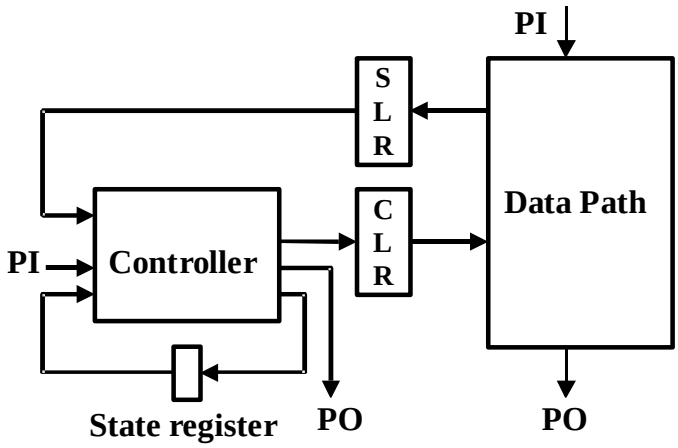


Figure 4.1: A controller data path circuit

4.2 The HTPT Data Path and Test Plan Generator

In chapter 3, we describe a method to transform a data path to an HTPT data path. However, in chapter 3, we have not considered the paths involving the control and status signals. After adding control signals to the arbitrary data path of Figure 3.1, we present it in Figure 4.2 and its HTPT equivalent is shown in Figure 4.3. The thru function and the test MUX are the added DFT elements. In chapter 3, we defined test plan of an RTL path as the set of control paths and an observation path that are sufficient to ensure the hierarchical two-pattern testability of the RTL path. Here we represent test plan in a different way. Referring to Figure 4.2, the test plan of the RTL path "R1-M3-M5-MULT-R5" consists of disjoint control paths "PI1-R1" and "PI2-M1-R2" and the observation path “R5-PO2”. Alternatively, we can represent a test plan as a sequence of control vectors [20]. The control vector sequence guides the propagation of two-pattern vectors from PIs

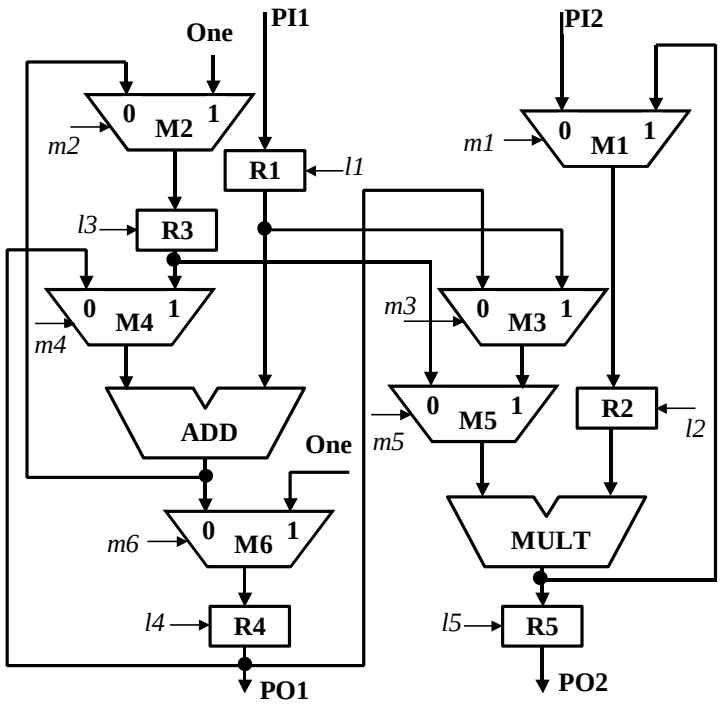


Figure 4.2: An arbitrary simple data path

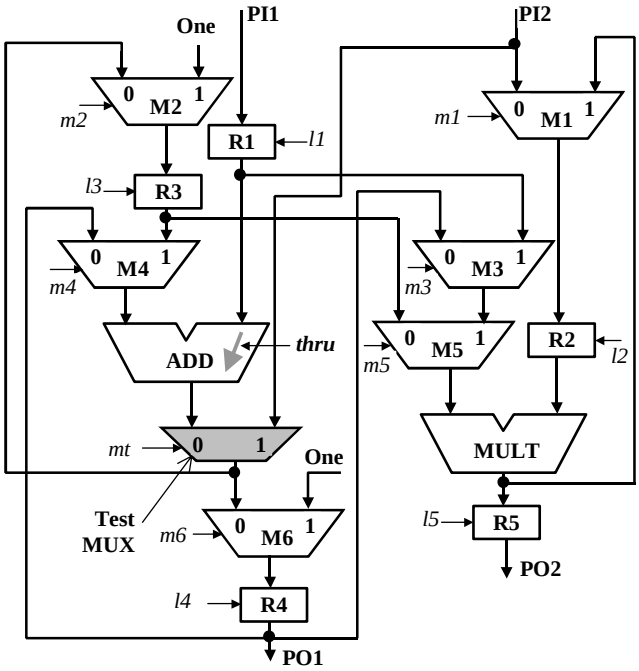


Figure 4.3: The HTPT equivalent of the data path of Figure 4.2

to appropriate registers and the propagation of the test responses from appropriate register to some PO. The sequence of Table 4.1 shows the test plan of the path "R1-M3-M5-MULT-R5" of Figure 4.2. Control signals that are not related to this test plan are not shown in this table.

Table 4.1: An example of a test plan

Time	PI1	PI2	m1	m3	m5	l1	l2	l5	PO2
1	C	C	0	X	X	1	1	X	-
2	C	C	0	1	1	1	1	X	-
3	X	X	X	1	1	X	X	1	-
4	X	X	X	X	X	X	X	X	O

C: apply test pattern to PI X: don’t care
O: observe test response at PO -: Need not be observed

In our scheme, we first convert a data path to an HTPT data path. Then, we need a test plan generator that can generate test plan for each RTL path of the data path. A test plan

generator is a sequential circuit. In Section 4.4, we discuss more about the test plan generator and how to integrate it to a controller data path circuit. The addition of the test plan generator ensures the testability of the RTL paths but the testability of the paths involving control signals and status signals still remains unsolved. In the following section we discuss the issue of control signals. In section 4.4, we discuss the paths involving status signals.

4.3 RTL Segments and Its Testing

In this chapter, we consider MUX select lines and register load lines as RTL segments. Each segment starts at the CLR and ends at a MUX or a register of the data path. A transition at the select line of a MUX ensures non-controlling or controlling values to certain lines internal to the MUX and thus allows the signals of one of the data inputs to propagate to the output. A transition at the load line of a register determines whether the register should load or not load the data from its input in the following clock. We sensitize a transition along an RTL segment only from CLR to respective MUXs or registers. We test delay faults along these segments using an approach that is consistent with the overall operation of a controller-data path circuit. Our approach is developed in the perspective of RTL i.e. in a broader sense we assume the registers and MUXs as primitive elements. The key to our approach is based on the fact that a delay fault along a MUX select line or a register load line is supposed to change the data flow in the data path. In the following discussion we clarify our approach.

4.3.1 Multiplexer Select Lines

Lemma 4.1: A delay fault along the path segment represented by a MUX select line prevents the propagation of appropriate data from the MUX's input side to the output side within specified time.

Proof: In Figure 4.4, a rising transition is shown at the start of the multiplexer select line. Before transition, B propagates to O and after transition A propagates to O. Therefore any delay along the select line will delay the propagation of signal from A to O. □

Corollary 4.1: The effect of a delay fault along a path segment represented by the select line of a two-input MUX can be successfully observed at its output if its input values are different.

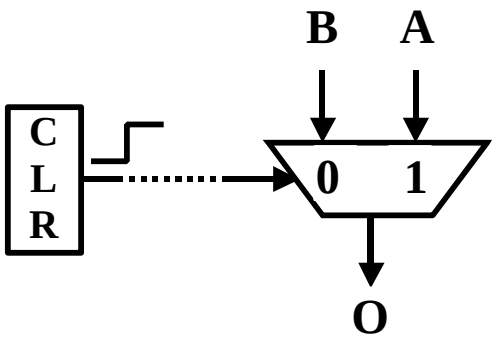


Figure 4.4: A MUX and its select line

Proof: Referring to Figure 4.3, before transition O=B and after transition O=A. If A=B then the transition has no effect at the output. Hence by observing the output it is not possible to decide whether the select line has a delay fault or not. But, if A≠B and the output is captured after nominal propagation delay then it is possible to detect whether the select line has a delay fault or not. □

Definition 4.1: The ***critical segment*** of a line in a data path is the longest possible segment of unity sequential depth that starts at the line and ends at a register (in the direction of the data flow) or a PO.

In the following example we explain a typical method of testing delay fault along a multiplexer select line.

Example 4.1: Referring to Figure 4.3, the inputs of *M3* are connected to *R1* and *R4* and *m3* is the select line of *M3*. Let in some clock t_0, any two values ***A*** and ***B*** such that ***A≠B*** are loaded to *R1* and *R4*, respectively and some known value ***C*** is loaded to *R2*. Also a rising transition is launched at the start of *m3* (at CLR) and the path from the output of *M3* via *M5* to *MULT* is selected in t_0. In t_1 (the following clock) the output of the *MULT* is loaded to *R2*. For correct data path operation, ***A*C*** (bits of lower half) should be loaded to *R2*. But if a delay fault along *m3* exists, ***B*C*** (bits of lower half) or some other incorrect value is loaded to *R2*. Since ***A≠B***, it is possible to detect the delay fault along *m3* (if present) by observing the value that *R2* assumes at the clock cycle t_1. However, the

values of ***A***, ***B*** and ***C*** should be such that bits of lower half of ***A*C*** does not match with those of ***B*C***. Notice that here the test response can also be loaded to *R5*. However, we choose *R2* because *R2* is at the end of the critical segment of the output of *M3*. If the test passes for the case of the critical segment, it will also pass for the other segments. From the perspective of RTL, it is enough that ***A≠B*** for this method of testing. However, with an insight to the gate-level structure of a MUX, we choose ***A*** and ***B*** such that every bit of ***A*** is different from the corresponding bit of ***B***. This partially relaxes our assumption that a MUX is a primitive element.

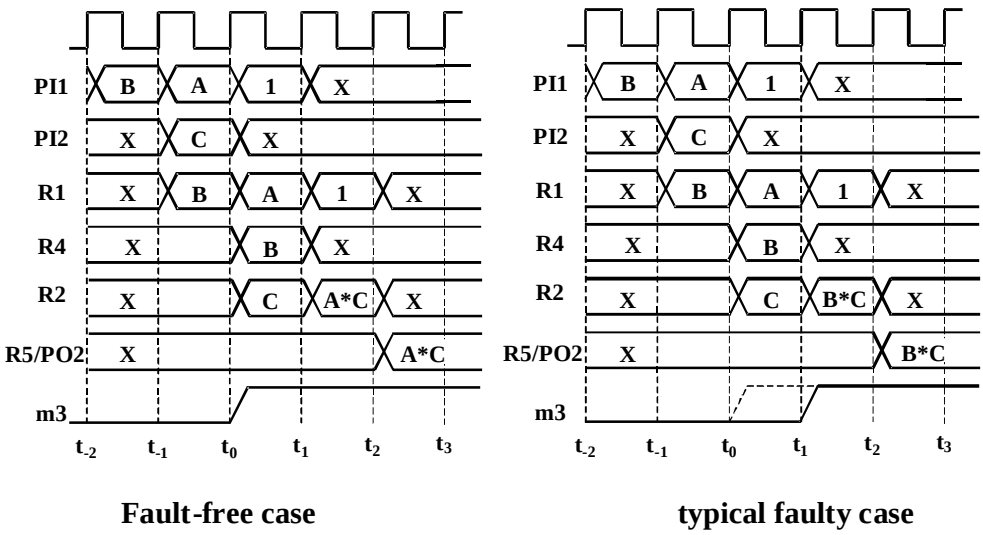


Figure 4.5: Waveforms related to Example 4.1

Now we explain how to apply the typical method descried above using the enhanced scan chain along the CLR. The enhanced scan chain can store and hold two bits for each control signal. Let *l1*=1, *thru*=1, *mt*=0, *m6*=0, *m3*=0, *m5*=X, *m1*=0, *l2*=1, *l5*=X are shifted into the enhanced scan chain as the first bits and *l1*=1, *thru*=X, *mt*=X, *m6*=X, *m3*=1, *m5*=1, *m1*=1, *l2*=1, *l5*=1 as the second bits. For other control signals both bits are don't care. The waveform of Figure 4.5 illustrates the events of the typical method described above for a fault-free case and a typical faulty case. The events are regulated by exercising a "hold-shift-hold" sequence of the enhanced scan chain. This means that we hold the first bits as control signals and apply necessary data from PIs of the data path

and then we apply the shift operation at an appropriate moment to launch the required transition. We then hold the second bits as control signals to propagate the test response to some PO of the data path. The rising transition at *m3* is launched in the clock t_0 using the shift operation. Before transition the first bits and after transition the second bits are held in the CLR for required number of clock cycles. In the clock t_1, numerical value 1 is loaded to *R1* to ensure a *thru* function to *MULT*. Similar method can be used to test a falling delay fault along a MUX select line.

Definition 4.2: The ***general controllability*** (***known controllability***) $\boldsymbol{C_g}$ ($\boldsymbol{C_k}$) of a line in a data path with respect to some clock is the ability of controlling the line in that clock to any desired value (some known value) by appropriately controlling the PIs and by using the hold and shift properties of the enhanced scan chain along the CLR.

Definition 4.3: The ***observability*** (***verifiability***) $\boldsymbol{O}$ ($\boldsymbol{V}$) of a line in a data path with respect to some clock is the ability to propagate the value (the signature of the value) that the line assumes in that clock to some PO by appropriately controlling the PIs and by using the hold and shift properties of the enhanced scan chain along the CLR.

Definition 4.4: The O (V) of a line in a data path with respect to some clock is called the ***critical observability*** (***critical verifiability***) $\boldsymbol{O_c}$ ($\boldsymbol{V_c}$) while (i) the process of propagation is performed through the critical segment of the line and (ii) the propagation reaches beyond the critical segment in the immediately following clock.

Theorem 4.1: The (rising/falling) delay fault along the select line of a MUX in a data path is testable if the following requirements are fulfilled with respect to a certain clock.

(i) The ability to launch a (rising/falling) transition at the start of the select line of the MUX.

(ii) $\boldsymbol{C_g}$ of both of the inputs of the MUX, or $\boldsymbol{C_g}$ of an input and $\boldsymbol{C_k}$ of another input.

(iii) $\boldsymbol{O_c}$ / $\boldsymbol{V_c}$ of the output of the MUX.

Proof: Requirement (i) is obviously necessary for delay testing. The delay fault along the select line of a MUX can be detected by applying two different values at the inputs of the MUX (*Corollary 1*). Requirement (ii) allows the application of two values to two inputs of the MUX, which are different from each other at every bit. Requirement (iii) allows the observation of the test response in a way that is consistent for delay testing. □

4.3.2 Register Load Lines

Lemma 4.2: The rising delay fault along the path segment represented by the load line of a register prevents it from loading data in the following clock (assuming that logic 1 enables the register).

Proof: The proof of this lemma is obvious. □

In the following example we explain a typical method of testing rising delay fault along a register load line.

Example 4.2: In Figure 4.3, *l3* is the load line of *R3*. Suppose initially *l3* is at logic 0 i.e. *R3* is disabled. Let, in the clock t_0, some known value ***A*** is loaded to *R1*, the path from *R1* to *R3* is selected (using the *thru* to ADD) and a rising transition is launched at the start of *l3*. For correct data path operation, ***A*** should be loaded to *R3* in the following clock t_1. But if there is a delay fault along *l3*, *R3* fails to load ***A*** in t_1. So, observing the value that *R3* contains in the clock t_1 can test the fault. Since *R3* is initially disabled the value it contains before loading ***A*** is unknown. Let this unknown value is ***U***. Hence the test is invalid, if somehow ***A*** completely matches with ***U***. If ***A*** is arbitrarily chosen then the probability that ***A*** completely matches with ***U*** is $1/2^n$, where n is the bit-width of the data path. To reduce the probability of the invalidity of the test, the test can be conducted more than once using different values of ***A***.

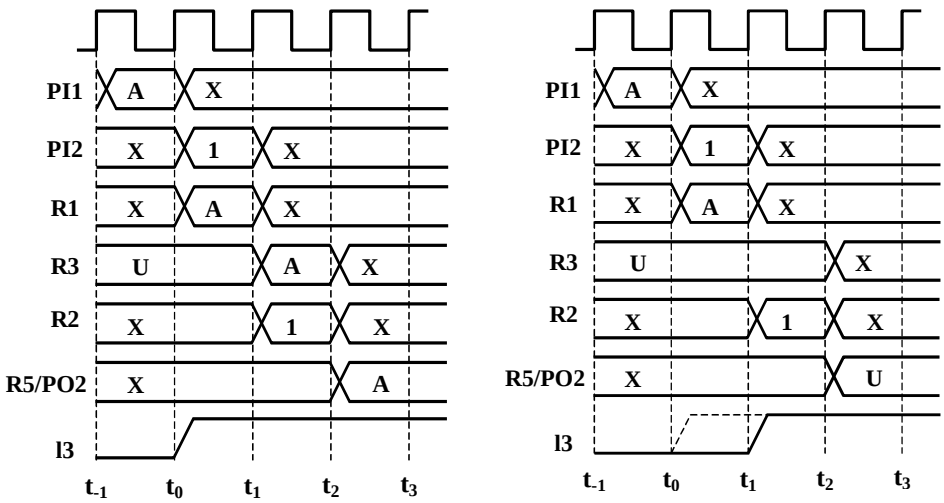


Figure 4.6: Waveforms related to Example 4.2

The waveform of Figure 4.6 illustrates the events of the typical method described above for a fault-free case and a typical faulty case. The explanation of Figure 4.6 is similar to that of the Figure 4.5 of *Example 4.1*. In this example too, exercising a "hold-shift-hold" sequence of the enhanced scan chain regulates the events. However here, *l1*=1, *thru*=1, *mt*=0, *m2*=0, *l3* = 0, *m5*=X, *m1*=X, *l2*=X, *l5* = X are the first bits and *l1*=X, *thru*=1, *mt*=0, *m2*=0, *l3*=1, *m5*=0, *m1*=0, *l2*=1, *l5* = 1 are the second bits.

Theorem 4.2: A rising delay fault along the load line of a register is testable if the following requirements are satisfied.
(i) The ability to launch a rising transition at the start of the load line at some clock.
(ii) ***Cg*** of the input line of the register with respect to the same clock
(iii) ***O/V*** of the output line of the register with respect to the following clock
Proof: The proof of this theorem is obvious in the context of *Lemma 4.2* and *Example 4.2*. □

Lemma 4.3: The falling delay fault along the path segment represented by the load line of a register causes an unwanted loading in the following clock.
Proof: The proof of this lemma is obvious. □
In the following example we explain a typical method of testing falling delay fault along a register load line.

Example 4.3: Referring to Figure 4.3, suppose *l3* is initially at logic 1 i.e. *R3* is enabled and the path from *R1* to *R3* is selected (using the *thru* to ADD). Let, in the clock t_{-1} some value ***A*** is loaded to *R1*. Then in the following clock t_0, ***B≠A*** is loaded to *R1*. Obviously ***A*** is loaded to *R3* in the clock t_0. Also, a falling transition is launched at the start of *l3* (at CLR) in the clock t_0. For correct data path operation, *R3* should retain ***A*** in t_1 (next clock after t_0). But if there is a falling delay fault along *l3*, ***B*** is loaded to *R3* in the clock t_1. Since ***A≠B***, observing the value that *R3* contains in the clock t_1 can test the delay fault along *l3*. To relax our assumption that a register is a primitive element, we choose ***A*** and ***B*** such that every bit of ***A*** is different from the corresponding bit of ***B.***

The waveform of Figure 4.7 illustrates the events of the typical method described above for a fault-free case and a typical faulty case. The explanation of Figure 4.7 is similar to that of the Figure 4.5 of *Example 4.1*. As it is the case for *Example 4.1* and *Example 4.2*,

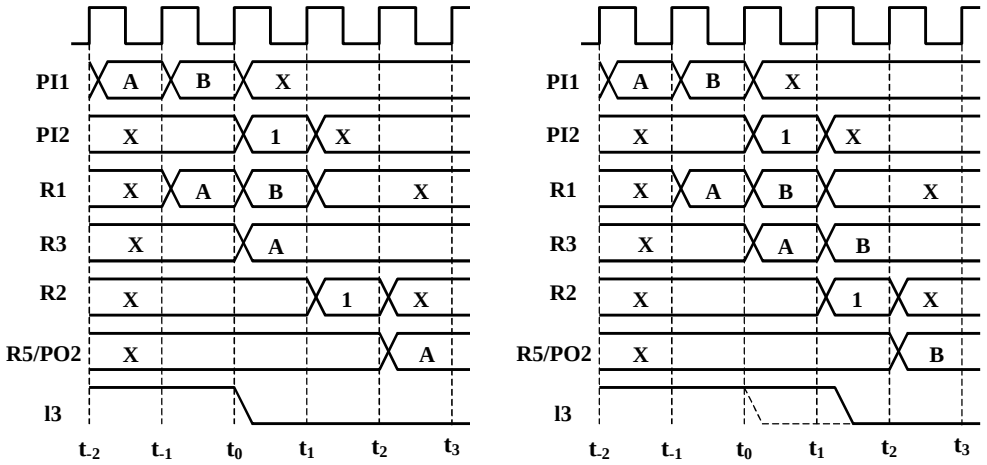


Figure 4.7: Waveforms related to Example 4.3

here too exercising a "hold-shift-hold" sequence of the enhanced scan chain regulates the events. However here, *l1*=1, *thru*=1, *mt*=0, *m2*=0, *l3*=1, *m5*=X, *m1*=X, *l2*=X, *l5* =X are the first bits and *l1*=X, *thru*=1, *mt*=0, *m2*=0, *l3*=0, *m5*=0, *m1*=0, *l2*=1, *l5*= 1 are the second bits.

Theorem 4.3: The falling delay fault along the load line of a register in a data path is testable if the following requirements are fulfilled.

(i) The ability of launching a falling transition at the start of the load line in some clock C.

(ii) ***Cg*** of the input line of the register with respect to two consecutive clocks – the clock before C and C.

(iii) ***O/V*** of the output line of the register with respect to the following clock of C.

Proof: The proof of this theorem is obvious in the context of *Lemma 4.3* and *Example4. 3*. □

Our target is to fulfill the requirements mentioned in *Theorem 4.1* for each of the MUX select lines and to fulfill the requirements mentioned in *Theorem 4.2* and *Theorem 4.3* for each of the register load lines. An HTPT data path usually fulfills these requirements but does not guarantee. For example, the HTPT data path of Figure 4.3 fulfills the requirements for all the control segments even if the order of the flip-flops of the enhanced scan chain along the CLR is arbitrary. However, sometimes an HTPT data path may not fulfill the requirements for some control segments. We first try to solve this problem by imposing some restrictions on the order of the flip-flops of the enhanced scan chain along the CLR. In such a case, a control segment is tested using an extended sequence of hold and shift operations of the enhanced scan chain instead of just "hold-shift-hold". However, changing the order of the flip-flops may not ensure the testability of all the control segments. Sometimes addition of *thru* function or MUX or both to the data path might be needed and we add those to the data path, as it is needed.

4. 4. The Proposed DFT Scheme

In this section, we discuss a scheme for two-pattern testability of controller-data path circuits. We classify the paths and segments of a controller-data path circuits into following four groups:

Group1: RTL paths (explained in Section 2) of the data path excluding the paths those end at SLR.

Group2: RTL paths those end at SLR.

Group3: The paths that start at the PIs or the state register of the controller or at SLR and end at the POs or the state register of the controller or at CLR

Group4: Segments represented by register load lines and MUX select lines.

Our aim is to develop a scheme that ensures two-pattern testability for all these paths and segments. In the process of developing the scheme, we first transform the data path to an HTPT one by following the method of [6]. We further modify the data path to fulfill the requirements for the testability of the control segments (discussed in Section 3). Figure 4.8 shows the detail of our scheme. The scan chain along the state register of the controller, the CLR and the SLR is an enhanced scan chain. Only one bit of some PI and

only one bit of some PO of the data path is shared for S_in and S_out ports of the ES chain, respectively. The test controller of Figure 4.8 is a sequential circuit. The test controller consists of a test plan generator (TG) and a test plan index register (TIR). A TG generates a test plan (discussed in Section 4.2) i.e. a sequence of control vectors depending on the contents of the TIR. The length of the TIR is $\log_2 n$ where n is the number of required test plans to test the RTL paths of the data path. The input of the TIR is connected to a PI of the data path. We add three extra pins to the circuit. They are L/H, T/N and S/P. L/H is connected to the load enable signals of the TIR. An inversion of L/H is connected to the load enable signals of the CLR. The T/N pin is used to multiplex the control signals of the normal controller and the test controller. Also, in normal mode the T/N pin apply logic zero to the control signals of the added DFT elements to the data path. The S/P pin is used to switch the enhanced scan chain from shift mode to parallel mode and vice versa. The S/P pin is also used to multiplex the data path output and the ES chain output. All these pins should be zero for normal circuit operation.

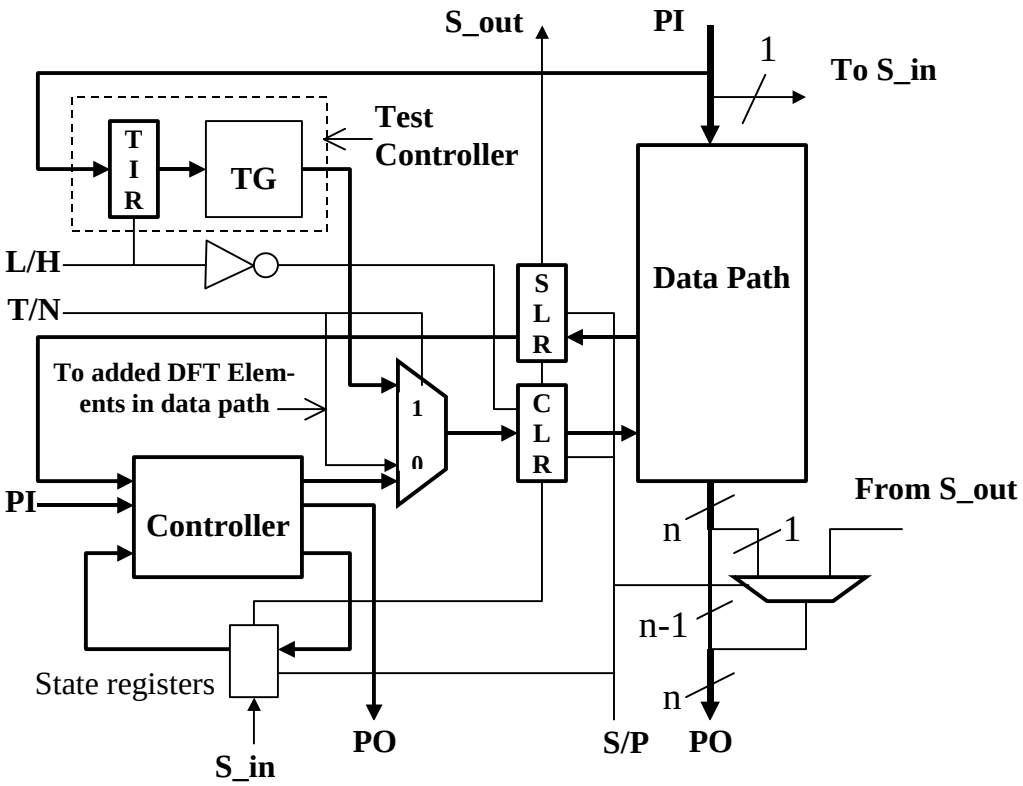


Figure 4.8: The proposed DFT scheme

Paths or segments of each group are tested separately. In the following we briefly discuss the procedure of testing the paths or segments of each group.

Group 1: First a test plan index is loaded to the TIR and then the TG generates the sequence of control signals necessary for that test plan. Test vectors are fed from the PIs and test responses are observed at a PO. Table 4.2 shows the required logic values for L/H, T/N, and S/P. The process is repeated for all the test plans.

Table 4.2: Logic values for L/H, T/N, S/P to test paths of group 1

Operation	L/H(1/0)	T/N(1/0)	S/P(1/0)
Loading TIR	1	X	X
Generating test plan	0	1	0

Group 2: The procedure of testing of the paths of this group is similar to the testing of the paths of group 1. However the test response is captured at SLR and then shifted out via the enhanced scan chain. Table 4.3 shows the required logic values for L/H, T/N, and S/P.

Table 4.3: Logic values for L/H, T/N, S/P to test paths of group 2

Operation	L/H(1/0)	T/N(1/0)	S/P(1/0)
Loading TIR	1	X	X
Generating test plan	0	1	0
Shifting out Test responses	0	X	1

Group 3: These paths are tested using the traditional enhanced scan approach. Test vectors are loaded to the enhanced scan chain and applied to the circuit by using shift operation. The test responses are captured in the enhanced scan chain and then shifted out for observation. Obviously the care bits for the test vectors span only the state register of the controller and the SLR. Table 4.4 shows the required logic values for L/H, T/N, and S/P.

Table 4.4: Logic values for L/H, T/N, S/P to test paths of group 3

Operation	L/H(1/0)	T/N(1/0)	S/P(1/0)
Loading enhanced scan chain	0	X	1
Applying vectors	0	0	1
Capturing test response	0	0	0
Shifting out test response	0	X	1

Group 4: The methods of testing the segments of this group have been described in Section 2. In short, each segment is tested separately by using the hold and shift properties of the enhanced scan chain and by controlling the PIs of the data path with appropriate data. Table 4.5 shows the required logic values for L/H, T/N, and S/P for hold and shift operations.

Table 4.5: Logic values for L/H, T/N, S/P to test segments of group 4

Operation	L/H(1/0)	T/N(1/0)	S/P(1/0)
Shift operation	0	X	1
Hold operation	1	X	X

4.5 Experimental Results

In this section we present experimental results to compare the area overhead incurred by our scheme with that of the enhanced scan approach. We applied our scheme to three benchmark circuits and a RISC processor provided by a semiconductor industry. The characteristics of these circuits are shown in Table 4.6. The columns “Controller” and “Data Path” contain the characteristics of the controller part and the data path part respectively. The columns PI, PO and area denote the numbers of primary inputs and primary outputs and area of respective parts. The logic synthesis tool Design Compiler (Synopsys) is used to generate the areas and they depend on the libraries we use for design synthesis. However, to imagine the circuit sizes, we mention here that the area of an AND gate is 2 units. The columns “state”, “status” and “control” in “Controller” denote the numbers of states, status inputs and control outputs respectively. The column

"bit", "reg" and "mod" in "Data Path" denote the bit width of the data path and the number of registers and operational modules in the data path.

Table 4.6: Circuit characteristics

Circuit	Controller						Data Path					
	PI	PO	state	status	control	area	PI	PO	bit	reg	mod	area
Paulin	1	0	6	0	16	86	32	32	16	7	4	6138
LWF	1	0	4	0	8	42	16	16	16	5	3	2120
Tseng	1	0	5	0	13	64	48	32	16	6	7	4573
RISC	1	2	11	54	61	580	32	96	32	40	19	59808

Table 4.7 presents the experimental results regarding area overhead. We divide the total area overhead incurred by our approach into three major parts and show them in three separate columns. We present the area overhead for the test controller and the MUXs we used in our scheme outside the data path under the column "Test controller and MUXs". The area overhead for the ES chain inserted between controller and the data path is shown under the column "ES chain". The area overhead needed to transform a data path to and HTPT data path or more is shown under the column "DFT added to data path". In both enhanced scan approach and our scheme, the area overhead decreases with the increase in bit-width of the data path. However, the rate of decrease is faster in our case. Area overhead incurred by ES approach is comparable to that of our approach for 8-bit circuits. But, for 16-bit and 32-bit circuits our scheme results in lower area overhead.

Table 4.7: Comparison of area overhead

Circuit	Bit-width	Area Overhead (%)				
		Enhanced scan approach	Our scheme			
			Test contr-oller and MUXs	ES chain	DFT added to data path	Total
Paulin	8	53.88	18.93	31.78	8.52	59.23
	16	34.88	7.01	11.77	5.53	24.37
	32	20.52	2.22	3.73	3.36	9.31
LWF	8	74.17	32.98	31.76	12.75	76.79
	16	68.24	16.50	16.23	11.28	44.01
	32	65.17	8.34	8.20	10.88	27.42
Tseng	8	54.72	17.10	31.08	4.76	52.94
	16	39.94	7.13	12.96	3.93	24.02
	32	26.83	2.57	4.68	2.62	9.87
RISC	32	38.66	4.54	5.07	2.04	11.65

To apply a two-pattern test in enhanced scan approach, both patterns are shifted into the enhanced scan chain serially. Then they are applied and the test response is captured. After capturing, the test response is serially shifted out of the enhanced scan chain. Therefore for a single two-pattern test it requires 4n+2 cycles where n is the number of flip-flops in the original circuit. However, if the scan-in operation of a test is overlapped by the scan-out operation of the previous test, then the test application time for m tests is m(2n+2) +2n cycles. Therefore the average time for a single test is approximately 2n+2 cycles. On the other hand, in our scheme the time required for a single two-pattern test is not constant. For example a two-pattern test for some path/paths of group 1 (mentioned in section 4) requires only a few cycles. But a two-pattern test for paths/segments of group 2, group 3 or group 4 involves shift-in and/or shift-out operations and require somewhat longer time. Therefore we calculate average number of cycles required for a single two-pattern test. In Table 4.8 we compare this average value with the number of cycles required for a single two-pattern test in the enhanced scan approach. This table roughly compares the test application time of our scheme with that of the enhanced scan

approach. The important thing to notice is that the test application time in our approach remains constant with the increase in the bit-width of the data path.

Table 4.8: Average number of cycles required for a single two-pattern test.

Circuit	Bit-width	Our Scheme	enhanced scan Approach
Paulin	8	37	114
	16	37	226
	32	37	450
Tseng	8	31	98
	16	31	194
	32	31	396
LWF	8	23	82
	16	23	162
	32	23	322
RISC	32	210	2562

4.6 Conclusions

In this chapter we have introduced a scheme for two-pattern testability of a controller-data path circuit. This is a hybrid scheme that makes use of both scan and non-scan techniques. The scheme is mainly based on path delay fault model. However the MUX select lines and register load lines are tested as RTL segments. The proposed scheme supports hierarchical test generation and can achieve fault coverage similar to the enhanced scan approach. In the context of hardware overhead, our approach is better for the circuits having data path of higher bit-width.

CHAPTER 5

Conclusions and Suggestions for Future Work

In this thesis, we present a hierarchical testability technique for delay faults. The increasing speed of digital designs is making their timing behavior critical and to ensure fault-free chips testing must be conducted. For timing faults two-pattern testing is required and traditional one-pattern stuck-at fault testing is not sufficient. Design-for-test (DFT) techniques such as the scan technique have been developed for stuck-at fault testing. For two-pattern testing extensions of the scan technique known as enhanced scan has been proposed. However, the drawbacks with enhance scan are long test application time and high area over-head. The approach we have presented in this thesis is an alternative to the enhanced scan approach. Our approach is developed on path delay fault model and applicable to controller-data path circuits.

In Chapter 3, we present an analysis on hierarchical two-pattern testability of a data path. We introduce the concept of RTL paths in a data path. Based on this we develop the definition of hierarchically two-pattern testable (HTPT) data path. We point out some necessary and sufficient conditions to support the propagation of two-pattern vectors via two or more control paths in a data path. We present these conditions as theorems together with proofs. A graph based analysis of the HTPT data path is also presented. Then a design for testability (DFT) method is discussed that can be applied to augment a data path to an HTPT data path. We propose a special type of DFT element called rotating enhanced flip-flop (REFF) that consists of two latches and a multiplexer. An REFF can hold two-bit data for as many cycles as it is required. Other DFT elements we use are multiplexers and thru functions. Our DFT method makes use of some graph algorithms such as breadth first search and finding shortest paths between nodes etc. We have conducted experiments by applying our method to several benchmark circuits and a RISC processor provided by a semiconductor industry. Through experiments we have showed that our method can achieve similar advantages to the enhanced scan approach at a much lower hardware overhead cost and test application time.

In Chapter 4, we propose a DFT scheme for two-pattern testability of a controller-data path circuit. The scheme makes use of both scan and non-scan techniques. The data path is first transformed into an HTPT data path. Then an enhanced scan chain is inserted on the control lines and the status lines. The enhanced scan chain is extended via the state register of the controller. In some cases, the data path is further modified by adding test multiplexers. Then a test controller is designed and integrated to the circuit. In this scheme, we test multiplexer select lines and register load lines as RTL segments. To test these control segments we propose some techniques that are compatible to overall operation of a controller data path circuit. For a given circuit, the area overhead incurred by our scheme decreases reasonably with the increase in bit-width of the data path of the circuit. The proposed scheme substantially lowers the test application time, supports hierarchical test generation and can achieve fault coverage similar to that of the enhanced scan approach.

The data path is the bigger part in a controller-data path circuit. The DFT method we propose in this thesis for a data path is very promising. Because, our method can provide similar advantages to the enhanced scan approach at a much lower hardware overhead cost and test application time. However, our scheme for a total circuit consisting of both controller and data path is not so good in the context of area overhead. Furthermore, the scheme is applicable to a controller-data path circuit where the registers separate the controller and the data path. As for future work, first we propose to improve this scheme by reducing the area overhead. Our second proposal is to find out some method that is applicable a general model of a controller data path circuit where the controller and the data path are not separated by registers. The method described in this thesis considers external testing environment. This is not so suitable for delay testing. Because, even with a very high-speed tester it is not always possible to have a timing accuracy comparable to the internal speed of a chip. A solution to this problem is to reduce the reliance on external testers by using Built In Self Test (BIST) techniques. Hence our third proposal is to develop a hierarchical BIST scheme for delay faults based on the method described in this thesis.

ACKNOWLEDGEMENTS

I express sincere appreciation to my supervisor Professor Hideo Fujiwara who at first taught me the basic concepts of design for testability with great care and then constantly guided me using his wisdom and experience throughout the course of my study at Nara Institute of Science and Technology (NAIST).

I would also like to appreciate Professor Katsumasa Watanabe of NAIST for his valuable comments and important suggestions concerning this thesis.

I am very much grateful to Associate Professor Michiko Inoue of our laboratory for her comments and suggestions at various stages of this research work.

Special thanks goes to Dr. Satoshi Ohtake, Assistant Professor of our Laboratory for his continuous co-operation and suggestions and more. I also thank Assistant Professor Tomokazu Yoneda of our Laboratory for friendly discussion.

I would also like to express my gratitude to Professor Toshimitsu Masuzawa of Osaka University for providing me with valuable information on RTL circuits and hierarchical testability.

I would also like to express my gratitude to Dr. Toshinori Hosokawa and Dr. Hiroshi Date of STARC (Semiconductor Technology Academic Research Center) for their valuable discussion and comments.

I also wish to thank Dr. Hiroki Wada of Hitachi Ltd. for helpful discussions while he was a PhD student of NAIST. I extend my gratitude to Mr. Sintharo Nagai and Tsuyoshi Iwagaki, PhD students of our laboratory for helping me to run the CAD tools and so on. I would also like to thank Dr. Satoshige Ukena, Mr. Kenichi Yamaguchi Ms. Chiiho Sano and Mr. Shunjiro Miwa, former and present students of Fujiwara laboratory for their help in leading my life as a foreigner in Japan.

I would like to extend my gratitude to Ms. Yoshiko Hirayama and Ms. Reiko Ohbayashi for helping me in different official correspondence.

Thanks are due to all other members of Fujiwara Laboratory for making my stay, within the confines of the laboratory, pleasurable.

I am grateful to the Ministry of Education, Culture, Sports, Science and Technology, Government of Japan for supporting me with Monbukagakusho scholarship throughout the course of my study.

This work was sponsored in part by NEDO (New Energy and Industrial Technology Development Organization) through the contract with STARC and supported by JSPS (Japan Society for the Promotion of Science) under the Grant-in-Aid for Scientific Research and by Foundation of Nara Institute of Science and Technology under the grant for activity of education and research. I express my gratitude to all these sponsoring organizations.

On a personal level, I wish to acknowledge my gratitude to my parents Altaf and Amena, brother Alam and sisters Nisa and Nahar for their endless love and counsel over long distance telephone calls. I also remember my wife Aziza and daughter Rafia for their company and encouragement.

References

[1] Angela Krstic and Kwang-Ting (Tim) Cheng, *Delay Fault Testing for VLSI Circuits*, Kluwer Academic Publishers, 1998.

[2] Rajsuman R. 1995. Iddq Testing for CMOS VLSI. ARTECH HOUSE, INC. 685 Canton Street, Norwood, MA 02062

[3] Angela Krstic and Kwang-Ting (Tim) Cheng, Resynthesis of Combinational Circuits for Path Count Reduction and for Path Delay Fault Testability. Journal of Electronic Testing: Theory and Application, pp. 43-54, August 1997

[4] S. Devdas and K. Kuetzer. Synthesis of Robust Delay-Fault Testable Circuits: Theory. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 11(1), pp. 87-101, January 1992

[5] S. Kundu, S. M. Reddy and N. K. Jha Design of Robustly Combinational Logic Circuits, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 10(8), pp.1036-1048, August 1991

[6] K. Roy, K. De, J. A. Abraham and S. Lusky. Synthesis of Delay Fault testable combinational logic. Proceedings of IEEE/ACM International Conference on Computer-Aided Design, pp. 418-421, November 1989

[7] T. J. Chakraborty, V. D. Agrawal and M. L. Bushnell, "Delay fault models and test generation for random logic sequential circuits," Proc. Of 29th Design Automation Conference, 1992, pp. 165-172.

[8] I. Pomeranz and S. M. Reddy, "At-speed delay testing of synchronous sequential circuits," Proc. Of 29th Design Automation Conference, 1992, pp. 177-181.

[9] S. Devdas, "Delay test generation for synchronous sequential circuits", Proceedings of International Test conf. 1989, pp. 144-152.

[10] J. Savir, "On board-side delay testing", Proc. of 12th IEEE VLSI Test Symposium, pp. 284-290, 1994.

[11] K.-T. Cheng, S. Devdas and K. Keutzer, "Delay-fault test generation and synthesis for testability under a standard scan design methodology", IEEE Transactions on Computer Aided Design Of Integrated Circuits and Systems, Vol. 12 No.8, pp. 1217-1231, 1993.

[12] K.-T. Cheng. Transition Fault Testing for Sequential Circuits, IEEE transactions on Computer-Aided Design of Integrated Circuits and Systems, 12(12), pp.1971-1983, December 1993.

[13] V. S. Iyengar, B. K. Rosen and I. Spillinger. Delay Test Generation 1- Concepts and Coverage Metrics. Proceedings of IEEE International Test conf., pp. 857-866, September 1998.

[14] V. S. Iyengar, B. K. Rosen and I. Spillinger. Delay Test Generation 2-Algebra and Algorithms. Proceedings of IEEE International Test conf., pp. 867-876, September 1998.

[15] A. K. Majhi , J. Jakob, L. M. Patnaik and V. D. Agrawal. On Test Coverage of Path Delay Faults. Proceedings of 9th International Conference on VLSI Design, pp. 418-421, January 1996.

[16] G. L. Smith. Model for Delay Faults Based upon Paths. Proceedings of IEEE International Test Conference, pp. 342-349, November 1985

[17] K. Haragu, J. H. Patel and V. D. Agrawal. Segment Delay Faults: Anew Fault Model. Proceedings of 14th IEEE VLSI Test Symposium, pp. 32-39, May 1996

[18] B. I. Dervisoglu and G. E. Stong, "Design for testability: using scan path techniques for path-delay test and measurement", Proc. of Int. Test Conf., pp. 365-374,1991.

[19] S. Dasgupta, R. G. Walther and T. W. Williams, “An enhancement to LSSD and some application of LSSD in reliability, availability and serviceability”, Proc. of Fault Tolerant Computing Symp., FTCS-11., pp. 32-34, 1981.

[20] S. Ohtake, H. Wada, T. Masuzawa and H. Fujiwara, "A non-scan DFT method at register-transfer level to achieve complete Fault efficiency", Proc. of the ASP-DAC, pp. 599-604, 2000.

[21] I. Ghosh, A. Raghunathan and N. K. Jha, "Design for hierarchical testability of RTL circuits obtained by Behavioral Synthesis", IEEE Tran. on Computer-Aided Design of Integrated Circuits and Systems, Vol. 16, No.9, pp. 1001-1014, 1997.

[22] Wei-Cheng Lai, Angela Krstic and Kwang-Ting (Tim) Cheng, "Functionally testable path delay faults on a microprocessor", IEEE Design & Test of Computers, pp. 6-14, 2000.

[23] H. Wada, T. Masuzawa, K. K. Saluja and H. Fujiwara, Design for strong testability of RTL data paths to provide complete fault efficiency", Proc. Of Int. Conf. on VLSI Design, pp.300-305, 2000.