CORE
Metadata, citation and similar papers at core.ac.uk
Provided by Document Repository

NAIST-IS-MT1051208

Master's Thesis

Acceleration of Seed Ordering and Selection for High Quality Delay Test

Ratna Aisuwarya

August 16, 2012

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

A Master'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
MASTER of ENGINEERING

Ratna Aisuwarya

Thesis Committee:

Professor Michiko Inoue	(Supervisor)
Professor Yasuhiko Nakashima	(Co-supervisor)
Professor Kazumi Hatayama	(Co-supervisor)
Assistant Professor Tomokazu Yoneda	(Co-supervisor)

Acceleration of Seed Ordering and Selection for High Quality Delay Test*

Ratna Aisuwarya

Abstract

Seed ordering and selection is a key technique to provide high-test quality with limited resources in Built-In Self Test (BIST) environment. We present a hard-to-detect delay fault selection method to accelerate the computation time in seed ordering and selection processes. This selection method can be used to restrict faults for test generation executed in an early stage in seed ordering and selection processes, and reduce a test pattern count and therefore a computation time. We evaluate the impact of the selection method both in deterministic BIST, where one test pattern is decoded from one seed, and mixed-mode BIST, where one seed is expanded to two or more patterns. The statistical delay quality level (SDQL) is adopted as test quality measure, to represent ability to detect small delay defects (SDDs). Experimental results show that our proposed method can significantly reduce computation time from 28% to 63% and base set seed counts from 21% to 67% while slightly sacrificing test quality.

Keywords:

seed ordering, BIST, delay test, SDQL

---

* Master's Thesis, Department of Information Processing, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-MT1051208, August 16, 2012.

Contents

1. Introduction **1**

2. Preliminaries **3**

2.1 Delay Test . . . 3

2.1.1 Transition Delay Fault Model (TDF) . . . 3

2.1.2 Transition Delay Fault Testing . . . 5

2.1.3 Small delay Defects (SDDs) . . . 5

2.2 Automatic Test Pattern Generation (ATPG) . . . 7

2.2.1 Timing-aware ATPG . . . 8

2.2.2 N-Detect ATPG . . . 9

2.3 Built-In Self-Test (BIST) . . . 9

2.3.1 Scan-based BIST . . . 10

2.3.2 LFSR-reseeding-based BIST . . . 10

2.4 Statistical Delay Quality Model (SDQM) . . . 12

2.5 Seed Ordering and Selection . . . 14

3. Hard-to-Detect fault Selection **15**

3.1 Test Pattern Generation . . . 16

3.2 Fault Selection . . . 17

4. Experimental Results **19**

4.1 Base Seed Set Generation . . . 20

4.2 Delay Test Quality . . . 21

4.2.1 Deterministic BIST . . . 21

4.2.2 Mixed-mode BIST . . . 23

4.3 Computation Time . . . 23

5. Conclusions **27**

Acknowledgements **28**

References **29**

List of Figures

1 Transition Delay Fault Example 4
2 Waveform of LOC Method 6
3 Fault Propagation Path on a delay Fault 7
4 Transition fault and timing-aware ATPG 8
5 BIST Architecture 10
6 Scan-based BIST Architecture 11
7 An LFSR connected to a scan chain 12
8 Timing Concept of Delay Defect Size 13
9 Delay Defect Distribution function and SDQL 13
10 Seed Ordering and Selection Flow of The Previous Work 15
11 Hard-to-detect Fault Selection Flowchart 16
12 Seed Ordering Flow of Previous Work and Proposed Method . . . 17
13 Fault Coverage and SDQL Transition 22
14 Computation Time for different circuits 26

List of Tables

1 Characteristics of Benchmark Circuits 19
2 Fault base set for Generation Pattern 20
3 Pattern Generation Results for Timing-aware ATPG 20
4 Seed Generation Results for Timing-aware ATPG 21
5 Seed Ordering Results for Different Mixed-mode BIST 24
6 FC and SDQL Loss from Previous Work in Mixed-mode BIST . . 24
7 Computation Time for Fault Selection 25
8 Computation Time for all Processes 26

1. Introduction

New challenges emerge for testing field engineering, as VLSI technologies are scale down to nanometer. This leads to the increasing probability of timing-related defects to occur. As a result, the stuck-at test cannot ensure high quality level of chips, and at-speed test is needed to cover these timing-related defects. One of the defects is small delay defects (SDDs), which is caused by resistive opens, resistive shorts and some other process variations might significantly impact the overall product quality especially for the 45nm technology and below if such defect is a critical path. Thus, serious consideration is growing rapidly in targeting these SDDs to minimize the test escape rate as well as improve defect coverage in some extent of in-field reliability [1]. In order to detect SDDs, propagation through long path is required. Conventional Automatic test pattern generation (ATPG) tools tend to generate test pattern that target the fault along the path which is the short path [2].

Therefore, commercial timing-aware ATPG tools, e.g., Synopsys TetraMAX SDD mode have been developed to overcome the lack of coverage of conventional timing-unaware ATPGs [3]. In spite of the ability to activates each undetected fault along paths with minimal timing slack, they result in significantly large CPU runtime and pattern count. The increasing pattern count is not practical to be applied for the testing environment with limited resources. To avoid the high cost of testing resources, novel methods are required to reduce the pattern count but still capable of targeting SDDs effectively.

Seed ordering and selection can be an effective method to reduce the storage for seeds [4]. LFSR reseeding based BIST was first introduced by Koenemann in [5] as a technique for coding test patterns into pseudo-random pattern generators (PRPGs). In terms of targeting SDDs, the proposed method in [6] considered the test compression for seed selection problem in LFSR-reseeding-based BIST, however it only utilized one seed for one pattern, or deterministic pattern in the compression method.

Since, we can apply some pseudo-random patterns combined with deterministic patterns (mixed-mode BIST) more seeds might be reduced and there is a chance that the patterns increase the defect coverage of SDDs. Recent seed ordering and selection method, proposed by Yoneda et al. [7] selects seeds based

on the gain in the sum of the longest path lengths sensitized by seeds, which is correlated with statistical delay quality level (SDQL). Experimental results show that this method can obtain significant seed count reduction under several mixed-mode BIST approaches, yet still considered to be time consuming, since it generated the entire faults into test patterns and later encoded into seeds. This is why we need another solution considered test time constraints as a compromise between the SDQL and the resources.

In this research, our purpose is a hard-to-detect fault selection method to reduce the computation time in seed ordering and selection process. This selection method can be used to restrict faults for test generation when it is impractical to target all delay faults that result in large test pattern count and long computation time. Especially for a much larger and complex circuit, It will be very useful if we put more consideration on resources, such as memory and storage.

To avoid the excessive usage of resources, in our proposed method, target faults are restricted based on the number of test patterns that detect each faults for a given test set. We examine three types of hard-to-detect fault selection method, *select-1*, *select-3*, and *select-5*, where *select-n* means the faults detected at most *n-times* are selected. We use seed ordering and selection method in [7] and evaluate the delay test quality based on SDQL as in [8]. The quality of seed generated from the base set of our proposed method will be compared to the previous work [7] in deterministic and mixed-mode BIST environments.

This thesis is organized into the following sections: section 2 explains the necessary theories and research topics that are related to our research. Section 3 gives the details of hard-to-detect fault selection method. Chapter 4 describes experimental results of the proposed method. The evaluation will be divided into three parts; base seed set, delay test quality, and computation time. Finally section 5 conclude this thesis.

2. Preliminaries

This section gives the necessary theories and research topics that are related to this thesis. Subsection 2.1 explains the basics of Delay Test and focus on transition delay fault model (TDF) and Small Delay Defects (SDDs). It briefly describes TDF model, then the techniques for testing TDFs. Additionally, as technology continues to scale down, more delay variations are occur, which can affect the performance of a circuit. One of those delay variations is caused by small delay defects (SDDs). This section will mention about SDDs, which are grown serious consideration to help increase defect coverage and hence test quality. Subsection 2.2 explains the automatic test pattern generation (ATPG) and it describes several types of ATPG that are used in our experiments. Subsection 2.3 gives an overview about build-in-self-test (BIST), one of Design for Testability (DFT) techniques in literature. This subsection only focuses on the BIST methodologies that are the most commonly used in industry. LFSR-reseeding-based BIST also introduced. Subsection 2.4 discusses some related methods for seed ordering and statistical delay quality level (SDQL). Finally subsection 2.5 explain about seed ordering and selection method in LFSR-reseeding-based BIST.

2.1 Delay Test

Delay faults cause errors in the functioning of a circuit based on its timing. They are caused by the rising or falling delay of the signals in the gates, as well as, the propagation delay of interconnects between the gates. The delay of the circuit has to be carefully evaluated to avoid such errors in the function of the circuit. Tests have to be generated specially to account for these faults.

2.1.1 Transition Delay Fault Model (TDF)

Transition delay fault model [9] assumes that the delay fault affects only one gate in the circuit. There are two transition faults associated with each gate; a *slow-to-rise* fault and a *slow-to-fall* fault. It is assumed that in the fault-free circuit each gate has some nominal delay. Delay faults result in an increase or decrease of this delay. Under the transition fault model, the extra delay caused by the fault is assumed to be large enough to prevent the transition from reaching any

primary output at the time of observation. In other words, the delay fault can be observed independent of whether the transition propagates through a long or a short path to any primary output.

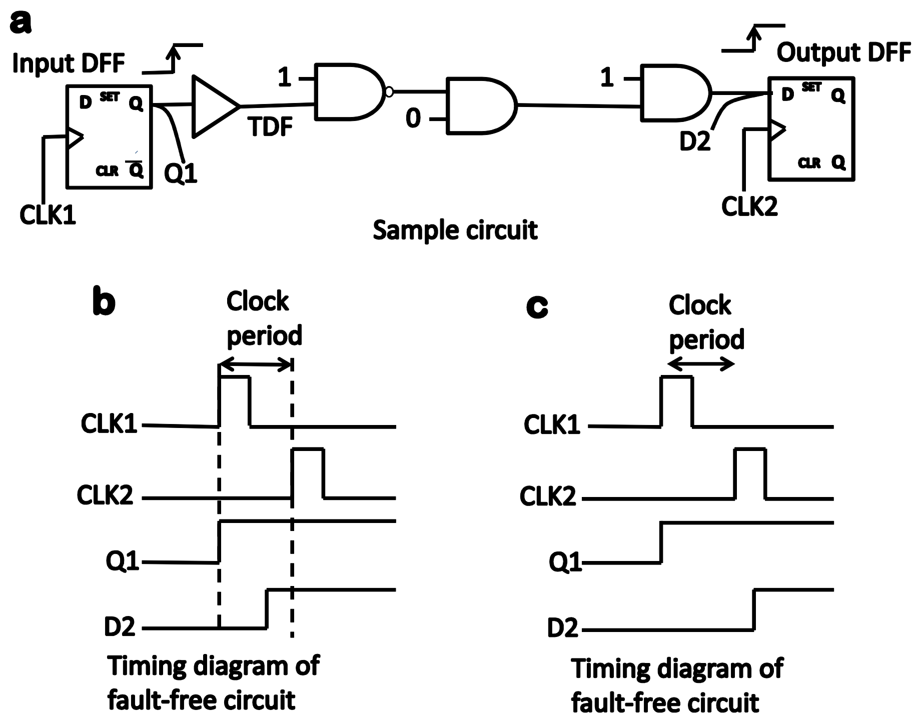


Figure 1. Transition Delay Fault Example

Figure 1 shows TDF's effected signal propagation on a circuit. In the example, the input flip-flop (shown as "input DFF") launches a rise-transition on its output pin $Q1$ at the rising edge of $CLK1$. Then the rise-transition at $Q1$ is propagated to the output flip-flop (shown as *"Output DFF").* If the circuit is fault-free, the rising transition arrive at *D2* before the capture clock on the output DFF *"CLK2"*, and the timing diagram should be similar to Fig.1.(b). However, if there is a TDF in the circuit as shown in Fig.1.(a), the signal propagation is slowed down and the transition exceed the specified clock period as shown in Fig.1.(c). As a result, the output DFF is not able to capture the correct value at *D2* at the rising edge of *CLK2*. In other words, the TDF causes the circuit to fail and alter the performance of the circuit.

The main advantage of TDF model is that the number of faults in the circuit is linear in terms of the number of gates. Also, the stuck-at-fault test generation and fault simulation tools can be easily modified for handling transition faults. On the other hand, the expectation that the delay fault is large enough for the effect to propagate through any path passing through the fault site might not be realistic because short paths may have a large slack. A delay defect can affect more than one gate and even though none of the individual delay faults is large enough to affect the performance of the circuit, several faults can together result in a performance degradation.

2.1.2 Transition Delay Fault Testing

To detect a transition delay fault, two input vectors V = *v1,v2* are applied. The first vector, *v1*, initializes the circuit, while the second vector, *v2*, activates the fault and propagates its effect to some primary output. Vector *v2* can be found using stuck-at-fault test generation tools. For example, for testing a *slow-to-rise* transition, the first vector initializes the fault site to 0, and the second vector is a test for stuck-at 0 fault at the fault site. A transition delay fault is considered detected if a transition occurs at the fault site and sensitized path extends from the fault site to some primary output.

Launch-off-capture(LOC) or *broadside* method [10] is one of transition fault pattern generation methods. Figure 2 shows the waveform of LOC method. In LOC, once the test pattern shifted in, the scan enable (SEN) signal transition from 1 to 0. In the LOC method, the launch cycle is separated from the shift operation. First pattern is applied and the circuit under test (CUT) is set to an initialized state at the end of shift-in mode, and at lauch clock the second pattern is applied. Launch and capture clocks are applied at a system speed, then the SEN signal is raised prior to shifting out.

2.1.3 Small delay Defects (SDDs)

The delay fault model covers many physical defects on real silicon, including the effects of process variations, temperature, on-chip power supply noise, crosstalk, resistive opens, resistive shorts, etc [1]. One of timing defect variations introduces a small amount of extra delay to the design, which is called Small Delay Defects

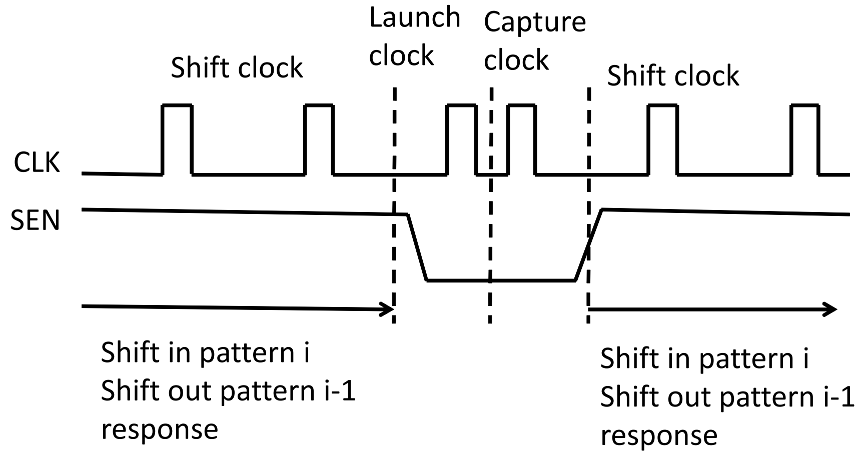


Figure 2. Waveform of LOC Method

(SDDs). Because of their small size relative to the timing margins allowed by the maximum operating frequency of a design, SDDs were not seriously considered in the testing of designs at higher technology nodes.

Although the delay introduced by each SDD is small, the overall impact can be significant if the sensitized path is long or critical path, especially when technology scales to 45nm and below [1]. As the shrinking of technology geometries and increasing of operating frequency of the design continues, the available timing slack becomes smaller. Therefore, SDDs have a good chance to add enough additional delay. Figure 3 shows three possible paths to detect a delay fault. TDF pattern generation typically generates a test pattern that activates fault along the path with the largest timing slack (path 3 in the figure). This pattern doesn't cover smaller delay defects in path 1 and path 2. In this case path 1 has the smallest slack.

TDF model has been widely used in industry, however, the TDF generation ignores the actual delays through the fault activation and propagation paths, and is more likely to detect a fault through a shorter path. As a result, the generated test set may not be capable of detecting SDDs. Therefore, most research on SDDs has been aimed at finding the longest path in a circuit. Due to the growing interest in SDDs, commercial timing-aware automatic test patterns generation (ATPG) tools were introduced recently, e.g., Mentor Graphics FastScan, Cadence

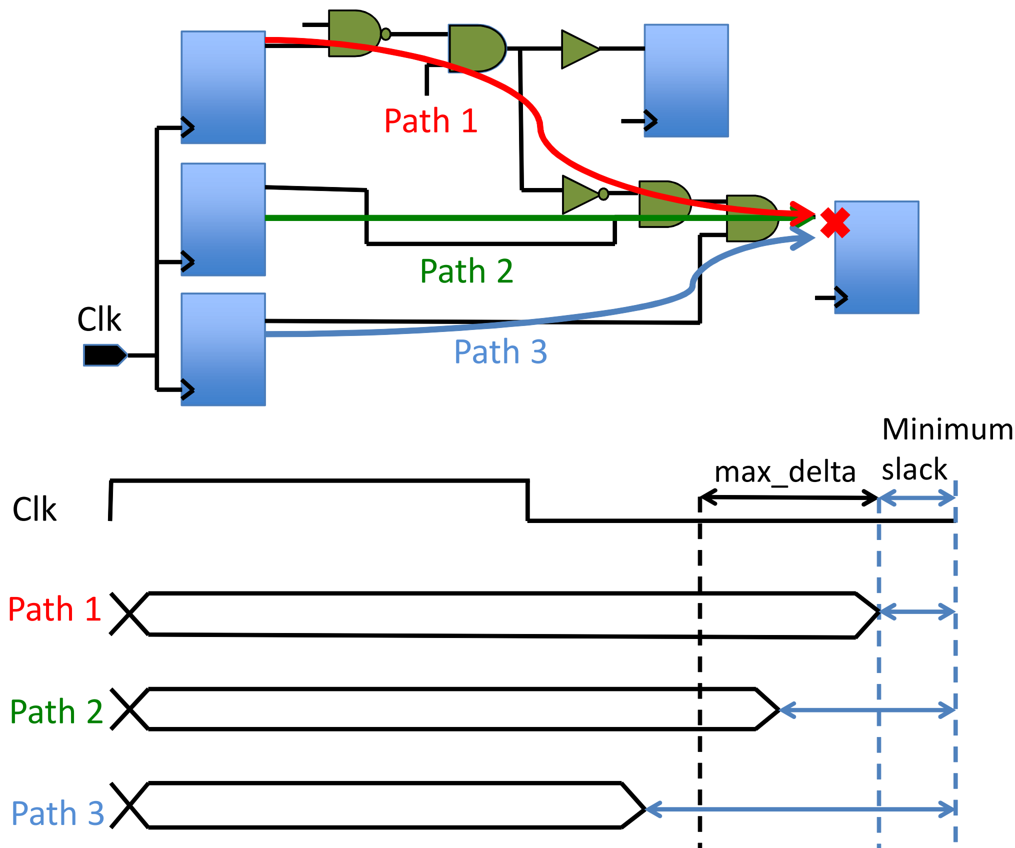


Figure 3. Fault Propagation Path on a delay Fault

Encounter Test, and Synopsys Tetramax tools [3]. However, timing-aware ATPG tools need longer CPU times and produce high pattern counts. Therefore, test data compression is needed especially for the field test with limited resources.

2.2 Automatic Test Pattern Generation (ATPG)

Automatic test pattern generation (ATPG) is the process of automatically generating a set of test patterns for detecting a specific group of faults. The inputs of the ATPG procedure are design data (e.g., netlist), fault list (specifying what faults are targeted), test protocol and test constraints, and the output is a set of test patterns. The test patterns are then applied to the design for fault detection. If a fault can be detected by the input test patterns, it is called a detected fault. Otherwise, it is called an undetected fault.

2.2.1 Timing-aware ATPG

Timing-aware ATPG is an improved version of transition fault, with timing information in-form of SDD. The faults are targeted similar to transition fault, i.e. every node in the design is a fault site. In transition faults, the tool launches and captures from randomly selected path. In contrast, timing-aware ATPG tool tries to launch from the longest path and captures based on the timing information provided in the loaded SDD file. Figure 4 explain the difference between transition fault and timing-aware ATPG.

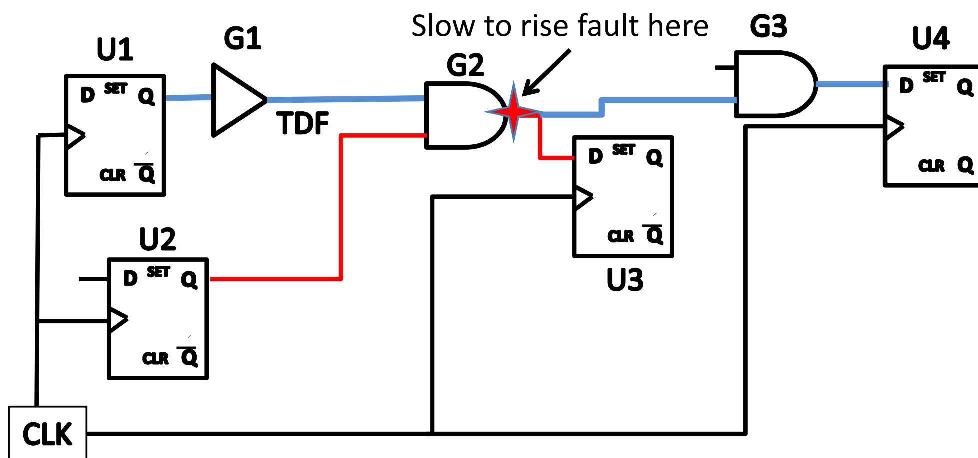


Figure 4. Transition fault and timing-aware ATPG

In Figure 4, there are four example paths to illustrate how timing-aware ATPG operates:

- R1: 4ns (U1-G1-G2-G3-U4)
- R2: 3ns (U1-G1-G2-U3)
- R3: 3ns (U2-G2-G3-U4)
- R4: 2ns (U2-G2-U3)

The path used for testing transition fault at the output of AND gate G2 is R4, which is just 2ns. The same fault is now being targeted with a path

R1, which is 4ns in timing-aware ATPG. To understand the impact of timing-aware ATPG, let us assume that the clock period in Figure 4 is 5ns and during manufacturing a small delay defect of 1.5ns introduced in output of G2. This small delay defect can be detected with timing-aware ATPG approach, but will escape the transition testing. The disadvantage of timing-aware ATPG is that it needs more time during fault simulation as the ATPG tools needs to calculate timing to select the right path for testing. This results in higher computation time. Secondly, timing-aware ATPG needs more pattern to achieve the same transition fault coverage, though timing-aware ATPG gets better delay coverage, which means better coverage for small delay defects.

2.2.2 N-Detect ATPG

In order to enhance the quality of a test set, and increase the coverage of all possible defects, we may generate a test set that achieves multiple detections of every fault under a given fault model. A fault is detected multiple times if it is detected with different vectors. By propagating the fault effect different ways, it is hoped that any defect close to a target fault will have an increased change of being detected.

In an n-detect setup, each fault must be targeted multiple times by an ATPG. In other words, all vectors generated that could detect a target fault are marked, and a fault is removed from further consideration when it has been detected n times. It has been shown that the size of an n-detect test set grows approximately linearly with respect to n [13].

2.3 Built-In Self-Test (BIST)

Build-in self-test (BIST) is a design for testability (DFT) technique that targets at detecting faulty components in a system by incorporating the test logic on-chip. The BIST has become a promising solution to VLSI testing problems and has been widely used in industry. Figure 5 shows a typical BIST hardware structure. In BIST, a test pattern generator (TPG) is used to generate test patterns and apply them to the circuit under test (CUT). The output response from the CUT is then compared with the reference signature stored in the ROM during BIST.

The entire process is controlled by a BIST controller.

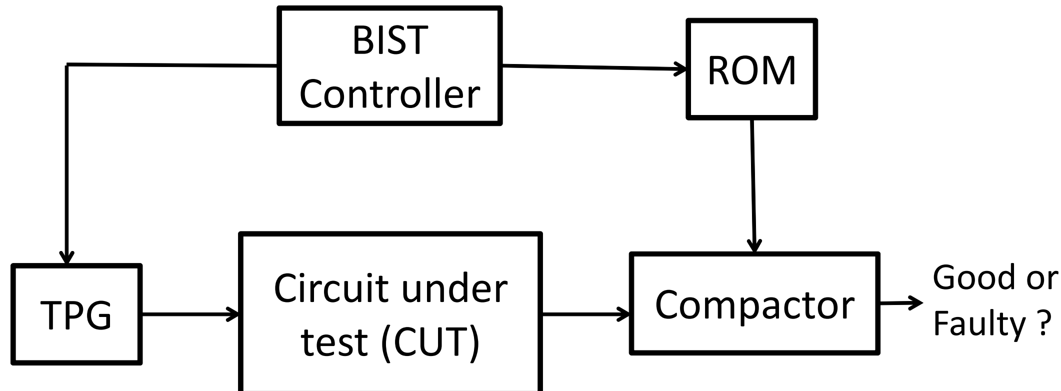


Figure 5. BIST Architecture

2.3.1 Scan-based BIST

Scan-based BIST is a type of DFT that allows test designer to design and add TPG, i.e. LFSR, to the scan architecture to generate pseudo-random patterns. The pseudo-random patterns are serially loaded into each scan chain of the CUT (Fig. 6). Response compactor is used for compacting the test responses. A scan-based BIST architecture requires long sequences of pseudo-random patterns in order to achieve acceptable fault coverage. The application of long test sequences takes a large amount of time, which limits the use of pseudo-random scan-BIST methods for field testing.

2.3.2 LFSR-reseeding-based BIST

In BIST, deterministic patterns are often encoded into seeds that are loaded into the LFSR used as pseudo-random pattern generation (PRPG) and then expanded into the desired patterns in the scan chains. A seed is an initial state for the LFSR which, computed by solving a system of linear equations based on the feedback polynomial of the LFSR [5]. Instead of storing each full test patterns on the tester, a much smaller LFSR seed is stored instead. Since, the seeds are much smaller than the full test patterns, the test data storage and other resources (memory or hardware) requirements for testing can be reduced.

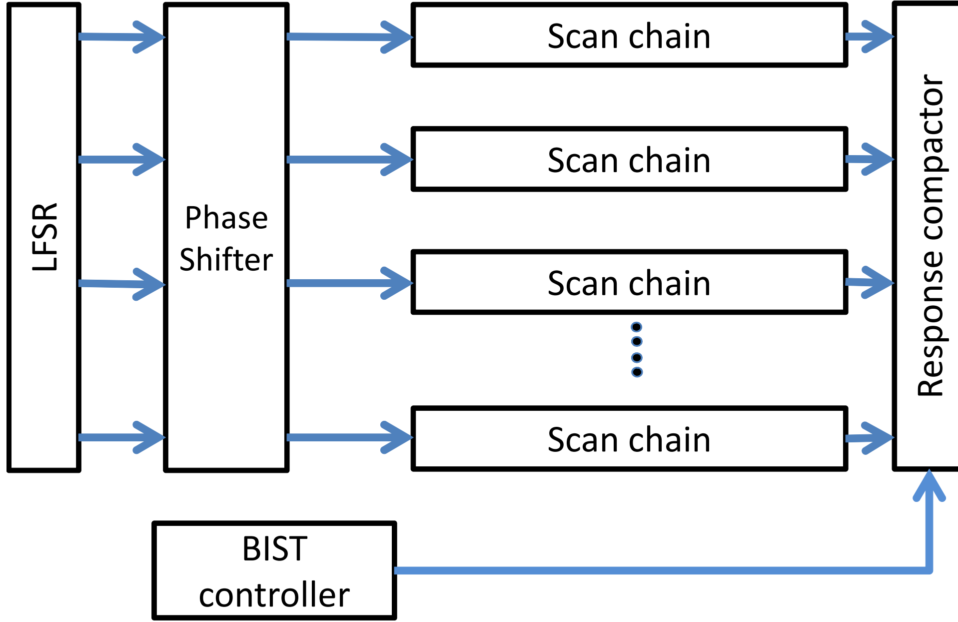


Figure 6. Scan-based BIST Architecture

When the LFSR is placed in this initial state it is expanded into a precomputed test pattern in the scan chains after *(m)* cycles, where *(m)* is the length of the longest scan chain. That is, it can produce the deterministic patterns. Reseeding refers to reinitializing the LFSR with a new seed. If the LFSR runs for another m cycles, another pseudo-random pattern is loaded into the scan chains. It is used to improve the fault coverage with pseudo-random patterns (mixed-mode BIST), since there is a chance that these patterns may detect more SDDs during testing.

Take as an example the LFSR used as a PRPG in Figure 7 for a single scan chain of 10 flip-flops (FFs). Where an LFSR consist of 4 FFs L_0, L_1, L_2, L_3, and feedback loops with an XOR gate; and a scan chain consists of 10 FFs S_0, S_1, ..., S_9. By initializing the LFSR at the state (0111) and running the clock for 10 clock cycles, the pattern (0011010111) will end up in the scan chain.

By solving a linear system of equations, seeds are encoded into deterministic test patterns, which is an algebraic representation of the linear expansion of the LFSR into the scan chains' flip-flops. In this thesis, we consider a mixed-mode BIST technique where each seed s_i is expanded into d_i patterns. The first pattern would be deterministic pattern and the remaining $d_i - 1$ patterns are

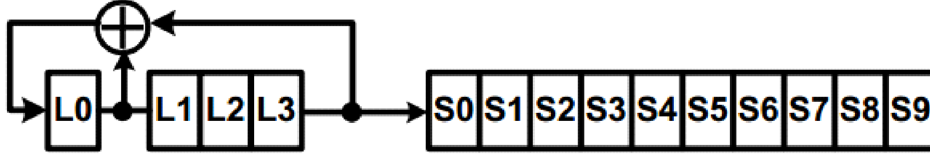


Figure 7. An LFSR connected to a scan chain

pseudo-random patterns.

2.4 Statistical Delay Quality Model (SDQM)

Many delay fault models were proposed to improve test pattern effectiveness. The transition delay fault model considers the propagation of lumped delay defects by logical transition to the observation pins or flip-flops. It is widely used because of its high fault coverage, but it cannot detect defects causing delays that are smaller than the test timing. A statistical delay quality model (SDQM) that reflects fabrication process quality, delay margin of design, and test timing accuracy have been proposed by Sato et al [8]. This model evaluate test quality based on delay defect distribution function.

The SDQM considers rising and falling delay faults on each of input and output pins of each gate. Though the number of faults is the same as transition faults, a delay defect size is associated with each fault. Figure 8 shows a concept of delay defect sizes that should be detected and can be detected by a given test set.

Let f be a fault, and let L_A and L_B be the lengths of the longest true path passing through f and the longest path passing through f that is actually sensitized by a given test set, respectively. Let T_{MC} and T_C be system clock timing and test timing, respectively. the difference $T^f_{mgn} = T_{MC} - L_A$ is the minimum delay defect size that can affect system behavior and therefore should be detected. the difference $T_{det} = T_c - L_b$ is the minimum delay defect size that can be actually detected by a given test set. Let N be the total number of faults and $F(t)$ is a delay defect distribution function. The statistical delay quality level (SDQL) represents the amount of delay defects escaped to be detected by the test set, can be expressed by:

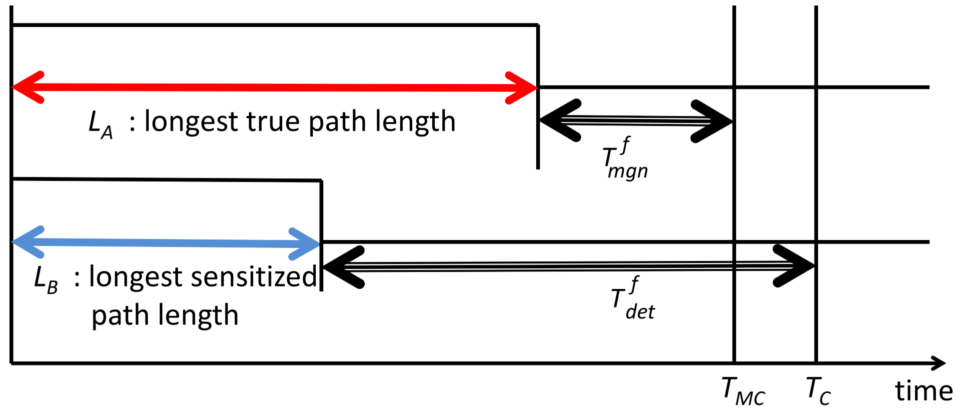


Figure 8. Timing Concept of Delay Defect Size

$$SDQL = \sum_{f \in N} \int_{T^f_{mgn}}^{T^f_{det}} F(t)dt \tag{1}$$

A shadow area in Figure 9 shows an amount of delay defect for one fault escaped during test. Therefore, smaller SDQL means better delay test quality.

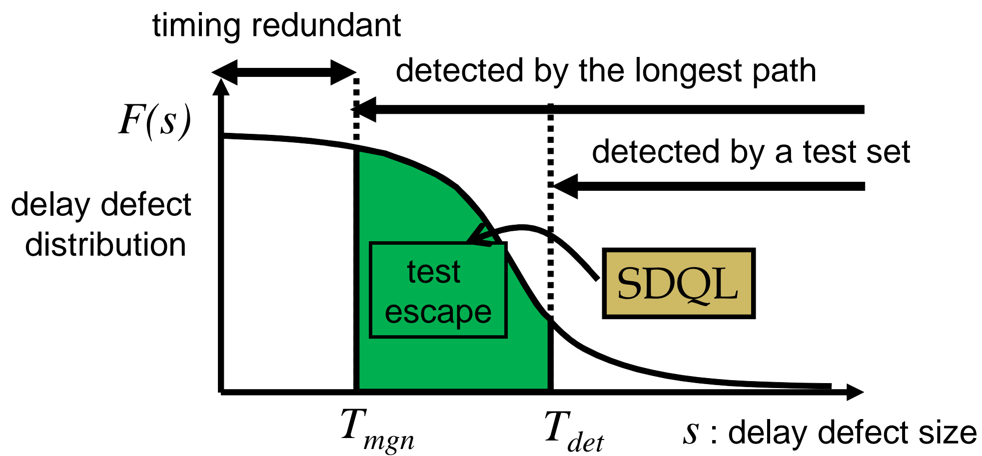


Figure 9. Delay Defect Distribution function and SDQL

2.5 Seed Ordering and Selection

In this subsection, seed ordering and selection technique are introduced. Seed ordering methods can be effective to reduce test data volume. Since, the seeds will be ordered based on the effectiveness. There is a chance that some of seeds may not need to be loaded, because the target faults are already detected by the pseudo-random patterns.

We adopt seed ordering and selection in LFSR-reseeding-based BIST proposed by Yoneda et al [7]. This method selects seeds based on the gain in the sum of the longest path lengths sensitized by seeds, which is highly correlated with SDQL. Consider a set S of $i + seeds$ that are selected. The $i - th$ seed is selected as follows. For each fault, the length of the longest path sensitized by a seed set calculated without delay defect distribution function $F(t)$ and fault simulation, once the length of the longest path sensitized by each seed is obtained. Assume L_f^S and l_f^s be the length of the longest path sensitized by the generated patterns from seed set S and a seed s for a fault f, respectively. Let L_s be the sum of the longest sensitized path lengths for the generated patterns from S. $L_{(S+s)}$ is obtained as follows.

$$L_{(S+s)} = L_S + \sum_{f \in N} max(l_f^s - l_f^S, 0) \tag{2}$$

The *gain* is defined as the sum of the longest sensitized path lengths when s is added to S as $Gain_{S,s}$.

$$Gain_{S,s} = L_{(S+s)} - L_S = \sum_{f \in N} max(l_f^s - l_f^S, 0) \tag{3}$$

Seed s with the largest $Gain_{S,s}$ is selected as the $i-th$ test pattern. Thus, the increase of the longest sensitized path length for a fault f implies the decrease of T_{det}^f, which implies the decrease of SDQL. This method apply fault simulation only one time to obtain a SDQL value and l_f^s for each seed in the base seed set (S_{base}). The first seed is selected based on SDQL value, seed with the minimum SDQL will be placed in the head of a sequence of seeds. Next, calculate $gain_{S,s}$ for each seed in S_{base} and order them based on the maximum $gain_{S,s}$ value. Notice that, since only the first seed is selected based on SDQL, therefore we can save some computation time in the ordering process.

3. Hard-to-Detect fault Selection

In the previous work [7] seed ordering and selection method for LFSR-reseeding-based have been proposed. From the processing flow in the figure 10, deterministic patterns are generated by ATPG tools, and the patterns are encoded into a seed set S_{base}. Seeds in S_{base} are ordered so that the SDQL improves by the maximum amount with the inclusion of each additional seed. In the selection process, if there is a seed count constraint k, select the top k seeds from the ordered sequence. If there is a SDQL constraint, select the seed from the top of the ordered sequence until the constraint is satisfied.

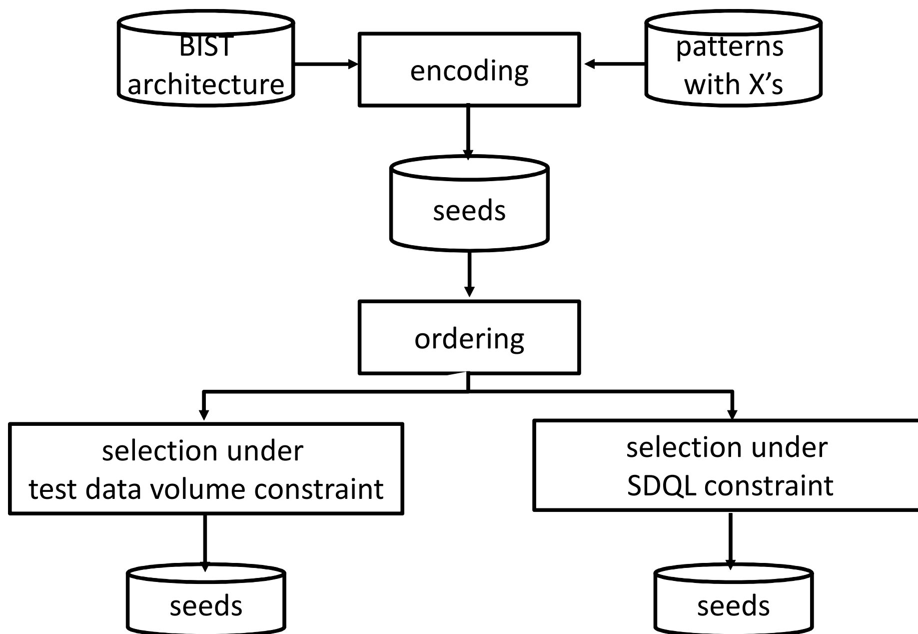


Figure 10. Seed Ordering and Selection Flow of The Previous Work

The previous work suffers in large pattern set and long computation time. These are our main concerns if we have limited resources in field test. Therefore, we proposed hard-to-detect fault selection method. Our purpose is to create smaller faults subset for test pattern generation, focused on *hard-to-detect* faults, which are faults with relatively few test patterns that can detect it. Moreover, test patterns for *hard-to-detect* faults are likely to detect a large number of *easier-to-detect* faults. In this thesis, we consider a way to characterize *hard-to-detect*

delay faults. The proposed method ranks faults according to their detection count. Based on this ranking, it is possible to select a subset of faults of the desired size as targets for test pattern generation. It is very important in order to save computation time on test pattern generation since, it wasted a lot of time in sensitized large number of faults. Our proposed method can be describe in figure 11. There are two important processes, test pattern generation and fault simulation to create subsets of hard-to-detect faults. We explain the details of our proposed method in the following subsections.

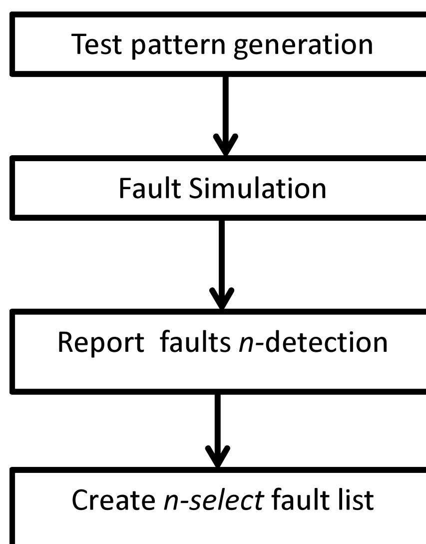


Figure 11. Hard-to-detect Fault Selection Flowchart

3.1 Test Pattern Generation

For test pattern generation for hard-to-detect fault selection in this thesis, we applied n-detect ATPG. Since, it is known that the main drawback of timing-aware ATPG is that they waste a lot of time operating on faults that do not contribute to SDD coverage resulting in a large number of test patterns. Experimental results [11][12] have demonstrated that timing-aware ATPGs will result in significantly larger computation time and pattern count. Furthermore, they seem ineffective in sensitizing large numbers of long paths.

Therefore, due to our purpose to accelerate computation time, for test pattern

generation we don't apply timing-aware ATPG, but n-detect ATPG instead. The n-detect ATPG can also be an effective method for SDD detection, even without timing information of the design. For each target fault, n-detect ATPG generates patterns trying to detect it n times through different paths. Furthermore, n-detect ATPG requires much lower computation time when compared with timing-aware ATPG.

To improve fault coverage in scan-based BIST, deterministic pattern generation is preferred than pseudo-random pattern. Since, in the next process we have to apply fault simulation to select hard-to-detect faults, therefore, an effective test patterns set is generated.

3.2 Fault Selection

In order to create *hard-to-detect* faults subset, fault simulation is applied after generating test patterns. By a given test patterns fault grading or fault simulation of those patterns is performed. fault simulation is set to detects a fault up to and including n times. This option allows fault detections to be active until the fault has been detected n times. We restrict the number of detection with very small n values, several subsets can be created. When a fault is detected up to n times, it is then placed in select-n subset. Furthermore, the *hard-to-detect* subset will be used in the timing-aware ATPG to generate patterns for targeting SDDs. Therefore it can generate faster since the fault list base is reduced.

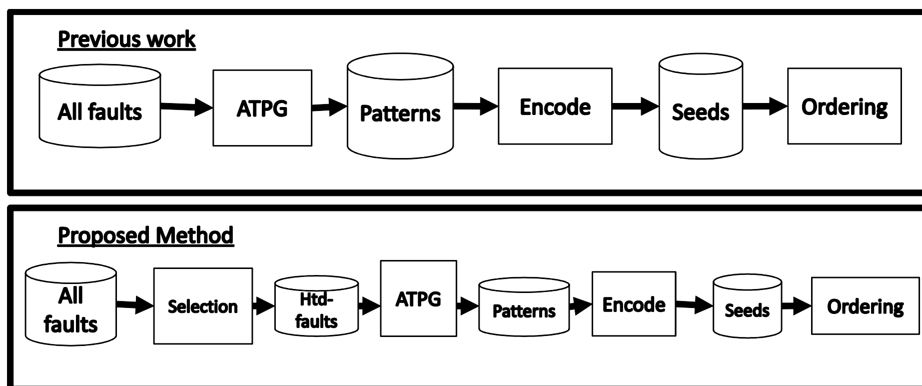


Figure 12. Seed Ordering Flow of Previous Work and Proposed Method

Figure 12 shows the comparison of seed ordering flow between the previous

work and our proposed method. We can observe that in previous work they try to generate all faults. Therefore, in timing-aware ATPG try to sensitize large number of long paths which is required very long computation time and a large pattern set. This large pattern set later will be encoded into seed, which is also large in size. Hence, this is the main reason why we apply fault *hard-to-detect* fault selection method to get smaller number of faults set in the earlier process before generating test pattern in timing-aware ATPG. So, we can accelerate all processes in pattern generation, encoding, and seed ordering.

4. Experimental Results

In order to evaluate the effectiveness of our proposed method, This section presents the results of experiments using several ITC'99 benchmark circuits. We compared our proposed method with previous work without fault selection as in [7]. To evaluate the results, we perform some simulation experiments. The results will be explained in this thesis by four difference parts. (1) we evaluate the base seed set generated by timing-aware ATPG; (2) delay test quality, in terms of fault coverage and SDQL with base seed set generated by timing-aware ATPG. Two different modes are used, deterministic BIST and mixed-mode BIST; (3) we also evaluate base seed set generated from different ATPG patterns; (4) Lastly, computation time, we observe the effect of our selection process to the total computation time.

The characteristics of the benchmarks circuits that we used are shown in the Table 1. Sysnopsis TetraMAX ATPG for small delay defect testing which targets a subset of the transition fault model were used in the experiments. TetraMAX will generate a specific set of transition fault tests that systematically try to find the longest paths. For the experiments, we specify the coefficients probability distribution function *F(t)* that is to be used in computing the SDQL.

$$F(t) = A.e^{-Bt} + C \tag{4}$$

The values of A, B, C is specified as follows, in this case we set $F(T_{MC})$=0.1 assumed that T_{MC} is the system clock timing of the circuit. To maintain the value, we calculate the above equation by given $A = 1$ and $C = 0$. Then we can get the calculated values for B as in Table 1. This Table show the characteristics of benchmark circuit that are used in the experiments.

Table 1. Characteristics of Benchmark Circuits

Circuit	#gates	#FFs	#faults	B in $F(t)$
b15	8,985	417	17,329	1.19
b17	2,776	1317	65,218	1.19
b18	79,400	3,020	172,403	0.71
b19	152,599	6,042	353,301	0.71

4.1 Base Seed Set Generation

Our proposed method started with the selection of the target faults for test pattern generation. Three different parameters are to specify hard-to-detect faults ($select - 1, select - 3, and, select - 5$). After fault simulation and selection process, hard-to-detect fault lists are obtained. Table 2 show that selection method significantly reduced the number of faults.

Table 2. Fault base set for Generation Pattern

Circuit	#Faults			
	Select-1	Select-3	Select-5	Previous
b15	3,930	6,423	7,245	17,329
b17	15,868	25,942	29,062	65,218
b18	39,265	60,812	67,453	172,403
b19	78,454	121,472	134,392	353,301

Test patterns with unspecified bits (X) are generated by timing-aware ATPG using the faults in table 2, then these patterns encoded into a base seed set. Pattern generation results and seed generation results are shown in Table 3 and Table 4 respectively, where #schains denotes the number of scan chains and #LFSR denotes the number of FFs in LFSR. For the base seed sets in the Table 4, we compared the proposed hard-to-detect fault selection method with the previous work) without fault selection. We can observe that the proposed method obtained significant reduction in the number of seeds compared to previous work up to 67,4%.

Table 3. Pattern Generation Results for Timing-aware ATPG

Circuit	BIST Architecture		#patterns				Reduction (%)		
	#LFSR	#schains	Select-1	Select-3	Select-5	Previous	Select-1	Select-3	Select-5
b15	96	8	490	543	568	727	32.6	25.3	21.9
b17	240	26	735	935	978	1,375	46.5	32.0	28.9
b18	384	60	1,479	1,690	1,760	3,293	55.1	48.7	46.6
b19	608	120	2,006	2,681	2,908	6,131	67.3	56.3	52.6

Table 4. Seed Generation Results for Timing-aware ATPG

Circuit	BIST Architecture		#seeds				Reduction (%)		
	#LFSR	#schains	Select-1	Select-3	Select-5	Previous	Select-1	Select-3	Select-5
b15	96	8	478	528	553	700	31.7	24.6	21.0
b17	240	26	706	891	931	1,319	46.5	32.4	29.4
b18	384	60	1,415	1,609	1,689	3,129	54.8	48.6	46.0
b19	608	120	1,906	2,560	2,784	5,850	67.4	56.2	52.4

4.2 Delay Test Quality

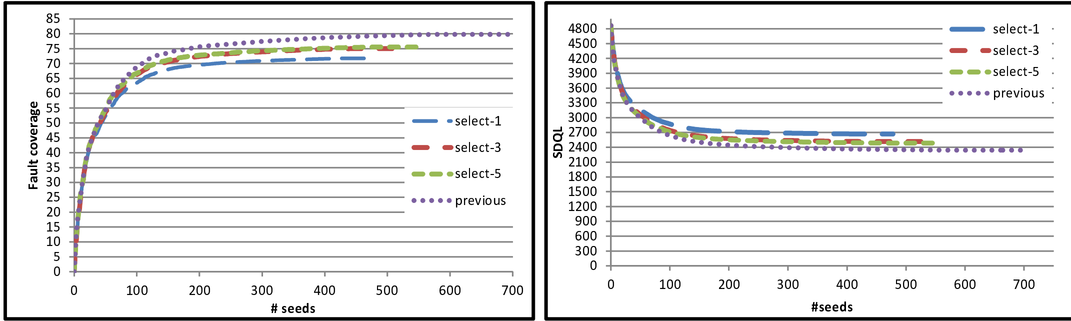
There are two important metrics in evaluating delay test quality of seed ordering in our experiments. Fault coverage (FC) and SDQL. FC can be calculated using following equation, where DT represents the number of detected faults, and TF denotes the number of total faults in the circuit under test (CUT).

$$FC = \frac{DT}{TF} \times 100\% \tag{5}$$

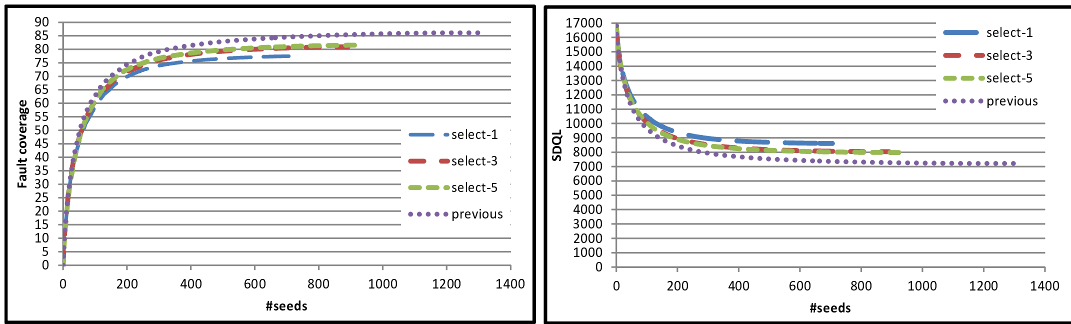
4.2.1 Deterministic BIST

As mentioned before, we evaluated the effectiveness of the proposed method using the base seed sets generated by BIST with d_i=1 for all seed s_i, which considered as deterministic pattern (one seed is expanded into one pattern). Figure 13 shows the fault coverage and SDQL transitions for different benchmark circuits respectively. From the figures we can observe that for only one deterministic pattern, seed generated from previous faults base set achieved higher fault coverage and lower SDQL compared to the proposed method. However, the previous method results in large seed counts with unnecessary seeds which have less contribution to SDQL in the base set set. If we are allowed to sacrifice SDQL a little, we can obtain significant reduction in seed counts with our proposed method. Furthermore, when a seed is loaded in the LFSR, some pseudo-random patterns can be applied and there is a possibility that these patterns will detect more faults. In this case large seed counts is not necessary.

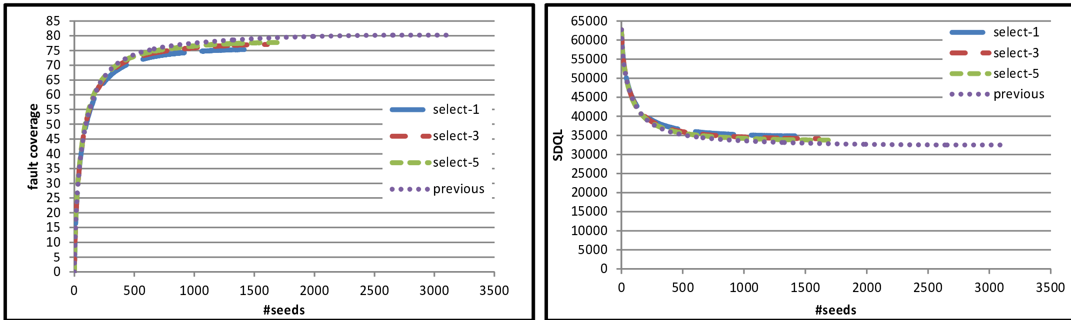
(a). b15



(b). b17



(c). b18



(d). b19

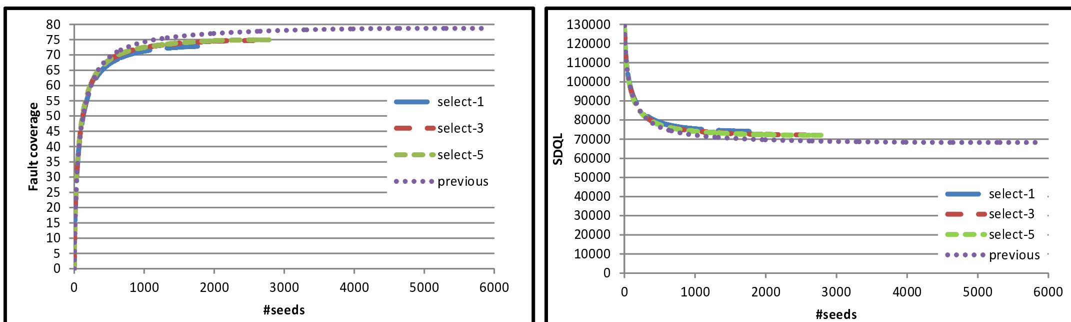


Figure 13. Fault Coverage and SDQL Transition

4.2.2 Mixed-mode BIST

In mixed-mode BIST contribution of pseudo-random patterns to delay test quality is evaluated. Three types of mixed-mode BIST approaches are applied.

- *Type I*: every seed s is expanded into d patterns, where d is set to 1, 2, 4, and 8.
- *Type II*: the first selected seed s_1 is expanded into d patterns, and 1 deterministic pattern is generated from the other seeds, where d is set to 1024, 2048, and 4096.
- *Type III*: the first two seeds s_1 and s_2 will be expanded to d patterns, and 1 deterministic patterns is generated from the other seeds, where d is set to 1024, 2048, and 4096.

Table 5 show seed ordering results of mixed-mode BIST for b18 and b19. We can observe that when d become large, SDQL value is decrease and the number of test patterns are increased. This also correlated to test application time. Furthermore, we compare the results between type I, type II and type III. In this case, type III generates more pseudo-random patterns compared to two other types. The results shows that the long pseudo-random patterns expanded from one seed are more effective than the very short expanded pattern for every seed in type I.

In our proposed method, we have to sacrifice SDQL depending on the selection types. For example if we choose $select - 1$ for $d = 1$ in b18 circuit, SDQL loss is 6.83% compare with *previous* type. However, if we chose $select - 5$, SDQL loss can be reduced to 3.79%. Moreover, if the reduction in seed counts is more concerned, $select - 1$ type can be the best option. Since, seed counts can be reduced by 31,7%, 24.6%, and 21% for $select - 1$, $select - 3$ and $select - 5$ respectively.

4.3 Computation Time

We evaluate the acceleration in computation time for our proposed method. In the experiments, two additional computation times are needed; computation time

Table 5. Seed Ordering Results for Different Mixed-mode BIST

Circuit	Type	d	Fault Coverage (%)				SDQL			
			Select-1	Select-3	Select-5	Previous	Select-1	Select-3	Select-5	Previous
b18	I	1	75.29	77.03	77.68	80.21	34,894.94	34,165.45	33,789.73	32,510.24
		2	75.63	77.3	77.89	80.33	34,735.76	34,030.25	33,665.68	32,428.76
		4	75.92	77.52	78.07	80.43	34,556.28	33,866.16	33,516.78	32,315.38
		8	76.17	77.71	78.2	80.54	34,390.90	33,731.80	33,389.95	32,203.47
	II	1024	75.62	77.2	77.85	80.26	34,685.55	34,012.80	33,640.54	32,426.86
		2048	75.74	77.3	77.94	80.3	34,600.98	33,937.69	33,564.83	32,384.50
		4096	75.88	77.46	78.06	80.36	34,483.89	33,819.18	33,462.49	32,323.50
	III	1024	75.78	77.31	77.93	80.28	34,599.11	33,943.39	33,575.58	32,387.12
		2048	75.93	77.46	78.03	80.34	34,477.68	33,831.20	33,478.04	32,324.42
		4096	76.1	77.64	78.15	80.43	34,330.71	33,687.38	33,351.41	32,232.54
b19	I	1	72.89	74.71	74.96	78.72	74,077.77	72,316.43	72,123.42	68,274.49
		2	73.2	74.9	75.15	78.79	73,755.27	72,093.86	71,891.89	68,156.53
		4	73.44	75.06	75.32	78.86	73,406.32	71,798.39	71,595.62	67,944.48
		8	73.71	75.23	75.5	78.99	73,050.98	71,530.23	71,310.90	67,688.74
	II	1024	73.08	74.84	75.05	78.75	73,772.55	72,077.48	71,913.30	68,137.08
		2048	73.2	74.91	75.12	78.77	73,592.33	71,950.11	71,786.90	68,069.92
		4096	73.39	75	75.21	78.81	73,338.09	71,760.22	71,605.27	67,950.66
	III	1024	73.21	74.9	75.13	78.78	73,610.69	71,972.19	71,793.07	68,072.68
		2048	73.38	75	75.23	78.81	73,363.38	71,789.97	71,625.00	67,970.68
		4096	73.6	75.1	75.34	78.86	73,027.66	71,542.90	71,378.98	67,804.02

Table 6. FC and SDQL Loss from Previous Work in Mixed-mode BIST

Circuit	Type	d	FC Loss (%)			SDQL Loss (%)		
			Select-1	Select-3	Select-5	Select-1	Select-3	Select-5
b18	I	1	6.13	3.96	3.15	6.83	4.84	3.79
		2	5.85	3.77	3.04	6.64	4.71	3.67
		4	5.61	3.62	2.93	6.48	4.58	3.58
		8	5.43	3.51	2.91	6.36	4.53	3.55
	II	1024	5.78	3.81	3.00	6.51	4.66	3.61
		2048	5.68	3.74	2.94	6.41	4.58	3.52
		4096	5.57	3.61	2.86	6.26	4.42	3.40
	III	1024	5.61	3.70	2.93	6.39	4.58	3.54
		2048	5.49	3.58	2.88	6.25	4.45	3.45
		4096	5.38	3.47	2.83	6.11	4.32	3.35
b19	I	1	7.41	5.09	4.78	7.83	5.59	5.34
		2	7.09	4.94	4.62	7.59	5.46	5.20
		4	6.87	4.82	4.49	7.44	5.37	5.10
		8	6.68	4.76	4.42	7.34	5.37	5.08
	II	1024	7.20	4.97	4.70	7.64	5.47	5.25
		2048	7.07	4.90	4.63	7.50	5.39	5.18
		4096	6.88	4.83	4.57	7.35	5.31	5.10
	III	1024	7.07	4.93	4.63	7.52	5.42	5.18
		2048	6.89	4.83	4.54	7.35	5.32	5.10
		4096	6.67	4.77	4.46	7.15	5.23	5.01

for generating patterns for transition delay fault (P.generation), and computation time for fault simulation to create fault list based on the detection counts (Fsim). Table 7 summarizes computation times for fault selection.

Table 7. Computation Time for Fault Selection

Circuit	P.generation (m)	Fsim (m)
b15	0.11	0.07
b17	1.08	0.41
b18	4.49	1.46
b19	7.25	4.49

To compare between the previous work and our proposed method, we evaluate each computation time in:

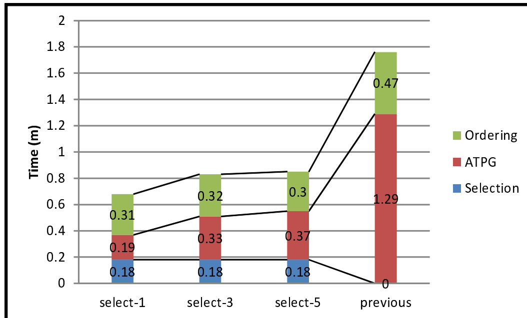
1. Fault selection. (Time to select faults in the proposed method).
2. ATPG. (Time for generating patterns).
3. Ordering. (Time for fault simulation in seed ordering).

Since, fault selection is only applied in our proposed method, for previous work we set this time to 0. Table 8 shows Computation time for all processes. The results show that the original work consumed longer time due to the fact that it targeted all faults during timing-aware ATPG. Therefore, if the test time is expensive, our proposed method can be applied to accelerate testing time.

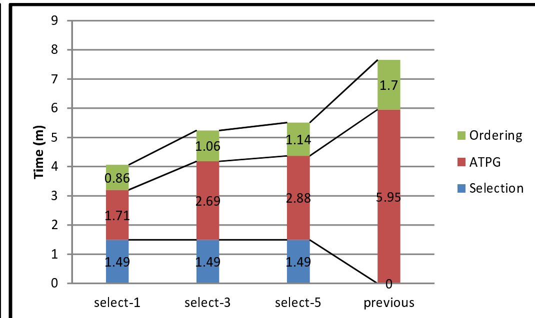
Figure 14 shows the computation time for different circuits. We can observed that the acceleration for each process is vary in different circuits. For example in b19 circuit, we can accelerate the processes up to 63.9% for $select-1$, 53.5% for $select-3$, and 50.2% for $select-5$ respectively. One advantage of this acceleration is that we can reduce testing cost in terms of computation time.

Table 8. Computation Time for all Processes

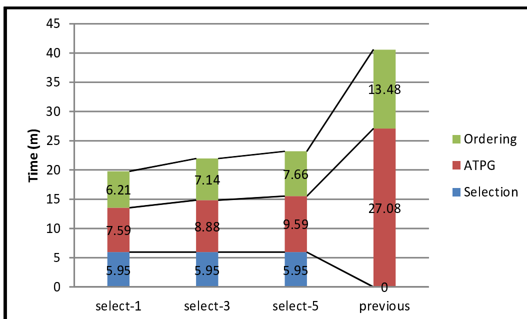
Circuit	Type	Select-1	Select-3	Select-5	Previous
b15	Selection	0.18	0.18	0.18	0
	ATPG	0.19	0.33	0.37	1.29
	Ordering	0.31	0.32	0.3	0.47
	Total (m)	0.68	0.83	0.85	1.76
	Acceleration (%)	61.4	52.8	51.7	0.0
b17	Selection	1.49	1.49	1.49	0
	ATPG	1.71	2.69	2.88	5.95
	Ordering	0.86	1.06	1.14	1.7
	Total (m)	4.06	5.24	5.51	7.65
	Acceleration (%)	46.9	31.5	28.0	0.0
b18	Selection	5.95	5.95	5.95	0
	ATPG	7.59	8.88	9.59	27.08
	Ordering	6.21	7.14	7.66	13.48
	Total (m)	19.75	21.97	23.2	40.56
	Acceleration (%)	51.3	45.8	42.8	0.0
b19	Selection	11.75	11.75	11.75	0
	ATPG	14.85	20.75	23.81	70.87
	Ordering	17.27	23.99	25.01	50.7
	Total (m)	43.87	56.49	60.57	121.57
	Acceleration (%)	63.9	53.5	50.2	0.0



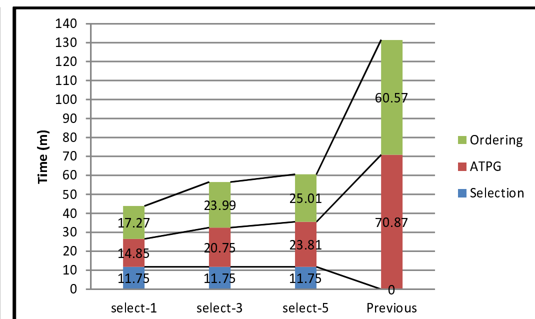
(a) b15



(b) b17



(c) b18



(d) b19

Figure 14. Computation Time for different circuits

5. Conclusions

Seed ordering and selection method based on exploiting the algebraic properties of the pseudo-random pattern generator (PRPG) to increase the number of patterns generated from one seed is an effective method to reduce the seed storage. In terms of targeting SDDs, LFSR-reseeding BIST offers solution for test pattern compression method. Other advantage is that we can apply some pseudo-random patterns combines with deterministic patterns (mixed-mode BIST), more seeds can be reduced and the patterns will increase the detection coverage of SDDs.

In this thesis seed ordering and selection method based on the gain in the sum of the longest path lengths sensitized by seeds, which is correlated with statistical delay quality level (SDQL) is applied. Experimental results show that this method can obtain significant seed count reduction under several mixed-mode BIST approaches, yet very time consuming, since it sensitizes large number of long paths.

We proposed a hard-to-detect delay fault selection method to accelerate toe computation time in seed ordering and selection process. This selection method can be used to restrict faults for test generation when it is impractical to target all delay faults that result in large test pattern count and long computation time. Target faults are restricted based on the number of test patterns that detect each faults for a given test set. We examine three types of hard-to-detect fault selection method, *select-1*, *select-3*, and *select-5*, where *select-n* means the faults detected at most *n-times* are selected. We use seed ordering and selection method in previous work and evaluate the delay test quality based on SDQL in deterministic and mixed-mode BIST environments.

Experimental results show that the proposed method significantly reduced seed counts from 21% up to 67%. We evaluate the effectiveness of the proposed method based on SDQL values, and found that the delay test quality is slightly decreased. However, our method can obtain significant acceleration in computation time from 28% up to 63% for overall processes.

Acknowledgements

Bismillahhirrahmanirrahim

In the name of God; The Most Gracious, the Most Merciful. First of all, I would like to express my gratitude to Allah Almighty for blessing me and giving me courage to finish my master thesis.

My sincerest gratitude goes to my supervisor, Prof. Michiko Inoue, who has supported me throughout my thesis with her patience and knowledge whilst allowing me to know the world of testing. I attribute the level of my Master degree to her patience guidance, encouragement and effort. Without her advices on my research, this thesis would not have been completed. I simply could not wish for a better or friendlier supervisor, who cared so much about my work, and who responded to my questions so promptly.

I would also like to thank my co-supervisors and staff; Prof. Kazumi Hatayama, Prof. Yasuhiko Nakashima, Assistant Prof. Tomokazu Yoneda, and Yuta Yamato for their helps, advices and comments during my research and writing my master thesis. And for all the members of Dependable System Laboratory; Mrs. Yoshiko Fujii, thank you for helping me all this time; and to M1 and M2 students for a good friendship and helping me in learning Japanese.

To my sweet child o' mine, Muhammad Adlil Khaliq, who I cherish with all of my heart. Thank you for being here and endlessly supporting me. You've got a smile that it seems to me, reminds me of childhood memories. Where everything was as fresh as the bright blue sky.

This thesis is dedicated for my late father, I always proud to be your daughter. For my mother who I love the most, my mother who did everything to cheered me up when I was down and homesick, and to all of my family who pray for my success.

To Indonesian government who gave me an opportunity to study here in Japan, it was such a great honor.

To all of my friends in NAIST, thank you for all of your support and I hope we still maintain our friendship. My big thanks to all Indonesian people who stayed in NAIST, with all of you here, I feel that I am home...

References

[1] R .Mattiuzo, D. Appello, and C. Allsup. Small Delay Defect Testing. *Test and Measurement World*, http://www.tmworld.com /article/CA6660051.html, 2009.

[2] N. Ahmed, M. Tehranipoor, and V. Jayaram. Timing-based Delay Test for Screening Small Delay Defects. *Proc. Design Automation Conference*, pp.320–325, July 2006.

[3] Synopsis. TetraMAX ATPG User Guide, Version C-2009.06-SP2, September 2009.

[4] A.A. Ahmad, M. Subhasish, and M.J. Edward. Optimized Reseeding by Seed Ordering and Encoding. *IEEE on Computer-aided Design of Integrated Circuits and Systems*, vol. 24, pp.264–270, February 2005.

[5] B. Koenemann. LFSR-coded Test Patterns for Scan Designs. *Proc. Euro Test Conference*, pp.237–242, 1991.

[6] M. Yilmaz, and K. Chakrabarty. Seed Selection in LFSR-reseeding-based Test Compression for The Detection of Small Delay Defects. *Proc. Design, Automation and Test in Europe*, pp.1488–1493, April 2009.

[7] T. Yoneda, M. Inoue, A. Taketani, H. Fujiwara. Seed Ordering and Selection for High Quality Delay Test. *Proc. Asian Test Symposium*, pp.313-318, December 2010.

[8] Y. Sato, S. Hamada, T. Maeda, A. Takatori, and S. Kajihara. Evaluation of The Statistical Delay Quality Model. *Proc. Asia and South Pasific Design Automation Conference*, pp.305-310, 2005.

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

[10] J. Savir and S. Patil. On Broad-Side Delay Test. *Proc. VLSI Test Symposium*, pp. 284-290, 1994.

[11] M. Yilmaz, K. Chakrabarty, and M. Tehranipoor. Interconnect-Aware and layout-Oriented Test-Pattern Selection for Small-Delay Defects. *Proc. IEEE International Test Conference*, 2008.

[12] M. Yilmaz, K. Chakrabarty, and M. Tehranipoor. Test-Pattern Grading and Pattern Selection for Small-Delay Defects. *Proc. IEEE VLSI Test Symposium*, 2008.

[13] S. M. Reddy, I. Pomeranz, and S. Kajihara. Compact Test Sets for High Defect Coverage. *IEEE Trans. Computer-Aided Designs*, pp.923-930, 1997.