DELAY TESTING IN INTEGRATED CIRCUITS: METHODOLOGIES FOR PATH DELAY FAULT DETECTION AND HARDWARE SECURITY

Palanichamy Manikandan

Institute of Engineering, Østfold University College, Fredrikstad, Norway

ABSTRACT

As semiconductor technology advances, high-speed circuits are becoming more common, which increases the chance of delay faults that can disrupt circuit timing. This paper explores the important issue of delay faults and reviews different testing methods, particularly focusing on path delay fault (PDF) testing and its role in hardware security. Various delay fault models and test generation techniques were discussed. The experiments on benchmark circuits showed that this approach could achieve a 12.7% to 19.6% increase in detecting path delay faults in circuits affected by hardware Trojans. These results highlight the need for effective delay testing methods to ensure the reliability of integrated circuits. This work recommends future research to focus on creating more efficient test generation methods and integrating security measures into standard testing practices.

KEYWORDS

Path delay fault (PDF) testing, test generation; path selection; hardware trojans (HT); HT detection

1. INTRODUCTION

As feature lengths of integrated transistors are decreasing, the amount of delay defects appearing in an IC is becoming a major concern while ensuring the timing correctness. This concern mandates appropriate tests to detect violations of the performance specifications of the circuit. Early approaches considered only logical correctness and assumed high stuck-at fault (SAF) coverage is sufficient to guarantee high-quality products. However, to increase the yield, industries have encouraged researchers to develop new test methods that can ensure timing correctness. Delay testing is considered the best solution. This area receives growing attention from both industries and academia. As a result, several delay fault models and numerous test methodologies have been proposed in the past two decades. This paper reviews a selection of existing delay test research results and encompasses basics with state-of-the-art techniques that address some of the current methodologies in delay testing.

Delay fault testing in digital circuits follows the steps of test generation, fault simulation, and fault grading. The creation of effective stimuli is an important part of the delay test procedure because it determines the fault coverage values [1] [2]. Models bridge the gap between physical reality and mathematical abstraction, and therefore physical defects of digital circuits can be modeled [3] as delay fault models when the defect affects the operating speed of the circuit. Test stimuli are then generated based on this delay fault model, which are used to verify the timing correctness of the circuit. The well-known delay fault models are the gate delay fault model, transition delay fault model, and path delay fault model. The gate delay fault model is delaydependent because it makes assumptions about circuit delays [4]. In this model, small-sized

delay defects may not be detectable, and the analysis may be invalidated if certain assumptions on delays in the circuit do not hold. The transition delay fault model and the path delay fault model, on the other hand, are delay-independent since they do not make any assumptions about circuit delays. The transition delay fault model [5] is like the gate delay fault model in which slow-torise (STR) and slow-to-fall (STF) faults are considered at gate inputs and outputs. The delay due to such slow transitions is assumed to be large enough to cause a delay fault when a signal propagates along any path through the fault site in the circuit. In the transition delay fault model, even though the test generation and fault simulation techniques are simple and require only minor modifications to stuck-at-fault (SAF) tools, it will not detect small delay defects (SDD). Segment delay fault model [6] helps to address the limitations of basic delay models. This model represents any general delay defect ranging from a spot defect to a distributed defect but restricts the length of segments and the number of segment faults that need to be considered.
However, the path delay fault model considers cumulative propagation delays along paths [7] in a circuit to detect delay faults. This model addresses the real situation of the circuit when there is a delay defect, and fault detection can be guaranteed by robust tests with no assumptions on circuit delays. Even though the path delay fault model supports an effective delay test method, it has several challenges in generating test patterns. These challenges include: (i) detecting small delay defects and shorter paths, (ii) pre-selection of longest paths in a circuit, and (iii) detection of faults that cannot support robust tests [9]. Path delay fault testing of large circuits remains an open problem, especially in achieving high fault coverage with cost-effective methods. These limitations have resulted in a lack of tools to cope with this type of testing and have prevented the adaptation of PDF testing. Motivated by this lack of tools, several recent research works have been engaged in developing techniques for PDF testing.

In this paper, we focus on surveying such selective test generation and fault simulation techniques based on the path delay fault model. Even though this study presents largely a qualitative view, we believe that this paper provides enlightening information on delay testing. A brief path delay fault (PDF) test primer in Section 2 provides background information. The path-selection techniques and test generation techniques are presented in Section 3 and section 4, respectively, followed by a PDF testing on hardware security in Section 5 and conclusions in Section 6.

2. PDF TEST PRIMER

The main objective of delay testing is to detect delay faults and ensure that the design meets the desired performance specifications.

2.1.Delay Defect and Delay Test

The delay defect in a circuit is manifested only when the delay of the propagated signal through a path arrives after the specified cycle time. Delay defect size (δ) is directly proportional to the path length (PL) or the difference between the designed cycle time (td) and actual arrival time (ta) [10]. If the defect is only considered at a particular point, such as a gate output or a signal on a path, it is called a point-defect or lumped delay defect. This can be used to model bridges and opens in a circuit.

If the defect is considered distributed along a path, then it is called a distributed delay defect. In this case, the applied signal passes through multiple lines in a path, and the accumulation of delay for those signals may be significant enough to impact circuit speed. This cumulative effect is used to model defects due to process variations [11]. Path delay faults model physical defects in a

circuit with a gate-level representation and consider cumulative propagation delays along paths. For example, too low doping in channels leads to higher resistance than specified, yielding increased delays on every transition. However, a path-delay fault test can also detect spot defects in cutting-edge sub-micron level along the path [12].

2.2. Fault Distribution

The path delay fault test considers two faults associated with each path in a circuit. Each path begins at a primary input, contains a chain of gates, interconnects, and ends at a primary output. The first fault is related to a falling transition at the source of the path, and the second fault is related to a rising transition at the source of the path. If the signal transition along the path accumulates too much delay due to defects, the rising/falling transition on the output will arrive late. This indicates that the presence of defects has been detected by the applied test stimulus. Two-pattern tests are required to detect delay faults in combinational circuits, while test sequences may be required to detect delay faults in sequential circuits [8]. The following procedure summarizes the two-pattern PDF test method. (i) Apply the first pattern (p1), which launches an initial transition and establishes the initial state of the circuit; (ii) Apply the second pattern (p2) after a time interval, launching the second transition in a path and propagating the signal value toward the output; (iii) Capture the response at the primary output after a pre-determined time interval. If there is a delay defect, an incorrect response will be captured.

This same test procedure can also be applied to sequential circuits with flip-flops at PIs and POs. In this typical delay test architecture, as indicated in [13], input and output latches are part of the circuit or are provided by the automatic test equipment (ATE) for testing. During test mode, the input and output latches are controlled by two different clocks: the input and output clocks. These independent clocks allow a phase delay to apply the two consecutive test patterns. As mentioned earlier, a two-pattern test assumes that all signals due to p1 have reached a steady state before applying p2. If the steady-state assumption is not true, transient signals may be present in the circuit, which could interfere with the testing of the targeted path [14]. To avoid this problem, test patterns (p1, p2) are applied at slower than the rated clock frequency. As indicated in the timing diagram of Figure 1, p1 is applied at t0, and p2 is applied at t1 of the input clock of the input latch. The time difference (t1 - t0) allows all signals in the circuit to stabilize under p1. Then, the output clock is latched by the time period, which is equal to the rated clock period. This allows the circuit to settle down with signal transitions due to p1 followed by p2. If the delay of the selected path is longer than the rated-clock period, then the faulty output will be observed in the output latch [15].

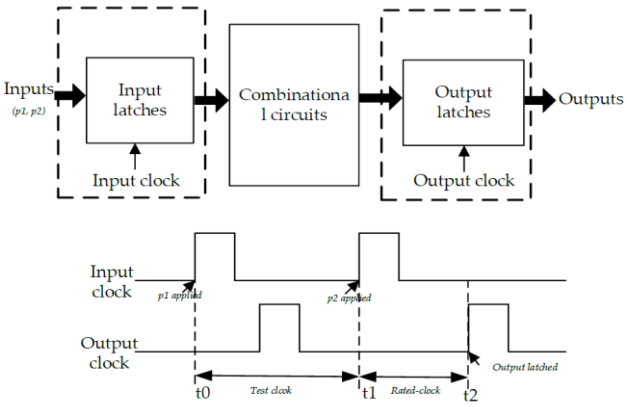

Figure 1. Delay test architecture with its timing diagram

2.3. Faults Detection in a Gate-Level Circuit

The goal of test generation based on the path delay fault model is to derive test patterns that can be used to test each manufactured circuit/component for timing correctness. [7] analyzed the major reasons for low path delay fault coverage in digital circuits. To achieve higher confidence in path delay fault tests, the number of untested faults in a circuit should be small. In practice, achieving high fault coverage is challenging mainly due to non-activatable paths, and it is difficult to produce two consecutive states effectively to create and propagate a transition through the paths [16]. Based on signal propagation criteria, fault detection under PDF test offers two different test methods: robust and non-robust tests. This subsection provides the basic principle of these two tests.

In robust testing, fault detection can be guaranteed with no assumptions about circuit delays. However, it requires stringent logic conditions for the detection of a delay fault. For instance, in robust testing, off-path inputs of logic gates along the targeted path are expected to have stable non-controlling values (NCV) while propagating either a falling or rising transition through onpath inputs, as shown in Figure 2(a). When the test pattern pair activates signal transitions on the targeted path, all off-path inputs of the gates along the path should be robust. This is called a robust sensitizable path, which results in a high-quality test. However, most of the circuit paths cannot be tested under robust conditions, leading to robust untestable path delay faults [17]. In non-robust testing, fault detection requires knowledge about delays in the circuit, even though it is less stringent than robust testing. For example, if there is an NCV → CV (control value) transition in the on-path input of the gate along the targeted path concerning the applied test patterns, then a CV → NCV transition is expected in off-path inputs of the same gate, as shown in Figure 2(b). The transition arrival time on off-path inputs should be earlier than the transition arrival time on on-path inputs to detect faults in the targeted path. If the off-path signal transition arrives after the on-path signal transition, the fault cannot be propagated to the output, resulting in an invalidated non-robust test [18].

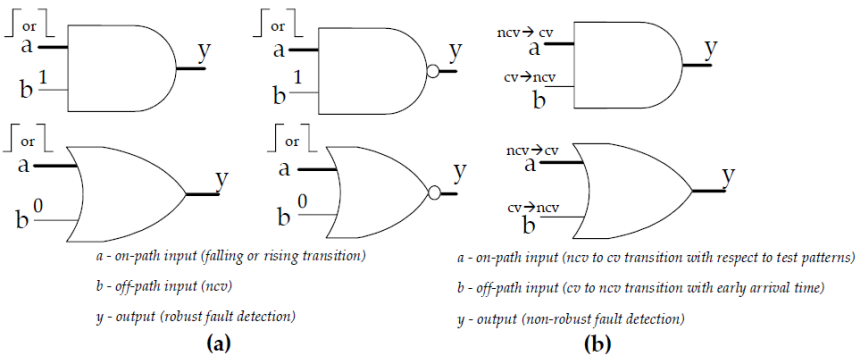

Figure 2. Principle of robust and non-robust PDF test

3. PATH SELECTION OF PDF TESTING

The number of testable paths can be huge in the CUT. It is impractical to consider all circuit paths because this would cause the size of the data structure inside the test patterns generation (TPG) tool to blow up. In order to keep the whole path store inside computer memory, the size of the path store with respect to the number of paths was set to a reasonable value. Generally, path selection approaches are based on either selecting paths with high fault probability, i.e., K-

longest path selection, or improving the quality of a path set of a specific size with the assumption of structural or spatial correlation and statistical timing analysis, i.e., time defects cause many paths to be faulty that pass through the defective site. However, a simple and effective approach is testing the longest paths through a fault site because it increases the fault detection probability.

Several studies have analysed the method of finding the longest paths through each gate or line in a circuit and described the fault propagation under certain sensitization criteria and finding the global longest paths. Many ATPGs, like NEST [19], DYNAMIT [20], RESIST [21], VIPER [22], and KLPG [23], have extensively studied the problem of finding k-longest testable paths through each gate in a circuit. Additional studies [24] have discussed the recursive path selection method, which does not require the iteration process for each path and correlation between paths. It recursively continues and selects a requested number of paths simultaneously. The branch-andbound algorithm-based statistical path tracing was introduced in [25] and reports the statistically most critical paths from industrial circuits with multi-million gates. However, this section gives an overview of well-known path selection methods and reviews a few selective methods.

3.1. Path Selection: Term Analysis

Generally, in gate-level circuits, timing analysis often relies on gate delays, interconnects, and signal propagation where the earliest, latest, and average signal arrival times are estimated for each PI to PO pin pairs. Based on these discrete timing values, the delay of a path can be defined as the accumulated delay on the path. Then, the set of critical paths can be constructed by selecting either a fixed number of the longest paths or all paths that fall into a pre-defined time range. Eventually, this set of longest paths is expected to ensure a complete topological coverage of the circuit. However, this basic definition for the path selection procedure may not be sufficient to model delay defects in deep sub-micron technologies. In deep submicron technologies, delay variations due to the manufacturing process, small defects, and/or signal noise cannot be detected with discrete timing assumptions and accumulated delay effects of longest paths. Even though there is no universal procedure defined for path selection in circuits, some techniques, like klongest path selection [26], recursive path selection [27], and statistical path selection methods [28], are generally used in PDF testing.

The commonly adopted method in both academia and industries is to select the K-longest testable paths in a circuit. In this method, the fault list is often set equal to the K-longest testable paths. This K value depends on the reasonable number of test patterns [23] [24]. Also, the selection of the longest or critical path depends on the timing length of a path or all testable path-delay faults considered to be longer than a predefined limit. It is often calculated using discrete delay models based upon worst-case timing scenarios. This K-longest path selection approach guides the test generation process along the longest paths with minimum slack. For instance, the difference between the minimum of the required time and the maximum of the arrival time at any given node is the slack. This simple and effective process selects long paths and obtains the difference between the arrival and required times by propagating fault effects along the longest paths. Even though the path selection technique is closely followed by the test generation, the next subsection only discusses the path selection for our simplification.

3.2. An Overview of Path Selection in ATPGs

An ATPG tool, NEST [19], generates paths in a non-enumerative way. This tool can handle a large number of paths and detect large numbers of path delay faults by propagating transitions robustly through parts of the circuit. It does not require enumerating the specific paths through

every selected subcircuit where the transitions are propagated. This tool uses labeling techniques that consider only lines in selected subcircuits and is used to determine test generation objectives effectively. However, NEST is only effective in highly testable circuits where a large number of path delay faults are testable. This method removes the most limiting restriction of the path delay fault model while handling a large number of path delay faults.

In contrast to NEST, the Delay Fault Oriented Automatic Test Pattern Generation System (DYNAMITE) [20] introduced an effective method to handle poorly testable circuits. The path sensitization procedure of this tool is used to identify large numbers of path delay faults as redundant by a single ATPG attempt. If the selected subset of paths is not well testable due to the presence of many redundant paths, this method allows dynamic switching to another subset of paths. This will eventually succeed in generating a test set for all testable path delay faults and identifying all redundant ones. However, in highly testable circuits, many faults are treated separately, which results in huge memory consumption. It shows that this method is not suitable for larger circuits.

Recursive selection and sensitization technique (RESIST) [21] overcomes the limitations of NEST and DYNAMITE. RESIST shows a cost effective method for a path selection and to test a large number of path delay faults in both highly testable and poorly testable circuits. It introduces an optimal search strategy for paths selection. For instance, many paths in a circuit have common subpaths which results dependent path delay faults. It is enough if these paths are sensitized only once. This tool addresses this issue by reducing the number of value assignments during path sensitization. To illustrate the basic idea of RESIST, let us use Figure 3. In Figure 3, a structural path (P) starts at a PI or present state lines (PS) and ends at a PO or next state lines (NS). P consists of several structural subpaths (S0, S1 … Sn). Each subpath leads from a PI/PS or a fanout stem to a PO/NS or a fanout stem.

To simplify the following discussion, we will assume that P is associated with two different faults. They are $p_r = (S_0^r \ldots S_n^{t_n})$ and $p_f = (S_0^f \ldots S_n^{t_n})$(rising and falling transitions at S0. The transitions at all other on-path signals $x_{(i,1)} \leq i \leq n$, by uniquely determined by gates along path P. Here only the bold paths starting from y0 (PI) and ending at yn (PO) will be considered. In which, there are 2n different subpaths $S_{i,i+1k}$ where $0 \leq i < n$ and $1 \leq k \leq 2$, between consecutive fanout stems.

Each path from y0 to yn consists of n subpaths $S_{i,i+1k}$. Altogether there are 2n different paths from y0 to yn. Conventional ATPGs sensitizes each path separately. Hence, it performs $S(n) := n \times 2^n$subpath sensitization steps (SPS) in any given circuit Cn. To reduce the number of SPS, RESIST uses a different strategy.

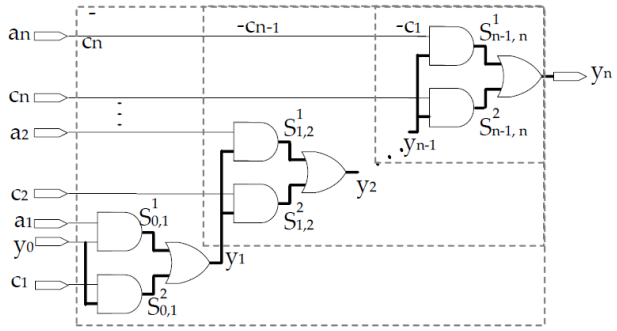

Figure 3. Dependency of PDFs due to Common sub-paths in main paths

Let us consider the sub-paths $S_0^{1},_1$and $S_0^{2},_1$from yo to fanout stem y1. Both subpaths are included in 2"-l paths from yo to yn. Hence, sensitizing $S_0^{1},_1$and $S_0^{2},_1$only once reduces the number of SPS from S(n) to,

$$2.\ [1+\ (n\ -\ 1).\ 2^{n-1}]\ =\ 2\ +\ 2.\ S(n\ -\ l), \qquad (1)$$

where S(n-1) := (n-1). 2n-1 is the number of SPS performed by conventional ATPG in circuit Cn-l. Applying the same principle at all subcircuits (Cn-1, Cn-2 … Cn), the number of SPS becomes

$$=2\ +\ 2.[2\ +\ 2.[2+2.S(1)]...]] \qquad (2)$$

$$=2^1+\ldots+2^{n-1}\ +\ 2^{n-1}.S(1)\ =\ \sum_{i=1}^{n} 2^i \qquad (3)$$

$$=\ 2^{n+1}-\ 2 \qquad (4)$$

Since S(1) = 2, compared with conventional ATPG, the number of SPS is reduced by,

$$\frac{n.2^n}{2^{n+1}-2}=\frac{n}{2-2^{\sim}n+1}\approx\frac{n}{2} \qquad (5)$$

For instance, if n=60 then the reduction factor is 30. The reduction factor increases with an increasing number of fanout branches (fanout > 2) converging to n. This simple sensitization procedure gives a speedup factor of this method that grows linearly with the circuit depth. Performing the sensitization step for a common subpath S only once requires that the TPG status is updated at fanout stems. Both mandatory value assignments for subpath sensitization and the corresponding unjustified lines have to be stored in order to exploit the TPG status for all paths including subpaths. Also RESIST identifies large sets of untestable paths without enumeration.

Another method to find a set of longest testable paths through different gates under unit delay assumption [29]. This ATPG technique automatically determines the longest testable path passing through a gate or wire in the circuit without first listing all long paths passing through it. The path selection is based on a graph traversal algorithm that can traverse all paths of a given length in a weighted directed acyclic graph (DAG). This method used search space pruning techniques while searching for the longest testable path through each node. Since this improved version of RESIST assumes a unit delay model, there is no obvious way to extend it to handle the problem of finding the K longest testable paths through each gate. For instance, this method fails when applied to C6288.

[30] introduced a timing analysis tool based on the recursive learning technique. Recursive learning [27] is a technique that can identify all necessary assignments required to satisfy a set of value assignments in a circuit. Necessary assignments are computed by temporarily injecting all combinations of possible values to the gate inputs that would justify the gates and observing the result after direct implication using a novel recursive ATPG technique. This timing analysis tool is used to find indirect conflicts during path building with the forward trimming technique. It efficiently identifies the global longest paths in combinational circuits and prevents the checking of multiple paths with equivalent constraints. Instead of generating many long structural paths and checking their testability, this tool grows paths from PIs, as shown in Figure 4.

3.3. Paths Selection/Generation Procedure

Step 1: First, during the preprocessing, the circuit netlist is loaded and represented as a graph with n primary inputs and m primary outputs. All circuit components (i.e., logic gates), PIs, and POs are represented as nodes in the graph with edges representing circuit interconnect. Some delay information such as Omax, Imin, and Imax about each gate need to be determined and loaded at this time. First, the maximum distance from each gate to a PO is calculated at each node in the circuit. It is represented as Omax(G). The maximum Omax(G) of all PIs is the delay of the longest structural path in the circuit. This can be calculated through a simple depth-first search. Imin and Imax represent the earliest and last possible time of signal transitions at gate G. These values are calculated assuming that all PI transitions will occur at time zero. But it is possible to relax this condition in order to allow input transitions to occur over a range, helping to reduce constraints applied to the circuit during path building.

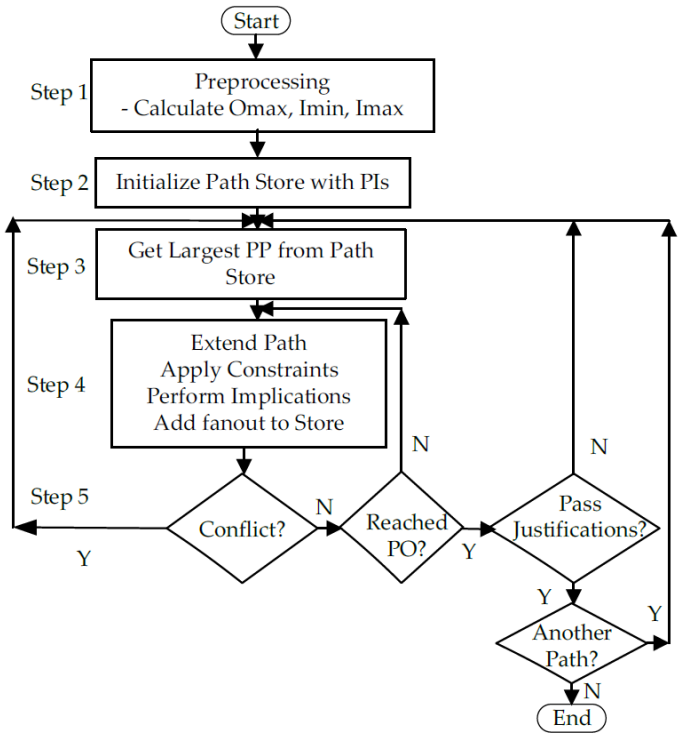

Figure 4. Incremental Path Generation Procedure

Step 2: The path store contains the partial paths. A partial path starts from PI but has not reached PO. The partial path can be extended by adding one gate along its path. When this extension reaches PO, it becomes a complete path. To compute the longest paths during the path generation process, the partial paths are sorted by max Esperance. The max esperance of a partial path is the upper bound of its delay when it grows to a complete path. It is calculated from the delay of the current gate and the largest potential delay to a PO.

Step 3: The path generation process begins by checking the path store for the largest partial path. The paths can be generated in decreasing order of path length based on the determined information. During the path generation process, after selecting a partial path, it is extended through the fanout with the highest esperance.

Step 4: The partial path extension begins by adding a new gate and applying the constraints to that gate. In VIPER [22], logic values are assigned to one or more PIs to satisfy the constraints on the newly added gate. However, this timing analysis tool uses direct implications, which are more efficient in finding local conflicts. It is used to discover blocked paths early.

Step 5: If there is a conflict then the whole search space which contains the already grown series of gates is trimmed off. This is called implicit false path elimination which guides the search toward the true longest path. It improves the search process during the path building phase of the incremental path generation routine. After direct implications if the path extension reaches PO without conflict, then it is important to perform the final step, called path justification. The reason is that the immediate prior process such as direct implications of the path generation can only implicitly eliminate false paths with local conflicts. But still there is a possibility to have indirect conflicts which may cause a generated path to be false. This path justification is also used to determine a set of primary input values that can be used to sensitize the path. A FAN style decision tree-based justification routine is used to sensitize the generated potential path. This process begins by applying all the constraints necessary to propagate a transition along the path. If the generated path passes the justification, then the path generator will go to another path. This method can handle C6288.

3.4. Observations

Section 3 shows an overview of some important path selection algorithms. Many path delay faults do not have robust tests, and therefore, it is necessary to develop efficient test generation procedures for non-robust tests [32]. While propagating signal transitions through the path nonrobustly, the worst-case delay may occur. It is also observed from the timing analysis that the delay of a path under non-robust test is longer than the delay of a path under robust test [17] [33]. This can cause a delay defect affecting the path and may not cause the circuit to fail at its designated speed of operation under a robust test. However, the circuit will fail under a non-robust test or may fail the same test (t) in the presence of the same fault (f1) if the path operates under its worst-case delay. Developing test generation procedures for path delay fault testing is a challenging task.

4. TEST GENERATION FOR PDF TESTING

The problem of finding the longest path through each gate or line in a circuit has been extensively studied in [34]. Several works inherit the framework of the existing techniques for the path selection and have developed different test generation procedures with improved experimental results. This section summarizes some existing test generation procedures with analysis. Generally, PDF tests for digital circuits (combinational and sequential circuits) are based on either non-scan methods or scan methods. These methods are implemented in circuits using self-test techniques because self-test offers the ability to apply the stimuli effectively and analyze at-speed test signals with better accuracy. Built-in Self-Test (BIST) techniques are well-suited in the path delay fault (PDF) testing of digital circuits [35]. This is a preferable technique for detecting a very large class of physical failures in the circuits. There are different ways to implement BIST [36] [37], but it usually comprises the following blocks: a test pattern generator (TPG), a circuit under test (CUT), a response analyzer, and a BIST controller. This section focuses on different automatic test pattern generation (ATPG) techniques for PDF testing.

4.1. Non-scan-based PDF Test Setup

In this subsection, we describe the pseudo random test stimuli generation for PDF testing. Figure 5 shows the non-scan-based BIST setup for PDF test. It comprises of netlist generator, path & test stimuli generator, PDF simulator and test controller with fault grading. The netlist generation is important in PDF test because netlist representation of the CUT simplifies the simulation algorithms and their implementation. It means that having an efficient netlist architecture is significant to achieve a better performance from the simulator [38]. The netlist typically contains logic gates and their associated interconnects along with attributes for the gates. First, the structure of the netlist can be represented using a Directed Acyclic Graph (DAG) with two event-list classes. The fan-outs associated with each gate are in one DAG and the fan-ins associated with each gate are in another. This helps to perform forward and backward propagation through the circuit easily.

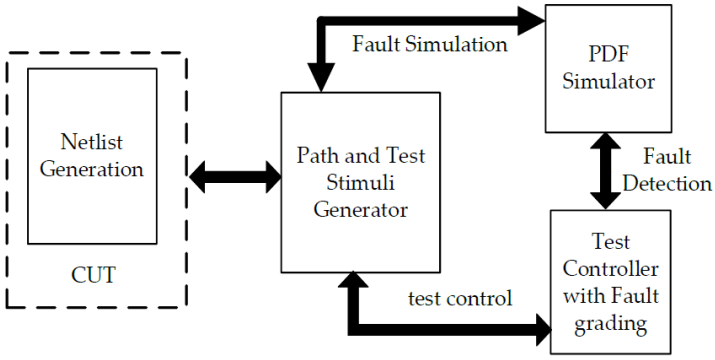

Figure 5. Non-scan-based BIST setup for PDF Test

Second, the main concern in PDF testing is the path generation. Section 3 reviews different path selection procedures to generate the longest testable paths. However, a K-longest path selection approach guides the test generation process along the longest paths with minimum slack. This approach selects long paths and obtains the difference between the arrivals and required times by propagating fault effects along the longest paths.

Next important part of the test setup is the test stimuli generator. Test stimuli generator of [39] is based on accumulator and mersenne twister methods. Both accumulators based and mersenne based pseudo random test stimuli generators are using different weighting schemes which are explained in the next subsection. In each weighting scheme, first weights are generated. Those are then used together with the pseudo random generators. The generated weights are based on the single stuck-at fault model and a deterministic test set. It is also important to create the basis patterns based on stuck-at faults. It can help in the path delay fault testing if the weight-based test patterns for stuck-at faults would yield a better fault coverage result or not. In the basis patterns, each bit is shifted once from 0 to 1 and once from 1 to 0. Therefore, testing a path delay fault over with a subsequence of successors and intrinsic complement operations on each path effectively tests for both rising and falling transitions on its inputs. In this way, every basic pattern produces 2N test vectors, and N is the number of inputs to the circuit. Figure 6 shows the basic concept of SIC stimuli generator which produces SIC stimuli by first establishing a basis pattern. Then each bit is toggled twice in order to generate both rising and falling transitions. The test patterns are encoded using smith's alphabet [40] as shown in Table 1. The concept of different test stimuli generation techniques and summary of different test stimuli generators are presented in the next subsections.

Table 1. Encoding – Smith's alphabet

S0	S1	P0	P1
0	1	0	1
0	1	1	0

Third, the PDF simulator starts to evaluate the generated test stimuli by applying them into the generated paths. In which, the generated paths are translated into the PDF simulator's input fault list format. This basically involves mapping the paths and its corresponding gates with levelized information. Then the test vectors need to be applied into the simulator. Thus, the PDF simulator computes the correct simulation value of all gates along the paths from the circuit netlist using the generated test vectors. During simulation, the correct signal value of each gate is computed one at a time in topologically sorted order. The levelization assures that the signal value of a gate is not computed before the signal values of the gates driving the inputs. All gates in the netlist are visited at least once. When a new test vector is applied during simulation, it can cause one or more signal changes at the inputs of the circuit.

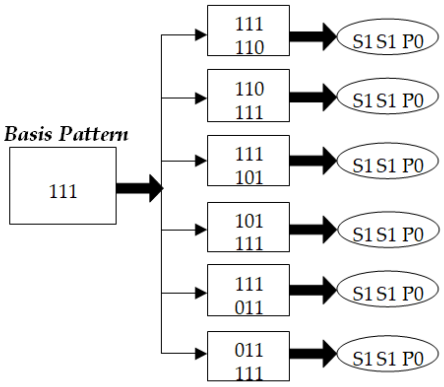

Figure 6. SIC Test Stimuli Generation

Fourth, The PDF test controller manages the process of fault simulation, applying test stimuli and fault grading. According to the fault grading algorithm, it assigns a unique number to each pathdelay fault and when a fault is detected the corresponding number is stored in a list. The algorithm [41] is intractable by definition since the number of path-delay faults is counted one by one. If the number of path-delay faults is very high then it is better to use a non-enumerative algorithm [42] even though it is more complex. It is fast and efficient while managing the huge number of longest paths and its corresponding faults. However, the fault grading requires a well-organized datastructure which can handle previously as well as currently detected faults efficiently for each test stimuli applied to the CUT.

4.2. PDF Test Stimuli Generation Techniques

This subsection describes a set of methods and techniques that can be used to design the test stimuli generators. Pseudo exhaustive test can give high fault coverage but applying 2^n test vectors, in the case of stuck-at faults, is only feasible for circuits with few inputs n. However, sometimes, it is possible to partition the circuit so that each sub circuit can be tested exhaustively. Pseudo exhaustive test generators may be realized in many ways, for instance, based on accumulators and as a part of the ABIST methodology. These two generators are given here.

ACC-FIXED: Optimal accumulator-based generators for single size subspaces: Arithmetic BuiltIn Self-Test (ABIST) [43] is a term introduced by Rajski and Tyszer. They pointed out the existing components in today's complex integrated circuits often contain ALUs and memory that can be reused for testing purposes. One efficient way to generate stimuli (measured in the number of clock cycles needed to generate a new vector) is to accumulate a constant as shown in the below equation.

$$Ai+1 = Ai + C \bmod 2^n, A0 = I \frac{n}{2} \quad (6)$$

There are 2^{2n} ways of choosing pairs of C and I, and the resulting generator exhibits some interesting properties for each pair. By carefully selecting the parameters C and I, it is possible to cover exhaustively every subinterval of size r within the first 2r test vectors. A pseudo exhaustive generator can be used to test modules with physically adjacent input lines (e. g. adders). The value of r should then be set equal to number of inputs to the partition with the largest number of inputs; ACC-RANGE: The best accumulator-based generators for subspaces within a range of sizes. The number of inputs to the partitions often varies, and in such cases a generator made for subspaces with fixed size might be suboptimal. However, It is not possible to synthesize values for C and I for equation 1 in such cases.

Pseudo-random technique: The most economical BIST techniques are based on pseudo-random pattern testing. Pseudo random pattern generators are used to generate test sequences that have the same properties as true random sequences even though the sequence is generated by a deterministic algorithm. The number of test vectors needed in order to detect all faults are usually much smaller than the number of test vectors generated during an exhaustive or pseudoexhaustive test. However, the test sequence might still be long due to random pattern resistant faults.

LFSR - Linear feedback shift register: The most popular pseudo random generator is the linear feedback shift register (LFSR) [44]. LFSRs are very efficient in hardware and are also easy to emulate in software.

TWISTER - Mersenne twister pseudo random generator: Mersenne Twister [45] is a pseudorandom generator which has a period of $2^{19937}– 1$. The generator is fairly complex and is not suitable for use in built-in self-test. However, there are many pitfalls when designing pseudorandom generators, and the Mersenne twister may thus be used as a verification tool in the design phase. If, for instance, an LFSR based generator in a BIST environment performs much poorer than the Mersenne twister, it may be caused by some structural or linear dependencies.

MAC - Multiply and accumulate based generator: In order to reduce the test application time of large sequential circuits with scan, the scan chain is usually broken down into several scan chains. These scan chains must then be fed by the test generator. LFSRs may, due to structural and linear dependencies, fail to produce some test patterns. Instead one can use a generator based on multiply and accumulate (MAC) operations.

Even though pseudo-random BIST provides an economical solution, it has couple of drawbacks such as low fault coverage and high power dissipation. Low fault coverage arises due to the presence of random pattern resistant (r.p.r.) fault, which have low detection probabilities. Solutions to this problem involve either modifying the circuit-under-test (CUT) by inserting test points to increase the detection probabilities, or by modifying the test pattern generator so that it generates patterns that detect the r.p.r. faults. The problem of high power dissipation comes from the fact that pseudo-random patterns cause much greater switching activity in the CUT than what

occurs during normal functional operation. This can result in overheating, as the chip package may only be capable of handling the power dissipation that occurs during functional operation. The problem of low fault coverage for pseudo-random BIST has been studied for a long time and quite a number of solutions have been proposed. One of the most attractive involves adding weight logic to bias the pseudo-random patterns towards those that detect the r.p.r. faults.

4.3.Weight Technique

Deterministic test set based weight computation (DTW): Generally, these methods generating weights are based on structural analysis of the CUT. There are several ways of creating weights for a weighted stimuli generator [46]. One common method used in conjunction with the single stuck-at fault model, is to create weights based on a deterministic test set for stuck-at faults. It is interesting to find out whether or not a weight set based on a deterministic test set for stuck-at faults would yield a good result. For instance, first a deterministic test set for the circuit under test can be created using TetraMax, an ATPG from Synopsys. This deterministic test set will contain number of test vectors as shown in Figure 7.

Then the weights can be determined by first counting the number of test vectors n1i with the symbols ’1’ and ’X’ present at input i, as well as the number of test vectors n0i with the symbols ’0’ and ’X’ present at input i. Values for n0 and n1 is shown in Figure 7. The probability for observing ’1’ at the input of the circuit under test at input i can be computed using p1 = n1/(n1 + n0). The ATPG can be used in order to extract the K-longest testable paths in a circuit together with a valid test vector. The weights were then computed based on the deterministic test set for path delay faults. These weights based deterministic test generators proved their efficiency in detecting pat delay faults (PDF-DTW).

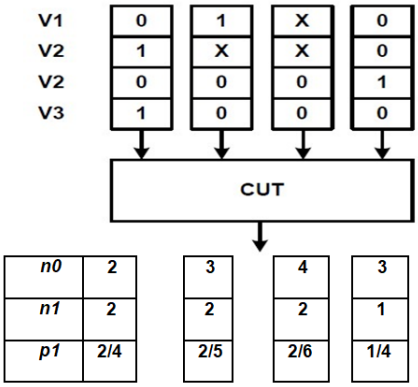

Figure 7. A deterministic test set and weights computation based on a deterministic test set for stuck-at faults

COUNTING-based weight computation (CBW): Weighs can also be generated based on fault coverage measurements. If we consider the CUT is attached with the pseudo random generator (PRG), the PRG can provide uniformly distributed basis patterns to the inputs of the CUT. Counters associated with each input of the CUT can be used to store the number of faults detected for each input values. When the desired number of basis patterns has applied the weighting factors can be computed using counter values. Generally, circuits will have some paths that are very easy to detect and other paths that are more difficult to detect. The easiest to detect faults are usually detected in the first few vectors anyway, thus it is a good idea to try to tune the weights on to the faults that are a bit more difficult to detect. When weights were generated using this method, 10M SIC patterns were applied to each CUT to compute weights.

Fault Subset based weight computation (FSW): Each path-delay faults starts at an input-node and ends at an output-node. This can be used to efficiently divide the set of all path-delay faults into smaller disjoint subsets containing only faults that ends at a particular output, starts at a particular input or both. Instead of trying to detect all faults in the fault set using one fixed weight set, it may be more efficient to restart the generator with weights that target one subset of the fault set at a time. The CBW technique can easily be adapted with this technique.

4.4.Implementation of PDF Test Stimuli Generators

Based on the above-mentioned techniques, [47] introduced different test stimuli generators. These ATPGs are intended to use in software based BISTs. The stimuli generators are based on accumulator and Mersenne twister based pseudo random generators. For instance, we describe here about the implementation of accumulator based pseudo random test stimuli generator (APRG) using assembly code. The fault simulation of this two-pattern test is assumed that the first pattern is held until it has propagated through the CUT. The second vector creates a transition and the signature compactor must sample the response one clock cycle later in order to capture any path delay faults. The test generators were realized in test program programs and test programs were implemented using the instruction set given in Table II. It is assumed that every instruction executes within one clock cycle. Test generators can easily be extended in order to include a response compactor as well. The test application was assumed that the CUT is connected to the processor through the register R0, and that R0 has the same width as the number of inputs to the CUT. This section discussed a review of several non-scan-based test stimuli generation techniques for PDF testing. The next sub-section gives the application path delay testing technique in hardware Trojans detection.

Table 2. Instruction Set

Instruction	Description
LOAD Rd, k LOADI Rd, k	Set the content of register Rd to [data stored in memory address k - immediate]
MOV Rd, Rr	Move content of Rr to Rd
ADD Rd, Rr ADDI Rd, k	Add the content of [reg. Rr — immediate] to Rd. Store the result in Rd
ADDC Rd, Rr ADDCI Rd, k	Add the content of [reg. Rr —immediate] with carry to Rd. Store the result in Rd
AND Rd, Rr ANDI Rd, k	Bitwise AND operation of [reg. Rr — immediate] and Rd. Result is stored in Rd
OR Rd, Rr ORI Rd, k	Bitwise OR operation of [reg. Rr—immediate] and Rd. Result is stored in Rd
XOR Rd, Rr	Bitwise XOR operation of Rr and Rd. Result is stored in Rd
NOT Rd	Invert the bits in Rd
ROL Rd	Rotate left the content of Rd
BRNE k	Set program counter to k if the equal flag is not set
CPI Rd, k	Compare register with immediate

5. TESTING IN HARDWARE SECURITY

Hardware Trojans (HTs) have become a significant concern in the trustworthiness of integrated circuits (ICs). If an untrusted foundry fabricates the IC, there is potential for an adversary to insert malicious behavior, known as Hardware Trojans (HTs). Detecting HTs is very challenging because of the diversity of Trojans and the stealthy modifications made to the hardware. HTs can be intelligently inserted in such a way that there is no impact on the function, area, power, and performance of the original design. They can also be inserted in a way that passes typical testing procedures [48], and they can be activated at any instant during runtime. Adversaries typically target weak sites, such as paths with large slack or paths with low switching activities, when inserting HTs. This section discusses detecting such HTs using the path delay fault testing approach. Our method focuses on detecting HTs by comparing the statistical path delay fault coverage (PDF Coverage) values before and after HT insertions for the particular selected paths and test seeds.

5.1. HT Detection Principle

The HT detection steps are as follows: i. First, select K shortest paths, ii. Generate m test patterns based on particular seeds, and compute the PDF coverage values, iii. Second, identify paths with large slack values among the selected K shortest paths, and insert an HT into the identified paths, iv. Generate the same m test patterns based on the same seeds, and compute the PDF coverage value for each circuit. This value will differ from the previous one because these patterns are not targeted to excite the inserted HTs [49], and the added interconnects and gates from HTs will affect the PDF coverage values. To observe the HT payload for each of these paths and cybersecurity threats [50], an appropriate vector must be generated to excite the path. This requires a different set of test generation conditions from typical path delay fault testing. In practice, PDF coverage values of HT-free ICs can be obtained by applying test patterns and performing PDF coverage measurement. After applying the patterns, a limited number of ICs are reverse engineered to ensure they are HT-free. Once the reference PDF coverage values are obtained, the same patterns are applied to the rest of the ICs. If the PDF coverage values differ from the reference values, the corresponding ICs are considered suspicious, potentially containing HTs. Different-sized HTs under varying process conditions can be detected by applying test patterns and observing the PDF coverage values.

The fault simulation algorithm together with fault grading technique and its test patterns generation are based on three-level (3-L) software platform that we have introduced earlier [47]. It works with a logic of SIC based test pattern generation technique in detecting K-shortest PDFs rather than K-longest PDFs. The SIC patterns are pairs of patterns and differs in only one bit. The pattern generator varies depending on how the seed patterns are generated. In our previous works, different methods for generating seed patterns were explored, including general arithmetic (GA) and the general Mersenne Twister (GT) test generators. It used the logic of pseudo-exhaustive and weighted random generation techniques.

The generated weights are associated with the single stuck-at fault model and based on a deterministic test set for stuck-at faults. Even though the test patterns are used to detect path delay faults, creating the seed patterns based on stuck-at faults is a sensible approach. It is also meaningful to test whether weight-based test patterns for stuck-at faults yield good coverage for path delay faults. The enumerative implementation and skip-list-based data structure were used to compute path delay fault coverage values. The skip-list data structure contains multiple pointers since elements can be in more than one list. This implementation makes the software easier to manage, and the open-source code for the skip list is particularly useful. When a path delay fault

is detected, a search in the skip-list is performed to find the proper position to store the fault. If the position is empty, the new path delay fault is stored; otherwise, the fault has already been detected.

5.2. Experimental Results

Our experiments considered 10K-shortest paths of ISCAS'85 benchmark circuits. 10 million (10M) single input change (SIC) test patterns were applied and the same was repeated ten times under the consideration of statistical variations. The number of test vectors are high enough to achieve high path delay fault coverage. Results are shown in Table 3. Each path is examined by SIC test vectors (rising/falling transitions) and detected faults associated with are logged.

In case 1, the generated test vectors applied to the CUT. For each detected fault, the simulator logs the path number in which the fault got detected. It makes it easier to account the detected faults and drop its corresponding paths from the path list, immediately. This helps in the realistic testing and not repeating the test again on the already tested paths. The number of detected faults produced by each test vector can be huge and it is important to use a right data structure to handle them. Also, one N-bit (N is the number of primary inputs to the CUT) basis pattern is re-used in 2N test vectors and therefore it is vital to create high quality seed patterns. It is significant to note that the fault coverage may differ based on the seed pattern because each good basis pattern may detect several new faults.

In case 2, we performed a timing analysis to compute the slack time at each circuit nodes of the selected k-shortest paths. Slack time at any timing node is the difference of its required arrival time and its arrival time. For example, at the primary output node of the path, the arrival time is 0+2=2, and the required arrival time is 6-2=4, and therefore a slack value is 4-2=2. The more accurate the estimation of the delay is, the better. This delay computation is therefore done after place and route of the original design to take into account delays of both gates and interconnects. Furthermore, this is what reflects the best information that an attacker can obtain from the GDSII sent to the foundry. Once this slack information is known for each of the selected path, it helps to decide about nodes with a large slack time. These nodes and their corresponding paths are dangerous. The reason is that they are insensitive to HT insertions, and they will not result any degradation in the overall timing performance of the original design due to HTs. We consider these dangerous paths to insert HTs. We assume that adversaries may not target other paths which holds a node with a slack time that is less than the threshold value we defined. The reason is that this kind of insertions can be easily detected by classical test methods.

Table 3. Experimental results

Benchmark	Circuit type	Inputs/Outputs	Gates/Levels	Considered paths / Upper bound	Case 1: PDF Coverage (Before HT insertion)	Case 2: PDF Coverage (After HT insertion)
c432	Channel Interrupt Controller	36/7	203/18	10K/132K	100%	84.2%
c880	8-bit ALU	60/26	469/25	10K/16652	100%	89.1%
c1355	32-bit SEC Circuit	41/32	619/25	10K/1110K	100%	82.3%
c1908	16-bit SEC Circuit	33/25	938/41	10K/355K	98.8%	78.1%
c2670	12-bit ALU and Controller	233/140	1566/33	10K/1306K	89.5%	72.2%
c3540	8-bit ALU	50/22	1741/48	10K/12330K	97.8%	75.4%
c5315	9-bit ALU	178/123	2608/50	10K/353K	98.7%	78.3%
c7552	32-bit adder/comparator	207/108	3827/44	10K/282K	98.3%	81.1%

Table 4. Analysis of PDF coverage-based HT Detection

Benchmark	PDF coverage variation in %		
	Statistical Simulation trials		
	VMax	VMin	VAve
c432	18.6%	15.1%	17.5%
c880	14.3%	11.9%	12.7%
c1355	19.3%	17.4%	19%
c1908	20.1%	19.2%	19.6%
c2670	18.4%	14.3%	17.8%
c3540	23.1%	18.6%	19.5%
c5315	21.6%	17.2%	18.9%
c7552	17.8%	16.1%	16.8%

Table 3 shows the experimental results for different benchmark circuits before and after HT insertions. The 10M test vectors were applied to each circuit. Each simulation was repeated ten times with different seed patterns. The PDF coverage values were logged each time after applied test vectors. However, to analyze further on statistical variations, and the quality of the seed patterns, the best, the worst and average case HT detection probabilities were considered, and the results are shown in Table 4. Table 4 gives the results for the statistical variation analysis over 10 trials. From the PDF coverage results of trial runs on each circuit, it presents values for 'VMax' (variation between the best-case values), 'VMin' (variation between the worst case values), and 'VAve' as (variation between the average values of before and after attacks). The experiment results conclude that the variation (before and after attacks) is observable such as 12.7% - 19.6%.

5.3.Discussion and Future Directions

Despite advancements in delay testing techniques, several challenges remain. Detecting small delay defects is still difficult, necessitating more sensitive testing methods. Additionally, generating effective test patterns for complex circuits is a complex task that requires automated solutions. Scalability poses another issue, as existing methods may struggle with larger circuit designs. The integration of hardware security is crucial, especially with the rising threat of hardware Trojans, and future research should explore combining delay testing with security measures. Furthermore, developing real-time testing techniques could provide immediate feedback on circuit performance. Lastly, cross-disciplinary collaboration with fields like machine learning may lead to innovative solutions for improving delay testing and enhancing hardware security.

6. CONCLUSIONS

Today's advanced ICs require a delay fault testing to ensure the manufactured circuits meet their timing specification. Development of efficient test generation and fault simulation algorithms for delay faults has been an active area of research in the last two decades. In this paper, a survey of related literature revealed that there is considerable scope for the development of path delay fault models for delay fault testing in combinational as well as sequential circuits. We discussed about the details of path delay fault test method, its principle, various paths selection and test generation procedures. We have also presented a new path delay fault coverage comparison of HT-free and infected integrated circuits. This proposed idea utilizes shortest paths of the circuit, and particularly targeted on paths that are prone to get attacked by adversaries. Our methodology does not require activation of the HT and does not require additional HT prevention mechanism (i.e. design-for trust circuits). The possibilities to detect HTs are very high using PDF coverage comparison method where the variation is more than 10% in all benchmarks under attacks.

REFERENCES

[1] I. Pomeranz and S. M. Reddy, "A delay fault model for at-speed fault simulation and test generation," in Proc. Int. Conf. Comput.-Aided Des. (ICCAD), San Jose, CA, USA, Nov. 2006, pp. 684–689. doi: 10.1145/1233501.1233521.

[2] N. Ahmed, M. Tehranipoor, and V. Jayaram, "Timing-based delay test for screening small delay defects," in *Proc. Annu. Des. Autom. Conf. (DAC)*, San Francisco, CA, USA, Jul. 2006, pp. 320–325. doi: 10.1145/1146909.1146993.

[3] I. Pomeranz, "Substituting transition faults with path delay faults as a basic delay fault model," in *Proc. Des., Autom. Test Eur. Conf. Exhib. (DATE)*, Dresden, Germany, Mar. 2014, pp. 1–6. doi: 10.7873/DATE.2014.241.

[4] S. Prasanna and K. Thanushkodi, "Implementation of delay measurement technique using signature register for small-delay defect detection," *IJRET: Int. J. Res. Eng. Technol.*, vol. 2, no. 4, pp. 446–450, Apr. 2013.

[5] J. Kõusaar, R. Ubar, S. Devadze, and J. Raik, "Transition delay fault simulation with parallel critical path back-tracing and 7-valued algebra," *Microprocess. Microsyst.*, vol. 39, no. 8, pp. 935– 945, May 2015. doi: 10.1016/j.micpro.2015.05.003.

[6] K. Heragu, J. H. Patel, and V. D. Agrawal, "Segment delay faults: A new fault model," in *Proc. IEEE VLSI Test Symp. (VTS)*, Apr. 1996, pp. 32–39. [Online]. Available: https://dl.acm.org/doi/10.5555/832296.836278.

[7] I. Pomeranz and S. M. Reddy, "Diagnosis of path delay faults based on low-coverage tests," *IET Comput. Digit. Tech.*, vol. 4, no. 2, pp. 124–133, Mar. 2010. doi: 10.1049/iet-cdt.2008.0154.

[8] Y. Higami, S. Wang, H. Takahashi, S. Kobayashi, and K. K. Saluja, "A method for diagnosing bridging fault between a gate signal line and a clock line," *IEICE Trans. Inf. Syst.*, vol. E100-D, no. 1, pp. 211–219, Jan. 2017. doi: 10.1587/transinf.2016EDL8210.

[9] Russell Tessier, "Understanding sequential circuit timing," University of Massachusetts Amherst, Jan. 2023. Available: http://www.ecs.umass.edu/ece/tessier/courses/221/lecture/timing-notes.pdf.

[10] K. Sreenath and S. R. Ramesh, "Statistical viability analysis and optimization through gate sizing," in *Proc. Int. Conf. Adv. Comput., Commun. Inform. (ICACCI)*, Bangalore, India, Sept. 2018, pp. 149–155. doi: 10.1007/978-981-10-8240-5_17.

[11] S. Abolmaali, M. Kamal, A. Afzali-Kusha, and M. Pedram, "An efficient false path-aware heuristic critical path selection method with high coverage of the process variation space," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 14, no. 3, pp. 1–21, Feb. 2018. doi: 10.1145/3177866.

[12] S. Kajihara, M. Fukunaga, X. Wen, T. Maeda, S. Hamada, and Y. Satô, "Path delay test compaction with process variation tolerance," in *Proc. Annu. Des. Autom. Conf. (DAC)*, San Diego, CA, USA, 2005, pp. 842–847. doi: 10.1145/1065579.1065802.

[13] M. Fukunaga, S. Kajihara, S. Takeoka, and S. Yosimura, "On effective criterion of path selection for delay testing," in *Proc. IEEE Asia South Pac. Des. Autom. Conf. (ASP-DAC)*, 2003, pp. 757–762. [Online]. Available: https://www.cs.york.ac.uk/rts/docs/SIGDA-Compendium-19942004/papers/2003/aspdac03/pdffiles/08c_2.pdf.

[14] M. Fukunaga, S. Kajihara, X. Wen, T. Maeda, S. Hamada, and Y. Satô, "A dynamic test compaction procedure for high-quality path delay testing," Jan. 1, 2006. doi: 10.1145/1118299.1118388.

[15] Ø. Gjermundnes and E. J. Aas, "Efficient stimuli generators for detection of path delay faults," in *Proc. Midwest Symp. Circuits Syst. (MWSCAS)*, 2005, pp. 1709–1712. doi: 10.1109/MWSCAS.2005.1594449.

[16] A. Makji, V. Agrawal, J. Jacob, and L. Patnaik, "Line coverage of path delay faults," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 8, no. 1, pp. 610–613, 2000. [Online]. Available: https://dl.acm.org/doi/abs/10.1109/92.894166.

[17] L. C. Chen, S. K. Gupta, and M. A. Breuer, "High quality robust tests for path delay faults," in *Proc. IEEE VLSI Test Symp. (VTS)*, Apr. 1997, pp. 88–93. doi: 10.1109/VTEST.1997.599447.

[18] E. S. Park and M. R. Mercer, "Robust and nonrobust tests for path delay faults in combinational circuits," in *Proc. IEEE Int. Test Conf. (ITC)*, Sep. 1987, pp. 1027–1034.

[19] I. Pomeranz, S. M. Reddy, and P. Uppaluri, "NEST: A nonenumerative test generation method for path delay faults in combinational circuits," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 14, no. 12, pp. 1505–1515, Dec. 1995. [Online]. Available: https://dl.acm.org/doi/abs/10.1109/43.476581.

[20] K. Fuchs, F. Fink, and M. H. Schulz, "DYNAMITE: An efficient automatic test pattern generation system for path delay faults," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 10, pp. 1323–1335, Aug. 2002. doi: 10.1109/43.88928.

[21] K. Fuchs, M. Pabst, and T. Rossel, "RESIST: A recursive test pattern generation algorithm for path delay faults considering various test classes," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 13, no. 12, pp. 1550–1562, Dec. 1994. [Online].

[22] H. Chang and J. A. Abraham, "VIPER: An efficient vigorously sensitizable path extractor," in *Proc. ACM/IEEE Des. Autom. Conf. (DAC)*, June 1993, pp. 112–117. [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/157485.158845.

[23] W. Qiu, J. Wang, D. M. H. Walker, D. Reddy, X. Lu, Z. Li, W. Shi, and H. Balachandran, "K longest paths per gate (KLPG) test generation for scan-based sequential circuits," in *Proc. IEEE Int. Test Conf. (ITC)*, 2004, pp. 223–231. [Online]. Available: https://people.engr.tamu.edu/dwalker/5yrPapers/ITC_KLPG_102004.pdf.

[24] W. Qiu and D. M. H. Walker, "An efficient algorithm for finding the K longest testable paths through each gate in a combinational circuit," in *Proc. IEEE Int. Test Conf. (ITC)*, 2003, pp. 592–601. [Online]. Available: https://people.engr.tamu.edu/dwalker/5yrPapers/ITC_KLPG_102003.pdf.

[25] J. Chung, J. Xiong, V. Zolotov, and J. A. Abraham, "Testability driven statistical path selection," in *Proc. ACM/IEEE Des. Autom. Conf. (DAC)*, San Diego, CA, USA, 2011, pp. 532–537. doi: 10.1145/2024724.2024822.

[26] M. Sharma and J. Patel, "Finding a small set of longest testable paths that cover every gate," in *Proc. IEEE Int. Test Conf. (ITC)*, 2002, pp. 974–982. doi: 10.1109/TEST.2002.1041853.

[27] W. Kunz and D. Pradhan, "Recursive learning: An attractive alternative to the decision tree for test generation in digital circuits," in *Proc. IEEE Int. Test Conf. (ITC)*, Sep. 1992, pp. 816–825. doi: 10.1109/TEST.1992.527905.

[28] V. Zolotov, J. Xiong, H. Fatemi, and C. Visweswariah, "Statistical path selection for at-speed test," in *Proc. IEEE Int. Conf. Comput.-Aided Des. (ICCAD)*, 2008, pp. 624–631. [Online]. Available: https://dl.acm.org/doi/10.5555/1509456.1509595.

[29] X. Lu, Z. Li, W. Qiu, D. M. H. Walker, and W. Shi, "Longest-path selection for delay test under process variation," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 24, pp. 1924–1929, 2005. [Online]. Available https://people.engr.tamu.edu/dwalker/5yrPapers/TCAD_pasta_02102005.pdf.

[30] J.-F. Lee and D. T. Tang, "An algorithm for incremental timing analysis," in *Proc. ACM/IEEE Des. Autom. Conf. (DAC)*, San Francisco, CA, USA, 1995, pp. 383–388. doi: 10.1145/217474.217613.

[31] W. Qiu and D. M. H. Walker, "Testing the path delay faults of ISCAS85 circuit c6288," in *Proc. IEEE Int. Workshop Memory Technol., Design, Test. (MTVCC)*, Dec. 2003, pp. 19–24. [Online]. Available: https://people.engr.tamu.edu/d-walker/5yrPapers/MTV_C6288_052003.pdf.

[32] K. T. Cheng and H. C. Chen, "Classifications and identification of non-robust untestable path delay faults," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 8, pp. 845–853, Aug. 1996. doi: 10.1109/43.511566.

[33] K.-T. Cheng and H.-C. Chen, "Generation of high-quality non-robust tests for path delay faults," in *Proc. ACM/IEEE Des. Autom. Conf. (DAC)*, Jun. 1994, pp. 365–369. [Online]. Available: https://dl.acm.org/doi/10.1145/196244.196423.

[34] S. Kundu, "An incremental algorithm for identification of longest (shortest) paths," *Integration, the VLSI Journal*, vol. 17, pp. 25–31, 1994. [Online]. Available: https://c.coek.info/pdf-anincremental-algorithm-for-identification-of-longest-shortest-paths-.html.

[35] C. Hartmann and M. Wieberneit, "Investigation on BIST assisted failure analysis on digital integrated circuits," *Microelectron. Rel.*, vol. 50, no. 9–11, pp. 1482–1485, Aug. 2010. doi: 10.1016/j.microrel.2010.07.148.

[36] V. P. Kolanchinathan and G. S. Kumar, "Design and implementation of the combinational circuits testing using accumulator-based BIST to reduce delay, power consumption and area," *Indian Soc. Educ. Environ.*, May 12, 2016. doi: 10.17485/ijst/2016/v9i16/92221.

[37] N. Li, G. Carlsson, E. Dubrova, and K. Petersén, "Logic BIST: State-of-the-art and open problems," *Cornell Univ.*, Mar. 16, 2015. doi: 10.48550/arXiv.1503.

[38] Y. Hau, S. M. Reddy, and I. Pomeranz, "Path delay fault test generation for standard scan designs using state tuples," in *Proc. IEEE ASP-DAC*, Jan. 2002, p. 767. [Online]. Available: https://dl.acm.org/doi/10.5555/832284.835496.

[39] M. Matsumoto and T. Nishimura, "Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator," *ACM Trans. Model. Comput. Simul.*, vol. 8, no. 1, pp. 3–30, 1998. [Online]. Available: https://dl.acm.org/doi/10.1145/272991.272995.

[40] G. L. Smith, "Model for delay faults based upon paths," *in Proc. IEEE Int. Test Conf. (ITC)*, Nov. 1985, pp. 342–349. [Online]. Available: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=4bd3e334fa5d2c6a817130e3c8cee5a69a6bc3e6.

[41] A. M. Somashekar, S. Tragoudas, R. Jayabharathi, and S. Gangadhar, "Non-enumerative generation of path delay distributions and its application to critical path selection," in *Proc. 34th Int. Conf. Comput.-Aided Design (ICCAD)*, Austin, TX, USA, 2016, pp. 156–162. doi: 10.1145/2940327.

[42] D. Kagaris and S. Tragoudas, "On the nonenumerative path delay fault simulation problem," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 21, no. 9, pp. 1095–1101, 2002. doi: 10.1109/TCAD.2002.801108.

[43] P. Manikandan, E. J. Aas, and B. B. Larsen, "Experiments with ABIST test methodology applied to path delay fault testing," in *Proc. IEEE Int. Conf. EWDTS-2010*, Sep. 2010, pp. 110–115. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5742129.

[44] S. Sridhar, K. Mounika, M. Anjali, G. Venkatesh, and M. M. Krishna, "VLSI implementation of linear feedback shift register (LFSR) based test pattern generator for pseudo exhaustive testing," *Blue Eyes Intell. Eng. Sci. Publ.*, May 20, 2020. doi: 10.35940/ijitee.g5624.059720.

[45] Ø. Gjermundnes, "Exploiting arithmetic built-in self-test techniques for path delay fault testing," Ph.D. dissertation, NTNU, Norway, Nov. 2006. [Online]. Available: https://ntnuopen.ntnu.no/ntnuxmlui/bitstream/handle/11250/2368813/124251_FULLTEXT01.pdf?sequence=1.

[46] J. Zhao, G. Ning, H. Lu, Y. B. Wang, Y. Cai, and J. Zhang, "A weight-based approach to combinatorial test generation," May 27, 2018. doi: 10.1145/3183440.3195018.

[47] P. Manikandan, E. J. Aas, and B. B. Larsen, "An enhanced path delay fault simulator for combinational circuits," in *Proc. IEEE Int. Conf. DSD-2011*, Sep. 2011, pp. 375–381. doi: 10.1109/DSD.2011.52.

[48] S. M. Sebt, A. Patooghy, H. Beitollahi, and M. A. Kinsy, "Circuit enclaves susceptible to hardware Trojans insertion at gate-level designs," *Inst. Eng. Technol.*, Sep. 14, 2018. doi: 10.1049/ietcdt.2018.5108.

[49] P. Behnam, "Validation of hardware security and trust: A survey," *Cornell Univ.*, Jan. 4, 2018. doi: 10.48550/arXiv.1801.

[50] A. D. Sontan and S. V. Samuel, "The intersection of artificial intelligence and cybersecurity: Challenges and opportunities," GSC Online Press, Feb. 28, 2024. doi: 10.30574/wjarr.2024.21.2.0607.

AUTHOR

Palanichamy Manikandan is an associate professor at Østfold University College, Norway within Institute of Engineering. Dr. Manikandan has over 12 years of experience after PhD, in both industries and academia. He has expertise in electrical and electronics systems development, cell designs and testing, industrial IoT systems, and lately focuses on sustainable engineering and green transition. He serves as a program committee member, associate editor and reviewer for several international conferences and journals including IEEE and Elsevier. He had completed two post doctorates from University of Montpellier-France and The Hongkong Polytechnic University-Hong Kong. He received his PhD in 2013 from Norwegian University of Science and Technology, Norway, and MS in 2007 from National Cheng University, Taiwan. He also served as a visiting researcher at the University of Iowa-USA. He has an extensive national and international network of collaboration partners, invited talks as a guest professor, contributing to research and industrial projects, with international work experience. He is also an active CDIO member and involves engineering education development in the network of more than 189 universities globally.