# *MERO*: A Statistical Approach for Hardware Trojan Detection[⋆]

Rajat Subhra Chakraborty[⋆⋆], Francis Wolff, Somnath Paul,
Christos Papachristou, and Swarup Bhunia

Department of Electrical Engineering and Computer Science,
Case Western Reserve University, Cleveland OH–44106, USA
rsc22@case.edu

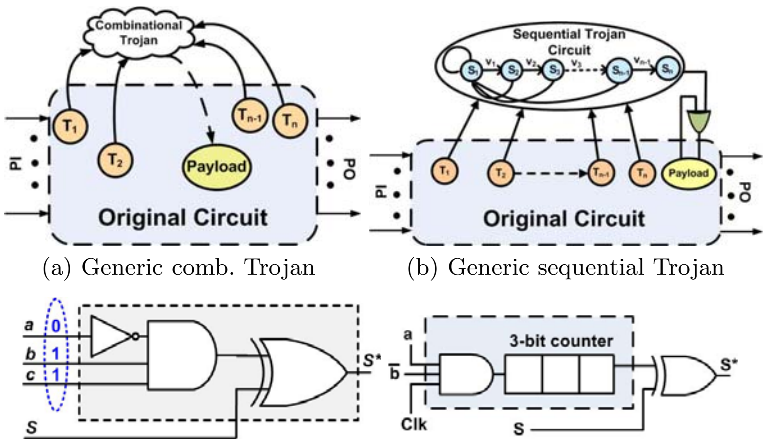
**Abstract.** In order to ensure trusted in–field operation of integrated circuits, it is important to develop efficient low–cost techniques to detect malicious tampering (also referred to as *Hardware Trojan*) that causes undesired change in functional behavior. Conventional post–manufacturing testing, test generation algorithms and test coverage metrics cannot be readily extended to hardware Trojan detection. In this paper, we propose a test pattern generation technique based on multiple excitation of rare logic conditions at internal nodes. Such a statistical approach maximizes the probability of inserted Trojans getting triggered and detected by logic testing, while drastically reducing the number of vectors compared to a weighted random pattern based test generation. Moreover, the proposed test generation approach can be effective towards increasing the sensitivity of Trojan detection in existing *side–channel* approaches that monitor the impact of a Trojan circuit on power or current signature. Simulation results for a set of ISCAS benchmarks show that the proposed test generation approach can achieve comparable or better Trojan detection coverage with about 85% reduction in test length on average over random patterns.

## 1 Introduction

The issue of *Trust* is an emerging problem in semiconductor integrated circuit (IC) security [1,2,3,8] This issue has become prominent recently due to widespread outsourcing of the IC manufacturing processes to untrusted foundries in order to reduce cost. An adversary can potentially tamper a design in these fabrication facilities by the insertion of malicious circuitry. On the other hand, third-party CAD tools as well as hardware intellectual property (IP) modules used in a design house also pose security threat in terms of incorporating malicious circuit into a design [3]. Such a malicious circuit, referred to as a *Hardware Trojan*, can trigger and affect normal circuit operation, potentially with catastrophic consequences in critical applications in the domains of communications, space, military and nuclear facilities.

[⋆] The work is funded in part by a DoD seedling grant FA-8650-08-1-7859.

[⋆⋆] Corresponding author.



(a) Generic comb. Trojan (b) Generic sequential Trojan

(c) Comb. Trojan example (d) Sequential Trojan example

**Fig. 1.** Generic model for combinational and sequential Trojan circuits and corresponding examples

An intelligent adversary will try to hide such tampering of IC's functional behavior in a way that makes it extremely difficult to detect with conventional post–manufacturing test [3]. Intuitively, it means that the adversary would ensure that such a tampering is manifested or *triggered* under very rare conditions at the internal nodes, which are unlikely to arise during test but can occur during long hours of field operation [13]. Fig. 1 shows general models and examples of hardware Trojans. The *combinational Trojans* as shown in Fig. 1(a) do not contain any sequential elements and depend only on the simultaneous occurrence of a set of rare node conditions (e.g. on nodes $T_1$ through node $T_n$) to trigger a malfunction. An example of a combinational Trojan is shown in Fig. 1(c) where the node $S$ has been modified to $S^{\star}$, and malfunction is triggered whenever the condition $a = 0, b = 1, c = 1$ is satisfied. The *sequential Trojans* shown in Fig. 1(b), on the other hand, undergo a sequence of state transitions ($S_1$ through $S_n$) before triggering a malfunction. An example is shown in Fig. 1(d), where the 3–bit counter causes a malfunction at the node $S$ on reaching a particular count, and the count is increased only when the condition $a = 1, b = 0$ is satisfied at the positive clock–edge. We refer to the condition of Trojan activation as the *triggering condition* and the node affected by the Trojan as its *payload*.

In order to detect the existence of a Trojan using logic testing, it is not only important to trigger a rare event at a set of internal nodes, but also to propagate the effect of such an event at the payload to an output node and observe it. Hence, it is very challenging to solve the problem of Trojan detection using conventional test generation and application, which are designed to detect manufacturing defects. In addition, the number of possible Trojan instances has a combinatorial dependence on the number of circuit nodes. As an example, even with the assumption of maximum 4 trigger nodes and a single payload, a relatively small

ISCAS–85 circuit such as *c880* with 451 gates can have $\sim 10^9$ triggers and $\sim 10^{11}$ distinct Trojan instances, respectively. Thus, it is not practical to enumerate all possible Trojan instances to generate test patterns or compute test coverage. This indicates that instead of an exact approach, a statistical approach for test vector generation for Trojans can be computationally more tractable.

In this paper, we propose a methodology, referred to as *MERO* (**M**ultiple **E**xcitation of **R**are **O**ccurence) for statistical test generation and coverage determination of hardware Trojans. The main objective of the proposed methodology is to derive a set of test patterns that is *compact* (minimizing test time and cost), while maximizing the Trojan detection coverage. The basic concept is to detect low probability conditions at the internal nodes, select candidate Trojans triggerable by a subset of these rare conditions, and then derive an optimal set of vectors than can trigger each of the selected low probability nodes *individually to their rare logic values multiple times* (e.g. at least $N$ times, where $N$ is a given parameter). As analyzed in Section 3.1, this increases the probability of detection of a Trojan having a subset of these nodes as its trigger nodes. By increasing the toggling of nodes that are random–pattern resistant, it improves the probability of activating a Trojan compared to purely random patterns. The proposed methodology is conceptually similar to *N-detect test* [5,6] used in stuck-at ATPG (automatic test pattern generation), where test set is generated to detect each single stuck-at fault in a circuit by at least $N$ different patterns, in the process improving test quality and defect coverage [6]. In this paper, we focus on digital Trojans [13], which can be inserted into a design either in a design house (e.g. by untrusted CAD tool or IP) or in a foundry. We do not consider the Trojans where the triggering mechanism or effect are analog (e.g. thermal).

Since the proposed detection is based on functional validation using logic values, it is robust with respect to parameter variations and can reliably detect very small Trojans, e.g. the ones with few logic gates. Thus, the technique can be used as *complementary to the side–channel Trojan detection approaches* [1,9,10,11] which are more effective in detecting large Trojans (e.g. ones with area $> 0.1\%$ of the total circuit area). Besides, the *MERO* approach can be used to increase the detection sensitivity of many side-channel Trojan detection techniques such as the ones that monitor the power/current signature, by increasing the activity in a Trojan circuit. Using an integrated Trojan coverage simulation and test generation flow, we validate the approach for a set of ISCAS combinational and sequential benchmarks. Simulation results show that the proposed test generation approach can be extremely effective for detecting arbitrary Trojan instances of small size, both combinational and sequential.

The rest of the paper is organized as follows. Section 2 describes previous work on Trojan detection. Section 3 describes the mathematical justification of the *MERO* methodology, the steps of the *MERO* test generation algorithm and the Trojan detection coverage estimation. Section 4 describes the simulation setup and presents results for a set of ISCAS benchmark circuits with detailed analysis. Section 5 concludes the paper.

## 2 Trojan Detection: Previous Work

Previously proposed Trojan detection approaches can be classified under two main classes: (1) destructive approaches and (2) non–destructive approaches. In the *destructive* approaches, the manufactured IC is de–metallized layer by layer, and chip microphotographs of the layers are integrated and analyzed by advanced software to detect any tampering [4]. However, the applicability of such approaches is limited by the fact that the hacker is most likely to modify only a small random sample of chips in the production line. This means that the success of detecting the Trojan depends totally on correctly selecting a manufactured IC instance that has actually been tampered. Also, destructive methods of validating an IC are extremely expensive with respect to time and cost and technology intensive, with validation of a single IC taking months [3]. Hence, it is important to investigate efficient non–destructive Trojan detection approaches.

Two non-destructive Trojan detection techniques can be categorized into two broad classes: (1) *Side-channel Analysis* based and (2) *Logic Testing* based techniques. The *side–channel* analysis based techniques utilize the effect of an inserted Trojan on a measurable physical quantity, e.g. the supply current [1,11] or path delays [10]. Such a measured circuit parameter can be referred as a *fingerprint* for the IC [1]. Side–channel approaches of detecting Trojans belong to a class of generic powerful techniques for IC authentication, and are conceptually applicable to Trojans of all operational modes and to designs of arbitrary size and complexity. Only local *activation* of the Trojans is sufficient to detect them, and methods have been proposed to maximize the possibility of locally activating Trojans [9]. However, there are two main issues with the side–channel based approaches that limit their practical applicability:

1. An intelligent adversary can craft a very small Trojan circuit with just a few logic gates which causes minimal impact on circuit power or delay. Thus it can easily evade side–channel detection techniques such as the ones described in [1,10].
2. The fingerprint is extremely vulnerable to process variations (i.e. *process noise*) and measurement noise. Even advanced de–noising techniques such as those applied in [1] fail to detect arbitrarily small Trojans under process variations.

Logic testing based approaches, on the other hand, are extremely reliable under process variations and measurement noise effects. An important challenge in these approaches is the inordinately large number of possible Trojans an adversary can exploit. Relatively few works have addressed the problem of Trojan detection using logic testing. In [12], a design methodology was proposed where special circuitry was embedded in an IC to improve the controllability and observability of internal nodes, thereby facilitating the detection of inserted Trojans by logic testing. However, this technique does not solve the problem of detecting Trojans in ICs which have not been designed following that particular design methodology.

## 3 Statistical Approach for Trojan Detection

As described in Section 1, the main concept of our test generation approach is based on generating test vectors that can excite candidate trigger nodes individually to their rare logic values multiple (at least $N$) times. In effect, the probability of activation of a Trojan by the simultaneous occurrence of the rare conditions at its trigger nodes increases. As an example, consider the Trojan shown in Fig. 1(c). Assume that the conditions $a = 0$, $b = 1$ and $c = 1$ are very rare. Hence, if we can generate a set of test vectors that induce these rare conditions at these nodes individually $N$ times where $N$ is sufficiently large, then a Trojan with triggering condition composed jointly of these nodes is highly likely to be activated by the application of this test set. The concept can be extended to sequential Trojans, as shown in Fig. 1(d), where the inserted 3–bit counter is clocked on the simultaneous occurrence of the condition $ab' = 1$. If the test vectors can sensitize these nodes such that the condition $ab' = 1$ is satisfied at least 8 times (the maximum number of states of a 3–bit counter), then the Trojan would be activated. Next, we present a mathematical analysis to justify the concept.

### 3.1 Mathematical Analysis

Without loss of generality, assume that a Trojan is triggered by the rare logic values at two nodes $A$ and $B$, with corresponding probability of occurrence $p_1$ and $p_2$. Assume $T$ to be the total number of vectors applied to the circuit under test, such that both $A$ and $B$ have been individually excited to their rare values *at least* $N$ times. Then, the expected number of occurrences of the rare logic values at nodes $A$ and $B$ are given by $E_A = T \cdot p_1 \geq N$ and $E_B = T \cdot p_2 \geq N$, which lead to:

$$T \geq \frac{N}{p_1} \quad \text{and} \quad T \geq \frac{N}{p_2} \tag{1}$$

Now, let $p_j$ be the probability of simultaneous occurrence of the rare logic values at nodes $A$ and $B$, an event that acts as the trigger condition for the Trojan. Then, the expected number of occurrences of this event when $T$ vectors are applied is:

$$E_{AB} = p_j \cdot T \tag{2}$$

In the context of this problem, we can assume $p_j > 0$, because an adversary is unlikely to insert a Trojan which would never be triggered. Then, to ensure that the Trojan is triggered at least once when $T$ test vectors are applied, the following condition must be satisfied:

$$p_j \cdot T \geq 1 \tag{3}$$

From inequality (1), let us assume $T = c \cdot \frac{N}{p_1}$. where $c \geq 1$ is a constant depending on the actual test set applied. Inequality (3) can then be generalized as:

$$S = c \cdot \frac{p_j}{p_1} \cdot N \tag{4}$$

where $S$ denotes the number of times the trigger condition is satisfied during the test procedure. From this equation, the following observations can be made about the interdependence of $S$ and $N$:

1. For given parameters $c$, $p_1$ and $p_j$, $S$ is proportional to $N$, i.e. the expected number of times the Trojan trigger condition is satisfied increases with the number of times the trigger nodes have been individually excited to their rare values. This observation forms the main motivation behind the *MERO* test generation approach for Trojan detection.
2. If there are $q$ trigger nodes and if they are assumed to be mutually independent, then $p_j = p_1 \cdot p_2 \cdot p_3 \cdots p_q$, which leads to:
$$S = c \cdot N \cdot \prod_{i=2}^{q} p_i \tag{5}$$
As $p_i < 1 \quad \forall i = 1, 2, \cdots q$, hence, with the increase in $q$, $S$ decreases for a given $c$ and $N$. In other words, with the increase in the number of trigger nodes, it becomes more difficult to satisfy the trigger condition of the inserted Trojan for a given $N$. Even if the nodes are not mutually independent, a similar dependence of $S$ on $q$ is expected.
3. The trigger nodes can be chosen such that $p_i \leq \theta \quad \forall i = 1, 2, \cdots q$, so that $\theta$ is defined as a *trigger threshold* probability. Then as $\theta$ increases, the corresponding selected rare node probabilities are also likely to increase. This will result in an increase in $S$ for a given $T$ and $N$ i.e. the probability of Trojan activation would increase if the individual nodes are more likely to get triggered to their rare values.

All of the above predicted trends were observed in our simulations, as shown in Section 4.

## 3.2 Test Generation

Algorithm 1 shows the major steps in the proposed reduced test set generation process for Trojan detection. We start with the golden circuit netlist (without any Trojan), a random pattern set ($V$), list of rare nodes ($L$) and number of times to activate each node to its rare value ($N$). First, the circuit netlist is read and mapped to a *hypergraph*. For each node in $L$, we initialize the number of times a node encounters a rare value ($A_R$) to 0. Next, for each random pattern $v_i$ in $V$, we count the number of nodes ($C_R$) in $L$ whose rare value is satisfied. We sort the random patterns in decreasing order of $C_R$. In the next step, we consider each vector in the sorted list and modify it by perturbing one bit at a time. If a modified test pattern increases the number of nodes satisfying their rare values, we accept the pattern in the reduced pattern list. In this step we consider only those rare nodes with $A_R < N$. The process repeats until each node in $L$ satisfies its rare value at least $N$ times. The output of the test generation process is a minimal test set that improves the coverage for both combinational and sequential Trojans compared to random patterns.

**Algorithm 1. Procedure *MERO***
Generate reduced test pattern set for Trojan detection

**Inputs:** Circuit netlist, list of rare nodes ($L$) with associated rare values, list of random patterns ($V$), number of times a rare condition should be satisfied ($N$)
**Outputs:** Reduced pattern set ($R_V$)

```
1: Read circuit and generate hypergraph
2: for all nodes in $L$ do
3:    set number of times node satisfies rare value ($A_R$) to 0
4: end for
5: set $R_V = \Phi$
6: for all random pattern in $V$ do
7:    Propagate values
8:    Count the # of nodes ($C_R$) in $L$ with their rare value satisfied
9: end for
10: Sort vectors in $V$ in decreasing order of $C_R$
11: for all vector $v_i$ in decreasing order of $C_R$ do
12:    for all bit in $v_i$ do
13:       Perturb the bit and re-compute # of satisfied rare values ($C_R^{'}$)
14:       if ($C_R^{'} > C_R$) then
15:          Accept the perturbation and form $v_i^{'}$ from $v_i$
16:       end if
17:    end for
18:    Update $A_R$ for all nodes in $L$ due to vector $v_i$
19:    if $v_i^{'}$ increases $A_R$ for at least one rare node then
20:       Add the modified vector $v_i^{'}$ to $R_V$
21:    end if
22:    if ($A_R \geq N$) for all nodes in $L$ then
23:       break
24:    end if
25: end for
```

### 3.3 Coverage Estimation

Once the reduced test vector set has been obtained, computation of Trigger and Trojan coverage can be performed for a given *trigger threshold* ($\theta$) (as defined in Section 3.1) and a given number of trigger nodes ($q$) using a random sampling approach. From the Trojan population, we randomly select a number of $q$–trigger Trojans, where each trigger node has signal probability less than equal $\theta$. We assume that Trojans comprising of trigger nodes with higher signal probability than $\theta$ will be detected by conventional test. From the set of sampled Trojans, Trojans with false trigger conditions which cannot be justified with any input pattern are eliminated. Then, the circuit is simulated for each vector in the given vector set and checked whether the trigger condition is satisfied. For an activated Trojan, if its effect can be observed at the primary output or scan flip-flop input, the Trojan is considered "covered", i.e. detected. The percentages of Trojans activated and detected constitute the *trigger coverage* and *Trojan coverage*, respectively.

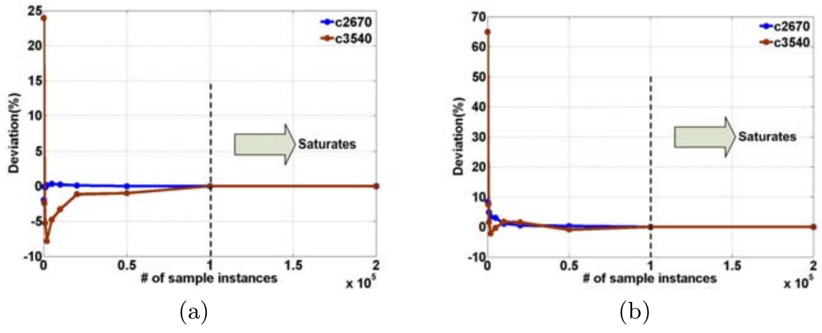


**Fig. 2.** Impact of sample size on trigger and Trojan coverage for benchmarks c2670 and c3540, $N = 1000$ and $q = 4$: (a) deviation of trigger coverage, and (b) deviation of Trojan coverage

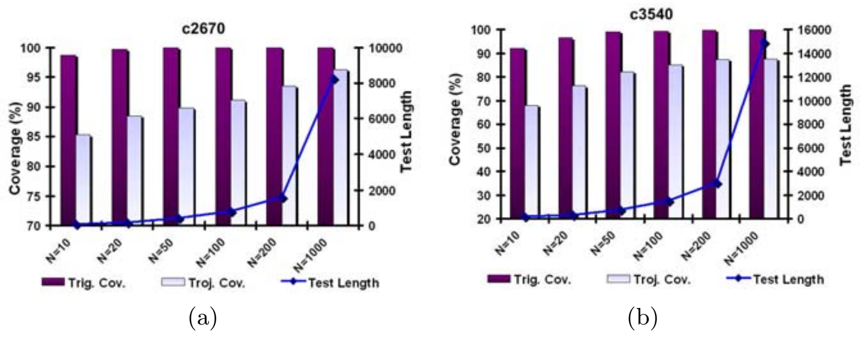


**Fig. 3.** Impact of $N$ (number of times a rare point satisfies its rare value) on the trigger/Trojan coverage and test length for benchmarks (a) c2670 and (b) c3540

## 3.4 Choice of Trojan Sample Size

In any random sampling process an important decision is to select the sample size in a manner that represents the population reasonably well. In the context of Trojan detection, it means further increase in sampled Trojans, renders negligible change in the estimated converge. Fig. 2 shows a plot of percentage deviation of Trigger and Trojan coverage ($q = 4$) from the asymptotic value for two benchmark circuits with varying Trojan sample size. From the plots, we observe that the coverage saturates with nearly 100,000 samples, as the percentage deviation tends to zero. To compromise between accuracy of estimated coverage and simulation time, we have selected a sample size of 100,000 in our simulations.

## 3.5 Choice of $N$

Fig. 3 shows the trigger and Trojan coverage for two ISCAS–85 benchmark circuits with increasing values of $N$, along with the lengths of the corresponding testset. From these plots it is clear that similar to *N–detect* tests for stuck-at fault where defect coverage typically improves with increasing $N$, the trigger and Trojan coverage obtained with the *MERO* approach also improves steadily with $N$, but then both saturate around $N = 200$ and remain nearly constant for larger values of $N$. As expected, the test size also increases with increasing $N$. We chose a value of $N = 1000$ for most of our experiments to reach a balance between coverage and test vector set size.

### 3.6 Improving Trojan Detection Coverage

As noted in previous sections, Trojan detection using logic testing involves simultaneous triggering of the Trojan and the propagation of its effect to output nodes. Although the proposed test generation algorithm increases the probability of Trojan activation, it does not explicitly target increasing the probability of a malicious effect at payload being observable. *MERO* test patterns, however, achieves significant improvement in Trojan coverage compared to random patterns, as shown in Section 4. This is because the Trojan coverage has strong correlation with trigger coverage. To increase the Trojan coverage further, one can use the following low-overhead approaches.

1. *Improvement of test quality*: We can consider number of nodes observed along with number of nodes triggered for each vector during test generation. This means, at step 13-14 of Algorithm 1, a perturbation is accepted if the sum of triggered and observed nodes improves over previous value. This comes at extra computational cost to determine the number of observable nodes for each vector. We note that for a small ISCAS benchmark *c432* (an interrupt controller), we can improve the Trojan coverage by 6.5% with negligible reduction in trigger coverage using this approach.
2. *Observable test point insertion*: We note that insertion of very few observable test points can achieve significant improvement in Trojan coverage at the cost of small design overhead. Existing algorithm for selecting observable test points for stuck-at fault test [14] can be used here. Our simulation with *c432* resulted in about 4% improvement in Trojan coverage with 5 judiciously inserted observable points.
3. *Increasing N and/or increasing the controllability of the internal nodes*: Internal node controllability can be increased by judiciously inserting few controllable test points or increasing $N$. It is well-known in the context of stuck-at ATPG, that scan insertion improves both controllability and observability of internal nodes. Hence, the proposed approach can take advantage of low-overhead design modifications to increase the effectiveness of Trojan detection.

## 4 Results

### 4.1 Simulation Setup

We have implemented the test generation and the Trojan coverage determination in three separate C programs. All the three programs can read a Verilog netlist

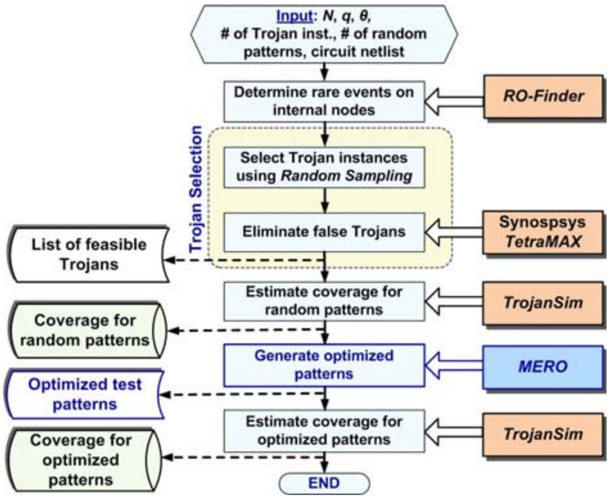

Input: N, q, θ, # of Trojan inst., # of random patterns, circuit netlist
Determine rare events on internal nodes
RO-Finder
Trojan Selection
Select Trojan instances using Random Sampling
Eliminate false Trojans
Synospsys TetraMAX
List of feasible Trojans
Estimate coverage for random patterns
TrojanSim
Coverage for random patterns
Generate optimized patterns
MERO
Optimized test patterns
Estimate coverage for optimized patterns
TrojanSim
Coverage for optimized patterns
END

**Fig. 4.** Integrated framework for rare occurrence determination, test generation using *MERO* approach, and Trojan simulation

**Table 1.** Comparison of Trigger and Trojan coverage among ATPG patterns [7], Random (100K, input weights: 0.5), and *MERO* patterns for $q = 2$ and $q = 4$, $N = 1000$, $\theta = 0.2$

| **Ckt.** | **Nodes (Rare/ Tot.)** | **ATPG patterns** | | | | **Random (100K patterns)** | | | | ***MERO* Patterns** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **q = 2** | | **q = 4** | | **q = 2** | | **q = 4** | | **q = 2** | | **q = 4** | |
| | | Trig. Cov. (%) | Troj. Cov. (%) | Trig. Cov. (%) | Troj. Cov. (%) | Trig. Cov. (%) | Troj. Cov. (%) | Trig. Cov. (%) | Troj. Cov. (%) | Trig. Cov. (%) | Troj. Cov. (%) | Trig. Cov. (%) | Troj. Cov. (%) |
| c2670 | 297/1010 | 93.97 | 58.38 | 30.7 | 10.48 | 98.66 | 53.81 | 92.56 | 30.32 | 100.00 | 96.33 | 99.90 | 90.17 |
| c3540 | 580/1184 | 77.87 | 52.09 | 16.07 | 8.78 | 99.61 | 86.5 | 90.46 | 69.48 | 99.81 | 86.14 | 87.34 | 64.88 |
| c5315 | 817/2485 | 92.06 | 63.42 | 19.82 | 8.75 | 99.97 | 93.58 | 98.08 | 79.24 | 99.99 | 93.83 | 99.06 | 78.83 |
| c6288 | 199/2448 | 55.16 | 50.32 | 3.28 | 2.92 | 100.00 | 98.95 | 99.91 | 97.81 | 100.00 | 98.94 | 92.50 | 89.88 |
| c7552 | 1101/3720 | 82.92 | 66.59 | 20.14 | 11.72 | 98.25 | 94.69 | 91.83 | 83.45 | 99.38 | 96.01 | 95.01 | 84.47 |
| s13207[‡] | 865/2504 | 82.41 | 73.84 | 27.78 | 27.78 | 100 | 95.37 | 88.89 | 83.33 | 100.00 | 94.68 | 94.44 | 88.89 |
| s15850[‡] | 959/3004 | 25.06 | 20.46 | 3.80 | 2.53 | 94.20 | 88.75 | 48.10 | 37.98 | 95.91 | 92.41 | 79.75 | 68.35 |
| s35932[‡] | 970/6500 | 87.06 | 79.99 | 35.9 | 33.97 | 100.00 | 93.56 | 100.00 | 96.80 | 100.00 | 93.56 | 100.00 | 96.80 |
| **Avg.** | **724/2857** | **74.56** | **58.14** | **19.69** | **13.37** | **98.84** | **88.15** | **88.73** | **72.30** | **99.39** | **93.99** | **93.50** | **82.78** |

[‡]These sequential benchmarks were run with 10,000 random Trojan instances to reduce run time of *Tetramax*.

and create a *hypergraph* from the netlist description. The first program, named as *RO-Finder* (**R**are **O**ccurence **Finder**), is capable of functionally simulating a netlist for a given set of input patterns, computing the signal probability at each node and identifying nodes with low signal probability as rare nodes. The second program *MERO* implements algorithm-1 described in Section 3.2 to generate the reduced pattern set for Trojan detection. The third program, *TrojanSim* (**Trojan Sim**ulator), is capable of determining both Trigger and Trojan coverage for a given test set using random sample of Trojan instances. A $q$-trigger random Trojan instance is created by randomly selecting the trigger nodes from the list of rare nodes. We consider one randomly selected payload node for each Trojan. Fig. 4 shows the flow-chart for the *MERO* methodology. Synopsys *TetraMAX* was used to justify the trigger condition for each Trojan and eliminate the false Trojans. All simulations and test generation were carried out on a Hewlett-Packard Linux workstation with a 2GHz dual-core Intel processor and 2GB RAM.

## 4.2 Comparison with Random and ATPG Patterns

Table 1 lists the trigger and Trojan coverage results for a set of combinational (ISCAS-85) and sequential (ISCAS-89) benchmarks using stuck-at ATPG patterns (generated using the algorithm in [7]), weighted random patterns and *MERO* test patterns. It also lists the number of total nodes in the circuit and the number of rare nodes identified by *RO-Finder* tool based on signal probability. The signal probabilities were estimated through simulations with a set of 100,000 random vectors. For the sequential circuits, we assume full-scan implementation. We consider 100,000 random instances of Trojans following the sampling policy described in Section 3.4, with one randomly selected payload node for each Trojan. Coverage results are provided in each case for two different trigger point count, $q = 2$ and $q = 4$, at $N = 1000$ and $\theta = 0.2$.

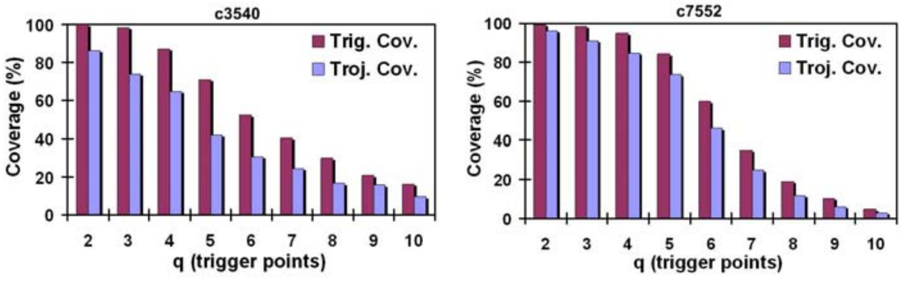
Table 2 compares reduction in the length of the testset generated by the *MERO* test generation method with 100,000 random patterns, along with the corresponding run-times for the test generation algorithm. This run-time includes the execution time for *Tetramax* to validate 100,000 random Trojan instances, as well as time to determine the coverage by logic simulation. We can make the following important observations from these two tables:

1. The stuck-at ATPG patterns provide poor trigger and Trojan coverage compared to *MERO* patterns. The increase in coverage between the ATPG and *MERO* patterns is more significant in case of higher number of trigger points.
2. From Table 2, it is evident that the reduced pattern with $N$=1000 and $\theta$ = 0.2 provides comparable trigger coverage with significant reduction in test

**Table 2.** Reduction in test length with MERO approach compared to 100K random patterns along with runtime, $q = 2$, $N$=1000, $\theta$=0.2

| **Ckt.** | ***MERO* test length** | **% Reduction** | **Run-time (s)** |
|---|---|---|---|
| c2670 | 8254 | **91.75** | 30051.53 |
| c3540 | 14947 | **85.05** | 9403.11 |
| c5315 | 10276 | **89.72** | 80241.52 |
| c6288 | 5014 | **94.99** | 15716.42 |
| c7552 | 12603 | **87.40** | 160783.37 |
| s13207† | 26926 | **73.07** | 23432.04 |
| s15850† | 32775 | **67.23** | 39689.63 |
| s35932† | 5480 | **94.52** | 29810.49 |
| **Avg.** | **14534** | **85.47** | **48641.01** |

†These sequential benchmarks were run with 10,000 random Trojan instances to reduce run time of *Tetramax*.



(a) (b)

**Fig. 5.** Trigger and Trojan coverage with varying number of trigger points ($q$) for benchmarks (a) c3540 and (b) c7552, at $N = 1000$, $\theta = 0.2$

length. The average improvement in test length for the circuits considered is about 85%.

3. Trojan coverage is consistently smaller compared to trigger coverage. This is because in order to detect a Trojan by applying an input pattern, besides satisfying the trigger condition, one needs to propagate the logic error at the payload node to one or more primary outputs. In many cases although the trigger condition is satisfied, the malicious effect does not propagate to outputs. Hence, the Trojan remains triggered but undetected.

### 4.3 Effect of Number of Trigger Points ($q$)

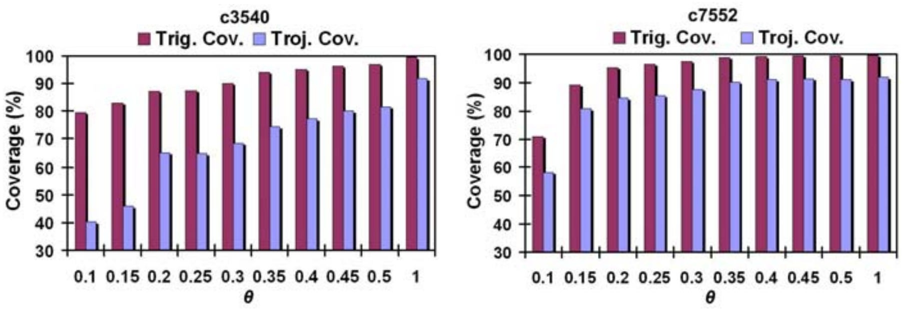
The impact of $q$ on coverage is evident from the Fig. 5, which shows the decreasing trigger and Trojan coverage with the increasing number of trigger points for two combinational benchmark circuits. This trend is expected from the analysis of Section 3.1. Our use of *TetraMAX* for justification and elimination of the false triggers helped to improve the Trojan coverage.

### 4.4 Effect of Trigger Threshold ($\theta$)

Fig. 6 plots the trigger and Trojan coverage with increasing $\theta$ for two ISCAS-85 benchmarks, at $N = 1000$ and $q = 4$. As we can observe, the coverage values improve steadily with increasing $\theta$ while saturating at a value above 0.20 in both the cases. The improvement in coverage with $\theta$ is again consistent with the conclusions from the analysis of Section 3.1.

### 4.5 Sequential Trojan Detection

To investigate the effectiveness of the *MERO* test generation methodology in detecting sequential Trojans, we designed and inserted sequential Trojans modeled following Fig. 1(d), with 0, 2, 4, 8, 16 and 32 states, respectively (the case with zero states refers to a combinational Trojan following the model of Fig. 1(c)). A cycle-accurate simulation was performed by our simulator *TrojanSim*, and the



(a) (b)

**Fig. 6.** Trigger and Trojan coverage with *trigger threshold* ($\theta$) for benchmarks (a) c3540 and (b) c7552, for $N = 1000$, $q = 4$

**Table 3.** Comparison of sequential Trojan coverage between random (100K) and MERO patterns, $N = 1000$, $\theta = 0.2$, $q = 2$

| **Ckt.** | **Trigger Cov. for 100K Random Vectors (%)** | | | | | | **Trigger Cov. for *MERO* Vectors (%)** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **Trojan State Count** | | | | | | **Trojan State Count** | | | | | |
| | 0 | 2 | 4 | 8 | 16 | 32 | 0 | 2 | 4 | 8 | 16 | 32 |
| s13207 | 100.00 | 100.00 | 99.77 | 99.31 | 99.07 | 98.38 | 100.00 | 100.00 | 99.54 | 99.54 | 98.84 | 97.92 |
| s15850 | 94.20 | 91.99 | 86.79 | 76.64 | 61.13 | 48.59 | 95.91 | 95.31 | 94.03 | 91.90 | 87.72 | 79.80 |
| s35932 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 |
| **Avg.** | **98.07** | **97.33** | **95.52** | **91.98** | **86.73** | **82.32** | **98.64** | **98.44** | **97.86** | **97.15** | **95.52** | **92.57** |

| **Ckt.** | **Trojan Cov. for 100K Random Vectors (%)** | | | | | | **Trojan Cov. for *MERO* Vectors (%)** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **Trojan State Count** | | | | | | **Trojan State Count** | | | | | |
| | 0 | 2 | 4 | 8 | 16 | 32 | 0 | 2 | 4 | 8 | 16 | 32 |
| s13207 | 95.37 | 95.37 | 95.14 | 94.91 | 94.68 | 93.98 | 94.68 | 94.68 | 94.21 | 94.21 | 93.52 | 92.82 |
| s15850 | 88.75 | 86.53 | 81.67 | 72.89 | 58.4 | 46.97 | 92.41 | 91.99 | 90.62 | 88.75 | 84.23 | 76.73 |
| s35932 | 93.56 | 93.56 | 93.56 | 93.56 | 93.56 | 93.56 | 93.56 | 93.56 | 93.56 | 93.56 | 93.56 | 93.56 |
| **Avg.** | **92.56** | **91.82** | **90.12** | **87.12** | **82.21** | **78.17** | **93.55** | **93.41** | **92.80** | **92.17** | **90.44** | **87.70** |

Trojan was considered detectable only when the output of the golden circuit and the infected circuit did not match. Table 3 presents the trigger and Trojan coverage respectively obtained by 100,000 randomly generated test vectors and the *MERO* approach for three large ISCAS-89 benchmark circuits. The superiority of the *MERO* approach over the random test vector generation approach in detecting sequential Trojans is evident from this table.

Although these results have been presented for a specific type of sequential Trojans (counters which increase their count conditionally), they are representative of other sequential Trojans whose state transition graph (STG) has no "loop". The STG for such a FSM has been shown in Fig. 7. This is a 8-state FSM which changes its state only when a particular internal node condition $C_i$ is satisfied at state $S_i$, and the Trojan is triggered when the FSM reaches state $S_8$. The example Trojan shown in Fig. 1(d) is a special case of this model, where the conditions $C_1$ through $C_8$ are identical. If each of the conditions $C_i$ is as rare as the condition $a = 1, b = 0$ required by the Trojan shown in Fig. 1(d),

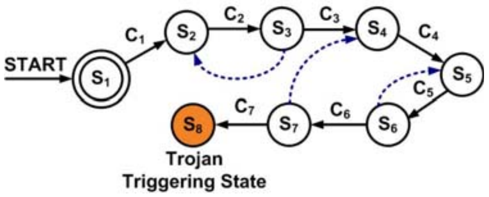


**Fig. 7.** FSM model with no loop in state transition graph

then there is no difference between these two Trojans as far as their rareness of getting triggered is concerned. Hence, we can expect similar coverage and test length results for other sequential Trojans of this type. However, the coverage may change if the FSM structure is changed (as shown with dotted line). In this case, the coverage can be controlled by changing $N$.

## 4.6 Application to Side-Channel Analysis

As observed from the results presented in this section, the *MERO* approach can achieve high trigger coverage for both combinational and sequential Trojans. This essentially means that the *MERO* patterns will induce activity in the Trojan triggering circuitry with high probability. A minimal set of patterns that is highly likely to cause activity in a Trojan is attractive in power or current signature based side-channel approach to detect hardware Trojan [1,9,11]. The detection sensitivity in these approaches depends on the induced activity in the Trojan circuit by applied test vector. It is particularly important to enhance sensitivity for the Trojans where the leakage contribution to power by the Trojan circuit can be easily masked by process or measurement noise. Hence, *MERO* approach can be extended to generate test vectors for side-channel analysis, which requires amplifying the Trojan impact on side-channel parameter such as power or current.

## 5 Conclusions

Conventional logic test generation techniques cannot be readily extended to detect hardware Trojans because of the inordinately large number of possible Trojan instances. We have presented a statistical Trojan detection approach using logic testing where the concept of multiple excitation of rare logic values at internal nodes is used to generate test patterns. Simulation results show that the proposed test generation approach achieves about 85% reduction in test length over random patterns for comparable or better Trojan detection coverage. The proposed detection approach can be extremely effective for small combinational and sequential Trojans with small number of trigger points, for which side-channel analysis approaches cannot work reliably. Hence, the proposed detection approach can be used as complementary to side-channel analysis based detection schemes. Future work will involve improving the test quality which will help in minimizing the test length and increasing Trojan coverage further.

## References

1. Agrawal, D., Baktir, S., Karakoyunlu, D., Rohatgi, P., Sunar, B.: Trojan detection using IC fingerprinting. In: IEEE Symp. on Security and Privacy, pp. 296–310 (2007)
2. Ravi, S., Raghunathan, A., Chakradhar, S.: Tamper resistance mechanisms for secure embedded systems. In: Intl. Conf. on VLSI Design, pp. 605–611 (2006)
3. DARPA BAA06-40: TRUST for Integrated Circuits, `http://www.darpa.mil/BAA/BAA06-40mod1/html`
4. Kumagai, J.: Chip Detectives. IEEE Spectrum 37, 43–49 (2000)
5. Amyeen, M.E., Venkataraman, S., Ojha, A., Lee, S.: Evaluation of the Quality of N-Detect Scan ATPG Patterns on a Processor. In: Intl. Test Conf., pp. 669–678 (2004)
6. Pomeranz, I., Reddy, S.M.: A Measure of Quality for n-Detection Test Sets. IEEE. Trans. on Computers. 53, 1497–1503 (2004)
7. Mathew, B., Saab, D.G.: Combining multiple DFT schemes with test generation. IEEE Trans. on CAD. 18, 685–696 (1999)
8. Adee, S.: The Hunt for the Kill Switch. IEEE Spectrum 45, 34–39 (2008)
9. Banga, M., Hsiao, M.S.: A Region Based Approach for the Identification of Hardware Trojans. In: Intl. Workshop on Hardware-oriented Security and Trust, pp. 40–47 (2008)
10. Jin, Y., Makris, Y.: Hardware Trojan Detection Using Path Delay Fingerprint. In: Intl. Workshop on Hardware-oriented Security and Trust, pp. 51–57 (2008)
11. Rad, R.M., Wang, X., Tehranipoor, M., Plusqellic, J.: Power Supply Signal Calibration Techniques for Improving Detection Resolution to Hardware Trojans. In: Intl. Conf. on CAD, pp. 632–639 (2008)
12. Chakraborty, R.S., Paul, S., Bhunia, S.: On-Demand Transparency for Improving Hardware Trojan Detectability. In: Intl. Workshop on Hardware-oriented Security and Trust, pp. 48–50 (2008)
13. Wolff, F., Papachristou, C., Bhunia, S., Chakraborty, R.S.: Towards Trojan-Free Trusted ICs: Problem Analysis and Detection Scheme. In: Design, Automation and Test in Europe, pp. 1362–1365 (2008)
14. Geuzebroek, M.J., Van der Linden, J.T., Van de Goor, A.J.: Test Point Insertion that Facilitates ATPG in Reducing Test Time and Data Volume. In: Intl. Test Conf., pp. 138–147 (2002)