MERS: Statistical Test Generation for Side-Channel Analysis based Trojan Detection

Yuanwen Huang
Department of Computer and Information Science and Engineering,
University of Florida,
Florida 32611, USA
yuanwenhuang@ufl.edu

Swarup Bhunia
Department of Electrical and Computer Engineering,
University of Florida,
Florida 32611, USA
swarup@ece.ufl.edu

Prabhat Mishra
Department of Computer and Information Science and Engineering,
University of Florida,
Florida 32611, USA
prabhat@ufl.edu

ABSTRACT

Hardware Trojan detection has emerged as a critical challenge to ensure security and trustworthiness of integrated circuits. A vast majority of research efforts in this area has utilized side-channel analysis for Trojan detection. Functional test generation for logic testing is a promising alternative but it may not be helpful if a Trojan cannot be fully activated or the Trojan effect cannot be propagated to the observable outputs. Side-channel analysis, on the other hand, can achieve significantly higher detection coverage for Trojans of all types/sizes, since it does not require activation/propagation of an unknown Trojan. However, they have often limited effectiveness due to poor detection sensitivity under large process variations and small Trojan footprint in side-channel signature. In this paper, we address this critical problem through a novel side-channel-aware test generation approach, based on a concept of Multiple Excitation of Rare Switching (MERS), that can significantly increase Trojan detection sensitivity. The paper makes several important contributions: i) it presents in detail the statistical test generation method, which can generate high-quality testset for creating high relative activity in arbitrary Trojan instances; ii) it analyzes the effectiveness of generated testset in terms of Trojan coverage; and iii) it describes two judicious reordering methods can further tune the testset and greatly improve the side channel sensitivity. Simulation results demonstrate that the tests generated by MERS can significantly increase the Trojans sensitivity, thereby making Trojan detection effective using side-channel analysis.

CCS Concepts

•**Hardware** → **Hardware test;** *Very large scale integration design; Hardware validation;*

CCS'16, October 24-28, 2016, Vienna, Austria

DOI: http://dx.doi.org/10.1145/2976749.2978396

Keywords

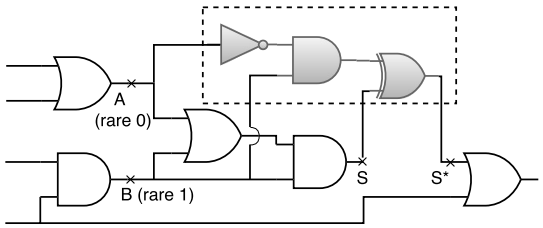
Hardware Security; Hardware Trojan Detection; Side-Channel Analysis; Statistical Test Generation.

1. INTRODUCTION

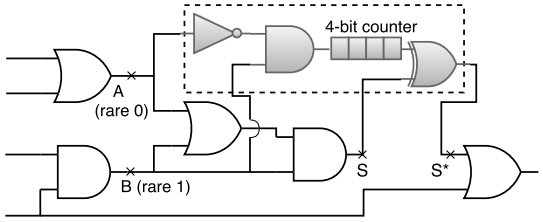
Hardware Trojan attacks relate to malicious modifications in the design of Integrated Circuits (ICs) at different stages of the design or fabrication process [1]. An adversary can introduce these modifications in a design in order to cause disruption in normal functional behavior and/or to leak secret information from a chip during operation in field. With the emerging trend of increased globalization of IC design and fabrication process and consequently reduced control on these steps by a trusted chip manufacturer, ICs are becoming increasingly vulnerable to these attacks. Since the threat of hardware Trojan in the form of a malicious implant in a design came into light about a decade ago through an US Department of Defense announcement [2], it has triggered wide array of research activities in threat analysis as well as design/validation solutions to evaluate this threat and protect against it. Hardware Trojan attacks are also being increasingly recognized in the semiconductor industry as a serious security concern.

A Trojan is expected to be covert and difficult to detect, i.e. an intelligent adversary will likely insert a Trojan circuit in a way that evades detection during post-manufacturing functional/parametric testing, but manifests itself during long hour of in-field operation. This can be achieved by externally triggering its operation or by making it dependent on rare circuit conditions inside an IC. The condition of Trojan activation as commonly referred to as *trigger condition*, which can be purely combinational or sequential, i.e. related to the clock or a sequence of rare events in the state elements (e.g. flip-flops of registers). The internal circuit nodes affected by a Trojan activation are referred to as *payload* of a Trojan. Fig. 1 shows some example Trojan circuits including a combinational and a sequential Trojan. For example, a Trojan circuit could be triggered only when a data bus attains a unique rare value or when the number of times it attains the rare value equals to a particular count. The malicious effects of Trojan payloads can range from passive, such as leakage of secret information to altering the original functionality of the chip in a critical or destructive fashion.

Protection against hardware Trojan attacks can be accomplished in two broad ways: (1) design-for-security (DfS) techniques that make Trojan insertion difficult or make a



(a) Combinational Trojan



(b) Sequential Trojan

Figure 1: Example of a combinational and a sequential Trojan with triggers from two rare internal nodes A and B.

Trojan easily detectable through post-silicon testing; and (2) manufacturing test approaches that aim at detecting an arbitrary Trojan by observing its effect into a circuit's operational behavior. The first class of techniques, primarily relies on different types of hardening approaches - e.g. insertion of dummy cells into empty spaces in a circuit layout; or key-based obfuscation of a design that make malicious alteration by an adversary provably hard. DfS techniques, however, come at the cost of additional design, verification, and test time, as well as additional design overhead. For example, key-based obfuscation, even though is capable of providing high level of robustness against Trojan attacks, come at a cost of 10% or more area overhead [3]. More importantly, design solutions, however, only work for new designs and not legacy designs, and hence has limited applicability. Hence, efficient test/validation approaches that can provide high level of confidence regarding IC trustworthiness in presence of Trojan threat provides an attractive solution to the IC manufacturers.

Existing test solutions for hardware Trojan detection can be broadly classified into: 1) logic testing and 2) side-channel analysis approaches. In logic testing approach, directed structural or functional tests are generated to activate rare events in a circuit and propagate the malicious effect of a Trojan in logic values to primary outputs. Such approaches are known to be more effective in detecting ultra-small Trojans (typically a few gates in size) reliably under large process variations. The main challenge with logic testing approaches, however, is the difficulty to trigger a Trojan and observe its effect, particularly the complex sequential Trojans, and the inordinately large number of possible Trojan instances an adversary can exploit. On the other hand, side-channel analysis approaches depend on measurement and analysis of physical "side-channel" parameters like power signature or path delay of an IC in order to identify a structural change in the design. Such approaches have the advantage that they do not require triggering a malicious change and observing its impact at the primary output. Side-channel analysis (SCA), primarily based on supply current, has been extensively investigated by large number of research groups and various solutions to increase the signal-to-noise (SNR) have been proposed. A disadvantage of SCA arises from the large process variations (e.g. 20X leakage power and 30% delay variations in 180nm technology [4]) which can potentially mask the minute effect of a Trojan in the measured side-channel parameter.

Even though SCA has shown tremendous promise in detecting unknown Trojans of various types during manufacturing test of an IC, a major issue with SCA is the low detection sensitivity. For a billion transistor modern IC, a Trojan with just a few logic gates would have a minuscule side-channel foot-print, which will require a measurement resolution and dynamic range in an instrument, that is hard to achieve. For example, a delta shift in several nano or pico amp of transient current in ten's of amp of background current, would be practically infeasible to detect even with most precise and expensive instrument. The problem is aggravated by orders of magnitude due to presence of both systematic and random variations in device parameters due to intrinsic process variations.

A solution to the sensitivity problem can be achieved by judicious test generation approach that aims at maximizing the sensitivity for an arbitrary Trojan in unknown circuit location. Through the remainder of the paper, we focus on transient current or power as our side-channel parameter of interest. Some of the concepts however can be applied to other side-channel parameters. To maximize sensitivity of a given Trojan, one needs to amplify activity inside the Trojan circuit and simultaneously minimize the background activity (i.e. activity in the original circuit). However, since the number of possible Trojan instances in a design can be inordinately large, a deterministic test generation method similar to conventional stuck-at fault test generation, cannot work. To address this issue, in this paper, we present a novel test generation framework that can maximize the detection sensitivity for an arbitrary Trojan.

1.1 Main Idea and Our Contributions

The goal of our work is to generate efficient test vectors for Trojan detection using side-channel analysis. Functional test can detect Trojan effect only when it is fully triggered and its payload is propagated to the primary outputs, which makes functional test infeasible to detect Trojans in most cases. Side channel analysis can detect well-hidden Trojans by inspecting the side channel signals, for example, transient current in the circuit. If the switching effect introduced by the Trojan circuit is distinguishable, in the presence of process variation, the Trojan will get identified. In this paper we propose a comprehensive test generation framework to assist side channel analysis for hardware Trojan detection.

We use the relative switching of the Trojan with respect to the whole circuit to indicate the sensitivity of the side channel signals. The statistical test patterns can maximize relative Trojan detection sensitivity under any process noise. Process variation is not expected to affect our side channel sensitivity computation since we consider switching activity instead of actual current or power values. The assumptions we have made are similar to the state-of-the-art side-channel analysis based Trojan detection approaches. The proposed method can be combined with any existing process calibration approaches (such as one in [21] or [22]) to minimize the false positives/negatives and maximize Trojan coverage.

To make side channel analysis successful in detecting Tro-

jans, we need to: (1) maximize the switching activity in the Trojan circuit; (2) minimize the switching activity in other parts of the circuit so that the relative switching effect is maximized. The main idea of this paper is to generate high quality test patterns which can achieve these two goals and increase the sensitivity of side channel analysis. The following are the major contributions of this paper:

1. It presents, for the first time in our knowledge, a statistical test generation approach for increasing side-channel analysis based Trojan detection sensitivity. The proposed approach can be applicable to any transient current based Trojan detection approach.

2. The methodology, referred to as MERS (Multiple Excitation of Rare Switching) for statistical test generation, is shown to derive a compact testset that can trigger each of the rare nodes to satisfy *rare switching* for multiple times. MERS can have a good coverage of all rare nodes and greatly increase the switching effect inside arbitrary Trojans in unknown locations of the circuit.

3. Two reordering methods are proposed to reduce the total switching of the circuit and thus further increase the sensitivity of side channel analysis. First, a simple and low-cost method based on Hamming distance of input vector pairs is introduced to reorder the tests. Next, we develop another simulation based method to more effectively balance switching in rare nodes and the total switching.

Our side-channel based approach is targeted towards detecting unknown Trojans, which means it will remain equally effective even if the adversary is aware of the proposed method. This is due to the following two reasons: (1) the proposed test generation method is statistical in nature - so, unlike conventional deterministic test approaches, it maximizes the activation probability for arbitrary Trojans designed with any trigger condition; and (2) it maximizes the detection sensitivity of unknown Trojans, however "stealthy", by amplifying its effect in side-channel signature. Our simulation platform inserts large number of arbitrary Trojans in a design and shows that the proposed approach is highly effective in detecting them.

The rest of the paper is organized as follows. Section 2 provides overview of hardware Trojan attacks and the broad classes of Trojan detection approaches. Section 3 presents related work in side channel analysis and functional test generation for Trojan detection. Section 4 presents the MERS test generation algorithm and the test reordering algorithms to improve sensitivity of side channel analysis. Section 5 describes the experiment setup and presents results on a set of ISCAS benchmarks with detailed analysis. Section 6 concludes the paper.

2. BACKGROUND AND PRELIMINARY

In this section we briefly describe the growing threat of hardware Trojan attacks and discuss two broad classes of Trojan detection approaches.

2.1 Hardware Trojan Attacks

Malicious modification of IC at different stages of its life cycle, known as hardware Trojan attacks, is an impending threat in the electronics industry. Increased reliance on third party hardware intellectual property (IP) blocks and design automation tools in the IC design flow as well as outsourcing of design/fabrication steps to external parties due to economic reasons are rapidly increasing the vulnerability to Trojan attacks. An adversary can mount such an attack with an objective to cause in-field operational failure or to leak secret information from inside a chip - e.g. the key in a cryptographic IC. Recent investigations have shown that an intelligent adversary can insert tiny Trojans of numerous forms and sizes into a million-transistor design, which can easily evade conventional manufacturing test that is not designed to isolate the stealthy Trojan attacks.

Depending on their mode of operation and structure, hardware Trojans can be grouped into several broad classes. A common classification of Trojans [1][7] is based on the activation mechanism (referred as *Trojan trigger*) and the effect on the circuit functionality (referred as *Trojan payload*). Trojans can be both combinationally and sequentially triggered. Typically, an adversary would choose an extremely rare activation condition so that it is highly unlikely for the Trojan to trigger during conventional manufacturing test. *Sequentially triggered* Trojans (the so-called "time bombs"), on the other hand, are activated by the occurrence of a sequence, or after a period of continuous operation. The simplest sequential Trojans are synchronous stand-alone counters, which trigger a malfunction on reaching a particular count. The trigger mechanism can also be *analog* in nature, whereby on-chip sensors are used to trigger a malfunction. For example, the Trojan gets activated when the temperature of a particular region of the IC exceeds a threshold [1]. Trojans can also be classified based on their *payload* mechanisms into two main classes - *digital* and *analog*. *Digital payload* Trojans can either affect the logic values at chosen internal payload nodes, or can modify the contents of memory locations. *Analog payload* Trojans, on the other hand, affect circuit parameters such as performance, power and noise margin.

2.2 Trojan Detection Approaches

Detecting hardware Trojan instances in an IC before it is used in an electronic system is of paramount importance. Even though DfS approaches that aim at hardening a design with respect to Trojan insertion or facilitating Trojan detection during manufacturing test are being actively researched [3], they have several major limitations: (1) they cannot provide provably robust defense against all forms of Trojan attacks; (2) they often incur unacceptable design overhead; and (3) they cannot be applied to legacy designs, which is difficult to change for incorporating DfS features. Hence, a Trojan detection step for trust validation during post-silicon manufacturing test is becoming crucial to isolate ICs affected with Trojans.

It is worth noting that conventional post-manufacturing test using functional / structural test patterns performs poorly to reliably detect hardware Trojans. This is because manufacturing test generation and application aim at detecting manufacturing defects with well-characterized behavior and model that cause deviation from functional or parametric specifications. They do not aim at detecting additional functionalities incorporated by a Trojan or deviation in circuit behavior triggered by rare events. Hence, conventional testing methods typically provide poor Trojan detection capability, as observed by researchers [5]. Destructive test-

ing of a chip by de-packaging, de-metallization and micro-photography based reverse-engineering is highly expensive (in time and cost) and not a feasible solution because an attacker may selectively insert Trojan into a small subset of the manufactured ICs [8].

Existing Trojan detection approaches fall into two major classes: (a)*functional testing* based, and (b) *side-channel analysis* based. Most Trojan detection techniques proposed in the literature are characterized by their efficiency in detecting particular classes of Trojan. These approaches typically fail to provide high confidence in detecting an inserted Trojan of arbitrary operating mode. The enormous variety of Trojans and the inordinately large Trojan population that might be present in a circuit makes it difficult to devise deterministic test patterns for them. The *functional testing* based Trojan detection approaches [5] aim to trigger rare events at internal nodes in the circuit to activate Trojans and then compare the obtained output logic values of the circuit with the expected golden values of the IC. On the other hand, the *side-channel analysis* based Trojan detection approaches [9][12][19] are based on observing the effect of an inserted Trojan on a physical parameter such as circuit transient current, power consumption or path delay, and then comparing it with the pre-characterized golden value for a Trojan-free IC (or a model of the IC). If the observed value of the measured parameter differs by more than a threshold from the golden value, the presence of a Trojan is suspected. Both classes of Trojan detection techniques have their relative pros and cons. The main challenge for functional testing based Trojan detection approaches is the enormously large *Trojan design space*, which makes complete enumeration and test generation for all feasible Trojan instances in a moderately-sized circuit computationally infeasible. This makes it extremely difficult to guarantee that an arbitrary inserted Trojan would be activated, cause circuit malfunction and thus get detected during the test application phase. On the other hand, the advantage of the side-channel analysis based approaches lies in the fact that even if the Trojan circuit does not cause observable malfunction in the circuit during test, the presence of the extra circuitry can be reflected in the measured side-channel parameter. Further, such techniques are suitable for arbitrarily complex Trojans because they do not need to make any assumption about the mode of operation of an inserted Trojan. However, the main challenges associated with side-channel analysis are large *process variation* and *design marginality* induced effects in modern nanometer technologies [1], and measurement noise, which can mask the effect of an inserted Trojan circuit, especially for small Trojans.

3. RELATED WORK

The underlying assumption for Trojan insertion is that an adversary is fully aware of the design functionality and therefore can hide the Trojan in a hard-to-find place. The adversary may use very rare internal transitions to trigger the Trojan, and it may be impossible to detect (due to exponential state space) during traditional testing and validation. One way to address this issue is to obfuscate [3] or encrypt [14] the design such that the adversary cannot figure out the actual functionality and therefore cannot insert the Trojan in a covert manner. Unfortunately, smart attacker can effectively bypass both obfuscation [15] and encryption [16] methods. A promising direction is to develop efficient techniques for hardware Trojan detection. Prior research on Trojan detection can be classified into two broad categories: side-channel analysis and functional test generation. A vast majority of the Trojan detection approaches are based on analysis of side-channel signatures such as delay, transient and leakage power [8][9][10][11][12][13]. The basic idea is to find a side-channel signature (if the Trojan activated) that is different from the normal signature. Unfortunately, these approaches are susceptible to thermal and process variations. Therefore, it would be difficult to detect small combinational Trojans.

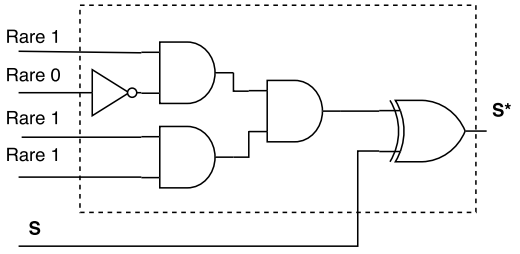
One promising direction to overcome process variation is to generate functional test patterns that are likely to activate the Trojans. These approaches rely on the fact that an adversary will choose a trigger condition for the Trojan using a set of rare nodes. Various approaches tried to maximize the rare node activation to increase the likelihood of activating Trojans. Some approaches [18][19] use the design-for-test (DFT) infrastructure (such as additional scan flip-flop) to increase the transition probability of low-transition nets. MERO [5] takes the advantage of N-detect test [20] to maximize the trigger coverage by activating the rare nodes. The test generation ensures that each of the nodes gets activated to their rare values for at least N times. They have shown that if N is sufficiently large, a Trojan with trigger conditions from these rare nodes, will be highly likely to be activated by the generated test set. Saha et al. [6] improves the test pattern generation of MERO by using genetic algorithm and boolean satisfiability, which could more effectively propagate the payload of possible Trojan candidates. However, these functional test generation approaches are not designed for side-channel analysis. Direct application of these test generation approaches for side-channel analysis would not be best for improving side-channel sensitivity for Trojan detection. The objective of increasing side-channel sensitivity is very different from the ones in both MERO as well as its enhanced version by Saha et al. Unlike these existing approaches, a side-channel aware test generation approach, as proposed in our paper, requires maximizing switching activity in an unknown Trojan circuit while minimizing the background switching.

Instead of aiming on finding a vector to activate a set of rare nodes, we focus on creating a set of vector pairs to maximize switching in rare nodes. Our algorithm creates multiple excitation of rare switching which is important in making side-channel based Trojan detection effective. Moreover, we also try to simultaneously minimize the background switching to maximize the relative switching.

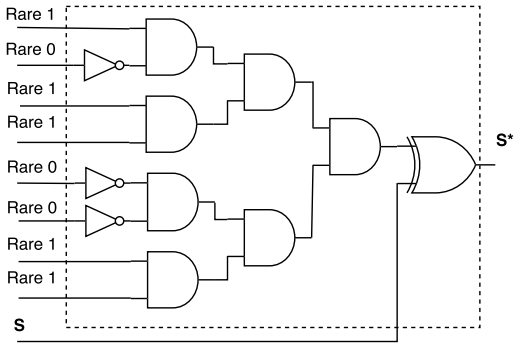
4. MERS METHODOLOGY

In this section, we present the proposed methodology for side-channel aware test generation in detail. The methodology is based on the concept of statistically maximizing the switching activity in all the rarely triggered circuit nodes.

The effectiveness of a test pattern for side channel analysis is measured in two ways: (1) the ability to create most switching inside a Trojan or to activate a Trojan; (2) the ability to create high Trojan-to-circuit switching. We measure *DeltaSwitch* as the switching introduced by the Trojan, which is the difference of number of switches between the golden circuit and the Trojan-infected circuit. We measure *RelativeSwitch* as the ratio of *DeltaSwitch* to the total number of switches (*TotalSwitch*) in the golden circuit.



(a) A 4-trigger Trojan



(b) An 8-trigger Trojan

Figure 2: Trojans with rare nodes as trigger conditions. The 4-trigger Trojan will only be activated by the rare combination 1011 and the 8-trigger Trojan will only be activated by the rare combination 10110011.

An effective test vector should be capable of creating large *DeltaSwitch*, and more importantly it should create large *RelativeSwitch*, as it is directly related to the sensitivity for side channel analysis.

$$RelativeSwitch = DeltaSwitch/TotalSwitch \quad (1)$$

The major challenges for generating high-quality test vectors are as follows: (1) we are not sure of the location where the Trojan is inserted in the circuit; (2) the Trojan is stealthy and has very low activity when it is not triggered. These characteristics have made random tests not effective in magnifying the side channel signal for Trojan detection. Fig. 2 shows two example Trojan instances. The 4-trigger Trojan will only be activated by the rare combination 1011 and the 8-trigger Trojan will only be activated by the rare combination 10110011. If the possibility of each rare node to take its rare value is 0.1, the probability to have these two Trojans fully triggered is 10^{-4} and 10^{-8}, respectively.

Our test generation approach (MERS) is based on creating a set of test vectors for each candidate rare node individually to have *rare switching* multiple (at least N) times. Our approach utilizes the principle of N-detect [20] tests to increase the likelihood of partially or fully activating a Trojan. MERS can generate a high-quality testset for these rare nodes individually to have rare switching for N times. If N is sufficiently large, a Trojan with triggering conditions from these rare nodes is likely to have high switching activity even though it might not be fully activated.

4.1 Multiple Excitation of Rare Switching

The basic idea of MERS is that if we can make a rare node switch N times where N is sufficiently large, it significantly improves the chance of switching in a Trojan associated with that rare node. The **rare switching** in our algorithm specially refers to a rare node switching from its non-rare value to its rare value. The reason to choose this criteria is twofold: (1) it is more difficult to switch from non-rare to rare value than from rare to non-rare value; (2) it defines the switching between the previous vector and the current vector, and it usually helps to create an extra switching between the current vector and the next vector. This will increase the probability of switching of a Trojan which has rare nodes as its trigger conditions. Our approach is also applicable to sequential Trojans, which requires the rare condition to occur a certain number of times to be fully triggered.

Algorithm 1 shows the steps of MERS to generate high quality tests for creating switching in rare nodes, so as to assist side channel analysis for hardware Trojan detection. The algorithm is fed with the golden circuit netlist, the list of random test patterns (V) and a list of rare nodes (R) (which is obtained by random vector based circuit simulation

Algorithm 1: Multiple Excitation of Rare Switching (MERS)

Input: Circuit netlist, rare switching requirement (N), list of rare nodes ($R = \{r_1, r_2, ..., r_m\}$), list of random patterns ($V = \{v_1, v_2, ..., v_n\}$)
Output: MERS test patterns (T)

```
// simulate and sort random vectors
for each random vector v in V do
    Simulate the circuit with the input vector v
    Count the number of nodes (R_V) in R with their rare values satisfied
end
Sort vectors in V in descending order of R_V
for each node r_i in R do
    Set its rare switching counter (S_i) to 0
end
// mutate vector to find improved vector pairs
Initialize previous vector t_p as a vector of all 0's
for each vector v_j in V do
    Simulate the circuit with vector pair (t_p, v_j)
    Count the number of rare switches (R_S)
    Set v'_j = v_j
    for each bit in v'_j do
        Mutate the bit and re-simulate the circuit with vector pair (t_p, v'_j)
        Count the number of rare switches (R'_S)
        if R'_S > R_S then
            Accept the mutation to v'_j
        end
    end
    Update S_i for all nodes in R due to vector v'_j
    if v'_j increases S_i for at least one rare node then
        Add the mutated vector v'_j to T
        Set t_p = v'_j
    end
    if S_i ≥ N for all nodes in R then
        Break
    end
end
return MERS test patterns T
```

beforehand). First, we simulate each random pattern and count the number of rare nodes (R_V) that take their rare values. We sort the random patterns in descending order of R_V, which means that the vector with ability to activate the most number of rare nodes goes first. Next, we initialize the rare switching counter S_i for each rare node to 0. In the next step, we mutate vectors from the random pattern set to generate high quality tests. We mutate the current vector one bit at a time and we accept the mutated bit only if the mutated vector can increase the number of nodes to have rare switching. In this step, only those rare nodes with $R_S < N$ are considered. The mutation process repeats until each rare node has achieved at least N rare switches. The output of the test generation process is a compact set that improves the switching capability in rare nodes, compared to random patterns. The complexity of the algorithm is $O(n*m)$, where n is the total number of test vectors mutated during the process, and m is the number of bits in primary inputs. The runtime to generate MERS tests can be found in Table 1.

The testset generated by MERS is expected to be very effective in increasing the likelihood of rare nodes to switch and thus increasing the activities in Trojans. In other words, MERS testset is capable of maximizing the DeltaSwitch (the numerator in Equation 1). MERS testset is already a very high quality testset in terms of criteria for DeltaSwitch. However, MERS testset also creates more switching in other parts of the circuit, when it is making efforts to switch rare nodes. This characteristic of increased TotalSwitch would be further illustrated in the Section 5. In order to maximize relative switching, we need to have TotalSwitch in control as well. In the following subsections, we propose two methods to tune the MERS testset, so that it can: (1) still be effective for DeltaSwitch, (2) reduce TotalSwitch and improve the effectiveness for RelativeSwitch. The first method is a heuristic approach based on hamming distance of test vectors, which can reduce the total switching. The second one is simulation based, in which we try to balance the rare switching and the total switching while we explore all the candidate vectors.

4.2 Hamming Distance based Reordering

If two consecutive input vectors have the same values in most bits, it is very possible that the internal nodes will also have a lot of values in common. A simple heuristic to reduce total switching in circuit is to have similar input vectors. We use the Hamming distance between two vectors to represent the similarity. Algorithm 2 shows our approach to reorder the testset by Hamming distance. The algorithm is a greedy approach to explore all candidate vectors and take the best one in terms of Hamming distance. We first check the Hamming distances between the previous vector and all the remaining vectors, then we select the vector which has the minimum Hamming distance as the next vector. The time complexity of Algorithm 2 is $O(n^2)$, where n is the testset size. Fortunately, it is of low cost to calculate the Hamming distance between two input vectors. The actual run-time is very short because n (number of test patterns produced by MERS) is small, in the order of tens of thousands.

Algorithm 2: Tests Reordering by Hamming Distance (MERS-h)

Input: List of Test Patterns ($T_{orig} = \{t_1, t_2, ..., t_n\}$) produced by Algorithm 1
Output: Improved Test Patterns (T_{hamm})

```
1:  Initialize T_hamm = {}
2:  Initialize previous test t_p as a vector of all 0's
3:  while T_orig is not empty do
4:      min_dist = int_max
5:      best_idx = −1
6:      for all remaining tests t_j in T_orig do
7:          if min_dist > hamming_dist(t_p, t_j) then
8:              min_dist = hamming_dist(t_p, t_j)
9:              best_idx = j
10:         end
11:     end
12:     Add t_best_idx to the end of T_hamm
13:     Remove t_best_idx from T_orig
14:     Update t_p = t_best_idx
15: end
16: return T_hamm
```

Algorithm 3: Tests Reordering by Simulation (MERS-s)

Input: List of Test Patterns ($T_{orig} = \{t_1, t_2, ..., t_n\}$) produced by Algorithm 1
Output: Improved Test Patterns (T_{sim})

```
1:  Initialize T_sim = {}
2:  Initialize previous test t_p as a vector of all 0's
3:  while T_orig is not empty do
4:      max_p = int_min
5:      best_idx = −1
6:      for all remaining tests t_j in T_orig do
7:          Simulate the circuit with vector pair (t_p, t_j)
8:          Count the number of RareSwitch and
            TotalSwitch
9:          profit = C * RareSwitch − TotalSwitch
10:         if max_p < profit then
11:             max_p = profit
12:             best_idx = j
13:         end
14:     end
15:     Add t_best_idx to the end of T_sim
16:     Remove t_best_idx from T_orig
17:     Update t_p = t_best_idx
18: end
19: return T_sim
```

4.3 Simulation based Reordering

The reordering problem to improve the relative switching is actually a multi-objective optimization problem: maximize the *DeltaSwitch* and minimize the *TotalSwitch* as in Equation 1. We do not know the *DeltaSwitch*, because the location and type of the Trojan is unknown. However, rare switching between two vectors is a good indicator for *DeltaSwitch*, which means a large number of rare switching would imply a large *DeltaSwitch* in Trojan. We redefine the optimization goal as to maximize the rare switching and

minimize the total switching at the same time between vector pairs. We formalize the problem as shown in Equation 2. We need to explore the best weights to balance between the two objectives:

$$maximize \quad (w_1 * RareSwitch - w_2 * TotalSwitch) \quad (2)$$

We propose an approach as shown in Algorithm 3 based on real simulation of the test vectors to maximize the combined objective. We introduce a concept of $profit$ to indicate the fitness of a test vector to follow the previous test vector. $profit$ is defined as $(C*RareSwitch-TotalSwitch)$, where C is the ratio of two weights w_1 and w_2. It is meant to maximize the rare switching (activity in Trojan circuits) and minimize the total switching of the whole circuit. In the experiment section, we will explore different weight ratios and check the influence of weight ratios on side channel sensitivity.

Algorithm 3 shows our approach to tune the testset by simulation with $profit$ as a reordering criterion. By exhaustively checking the $profit$ between the previous vector and all the remaining vectors, we select the vector which has the maximum $profit$ as the next following vector. The time complexity of Algorithm 3 is $O(n^2)$, where n is the test length. However, it is much slower than Algorithm 2, because it is time-consuming to simulate input vector pairs and calculate $profit$.

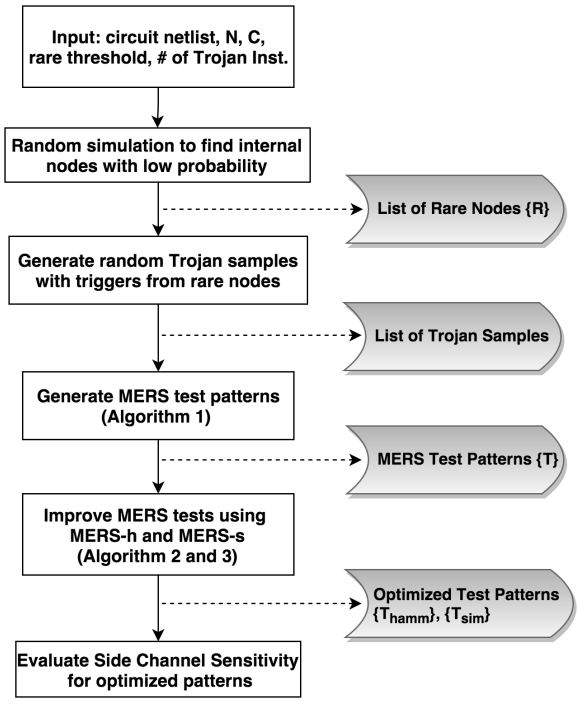


Figure 3: Test generation framework for side-channel analysis based Trojan detection.

5. EXPERIMENTS

5.1 Experimental setup

The test generation framework, including the MERS core algorithms and the evaluation framework, is implemented using C. As shown in Fig. 3, the test generation framework can simulate circuit netlists, generate MERS testset, further tune the testset, and evaluate the effectiveness of testsets on random Trojans. We evaluated our approach on a subset of ISCAS-85 and ISCAS-89 benchmark circuits. The sequential circuits are converted into full scan mode. We also implemented the MERO [5] approach with parameter N of 1000 for comparison. We did our experiments on a server with AMD Opteron Processor 6378 (2.4GHz). The runtime for different benchmarks and different methods is shown in Table 1. The table also shows the number of rare nodes in each benchmark. We used 0.1 as the rare threshold to select rare nodes.

Table 1: Runtime for MERS test generation, reordering using hamming and reordering using simulation, with N=1000, rare threshold = 0.1

Benchmark	Nodes (rare / total)	Run-time (s)		
		MERS	MERS-h reordering	MERS-s reordering
c2670	63 / 1010	13370.86	7.24	4925.23
c3540	331 / 1184	6097.51	9.43	18166.94
c5315	255 / 2485	45595.97	11.04	39073.81
c6288	45 / 2448	4154.62	0.31	2802.85
c7552	306 / 3720	81405.89	25.2	63502.19
s13207	592 / 2504	12511.95	365.02	29064.72
s15850	679 / 3004	19903.44	728.14	38181.49
s35932	896 / 6500	7295.74	39.53	31201.04

5.2 Evaluation Criteria

When applying a testset to a circuit with Trojan, there are four criteria to evaluate the effectiveness of the testset:

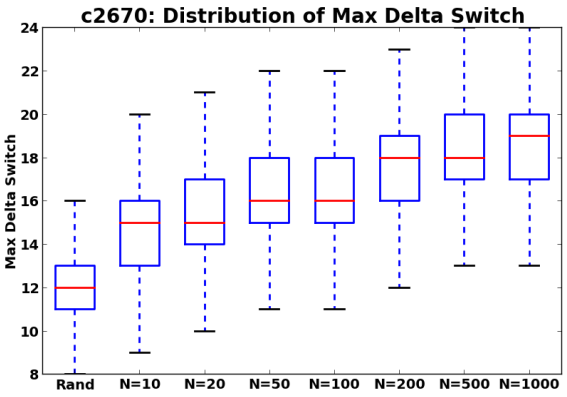
- **AvgDeltaSwitch**: the average delta switch when applying the testset on this Trojan-infected circuit.
- **MaxDeltaSwitch**: the maximum delta switch when applying the testset.
- **AvgRelativeSwitch**: the average relative switch when applying the testset.
- **MaxRelativeSwitch**: the maximum relative switch when applying the testset. We choose this criterion as the **Side Channel Sensitivity** because this directly determines whether a Trojan can be detected through side-channel analysis.

AvgDeltaSwitch and *MaxDeltaSwitch* reflect the activity in Trojan, and *AvgRelativeSwitch* as *MaxRelativeSwitch* reflect the sensitivity of the side channel signal in detecting the Trojan.

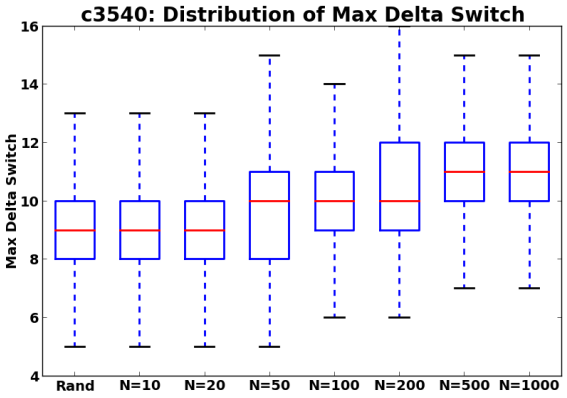
As for evaluation of testsets, we would expect a high-quality testset to have a good coverage over all possible Trojans. In our experiments, we apply the testset to 1000 randomly inserted Trojan samples and compute these four values for each Trojan instance. We would then take the average of these four metrics, which would reflect the capability of the testset to enable detection of different Trojans through side-channel analysis. The average $MaxRelativeSwitch$ would be most suitable for Side Channel Sensitivity evaluation, which is to maximize the sensitivity for an arbitrary Trojan in unknown circuit location.

5.3 Exploration of N

Fig. 4 shows the distribution of MaxDeltaSwitch over 1000 random 8-trigger Trojan samples for two ISCAS-85



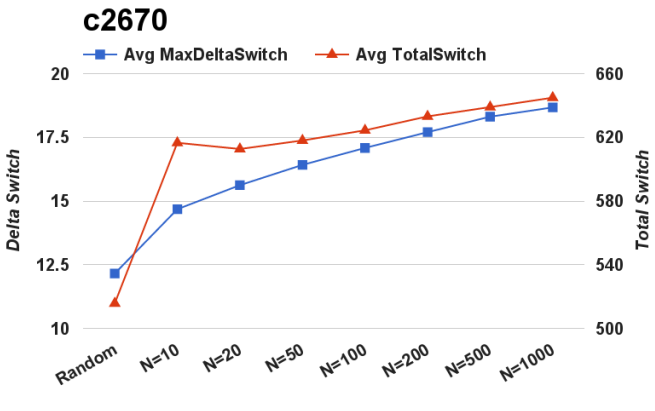
(a) c2670: Distribution of MaxDeltaSwitch over 1000 random samples of 8-trigger Trojans.



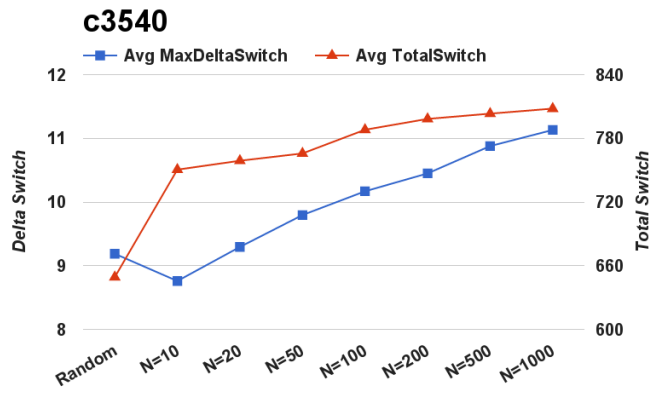
(b) c3540: Distribution of MaxDeltaSwitch over 1000 random samples of 8-trigger Trojans.

Figure 4: Impact of N (number of times that a rare node have rare switching) on MaxDeltaSwitch for benchmarks (a) c2670 and (b) c3540.

benchmarks. We choose different N to generate MERS testsets, to compare with the Random (10K vectors) testset. For each testset, the box plot shows (minimum, first quartile, median, third quartile, maximum) values of MaxDeltaSwitch of the 1000 Trojan samples. It is clear from these plots that the distribution of MaxDeltaSwitch is constantly improving with increasing N. For $c2670$, the average MaxDeltaSwitch (as shown by the red lines) can reach 18.67 for MERS (N = 1000), while Random testset can achieve only 12.15. For $c3540$, the average MaxDeltaSwitch can reach 11.13 for MERS (N = 1000), while for Random testset it is only 9.19. The fact that the quality of MERS tests improves with increasing N is not surprising. It is similar to N-detect tests for stuck-at faults, where fault coverage is expected to improve with increasing N. The testset size also increases with N. The sizes of testsets for MERS (N = 10, 20, 50, 100, 200, 500, 1000) are (71, 140, 347, 656, 1262, 3142, 6199) for $c2670$, and (161, 302, 742, 1441, 2858, 7070, 14250) for $c3540$. In most of our experiments, we choose a value of N = 1000, which is a good balance between testset quality and testset size. For fair comparison with Random testset, we will only take the first 10K vectors of MERS testset if it is larger than 10K.



(a)



(b)

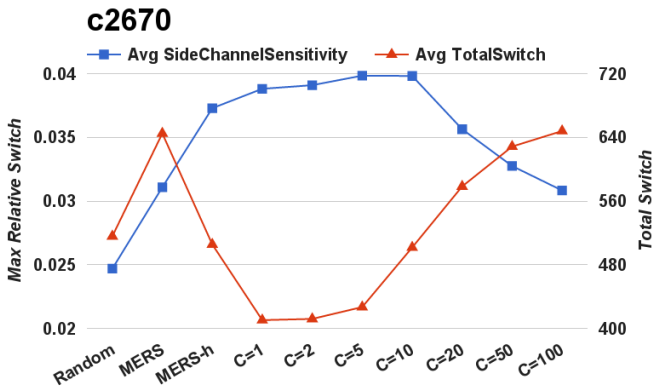
Figure 5: MaxDeltaSwitch versus TotalSwitch for different N for benchmarks (a) c2670 and (b) c3540. MERS creates more switching in Trojan, as well as more switching in other parts of the circuit (which results in increased total switching).

5.4 Effect of Increased Total Switching

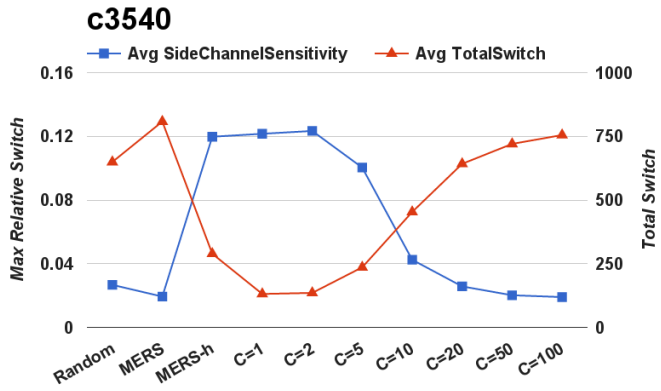
Fig. 5 shows the average $MaxDeltaSwitch$ and the average $TotalSwitch$ of the testsets for 1000 8-trigger Trojan samples for different values of N. For both of the two benchmarks, the average $TotalSwitch$ increases with N as well as the average $MaxDeltaSwitch$. It is obvious that all the MERS testsets have much larger average $TotalSwitch$, compared with the Random testset. For $c2670$, the average $TotalSwitch$ for MERS (N = 1000) is 644.9, which is about 1.25X times of that of the Random testset (515.7). For $c3540$, the average $TotalSwitch$ for MERS (N = 1000) is 808, while Random testset is only 649.2. The insight that we can get from here is that MERS tends to increase the $TotalSwitch$ of the circuit, although it is designed to increase switches in rare nodes. The following subsection will show that the proposed reordering methods would be effective to reduce $TotalSwitch$ and thus increase side channel sensitivity.

5.5 Effect of Weight Ratio (C)

The effectiveness of the two reordering methods can be observed in Fig. 6 and Fig. 7. As shown in Fig. 6, MERS-h can reduce $TotalSwitch$ and thus increase the relative switching

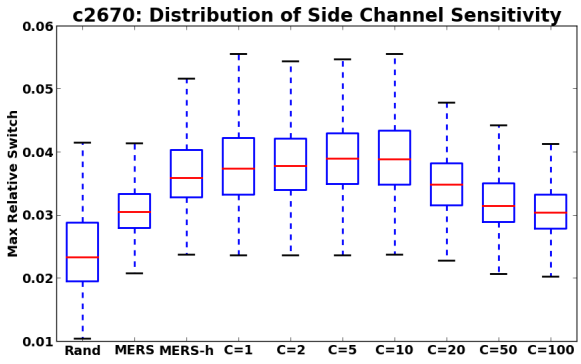


(a)

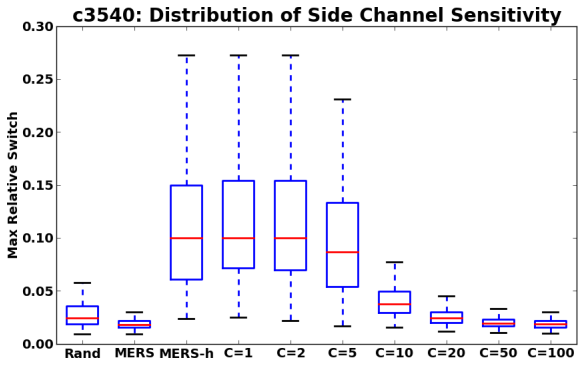


(b)

Figure 6: Side Channel Sensitivity versus $TotalSwitch$ for Random, the original MERS, MERS-h and MERS-s (with different C) for benchmarks (a) c2670 and (b) c3540. Both MERS-h and MERS-s (with a small C) are effective in reducing the total switching.



(a) c2670: Distribution of Side Channel Sensitivity over 1000 random samples of 8-trigger Trojans.



(b) c3540: Distribution of Side Channel Sensitivity over 1000 random samples of 8-trigger Trojans.

Figure 7: Distribution of Side Channel Sensitivity for Random, the original MERS, MERS-h and MERS-s (with different C) for benchmarks (a) c2670 and (b) c3540.

(i.e. the Side Channel Sensitivity), compared with the original MERS testset. For MERS-s with different weight ratio C, side channel sensitivity improves steadily with a small C, and then goes down when C is too large. As the weight ratio tries to balance $DeltaSwitch$ and $TotalSwitch$, a large C will outweigh the influence of $TotalSwitch$, which will make it less different from the original MERS testset. In the following experiments, we choose the weight ratio as $C = 5$, as it provides a good balance between the total switching and rare switching.

Fig. 7 shows detailed distribution of Side Channel Sensitivity for 1000 8-trigger Trojan samples with different choices of C. The reordering methods are working well to improve Side Channel Sensitivity, which is built on the fact that the original MERS testset is already of high quality in terms of $DeltaSwitch$, or switching in Trojans.

5.6 Increase in Trojan Activity

Table 2 shows that MERS (N=1000) is very effective in creating $DeltaSwitch$ caused by arbitrary Trojans due to its statistical nature. The average *Max Delta Switch* increases by 31.11% and the average *Avg Delta Switch* by 187.33% on average for different benchmarks, compared with Random testset. This shows the effectiveness of MERS in creating Trojan activity.

Table 3 shows that MERS is also helpful in improving RelativeSwitch. The average AvgRelativeSwitch increased by 158.16%, compared with Random testsets. For average MaxRelativeSwitch (Side Channel Sensitivity), MERS has an average improvement of 18.89%. However, Side Channel Sensitivity values for benchmark $c3540$ and $c6288$ are not as good as those of Random testsets. This is due to the fact that MERS testset also increases the total switching, when it is making efforts to cause rare nodes switching. This phenomenon is illustrated and explained in Fig. 5 and Fig. 6, and this side effect can be improved by the two reordering algorithms as shown in Table 4 and 5.

5.7 Side Channel Sensitivity Improvement

To this point, we have explored the parameters: N for MERS and C for MERS-s. We choose $N = 1000$ and $C = 5$ in the following experiment to compare our proposed schemes with Random testset and MERO. Table 4 and 5 show the improvement of proposed approaches on Side Channel Sensitivity for 4-trigger and 8-trigger Trojans.

Table 4 shows that MERS, MERS-h and MERS-s have 10.37%, 138.44% and 152.26% improvement over the Ran-

Table 2: Comparison of MERS (N=1000) with Random (10K) for average MaxDeltaSwitch and average AvgDeltaSwitch, over 1000 random samples of 8-trigger Trojans.

Benchmark	**average MaxDeltaSwitch**			**average AvgDeltaSwitch**		
	Random	MERS	Improve.	Random	MERS	Improve.
c2670	12.15	18.67	53.67%	1.4289	6.8561	379.83%
c3540	9.19	11.13	21.16%	1.3716	2.9058	111.85%
c5315	9.51	13.80	45.16%	1.3116	3.9300	199.64%
c6288	6.63	7.26	9.63%	1.0636	4.8448	355.50%
c7552	8.53	12.00	40.76%	1.3488	2.7700	105.36%
s13207	6.63	8.83	33.18%	0.6428	0.9771	52.01%
s15850	7.53	10.84	43.99%	0.7465	1.3609	82.29%
s35932	15.16	15.37	1.35%	2.1803	6.8060	212.16%
Avg. Improve.	–	–	31.11%	–	–	187.33%

Table 3: Comparison of MERS (N=1000) with Random (10K) for average $MaxRelativeSwitch$ (Side Channel Sensitivity) and average $AvgRelativeSwitch$, over 1000 random samples of 8-trigger Trojans.

Benchmark	**average MaxRelativeSwitch (Side Channel Sensitivity)**			**average AvgRelativeSwitch**		
	Random	MERS	Improvement	Random	MERS	Improvement
c2670	0.02469	0.03108	25.90%	0.00255	0.01054	314.14%
c3540	0.02670	0.01933	-27.59%	0.00214	0.00361	69.12%
c5315	0.00526	0.00766	45.72%	0.00075	0.00200	165.65%
c6288	0.00534	0.00395	-26.06%	0.00059	0.00219	270.68%
c7552	0.00452	0.00852	88.48%	0.00058	0.00113	94.65%
s13207	0.00756	0.00844	11.64%	0.00066	0.00085	28.22%
s15850	0.00593	0.00716	20.70%	0.00053	0.00082	54.25%
s35932	0.00523	0.00587	12.29%	0.00060	0.00223	268.54%
Avg. Improve.	–	–	18.89%	–	–	158.16%

dom testsets, respectively. While the original MERS testsets is 23.95% worse than MERO testsets, MERS-h and MERS-s have 52.62% and 62.01% improvement over MERO. Table 5 shows the results for 8-trigger Trojans. Compared to Random testsets, MERS, MERS-h and MERS-s can have 18.89%, 107.53% and 96.61% improvement, respectively. The original MERS testsets is 12.43% worse than MERO testsets. MERS-h and MERS-s testsets can improve the Side Channel Sensitivity by 40.79% and 38.50%, respectively.

In this section, we explore the impact of different values of N for MERS and observe the effectiveness of MERS to maximize Trojan activity as N increases. We confirm the superiority of MERS testsets over Random testsets in Section 5.6 on creating switching activity in randomly sampled Trojans. We observed that the total switching was also likely to increase while MERS made efforts to maximize rare switching in Trojans. The two reordering methods (MERS-h and MERS-s) successfully had the total switching under control while maintaining the rare switching high. The comparison with Random and MERO testsets shows the effectiveness of our test generation framework in maximizing Side Channel Sensitivity for Trojan detection.

5.8 Process Calibration and Multiple-Parameter Side-Channel Analysis

MERS can be combined with existing process calibration approaches [21][22][23] to minimize the false positives/negatives and maximize Trojan coverage. Most side-channel analysis based approaches perform process variation calibration by using golden chips at different process corners. This helps us obtain the limiting threshold values, beyond which any chip is classified as Trojan-infected. MERS can simultaneously maximize the switching in Trojan and minimize the background switching, so as to maximize the relative switching. By calibration or reference to that of a golden chip, MERS helps side channel analysis to reduce the intra-die systematic process variations. Moreover, as shown in [23], various measurable parameters can be used for multiple-parameter side-channel-based Trojan detection where at least one parameter is affected by the Trojan and other parameters are used to calibrate the process noise. For example, the dynamic current (I_{DDT}), the quiescent or leakage current (I_{DDQ}) and the maximum operating frequency (F_{max}) may be influenced when there is a Trojan. They can serve as side channel references to calibrate process noise. Authors in [23] have shown Trojan and process variation effects on these three variables (I_{DDT}, I_{DDQ} and F_{max}). MERS can increase I_{DDT}, which would greatly improve the accuracy of [23] to isolate a Trojan-infected chip in the multiple-parameter space from process induced variations.

5.9 Scalability to Large Designs

For a large design, the supply current of a golden chip for a high-activity vector can be very large compared to the additional current consumed by a small Trojan. The variation in the current value due to process noise can also be very large, which would mask the effect of the Trojan on the measured current and create difficulty for accurate Trojan detection. Scalability of MERS to larger designs can be enhanced by combining it with region-based test generation

Table 4: Comparison of average **Side Channel Sensitivity** between Random (10K), MERO, and MERS testsets, N=1000, C=5 for MERS-s, over 1000 random samples of 4-trigger Trojans.

Benchmark	Comparison Testsets		Proposed Schemes			Improvement to Random			Improvement to MERO		
	Random	MERO	MERS	MERS-h	MERS-s	MERS	MERS-h	MERS-s	MERS	MERS-h	MERS-s
c2670	0.01703	0.02571	0.02231	0.03035	0.03308	31.01%	78.27%	94.31%	-13.23%	18.07%	28.69%
c3540	0.02144	0.04238	0.01336	0.10677	0.11067	-37.71%	397.97%	416.16%	-68.48%	151.96%	161.16%
c5315	0.00445	0.01082	0.00747	0.01287	0.01586	67.79%	188.97%	256.29%	-30.97%	18.89%	46.59%
c6288	0.00480	0.00395	0.00313	0.00741	0.00896	-34.81%	54.47%	86.85%	-20.88%	87.50%	126.80%
c7552	0.00351	0.00737	0.00491	0.01250	0.01168	39.61%	255.63%	232.38%	-33.46%	69.50%	58.42%
s13207	0.00568	0.00617	0.00619	0.00773	0.00826	9.07%	36.24%	45.49%	0.31%	25.29%	33.80%
s15850	0.00447	0.00487	0.00474	0.00691	0.00634	6.14%	54.83%	42.06%	-2.75%	41.86%	30.17%
s35932	0.00354	0.00463	0.00361	0.00500	0.00512	1.89%	41.17%	44.53%	-22.12%	7.90%	10.48%
Avg. Improve.	–	–	–	–	–	10.37%	138.44%	152.26%	-23.95%	52.62%	62.01%

Table 5: Comparison of average **Side Channel Sensitivity** between Random (10K), MERO, and MERS testsets, N=1000, C=5 for MERS-s, over 1000 random samples of 8-trigger Trojans.

Benchmark	Comparison testsets		Proposed Schemes			Improvement to Random			Improvement to MERO		
	Random	MERO	MERS	MERS-h	MERS-s	MERS	MERS-h	MERS-s	MERS	MERS-h	MERS-s
c2670	0.02469	0.03204	0.03108	0.03729	0.03984	25.90%	51.05%	61.40%	-3.01%	16.37%	24.35%
c3540	0.02670	0.05532	0.01933	0.11974	0.10037	-27.59%	348.53%	275.96%	-65.05%	116.47%	81.44%
c5315	0.00526	0.00875	0.00766	0.01020	0.01129	45.72%	94.03%	114.78%	-12.38%	16.66%	29.14%
c6288	0.00534	0.00412	0.00395	0.00649	0.00790	-26.06%	21.55%	47.97%	-4.20%	57.49%	91.72%
c7552	0.00452	0.00914	0.00852	0.01437	0.01149	88.48%	217.78%	154.00%	-6.70%	57.31%	25.74%
s13207	0.00756	0.00838	0.00844	0.01053	0.01112	11.64%	39.24%	47.05%	0.69%	25.58%	32.63%
s15850	0.00593	0.00722	0.00716	0.00923	0.00818	20.70%	55.69%	37.94%	-0.87%	27.86%	13.28%
s35932	0.00523	0.00638	0.00587	0.00692	0.00700	12.29%	32.39%	33.80%	-7.90%	8.58%	9.74%
Avg. Improve.	–	–	–	–	–	18.89%	107.53%	96.61%	-12.43%	40.79%	38.50%

approaches, which segment a circuit into nearly-isolated regions (i.e. with low connectivity between them). In this case, MERS can be applied separately to each region. For example, in case of a processor, MERS can be employed separately to its building blocks, such as, integer execution unit, floating point datapaths, control logic, and result bus logic. MERS can work with schemes proposed in [23] to isolate a region and prevent unwanted switching in independent functional modules by taking advantage of the power gating techniques conventionally used by low-power designs, such as clock gating, supply gating, or operand isolation. MERS can also be applied a more flexible region-based side channel analysis approach proposed in [24]. They perform a functional decomposition to divide a large design into several small blocks or regions, so that they can activate them one region at a time. MERS can be used as the test generation algorithm to generate vectors that maximize the activity within each region. The decision to report a chip as Trojan-infected would be based on the deviation of its region current matrix with respect to the golden chip. Future work will include integration of MERS with region-based circuit partitioning techniques to further enhance its effectiveness and its evaluation on larger industry-standard designs.

6. CONCLUSIONS

We have presented a framework for statistical test generation, called MERS, which can significantly improve the Trojan detection sensitivity in side-channel analysis based Trojan detection. The approach aims at statistically increasing switching activity in an unknown Trojan to amplify the Trojan effect in presence of large process variations. Such a test generation approach will, in general, be effective for any side-channel analysis approaches that rely on activity in Trojan circuits (e.g. transient current, dynamic power profile, or electromagnetic emanation based methods). Furthermore, MERS is effective for any Trojan forms/sizes, as long as a Trojan is implanted through alterations in a circuit structure - the most dominant mode of Trojan implantation. Our simulation results on a set of benchmark circuits show that the proposed approach can improve the side channel sensitivity by more than 96.61%, compared with random tests for a large set of arbitrary Trojans. It shows that a judicious statistical test generation such as MERS can serve as an essential component in a side-channel Trojan detection approach. Future work will include further improvement in scalability to larger designs and evaluation of MERS with test chip measurements.

7. ACKNOWLEDGMENTS

This work was partially supported by grants from National Science Foundation (1441667, 1603475, 1603483), Semiconductor Research Corporation (2014-TS-2554) and Cisco Systems (F020375). Any opinions, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the funding agencies.

8. REFERENCES

[1] R. Chakraborty, S. Narasimhan and S. Bhunia. Hardware Trojan: Threats and emerging solutions. IEEE International High-Level Design Validation and Test Workshop (HLDVT), 2009.

[2] DARPA: TRUST in Integrated Circuits (TIC), 2007. [Online]. Available: http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA503809

[3] R. Chakraborty and S. Bhunia. Security against hardware Trojan through a novel application of design obfuscation. ACM International Conference on Computer-Aided Design (ICCAD), pp. 113-116, 2009.

[4] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi and V. De. Parameter variations and impact on circuits and microarchitecture. ACM/IEEE Design Automation Conference (DAC), pp. 338-342, 2003.

[5] R. Chakraborty, F. Wolff, S. Paul, C. Papachristou and S. Bhunia. MERO: A Statistical Approach for Hardware Trojan Detection. International Workshop on Cryptographic Hardware and Embedded Systems (CHES), pp. 396-410, 2009.

[6] S. Saha, R. Chakraborty, S. Nuthakki, Anshul, and D. Mukhopadhyay. Improved Test Pattern Generation for Hardware Trojan Detection Using Genetic Algorithm and Boolean Satisfiability. International Workshop on Cryptographic Hardware and Embedded Systems (CHES), pp. 577-596 (2015).

[7] Y. Jin and Y. Makris. Hardware Trojan detection using path delay fingerprint. IEEE International Symposium on Hardware Oriented Security and Trust (HOST), 2008.

[8] M. Banga and M. Hsiao. A region based approach for the identification of hardware Trojans. IEEE International Workshop on Hardware-Oriented Security and Trust (HOST), 2008.

[9] M. Banga, M. Chandrasekar, L. Fang and M. Hsiao. Guided test generation for isolation and detection of embedded Trojans in ICs. ACM Great Lakes Symposium on VLSI (GLSVLSI), pp. 363-366, 2008.

[10] Y. Jin and Y. Makris. Hardware Trojan detection using path delay fingerprint. IEEE International Workshop on Hardware-Oriented Security and Trust (HOST), 2008.

[11] S. Wei and M. Potkonjak. Scalable hardware Trojan diagnosis. IEEE Transactions on Very Large Scale Integration Systems (TVLSI), 20(6), pp. 1049-1057, 2012.

[12] R. Rad, J. Plusquellic and M. Tehranipoor. A sensitivity analysis of power signal methods for detecting hardware Trojans under real process and environmental conditions. IEEE Transactions on Very Large Scale Integration Systems (TVLSI), 18(12), pp. 1735-1744, 2010.

[13] H. Salmani and M. Tehranipoor. Layout-Aware Switching Activity Localization to Enhance Hardware Trojan Detection. IEEE Transactions on Information Forensics and Security, 7(1), pp. 76-87, 2012.

[14] S. Dupuis, P. Ba, G. Natale, M. Flottes, and B. Rouzeyre. A novel hardware logic encryption technique for thwarting illegal overproduction and Hardware Trojans. IEEE 20th International On-Line Testing Symposium (IOLTS), pp. 49-54, 2014.

[15] J. Rajendran, Y. Pino, O. Sinanoglu and R. Karri. Security analysis of logic obfuscation. ACM/IEEE Design Automation Conference, pp. 83-89, 2012.

[16] S. Shekarian, M. Zamani and S. Alami. Neutralizing a design-for-hardware trust technique. International Symposium on Computer Architecture and Digital Systems (CADS), pp. 73-78, 2013.

[17] X. Mingfu, H. Aiqun and L. Guyue: Detecting Hardware Trojan through Heuristic Partition and Activity Driven Test Pattern Generation. Communications Security Conference (CSC), pp. 1-6, 2014.

[18] H. Salmani, M. Tehranipoor and J. Plusquellic. A novel technique for improving hardware Trojan detection and reducing Trojan activation time. IEEE Transactions on Very Large Scale Integration Systems (TVLSI), 20(1), pp. 112-125, 2012.

[19] B. Zhou, W. Zhang, S. Thambipillai, and J. Teo. A low cost acceleration method for hardware Trojan detection based on fan-out cone analysis. ACM International Conference on Hardware Software Codesign and System Synthesis, p. 28, 2014.

[20] I. Pomeranz and S. Reddy. A measure of quality for n-detection test sets. IEEE Transactions on Computers, 53(11), pp. 1497-1503, 2004.

[21] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi and B. Sunar. Trojan Detection using IC Fingerprinting. IEEE Symposium on Security and Privacy, pp. 296-310, 2007.

[22] X. Wang, H. Salmani, M. Tehranipoor and J. Plusquellic. Hardware Trojan Detection and Isolation Using Current Integration and Localized Current Analysis. IEEE International Symposium on Defect and Fault Tolerance of VLSI Systems, pp. 87-95, 2008.

[23] S. Narasimhan, D. Du, R. Chakraborty, S. Paul, F. Wolff, C. Papachristou, K. Roy and S. Bhunia. Hardware Trojan detection by multiple-parameter side-channel analysis. IEEE Transactions on Computers, 62(11), pp. 2183-2195, 2013.

[24] D. Du, S. Narasimhan, R. Chakraborty and S. Bhunia. Self-referencing: a scalable side-channel approach for hardware trojan detection. International Workshop on Cryptographic Hardware and Embedded Systems (CHES), pp. 173-187, 2010.