# Efficient Test Generation for Trojan Detection using Side Channel Analysis

Yangdi Lyu and Prabhat Mishra
Department of Computer and Information Science and Engineering
University of Florida, Gainesville, Florida, USA

*Abstract*—**Detection of hardware Trojans is vital to ensure the security and trustworthiness of System-on-Chip (SoC) designs. Side-channel analysis is effective for Trojan detection by analyzing various side-channel signatures such as power, current and delay. In this paper, we propose an efficient test generation technique to facilitate side-channel analysis utilizing dynamic current. While early work on current-aware test generation has proposed several promising ideas, there are two major challenges in applying it on large designs: (i) the test generation time grows exponentially with the design complexity, and (ii) it is infeasible to detect Trojans since the side-channel sensitivity is marginal compared to the noise and process variations. Our proposed work addresses both challenges by effectively exploiting the affinity between the inputs and rare (suspicious) nodes. We formalize the test generation problem as a searching problem and solve the optimization using genetic algorithm. The basic idea is to quickly find the profitable test patterns that can maximize switching in the suspicious regions while minimize switching in the rest of the circuit. Our experimental results demonstrate that we can drastically improve both the side-channel sensitivity (30x on average) and time complexity (4.6x on average) compared to the state-of-the-art test generation techniques.**

## I. Introduction

Hardware Trojans are malicious modifications incorporated in simple Integrated Circuits (ICs) or complex System-on-Chip (SoC) designs [1]. As IC design and fabrication process become more and more globalized, the threat of hardware Trojan attack is increasing due to potential malicious modifications at different stages of the design and fabrication process [2]. There are several test generation efforts for detection of hardware Trojans [3]–[6].

The existing test generation approaches can be broadly categorized as logic testing and side-channel analysis. Side-channel analysis does not require the full activation of the Trojan or propagation of the Trojan effect to the observable outputs. However, detection of small Trojans can be hard since the change in side-channel signatures due to the Trojans can be negligible compared to the noise or process variations. Logic testing is immune to noise and process variations, but requires both the activation of the Trojan and propagation of the Trojan effects to the observable outputs. Since the number of possible input patterns are exponential [7], Trojan detection using logic testing can be infeasible for large designs. While MERS [3], [14] tried to combine the advantages of logic testing and side channel analysis, there are two major challenges in applying it on large designs. The test generation time using MERS grows exponentially with the design complexity. Moreover, it is infeasible to detect Trojans since the increase in side-channel sensitivity is marginal compared to the noise and process variations. Specifically, MERS can provide up to 3% sensitivity whereas typical process variations can be more than 10% [18]. Our proposed approach addresses both challenges.

This paper introduces an efficient approach to generate test patterns to maximize the side-channel sensitivity for Trojan detection. *In this paper, we target dynamic current as our side-channel signature.* However, this approach can also be extended to other side-channel parameters with suitable modifications of the evaluation criteria. Specifically, this paper makes the following major contributions:

- Exploits the input affinity to identify test patterns that can maximize switching in the suspicious (target) region while minimize switching in the rest of the circuit in order to significantly improve the side-channel sensitivity.
- Utilizes genetic algorithm to quickly find the profitable test patterns in order to improve the test generation time.
- The significant improvement in sensitivity enabled our approach to detect the majority of Trojans (out of randomly inserted 1000 Trojans), while the state-of-the-art approaches can detect less than 1% Trojans.

The paper is organized as follows. Section II describes existing Trojan detection techniques. Section III provides problem formulation and motivates the need for our work. Section IV describes our test generation framework. Section V presents experimental results. Finally, Section VI concludes the paper.

## II. Related Work

Existing Trojan detection techniques can be broadly classified into two categories: logic testing and side-channel analysis. Logic testing in Trojan detection has been extensively explored, such as ATPG based [8]–[10] and N-detect test [6]. The MERO approach presented in [6] utilized the idea of N-detect [17] to achieve high coverage over randomly sampled Trojans, assuming the trigger conditions of the Trojans consist of rare nodes only. The authors observed that if the generated test patterns are able to satisfy all rare values N times, it is highly likely that rare trigger conditions are satisfied when N is sufficiently large.

Logic testing approaches have several limitations such as lack of scalability due to long test generation time even for small benchmarks, restrictions of the trigger conditions

This work was partially supported by Cisco Systems.

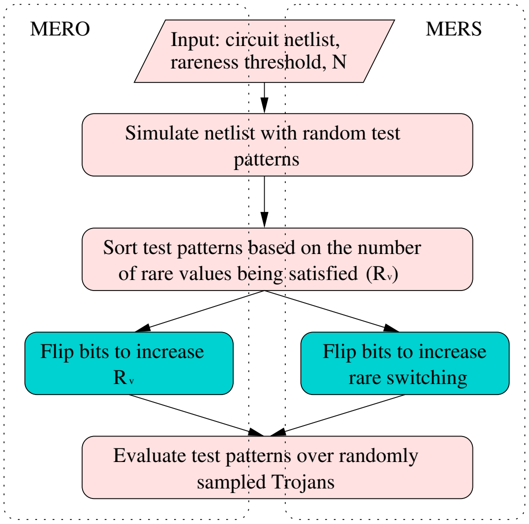

MERO
MERS
Input: circuit netlist, rareness threshold, N
Simulate netlist with random test patterns
Sort test patterns based on the number of rare values being satisfied (Rᵥ)
Flip bits to increase Rᵥ
Flip bits to increase rare switching
Evaluate test patterns over randomly sampled Trojans

Fig. 1. MERO versus MERS. While MERS counts the number of rare switching, and MERO counts the number of activated rare values.

being fully activated, and the effect of the inserted Trojan propagating to the observable points. Side-channel analysis overcomes these disadvantages. Trojan detection using side-channel analysis [11]–[16] measures transient current, power consumption, or path delay both in the golden design and the design under test. If the measured signals from these two designs vary by a threshold, a Trojan is suspected to be present.

Huang et al. [14] extended the idea of N-detect test for side-channel analysis, and proposed a test generation framework called MERS to maximize the sensitivity of dynamic current. The frameworks of MERS and MERO are similar as shown in Figure 1. MERS generates compact test patterns to let each rare node switch from its non-rare value to its rare value N times, increasing the probability of partially or fully activating a Trojan. The side-channel sensitivity of MERS is too small, typically less than 3% in most benchmarks [14], compared to large (7-17% [18]) environmental noise and process variations in today's CMOS circuits. The low side-channel sensitivity in [14] is due to the inherent restriction of reordering within the set of test patterns generated by MERS. Our proposed approach is able to effectively search for efficient tests that can drastically improve the side-channel sensitivity - making Trojan detection feasible in practice.

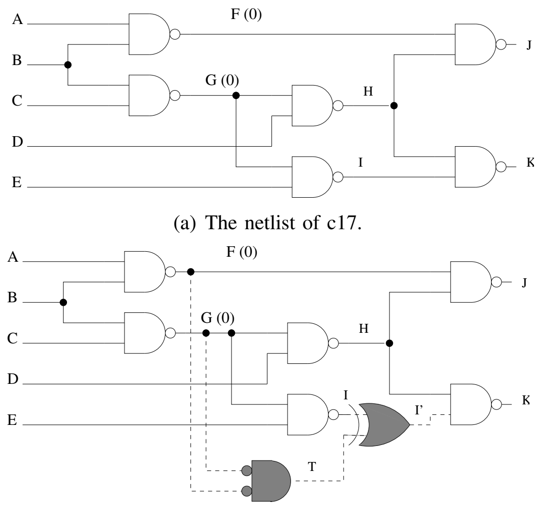
Searching for the best solution in a given search space is a common optimization problem. Genetic algorithm (GA) is a commonly used evolutionary search algorithm inspired by natural selection [19]. In test generation domain, genetic algorithm is shown to be successful in fault coverage [20] and Trojan detection [21]. To the best of our knowledge, our work is the first attempt in utilizing genetic algorithm for side-channel analysis aware test generation.

## III. Problem Formulation and Motivation

### A. Problem Formulation

Our goal is to generate $l$ compact test pattern pairs $(u_i, v_i)$ $(i = 1, 2, ..., l)$ that can maximize the dynamic current based side-channel sensitivity. For each pair of test patterns $(u_i, v_i)$, the current switching in the golden design $G$ is measured by applying $u_i$ followed by $v_i$, i.e., $switch^G_{u_i,v_i}$. The current switching in the Trojan-inserted design $G^T$ is defined in the same way, i.e., $switch^{G^T}_{u_i,v_i}$. The *relative switching* is computed as $|switch^G_{u,v} - switch^{G^T}_{u,v}|/switch^G_{u,v}$. Given the test pattern pairs, the ***sensitivity*** of a Trojan $T$ is defined as the maximum of the relative switching over all pairs, as shown in Equation 1.

$$sensitivity_T = max_{(u_i,v_i)}(\frac{|switch^G_{u_i,v_i} - switch^{G^T}_{u_i,v_i}|}{switch^G_{u_i,v_i}}) \quad (1)$$



(a) The netlist of c17.

(b) The netlist of c17 with Trojan inserted.

Fig. 2. The netlist of c17 from ISCAS'85 benchmark. We assume 0.3 as the threshold for rare nodes, F and G are the two rare nodes in this design, with their rare values 0. The top shows the golden design and the bottom shows the design with a Trojan inserted.

### B. An Illustrative Example

To illustrate how to improve sensitivity in dynamic current based side-channel analysis, we first use a small benchmark c17 from ISCAS'85 as an example with its netlist shown in Figure 2(a). We set *Rareness threshold* to be 0.3, which means rare values are satisfied with less than 30% probability in random simulation, e.g., F and G with rare value 0 in Figure 2(a).

Assume an attacker uses rare nodes F and G as the trigger condition and constructs a Trojan as shown using dashed lines in Figure 2(b). To detect this Trojan, applying the pair of test patterns $(u, v) = (11100, 10100)$ on inputs (<A,B,C,D,E>) is an ideal choice, with only B switches from '1' to '0'. The current switching in the golden design $switch^G_{u,v}$ is 4 (switching of signals B, F, G, and J) and the current switching in the Trojan inserted design $switch^{G^t}_{u,v}$ is 7 (switching of signals B, F, G, J, T, I', and K). Thus the sensitivity is 75% in this example. *An important observation is that by flipping only a small number of relevant inputs (B in this example) while preserving the others, the switching activities in the Trojan area are maximized while the current switching in the golden design is minimized.* In other words, if we can exploit the ***affinity*** between inputs and the rare nodes while creating a pair of test patterns ($u$ followed by $v$), it can lead to significant improvement in sensitivity for Trojan detection. Section V-D (Figure 6) demonstrates that affinity is useful in practice.

## IV. Generation of Effective Test Patterns

Figure 3 shows an overview of our proposed approach. It has three important steps. The first step finds the profitable initial test patterns (Section IV-B). The next step forms the effective pair of test patterns (Section IV-C). Finally, we evaluate the quality of the generated pairs of test patterns (Section V-C).

### *A. Motivation and Research Challenges*

By inspecting the capability of $(11100, 10100)$ for c17, we want to divide the task of searching for effective pairs of test patterns into two sub-problems. (1) Generation of good initial test patterns that can trigger rare conditions, e.g., 11100 in the previous example. As the difference of current switching in designs with/without Trojans comes from the inserted circuits and the switching after payloads are activated and propagated, the sensitivity can be improved if the test patterns can trigger rare conditions. (2) Given any test pattern $u$ generated in the previous step, searching for the best succeeding pattern $v$ to maximize the sensitivity, e.g., 10100 in the previous example.

However, there are three main challenges in searching for the best succeeding pattern for $u$.

1) Randomly selected pairs may not lead to high sensitivity, even if the two patterns are similar. For example, if we apply $(u, v) = (11100, 11101)$ to the previous example, the current switching in $G$ and $G^T$ remains the same, revealing no side-channel footprint.
2) Large search space. The whole search space is exponentially large ($2^n$, where $n$ is the number of inputs in the design). So, searching for the whole space is not feasible. Based on affinity heuristic, the neighbor of $u$ with Hamming distance less than $k$ is the optimized search space. One naive way is to use breadth-first-search (BFS) according to the Hamming distance. However, the searching complexity is still $O(n^k)$.
3) There is a tradeoff between introducing switching in the rare nodes and minimizing switching in the golden design. We need to introduce as much switching in all rare nodes as possible, since we have no knowledge of the trigger condition. However, for a design with thousands of rare nodes, introducing switching for all of them can lead to significant increase in switching of the golden design. In that case, even if the Trojan is fully activated, the sensitivity (extra switching) can be too small compared to process and noise margins.

Our approach addresses these challenges by using genetic algorithm as an approximate and optimized replacement of BFS. The first population in GA is initialized with random test patterns that have fixed small Hamming distance from $u$. By crossover and mutation, the Hamming distance is expected to grow slowly. After several generations, majority of the profitable test patterns in the expected search space are likely to be visited.

### *B. Generation of Profitable Initial Test Patterns*

The sensitivity of side channel analysis is maximized if the test pattern pairs are able to partially or fully activate trigger condition. Thus, our first task is similar to other logic testing techniques, such as ATPG or N-detect approach. In this paper, we choose to use MERO [6] to generate N-detect test patterns. As introduced in [6], the generated test patterns are compact and can statistically achieve good coverage when N increases. MERO is used as a black box in our approach, and the parameters are introduced in Section V. We denote the generated $l$ test patterns as $\{u_i\}$ $(i = 1, 2, ..., l)$.

### *C. Searching for the Best Succeeding Pattern*

The second task is to find the best succeeding pattern $v_i$ for each $u_i$ (identified in Section IV-B), such that the relative switching is maximized. To achieve both high-quality pairs and test generation efficiency, we use genetic algorithm as our searching algorithm.

Genetic algorithm forms the main part of Algorithm 1, which consists of four major steps: initialization, fitness computation, selection, and crossover and mutation. The ***fitness*** is defined in Equation 2, where $rare\_switch^G_{u,v}$ represents the current switching of all rare nodes in $G$ when applying the test pattern $u$ followed by $v$. A profitable test pattern should maximize the current switching in rare nodes to increase the probability of activating a Trojan, and minimize the switching in the golden design. The best succeeding pattern $v_i$ for a given preceding $u_i$ is the one achieving highest fitness value over all generations (line 12). The first iteration of GA for c17 is shown in Figure 4, assuming 4 individuals in each generation.

$$fitness_u(v) = \frac{rare\_switch^G_{u,v}}{switch^G_{u,v}} \qquad (2)$$

**Algorithm 1** $TestGeneration$

```
    Input: circuit netlist, N
    Output: pairs of test patterns
 1: Simulate the circuit netlist with random test patterns
 2: Generate N-detect test patterns $\{u_i\}$ $(i = 1, 2, ..., l)$
 3: for $i = 1$ to $l$ do
 4:     Initialization of GA with $u_i$
 5:     For each individual $v$, compute $fitness_{u_i}(v)$ by simulating the netlist with the pair of test patterns $(u_i, v)$
 6:     for $gen = 1$ to $generations$ do
 7:         Selection of parents from the $gen^{th}$ generation based on fitness values
 8:         Single point crossover to produce children
 9:         Single point mutation according to mutation rate
10:         Compute fitness for the children ($(gen + 1)^{th}$ generation)
11:     end for
12:     Select the best individual over all generations as $v_i$
13: end for
14: Return the pairs of test patterns $(u_i, v_i)$ $(i = 1, 2, ..., l)$
```

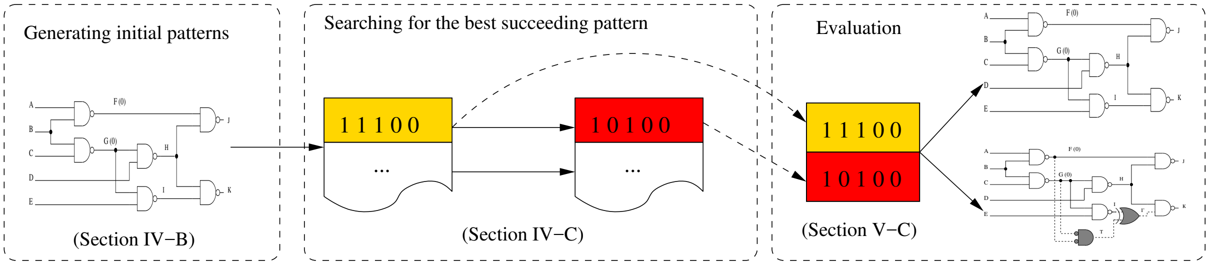


Fig. 3. The overview of our approach. We divide the task of test generation into two sub-problems: (i) generation of good initial test patterns that can trigger rare conditions (ii) given any test pattern generated in the previous step, searching for the best succeeding pattern that can maximize side-channel sensitivity.

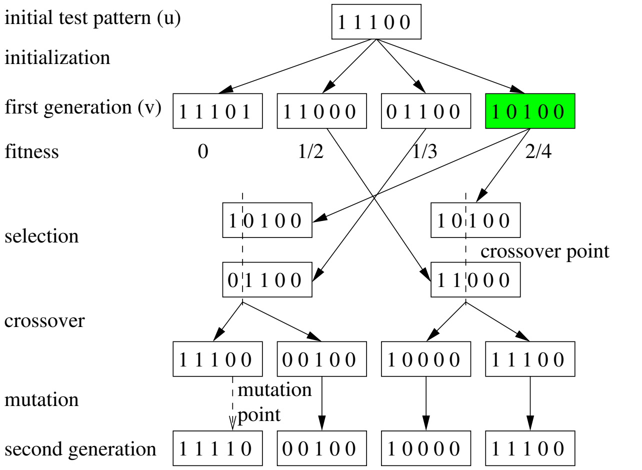


Fig. 4. The first iteration of GA for generating the best succeeding pattern for 11100. The initial Hamming distance is 1 and the number of individuals for each generation is 4. Crossover point and mutation point are selected randomly. Fitness values are shown as the numerator representing rare switching $rare\_switch^G_{u_i,v}$ and the denominator representing total switching $switch^G_{u_i,v}$. The best succeeding pattern $v_i$ is the individual with the largest fitness value over all generations as shown in the green box.

*1) Initialization:* The first population is initialized with random test patterns that are similar to $u_i$. Each individual in the initial population has Hamming distance $k$ from $u_i$. During experiments, we choose $k$ to be $\max(0.4\%|u_i|, 1)$.

*2) Fitness Computation:* For each individual $v$, the golden design $G$ is simulated with the pair of test patterns $(u_i, v)$. Then the fitness of $v$ is computed by Equation 2. For example, the fitness values for four candidates are shown in Figure 4.

*3) Selection:* Selection is based on the fitness of each individual. Individuals with higher fitness are more likely to be selected. The selection shown in Figure 4 demonstrates that the individuals with higher fitness values (such as 10100) are more likely to be selected than the ones with lower fitness values (such as 11101).

*4) Crossover and Mutation:* During crossover, a single crossover point is randomly selected and crossover is performed on parents to produce two children. During mutation, a randomly selected position is mutated with a low mutation rate. For example, Figure 4 shows only 1 mutation for 4 individuals.

Although the Hamming distance of all individuals in the initial generation and $u_i$ is small, crossover and mutation will increase the Hamming distance between each generation and $u_i$. Theoretically, the largest possible Hamming distance between the $i^{\text{th}}$ generation and $u_i$ is at most $2^i * |k|$ considering only crossover. In order for all test patterns to be evaluated with some probability, the total number of generations should be large enough to allow $|u_i|$ Hamming distance. However, as the affinity heuristic suggests that we may need only a small number of generations. During experiments, we fix the number of generations to be 5. So the maximum Hamming distance could be $2^5 \times 0.4\%|u_i|$ which is around $10\%|u_i|$. By exploring around $u_i$ with low Hamming distance, we expect to get high quality pairs efficiently. As shown in Figure 6, most profitable pairs of test patterns have small Hamming distances that provided significant improvement in sensitivity.

## V. Experiments

### A. Experimental Setup

To evaluate the effectiveness of our approach, we implemented our framework in C++. Since MERS is the state-of-the-art (closest to our approach), we used exactly the same benchmarks as MERS - a subset of ISCAS-85 and ISCAS-89 gate-level benchmark circuits. We performed our experiments on a machine with Intel Xeon E5-2698 CPU @2.20GHz. We compared the results of our approach to MERS-s (MERS with simulation based reordering) with C = 5.0 (best result settings from [14]). We did not compare with random tests and MERS (with Hamming distance) since MERS-s outperforms them.

### B. Generation of Initial Test Patterns

We first simulate the benchmarks with random test patterns, and calculate the probability of each node achieving each possible value. All the nodes with probability lower than the rareness threshold is marked as rare nodes. To enable a fair comparison, similar to MERS, we set the number of random test patterns and rareness threshold to 10,000 and 0.1, respectively, for all benchmarks.

By applying N-detect approach with $N = 1000$, we generate the initial test patterns. The length of test patterns and the running time of MERO are reported in Table I. The length of each test pattern is the number of primary inputs for combinational circuits, and the number of primary inputs plus the number of flip-flops in sequential circuits (same full-scan assumption as MERS [14]). As the algorithms of MERS and

MERO are almost the same, the running time and the length of test patterns by these two approaches are similar. The small difference in the length of test patterns is because MERO asks for N times activation of rare values, while MERS asks for N times switching from non-rare values to rare values.

TABLE I
MERO AND MERS TEST PATTERN LENGTH AND RUNTIME

| Benchmarks | #Rare nodes | #Test patterns MERO / MERS | Length of One Test | Running time (s) |
|---|---|---|---|---|
| c2670 | 64 | 5300 / 5306 | 233 | 444 |
| c5315 | 255 | 7927 / 8066 | 178 | 1589 |
| c7552 | 306 | 8971 / 7935 | 207 | 2884 |
| s13207 | 604 | 9304 / 9659 | 687 | 5517 |
| s15850 | 681 | 9445 / 9512 | 590 | 6042 |
| s35932 | 896 | 3034 / 3083 | 1764 | 11047 |

Trojans are randomly generated with 8 triggers from rare nodes and one payload each (same as MERS [14]). We randomly sampled 1000 Trojans from each benchmark. The probability of activating these Trojans using random simulation is at most $10^{-8}$ if the triggers are independent.

### C. Evaluation

In our GA-based framework, we set the number of individuals to be 200 in each generation, the number of generations to be 5, and mutation rate to be 0.1. For each test pattern $u_i$ generated by N-detect approach, GA generates a test pattern $v_i$ to form a pair. Then, we apply test patterns $\{(u_i, v_i)\}$ to both the golden design and the Trojan inserted design.

Table II shows the result of our approach compared to MERS-s. The sensitivity of test patterns is measured by the average of the sensitivities over 1000 randomly sampled Trojans. The original switching represents the current switching in the golden design. The two columns of time in Table II represent the running time for simulation-based reordering for MERS-s, and searching for succeeding patterns in our approach, respectively. We use the sensitivity threshold of 10% [15]. The percentage of Trojans detected in the last columns of MERS-s and our approach shows the fraction of Trojans whose sensitivity is above this threshold.

*1) Test length comparison:* As our approach finds a pair for each test pattern generated by N-detect, the total length of the test patterns generated by our approach is twice the number of the test patterns shown in Table I. MERS-s reorders the test patterns generated by MERS without generating any new test pattern. So the length of test patterns by our approach is twice the number of test patterns by MERS-s. However, as we measure the current switching for each pair, and MERS-s measures the current switching between each two sequential test patterns, the total numbers of measurements are the same.

*2) Sensitivity comparison:* The overall sensitivity of our approach improves by a factor of 30.5 compared to MERS-s. Table II indicates that the advantage of our approach compared to MERS lies in effective reduction of original switching. As the reordering of MERS restricts each test pattern to find its pair from MERS test patterns, the minimum original switching that reordering can achieve is bounded by the optimum pairs inside these test patterns. On the other hand, our approach fixes the preceding pattern to be good at activating trigger condition and searches the whole space for profitable succeeding patterns to minimize the original switching.

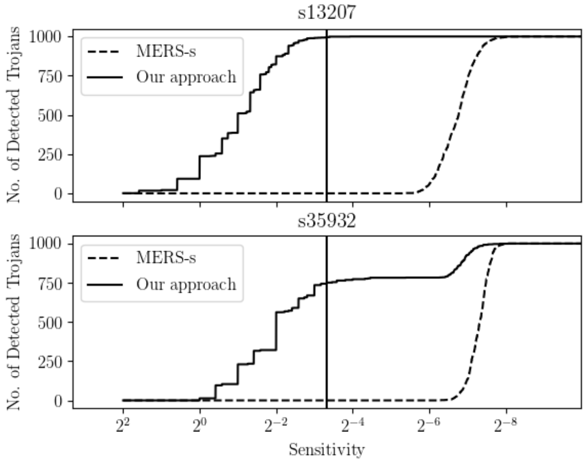


Fig. 5. The comparison of cumulative distributions of sensitivities by our approach versus MERS-s over 1000 Trojans. The $x$-axis shows the sensitivity in logarithmic axis, and $y$-axis is the number of Trojans that have sensitivities greater than $x$. The vertical line represents 10% process variations as a threshold to detect Trojans.

Next, we inspect the sensitivity for each Trojan independently. The cumulative distribution of the sensitivities over 1000 Trojans in s13207 and s35932 are shown in Figure 5 for our approach and MERS-s. The $x$-axis is the sensitivity, $y$-axis is the number of Trojans that have sensitivities greater than $x$, and the vertical line represents 10% sensitivity. For example in s13207, almost all the Trojans have sensitivities greater than the sensitivity threshold in our approach, while in MERS-s this number is 0. In other words, if we assume the process variation is 10%, our approach can detect the majority of these randomly sampled Trojans with high confidence, while MERS-s can not detect any of them. The exact numbers are reported in the last columns of both approaches in Table II. Overall, our approach can detect majority of the Trojans in most of the benchmarks due to higher sensitivity provided by our test patterns, whereas the test patterns generated by MERS fail to detect any of them.

*3) Test generation time:* Finally, we compare the test generation time of our approach with MERS-s. Table II indicates that our approach is much more efficient than MERS-s by an improvement factor of 4.6. The running time of reordering in MERS-s is $O(n^2/2)$ to simulate each pair of MERS test patterns, where $n$ is the number of test patterns generated by MERS. The time complexity of our approach is $O(n * 1000)$, as for each test pattern in MERO, we evaluate 5 generations and 200 individuals for each generation. As the numbers of MERS test patterns and MERO test patterns are almost the same, our approach is significantly faster than MERS-s (up to 6.2x, 4.6x on average).

### D. Validation of Affinity Heuristic

For each test pattern $u$ from Section V-B, the best succeeding pattern $v$ is found by GA with 5 generations. Thus,

TABLE II

Comparison of MERS-s (C = 5.0) with our approach over 1000 random 8-trigger Trojans. Our approach can provide an order-of-magnitude improvement in sensitivity while reduces the test generation time by 4.6x on average. As a result, our approach can detect almost all the Trojans in majority of the benchmarks while MERS fails to detect any of them.

| Benchmarks | MERS-s (C = 5.0) | | | | Our Approach | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Avgerage Orig Switch | Sensitivity | Time (s) | % of Trojans detected [1] | Avgerage Orig Switch | Sensitivity | | Running Time | | % of Trojans detected [1] |
| | | | | | | Sensitivity | Impro. factor | Time (s) | Impro. factor | |
| c2670 | 271.3 | 3.8% | 3476 | 0% | 13.0 | 41.2% | 10.8x | 1008 | 3.4x | 100% |
| c5315 | 888.5 | 1.1% | 25178 | 0% | 29.4 | 16.9% | 15.4x | 4795 | 5.3x | 59.2% |
| c7552 | 477.0 | 1.1% | 49978 | 0% | 34.0 | 12.2% | 11.1x | 8002 | 6.2x | 51.9% |
| s13207 | 556.4 | 1.0% | 24385 | 0% | 5.4 | 61.9% | 61.9x | 4381 | 5.6x | 99.6% |
| s15850 | 698.4 | 0.9% | 29734 | 0% | 12.4 | 34.9% | 38.8x | 5795 | 5.1x | 87.9% |
| s35932 | 1338.6 | 0.6% | 6795 | 0% | 373.6 | 27.0% | 45.0x | 3696 | 1.8x | 75.1% |
| **Average** | 572.6 | 1.4% | 19560 | 0% | 104.9 | 32.4% | 30.5x | 3956 | 4.6x | 79% |

[1] We use sensitivity threshold of 10% based on [18] . A Trojan $t$ is detected if the $sensitivity_t > 10\%$.

the maximum possible Hamming distance between $u$ and $v$ can be as large as $10\%|u|$ (see Section IV-C). To empirically demonstrate the validity of the affinity heuristic, we want to show that the optimum solution $v$ comes from the small neighborhood of $u$.

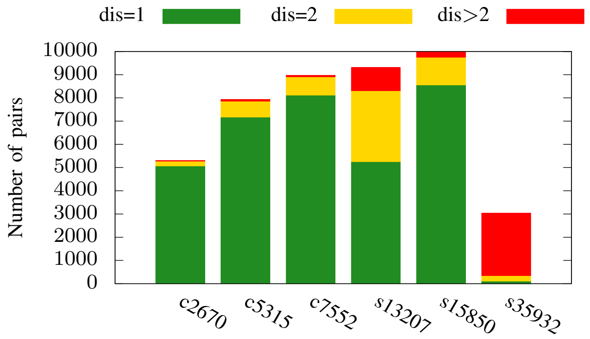


Fig. 6. Hamming distance of all test pattern pairs.

The Hamming distances of all optimum test pattern pairs are plotted in Figure 6. We can see that the majority of Hamming distances are less than 3. Actually, most of the distances are 1 for all benchmarks except s35932. The results indicate why a small number of generations (faster runtime) can provide significant sensitivity improvement.

## VI. Conclusion

Side-channel analysis provides a promising approach for Trojan detection. State-of-the-art test generation technique (e.g., MERS [8]) is not beneficial for large designs due to its high runtime complexity. Most importantly, the sensitivity obtained by existing approaches is very low compared to environmental noise and process variations, making them useless in practice. Our proposed approach addresses both limitations by developing a genetic algorithm based test generation algorithm that can lead to drastic increase in sensitivity while significantly reduce the test generation time. Our approach breaks down the problem into two sub-problems. The first task generates efficient test patterns to maximize the excitation of rare values, such that the trigger conditions are satisfied statistically. The second task finds the best matching pair for each test pattern generated by the first task to maximize the sensitivity. In this paper, we demonstrated that the combination of N-detect test generation with genetic algorithm can generate significantly better test patterns than MERS. Our proposed test generation approach can improve both side-channel sensitivity (up to 61.9x, 30.5x on average) and test generation time (up 6.2x, 4.6x on average) compared to MERS. Experimental results demonstrated that our approach can detect the majority of Trojans in the presence of process variation and noise margin while state-of-the-art approaches fail.

## References

[1] P. Mishra, S. Bhunia and M. Tehranipoor (Editors), "Hardware IP Security and Trust," *Springer*, ISBN: 978-3-319-49024-3, 2017.

[2] F. Farahmandi *et al.*, "Trojan localization using symbolic algebra," in *Asia and South Pacific Design Automation Conference*, 2017.

[3] Y. Huang, S. Bhunia and P. Mishra, "Scalable test generation for Trojan detection using side channel analysis," in *TIFS*, 13(11), 2018.

[4] A. Ahmed *et al.*, "Scalable hardware Trojan activation by interleaving concrete simulation and symbolic execution," in *ITC*, 2018.

[5] Y. Lyu, A. Ahmed and P. Mishra, "Automated activation of multiple targets in RTL models using concolic testing," in *DATE*, 2019.

[6] R. Chakraborty *et al.*, "MERO: a statistical approach for hardware Trojan detection," in *CHES Workshop*, 2009.

[7] M. Chen *et al.*, "System-Level Validation: High-Level Modeling and Directed Test Generation Techniques," *Springer*, 2012.

[8] M. E. Amyeen *et al.*, "Evaluation of the quality of N-detect scan ATPG patterns on a processor," in *International Test Conference*, 2004.

[9] F. Wolff *et al.*, "Towards trojan-free trusted ICs: Problem analysis and detection scheme," in *Design, Automation and Test in Europe*, 2008.

[10] J. Cruz *et al.*, "Hardware Trojan detection using ATPG and model checking," in *International Conference on VLSI Design*, 2018.

[11] 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. Trans. in VLSI*, 2010.

[12] Y. Lyu and P. Mishra, "A survey of side channel attacks on caches and countermeasures," *Journal of Hardware and Systems Security*, 2(1), 2018.

[13] H. Salmani and M. Tehranipoor, "Layout-aware switching activity localization to enhance hardware Trojan detection," *IEEE TIFS*, 2012.

[14] Y. Huang, S. Bhunia, and P. Mishra, "MERS: Statistical test generation for side-channel analysis based Trojan detection," in *ACM CCS*, 2016.

[15] D. Agrawal *et al.*, "Trojan detection using IC fingerprinting," in *2007 IEEE Symposium on Security and Privacy*, 2007.

[16] Y. Jin and Y. Makris, "Hardware Trojan detection using path delay fingerprint," in *Hardware-Oriented Security and Trust*, 2008.

[17] I. Pomeranz and S. M. Reddy, "A measure of quality for n-detection test sets," *IEEE Transactions on Computers*, 2004.

[18] B. Balaji *et al.*, "Accurate characterization of the variability in power consumption in modern mobile processors," in *HotPower*, 2012.

[19] M. Mitchell, *An introduction to genetic algorithms*. MIT Press, 1996.

[20] E. M. Rudnick *et al.*, "A genetic algorithm framework for test generation," *IEEE Transactions on CAD*, 1997.

[21] S. Saha *et al.*, "Improved test pattern generation for hardware Trojan detection using genetic algorithm and boolean satisfiability," *CHES*, 2015.