Automated Trigger Activation by Repeated Maximal Clique Sampling

Yangdi Lyu Prabhat Mishra

Department of Computer and Information Science and Engineering
University of Florida
Gainesville, FL 32611
e-mail: {lvyangdi, prabhat}@ufl.edu

Abstract— Hardware Trojans are serious threat to security and reliability of computing systems. It is hard to detect these malicious implants using traditional validation methods since an adversary is likely to hide them under rare trigger conditions. While existing statistical test generation methods are promising for Trojan detection, they are not suitable for activating extremely rare trigger conditions in stealthy Trojans. To address the fundamental challenge of activating rare triggers, we propose a new test generation paradigm by mapping trigger activation problem to clique cover problem. The basic idea is to utilize a satisfiability solver to construct a test corresponding to each maximal clique. This paper makes two fundamental contributions: 1) it proves that the trigger activation problem can be mapped to clique cover problem, 2) it proposes an efficient test generation algorithm to activate trigger conditions by repeated maximal clique sampling. Experimental results demonstrate that our approach is scalable and it outperforms state-of-the-art approaches by several orders-of-magnitude in detecting stealthy Trojans.

I. Introduction

Due to increasing System-on-Chip (SoC) complexity and stringent time-to-market constraints, SoC supply chain involves multiple third parties. A malicious third-party can insert hardware Trojans [1], [2] during any stages in the development cycle starting from design implementation to fabrication. These malicious modifications may alter the original functionality or leak secret information. To remain covert under in-field testing, a hardware Trojan is carefully designed to be triggered by extremely rare circuit input events. An example hardware Trojan is shown in Figure 1 with corresponding *trigger* and *payload*. The trigger condition is usually constructed by a few signals that can be activated under rare conditions. The figure illustrates a beneficial way to assemble the rare signals (A, B and C) to form a rare input event. If the selected signals are independent, the probability of triggering this condition is multiplication of all the probabilities of these signals. Due to the stealthy nature of these Trojans, the trigger condition may not be activated during traditional validation and regression testing. Therefore, it is paramount to have efficient validation approaches that can activate rare trigger conditions to enable Trojan detection.

To detect hardware Trojans, various approaches have been proposed including logic testing [3]–[7] and side-channel analysis [8]–[15]. However, existing approaches are neither effective nor scalable to large designs with extremely rare trigger conditions. Logic testing requires test vectors to fully activate trigger condition and also propagate the effects to observable outputs. In contrast to logic testing, side-channel analysis detects hardware Trojans by observing the side effects of inserted gates. Since the Trojans are very small (few gates in a million-gate design), their side-channel footprint can easily hide within process variation and environmental noise margins. Although side-channel analysis does not require activation of trigger conditions, the activation could significantly improve side-channel sensitivity. Therefore, trigger activation is a fundamental problem in both logic testing and side-channel analysis based Trojan detection.

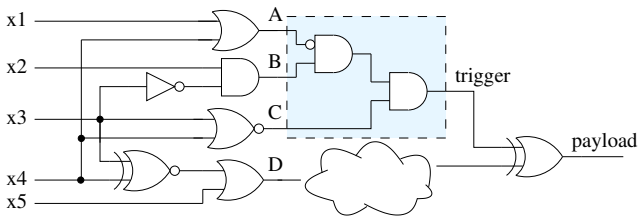


Fig. 1: An example hardware Trojan with a trigger condition constructed by three rare signals (A, B, C).

Trigger activation is a major challenge due to the exponentially large space that an adversary can exploit to construct trigger conditions. Conventional validation approaches simulate the design using millions or billions of random/constrained-random test vectors, and hope that one of these tests will activate the trigger condition. MERO [3] is one of the constrained-random test generation approaches that is similar to $N-$detect stuck-at ATPG [16], [17]. However, it is not effective in large designs with extremely rare trigger conditions as demonstrated in Section III. Existing directed test generation techniques are beneficial for known targets, but not useful for unknown targets (trigger conditions) since it leads to exponential complexity as discussed in Section II.

In this paper, we solve the trigger activation problem by mapping it to the problem of covering maximal cliques in a graph. Our goal is to activate extremely rare trigger conditions that can be covert during traditional validation. The major contributions of this paper are as follows:

1) To the best of our knowledge, our approach is the first attempt to map trigger activation problem to maximal clique cover problem.
2) We propose an efficient and scalable test generation algorithm for Trigger Activation by Repeated MAximal Clique sampling (TARMAC).
3) Experimental results demonstrate that TARMAC outperforms the state-of-the-art test generation techniques by several orders-of-magnitude for extremely rare-to-activate trigger conditions in large designs.

978-1-7281-4123-7/20/$31.00

The rest of this paper is organized as follows. Section II surveys prior efforts in trigger activation. In Section III, we motivate the need for this work by highlighting the drawbacks of $N-$detect paradigm as well as the limitations of the state-of-the-art test generation approaches. Section IV describes our proposed test generation framework. Section V presents the experimental results. Section VI concludes this paper.

II. Related Work

Random and constrained-random tests are widely used in traditional functional validation methodology. Unfortunately, even billions or trillions of constrained-random tests cannot cover many complex and corner-case scenarios in today's industrial designs. Directed tests are promising in such cases to activate the specific targets that were not covered by random or constrained-random tests. There are a wide variety of directed test generation techniques [18]–[22] for functional validation. Unfortunately, these techniques are not beneficial for unknown Trojans with extremely rare trigger conditions since it leads to exponential complexity.

Statistical test generation is a promising alternative to directed tests. The basic idea is to activate the rare signals as much as possible to increase the likelihood of activating the actual (unknown) trigger consisting of rare signals. In [3], the authors proposed a tool named MERO to generate $N-$detect test for logic testing. MERO achieves $N-$detect criteria by constrained random approach. It starts with a large number of random test vectors, and flips each bit of random vectors to increase $N-$detect criteria. MERO is shown to be effective in small designs (e.g., ISCAS benchmarks [23], [24]) with relatively easy-to-activate trigger conditions. However, MERO is unsuitable for large designs (scalability problem) as well as hard-to-detect triggers as demonstrated in Section III.

III. Motivation

$N-$detect paradigm has been successful in both logic testing [3], [17] and side-channel analysis [15]. $N-$detect paradigm requires the test set to activate each rare signal N times and is statistically effective for trigger activation given "sufficiently" large N [3]. The probability of activating trigger conditions will significantly decrease when the trigger condition is composed of very rare signals. It is expected that increasing N can increase the chances of hitting trigger conditions. However, larger N will significantly deteriorate the test generation performance and increase the required test length. MERO incorporated $N-$detect [3] with deterministic flipping method. There are two major problems that make it ineffective for activating hard-to-detect trigger conditions in large designs.

Scalability Problem: The test vectors generated by MERO cannot guarantee that each rare signal is activated at least N times. To ensure $N-$detect for all rare signals, the number of initial random vectors should be extremely large even for small benchmarks. For example, to achieve $N = 1000$ for all rare signals, 200K initial random vectors are not enough for small designs, as shown in Figure 2. We observed that some extremely rare signals are almost never activated, while easy-to-activate signals are activated more than N times. It is expected that for large designs, billions of random vectors

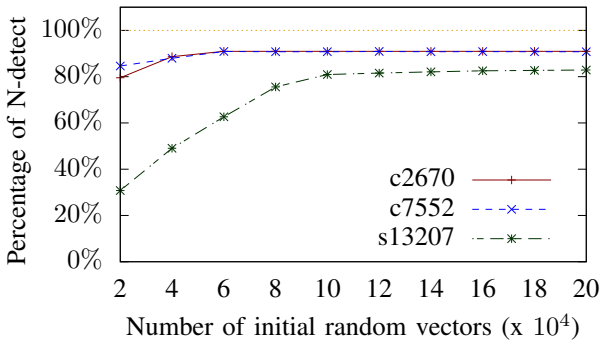


Fig. 2: The percentage of rare signals that are activated at least N times by MERO [3] with different number of random vectors.

are required to satisfy $N = 1000$. Since MERO requires *one simulation per bit flipping*, the total number of simulations would be in the order of billions or trillions, which makes this approach impractical for large designs.

Poor Trigger Coverage: MERO uses a vague notion of N being "sufficiently" large to ensure high trigger coverage. In fact, MERO simply selected $N = 1000$ in [3] for all benchmarks. Despite the fact that all rare signals are activated at least 1000 times in the small benchmark, such as c5315, the trigger coverage is only 50.6% (see Section V-V-B). In other words, $N = 1000$ is not "sufficiently" large for such a small benchmark. For larger designs with more trigger points and lower rareness threshold, larger N is required to reach a reasonable coverage, which needs drastically larger number of initial random vectors as discussed above, making scalability issue even worse.

Given poor trigger coverage and scalability issue of MERO and $N-$detect, new paradigms are need to solve trigger activation problem. In this paper, we address the fundamental challenge of trigger activation by mapping it to clique cover problem and finding the test patterns to cover maximal cliques, as outlined in the next section.

IV. TARMAC Methodology

In this section, we propose a new paradigm to solve trigger activation problem by mapping it to maximal clique cover problem, as shown in Figure 3. Our approach first constructs a satisfiability graph based on the design (e.g., gate-level netlist). Then, it finds maximal satisfiable cliques ($\mathcal{MSC}$s) in the satisfiability graph. Finally, it utilizes a satisfiability modulo theories (SMT) solver to generate one test for each maximal satisfiable clique.

A. Definition and Notations

Without loss of generality, we consider gate-level implementation of designs. We call the graph level representation of the design *a Design Graph* ($\mathcal{DG}$), where each vertex represents a signal and each edge represents the connectivity (via a gate). For each signal, we compute its logic expression (le) from its corresponding logic cone. Assuming design-for-debug architecture (e.g., scan chain), we allow all registers to be free variables. For example, the logical expression of vertex A in Figure 1 is $A.le = x_1 \vee x_4$.

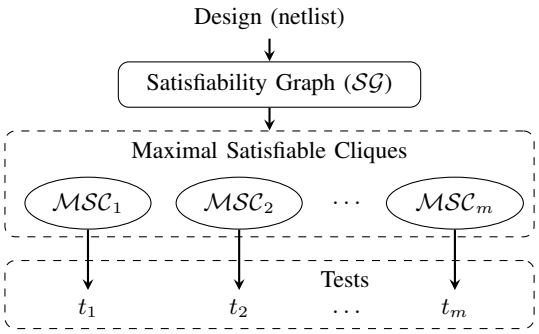


Fig. 3: Overview of our proposed (TARMAC) paradigm.

For each design, trigger conditions can be constructed from a subset of its signals and their corresponding rare value rv, which we refer as ***potential trigger signals*** *(PTS)*. PTS could be any subset of whole signals. In [3], PTS is the set of rare signals that are used to construct hard-to-activate trigger condition. We call a trigger signal is *activated* if it satisfies its rare value. We define *satisfiability graph* as follows.

Definition 1. *A* **Satisfiability Graph** *($\mathcal{SG}$) consists of vertices representing PTS and their satisfiability connections, $\mathcal{SG} = \{\mathcal{V}, \mathcal{E}\}$ where $\mathcal{V} = PTS$. If $(u.le == u.rv) \wedge (v.le == v.rv)$ is satisfiable, then there exists an edge between u and v, i.e., $u \in \mathcal{E}(v)$ and $v \in \mathcal{E}(u)$.*

Figure 4 shows the satisfiability graph for the example in Figure 1 with four PTS (A, B, C, D) and their corresponding rare values (0, 1, 1, 0). For example, the edge between A and B exists since input pattern 01000 satisfies the condition $(A.le == 0) \wedge (B.le == 1)$. However, there is no input pattern that satisfies $(C.le == 1) \wedge (D.le == 0)$, i.e., there is no edge between C and D.

B. Mapping Trigger Activation to Clique Cover Problem

A fundamental contribution of this paper is to show that trigger activation problem can be mapped to clique cover problem. First, we show that any valid trigger condition forms a clique in satisfiability graph $\mathcal{SG}$.

Lemma 1. *For any valid trigger condition with k rare signals $\{v_1, v_2, ..., v_k\}$, the vertices $\{v_1, v_2, ..., v_k\}$ form a $k-$clique in the satisfiability graph $\mathcal{SG}$.*

Lemma 1 holds since a valid trigger condition should have at least one input pattern that activates all signals in $\{v_1, v_2, ..., v_k\}$ simultaneously. Therefore, $(v_i.le == v_i.rv) \wedge (v_j.le == v_j.rv)$ is satisfiable for any i, j.

Note that it is possible to have a clique in the satisfiability graph that does not represent a valid trigger condition. For example, consider the clique ABD in Figure 4. There is no input pattern that satisfies the condition $(x_1 \vee x_4 == 0) \wedge (x_2 \wedge \neg x_3 == 1) \wedge (\neg(x_3 \oplus x_4) \vee x_5 == 0)$, although there are edges between any two of the three vertices. In other words, ABD forms a clique in $\mathcal{SG}$, but it does not represent a valid trigger condition. Clearly, an adversary will not use it as a Trojan trigger since it cannot be triggered. For the ease of illustration, we define *satisfiable clique* in Definition 2.

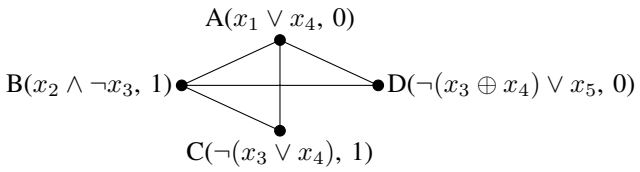


Fig. 4: Satisfiability graph with 4 PTS (A,B,C,D) from Figure 1, with logic expressions and rare values in parentheses.

Definition 2. *A satisfiable clique $\mathcal{SC}$ is a clique in a satisfiability graph $\mathcal{SG}$, where all the vertices of $\mathcal{SC}$ can be activated by the same input vector.*

It is clear to see that a satisfiable clique $\mathcal{SC}$ represents a valid trigger condition since the vertices can be activated by the same input. Similarly, any valid trigger condition can represent a satisfiable clique, due to Lemma 1 and its validity. Next, we explore the relationship between satisfiable cliques and valid trigger conditions as shown in Theorem 1, which points out a new way to solve trigger activation problem, i.e., finding test vectors to cover satisfiable cliques in a satisfiability graph.

Theorem 1. *The mapping between the set of valid trigger conditions and the set of satisfiable cliques is a bijection.*

Proof. As different trigger conditions consist of at least one different trigger signal, the corresponding satisfiable cliques have at least one different vertex. Hence, no two valid trigger conditions map to the same satisfiable clique, i.e., the mapping from the set of valid trigger conditions to the set of satisfiable cliques is an injection. Similarly, we can prove that the mapping from the set of satisfiable cliques to the set of valid trigger conditions is also an injection. Therefore, we have a one-to-one mapping between these two sets. □

C. Test Generation Algorithm

The rationale behind our test generation approach is that if we are able to find a test vector that can satisfy a clique, it is not necessary to generate any more test for all the trigger conditions represented by its subgraphs. Therefore, the most profitable test vector is the one that can satisfy the largest clique. Similar to cliques in graph theory, we define a *maximal satisfiable clique*.

Definition 3. *A maximal satisfiable clique ($\mathcal{MSC}$) is a satisfiable clique to which no more vertices can be added.*

Let $\{\mathcal{MSC}_1, \mathcal{MSC}_2, \ldots, \mathcal{MSC}_m\}$ represents the complete set of maximal satisfiable cliques, where m is the total number of maximal satisfiable cliques. For example, $\{\mathcal{MSC}_1$ = ABC, $\mathcal{MSC}_2$ = AD, $\mathcal{MSC}_3$ = BD$\}$ represents the complete set of maximal satisfiable cliques in Figure 4. Our goal is to find a test set that is able to activate all these maximal satisfiable cliques. However, finding all maximal satisfiable cliques is not scalable. The worst running time of finding all maximal cliques is $O(3^{n/3})$ [28], where n is the number of vertices. It is expected that finding all maximal satisfiable cliques takes longer than finding maximal cliques since it needs to check sub-graphs when a clique is not satisfiable.

Algorithm 1 Test Generation using Random Sampling and Lazy Construction (TARMAC)

1: **procedure** $TestGeneration$(circuit netlist CN, potential trigger signals PTS, maxVectorNumber VN)
2: $\mathcal{DG}$ = ConstructDesignGraph (CN)
3: Compute logic expressions for PTS in $\mathcal{DG}$
4: $\mathcal{SG}.\mathcal{V} = PTS$, $\mathcal{SG}.\mathcal{E}(u) = \mathcal{SG}.\mathcal{V} \setminus \{u\}$
5: **for** $i = 1$ to *VN* **do**
6: t_i = CliqueSampling($\mathcal{SG}$)
7: **end for**
8: *return* $Tests = \{t_1, t_2, \ldots, t_{VN}\}$
9: **end procedure**

10: **procedure** $CliqueSampling(\mathcal{SG})$
11: constraints $cns = true$, P = $\mathcal{SG}.\mathcal{V}$
12: **while** P is not empty **do**
13: randomly pick and remove a vertex v from P
14: **if** satisfiabile($cns \wedge (v.le == v.rv)$) **then**
15: $cns = cns \wedge (v.le == v.rv)$
16: $P = P \cap \mathcal{SG}.\mathcal{E}(v)$
17: **else if** cns has only one constraint $u.le == u.rv$ **then**
18: $\mathcal{SG}.\mathcal{E}(v) = \mathcal{SG}.\mathcal{E}(v) \backslash \{u\}$
19: $\mathcal{SG}.\mathcal{E}(u) = \mathcal{SG}.\mathcal{E}(u) \backslash \{v\}$
20: **end if**
21: **end while**
22: Use SMT solver to solve cns and *return* the test
23: **end procedure**

We propose an on-the-fly technique (TARMAC) in Algorithm 1 that utilizes lazy construction of the satisfiability graph and random sampling of maximal satisfiable cliques. For each sampled maximal satisfiable clique, TARMAC generates one test vector for it. As shown in Algorithm 1, initially every vertex is connected to all the other vertices in line 4. The edge between two unsatisfiable vertices is removed in line 17-19, where cns contains only one constraint $u.le == u.rv$ and the new constraint $v.le == v.rv$ cannot be satisfiable together with cns. Lazy construction benefits large designs by generating test vectors as soon as possible. Clique sampling is done by maintaining two sets of variables: cns to keep track of constraints that are satisfiable (represents vertices that are already found in a satisfiable clique), and P to represent candidate vertices that may potentially be added to the clique. Initially, cns is true and P contains all the vertices. We first randomly select and remove a vertex v from candidate set P. If cns can be augmented by $v.le == v.rv$, we put it into cns and remove all vertices in P that are not connected to v (line 16). It is easy to verify that cns represents a maximal satisfiable clique when P is empty. Parameter *VN* is used to control how many times we should sample maximal satisfiable cliques, i.e., the number of generated test vectors.

V. Experiments

A. Experimental Setup

The framework TARMAC is implemented in C++ with Z3 [27] as our SMT solver. We conducted a wide variety of experiments on a machine with Intel Xeon E5-2698 CPU @2.20GHz to evaluate the performance of TARMAC compared to random test vectors and $N-$detect approach (MERO [3]). In this paper, we used the same benchmarks (ISCAS-85 [23] and ISCAS-89 [24]) from [3] to enable a fair comparison with MERO. We have also used two large designs (memory controller (MEM) from TrustHub [26] and MIPS processor from OpenCores [25]) to demonstrate the scalability of our approach. The experimental setup is shown in Figure 5. Similar to [3], we constructed valid trigger conditions from rare signals. We first ran a number of random simulations (100K for ISCAS and one million for MEM and MIPS) on the design and computed the probability of each signal. Rareness threshold is set to be 0.1 for ISCAS benchmarks and 0.005 for the other designs. For each benchmark, 1000 trigger conditions were randomly sampled and validated using ATPG. After sampling 1000 valid trigger conditions, each of them was individually integrated into the original design to construct a design under test (DUT). In other words, there are 1000 DUTs from each benchmark with one trigger condition for evaluation. We applied TARMAC (Algorithm 1) to generate the test set with the rare signals as potential trigger signals (PTS). Finally, we applied test sets to each DUT and collected trigger condition coverage. For all experiments, we fixed $N = 1000$ [3] for $N-$detect approaches.

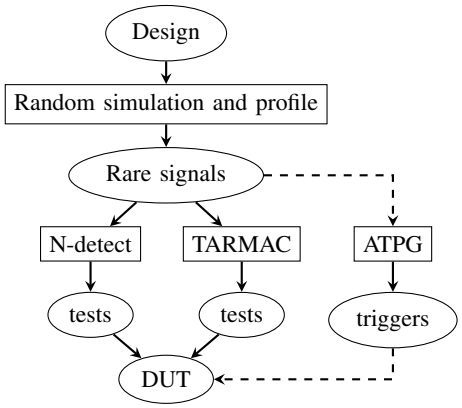


Fig. 5: Setup for evaluation of TARMAC compared to $N-$detect (MERO). Triggers are randomly sampled and validated by ATPG.

B. Performance Evaluation

We compared the trigger coverage of TARMAC to random approach and MERO, on a subset of ISCAS benchmarks and two large benchmarks (MEM and MIPS). To get a fair comparison of trigger coverage to MERO, we evaluated the trigger coverage with the same number of test vectors. The length of MERO test vectors cannot be controlled arbitrarily since it depends on $N-$detect and the number of initial random vectors R. Hence, we first ran MERO with 100K random vectors for ISCAS benchmarks as suggested in [3] and 1 million random vectors for MEM and MIPS. After MERO finished, we ran TARMAC to generate the same number of test vectors as MERO for each benchmark.

The trigger coverage comparison of TARMAC with random and MERO test vectors is shown in Table I. For ISCAS benchmarks, we can see that TARMAC can achieve huge

TABLE I: Comparison of TARMAC with random simulation and MERO for trigger activation coverage over 1000 randomly sampled 8-trigger conditions. The test length of TARMAC is the same as MERO.

Bench	Number of rare signals	Random		MERO [3]			TARMAC				
		Test Length	Cov. (%)	Test Length	Cov. (%)	Time (s)	Test Length	Cov. (%)	Impro. / Random	Impro. / MERO	Time (s)
c2670	43	100K	0.3	6820	38.2	1268	6820	100	333x	2.6x	257
c5315	164	100K	1.1	9232	50.6	4396	9232	98.8	89.8x	1.9x	682
c6288	169	100K	18.9	5044	76.6	596	5044	95.0	5.0x	1.2x	638
c7552	278	100K	0	14914	5.6	7871	14914	66.5	∞	11.9x	2185
s13207	604	100K	0	44534	1.9	15047	44534	94.4	∞	49.7x	5417
s15850	649	100K	0	39101	3	17000	39101	88.7	∞	29.6x	11337
s35932	1152	100K	100	4047	100	49616	4047	100	1x	1x	1947
MEM	1306	1M	0	28542	0	89747	28542	98.6	∞	∞	15753
MIPS	906	1M	0	25042	0.2	273807	25042	95.6	∞	472x	19458
avg.	**586**	**300K**	**13.4**	**19697**	**30.7**	**51039**	**19697**	**93.1**	**$>$ 107x**	**$>$71x**	**6408**

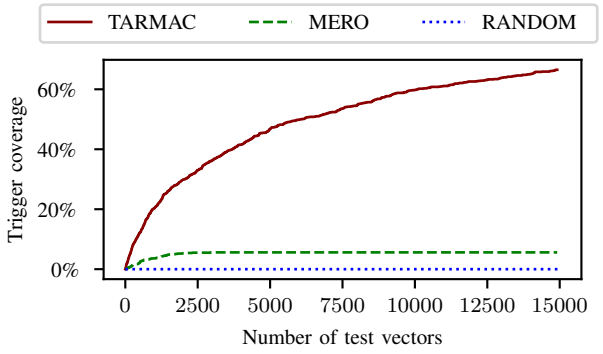
trigger coverage improvement (up to 49 times) over MERO with only around $1/4$ of time to generate the same number of test vectors. In large benchmarks such as c7552, s13207 and s15850, the performance of MERO is very poor, with less than 6% trigger coverage. TARMAC outperformed MERO in all benchmarks with more than 90% trigger coverage for most of them. With the same number of test vectors, TARMAC can cover the extremely hard-to-activate trigger conditions that are left after applying both random test vectors and MERO with significantly less effort.

For the two large benchmarks, since each trigger condition contains 8 rare signals with rareness threshold as 0.005, the probability of trigger conditions could be less than 10^{-18}. It is expected that 1 million random simulations could not achieve good coverage. The test vectors generated by MERO also achieved poor coverage, 0% in memory controller, and 0.2% in MIPS. On the other hand, TARMAC is able to cover majority of the trigger conditions efficiently. For example, TARMAC covered 95.6% of trigger conditions in MIPS using the same amount of test vectors as MERO, and finished in 6 hours. Note that the average test generation of TARMAC for one test vector is less than one second. This demonstrates that TARMAC is scalable for large designs, while MERO is not suitable.

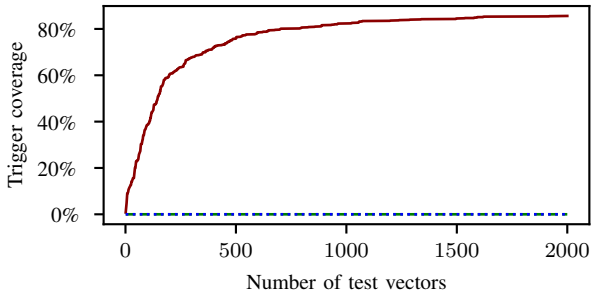
One observation is that the quality of MERO is partially dependent on the quality of random test vectors. For example, with 18.9% and 100% trigger activation coverage from random test vectors for c6288 and s35932, respectively, test vectors from MERO can cover 76.6% and 100%. However, for benchmarks such as c7552 and s31207, test vectors of MERO can only achieve 5.6% and 1.9%, respectively, corresponds to low coverage from random test vectors. The limited improvement from random test vectors to MERO is due to the simple flipping bits approach to search for good vectors in MERO.

C. Compactness and Efficiency

For better illustration of trigger coverage, we plotted the trigger coverage with respect to the number of test vectors for c7552 and MIPS in Figure 6. The x-axis represents the number of tests applied to DUTs, and the y-axis represents the percentage of activated trigger conditions. The efficiency in trigger coverage is the gradient of trigger coverage curves. TARMAC has much steeper slopes than MERO and the curves of random approach are almost flat. The results demonstrated



(a) c7552



(b) MIPS

Fig. 6: TARMAC can achieve significantly higher coverage using the same number of test vectors compared to random and MERO.

that TARMAC can cover more trigger conditions faster (with significantly less test vectors) than MERO and random approach.

The results suggest that each vector in TARMAC is able to activate more potential trigger conditions than MERO. Since each test vector can cover all the subgraphs of a satisfiable clique, if one test vector can activate more rare signals, it covers a larger clique and is likely to activate more potential trigger conditions. The number of rare signals satisfying their rare values (*rare signal hits*, for short) for each test vector is shown in Figure 7. The results show that the numbers of rare signal hits are significantly larger in TARMAC, which is consistent with the slope of coverage in Figure 6. From Algorithm 1, the number of rare signal hits is the same as the size of each sampled maximal satisfiable clique in

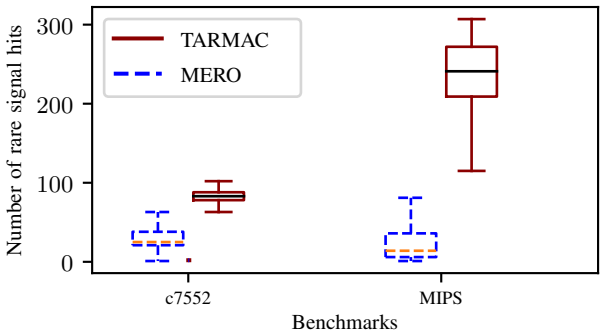


Fig. 7: The distribution of rare signal hits of each test vector generated by MERO and TARMAC.

TARMAC. While in MERO, the number of rare signal hits is the best number of hits after one round of bit flipping from a random test vector. The rare signal hits from MERO should be statistically lower than TARMAC. Moreover, the quality of test vectors in MERO is not guaranteed, since it partially depends on the initial random vectors. As a result, MERO has low rare signal hits (normally less than 50), which is significantly smaller than rare signal hits in TARMAC.

VI. Conclusion

Trigger activation is a fundamental challenge in detection of hardware Trojans. While prior efforts using statistical test generation are promising, they are neither scalable for large designs nor suitable for activating extremely rare trigger conditions in stealthy Trojans. In this paper, we introduced a new paradigm to solve trigger activation problem. This paper made the following important contributions. (1) Our approach is the first attempt in mapping the problem of test generation for trigger activation to the problem of covering maximal satisfiability cliques. (2) We proved that valid trigger conditions and satisfiability cliques are one-to-one mapping. (3) We presented efficient test generation algorithms to repeatedly sample maximal satisfiability cliques and generate a test vector for each of them. Our experimental results demonstrated that our approach is both scalable and effective in generating efficient test vectors for a wide variety of trigger conditions. Our approach outperforms the state-of-the-art techniques by several orders-of-magnitude in terms of trigger coverage, test length as well as test generation time. Our test generation algorithms can be utilized for activating extremely rare trigger conditions to fulfill diverse requirements such as improvement of functional (trigger) coverage as well as side-channel sensitivity.

Acknowledgments

This work was partially supported by grants from NSF (CCF-1908131) and SRC (2020-CT-2934).

References

[1] M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection," *IEEE Design & Test*, 2010.
[2] R. Karri, J. Rajendran, K. Rosenfeld, and M. Tehranipoor, "Trustworthy hardware: identifying and classifying hardware Trojans," *Computer*, 2010.
[3] R. S. Chakraborty, F. Wolff, S. Paul, C. Papachristou, and S. Bhunia, "MERO: A statistical approach for hardware Trojan detection," in *Cryptographic Hardware and Embedded Systems (CHES)*, 2009.
[4] A. Waksman, M. Suozzo, and S. Sethumadhavan, "FANCI: Identification of stealthy malicious logic using boolean functional analysis," in *ACM Conference on Computer and Communications Security (CCS)*, 2013.
[5] A. Bazzazi, M. T. M. Shalmani, and A. M. A. Hemmatyar, "Hardware trojan detection based on logical testing," *Journal of Electronic Testing*, 2017.
[6] M. A. Nourian, M. Fazeli, and D. Hely, "Hardware Trojan detection using an advised genetic algorithm based logic testing," *Journal of Electronic Testing*, 2018.
[7] F. Farahmandi and P. Mishra, "Automated Test Generation for Debugging Multiple Bugs in Arithmetic Circuits," *IEEE Transactions on Computers (TC)*, 2019.
[8] Y. Lyu and P. Mishra, "A Survey of Side-Channel Attacks on Caches and Countermeasures", *Journal of Hardware and Systems Security*, 2018.
[9] Y. Lyu and P. Mishra, "Efficient Test Generation for Trojan Detection using Side Channel Analysis", in *Design Automation and Test in Europe (DATE)*, 2019.
[10] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar, "Trojan detection using IC fingerprinting," in *IEEE Symposium on Security and Privacy (SP)*, 2007.
[11] Y. Jin and Y. Makris, "Hardware Trojan detection using path delay fingerprint," in *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2008.
[12] J. Aarestad, D. Acharyya, R. Rad, and J. Plusquellic, "Detecting trojans through leakage current analysis using multiple supply padI_{ddq}s," *IEEE Transactions on Information Forensics & Security (TIFS)*, 2010.
[13] G. Zarrinchian and M. S. Zamani, "Latch-based structure: A high resolution and self-reference technique for Trojan detection," *IEEE Transactions on Computers (TC)*, 2017.
[14] S. Wei and M. Potkonjak, "Scalable hardware Trojan diagnosis," in *IEEE Transactions on VLSI*, 2012.
[15] Y. Huang, S. Bhunia, and P. Mishra, "Scalable Test Generation for Trojan Detection using Side Channel Analysis," *IEEE Transactions on Information Forensics & Security (TIFS)*, 2018.
[16] I. Pomeranz and S. M. Reddy, "A measure of quality for n-detection test sets," *IEEE Transactions on Computers (TC)*, 2004.
[17] M. E. Amyeen, S. Venkataraman, A. Ojha, and S. Lee, "Evaluation of the quality of n-detect scan ATPG patterns on a processor," in *International Conference on Test (ITC)*, 2004.
[18] Y. Lyu, A. Ahmed, and P. Mishra, "Automated Activation of Multiple Targets in RTL Models using Concolic Testing", in *Design Automation and Test in Europe (DATE)*, 2019.
[19] Y. Lyu, X. Qin, M. Chen, and P. Mishra, "Directed Test Generation for Validation of Cache Coherence Protocols", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2018.
[20] D. A. Mathaikutty, S. Ahuja, A. Dingankar, and S. Shukla, "Model-driven test generation for system level validation," in *IEEE International High Level Design Validation and Test Workshop*, 2007.
[21] F. Wolff, C. Papachristou, S. Bhunia, and R. S. Chakraborty, "Towards Trojan-free trusted ICs: Problem analysis and detection scheme," in *Design Automation and Test in Europe (DATE)*, 2008.
[22] H. D. Foster, "Trends in functional verification: A 2014 industry study," in *Design Automation Conference (DAC)*, 2015.
[23] ISCAS85 combinational benchmark circuits, https://filebox.ece.vt.edu/~mhsiao/iscas85.html.
[24] ISCAS89 sequential benchmark circuits, https://filebox.ece.vt.edu/~mhsiao/iscas89.html.
[25] Opencores, https://www.opencores.org/.
[26] Trusthub, https://www.trust-hub.org/.
[27] Moura & Bjørner, "Z3: An efficient smt solver," *TACAS*, 2008.
[28] E. Tomita et al., "The worst-case time complexity for generating all maximal cliques and computational experiments," *Theor. Comput. Sci.*, 2006.