Static Probability Analysis Guided RTL Hardware Trojan Test Generation

Haoyi Wang
Tsinghua University
Beijing, China
why16@mails.tsinghua.edu.cn

Qiang Zhou
Tsinghua University
Beijing, China
zhouqiang@mail.tsinghua.edu.cn

Yici Cai
Tsinghua University
Beijing, China
caiyc@mail.tsinghua.edu.cn

ABSTRACT

Directed test generation is an effective method to detect potential hardware Trojan (HT) in RTL. While the existing works are able to activate hard-to-cover Trojans by covering security targets, the effectiveness and efficiency of identifying the targets to cover are ignored. We propose a static probability analysis method for identifying the hard-to-active data channel targets and generating the corresponding assertions for the HT test generation. Our method could generate test vectors to trigger Trojans from Trusthub, DeTrust, and OpenCores in 1 minute and get 104.33X time improvement on average compared with the existing method.

ACM Reference Format:
Haoyi Wang, Qiang Zhou, and Yici Cai. 2023. Static Probability Analysis Guided RTL Hardware Trojan Test Generation. In *28th Asia and South Pacific Design Automation Conference (ASPDAC '23), January 16–19, 2023, Tokyo, Japan.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3566097.3567921

1 INTRODUCTION

Hardware Trojans (HTs) are identified as one of the major concerns of Integrated Circuits (IC) designers. HTs may change the IC functionality, leak sensitive information or even cause a denial of service [1]. The globalization of the semiconductor industry and the widely used third-party intellectual property (3PIP) core have also raised the risk of hardware Trojans by rogue entities in the IC supply chain. Therefore, it is extremely important to find efficient approaches to detect hardware Trojans if they exist.

Recently, directed test generation frameworks [2, 3] are proved to be effective for activating hardware Trojans in complex register-transfer level (RTL) models. A typical flow of the directed test generation framework consists of 1) security target identification task and 2) directed test generation task, as shown in Figure 1. Firstly, the security targets are identified according to the security knowledge such as the threat model, and the corresponding security assertions are generated for test generation. Then, the directed test generation algorithm generates test patterns violating the assertions to activate the Trojan.

The existing directed test generation methods, Concolic testing engine [2], and C-level symbolic execution framework [3], mainly focus on the efficiency of the directed test generation task. However,

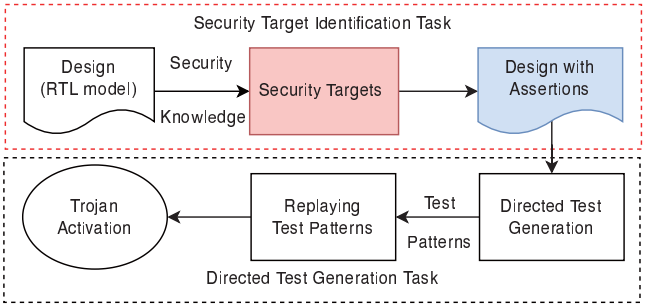

Figure 1: The overview of a directed test generation framework.

the effectiveness and efficiency of the security target identification task are ignored. The rare branch security targets in those works are identified by analyzing the random simulation traces, incurring large simulation time overhead. Also, while the rare branch security targets are able to be used for activating hard-to-cover Trojans by ensuring the branch coverage, many known HTs may evade the branch coverage detection [4].

As a result, we identify the data channel security target instead of the branch security target to improve the detection capability of the directed test generation. And those hard-to-active targets could be identified by a static probability analysis algorithm called Data Channel Active Probability Estimation algorithm (DCAPE). To the best of our knowledge, *our approach is the first attempt at RTL automatic security targets/assertions generation without the help of simulation knowledge.*

There are three major challenges in applying the static probability analysis: 1) the static analysis of the circuit sequential behaviors, 2) estimating the data channel active probability efficiently and 3) the security target identification and assertion generation. The DCAPE consists of three steps to deal with those challenges respectively: the register Probability Distribution (PD) estimation, the register PD propagation, and the security assertion generation.

In order to analyze the sequential circuit statically without the help of simulation information, the circuit sequential behaviors are captured and represented in the register PDs. The register PDs are modeled by a system of nonlinear polynomial equations generated from the RTL code, solved by the Levenberg-Marquardt algorithm. Then, in the register PD propagation step, the register PDs are propagated on the control and data flow graph (CDFG) according to the operator characteristic to estimate each data channel's active probability. Finally, those data channels with a low but not zero active probability are identified as data channel security targets

ASPDAC '23, January 16–19, 2023, Tokyo, Japan

ACM ISBN 978-1-4503-9783-4/23/01.
https://doi.org/10.1145/3566097.3567921

and the security assertions are generated by negating each security target's symbolic active conditions.

To summarize, the contributions of this paper are as follows:

(1) Our proposed approach is the first attempt to apply the RTL static probability analysis to generate security targets and assertions without the help of simulation knowledge. The security assertions are used for directed test generation.
(2) We utilize the data channel security target to improve the detection capability of the directed test generation. A static probability analysis algorithm DCAPE is proposed to identify the security targets and generate security assertions efficiently.
(3) The circuit sequential behaviors are captured and represented in the register probability distribution in the static analysis. The register probability distributions are modeled by a nonlinear polynomial equations system, solved by the Levenberg-Marquardt algorithm efficiently.
(4) The experiments demonstrate that the proposed method can detect more Trojans from Trust-hub, DeTrust, and OpenCores in 1 minute and get 104.33X time improvement on average compared with the existing method.

The remainder of this paper is organized as follows. Section 2 provides the preliminary. Section 3 introduces the threat model. The static probability analysis algorithm is introduced in Section 4. Section 5 provides the experimental results. The conclusion and discussion are summarized in Section 6.

2 PRELIMINARY

2.1 Hardware Trojan Detection

There are several existing efforts to address the problem of detecting hardware Trojans with respect to the RTL security target identification.

Although **formal verification** [5] has proven effective to detect HTs violating the security properties, the security properties are time-consuming and error-prone to write, which is a challenging task for designers.

The existing **information flow tracking** methods [6] are mostly based on various levels of information flow tracking (IFT). The IFT methods focus on the information leakage HT and they could not deal with the functional HT threats.

Approaches based on **statistical analysis** like [7–9] require a full simulation of the design. They suffer from scalability for large designs and may not deal with the DeTrust Trojans.

Machine learning based method [10] identifies the presence of HT fast with high precision and recall rate. However, it's hard to locate the Trojan and generate the security assertions.

The **functional testing** methods consist of applying test stimuli to ICs under evaluation to trigger the HTs and comparing the responses with the expected specification. Recently, directed test generation frameworks [2, 3] are proved to be effective for activating Trojans. However, the rare branch security target identification in those methods is time-consuming and there are many known HTs evading the detection [4]. The detection capability of directed test generation methods is dominated by the quality of security targets. There is an urgent need for the effectiveness and efficiency of the security targets identification method.

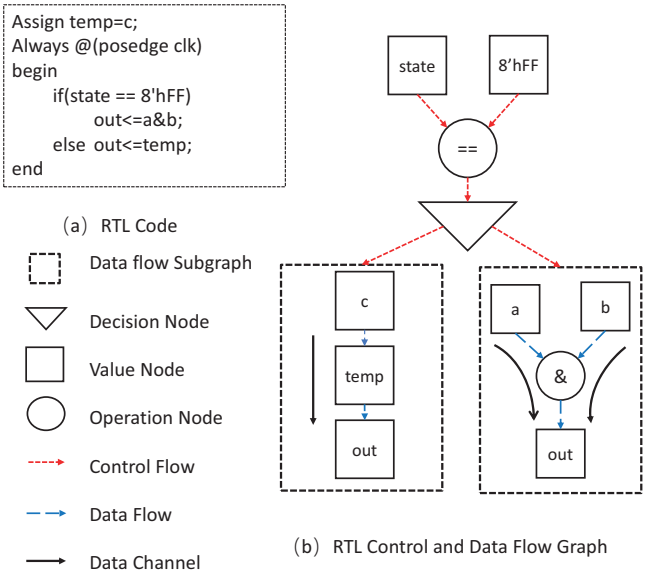

Figure 2: The example of the RTL Control and Data Flow Graph.

2.2 Control and Data Flow Graph

A RTL design is composed of control flows and data flows. The control flows determine which data flow will be executed. The control and data flow graph (CDFG) is constituted by control flow and data flow. Because the CDFG contains all RTL control and data propagation information and can be easily constructed from the RTL code, we utilize it to analyze the RTL design code.

In CDFG, both control flow and data flow are represented in the directed acyclic graphs (DAG) and can be formally expressed as $G(V, E)$. V denotes the set of nodes and there are two types of nodes in the data flow: value node and operation node. In the control flow, there are two more node types: decision node and data flow subgraph node.

- **Operation node** represents each operator in RTL code.
- **Value node** is corresponding to reading/writing the value of the constant value, the wire, primary output, primary input, or the register.
- **Decision node** determines which data flow will be executed.
- **Data flow supernode** represents a data flow subgraph that is controlled by the control flow.

$E \subseteq V \times V$ is the set of edges representing the transfer of either data or control from one node to another. In our work, each one-time-cycle circuit combinational behavior among registers is represented in the same CDFG. Figure 2(a) and Figure 2(b) show an example of the RTL code and corresponding CDFG.

3 THREAT MODEL

In our threat model, the RTL hardware design can be covertly modified by HTs inserted into the RTL code, inserted by the rogue adversaries in the design team, or existing in third-party intellectual property (3PIP) cores to be integrated into the system. We assume that the RTL design under detection is delivered to the

defender as the RTL code in VHDL or Verilog HDL formats without trustworthiness.

The Trojan is triggered to change the IC functionality, leak sensitive information or even cause a denial of service. The Trojan triggers developed so far come in two forms: combinational and sequential. To escape detection during different steps of the verification procedure, Trojans are designed to be hard-to-trigger during normal execution and activated under rare conditions. Otherwise, traditional simulation-based techniques using random or constrained-random tests may detect them, and the attacker's attempt would fail. Our methods are designed to detect those hard-to-activate RTL Trojans.

4 STATIC PROBABILITY ANALYSIS

Our work proposes a static probability analysis method called Data Channel Active Probability Estimation algorithm (DCAPE) to generate the data channel security targets and the corresponding assertions for directed test generation. The overview of DCAPE is shown in Figure 3. There are three major steps in DCAPE. 1) Register PD estimation: estimating the register PDs capturing the circuit sequential behaviors; 2) Register PD propagation: propagating the register PDs on CDFG to estimate each data channel's active probability; 3) Security assertion generation. The data channel security target is first introduced and then those steps of DCAPE are introduced.

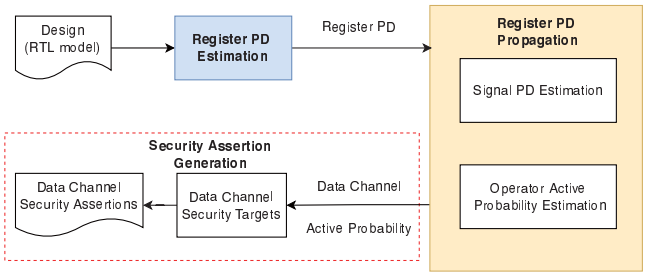

Figure 3: Data Channel Active Probability Estimation Algorithm (DCAPE) Overview.

4.1 Data Channel Security Target

In the previous study [2, 3], the branch security targets are used for directed test generation to trigger the Trojan parts in the rare branch. However, it's not hard to eliminate the Trojan rare branch by implying the control logic into the data flow without the logic function changing. Instead, we identify the hard-to-active data channel as our security target in which the data channel behavior is taken into consideration.

Data Channel: The data channel $x \longmapsto y$ is a path in the CDFG data flow subgraph from a register or primary input value node x to another register or primary output value node y. For example, in Figure 2, there are three data channels $c \rightarrow temp \rightarrow out$, $a \rightarrow \& \rightarrow out$, and $b \rightarrow \& \rightarrow out$.

Channel Behavior Function: In the data channel $x \longmapsto y$, there is a sequence of operators through which the value in x could produce the y value. The data channel behavior can be considered as a *Channel behavior function* $y = f(x, Z)$. $Z = (z_1, z_2, \ldots, z_n)$ is the vectors of input z_i in the data channel except x. The boolean function specifies the value of node y after one time cycle given the specific value of x and Z.

Data Channel Activation: For a data channel $x \longmapsto y$ and its channel behavior function $y = f(x, Z)$, given the specific value of function input x and Z, if the change of x results in the change of y when the Z remains unchanged, the data channel is activated.

The nearly inactive data channels are hard to be sensitized during regular testing and likely to be a part of hard-to-activate HT. If a data channel is inactive, the input x value and some operators in that data channel are unnecessary to produce the correct output y. There are unused operations and behaviors hidden in the inactive channel. Those data channels with a low but not zero active probability are identified as **Data Channel Security Target**s for hard-to-activate HTs directed test generation.

4.2 Register Probability Distribution Estimation

To identify the data channel security target, the register PDs are estimated first to produce the necessary data for the data channel active probability estimation in the register PD propagation.

In static analysis, the sequential circuit behaviors are hard to be analyzed without the help of simulation information. In our work, the circuit sequential behaviors are captured and represented in the register PDs.

The register PD estimation problem has been well studied in RTL power estimation and RTL design for test [11]. With the assumption that each register probability is independent, each register probability P_i [1] is given by a function of all register probabilities P_N and input probabilities I_M, $f_i(P_1, P_2, \ldots, P_N, I_1, I_2 \ldots, I_M)$. The function is determined by the input cone of the register and is calculated by parsing and analyzing the RTL code. Finally, a set of polynomial equations (1) is constructed to estimate register probabilities. Most register probability estimation deviations are smaller than 0.1 compared with the random simulation result as the experiment results listed in [11].

$$\begin{aligned} P_1 &= f_1(P_1, P_2, \ldots, P_N, I_1, I_2, \ldots, I_M) \\ P_2 &= f_1(P_1, P_2, \ldots, P_N, I_1, I_2, \ldots, I_M)) \\ &\vdots \\ P_N &= f_1(P_1, P_2, \ldots, P_N, I_1, I_2, \ldots, I_M) \end{aligned} \tag{1}$$

```
Always @(posedge clk)
Begin
    a <= ~a;
end
```

(a) Picard-Peano Counter Example

```
Always @(posedge clk)
begin
    a <= a|b;
    b <= a|b;
end
```

(b) Newton Counter Example

Figure 4: Picard-Peano and Newton Method Counter Example.

The set of polynomial equations (1) is suggested to be solved using the Picard-Peano method or Newton method in [11]. However,

[1]The signal probability P_i refers to the probability of the signal being 1.

the Picard-Peano method suffers from the problem of not being convergent. In Figure 4(a) example, the equation $P_1 = 1 - P_1$ should be solved and the Picard-Peano method fails to converge when the initial P_1 is not 0.5. Furthermore, the Newton method may not be valid when the Jacobian matrix is invertible. In Figure 4(b) example, the equations system $P_1 = P_1 + P_2 - P_1 * P_2$ and $P_2 = P_1 + P_2 - P_1 * P_2$ can be derived from the RTL code. The Jacobian matrix isn't invertible with any P_1 and P_2. Instead of utilizing the Picard-Peano method or Newton method to solve the system of polynomial equations, we transfer the problem into a least-squares optimization problem as shown in Equation (2).

$$\begin{aligned} &\min \sum_{i=1}^{N} (P_i - f_i(P_1, P_2, \ldots, P_N))^2 \\ &s.t.\ \ 0 \le P_i \le 1 \end{aligned} \tag{2}$$

The least-squares optimization problem could be solved efficiently by the Levenberg-Marquardt (LM) algorithm. Since all probability functions f_i are polynomial functions, the Jacobian matrix must exist and the LM algorithm is available for all circuits.

4.3 Register Probability Distribution Propagation

The DCAPE utilizes the register PDs produced by the register PD estimation step to estimate each data channel's active probability through a register PD propagation on CDFG.

The activation of data channel $x \longmapsto y$ should satisfy two conditions.

- **Condition I:** The control flow is in the appropriate processes in which the data flow could execute.
- **Condition II:** The input vector Z of the corresponding boolean function $y = f(x, Z)$ should follow a special pattern so that changing x causes a change in y.

Condition I makes sure that the data channels in the data flow could be executed and Condition II makes sure that when the data flow executes, the change of value x could affect the output y value and the data channel is active.

According to the definition of data channel activation, when these two conditions are met, x and all the operators in the path cannot be reduced and the data channel is active. Taking the data flow $a \rightarrow \& \rightarrow out$ in Figure 2(b) as an example, if Condition I $state == 8'hFF$ satisfies and Condition II $b! = 0$ satisfies, then the change of a value will result in a different out value and the data channel is active.

The relationship among the data channel active probability $P_{channel}$, the Condition I satisfied probability C_1, and the Condition II satisfied probability C_2 is

$$P_{channel} = C_1 \times C_2 \tag{3}$$

So, the data channel active probability estimation can be divided into estimating C_1 and C_2 separately.

4.3.1 Condition I Probability Estimation. The control flow in CDFG determines which data flow should be executed. A data flow is executed if and only if the corresponding control flows output a True value. Estimating the Condition I probability C_1 is to estimate the signal PDs in the control flow.

For all operators in CDFG, the operators' truth table is fixed and the output signal PD can be calculated from the operators' input signal PDs. The *PD calculation rules* of many RTL operators, such as n-bit adders, n-bit comparators, n-bit multipliers, and so on, are thoroughly discussed in [12]. According to those rules, the ***signal PD estimation algorithm*** traverses the entire CDFG to calculate all nodes' signal PDs. As shown in Algorithm 1, all nodes in CDFG are first sorted by the topological order in line 1. Then all nodes in CDFG are visited in topological order and the signal PD of each node is calculated when visited in line 3 according to the parent nodes' signal PDs and the PD calculation rules. The register node PDs are provided by the register probability distribution estimation algorithm introduced in Section 4.2. All signal PDs including those control flow output signal PDs in the decision node are estimated, so that each data channel's Condition I probability C_1 is determined.

Algorithm 1 Signal PD Estimation Algorithm

Input: The CDFG Graph $G = (V, E)$, the register probability vector R;
Output: The CDFG Graph $G = (V', E')$ with signal PD;
1: TopologicalSort(G);
2: **for** each node v in $G = (V, E)$ in topological order **do**
3: $v.p = calByPDRules(v.parent.p)$
4: **return** $G = (V', E')$;

4.3.2 Condition II Probability Estimation. Condition II probability C_2, also has a relationship with the signal PDs in the data channel.

Taking the data channel $a \rightarrow \& \rightarrow out$ in Figure 2(b) as an example, the change of a can result in the change of out if and only if another input of & node $b! = 0$. In order to change the output value, the change in the data channel input must be propagated through the entire channel. If the probability of propagating through the operation j in the channel is the operator active probability p_j, the Condition II probability C_2 is estimated by Equation 4:

$$C_2 = \prod p_j \tag{4}$$

Considering that each operator's truth table is fixed, the *propagating rule* of each operator can be derived and p_j is determined by the input signal PD. Boubezari *et al.* [12] defines the observability of the operator input I on the operator output O as the probability of a signal change on I resulting in a signal change on the output O which is the same with the active probability of the operators. The so-called observability formula of each RTL operator is used as the propagation rule to estimate the propagating probability p_j according to the operators' input signal PDs.

Algorithm 2 Operator Active Probability Estimation Algorithm

Input: The CDFG Graph $G = (V', E')$ with signal PD;
Output: Data channel set D with operator active probability C_2
1: **for** each data channel dc in $G = (V, E)$ **do**
2: **for** each operator node op_j in dc **do**
3: $p_j = calByPropagatingRules(op.parent.p)$;
4: $dc.C_2 = \prod p_j$;
5: Add dc to data channel set D;
6: **return** D;

Table 1: THE DETECTION CAPABILITY OF DCAPE ON THE BENCHMARKS FROM TRUST-HUB DETRUST AND OPENCORE

Benchmark	Lines of Code[a]	Unrolled Cycle	Rare Branch Security Target Identification[b]					DCAPE [c]					Overall Time Improvement
			Detected ?	#Covered	Ass. (Sec)	EBMC (Sec)	Memory (MB)	Detected ?	#Covered	Ass. (Sec)	EBMC (Sec)	Memory (MB)	
Wb_conmax-T200	70k	10	✓	20	1274.7	8.38	2141.6	✓	21	27.61	9.09	2219.2	34.96X
Wb_conmax-T300		10	✓	17		9.42	2453.5	✓	18	27.35	9.91	2531.1	34.47X
Wb_conmax-T200D[d]		10	✓	20		8.42	2120.3	✓	29	32.11	8.99	2806.7	31.22X
AES-T1000	358k	10	✓	2	4510.3	0.04	18.0	✓	2	18.92	0.04	18.0	237.85X
AES-T1100		10	✓	5		0.06	23.3	✓	9	18.49	0.07	26.4	243.00X
AES-T1300		10	✓	9		0.04	18.7	✓	9	19.61	0.04	18.7	229.53X
AES-T2000		10	✓	5		0.02	8.48	✓	9	29.78	0.02	8.8	151.32X
AES-T700D [d]		10	✓	1		0.04	21.81	✓	9	19.31	0.09	39.7	232.44X
RS232-T800	0.5k	300	X	0	7.3	-	-	✓	17	0.04	1.2	394.2	5.88X
RS232-T900		300	✓	9		0.64	192.4	✓	10	0.04	0.66	196.3	11.25X
RS232-T800D [d]		300	X	0		-	-	✓	3	0.036	0.49	130.0	13.84X
OR1200_ctrlD[d]	1k	10	✓	1	28.8	0.01	8.0	✓	9	1.078	0.02	17.5	26.25X

[a] After hierarchy flattening.

[b] Rare branches are selected by simulating the design with random inputs for one million cycles.

[c] The threshold is chosen as 10^{-6}.

[d] The benchmark is updated from the genuine RTL designs with the strategy of DeTrust.

Therefore, the ***operator active probability estimation algorithm*** could estimate the active probability p_j of each operator j and the Condition II probability in a single CDFG traversal. As shown in Algorithm 2, for each data channel found in a CDFG, the operator node active probability in the data channel is calculated by the parent signal PDs and the propagating rules in line 3. Finally, the Condition II probability C_2 of each dc is estimated in line 4.

4.4 Data Channel Security Assertion Generation

In the last phase of DCAPE, the data channel security targets are identified and the corresponding security assertions are generated for the directed test generation.

The security targets are those data channels with active probability less than the threshold. And the security assertions are the negated active conditions of security targets. The assertions could be used for the directed test generation to generate test vectors to activate the security targets for the Trojan behaviors observation.

The security assertions generation algorithm is similar to the register PD propagation algorithm. The data channel active probability is estimated through the concrete register PD propagation in section 4.3. While in the security assertion generation algorithm, the concrete register PDs are replaced with the symbolic register probability. Then the negated symbolic active conditions of each data channel are generated as security assertions.

5 EXPERIMENT

5.1 Experiment Setting

To verify the effectiveness and efficiency of the static probability analysis method DCAPE, we developed a Verilog RTL parser based on Yosys [13]. The algorithm could also be applied to VHDL or any other common HDL. The EBMC model checker [14] is used to perform directed test generation and other directed test generation engines, like Concolic testing [2] and SymbA [3] are also available. And the systems of polynomial equations are solved by an open-source Levenberg-Marquardt algorithm toolkit, Levmar [15].

The off-the-shelf benchmarks are mainly from the TrustHub [16], a popular benchmark suite for the work on hardware Trojans. The benchmarks from [2, 3] are selected for the purpose of comparison and cover a wide range of hardware Trojans. The scale of benchmark is up to 358k lines RTL code after hierarchy flattening. The scale of each benchmark is listed in the column 2 of the Table 1. We also apply the algorithm to the Trojan-infected designs proposed by DeTrust [17] and update the genuine RTL designs from OpenCores [18] with the strategy of DeTrust. The code is implemented in C++ language and executed on a Ubuntu 16.04LTS server with Intel(R) Core(TM) i9-9900k CPU@3.60GHZ and 64GB RAM.

5.2 Results

The detection capability results of both rare branch security targets identification method [2, 3] and DCAPE are reported in Table 1. In rare branch security target identification method, rare branches are selected by simulating the design with random inputs for one million cycles as suggested in [2, 3]. The Icarus Verilog [19] is chosen as the simulator in the experiment. In DCAPE, the data channels with active probability less than 10^{-6} are flagged as security targets. All benchmarks are sliced based on the Cone of Influence (COI) of the targets and only target-related codes are kept for EBMC test generation. And the benchmarks are unrolled in a specific cycle listed in column 3. Five experiment indexes are reported to indicate the detection capability and efficiency of methods: whether the Trojan is detected, the number of targets covered by the generated

tests, the security assertions generation time, the time and the memory cost in EBMC test generation. They are listed in columns 4-8 and columns 9-13 respectively.

In terms of **Detection Capability**, the DCAPE could generate effective security targets and assertions for directed test generation to trigger all Trojans from Trust-hub, DeTrust, and OpenCores. However, the rare branch security target may not identify the Trojan evading the branch coverage detection like RS232-T800 Trojan. The data channel security target takes both control flow logic and data channel behaviors into consideration. As a result, the capability of detection is improved.

In terms of **Detection Efficiency**, the DCAPE based directed test generation could generate tests from Trust-hub, DeTrust, and OpenCores in 1 minute. Despite DCAPE generating more data channel targets and resulting in a slight time and memory overhead in EBMC test generation, the static analysis method saves lots of time in the security target identification task. The overall time gets 104.33X time improvement on average compared with the rare branch security target identification.

5.3 Comprasion with the existing works

In Table 2, we summarize and compare the detection capability of DCAPE with existing HT detection methods that are promising for the security target identification task in the directed test generation framework. The Trojan benchmarks are distorted from the different original circuits with representative trigger and payload characteristics.

The existing HT detection methods could not address all HT threats. The information flow tracking (IFT) could only identify the information leakage HT targets. Detrust Trojan may evade statistical methods (SM) identification. And the existing functional test (FT) generation methods are hard to cover branch coverage uncovered HTs. Also, the statistical methods and the existing functional test generation suffer from the large random simulation time overhead. Only our DCAPE could identify security targets, generate assertions for directed test generation and finally generate test vectors to trigger all Trojans effectively and efficiently.

Table 2: HT DETECTION CAPABILITY COMPARISONS

Benchmark	Trigger Payload	Ours	IFT [6]	SM [7–9]	FT [2, 3]
RS232-T800	Combinational Deny Service	✓	X	✓	X
AES-T1100	Sequential Leak Data	✓	✓	✓	✓
Wb_conmax-T200	Sequential Change Function	✓	X	✓	✓
OR1200_ctrlD	DeTrust Change Function	✓	X	X	X

6 CONCLUSION AND DISCUSSION

In this paper, we propose a static probability analysis method DCAPE, identifying the hard-to-active data channel targets and generating the corresponding assertions for the HT test generation. To the best of our knowledge, our approach is the first attempt at automatic RTL security targets/assertions generation without the help of simulation knowledge. The DCAPE consists of three major steps. First of all, in order to analyze the sequential circuit statically without the help of simulation information, the circuit sequential behaviors are captured and represented in the register probability distributions. The register probability distribution is modeled by a nonlinear polynomial equations system and solved by the Levenberg-Marquardt algorithm. Then the register probability distribution is propagated on the CDFG to estimate each data channel's active probability. Finally, those data channels with a low but not zero active probability are identified as data channel security targets and the security assertions are generated by negating each security target's symbolic active conditions. DCAPE based directed test generation could generate test vectors to trigger Trojans from Trust-hub, DeTrust, and OpenCores in 1 minute and get 104.33X time improvement on average compared with the existing method.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (NSFC) under Grant No. 61834002.

REFERENCES

[1] K. Xiao, D. Forte, Y. Jin, R. Karri, S. Bhunia, and M. Tehranipoor, "Hardware trojans: Lessons learned after one decade of research," *Acm Transactions on Design Automation of Electronic Systems*, vol. 22, no. 1, p. 6, 2016.
[2] A. Ahmed, F. Farahmandi, Y. Iskander, and P. Mishra, "Scalable hardware trojan activation by interleaving concrete simulation and symbolic execution," in *2018 IEEE International Test Conference (ITC)*, 2018.
[3] A. Vafaei, N. Hooten, M. Tehranipoor, and F. Farahmandi, "Symba: Symbolic execution at c-level for hardware trojan activation," in *2021 IEEE International Test Conference (ITC)*, 2021, pp. 223–232.
[4] J. Zhang and Q. Xu, "On hardware trojan design and implementation at register-transfer level," in *2013 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, 2013, pp. 107–112.
[5] J. Rajendran, V. Vedula, and R. Karri, "Formal security verification of third party intellectual property cores for information leakage," in *International Conference on Vlsi Design & International Conference on Embedded Systems*, 2016.
[6] A. Ardeshiricham, W. Hu, J. Marxen, and R. Kastner, "Register transfer level information flow tracking for provably secure hardware design," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, 2017, pp. 1691–1696.
[7] A. Waksman, M. Suozzo, and S. Sethumadhavan, "FANCI: identification of stealthy malicious logic using boolean functional analysis," in *ACM SIGSAC Conference on Computer and Communications Security, CCS*, 2013, pp. 697–708.
[8] J. Zhang, F. Yuan, L. Wei, Y. Liu, and Q. Xu, "Veritrust: Verification for hardware trust," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 7, pp. 1148–1161, 2015.
[9] C. Wang, Y. Cai, Q. Zhou, and H. Wang, "Asax: automatic security assertion extraction for detecting hardware trojans," in *Asia and South Pacific Design Automation Conference*, 2018, pp. 84–89.
[10] R. Yasaei, S.-Y. Yu, and M. A. Al Faruque, "Gnn4tj: Graph neural networks for hardware trojan detection at register transfer level," in *2021 Design, Automation and Test in Europe Conference and Exhibition (DATE)*, 2021, pp. 1504–1509.
[11] J. M. Fernandes, M. B. Santos, A. L. Oliveira, and J. C. Teixeira, "A probabilistic method for the computation of testability of rtl constructs," in *Proceedings Design, Automation and Test in Europe Conference and Exhibition*, vol. 1, 2004, pp. 176–181 Vol.1.
[12] S. Boubezari, E. Cerny, B. Kaminska, and B. Nadeau-Dostie, "Testability analysis and test-point insertion in rtl vhdl specifications for scan-based bist," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, no. 9, pp. 1327–1340, 1999.
[13] C. wolf. yosys open synthesis suite. http://www.clifford.at/yosys/.
[14] R. Mukherjee, D. Kroening, and T. Melham, "Hardware verification using software analyzers," in *Vlsi*, 2015.
[15] M. Lourakis, "levmar: Levenberg-marquardt nonlinear least squares algorithms in C/C++," http://www.ics.forth.gr/ lourakis/levmar/+.
[16] Trust-hub. http://www.trust-hub.org.
[17] Z. Jie, Y. Feng, and X. Qiang, "Detrust: Defeating hardware trust verification with stealthy implicitly-triggered hardware trojans," in *Acm Sigsac Conference on Computer and Communications Security*, 2014.
[18] Opencores. http://www.opencores.org/.
[19] S. Williams. Icarus verilog. http://iverilog.icarus.com.