Provably Complete Hardware Trojan Detection Using Test Point Insertion

Sheng Wei† Kai Li‡ Farinaz Koushanfar‡ Miodrag Potkonjak†

†Computer Science Department
University of California, Los Angeles
Los Angeles, CA 90095
{shengwei, miodrag}@cs.ucla.edu

‡Department of Electrical and Computer Engineering
Rice University
Houston, TX 77005
{kai.li, farinaz}@rice.edu

ABSTRACT

This paper proposes a novel minimal test point insertion methodology that provisions a provably complete detection of hardware Trojans by noninvasive timing characterization. The objective of test point insertion is to break the reconvergent paths so that target routes for Trojan delay testing are specifically observed. We create a satisfiability-based input vector selection for sensitizing and characterizing each single timing path. Evaluations on benchmark circuits demonstrate that the test point-based Trojan detection can cover all circuit locations and can detect Trojans accurately with less than 5% performance overhead.

1. INTRODUCTION

The excessive cost of fabricating ICs in miniature-scale technologies has resulted in the dominance of a contract-foundry semiconductor business model. In this model, the designers, the Intellectual Property (IP) providers, and the manufacturing plants are different entities, exposing the ICs to multiple threats including piracy, hardware Trojan (malware) insertion, and unauthorized IP usage [7]. The ICs are the kernels of all digital computing and communications for the modern business, military, and government affairs and their vulnerabilities could permeate to all software and applications. Thus, providing trust in presence of an untrusted foundry and an unverified supply chain is a major problem.

Hardware Trojans (HTs), or simply Trojans, are unwanted components added to integrated circuits (ICs) during the manufacturing process to maliciously alter, tamper, or enable later monitoring, spying, or other exploits [18, 28, 29, 11]. Trojans are often behaviorally dormant and are rarely functionally activated (triggered) as desired. Development of methods for noninvasive chip testing for Trojan detection has been identified as an important research topic. Several challenges need to be addressed for dealing with the excessive complexity and scale of modern chips with inherently nontransparent internal parts. The growing process variations (PV) [5] cause random fluctuations in chip structural properties that are hard to distinguish from the Trojan impact. The large degree of freedom in the form and placement of possible exploits further complicates Trojan detection. Traditional IC scanning and testing methods have a limited capability in fully characterizing the chips. Destructive tests and reverse-engineering are costly and slow.

A number of efficient methods for Trojan detection based on structural side-channel timing tests [12, 8], dynamic or leakage power tests [16, 24], or a combination of them [11, 20, 23] were recently proposed. Comprehensive surveys can be found in [18]. A number of works have focused on Trojan detection based on gate-level characterization [16, 20, 11]: after applying a set of input test vectors and measuring the side-channel, the measurements are linearly translated to gate-level characteristics. Leakage power is the most often monitored side-channel for gate-level methods, because any unexpected malicious circuit component would result in a systematic bias in the total leakage consumption [16]. Thus, leakage tests could provide a full circuit coverage for identifying Trojans. However, existing power-based approaches require creating and solving huge sizes of power equations that cover essential gates in an IC, making it difficult for these methods to scale to modern ICs with millions of gates. Furthermore, leakage has an exponential dependence on PV and noise, making the detection hard.

Delay(timing)-based detection has a higher resolution and lower noise when compared to the current testing because of the linear dependence of delays on PV and the limited number of components on each path under test. However, an unresolved difficultly of timing measurements is the inability of individually sensitizing and characterizing each component using the test vectors. This is because of the existence of parallel routes that reconverge to a single point which make it difficult to map the measured path delay to a specific path for the consideration of Trojans. As shown in a small example in Figure 1, even though we can measure the delay between input x and output y, we are unable to determine whether the measured delay is for path 1 or path 2. The presence of PV would further complicate the case, since the delay of path 1 may be smaller than that of path 2 on one chip but could be larger on another.

In this paper, we create a gate-level delay characterization approach that enables full coverage of all circuit locations for Trojan detection. In particular, we introduce a test point (TP) insertion methodology that separates the paral-

IEEE/ACM International Conference on Computer-Aided Design (ICCAD) 2012, November 5-8, 2012, San Jose, California, USA

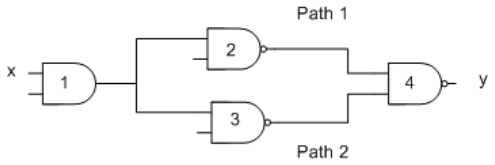


Figure 1: Example of reconvergent path.

lel paths and enables their observability via delay measurements. Next, we develop an input vector selection scheme for each single path by transforming the problem to a Boolean satisfiability (SAT) problem. Leveraging SAT problem formulation, we obtain gate-level delay properties for all the gates in the circuit. Trojan detection is performed by introducing and characterizing a Trojan indicator variable. To summarize, our innovations and contributions include:

- a consistency-based Trojan detection using delay characterization;
- a provably complete test points insertion to break the reconvergences and identify the measured paths;
- a SAT-based input vector selection for characterizing each single path; and
- automated integrability of the methods within the standard IC design and test flow and proof-of-concept evaluations on combinational and sequential benchmarks.

2. BACKGROUND

2.1 Hardware Trojan Attack

There have been many types of malicious modifications to ICs discussed in the hardware security community. In this paper, we consider the post-silicon IC attacks that are during or after the manufacturing process.

The HT threat model is to add to the circuit additional components (e.g. gates) that are seldom activated but capable of conducting malicious attack (e.g. leak confidential information or consume huge amount of energy) when a certain (rare) condition is satisfied. In most cases, the attackers tend to design and place the HT in a wise manner so that the alteration caused by the HT is easily hidden under PV or other manufacturing factors. In this way, the HTs are unlikely to be detected by the side channel based approaches, simply because the variations in side channels are negligible. For example, the attacker may intentionally use a small-size gate as the HT to minimize the exposed leakage power. In this paper, we focus on timing-based HT attack, where an attacker hides the embedded malicious circuitry under paths that cannot be measured for delay using standard methods.

2.2 Timing Models and Test Point Insertion

The delay of a single logic gate can be expressed as $d = gh + p$, where g and h are logical effort and electrical effort, respectively; and p is parasitic delay. In particular, We use the delay model in [15] that connects the gate delay to its sizing and operating voltages:

$$Delay = \frac{k_{tp} \cdot k_{fit} \cdot L^2}{2 \cdot n \cdot \mu \cdot \phi_t^2} \cdot \frac{V_{dd}}{(ln(e^{\frac{(1+\sigma)V_{dd}-V_{th}}{2 \cdot n \cdot \phi_t}} + 1))^2} \cdot \frac{\gamma_i \cdot W_i + W_{i+1}}{W_i} \quad (1)$$

where L is effective channel length, V_{th} is threshold voltage, W is gate width, V_{dd} is supply voltage, n is substreshold slope, μ is mobility, C_{ox} is oxide capacitance, D is clock period, ϕ_t is thermal voltage $\phi_t = kT/q$, σ is drain induced barrier lowering (DIBL) factor, subscripts i and $i+1$ represent the driver and load gates, respectively, γ is the ratio of gate parasitic to input capacitance, and k_{tp} and k_{fit} are fitting parameters. Note that we adapt the quad-tree model for the timing PV [5].

To measure a path's timing, we use the known delay-fault testing techniques [14]. The test points inserted by our method will be only used during the test mode. At each test point, we insert a test flip-flop (FF). During the test mode, the inserted FFs are enabled to capture the test point value. In other modes, the added FFs are disabled. The performance overhead of our method on the normal operation is only due to the loading of the disabled test point FFs on the original circuit.

2.3 Gate-Level Characterization (GLC)

In GLC [20, 21, 23, 25], the measured side-channel value is decomposed to its constituent components. For example, let us assume that a test vector sensitizes a path consisting of an interconnection of K gates where each gate changes its state after applying the incident input. For simplicity of example, let us ignore the wire delays for the moment. The overall measured path timing can be written as the sum of the delays of the K individual gates with an added measurement error term. Similarly, it is possible to test multiple paths and then write a linear system of timing equations. Noninvasive GLC aims at finding the individual gate delay values from the noisy equations by forming a linear optimization that attempts to minimize the measurement errors.

Assume the j-th test vector($j = 1, \ldots, J$) measures the delay of the path P_j denoted by $T_{meas}(P_j)$ and the measurement incurs the error $err_T(P_j)$. The path consists of the gates G_{k_j} with delay $T(G_{k_j})$, where $k_j = 1, \ldots, K_j$ is the index of the elements on path j. The optimization problem objective function (OF) and constraints are then:

$$OF: \quad \min_{1 \le j \le J} \mathcal{F}(err_T(P_j)) \quad (2)$$

$$C's: \quad \Sigma_{k_j=1}^{K_j} T(G_{k_j}) = T_{meas}(P_j) + err(P_j),$$
$$P_j = \{G_{k_j}\}_1^{K_j}, \qquad 1 \le j \le J.$$

$\mathcal{F}$ is a metric for quantifying the measurement errors; commonly used form of $\mathcal{F}$ is the l_2 norm of errors. $T(G_{k_j})$ is sometimes expressed as a product of its nominal value $T_{nom}(G_{k_j})$ and a scaling factor (because of PV) $\delta(G_{k_j})$, i.e., $T(G_{k_j}) = \delta(G_{k_j})T_{nom}(G_{k_j})$. Furthermore, for the purpose of HT detection, we add one Trojan variable H_j in each constraint as an indicator for the HT. Our idea is that if there is any Trojan embedded, it will cause a systematic bias in one of the paths in the circuit, which can be captured by H_j.

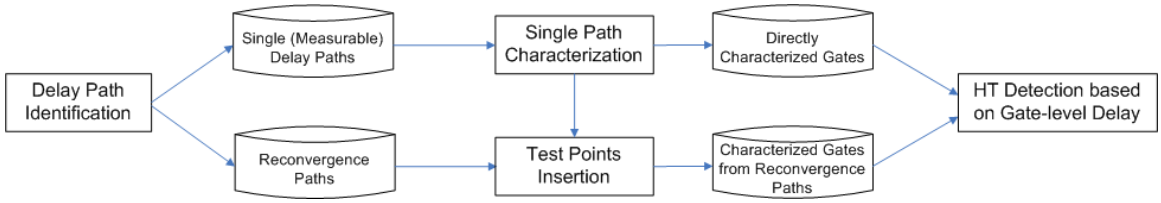


Figure 2: Overall flow of Trojan detection using delay characterization.

2.4 Related Work

[1] proposed one of the first Trojan detection techniques in 2007. They used the side-channel measurements (e.g., power and temperature) to construct fingerprints for a specific design by destructively testing a few chips. The IC instances were authenticated by comparing to the measured fingerprints. The technique had a number of limitations since it did not consider PV, and since destructive tests are both slow and expensive. A number of other early Trojan detection methods employed functional test and verification techniques. Functional tests simulate the circuit inputs and monitor the outputs to see if they match the expected patterns. For example, [26] proposed generation of test vectors that maximize the Trojan detection likelihood consisting of 2-input gates that rarely switch; [3] suggested automatic test pattern generation (ATPG) techniques within the divide-and-conquer paradigm. [9] developed a periodic on-line self-test approach by using on-chip clock control to monitor delay shifts over time and detect early-life failure.

Recently, a number of Trojan detection methods using side-channel based analysis have been developed, including [8, 2, 22]. For example, two types of Trojan detection techniques analyzed pertinent ICs in terms of their delay from one flip-flop to another using either deterministic [12] or statistical methods [8]. A number of Trojan detection techniques advocated the use of switching power measurements[4]. [18] presented a comprehensive survey of hardware Trojan detection. The common assumption for most Trojan detection scenarios is that a correct golden model (in form of post-layout simulations) of the IC is available. The manufactured IC's properties are then compared to this model.

GLC is an attractive method that greatly improves the accuracy of post-silicon characterization [16, 27, 20, 10]. The existing GLC methods are either purely based on the leakage power measurements, e.g., [16], or if they include the timing tests they do so within the constraints of the path observability using the standard delay test vectors without adding new test points [20, 10]. The existing GLC evaluations are mostly reported on smaller benchmarks. The positive aspects of GLC is its ability in handling the PV and its detection accuracy for the observable paths.

Note that our approach is different from the existing delay measurement-based HT detection methods, such as ATPG [9], in that we characterize the gate-level delay properties in order to capture even a small variation caused by the embedded HT. In order to ensure the feasibility of the solutions for gate-level characterization, we employ both test point insertion and SAT-based input vector selection approaches to create timing-based independent linear equations. Therefore, we are able to solve challenging HT problems, such as minimum exposed delay or power, which cannot be solved by the existing HT detection approaches.

3. DELAY PATHS IDENTIFICATION AND SELECTION

3.1 Overall Flow

A thorough delay-based Trojan detection requires us to characterize the timing of all the existing gates under the impact of PV, since the Trojan may be embedded at any circuit location. To achieve this goal, we design a systematic way of identifying the delay paths, break the reconvergence points, and characterize the gate-level delay properties.

The overall flow of our approach is shown in Figure 2. We first analyze the structure of the netlist and identify two types of delay paths between a specific pair of input and output: (1) those that only include one single path and thus are characterizable by direct delay measurements; and (2) those that include multiple paths in parallel and thus are not differentiable by direct delay measurements. After obtaining the two groups of paths, we first conduct GLC for the path group (1) by leveraging SAT for determining the input vectors that switch the gates and create multiple independent linear equations. Next, we select paths from group (2) to cover all the remaining gates in the circuit. In order to make the paths in group (2) characterizable, we insert additional test points at the reconvergence point of each set of parallel paths. As a result, any path that is originally in parallel with other paths would get an additional observation point for delay measurements. The inserted test points enable us to treat the parallel paths the same as the single paths in group (1). Thus, we would be able to conduct GLC on these paths and characterize all the gates.

3.2 Test Point Structure Design

To measure the delay for one of the parallel paths, we insert a test point flip-flop (D-type) at each reconvergence point. In particular, we feed the data input of the D flip-flop with the output of the last gate on the path which enables us to measure the path delay by using clocking and standard scan chain delay-fault testing methods.

We demonstrate a small example in Figure 3 regarding the test point insertion method that breaks initially unmeasurable delay paths (due to reconvergence). In this example, path a (3→10→22) and path b (3→11→16→22) originate from the same input and reconverge to the same output. Consequently, if we measure the delay from input 3 to output 22, it is difficult to determine whether the measured delay is for path a or for path b; therefore, we would not be able to create correct linear equations for the GLC of these two paths. We address this issue by inserting a D flip-flop at the end of each path, i.e., at pin 10 and pin 16. Now, we are able to bypass the reconvergence problem by measuring the delay between 3 and 10 (for path a) and between 3 and 16 (for path b) separately. This enables us to treat both

paths a and b as single paths and create a system of linear equations to find their gate-level delay characteristics.

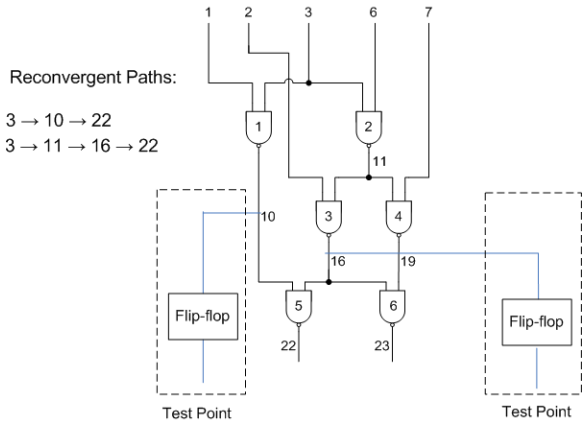


Figure 3: Example of test points insertion for delay characterization.

3.3 Reconvergence Identification

Reconvergence identification is an important step in our test point-based delay characterization. To do so, we analyze the structure of the netlist and identify the reconvergence points between each input/output pair. Assuming that the circuit netlist is a directed graph, with each interconnect as an edge $e_i \in E$, and each gate (or input, output) as a node $n_i \in N$, we define a reconvergence point as a node in the circuit graph that has an in-degree larger than 1.

The identification of reconvergence points in the target circuit is straightforward, because it is already implied in the structure of the circuit graph whether a node is a reconvergence point or not. However, for the purpose of delay characterization of all gates, it is infeasible to insert test point at every reconvergence point, since it would create a very high overhead. Our goal in test point insertion is to minimize the number of added test points based on the consideration that a subset of gates are already characterizable by single path characterizations, and the fact that each gate may appear in multiple paths.

Our test point insertion algorithm is shown in Algorithm 1. We first characterize all gates that are measurable on single paths. Then, for the remainder gates, we search the circuit graph and find their transitive fan-in (inputs that control these gates) and transitive fan-out (outputs that these gates drive either directly or indirectly). Next, for each pair of transitive fan-in and transitive fan-out, we develop a backtracing-based search algorithm to determine all the paths between them. We conduct a depth-first search from a specific output towards the inputs. During this process, we keep pruning the edges using backtracking to trace all the possible paths towards a specific input. After determining the list of paths for each pair of transitive fan-in and fan-out of all the unsolved gates, we obtain a list of candidate paths that we may consider to solve by adding test points. Next, we sort the list of paths in an ascending order by their length (i.e., the number of unsolved gates on the path) and (in the same order) use GLC to characterize their delay. Note that we conduct GLC for these paths in an iterative and incremental manner. In other words, the solved gates in the original paths are considered known in the next iteration, so that the linear system's size can be reduced because many paths would have overlapping gates. The ascending order requires less run time and overhead since it is easier to solve the overlapping gates on a shorter path.

Algorithm 1 Reconvergence identification and test point insertion for characterizing all gates.

Input: Circuit Netlist (Net), Gate Set (G);
Output: A set of test points (TP) for delay characterization;
1: $G_1 \leftarrow$ all measurable gates on single paths;
2: Characterize G_1 using delay measurments;
3: **for** each gate $g_i \in G - G_1$ **do**
4: $TI_i \leftarrow$ transitive fan-in of g_i;
5: $TO_i \leftarrow$ transitive fan-out of g_i;
6: Find all the paths P_i between TI_i and TO_i using backtracking-based depth-first search;
7: $len_i \leftarrow$ number of gates in $G - G_1$ that are covered by P_i;
8: Insert P_i in P;
9: **end for**
10: Sort P in ascending order based on len_i;
11: $i \leftarrow 0$;
12: **while** Not all gates in $G - G_1$ are characterized **do**
13: Insert test point TP_i to separate paths P_i;
14: Insert TP_i in TP;
15: Characterize all gates covered by P_i;
16: $i \leftarrow i + 1$;
17: **end while**
18: Return TP;

4. SAT-BASED APPROACH FOR CHARACTERIZING ALL GATES

In this section, we discuss in details our SAT-based input vector selection approach for characterizing all the gates in the circuit. The approach operates on the circuit with inserted test points as described in Algorithm 1. Our goal is to select input vectors that switch the gates in such a way that independent linear equations (described in Equation (2)) can be created to characterize all gates.

4.1 Problem Formulation

For gate-level delay characterization of a single path, or separated reconvergent paths by using test points, we must select a set of input vectors that switch all the gates on the measured path, so that we can create the linear constraints as described in Equation (2). To ensure that the gate-level delays are solvable from the linear program, the coefficients in the linear constraints must be independent from each other, or in a more formal statement, the rank of the matrix formed by $\delta(G_{k_j})$ must be larger than the number of gates on the measured path.

Our observation from the delay model (i.e., Equation (1)) is that the nominal delay values (i.e., the coefficients) are dependent on the switching patterns of the corresponding gates (e.g., either switching from low-to-high or high-to-low). Therefore, to increase the coefficient matrix rank, we must create a variety of independent combinations in terms of the switching patterns for all the gates. The input vector selection problem can be formulated as follows:

Input Vector Selection Problem. Given an IC with N gates, where each gate $i(1 \leq i < N)$ has a required signal for each of its inputs in order to create solvable linear program in the form of Euqation (2), the input vector selection problem aims to find the input vectors that satisfy the signal requirements of all the gates.

4.2 Input Vector Selection Using SAT

In order to solve the input vector selection problem, we convert it to a SAT problem, where the signal of each gate can be represented by a Boolean expression in a clause. The SAT problem aims to find all the variable (gate/pin signal) values that evaluate all the clauses to be true. In particular, the SAT problem is formulated as follows:

$$C_1 \wedge C_2 \wedge \ldots \wedge C_m = true \tag{3}$$

and

$$C_i = \begin{cases} O_i(I_1, I_2, ...I_k), & s_i = 1; \\ !O_i(I_1, I_2, ...I_k), & s_i = 0; \\ 1, & s_i = don't - care; \end{cases}$$

where C_i is the clause for setting pin i to its objective signal; $O(I_1, I_2, ...I_k)$ is the Boolean expression for the signal of pin i based on inputs I_1 to I_k; s_i is the objective signal of pin i; m is the number of pins in the circuit; and k is the number of primary inputs of the circuit.

The solution of the SAT problem provides us with specific input signals of $O(I_1, I_2, ...I_k)$ that satisfy Equation (3), or in certain cases, reports that the solution is infeasible for the specified objective signals. SAT problem is an NP-complete problem, for which there has been a large number of SAT solvers proposed in the SAT community. For the discussion of this paper, we employ sat4j [17] to solve the SAT problem in finding the input vectors.

By solving the SAT problem using a SAT solver, we can determine the input vectors that create the switching patterns. In the case where the resulting SAT problem is not solvable, we iteratively add additional test points to break the single path into multiple paths, until all the paths are characterizable using delay measurements. In this way, we obtain a provably complete coverage of all the gates in the target circuit in terms of delay characterization.

5. EVALUATION RESULTS

We evaluate our delay characterization-based Trojan detection method on a set of ISCAS'85, ISCAS'89, and ITC'99 benchmarks. For each benchmark, we simulate the 45nm technology and generate the variation of threshold voltage following the Gaussian PV model. Also, we consider the spatial correlation of effective channel length following the quad-tree model [5]. Furthermore, we add test points to the original design netlist to make all the gates visible for delay measurements and characterization. For the implementation, we use Synopsys Design Compiler Version E-2010.12-SP5 to synthesize. The target library is the FreePDK 45nm Standard cell library [13]. Finally we use Cadence Soc Encounter Version 9.1 to place and route our design. An example of the generated layouts before and after test point insertion is presented in Appendix A.

We consider the following evaluation metrics: the Trojan detection coverage (i.e., how many gates we are able to cover in terms of the delay-based Trojan detection), characterization accuracy (i.e., what is the characterization error between the characterized delay values and the the actual ones), and the overhead of test point insertion (i.e., how much is the increase of delay due to the inserted test points). Based on the evaluation of delay characterization, we further simulate the Trojan detection process on the same set of benchmarks and quantify the resulting false positives and false negatives.

5.1 Accuracy of Delay Characterization

Table 1 compares the number of gates that are covered by the characterization approach with and without test points inserted. In case of no test points, the only possibility to conduct delay characterization is that there are no reconvergences from a specific input to a specific output in the original design. It can be seen that there is no full coverage of gate delays in any of the tested benchmarks. The highest possible coverage rate is 60%, which still leaves a large portion of the circuit under the risk of Trojan attacks. However, after we insert test points to break the reconvergences, we could characterize all the gates in each benchmark.

Furthermore, the rightmost column shows the average characterization errors in the test point-based approach. The simulations were conducted under the assumption of 1% delay measurement errors. Note that the delay measurement errors reported in [19] and [6] are far less than 1% and, therefore, we are overestimating the measurement errors that can be further improved in real scenarios. For all benchmarks, we observe less than 1% error for gate-level delay characterizations. The accuracy in delay characterization that covers all gates in the target circuits serves as a foundation for the Trojan detection process.

5.2 Test Points Overhead

The test point-based delay characterization process incurs two sources of overhead: (1) the area cost for the inserted test points; and (2) the delay increase on critical paths due to the insertion of test points. We evaluate (1) by identifying the number of test points (i.e., flip-flops) that have to be added to cover all gates. As shown in Table 2, the number of test points and characterized paths are within a limited range, making it acceptable for the design and manufacturing of the Trojan-detectable chips. For the evaluation of (2), we further conducted simulations on the ICs with all the test points inserted in terms of the critical path delays and compare them with the original values before any test points are added. The results are shown in Figure 4. We can see from the results that the average delay increase is within 5% of the original delay.

5.3 Quality of Trojan Detection

We evaluate the Trojan detection process based on the test point insertion and the gate-level delay characterization. As shown in Figure 5, we simulate two cases for each benchmark circuit: (1) there is a Trojan gate embedded at a random location in the target circuit; and (2) there are no Trojans embedded. To capture the impact of Trojan gates by delay characterization, we characterize the Trojan variable H_j as defined in Section 2.3 by solving the linear program (Equation (2)). We observe from the results in Figure

Table 1: Results of delay characterization with and without test points. The number of characterizable gates without test points is no more than 60% of all the gates, leaving a large portion of the circuit under the risk of Trojan attacks. However, with inserted test points, we can characterize 100% of the gates accurately in all the tested benchmark circuits.

Benchmark	# Gates	# Inputs	# Outputs	# Characterized Gates (No TPs)	# Characterized Gates (With TPs)	GLC Error (%)
C499	202	41	32	122 (60.4%)	202 (100%)	3.1E-03
C880	383	60	26	178 (46.5%)	383 (100%)	0.10
C1355	546	41	32	0 (0%)	546 (100%)	0.10
C1908	880	33	25	141 (16.0%)	880 (100%)	0.10
C2670	1,193	233	140	262 (22.0%)	1,193 (100%)	0.98
C3540	1,669	50	22	127 (7.61%)	1,669 (100%)	0.95
C5315	2,307	178	123	383 (16.6%)	2,307 (100%)	0.11
C7552	3,512	207	108	960 (27.3%)	3,512 (100%)	0.51
S38584	19,253	38	304	3,022 (15.7%)	19,253 (100%)	0.24
B17	32,192	37	97	12,909 (40.1%)	32,192 (100%)	0.27

Table 2: Overhead of test point insertion in terms of number of test points (i.e., area cost) and increased delay of critical path (i.e., performance overhead).

Benchmark	# Gates	# Inputs	# Outputs	# Characterized Paths	# TPs	Increased Delay (%)
C499	202	41	32	107	32	2.77
C880	383	60	26	156	19	1.74
C1355	546	41	32	208	104	4.68
C1908	880	33	25	219	88	2.48
C2670	1,193	233	140	368	94	2.95
C3540	1,669	50	22	659	136	2.58
C5315	2,307	178	123	1,025	278	1.35
C7552	3,512	207	108	797	97	1.00
S38584	19,253	38	304	160	988	0.74
B17	32,192	37	97	1,480	370	0.83

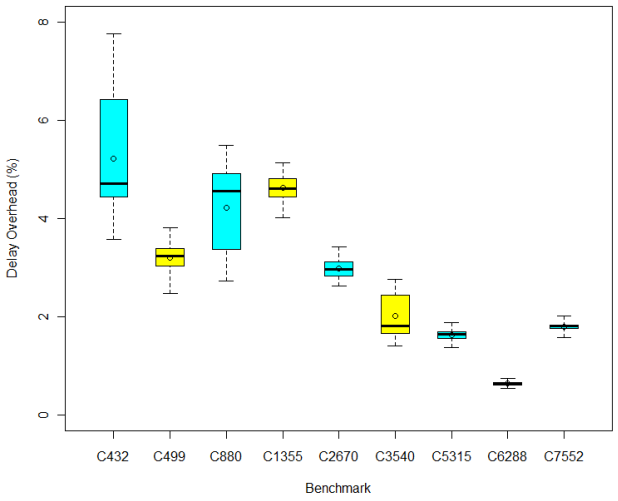


Figure 4: Delay overhead under process variation.

5 that the Trojan variable has a large value in the Trojan-present case (shown in yellow bars), compared to close-to-0 value in the no-Trojan case (shown in blue bars). There is a large gap between the values of Trojan variables in the two cases, with which we can draw a decision line and distinguish them. Since the values of the Trojan variable do not have any overlap between the two cases, one could obtain zero false negatives and zero false positives in Trojan detection, considering the high resolution of the delay measurements.

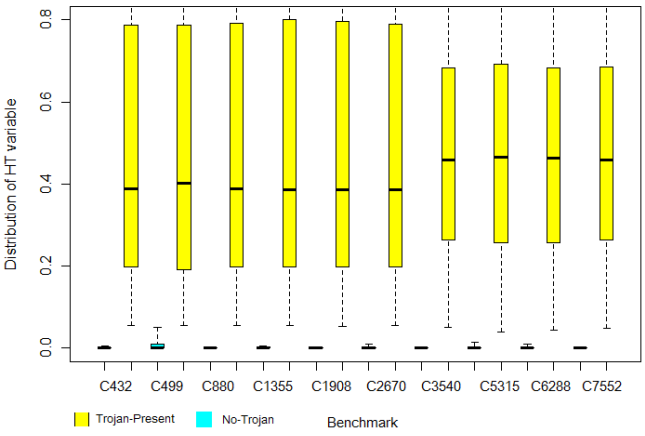


Figure 5: Trojan detection results. The yellow bars and blue bars show the Trojan variable in the Trojan-present and no-Trojan cases, respectively.

6. CONCLUSION

We have developed a test point insertion method that can be used for gate-level delay characterization to detect the existence of hardware Trojans. The test point insertion approach ensured that all the components on the target circuit are provably observable through delay measurements, so that the Trojan detection scheme has a full coverage over all possible circuit locations. We evaluated the approach on a set of ISCAS and ITC benchmarks. The results showed that the proposed scheme can detect Trojan accurately with less than 5% average performance overhead.

7. ACKNOWLEDGEMENT

This work was supported in part by the NSF under Awards CNS-0958369, CNS-1059435, CCF-0926127, CNS-1059416, and CCF-1116858, in part by the DARPA/MTO Grant N66001-11-1-4103, and is based upon research performed in collaborative facilities renovated with funds from the National Science Foundation under Grant No. 0963183, an award funded under the American Recovery and Reinvestment Act of 2009 (ARRA).

8. REFERENCES

[1] M. Agarwal, B. Paul, M. Zhang, and S. Mitra. Circuit failure prediction and its application to transistor aging. In *VLSI Test Symposium (VTS)*, pages 277–286, 2007.

[2] Y. Alkabani and F. Koushanfar. Consistency-based characterization for IC Trojan detection. In *IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 123–127, 2009.

[3] M. Banga and M. Hsiao. A region based approach for the identification of hardware Trojans. In *IEEE International Workshop on Hardware-Oriented Security and Trust (HOST)*, pages 40–47, 2008.

[4] M. Banga and M. Hsiao. VITAMIN: Voltage inversion technique to ascertain malicious insertions in ICs. In *IEEE International Workshop on Hardware-Oriented Security and Trust (HOST)*, pages 104–107, 2009.

[5] B. Cline, K. Chopra, D. Blaauw, and Y. Cao. Analysis and modeling of CD variation for statistical static timing. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 60–66, 2006.

[6] R. Datta, G. Carpenter, K. Nowka, and J. Abraham. A scheme for on-chip timing characterization. In *IEEE VLSI Test Symposium (VTS)*, pages 24–29, 2006.

[7] Defense Science Board (DSB) study on high performance microchip supply, http://www.acq.osd.mil/dsb/reports/2005-02-hpms_report_final.pdf.

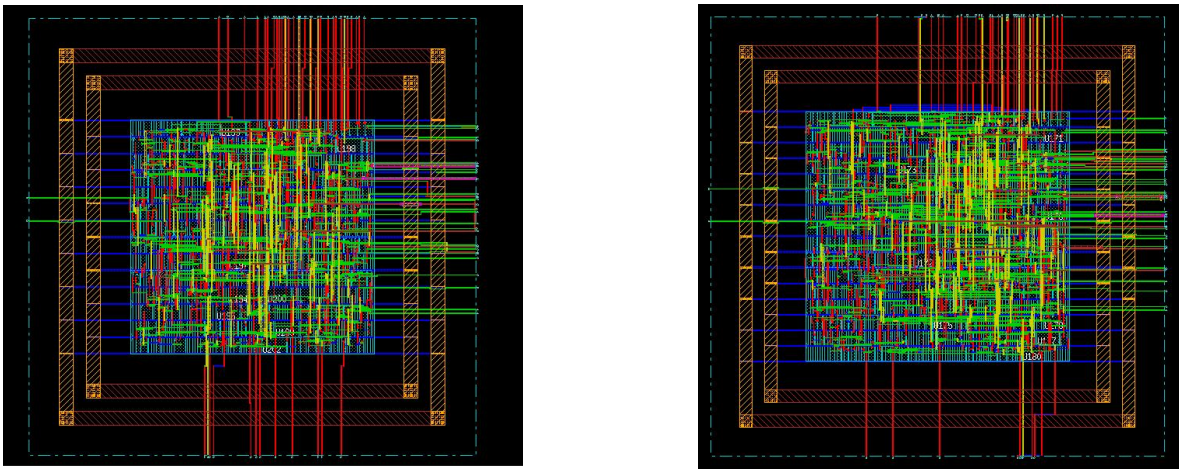
[8] Y. Jin and Y. Makris. Hardware Trojan detection using path delay fingerprint. In *IEEE International Workshop on Hardware-Oriented Security and Trust (HOST)*, pages 51–57, 2008.

[9] Y. Kim, Y. Kameda, H. Kim, M. Mizuno, and S. Mitra. Low-cost gate-oxide early-life failure detection in robust systems. In *IEEE Symposium on VLSI Circuits (VLSIC)*, pages 125 –126, 2010.

[10] F. Koushanfar and A. Mirhoseini. A unified framework for multimodal submodular integrated circuits Trojan detection. *IEEE Transactions on Information Forensics and Security*, 6(1):162–174, 2011.

[11] F. Koushanfar, A. Mirhoseini, and Y. Alkabani. A unified submodular framework for multimodal IC Trojan detection. In *Information Hiding*, pages 17–32, 2010.

[12] J. Li and J. Lach. At-speed delay characterization for IC authentication and Trojan horse detection. In



(a) Before test point insertion

(b) After test point insertion

Figure 6: Layout of benchmark C880 before and after test point insertion.

IEEE International Workshop on Hardware-Oriented Security and Trust (HOST), pages 8–14, 2008.

[13] http://www.eda.ncsu.edu/wiki/freepdk.

[14] M. Majzoobi, E. Dyer, A. Elnably, and F. Koushanfar. Rapid FPGA characterization using clock synthesis and signal sparsity. In *International Test Conference (ITC)*, pages 1–10, 2010.

[15] D. Markovic, C. Wang, L. Alarcon, T. Liu, and J. Rabaey. Ultralow-power design in near-threshold region. *Proceedings of the IEEE*, 98(2):237–252, 2010.

[16] M. Potkonjak, A. Nahapetian, M. Nelson, and T. Massey. Hardware Trojan horse detection using gate-level characterization. In *Design Automation Conference (DAC)*, pages 688–693, 2009.

[17] http://www.sat4j.org.

[18] M. Tehranipoor and F. Koushanfar. A survey of hardware Trojan taxonomy and detection. *IEEE Design Test of Computers*, 27(1):10–25, 2010.

[19] X. Wang, M. Tehranipoor, and R. Datta. A novel architecture for on-chip path delay measurement. In *International Test Conference (ITC)*, pages 1–10, 2009.

[20] S. Wei, S. Meguerdichian, and M. Potkonjak. Gate-level characterization: Foundations and hardware security applications. In *Design Automation Conference (DAC)*, pages 222–227, 2010.

[21] S. Wei, S. Meguerdichian, and M. Potkonjak. Malicious circuitry detection using thermal conditioning. *IEEE Transactions on Information Forensics and Security*, 6(3):1136–1145, 2011.

[22] S. Wei, A. Nahapetian, M. Nelson, F. Koushanfar, and M. Potkonjak. Gate characterization using singular value decomposition: Foundations and applications. *IEEE Transactions on Information Forensics and Security*, 7(2):765–773, 2012.

[23] S. Wei and M. Potkonjak. Scalable segmentation-based malicious circuitry detection and diagnosis. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 483–486, 2010.

[24] S. Wei and M. Potkonjak. Scalable consistency-based hardware Trojan detection and diagnosis. In *International Conference on Network and System Security (NSS)*, pages 176–183, 2011.

[25] S. Wei and M. Potkonjak. Scalable hardware Trojan diagnosis. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 20(6):1049–1057, 2011.

[26] F. Wolff, C. Papachristou, S. Bhunia, and R. Chakraborty. Towards Trojan-free trusted ICs: Problem analysis and detection scheme. In *Design, Automation and Test in Europe (DATE)*, pages 1362–1365, 2008.

[27] W. Zhang, X. Li, and R. Rutenbar. Bayesian virtual probe: minimizing variation characterization cost for nanoscale IC technologies via Bayesian inference. In *Design Automation Conference (DAC)*, pages 262–267, 2010.

[28] J. Zheng, E. Chen, and M. Potkonjak. A benign hardware Trojan on FPGA-based embedded systems. In *International Conference on Field Programmable Logic and Applications (FPL)*, to be published, 2012.

[29] J. Zheng and M. Potkonjak. Securing netlist-level FPGA design through exploiting process variation and degradation. In *International Symposium on Field-Programmable Gate Arrays (FPGA)*, pages 129–139, 2012.

APPENDIX

A. LAYOUT EXAMPLE

Figure 6 shows an layout example of ISCAS'85 benchmark C880 that we implemented using Synopsys Design Compiler Version E-2010.12-SP5 and Cadence Soc Encounter Version 9.1. The target library is the FreePDK 45nm Standard cell library [13]. In this example, there are 19 test points (Flip-flops) that are added for the coverage of all gates in the circuit in terms of delay characterization. We present both the original layout and the one after test point insertion. The inserted test points result in very limited area overhead and layout changes.