Trojan Detection via Delay Measurements: A New Approach to Select Paths and Vectors to Maximize Effectiveness and Minimize Cost

Byeongju Cha and Sandeep K. Gupta
Ming Hsieh Department of Electrical Engineering
University of Southern California, Los Angeles, USA
{byeongjc, sandeep}@usc.edu

***Abstract*—One of the growing issues in IC design is how to establish trustworthiness of chips fabricated by untrusted vendors. Such process, often called Trojan detection, is challenging since the specifics of hardware Trojans inserted by intelligent adversaries are difficult to predict and most Trojans do not affect the logic behavior of the circuit unless they are activated. Also, Trojan detection via parametric measurements becomes increasingly difficult with increasing levels of process variations.**

In this paper we propose a method that maximizes the resolution of each path delay measurement, in terms of its ability to detect the targeted Trojan. In particular, for each Trojan, our approach accentuates the Trojan's impact by generating a vector that sensitizes the shortest path passing via the Trojan's site. We estimate the minimum number of chips to which each vector must be applied to detect the Trojan with sufficient confidence for a given level of process variations. Finally, we demonstrate the significant improvements in effectiveness and cost provided by our approach under high levels of process variations. Experimental results on several benchmark circuits show that we can achieve dramatic reduction in test cost using our approach compared to classical path delay testing.

***Keywords*- Hardware Trojan; security; parametric test.**

I. Introduction

In semiconductor industry, to reduce costs, many steps of digital IC design are now conducted by outside vendors. In addition, it is impossible for the designers of relatively low volume applications to develop state-of-the-art fabrication facilities by themselves and hence they are increasingly forced to use the services of outside fabricators. Due to these reasons, it is increasingly common for a new IC's original designers to lose direct control of many design and fabrication steps. This increases the opportunities for intelligent and resourceful adversaries to tamper with the circuit by introducing hardware Trojans, especially during fabrication steps. Detecting *hardware Trojans* by destructive physical inspection or reverse engineering is costly and might fail as the scaling down of the IC device dimensions makes a well-designed Trojan circuitry very difficult to detect. Hence, it is important to develop a new framework to detect possible hardware Trojans within ICs.

Existing Trojan detection methods can be categorized into two major types: *logic test methods*, which apply vectors and examine logic values at the circuit's outputs [1][2][3], and *parametric test methods*, which apply vectors and measure values of parameters, such as power/ground currents [4] or path delays [5][6]. However, logic test methods require Trojan activation, which has been shown to be extremely difficult [7][8], since the specifics of the Trojan are unknown and we can never be sure of activating the Trojan. Also, some parametric test methods measuring power or current are inefficient since many Trojans change the power/ground current by a very small percentage. As power/ground measurements are performed over large regions of a chip, namely, each power/ground pin or even the entire chip [1], it is difficult to obtain sufficient resolution under process variations.

In contrast to other parametric test methods, delay measurements benefit from the fact that the delay of each path can be measured separately. Thus, the resolution of delay measurement for one path is independent of the other paths in a logic block and other logic blocks on the chip. *Hence, we pursue Trojan detection via delay measurements.* However, increasing levels of process variations make it more difficult to detect Trojans. In addition, Trojan designers will try to design Trojans that are minimally invasive in terms of circuit parameters, in order to prevent them from being detected by any test approach. Thus, the main challenge of our approach is *how to detect a Trojan that gives the smallest impact on delays under increasing levels of process variations,* since the amount of extra delay induced by a minimally delay-invasive Trojan becomes smaller in magnitude compared to the impact of process variations [9].

In [10], we presented our model of a Trojan that does not assume any specific type of Trojan. Instead, we capture characteristics of the most difficult type of Trojans for our approach to detect and identify them as a set of *surrogates*, to derive the most conservative and yet a general Trojan model. Also this approach focuses on the effect of process variations (σ) on the average value of the total path delay (μ) and on reducing its effect (σ/μ) via calibration, assuming that a Trojan is minimally delay-invasive, i.e., its impact on delay (Δ) is minimal. This approach uses a new hypothesis testing method which can statistically decide whether a Trojan exists or not. While this approach is guaranteed to detect even minimally delay-invasive Trojans, it requires substantial numbers of chips to be tested to ensure high confidence in the results.

In this paper, we focus on another parameter, namely the impact of a Trojan on the path delay (Δ/μ). Especially, we develop a ***path selection scheme*** for a target Trojan which maximizes the impact of a Trojan on the measured delay. As Trojans are expected to cause minimal delay deviations, our goal is to select paths which maximize the additional delay induced by the Trojan with respect to the nominal path delays and effects of process variations. In contrast to existing methods that target critical paths [11], ***our path selection scheme targets paths having the smallest path delay values to***

* This research was supported by National Science Foundation under grant CNS-1018937

978-3-9815370-0-0/DATE13/©2013 EDAA

maximize the impact of a Trojan on each path's delay. We also derive ***new logic and timing conditions*** that sequences of vectors must satisfy to detect any particular Trojan. The effectiveness of our approach is demonstrated using an industrial 65nm technology for high levels of process variations provided by a foundry.

The rest of the paper is organized as follows. Section II presents background including our models of process variations and Trojans used in our approach. We also summarize our previous approach [10]. In Section III, we propose our new approach to improve the resolution of the tests by targeting shorter paths and developing a new test generation procedure for these paths. Also, we formulate this problem as hypothesis testing that minimizes test cost with a desired level of confidence under a given level of process variations. We present an integrated Trojan detection algorithm in Section IV and present experimental results in Section V. Finally, conclusions are drawn in Section VI.

II. BACKGROUND

A. *Process variations*

For any given vector, we characterize delays of paths in benchmark circuits using realistic delay values and under realistic levels of process variations supplied by the vendors for the fabrication process in the form of technology files. In particular, we use an industrial 65nm technology and use the delay model, including inter- and intra-die variations, provided by the foundry which fabricates chips using this technology. We perform Monte Carlo simulations to obtain realistic distributions of path delay values, using the Cadence Spectre simulator in a manner where it uses the foundry-supplied model of process variations in terms of variations in about 50 device parameters including, L_{eff}, V_{th}, t_{ox}, etc. [12].

B. *Characterization of Trojans*

Any existing strategy that enumerates specific types of Trojans is likely to be incomplete since Trojans are continuously developed by intelligent adversaries. In addition, Trojans will be designed to give only the smallest impact on circuit parameters as discussed in Section I. To improve completeness of our models of Trojans, we have proposed a new approach for capturing the necessary characteristics of Trojans. In [10], we introduced ***a set of surrogate targets*** which characterizes all possible Trojans by deriving a set of necessary conditions that must be satisfied by any Trojan designed to be minimally invasive in terms of their impacts on delays, i.e., ***the most challenging Trojans***.

(1) A Trojan must involve a connection between a Trojan site in at least one original circuit block and the newly added Trojan block(s).

(2) This connection will take the form of an *additional fanout of minimum load* at the Trojan site, as this is the most difficult type of Trojan for our approach to detect.

Since such a Trojan has the minimum impact on the original circuit's delay, only giving the smallest change to the original circuit, we believe that our surrogate model is the most challenging Trojan for our approach to detect. Since we do not make any additional assumptions besides the above two points, our approach also can be generally applied to any type of Trojans.

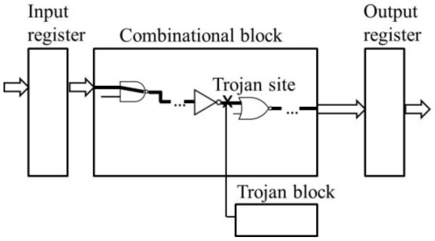


Figure 1. The model of a Trojan

C. *Previous approach* [10] *– Reducing the effect of process variations on delays via calibration*

Based on the model of a Trojan discussed above, we developed a calibration method which reduces the effect of process variations dramatically via some additional measurements on test structures inserted into the circuit. The main idea of this approach is that the effects of global variations, or inter-chip variations, is almost identical among delays measured on a particular chip, where local variations, or intra-chip variations, affect differently delays of different paths of the chip. Thus, we arrived at the observation that *the effect of global variations can be eliminated via calibration, provided that measurements are performed on test structures to measure the effect of global variations*.

In [10], we targeted the effect of process variations and focused on how to minimize its impact on delay. In contrast, in this paper, we will show how to maximize the impact of a Trojan on delay. All our approaches are based on our foundation which enables us to detect any type of a Trojan, and will be discussed in the next section.

III. MAIN IDEA TO MINIMIZE TROJAN DETECTION COST

A. *Problem statement*

We formulate the problem of Trojan detection at an arbitrary line (Trojan site) in the circuit. Let *m* be the number of lines in the original design of a logic block, say *C*. *The question is, how do we detect a Trojan which is suspected to exist on an arbitrary line, say i, where the Trojan induces a minimum additional load at the line*. We assume that the same Trojan will be inserted in every copy of the design *C*, i.e., in every fabricated chip since inserting Trojans into a subset of chips requires additional masks and is very expensive [7].

The total delay of an arbitrary path passing via line *i*, say P, using vector V can be expressed as below:

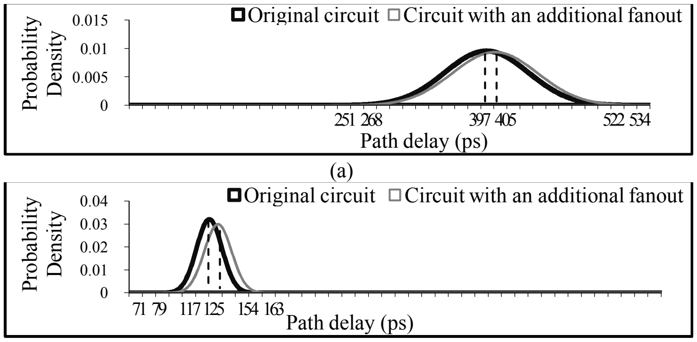
$$D(P,V) = D_N(P,V) + \Delta D_{var}(P,V) + \Delta D_T(P,V),$$

where three parameters are (1) nominal delay of path P, $D_N(P,V)$, (2) the effect of process variations on the delay of P, $\Delta D_{var}(P,V)$, which is the overall effect of inter- and intra-chip variations on the path delay, and (3) extra delay induced by a Trojan at line i, $\Delta D_T(P,V)$. Among these three parameters, *the effect of process variation follows random distribution with standard deviation* (σ) *which is typically bi-directional around a mean* (μ), e.g., normal or truncated normal distribution [13]. *In contrast, extra delay induced by Trojan* (Δ) *is uni-directional* and always increases the total delay of path P. For every copy of the design, i.e., for every fabricated chip, for the design with the Trojan, the delay of the gate/line at the Trojan site increases. This observation enables us to *prove that a* *minimally delay-invasive Trojan can always be detected, provided that we make measurements on a sufficient number of chips for specifically generated vectors*. However, this observation only guarantees the possibility of Trojan detection and the cost of Trojan detection is still too high.

B. *Path selection – Targeting the shortest paths*

In this paper, we solve the above problem by making a second observation: ***The greater the uni-directional increase due to a Trojan (Δ) compared to the standard deviation for the delay of path due to process variations (σ), the smaller the number of chips that need to be tested.*** Hence, to maximize the impact of a Trojan, ***we select the path with the smallest delay that passes via the site of the Trojan being targeted***, since the standard deviation of variations is approximately proportional to the nominal delay of the path and the impact of a Trojan on delay is almost identical among every path passing via the Trojan site. The example shown below supports our idea. (Every example we have studied exhibits the same trend.)

To show the effectiveness of selecting the shortest path, we choose two different paths with significantly different path delays that pass via the same Trojan site, line 371, in s420 benchmark circuit. We perform Monte Carlo simulations to obtain delay values shown in Fig. 2. The distribution of delay for the original version of the s420 benchmark circuit for a specific vector is shown by the darker curve in Fig. 2(a). We then obtain a version of this circuit with a Trojan by inserting a Trojan at line 371 of the circuit, and repeat the simulations to obtain the distribution shown by the lighter curve in Fig. 2(a). Note that the Trojan causes a relatively small change in delays, namely 8ps, compared to the nominal delay for the original benchmark circuit, and the variations in the delay of the original circuit caused by process variations.



(a)

(b)

Figure 2. The distribution of delay at an output of s420 considering process variations for the original circuit version, and a version with a Trojan sited on line 371. For a vector that excites (a) a long path, and (b) a shorter path.

For the same circuit and the same Trojan, we repeat the simulations for a different vector selected to excite a much shorter path that passes via the same Trojan site. It is easy to see in Fig. 2(b), that while the expected value of the additional delay due to the Trojan remains around 8ps, i.e., the same level as in Fig. 2(a), the impact of the Trojan increases significantly as a percentage of the average delay of the path. Trojan's impact also increases with respect to the variance due to process variations and hence it is easier to detect.

The idea of targeting the shortest paths is also useful due to the fact that a Trojan always increases the total path delay. The conventional delay testing method targets the longest paths and check erroneous logic values at primary outputs, which arrive later than the desired clock period. Even though the impact of a Trojan on delay is very small, the Trojan might be detected if it increases the delay of any of the longest path and the path delay goes beyond the clock period. Since the adversary is aware of every commonly used conventional testing method, he/she will try to insert a Trojan to paths other than the longest paths to avoid detection. Also, another benefit of targeting short paths is that shorter paths tend to have fewer off-path inputs than longer paths. Due to this reason, the probability that a test vector that satisfies all our conditions exists is greater for a shorter path.

Hence, in order to *detect a Trojan at line i using path delay measurement, we select the shortest path passing via line i*. We consider the following set of paths as *surrogate paths*

$$P = \{P_1, P_2, \cdots, P_m\},$$

where P_i is the shortest delay path passing via line *i*. As we select a surrogate path for each Trojan, the total number of surrogate paths is only proportional to the number of lines in the circuit. Testing of all possible Trojans using surrogate paths in the circuit only needs $O(m)$ time, in the worst case.

C. *Test generation conditions for a Trojan*

Our next task is to generate a vector for a particular surrogate path that invokes the delay of the corresponding Trojan. This task is similar to the problem of test vector generation for path delay testing during high-volume manufacturing (HVM) testing with two important differences. First, here we have selected the shortest paths, in contrast to the longest paths in delay testing. Second, our objective is to excite the delay of the selected path, whereas in delay testing the goal is to invoke a delay that is either greater than or equal to that of the selected path.

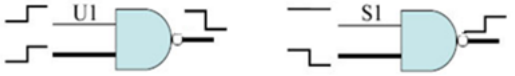


Figure 3. Conditions for robust path delay testing for an on-path NAND gate. The thick and thin lines denote on-path and off-path lines, respectively.

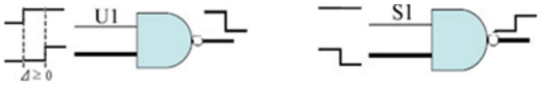


Figure 4. Conditions for a NAND gate along a surrogate path, to detect above category of Trojans

We have derived conditions that must be satisfied by a vector to ensure that our above objective is satisfied for any path selected as a surrogate path. Fig. 3 shows the conditions that must be satisfied by a vector generated for path delay testing, for an on-path NAND gate. In contrast, Fig. 4 shows the conditions we have derived for generating a vector for a surrogate path for a Trojan. In the first case, where the on-path input of the NAND gate has a rising transition, robust testing only focuses on invoking delay that is equal to or greater than the delay of the path [11]. However, as our goal is to invoke the delay of the target path, we modify these conditions to preclude cases where a late transition at an off-path input invokes delay greater than that of the target path. For the second case, where the on-path input of the NAND gate has a falling transition, delay testing as well as Trojan detection both require the off-path input to have a steady-1. This is because in both cases we must ensure that a transition at any off-path input does not decrease the delay of the target path.

If the transition at a gate's input on the path being tested is from a controlling value (c) to a non-controlling value ($\bar{c}$), then we have two conditions:

Condition I: The off-path signal values should be of the form $< x, \bar{c}>$, where x can be either 0 or 1.

Condition II: The off-path signal values should change to non-controlling value before the on-path input arrives.

We have derived new conditions for all types of gates and integrated these into our vector generation framework.

D. Path delay measurement

Finally, we consider the problems of testing short paths [7][9]. First, measurement noise might occur at connections between tester probes and circuit pins. Also, it is important to measure path delays in a manner that it can capture suitably small differences in delay caused by a Trojan, around several pico-seconds in our 65nm technology. In addition, targeting short paths may require very fast clocks which are not available in most testers. Even when available, fast clock can cause excessive heat dissipation. Finally, our approach must measure delays at the flip-flop at the output of the selected path, and not at every flip-flop of every combinational block. This is unlike conventional delay testing which checks timing violations at all flip-flops.

To satisfy one of the above requirements, we choose the *on-chip delay measurement architecture* and adapt the architecture provided in [14]. This method introduces shadow registers to the original scan registers connected to the inputs and outputs of logic blocks to measure delays while controlling the size of the skew, *Δ*, between two clocks as shown in Fig. 5. This approach is reported in [14] to provide sufficient measurement resolution, which is dependent upon how precisely we can control the skew of the shadow clock. This approach uses digitally variable resistors to control the skew size to 1ps which provides sufficient resolution, Δ_{min}, for our purposes [15]. In addition, measurement errors which might be caused by temperature and voltage changes are reported to be considered in this approach, by estimating the effect of measurement noise by monitoring circuit parameters.

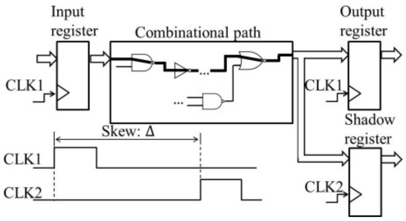


Figure 5. Path delay measurement architecture.

Moreover, we can avoid using fast clocks by introducing multiple clocks, each with the same frequency as the original clock, but with controllable phase shifts to obtain desired skews. We use a combination of multiple skewed clocks followed by the original clock (a slight modification of the classical approach used for path delay testing [11]) and capture logic values for different skew sizes. Thus, our approach does not require high frequency clocks and avoids all problems associated with excessive heat dissipation during testing of short delay paths.

Last, area overhead caused by using this architecture is relatively low. This overhead is higher if surrogate paths arrive at many different flip-flops. Thus, we have the option of reducing the area overhead by choosing surrogate paths that terminate at fewer outputs. This leads to an interesting tradeoff, since this might increase the number of chips to be tested.

Also note that our approach targets the most difficult type of a Trojan to detect. *Thus, our approach can be used in conjunction with other types of delay measurement architectures that offer different (lower) resolutions, depending on which Trojan threat models are used.* So the advantage of our approach for path selection and vector generation remain, independent from the type of delay measurement architecture.

E. Estimating the number of chips to be tested

Based on measured delay values for selected paths and generated vectors, we determine whether a Trojan exists or not. We choose *hypothesis testing* to solve this problem. Two main categories of hypothesis testing methods are parametric tests and non-parametric tests. Parametric tests such as Student's t-test assume that the data follows a general distribution, where distributions can be characterized by generic metrics such as mean and variance. However, the actual delay values, strictly speaking, do not follow a Gaussian distribution and parametric tests are not adequate to solve our problem. Non-parametric tests like goodness-of-fit can be performed for any kind of fully-specified distribution. But these tests require higher numbers of samples to achieve a certain level of confidence than parametric tests [16]. Finally, our problem is to select a more *likely* model between two competing models, "Trojan-free" and "circuit with the target Trojan". However, existing hypothesis testing methods make decisions whether the data follows a certain distribution or not. Thus no existing statistical method is an ideal method for solving our selection/classification problem.

In [10], we developed a new non-parametric hypothesis testing method based on likelihood-ratio test, which computes the ratio between conditional probabilities from the above two competing models. And it chooses a more *likely* model by comparing the computed likelihood-ratio with the threshold value determined by two constraints, Type-I and Type-II error probabilities. Type-I error probability refers to the probability that the circuit without a Trojan is falsely identified as the circuit with a Trojan (false positive), and the Type-II error probability is the probability that the circuit with a Trojan is falsely identified as a Trojan-free circuit (false negative), where the level of confidence is just another form of Type-I error probability and can be computed as (1 - Type-I error probability)*100(%). The above two parameters can be controlled to adjust the number of chips to be tested as a tradeoff of the accuracy of the test.

To show the effectiveness of our new approach, we choose Student's t-test as a baseline method, which is the most commonly used hypothesis testing method. Though it requires a specific assumption on the delay distribution to be Gaussian, this method results in smaller number of chips to be tested compared to non-parametric tests and hence makes our comparison less optimistic. In the rest of the paper, we choose 95% and 5% for the level of confidence and Type-II error probabilities, respectively, for both Student’s t-test and likelihood-ratio based method to ensure fair comparison.

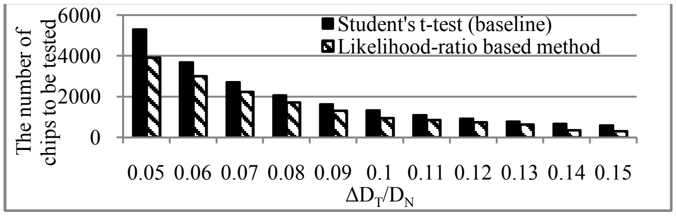


Figure 6. The number of chips to be tested using Student’s t-test and likelihood-ratio based test for different ratio between Trojan induced delay (ΔD_T) with respect to the total delay (D_N).

The main advantage of the proposed method is that, in contrast to Student’s t-test, it does not require the distribution of measured data points to be a general distribution, such as normal distribution. *Thus, the proposed method is applicable to detect a Trojan under any type of process variations.* Also for the same path and Trojan, our likelihood-ratio based test reduces the number of chips to be tested significantly when compared to Student’s t-test and the average reduction in the number of chips to be tested is 29.7%, as shown in Fig. 6. This is because our likelihood-ratio based test is designed to solve our classification problem, where Student’s t-test only gives out a decision whether delay values follow a certain Gaussian distribution and hence requires more chips to be tested to make a decision between two competing models.

IV. Our Trojan detection algorithm

```
Given a circuit with $m$ lines, our model classifies $m$ Trojans
$P_i$: The shortest path passing via line $i$ that can be sensitized by a vector, to cover the Trojan sited on line $i$
**P**: A set of $P_i$ for every detectable Trojan in the circuit
**S**: A set of every detectable Trojan
1.   Initialize $P_i$ = NULL for every $i$ = 1,…, $m$;  $\mathbf{P} = \emptyset$; and $\mathbf{S} = \emptyset$
2.   for every Trojan sited on line $i$ = 1,…, $m$ in the circuit  begin
       Enumerate every path passing through the target Trojan’s site
       Sort paths in increasing delay order $Q_1$,…, $Q_k$
3.     for $j$ = 1,…,$k$
         if (test generation for $Q_j$ is successful)
           Update $P_i = Q_j$, $\mathbf{P} = \mathbf{P} \cup \{P_i\}$ and $\mathbf{S} = \mathbf{S} \cup \{i\}$  break
4.   for each Trojan $i$ in S  begin
       for each path $j$ in P  begin
 4-1.    Simulate and measure delay of path $j$, without and in presence of
         Trojan $i$, using Monte-Carlo simulations
 4-2.    Calibrate the effect of process variations by eliminating
         global components of the total variations (optional)
 4-3.    Compute the number of fabricated chips, $n_{i,j}$, to be tested
         using vector generated for path $j$ to detect Trojan $i$.
```

Figure 7. Trojan detection algorithm for vector generation for paths for Trojans

Using the Trojan detection test procedures and test vector generation method discussed above, we have integrated and implemented all our above results as a single framework. Fig. 7 provides a high-level overview of our integrated approach. The algorithm generates paths and computes the minimum number of chips to be tested for every Trojan. For every line i in the circuit, the algorithm enumerates all the paths that pass via the line i and sorts them in increasing delay order in Step 2. In Step 3, the algorithm selects the path with the smallest delay (the shortest path) and generates a test vector for the selected path, until the test generation is successful. We have updated a timing-aware ATPG tool [17] to use our special test generation conditions and for the industrial technology. A set of detectable Trojans and corresponding set of paths are updated depending on the result of test generation. Then we measure delays and variations in delays using Monte-Carlo simulation for each path and its corresponding Trojans in Step 4. The algorithm computes the number of chips required for the test to detect every detectable Trojan via measuring delays using Cadence Spectre and the industrial technology. We can also apply our previous approach [10] in Step 4-2 to further reduce the number of chips.

It is possible that some paths may be used to detect more than one Trojan and there might be multiple paths passing via the same Trojan site. The goal is to find a minimal set of paths that detects every detectable Trojan to minimize the test cost. This problem can be stated as an integer linear programming (ILP).

$$\begin{aligned} &\text{Objective:} && \text{minimize } |\mathbf{V}|,\ \mathbf{V} \subseteq \mathbf{P} \\ &\text{Constraints:} && \textstyle\sum_{j \subseteq \mathbf{P}} x_{i,j} \geq 1 \text{ for } 1 \leq i \leq m \\ & && x_{i,j} = 1, \text{ if } i \in \mathbf{S} \text{ and } j \in \mathbf{P} \\ & && x_{i,j} = 0, \text{ if } i \notin \mathbf{S} \text{ or } j \notin \mathbf{P}, \end{aligned}$$

where $x_{i,j}$ is an indicator that shows whether Trojan i is detectable using path j ($x_{i,j} = 1$) or not ($x_{i,j} = 0$) and **V** is a minimal set of paths to be used for testing. The above problem can be solved using greedy heuristics. Two metrics, test cost and Trojan coverage are computed as follows:

$$\text{Test Cost} = \textstyle\sum_{j \in \mathbf{V}} (\max_i n_{i,j}),$$

$$\text{Trojan Coverage (TC)} = \frac{|\mathbf{V}|}{m} \times 100\ (\%),$$

where test cost is sum of the number of required chips to be tested for every path in **V**. In the next section, we will present two different test costs, namely the cost obtained from our approach only, and both our previous and current approaches by applying Step 4-2, to maintain fairness in comparison and show benefits of these two approaches independently.

V. Experimental Results

For our experiments, we use the combinational parts of nine ISCAS89 benchmark circuits. The timing-aware ATPG tool for the proposed test generation procedure has been implemented on Intel Core i7 with 2.67GHz processors and 4GB of main memory, for an industrial 65nm technology [17]. In Table II, the number of paths does not exceed the number of lines of each circuit, so the complexity of simulations for each circuit is *O(m)*, as shown in Section III-B. The test generation time, including shortest path selection and vector generation with special test conditions, has been greatly reduced by using a fast timing-aware ATPG.

Using obtained paths, path delay values are measured from Monte-Carlo simulations using Cadence Spectre for an industrial technology and using levels of process variations provided by a fabricator, as described in Section II-A. As a Trojan which induces a minimum load at a line in the circuit, we use a minimum-sized inverter as an extra fanout that induces extra delay at each particular Trojan site. The hypothesis testing methods and our Trojan detection algorithm are implemented using MATLAB. In Table II, we compute the Trojan detection cost for nine benchmark circuits of three different methods. We compare the costs of (i) and (ii), and (i) and (iii), to see improvements in the cost of Trojan detection independently. Note that method-(i) is the baseline method including classical delay testing method and Student's t-test, and method-(ii) is the method targeting the shortest paths we have proposed in this paper, but still exploits Student's t-test. Our new approach (proposed here) with the shortest path (method-(ii)) dramatically improves 2.1X in the test cost compared to the classical delay testing targeting the longest paths with Student's t-test (method-(i)), for the same value of Trojan coverage.

Furthermore, we can see that even higher improvement can be obtained by applying the new method proposed in this paper (method-(iii)) in conjunction with the idea of calibration of process variations with the likelihood-ratio based method that we proposed in [10]. This is the case since our new and previous methods target two independent parameters, the impact of a Trojan and the effect of process variations on delays, respectively. Thus, advantages of our two approaches are orthogonal and finally we get 4.51X of improvement in the test cost. For sensitive chips that are fabricated in small volumes, our approach may be the only one that can perform Trojan detection with desired confidence level.

In summary, the results clearly demonstrate that our approach which targets the shortest paths gives better results than classical delay testing method. In addition, the results show that the Trojan detection procedure that also uses our hypothesis testing method and our previous calibration method [10] further reduces test cost significantly.

VI. CONCLUSION

We have identified several principles for selection of target paths and generation of vectors to enable identification of Trojans using our notion of surrogates. Our approach is also demonstrated as being efficient in the presence of increasing levels of process variations, which cannot be tackled by classical testing and validation approaches. The experimental results show that the proposed approach reduces test cost significantly compared to classical methods. Last, benefits of the new approach we present here and the one we presented earlier [10] are shown to be independent and collectively lead to dramatic decrease in test costs.

REFERENCES

[1] F. Wolff, et al, "Towards Trojan-free trusted ICs: Problem analysis and detection scheme," Design and Test in Europe, 2008, pp. 1362-1365.

[2] S. Jha and S. K. Jha, "Randomization Based Probabilistic Approach to Detect Trojan Circuits," High Assurance Systems Engineering Symposium, 2008, pp. 117-124.

[3] H. Salmani, M. Tehranipoor, and J. Plusquellic, "A Novel Technique for Improving Hardware Trojan Detection and Reducing Trojan Activation Time," IEEE Transactions on VLSI, Vol. 20, Issue 1, 2012, pp. 112-125.

[4] M. Banga and M. S. Hsiao, "A Region Based Approach for the Identification of Hardware Trojans," IEEE International Workshop on Hardware-Oriented Security and Trust, 2008, pp. 40-47.

[5] Y. Jin and Y. Makris, "Hardware Trojan Detection Using Path Delay Fingerprint," IEEE International Workshop on Hardware-Oriented Security and Trust, 2008, pp. 51-57.

[6] M. Potkonjak, et al., "Hardware Trojan horse detection using gate-level characterization," Design Automation Conference, 2009, pp. 688-693.

[7] M. Tehranipoor, et al., "Trustworthy Hardware: Trojan Detection Solutions and Design-for-Trust Challenges," IEEE Computer Magazine, Vol. 44, Issue 7, 2011, pp. 66-74.

[8] M. Tehranipoor and F. Koushanfar, "A Survey of Hardware Trojan Taxonomy and Detection," IEEE Design & Test of Computers, 2010.

[9] D. Rai and J. Lach, "Performance of delay-based Trojan detection techniques under parameter variations," IEEE International Workshop on Hardware-Oriented Security and Trust, 2009, pp. 58-65.

[10] B. Cha and S.K. Gupta, "Efficient Trojan detection via calibration of process variations," Asian Test Symposium, 2012

[11] N. Jha and S. K. Gupta, Testing of Digital Systems, Cambridge, U.K.: Cambridge Univ. Press, 2003.

[12] K. Bernstein et al., "High- Performance CMOS variability in the 65-nm regime and beyond," IBM Journal of Research and Development, Vol. 50, Issue 4.5, 2006, pp. 433-449.

[13] F. Liu, "A General Framework for Spatial Correlation Modeling in VLSI Design," Design Automation Conference, 2007, pp. 817-822.

[14] J. Li and J. Lach, "Negative-Skewed Shadow Registers for At-Speed Delay Variation Characterization," International Conference on Computer Design, 2007, pp. 354-359.

[15] M. Saint-Laurent and M. Swaminathan, "A digitally adjustable resistor for path delay characterization in high frequency microprocessors," Southwest Symposium on Mixed-Signal Design, 2001, pp. 61-64.

[16] R. V. Hogg and E. A. Tanis, Probability and Statistical Inference, Pearson Education, 2008, pp. 407-463.

[17] P. Das and S.K. Gupta, "On Generating Vectors for Accurate Post-Silicon Delay Characterization," Asian Test Symposium, 2011, pp. 251-260.

TABLE II. THE COST AND TROJAN COVERAGE (TC) FOR BENCHMARK CIRCUITS. (i) BASELINE METHOD WITH CLASSICAL DELAY TESTING METHOD AND STUDENT'S T-TEST. (ii) OUR NEW APPROACH BASED ON SELECTING SHORTEST PATHS AND STUDENT'S T-TEST. (iii) OUR NEW APPROACH (SELECTING SHORTEST PATHS), IN CONJUNCTION WITHOUR PREVIOUS APPROACH IN [10] (LIKELIHOOD-RATIO BASED TEST AND CALIBRATION OF PROCESS VARIATIONS).

Benchmark circuit	Number of lines	Tested paths / Total paths	TC (%)	Test cost			Improvement in test cost		Test generation time (sec)
				(i)	(ii)	(iii)	(i)/(ii)	(i)/(iii)	
c17	17	8/22	100	960	470	235	2.04X	4.09X	41
s298	298	112/462	72.8	63950	18278	9299	3.50X	6.88X	58
s386	386	162/414	84.2	187199	115550	87097	1.62X	2.15X	69
s420	420	131/738	69.3	228157	110054	44461	2.07X	5.13X	91
s510	510	247/738	94.9	327474	94835	47500	3.45X	6.89X	92
c880	880	272/17284	78.9	345825	332553	145870	1.04X	2.37X	3300
s1238	1238	368/7116	80.6	1021218	509990	215080	2.00X	4.75X	1440
s1488	1488	365/1924	69.5	683190	366669	176150	1.86X	3.88X	960
s5378	5378	844/27084	67.5	1442322	1083721	326840	1.33X	4.41X	9300