Trojan detection via delay measurements: An 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
{byeongjc, sandeep}@usc.edu

Abstract

One of the growing issues in IC design is how to authenticate chips fabricated by untrusted vendors. Such authentication, 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 introduce the new notion of a **set of surrogate Trojan targets**, where the surrogates capture the necessary conditions that every Trojan must satisfy. We also propose **a method that maximizes the resolution of each path delay measurement, in terms of its ability to detect the targeted surrogate**. In particular, for each surrogate, our approach accentuates the surrogate's impact by generating a vector that sensitizes the **shortest path** passing via the surrogate's site. We estimate the minimum number of chips to which each vector must be applied to detect the surrogate 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 process variations. Experimental results on several benchmark circuits show that we can achieve 3.48X reduction in test cost using our approach compared to classical path delay testing with Student's t-test.

1. 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 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 and tools to detect possible hardware Trojans within ICs.

Recently, several Trojan detection techniques have been developed. These approaches include *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][5][6] or path delays [7][8][9]. In addition, a taxonomy and necessary elements of a Trojan are introduced to classify types of Trojans and to be used for evaluation of different Trojan detection strategies [10][11][12]. However, logic test methods require Trojan activation, which has been shown to be extremely difficult [10][13]. Since the specifics of the Trojan are unknown and we can never be sure of activating the Trojan, these approaches are ineffective in most scenarios. Many Trojans change the power/ground current by a very small percentage, since power/ground measurements are performed over large regions of a chip, namely, each power/ground pin or even the entire chip [1]. In contrast to these and other similar 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 the block and other blocks on the chip. *Hence, we pursue Trojan detection via delay measurements.* In this paper, we provide an approach to model Trojans, select paths, and generate vectors to detect Trojans with minimum cost and high accuracy.

In particular, our approach tackles several major challenges for Trojan detection that have been mentioned in many reports [10][13][14] and provides efficient solutions.

First, all existing approaches try to classify Trojans and make many assumptions regarding specific characteristics of Trojans. However, the type, size, and physical distribution of Trojan may vary with the intent and ingenuity of the Trojan designer. Since an intelligent adversary will continually develop new types of Trojans to retain his/her advantage, any attempt to enumerate every possible specific type of Trojans will fail. We assume that the adversary will insert the most difficult to detect Trojan that is likely to pass typical manufacturing tests and validation. At the same time we assume that the adversary is likely to insert the same Trojan into every fabricated chip, since inserting Trojans into a subset of chips requires additional masks and is very expensive [13]. Hence, our methodology does not require activation of the Trojan and does not assume any specific type of Trojan. Instead, we capture characteristics of the most difficult type of Trojans for our approach to detect, i.e., Trojans that minimally

alter the original circuit's delay. And we identify a set of *surrogates* that includes the minimal change in the delay of the original circuit caused by the most difficult type of Trojans to detect, in order to derive the most conservative and yet a general Trojan model.

Second, increasing levels of process variations make it more difficult to detect Trojans, since the amount of extra delay induced by a minimally-invasive Trojan becomes smaller in magnitude compared to the impact of process variations [14]. To overcome this challenge, our approach exploits a fundamental difference between the effects of process variations on delay and the additional delay induced by a Trojan: process variation has random (*bi-directional*) effect on delay, while a Trojan always changes the total delay in a fixed direction (*uni-directional shift*). This fundamental difference causes a change in the delay distribution as depicted in Figure 3(a). Hence, the change in the distribution can always be detected if we measure delays for a sufficient number of chips. In Section 3, we present a more detailed analysis of this observation. Based on this observation, the next question is: Which paths and vectors should we choose to detect this change effectively while maximizing the test resolution and minimizing the number of chips to be tested?

Hence, we develop a path selection scheme for a target Trojan. 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 [13], our path selection scheme targets paths having the *smallest* path delay values to 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 at a desired level of confidence and at a minimum cost.

We have also developed a new hypothesis testing method based on likelihood-ratio test that improves the resolution of Trojan detection while minimizing test cost. The new hypothesis testing method decides whether a target Trojan in the circuit exists or not based on measured delay values from fabricated chips. The effectiveness of our approach is demonstrated using an industrial 65nm technology for high levels of process variations provided by a foundry and benchmark circuits.

The rest of the paper is organized as follows. Section 2 introduces our approach for characterization of Trojans as a set of surrogates that capture necessary conditions that every Trojan satisfies. In Section 3, we propose our approach to improve the resolution of the test by targeting shorter paths. Section 4 presents test generation procedure for paths having the smallest delays. In Section 5, 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 6 and present experimental results in Section 7. Finally, conclusions are drawn in Section 8.

2. 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. To improve completeness of our models of Trojans, we propose a new approach for capturing the necessary characteristics of Trojans.

Our method focuses on detecting Trojans by measuring path delays due to the several advantages of delay measurements discussed in Section 1. To ensure that we evaluate our Trojan detection method under the most challenging conditions, we assume that our adversary has designed Trojans that will cause minimal changes in circuit delays. Hence, in this paper we characterize all possible Trojans by deriving a set of necessary conditions that any Trojan must satisfy in terms of minimal impact on delays.

Every sequential circuit consists of combinational logic blocks and flip-flops. Since making too many changes to a logic block in a given design (referred to as the original design) will change delays of many paths, we focus on the alternative where the adversary designs its logic as a separate *Trojan block*, as shown in Figure 1. (We assume that a Trojan block can be spatially distributed, placed in the unused areas within and between logic blocks and under interconnects, but we depict it in a simplified form as a single block in Figure 1.) It is necessary for such a Trojan block to have at least one connection with the original logic blocks, e.g., line x in block C1 in Figure 1. In absence of at least one connection with the original logic, a Trojan will be totally harmless as it will (a) not affect the operation of the original block in any manner, and (b) not able to copy any values from the original logic. We call a line in the original block that has a connection with the Trojan block e.g., line x in block C1 in Figure 1, a *Trojan site*.

In this context, two cases are possible. Case-1: the adversary inserting the Trojan redesigns the original logic block to hide the impact of the additional delay of this connection, e.g., redesigns C1 in Figure 1 to hide the impact of the connection at line x. Case-2: the adversary leaves the original block unchanged but makes the connection to the Trojan block at a Trojan site in the original block, e.g., line x in Figure 1, in a manner that minimally changes the delay at that line.

In this paper we focus on Case-2, since Case-1 will change delay values of many paths and will be relatively easier to detect. (A demonstration of the ease of detection of Trojans in Case-1 is a subject of our ongoing research.)

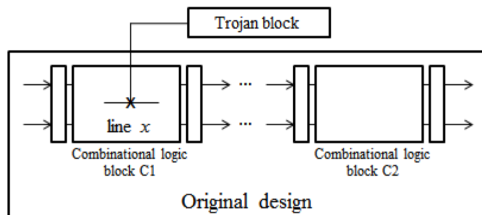

Figure 1: Trojan block connected to the original design

The above reasoning provides us with a set of Trojans with the following characteristics.

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

2) This connection may use the value at the Trojan site, say line *x*, in the original circuit block to drive an input line of the Trojan block. In the least intrusive case, this will take the form of an *additional fanout of minimum load* at line *x*, which is the most difficult type of Trojan for our approach to detect. This will cause a small additional delay at line *x*. Alternatively, this connection may be used to alter the logic function implemented by the original logic block by injecting a value from an output of a Trojan block at the Trojan site. This will require either insertion of another gate at the Trojan site or modification of an existing gate. Again, this will cause a small additional delay at line *x*.

We capture the above two conditions using *a set of surrogate Trojan targets* which contains one surrogate at each line *x* in the original circuit, where the surrogate at line *x* represents a minimal additional delay at that line. Under this model, for an original logic block with *m* lines, the set of surrogates will have *m* surrogate targets.

In the next section we introduce our approach for selecting paths that pass via surrogate Trojans and generating vectors for Trojans that satisfy the above necessary conditions.

3. Impact of Trojans on Delay

Based on the basic properties of surrogates, we propose a path selection scheme that maximizes the impact of Trojans on delay. We also describe the models used by our scheme, including process variations and delay measurements.

3.1 Process Variations

First, 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. We consider inter-die as well as intra-die process variations. 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 (L_{eff}, V_{th}, t_{ox}, etc.) [15].

3.2 Delay Measurements

To detect a target surrogate, we apply a test vector that we generate to excite a selected path via the target surrogate. In some ways this is similar to classical delay testing. However, in contrast to classical delay testing here we are trying to capture the increase in the path delay caused by the Trojan. As discussed in Section 2, we assume that the Trojan induces a minimum additional load at a line in the circuit and hence it minimally alters the delays of paths that pass via the surrogate site. Thus, it is important to precisely measure path delays to capture suitably small differences in delay, around 8ps in our 65nm technology. Also, to increase the relative importance of the Trojan's impact on the delay, we test paths with short delays. While postponing the discussion of why and how we select paths for testing to the next subsection, here we focus on how we apply vectors and measure responses to serve our purpose.

We focus on how to make measurements so we can (1) measure small differences in delays while considering the capabilities of available measurement approaches, and (2) eliminate any concerns about excessive heat dissipation that might otherwise occur when testing focuses on short delay paths [13].

In modern ICs, scan registers are connected to the inputs and outputs of each combinational logic block to apply test vectors and capture response values. Hence we try to measure delays by modifying the architecture and the structure of a scan register. For example, a path delay measurement architecture using a shadow register to measure register-to-register delay is introduced in [8][16]. In this approach, a scan-register connected to each output pin includes an additional shadow register operated by a separate *shadow* clock to measure the path delay at the primary outputs. This approach provides path delay value by comparing logic values captured at both registers while controlling the skew size. Recent approaches, e.g., the one proposed in [17], use digitally variable resistors to control the skew size to a resolution of 1ps. While this resolution is sufficient for our purposes, we can further improve the resolution by introducing two different clocks spaced with controllable skew steps. In Figure 2, the first clock controls the scan-register at an input pin of a combinational path and the second clock which is skewed by Δ is applied to the output scan register. Now we use a combination of multiple slow clocks followed by one fast clock (a slight modification of the classical approach used for path delay testing [18]) and capture logic values for different skew sizes. Note that our approach does not require high frequency clocks and hence avoids all problems associated with excessive heat dissipation during testing of short delay paths.

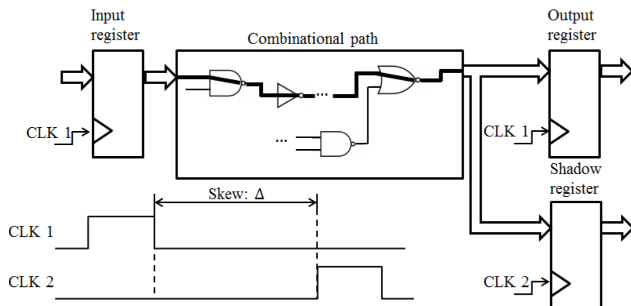

Figure 2: Path delay measurement architecture.

3.3 Path Selection: Targeting Shortest Paths

The total delay of a path p can be expressed as the sum of three parameters as shown below:

$$d = d_0 + \Delta d_{variation} + \Delta d_{Trojan},$$

where three parameters are (1) nominal delay of path p, d_0, (2) the effect of process variations on the delay of p, $\Delta d_{variation}$, which is the overall effect of inter- and intra-chip variations on the path delay, and (3) extra delay induced by a surrogate at a site, say line x, along path p, Δd_{Trojan}. Among these three parameters, *the effect of process variation follows random distribution, which is typically **bi-directional** around a mean*, e.g., normal or truncated normal distribution [19]. *In contrast, extra delay induced by surrogate 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 surrogate, the delay of the gate/line at the surrogate site increases.

This observation is one foundation of our approach for Trojan detection. In fact, by itself this observation enables us to *prove that a minimal-delay surrogate can always be detected, provided that we make measurements on a sufficient number of chips for specifically generated vectors*.

The next question is: How can we minimize the cost of testing for Trojan detection? We address this problem by making a second observation: *The greater the uni-directional increase due to a surrogate compared to the standard deviation for the delay of path due to process variations, the smaller the number of chips that need to be tested.* So to maximize the impact of a surrogate, *we select the path with the smallest delay that passes via the surrogate site*, since the standard deviation of variations is approximately proportional to the nominal delay of the path. 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 surrogate site in the s420 benchmark circuit. We perform Monte Carlo simulations to obtain delay values shown in Figure 3.

The distribution of delay for the original version of the s420 benchmark circuit for a specific vector is shown by the darker curve in Figure 3(a). We then obtained a version of this circuit with a surrogate by inserting an additional fanout at line 371 of the circuit, where the additional fanout was configured to drive the input of an inverter with minimum-sized transistors. We then repeated the simulations to obtain the delay distribution shown by the lighter curve in Figure 3(a). Note that the surrogate 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.

For the same circuit and the same surrogate, we repeated the simulations for a different vector that was selected to excite a much shorter path that passed via the surrogate's site, namely line 371. It is easy to see in Figure 3(b), that while the expected value of the additional delay due to the surrogate remains at around 8ps, i.e., the same level as in Figure 3(a), the impact of the surrogate increases significantly as a percentage of the average delay of the path. Surrogate's impact also increases with respect to the values of the variance due to process variations.

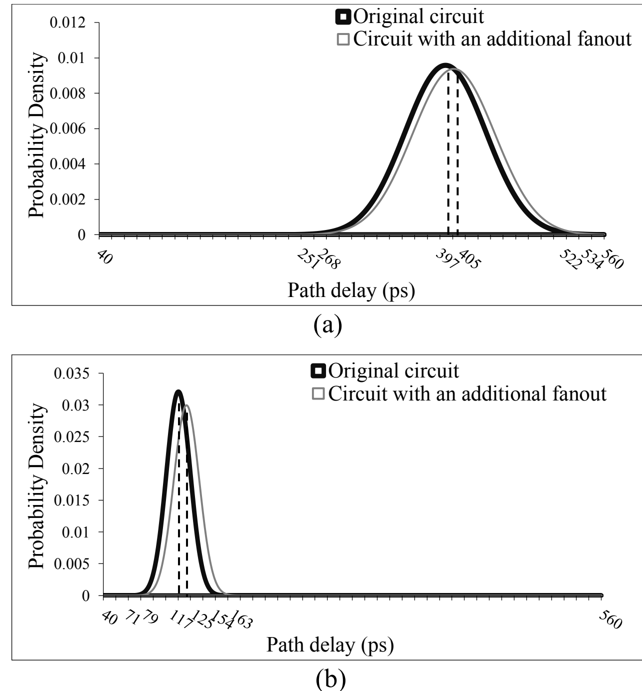

Figure 3: The distribution of delay at an output of benchmark circuit s420 for a particular vector, considering realistic process variations for the original circuit version, and a version with an additional fanout driving a minimum load at a surrogate site along the path (line 371). For a vector that excites (a) a long path, and (b) a shorter path.

The idea of targeting the shortest paths is also useful due to the fact that a surrogate always increases the total path delay. The conventional delay testing method targets the longest paths and captures logic values at primary outputs. Thus any logic values which arrive at primary outputs later than the desired clock period will produce errors. Even though the size of additional delay due to the surrogate is very small, it might be detected if it increases delay of any longest path and the path delay goes beyond the clock period. Since the adversary is aware of every kind of conventional testing method, they will try to insert a Trojan to paths other than the longest paths to avoid detection.

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 exists is greater for a shorter path.

Hence, in order to detect a surrogate at a specific site, say line x, using path delay measurement, we select the

shortest path passing via line *x*. 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, and m is the number of lines in the circuit. As we select a surrogate path for each surrogate target, the total number of surrogate paths is only proportional to the number of lines in the circuit. Testing of all possible surrogates using surrogate paths in the circuit only needs $O(m)$ time.

For each of these paths, we must generate an appropriate vector that excites the path. This requires a very different set of test generation conditions from the classical delay testing which typically focuses on the longest paths [18]. Later in this paper, we derive our conditions and compare our approach with the classical approach of using vectors that excite the longest delay paths and show dramatic benefits.

4. Conditions for Generating a Test for a Trojan

Our next task is to generate a vector that invokes the delay of a given surrogate. 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.

For any path selected as a surrogate path, we have derived conditions that must be satisfied by a vector to ensure that our objective is satisfied. Figure 4 shows the conditions that must be satisfied by a vector generated for path delay testing. In contrast, Figure 5 shows the conditions derived for generating a vector for a surrogate path, for an on-path NAND gate. For the first case, where the on-path input of the NAND gate has a falling transition, delay testing as well as surrogate 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.

Figure 4: 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 5: Conditions for a NAND gate along a surrogate path, to detect above category of surrogates.

In the second case, where the on-path input of the NAND gate has a rising transition, robust testing only focuses on invoking delay that that is equal to or greater than the delay of the path [18]. Since in surrogate detection our goal is to invoke the delay of the target path, we must modify these conditions to preclude cases where a late transition at an off-path input invokes delay greater than that 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.

5. Estimating the Number of Chips to be Tested

In this section, we formulate the problem of determining whether a surrogate exists or not, based on measured delay values for selected paths and specifically generated vectors. Given a circuit design *C* and the values of delay parameters and variations, we apply a vector we generate, *V*, to a number, *n*, of fabricated chips provided by the foundry for design *C*. For each chip, we measure the delay at an output with the goal of detecting a particular instance of the surrogate. *What is the minimum value of n sufficient to provide the correct disposition of the fabricated chips – either "surrogate-free" or "has the particular surrogate" – with low probability of decision error?*

We have developed a method to solve the above problem using hypothesis testing. There are two main categories of the hypothesis testing methods: 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 or mean-squared error (MSE). 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 tests are not based on strong distribution assumptions and they can be performed on any kinds of fully-specified distributions. But these tests require the basic assumption that a large number of samples exist. In addition, non-parametric tests require higher numbers of samples to achieve a certain level of confidence than parametric tests [20]. Finally, our problem is to select a more *likely* model between two competing models, "surrogate-free" and "circuit with the target surrogate". However, existing hypothesis testing methods make decisions whether the data follows a certain distribution or not. Thus any existing test cannot serve as a proper method for solving our selection/classification problem.

Instead, we propose a more efficient non-parametric test to identify the existence of the target surrogate using likelihood-ratio test. It utilizes every single delay value of a path measured from fabricated chips using a vector and computes the exact conditional probabilities for both "surrogate-free" and "circuit with the target surrogate" models. Because our goal is to choose a more probable model between these two competing models, we use the

likelihood-ratio between conditional probabilities to make a decision. We also derive equations that estimate the number of chips required for measurements for a given vector. To show the effectiveness of our new approach, we choose Student's t-test as a baseline method because Student's t-test requires less number of chips to be tested than non-parametric methods for the same level of confidence. Later, we will compare the number of chips to be tested required by both methods, Student's t-test and our likelihood-ratio based test, for the same level of confidence.

5.1 Baseline for comparison: Student's t-test Method

Given the level of confidence ρ and the number of fabricated chips n, we determine the confidence interval for the mean value of the path delay μ obtained by measurements on n fabricated copies of the chip. Suppose the mean delay derived via simulation for the same vector, for circuit versions without and with the surrogate are μ_A and μ_B, respectively. Let σ be the standard deviation of the delay values obtained via simulations for both circuit versions. Then we set null hypothesis and its corresponding counter hypothesis as follows.

H0: $\mu = \mu_A$, i.e., target surrogate does not exist,
H1: $\mu > \mu_A$, i.e., target surrogate does exist.

z-value for testing H0 is computed as follows.

$$z_1 = \frac{\mu - \mu_A}{\frac{\sigma}{\sqrt{n}}}. \quad (1)$$

H0 is rejected if $|z_1| > z_\rho$ (surrogate is detected), where z_ρ is z-value for level of confidence ρ. Otherwise H0 is not rejected (surrogate is not detected). By conducting hypothesis testing with n chips, a decision error may occur *when hypothesis testing falsely identifies a chip as having the surrogate while the surrogate does not exist* (Type-I error, where probability of occurrence is (100- ρ)%), or *falsely identifies a chip as the surrogate-free chip while the surrogate does exist* (Type-II error). Since it is dangerous to let a chip containing the surrogate to pass the test, Type-II error should be avoided. The probability of Type-II error, β, is

$$\beta = P[\text{H0}|\text{H1}] = \Phi\left(z_\rho - k\sqrt{n}\right) - \Phi\left(-z_\rho - k\sqrt{n}\right), \quad (2)$$

where k is the expected value of additional delay due to the surrogate with respect to the value of standard deviation σ. The probability of decision error decreases as the value of n increases and the value of k is bigger. The minimum value of n is given by Eq. 2 depending on the value of k and β and with the level of confidence ρ.

5.2 Our Approach Based on Likelihood-Ratio Test

Suppose that fabricated chips follow some unknown distribution F and we have *i.i.d.* delay values (data points) measured from n samples from fabricated chips $X_1 \dots, X_n$ using a test vector. For a given test vector, we have a probability distribution A and its probability density function f_A for delay values of the original circuit, and B and f_B for the circuit with the target surrogate. The likelihood-ratio test decides between the following hypotheses.

H0: $F = A$, i.e., target surrogate does not exist,
H1: $F = B$, i.e., target surrogate does exist.

The entire sample space is divided into r mutually-exclusive intervals and each interval has the size Δ, except the first and the last interval as we divide the entire sample space $[-\infty, +\infty]$ into r intervals. And we compute p_{k0} and p_{k1}, the conditional probabilities that one data point belongs to the interval k given that either one of two above hypotheses is true.

$$p_{k0} = P[X_1 \in (interval\ k)|\text{H0}], k = 1, \dots, r. \quad (3)$$

For n data points and r intervals, every data point belongs to one unique interval. And numbers of samples that belong to the interval $k = 1, \dots, r$ are $n_1, \dots, n_r$, respectively, where $\sum_{i=1}^{r} n_i = n$. Let the event $\boldsymbol{X}$ be a combination of $n_1, \dots, n_r$, i.e., $\boldsymbol{X} = \{n_1, \dots, n_r\}$. The number of possible combinations of $\boldsymbol{X}$ is $M = \binom{n+r-1}{n}$ and the conditional probability of the event $\boldsymbol{X}$ given that H0 is true is

$$p(\boldsymbol{X}|\text{H0}) = \prod_{k=1}^{r} p_{k0}^{n_k} \cdot \frac{n!}{n_1!\, n_2! \dots n_r!}. \quad (4)$$

The expression for p_{k1} and $p(\boldsymbol{X}|\text{H1})$ can be obtained in a similar way.

The likelihood ratio test statistic can be written as

$$\Lambda(\boldsymbol{X}) = \frac{p(\boldsymbol{X}|\text{H0})}{p(\boldsymbol{X}|\text{H1})}, \quad (5)$$

where $\Lambda(\boldsymbol{X})$ is a ratio between $p(\boldsymbol{X}|\text{H0})$ and $p(\boldsymbol{X}|\text{H1})$ and is called a likelihood-ratio of the test.

In the above equation, the likelihood ratio $\Lambda(\boldsymbol{X})$ is small if the alternative hypothesis H1 is better than the null hypothesis H0. It provides the decision rule as below.

If $\Lambda(\boldsymbol{X}) < c$, do not reject H0,
otherwise, reject H0,

where c is a threshold value.

In addition, the probability of Type-II error, β, is computed as

$$\beta = \sum_{\Lambda(\boldsymbol{X}) \geq c} p(\boldsymbol{X}|\text{H1}). \quad (6)$$

β represents the sum of conditional probabilities given that H1 is true when the test decides to accept H0 (i.e., when $\Lambda(\boldsymbol{X}) \geq c$). Hence, the probability of Type-II error can be controlled by adjusting the threshold value c. This problem can be expressed as integer linear programming (ILP) shown below:

Objective: minimize n for H0 and H1
Constraints: $c \geq 1$,
$\beta_{max} \geq \sum_{\Lambda(\boldsymbol{X}) \geq c} p(\boldsymbol{X}|\text{H1})$,
$\alpha_{max} \geq \sum_{\Lambda(\boldsymbol{X}) < c} p(\boldsymbol{X}|\text{H0})$,

where α_{max} and β_{max} are maximum values of Type-I and II error probabilities allowed for the test.

The main advantage of the proposed method is that it does not require the distribution of measured data points to be a general distribution such as normal distribution, in contrast to Student's t-test. The conditional probability of each interval is computed based on the probability distribution that results from process variation, where the actual distribution is not Gaussian. *Thus, the proposed method is applicable to detect a surrogate under any type of process variations.* Also, recall that despite its applicability, we use Student's t-test as a baseline for comparison since it requires fewer chips than most non-parametric methods.

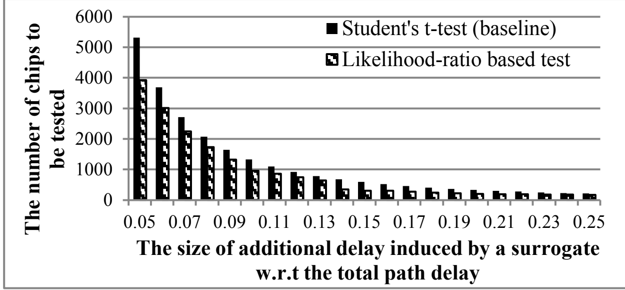

Figure 6: The number of chips to be tested using Student's t-test and likelihood-ratio based test for different sizes of additional delay induced by a surrogate with respect to the total path delay.

The proposed likelihood-ratio based test dramatically reduces the number of chips to be tested compared to Student's t-test. Figure 6 shows the number of chips to be tested using the above two methods, Student's t-test and likelihood-ratio based test, for different sizes of additional delay induced by a surrogate with respect to the total path delay with the fixed sizes of desired Type-I and II error probabilities of 5% and 5%, respectively. It clearly shows that the number of chips to be tested increases as the percentage of the amount of the extra delay induced by a surrogate to the total path delay decreases. For the same path and surrogate, 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%. It 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.

6. Our Trojan Detection Algorithm

Using the surrogate detection test procedures and test vector generation method discussed above, we have integrated and implemented all our above results as a single framework. Figure 7 provides a high-level overview of our integrated approach. The algorithm generates surrogate paths and computes the minimum number of chips to be tested for every surrogate. 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 1. In Steps 2 and 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. Surrogate targets and a set of detectable surrogates are updated depending on the result of test generation. In Step 4, the algorithm computes the number of chips required for the test to detect every detectable surrogate. We use surrogate coverage to show the effectiveness of our approach.

```
1.    Initialize P_i = NULL (surrogate path i), S = ∅ (set of
      detectable surrogates), P = ∅ (set of surrogate paths) and
      S_{P_i} = ∅ (subset of surrogates that are detectable by
      surrogate path i)

      for every line in the circuit i = 1,…, m, enumerate every
      path passing via the target line i.
      Sort paths in increasing delay order and add them into S_i
       begin
2.        while test generation is successful or S_i ≠ ∅
              Choose the path k having the smallest delay in S_i
              Generate a test vector for corresponding path k
 2-1.         if (test generation is successful) then
                  Update P_i = k
                  Add P_i to the set of surrogate paths, P
                  Add line i to S
                  Add every line along the path P_i to S_{P_i}
                  break
 2-2.         else then
                  Remove k from S_i
              end if;
          end loop;
3.        if (P_i = ∅) then
              Surrogate path for line i does not exist
          end if;
       end loop;
4.    for every surrogate i in every non-empty S_{P_k}
        begin
5.        Compute the number of fabricated chips, n_{i,k},
            to be applied to P_k to detect surrogate i.
        end
    Surrogate coverage = |S|/m ×100 (%)
```

Figure 7: Trojan detection algorithm for generating vectors for surrogate paths to detect surrogates.

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

Objective: minimize $|T|,\ T \subseteq P$

Constraints: $\sum_{P_j \subseteq T} x_{i,j} \geq 1$ for $1 \leq i \leq m$

$x_{i,j} = 1$, if $i \in S_{P_j}$ and $P_j \in P$

$x_{i,j} = 0$, if $i \notin S_{P_j}$ or $P_j \notin P$,

where $x_{i,j}$ is an indicator that shows whether surrogate i is detectable using surrogate path j ($x_{i,j} = 1$) or not ($x_{i,j} = 0$) and T is a minimal set of surrogate paths to be used for

testing. The above problem can be solved using greedy heuristics and test cost is computed as $\sum_{P_j \in T} (\max_i n_{i,j})$ and it is sum of the number of required chips to be tested for every surrogate path in *T*.

7. Experimental Results

We have used our prototype tool to demonstrate the effectiveness of our approach. We have also compared our approach with an adaptation of the existing delay testing approach to demonstrate the dramatic improvement in effectiveness (surrogate coverage) and dramatic decrease in test cost.

For our experiments, we use the combinational parts of eight ISCAS89 benchmark circuits. Path delay values are measured from Monte-Carlo simulations using Spectre for an industrial 65nm technology while using levels of process variations provided by a fabricator, as described in Section 3. The ATPG tool for the proposed test generation procedure has been implemented in C. As a surrogate which induces minimum load to a line in the circuit, we use a minimum-sized inverter as an extra fanout that induces extra delay at each particular surrogate site. After we obtain delay values via simulations under process variations, we apply two different hypothesis testing methods, Student's t-test and our likelihood-ratio based test, to compare test costs required by these two methods.

In Table 1, we showed the number of surrogate paths required to achieve the maximum surrogate coverage using our approach. Our approach significantly reduces the number of path delays to be measured since it only focuses on the shortest path passing via each possible surrogate site.

Table 1: Maximum surrogate coverage and number of surrogate paths required for selected benchmark circuits.

Benchmark circuit	Number of circuit lines (m)	Maximum surrogate coverage (%)	Surrogate paths/ Total paths
c17	17	100	8/22
s298	298	72.8	112/462
s368	368	84.2	162/414
s420	420	69.3	131/738
s510	510	94.9	247/738
c880	880	78.9	272/17284
s1488	1488	69.5	365/1924
s5378	5378	67.5	844/27084

Figure 8 shows the number of chips to be tested for each surrogate path P_i with our approach and classical delay testing method. It can be seen that *using shortest paths as surrogate paths we significantly reduce the required number of chips for every possible surrogate site* in c17. In addition, *our hypothesis testing method is significantly more effective than Student's t-test* since it chooses one likely model between two competing models, "surrogate-free" and "circuit with the surrogate", where Student's t-test focuses on the closeness of measured delay values to the surrogate-free model.

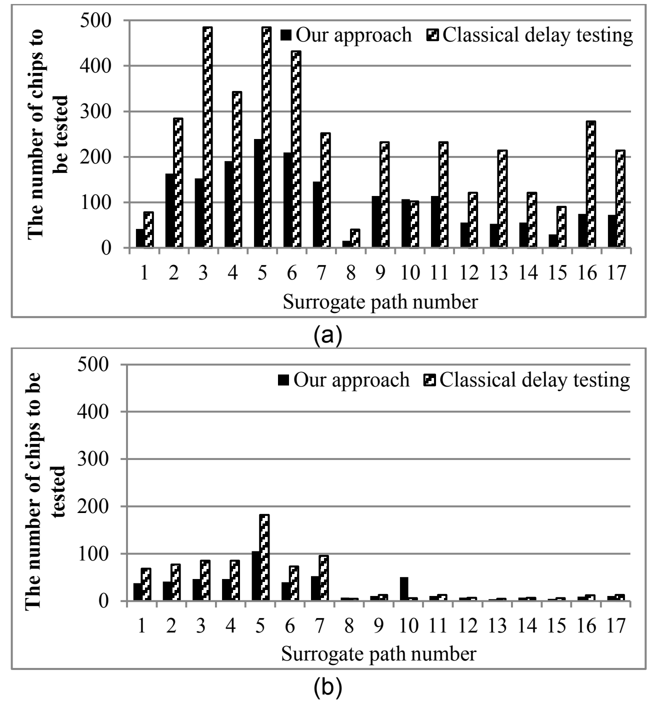

Figure 8: The number of chips to be tested for surrogate paths in c17 with our approach and classical delay testing method. (a) Student's t-test. (b) Likelihood-ratio based test.

Table 2 compares the cost for eight benchmark circuits of four different methods: (i) Classical delay testing method with Student's t-test, (ii) Our approach with Student's t-test, (iii) Classical delay testing method with likelihood-ratio based test, and (iv) Our approach with likelihood-ratio based test. In each method, we use 95% confidence level and 5% Type-II error probability. Note that method (i) is the classical method, method (iv) is the proposed method, and other two are intermediate methods. In order to show improvement in the quality of the test, we also compute test costs required for various surrogate coverage levels for all the four methods. The maximum values of surrogate coverage for four methods are the same, because the maximum surrogate coverage is determined by the existence of a testable surrogate path for each surrogate site. However, our new approach with likelihood-ratio based test (method (iv)) dramatically improves 3.48X in the test cost compared to the existing delay testing targeting the longest paths with Student's t-test (method (i)), for the same value of surrogate coverage. For sensitive chips that are fabricated in small volumes, our approach may be the only one that can perform Trojan detection with desired level of confidence. 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 incorporated with our hypothesis testing method reduces test cost significantly.

Table 2: The cost and surrogate coverage (SC) for benchmark circuits for different values of surrogate coverage. (i) Classical delay testing method with Student's t-test. (ii) Our approach with Student's t-test. (iii) Classical delay testing method with likelihood-ratio based test. (iv) Our approach with likelihood-ratio based test.
*** Improvement in test cost: {Test cost of method (i)}/{Test cost of method (iv)}**

c17

SC (%)	Test cost (i)	(ii)	(iii)	(iv)	Improvement in test cost
76.5	68	55	43	47	1.45X
82.4	179	101	115	61	2.93X
88.2	396	311	227	188	2.11X
94.1	619	393	355	230	2.69X
100	960	470	542	275	3.49X

s298

SC (%)	Test cost (i)	(ii)	(iii)	(iv)	Improvement in test cost
55	10793	3588	5848	2110	5.12X
60	17097	6750	10062	3804	4.49X
65	28302	10305	17589	5649	5.01X
70	48754	14874	30629	8736	5.58X
72.8	63950	18278	40934	10706	5.97X

s386

SC (%)	Test cost (i)	(ii)	(iii)	(iv)	Improvement in test cost
65	87742	46047	55712	34168	2.57X
70	102814	63311	67190	45473	2.26X
75	124239	81067	90358	60573	2.05X
80	155495	100089	130320	87871	1.77X
84.2	187199	115550	156000	109600	1.71X

s420

SC (%)	Test cost (i)	(ii)	(iii)	(iv)	Improvement in test cost
50	32723	5439	19531	2980	10.98X
55	41423	9141	24616	4667	8.88X
60	66715	16084	38396	9222	7.23X
65	95771	22545	59570	14587	6.57X
69.3	228157	110054	117080	55859	4.08X

s510

SC (%)	Test cost (i)	(ii)	(iii)	(iv)	Improvement in test cost
75	73327	23777	44207	13780	5.32X
80	97323	30827	55348	17805	5.47X
85	158462	45302	93494	28137	5.63X
90	232830	70233	140860	41552	5.60X
94.9	327474	94835	191330	57688	5.68X

c880

SC (%)	Test cost (i)	(ii)	(iii)	(iv)	Improvement in test cost
60	35079	52547	22452	34634	1.01X
65	79133	57215	45503	39752	1.99X
70	143292	120899	95549	76798	1.87X
75	255270	259200	157480	130950	1.95X
78.9	345825	332553	192440	175740	1.97X

s1488

SC (%)	Test cost (i)	(ii)	(iii)	(iv)	Improvement in test cost
50	239100	60443	148320	37738	6.34X
55	299358	88430	195980	54385	5.50X
60	397046	134034	265950	77942	5.09X
65	480751	227759	311680	140760	3.42X
69.5	683190	366669	385610	223510	3.06X

s5378

SC (%)	Test cost (i)	(ii)	(iii)	(iv)	Improvement in test cost
50	762350	452831	521020	382930	1.99X
55	842330	659029	732930	469280	1.79X
60	930023	782729	803820	523920	1.78X
65	1232523	837293	837260	653840	1.89X
67.5	1442322	1083721	1102340	769280	1.87X

8. Conclusions

We have developed new principles for characterization of Trojans inserted by intelligent adversaries using our new notion of surrogates. We have also identified several principles for selection of target paths and generation of vectors to enable identification of surrogates in presence of increasing levels of process variations, that cannot be detected by classical testing and validation approaches. The experimental results show that the proposed approach reduces test cost significantly compared to classical methods.

We are in the process of applying our principles to identify other surrogate sets of Trojans and developing an extensive set of effective and efficient approaches for their detection.

9. References

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

[2] S. Jha and S. K. Jha, "Randomization Based Probabilistic Approach to Detect Trojan Circuits," *High Assurance Systems Engineering Symposium (HASE)*, 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] X. Wang, H. Salmani, M. Tehranipoor and J. Plusquellic, ''Hardware Trojan Detection and Isolation Using Current Integration and Localized Current Analysis," *IEEE International Symposium on Defect and Fault Tolerance of VLSI Systems (DFT)*, 2008, pp. 87-95.

[5] 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 (HOST)*, 2008, pp. 40-47.

[6] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi and B. Sunar, "Trojan Detection using IC Fingerprinting," *Symposium on Security and Privacy*, 2007, pp. 296-310.

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

[8] J. Li and J. Lach, ''At-Speed Delay Characterization for IC Authentication and Trojan Horse Detection,'' *IEEE International Workshop on Hardware-Oriented Security and Trust (HOST)*, 2008, pp. 8-14.

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

[10] M. Tehranipoor and F. Koushanfar, "A Survey of Hardware Trojan Taxonomy and Detection," *IEEE Design & Test of Computers*, 2010, pp. 10-25.

[11] X. Wang, M. Tehranipoor, and J. Plusquellic, "Detecting Malicious Inclusions in Secure Hardware: Challenges and Solutions," *IEEE International Workshop on Hardware-Oriented Security and Trust (HOST)*, 2008, pp. 15-19.

[12] R.S. Chakraborty, S. Narasimhan and S. Bhunia, "Hardware Trojan: Threats and emerging solutions," *High Level Design Validation and Test Workshop (HLDVT)*, 2009, pp. 166-171.

[13] M. Tehranipoor, H. Salmani, X. Zhang, X. Wang, R. Karri, J. Rajendran and K. Rosenfeld, "Trustworthy Hardware: Trojan Detection Solutions and Design-for-Trust Challenges," *IEEE Computer Magazine*, Vol. 44, Issue 7, 2011, pp. 66-74.

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

[15] 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.

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

[17] 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.

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

[19] F. Liu, "A general framework for spatial correlation modeling in VLSI design," *Design Automation Conference. (DAC)*, 2007, pp. 817-822.

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