IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received June 1, 2021, accepted June 13, 2021, date of publication June 28, 2021, date of current version July 6, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3093160

AVATAR: NN-Assisted Variation Aware Timing Analysis and Reporting for Hardware Trojan Detection

ASHKAN VAKIL[1], ALI MIRZAEIAN[1], HOUMAN HOMAYOUN[2], NAGHMEH KARIMI[3], (Member, IEEE), AND AVESTA SASAN[1], (Senior Member, IEEE)

[1]Department of Electrical and Computer Engineering, George Mason University, Fairfax, VA 22030, USA
[2]Department of Electrical and Computer Engineering, University of California at Davis, Davis, CA 95616, USA
[3]Department of Computer Science, University of Maryland Baltimore County, Baltimore, MD 21250, USA

Corresponding author: Ashkan Vakil (avakil@gmu.edu)

This work was supported by the Defense Advanced Research Projects Agency Defense Advanced Research Projects Agency-Air Force Research Lab (DARPA-AFRL) under Grant FA8650-18-1-7820.

This work did not involve human subjects or animals in its research.

ABSTRACT This paper presents AVATAR, a learning-assisted Trojan testing flow to detect hardware Trojans placed into fabricated ICs at an untrusted foundry, without needing a Golden IC. AVATAR is a side-channel delay-based testing solution that is assisted by a learning model (process watchdog) for tracking the process drift and systematic process variation. AVATAR's process watchdog model is trained using a limited number of test samples, collected at test time, to tightly correlate the Static Timing Analysis results (generated at design time) to the test results (generated from clock frequency sweeping test). The experimental results confirm that AVATAR detects over 98% of (small) Trojans inserted in the selected benchmarks. We have complemented our proposed solution with a diagnostic test that 1) further reduces the false-positive rate of AVATAR Trojan detection to zero or near zero, and 2) pinpoints the net-location of the Trojan Trigger or Payload.

INDEX TERMS Hardware trojan, clock frequency sweeping test, neural network, side channel analysis, process variation, process drift.

I. INTRODUCTION

To reduce the fabrication cost, scale with market demand, and access to the state-of-the-art technology, the manufacturing supply chain of Integrated Circuits (IC) is widely and globally distributed [1]. Such broad globalization has raised many concerns over the security and trustworthiness of ICs when untrusted providers and facilities are included in the supply chain [2]. One such security concern is the risk of Hardware Trojan (HT) insertion by an adversary in the supply chain. An HT is a malicious modification to a circuit to control, modify, disable, or monitor it's logic or leak sensitive data [2]. The concerns over Hardware Trojan was raised by US DoD in 2005 [3] The spectrum of harm caused by HT is broad. It can range from passive HT for activity monitoring or stealing information to weaponized HT that could cause catastrophic consequences in the critical military, space, or medical applications [4]. Moreover, many recent studies [5]–[7] explained the feasibility of such attacks adding to the concern. An example of such study is the A2 Trojan using a single gate in [8], illustrating how an adversary could weaponize this single gate Trojan for raising her user privilege from an ordinary to root privilege through a sequence of rare yet well-defined operation (e.g., sequence of divide by zeros). In addition, several military mishaps in the past are attributed to the presence of hardware modification [9]–[11]. Thereby, detecting HT is highly crucial, and it has become a significant concern for governments and industries.

One solution for detecting HT is through destructive reverse-engineering schemes to check and ensure that the manufactured chips' logical structure and functional integrity is untouched. However, IC reverse engineering requires significant investment, is extremely challenging in advanced geometries, and is quite a time and resource consuming.

The associate editor coordinating the review of this manuscript and approving it for publication was Tyson Brooks.

One may argue that a Trojan-induced logic-change can be detected during the manufacturing test process. However, HTs are stealthy in nature, and are designed such that they are rarely activated. This makes detecting the HT during manufacturing testing highly difficult if not impossible.

Another approach to detect HT is through comparison with a Golden Model. Such comparison could be made based on dynamic, or leakage power signature [12]–[15] or expected delay of individual timing paths [16]–[18]. However, these side-channel Trojan detection solutions are not applicable if such a Golden IC cannot be obtained or produced. More precisely, in scaled geometries, the number of target foundries is limited to one or only a few. In addition, the process and parametric signatures of the ICs manufactured (in the same technology node) are different across foundries due to the subtle differences in their processing technology.

Considering the shortcoming of the current Trojan detection solutions mentioned earlier, there is an imperative demand for a low-cost, precise, and efficient alternative Golden-IC-free Trojan detection scheme. Accordingly, in this paper, we propose a robust flow for HT detection, denoted as AVATAR, that is based on side-channel delay test and analysis. Our proposed scheme, unlike the previous work [16], does not rely on the availability of a Golden IC. Instead, we use a machine learning approach in which a learning model is trained to predict the slack of each timing path via path features extracted from the design database. In other words, we train a Neural Network model that tracks the non-linear changes in the behavior of timing paths. The learning model is trained at test time using a limited number of delay samples (used as the label for our training set) collected from the Clock Frequency Sweeping Test (CFST). More specifically, the contributions of this paper are as follow:

- Highlighting how process drift (defined as improvement in fabrication process overtime that is not reflected in the SPICE models released to a design house, after the process is available) could result in unseen and unaccounted timing slack that could be exploited to insert stealthy Trojan;
- Proposing a learning-assisted process-tracking watchdog that is trained at test time (using a limited number of test samples collected from Clock Frequency Sweeping Test) to track the impact of process drifts on the delay/slack of each timing-path in the fabricated IC;
- Proposing a Golden-IC-free Trojan detection test flow that uses the neural-assisted process-tracking watchdog to correlate the Static Timing Analysis (STA) results to the test results, and illustrating its effectiveness and superior sensitivity for Trojan detection.
- Proposing a diagnostic solution (which is conducted if our flow detects a Trojan) for 1) reducing the False Positive rate of our proposed Trojan detection solution; 2) locating the HT's possible trigger and payload location in the IC under test, to facilitate and speed up further physical and optical investigations, by removing the need for scanning the entire IC for HT.

The rest of this paper is organized as follow: Section II describes the previous works on HT detection. Section III describes the adversarial HT threat model used in this paper and explains a body of challenges posed on HT detection when using side-channel delay analysis due to process variation and process drift (variability). Section IV describes our proposed solution, AVATAR, in detail and explains how aforementioned challenges are addressed. Section V presents the results, and finally, Section VI concludes this work.

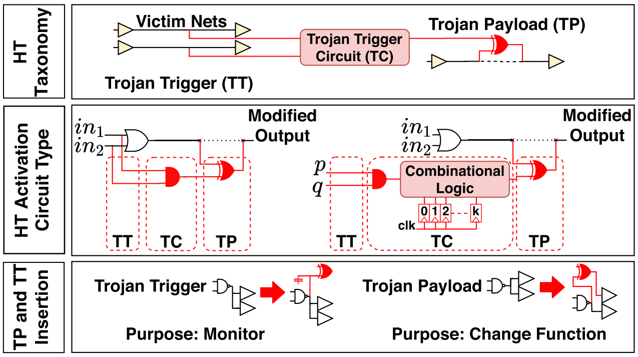


FIGURE 1. **(Up): Trojan taxonomy, (middle): Trojan trigger circuit types, (bottom): Trojan impact.**

II. PREVIOUS WORK ON HW TROJAN DETECTION

In practice, an HT can be inserted at any stage of the design flow [2], [11], [19]–[21]. As depicted in Figure 1, A Trojan mainly consists of 1) Trojan's Trigger (TT), 2) Trojan (sequential or combinational) triggering Circuit (TC), and 3) Trojan Payload (TP). Upon activation of the TC, a Trojan delivers its payload which can result in a denial of service in the whole or part of the circuit, corruption of the circuit's functionality, an alteration in the characteristic of the circuit such as aging factors, or leaking secret information [2], [11].

Countermeasures against HT can be categorized into the design-for-security, run-time monitoring, and detection solutions [11]. The design-for-security approaches opt to reduce the chances of Trojan insertion (e.g., through hardware obfuscation). However, they can neither guarantee a Trojan free IC nor detect them. The run-time techniques monitor the functionality of the IC (usually through snapshots of its operation) at run-time [22], and compare it against known behavior signatures. However, if the Trojan impact does not persist, it does not create the expected signature, or affects the IC's behavior in a way that is not modeled (in the monitoring solution), the monitoring schemes will fail to detect the Trojan. On the other hand, detection approaches aim to directly or indirectly detect the presence of HT. Detection solutions could be destructive or non-destructive [11]. The destructive solutions, that could provide an ultimate proof for Trojan's presence in the selected IC, require full reverse engineering of the IC.

The non-destructive detection approaches can detect the HT by either activating them or via side-channel analysis [2], [11], [23]. The former relies on finding a set of input patterns that trigger the possible Trojan such that the Trojan results in a noticeable impact (e.g., change in expected output). On the other hand, the side-channel based detection methods attempt to identify the Trojan presence through side-channel information obtained from an IC, e.g., power consumption [24]–[27], electromagnetic emanations [28], or path delays [16]–[18].

Detecting HT by activating them during manufacturing test is significantly challenging. In principle, HT are designed to be activated through a rare sequence or combination of events, only known to the adversary [11], [16]. Testing an IC exhaustively using all sets of possible patterns is not practically feasible [16]. Note that not all HT alter the functionality. For example, an HT may be designed to leak secret information (with antenna or noise); making such Trojan immune to activation-based solutions as the functional impact of such HT is not observable.

Trojan activation solutions' limited coverage has encouraged the research to focus on side-channel analysis-based detection techniques. One widely studied Trojan detection direction is through side-channel power analysis that focuses on power consumed by The Trojan Circuit (TC) [12]–[15], [26]. However, Trojan circuitry is small and it induces little change in power that may not be easily differentiated from process variation related change (from one IC to another). Hence, to improve detection, the TC should be partially or fully activated. Therefore it is better suited for HT, trigger of which is connected to shorter timing paths with a higher degree of controllability. At the same time, the power signature of the TC should be significant enough to stand out (make a noticeable change in the power consumption of the IC) as the demanded current of an IC can be monitored with limited precision (through package balls or, at best case, through power delivery networks pads). Hence, the observed current signature is the accumulation of the transistors' current needed for the normal function of the IC and those added for implementation of TC. Therefore, the size of a TC should be large enough to be observable using such techniques.

The delay side-channel test, on the other hand, focuses on the change of the delay and measures path delays to detect a Trojan [29]. The delay analysis proposed in [17] monitors the critical timing-paths to detect HT. However, it fails to consider the near-critical or shorter timing paths. The authors of [18] insert shadow registers to measure the delay of each timing-path. However, this results in a large area overhead. Finally, the solution suggested by [16] uses Clock Frequency Sweeping Test (CFST) to detect HT. However, it relies on the existence of a Golden IC for delay comparison.

There exist a body of prior art solutions that use machine learning to remedy the impact of noise in the power/current profile of side-channel analysis [30]–[32]. Authors in [30] use the Extreme Learning Machine algorithm to detect HT by analyzing the power consumption and observing the resulting current portfolio as an input feature to their machine learning solution. The major limitation in this work is the size of detectable Trojans. According to this paper, the HT size ranges from 0.15%, 0.40%, or 0.81% of the original chip to create a distinguishable current profile. In comparison, AVATAR can detect small hardware Trojans payload or trigger. Hence, it could detect HTs constructed using few or even one gate.

In [31], authors extracted the timing signature of Golden IC and reduced the dimension of attributes using Principle Component Analyzer (PCA). This data is then used to train a Decision Tree, a Bayesian Classifier, and a K-Nearest Neighbor model. After optimizing the models via cross-validation, they determine the trustworthiness of the IC under the test. Authors in [32] evaluated their approach on fabricated chips. They applied PCA and a Support Vector Machine to analyze the transmission power information of the chip. This work only detects HTs that distort the transmission power, and it is not applicable for other cases. Besides, their training depends on the existence of a Golden IC.

AVATAR provides several advantages compare to side-channel power detection solutions including [30]–[32]. 1) It doesn't require access to a Golden IC for Trojan detection. 2) it doesn't require full or partial activation of the Trojan circuit. An inactive Trojan can still be detected using AVATAR as the added delay, either due to added capacitive load of TT or added delay of TP, is permanently present and measurable.

III. TROJAN AND VARIABILITY MODEL

This section first describes our Trojan treat model. It then explains the challenges that process variability and process drift pose to detect HT detection when using side-channel delay analysis.

A. TROJAN THREAT MODEL

We assume that the adversary is an untrusted foundry with access to the Graphic Database System format (GDSII) of the design, aiming at inserting a Trojan that is triggered based on a combination or sequence of rare events. We assume that the Trojan has several Triggers and at least one payload. However, our proposed Trojan detection solution is also applicable to Trojans with no Payload (inserted for monitoring purposes). We further assume that the same HT is inserted in all fabricated dies. We also assume that the foundry can skew the process (making faster transistors) to create available slack for the insertion of HT in desired timing paths without making the overall delay of timing path larger than the delay reported (or expected) by STA at design time. Trojan detection, in our solution, is performed in a trusted facility.

B. TROJAN DETECTION CHALLENGE: VARIABILITY

The TT of the Trojans considered in this study poses an additional capacitive load on its driving cell, resulting in a slower rise and fall, while its TP adds at least one gate delay to its victim timing path. Note that the added delay could be more than a single gate; this is because a large and complex

TC may also affect the delay of timing path hosting the TP if the sum of worse-case trigger sub-path delay and TC delay is larger than the delay of sub-path leading to the TP. In this paper (to address a more challenging scenario), we assume that TC is small, and the increase in the delay of the timing path hosting the TP is limited to a single gate delay.

The AVATAR scheme relies on side-channel delay analysis and detects HT by tracking and analyzing the changes in the delay resulted from tested timing-paths. AVATAR does not rely on the availability of a Golden IC yet relies on the timing model generated using Static Timing Analysis (STA) at the design time. However, the STA data can be significantly different from delay information calculated at the test time. The difference is due to pessimistic margins considered in generating GDSII file to account for various sources of variability including Process Variation and process drift. What follows discusses the sources of such variations, and how they can be exploited by an adversary to insert stealthy HT. Then, in Section IV, we describe the modeling of different sources of variability and how to improve the probability of Trojan detection by mitigating or modeling their impact.

1) RANDOM PROCESS VARIATION

The random process variation refers to the variations in the physical and electrical properties of transistors due to the physical limitations faced during the fabrication process [33]. The random process variation impacts the delay and drive strength of fabricated transistors and makes Trojan detection more difficult as the test engineer needs to differentiate between the delays imposed by random process variation and the timing impact of an HT. Figure 2 illustrates the effect of the random process variation on the slack of timing paths.

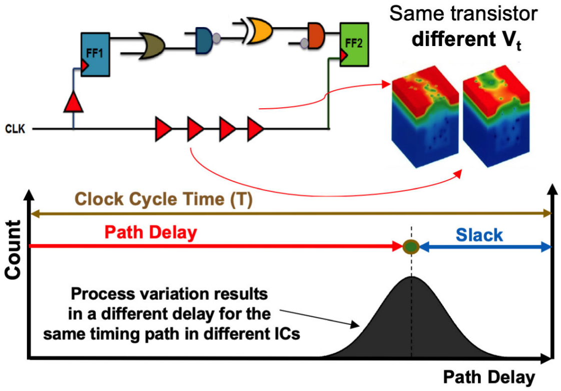


FIGURE 2. The impact of random process variation on the delay of a timing-path when sampled across multiple dies (after fabrication).

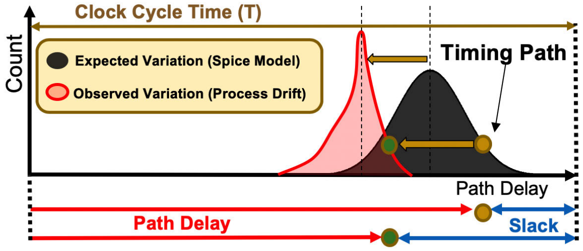

Clock Cycle Time (T)
Count
Expected Variation (Spice Model)
Observed Variation (Process Drift)
Timing Path
Path Delay
Path Delay
Slack

FIGURE 3. Improvement in the process over time non-linearly changes the delay of different timing-paths (process drift). The process drift affects each timing-path differently.

2) PROCESS DRIFT

The SPICE model for the fabrication process in a new technology node is released soon after the process is stabled and is used to characterize the standard cell libraries deployed in a physical design house. The SPICE model and standard cell libraries are padded with carefully crafted margins to guarantee a high yield. Furthermore, the foundry keeps improving the process over time to improve yield and reduce cost and may update the process by deploying newer and more capable stepping devices. Hence, the fabrication process and the released SPICE model drift apart over time. The improvement in the process builds large unused slacks in a fabricated IC that is designed using the older SPICE model. This practice poses a security problem as these unused and hidden timing slacks (to the test engineer) can be used by an adversary in the untrusted foundry to design stealthy hardware Trojan(s). Figure 3 illustrates the impact of the Process Drift on the slack of timing paths.

3) SYSTEMATIC PROCESS VARIATION

Systematic Process Variation is the result of imperfection in one or several process steps, as a result of which, a systematic shift occurs in the behavior of transistors or wires. For example, the systematic shift may speed up all NMOS transistors, increase the capacitance of a given metal layer, or reduce PMOS transistors' strength. Unlike random process variation (mitigation of which is disclosed when we describe our IC classification methodology), the systematic (inter-die) process variation affects all devices similarly. Therefore, systematic process variation behaves similarly to process drift, with the difference that process drift is the intended consequence of improving the fabrication process. On the other hand, the systematic process variation is an unintended consequence of imperfection in one or several processing steps. For example, if during the Chemical Mechanical Polishing step, the height of a specific metal layer, e.g., M4, was less or more than the process defined height, the expected resistance, and capacitance of all M4 net segments systematically shifts. In practice, the systematic process drift can be treated similarly to process drift.

IV. PROPOSED HW TROJAN DETECTION SOLUTIONS

Our proposed HT detection solution, denoted by AVATAR, integrates multiple variation modeling and mitigation techniques into a side-channel delay analysis solution for HT testing. Employing these modeling and mitigation solutions allows us to characterize and alleviate the impact of process variation and process drift to improve the correlation between the expected timing model and the fabricated ICs' timing behavior.

The process drift and systematic process variation are modeled with an NN-watchdog, which is trained using sampled timing path delays at test time. The details of our proposed NN-watchdog is further discussed in Section IV-A.

We model the random process variation using Parametric On-Chip Variation (POCV) [34] at design time and deploy a delay-averaging technique on path delay measurements obtained across multiple ICs to cancel the delay impact of process variation. The details of our proposed solutions are also discussed in Section IV-B.

A. MODELING AND TRACKING THE PROCESS DRIFT

Process drift results in a non-uniform shift in the delay of different timing-paths. To model the timing impact of process drift, we train a Neural Network (NN) that acts as a process tracking watchdog (NN-watchdog). This NN-watchdog predicts the slack difference between the delay reported by STA at design time and sampled delay from fabricated IC at test time. To train the NN-watchdog, we use a labeled data-set in which each labeled data point is a collection of 48 input feature values and a label (output) value. The input features, detail of which is captured in Table 1, are extracted from physical design and timing engine EDA tools. The label for each data point is the difference between the slack reported by STA at design time and that obtained from Clock Frequency Sweeping Test (CFST) at test time.

TABLE 1. Description for each of 48 features, extracted from each timing-path to build the NN training set. (LP: Launch portion of timing-path, CP: Capture portion of timing-path, DP: Data portion of timing-path, M: Metal Layer, x: drive strength of the gate).

Total of 48 Features, 3 Feature Extracted from each timing-path		
Setup Time	Path delay reported in STA	Sum of fanout over cells in DP
45 Feature Extracted, 15 from each sub-path (CP, LP and DP)		
number of gates	subpath Delay	# cells of x0 strength
# cells of x1 strength	# cells of x2 strength	# cells of x4 strength
# cells of x8 strength	# cells of x16 strength	# cells of x32 strength
Total Length of M1	Total Length of M2	Total Length of M3
Total Length of M4	Total Length of M5	Total Length of M6

To assess the effectiveness of NN-watchdog (and for the lack of access to fabricated ICs), we modeled the process drift (and systematic process variation) by extracting the shift in delay values from SPICE simulations, performed using a skewed SPICE model. For this purpose, we first extracted the SPICE model for each timing-path in the input training. Then, to mimic a systematic process drift, the SPICE model was skewed such that the NMOS and PMOS transistors were ∼X% faster, and the Metal capacitance for Metal layers 1 to 7 derated by Y%. The selection of X and Y gives us a consistently faster or slower process model. For example, the selection of $(X, Y) = (5, 5), (0, 0), (-5, -5)$ produces Fast, Typical, and Slow process models in our simulations.

In this paper, we have evaluated 3 different models for predicting the process-induced change in the timing path delays. Details of each model is given next:

1) LINEAR REGRESSION (RIDGE REGRESSION) MODEL (BASELINE)

Ridge Regression [35] is a regularized linear regression model and it is useful for modeling and tracking multi-collinearity phenomena.

2) MULTI-LAYER PERCEPTRON REGRESSION

Multi-Layer Perceptron (MLP), is a non-linear neural network composed of an input layer, one or more hidden layers, and an output layer (Fig. 4-top). Details and setup of MLP regressor used in this paper is summarized in Table 2.

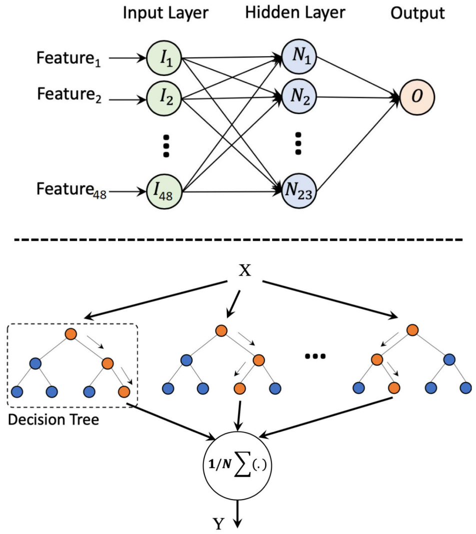


FIGURE 4. (Top): Abstract view of a Multi-Layer Perceptron (MLP) constructed as a fully-connected NN. (Bottom): A Random forest (Rf) as two base models to form test-time process watchdog.

TABLE 2. Hyper-parameters of regressor models used in this table.

Model	**Hyper-Parameters**
Ridge	alpha=1, max_iter=5000
Lasso	alpha=0.001, max_iter=5000
RF	n_estimators=1024, bootstrap=True, min_leaf=1, min_split=2
Xgb	n_estimators=1024, learning_rate=0.05
Enet	alpha=0.001, max_iter=1000
MLP	in_layer=48, hidden_layer=23, out_layer=1, activation='tanh', optimizer='adam', learning_rate='adaptive', start_lr='0.1'

3) STACKING REGRESSION MODEL

The structure of Stacking Regression model [36], which is also known as stacked generalization [37], is depicted in Figure 5. The Stacking Regression is an ensemble learning technique in which different estimators are arranged into two layers to form a regressor with lower variance in comparison to each (member) regressor. More precisely, at a two-layer stacked regressor (Figure 5-top), we used regressors XGB [38], Enet [39], Lasso [40], Ridge [35], MLP [41], and RandomForest [42] for our first layer regression. The predictions of these regressors, $\hat{y}^{\eta_1}$ to $\hat{y}^{\eta_6}$, are stacked together

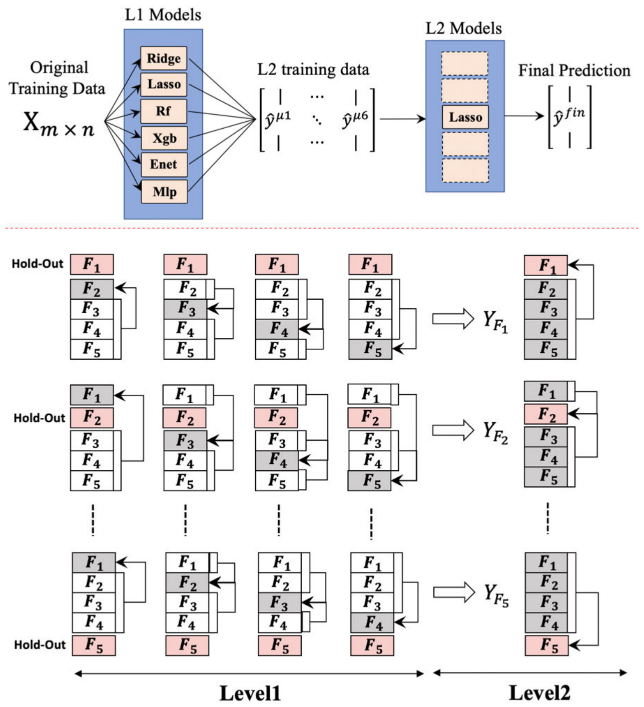


FIGURE 5. Top figure shows a two-layer stacked regressor. Bottom figure shows the cross-validation method used for obtaining hyper parameters at a two layer stacked regressor.

and fed to the second layer of regressor(s). In general, the second layer may also consist of multiple regressors. The overall prediction $\hat{y}^{fin}$ is obtained by averaging the results of the second layer regressors. In our model, we have only deployed a single Lasso [40] regressor in the second layer, as including additional regressor result in only negligible improvement in the model's prediction performance at the cost of increased complexity.

Among the regressors used in two-layer stacked regressors, Enet, Lasso and Ridge are linear. In these models, the difference in the performance stems from their associated regularization penalty. More precisely, Equation 1 shows the optimization problem formulated for Enet.

$$\hat{W} = \underset{W}{argmin}(\|y - XW\|^2 + \lambda_2 \|W\|^2 + \lambda_1 \|W\|) \qquad (1)$$

In this equation, X is the input features, see Table 1, W denotes the corresponding weight for each feature, and λ_1 and λ_2 are the hyper-parameters for tuning the contribution of L_1 and L_2 norms of weights, respectively. Lasso and Ridge are special cases of Enet in which Lasso only considers L_1 norm of parameters ($\|W\| = \sum_{j=1}^{N} |W_j|$) while Ridge considers L_2 norm of parameters ($\|W\|_2 = \sum_{j=1}^{N} |W_j|^2$) for regularization.

Random Forest and XGB can be considered as ensembles of decision-trees. The main difference between these two categories is the way that decision trees are combined. In Random Forest, also known as a bagging-based algorithm, a subset of features is randomly selected to form a forest of decision trees, see Fig. 4-bottom. Each of these trees is trained independently, and the final regression model is determined by averaging the result of each decision tree. In XGB, also known as a boosting-based algorithm, decision trees depend on each other, and through cascading, the error of previous trees is minimized (Boosting). Details and setup of each regressor used in the stacked model (which is used in this paper) are summarized in Table 2.

The six regresses in the stacking model were down-selected from a larger pool of regression models that we tested. Each of the member models was selected because they could reach a higher regression accuracy on a subset of input samples. After combining these regressors, the resulting stacking model had a lower variance compare to each member. Moreover, this selection represents different classes of regression models, each trained with a different loss function. Note that linear models contribute the least to the final regression accuracy, and removing one or two of the linear models from the stacking model has a small impact on regression accuracy (e.g., 2% drop if two of linear models, Lasso and Ridge, are removed). However, in the case of HT detection, even 2% improvement in model accuracy is substantial and counts. Considering that our goal was to achieve the highest possible regression accuracy, we only eliminated regressors (from our original pool) whose elimination affected the accuracy of the regression model less than 0.5%.

Figure 5-bottom shows the cross-validation technique for defining the hyper-parameters of each one of the used regressors. Cross-validation consists of four steps: 1) Randomly partitioning the training set into k equal sets, also known as K-fold. 2) Holding out one of the training sets, highlighted with red, from the (k-1)-remaining folds, and training on the (k-2)-fold. The left-out fold, highlighted with gray, is used for validation. This procedure continues for (k-1) times, which results in a stack of prediction of (k-1)-folds, which is stored in Y_{F_i}. 3) Training the layer two regressors based on the obtained dataset, Y_{F_i}, and evaluating the level-2 regressors based on the holdout set, F_i. 4) Selecting the hyper-parameters that result in a lower average loss. Once the hyper-parameters are corrected, both layers are trained on the whole training set, without k-folding, and the final results are reported by evaluating the trained stacked model on the test set.

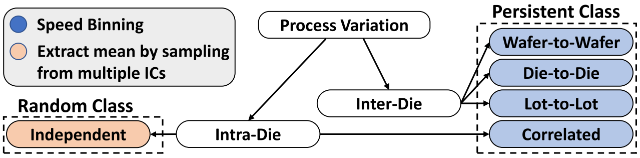


FIGURE 6. Process variation classification.

B. MODELING AND MITIGATING PROCESS VARIATION

Figure 6 depicts the classification of process variation. In this paper, process variation is categorized into two major classes: 1) *Random* Class that includes the independent intra-die

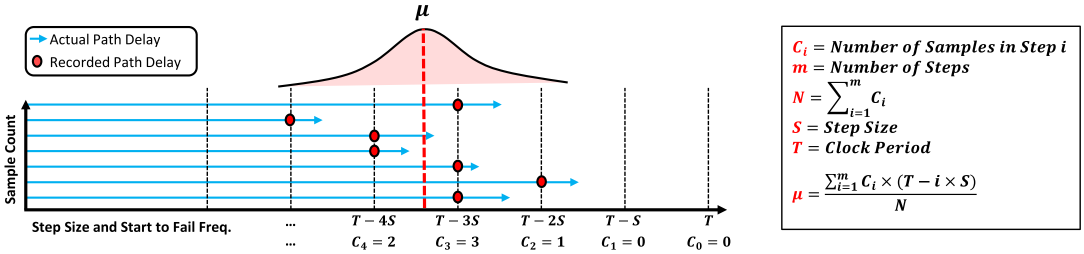


FIGURE 7. Computing the mean delay (μ) of a timing path using CFST delay measurements with step size S, clock period T across N sample dies, where $N=\sum_{i=1}^{m} C_i$.

random process variation, and 2) *Persistent* class including all forms of inter-die variability and the correlated intra-die variations.

1) MODELING THE PROCESS VARIATION

In this paper, the persistent process variation is modeled similarly to process drift. To emulate the persistent process variation (within the same process corner), we created two additional derivatives (slightly modified copy) for each of our skewed SPICE models. Each of the skewed SPICE models (originally used to model process drift) are additionally altered to make the transistors in the first derivative 1% slower, and the second derivative 1% faster.

To model the random process variation, each of the SPICE simulations is subjected to 100 Monte Carlo simulations. This is to emulate the CFST performed on 100 different dies in the same speed-bin where the threshold voltage (V_{th}), Oxide thickness (T_{ox}), and channel Length (L) are varied based on a normal distribution. Accordingly, we model the variation of path delays from chip to chip according to the expected variations in a 32nm technology node.

2) MITIGATING THE PROCESS VARIATION

To mitigate the process variation at test time, we use two different mechanisms to deal with random and persistent process variations:

a: PERSISTENT PROCESS VARIATION

We perform speed binning on fabricated ICs and divide them into different speed bins (Fast, Normal, and Slow), arguing that ICs in the same bin are similarly affected by the persistent process variation. Then for each bin, we train an NN-watchdog that could predict the impact of persistent process variation (and simultaneously process drift) on each timing path delay.

b: RANDOM PROCESS VARIATION

To reduce the impact of random process variation, using the formulation presented in Figure 7, we collect the delay of each timing path (in our test set) from many ICs and compute their average delay to be used in our HT detection solution. When the timing-path delay is averaged across N different dies, the standard deviation of the random variable representing the average delay is N times smaller than the standard deviation of individual samples. To illustrate this, assume that there are N independent samples (considering that variation has resulted from local and random process variation) for the timing path P_i, where each $X_{(i,n)}$ is a random variable representing a delay measurement for a given timing-path from a tested IC with index (i, n). Assume that the mean and standard deviation of $X_{(i,n)}$ are $\mu_{(i,n)}$ and $\sigma_{(i,n)}$, and the mean and standard deviation of the averaged random variables $\bar{X}_i$ are $E(X_i)$ and V $= VAR(X_i)$, respectively. Equation 2 describes the average of the random variable over N samples.

$$\bar{X} = \frac{1}{N}\sum_{n=1}^{N} X_{(i,n)} = E(\bar{X}_i) + VAR(\bar{X}_i) \tag{2}$$

Accordingly we have:

$$E(\bar{X}_i) = E(\frac{1}{N}\sum_{n=1}^{N} X_{(i,n)}) = \frac{1}{n}E(\sum_{n=1}^{N} X_{(i,n)}) \tag{3}$$

$$\begin{aligned} VAR(\bar{X}_i) &= VAR(\frac{1}{N}\sum_{n=1}^{N} X_{(i,n)}) \\ &= \frac{1}{N^2}VAR(\sum_{n=1}^{N} X_{(i,n)}) = \frac{\sigma^2}{N} \end{aligned} \tag{4}$$

As illustrated, by averaging the delay over N ICs, the standard deviation (σ) could be made $\sqrt{N}$ times smaller, while the mean (μ) remains unchanged. Now consider the case where an HT is added to the timing path: the averaged (mean) delay will be shifted compared to that expected from the process (which is predicted by NN-adjusted slack obtained from STA) and its impact could not be hidden by process variation.

The above equation assumes that the value of each random variable could be precisely measured. However, when it comes to measuring the delay of timing paths, we are limited by the barrier posed on us by the tester's step size (when performing the clock frequency sweeping). In other words, the tester's step size is a discrete value (for example, 10ps). Hence, our equation for mean value has to consider the step size of the tester. This concept is illustrated in Figure 7. Assume that the T is the clock period of the IC under clock frequency sweeping test, and we perform a clock frequency sweeping with step size S. Let m be the number of steps tested. For a given timing path, assume that C_i is the number of ICs (samples) in which the selected timing path (under test) passes the setup timing test at step i, but it fails at step $i+1$. We refer to this step as a start-to-fail frequency step. Using this set of assumptions, the mean value of the quantized start-to-fail frequencies is obtained from:

$$\mu = \frac{\sum_{i=1}^{m} C_i \times (T - i \times S)}{\sum_{i=1}^{m} C_i} \tag{5}$$

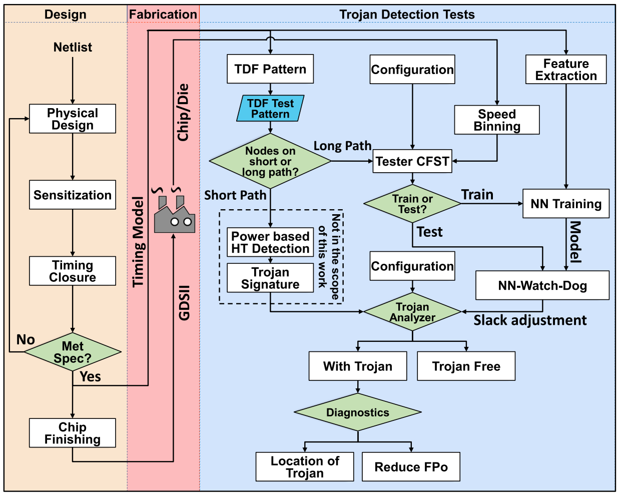


FIGURE 8. AVATAR Trojan Detection Flow: The model includes changes in the design and test stages. In the test stage, if the timing paths are too short and beyond the CFST test's reach, they will be subjected to power side-channel Trojan detection as described in [26] (not covered in this paper). Otherwise, timing paths are subjected to side-channel delay analysis. Our model uses an NN that servers as a process watchdog for tightly correlating the STA reported data to delay collected at test time. The NN model is trained using a limited number of delay samples collected using the CFST test and path features collected from the design database. If the Trojan detection flow flags detection of one or more HT, we perform a subsequent diagnostic test in which a) the false positive rate is significantly reduced, and b) the nets hosting the TT or TP are identified.

C. AVATAR TROJAN DETECTION FLOW

Figure 8 shows the overall flow of the AVATAR Trojan detection solution. In our model, the design house is trusted. However, the fabrication and functional testing of the IC may be carried in an untrusted foundry. The fabricated ICs are then shipped to a trusted facility for Trojan testing. Our proposed changes to the design flow and steps added for Trojan testing is described next:

1) SENSITIZATION

This step involves a set of physical design constraints, design methodologies, and Engineering Change Orders (ECO) that make the design more sensitive to TT or TP of a hardware Trojan; thereby making HT more visible if inserted. Techniques used for sensitizing the design include 1) limiting the size and drive strength of logic cells to X4 or X8 as the delay of smaller gates is more sensitive to payload change. 2) Running aggressive low power optimization ECOs at timing closure. The above techniques result in under-driving each gate's fanout, making them more sensitive to payload change. 3) reducing routing resources in areas in need of protection for HT requiring an adversary to rely on a complicated and long route (with significant capacitive impact), and 4) Reducing unused space (via increasing the area utilization) to limit the chances of Trojan insertion.

Algorithm 1 Generating a Training Set for the NN-Watchdog

```
$S_{size} \leftarrow$ Required test set size
M $\leftarrow$ Number of timing path segments
$T \leftarrow$ Operating frequency of the IC
$T_{max} \leftarrow$ Max frequency of the tester equipment
$\delta \leftarrow (T_{nom} - T_{max})/M$
for (S = 0; S < M; S++) do
    $TPaths \leftarrow$ Select $S_{size}/M$ worse timing path in the range $(T-(S+1)\times\delta <$ delay(Timing path) $< T-S\times\delta)$
end for
for all path in TPaths do
    feature(path) $\leftarrow$ Extract path features from STA
    STA(path) $\leftarrow$ Extract path slack from STA
    Slack(path) = 0
end for
for all die in Dies do
    for all path in TPaths do
        CFST(die, path) $\leftarrow$ path Slack of CFST on die
        Slack(path) + = CFST(die, path)
    end for
end for
for all path in TPaths do
    Slack(path) = Slack(path)/Size(Dies);
    $\Delta_{slack}(path)$ = Slack(path) − STA(path)        ▷ label
    data-points(path) $\leftarrow$ (features(path), $\Delta_{slack}(path)$)
end for
```

2) FABRICATION

The final GDSII submitted to the foundry for fabrication, the functionality of the fabricated ICs tested in the untrusted foundry where the working ICs will be sent to a trusted facility for Trojan detection.

3) GENERATING A TRAINING SET FOR THE NEURAL NETWORK

The flow for generating the set used for training the process-tracking neural network model is shown in Algorithm 1. To generate such a set, we need to select and extract features from a selected group of timing paths. The selected group should represent a variety of timing-path topologies that exist in the design. Simultaneously, the selected timing paths should have large enough delays to be testable using CFST at test time. However, accounting and including all path topologies (all register-to-register timing paths) in a training set, in a large design, will result in a massive test set, and it is beyond the need of NN for training. To simplify the training set generation, we decided to break the range of testable frequencies into M

(in our case, M = 10) segments. Let's consider the nominal frequency of the netlist to be $F = 1/T$, and the maximum testable frequency to be $F_{max} = 1/T_{min}$. By breaking the testable range into M segments, each segment covers a range of delays equal to $\frac{T-T_{min}}{M}$. The desired number of timing paths, S_{size} in the training set of our model, is 50× to 100× of the number of input features, which is about 2500 to 5000 timing paths. Then to select the timing paths, we query the STA engine for S_{size}/M worst timing paths in each of the M testable delay segments. If a delay segment does not contain enough timing paths to satisfy the training size requirements, we add additional timing paths from other delay segments to reach the desired training set size. The goal here is to select timing paths covering a range of different delays and timing path topology. In the next step for generating the training set, as shown in Algorithm 1, the path's features are extracted from the design-time EDA tools. Some of the path features are extracted from STA and some from the PnR tool. Then each selected timing path is subjected to the CFST test, where the timing slack of each selected timing path is recorded (by measuring it across multiple ICs and averaging the result) as the label. This training set is used in the next step to train a NN that acts as a process (drift and systematic variation) tracking watchdog. Note that the delay of the timing paths (in lines 17 and 21 of the algorithm) is averaged across many dies to extract the mean delay of the timing path and to remove the effect of process variation.

4) TRAINING THE NEURAL NETWORK

The neural network that acts as a process tracking watch-dog is trained using the methodology described in section IV-A. As previously described, the trained neural network will track the change in delay due to process drift and systematic process variation. When provided with features of a test sample (of a timing path), the NN-watchdog will generate an estimate on the expected difference between CFST reported test delay of the timing path, and the delay reported by the static timing model (generated at design time).

5) GENERATING TEST SET FOR TROJAN DETECTION

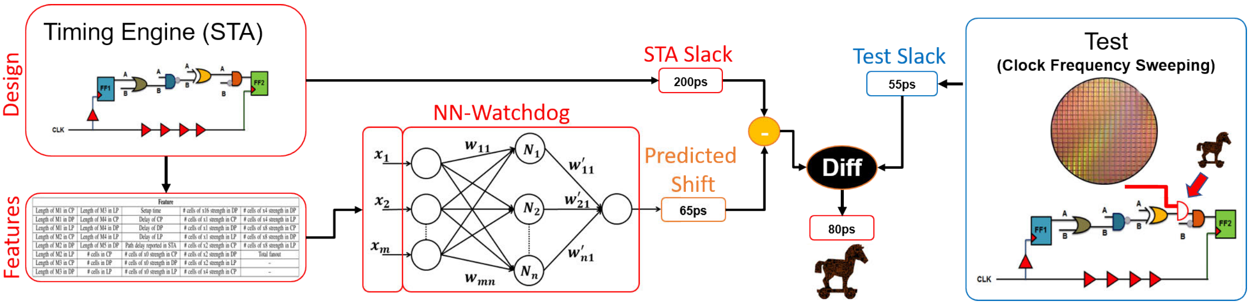
To detect a Trojan, we need to detect the change of slack in affected timing paths due to TT or TP presence. As Figure 1 shows, the TT adds capacitive load to the driving cell of an observed net, and the TP inserts one (or more) additional gate(s) in the victim net. Without a Golden IC, we do not know which nets have been victimized. Hence, we need to check for the delay change of timing paths by investigating each net included in the suspicious timing paths, i.e., the timing paths whose slack appeared to be larger than expected when doing frequency sweeping test. We define a Pin-to-Pin Wire (P2P-wire) as a net connecting the output pin of a driver cell (or a primary-input) to the input pin of one of its fanout cells (or a primary-output). Hence a gate with a fanout of 4 has 4 P2P-wires. Each P2P-wire will be tested for rise and fall transitions. To increase the detection rate and to account for random process variation, this process may repeat for N different timing-paths passing through each net (similar approach to N-detect testing). The second criterion for selecting the timing-paths is the maximum frequency of the tester equipment; the selected paths' delay should be larger than the limit imposed by the maximum reachable frequency of the tester equipment. If the P2P-wire in no timing-path is long enough for CFST, it cannot be subjected to side-channel delay testing. However, such timing-path could be used as a candidate for Trojan detection via power-based detection schemes. This is because timing-paths with a smaller number of cells often exhibit better controllability and are better suited for the power-based Trojan detection schemes (e.g., [24], [25], [27], [29]) that rely on full or partial activation of a Trojan. We generate the Path Delay Fault (PDF) test vectors for the long timing-path candidates using an Automatic Test Pattern Generation tool (ATPG). Suppose the ATPG fails to generate a test pattern for exclusive PDF testing of a given net in a suspicious timing path (in other words, no other timing path exists that contains the targeted net while it does not contain any of the other nets in the suspicious timing path). In that case, a sequence of nets (preceding or proceeding the target net) is selected for test pattern generation.

6) TROJAN SIGNATURE DETECTION

Fig. 9 illustrates the conceptual view of the steps taken in the proposed Trojan signature detection scheme. In simple terms, the Trojan detection is the comparison of the slack reported by STA, which is adjusted by NN-watchdog, against the collected CFST test data. If the shift in slack is larger than the expected value (a large discrepancy between the NN-adjusted slack reported by STA and that recorded at test time), the timing path is considered as likely-affected by Trojan.

Algorithm 2 describes our proposed Trojan detection flow. As depicted, after selecting a set of timing paths for PDF testing, we speed-bin the fabricated dies. In the next step, we collect the NN-watchdog training data using the flow described in algorithm 1. Then, we train a process tracking NN-Watchdog for each bin and extract the standard deviation of each NN-Watchdog in predicting the shifted delays. For each bin, we perform CFST and measure the start-to-fail frequencies for the selected timing-paths. The slack difference (δ) between the mean of slacks reported by the CFST and the NN-watchdog adjusted slack from STA (in the same bin) represents the likelihood of a timing path being affected by a Trojan. To make a binary decision, we use a detection threshold to assess the significance of δ and classify the timing paths into genuine, or Trojan affected classes.

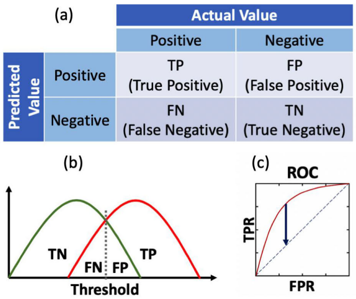
In choosing a value for the HT-Detection threshold, we face a trade-off between false-positive rate and HT detection accuracy. The false-positive could result from 1) inaccuracy in the STA, 2) inaccuracy of NN, and 3) random process variation for sampling over a small number of ICs. The threshold used for detection should be large enough to account for the uncertainties above to reduce the false-positive rate, and small enough to prevent false-negatives. Since we average



Feature				
Length of M1 in CP	Length of M3 in LP	Setup time	# cells of x16 strength in DP	# cells of x4 strength in DP
Length of M1 in DP	Length of M4 in CP	Delay of CP	# cells of x1 strength in CP	# cells of x4 strength in LP
Length of M1 in LP	Length of M4 in DP	Delay of DP	# cells of x1 strength in DP	# cells of x8 strength in CP
Length of M2 in CP	Length of M4 in LP	Delay of LP	# cells of x1 strength in LP	# cells of x8 strength in DP
Length of M2 in DP	Length of M5 in DP	Path delay reported in STA	# cells of x2 strength in CP	# cells of x8 strength in LP
Length of M2 in LP	# cells in CP	# cells of x0 strength in CP	# cells of x2 strength in DP	Total fanout
Length of M3 in CP	# cells in DP	# cells of x0 strength in DP	# cells of x2 strength in LP	-
Length of M3 in DP	# cells in LP	# cells of x0 strength in LP	# cells of x4 strength in CP	-

FIGURE 9. Our proposed Trojan detection flow's conceptual view: The timing path features are extracted from STA and PnR EDA tools at design time. The features are input to a trained NN-watchdog that predicts the shift in the timing path slack due to process drift and systematic process variation. The timing path is also subjected to the CFST test at test time, and the timing path slack is observed (by averaging across multiple ICs). If the difference between the STA reported delay adjusted based on NN-watchdog's shift-prediction and the CFST suggested delay is greater than a safe threshold, the timing path is marked as a suspicious Trojan timing path. Please also see our proposed diagnostic solution, following the Trojan detection for reducing the false positive rate.

the delay of each timing-path over many IC samples, timing impact of random process variation in the average delay could be reduced to a desirable range. However, we still have to account for the inaccuracy of the NN and persistent variation. Hence, we define the detection threshold to be $T_{th} = n \times max(\sigma_{NN}, \sigma_{PV})$, in which the σ_{PV} is the expected variance of persistent process variation (excluding random) and σ_{NN} is the standard deviation of the NN. Since σ_{NN} is the aggregated impact of NN inaccuracy (for under or over-fitting) and impact of persistent process variation, the variance of σ_{NN} tends to be larger than σ_{PV}, and we can simply use $T_{th} = n \times \sigma_{NN}$ (n is selected as 4 in algorithm 2).


(a)

		Actual Value	
		Positive	Negative
Predicted Value	Positive	TP (True Positive)	FP (False Positive)
	Negative	FN (False Negative)	TN (True Negative)

(b)

(c)

FIGURE 10. a) Contingency table (confusion matrix) for a binary classifier; b) distribution of true negative and true positive samples; c) Youden [43] method for threshold extraction.

To verify the choice of threshold values T_{th}, we utilized Youden [43] method to extract the threshold value from a Receiver Operating Characteristic (ROC) curve that we generate over our SPICE simulation data (details in Section V). Figure 10 defines a few analysis related terms (True Positive (TPos), False Positive Rate (FPos), False Negative (FNeg), and True Negative (TNeg)), and illustrates how the ROC curves are obtained. As illustrated in this figure, the Youden method chooses the threshold value that corresponds to the optimum cut-off point between *TPos* and *FPos*. Note that at test time, we do not know which timing-paths are affected by HT. Hence, the optimal threshold of detection cannot be determined by Youden method.

Change in the temperature affects the speed of transistors and alters the RC characteristics of the connecting wires. However, the temperature change is an intensely slow phenomenon. At test time, a test vector is loaded into the scan chain using a slow clock then the circuit operates at-speed for two cycles (launch and capture) using a fast clock. Finally, the scan offloaded using a slow clock. The heat dissipation when using a slow clock is quite low, and the duration of the at-speed test is only two cycles for each test pattern, limiting the extent of temperature changes to a fraction of a degree Celsius. Hence, the die temperature can be tightly controlled at test to discount the delay impact of temperature variations.

To proceed with Trojan detection, as described in Algorithm 2, we first separate the fabricated dies according to their speed and performance into different speed-bins. Then, we train a process tracking NN-Watchdog for each bin and extract the standard deviation of each NN-Watchdog in predicting the shifted delays. In the next step, we perform CFST and measure the start-to-fail frequencies for different timing-paths. If the mean value of the slack difference between the slack reported by the CFST and the NN-watchdog adjusted slack from STA (in the same bin) is less than our detection threshold, the path classified as Trojan-Free.

A Trojan is detected if one of its TTs or TPs (Trojan signatures) is detected in the previous step. In the result section, in addition to reporting the rate of TT and TP detection, we also report the overall Trojan detection performance of our solution. In this case, a Trojan is detected if our proposed solution can detect at least one of its triggers or payload(s).

D. DIAGNOSTIC ANALYSIS

AVATAR could classify a timing path as likely affected by Trojan for two reasons: 1) The timing path hosts the trigger or payload of HT, resulting in a true-positive, 2) the timing path is severely affected by process variation such that the resulting increase in the path delay is greater than the threshold, resulting in a false-positive. Our proposed diagnostic test, as illustrated in Fig. 11 opts to deeply investigate the detected HT and extract the reason behind the increase in each

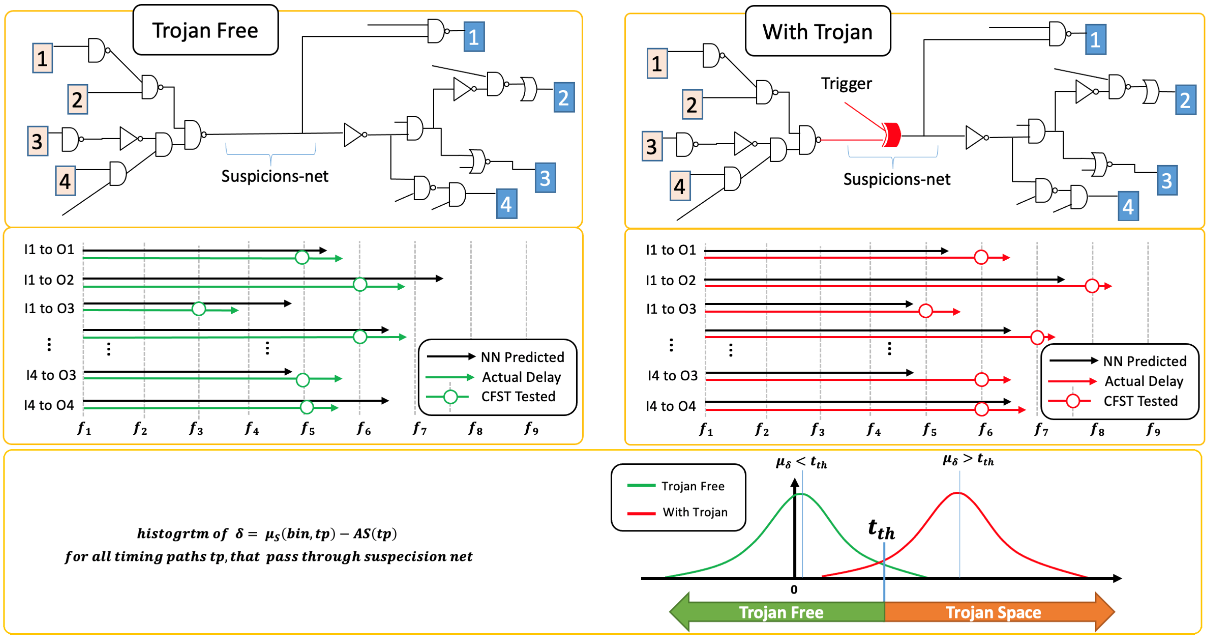


FIGURE 11. Diagnostic Test: (left): an HT free design where a suspicious net is tested for HT through many timing paths passing through it. The delta difference between predicted delay (NN adjusted STA delay) and CFST test delay (collected from multiple ICs) for each timing path is computed. In a HT-free design, the process variation results in a variation of the resulting delta value, but the mean of this delta difference is close to 0. (right): A design with HT on the suspicious net. The delta difference, in this case, is also a distribution. However, the existence of HT pushes the mean of this distribution away from 0. (bottom): The distribution of delta difference for the design with and without HT is shown. The HT is detected if the mean value of the delta difference distribution is greater than the detection threshold obtained from Algorithm 2.

timing path's delay resulted in detecting such a Trojan. This diagnostic step's benefit is twofold: 1) reducing false positive, 2) pinpointing to the location of HT.

Algorithm 3 captures the flow of our proposed diagnostic solution. The overall strategy is quite simple and based on the following assumption: In a false positive case, the increase in the delay is due to process variation, and the total delay increase is distributed over different segments of the suspicious timing paths. Thus, by selecting N timing paths (that share least number of nets with the suspicious timing path whose net segments are being tested), we expect a small increase in the delay of N chosen paths compared to the NN-adjusted timing model prediction.

To run our diagnostic test, we select N timing paths that pass through the selected net for each net in the suspect timing path such that the selected timing path and the suspicious path do not include any common path segments other than the selected net. Then the delay of each of these timing paths is assessed against our NN-adjusted timing model. The delta δ between the average delay of that timing path reported by the CFST test and the delay of that timing path expected from our model is computed using:

$$\delta = \mu_S(bin, tp) - AS(tp) \tag{6}$$

We then compute the mean of δ (μ_δ) across all selected timing paths passing through that net. If the μ_δ is smaller than a threshold value (close to 0), then the CFST slack and adjusted slack for all timing paths through that net match (no mean-shift), indicating no Trojan and the net is removed from the suspicious list. However, suppose μ_δ is a positive number larger than the threshold. In that case, that indicates a mean-shift in the difference between predicted delay (NN adjusted delay) and recorded CFST delay across all timing paths passing through that net. This is an indication of a Trojan. In this case, the algorithm also pinpoints the location of the Trojan (the net where the means shift in the distribution of delays occurs). If it is not possible to select N exclusive timing paths passing through a single net, a set of nets are selected to point to the HT location. Using this scheme, we reduce the false-positive rate of AVATAR, thus increasing the detection precision.

V. RESULTS AND DISCUSSION

This section depicts the accuracy of the NN-Watchdog in tracking the process drift, and presents the results of applying our proposed test flow, AVATAR, for Trojan detection.

A. NN-WATCHDOG ACCURACY

The NN-watchdog model is trained to predict the difference between the slack reported by the STA engine (at design time) and CFST (at test time). We evaluated the effectiveness of NN-watchdog on the three largest IWLS benchmarks [44] (Ethernet, S38417, and AES128). For training and test purposes, We collected a labeled dataset for each of these benchmarks using the methodology described in Algorithm 1 in section IV-C. We divided the dataset into a training and a test

TABLE 3. The Accuracy of Three NN-Watchdog regression model (Ridge Regression, MLP and Stacking-regressor) trained for different benchmarks on NGTM-10. The μ and σ are the Mean and Standard deviation of the regression error over the validation set. As discussed in Section IV-A, the Fast, typical and slow process are simulated using skewed SPICE model with (X,Y) = (5,5), (0,0), (−5, −5), respectively. μ and σ are reported in pico seconds.

Benchmark	# Gates	Size		Fast						Typical						Slow					
		Train	Test	Ridge		MLP		Stacked		Ridge		MLP		Stacked		Ridge		MLP		Stacked	
				μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
AES128	114K	21K	4K	0.17	9.26	-0.14	7.45	0.09	5.16	-0.03	12.54	0.04	8.12	0.02	5.13	0.07	13.52	-0.02	7.15	0.01	5.14
Ethernet	40K	20K	4K	0.09	28.43	0.79	9.65	0.03	6.52	-0.12	28.82	0.28	9.13	0.01	6.51	0.08	29.43	-0.65	8.36	0.01	6.50
S38417	6K	4K	1K	0.07	12.41	0.12	6.87	0.09	5.03	0.17	13.93	0.08	7.07	0.09	5.02	-0.03	12.91	0.25	6.38	0.07	5.04

Algorithm 2 AVATAR Trojan Signature Detection Flow

```
$N$ = # paths to be tested through each net in the design
$Nets \leftarrow$ all nets in the design.
for all $net$ in $Nets$ do                ▷ net selection for PDF test
    $Tpath += $ select $N$ paths passing through $net$
end for
Speed-bin all dies and assign them to B bins.
for all $bin$ in $B$ do                   ▷ NN training
    $NN_{bin} \leftarrow$ Use Algorithm 1 to Train a NN-watchdog
    $\sigma_{NN_{bin}} \leftarrow$ the standard deviation of $\mathrm{NN}_{bin}$
    for all $die$ in $bin$ do
        Slack = 0
        for all $path$ in $Tpath$ do
            CFST(bin,die,path) $\leftarrow$ Timing $path$ slack measured by CFST $die$ in the $bin$
            Slack(bin,path) += CFST(bin,die,path)
        end for
    end for
    for all $path$ in $Tpath$ do
        $\mu_S$(bin,path) = Slack(bin,path)/sizeof(bin)
    end for
end for
$T_{th} = 4\times \sigma_{NN_{bin}}$     ▷ $4\sigma$ threshold to reduce false positive
for all $path$ in $Tpath$ do
    STA(path) $\leftarrow$ query the slack of path from STA
    $\mathrm{NN}_{Watchdog}$(path) $\leftarrow$ slack shift from $NN_{bin}(path)$
    AS(path) = STA(path) + $\mathrm{NN}_{Watchdog}$(path)
    $\delta = \mu_S$(bin,path) − AS(path)          ▷ Adjusted Delay
    if $(\delta > T_{th})$ then                     ▷ Trojan Classifier
        Likely Trojan Set $\leftarrow$ $path$
    end if
end for
```

Algorithm 3 Diagnostics

```
$bin \leftarrow$ the speed bin that this IC is allocated to
$\mathrm{NN}_{bin} \leftarrow \mathrm{NN}_{bin}$ trained for $bin$ in Alg. 2
$\mu_S \leftarrow$ vector of mean values from Alg. 2
$M_P \leftarrow$ Malicious Paths detected by AVATAR in Alg. 2
$t_{th} \leftarrow$ Detection threshold from Alg. 2
$n \leftarrow 50$          ▷ number of new paths for diagnostics test
$d_{th} \leftarrow$ threshold for diagnostics
for all $path$ in MP do
    $L_{Nets} \leftarrow$ list of nets in $path$
end for
$U$ = unique($L_{Nets}$)      ▷ remove repeated nets in the list
for all $net$ in $U$ do
    $P_{inc} \leftarrow$ timing paths in $M_P$ that include net $net$
    for all $path$ in $P_{inc}$ do
        $L_{exc} \leftarrow$ all nets in timing path $path$ except $net$
    end for
    $L_{test} \leftarrow$ get $n$ timing paths that include $net$ and contains least number of nets in $L_{exclude}$
    for all $tp$ in $L_{test}$ do
        STA($tp$) $\leftarrow$ query the slack of $tp$ from STA
        $\mathrm{NN}_{watchdog}(tp) \leftarrow$ slack shift from $\mathrm{NN}_{bin}(tp)$
        AS($tp$) = STA($tp$) + $\mathrm{NN}_{watchdog}(tp)$
        $\Delta += \mu_S$(bin, $tp$) − AS($tp$)
    end for
    $\mu_\delta = \Delta$ / size($L_{test}$)      ▷ average shift over all paths
    if $(\mu_\delta > t_{th})$ then
        $L_{diagnostic}$ append($net$)
    end if
end for
```

set with 80% and 20% of samples in each set accordingly. Besides, we evaluated the effectiveness of three regressors 1) Ridge, 2) MLP, and 3) Stacking-Regressor. Ridge is a linear regressor that has enhanced efficiency when it comes to parameter estimation problems. We use this model as a baseline. MLP, see Figure 4, is a non-linear regression (used in our previous work, LASCA in [45]). The stacking-Regressor (Fig. 5) is an ensemble of both linear and non-linear regressors, and represent our improved NN-watchdog model in this paper.

Table 3 depicts the mean and standard deviation of the NN-watchdog in predicting the shift in the delay of timing-paths when subjected to process drift. The process drift is represented using Fast, Typical, and Slow process corners which are simulated using skewed SPICE model with (X,Y) = (5,5), (0,0), (−5, −5), respectively as discussed in Section IV-A. As illustrated, all models generate acceptable value for mean delay. The non-linear models (MLP and Stacked) result in a considerably smaller standard deviation across the board. As illustrated, the standard deviation in stacked regressor has also considerably improved compared to MLP, showcasing ensemble learning solutions' power in generating superior models. The lower standard deviation of the stacking-regressor model makes its prediction (of slack change) more accurate, resulting in a more precise Trojan detection solution, when integrated into our flow.

B. HW TROJAN DETECTION ACCURACY

In this section, we first describe our experimental setup, and then describe our simulations results:

1) SETUP

We have evaluated AVATAR Trojan detection accuracy on three largest IWLS benchmarks [44] (Ethernet, S38417 and AES128). We designed and inserted 90 HT into each benchmark. These HT are simple combinational HT with 4 input triggers that are connected to selected nets (as Trigger Nets), an AND-tree that generates the Trojan activation signal, and a single 2-input XOR gate to deliver the payload of Trojan to a target net. Nets selected for Trojan Trigger and Payload insertion are carefully chosen from non-critical timing paths with large available timing slack. The TT capacitive delay impact is controlled by the distance of Trojan Circuit placement (and first gate of the AND-tree) from the triggering net. To keep the Trigger impact small, we place the Trojan Circuit (first gate) within a 20um radius of the Trojan Trigger nets (assuring small delay impact). The Trojan Circuit's AND-tree is also constructed using the smallest AND gate available in the standard cell library, to minimize the gated capacitance impact of the driver gate on the delay of the Trojan triggering net. To illustrate our solution's sensitivity, each Trojan Circuit is added in a separate placed-and-routed netlist, meaning we would have 90 placed-and-routed netlists for each benchmark, each containing a single HT circuit. Each benchmark is hardened (physical design), and timing closed at 1.4GHz in 32nm technology.

During NN-watchdog training, we do not know if a timing-path selected for training contains a Trojan. Hence, we also evaluated the impact of including Trojan affected timing paths in the training set. We trained 5 NN-watchdogs with 0, 1, 5, 10, and 15 Trojan paths included in their training set. Our goal is to evaluate if the inclusion of Trojan affected timing paths (including trigger nets or payload nets) in the model could poison the model to a point resulting in Trojan evasion when affected timing paths are tested using AVATAR.

The silicon CFST test emulated using SPICE simulation, where the slack reported for each timing-path adjusted to the neighboring higher clock sweeping frequency step, modeling the CFST step size. The step size in the state-of-the-art tester equipment can be as small as 10-15ps. Hence, we selected the step size of the tester as 15ps. At this point, we also consider the impact of random process variation (see Figure 6). For this purpose, each SPICE simulation is subjected to 200 Monte Carlo simulations (modeling CFST performed on 200 different dies in the same speed-bin), where the threshold voltage (V_{th}), Oxide thickness (T_{ox}) and channel Length (L) are varied to model the variation of path delays from chip to chip. Similar to [16], we have limited the random process variation to 5%.

In our simulations, we assessed the effectiveness of Trojan detection using two mechanisms for building our reference (Golden) timing model:

a: SHIFTED STA (SSTA)

In which we use the STA results as our reference Timing Model to detect HT. The process drift makes the direct usage of STA results quite ineffective. To account for process drift in SSTA, we have computed a static shift value, obtained from averaging the observed shift from many sampled timing-paths, and have shifted all reported slacks by STA using this value. For this approach, we have set the detection threshold to the fixed value of *45ps*, which is the delay of a 2-input NAND gate in our standard cell library.

b: NEURAL SHIFTED GOLDEN TIMING MODEL (NGTM)

In which we modeled the process drift and systematic process variation using our proposed NN-watchdog. Then we augment the STA results with the predicted shift by NN-watchdog (generating path-specific change in slack based on path topology/features). To show the stacked learning model's effectiveness, we have also evaluated the usage of both MLP and stacked-regression as NN-watchdog. When collecting a dataset for training the NN-watchdog, there is no guarantee that the Trojan affected timing path(s) is not included in the training set. Therefore, we have investigated the accuracy of NGTM when the training set contains 0, 1, 5, 10, and 15 timing-paths affected by HT. In this approach, we have set our Trojan detection threshold to $4 \times \sigma$ of regressor standard deviation. The choice of $4 \times \sigma$ significantly reduces the number of false positives. Comparing the standard deviation of the stacked and MLP model, based on Table 3, provides a valuable insight on why the NN-watchdog designed using the stacked-regression model is expected to be more sensitive/accurate compare to the MLP-regression model: It benefits from a lower detection threshold, while statistically benefit from a similar false-positive rate.

To evaluate the quality of selected threshold values for our proposed Trojan detection flow, we have extracted and reported the optimal threshold from the ROC-curve using Youden [43] method. Note the optimal threshold cannot be extracted in real-life examples as it requires the ground-truth table (knowing exactly which timing path are and are not affected by Trojan), and could only be used for quality assessment purposes. The Youden method generates a different detection threshold for each of TT and TP based ROC curve.

Figure 12 captures the result of TP detection in Fast (X,Y) = (5,5) speed bin. The top row compares the accuracy of SSTA and NGTM in detecting TPs. The bottom row reports the false positive rate of detection for each model across different benchmarks. This figure evaluates Trojan detection's effectiveness when each of the Stacked-regression and MLP-regression models (for predicting the shift in slack) is used. The NGTM model is reported five times (NGTM-X), where each has been trained with X HT included in their training set, where $X \in [0, 1, 5, 10, 15]$. As reported, the inclusion of a small number of HT samples in our data set minimally impact the detection rate of AVATAR (using NGTM) on the test set, as the detection rate and false-positive rate

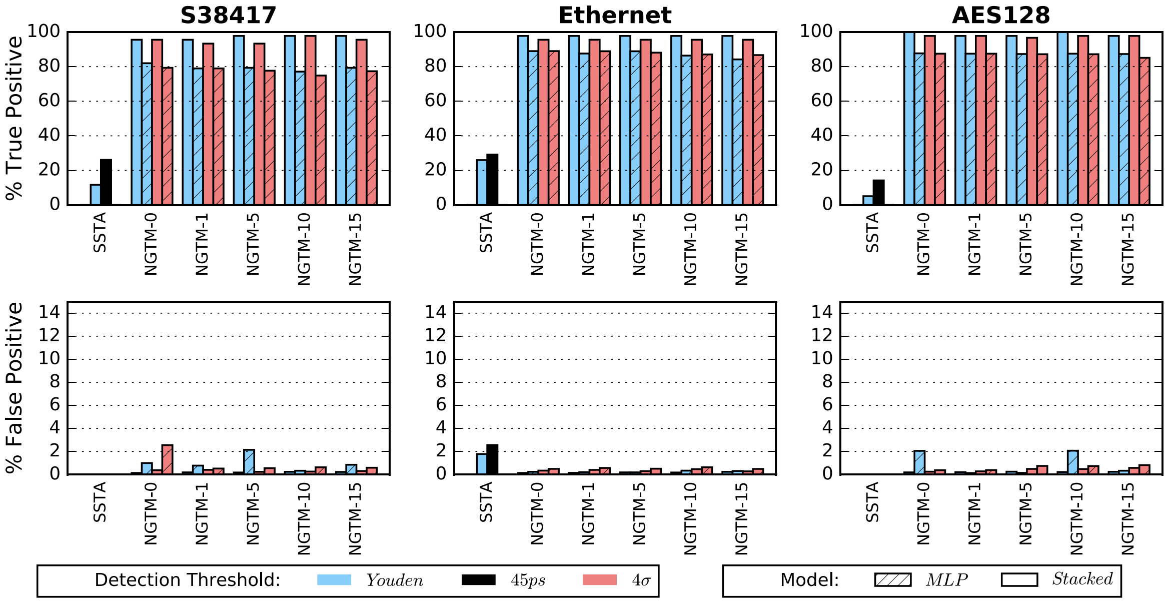


FIGURE 12. Trojan Payload detection results for 3 benchmarks. (top): Detection rate, (bottom): False positive rate. The SSTA bar represents the HT Payload detection using a (Mean shifted) STA. The NGTM bars represent the Trojan Payload detection when Neural-assisted timing model is deployed. Each bar shows the NN trained when X HT are included in the training set, with $X \in \{0, 1, 5, 10, and 15\}$.

of AVATAR for NGTM-0 is similar to the NGTM-X for $X \in [0, 1, 5, 10, 15]$. The similarity of detection rate and false-positive rate is simply because the number of HT is not statistically significant to affect the training (e.g., 15 Trojan data versus 20K Trojan free data points) process.

The NGTM not only results in a significant increase in the TP detection rate (to over 95%) but also significantly depresses the false positive rate. This confirms NN-watchdog's ability in modeling the complicated, non-linear, path-specific shift of delays resulting from process drift. Finally, note that a small number of HT in the training set does not affect the accuracy of trained NN-watchdog as the impact of a few samples in a large training set is statistically insignificant.

Figure 13 depicts the result of our TT detection in the FAST speed bin with (X,X) = (5,5). Like the TP case, it compares the effectiveness of SSTA and multiple forms of NGTM (Trojan contaminated model) for detecting TTs. The figure shows how the improvement in the standard deviation of NN-watchdog significantly improves the detection rate for TTs (over 40% in some cases). As shown, NGTM has a lower rate for detecting TTs than TPs due to the smaller impact of TT on the delay of affected observed nets compared to TP (which is at least equal to one gate delay). Like the TP case, we observe that contamination of the training set with few HT data points does not impact the accuracy of trained NN-Watchdog. As illustrated, the choice of the learning model (MLP vs. Stacked) for training the NN-watchdog has a significant impact on Trojan detection accuracy. As illustrated the stacking-regression, for having a smaller threshold (selected based on $4 \times \sigma_{NN}$ of regression model error) could improve the detection rate by 10% to 15%, resulting in over 95% Trojan detection rate. This is when the false-positive rate of the overall solution, when constructed based on the stacking-regression model, is equal to or lower than its MLP-based counterpart. The selection of the Youden threshold for detection, although significantly improves the TT detection, results in higher false positive, and perhaps is not a preferred mechanism for setting the detection threshold.

Figure 14 illustrates the ROC curve from which the Youden threshold (as described in Section IV-C) is extracted for NGTM-10. The Youden value for other NGTM models is extracted using similar ROC curves. Table 4 compares the threshold values obtained from using the Youden method with our proposed threshold of $4 \times \sigma_{NN}$ of regression model error. For having a different ROC curve, the threshold obtained from the Youden method is different for TT and TP detection. However, the $4 \times \sigma_{NN}$ threshold is fixed for both TT and TP (because the same NN for detection of TT and TP is used). As illustrated, the Youden threshold is smaller (for both TT and TP detection) than our proposed threshold. This explains why HT detection using the Youden threshold results in a higher detection rate in Fig. 13 and 12 for TT and TP detection. But, also as illustrated in these figures, the smaller threshold comes at the expense of a significantly higher false-positive rate. This table also compares the threshold values obtained using the stacking learning solution (in this work) and when using an MLP solution (in our previous work in [45]. As illustrated, the higher accuracy of the stacking model has resulted in a significant improvement in resulting threshold values. This explains why the AVATAR solution

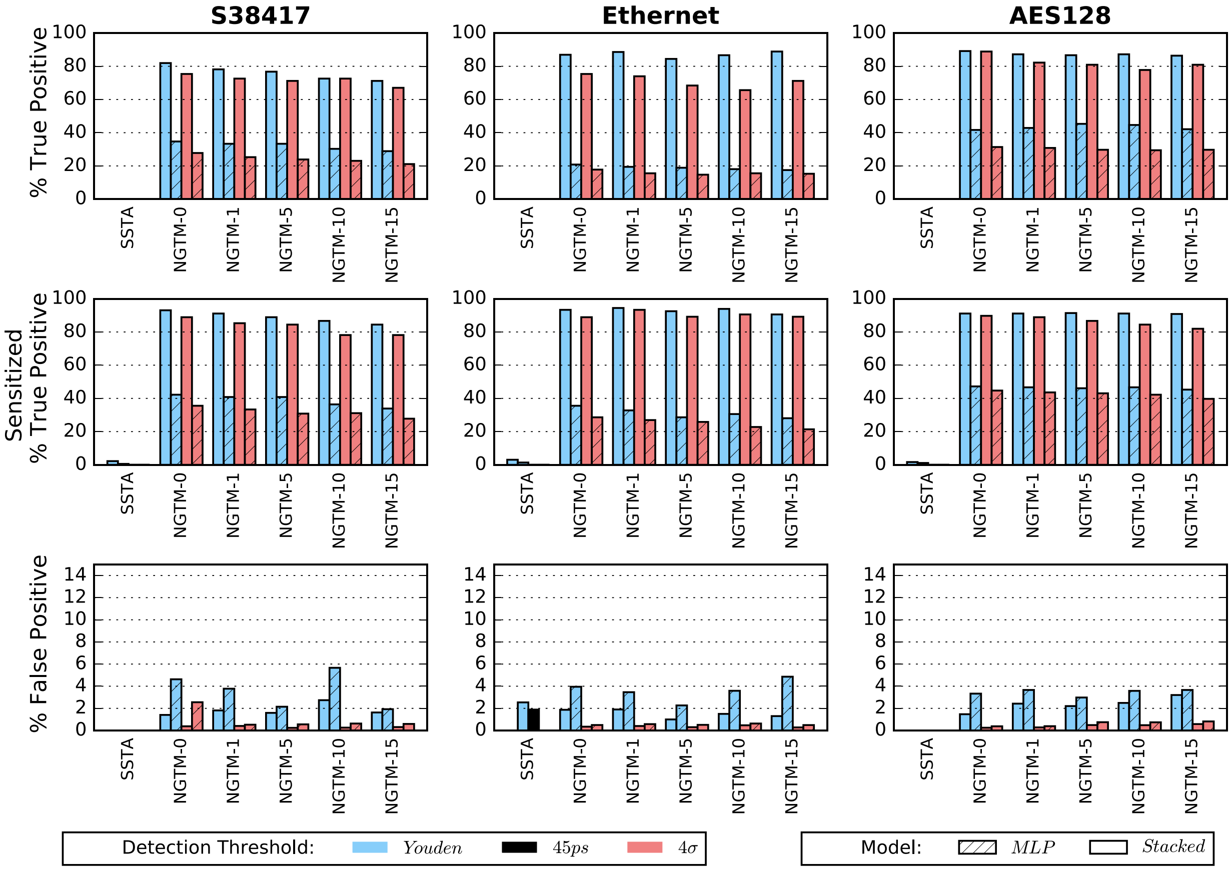


FIGURE 13. Trojan trigger detection results for 3 benchmarks. (top): Detection rate, (middle): Detection rate for sensitized designs, and (bottom): False positive rate.

TABLE 4. Threshold values used for TT and TP Trojan detection in Fast-bin in Algorithm 2 when using NGTM-10 model.

Benchmarks	LASCA				AVATAR			
	TP		TT		TP		TT	
	Youden	$4 \times \sigma_{NN}$	Youden	$4 \times \sigma_{NN}$	Youden	$4 \times \sigma_{NN}$	Youden	$4 \times \sigma_{NN}$
AES128	27.1	32.48	16.3	32.48	18.66	20.52	11.68	20.52
Ethernet	35.5	36.52	15.4	36.52	22.48	26.04	13.17	26.04
S38417	24.7	28.28	17.2	28.28	17.82	20.08	12.05	20.08

by using a stacking learning model outperforms our previous work (LASCA).

The TTs are more stealthy as they can be engineered to have a minimal delay impact on affected timing-paths. Connecting TTs to gates with high drive strength and low threshold voltage, and limiting their capacitive delay by making the TT nets short could result in a very small change in the delay of affected timing-paths. However, as discussed in section IV-C, we can sensitize the design to Trojan triggers by placing a cap on the size of standard cells used in the design, increasing the utilization in areas-to-be-protected, and by forcing high routing density over area-to-be-protected. This would force the adversary to connect the TT to cells with smaller drive strength and use longer nets to connect the Trojan trigger to the Trojan logic (TT placed further away). Figure 15 shows our SPICE setup for measuring the delay impact of a Trojan Trigger when placed in different distances from its driving cell. We have used a distributed RC model for metal 3 of a process in 32nm technology. We have simulated the impact of increasing the TT distance (and its capacitive delay) on the delay of a timing-path constructed using 5 NAND gates. Let's assume our detection threshold is 25ps; As illustrated, the TT needs to introduce an additional capacitance equal to a net that drives the TT logic placed 40um away from the affected net to add a 25ps delay increase in the delay of the timing-path. Hence, sensitization could be an effective means for increasing the detection rate of HT.

Table 5 captures the results of TP Trojan detection in all speed bins and also compares the impact of avoiding speed binning (no-binning). As reported, the speed binning provides more accurate results for TP detection compared to the No-speed-binning case. This is because the standard

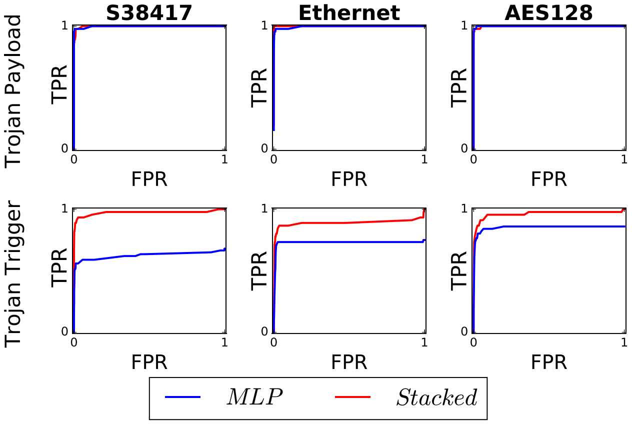


FIGURE 14. Associated ROC curve for (top): TP, and (bottom): TT, when NGTM-1 models are used. ROC curves capture the True positive rate versus False positive rate.

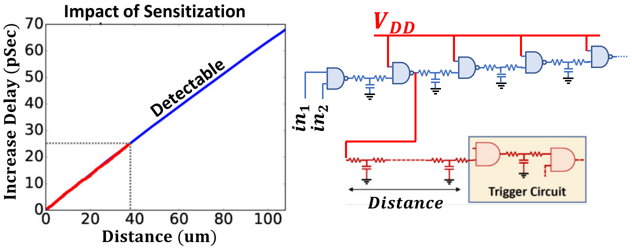


FIGURE 15. Impact of the distance of Trigger gates from the targeted path, on path delay.

TABLE 5. Percentage of False Positives (FPos) and True Positives (TPos) when AVATAR, as described in Algorithm 2 is used for detection of TP with different binning strategies (Slow, Typical, Fast, and no Binning). For this simulation, the NN (process-watchdog) is trained using a Trojan in dataset (NGTM-1 model).

Benchmarks	Slow		Typical		Fast		No-Binning	
	TPos	FPos	TPos	FPos	TPos	FPos	TPos	FPos
AES128	98.88	0.11	97.78	0.17	94.44	0.18	86.67	0.31
Ethernet	96.67	0.17	95.56	0.12	92.22	0.15	88.89	0.48
S38417	96.67	0.19	94.44	0.23	91.11	0.39	82.22	0.45

deviation of the NN-watchdog when trained over individual bins is smaller as delays are less affected by persistent process variation.

C. OVERALL TROJAN DETECTION

As discussed in section IV-C, a Trojan is detected even if one of its Triggers and Payload(s) is detected. The probability of Trojan detection could be expressed as $P(Detection) = 1 - P(Evasion)$. Trojan evasion happens when none of the Triggers or Payload(s) are detected. Consider the Probability of detecting the jth Payload to $P(P_j)$ and probability of detecting the ith Trigger of the Trojan to the $P(T_i)$. In this case, $P(Detection)$ for a Trojan with N payloads and M triggers could be expressed as:

$$P(Detection) = 1 - \prod_{j=1}^{M}(1 - P(P_j)) \times \prod_{i=1}^{N}(1 - P(T_i)) \quad (7)$$

For example, let's consider the benchmark Ethernet, using the NGTM-0 model. The HT used in this paper have 4 triggers and 1 payload. The probability of detecting Payload (in our experimental setup) is 96.67% (missing 3/90 payloads), while the probability of detecting a Trigger is 75.27% (missing 89/360 Triggers). plugging this information in equation 7 indicates 99.98% (close to 90/90) chances of Trojan detection. Therefore, we expect the rate of Trojan detection to be higher than that of TT or TP detection. Table 6 captures the result of our Trojan detection and compare that to Trojan detection of our baseline (SSTA). As reported, the overall Trojan detection of our proposed solution is between 97.78% to 100%, which is higher than TT (67.78% to 88.89%) or TP (93.33% to 97.78%) detection rate. Note that in this paper, we have considered small HT with 4 Triggers and a single XOR as Payload.

TABLE 6. Trojan detection performance of AVATAR. Percentage of True positive for Trojan payload detection is denoted as TD(payload), Trojan trigger detection is denoted by TD(trigger), and overall Trojan detection is denoted by TD(Trojan). The SSTA is the baseline used for comparison as statically shifted STA results used for Trojan detection.

Benchmark	Model	TD(Payload)	TD(Trigger)	TD(Trojan)
AES128	SSTA	14.22	0.00	14.22
	NGTM-0	97.78	88.89	100
	NGTM-1	97.78	82.22	100
	NGTM-5	96.67	80.83	98.89
	NGTM-10	97.78	77.78	100
	NGTM-15	95.56	80.83	100
Ethernet	SSTA	29.16	0.00	29.16
	NGTM-0	96.67	75.27	100
	NGTM-1	95.56	73.89	100
	NGTM-5	95.56	68.89	100
	NGTM-10	94.44	67.78	97.78
	NGTM-15	95.56	71.39	98.89
s38417	SSTA	26.11	0.00	26.11
	NGTM-0	95.56	75.27	100
	NGTM-1	94.44	72.78	98.89
	NGTM-5	93.33	73.89	97.78
	NGTM-10	97.78	72.78	100
	NGTM-15	95.56	68.89	100

TABLE 7. In this table, the result of our diagnostic test for reducing the false-positive rate of our proposed model is reported. The diagnostic test is also able to pinpoint the location of nets hosting the Trojan Trigger or Payload. The expected number of suspect nets (by the model) after running the diagnostic test is indicated by E(n).

Benchmark	NN Model	# FPo Before Diagnostics	# FPo After Diagnostics	E(n)
AES128	NGTM-0	99	0	1.39
	NGTM-1	111	0	1.41
	NGTM-5	195	0	1.47
	NGTM-10	191	0	1.48
	NGTM-15	231	0	1.47
Ethernet	NGTM-0	137	0	1.61
	NGTM-1	161	0	1.62
	NGTM-5	119	0	1.65
	NGTM-10	187	3	1.65
	NGTM-15	111	2	1.68
S38417	NGTM-0	19	0	1.66
	NGTM-1	21	4	1.66
	NGTM-5	12	2	1.65
	NGTM-10	13	1	1.57
	NGTM-15	16	0	1.54

D. RESULTS OF DIAGNOSTIC ANALYSIS

Our diagnostic test flow, as described in section IV-D, significantly reduces the false-positive rate of our Trojan detection solution. The result of running our diagnostic test is reported in Table 7. As shown in this table, the diagnostic test can

TABLE 8. **HT circuits used from the TrustHub benchmarks.**

Benchmark	HT Model	Description
S38417	T100	The TT is a comparator with activation probability of 1.4243e-70. The Trojan gains control over an internal signal after activation.
	T200	The TT is a comparator with activation probability of 1.6616e-44. The TP propagates erroneous values over four internal signals.
	T300	The TT is a comparator with activation probability of 1.6616e-44. The TP leaks the value of a specific net through side-channel signals (ring-oscillator with 29 stages).
S35932	T100	The TT is a comparator and gets activated only in the functional mode. After activation, the Trojan enables the scan enable of a part of one scan chain and leaks the value of one internal signal through a test output pin.
	T200	The TT is a comparator which only gets activated in the functional mode. After activation, it bypasses four gates of the main design by applying a corresponding dominant input value.
	T300	The TT is a comparator which gets only activated in the functional mode. The TP is a ring oscillator along a path. It is expected the path slows down when the ring oscillates.
xge_mac_scan	T700	The TT is a sequential circuit which seeks a specific 16-bit vector. The probability of Trojan activation is 1.4761e-08. Whenever Trojan gets triggered, its payload gains control over an internal signals.
	T710	The TT is a sequential circuit which seeks a specific 16-bit vector. The probability of Trojan activation is 04.557e-2. Whenever Trojan gets triggered, its payload gains control over an internal signals.
	T720	The TT is a sequential circuit which seeks a specific 16-bit vector. The probability of Trojan activation is 1.3825e-4. Whenever Trojan gets triggered, its payload gains control over an internal signals.
	T730	The TT is a sequential circuit which seeks a specific 16-bit vector. The probability of Trojan activation is 7.2317e-2. Whenever Trojan gets triggered, its payload gains control over an internal signals.

significantly reduce the false positive rate, bringing it down to zero false positives for most NGTM, regardless of regressor choice (MLP vs. Stacking). Note that the choice of regression model determines our Trojan detection accuracy, and the diagnostic test flow does not help with increasing the detection accuracy. Hence, although our diagnostic test could impressively suppress the false-positive rate, the stacking learning model still has a clear advantage over the MLP model (or other liner models). In a few cases, the diagnostic test cannot reduce the false positive to zero. The reason for this observation is that the majority of timing paths selected for the diagnostic test are affected by process variation such that they see a slight increase in their delay, which is comparable to the Trojan Trigger impact. Besides, in some cases, the diagnostic test cannot generate enough timing paths that consist of a small portion of a malicious path, limiting the analysis on each segment (net) of a malicious timing path. To further reduce the false positive, one may a) increase the number of timing paths chosen for the diagnostic test, b) run the diagnostic test on a different die (if the Trojan is inserted in all dies), c) change the selection of timing paths used for diagnosis, or d) increase the detection-threshold for Trojan detection (trading off accuracy vs false-positive rate). We have purposely not repeated the experiment with additional timing paths to illustrate that the diagnostic test could still miss some of the false positives and highlight the need for repeating the diagnostic test with one of the solutions proposed above.

Another advantage of our proposed diagnostic test, as previously suggested, is its ability to pinpoint the location (net) containing the Trojans' TT or TP. In some cases, however, we cannot produce enough test patterns for a single suspicious net, and we have to consider a set of nets for the diagnostic test. In such a case, the number of nets that should be diagnosed for Trojan is more than one. Table 7 summarizes the result of our diagnostic test. As indicated, the expected number of timing paths that may have a TT or TP Trojan, denoted by E(n), is between 1 and 2 timing paths. E(n) 's small value would significantly help with next-step verification for partially or fully-invasive Trojan detection (for verification) and save significant time scanning the IC for the Trojan.

We finally like to mention that the tester equipment's max frequency limits the type of timing-paths that could be subjected to AVATAR for a Trojan test. Besides, the minimum step size (min change in clock frequency) of the tester equipment limits the size of Trojan that could be detected by AVATAR. For example, if the Trojan delay is 20ps, with a tester step size of 100ps there is a good chance that the Trojan skips detection if the timing paths' original delay is not close to the boundary of frequency bucket, which is set by the tester's step size.

E. OTHER TROJAN BENCHMARKS

Regardless of the HT activation mechanism (e.g., combinational or sequential), the Trigger or Payload of an HT affects the delay of all timing paths passing through the affected net. Hence, as a side-channel delay solution, the accuracy of the AVATAR is independent of the HT activation mechanism and only a function of how significant the added capacitance of Trigger or added delay of Payload is. That is why in the first section of our experimental results, we investigated the chances of HT detection using a single Payload and small triggers by only using simple combinational HT circuits. However, for the sake of completeness, we have also tested our solutions using several HT benchmarks in TrustHub benchmark suit [46], [47]. Unfortunately, all HT benchmarks at the ''Fabrication stage'' on TrustHub only contain the HT-inserted circuit's Design Exchange Format (DEF). They are missing the technology library, constraint file, and the GDSII of the HT-free circuit. Please note that we assume

TABLE 9. Trojan detection result for the TrustHub benchmarks.

Benchmark	Trojan Type	HT Circuit Detection	Number FPo		Number of TT nets		Number TP nets		NN-Watchdog	
			Before Diagnostics	After Diagnostics	Total	Detected	Total	Detected	Mean	STD
S38417	T100	Yes	14	0	2	2	1	1	-0.058	7.991
	T200	Yes	23	0	8	6	1	1	0.055	7.825
	T300	Yes	20	1	4	4	NaN	NaN	-0.050	8.439
S35932	T100	Yes	30	0	8	7	1	1	0.973	9.787
	T200	Yes	27	1	15	13	NaN	NaN	0.076	10.261
	T300	Yes	31	2	6	6	NaN	NaN	-0.013	9.892
xge_mac_scan	T700	Yes	271	1	4	4	1	1	-0.041	1.682
	T710	Yes	377	0	6	4	1	1	0.002	1.748
	T720	Yes	103	0	6	6	1	1	-0.030	1.316
	T730	Yes	104	0	9	9	1	1	-0.007	1.387

that AVATAR has access to GDSII of HT-free circuit in our flow, and only the fabricated IC is untrusted. To settle this shortage, we take the HT benchmarks from the Gate Abstraction level (where both HT-free and HT-inserted netlists are available) and take the gated netlists through physical design in 32nm technology. Later, we extract the HT circuit from the malicious benchmark and manually add it to the clean and routed netlist. We then run incremental placement and routing for legalization. To simulate the fabricated chips and to assign the labels in a dataset, we take the malicious benchmarks through SPICE simulation with the settings similar to Section V-B to model a typical process drift (mentioned in Section IV-A), process variation, and tester's step size limit. Finally, we extract the slack of timing paths, and following Algorithm 1, we prepare the train-set of the NN-watchdog.

The training and test set preparation and the training of the NN-watchdog (stack learning model) are done using the methodology described previously with the difference that this time, we made sure that the HT is included in the training set to model the worst-case scenario. Table 8 explains the HT types used from the TrustHub benchmark. The result of our HT detection solution on these benchmarks are reported in Table 9. Note that HT benchmarks range within "sequential Trojans, combinational Trojans, or Trojans with no Payload." As shown in Table 9, AVATAR was able to detect all HT benchmarks by detecting most triggers and all payload for each HT circuitry. This is consistent with the results reported for our simple HT circuit, where the Trojan payload detection rate was significantly higher than the Trojan trigger detection rate. Also, note that it is only enough to detect one of its Triggers or payload(s) to detect an HT. Also, note that the number of false positives is quite low, and our diagnostic test, as expected, can significantly reduce this false positive rate.

VI. CONCLUSION

In this paper, we presented a promising learning-assisted methodology for Trojan detection based on side-channel delay analysis (using clock frequency sweeping) that does not require the availability and usage of a Golden IC. For Trojan detection, Our proposed solution relies on 1) sensitizing the netlist at design time to amplify the impact of HT, 2) training a Neural Network at test time that is used as a process tracking watchdog to model the process drift (while accounting for process variation), and 3) clock frequency sweeping test to find the start to fail frequencies for different timing paths. Our proposed solution is complemented with a diagnostic test that could reduce the false positive rate of our Trojan Trigger and Trojan Payload detection to zero (or near zero), while pinpointing to the location of HT for True-positive cases. We have reported that AVATAR could achieve over ~98% Trojan detection rate over selected benchmarks.

REFERENCES

[1] A. Yeh, "Trends in the global IC design service market," *DIGITIMES Res.*, Mar. 2012.

[2] M. Tehranipoor and F. Koushanfar, "A survey of hardware Trojan taxonomy and detection," *IEEE Des. Test Comput.*, vol. 27, no. 1, pp. 10–25, Jan. 2010.

[3] D. S. Board. (2005). *Task Force on High Performance Microchip Supply*. [Online]. Available: http://www.acq.osd. mil/dsb/reports/ADA435563.pdf

[4] N. Karimi, J.-L. Danger, and S. Guilley, "On the effect of aging in detecting hardware Trojan horses with template analysis," in *Proc. IEEE 24th Int. Symp. On-Line Test. Robust Syst. Design (IOLTS)*, Jul. 2018, pp. 281–286.

[5] A. Rawnsley, "Fishy chips: Spies want to hack-proof circuits," *Wired*, Jun. 2011.

[6] R. Johnson, "The Navy bought fake Chinese microchips that could have disarmed US missiles," Business Insider, New York, NY, USA, Tech. Rep., Jun. 2011.

[7] S. Skorobogatov and C. Woods, "Breakthrough silicon scanning discovers backdoor in military chip," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.* Berlin, Germany: Springer, 2012, pp. 23–40.

[8] K. Yang, M. Hicks, Q. Dong, T. Austin, and D. Sylvester, "A2: Analog malicious hardware," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2016, pp. 18–37.

[9] S. Adee, "The hunt for the kill switch," *IEEE Spectr.*, vol. 45, no. 5, pp. 34–39, May 2008.

[10] Y. Alkabani and F. Koushanfar, "Consistency-based characterization for IC Trojan detection," in *Proc. Int. Conf. Comput.-Aided Design (ICCAD)*, 2009, pp. 123–127.

[11] S. Bhunia, M. S. Hsiao, M. Banga, and S. Narasimhan, "Hardware Trojan attacks: Threat analysis and countermeasures," *Proc. IEEE*, vol. 102, no. 8, pp. 1229–1247, Aug. 2014.

[12] H. Salmani and M. Tehranipoor, "Layout-aware switching activity localization to enhance hardware Trojan detection," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 1, pp. 76–87, Aug. 2011.

[13] R. Rad, J. Plusquellic, and M. Tehranipoor, "Sensitivity analysis to hardware Trojans using power supply transient signals," in *Proc. Int. Workshop Hardw.-Oriented Secur. Trust*, 2008, pp. 3–7.

[14] M. Potkonjak, A. Nahapetian, M. Nelson, and T. Massey, "Hardware Trojan horse detection using gate-level characterization," in *Proc. 46th Annu. Design Automat. Conf.*, 2009, pp. 688–693.

[15] D. Du, S. Narasimhan, R. S. Chakraborty, and S. Bhunia, "Self-referencing: A scalable side-channel approach for hardware Trojan detection," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.* Berlin, Germany: Springer, 2010, pp. 173–187.

[16] K. Xiao, X. Zhang, and M. Tehranipoor, "A clock sweeping technique for detecting hardware trojans impacting circuits delay," *IEEE Des. Test*, vol. 30, no. 2, pp. 26–34, Apr. 2013.

[17] Y. Jin and Y. Makris, "Hardware Trojan detection using path delay fingerprint," in *Proc. IEEE Int. Workshop Hardw.-Oriented Secur. Trust*, Jun. 2008, pp. 51–57.

[18] J. Li and J. Lach, "At-speed delay characterization for IC authentication and Trojan horse detection," in *Proc. IEEE Int. Workshop Hardw.-Oriented Secur. Trust*, Jun. 2008, pp. 8–14.

[19] H. Li, Q. Liu, and J. Zhang, "A survey of hardware Trojan threat and defense," *Integration*, vol. 55, pp. 426–437, Sep. 2016.

[20] N. Jacob, D. Merli, J. Heyszl, and G. Sigl, "Hardware Trojans: Current challenges and approaches," *IET Comput. Digit. Techn.*, vol. 8, no. 6, pp. 264–273, Nov. 2014.

[21] K. Xiao, D. Forte, Y. Jin, R. Karri, S. Bhunia, and M. Tehranipoor, "Hardware Trojans: Lessons learned after one decade of research," *ACM Trans. Design Autom. Electron. Syst.*, vol. 22, no. 1, pp. 1–23, Dec. 2016.

[22] S. R. Hasan, C. A. Kamhoua, K. A. Kwiat, and L. Njilla, "Translating circuit behavior manifestations of hardware Trojans using model checkers into run-time Trojan detection monitors," in *Proc. IEEE Asian Hardw.-Oriented Secur. Trust (AsianHOST)*, Dec. 2016, pp. 1–6.

[23] F. Wolff, C. Papachristou, S. Bhunia, and R. S. Chakraborty, "Towards Trojan-free trusted ICs: Problem analysis and detection scheme," in *Proc. Design, Automat. Test Eur.*, Mar. 2008, pp. 1362–1365.

[24] R. Rad, J. Plusquellic, and M. Tehranipoor, "A sensitivity analysis of power signal methods for detecting hardware Trojans under real process and environmental conditions," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 12, pp. 1735–1744, Oct. 2010.

[25] H. Salmani, M. Tehranipoor, and J. Plusquellic, "New design strategy for improving hardware Trojan detection and reducing Trojan activation time," in *Proc. IEEE Int. Workshop Hardw.-Oriented Secur. Trust*, Jul. 2009, pp. 66–73.

[26] M. Lecomte, J. Fournier, and P. Maurine, "An on-chip technique to detect hardware Trojans and assist counterfeit identification," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 12, pp. 3317–3330, Dec. 2017.

[27] S. Wei and M. Potkonjak, "Scalable hardware Trojan diagnosis," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 6, pp. 1049–1057, Jun. 2012.

[28] O. Söll, T. Korak, M. Muehlberghuber, and M. Hutter, "EM-based detection of hardware Trojans on FPGAs," in *Proc. IEEE Int. Symp. Hardw.-Oriented Secur. Trust (HOST)*, May 2014, pp. 84–87.

[29] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar, "Trojan detection using IC fingerprinting," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2007, pp. 296–310.

[30] S. Wang, X. Dong, K. Sun, Q. Cui, D. Li, and C. He, "Hardware Trojan detection based on ELM neural network," in *Proc. 1st IEEE Int. Conf. Comput. Commun. Internet (ICCCI)*, Oct. 2016, pp. 400–403.

[31] F. K. Lodhi, I. Abbasi, F. Khalid, O. Hasan, F. Awwad, and S. R. Hasan, "A self-learning framework to detect the intruded integrated circuits," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2016, pp. 1702–1705.

[32] Y. Liu, Y. Jin, A. Nosratinia, and Y. Makris, "Silicon demonstration of hardware Trojan design and detection in wireless cryptographic ICs," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 4, pp. 1506–1519, Apr. 2017.

[33] V. Wang, K. Agarwal, S. Nassif, K. Nowka, and D. Markovic, "A design model for random process variability," in *Proc. 9th Int. Symp. Qual. Electron. Design (ISQED)*, Mar. 2008, pp. 734–737.

[34] A. Mutlu, J. Le, R. Molina, and M. Celik, "A parametric approach for handling local variation effects in timing analysis," in *Proc. 46th Annu. Design Automat. Conf.*, 2009, pp. 126–129.

[35] M. Gruber, *Improving Efficiency By Shrinkage: The James–Stein and Ridge Regression Estimators*. Evanston, IL, USA: Routledge, 2017.

[36] L. Breiman, "Stacked regressions," *Mach. Learn.*, vol. 24, no. 1, pp. 49–64, Jul. 1996.

[37] D. H. Wolpert, "Stacked generalization," *Neural Netw.*, vol. 5, no. 2, pp. 241–259, Jan. 1992.

[38] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 785–794.

[39] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *J. Roy. Stat. Soc., B, Stat. Methodol.*, vol. 67, no. 2, pp. 301–320, Apr. 2005.

[40] R. Tibshirani, "Regression shrinkage and selection via the Lasso," *J. Roy. Stat. Soc., B, Methodol.*, vol. 58, no. 1, pp. 267–288, Jan. 1996.

[41] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York, NY, USA: Springer, Aug. 2009, doi: 10.1007/b94608.

[42] T. K. Ho, "Random decision forests," in *Proc. 3rd Int. Conf. Document Anal. Recognit.*, vol. 1, 1995, pp. 278–282.

[43] N. J. Perkins and E. F. Schisterman, "The inconsistency of 'optimal' cutpoints obtained using two criteria based on the receiver operating characteristic curve," *Amer. J. Epidemiol.*, vol. 163, no. 7, pp. 670–675, 2006.

[44] *IWLS 2005 Benchmarks*. Accessed: Apr. 17, 2018. [Online]. Available: http://iwls.org/iwls2005/benchmarks.html

[45] A. Vakil, F. Behnia, A. Mirzaeian, H. Homayoun, N. Karimi, and A. Sasan, "LASCA: Learning assisted side channel delay analysis for hardware Trojan detection," in *Proc. 21st Int. Symp. Qual. Electron. Design (ISQED)*, 2020, pp. 40–45.

[46] B. Shakya, T. He, H. Salmani, D. Forte, S. Bhunia, and M. Tehranipoor, "Benchmarking of hardware Trojans and maliciously affected circuits," *J. Hardw. Syst. Secur.*, vol. 1, no. 1, pp. 85–102, Mar. 2017.

[47] H. Salmani, M. Tehranipoor, and R. Karri, "On design vulnerability analysis and trust benchmarks development," in *Proc. IEEE 31st Int. Conf. Comput. Design (ICCD)*, Oct. 2013, pp. 471–474.

[48] A. Apache. *Redhawk*. Accessed: Apr. 17, 2018. [Online]. Available: https://www.apache-da.com/products/redhawk

ASHKAN VAKIL received the B.Sc. degree in electrical engineering from Mazandaran University, Mazandaran, Iran, and the M.Sc. degree in electrical engineering from the University of Bridgeport, CT, USA, in 2015. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, George Mason University. He has worked earlier on nanoelectronic and carbon nano tube conductivity. His research interests include the areas of low power design, variation modeling, and hardware-trojan detection and security.

ALI MIRZAEIAN received the B.Sc. and M.Sc. degrees in computer engineering from the Isfahan University of Technology (IUT), Isfahan, Iran, in 2013 and 2016, respectively. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, George Mason University. His research interests include accelerator design for neural networks and neural network security.

HOUMAN HOMAYOUN received the B.Sc. degree in electrical engineering from the Sharif University of Technology, Tehran, Iran, in 2003, the M.Sc. degree in computer engineering from the University of Victoria, Victoria, BC, Canada, in 2005, and the Ph.D. degree from the Department of Computer Science, University of California at Irvine, Irvine, CA, USA, in 2010. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, University of California at Davis, Davis, CA. He was the Technical Program Co-Chair of the GLSVLSI 2018 and the General Chair of the 2019 GLSVLSI Conference. Since 2017, he has been serving as an Associate Editor for the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS.

NAGHMEH KARIMI (Member, IEEE) received the B.Sc., M.Sc., and Ph.D. degrees in computer engineering from the University of Tehran, Iran, in 1997, 2002, and 2010, respectively. She was a Visiting Researcher with Yale University, and a Postdoctoral Researcher with Duke University and New York University. She joined University of Maryland Baltimore County, as an Assistant Professor, in 2017, where she leads the Secure, Reliable and Trusted Systems (SECRETS) Research Laboratory. Her current research interests include hardware security and VLSI testing and reliability. She was a recipient of the National Science Foundation CAREER Award, in 2020. She serves as an Associate Editor for the *Journal of Electronic Testing: Theory and Applications* (JETTA) (Springer).

AVESTA SASAN (Senior Member, IEEE) received the B.S. degree in computer engineering, the M.S. and Ph.D. degrees from the Department of Electrical and Computer Engineering, University of California at Irvine, Irvine, in 2005, 2006, and 2010, respectively. In 2010, he joined the office of the Chief Technology Officer at Broadcom Corporation. He worked on the physical design and implementation of ARM processors, serving as a Physical Designer, Timing Sign off Specialist, and the Lead of signal and power integrity sign off in this team. In 2014, he was recruited by Qualcomm's office of VLSI technology. In this role, he developed different methodologies and in-house EDA's for accurate sign-off and analysis of hardened ASIC solutions. He joined George Mason University, in 2016, and he is currently serving as an Associate Professor with the Department of Electrical and Computer Engineering. His research interests include low power design and methodology, hardware security, computer security, cybersecurity, accelerated computing, approximate computing, and neuromorphic computing.