IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 11, 2019, accepted January 5, 2020, date of publication January 8, 2020, date of current version January 17, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2965016

A Survey on Machine Learning Against Hardware Trojan Attacks: Recent Advances and Challenges

ZHAO HUANG[1], QUAN WANG[1], (Member, IEEE), YIN CHEN[2], (Member, IEEE), AND XIAOHONG JIANG[3], (Senior Member, IEEE)

[1]School of Computer Science and Technology, Xidian University, Xi'an 710071, China
[2]Graduate School of Media and Governance, Keio University Shonan Fujisawa Campus, Fujisawa 252-0882, Japan
[3]School of Systems Information Science, Future University Hakodate, Hakodate 041-8655, Japan

Corresponding author: Quan Wang (qwang@xidian.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61572385 and Grant 61702395, and in part by the National Institute of Information and Communications Technology (NICT), Japan.

ABSTRACT The remarkable success of machine learning (ML) in a variety of research domains has inspired academic and industrial communities to explore its potential to address hardware Trojan (HT) attacks. While numerous works have been published over the past decade, few survey papers, to the best of our knowledge, have systematically reviewed the achievements and analyzed the remaining challenges in this area. To fill this gap, this article surveys ML-based approaches against HT attacks available in the literature. In particular, we first provide a classification of all possible HT attacks and then review recent developments from four perspectives, i.e., HT detection, design-for-security (DFS), bus security, and secure architecture. Based on the review, we further discuss the lessons learned in and challenges arising from previous studies. Despite current work focusing more on chip-layer HT problems, it is notable that novel HT threats are constantly emerging and have evolved beyond chips and to the component, device, and even behavior layers, therein compromising the security and trustworthiness of the overall hardware ecosystem. Therefore, we divide the HT threats into four layers and propose a hardware Trojan defense (HTD) reference model from the perspective of the overall hardware ecosystem, therein categorizing the security threats and requirements in each layer to provide a guideline for future research in this direction.

INDEX TERMS Machine learning, hardware Trojan detection, design-for-security, bus security, secure architecture.

I. INTRODUCTION

The tremendous advancements in semiconductor technology have resulted in a large number of participants coordinating during the design and manufacturing process of integrated circuits (ICs) [1], [2]. Particularly given the continuously increasing complexity of ICs, increasingly greater numbers of specialized teams and/or companies, typically dispersed geographically, are involved in this complex process to increase efficiency and manufacturability. However, due to its highly distributed nature, any stage of this complicated supply chain can be compromised by an adversary (or attacker) by implanting malicious circuits [2], [11]. Fig. 1 illustrates the stages that could be exploited in a typical design-fabrication process of modern ICs [3], [4]. Such malicious circuits, often referred to as hardware Trojans (HTs), will endanger the security and trustworthiness of the underlying hardware by, e.g., disclosing confidential information, impeding normal executions at vital points, and even prompting irreversible and fatal damage to the system [3], [11]. This fact also poses a serious threat to semiconductor suppliers and end IC users, which may include critical applications and cyber infrastructure such as mobile communications, aerospace agencies, medical electronics, military weapons, and nuclear reactors [4], [12]. Given this situation, it is both critical and challenging to study defensive strategies to alleviate the potential security threats posed by the so-called HT attacks.

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

A. HT DEFINITION, TAXONOMY, AND PROTECTION

1) HT DEFINITION AND CLASSIFICATION

HTs are defined as malicious, intentional inclusions/deletions/alterations, or inadvertent design defects of ICs or intellectual property (IP) cores that can be exploited by knowledgeable adversaries to achieve the purpose of an attacker, which may

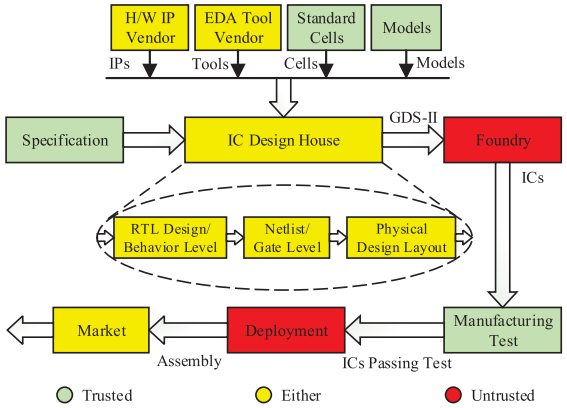

FIGURE 1. Potential stages of modern IC supply chain that can be exploited to implant Trojans [4].

result in vast economic losses and great harm to society [13], [14], [16].

HT circuits can be categorized in a variety of forms according to different features. Several studies have introduced detailed taxonomies to cover a wide range of HT instances. Early work can be traced back at least to Banga et al., who classified HTs into two categories according to the logical types, i.e., *combinational* and *sequential* [5]. However, this classification failed to take the impacts of analog trigger conditions and effective payload into account. Tehranipoor et al. presented classifying HTs into three categories based on their physical, activation, and action characteristics [6]–[8]. Although this HT taxonomy was more detailed than previous approaches, it contained only a limited number of Trojan examples, and its correlation with the IC development cycle was not obvious. Zhang et al. classified HTs into two categories, namely, *bug-based* HTs and *parasite-based* HTs, based on their impacts on the normal functionalities of the circuits [9]. However, there was not much discussion about their trigger conditions and implantation stage. Bhunia et al. proposed classifying HTs as *analog* Trojans and *digital* Trojans based on the trigger and payload mechanisms [4], [8], wherein the digital Trojan also included both *combinational* and *sequential* types.

Karri et al. integrated and extended the above classification methods and introduced an *attributed-based* HT taxonomy [36], [37]. Based on the study of Wang et al., they classified HTs based on *five* different attributes, adding considerations of design phase, location, and abstraction level. Subsequently, Moein et al. separated HTs based on *eight* different *attributes* and included *three* more attributes, i.e., logic type, physical layout, and functionality, to further improve the HT classification [17], [38]. In particular, Table 1 lists these typical studies on HT taxonomies. These previous taxonomies continue to evolve as newer attacks and Trojan types are discovered.

Here, we present a new, systematic taxonomy based on the relationship between the *locations* of HT circuits inserted into a system-on-chip (SoC) and the *targets* affected by the

TABLE 1. Typical Studies on HT Taxonomies.

Article	Class	Features	Each Category
[5]	2	Logical Types	Combinational and Sequential
[6], [7]	3	Trojan Features	Physical, Activation, and Action
[9]	2	Trojan Impacts	Bug-based and Parasite-based
[4]	2	Circuit Features	Analog and Digital
[36], [37]	5	Attributes	Design Phase, Abstraction Level, Activation, Effects and Location
[17], [38]	8	Attributes	Design Phase, Abstraction Level, Activation, Effects, Location, Logic Types, Physical Layout, and Functionality

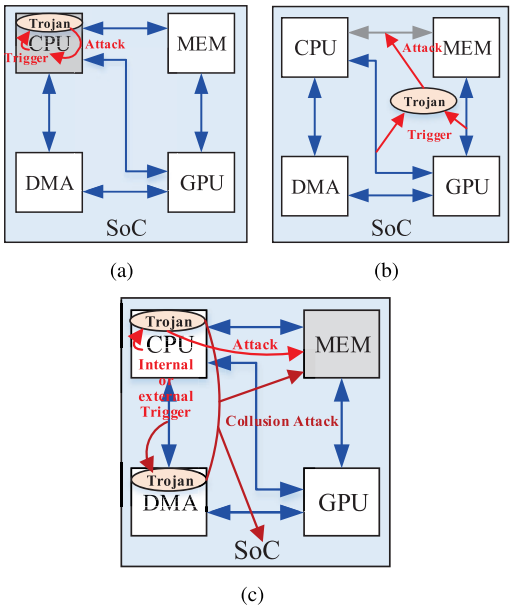

FIGURE 2. Typical representatives of the three types of Trojan threats [18]. (a) IP-level HTs. (b) Bus-level HTs. (c) SoC-level HTs.

Trojan attacks when triggered. Thus, the HT circuits can be categorized into three types: (1) IP-level HTs, (2) bus-level HTs, and (3) SoC-level HTs, as shown in Fig. 2.

- **IP-level Trojans –** this type of HT is implanted into the individual function IP cores of the SoCs (or into simple logic circuits) and triggered by internal conditions such as rare nodes/nets. When activated, they can only affect the specific IP cores in which they are implanted (as shown in Fig. 2 (a)) [18].
- **Bus-level Trojans –** this type of HT is connected with the network fabrics of buses (i.e., linkers [26], [27]) or embedded into the modules of the on-chip buses (such as routing nodes [19], [108], control units, and network interfaces [28]). They are mainly triggered by internal rare signals or by data flowing through the on-chip buses and can tamper with the integrity and reliability of interconnection fabrics and routers in the SoC designs (as shown in Fig. 2 (b)) [18]–[20].
- **SoC-level Trojans –** this type of HT is inserted into the individual function IP cores of the SoCs and can affect the behaviors of other IP cores. Alternatively,

TABLE 2. Typical Studies on Broad Classifications of HT Countermeasures.

Article	Class	Each Category
[6]	1	HT Detection
[8], [35]	2	HT Detection and HT Prevention
[7], [10]	2	HT Detection and Design for Hardware Trust (DFHT)
[4]	3	HT Detection, Design for Security (DFS), and Runtime Monitoring
[16]	3	HT Detection, Design for Trust (DFT), and Split Manufacturing for Trust (SMFT)
[13], [14]	3	HT Detection, HT Diagnosis, and HT Prevention

several untrusted third-party (3P) IP cores, which are obtained from 3PIP vendors, may collaborate to perform collusion attacks on other IP cores or even the SoCs. Moreover, the trigger conditions of this type of HT vary, e.g., rare conditions occurring within the implanted IP cores, special instructions or external sequences, etc. This type of HT is designed to affect the overall system functions rather than the infected IP cores (as shown in Fig. 2 (c)) [20].

2) CLASS OF MITIGATION STRATEGIES

Substantial research efforts, from both academia and industry, have been devoted to the studies of HTs [11], [21], [22]. Table 2 lists several typical studies on the classification of HT protections. The initial mitigation strategies were mainly focused on HT detection approaches and design for security (DFS) [6]–[8], [35], [39]. HT detection is a pre-/post-silicon detection technique that is used to determine whether the hardware is contaminated by malicious logic or by an undesired functional unit [13], [14], [16], [23], whereas DFS[1] methods mainly prevent the inclusion of malicious circuits at hardware design time or facilitate detection at test time or run time [21], [24].

Bhunia et al. attempted to classify the mitigation strategies as HT detection, DFS, and runtime monitoring according to the effective stage of each kind of technique [4]. The runtime validation strategies were separated from DFS and employed as a major class of online monitoring approaches to safeguard against potentially undetected HT threats at run time. Xiao et al. categorized the defensive methods as HT detection, design for trust (DFT), and split manufacturing for trust (SMFT) [16]. Similarly, the split manufacturing approaches were used as a separate class of protections since these approaches can prevent the insertion of HTs at fabrication time. Furthermore, Li et al. also categorized the countermeasures for HTs into 3 classes: HT detection, HT diagnosis, and HT prevention [13], [14]. Different from the works of Bhunia and Xiao et al., they added a new class of protections to the previous 2 categories, i.e., HT diagnosis. An HT diagnosis approach is designed to identify the locations, types and trigger conditions of the HT circuits in an IC or IP core [14]. However, research on the classification of HT protections still faces some challenges, especially with the emergence of new HT countermeasures, and a new category of protection strategies for HTs is still required.

Based on the HT taxonomy discussed in Section I.A.1, we present that protections against HT threats can be categorized into 4 broad classes of solutions, namely, 1) HT detection, 2) DFS, 3) bus security, and 4) secure architecture. *HT detection and DFS strategies are intended to defend against IP-level HTs.* Moreover, research on HT defense techniques has also been expanded to bus security (e.g., network-on-chips, abbreviated as NoCs, [25]–[27], [108], and advanced-high-performance [29], [30], etc.) and secure architecture [18], [20], [31]–[34]. *These countermeasures are special DFS strategies and are meant to address the integrity problems of on-chip buses incurred by bus-level HTs and suspicious behaviors created by SoC-level HTs at runtime, respectively.* In particular, Table 3 lists the key distinctions between IP-level HTs, bus-level HTs and SoC-level HTs, as well as the corresponding mitigation countermeasures.

From Table 3, it can be concluded that IP-level HTs appear to be a special case of SoC-level HTs since they have some overlap in their locations and trigger mechanisms. However, they have significant differences in the following respects. (1) The design and attacker location are different. IP-level HTs are relatively simple and are designed to be implanted in an individual IP core (or simple logic circuit) by adversaries during the design or manufacturing stage, whereas SoC-level HTs come in many varieties and may exist in one or more 3PIP cores provided by untrusted IP vendors. (2) The impacts are different. IP-level HTs only affect the specific IP cores in which they are embedded, whereas SoC-level HTs more strongly impact other IPs or even the general SoC functions, rather than the implanted IP cores. (3) Most importantly, the defense strategies against these two types of HT threats are different. IP-level HTs are typically detected via circuit features, such as functional or structural features, or parametric characteristics, e.g., power or path delay, whereas SoC-level HTs are identified by using dynamic behavior analysis of the whole SoC design. (4) In addition, the effective application stages of the respective defense strategies are also different (see Table 3).

On the other hand, bus-level HTs are also clearly different from SoC-level HTs, which can be embodied in the following aspects. (1) The design and attacker location are different. SoC-level HTs are implanted inside one or more IPs, while bus-level HTs are embedded into the modules of the on-chip buses or connected with the buses, not inside the IPs. (2) The impacts are different. Bus-level HTs can only tamper with the interconnection among cores, while SoC-level HTs affect the function of other IPs and even the whole SoCs. (3) Most importantly, the defense countermeasures to address these two types of HT threats are also different. The defense strategies for alleviating the bus-level HTs only concern the linker and router behaviors, while the defense strategies for resisting SoC-level HTs incorporate the behaviors of IP cores or the overall system together.

[1]In most of the survey articles on HTs, DFS, HT prevention, DFHT, and DFT have similar meanings. Therefore, this paper will uniformly adopt DFS in the following description.

TABLE 3. Key Distinctions between IP-level, bus-level and SoC-level Trojans.

Property \ Type	IP-level HTs	Bus-level HTs	SoC-level HTs
Location	Individual function IP cores (or simple logic circuits)	On-chip buses such as linkers, routers, or interfaces	One or more function IP cores
Attack Target	IP cores (or simple logic circuits) in which HTs implanted	Fabrics, linkers, routers, or interfaces	Other IP cores or even the entire SoCs
Trigger Condition	Internal rare nodes/nets	Internal rare signals or data flowing through the on-chip buses	Internal rare nodes/nets, or special instruction/sequence external
Attack Pattern	Relatively simple	Normal	Diversity
Trojan Impact	Only affect the specific IP cores in which HT implanted	Tamper the integrity and reliability of interconnection fabrics or routers in SoCs	Impact the behavior of other IPs or even the overall SoC functions
Defense Strategy	Exploit static trust verification techniques for HT detection	Extract and identify traffic patterns of on-chip buses	Validate potentially suspicious activities by dynamic or formal analysis
Effective Stage	Design / Test time	Run time	Run time
Technique Feature	Feasibility and impact analysis of Trojans in individual IP cores (or simple logic circuit)	Real-time matching of abnormal traffic linker or router features	Fine-grained SoC behavior analysis and assertion-aware security decision
Deficiency	Fail to address error correction or recovery	Continuous verification of data features exchanged across on-chip buses	Focus mainly on in-field protections

B. MOTIVATION AND CONTRIBUTIONS

Several excellent surveys have been devoted to designs, categorizations, and protections for HT problems [1], [4], [7], [8], [13]–[17], [21], [35]–[39]. For example, Sumathi et al. provided a comprehensive review of complex HT threats and illustrated feasible countermeasures against HT attacks for PLD and ASIC life cycles [1]. Zhang et al. highlighted the potential security and trust threats associated with FPGA-based systems from a market perspective and discussed the relevant solutions available for each party [15]. In 2016, Bazzazi et al. explained the taxonomies of HT threats and presented a new classification of HT detection and counterattack techniques [8]. Li and Zhang et al. reviewed the specific HT threats faced by the parties involved in the IC development process and described up-to-date HT defense solutions [13], [14], [35]. Subsequently, Xiao et al. summarized the research advancements and lessons learned concerning HT problems in recent years and explored possible future trends [16]. In addition, Jacob et al. also elaborated the feasibility of Trojan insertions and corresponding defense strategies at each stage of the IC development and production chain [39]. Table 4 summarizes the primary research contents of several related surveys published in recent years.

TABLE 4. Primary Research Contents of Several Related Surveys.

Article	Main Research Contents
[1]	Review the potential HT threats and defense solutions at each stage of PLD and ASIC life cycle.
[4]	Describe the complex HT threat models and feasible protections against HT attacks in specific fields of ICs.
[8]	Explore the taxonomy of HT threats and provide an overview of HT detection and counterattack techniques.
[15]	Discuss the potential security and trust threats for the parties involved in the FPGA-based market model.
[13], [14]	Explain the feasibility of HT insertions faced by the parties involved in the IC life cycle models and describe up to date HT defense solutions.
[16]	Introduce a comprehensive adversarial model taxonomy, review the defense strategies corresponding to each class, and explore possible future trends.
[17], [38], [36], [37]	Present the attribute-based HT classification methods.
[21]	Illustrate the robustness of specific DFS techniques to prevent HT threats and examine their limitations.
[35]	Analyze the possible infection scenarios and the detection mechanisms of HTs for hardware development process.
[39]	Evaluate the feasibility of Trojan insertions and the practicability and cost of Trojan detection techniques at each stage of the IC development and production chain.

Note that the aforementioned surveys primarily focus on HT threats and corresponding HT detection (such as reverse engineering [40]–[43], logic testing [44], [45], and side-channel analysis [46]–[49]) and prevention (such as detection assistance [50]–[52] and implantation prevention [53]–[55]) techniques. Compared with the existing surveys, this paper intends to review the specific HT threats and defense solutions from a different perspective of SoC life cycle. We identify the potential HT types at each stage of SoC development and examine the state-of-the-art HT countermeasures related to each stage. The recent success of machine learning (ML) techniques in many research domains has inspired both academic and industrial communities to explore the potential of applying ML to address various HT attacks [11], [56]. A number of achievements utilizing ML for HT defenses have emerged in the last decade. However, this important progress has not been systematically reviewed in previous surveys [4], [13], [14], [16], [21], [35], [39]. Elnaggar et al. conducted a survey on the applications of ML in the general area of hardware security, including HT attacks, reverse engineering, IC counterfeiting, side-channel attacks, and IC overbuilding [56]. However, a dedicated survey on the applications of ML to HT defenses is still not available in the literature. In addition, most of the aforementioned surveys focused on IP-level HT threats and may overlook bus-level and SoC-level HT attacks or corresponding mitigation strategies.

To fill this gap, we first investigate publications with regard to the HT problems found in the IEEE Xplore

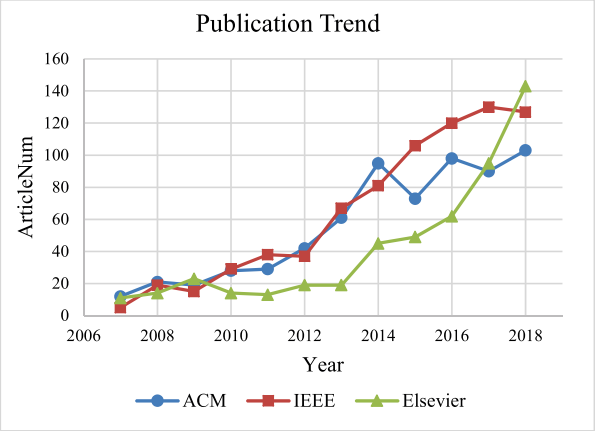

FIGURE 3. Publication trends on HTs from IEEE, ACM, and Elsevier.

digital library,[2] the ACM DL digital library,[3] and Elsevier ScienceDirect.[4] Fig. 3 illustrates the number of articles on HTs published in the digital libraries annually starting from 2007 to 2018. In particular, we have specifically reviewed the recent progress in utilizing ML techniques against HT threats and revealed the potential problems, challenges and possible future research directions in this domain. Note that the up-to-date bus-level and SoC-level HT attacks and ML-related mitigation strategies have also been elaborated in this paper. It is expected that this survey will serve as a general guidebook for those who want to contribute to the study of HTs.

The main contributions of this article are summarized as follows:

(1) This article, for the first time to the best of our knowledge, systematically investigates the latest research progress in applying ML technology in the fields of HT detection and prevention.
(2) The explicit analysis highlights the new advances in using ML-based techniques in HT defense domains to address the IP-level, bus-level and SoC-level Trojan threats from the perspectives of HT detection, DFS, bus security, and secure architecture.
(3) We also analyze some important aspects related to the application of ML algorithms in HT defense domains and discuss its potential challenges facing the state-of-the-art.
(4) In addition, we propose a reference model for recommending possible strategies that each layer should be able to utilize along with future directions for HT research based on the analysis of the above situations.

The remainder of this article is organized as follows. In Section II, we present typical ML models used in HT defense studies. A detailed discussion of the HT threat models and the advancements in applying ML for HT defenses is presented in Section III. Section IV discusses the lessons learned and challenges arising from the previous studies. Section V gives a reference model of HTs from the perspective of the overall hardware ecosystem and provides a discussion on future research directions. Finally, we conclude this article in Section VI.

[2]http://ieeexplore.ieee.org/Xplore/home.jsp
[3]http://dl.acm.org/
[4]http:/www.sciencedirect.com/

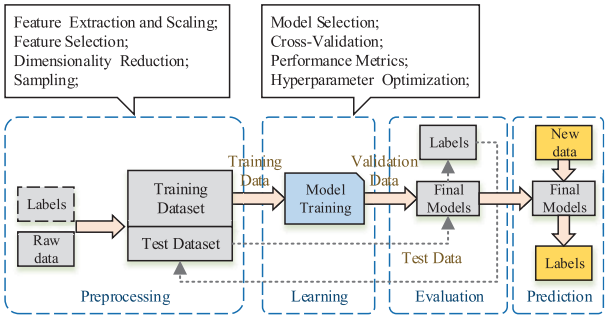

FIGURE 4. The general flow of ML framework.

II. ML MODELS USED IN HT DEFENSES

This section presents typical ML models that have been widely utilized to solve HT problems. Section II.A depicts the general procedure of applying ML, as well as commonly exploited ML algorithms. Section II.B analyzes the current publication trend of ML in the HT defense domain.

A. ML MODELS

ML is a class of algorithms that primarily focuses on generating the "models", namely, learning algorithms, from a large amount of historical "data" and then utilizing these trained models for prediction or classification [57]. As illustrated in Fig. 4, the general procedure of applying ML algorithms involves the following: (1) In the preprocessing phase, the relevant features are first selected, and then, data with these features are extracted from the raw data so that they can be utilized to distinguish the different values of the target outputs. After that, several data cleansing and feature engineering operations, such as principal feature selection, scaling, and dimensionality reduction, are executed to generate the sample dataset for learning. (2) In the learning phase, the appropriate learning algorithms are selected and executed to derive the models from the training dataset. Such models correspond to a certain underlying law about the data. Then, operations such as cross-validation, result evaluation, and hyper-parameter optimization are carried out to acquire the final models. (3) In the evaluation phase, the final models are tested on the test dataset to evaluate their performance. In practice, evaluation criteria can be selected and customized according to different scenarios. (4) In the prediction phase, the final models are exploited to infer the predicted values of the target output for the newly input data.

According to the nature of the data types processed by ML, there are two major categories of learning tasks that are widely explored in HT defense domains: supervised learning and unsupervised learning. The supervised learning technique utilizes the labeled data to perform model training

and selects the final models to predict the target classes of the newly input data; in contrast, unsupervised learning techniques focus more on revealing the potential relations characterizing the data by learning from them when the data labels for all the target classes are unavailable. In particular, the major difference between supervised learning and unsupervised learning is whether the data labels are available.

Below, we present an overview of the most frequently explored ML algorithms, feature selection and dimensionality reduction methods, and several optimization and model enhancement techniques in HT defense studies.

1) SUPERVISED LEARNING

- **Artificial Neural Networks (ANNs)** [58] are an abstraction and simulation of the human brain. They can be viewed as a mathematical model from an algorithmic perspective and construct various networks according to different connection patterns. ANNs have been successfully implemented in diverse applications for classification, including HT detection.
- **Support Vector Machine (SVM)** [59] is a two-class classification model with the largest interval in the feature space. SVM's learning strategy is to maximize the interval, which can ultimately be transformed into solving convex quadratic programming problems. SVM is suitable for HT detection and has been widely applied in this area.
- **Bayesian Classifiers** [63] are a type of classification algorithm based on probability statistics and misjudgment losses. Bayesian classifiers can achieve the minimum error rate.
- **One-class Classifiers** are variants of supervised models; they are used when one wants to determine whether the new training data belong to a particular class (i.e., whether the data are normal or abnormal data). Thus, one-class classifiers have been proposed for novelty and anomaly detection, which can be regarded as one-class classification problems. At this point, the boundary of the training data needs to be learned rather than the maximum margin. **One-class ANNs** [64] and **One-class SVM** [65] are two representative one-class classifiers.
- **Back-propagation NNs (BPNNs)** [66], [67] are a kind of feed-forward ANN that may contain multiple hidden layers. BPNNs can adjust the network weights and thresholds during training to achieve a nonlinear mapping of the input and output as well as better generalization ability.
- **Extreme Learning Machine (ELM)** [68] is a single-hidden-layer feed-forward ANN (SLFN) that can randomly initialize the input weight and bias and obtain the corresponding output weight. ELM has been widely used in many fields due to its high learning speed and good generalization ability.
- **Decision Tree (DT)** [88] is a tree-structure-based learning model that contains a root node, several internal nodes, and several leaf nodes. The leaf nodes correspond to the decision results, while the other nodes correspond to an attribute test. The decision-learning process of the DT method is similar to the method humans use to make choices. In this process, each judgment question raised is a "test" of a certain attribute, and the final conclusion corresponds to the decision result that we expect. Classification tree and regression tree are commonly used DT models.
- **K-Nearest Neighbors (K-NN)** [89] classify the training dataset by measuring the distance between different eigenvalues. The general idea is that if the majority of the k most similar samples (i.e., the nearest neighbor in the feature space) in a feature space belong to a certain category, then the sample also belongs to this category, where k is usually an integer less than 20. In K-NN, the selected neighbors are all objects that have been correctly classified.

2) UNSUPERVISED LEARNING

- **Clustering Algorithms (CAs)** [72] intend to group unlabeled data into different clusters. CAs do not require priori experience as input. One major reason for using CAs in the HT protection field is the unobtainable features of golden designs/ICs, as CAs will be unaffected by this issue.
- **K-means Clustering** [69] is a variant of prototype-based CAs that attempts to discover a user-defined number (k) of clusters and partitions a set of n data points into k clusters in such a way that the resulting intra-cluster similarity is high but where the inter-cluster similarity is low.
- **Density-Based Spatial Clustering (DBSCAN)** [71] and **Ordering Points to Identify the Clustering Structure (OPTICS)** [85] are typical density-based CAs. They can find points with higher density starting from the estimated density of the corresponding nodes, and then gradually connect the high-density points into one block to generate various clusters. The advantage of density-based CAs is that clusters of various shapes and sizes can be revealed in data possessing noise.

3) FEATURE SELECTION AND DIMENSIONALITY REDUCTION

- **Genetic Algorithms (GAs)** [72] are popular heuristic algorithms. A GA can be applied for the feature selection of classification problems whose goal is to find a small subset of variables from the dataset that provides the highest classification accuracy.
- **Principal Component Analysis (PCA)** [73] is a widely used data dimensionality reduction technique in which n-dimensional features are mapped to a k-dimensional space. The newly reconstructed k-dimensional features, also known as principal components, are orthogonal. Each of the principal components can reflect most of the original variables, and the information contained therein is not repeated. In this process, the dimension k of the

low-dimensional space is usually specified by the users in advance and is much lower than n.

- **2-Dimensional PCA (2DPCA)** [71] is a variant of the PCA technique in which the covariance matrix can be constructed directly based on 2D matrices rather than being transformed previously into 1D vectors. 2DPCA can overcome the deficiencies of PCA such as high computational complexity and time consumption.

4) DESIGN OPTIMIZATION AND MODEL ENHANCEMENT

- **Adaptive Iterative Optimization Algorithm (AIOA)** [74] is a model-enhancing technique in which a specific classifier is iteratively improved using the input training data weighted by a weight vector to reduce the errors.
- **Multi-Objective Evolutionary Algorithm (MOEA)** [75] is an enhanced optimization algorithm that starts from randomly generated populations and then increasingly approaches the Pareto-optimal solutions by applying multiple-generation continuous optimization operations with adaptability.
- **Particle Swarm Optimization (PSO) Algorithm** [104], [125] is an optimizer that attempts to find the optimal solution through the cooperation and information sharing among individuals in a group. PSO is simple and easy to implement, and it does not require complex parameter adjustments. The quality of the optimal solution can be evaluated by the fitness.

Table 5 illustrates the advantages and disadvantages of each type of ML algorithm. As shown in Table 5, supervised learning is effective in addressing cases with few features and can achieve relatively better classification results. Especially for HT detection, a definitive output can be presented for each input. However, supervised learning requires golden designs/ICs, i.e., data with labels, as a reference and is also unsuitable for training on large datasets. Unsupervised learning can overcome these deficiencies. However, unsupervised learning is sensitive to noise and readily falls into local optima. In particular, the output results for each training process cannot be determined clearly. On the other hand, feature selection and dimensionality reduction methods can reduce the related features and decrease the data dimensions; however, the HT features might be deleted as redundant information because they have little impact on circuits, and the threshold values need to be determined manually. In addition, some DFS strategies can be improved by design optimization algorithms. However, this process is time consuming and takes multiple iterations to achieve the optimal solution.

B. PUBLICATION TREND OF ML APPLIED IN HT DEFENSES

Fig. 5 shows publications on ML-based techniques exploited for HT defense in the IEEE Xplore digital library, which can be used to represent the research trends in HT defense studies to a certain extent (as shown in Fig. 3). In particular, the mitigation strategies that are tackled using ML primarily involve HT detection, DFS, bus security, and secure architecture.

TABLE 5. Advantages and Disadvantages of each Type of ML in HT Defense Studies.

Machine Learning Algorithms	Advantages	Disadvantages
Supervised Learning	(1) Handle quickly for cases with fewer features; (2) Having clear output to the input; (3) Relatively better classification results; (4) Learning models is non-sensitive for noise.	(1) Require golden designs or ICs as reference; (2) Easy to over-fitting and under-fitting; (3) Improper to train large dataset; (4) Unsuitable to address multi-classification problems.
Unsupervised Learning	(1) Need no golden designs or ICs as reference; (2) Unexpected output; (3) For addressing large dataset, it is scalable and efficient; (4) Model is simple, easy to implement, and the performance does not depend on parameter choosing.	(1) Relatively poor classification results; (2) Easy to slip into local optimal solutions; (3) Learning models is sensitive for noise; (4) Cluster number is difficult to select; (5) Sensitive to initial cluster center value.
Feature Selection and Dimensionality Reduction	(1) Select effective features, decrease data dimensions and remove redundant features; (2) Improve the accuracy of HT detection and prevent over-fitting; (3) Reduce attribute space.	(1) This process will take a little more time. (2) HT features may be lost as redundant data; (3) Require to determine the threshold manually; (4) Need to make multiple attempts and adjustments.
Design Optimization and Model Enhancing	(1) Mostly used to enhance DFS strategies; (2) Decrease rare signals; (3) Improve the performance of learning models.	(1) This process is time consuming; (2) Only effective for simple combinational circuits; (3) Need many iterations to select the best solutions.

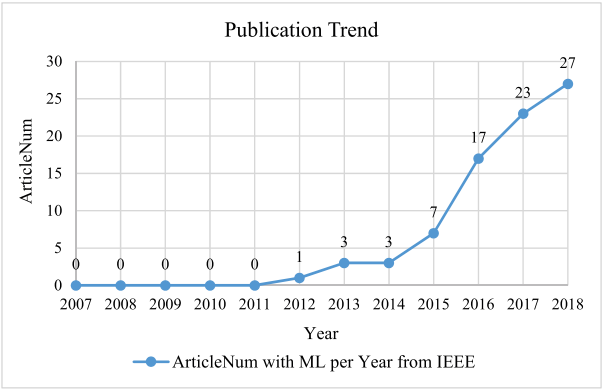

FIGURE 5. Overall publication trend of ML in HT defenses.

As can be seen from Fig. 5, approximately 80 articles were published on the studies of ML-based HT defense from 2007 to 2018. Early work in this area can be traced back at least to Jin et al., who proposed in 2012 that a one-class ANN model could be used to determine whether chips have been infected by Trojan circuits [76]. The publication count was scarce and remained basically stable from 2012 to 2014. Since 2014, the research achievements in this area have been expanding rapidly each year. As a result, it is expected that increasingly more articles on ML-based solutions will be published in this regard.

TABLE 6. HT Threat Models at Chip Layer.

Scenario	Attacker Location	Trojan Type	Specific Attack Goal	Insert Stage	Defender	Defend Strategy
A	Untrusted Foundry	IP-level Trojans	Leak Sensitive Information / Modify Functionality / Reduce Reliability / Irreversible Destruction etc. (by maliciously modifying circuit structures, functionalities, or parameters)	Fabrication	3PIP Vendor / SoC Developer	HT Detection
B	Untrusted EDA Tool or Employee			Design	3PIP Vendor / SoC Developer / Foundry	DFS / HT Detection
	Untrusted 3P Design House					
C	Untrusted 3PIP Vendor			Design	SoC Developer / Foundry	DFS / HT Detection
D	Untrusted SoC Developer	Bus-level Trojans	DoS / Traffic Diversion / Routing Loop / Core Spoofing (through untrusted links and/or routers)	Design / Fabrication	Foundry / System Designer / Integrator / End User	Bus Security
E	Untrusted SoC Developer or 3PIP Vendor	SoC-level Trojans	Interception / Interruption / Divert / Tamper / Masquerader / Fault Injection / Failure / Collusion Attack (through untrusted 3PIP cores)	Design / Fabrication	Foundry / System Designer / Integrator / End User	Secure Architecture

III. THREAT MODELS AND ML-BASED COUNTERMEASURES

This section introduces the major advances of ML techniques applied in HT defenses. Section III.A reviews the HT threat models in the modern IC supply chain from the perspective of the chip layer. Section III.B details the advances in ML-based countermeasures from the aspects of HT detection, DFS, bus security and secure architecture.

A. THREAT MODELS

HTs can be implanted at any stage of the design and fabrication process, which leads to diverse threat models [16]. From the perspective of the chip layer, the overall design-fabrication process of an SoC design can be divided into three stages: IP core development, SoC development, and fabrication [77]. Therefore, *five* types of participants in this process, i.e., 3P design houses or employees, commercial electronics design automation (EDA) tools, 3PIP vendors, SoC developers, and foundries, have opportunities to implant HTs. Table 6 systematically summarizes five typical scenarios of the chip-layer HT attacks that have appeared in the literature.

Each scenario is explained as follows:

- **Scenario A (i.e., untrusted foundry):** An attacker at the foundry could implant an HT instance into the design by manipulating the lithographic masks. These HTs appear in the forms of modification to the transistors, gates, and interconnects into the original layout.
- **Scenario B (i.e., untrusted EDA tool, employee, or 3P design house):** Because increasingly more specialized IC designers and tools have become involved in the design-fabrication process, HTs might be inserted by untrusted 3P commercial EDA tools or rogue designers within the in-house team (also called insider threats) [16]. Moreover, customers may also outsource their specifications to offshore 3P design houses, and these untrusted design houses may add additional modules or functions to the original design.
- **Scenario C (i.e., untrusted 3PIP vendor):** SoC developers may purchase and employ 3PIP cores to complete their SoC designs. These IP cores provided by untrusted 3PIP vendors to customers may contain malicious logic or backdoors.
- **Scenario D (i.e., untrusted routers or links in on-chip bus):** An adversary may compromise the integrity of the on-chip buses using malicious routing nodes or traffic links that have been infected with HTs. Such malicious bus fabrics would then be integrated into the SoCs.
- **Scenario E (i.e., untrusted SoC developer):** This attack scenario assumes that an untrusted SoC developer may develop SoC designs infected with SoC-level HTs or integrate the soft/firm/hard IP cores from untrusted 3PIP vendors containing such HT circuits that can separately or jointly impact other IP cores or the functions of the overall SoC functions.

B. ML-BASED COUNTERMEASURES

ML has been extensively exploited in a variety of HT defenses. Here, we consider the *four* primary types of countermeasures that can be applied to one or more attack scenario: 1) HT detection that utilizes available circuit features, e.g., structures, functionalities, and parameters, to reveal the HTs (i.e., Scenarios A, B, and C); 2) DFS that exploits security design strategies to enhance the trustworthiness of IC designs or intends to facilitate HT detection, prevention, and monitoring via on-chip modules (i.e., Scenarios B and C); 3) bus security that aims to identity malicious on-chip traffic behaviors created by bus-level HTs (i.e., Scenario D); and 4) secure architecture that is designed to resist SoC-level HT threats and secure the original IC designs from an architectural point of view (i.e., Scenario E).

1) HT DETECTION

HT detection techniques are typically used to verify whether any unwanted circuit is implanted in designed or fabricated ICs. Thus, ML-based techniques have seen progress in terms of (1) reverse engineering, (2) circuit feature analysis, and (3) side-channel analysis. Fig. 6 shows a summary of ML techniques applied in these aspects. In particular, Table 7 briefly elaborates the primary research contributions and innovations of ML algorithms applied in HT detection according to Fig. 6.

TABLE 7. Main Research Contributions and Innovations of ML in HT Detection.

HT Detection	Article	Main Research Contributions and Innovations
Reverse Engineering	[65], [69]	One-class SVM or K-means can classify or cluster IC images in destructive RE.
	[82]	Pattern mining can reverse the gate-level netlists of IC designs to input-output track model.
Circuit Feature Analysis	[70]	Dynamic CAs can cluster LTS features in the gate-level netlists of IP cores.
	[59], [61]	Two-class SVM or multiple ANN can classify Trojan-related features extracted from the net of ICs.
	[60]	RF can be used to select the best set of Trojan-infected features from the original IC features.
	[62]	Voting Ensemble of RF, Bayes and AdaBoot can classify functional and structural features of ICs and increase the HT detection accuracy.
Side-Channel Analysis	[68]	ELM can classify the electrical current features of ICs.
	[66], [67], [88], [89], [90], [91]	BPNN, KNN, DT, NBC, DL, PCA+Markv. CAs and CA ensemble can classify or cluster power-consumption, trace, or profiling features of ICs.
	[86]	One-class SVM with the RBF kernel can classify EM traces infected by HTs.
	[73]	PCA+one-class SVM can preprocess and classify the transmission power waveform.
	[87]	DFT+two-class SVM can preprocess and classify the frequency domain of power-consumption converted from time domain.
	[71]	2DPCA+DBSCAN can preprocess, cluster and locate HTs by using the power mapping of thermal radiation.

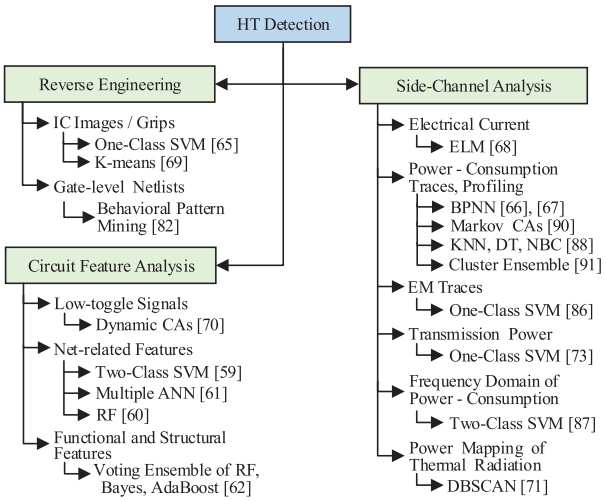

FIGURE 6. Summary of ML in HT detection.

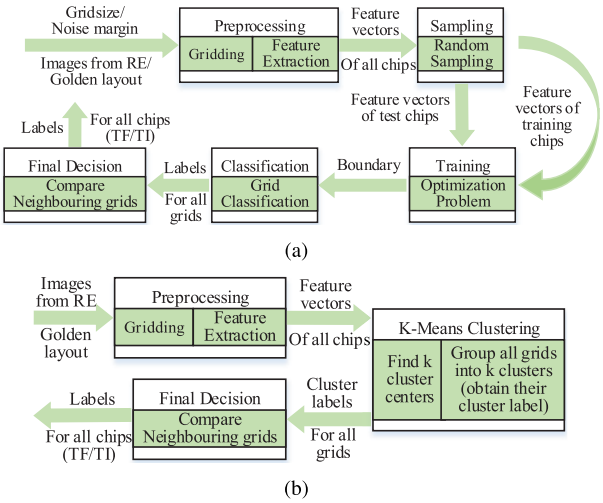

FIGURE 7. Block diagram of ML-based HT detection methods. (a) SVM-based Trojan Detection Method [65]. (b) K-means-based Trojan Detection Method [69].

As shown in Table 7, the main application of ML methods in these aspects is to classify or cluster IC-related features to improve the HT detection accuracy. Next, we present the main elements of each ML-based approach, as well as the advantages and disadvantages between ML-based and non-ML-based methods in detail.

- **Reverse Engineering**

A reverse engineering (RE) technique is utilized to repackage the ICs and obtain microscopic images of each layer to reconstruct the original design of the end products [4]. This technique has the potential to achieve very high accuracy in identifying any modifications to the ICs. However, such an irreversible approach might require several weeks or even months to apply this intrusive process to a complex IC, making its application usually limited to the testing of a limited number of IC samples [78].

However, it is still attractive to perform destructive reverse engineering on a limited number of IC samples to obtain the characteristics of a golden batch of ICs [79], [80]. For instance, Bao et al. presented an efficient and robust reverse engineering-based method to identify HT-free ICs via ML [65], [69]. By using the one-class SVM and K-means technique successively, a classifier was developed to automatically learn how to distinguish between the expected and suspicious structures in an IC, as shown in Fig. 7. Such procedures can simplify traditional 5-step reverse engineering to 3 steps and avoid the need to generate and manually enter the gate-level netlists extracted from the image-based IC designs in the last 2 steps. Moreover, the performance of K-means clustering does not depend on the choice of parameters, making it easier to train and adjust than SVM. However, this intrusive process still requires golden designs, i.e., Trojan-free designs.

Moreover, non-destructive reverse engineering, which attempts to reverse the gate-level netlist of an IC design into a high-level description of the control logic, such as finite state machines [42], [43], [81], has also been investigated. For instance, Li et al. proposed reversing the unknown ICs into the

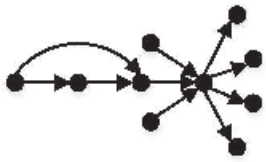

FIGURE 8. LTS-based directed graph caused by HTs [70].

input-output track model via behavioral pattern mining and discovering the Trojan circuitry through behavioral pattern matching [82]. Though non-intrusive, such a process only outlines an approximate functional model of the original design and cannot fully represent its actual function design. In particular, non-destructive reverse engineering still suffers the same deficiencies as destructive engineering, e.g., only being valid for simple logic circuits and being costly and time consuming.

- **Circuit Feature Analysis**

An adversary may create a HT instance that is activated under rare conditions or implanted into IC designs as a redundancy module. Due to their stealthy nature, such Trojans cannot be detected during traditional formal testing [84], [85]. Therefore, circuit features, e.g., functional or structural features, extracted from the gate-level netlists can be quantified and analyzed to potentially identify whether a net or gate is suspicious, wherein the switching activity and net feature are two quantitative metrics that are commonly utilized for HT detection [83].

Zhou et al. presented a structural feature matching method to detect sequential HTs in 3PIP cores [70]. By analyzing the structural features of less-toggled signals (LTS) in the gate-level circuits of IP cores, this method first abstracts the LTS caused by HTs into a directed graph (as illustrated in Fig. 8) and then utilizes dynamic CAs for structural feature matching. On the other hand, n features extracted from each net can be utilized to differentiate Trojan-infected nets from normal nets. For example, Kasegawa et al. extracted five HT-net feature values from each net in gate-level netlists and learned them using a classifier such as SVM or ANNs [59]. After that, the trained classifier can be used to classify a set of features from an unknown gate-level netlist. This method can increase the true positive rate (TPR) of detecting an HT circuit; however, there are deficiencies in terms of the true negative rate (TNR) and average accuracy [61]. Additionally, they applied RF to select the best set of Trojan features and reduce the original 51 features to the most effective 11 features, which are then utilized to classify the target nets [60]. In addition, Hoque et al. proposed a systematic learning-based approach to verify the trustworthiness of untrusted hardware 3PIPs [62]. Unlike existing learning models that rely only on structural features, they incorporated both functional and structural features to create a robust training set and used an average probability voting ensemble combined with multiple learning models to improve the HT detection efficiency.

- **Side-channel Analysis**

Side-channel analysis refers to measuring circuit parameters, such as power, path delay, temperature, and electromagnetic (EM) radiation profiles, to isolate a Trojan-infected IC from the golden ICs. This analysis fully utilizes the advantage of the parametric variations in side-channel information created by extra circuitry and/or the activities from HTs. However, the effectiveness of HT detection through side-channel analysis largely relies on the signal-to-noise ratio (SNR) and the Trojan-to-circuit ratio (TCR) [4] because the side-channel features could be affected by process variations (PVs) and noise.

ML is expected to overcome the deficiencies when incorporated with side-channel analysis, thereby improving the SNR [65], [86], and research along this direction has drawn increased attention in recent years.

Some researchers have attempted to apply ANNs to side-channel analysis. For example, Wang et al. presented an HT detection method to indicate whether the ICs are Trojan-infected by sampling and classifying the current features using an ELM [68]. Li et al. extracted the nonlinear features from the power-consumption through the HT detection model established by the BPNN [66], [67]. The ELM and BPNN can be used for feature extraction and inference because they can better retain and extract useful information for analysis, thereby avoiding the inaccuracy of manual modeling. However, there is a lack of preprocessing of the sampled side-channel features and the results of each training may be slightly volatile.

SVM can overcome the instability of ANNs during training and theoretically enhance the HT detection capability and classification accuracy. SVM is also a relatively general classification algorithm employed in side-channel analysis. For instance, Dimanto et al. proposed that HT detection can be modeled as an outlier detection problem, and the effects of an HT instance on EM traces can be directly identified through one-class SVM with the RBF kernel [86]. However, this process only contains a simple feature selection and fails to substantially improve the SNR. In contrast, Liu et al. presented a combined PCA and SVM approach (i.e., PCA+SVM) to detect the covert communication-type Trojans by utilizing the transmission power waveform [73]. Iwase et al. suggested converting the power waveform data from the time domain to the frequency domain through a discrete Fourier transform (DFT) and then conducting HT detection using SVM [87]. The PCA+SVM and DFT+SVM methods first preprocess the sample features and extract the effective part, which is more conducive to detecting and classifying HT instances. In addition, the PCA+SVM method is more accurate than the combined PCA and minimum volume enclosing ellipsoid (MVEE) method (i.e., PCA+MVEE) [73].

Furthermore, Lodhi et al. developed a runtime HT detection approach based on an online adaptive ML model [88]. The power profiling of a given micro-controller instruction set is first extracted and then classified by K-NN, DT, naive Bayesian (NBC), and deep learning (DL). This approach is

TABLE 8. Comparisons of Positives and Negatives between ML-based and non-ML-based Approaches in HT Detection.

HT Detection Technology	Non-ML based Strategies		ML based Strategies	
	Positives	Negatives	Positives	Negatives
Reverse Engineering	• Include physical level and design level • Physical level reverse is irreversible • Design level reverse is a non-destructive method • Accuracy could be 100%	• Costly, time consuming • Physical level reverse is a destructive method • Effective for simple logic circuits • Require golden ICs as reference	• No golden chips are available. • Automatically learn to identify suspicious circuits • Simplify traditional 5-step reverse engineering to 3 steps • Avoid generating and entering the gate-level netlists manually	• Require golden designs as reference • Performance of classification models relies on parameter • Clustering models are sensitive to noise • Mainly valid during test time
Circuit Feature Analysis	• Typical heuristic methods • Easy to implement and effective in practice • Extract and quantify circuit characteristics • Can detect the suspicious statements or modules	• Require golden designs as reference • Fail to detect implicit HT • Manual post-processing is required to analyze if it is (part of) a HT circuit • Suitable for simple logic circuits	• Can identify and classify HT-net features automatically • Increase the TPR and efficiency • Extract the most important HT-net features • decrease the size of feature vectors, i.e., attribute spaces	• Require golden designs as reference • Fail to detect implicit Trojan • TNR and accuracy need to further improve • Execution time depends on the circuit scale and the quantitative metrics selected
Side-channel Analysis	• Non-invasive approaches • Easy to operate and implement, applied widely • High detection accuracy • No require to activate HT • Can effectively identify large Trojans	• Can be affected by noise and PVs easily • Perform poorly on the detection of small Trojans • Require golden designs/ ICs as reference • High precision requirement on equipment	• Decrease the impacts of noise and PVs • Can effectively extract relevant features, reduce data dimensions • Partly offset the golden ICs • Improve the accuracy, TPR, etc.	• HT effects may be removed as irrelevant features or noise • Performance relies on the selection of relevant features, ML models, and parameters, etc. • Increase time overhead

able to compare and analyze various ML models and choose the most suitable model [89]. However, such a process is very time consuming, and the selection of the most appropriate ML algorithm heavily relies on the detection cases and the number of attributes required.

Most of the aforementioned classification algorithms for side-channel analysis assume that there are available golden ICs (i.e. Trojan-free fabricated ICs) to refer to during training. However, such an assumption is difficult to realize in practice. To this end, researchers have also attempted to apply CAs to side-channel analysis because they do not need priori experience as input and can eliminate the dependence on golden designs/ICs to a certain degree [91].

Nowroz et al. introduced a post-silicon multi-model approach that utilizes runtime thermal radiation and power maps for HT detection [71]. Multiple scenarios are considered there. If trusted data from known chips are available for training, a supervised thresholding method can be applied to classify the chips under detection by the Euclidean distance of the feature matrix and the golden design. Otherwise, 2DPCA can be performed to address the high-dimensional power maps converted from thermal radiation data; then DBSCAN can be exploited to detect and locate the HTs. Cui et al. developed an HT detection method with the Markov distance as the clustering center [90]. The collected power-consumption is first analyzed by PCA, and then, the clustering operation is completed with the Mahalanobis distance as the class center. Multi-mode-based HT detection takes both supervised learning and unsupervised learning into account to combine their advantages. Moreover, CA based on the Mahalanobis distance performs better than the Euclidean distance in that it is designed to optimize the detection process and can overcome the shortcomings of the Euclidean distance so that it can achieve lower computational complexity and higher accuracy.

Table 8 illustrates the advantages and disadvantages between ML-based and non-ML-based methods in HT detection. For reverse engineering, conventional solutions, including physical-level and design-level reverse engineering, can achieve 100% accuracy in detecting Trojans. However, these schemes are costly, time consuming, and only effective for simple logic circuits. They are also difficult to apply in practice. In contrast, ML can overcome some of the above-mentioned deficiencies. For example, ML can simplify the reverse engineering process and learn to identify suspicious circuits automatically. However, the performance depends on the selection of learning models and corresponding parameters, and the golden designs are still required as references.

For circuit feature analysis, current studies mainly perform coverage analysis for the gate-level netlists of circuits under detection to identify suspicious statements or modules. Then, quantitative metrics are utilized to mark the signals or gates suspected of being a Trojan. Circuit feature analysis is a typical heuristic detection method and is effective in practice. However, the limitations are that such methods are only suitable for simple logic circuits. Even when the structural or functional attributes of the circuit under detection fully pass the coverage test, there is no guarantee that Trojans are not present. Manual post-processing is still needed to further analyze suspicious signals or gates to determine whether they were HTs (or part of a Trojan circuit). On the other hand, ML-based approaches can extract and identify Trojan-infected features automatically and screen out the most effective features for classification. This can not only enhance the TPR but also reduce the size of the sample feature vectors, thereby increasing the efficiency of the circuit

TABLE 9. Main Research Contributions and Innovations of ML in DFS.

DFS	Article	Main Research Contributions and Innovations
Trojan Detection Assistance	[72]	GA can select the optimal set of ROs from Power RON and one-class SVM then classifies them, thus achieving a high HT detection accuracy.
	[95]	LightGBM can extract the key features from scan-chain netlist-based features to reduce feature dimensions.
	[58]	One-class ANN can classify continuously transient currents measured by on-chip current sensor.
	[96]	Regression can classify PV-related parameters obtained from transient current waveforms and circuit nets.
	[63]	Bayesian can calibrate PV distributions by PV monitors to facilitate HT detection by leakage current.
Implantation Prevention	[75]	MOEA can reduce rare signals and maximize the efficiency of logic encryption at IC design time.
Trusted Library	[103]	One-class SVM can establish golden power or delay library by PCM units.
	[71], [90], [91]	CAs can cluster side-channel features to build golden side-channel fingerprinting library.
	[74], [104]	AIOA and PSO can enhance the performance of learning models such as LR, NBC, and NN by multiple iterations of model related parameters.

feature analysis methods. However, there are still several potential challenges that remain to be addressed. First, golden designs are needed to establish the training dataset for the structural or functional attributes of the gate-level netlists. Second, these solutions are only valid for identifying explicit Trojans and are not effective at detecting implicit Trojans. Third, the TNR and average accuracy of HT detection need to be further improved. In addition, the execution time and reliability of ML-based approaches depend on the circuit scale, i.e., the number of nets within the circuits, and the quantization metrics selected. They are less effective when analyzing larger circuits.

Side-channel analysis is a non-invasive method. Because this solution can easily generate test vectors and does not need to fully activate the HTs, it is easy to implement and widely used in practice. Especially for large Trojans, it is possible to effectively identify and achieve high detection accuracy. However, this method is susceptible to PVs and noise and performs poorly in the detection of small Trojans. In particular, the main limitation of such a method is to assume that there are golden ICs available as a reference. The requirement for the precision of the equipment is high. For some methods that attempt to sidestep the dependence of golden ICs, the results are less satisfactory in terms of accuracy. ML can effectively preprocess relevant features, such as extracting relevant features and reducing the data dimensionality, to effectively reduce the impact of PVs and noise and improve the accuracy and TPR. However, in this procedure, since Trojans have negligible effects on circuits, they may be removed as unrelated features or noise, thus affecting the HT detection accuracy. Furthermore, the performance of ML-based methods is highly dependent on the selection of relevant features, learning models, and parameters, which may increase the time overhead. In addition, ML methods, such as unsupervised learning, can partially reduce their reliance on golden ICs but still require golden designs to build a training dataset. This, however, remains a limitation in practical applications.

2) DESIGN FOR SECURITY

Another important application of ML in HT defense domains is DFS. Several ML-based countermeasures have been proposed for (1) Trojan detection assistance, (2) implantation prevention, and (3) trusted library. Fig. 9 summarizes the existing progress in DFS. In particular, Table 9 briefly summarizes the primary research contributions and innovations of ML applied in DFS based on Fig. 9. From Table 9, it can be concluded that ML, especially optimization algorithms, can be incorporated with design-time mechanisms to increase the accuracy of HT detection, the efficiency of design protection strategies and the performance of learning models. Below, we introduce the primary elements of each ML-based approach, as well as the advantages and disadvantages between ML-based and non-ML-based methods in detail.

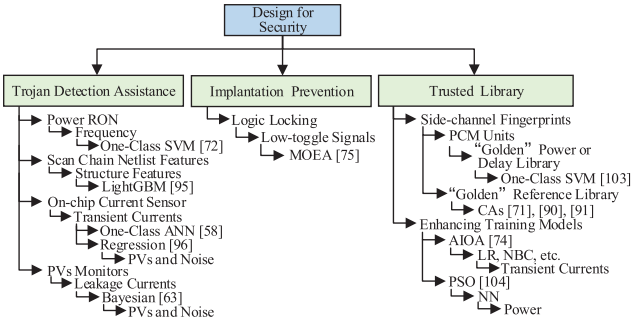

FIGURE 9. Summary of ML applied in DFS.

- **Trojan Detection Assistance**

HT detection can benefit dedicated on-chip modules. Such embedded solutions can improve the sensitivity of HT detection at design and test time or be used to identify the abnormal behaviors caused by HTs at runtime [92], [93]. In particular, ML can be incorporated with on-chip modules to enhance the accuracy of HT detection at the expense of extra silicon areas in the ICs.

For instance, Karimian et al. proposed exploiting the power ring oscillator network (RON) structure to assist in HT detection [72]. They first utilized the GA to select an optimal set of RO measurements from the power RON and then applied one-class SVM to classify the measurements to identify HT-infected designs. This method can achieve a higher accuracy but requires substantially more training time than the PCA+convex hull and pure SVM-based methods and has

a high area overhead [72]. Dong et al. introduced a scan-chain feature-based analysis method to identify gate-level Trojans [95]. They first converted the gate-level HTs to scan-chain netlists and then utilized a lightweight gradient lifting algorithm called lightGBM to extract the key features for training. LightGBM can reduce the feature dimensions, thereby improving the HT detection accuracy and optimizing the training speed. However, certain HTs with special structures and trigger modules cannot be effectively classified.

On the other hand, it is extremely difficult to trigger HTs of various types and sizes during the test phase. Hence, runtime monitoring provides the last line of defense against Trojan attacks, and ML can be incorporated into ICs as dedicated security primitives to enable dynamic verification at runtime [47], [94]. Liu et al. proposed an HT online monitoring and parallel detection approach [58]. This approach combined the normal behavioral operations of ICs with continuously transient currents and employed a one-class ANN classifier to concurrently distinguish Trojan-free and Trojan-infected ICs. However, such a one-class ANN classifier can only perform partial monitoring and cannot validate the ICs in a comprehensive manner. For certain complex large-scale designs, this method might be somewhat ineffective. The circuit parameter values measured by on-chip sensors are susceptible to PVs and noise, which may mask the HT effects.

One way to solve this problem is to select the appropriate features and then input them into ML methods, thereby eliminating the impact of PVs and noise in circuit parameters. For instance, Shanyour et al. constructed a regression model between multiple PV-related parameters obtained from the transient current waveform and the circuit lines to predict the PV changes [96]. For each targeted circuit line L, all measured values extracted from the sensors are provided to the regression models, and any small deviation from the expected value will indicate the presence of the HTs. In addition, Chen et al. introduced a Bayesian inference-based technique to calibrate the PV distribution for each individual chip using the maximum-a-posteriori (MAP) estimation to enhance the accuracy of HT detection through leakage current analysis [63]. The proposed framework can reduce the measurement cost and achieve a high HT detection rate even under large measurement errors. However, the golden designs are still required as a reference in the pre-silicon simulation, and the accuracy of calibrated HT-free scaling factors may depend on the number and location of PV monitors and the batches of chips measured.

- **Implantation Prevention**

Implantation prevention aims to utilize design obfuscation or layout-filler approaches to protect ICs/IPs from HT insertion and activation, reverse engineering, or theft [97], [98]. ML has been applied in the EDA design and testing domains, and there have also been several advances in HT prevention [99], [100].

Built-in locking mechanisms, such as logic encryption, are usually exploited to hide the real functionalities of the original designs [101], [102], as shown in Fig. 10. To this end, Andrea et al. introduced a MOEA-based obfuscation design strategy that can explicitly reduce the rare signals and maximize the efficiency of logic encryption to effectively mitigate the threats of HT attacks and overproduction [75]. However, such a technique is only suitable for small combinational circuits. With increasing VLSI circuit scales, MOEA may require greater execution times to obtain the local optimal solutions.

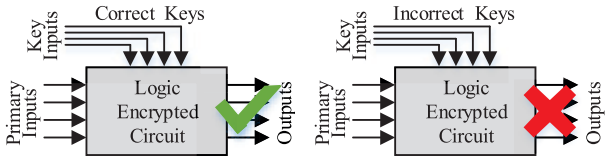

FIGURE 10. Block diagram of Logic Encryption [102].

- **Trusted Library**

A trusted library refers to providing a golden reference library, such as trusted datasets that contain the golden side-channel fingerprints and circuit features, or enhanced training models for HT detection. Current research usually assumes that golden designs/ICs are available for reference. This, however, severely restricts the practical feasibility of current defense techniques.

There has been significant progress in the construction of golden reference libraries when incorporated with ML. Several studies have attempted to utilize Monte Carlo (MC) to simulate golden designs to obtain golden data at design time. For example, Liu et al. incorporated process control monitor (PCM) units into ICs to sample and measure the MC simulation-based features of side-channel information [103]. After that, the classification boundary of these statistical side-channel fingerprints could be learned via one-class SVM during the pre-manufacturing stage. This procedure can offset the requirements for golden ICs and utilize simulation-based data to represent them; however, this technique still needs golden designs as a reference. Moreover, researchers have attempted to exploit CAs to build trusted reference libraries to shift the demands for trusted ICs [71], [90], [91]. However, all the aforementioned approaches are still in the experimental simulation stage, and there is still a long way to go before practical applications are realized.

Researchers have also attempted to explore design optimization algorithms to enhance the training models. For instance, Xue et al. presented an enhanced training model for HT detection, in which the higher weights were assigned to incorrectly classified instances after each iteration [74]. Wang et al. used PSO to find the global optimal extremum matrix of a NN after multiple iterations of the NN weights and bias values to minimize the squared error of the NN [104]. Compared to the BPNN, PSO+NN can increase the accuracy of HT detection but at the cost of greater time consumption.

Table 10 illustrates the advantages and disadvantages between ML-based and non-ML-based approaches in DFS.

TABLE 10. Comparisons of Positives and Negatives between ML-based and non-ML-based Approaches in DFS.

DFS Technology	Non-ML based Strategies		ML based Strategies	
	Positives	Negatives	Positives	Negatives
Trojan Detection Assistance	• Intrusive detection method • Add extra on-chip modules or probes • Reduce rare nodes/signals • Amplify the effect of HTs on circuit parameters • Enhance accuracy	• Increase area overhead • Need golden designs/ ICs as reference • Partially monitor the circuit parameters • Affect the confidentiality and integrity of ICs	• Enhance accuracy effectively • Reduce the measurement cost • Eliminate the impact of PVs and noise in IC parameters • ML can be embedded into circuits as new on-chip models	• Require much more time for feature extracting and training • Increase area overhead • The security of ML model itself cannot be ensured
Implantation Prevention	• Intrusive defense approach • Obfuscate and disguise the original circuits, or fill the die spaces in ICs • Reduce possibility for HT implanting or triggering	• Increase area, time, and power overhead • May influence the performance of circuits • Cannot resist reverse engineering, parametric HT	• Can reduce the rare signals explicitly • Maximize the efficiency of logic encryption • Mitigate HT attack and overproduction	• Only suitable for small combinational circuits • Multiple iterations • Take more time to achieve local optimum solutions
Trusted Library	• Build trusted libraries as reference • Reverse engineering ICs	• Difficult to complete • Costly, time consuming • Process deviation impact	• MC simulate the IC designs • Extract features by on-chip modules or probes • Can enhance the performance of training models	• Retain experiment stage • Need golden designs • More time consumption • Multiple iterations to achieve best performance

As shown in Table 10, DFS is a design-time consideration mechanism that uses a series of security design strategies to enhance HT detection or prevent HT implantation/activation. For Trojan detection assistance, existing research is similar to HT detection; however, there are differences which we describe as follows. (1) It is possible to reduce the number of rare nodes/signals inside the circuit by implanting additional logic units, such as dummy scan flip-flop (DSFF) to increase the controllability and observability of the internal nodes. However, this scheme introduces additional hardware resource overhead. Moreover, the toggle rate of the internal nodes inside a circuit is related to the test vectors applied and the topology of the circuit itself, which may result in a difference in the location and number of logic units implanted. (2) It is also possible to employ the on-chip sensors/logic modules/probe points implanted at design time to collect circuit parameters or to convert the influences of Trojans on the circuit parameters into an observable format to amplify the impacts of HTs on the side-channel parameters. Therefore, such solutions can enhance the accuracy and reliability of HT detection. However, in addition to increasing the hardware overhead, these approaches may also require golden designs/ICs as a reference. As intrusive means, the added logic modules/sensors/probe points could also be exploited by attackers, resulting in information leakage. ML-based methods can select the most effective HT features from the features extracted from the added logic modules/sensors/probe points or eliminate the influences of the PVs and noise, thereby effectively improving the accuracy and reducing the measurement cost. However, the feature extraction and training processes will take more time, which may impact the performance of the ICs. In addition, ML can also be incorporated as on-chip modules into a circuit to provide runtime monitoring for the target circuits. However, it can only perform partial inspection and will fail to provide protection in a comprehensive manner. Simultaneously, the security of ML itself cannot be guaranteed. An attacker may also embed the Trojan circuit into the ML module, thereby affecting the detection performance.

For implantation prevention, the current study mainly exploited design obfuscation, camouflage technology, and functional cell filling strategies to prevent Trojan implantation or activation. Such schemes can confuse opponents and mitigate the threats of Trojan attacks. However, they will cause large-area circuit redundancy, as well as high resource requirements and costs. Moreover, they may also have serious influences on the performance of target circuits, such as readily producing crosstalk. In particular, such methods cannot protect against opponents who are highly familiar with the entire design. Similar concerns also appear in reverse engineering and parametric Trojans. On the other hand, ML-based approaches can gain the effects of design obfuscation on target circuits and reduce the number of rare nodes/signals, thus preventing the activation of Trojans. However, such approaches are appropriate for small combinational circuits. In regard to large ICs, they may spend more time iteratively obtaining the optimal solutions.

Regarding trusted library, most work has applied reverse engineering to acquire the trusted circuit feature dataset. The drawbacks here are obvious. On the one hand, the ''trusted'' IC dies are difficult to obtain; on the other hand, they are relevant to the limitations of the reverse engineering technique itself (see Table 8). ML-based approaches are intended to perform MC simulation on the golden designs, and then obtain the trusted circuit feature datasets through classification or clustering algorithms. However, the obtained simulation data are different from the actual measurement results and still need further improvement in practice. In terms of training model enhancements, design optimization algorithms could improve the performance of learning models; however, multiple iterations are required to achieve

TABLE 11. Main Research Contributions and Innovations of ML in Bus Security.

Bus Security	Article	Main Research Contributions and Innovations
Router Behavior Features	[107]	K-NN, SVM, LR, and DT can classify the router features, such as source core, distance core, transfer path, and distance, etc., extracted from on-chip traffic to identify the known HT attacks to routers.
	[108]	Learning models can be automatically updated by MBW to consider the unknown HT attacks launched to routers.
	[112]	Unsupervised algorithms, such as K-means, FF, EM, and HC, can cluster the HT-infected router features.
Traffic Links per Router	[111]	SNN can classify the spatial and temporal features extracted from NoC data exchanged across traffic links between adjacent routers to identify DoS attack caused by HTs.

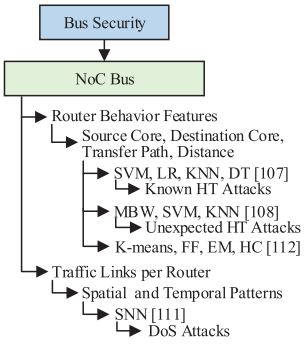

FIGURE 11. Summary of ML applied in Bus Security.

the optimal performance. Additionally, this process is time consuming.

3) BUS SECURITY

Bus security precautions intend to provide effective countermeasures against malicious on-chip traffic behaviors to ensure the security of communication and the reliability of data transmission between multiple cores in a multiprocessor SoC environment [105], [106]. The major malicious behaviors along this line include denial of service (DoS) [107], linker behavior [26], [27], and router behavior [108], which are launched by HTs implanted in links or routers. Attackers may exploit covert communication [73], [109], bus idle states [110], and peripheral interfaces [28] to disclose confidential information, tamper with communication data, interfere with normal operations, and perform DoS attacks.

In recent years, substantial progress has been achieved in applying ML to secure on-chip buses, as shown in Fig. 11. Table 11 briefly presents the main research contributions and innovations of ML applied in bus security based on Fig. 11. As described in Table 11, ML, especially supervised learning algorithms, has high accuracy in indicating the abnormal on-chip traffic behaviors caused by Trojan attacks. Then, we provide the main contents of each ML-based approach, as well as the advantages and disadvantages between ML-based and non-ML-based methods in detail.

Kulkarni et al. exploited router behaviors to determine whether routers in a multi-core IC were infected by HTs [107], [108]. These router behaviors are related to the packet source and destination addresses, transfer path, and transfer distance. They first extracted the features from on-chip traffic and then classified them by several ML models, including K-NN, SVM, linear regression (LR), and DT, to identify HT-infected routers [107]. For attacks that are not considered in the training phase, Kulkarni et al. also presented a method to dynamically update the trained model by using a modified balanced winnow (MBW) [108]. In addition, several unsupervised learning algorithms such as K-means, farthest first (FF), estimation maximization (EM), and hierarchical clustering (HC), were also explored to identify the malicious router behavior features [112]. Compared to SVM and K-NN, the MBW-based technique achieves a higher detection accuracy, a lower area overhead and a lower detection latency. Moreover, unsupervised learning algorithms are generally less accurate than supervised learning algorithms, possibly because the accuracy of these algorithms depends on the compactness and separation of clustering. Additionally, they are not effective against spoofing attacks [112].

Madden et al. presented a spiking neural network (SNN)-based HT detection approach that was designed to detect DoS attacks through abnormal traffic patterns in NoC data [111]. They first extracted the spatial and temporal features of digital data exchanged across traffic links between adjacent routers. Then, these features were input into the SNNs and learned to reveal the abnormal operations. This method can effectively identify unseen attacks; however, the accuracy strongly depends on the length of the attacks.

Table 12 illustrates the advantages and disadvantages between ML-based and non-ML-based methods in bus security. As shown in Table 12, bus security has made some progress in terms of NoC bus and AHB bus. In prior work that does not explore ML, researchers have attempted to use the on-chip modules/probes to monitor the activities of NoC buses to prevent information leakage attacks caused by Trojans. Other techniques to protect the security of data flowing within the NoCs include encryption, firewall, and data tags. In addition, state obfuscation and secure routing techniques could also be used to improve the probability of HT detection for ensuring the integrity and confidentiality of data in the NoC buses. However, the above methods still have some shortcomings. For example, these approaches mainly identify the abnormal behaviors created by Trojans at runtime. As intrusive solutions, they might increase the power and time consumptions of the NoCs and affect the performance of the system, such as increasing the system delay. In particular, these strategies mainly focus on resisting several common attacks such as DoS attacks. However, novel

TABLE 12. Comparisons of Positives and Negatives between ML-based and non-ML-based Approaches in Bus Security.

Bus Security Technology	Non-ML based Strategies		ML based Strategies	
	Positives	Negatives	Positives	Negatives
NoC Bus	• On-chip modules, probes • Obfuscation techniques • Encryption, hash, etc. • Securing routing • Data tag, counter, etc. • Enhance integrity of NoC	• Runtime defenses • Increase time, power, and area overhead • Extra latency effects • Mainly to defend against DoS attacks, etc.	• Insert on-chip modules • Extract router/linker behavior from traffic • Add on-chip ML units • Improve accuracy • Reduce area overhead, detection latency	• Runtime defenses • Accuracy relies on the length of HT attacks • Need training dataset • May be invalid for unknown attacks • Continuous monitoring
AHB Bus	• Dynamic on-chip function replacement • Seamless system operation	• AHB bus arbitration mode • Require embedded reconfigurable logic	—	—

types of Trojans, such as covert communication-type Trojans, have received less attention. In contrast, ML may alleviate the above problems to some extent. In particular, most ML-based methods utilize on-chip modules/probes to extract the router and linker behaviors from the traffic of the NoC bus and then train them through ML, which not only improves the accuracy of HT detection but also reduces the area overhead and detection delay. However, the accuracy of these methods highly depends on the type of HTs, the length of the attacks, and the traffic patterns selected. Additionally, golden designs are required to build the training dataset, and certain unknown Trojan attacks may not be well defended against.

For AHB bus security, the current study adopted a method utilizing the dynamic replacement of functions to recover from unexpected operations disabled by HTs during runtime, thereby eliminating the effects of Trojans. This technique could achieve a seamless connection of system operations but requires embedded reconfigurable logic to implement the above scheme, which increases the area overhead of the system. Additionally, AHB utilizes the bus arbitration mode to share the bus, and the above solution may affect the real-time response of certain critical missions. Note that ML has yet to be explored for the AHB bus.

4) SECURE ARCHITECTURE

With the increasing complexity of modern ICs and the reuse of large numbers of 3PIPs, some researchers have attempted to study HTs from an architectural perspective. An adversary may exploit vulnerabilities from the architectural level to undermine the security of hardware. Therefore, the demand for a secure architecture is strong, and ML technology has made progress in the area of secure architecture, as shown in Fig. 12. In particular, Table 13 briefly describes the main research contributions and innovations of ML algorithms applied in secure architecture depending on Fig. 12. As illustrated in Table 13, ML can be applied as an on-chip security module to enhance the security of SoC at the architectural level. Below, we introduce the main contents of each ML-based approach, as well as the advantages and disadvantages between ML-based and non-ML-based methods in detail.

Jin et al. advised that ML algorithms can be exploited for secure architecture design in [113]. They also proposed a

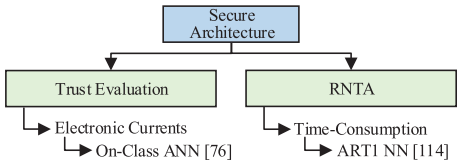

FIGURE 12. Summary of ML applied in Secure Architecture.

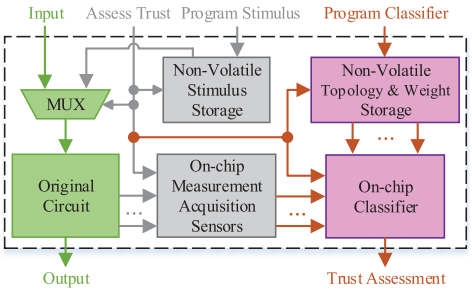

FIGURE 13. Block diagram of proposed Post-deployment Trust Evaluation Architecture [76].

trust evaluation architecture for the real-time protection of deployed chips [64], [76]. Such a secure architecture applied an on-chip analog ANN classifier to analyze current parameters that are acquired and measured from on-chip sensors and then classify as either Trojan-free or Trojan-infected chips. Fig. 13 presents the block diagram of the proposed trusted evaluation architecture. However, the accuracy and stability of this scheme depend on the primary inputs, and the tradeoff between security and performance overhead also needs to be considered.

Krishnendu et al. introduced a runtime trust neural architecture (RTNA) based on adaptive resonance theory (ART1) NNs [114]. RTNA focuses on preventing the confidentiality-undermining attacks launched by HTs and masks the effects of HTs as circuit aging through unsupervised learning strategies. Although it can eliminate the need for a golden model, the sensitivity of RTNA depends on the periodicity of the internal clock in an SoC.

Table 14 illustrates the advantages and disadvantages between ML-based and non-ML-based methods in secure architecture. As shown in Table 14, the secure architecture

TABLE 13. Main Research Contributions and Innovations of ML in Secure Architecture.

Secure Architecture	Article	Main Research Contributions and Innovations
Trust Evaluation	[76]	One-class ANN can be employed as on-chip classifier to classify the electronic currents measured by on-chip current sensors, so as to detect HTs.
RNTA	[114]	ART1 NN can be explored as on-chip modules to mask the effects of HTs to time-consumption.

TABLE 14. Comparisons of Positives and Negatives between ML-based and non-ML-based Approaches in Secure Architecture.

Secure Architecture Technology	Non-ML based Strategies		ML based Strategies	
	Positives	Negatives	Positives	Negatives
Wireless Crypto IC, COTS, or FPGA	• Trust computing, voting • Dynamic reconfiguration • Soft and hard co-defense • Function verification • Ensure safe operations	• Increase area overhead • Affect the performance • Lack of error correction or recovery policies • Cannot resist collusion	• Insert on-chip classifier, sensors, storage, etc. • Reduce FPR and FNR • Effective to detect HTs activated in-field	• Require extra components • Trust evaluation • Need golden designs • Accuracy depends on test vectors applied
SoC Chips	• SoC-level protections • Resist HTs in untrusted 3PIP cores • Demand dedicated on-chip security modules • Fine-grained policies	• Runtime defenses • Security relies on the primitives embedded • Need golden designs • Incur power and area overhead	• Utilize ART1 as on-chip security modules • Provide confidentiality protection for SoCs • Mask HT effects • No require golden model	• Passive protections • Valid at runtime • Need HT activation • Sensitivity depends on periodicity of internal clock

mechanism has made substantial program in terms of wireless crypto ICs, COTS, FPGA, and SoC. For conventional defense strategies, early work combined a trust computing and voting mechanism to achieve the reliable execution of programs in untrusted COTS that may infect HTs. Subsequently, researchers incorporated dynamic reconfiguration, software-hardware co-defense, and function monitoring or verification techniques to enhance the security of the architecture level. Although such methods effectively resist HT attacks and ensure safe operations, several disadvantages still remain. For example, these solutions can increase the area overhead and affect the performance of the system. Furthermore, such methods focus on detecting the HTs in untrusted units/cells/modules, therein lacking corresponding error correction or recovery policies. These solutions may also fail to detect collusion attacks. In addition, the countermeasures mentioned above may not be applicable to SoC chips due to the heterogeneous characteristics of current SoCs [18]. To this end, architects and researchers have attempted to construct secure architectures specifically for SoCs such as IIPS and E-IIPS. Such state-of-the-art methods incorporate dedicated on-chip security IPs into an SoC to provide fine-grained protection for SoCs to resist IP-level and SoC-level HT attacks. However, these schemes work mainly at runtime, and the effectiveness depends highly on the hardware security primitives applied. In addition, golden designs are still required as reference for some secure architectures.

On the other hand, ML-based strategies in secure architecture have similar advantages as non-ML-based methods. However, the difference is that the former integrates the classifiers as on-chip modules in the ICs. Especially for wireless crypto ICs, for example, trust evaluation architecture established through on-chip ANNs [76], the incorporation of ML models can reduce the false positive rate (FPR) and false negative rate (FNR) and can effectively identify Trojans activated during operation. However, this method only evaluates the reliability of the circuits under detection; to assist the on-chip classifier, additional support components, such as on-chip sensors, memory, etc., are also required, which increases the area overhead. In addition, the construction of the training dataset still requires golden designs, and the accuracy of HT detection depends on the test vectors applied. For SoC chips, for example, RNTA [114], ART1 NNs can be applied as an on-chip module to provide confidentiality protection for SoCs. This method does not require a golden model and can effectively eliminate the impacts of Trojans. However, this is a passive protection strategy that works if and only if the HTs are actually activated and affect the SoCs.

IV. PRIOR LESSONS AND POTENTIAL CHALLENGES

This section discusses the lessons learned and challenges arising from previous studies. Section IV.A analyzes the usage trends of ML models in HT defense domains. In Section IV.B, we explore the effects of ML on the performance of HT countermeasures from several perspectives. The possible problems and potential challenges remaining when applying ML are discussed in Section IV.C.

A. RESEARCH TREND ANALYSIS

A review of available studies has indicated that ML is highly attractive for enhancing the security and trustworthiness of hardware. Here, we analyze the usage and research trends of ML techniques that have been explored in the HT defense domains. Fig. 14 summarizes the frequency of each ML model applied for HT defense based on the references in this survey.

From Fig. 14, it can be concluded that the most extensive use case of ML is HT detection, accounting for approximately 58.3% of all cases. This is because HT detection is the simplest and most common method of identifying HTs,

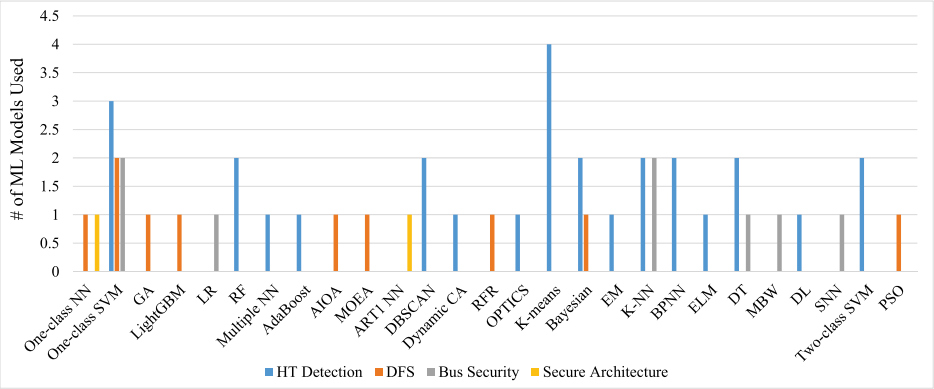

FIGURE 14. Frequency of ML models applied in HT defense domains.

and ML can be inherently and effectively incorporated to address HTs [16], [56]. Furthermore, DFS and bus security have also seen remarkable applications of ML, with percentages of 20.8% and 16.7%, respectively. This trend is reasonable because more untrusted entities are involved in the design and fabrication process of modern ICs. These entities may insert HTs at design time, especially SoC-level HTs, which cannot be recognized only by HT detection. The remaining application, secure architecture, is rarely explored, accounting for only 4.2% of all cases. This approach can be considered powerful and partially overcome the requirements for golden designs/ICs. However, it is subjective, and security depends on the experience, funding, vulnerability analysis and evaluation strategies used. In addition, this process also requires substantial time and effort when incorporating ML.

Further analysis of Fig. 14 reveals that ML algorithms with the largest amount of published literature in the field of HT defense are CAs, wherein K-means appear most frequently. This is primarily because they are not restricted by the requirements of golden ICs. On the other hand, ANNs are the most widely exploited ML algorithm, being involved in almost all HT defense areas. This can be explained by the fact that ANNs have a variety of variants, both supervised and unsupervised; hence, they can be adapted to different application scenarios. From Fig. 14, we can see that several ML algorithms have been explored in only one or a few aspects. For example, K-means, ELM, DL and BPNN are generally applied for HT detection; however, they have not yet been exploited for DFS, bus security, or secure architecture. Therefore, this could provide researchers with an opportunity to enhance the accuracy of HT protection by incorporating these models into these aspects. In addition, note that some ML optimization algorithms have appeared for HT defense such as GAs for feature selection [72].

In the following sections, we will specifically analyze and discuss the usage and research trends of ML exploited in each HT defense domain.

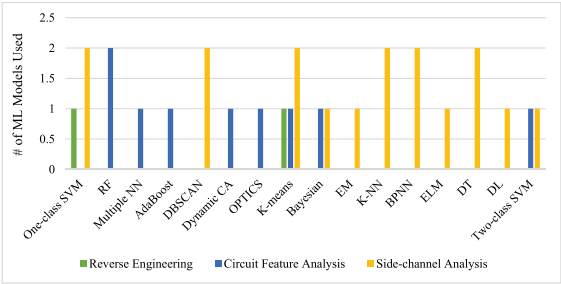

FIGURE 15. Frequency of ML used for HT detection.

1) USAGE OF ML IN HT DETECTION

Fig. 15 shows the usage of ML in HT detection. It is clear that ML algorithms are used mostly in side-channel analysis, probably because the side-channel information of ICs is easy to sample and vectorize. The second dominant feature is circuit feature analysis. However, it is rarely used in reverse engineering because the process of feature extraction is complex and time consuming, making it less practical.

From Fig. 15, we can conclude that SVM, including one-class and two-class SVM, is the most widely used supervised ML algorithm for identifying HT-infected ICs. However, SVM assumes that there are golden ICs available for training. Furthermore, K-means approaches are also relatively popular unsupervised learning algorithms for HT detection. However, they are not constrained by the above condition.

2) USAGE OF ML IN DFS

Fig. 16 shows the usage of ML in DFS. It can be seen from Fig. 16 that publications on ML-based approaches in Trojan assistance and trusted library are minimal in number, while there is only one publication on implantation prevention. This is primarily because Trojan detection assistance and trusted library can benefit from supervised or unsupervised methods to further enhance the detection and diagnosis of HTs. However, implantation prevention cannot be effectively incorporated with these strategies, except by applying several

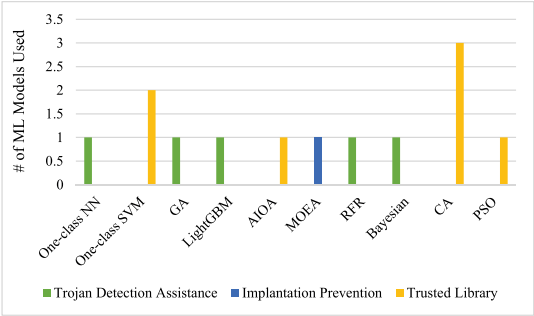

FIGURE 16. Frequency of ML used for DFS.

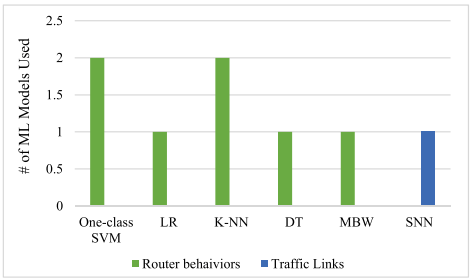

FIGURE 17. Frequency of ML used for bus security.

optimization algorithms to prevent the implantation or activation of HTs.

In addition, several design optimization technologies, such as GAs, LightGBM, MOEA, AIOA, and PSO, have been exploited in DFS alone. These algorithms will increase the performance of feature extraction and selection and optimize the performances of learning models (see Section III.B.2).

3) USAGE OF ML IN BUS SECURITY

Fig. 17 shows the usage of ML in bus security. Router behaviors and traffic links are typically utilized as hardware behavior-related features to identify the anomalies introduced only through the on-chip routers of a many-core chip. These routing nodes might be infected by system-level HTs, which can launch anomalous behavior patterns.

As shown in Fig. 17, the features of router behaviors are more frequently exploited to train several supervised ML models, e.g., one-class SVM, LR, K-NN, and DT, for hardware behavior analysis. However, traffic links have been studied less frequently, and only SNN-based methods have explored traffic links. One possible reason for this is that the traffic patterns are somewhat more difficult to extract and also require the assistance of adjacent routing nodes at runtime.

4) USAGE OF ML IN SECURE ARCHITECTURE

Fig. 18 shows the usage of each ML algorithm that has been utilized for secure architecture. Current solutions in this aspect generally employ 1) on-chip classifiers, 2) dedicated security IPs, or 3) design-for-debug (DfD) features to ensure the security and trustworthiness of multi-core ICs from

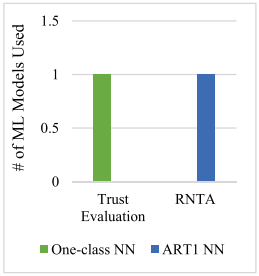

FIGURE 18. Frequency of ML used for secure architecture.

an architectural perspective. However, ML has only been explored in the first two schemes.

From Fig. 18, architects have only applied ANN algorithms, for example, one-class NNs and ART1 NNs, to build secure architectures for many-core chips. There have been few publications on this aspect, especially studies exploring ML. This, however, may also provide an opportunity to construct architectures with better performance for multi-core chips when learning models other than ANNs are incorporated.

B. PRIOR LESSONS

The previous sections have systematically explored the advances in applying ML to HT defense (see Section III.B); however, the effective phases of ML in HT protection are not indicated, and the necessary data information is also lacking. Since ML-based approaches essentially follow a data-driven pattern, most of the solutions have similar procedures, including feature extraction, main feature selection, PV and noise elimination, and classification or clustering (see Fig. 4). However, through an in-depth analysis, it is found that the specific effective phase of each ML algorithm in HT protection are not the same. Fig. 19 highlights the specific effective phase of each ML explored in HT protection.

From Fig. 19, it can be concluded that most ML methods are concentrated in the model selection and training phase. This is primarily because researchers usually utilized ML to determine whether the IC or IP under detection was infected by a Trojan instance. Particularly, the classification or clustering algorithms are selected according to whether the data labels are available. Next is the dimensionality reduction phase, and several commonly applied pretreatment techniques are implemented in this phase to overcome dimensional disaster, acquire the essential features, and simplify the learning process. For other phases, such as PV and noise elimination, main feature selection, and performance optimization, ML has also been implemented for selecting the best feature subset, eliminating PVs and measurement noise, and tuning the parameters. In addition, note that some optimization algorithms, which do not conform to the general procedure in Fig. 19, are only effective during the design stage of ICs such as MOEA.

To further illustrate the effects of ML-based solutions on the performance of HT protection strategies, we also

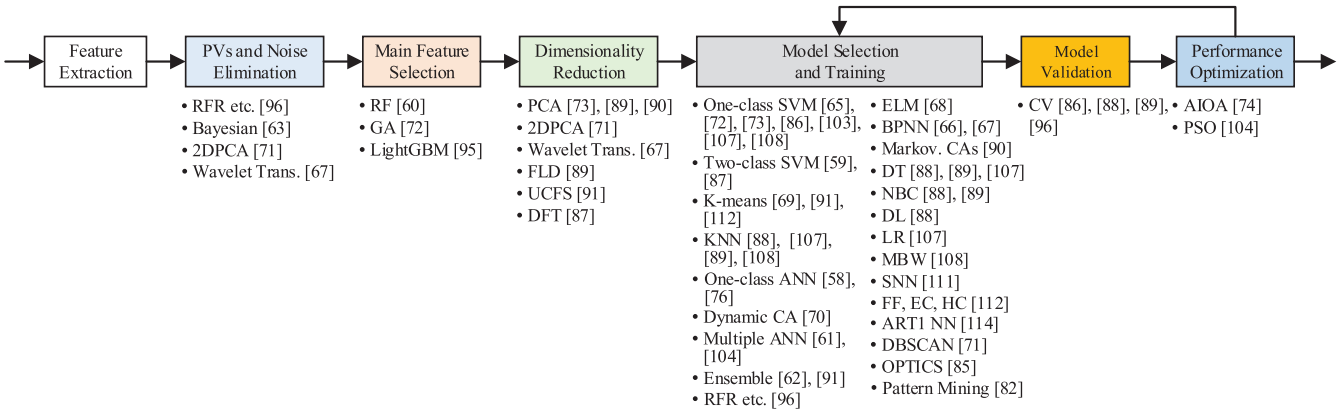

FIGURE 19. Effective phase of each ML algorithm applied in HT defense domains.

summarize and analyze the properties of the data used with respect to the ML workflow, which may be of great interest to the reader. In particular, Tables 15 – 18 systematically summarize each ML-based approach in the reviewed state-of-the-art. In the following sections, we will analyze and discuss these techniques and lessons learned from the following seven aspects: (1) extraction and selection of relevant features, (2) dimensionality reduction and pretreatment methods, (3) size of the training and testing datasets, (4) reasonable selection of the learning models, (5) performance optimization strategies, (6) effect of the PVs and noise, and (7) performance evaluation criteria.

1) EXTRACTION AND SELECTION OF RELEVANT FEATURES

There are a variety of relevant features that can be exploited to identify HT-infected ICs. From Tables 15 – 18, the widely used features involve circuit net-related features, circuit parameter features, and behavior-related features. Therefore, it is important to select effective relevant features from the raw dataset, which can significantly affect the performance of the learning models. For example, when the K-NN model is trained based on power-consumption instead of propagation delay, the HT detection accuracy increases from 93.12% to 99.02% [88], [89].

On the other hand, the size of the feature vectors, i.e., the attribute spaces, could also affect the performance of the models. For instance, when the learning model is trained with the best 11 gate-level Trojan features selected from the original 51 features using RF, the F-measure is increased from 66.9% to 77.8% and to even 100% in certain cases [60]. In addition, Karimian et al. applied a GA to select the optimal set of RO measurements for training. After that, the HT detection accuracy increased to 99.6%, and the equal error rate (EER) decreased to even 0.8% in the best case [72]. However, this process consumes more time overhead (190 and 120 times more than PCA+convex hull and pure SVM, respectively), because it requires many iterations to select the best feature set.

2) DIMENSIONALITY REDUCTION AND PRETREATMENT

An increase in the number of relevant features may improve the performance of the models; however, this is not always a better option. Especially in the cases in which the number of relevant features is very large and mutually correlated, redundant features will affect the trained models. Therefore, the dataset of the relevant features can be analyzed and rectified using dimensionality reduction techniques to obtain the optimized dataset for training to both prevent over-fitting and improve the performance of the models.

PCA is an effective dimensionality reduction method and is mostly utilized to address circuit parameter-related features (see Tables 15 – 16). For instance, Cui et al. employed the PCA method to reduce the original 10000-dimensional power data to 2050 dimensions, thus greatly reducing the amount of computations [90]. In [104], after using PCA to reduce the number of data dimensions from 8000 to 17, the structure of the NN was decreased from $8000\times89\times1$ to $17\times5\times1$, which increased the HT detection accuracy from 41% to 100% and reduced the computational complexity of the NN by more than 8000 times.

In addition to PCA, there are several other pretreatment techniques that have been applied to side-channel features, e.g., wavelet translation [67], 2DPCA [71], and DFT [87] (see Table 15). Moreover, Xue et al. introduced a modified unsupervised correlation-based feature selection method (UCFS) and adopted it to preprocess the raw power traces of ICs [91]. The detection accuracy of the clustering models subsequently increased by approximately 5%. However, this process incurs a slight increase in time overhead, increasing by up to 0.135% in the EDA evaluation and 3.23% in the FPGA evaluation.

3) EFFECT OF THE TRAINING AND TESTING DATASET SIZES

The study of ML cannot be separated from the dataset because the dataset is an essential component of any ML-based approach and can affect the prediction results of the learning models. Tables 15 – 18 list the size of the training and testing datasets for each HT defense technique. Note that the size of the training and testing datasets varies greatly in

TABLE 15. ML Models in HT Detection.

Article	Feature Properties				Pretreatment Techniques	Learning Model	Repeating (# of trials)	Evaluation Metrics
	Feature Selection	Training Dataset (# of instances)	Testing Dataset (# of instances)	Feature Vector (# of features)				
[65]	IC Images /Grips	29	40	5		One-class SVM	10	Accuracy = 100%
[69]	IC Images /Grips	N/A	2000	2		K-means	500	Accuracy = 97.2% – 100%
[70]	LTS	N/A	16	1		Dynamic CAs		Accuracy = 100%
[84]	Controllability /Observability	N/A	>500K Signals	2		K-means	500	FPR = FNR = 0%
[85]	Signal Correction	N/A	8	2		OPTICS		TPR = 28% – 100%, SPC = 94% – 99%
[59]	Net-related Features	16	1	5		Two-class SVM	0 – 20	TPR = 0% – 100%, TNR = 22% – 100%
[60]	Net-related Features	12	4	11		RF		TPR = 5.3% – 100%, TNR = 98.2% – 100%, Precision = 33.3% – 100%, F-measures = 9.1% – 100%
[61]	Net-related Features	16	1	11		Multi-layer ANN	3	TPR = 74.3% – 88.9%, TNR = 53.3% – 70.1%
[62]	Functional & Structural Features	800	11	14		Voting Ensemble		TPR = 73% – 100%, TNR = 97% – 98%
[68]	Electronic Current	1	3	500 Current Curves		ELM	4	Accuracy = 88.2% – 100%
[66]	Power Consumption	4	3	12000 Sample Points		BPNN		
[67]	Power Consumption	4	1	12000 Sample Points	Wavelet Translation	BPNN		Sensitivity = 99.2%
[90]	Power Consumption	360K	120K	1	PCA	Markov. CAs		Accuracy = 92% – 100%
[88]	Power Profile	1260	6300	6		K-NN, DT, NBC		Accuracy = 86.4% – 99%, Precision = 68.4% – 100%
[89]	Propagation Delay	271	1329	16	PCA/FLD	K-NN, DT, BC	500	Accuracy = 93.1% – 95.1%, Precision = 79.7% – 95.9%
[91]	Transient Supply Current	N/A	>560	1	UCFS	Cluster Ensemble	4	Accuracy = 91.5% – 93.75%
[86]	EM Traces	150	50	50 EM Traces	PCA	One-class SVM	50	FPR = 1.1% – 14.4%, FNR = 0% – 18.4%, Accuracy = 57.6% – 99.4%
[73]	Transmission Power	30	90	6	PCA	One-class SVM	30	FPR = FNR = 0%
[87]	Frequency domain of Power Consumption	1	12	1	DFT	Two-class SVM	2918	Accuracy = 100%
[71]	Thermal Radiation	N/A	>100K	2	2DPCA	DBSCAN	5	FPR = 0% – 13%, TPR = 10% – 100%

the literature. For example, the size of the training dataset exploited in [68] and [87] could be as small as 1 instance, while the size of the testing dataset utilized in [84] was as large as 500K samples (see Table 15).

Furthermore, increasing the size of the training dataset can typically prevent over-fitting and improve the generalizability of the models, thereby enhancing the accuracy in predicting the target outputs of previously unseen inputs. For example, the size of the training dataset in [72] varies from 8 instances to 24 instances, and the corresponding HT detection accuracy increases from 96.9% to 99.6% (see Table 16). At this point, the time cost for training each classifier increased by 0.24sec, 0.89sec, and 94.35sec (i.e., 19.35%, 67.94%, and 49.69%), respectively. However, the extra training time required due to the changes in dataset size varies for different ML models. In general, the ratio of the dataset size to the feature size should be sufficiently large, typically 5–10 times the feature size [115].

Note that in addition to Markv distance-based CA, unsupervised learning models only require testing datasets as input such as in [70], [85], [91], [111], and [114] (see Tables 15 – 18). Some researchers use sample datasets with labels as unlabeled data, for example, the work of Cui et al. in [90], to train the learning models and then validate them using the testing datasets.

4) REASONABLE SELECTION OF THE LEARNING MODELS

The selection of learning models should be based on the application scenario, feature type, and availability of training datasets. An appropriate learning model can achieve the best prediction outputs in HT problems. This is reasonable because even if the features learned are identical, the

TABLE 16. ML Models in DFS.

Article	Feature Properties				Design Units and/or Optimiz -ation Methods	Learning Model	Repeating (# of trials)	Evaluation Metrics
	Feature Selection	Training Dataset (# of instances)	Testing Dataset (# of instances)	Feature Vector (# of features)				
[72]	Frequency	8 / 16 / 24	25 / 17 / 9	8	Power RON + GA	One-class SVM	10 (Iteration)	Accuracy = 96.9% – 99.6%
[95]	Net-related Features	4	15	13	Scan Chain Structure	LightGBM	3000	TPR = 93.7% – 100%, TNR = 54.5% – 100%, Precision = 98.6% – 100%, F-measure = 96.7% – 99.9%, Accuracy = 94.2% – 99.9%
[58]	Transient Current	50	50	2	On-chip Curr- ent Sensor	One-class ANN	100	FPR = 6%
[96]	Transient Current	200	200	3	Built-in Curre- nt Sensor	RFR etc.	4	Accuracy = 94% – 100%
[63]	Leakage Current	10000	360	16	PVs Monitors	Bayesian	36 – 51	TPR = 98.5% – 100%, TNR = 97.5% – 100%
[75]	Rare Signals	100 (# of po- pulation size)	100 (# of new individuals per generation)	2 (selective pressure)	Logic Encrypt- ion Key	MOEA	10000 or 12 hours (Iteration)	Hamming Distance = 50%
[103]	Output power Fingerprints	100	120	6	PCM Unit + PCA	One-class SVM	5	FPR = 0%, TNR = 7.5%
[74]	Transient Current	100	100	501 Current Points	AIOA	LR, NBC, etc.	10 (Iteration)	Accuracy = 70% – 98%, Recall = 0.75 – 0.98
[104]	Power Consumption	1600	400	1	PCA + PSO	ANN	60 (Iteration)	Accuracy = 99.1% – 100%

TABLE 17. ML Models in Bus Security.

Article	Feature Properties				Design Units	Learning Model	Repeating (# of trials)	Evaluation Metrics
	Feature Selection	Training Dataset (# of instances)	Testing Dataset (# of instances)	Feature Vector (# of features)				
[107]	Router Behavior	716	290	8	ADM	One-class SVM		Accuracy = 96.9% – 99.6%
[108]	Router Behavior	7260	10055	10	ADM	MBW		Accuracy = 67% – 96%
[111]	Traffic Pattern	N/A	49790	2		SNN	5	Accuracy = 76% – 93%

TABLE 18. ML Models in Secure Architecture.

Article	Feature Properties				Design Units	Learning Model	Repeating (# of trials)	Evaluation Metrics
	Feature Selection	Training Dataset (# of instances)	Testing Dataset (# of instances)	Feature Vector (# of features)				
[76]	Transmission Power or Current	1K	4K	2	Built-in Current Sensor	One-class ANN	4	TPR > 97.2%, TNR > 99.8%
[114]	Time Consumption	N/A	4	3		ART1	5	Increase of Area Overhead < 6.7%, Increase of Power Overhead < 0.3%

accuracy will vary under different applied learning models. For instance, the work in [108] applied MBW, SVM, and K-NN to identify the unexpected attacks created by HT-infected routers, and the overall accuracy for MBW was 5% to 8% higher than SVM and K-NN, respectively (see Table 17).

On the one hand, ML algorithms can be selected according to the given application scenario. For instance, to minimize the number of rare signals and maximize the performance of logic encryption, an MOEA method has been applied to achieve the best obfuscation with an average 50% Hamming distance and prevent the insertion of HTs [75]. Other application scenarios include feature selection, dimensionality reduction, classification/clustering, and performance enhancement of the learning models.

On the other hand, ML algorithms, especially supervised or unsupervised learning-based approaches, can be selected according to whether the training datasets are available. For example, the works of [59], [61], [73], and [86], among others, selected a classification-based approach for HT detection because they assumed that golden designs or ICs are available as training datasets (see Table 15).

In addition, each ML-based approach has its own characteristics such as distinct storage space and computing requirements. These characteristics can also affect the precision of the predictions. Therefore, a single model may not achieve good results. It has been suggested that several learning models can compete with each other to select the most suitable model for current applications [88], [89].

5) PERFORMANCE OPTIMIZATION STRATEGIES

The performance when applying ML varies greatly according to various factors. Except for the aforementioned points, there are several other factors that can also greatly affect the performance of the learning models such as different model parameter settings and iteration numbers.

First, model parameters should be carefully tuned to ensure that the final models will achieve optimal performance. The parameters could be checked and determined by performing cross-validation (CV) [86]. For instance, the ν parameter, $\nu \in (0,1)$, is fine-tuned for the one-class ν-SVM to improve the performance of classification [65]. In particular, taking an s298 chip as an example, when the ν parameter is set to 5×10^{-4}, the classification accuracy on HT-free chips is 100%, while the accuracy on HT-infected chips varies from 0% to 9% according to the HT type. When ν is set to 7×10^{-3}, the detection accuracy on both HT-free and HT-infected chips is 100%. Additionally, when ν is set to 5×10^{-2}, the detection accuracy on HT-free chips is 0%, while all HT-infected chips are detected. However, the time consumption of this process is at least 4 times greater for determining the optimal parameter settings. Furthermore, several researchers have attempted to obtain the best parameter settings for their models by using optimization algorithms. For example, when PSO is applied to ANNs for network weight optimization, the HT detection accuracy with a TCR of 0.32% increases from 91.9% to 99.1% compared with the BPNN [104]. However, the optimization program also results in at least 75% extra time overhead.

Second, multiple iterations of the training or testing procedure can enhance the performance of the selected learning models. The number of iterations varies greatly. For instance, the iterations of MOEA necessary to find the most effective logic encryption key can be as large as 10000, or 12 hours (see Table 16). On the other hand, the experimental process could also be iterated multiple times to determine the most effective structure of the learning models. For example, the work in [61] obtained the most efficient structure of the NN classifier by iteratively changing the number of the middle layers and the number of units in each middle layer, thereby maximizing the TPR and TNR to 88.9% and 70.1%, respectively. Similar to the CV approaches, this process also spent much more time to obtain the most efficient structure of the learning models.

In addition to choosing appropriate model parameters and iterating, the performance of the learning models can also be improved through ensembling strategies. For instance, some progress has been made on voting ensembling [62] and clustering ensembling [91] approaches to decrease FPR and increase the HT detection accuracy.

According to the findings reported in previous research, we suggest that ensemble learning assisted by an efficient structure of learning models, parameter tuning and iteration optimization strategies can enhance the performance of ML-based approaches.

6) EFFECT OF PROCESS VARIATIONS AND NOISE

Random PVs and noise are two challenging factors that can significantly affect the HT detection sensitivity. They can mask effects of Trojan instances into the measurable parameters of an IC, especially in side-channel analysis. This concern becomes increasingly obvious with increasing IC scale, making random PVs and noise difficult to separate from the effects of the Trojan [4]. For instance, when random PVs increased from 20% to 40%, the HT detection accuracy in AES chips decreased from 89% to 44% and the FPR increased from 5% to 8% (under LTPD = 0.111, unsupervised clustering) [71]. Here, we outline several ML-related solutions that have appeared in studies to decrease the impact of random PVs and noise.

For the random PV problems, regression models, e.g., MLR, SVR, DTR, and RFR [96], can be applied to predict the PV changes in circuit parameters that are highly correlated; as a result, the RFR model obtains the lowest average prediction error (PE) and PE standard deviation and can facilitate the detection of HT-infected chips. This method did not cause any area overhead, and the increase in performance overhead was less than 2.25%. The study in [63] exploited a Bayesian inference-based MAP algorithm to calibrate and recover the spatial PV distribution for each fabricated chip in the batch (see Section III.B.2). Furthermore, increasing the number of PV monitors and effectively placing them can further offset the effects of random PVs and enhance the Trojan detection performance. However, this will result in extra hardware overhead. In addition to the above techniques, several pretreatment methods, such as 2DPCA, can also be exploited to overcome this problem [71].

For noise problems, real data collected from hardware should be adequately de-noised when working with data with high levels of noise. For example, the collected data can be sampled multiple times, and then, the mean value can be calculated for noise elimination. Several studies have applied pretreatment methods to eliminate noise effects. Ni et al. employed the wavelet transform technique to reduce the high-frequency noise and improve the HT detection accuracy from 92.2% to 99.2% [67]. It is suggested in [56] that noise-sensitive models such as K-means should be avoided for data training. In addition, model parameter tuning strategies can also be explored to reduce the effects of noise [65].

7) PERFORMANCE EVALUATION CRITERIA

A variety of performance indicators have been used to evaluate the effectiveness of ML-based methods. However, unified

criteria for performance evaluation have yet to be developed. As shown in Tables 15 – 18, the commonly utilized values for evaluating the results include Accuracy, TPR, TNR, FPR, and FNR.

Accuracy is the measure with the most extensive usage in ML-based methods, especially in supervised learning, and is related to the sum of true positive data (TP) and true negative data (TN) [56]. However, it is probably not the best measure for performance evaluations. For example, the HT detection accuracy and TPR of s38417-T300 in [95] are as high as 99.8% and 99.9%, respectively, whereas the TNR is only 54.5%. In contrast, the accuracy, TPR, and TNR of HT detection for s38417-T100 and s38417-T200 are almost 100%. From this perspective, it seems unreasonable to use only one indicator to evaluate the prediction results obtained by the learning models. Several measures, such as TNR, TPR, and precision, can be collectively exploited to evaluate the predictions of the learning models. For instance, the research in [60] utilized *four* values to assess the results. Similar concerns also exist in [86] and [95] (see Tables 15 – 16).

Note that the criteria used for the performance evaluation may rely on the given specific application and ML algorithm. For general ML-based HT detection methods, the indicators mentioned above can be exploited to evaluate the effectiveness of the prediction results. For other studies, such as the work in [75], the Hamming distance is applied to evaluate the logic encryption keys because its goal is to reduce the occurrence of rare signals and maximize the efficiency of logic encryption; for secure architecture with unsupervised learning, there are *two* indicators, e.g., the hardware area and power overhead, utilized to assess the performance [114]. In addition, only a few efforts have assessed the time overhead of the ML-based approaches.

C. POTENTIAL PROBLEMS AND CHALLENGES

The previous section discusses the important research advances, trends and lessons learned regarding the application of ML techniques in HT defense. Although it is advantageous to improve the security and reliability of hardware via ML-based approaches, a variety of problems and challenges remain and are summarized below:

1) EFFECTIVE SELECTION OF RELEVANT FEATURES

As described in Section IV.B.1, the size of the feature vectors can affect the performance of the learning models. If the size is too large, much time is required to learn them, thus resulting in a large overhead. Conversely, if the size is too small, they cannot be classified correctly. Hence, selecting the appropriate number of relevant features for the ML models is important for the accuracy and computational efficiency of HT detection. However, in the current ML-based approaches, only a small amount of work has focused on this aspect. Therefore, how to apply ML algorithms to determine the correct number of relevant features that should be selected is an important factor and challenge to be considered.

2) PROPER CHOICE OF PRETREATMENT TECHNIQUES

The dimensionality reduction and pretreatment of HT-related features can also enhance the performance and reduce the utilization overhead of the learning models. However, pretreatment techniques are primarily effective for side-channel analysis-based methods. When the impacts of HT instances are very small, they may be filtered out as noise. In particular, for some other features, such as circuit net-related and circuit behavior-related features, pretreatment techniques have not been utilized. As a result, how to consider the proper choice of pretreatment technology to amplify the influence of HTs on circuit features is a challenge, which can significantly alter the quality of the output results.

3) REQUIRED OBSERVATION NUMBER FOR THE TRAINING DATASET

The required observation number for the training data refers to the minimal effective feature dataset that can be utilized to train a model. This number depends on the selected features and training models. For instance, different features of the same ICs or different training models for the same features need different numbers of observations for learning the underlying pattern. In addition, these numbers also depend on the environment in which the procedures are carried out. As a result, a balance is needed between the training dataset size and the learning models to achieve the optimal performance, therein reducing the risks of over-fitting and decreased generalization.

4) EFFECTIVE DEPLOYMENT OF ML KERNEL

The effective deployment of ML kernels should also be taken into account. A ML kernel mainly contains feature extraction logic, the training or testing datasets, and the execution strategies. As specialized security primitives, ML kernels can be deployed as offline policies, for example, executed on a PC or server [67], [86], [95], or implemented as online policies through dedicated on-chip hardware modules, e.g., as in the studies in [58], [76], [107], [108], [114]. Although the current approaches can effectively provide dynamic verification and trust evaluation, however some challenges and risks still remain.

On the one hand, if the ML kernels were deployed offline, it may not be possible to achieve automated defenses against HT attacks during runtime. On the other hand, the hardware implementation of ML kernels also faces several challenges, such as extraction and pretreatment of relevant features, complexity of computational models, and managing memory transfers [107], [108]. Moreover, the security strategies that are built on these dedicated on-chip classifiers might produce a few false alarms compared to the software versions, while the latter will lead to high resource utilization and execution time overheads [76], [114].

In addition, several ML models cannot be trained offline, such as K-NN and MBW. Although training ML algorithms offline can reduce the overhead, this is also a challenge when deploying this kind of learning model.

5) BEST BALANCE FOR ML PRACTICES

As can be concluded from Section IV.B, ML has been explored in each stage of HT defense, e.g., (i) effective extraction of relevant features, (ii) attribute space generation, (iii) appropriate data pretreatment, (iv) PVs and noise elimination, (v) proper training model selection, (vi) model parameter tuning, and (vii) iteration improvement. However, when applying ML to HT defenses, each stage of this process should be properly balanced to achieve the best performance of the models.

6) SECURITY OF ML-BASED DEFENSES

Since ML-based techniques can protect against HT attacks, defense mechanisms themselves may also be compromised. Adversaries may maliciously change the model parameters and the training datasets or inject an attack to tamper with the execution procedure of the classifiers, resulting in reduced trustworthiness. Therefore, the security and trustworthiness of the explored ML models also present a challenge in practical application.

7) NEWLY EMERGING THREATS

With the continuous emergence of new techniques, novel methods can be exploited by an adversary to implement more intelligent attacks. While effort should be devoted to solving the remaining outstanding issues, newly emerging threats are also a concern.

New three-dimensional (3D) IC technology will change the current IC supply chain model and may produce new vulnerabilities. For instance, Hasan et al. proposed a novel HT that utilized the unique structure of 3D ICs [116]. Such a Trojan circuit can only be triggered by the thermal effect of middle tiers in 3D ICs.

In addition, attackers may also utilize ML techniques to perform more covert and advanced attacks. Chen et al. showed that the simulated annealing algorithm (SAA) can be adopted to produce a new type of HT [117].

V. HT THREATS BEYOND CHIPS

This section describes the HT threats as well as corresponding security policies in the hardware ecosystem. Section V.A summarizes the new HT threats emerging in higher layers. Section V.B proposes a reference model of HTs from the perspective of the full hardware ecosystem to define the corresponding security protections that the hardware ecosystem should possess. A discussion on future research directions is provided in Section V.C.

A. SUMMARY OF HT PROBLEMS

HT attacks and HT defense are similar to a game between a spear and a shield. Previous sections have analyzed and discussed the advances, lessons learned, and challenges in applying ML-based techniques for HT defense from the perspective of the chip layer. However, this is still not sufficient in HT defense domains, and HT attacks have gone beyond the chip layer. Beyond chips, novel HT attacks have appeared at in higher layers and may corrupt the security of the overall hardware ecosystem [118].

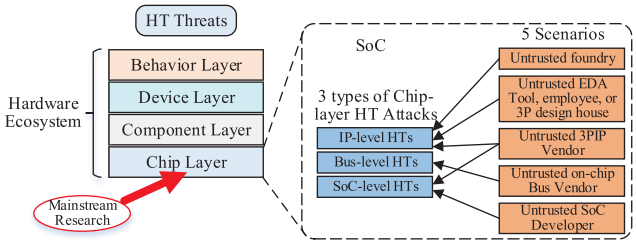

FIGURE 20. 4 layers of HT threats in the hardware ecosystem.

Ensuring the security and trustworthiness of the modern hardware ecosystem requires not only ensuring the security of the modern IC supply chain but also that of the entire hardware ecosystem. In particular, given the development of Internet of Things (IoT) technology, the security of electronic devices is facing unprecedented challenges.

In the authors' opinion, HT threats should be re-examined from the viewpoint of the overall hardware ecosystem. Thus, depending on the new threats, we categorize the HT threats to the hardware ecosystem into *four* layers: the chip layer, component layer, device layer and behavior layer. Fig. 20 generalizes the *four* layers of Trojan threats in the hardware ecosystem. Note that the chip-layer HT threats (the first layer of the *four* layers) include three types: IP-level HTs, bus-level HTs and SoC-level HTs. These threats are introduced by *five* typical scenarios during the IC design and manufacturing process (see Section III.A), which represents the mainstream of current HT research. In particular, the HT threats in each layer are described as follows:

- **Chip Layer.** IP-level, bus-level and SoC-level HT attacks, i.e., chip HTs, are maliciously injected into chips at any stage of the modern IC supply chain and can incur IP-level, bus-level and SoC-level impacts on the chips (see Table 6).
- **Component Layer.** Printed circuit board (PCB) designs can be deliberately modified by tampering with the interconnect lines at the internal layers or by altering the components, thereby introducing PCB HTs (or component HTs) [118], [119]. For example, an adversary may inject such macro HTs into commercial-off-the-shelf (COTS) modules or separate functional modules or use PCB features, such as the joint test action group (JTAG) interface and test hooks, to cause malfunctions or secret information leakage [120], [122]. PCB HTs can specifically affect the components themselves.
- **Device Layer.** Device HTs may exist in the functional components and COTS, e.g., Bluetooth and WiFi modules, of electronic devices and can affect the communication and data transmission between each function module of the electronic device. For example, an external memory module containing device HTs might tamper with data during communication or maliciously

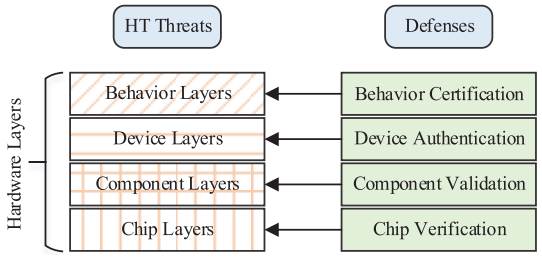

FIGURE 21. Overview of the proposed HTD reference model.

intercept and deliver such data through the connected multiplexing interface.

- **Behavior Layer.** Abnormal behaviors and operations by electronic devices during runtime can be produced by extremely hidden HTs in hardware system or functional modules. In addition, behavior HTs in boot programs can mislead the device and make it load untrusted programs or change the ownership of an IoT device, thus affecting the identity authentication for device-to-device and device-to-user communication. For example, a malicious universal serial bus (USB) flash disk (i.e., BadUSB) may disguise itself as another component, thereby affecting system functionality.

Prior security solutions to the threats facing chips (e.g., SoCs) have typically focused on IP-level, bus-level, and SoC-level HT attacks, i.e., ensuring that the chips used by consumers are HT-free. During the design and development of electronic devices, the above security features should still be followed but are largely overlooked in the literature.

B. REFERENCE MODEL

Given the aforementioned HT threats, we suggest that the defense policies for electronic devices against HT threats should also be classified into multiple layers, and the threats facing each layer should be addressed accordingly. Therefore, we introduce an HT defense (HTD) reference model based on summarizing the 4-layer HT threats that should be mitigated in HT defense domains. In particular, the reference model also illustrates and summarizes the security strategies that each layer should apply to protect hardware devices from HT attacks.

The proposed HTD reference model defines the corresponding security protection based on four layers: the chip verification layer, component validation layer, device authentication layer, and behavior certification layer. Fig. 21 illustrates the security policies that each layer should apply to address the corresponding HT issues. The following depicts the security responsibilities of each layer in detail.

1) CHIP VERIFICATION LAYER

Chip verification aims to ensure the security of chips that are used in hardware devices. This technique can ensure that the chips have not been threatened by HTs. However, the current IC supply chain model has many vulnerabilities, which can

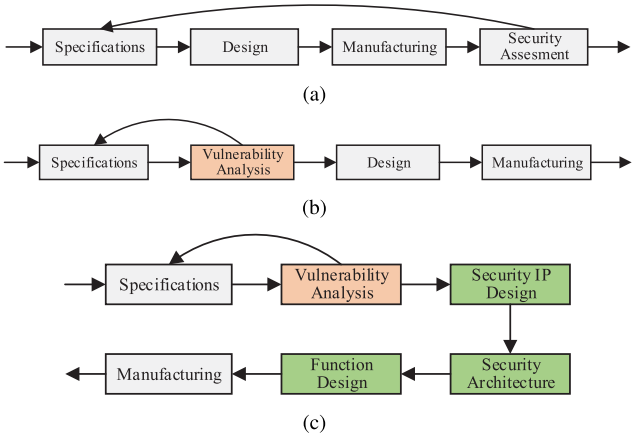

FIGURE 22. Block diagram of IC design flow. (a) Current IC design flow without vulnerabilities analysis [123]. (b) AMS-IP based design flow with vulnerabilities analysis [123]. (c) Security IP based design flow with vulnerabilities analysis [124].

be exploited by an adversary to perform HT attacks. This severely affects the security and trustworthiness of chips (see Section I). Therefore, researchers must strive to construct a strong-secure IC supply chain model to resist HT threats of the chip layer to provide fundamental security protection for hardware devices at the chip layer.

Some effort has been devoted in this area. For instance, Boher et al. proposed modifying the current IC design process, as shown in Fig. 22 (a), and exposing all existing vulnerabilities during the AMS-IP design process to identify potential security risks [123], as shown in Fig. 22 (b). Huang et al. introduced a novel security IP-based SoC design flow that considered the threats from the very beginning of the SoC design process [124], as shown in Fig. 22 (c). In addition, Sengupta et al. applied the PSO algorithm to explore optimized Trojan secure scheduling at design time [125].

2) COMPONENT VALIDATION LAYER

Component validation refers to providing protections for the hardware function modules, such as PCB or COTS, against PCB HTs. An adversary may implant a malicious design into the COTS and hardware components or design them using HT-infected chips, which may have catastrophic consequences. To effectively mitigate these types of threats, preventive strategies through hardening have been presented to address PCB HT attacks.

For example, Ghosh et al. adopted proactive approaches, such as secure interfaces and secure PCB, to protect against HT attacks at the PCB level [118]. Guo et al. proposed protecting PCBs from tampering via board-level RO and temperature compensation [120]. In addition, Nin et al. suggested that changes in the impedance of wiring could be used for component-based HT detection [121].

3) DEVICE AUTHENTICATION LAYER

Device authentication primarily protects electronic devices composed of one or more hardware function components

or subsystems. Device authentication is a direct technique for securing electronic devices and overcoming device HT threats from the following respects.

First, hardware devices can be integrated with a hardware security module (HSM) to specifically address the attacks created by the device-based HTs such as tamper-proofing and tamper-evidencing functions [126]. For example, ARM have introduced a platform security architecture (PSA) to secure IoT devices [127], [128]. Second, security authentication mechanisms for devices should be enforced to ensure that the underlying hardware is HT-free before application. For instance, Fern et al. utilized commercial formal equivalence checking tools to detect HTs inserted into the JTAG instruction set [129]. Third, devices can be developed using a secure architecture that can implement trusted executions based on untrusted COTS or hardware components that might contain HTs. For example, the trustworthy computing-based techniques in DFS, which are designed to overcome the untrusted 3PIP issues, can be extended to also address the issues of device authentication [16], [52].

4) BEHAVIOR CERTIFICATION LAYER

Behavior certification is a newly introduced topic that aims to reveal the abnormal behaviors of a system. The behavior features can be formally described as hardware-based security metadata. By modeling and analyzing the behavior features of hardware devices, components and COTS in normal operation, this technique can identify abnormal behaviors through dynamic certification, thereby ensuring the security and reliability of the system at runtime. This method can also be called a trusted audit, and it is expected to be able to effectively defeat behavior HT attacks.

Some progress has been made in this area by academia and industry. For example, Choi et al. proposed a secure firmware validation and update scheme for consumer devices [130]. The Mentor Company introduced root of trust and secure boot strategies to build a chain of trust policy for electronic devices [131]. Erdin et al. presented a novel threat detection and prevention framework to dynamically analyze USB cases through individual hardware-assisted modules [132].

C. FUTURE TRENDS AND DIRECTIONS

ML algorithms have shown great promise in overcoming chip-layer HT threats. Beyond the chip layer, ML-based techniques for HT defense deserve further exploration. We can observe from the proposed HTD reference model that HT problems in higher layers have similar concerns to those at the chip layer. Therefore, the insights obtained from research at the chip layer indicate that ML will be helpful in solving HT problems at higher layers. In addition, the security practices currently applied in industry usually adopt hardware-supported security products, e.g., HSM, isolation mechanisms, and encryption units, combined with software to secure the system. Compared with current industry security practices, our HTD reference model can be considered as supplementary to such practices and focuses on the HT problems from an overall viewpoint.

Future trends and directions may include the following:

(1) Chip layer:

- The application of ML methods to logic testing is worthy of further study because there is no relevant work in this field at present.
- Research on HT problems in ANN-based accelerators and computing units is lacking [133] and should be considered. This is primarily because as an acceleration engine, ANNs have made significant advances in many application areas, and the design of ANN-based hardware accelerator IP is a major trend for the future. Therefore, its security cannot be overlooked.
- Additional studies on secure architecture, especially those exploiting ML technologies to develop more robust, security-aware architectures for modern SoCs to resist bus-level and SoC-level HT threats rather than IP-level HT threats, remain a challenging and essential aspect of HT defense domains.
- Currently, ML-based approaches focus mainly on the detection, diagnosis and prevention of HT attacks, whereas there are few studies on error correction and recovery. Therefore, research on how to explore ML methods for providing error correction and recovery is also an interesting future direction.

(2) Beyond Chip layer:

- Recent ML-based approaches are aimed at HT problems at the chip layer (see Section III.B). However, several recent advances have indicated that HTs may threaten the security and trustworthiness of hardware at higher layers [118], [119]. Hence, exploring high-layer HT countermeasures via ML might also be an interesting future direction.
- The studies in [75] and [125] explored using optimization algorithms to enhance the reliability of IC design flows. As a result, such discussions could be used as references to establish a strong IC supply chain or even hardware ecosystem.

(3) Application Scenarios:

- There is minimal work incorporated into important application scenarios applying ML to solve HT problems. Future trends and research directions may consider both HT problems with practical scenarios to build scenario-related HT defense strategies. For example, the work in [134] considered HT issues in implantable medical devices, while the research in [135] focused on HT problems facing smart grids and home area networks.

(4) Other Directions:

- In the implementation phase, most ML algorithms are applied mainly to data collection and experimental validation (see Fig. 19). However, there are only a few papers about the ML implementation. Note that the specific deployment and implementation of ML-based techniques still have several challenges (see Section IV.C.4),

which could be an important future direction and should be considered in practice.
- The security of the ML models themselves against various threats should be considered in constructing model-trust HT defense strategies (see Section IV.C.6).
- Rather than focusing on HT defenses, how ML could be exploited by attackers is beginning to attract the attention of researchers (see Section IV.C.7).

VI. CONCLUSION

HT threats have drawn increased attention in academic and industrial research. In this article, we elaborated on the state-of-the-art applications of ML-based approaches in HT defense studies. By analyzing the relevant achievements, potential challenges and problems facing current research are identified. Since the HT threats have evolved beyond the chip layer, a reference model is presented to illustrate corresponding HT defense strategy requirements from the perspective of the entire hardware ecosystem.

The primary purpose of this article is to demonstrate the latest advances in the application of ML-based techniques in HT defense fields to provide a general understanding and a guidebook to those who want to engage in HT defense research.

ACKNOWLEDGMENT

The authors would like to thank Dr. J. Zhu, D. Lin, P. Yang, X. Li, Y. Dang, and so on for their suggestions during the revision of the article.

REFERENCES

[1] G. Sumathi, L. Srivani, D. T. Murthy, K. Madhusoodanan, and S. S. Murty, "A review on HT attacks in PLD and ASIC designs with potential defence solutions," *IETE Tech. Rev.*, vol. 35, no. 1, pp. 64–77, Jan. 2018, doi: 10.1080/02564602.2016.1246385.

[2] O. Sinanoglu, "Do you trust your chip?" in *Proc. Int. Conf. Design Technol. Integr. Syst. Nanosc. Era (DTIS)*, Apr. 2016, doi: 10.1109/dtis.2016.7483804.

[3] A. Antonopoulos, C. Kapatsori, and Y. Makris, "Trusted analog/mixed-signal/RF ICs: A survey and a perspective," *IEEE Des. Test*, vol. 34, no. 6, pp. 63–76, Dec. 2017, doi: 10.1109/mdat.2017.2728366.

[4] 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, doi: 10.1109/jproc.2014.2334493.

[5] M. Banga and M. S. Hsiao, "A region based approach for the identification of hardware Trojans," in *Proc. IEEE Int. Workshop Hardw.-Oriented Secur. Trust*, Jun. 2008, pp. 40–47, doi: 10.1109/hst.2008.4559047.

[6] X. Wang, M. Tehranipoor, and J. Plusquellic, "Detecting malicious inclusions in secure hardware: Challenges and solutions," in *Proc. IEEE Int. Workshop Hardw.-Oriented Secur. Trust*, Jun. 2008, pp. 15–19, doi: 10.1109/hst.2008.4559039.

[7] M. Tehranipoor and F. Koushanfar, "A survey of hardware Trojan taxonomy and detection," *IEEE Des. Test Comput.*, vol. 27, no. 1, pp. 10–15, Jan./Feb. 2010, doi: 10.1109/MDT.2010.7.

[8] A. Bazzazi, S. M. TaghiManzuri, and A. Hemmatyar, "Trojan counteraction in hardware: A survey and new taxonomy," *Indian J. Sci. Technol.*, vol. 9, no. 18, pp. 1–9, May 2016, doi: 10.17485/ijst/2016/v9i18/93764.

[9] J. Zhang, F. Yuan, L. Wei, Y. Liu, and Q. Xu, "VeriTrust: Verification for hardware trust," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 7, pp. 1148–1161, Jul. 2015, doi: 10.1109/tcad.2015.2422836.

[10] M. Tehranipoor, H. Salmani, X. Zhang, M. Wang, R. Karri, J. Rajendran, and K. Rosenfeld, "Trustworthy hardware: Trojan detection and design-for-trust challenges," *Computer*, vol. 44, no. 7, pp. 66–74, Jul. 2011, doi: 10.1109/mc.2010.369.

[11] M. Tehranipoor, "New directions in hardware security," in *Proc. 29th Int. Conf. VLSI Design 15th Int. Conf. Embedded Syst. (VLSID)*, Jan. 2016, pp. 50–52, doi: 10.1109/vlsid.2016.149.

[12] Y.-Q. Lv, Q. Zhou, Y.-C. Cai, and G. Qu, "Trusted integrated circuits: The problem and challenges," *J. Comput. Sci. Technol.*, vol. 29, no. 5, pp. 918–928, Sep. 2014.

[13] H. Li, Q. Liu, J. Zhang, and Y. Lyu, "A survey of hardware Trojan detection, diagnosis and prevention," in *Proc. 14th Int. Conf. Comput.-Aided Design Comput. Graph. (CAD/Graph.)*, Aug. 2015, pp. 173–180, doi: 10.1109/cadgraphics.2015.41.

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

[15] J. Zhang and G. Qu, "Recent attacks and defenses on FPGA-based systems," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 12, no. 3, pp. 14:1–14:24, Aug. 2019.

[16] K. Xiao, D. Forte, Y. Jin, R. Karri, S. Bhunia, and M. Tehranipoor, "Hardware Trojans: Lessons learned after one decade of research," *ACM Trans. Des. Automat. Electron. Syst.*, vol. 22, no. 1, pp. 1–23, May 2016, doi: 10.1145/2906147.

[17] S. Moein, S. Khan, T. A. Gulliver, F. Gebali, and M. W. El-Kharashi, "An attribute based classification of hardware Trojans," in *Proc. 10th Int. Conf. Comput. Eng. Syst. (ICCES)*, Dec. 2015, pp. 351–356, doi: 10.1109/icces.2015.7393074.

[18] X. Wang, Y. Zheng, A. Basak, and S. Bhunia, "IIPS: Infrastructure IP for secure SoC design," *IEEE Trans. Comput.*, vol. 64, no. 8, pp. 2226–2238, Aug. 2015, doi: 10.1109/tc.2014.2360535.

[19] L. Daoud, "Secure network-on-chip architectures for MPSoC: Overview and challenges," in *Proc. IEEE 61st Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2018, pp. 542–543, doi: 10.1109/mwscas.2018.8623831.

[20] A. Basak, S. Bhunia, T. Tkacik, and S. Ray, "Security assurance for system-on-chip designs with untrusted IPs," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 7, pp. 1515–1528, Jul. 2017, doi: 10.1109/tifs.2017.2658544.

[21] M. I. Khan and M. A. Kabir, "Review of design for security techniques: Advancement and challenges," in *Proc. Int. Conf. Electr., Comput. Commun. Eng. (ECCE)*, Feb. 2017, pp. 406–410, doi: 10.1109/ecace.2017.7912939.

[22] Y. Xie, C. Bao, C. Serafy, T. Lu, A. Srivastava, and M. Tehranipoor, "Security and vulnerability implications of 3D ICs," *IEEE Trans. Multi-Scale Comput. Syst.*, vol. 2, no. 2, pp. 108–122, Apr. 2016, doi: 10.1109/tmscs.2016.2550460.

[23] S. Dupuis, G. Di Natale, M.-L. Flottes, and B. Rouzeyre, "On the effectiveness of hardware Trojan horse detection via side-channel analysis," *Inf. Secur. J., A Global Perspective*, vol. 22, nos. 5–6, pp. 226–236, Nov. 2013, doi: 10.1080/19393555.2014.891277.

[24] J. Dofe, Q. Yu, H. Wang, and E. Salman, "Hardware security threats and potential countermeasures in emerging 3D ICs," in *Proc. 26th Ed. Great Lakes Symp. VLSI (GLSVLSI)*, 2016, pp. 69–74, doi: 10.1145/2902961.2903014.

[25] T. Boraten and A. K. Kodi, "Packet security with path sensitization for NoCs," in *Proc. Design, Automat. Test Eur. Conf. Exhibit. (DATE)*, 2016, pp. 1136–1139.

[26] T. Boraten, D. Ditomaso, and A. K. Kodi, "Secure model checkers for network-on-chip (NoC) architectures," in *Proc. 26th Ed. Great Lakes Symp. VLSI (GLSVLSI)*, 2016, pp. 45–50, doi: 10.1145/2902961.2903032.

[27] T. Boraten and A. K. Kodi, "Mitigation of denial of service attack with hardware Trojans in NoC architectures," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. (IPDPS)*, May 2016, pp. 1091–1100, doi: 10.1109/ipdps.2016.59.

[28] J. Frey and Q. Yu, "Exploiting state obfuscation to detect hardware Trojans in NoC network interfaces," in *Proc. IEEE 58th Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2015, pp. 1–4, doi: 10.1109/mwscas.2015.7282167.

[29] N. Fern, I. San, C. K. Koc, and K.-T.-T. Cheng, "Hiding hardware Trojan communication channels in partially specified SoC bus functionality," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 36, no. 9, pp. 1435–1444, Sep. 2017, doi: 10.1109/tcad.2016.2638439.

[30] P. Poorani and B. M. E. Vijayashree, "Implementation of AHB bus protocol for system on chip security," in *Proc. 2nd Int. Conf. Electron. Commun. Syst. (ICECS)*, Feb. 2015, pp. 1407–1413, doi: 10.1109/ecs.2015.7124816.

[31] F. Koushanfar and A. Mirhoseini, "A unified framework for multimodal submodular integrated circuits Trojan detection," *IEEE Trans. Inf. Forensics Security*, vol. 6, no. 1, pp. 162–174, Mar. 2011, doi: 10.1109/tifs.2010.2096811.

[32] C. Liu, P. Cronin, and C. Yang, "A mutual auditing framework to protect IoT against hardware Trojans," in *Proc. 21st Asia South Pacific Design Automat. Conf. (ASP-DAC)*, Jan. 2016, pp. 69–74, doi: 10.1109/aspdac.2016.7427991.

[33] A. P. Johnson, R. S. Chakraborty, and D. Mukhopadhyay, "A PUF-enabled secure architecture for FPGA-based IoT applications," *IEEE Trans. Multi-Scale Comput. Syst.*, vol. 1, no. 2, pp. 110–122, Apr. 2015, doi: 10.1109/tmscs.2015.2494014.

[34] L.-W. Kim and J. D. Villasenor, "Dynamic function verification for system on chip security against hardware-based attacks," *IEEE Trans. Rel.*, vol. 64, no. 4, pp. 1229–1242, Dec. 2015, doi: 10.1109/tr.2015.2447111.

[35] J. Francq and F. Frick, "Introduction to hardware Trojan detection methods," in *Proc. Design, Automat. Test Eur. Conf. Exhibit. (DATE)*, 2015, pp. 770–775, doi: 10.7873/date.2015.1101.

[36] R. Karri, J. Rajendran, K. Rosenfeld, and M. Tehranipoor, "Trustworthy hardware: Identifying and classifying hardware Trojans," *Computer*, vol. 43, no. 10, pp. 39–46, Oct. 2010, doi: 10.1109/mc.2010.299.

[37] J. Rajendran, E. Gavas, J. Jimenez, V. Padman, and R. Karri, "Towards a comprehensive and systematic classification of hardware Trojans," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2010, pp. 1871–1874.

[38] S. Moein, T. A. Gulliver, F. Gebali, and A. Alkandari, "A new characterization of hardware Trojans," *IEEE Access*, vol. 4, pp. 2721–2731, 2016, doi: 10.1109/access.2016.2575039.

[39] N. Jacob, J. Heyszl, G. Sigl, and D. Merli, "Hardware Trojans: Current challenges and approaches," *IET Comput. Digit. Techn.*, vol. 8, no. 6, pp. 264–273, Nov. 2014, doi: 10.1049/iet-cdt.2014.0039.

[40] F. N. Esirci and A. A. Bayrakci, "Hardware Trojan detection based on correlated path delays in defiance of variations with spatial correlations," in *Proc. Design, Automat. Test Eur. Conf. Exhibit. (DATE)*, Mar. 2017, pp. 163–168, doi: 10.23919/date.2017.7926976.

[41] F. S. Hossain, T. Yoneda, M. Inoue, and A. Orailoglu, "Detecting hardware Trojans without a Golden IC through clock-tree defined circuit partitions," in *Proc. 22nd IEEE Eur. Test Symp. (ETS)*, May 2017, pp. 1–6, doi: 10.1109/ets.2017.7968246.

[42] A. Nahiyan, K. Xiao, K. Yang, Y. Jin, D. Forte, and M. Tehranipoor, "AVFSM: A framework for identifying and mitigating vulnerabilities in FSMs," in *Proc. 53rd Annu. Design Automat. Conf. (DAC)*, 2016, pp. 1–6, doi: 10.1145/2897937.2897992.

[43] T. Meade, S. Zhang, and Y. Jin, "Netlist reverse engineering for high-level functionality reconstruction," in *Proc. 21st Asia South Pacific Design Automat. Conf. (ASP-DAC)*, Jan. 2016, pp. 655–660, doi: 10.1109/aspdac.2016.7428086.

[44] F. Farahmandi, Y. Huang, and P. Mishra, "Trojan localization using symbolic algebra," in *Proc. 22nd Asia Pacific Design Automat. Conf. (ASP-DAC)*, Feb. 2017, pp. 591–597, doi: 10.1109/ASPDAC.2017.7858388.

[45] M. Yoshimura, T. Bouyashiki, and T. Hosokawa, "A hardware Trojan circuit detection method using activation sequence generations," in *Proc. IEEE 22nd Pacific Rim Int. Symp. Dependable Comput. (PRDC)*, Jan. 2017, pp. 221–222, doi: 10.1109/prdc.2017.40.

[46] J. Aarestad, D. Acharyya, R. Rad, and J. Plusquellic, "Detecting Trojans through leakage current analysis using multiple supply pad I_{DDQ}s," *IEEE Trans. Inf. Forensics Security*, vol. 5, no. 4, pp. 893–904, Dec. 2010, doi: 10.1109/tifs.2010.2061228.

[47] J. Dubeuf, D. Hely, and R. Karri, "Run-time detection of hardware Trojans: The processor protection unit," in *Proc. 18TH IEEE Eur. TEST Symp. (ETS)*, May 2013, pp. 1–6, doi: 10.1109/ets.2013.6569378.

[48] D. Forte, C. Bao, and A. Srivastava, "Temperature tracking: An innovative run-time approach for hardware Trojan detection," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2013, pp. 532–539, doi: 10.1109/iccad.2013.6691167.

[49] T. Wehbe, V. J. Mooney, A. Q. Javaid, and O. T. Inan, "A novel physiological features-assisted architecture for rapidly distinguishing health problems from hardware Trojan attacks and errors in medical devices," in *Proc. IEEE Int. Symp. Hardw. Oriented Secur. Trust (HOST)*, May 2017, pp. 106–109, doi: 10.1109/hst.2017.7951807.

[50] S. K. Haider, C. Jin, M. Ahmad, D. M. Shila, O. Khan, and M. Van Dijk, "Advancing the state-of-the-art in hardware Trojans detection," *IEEE Trans. Dependable Secure Comput.*, vol. 16, no. 1, pp. 18–32, Jan. 2019.

[51] G. Zarrinchian and M. S. Zamani, "Latch-based structure: A high resolution and self-reference technique for hardware Trojan detection," *IEEE Trans. Comput.*, vol. 66, no. 1, pp. 100–113, Jan. 2017, doi: 10.1109/tc.2016.2576444.

[52] K. Xiao and M. Tehranipoor, "BISA: Built-in self-authentication for preventing hardware Trojan insertion," in *Proc. IEEE Int. Symp. Hardw.-Oriented Secur. Trust (HOST)*, Jun. 2013, pp. 45–50, doi: 10.1109/hst.2013.6581564.

[53] H. Salmani and M. M. Tehranipoor, "Vulnerability analysis of a circuit layout to hardware Trojan insertion," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 6, pp. 1214–1225, Jun. 2016, doi: 10.1109/tifs.2016.2520910.

[54] J. J. Rajendran, O. Sinanoglu, and R. Karri, "Building trustworthy systems using untrusted components: A high-level synthesis approach," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 9, pp. 2946–2959, Sep. 2016, doi: 10.1109/tvlsi.2016.2530092.

[55] B. Khaleghi, A. Ahari, H. Asadi, and S. Bayat-Sarmadi, "FPGA-based protection scheme against hardware Trojan horse insertion using dummy logic," *IEEE Embedded Syst. Lett.*, vol. 7, no. 2, pp. 46–50, Jun. 2015, doi: 10.1109/les.2015.2406791.

[56] R. Elnaggar and K. Chakrabarty, "Machine learning for hardware security: Opportunities and risks," *J. Electron. Test.*, vol. 34, no. 2, pp. 183–201, Apr. 2018, doi: 10.1007/s10836-018-5726-9.

[57] E. Alpaydin, "Introduction," in *Introduction to Machine Learning*. Cambridge, MA, USA: MIT Press, 2014, pp. 1–18.

[58] Y. Liu, G. Volanis, K. Huang, and Y. Makris, "Concurrent hardware Trojan detection in wireless cryptographic ICs," in *Proc. IEEE Int. Test Conf. (ITC)*, Oct. 2015, pp. 1–8, doi: 10.1109/test.2015.7342386.

[59] K. Hasegawa, M. Yanagisawa, and N. Togawa, "A hardware-Trojan classification method using machine learning at gate-level netlists based on Trojan features," *IEICE Trans. Fundam.*, vol. E100-A, no. 7, pp. 1427–1438, Jul. 2017.

[60] K. Hasegawa, M. Yanagisawa, and N. Togawa, "Trojan-feature extraction at gate-level netlists and its application to hardware-Trojan detection using random forest classifier," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2017, pp. 1–14.

[61] K. Hasegawa, M. Yanagisawa, and N. Togawa, "Hardware Trojans classification for gate-level netlists using multi-layer neural networks," in *Proc. IEEE 23rd Int. Symp. On-Line Test. Robust Syst. Design (IOLTS)*, Jul. 2017, pp. 227–232.

[62] T. Hoque, J. Cruz, P. Chakraborty, and S. Bhunia, "Hardware IP trust validation: Learn (the untrustworthy), and verify," in *Proc. Int. Test Conf. (ITC)*, Oct. 2018, pp. 1–10.

[63] X. Chen, L. Wang, Y. Wang, Y. Liu, and H. Yang, "A general framework for hardware Trojan detection in digital circuits by statistical learning algorithms," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 36, no. 10, pp. 1633–1646, Oct. 2017, doi: 10.1109/tcad.2016.2638442.

[64] Y. Jin, D. Maliuk, and Y. Makris, "A post-deployment IC trust evaluation architecture," in *Proc. IEEE 19th Int. On-Line Test. Symp. (IOLTS)*, Jul. 2013, pp. 224–225, doi: 10.1109/iolts.2013.6604083.

[65] C. Bao, D. Forte, and A. Srivastava, "On application of one-class SVM to reverse engineering-based hardware Trojan detection," in *Proc. 15th Int. Symp. Qual. Electron. Design*, Mar. 2014, pp. 47–54, doi: 10.1109/isqed.2014.6783305.

[66] J. Li, L. Ni, J. Chen, and E. Zhou, "A novel hardware Trojan detection based on BP neural network," in *Proc. 2nd IEEE Int. Conf. Comput. Commun. (ICCC)*, Oct. 2016, pp. 2790–2794, doi: 10.1109/compcomm.2016.7925206.

[67] L. Ni, J. Li, S. Lin, and D. Xin, "A method of noise optimization for Hardware Trojans detection based on BP neural network," in *Proc. 2nd IEEE Int. Conf. Comput. Commun. (ICCC)*, Oct. 2016, pp. 2800–2804, doi: 10.1109/compcomm.2016.7925208.

[68] 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, doi: 10.1109/cci.2016.7778952.

[69] C. X. Bao, Y. Xie, Y. Liu, and A. Srivastava, "Reverse engineering-based hardware Trojan detection," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 35, no. 1, pp. 49–57, Jan. 2016, doi: 10.1109/TCAD.2015.2488495.

[70] E.-R. Zhou, S.-Q. Li, J.-H. Chen, L. Ni, Z.-X. Zhao, and J. Li, "A novel detection method for hardware Trojan in third party IP cores," in *Proc. Int. Conf. Inf. Syst. Artif. Intell. (ISAI)*, Jun. 2016, pp. 528–532, doi: 10.1109/isai.2016.0118.

[71] A. N. Nowroz, K. Hu, F. Koushanfar, and S. Reda, "Novel techniques for high-sensitivity hardware Trojan detection using thermal and power maps," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 33, no. 12, pp. 1792–1805, Dec. 2014, doi: 10.1109/tcad.2014.2354293.

[72] N. Karimian, F. Tehranipoor, M. T. Rahman, S. Kelly, and D. Forte, "Genetic algorithm for hardware Trojan detection with ring oscillator network (RON)," in *Proc. IEEE Int. Symp. Technol. Homeland Secur. (HST)*, Apr. 2015, pp. 1–6, doi: 10.1109/ths.2015.7225334.

[73] Y. Liu, Y. Jin, A. Nosratinia, and Y. Makris, "Silicon demonstration of hardware Trojan design and detection in wireless cryptographic ICs," *IEEE Trans. VLSI Syst.*, vol. 25, no. 4, pp. 1506–1519, Apr. 2017, doi: 10.1109/tvlsi.2016.2633348.

[74] M. Xue, J. Wang, and A. Hux, "An enhanced classification-based golden chips-free hardware Trojan detection technique," in *Proc. IEEE Asian Hardw.-Oriented Secur. Trust (AsianHOST)*, Dec. 2016, pp. 1–6, doi: 10.1109/asianhost.2016.7835553.

[75] A. Marcelli, M. Restifo, E. Sanchez, and G. Squillero, "An evolutionary approach to hardware encryption and Trojan-horse mitigation," in *Proc. Design, Automat. Test Eur. Conf. Exhibit. (DATE)*, Mar. 2017, pp. 1597–1602, doi: 10.23919/DATE.2017.7927244.

[76] Y. Jin, D. Maliuk, and Y. Makris, "Post-deployment trust evaluation in wireless cryptographic ICs," in *Proc. Design, Automat. Test Eur. Conf. Exhibit. (DATE)*, Mar. 2012, pp. 965–970, doi: 10.1109/date.2012.6176636.

[77] M. Rostami, F. Koushanfar, and R. Karri, "A primer on hardware security: Models, methods, and metrics," *Proc. IEEE*, vol. 102, no. 8, pp. 1283–1295, Aug. 2014, doi: 10.1109/jproc.2014.2335155.

[78] F. Courbon, P. Loubet-Moundi, J. J. Fournier, and A. Tria, "SEMBA: A SEM based acquisition technique for fast invasive Hardware Trojan detection," in *Proc. Eur. Conf. Circuit Theory Design (ECCTD)*, Aug. 2015, pp. 1–4, doi: 10.1109/ecctd.2015.7300097.

[79] R. Torrance and D. James, "The state-of-the-art in semiconductor reverse engineering," in *Proc. 48th Design Automat. Conf. (DAC)*, Aug. 2011, pp. 333–338.

[80] Y. Shiyanovskii, F. Wolff, A. Rajendran, C. Papachristou, D. Weyer, and W. Clay, "Process reliability based Trojans through NBTI and HCI effects," in *Proc. NASA/ESA Conf. Adapt. Hardw. Syst.*, Jun. 2010, pp. 215–222, doi: 10.1109/ahs.2010.5546257.

[81] J. Smith, "Non-destructive state machine reverse engineering," in *Proc. 6th Int. Symp. Resilient Control Syst. (ISRCS)*, Aug. 2013, pp. 120–124, doi: 10.1109/isrcs.2013.6623762.

[82] W. Li, Z. Wasson, and S. A. Seshia, "Reverse engineering circuits using behavioral pattern mining," in *Proc. IEEE Int. Symp. Hardw.-Oriented Secur. Trust*, Jun. 2012, pp. 83–88, doi: 10.1109/hst.2012.6224325.

[83] X. Chen, Q. Liu, S. Yao, J. Wang, Q. Xu, Y. Wang, Y. Liu, and H. Yang, "Hardware Trojan detection in third-party digital intellectual property cores by multilevel feature analysis," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 7, pp. 1370–1383, Jul. 2018, doi: 10.1109/tcad.2017.2748021.

[84] H. Salmani, "COTD: Reference-free hardware Trojan detection and recovery based on controllability and observability in gate-level netlist," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 2, pp. 338–350, Feb. 2017, doi: 10.1109/tifs.2016.2613842.

[85] B. Çakir and S. Malik, "Hardware Trojan detection for gate-level ICs using signal correlation based clustering," in *Proc. Design, Automat. Test Eur. Conf. Exhibit. (DATE)*, 2015, pp. 471–476.

[86] D. Jap, W. He, and S. Bhasin, "Supervised and unsupervised machine learning for side-channel based Trojan detection," in *Proc. IEEE 27th Int. Conf. Appl.-Specific Syst., Archit. Processors (ASAP)*, Jul. 2016, pp. 17–24, doi: 10.1109/asap.2016.7760768.

[87] T. Iwase, Y. Nozaki, M. Yoshikawa, and T. Kumaki, "Detection technique for hardware Trojans using machine learning in frequency domain," in *Proc. IEEE 4th Global Conf. Consum. Electron. (GCCE)*, Oct. 2015, pp. 185–186, doi: 10.1109/gcce.2015.7398569.

[88] F. K. Lodhi, S. R. Hasan, O. Hasan, and F. Awwadl, "Power profiling of microcontroller's instruction set for runtime hardware Trojans detection without golden circuit models," in *Proc. Design, Automat. Test Europe Conf. Exhibit. (DATE)*, Mar. 2017, pp. 294–297, doi: 10.23919/date.2017.7927002.

[89] 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, doi: 10.1109/iscas.2016.7538895.

[90] Q. Cui, K. Sun, S. Wang, L. Zhang, and D. Li, "Hardware Trojan detection based on cluster analysis of Mahalanobis distance," in *Proc. 8th Int. Conf. Intell. Hum.-Mach. Syst. (IHMSC)*, Aug. 2016, pp. 234–238, doi: 10.1109/ihmsc.2016.65.

[91] M. Xue, R. Bian, W. Liu, and J. Wang, "Defeating untrustworthy testing parties: A novel hybrid clustering ensemble based golden models-free hardware Trojan detection method," *IEEE Access*, vol. 7, pp. 5124–5140, 2019.

[92] S. Narasimhan, W. Yueh, X. Wang, S. Mukhopadhyay, and S. Bhunia, "Improving IC security against Trojan attacks through integration of security monitors," *IEEE Des. Test. Comput.*, vol. 29, no. 5, pp. 37–46, Oct. 2012, doi: 10.1109/mdt.2012.2210183.

[93] H. Salmani, M. Tehranipoor, and J. Plusquellic, "A novel technique for improving hardware Trojan detection and reducing Trojan activation time," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 1, pp. 112–125, Jan. 2012, doi: 10.1109/tvlsi.2010.2093547.

[94] T. F. Wu, K. Ganesan, Y. A. Hu, H.-S.-P. Wong, S. Wong, and S. Mitra, "TPAD: Hardware Trojan prevention and detection for trusted integrated circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 35, no. 4, pp. 521–534, Apr. 2016, doi: 10.1109/tcad.2015.2474373.

[95] C. Dong, G. He, X. Liu, Y. Yang, and W. Guo, "A multi-layer hardware Trojan protection framework for IoT chips," *IEEE Access*, vol. 7, pp. 23628–23639, 2019.

[96] B. Shanyour and S. Tragoudas, "Detection of low power Trojans in standard cell designs using built-in current sensors," in *Proc. IEEE Int. Test Conf. (ITC)*, Oct. 2018, pp. 1–10.

[97] S. Koteshwara, C. H. Kim, and K. K. Parhi, "Key-based dynamic functional obfuscation of integrated circuits using sequentially triggered mode-based design," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 1, pp. 79–93, Jan. 2018, doi: 10.1109/tifs.2017.2738600.

[98] B. Liu and B. Wang, "Reconfiguration-based VLSI design for security," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 5, no. 1, pp. 98–108, Mar. 2015, doi: 10.1109/jetcas.2014.2372431.

[99] S. Cadambi, I. Durdanovic, V. Jakkula, M. Sankaradass, E. Cosatto, S. Chakradhar, and H. P. Graf, "A massively parallel FPGA-based coprocessor for support vector machines," in *Proc. 17th IEEE Symp. Field Program. Custom Comput. Mach.*, Apr. 2009, pp. 115–122, doi: 10.1109/fccm.2009.34.

[100] D. Maliuk, H.-G. Stratigopoulos, H. Huang, and Y. Makris, "Analog neural network design for RF built-in self-test," in *Proc. IEEE Int. Test Conf.*, Nov. 2010, pp. 1–10, doi: 10.1109/test.2010.5699272.

[101] J. Rajendran, H. Zhang, C. Zhang, G. S. Rose, Y. Pino, O. Sinanoglu, and R. Karri, "Fault analysis-based logic encryption," *IEEE Trans. Comput.*, vol. 64, no. 2, pp. 410–424, Feb. 2015, doi: 10.1109/tc.2013.193.

[102] R. Karmakar, N. Prasad, S. Chattopadhyay, R. Kapur, and I. Sengupta, "A new logic encryption strategy ensuring key interdependency," in *Proc. 30th Int. Conf. VLSI Design 16th Int. Conf. Embedded Syst. (VLSID)*, Jan. 2017, pp. 429–434, doi: 10.1109/vlsid.2017.29.

[103] Y. Liu, K. Huang, and Y. Makris, "Hardware Trojan detection through golden chip-free statistical side-channel fingerprinting," in *Proc. 51st ACM/EDAC/IEEE Design Automat. Conf. (DAC)*, Jun. 2014, pp. 1–6, doi: 10.1109/dac.2014.6881482.

[104] C.-X. Wang, S.-Y. Zhao, X.-S. Wang, M. Luo, and M. Yang, "A neural network Trojan detection method based on particle swarm optimization," in *Proc. 14th IEEE Int. Conf. Solid-State Integr. Circuit Technol. (ICSICT)*, Oct. 2018, pp. 1–3.

[105] S. Sethumadhavan, A. Waksman, M. Suozzo, Y. Huang, and J. Eum, "Trustworthy hardware from untrusted components," *Commun. ACM*, vol. 58, no. 9, pp. 60–71, Aug. 2015, doi: 10.1145/2699412.

[106] A. Deorio, D. Fick, V. Bertacco, D. Sylvester, D. Blaauw, J. Hu, and G. Chen, "A reliable routing architecture and algorithm for NoCs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 31, no. 5, pp. 726–739, May 2012, doi: 10.1109/tcad.2011.2181509.

[107] A. Kulkarni, Y. Pino, and T. Mohsenin, "SVM-based real-time hardware Trojan detection for many-core platform," in *Proc. 17th Int. Symp. Quality Electron. Design (ISQED)*, Mar. 2016, pp. 362–367, doi: 10.1109/isqed.2016.7479228.

[108] A. Kulkarni, Y. Pino, and T. Mohsenin, "Adaptive real-time Trojan detection framework through machine learning," in *Proc. IEEE Int. Symp. Hardw. Oriented Secur. Trust (HOST)*, May 2016, pp. 120–123, doi: 10.1109/hst.2016.7495568.

[109] N. Fern, I. San, Ç. K. Koç, and K.-T. Cheng, "Hardware Trojans in incompletely specified on-chip bus systems," in *Proc. Design, Automat. Test Eur. Conf. Exhibit. (DATE)*, Mar. 2016, pp. 527–530.

[110] T. Wehbe and X. Wang, "Secure and dependable NoC-connected systems on an FPGA chip," *IEEE Trans. Rel.*, vol. 65, no. 4, pp. 1852–1863, Dec. 2016, doi: 10.1109/tr.2016.2606883.
[111] K. Madden, J. Harkin, L. Mcdaid, and C. Nugent, "Adding security to networks-on-chip using neural networks," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Nov. 2018, pp. 1299–1306.
[112] A. Kulkarni, Y. Pino, M. French, and T. Mohsenin, "Real-time anomaly detection framework for many-core router through machine-learning techniques," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 13, no. 1, pp. 10:1–10:22, Jun. 2016.
[113] Y. Jin and D. Sullivan, "Real-time trust evaluation in integrated circuits," in *Proc. Design, Automat. Test Europe Conf. Exhibit. (DATE)*, 2014, pp. 1–6, doi: 10.7873/date2014.104.
[114] K. Guha, D. Saha, and A. Chakrabarti, "RTNA: Securing SOC architectures from confidentiality attacks at runtime using ART1 neural networks," in *Proc. 19th Int. Symp. VLSI Design Test*, Jun. 2015, pp. 1–6, doi: 10.1109/isvdat.2015.7208048.
[115] S. Y. Yang and H. Zhang, "Feature selection and optimization," in *Pattern Recognization and Intelligent Computing*, 3th ed. Beijing, China: PHEI, 2015, ch. 2.1, sec. 2, pp. 27–28.
[116] S. R. Hasan, S. F. Mossa, O. S. A. Elkeelany, and F. Awwad, "Tenacious hardware Trojans due to high temperature in middle tiers of 3-D ICs," in *Proc. IEEE 58th Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2015, pp. 1–4, doi: 10.1109/mwscas.2015.7282148.
[117] Z. Chen, P. Zhou, T.-Y. Ho, and Y. Jin, "How secure is split manufacturing in preventing hardware Trojan?" in *Proc. IEEE Asian Hardw.-Oriented Secur. Trust (AsianHOST)*, Dec. 2016, pp. 1–6, doi: 10.1109/asianhost.2016.7835561.
[118] S. Ghosh, A. Basak, and S. Bhunia, "How secure are printed circuit boards against Trojan attacks?" *IEEE Des. Test*, vol. 32, no. 2, pp. 7–16, Apr. 2015, doi: 10.1109/mdat.2014.2347918.
[119] D. Fujimoto, S. Nin, Y.-I. Hayashi, N. Miura, M. Nagata, and T. Matsumoto, "A demonstration of a HT-detection method based on impedance measurements of the wiring around ICs," *IEEE Trans. Circuits Syst., II, Exp. Briefs*, vol. 65, no. 10, pp. 1320–1324, Oct. 2018.
[120] Z. Guo, X. Xu, M. M. Tehranipoor, and D. Forte, "MPA: Model-assisted PCB attestation via board-level RO and temperature compensation," in *Proc. Asian Hardw. Oriented Secur. Trust Symp. (AsianHOST)*, Oct. 2017, pp. 25–30.
[121] S. Nin, D. Fujimoto, and Y. I. Hayashi, "HT-detection method based on impedance measurements of ICs," in *Proc. IEEE Intl. Symp. Electromagn. Compat. IEEE Asia–Pacific Symp. Electromagn. Compat. (EMC/APEMC)*, May 2018, p. 11.
[122] W. E. Cobb, E. D. Laspe, R. O. Baldwin, M. A. Temple, and Y. C. Kim, "Intrinsic physical-layer authentication of integrated circuits," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 1, pp. 14–24, Feb. 2012, doi: 10.1109/tifs.2011.2160170.
[123] N. Beringuier-Boher, D. Hely, V. Beroulle, J. Damiens, and P. Candelier, "Increasing the security level of analog IPs by using a dedicated vulnerability analysis methodology," in *Proc. Int. Symp. Qual. Electron. Design (ISQED)*, Mar. 2013, pp. 531–537, doi: 10.1109/isqed.2013.6523662.
[124] Z. Huang and Q. Wang, "MSIPS: Multi-tiered security IPs architecture for secure SoC design," in *Proc. Int. Conf. Netw. Netw. Appl. (NaNA)*, Oct. 2017, pp. 203–208, doi: 10.1109/nana.2017.43.
[125] A. Sengupta, S. Bhadauria, and S. P. Mohanty, "TL-HLS: Methodology for low cost hardware Trojan security aware scheduling with optimal loop unrolling factor during high level synthesis," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 36, no. 4, pp. 655–668, Apr. 2017, doi: 10.1109/tcad.2016.2597232.
[126] L. Sustek, "Hardware security module," in *Encyclopedia of Cryptography and Security*. Boston, MA, USA: Springer, 2011, pp. 535–537.
[127] R. Coombs. (2018). *How PSA APIs Will Enable Secure Devices and a Consistent Developer Experience*. ARM. [Online]. Available: https://community.arm.com/iot/b/blog
[128] B. Mann. (2017). *Platform Security Architecture-Scalable Security for the IoT*. ARM. [Online]. Available: https://community.arm.com/processors/b/blog
[129] N. Fern and K.-T. Cheng, "Pre-silicon formal verification of JTAG instruction opcodes for Security," in *Proc. IEEE Int. Test Conf. (ITC)*, Oct. 2018, pp. 1–9.
[130] B.-C. Choi, S.-H. Lee, J.-C. Na, and J.-H. Lee, "Secure firmware validation and update for consumer devices in home networking," *IEEE Trans. Consum. Electron.*, vol. 62, no. 1, pp. 39–44, Feb. 2016, doi: 10.1109/tce.2016.7448561.
[131] Mentor. (2011). *Security topic in Internet of Things*. [Online]. Available: https://www.mentor.com/embedded-software/iot/security
[132] E. Erdin, H. Aksu, S. Uluagac, M. Vai, and K. Akkaya, "OS independent and hardware-assisted insider threat detection and prevention framework," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, Oct. 2018, pp. 926–932.
[133] J. Ye, Y. Hu, and X. Li, "Hardware Trojan in FPGA CNN accelerator," in *Proc. IEEE 27th Asian Test Symp. (ATS)*, Oct. 2018, pp. 68–73.
[134] Z. Kahleifeh, S. D. Kumar, and H. Thapliyal, "Hardware Trojan detection in implantable medical devices using adiabatic computing," in *Proc. IEEE Int. Conf. Rebooting Comput. (ICRC)*, Nov. 2018, pp. 1–6.
[135] H. Mohammed, J. Howell, S. R. Hasan, N. Guo, F. Khalid, and O. Elkeelany, "Hardware Trojan based security issues in home area network: A testbed setup," in *Proc. IEEE 61st Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2018, pp. 972–975.

ZHAO HUANG received the B.S. degree from the College of Computer Science and Technology, Shandong University of Technology, Zibo, China, in 2012, and the M.S. degree from the School of Computer Science and Technology, Xidian University, Xi'an, China, in 2015, where he is currently pursuing the Ph.D. degree. His current research interests include hardware security, hardware Trojan detection, physically unclonable function (PUF), secure architecture, and bus security. He has been a Student Member of China Computer Federation (CCF), since 2016.

QUAN WANG (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in computer science from Xidian University, Xi'an, China, in 1992, 1997, and 2008, respectively. He is currently a Professor with the School of Computer Science and Technology and the Vice President of Xidian University. His research interests include hardware security, embedded systems, wireless networks, and 3-D printing. He is also a Distinguished Member of China Computer Federation (CCF) and a Councilor of ACM Xi'an Section.

YIN CHEN (Member, IEEE) received the B.S. and M.S. degrees from the School of Computer Science and Technology, Xidian University, Xi'an, China, in 2008 and 2011, respectively, and the Ph.D. degree from the School of Systems Information Science, Future University–Hakodate, Japan, in 2014. From April to October 2014, he was a Postdoctoral Researcher with Future University–Hakodate. He is currently serving as a Senior Research Assistant Professor with the Graduate School of Media and Governance, Keio University–Fujisawa, Japan. His research interests are in the wide area of wireless communication and networks and their applications in the IoT and smart cities. He is a member of ACM and IPSJ.

XIAOHONG JIANG (Senior Member, IEEE) received the B.S., M.S., and Ph.D. degrees from Xidian University, Xi'an, China, in 1989, 1992, and 1999, respectively. Before joining Future University–Hakodate, he was an Associate Professor at Tohoku University, Japan. He was an Assistant Professor with the Japan Advanced Institute of Science and Technology (JAIST). He is currently a Full Professor of Future University–Hakodate, Japan. He has authored and coauthored over 300 publications in journals, books, and international conference proceedings, which include the IEEE/ACM TRANSACTIONS ON NETWORKING and the IEEE JOURNAL OF SELECTED AREA ON COMMUNICATIONS. His current research interests include wireless networks and optical networks. Dr. Jiang is a member of IEICE.