Journal of Electronic Testing (2023) 39:447–463
https://doi.org/10.1007/s10836-023-06079-2

Threshold Analysis Using Probabilistic Xgboost Classifier for Hardware Trojan Detection

Tapobrata Dhar[1] · Ranit Das[2] · Chandan Giri[1] · Surajit Kumar Roy[1]

Received: 17 March 2023 / Accepted: 7 August 2023 / Published online: 7 September 2023

Abstract
The fabless nature of integrated circuits manufacturing leaves them vulnerable to modifications by ill-intentioned third party. There arises a necessity for security measures during their manufacturing to protect them from covert modifications known as hardware Trojans. Static analysis of gate-level synthesized integrated circuits can prove helpful in detecting the presence of unwanted circuitry within the host. This paper proposes a static analysis technique of gate-level integrated circuits using supervised probabilistic classifier through effective threshold analysis. New and existing relevant features are extracted that relates to hardware Trojan properties and normalised accordingly. Effective features are selected using their feature importance values. Variance threshold has been used to create a high entropy feature subset to train a supervised model using XGBoost algorithm with relevant hyperparameters. Threshold values of the probabilistic classifier are determined through analysis of threshold obtained using receiver operating characteristic and precision-recall curves. The chosen techniques showcase hardware Trojan detection with high accuracy over gate-level synthesized circuits.

Keywords Hardware Trojan · XGBoost · Variance Threshold · Receiver operating characteristic curve

1 Introduction

With the recent developments of the High Performance Computers, there has been a surge in Integrated Chip (IC) manufacture. The production of an IC is a lengthy and exhausting process, which often requires collaboration with third-party intellectual property (IP) core suppliers and foundries. Every component in the supply chain of IC manufacture is susceptible to hardware attacks. At any stage of IC design and manufacturing, a Trojan may be inserted. The security of hardware devices is often unnoticed. Any undesirable modification to an existing circuit by the addition of malicious circuitry is called a hardware Trojan (HT). It could be done by someone in the foundry or 3rd party house. A hardware Trojan has two components: A trigger and a payload. When the trigger circuitry provides an activation signal, the payload performs the main Trojan functionality. Traditional testing approaches fail to detect the presence of an HT.

HTs can affect a batch of chips by manipulating the design or fabrication of an IC. However, side-channel analysis and reverse engineering are not suitable for multiple devices [14]. Therefore, it is necessary to have a detection technique in the IC design step. The insertion of HTs into an IC is possible in two ways. The lithographic masks can be manipulated in the form of addition, deletion or modification of gates. Or there can be a malicious IP designed where the attacker may be from the third-party design house or in-house [17]. Yang et al. [22] divided HT detection algorithms in terms of (1) search-based, (2) threshold-based, and (3) machine learning-based methods. Search-based detection techniques directly search for nets that have

Responsible Editor: C. A. Papachristou

Ranit Das, Chandan Giri, and Surajit Kumar Roy have contributed equally to this work.

✉ Tapobrata Dhar
tapobrata.dhar91@gmail.com

Ranit Das
ranit3.das@gmail.com

Chandan Giri
chandan@it.iiests.ac.in

Surajit Kumar Roy
suraroy@gmail.com

1 Indian Institute of Engineering Science and Technology, Shibpur, Howrah, India

2 Samsung Research Institute, Noida, India

particular Trojan features in the circuit. Threshold-based detection techniques compare quantified parameters with Trojan thresholds. Machine learning (ML) techniques also have been a great utility in the field of HT detection.

ML-based techniques have become useful in reverse engineering detection techniques, circuit feature analysis, and side-channel analysis [13]. Reverse engineering (RE) technique obtains images of each layer of the IC in order to analyse them. However, it is a time-consuming and costly process. In circuit features analysis, gate-level netlists are used to extract functional or structural features and analysed to potentially spot the presence of a suspicious net or a gate. Side-channel analysis (SCA) analyses parametric changes in side-channel information by modification due to HTs. The disadvantage of SCA is that the parametric changes are also vulnerable to noise. The HT-net features can be identified and classified automatically in circuit feature analysis. Application of ML techniques has been found to enhance the True Positive Rate (TPR) and efficiency of HT-detection. One more advantage of this method is dimensionality reduction can be performed for the ML algorithms, thereby providing an option to judge the most important H-net features. Li et al. [16] categorized the detection of hardware Trojans into two stages: pre-silicon stage and post-silicon stage.

HTs inserted in the pre-silicon stage of IC manufacturing can be detected through static analysis of gate-level circuit features by machine learning algorithms, such as XGBoost. XGBoost is a scalable and highly performant supervised algorithm that comprises of gradient boosted decision trees. The performance of the algorithm is attributed to efficient parallelisation of computation in the distributed decision trees. The flexibility of the model enables it to address the class imbalance of the HT nets over normal nets in an IC. The model is also capable of probabilistic classification, whose threshold can be determined through performance analysis [3]. Machine learning models generally rely on features with low correlation selected through methods such as Pearson's correlation analysis for efficient performance. However, in case of highly class imbalanced HT detection problem, there arises a necessity to consider entropy over correlation amongst features. This is done to select features that contain more information while keeping the feature count low through methods such as variance threshold analysis.

The contributions of this paper are as follows:

1. New and existing features relevant to the properties of HT have been extracted from gate-level netlist.
2. Selection of high entropy features is conducted through utilisation of feature importance paired with variance threshold analysis.
3. Construction of supervised probabilistic classifier through effective hyperparameter selection using randomised search.
4. Determination of the most effective threshold for the classifier in order to effectively detect the presence of HTs by receiver operating curve and precision-recall curve analysis.
5. Highlighting the effectiveness of high entropy feature subset over low correlation feature subset for HT classification.

The outline of the paper is as follows: Section 2 describes the various static analysis of HTs in literature and the motivation behind the proposed work. Section 3 details the feature set extracted from the gate-level ICs. Section 4 describes the selection of features using XGBoost's feature importance along with variance threshold. Section 5 contains the details of the supervised algorithm utilised to train the model, its hyperparameter selection procedure, and the threshold measurement of the probabilistic classifier. Section 6 showcases the results of the cross-validation and testing experiments and comparison of the results with previous works. Section 7 ends the paper with concluding statements.

2 Motivation

HTs rely on the presence of nets IC that are significantly less active than the rest of the nets in the IC. The lack in activity in those nets during normal functioning of the IC gives the HTs avenue for circumventing detection through side-channels. However, the lack of activity in the IC and HT nets makes them exhibit features that are attributed to their covert nature. Supervised and unsupervised ML techniques can be utilised to create classifier model using the features attributed to the covertness of the nets. Recently, state-of-the-art hardware security research has seen a great utilization of machine learning algorithms. Hasegawa et al. first proposed an approach applying ML to detect hardware Trojans at gate-level netlist [9]. They extracted 5 Trojan features based on various netlists known to be Trojan infected. Several nets were trained in a binary support vector machine (SVM) classifier and an unknown set of nets from a netlist are tested in the classifier. The method turned out to be unfavourable from viewpoint of True Negative Rate (TNR) in the testing phase.

Hasegawa et al. [10] proposed a method where they extracted 51 hardware Trojan features. 11 of them are selected to be trained on a Random Forest classifier to obtain the best F-measures for Trojan classification. Several Trust-Hub benchmarks [19] were tested in the classifier. However, performance degrades when the predictions are carried on circuits with significantly larger class imbalance. In [11], 11 gate-level Trojan-net feature values previously obtained from [10] were extracted for each net and trained them using a multi-layer neural network. Finally, an unknown set of nets from a netlist were put in the learned multi-layer neural network for classifying them into Trojan class or normal class, producing an average TNR of 70%. Hoque et al. [12] introduced functional features along with structural features in supervised learning for HT detection using gate-level netlists. Multiple machine learning algorithms were incorporated using a Voting Ensemble in the framework. Nevertheless, they created new Trojan data for class balancing before training. Finally, the suspected IP is verified by the framework.

Kok et al. [15] extracted Testability features and performed class balancing using the ADASYN algorithm. They trained the dataset on four unsupervised classifiers and compared the results. Evaluation results showed that the best performance was obtained by the classifier based on Bagged Trees. However, out of all synthetic data created in the Trojan class, some of them tend to get similar to the Trojan-free class. This produces small true positives/negatives. Dong et al. proposed five new features by analysing the traditional fifty-one features [6]. They extracted 49 effective out of the 56 features, using a scoring mechanism of the XGBoost Algorithm. The framework provided an average accuracy of 99.83% but requires too much computation due to the high dimensionality of the model. Sharma et al. propose a new class-weighted XGBoost-based HT detection technique which tackles the issue of the imbalance in classes [20]. The minority Trojan class is assigned higher weights, thereby removing the purpose of applying the creation of synthetic data or oversampling.

Therefore, the problem statement can be established as follows: *Given a gate-level netlist of an IC with various interconnections between nets, there arises a need for:*

- *extraction of relevant features that are relevant to the properties of HTs,*
- *reducing the set of extracted features for performance improvement of supervised algorithm,*
- *creating the supervised learning model with appropriate choices of hyperparameters, and*
- *effective threshold measurement for the probabilistic classifier that maximises accuracy while keeping high rate of positive detection.*

The proposed work theorised in the following sections comprises of methodologies to address the issues put forth by the problem statement. New and established HT features have been extracted from gate-level netlists. The effectiveness of the extracted features have been analysed through evaluation of their importance values in the XGBoost decision trees. Feature subset has been selected from the extracted features by using feature importance paired with variance threshold. Randomised cross validation has been used to establish the most effective hyperparameters for training the probabilistic classifier. Threshold analysis has also been conducted with regards to the classifier to select threshold which yields results of high accuracy.

The model detects covert, functional HTs that uses malicious signals to introduce malfunction or information leakage upon trigger activation.

Probabilistic threshold based XGBoost classifier has been utilised in [4] for HT detection. This paper expands on the established work in the following ways:

- New features have been introduced and number of extracted gate-level features have been expanded from 35 to 85.
- Feature importance has been used to select effective features from feature groups.
- Variance threshold has been used to create high entropy feature subset to train the supervised classifier.

3 Feature Extraction and Transformation

The covert nature of HTs are attributed to various gate-level properties within the host IC that render them undetectable during fault testing. The properties that render them undetectable can be analysed to create a model based on the nature of covertness. Therefore, effective consideration of HT feature needs to be conducted for the model to accurately detect the presence of purposefully hidden nets. In this work, a total of 85 HT features have been extracted and normalised, as shown in Table 1.

3.1 Features Extracted

The extracted feature set comprises of features that are both established in recent literature and newly introduced. The full list of extracted features have been listed in Table 1. The Feature ID column specifies the ID used to refer to the features throughout the paper. The entire feature set can be broadly categorised as level specific and non-level specific feature groups. Level specific feature groups are localised features extracted from the immediate neighbourhood cones of the target net. Non-level specific feature groups include just one feature that is obtained by analysis of the entire IC. Feature groups $1-10$ and $30-33$ include level specific features, while $11-29$ include non-level specific features. The values of non-level specific features do not rely on a predefined level. However, level specific features are directly influenced by the specification of depth in logic levels of the neighbourhood cones from the target net. The range of considered levels have been depicted in the Levels column for each level specific feature group.

For example, *mux_fanin* level specific feature group denotes the number of MUXs in numerous fan-in cones of

Table 1 Extracted features from gate-level netlists

Existing Features			
No.	**Feature ID**	**Description**	**Levels**
1	*net_fanin*	No. of gates in the fan-in	1-5
2	*net_fanout*	No. of gates in the fan-out	1
3	*ff_fanin*	No. of flip-flops in the fan-in	1-5
4	*ff_fanout*	No. of flip-flops in the fan-out	1-5
5	*mux_fanin*	No. of MUXs in the fan-in	1-5
6	*mux_fanout*	No. of MUXs in the fan-out	1-5
7	*loop_fanin*	No. of loops in the fan-in	1-5
8	*loop_fanout*	No. of loops in the fan-out	1-5
9	*const_fanin*	No. of constants in the fan-in	1-5
10	*const_fanout*	No. of constants in the fan-out	1-5
11	*min_dff_fanin*	Min. levels from a DFF in the fan-in	-
12	*min_dff_fanout*	Min. levels from a DFF in the fan-out	-
13	*min_mux_fanin*	Min. levels from a MUX in the fan-in	-
14	*min_mux_fanout*	Min. levels from a MUX in the fan-out	-
15	*min_dif_pi*	Min. levels from a primary input	-
16	*min_dif_po*	Min. levels from a primary output	-
17	*pi*	No. of influential primary inputs	-
18	*po*	No. of influenced primary outputs	-
19	*ld*	Logical depth	-
20	*pfdep*	Fan-out Positively dependant gates	-
21	*nfdep*	Fan-out Negatively dependent gates	-
22	*pbdep*	Fan-in Positively dependant gates	-
23	*nbdep*	Fan-in Negatively dependent gates	-
24	$cc0$	Combinational controllability of signal 0	-
25	$cc1$	Combinational controllability of signal 1	-
26	*co*	Combinational observability	-
27	$sc0$	Sequential controllability of signal 0	-
28	$sc1$	Sequential controllability of signal 1	-
29	*so*	Sequential observability	-
Introduced Features			
No.	**Feature ID**	**Description**	**Levels**
30	*div_fanin*	No. of diverse gates in the fan-in	1-5
31	*uni_fanin*	No. of uniform gates in the fan-in	1-5
32	*div_fanout*	No. of diverse gates in the fan-out	1-5
33	*uni_fanout*	No. of uniform gates in the fan-out	1-5

the target net of various logical depths. For *mux_fanin*, the range $1-5$ in the Levels column of Table 1 denotes 5 separate MUX counts extracted from 5 different levels of logical depth in the fan-in cone. The 5 separate MUX counts are treated as 5 separate features belonging to the *mux_fanin* feature group. Level specific feature groups 1, $3-10$ and $30-33$ include neighbourhood cone logic level range $1-5$, and level specific group 2 include just 1 logic level of the cone. There are 19 non-level specific feature groups containing 1 feature each. Thus, the total number of extracted features have been evaluated as $13 \times 5 + 1 + 19 = 85$ features.

Fan-in and fan-out cones of $1-5$ level depth range is considered to be the immediate neighbourhood of a target net. Feature components beyond 5 level differences have not been considered to be part of the level specific feature groups. This is because features that are farther than 5 levels away have been considered to not be related to the target net [10]. Consideration of localised features over 5 levels leads to erroneous classification of normal nets as infected by the model trained on such features. In contrast, the non-level specific features do not rely on a pre-defined level difference.

Features 1 – 16 have been utilised in [7] and [11] for training their unsupervised model and neural networks respectively. Features 17 – 19 are derivatives of the established features where *pi* is the number of primary inputs (PIs) influencing a net, *po* is the number of primary outputs (POs) influenced by the net, and *ld* is the logical depth of the net. Features 20 – 23 have been used in [5] for determination of the vulnerability of nets for HT insertion.

Sandia Controllability and Observability (SCOAP) measures [8] 24. – 29. have been utilised as features to train unsupervised model in [21]. In general, HT nets have the tendency to exhibit low toggling in signals. Such low probabilities in signal transitions makes them exhibit high testability values. High values of combinational and sequential testability measures is a property of combinational and sequential HT nets respectively. Thus, the testability measures serve as an effective indicator for the presence of HTs within the host.

Features 1 – 19 showcase various combinational and sequential structural properties of HT. Features 24 – 29 showcase the ability of the tester to manipulate and observe the signals within the internal nets of the IC. However, the covert nature of HT is also attributed to the low switching of the signals within the nets of its triggers to a large degree. Features 20 – 23 address the low switching of the signals of a net, but the evaluation is conducted on a global level without restriction in logical depth. Local evaluation of the impact of neighbourhood gates on a target net also need to be considered to predict the degree of switching of signals effectively. Such evaluation need to be conducted in conjunction with the established feature, such that every possible situation that lead to low switching in signals is considered. Thus, there arises a necessity for effective evaluation on the impact of signal switching by the immediate neighbourhood gates. It is for the same reason that features 30 – 33 have also been introduced to be used to create the feature super-set to accompany the existing features.

These features are centred around the degree of uniformity and the diversity of signals at the gate output. HTs are made covert by suppressing activities within the infected nets so they do not get highlighted in side-channel analysis. The suppression of activities is achieved by reduced switching within the signals of the gate output. Such suppression is more prevalently noticed in gates with a certain signal likelihood. For example, *AND* and *NOR* gates have output signal likelihood of 0 and *OR* and *NAND* gates have likelihood of 1. The logic gates with special likelihood of a specific signal in their outputs can be categorised into two sets:

$$Lk_0 = \{AND, NOR\}$$
$$Lk_1 = \{OR, NAND\}$$

The likelihood of the output signals is utilised to create 20 new features. The newly introduced features are as follows:

1. *div_fanin*: When the output of a gate has likelihood of a specific signal (0 or 1), the frequency in signal switching depends on its fan-in. Fan-in of a gate containing majority of gates with the same signal likelihood would result in lack of diversity in signals in the output of the target net. Conversely, fan-in containing majority of nets having the opposite signal likelihood increases the chance of toggling in signals in the output of the target net. Therefore, increased diverse nets in fan-in of a target gate would imply increased visibility during analysis. The number of gates in the fan-in of the target gate that exhibits likelihood for the opposite signal in its output is considered as a feature as *div_fanin*. In order to measure the degree of diversity in fan-in of a target gate *X*, the following set $FIDi_X$ is evaluated as follows:

   $$FIDi_X = \{(X, Y_j) \in Lk_0 \times Lk_1 \cup Lk_1 \times Lk_0,\\ \forall Y_j \in \text{fan-in of } X \text{ up to level } n\}$$

   where $FIDi_X$is the set for measure of diversity in nodes in the fan-in cone of the target gate *X*. *div_fanin* is evaluated as the cardinality of *FIDi* set. The *div_fanin* measure is evaluated up to a maximum difference of 5 levels from the target gate for each gate.
2. *div_fanout*: Similar to *div_fanin*, the number of gates with likelihood for inverse signals than the target gate in its output is also considered. The lack of toggles of the HT nets impacts the toggling of the nets in its fan-out as well. This is even exacerbated when the gates in the fan-out have the likelihood for the same signal as the target gate. However, when the majority of gates in the fan-out have net outputs with opposite signal likelihood, it leads to increased toggles within the signals. Therefore, the count of nets with opposite signals from the target net in the fan-out also serves as an effective HT feature. The number of gates with the likelihood for the opposite signal is evaluated for each gates in the IC as *div_fanout*. The fan-out diversity set for target gate *X* can be formulated as:

   $$FODi_X = \{(X, Y_j) \in Lk_0 \times Lk_1 \cup Lk_1 \times Lk_0,\\ \forall Y_j \in \text{fan-out of } X \text{ up to level } n\}$$

   where $FODi_X$ is the set for measure of diversity in nodes in the fan-out cone of the target gate *X*. *div_fanout* is the cardinality of the set *FODi*. The feature is evaluated up to a maximum difference of 5 levels from the target gate.
3. *uni_fanin*: Uniformity in signals within the gate outputs in the fan-in of a target gate leads to decreased toggle in signal in its output. When the likelihood of the target

gate towards a particular signal is shared by the majority of gates in its fan-in, the signal toggling is drastically reduced. The reduction in the toggling of signals in the output of a gate leaves it with more propensity to be used as trigger inputs of HTs. Therefore, the count of gates in fan-in that shares the same signal likelihood can serve as an effective HT feature, and such count is labelled *uni_fanin*. The fan-in uniformity set for a target gate X is as follows:

$$FIUn_X = \{(X, Y_j) \in Lk_0 \times Lk_0 \cup Lk_1 \times Lk_1, \\ \forall Y_j \in \text{fan-in of } X \text{ up to level } n\}$$

uni_fanin is the cardinality of the set $FIUn$. This feature is evaluated up to a maximum level difference of 5 levels from the target gate.

4. *uni_fanout*: Similar to *uni_fanin*, *uni_fanout* contains the number of gates with similar likelihood of the target gate in its fan-out. For target gate X, its fan-out uniformity set is evaluated as:

$$FOUn_X = \{(X, Y_j) \in Lk_0 \times Lk_0 \cup Lk_1 \times Lk_1, \\ \forall Y_j \in \text{fan-out of } X \text{ up to level } n\}$$

uni_fanout is the cardinality of the set $FOUn$. The measurement is evaluated up to 5 levels from the target net.

3.2 Normalisation of Features

The features of the HTs exhibits significant class imbalance within the extracted feature super-set. One of the major contributing factor is the small footprint the HT nets tend to leave with regards to their numbers. Far lower HT net count compared to the number of total nets in the host IC leads to such imbalance. However, accurate analysis of the data by the supervised algorithm requires a normally distributed dataset. Thus, box-cox transformations [18] can be used to normalise a dataset with severe class imbalance. Every extracted feature has the propensity to be left or right skewed to various degree. Box-cox transformations have thus been utilised with varied lambda values to distribute the data for each feature normally.

Thus, various gate-level features have been extracted from ICs. The extracted features pertain to the covert and malicious nature of the HTs and have been normalised suitably for accurate training of a supervised model.

4 Feature Importance Based Feature Selection

Training the supervised model on 85 features is a very computationally intensive prospect. Many of the features tend to exhibit correlation amongst each other. This makes choosing of all features to train the model a superfluous option. There arises a need to select a subset of features from the extracted superset. This is done keeping in mind that the effectiveness of the model is not hampered. Therefore, careful choice in feature selection process needs to be conducted that satisfies all the requirements. Feature importance values of XGBoost along with variance threshold measure helps to do just that.

4.1 Feature Importance

The feature importance of XGBoost [3] model is a value assigned to each feature by the model. XGBoost algorithm involves creation of multiple gradient boosted decision trees. Each tree is constructed by introducing decision splits by a particular feature. A feature is considered to be more impactful to the algorithm if the feature is used to create such decision splits within the trees more number of times than the other features. The importance value of a particular feature is evaluated by the number of times it is used to split the dataset across all the trees. Higher the number of times the feature is used to create a split, the more impact it has on the decision-making process of the classifier. Increased impact on the decision-making leads to increased value of its importance in the algorithm. The value generated by the model signifies the impact of a feature during the construction of the model. Higher the impact of a particular feature in the construction of the model, higher is the value of its importance. Such importance value is utilised to select one feature amongst the multi-level features extracted from the gate-level netlist. Features that have been extracted up to multiple levels, for example *ff_fanin*, tend to exhibit maximum correlation amongst the different levels. Feature importance values are used to select the level at which that feature has the maximum importance value.

The feature importance measure is utilised to select the logic depth of the neighbourhood cone for level specific feature groups that exhibits highest importance value for that feature. For a level specific feature group X and the logic level y of the neighbourhood cone that exhibits highest feature importance in the group, the selected feature is denoted as *X_y*. For example, for the *div_fanout* level specific feature group, the importance of its features *div_fanout*_1, *div_fanout*_2, *div_fanout*_3, *div_fanout*_4 and *div_fanout*_5 are 0.02152, 0.57268, 0.09958, 0.14090 and 0.58237 respectively. Feature *div_fanout*_5 is selected from the *div_fanout* level specific feature group since it exhibits the highest importance value in the XGBoost algorithm for HT detection. Thus, the selected features from the various level specific feature groups are: *const_fanin*_2, *div_fanout*_5, *uni_fanout*_1, *ff_fanout*_4, *mux_fanout*_3, *ff_fanin*_4, *const_fanout*_5, *uni_fanin*_3, *div_fanin*_4, *mux_fanin*_3, *net_fanin*_1, *loop_fanout*_4, *loop_fanin*_2. This process whittles the 85 extracted features to 33 features, which are then further reduced by variance threshold.

4.1.1 Variance Threshold

The measure of variance in a feature is an indicator of how much information it holds. Higher variance of a feature would mean it exhibits higher entropy. Features with higher variance tends to have a lot more impact on the accuracy of a model, compared to the ones with lower variance. In order to whittle the feature list down to the features that contributes the most information, variance threshold is used. The mean of variance of all features have been collected, and this mean is considered as the threshold. Features that have variance values lower than this threshold are rejected. This process results in consideration of a total of 15 features from the 33 features.

The selected features are *min_dff_fanin*, *ld*, *so*, *min_dif_pi*, *ff_fanin*_4, *po*, *cc*0, *co*, *sc*0, *net_fanin*_1, *cc*1, *min_dff_fanout*, *loop_fanout*_4, *sc*1 and *loop_fanin*_2. The correlogram of the selected features using variance threshold is shown in Fig. 1. The value in each cell of the figure represents the correlation coefficient of the two features. The correlation coefficient $cr_{x,y}$ between feature x and feature y in dataset of size N is calculated as:

$$cr_{x,y} = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2 \sum_{i=1}^{N}(y_i - \bar{y})^2}}$$

where $\bar{x}$ and $\bar{y}$ are averages of dataset x and y respectively. The feature pairs with light hue represent negative correlation, while the feature pairs with dark hue represent positive correlation. The feature pairs with medium hue represent lack of correlation between the features. The diagonal matrix always has correlation coefficient value of 1, since each of its cells includes correlation calculation with the same features. Feature pairs *co* and *so*, and *cc*1 and *sc*1 are highlighted to exhibit the highest degree of correlation with value 0.95. However, the feature subset in Fig. 1 depicts features with high entropy, despite the existence of significant correlation.

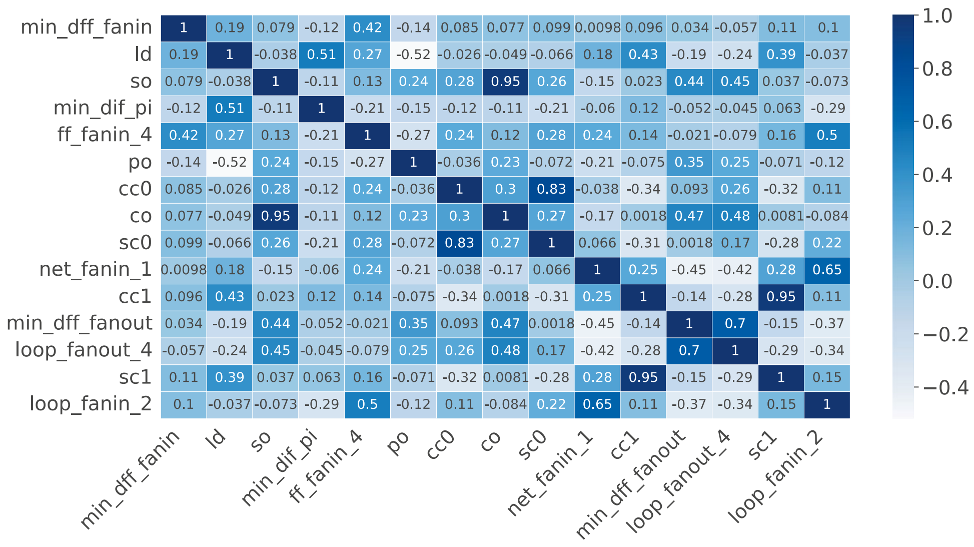
4.1.2 Pearson's Correlation

Pearson's correlation is used in [4] to select features with low correlation from the extracted feature super-set. To compare the effectiveness of high entropy within features obtained from variance threshold over features with low correlation, a separate low correlation feature subset is obtained. The Pearson's Correlation [1] is a measure that shows the degree of associations between the features. Higher correlation values indicate dependency between the features. When dependency is detected amongst two or more features, that indicates superfluousness in the feature set. Therefore, Pearson's correlation coefficient is used to deduce the highly correlated features from the less correlated ones.

The modular average of the coefficients of all the features is evaluated. The features with coefficient values lesser than the calculated mean value is chosen to create Pearson's correlation feature set. A total of 19 features are chosen using this method. The correlogram of the selected feature sets is depicted in Fig. 2 It is observed that the highest correlation occurs between the features *min_dif_po* and *min_dff_fanout* with value 0.76. Thus, selection of features with Pearson's correlation yields feature subset with relatively low correlation amongst the features, even when the number of features selected is relatively high.

Thus, feature importance values of XGBoost classifier has been used to reduce the number of level-specific features

Fig. 1 Correlogram of features selected using Feature Importance and Variance Threshold



from the extracted feature set. The feature dimension is further reduced through utilisation of variance threshold to extract a feature subset that exhibits high entropy. The impact of high entropy features with low correlation features on the effectiveness of the models is afterwards compared during cross-validation.

5 Machine Learning Model

The extracted feature sets need to be analysed through appropriate algorithm in order to create a model that can effectively detect the presence of HT. The proposed methodology of extracting the appropriate threshold of probabilistic classifier trained with variance threshold features for HT nets detection is depicted in Fig. 3. Relevant gate-level HT features are extracted from the ICs. Hyperparameters of the XGBoost model are obtained by using randomised search over the XGBoost algorithm with the extracted features. Feature importance values are evaluated for each level specific features, and is used to choose the most effective feature for a level specific feature group. Variance threshold is used to find a feature subset with high entropy from the selected level specific and non-level specific features. Supervised probabilistic XGBoost classifier is trained with the selected feature subset with relevant hyperparameters. Receiver operating curve (ROC-AUC) curve analysis and precision-recall (PR) curve analysis is performed to obtain the respective thresholds for the classifier. The performance of the model over the thresholds is analysed to choose the most effective threshold analysis technique.

The chosen algorithm is required to operate effectively in the presence of high class imbalance within the datasets. Decision tree based supervised models such as XGBoost is utilised for this very purpose. XGBoost [3] is an optimized

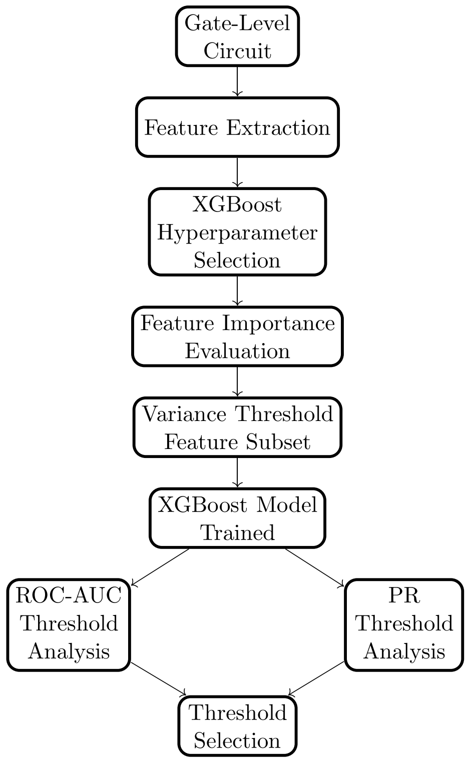

Gate-Level Circuit
Feature Extraction
XGBoost Hyperparameter Selection
Feature Importance Evaluation
Variance Threshold Feature Subset
XGBoost Model Trained
ROC-AUC Threshold Analysis
PR Threshold Analysis
Threshold Selection

Fig. 3 Threshold analysis procedure using variance threshold feature subset for HT detection

distributed gradient boosting algorithm. The algorithm exhibits increased performance and speed in tree-based (sequential decision trees) ML algorithms. It is known to be highly efficient, flexible and portable, where boosting plays a key part. Boosting is an ensemble technique where the training errors made by existing models are minimized by combining weak models. Models are added repeatedly as long as no noticeable improvements are observed.

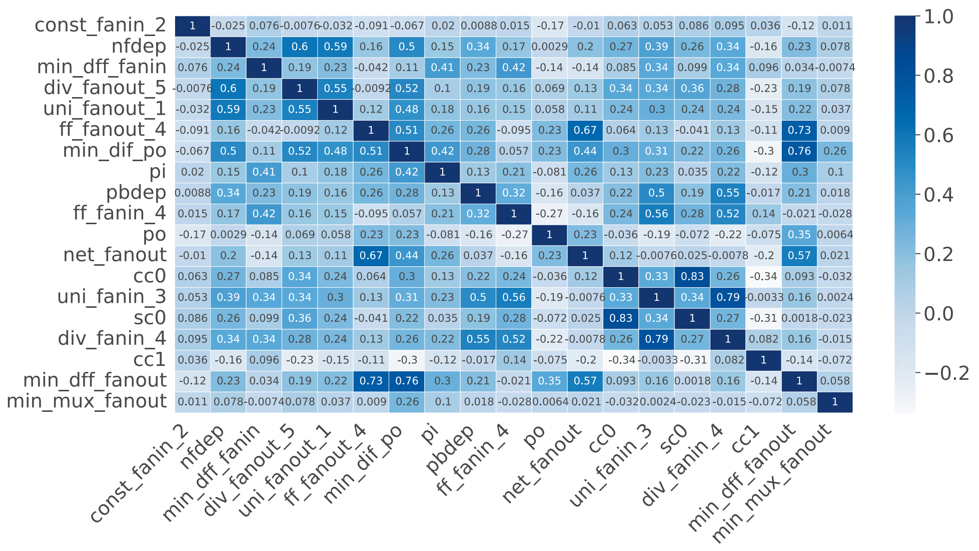


Fig. 2 Correlogram of features selected using Feature Importance and Pearson's Correlation

In the Gradient Boosting algorithm, new models are developed that predict the remnants of previous models. The models are then combined together to perform the final prediction. Loss function is minimized using the gradient descent algorithm when adding new models. Trees are built in a parallel fashion in XGBoost, unlike Gradient Boosting which follows the sequential building of trees. The algorithm follows a level-wise strategy, where it scans across gradient values. Whenever there is a split in the training set it uses these partial sums to gauge the status of splits. The scalability of the XGBoost algorithm in all scenarios has been its most important advantage. The XGBoost provides a mechanism to handle the class imbalance problem, without creating any synthetic data. The errors made by the model during training on the minority class can be scaled to a certain value. This prevents the errors of the minority to go undetected thereby leading the model to over-correct them. The overall performance of the model will now enhance while making predictions on the minority class.

5.1 Probabilistic Classifier

A classification model (or classifier) categorizes instances into one or more sets of classes. There are two types of outputs a classifier model can produce: discrete outputs, and continuous outputs. Discrete classifiers produce discrete output values, and the instance is mapped directly to a discrete class label. In contrast, classifier models with a continuous output produce a probability estimate (score). The estimate denotes how likely an instance is to be classified to a particular class/label. Such classifiers are called Probabilistic Classifiers. Probability estimates can only be calculated when the estimator can make probabilistic predictions. The estimations may appear only after fitting the model to training data. Probability estimates are evaluated by its threshold calculation through analysis of receiver operating curve (ROC) or precision-recall curve (PR). Comparison of results obtained using the two different threshold choices are conducted. The threshold that yields the results with higher accuracy and true positive rate is considered to be the more effective amongst the two.

5.2 Model Development

The HT infected ICs suffer from a class imbalance problem where the number of Trojan-infected nets is much less than the number of normal nets. Such an imbalance performs a very biased training, which in turn leads to overfitting. Hyperparameters of a supervised learning model are parameters that influence the learning process of the model. The Python implementation of XGBoost gives a hyperparameter designed to track the behaviour of the set of rules for imbalanced classification problems. This is the *scale_pos_weight* hyperparameter. It scales errors made by the model during training on the minority class, leading the model to over-correct them and helping the model obtain higher overall performance while making predictions about the minority class. This parameter is set as the ratio between the normal nets and Trojan-infected nets.

Careful tuning of the model is required in order for it to perform with high efficiency. XGBoost has a large number of hyperparameters which makes the tuning very exhausting. Tuning can be done in a grid or randomised search. Grid search works well when there is less number of hyperparameters, and each hyperparameter has about the same magnitude impact on the validation score. Randomised search is a better option when the magnitude impacts of hyperparameters are imbalanced. This paper uses the randomised search. Model performance is evaluated with respect to the accuracy score during the selection of the model. Various hyperparameters obtained from randomised search process in XGBoost model trained over gate-level datasets of ICs infected with HT are showcased in Table 2.

The detection of Trojan nets is carried out using the leave-one-out cross-validation method from a set of HT infected netlists. Every time, one of the circuit infected with HT is considered for testing, while the rest of the circuits are used for the model training. Instead of classifying a net as a Trojan or normal net, the probabilistic classifier model produces a continuous output. This continuous output is an estimation of the class membership for the instance. In other words, it represents how likely a net belongs to the Trojan class.

Thus, an optimised supervised probabilistic classifier has been chosen to train the gate-level dataset that addresses the class imbalance issue of HT detection problem. Randomised search has been utilised to select the most effective hyperparameters for the probabilistic classifier model.

Actual classifications are then produced using a threshold. If the probability is more than the threshold, the net is classified as Trojan infected. The problem comes down to finding an optimal threshold for each test instance and finally producing actual classifications. For the proposed work, two techniques have been used for the best threshold calculations.

Table 2 Selected XGBoost Hyperparameters through Randomised Search Process

Hyperparameter	Value
colsample_bytree	0.3
gamma	0.2
learning_rate	0.3
max_depth	12
min_child_weight	1
n_estimators	100
scale_pos_weight	4

5.2.1 Receiver Operating Curve (ROC)

The Receiver Operating Curve plots the True Positive Rate (TPR) vs False Positive Rate (FPR). For each test circuit, the ROC curve is plotted. Any point in the graph represents the performance of the classifier at some threshold value in the range [0, 1]. The cut-off threshold for best classifications can be determined by the Youden index. Youden's index is defined as ($sensitivity + [100\% - specificity]$). Briefly speaking, the point on the ROC curve at which Youden's index defined as is maximal, is considered to be the optimal cut-off threshold value. By using these optimal cut-off values, the performance metrics are calculated.

5.2.2 Precision Recall Curve (PR-Curve)

The precision-recall (PR) curve is plotted by calculating the precision against the recall for a probabilistic classifier at different thresholds. The curve showcases the trade-off between the two parameters. It is a convenient metric of prediction when the data suffers from a large class imbalance. When a system has high recall but low precision, it returns many positive results. However, most of the predicted labels end up as incorrect. Conversely, a system with high precision but low recall returns less number of positive results where most of its predicted labels are correctly predicted. A combination of precision and recall into a single performance metric is brought by the F1 score. It is defined as the harmonic mean between Precision and Recall. The probability threshold is chosen at the point in the PR curve that exhibits the highest F1 score.

Two different thresholds have been determined by using both the ROC-AUC curve and PR curve analysis for every IC separately for a model trained with certain features. The threshold extraction procedure that yields the most consistent and effective results for HT detection is chosen as the more viable threshold determination procedure.

The performance of a trained probabilistic classifier model with a predetermined threshold is measured through numerous metrics. True positives (TP) are the number of nets correctly identified as HT nets by the trained model. True negatives (TN) are the number of nets correctly identified as normal nets by the trained model. False positives (FP) are the number of normal nets incorrectly labelled as HT nets. False negatives (FN) are the number of HT nets incorrectly labelled as normal nets in the tested IC. True positive rate (TPR), or recall is the proportion of correct HT net detections with respect to total HT nets in the IC and is formulated as:

$$TPR = \frac{TP}{TP + FN}$$

True negative rate (TNR) is the proportion of correct normal net evaluations with respect to total normal nets in the tested IC and is formulated as:

$$TNR = \frac{TN}{TN + FP}$$

Accuracy is the ratio of the correct predictions to total predictions, and is formulated as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision is the measure of correctness of HT net prediction by the model, and is formulated as:

$$\text{Precision} = \frac{TP}{TP + FP}$$

F-measure combines the precision and recall metrics to create a harmonic mean of the two, and is formulated as:

$$\text{F-measure} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Analysis of these metrics provide a detailed perspective of the efficiency of the trained model and selected threshold for HT detection.

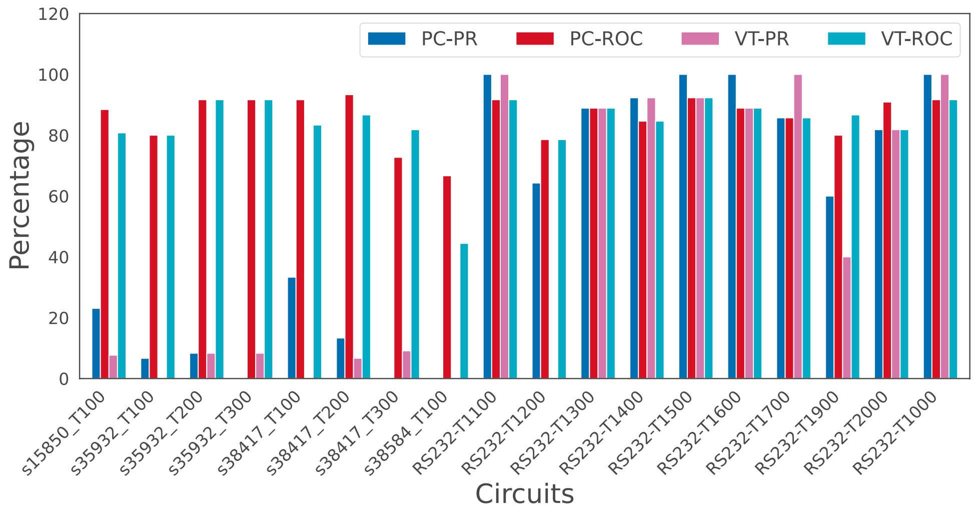
6 Experimental Results

The supervised model has been trained using the feature sets and threshold measures and cross-validation and test results have been recorded. The experiments have been conducted in a machine with Intel i5-8250U 8-core processor with maximum clock speed of 3.4 GHz, 16 GB RAM, NVIDIA GTX 1050Ti GPU, running Ubuntu 22.04.2 OS. Gate-level netlists infected with HTs from Trust-Hub site [19] have been utilised as the source of datasets. The program for dataset extraction has been written in C-language, and model training, cross-validation, and testing have been conducted using Python.

6.1 Feature Set and Threshold Measure Evaluation

Various threshold selection methods have been used for each IC under test to select the most effective threshold selection procedure. Feature importance measures have been used to select the most influential level for each level specific features. Variance threshold has been used over the selected level specific features along with non-level specific features to generate feature subset *VT*. The selected feature set exhibit high entropy and high correlation between the features. In contrast, Pearson's Correlation

Fig. 4 Comparison of TPR between the different models



procedure in [4] generated feature subset with low correlation amongst the features. Therefore, a separate subset of features (*PC*) has been extracted by using the Pearson's Correlation measure over the extracted features. This is done to highlight the effectiveness of feature set with high entropy and high correlation to train the model, as opposed to feature set with low correlation. Threshold measures for each IC under test have been extracted using the ROC-AUC curve analysis and PR-curve analysis over the model trained with *VT* feature subset. The threshold extraction process is repeated with model trained with *PC* feature subset. Therefore, 4 different sets of results are extracted from these parameters with varying performance. The models are used to evaluate cross-validation results conducted using the leave-one-out method. Figure 4 compares the TPR of the cross-validation result set. Figure 5 compares the TNR of the cross-validation result set.

Figure 4 compares the TPR results obtained from:

1. model trained with feature set obtained through feature importance with Pearson's correlation and probability threshold obtained through PR curve (*PC-PR*),
2. model trained with feature set obtained through feature importance with Pearson's correlation and probability threshold obtained through ROC (*PC-ROC*),
3. model trained with feature set obtained through feature importance with variance threshold and probability threshold obtained through PR curve (*VT-PR*), and
4. model trained with feature set obtained through feature importance with variance threshold and probability threshold obtained through ROC (*VT-ROC*).

It is revealed in Fig. 4 that threshold measure obtained using ROC-AUC curve fares much better with regards to TPR than the threshold measure obtained using the PR curve. The higher TPR value if observed for both feature selection processes. The difference in performance is far

Fig. 5 Comparison of TNR between the different models

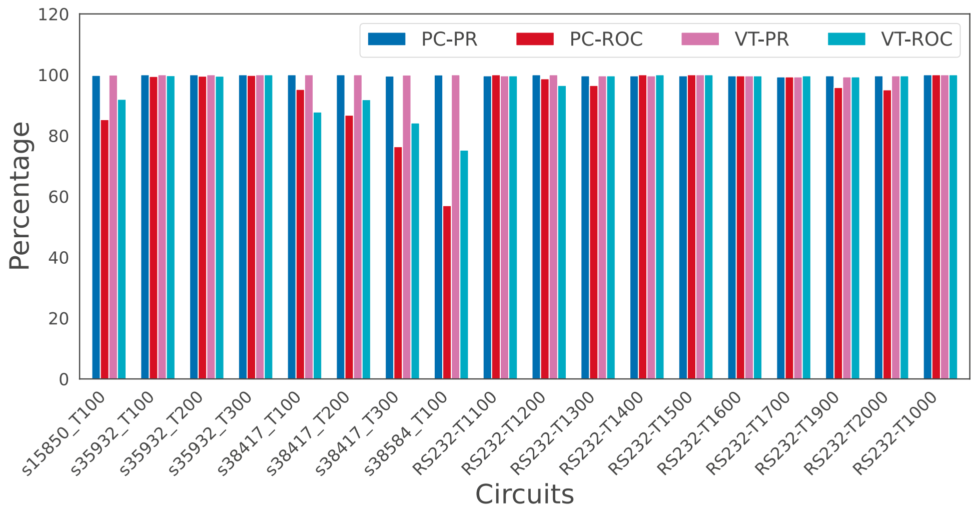


Table 3 Mean and variance evaluation of classifiers with selected thresholds

		PC-PR	PC-ROC	VT-PR	VT-ROC
TPR	**Mean**	53.21	86.08	45.80	83.95
	Variance	1715.90	57.65	1983.15	117.68
TNR	**Mean**	99.78	93.56	99.81	95.79
	Variance	0.05	125.90	0.06	49.50

more stark in the ISCAS’89 [2] ICs. The threshold measure obtained using PR curve analysis exhibits far less TPR in the ISCAS’89 ICs, especially with s38584-T100 showcasing 0 true positive. In contrast, the threshold measure obtained using ROC-AUC curve analysis exhibits fairly high TPR across all the tested ICs for both feature sets. Therefore, the ROC-AUC proves to be the better threshold measure to be chosen for test evaluation than the PR curve. Similar to Fig. 4, Fig. 5 showcases the TNR of the cross-validated ICs. It is observed that the model trained with the feature set extracted using VT has higher TNR compared to the model using PC feature set. The improvement becomes more apparent when the threshold evaluations of ROC is solely considered, since that is the threshold measure that yields higher TPR.

Statistical analysis of the extracted results have been depicted in Table 3. The mean of TPR for all circuits are observed to be considerably higher with threshold selection procedure using ROC-AUC curve. Large variance in TPR is also observed in threshold measure using PR-curve analysis compared to ROC-AUC curve analysis. Therefore, it is inferred that threshold selection procedure using ROC-AUC curve analysis is best for extracting results with high TPR. Impact of different feature selection procedures is also observed in the performance measures. With the ROC-AUC threshold selection procedure, it is observed that the mean of TNR for model trained with *PC* is comparatively similar to the model trained with *VT*. However, the variance in TNR is observed to be higher in the model trained with *PC* feature subset as opposed to the model trained with *VT* feature subsets. Low variance in TNR with high mean value is an indicator of a more consistent and effective performance with the *VT* features. High TNR along with high TPR is an indication of a better performing model. Therefore, feature sets using the variance threshold method along with threshold determination using ROC-AUC is considered as the best choice for testing.

Feature importance plays a big part in the efficiency of the trained model. In order to compare the effect of importance evaluation of the features, a feature subset is obtained by not reducing the level specific group using feature importance. The model is trained with features selected solely using variance threshold and probability threshold evaluated using ROC. The feature set selected using solely variance threshold contained a total of 23 features, compared to the 15 features selected with importance evaluation. The features selected by using just the variance threshold analysis yields the selection of features net_fanin_1, net_fanin_2, net_fanin_3, net_fanin_4, net_fanin_5, ff_fanin_3, ff_fanin_4, ff_fanin_5, po, $loop\_fanin\_2$, $loop\_fanin\_3$, $loop\_fanin\_4$, $loop\_fanin\_5$, $loop\_fanout\_4$, $loop\_fanout\_5$, ld, min_dff_fanin, min_dff_fanout, min_dif_pi, $cc0$, $cc1$, co, $sc0$, and $sc1$. It is to be observed that the selected feature subset exhibit multiple features that belong to the same level specific feature group. Selection of features from the same level specific feature group reduces the amount of useful information contributed by the features, thus degrading the performance of the model. Results are extracted from the trained model and they are compared with the results of the model that included feature importance evaluation.

Figure 6 showcases the accuracy comparison between the two models. It is observed that the accuracy measure

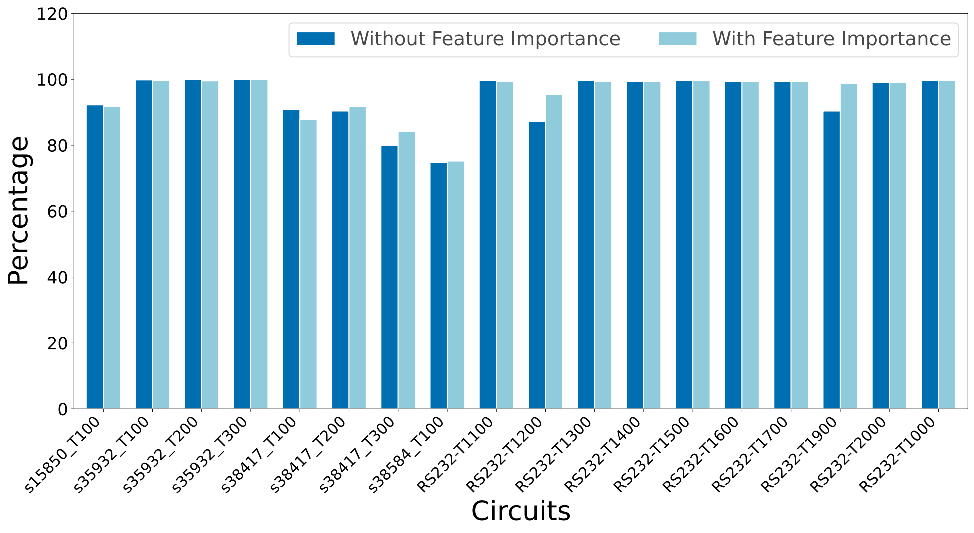


Fig. 6 Accuracy comparison between models obtained with and without feature importance

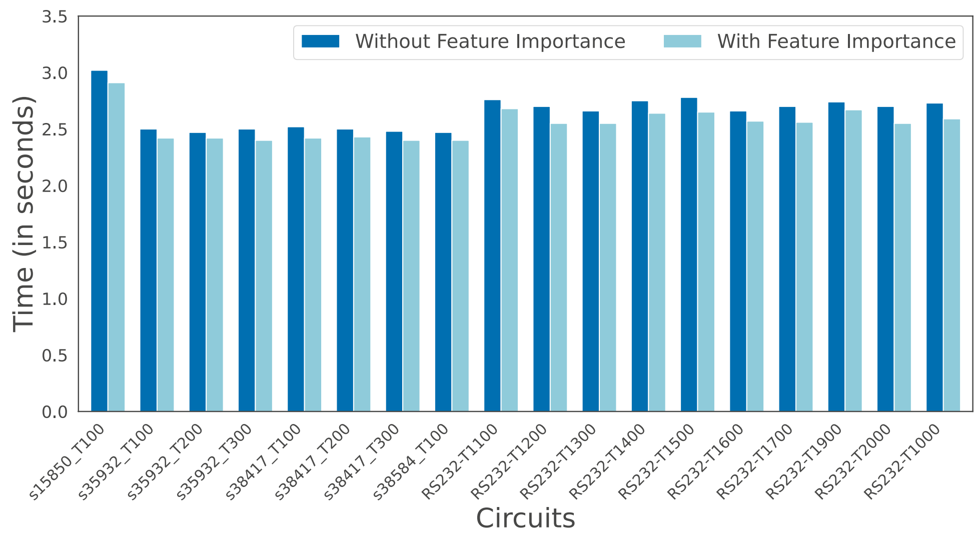


Fig. 7 Comparison of time elapsed (in seconds) for each test

remain very similar for most circuits. Noticeable improvements in accuracy is observed in RS232-T1200 and RS232-T1900 circuits in the model that used the importance values to select its features. The comparison of mean and variance of accuracy measures of the models trained by variance threshold without feature importance features and variance threshold with feature importance features have been depicted in Table 4. It is observed in the table that the mean of the accuracy is higher in the model that relies on feature importance than the model that does not involve feature importance for feature selection. It is also observed that the model with feature importance evaluation exhibits lower variance, and in turn, higher consistency in accuracy compared to without feature importance. From the results it can thus be concluded that merely relying on entropy of the features is not sufficient for effective feature selection. The importance of features play a valuable role in conjunction with variance threshold analysis to select relevant features and ensure accurate and effective detection of HT nets. The model can be improved in terms of its efficiency when features are selected using the importance values computed through XGBoost.

Figure 7 showcases the performance measure of the two models while conducting the classification process. Lower feature count in the trained model is helpful towards increasing the performance of the model during classification. The exclusion of importance values while selecting features leads to a larger feature set. The larger size of the feature set contributed to the longer time that the model had taken to perform the classification process, as observed in Fig. 7.

Table 4 Comparative analysis of impact on accuracy in importance based feature selection procedure over non-importance based

	Without Feature Importance	**With Feature Importance**
Mean	94.50	95.49
Variance	58.22	47.77

The HTs exhibit diverse functional and structural properties depending on their nature and purpose. For example, sequential HTs involve the usage of flip-flops in their trigger nets that are in close proximity to nets exhibiting rare signal switching. Level specific features aid in localised analysis of such various functional and structural properties of HTs. The limit of 5 logical depth in neighbourhood cones of the target net serve as an efficient range to gather localised HT features. Higher logical depth of neighbourhood cone leads to the features to be increasingly unrelated to the target net. Unrelated feature evaluation leads to flagging more number of normal nets as HT nets that results in decreased accuracy in HT net classification. To showcase the importance of limiting the range of the neighbourhood cone logical depth from 1 to 5, the HT classification process has been conducted by increasing the range up to 6. Table 5 depicts the statistical comparison of accuracy in results obtained from Trust-Hub HT detection process. It is observed in the table that the model trained with level specific features up to logic level 6 has lower mean and higher variance in accuracy compared to model trained with level specific features up to logic level 5. Lower mean of accuracy implies that considering level specific features up to 6 levels leads to more erroneous flagging of normal nets as HT nets. Higher value of variance also insinuates decreased consistency in accurately identifying HT nets within the host IC.

6.2 Results and Comparison

Table 6 showcases the detailed cross-validation results of various ISCAS'89 and RS Trust-Hub circuits. It is observed that

Table 5 Accuracy comparison of model trained over level specific features extracted up to 5 logic levels against features extracted up to 6 logic levels in neighbourhood cone of target nets

	5 Logic Levels	**6 Logic Levels**
Mean	95.49	94.16
Variance	47.77	58.27

the threshold value of the ROC-AUC curve tends to remain towards 0, with only a handful of ICs having high threshold determination. The TNR and accuracy is also observed to be consistently high across all ICs. TPR values are observed to be fairly high, with the RS circuits performing better than the ISCAS'89 circuits. s35932-T300, RS232-T1400, RS232-T1500 and RS232-T1000 have a TNR of 100%, which means all normal nets have been detected correctly by the trained model. 100% TNR value of the ICs leads to the precision scores to be 100%. It is observed that the F-measure is also 100% for these ICs because of the high TPR paired with perfect TNR score. High precision and F-measure scores indicate high degree of correctness in predictions. The minimum TPR is exhibited by s38584_T100 with a TPR value of 44.44%. The IC also exhibits a minimum TNR of 75.22%, making it the lowest performing tested Trust-Hub IC. However, a greater than 0% value of TPR indicates detection of at least one HT net for every tested IC. s35932_T300 is observed to be the most accurately predicted Trust-Hub IC with an accuracy measure of 99.98%. It is to be noted that the TNR and accuracy measure is consistently high across all the tested ICs with an average value of 95.79% and 95.49%, respectively. Consistently high TNR and accuracy measure indicates the effectiveness of the selected threshold for HT detection for all ICs. Overall, the best performance is seen with s35932-T300, RS232-T1400, RS232-T1500 and RS232-T1000 circuits which all have 100% F-measure and precision scores. The results of the circuits that exhibit the highest performance by the model are emboldened in the table. The results obtained are compared with previous supervised works in literature and the comparison is depicted in Table 7.

Random forest model is used in [10], neural networks are utilised to train the model in [11], and supervised model using SVM [9] have been used to extract results from Trust-Hub circuits. [4] utilised XGBoost with feature set obtained using Pearson's correlation. It is observed in Table 7 that both the TPR and TNR values of the proposed model have improved over the previous works in varying degree. Unlike the previous works, the TPR and TNR values are also observed to be much more consistent as well. Such consistency speaks to the reliability of the proposed model compared to the previous works.

6.3 Test Results of Custom HT

Custom HTs that are designed to exhibit low toggle within their nets have been inserted in ISCAS'89 circuits s386 and

Table 6 Cross validation results using variance threshold feature sets and ROC-AUC threshold

CircuitName	**Threshold**	**TNR%**	**TPR%**	**Accuracy%**	**F-measure%**	**Precision%**
s15850_T100	0.00	91.93	80.77	91.81	18.19	10.10
s35932_T100	0.00	99.71	80.00	99.66	57.10	40.00
s35932_T200	0.00	99.54	91.67	99.52	43.09	27.50
s35932_T300	0.02	**100.00**	91.67	**99.98**	**100.00**	**100.00**
s38417_T100	0.00	87.75	83.33	87.74	2.80	1.42
s38417_T200	0.00	91.82	86.67	91.80	5.30	2.73
s38417_T300	0.00	84.15	81.82	84.15	1.95	0.99
s38584_T100	0.00	75.22	44.44	75.19	0.45	0.23
RS232-T1100	0.95	99.65	91.67	99.33	95.49	91.67
RS232-T1200	0.00	96.48	78.57	95.44	72.36	57.89
RS232-T1300	0.28	99.65	88.89	99.32	93.96	88.89
RS232-T1400	1.00	**100.00**	84.62	99.32	**100.00**	**100.00**
RS232-T1500	0.44	**100.00**	92.31	99.66	**100.00**	**100.00**
RS232-T1600	0.55	99.64	88.89	99.31	93.96	88.89
RS232-T1700	1.00	99.65	85.71	99.31	92.16	85.71
RS232-T1900	0.00	99.30	86.67	98.67	92.55	86.67
RS232-T2000	0.00	99.65	81.82	98.98	94.58	90.00
RS232-T1000	1.00	**100.00**	91.67	99.66	**100.00**	**100.00**
Average		95.79	83.95	95.49	64.66	59.59

Certain results in Table 6 have been emboldened to highlight the performance metrics of the circuits that exhibit significantly higher performance than the rest of the circuits

Table 7 Comparison of obtained results with previous supervised works

	True Positive Rate					True Negative Rate				
Circuit	[10]	[11]	[9]	[4]	VT-ROC	[10]	[11]	[9]	[4]	VT-ROC
RS232-T1000	100	100	53	93.18	91.67	98.90	24	31	95.54	100
RS232-T1100	50	78	58	95.35	91.67	98.20	25	27	97.05	99.65
RS232-T1200	88.20	91	80	88.37	78.57	100	55	26	96.70	96.48
RS232-T1300	100	86	89	96.77	88.89	100	65	26	97.47	99.65
RS232-T1400	97.80	100	83	95.92	84.62	100	15	22	96.58	100
RS232-T1500	94.90	82	83	95.74	92.31	99.60	47	24	97.03	100
RS232-T1600	93.10	97	89	90.62	88.89	99	28	26	97.83	99.64
s35932_T100	73	80	93	79.41	80	100	99	60	91.91	99.71
s35932_T200	8.30	67	100	65.85	91.67	100	88	59	93.90	99.54
s35932_T300	81.10	100	27	88.14	91.67	100	97	58	98.28	100
s38417_T100	33.30	83	100	71.43	83.33	100	98	76	78.76	87.75
s38417_T200	46.70	93	73	62.86	86.67	100	74	76	84.44	91.82
s38417_T300	75	100	100	87.10	81.82	100	94	72	63.96	84.15

s38584. The various HTs inserted within the host ICs are shown in Fig. 8. The insertion of the HTs have been conducted so as to make them be a part of the IC that has minimal toggling within their nets. The trigger inputs (*Tin*) are nets of the host IC that exhibit low toggle in their signals during normal functioning of the IC. Upon full activation, the HTs inject the malicious signal in the host IC through the payload signal (*P*). s386-HT1 is obtained using HT1 nets in s386 circuit, s386-HT2 is obtained using HT2 nets in s386, s386-HT3 and s38584-HT3 is obtained using HT3 nets in s386 and s38584 circuits respectively. The supervised model trained with feature importance and variance threshold features are used to examine the datasets of the ISCAS'89 ICs infected by the custom HTs. The results of the analysis is depicted in Table 8.

It is observed in Table 8 that majority of the HT nets have been detected by the trained model. In addition to consistent detection of infected nets, the TNR, and thus the accuracy of s38584-HT3 is also observed to be relatively high. Therefore, it is observed that the trained model is sufficiently capable of detecting covert nets within the IC even when the tested HTs exhibit different design from the training set. They are able to detect them even when they are designed to bypass standard fault testing techniques.

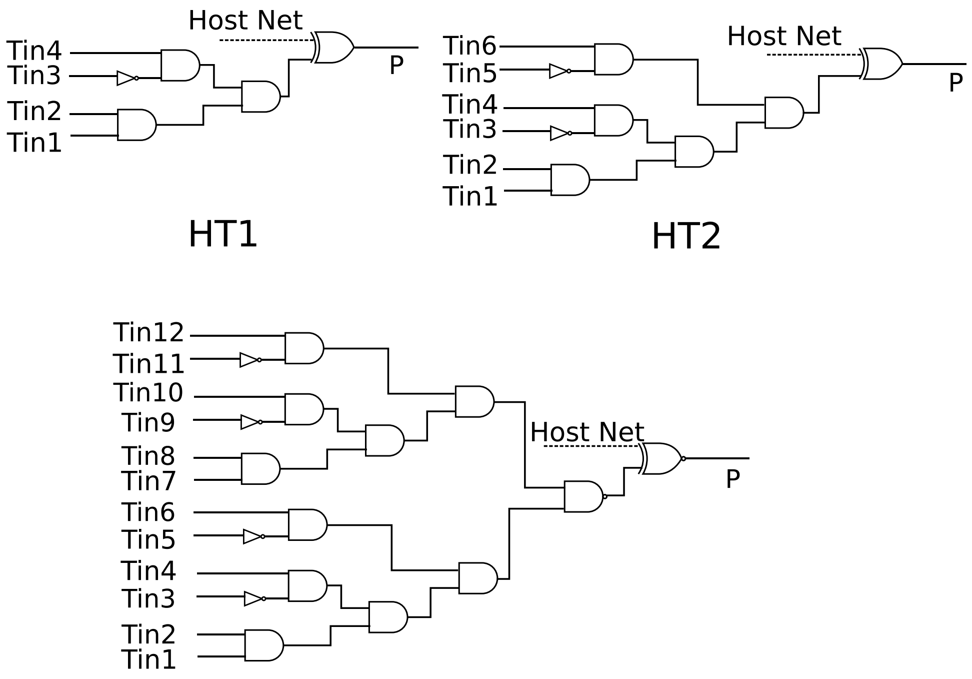


Fig. 8 Custom HTs inserted in ICs

Table 8 Test results of the ICs infected with custom HT

CircuitName	Threshold	TNR%	TPR%	Accuracy%	F-measure%	Precision%
s386-HT1	0.00	53.85	66.67	54.07	4.78	2.50
s386-HT2	0.00	51.48	80.00	52.30	8.53	4.65
s386-HT3	0.00	47.93	81.82	50.00	15.55	9.28
s38584-HT3	0.00	93.04	54.55	93.02	0.83	0.42

7 Conclusion

The covertness of the HTs tend to make them difficult to detect using standard testing procedures during IC manufacturing. This paper analyses effective threshold calculation procedure in probabilistic XGBoost classifier to facilitate their detection. Relevant HT features have been considered from gate-level netlists. The impact of the extracted features on decision tree classifier models is analysed by evaluating their importance values. Feature subset exhibiting high entropy has been obtained using the XGBoost feature importance along with variance threshold. The feature subset is used to create probabilistic classifier models using XGBoost trained with effective hyperparameters whose threshold is analysed using ROC-AUC curve and PR curve. The impact of the resultant classifier exhibits high TNR and moderate to high TPR values in the tested circuits. Effectiveness of high entropy in features over low correlation for HT detection is also compared by using Pearson's correlation to select low correlated feature subset.

Data Availability Raw data has been generated from the gate level Trust-Hub benchmark circuits available in https://trust-hub.org/#/benchmarks/chip-level-trojan. Derived data supporting the findings of this study are available from the corresponding author upon request.

Declarations

Conflicts of Interests The authors have no conflicts of interests to declare that are relevant to the content of this article.

References

1. Benesty J, Chen J, Huang Y, Cohen I (2009) Pearson correlation coefficient. In: Proc. Noise Reduction in Speech Processing. Springer Topics in Signal Processing, pp. 1–4. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-00296-0_5
2. Brglez F, Bryan D, Kozminski K (1989) Combinational profiles of sequential benchmark circuits. In: Proc. 1989 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1929–19343. https://doi.org/10.1109/ISCAS.1989.100747
3. Chen T, Guestrin C (2016) XGBoost: A scalable tree boosting system. In: Proc. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '16, pp. 785–794. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/2939672.2939785
4. Das R, Dhar T, Roy SK (2022) A threshold based hardware trojan detection technique using xgboost algorithm. In: Proc. 2022 IEEE International Test Conference India (ITC India), pp. 1–6. https://doi.org/10.1109/ITCIndia202255192.2022.9854735
5. Dhar T, Roy SK, Giri C (2021) Hardware Trojan Horse Detection through Improved Switching of Dormant Nets. ACM J Emerg Technol Comput Syst 17(3):33–13322. https://doi.org/10.1145/3439951
6. Dong C, Chen J, Guo W, Zou J (2019) A machine-learning-based hardware-Trojan detection approach for chips in the Internet of Things. Int J Distrib Sens Netw 15(12):1550147719888098
7. Dong C, Liu Y, Chen J, Liu X, Guo W, Chen Y (2020) An unsupervised detection approach for hardware trojans. IEEE Access: Practical Innovations, Open Solutions 8:158169–158183. https://doi.org/10.1109/ACCESS.2020.3001239
8. Goldstein LH, Thigpen EL (1980) SCOAP: Sandia Controllability/Observability analysis program. In: Proc. 17th Design Automation Conference, pp. 190–196. https://doi.org/10.1109/DAC.1980.1585245
9. Hasegawa K, Oya M, Yanagisawa M, Togawa N (2016) Hardware trojans classification for gate-level netlists based on machine learning. In: Proc. 2016 IEEE 22nd International Symposium on On-Line Testing and Robust System Design (IOLTS), pp. 203–206. https://doi.org/10.1109/IOLTS.2016.7604700
10. Hasegawa K, Yanagisawa M, Togawa N (2017) Trojan-feature extraction at gate-level netlists and its application to hardware-Trojan detection using random forest classifier. In: Proc. 2017 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1–4. https://doi.org/10.1109/ISCAS.2017.8050827
11. Hasegawa K, Yanagisawa M, Togawa N (2017) Hardware Trojans classification for gate-level netlists using multi-layer neural networks. In: Proc. 2017 IEEE 23rd International Symposium on On-Line Testing and Robust System Design (IOLTS), pp. 227–232. https://doi.org/10.1109/IOLTS.2017.8046227
12. Hoque T, Cruz J, Chakraborty P, Bhunia S (2018) Hardware IP Trust Validation: Learn (the Untrustworthy), and Verify. In: Proc. 2018 IEEE International Test Conference (ITC), pp. 1–10. https://doi.org/10.1109/TEST.2018.8624727
13. Huang Z, Wang Q, Chen Y, Jiang X (2020) A survey on machine learning against Hardware trojan attacks: Recent advances and challenges. IEEE Access 8:10796–10826. https://doi.org/10.1109/ACCESS.2020.2965016
14. Jacob N, Merli D, Heyszl J, Sigl G (2014) Hardware Trojans Current challenges and approaches. IET Comput Digit Tech 8(6):264–273. https://doi.org/10.1049/iet-cdt.2014.0039
15. Kok CH, Ooi CY, Moghbel M, Ismail N, Choo HS, Inoue M (2019) Classification of trojan nets based on SCOAP values using supervised learning. In: Proc. 2019 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1–5. https://doi.org/10.1109/ISCAS.2019.8702462
16. Li H, Liu Q, Zhang J, Lyu Y (2015) A survey of hardware trojan detection, diagnosis and prevention. In: Proc. 2015 14th International Conference on Computer-Aided Design and Computer Graphics (CAD/Graphics), pp. 173–180. https://doi.org/10.1109/CADGRAPHICS.2015.41

17. Rostami M, Koushanfar F, Karri R (2014) A primer on hardware security: Models, methods, and metrics. Proc IEEE 102(8):1283–1295. https://doi.org/10.1109/JPROC.2014.2335155
18. Sakia RM (1992) The box-cox transformation technique: A review. J R Stat Soc Ser A Stat Soc Series D 41(2):169–178
19. Salmani H, Tehranipoor M (2023) Trust-Hub.Org. https://trust-hub.org/#/home
20. Sharma R, Valivati NK, Sharma GK, Pattanaik M (2020) A new hardware trojan detection technique using class weighted XGBoost classifier. In: Proc. 2020 24th International Symposium on VLSI Design and Test (VDAT), pp. 1–6. https://doi.org/10.1109/VDAT50263.2020.9190603
21. Tebyanian M, Mokhtarpour A, Shafieinejad A (2021) SC-COTD: Hardware trojan detection based on sequential/combinational testability features using ensemble classifier. J Electron Test 37(4):473–487. https://doi.org/10.1007/s10836-021-05960-2
22. Yang Y, Ye J, Cao Y, Zhang J, Li X, Li H, Hu Y (2020) Survey: Hardware trojan detection for netlist. In: Proc. 2020 IEEE 29th Asian Test Symposium (ATS), pp. 1–6. https://doi.org/10.1109/ATS49688.2020.9301614

Tapobrata Dhar received his Bachelor of Technology (B.Tech.) degree in Information Technology from National Institute of Technology, Durgapur, India, in 2013. He received his Master of Engineering (M.E.) degree in Software Engineering from Jadavpur University, India, in 2017. He is currently a research scholar in the Indian Institute of Engineering, Science and Technology, Shibpur, India, pursuing doctorate degree. His main research interests include hardware security, machine learning and cryptography.

Ranit Das received his Bachelor of Technology (B. Tech.) degree from RCC Institute of Information Technology, Kolkata, India, in 2015. He received his Master of Technology (M.Tech.) in Information Technology from Indian Institute of Engineering Science and Technology (IIEST), Shibpur, India, in 2022. He is currently employed as an engineer at Samsung Research Institute, Noida, India, working with video codes in Samsung mobile devices, under Visual Solutions team.

Chandan Giri has been at Indian Institute of Engineering Science and Technology, Shibpur since 2008 as Associate Professor, Dept. of Information Technology. His current research is focused on testing and design-for-testability of integrated circuits (especially 3D and multi-core chips) and Wireless Sensor Network. His research project has included 3D multi-core IC testing. Research support is provided by the University Grant Commission, Govt. of India. C. Giri received his Bachelor of Technology (B.Tech) in Computer Science & Engineering from Calcutta University, India in 2000 and subsequently Master of Engineering (ME) in Computer Science & Engineering from Jadavpur University, Kolkata, India in 2002. He was awarded PhD degree from Dept. of Electronics and Electrical Communication Engineering of Indian Institute of Technology, Kharagpur in 2008. He also presented his research papers in several International Conferences. He is a member of IEEE and ACM.

Surajit Kumar Roy received the BSc (Hons. in Physics) from Calcutta University, India. He also received Bachelor of Technology in computer science and engineering and subsequently Master of Technology in computer science and engineering from Calcutta University, India in 2002 and 2004. He was awarded PhD degree from Indian Institute of Engineering Science and Technology (IIEST), Shibpur in 2016. Currently he is working at Indian Institute of Engineering Science and Technology, India as Associate Professor, in the department of Information Technology. His research interest includes VLSI testing, embedded Systems, hardware security.