Early Stage DRC Prediction Using Ensemble Machine Learning Algorithms
Prédiction de la DRC à un stade précoce à l'aide d'un ensemble d'algorithmes d'apprentissage machine

Riadul Islam, *Senior Member, IEEE*

***Abstract*—At leading technology nodes, the industry is facing a stiff challenge to make profitable integrated circuits (ICs). One of the primary issues is the design rule checking (DRC) violation. This research cohort with the DARPA IDEA program aims for "no-human-in-the-loop" and 24-h turnaround time to implement an IC from design specifications. In order to reduce human effort, this work introduces the ensemble random forest, gradient boosting, and Adaboost algorithms to predict DRC violations before detailed routing, which is considered the most time-consuming step in an IC design flow. In addition, this work identifies the features that critically impact DRC violations. The proposed algorithm has a 2% better F1-score compared to the existing support-vector machine (SVM) classifiers. The proposed ensemble approach has up to an area-under-the-curve–receiver operating characteristics (AUC–ROC) curve mean of 0.940 with ± 0.011 standard deviation compared to the state-of-the-art SVM classifier with an AUC–ROC curve mean of 0.854 with ± 0.01 standard deviation. The proposed ensemble approach exhibits up to 28.7% better DRC violation prediction rate compared to those using SVM algorithms on the test data. In addition, the proposed gradient boosting algorithm requires 37.5× lower average training time and 50× lower average testing time compared to the existing SVM methodologies.**

***Résumé*—Dans les nœuds technologiques de pointe, l'industrie doit relever un défi de taille pour fabriquer des circuits intégrés (CI) rentables. L'un des principaux problèmes est la vérification de la violation des règles de conception (DRC). Cette cohorte de recherche du programme IDEA de la DARPA a pour objectif d'implémenter un circuit intégré à partir des spécifications de conception sans intervention humaine dans la boucle et dans un délai de 24 heures. Afin de réduire l'effort humain, ce travail introduit les algorithmes de forêt aléatoire d'ensemble, amplification de gradient et d'Adaboost pour prédire les violations de DRC avant le routage détaillé, qui est considéré comme l'étape la plus longue dans un flux de conception de CI. En outre, ce travail identifie les caractéristiques qui ont un impact critique sur les violations de la DRC. L'algorithme proposé obtient un score F1 supérieur de 2 % à celui des classificateurs SVM (Machine à Vecteur de Support) existants. L'approche d'ensemble proposée présente une zone sous la courbe - caractéristiques de fonctionnement du récepteur (AUC-ROC) moyenne de 0,940 avec un écart type de ± 0,011 par rapport au classificateur SVM de pointe dont la moyenne de la courbe AUC-ROC est de 0,854 avec un écart type de ± 0,01. L'approche d'ensemble proposée présente un taux de prédiction de violation de la DRC jusqu'à 28,7 % supérieur à celui des algorithmes SVM sur les données de test. En outre, l'algorithme d'amplification du gradient proposé nécessite un temps d'entrainement moyen 37,5 fois inférieur et un temps de test moyen 50 fois inférieur par rapport aux méthodologies SVM existantes.**

***Index Terms*—Adaboost, design rule checking (DRC), ensemble learning, gradient boosting, physical design, random forest.**

I. Introduction

THE modern microprocessor industry started putting billions of transistors in a chip, which is a collaborative effort of industry, academia, and government to continuous technology scaling and improved electronic design automation (EDA) tools. The standard integrated circuit (IC) design flow starts from the behavioral design specification to the physical layout generation, commonly used by the design automation community. Fig. 1 shows a traditional physical design flow, starting from high-level hardware description language (HDL) and ending with GDSII generation. In recent years, however, the IC industry faces an enormous challenge in making reliable ICs [1], [2]. This is primarily due to hostile technology scaling, severe timing, and area budgets. Hence, the manufacturing cost of the ICs is increasing in the below 10-nm technology nodes [3], [4], [5]. Besides meeting timing constraints, another major issue that critically affects the performance of the EDA tool is the design rule checking (DRC) violation [2], [6], [7].

A. Motivation

The first step of physical IC synthesis flow is logic synthesis and followed by place and route, as shown in Fig. 1. The place and route comprises partitioning, floorplanning, placement,

Manuscript received 26 April 2021; revised 7 January 2022, 2 April 2022, and 10 June 2022; accepted 11 August 2022. This work was supported in part by the National Science Foundation (NSF) under Award 2138253, in part by UMBC Startup Grant, and in part by the Office of Naval Research (ONR) under Award N00014-21-1-2531.

The author is with the Department of Computer Science and Electrical Engineering, University of Maryland at Baltimore County, Baltimore, MD 21250 USA (e-mail: riaduli@umbc.edu).

Associate Editor managing this article's review: Shan Du.
Digital Object Identifier 10.1109/ICJECE.2022.3200075

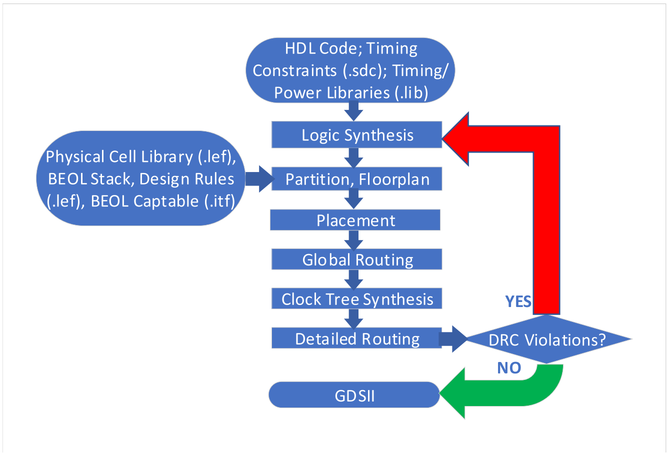

HDL Code; Timing Constraints (.sdc); Timing/ Power Libraries (.lib)
Physical Cell Library (.lef), BEOL Stack, Design Rules (.lef), BEOL Captable (.itf)
Logic Synthesis
Partition, Floorplan
Placement
Global Routing
Clock Tree Synthesis
Detailed Routing
DRC Violations?
YES
NO
GDSII

Fig. 1. Conventional physical IC synthesis flow uses high-level HDL code and design and technology libraries to generate a final GDSII layout; however, at a low-technology node, a large number of DRC violations make a design unroutable and increases design time significantly.

global routing, clock tree synthesis, and detailed routing. The final step, detailed routing, consumes more time than all other physical design flow stages. It can take days to several weeks and require considerable computing resources, depending on the design's size [2]. In a design, the EDA tools locate a significant number of DRC violations after routing is done. Fixing DRC violations require expert skills, and a designer can remove about 200 DRC violations in a day [8]. Hence, resolving a large number of DRC violations is expensive and requires a large number of person-hours. It is mostly impossible to extract a routable design in the first pass with all the careful steps and requires many iterations, as shown in the feedback path in Fig. 1. As a result, early stage routing prediction attracts great attention from the research community [2], [7], [9], [10]. The existing synthesis tools use only global routing congestion report for routability prediction, however, disputed by the research community on the effectiveness of this method [2], [10]. The EDA researchers identified some critical features that are highly correlated with DRC violations. Researchers started using machine learning algorithms to predict DRC violations by building datasets from the synthesis tools [7]. The primary idea is to convert a design into virtual grids and extract features from those grids. Once the dataset is built, the researchers identify the DRC violations' locations by applying a supervised support-vector machine (SVM) algorithm with moderate accuracy [2].

B. Primary Contributions

This article proposes an ensemble learning algorithm that inherently combines multiple learning models to subdue each model's shortcomings to generate a weighted result. It successfully predicts DRC violations using ensemble learning-based random forest [11], gradient boosting, and Adaboost classifiers. Ensemble learning models incorporate multiple learning models to come to a weighted and better prediction result. For example, ensemble random forest generates random sets of uncorrelated decision trees to the outcome to the best possible prediction. This research shows that the proposed algorithm performs comparable to or better compared to the existing methodologies in terms of precision, recall, and F1-scores. This work also introduced the area-under-the-curve–receiver operating characteristics (AUC–ROC) curve and feature importances. To be precise, the primary contributions of this article are given as follows.

1) From what the author knows, this is the first methodology to predict early stage routing using three-ensemble learning-based algorithms.
2) The proposed methodology identifies the best-suited ensemble approach for early DRC violation prediction.
3) This work identifies the features that critically influence the DRC violations.
4) It achieves the best precision, recall, and F1-scores of 99%, 99%, and 99%, respectively, which are better results than those of the existing prediction models.
5) The proposed algorithms' AUC–ROC curve exhibits the highest ratio among the true positive rate (TPR) and false positive rate (FPR) of 0.94.

C. Article's Organization

The remainder of this article is organized as follows. In Section II, it gives a brief overview of existing machine learning applications in the physical design flow. Section III presents the proposed ensemble schemes. In Section IV, the prediction accuracy and performance efficiency of the proposed algorithms compared with state-of-the-art schemes are investigated. Finally, Section V concludes with the key findings of this work.

II. Overview of Machine Learning Application in Physical Design Flow

Machine learning has widespread applications, including text sentiment classification [12], [13], [14], [15], [16], [17], natural language processing [18], [19], [20], and sarcasm identification [21], while, in recent years, researchers have shown a tremendous interest in machine learning applications in IC physical design tools [2], [5], [6], [7], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34]. Static timing analysis (STA) is a key step in the IC design flow, and machine learning algorithms can play a key role in estimating delay. The primary reason for this is that machine learning algorithms can relate the complex interactions between different design parameters, for example, wire and gate parasitics, wire length, buffers, a number of gates, and a number of primary inputs/outputs.

An incremental STA (iSTA) tool was proposed to model wire delay/slew, and an offset-based timing correlation was proposed to obtain an internal STA tool [35]. In order to model wire delay and slew, this approach uses machine learning least-squares regression (LSQR) [36] to correlate STA tool values. Another machine learning approach estimated path delay by fitting the slack delta before and after threshold swap [37]. This approach primarily tried to reduce leakage power without affecting the path delay. An artificial neural network (NN) was used, where different gate parameters were taken as input and the predicted gate delay was the output [38]. Other researchers used a multivariate regression-based machine learning technique to compute the circuit delays considering IC manufacturing-related process variation in the presence of temperature-related temporal variation [39]. This method is also useful in predicting the dynamic behavior of a circuit under various operating conditions.

In addition to STA, researchers have applied a machine learning multivariate adaptive regression spline (MARS) algorithm to predict global routing using pin density and congestion maps as design features [26]. Using this prediction model, the authors claimed 13% DRC violation reduction compared to the existing analytical model-based routing predictor. An ensemble RUSBoost algorithm was proposed by Tabrizi *et al.* [30] for DRC violations prediction. However, this approach achieved only 84% overall prediction accuracy. Another interesting approach used a supervised multivariate adaptive regression algorithm for routability or DRC violation prediction considering different design features collected after the placement stage [7]. This method achieved 79.8% prediction accuracy. Another approach, the back-end-of-line (BEOL) stack-aware routability predictor, used machine learning and data mining algorithms [2]. Besides the BEOL stack, this approach can predict core utilization and aspect ratio with a more than 85.4% accuracy rate.

Pin access point information and self-crossing net information were used by Yu *et al.* and proposed a convolutional neural network (CNN)-based pin pattern recognition (PPR) and the design feature-aware PPR (DFPPR) models. However, unlike the proposed algorithm, which considers all kinds of DRC violations, this approach only considered metal 2 related issues. Another CNN-based DRC violation prediction method, namely, RouteNet, uses features associated with macros [29]. However, this method suffers from low accuracy due to the lack of incorporation of other physical properties of layouts. Another deep CNN model used pin-related features, region-based route map, and trial routing-based features to predict DRC violations [33]. Similar to RouteNet, this approach exhibits low prediction accuracy. An NN-based ensemble approach used soft voting architecture and a principal component analysis-based subset selection scheme to predict DRC violations [34]. Unlike the proposed method, this technique utilized global routing information to build the prediction model.

III. Proposed Algorithm

This article proposes ensemble learning-based algorithms that inherently combine multiple learning models to subdue each model's shortcomings to generate a weighted result. The proposed early stage DRC violation prediction methodology is shown in Fig. 2. The proposed methodology collects features after the placement stage of the conventional physical design flow. Once features are collected, it applies different ensemble machine learning algorithms to predict the routability of design, as shown in Fig. 2.

A. Feature Selection

The proposed methodology collected data by implementing different designs and will be discussed in detail in Section IV-A. For data collection, it partitioned each design by the same size grids. Each grid consists of 45 metal 2 track × 45 metal 2 track. Then, it collected data features from each grid. This article considered the following features for building the dataset and experiments: total area, pin density, cell density, buried nets, buried pins, intersecting cells, standard cell count, standard cell area, cell utilization, intersecting nets, and intersecting pins. Now, it will discuss each feature briefly.

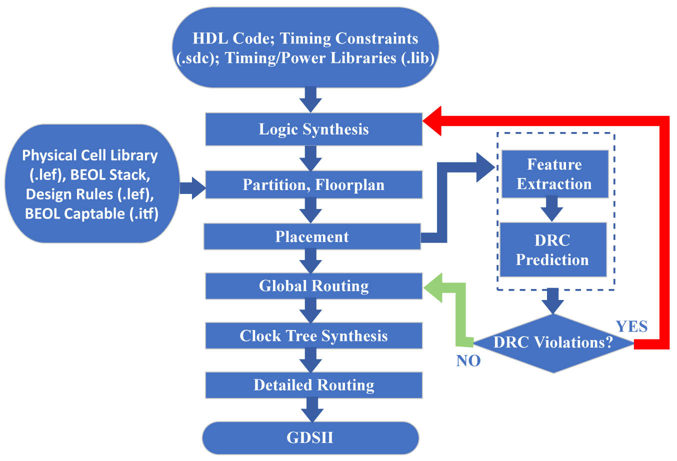


Fig. 2. Unlike existing methodologies, the proposed algorithm collects features after the placement stage and applies ensemble algorithms to predict a design's DRC violation issues.

1) *Total Area:* It is the total area of the grid.
2) *Pin Density:* It is the ratio of the number of pins in a grid region and the total area of that region.
3) *Cell Density:* It is the ratio of the number of cells in a grid region and the total area of that region.
4) *Buried Nets:* It refers to the number of nets within a grid.
5) *Buried Pins:* It indicates the number of pins within a grid.
6) *Intersecting Cells:* It indicates that the number of cells intersects between two grid networks.
7) *Std Cell Count:* It refers to the number of standard cells within a grid.
8) *Std Cell Area:* It refers to the total area occupied by standard cells within a grid.
9) *Cell Utilization:* It is the ratio of total cell area of a grid region and total area of a grid region.
10) *Intersecting Nets:* It refers to the number of nets intersect between two grids.
11) *Intersecting Pins:* It refers to the number of pins intersect between two grids.

Empirically, these features directly influence DRC violations in a design, and the experimental results in Section IV-B will provide more concrete evidence. After the features are selected, they are classified into DRC-violated and DRC-violation-free datasets. In order to identify the performance of different ensemble algorithms on DRC violations prediction, this work considers random forest, gradient boosting, and Adaboosting algorithms.

B. Random Forest Algorithm

The ensemble learning algorithm random forests or random decision trees fall in the supervised machine learning algorithm category. The random forest can be utilized for predicting both classification and regression problems [40]. The individual ensemble model uses various learning methods to reach a better predictive outcome. By creating a distinct forest of random decision trees, this algorithm provides the best possible decision.

The decision tree-based random forest algorithm use a top-down method in which the root node continuously performs binary splitting until it reaches a specific criterion. As a result, a predicted value is constructed on the inner nodes reflecting the outer nodes. For classification algorithms, a decision tree will produce an expected destination group for every new leaf nodes. However, when the random forest uses diverse training, they suffer from high variance and overfitting issues—these issues due to the use of separate training and test datasets from the same data. As a result, random forest often performs inadequately on new data and limits their usage. However, researchers surmount this issue with ensemble techniques.

The random forest uses a standard bootstrap aggregating or bagging method to form an ensemble of trees. This method builds various training sets with replacement and the likelihood of reusing data. Then, the algorithm uses each subsample to train a model. At this point, a majority-voter-type model is introduced to subdue the variance by taking the average of the results. Besides, the bagging tree uses the entire feature space for splitting the nodes. This algorithm improves the prediction accuracy by permitting the trees to develop without pruning, reducing tree-depth sizes, and lowering bias at the expense of high variance. It also compensates for the correlation issue by selecting only a subsample of the feature space during each split—a termination condition used to prune the trees and keep them away from correlation.

For generating the real dataset for modeling a decision tree algorithm, this study divides each design into equal-sized grids and spends months collecting the required feature parameters from each grid. The ensemble method uses the Gini impurity and entropy metrics to evaluate the uniformity of the splits. These metrics generally provide foreknowledge about the critical variables applied in the training of the model. The entropy computes a pseudolinear logarithmic function. As a result, it is computationally more expensive than the Gini impurity. The impurity reaches zero when all target class labels are alike. In this process, the author learned what variables played a vital role in the model predictions.

The proposed random forest DRC prediction methodology is shown in Algorithm 1. It accepts grid data from synthesized layouts as inputs and predicts whether the data (i.e., grid) are DRC violated or clean. This research processes the data and performs labeling using processData(grid_data) in Line 4. It splits the data and labels into training and test sets and fits the data into a random forest classifier model (RFmodel) in Lines 5 and 6, respectively. It performs DRC violation prediction in Line 7.

Algorithm 1 Random Forest DRC Prediction Algorithm

```
1: Input: Extracted grid data from layouts (grid_data);
2: Output: Prediction;            ▷ DRC violated or DRC clean
3:
4: $\{data, labels\} = processData(grid\_data)$
5: $train, test, trainLabels, testLabels = trainTestSplit(data, labels)$   ▷ Splitting the data and labels for training and testing
6: $RFmodel = RFClassifier(train, trainLabels)$   ▷ Training data using a random forest classsifiers
7: $predictions = RFmodel.predict(test)$   ▷ Making the predictions
```

C. Gradient Boosting Algorithm

Boosting can be categorized as observation filtering where the hypothesis uses weak learners' compatible observations and builds new weak learners for difficult observations. A gradient boosting algorithm has three elements: 1) an optimization loss function; 2) a weak learner for prediction; and 3) a cumulative model to accumulate weak learners to minimize the loss function [41], [42]. In general, gradient boosting supports the differentiable loss function $[L(y, F_m(x))]$, where y is the output and $F_m(x)$ with ν learning rate can be represented as

$$F_m(x) = F_{m-1}(x) + \nu\gamma_m h_m(x) \tag{1}$$

where $h_m(x)$ is the expected fitting decision tree or pseudo-residuals at the mth step and γ_m is the multiplier. The value of γ_m can be computed using the line search, which minimizes the loss function.

The advantage of this algorithm is that it does not need to provide loss functions for each new boosting algorithm; rather, one can use any generic differentiable loss function [41]. It fits a regression tree on the gradient of the least-squares regression loss function. However, the least absolute deviation is very popular due to its robustness, while the Huber loss function combines the first two and the quantile loss function allows quantile regression.

Gradient boosting uses decision trees as weak learners. It constructs the trees using the best split points based on a minimum loss function. In order to keep weak learners stable, it is common practice in weak learners to add constraints considering the maximum number of layers, nodes, and splits. The learning rate shrinks the contribution of each tree. Empirically, small ν yields exhibit better generalization ability compared to gradient boosting without shrinking. This work used $\nu = 0.05$ for the analysis.

In general, gradient boosting is considered to be robust to overfitting. As a result, a high number of boosting stages improves the accuracy. This model considered 200 boosting stages for optimal performance. Empirically, shorter trees are preferred over deeper complex trees, and trees of 4–8 levels exhibit better results. In addition, it considered up to four individual regression estimators for optimal results. For gradient boosting, it used a similar methodology, as shown in Algorithm 1. However, Line 6 replaced with a gradient boosting classifier.

D. Adaboost Algorithm

Similar to gradient boosting, the Adaboost algorithm tries to fit on a sequence of weak learners on continuously modified version of data [43]. However, the weak learners are small decision trees, marginally better than random guessing. The final prediction is produced using the weighted sum of all the weak predictors. At each boosting iteration, the data modification is performed by applying a set of weights $(w_1, w_2, \ldots, w_n)$ to each training dataset. The algorithm confirms the first sample to be a weak learner by initializing all weights, $w_i = (1/n)$. At each iteration, these weights are modified and the learning algorithm is continued to apply on the updated data. The prediction is improved by continuously increasing the incorrect predictors. On the other hand, the better predictors' weights are decreased due to the nature of the algorithm. As a result, the corner cases receive the greatest weight variation at each iteration. Hence, weak learners improve prediction accuracy in the subsequent iteration.

This model uses up to 200 learning estimators. However, if the model fits the dataset earlier than the given estimator value, the algorithm converges early. Similar to gradient boosting, it uses a learning rate of 0.05 for optimal performance. For Adaboost, it used an identical methodology, as shown in Algorithm 1. However, Line 6 replaced with an Adaboost classifier.

IV. Experiments and Results

A. Experimental Setup

For experiments, this work utilized community-developed open-source hardware consisting of crypto cores: MD5 pipeline, AES128, AES192, and AES256; arithmetic core: field-programmable gate array (FPGA) median; processor: OpenMSP and RISC16f84; co-processor: floating-point unit; and video controller: JPEGEncoder benchmarks from OpenCore [44]. It utilized Synopsys Design Compiler to perform logic synthesis and extract gate-level netlist. The synthesis features (i.e., clock period, aspect ratio, and utilization) are shown in Table I. For example, this work considered clock frequency ranging from 100 MHz to 2.5 GHz, aspect ratio from 0.7 to 0.95, and core utilization from 70% to 95%. In addition, it used the Synopsys 28-nm CMOS technology library in the synthesis.

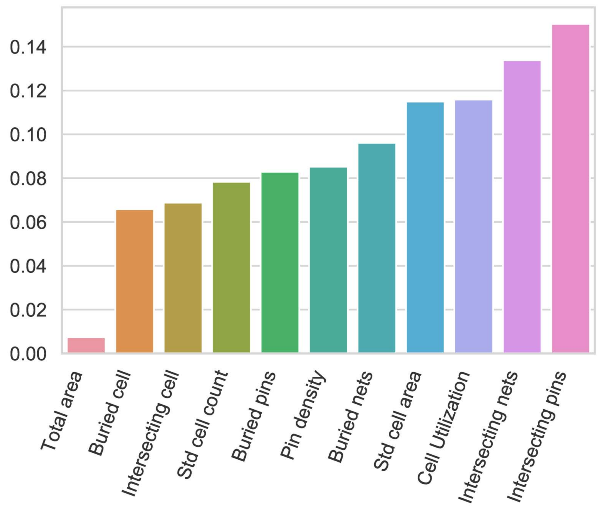
The author prepared a tool command language (TCL) script to perform place and route using Synopsys IC Compiler. The dataset size is nearly 60k grid points from the designs mentioned above. It considered 80% of the DRC-clean data and 60% of the DRC-violated data for training. Finally, it utilized the rest of the data for testing. This research implements the proposed algorithm using the Python programming language.

B. Experimental Results

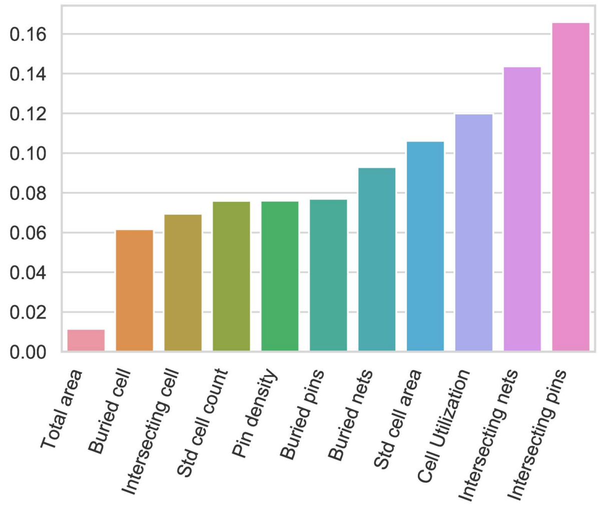
1) Features Importance: To identify the features that influence the DRC violations most, it used both the entropy and the Gini impurity index. The Gini impurity index-based and the entropy-based feature importance are shown in Fig. 3(a) and (b), respectively. The results are identical in

TABLE I
THIS RESEARCH USED COMMON SYNTHESIS FEATURES CONSIDERING CLOCK FREQUENCY RANGES FROM 100 MHz TO 2.5 GHz, ASPECT RATIO FROM 0.7 TO 0.95, AND CORE UTILIZATION FROM 70% TO 95%

Design types	Clock frequency (GHz)	Aspect ratio	Utilization (%)
Crypto cores	0.1–1.0	0.75–0.85	80–90
Arithmetic cores	1.0–2.5	0.70–0.95	70–95
Processors	0.1–0.5	0.80–0.85	80–85
Co-processors	1.0–2.5	0.70–0.95	70–95
Video controller	0.1–0.5	0.70–0.90	70–90

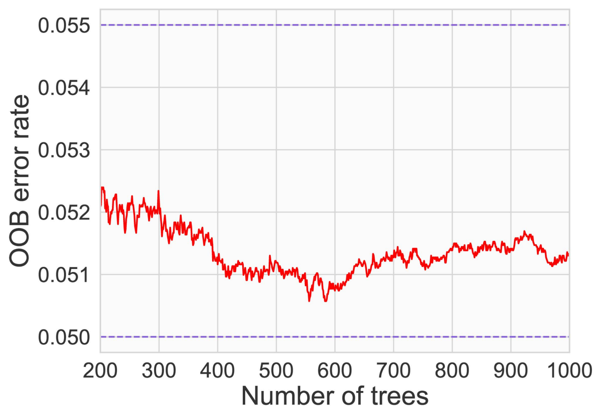


(a)

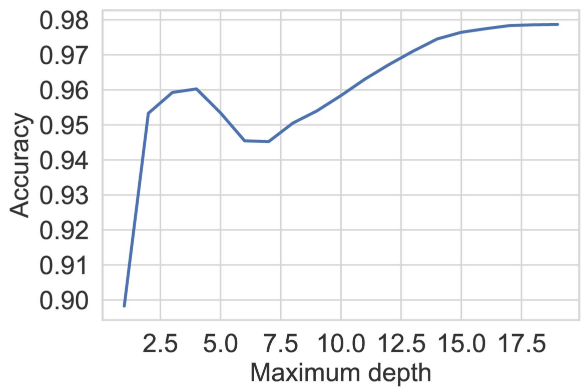


(b)

Fig. 3. Gini impurity and entropy for identifying the feature importance and considered features that have more than 5% impact on the proposed prediction model. (a) Gini impurity to identify feature importance, and it helps the algorithm to eliminate features that have a lesser impact on DRC violations. (b) Validated Gini impurity-based feature importance results using the entropy-based classifier.

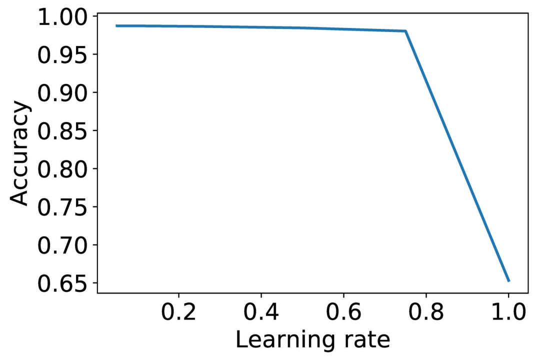


(a)

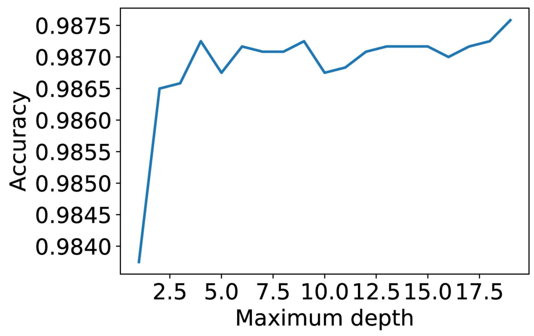


(b)

Fig. 4. Random forest classifier key features tuning. (a) For random forest classifier, the lowest OOB error was ∼5% for the number of trees in the range of 400–600. (b) It achieved the highest accuracy of 98% with the tree depth of 17.

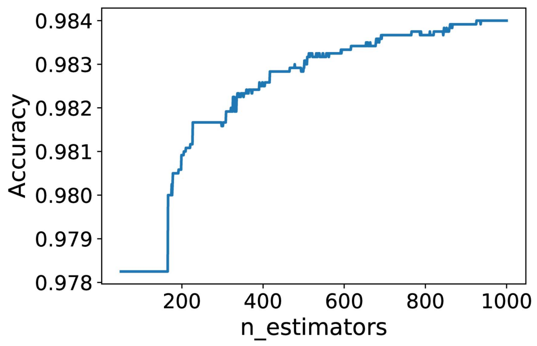


(a)

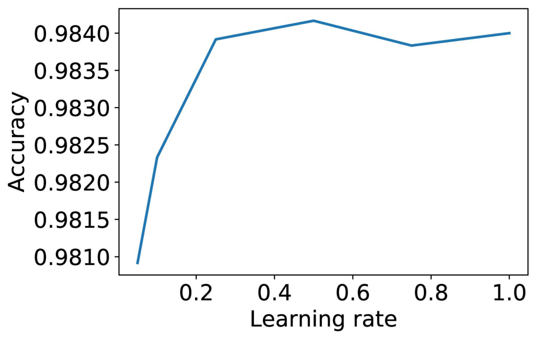


(b)

Fig. 5. Gradient boosting classifier key features tuning. (a) For gradient boosting classifier, the highest accuracy 98.73% achieved using a $\nu = 0.05$ value. (b) For the gradient boosting classifier, the changes in maximum depth have a negligible (i.e., less than 1%) impact on model accuracy.



(a)



(b)

Fig. 6. Adaboost classifier key features tuning. (a) For the Adaboost classifier, the changes in n_estimator parameter have negligible (i.e., less than 1%) impact on model accuracy. (b) For the Adaboost classifier, the accuracy varied less than 1% with the change of learning rate.

each method; however, the entropy-based classifier has slightly more amplitude than the Gini impurity-based method. In addition, using Gini impurity, the pin density has slightly more influence than buried pins in DRC violations, while when using the entropy-based method, the buried pins have more influence than the pin density on DRC violations. The proposed algorithm considered only those features that had more than a 5% impact on the DRC violations. As a result, only the total area feature is removed from the analysis, and this result is consistent between both methodologies.

2) *Key Parameters Tuning:* The random forest algorithm used two-thirds of the data for training each tree when building the forest. As a result, one-third of unseen data could improve accuracy without expensive cross validation. The bootstrap aggregation fits new trees from the training observations [45]. However, the algorithm experienced an error in each observation for not using the entire data. The average prediction error is called the out-of-bag (OOB) error. The proposed methodology varied the number of trees and computed the OOB, and according to the analysis, the number of trees in the range of 400–600 achieved the lowest OOB, as shown in Fig. 4(a). Another key feature that influences the performance of a random forest classifier is the depth of the tree. Empirically, the algorithm captures more information about the data with the increase of the tree's depth. In this analysis, the author considered the depth of 1–20. The observed highest accuracy was about 98% with the tree depth of 17, as shown in Fig. 4(b).

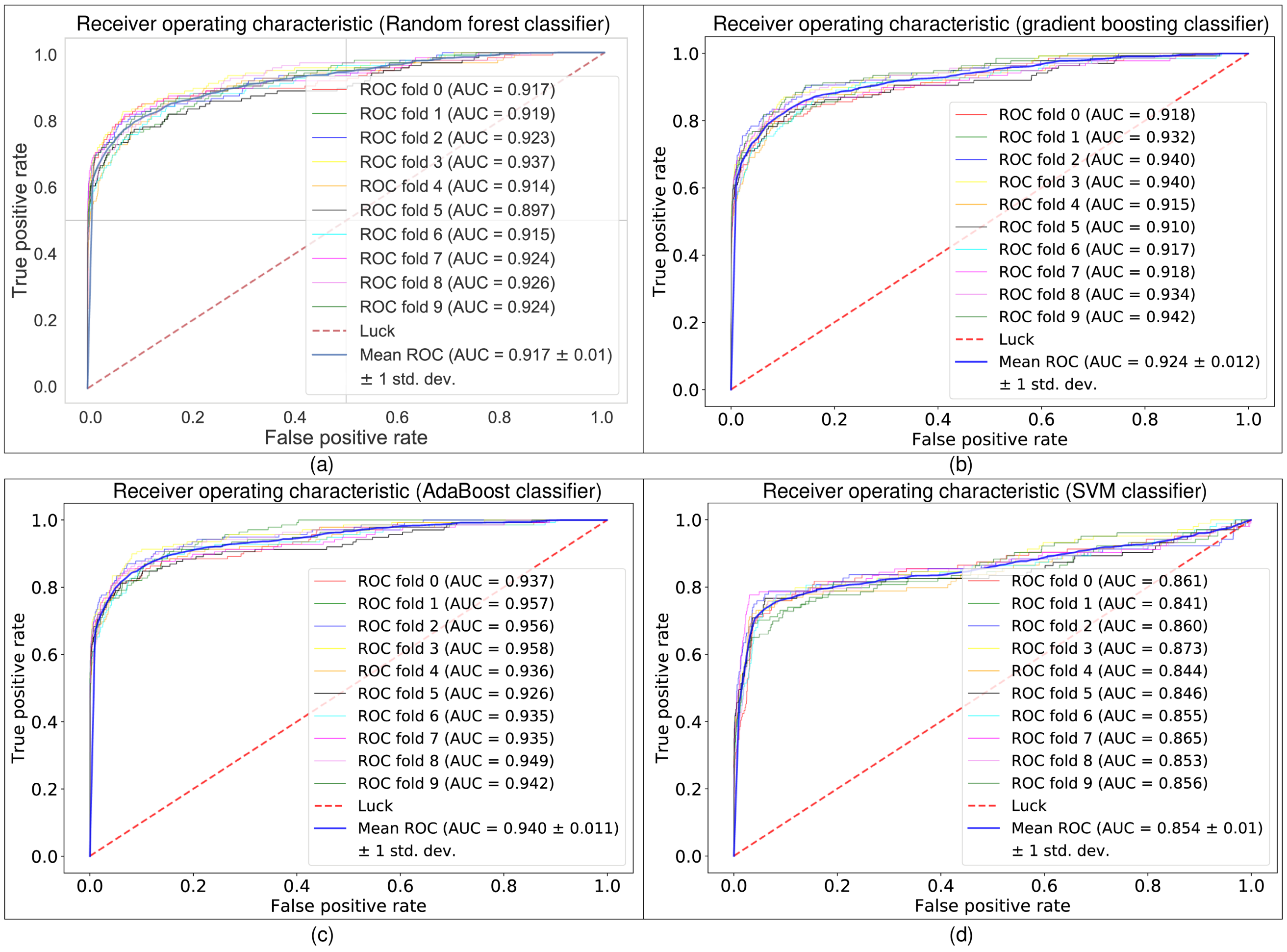


Fig. 7. Proposed ensemble algorithms exhibit better receiver operating characteristics compared to the existing SVM classifier. (a) Random forest classifier exhibits a mean area under the curve of 0.917 ± 0.01. (b) Gradient boosting classifier exhibits a mean area under the curve of 0.924 ± 0.012. (c) Adaboost classifier exhibits a mean area under the curve of 0.94 ± 0.011 compared to (d) SVM's 0.854 ± 0.01.

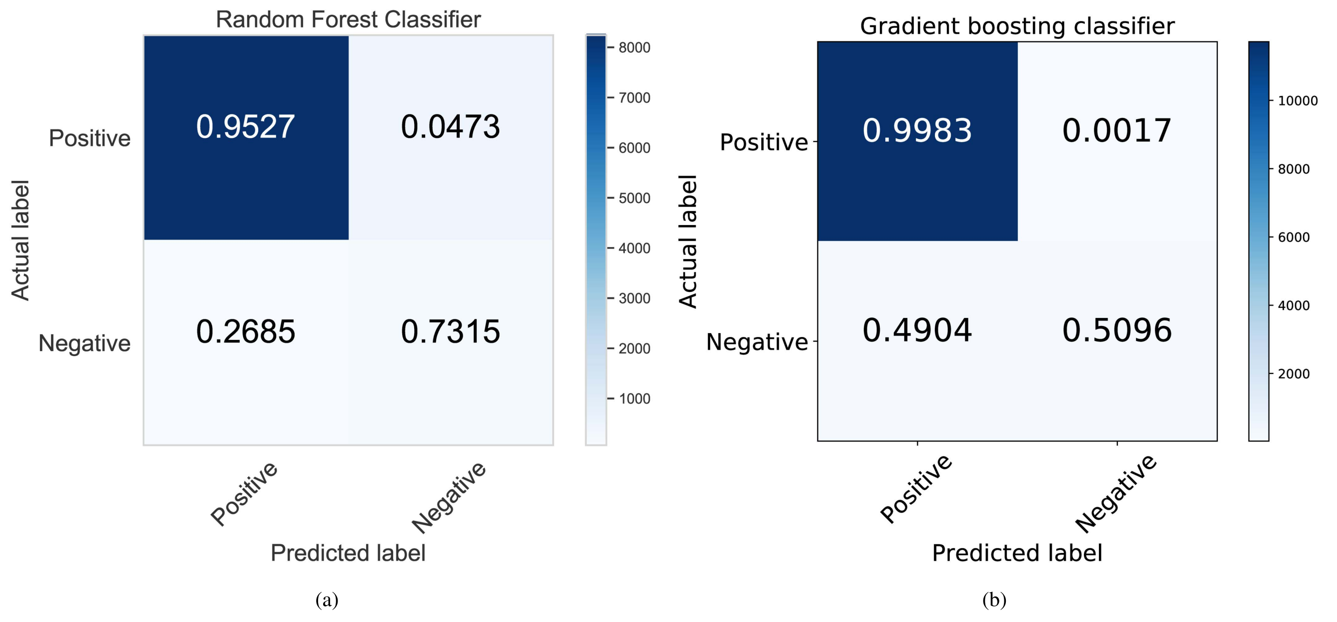
The gradient boosting algorithm uses an initial estimate and then updates using the output of each tree. The ν parameter controls the value of this estimate. A lower ν value makes the model robust to the tree's specific characteristics and generalizes well. The analysis of ν versus accuracy also justifies this statement, as shown in Fig. 5(a). The maximum depth or number of nodes in the tree is crucial for improving the gradient boosting algorithm's performance. Similar to the random forest, this research considers the depth of 1–20. However, unlike random forest, the accuracy varies less than 1% with the change in maximum depth, as shown in Fig. 5(b).

The Adaboost algorithm used the n_estimator parameter to control the number of weak learners. This is a crucial parameter where the model stopped learning if it could not fit a smaller value. According to this analysis, the model achieved its maximum accuracy using the ∼200 value, as shown in Fig. 6(a). Unlike the gradient boosting classifier, the Adaboost was relatively robust with the variation of learning rate. The accuracy varied less than 1% with the change in the learning rate, as shown in Fig. 6(b).

3) Accuracy Measurements: In order to verify the proposed methodology's robustness, this research shuffled the collected dataset into conventional ten-fold cross validation. It considered nine sets for training and the remaining one for testing in each step. To estimate the efficiency of the proposed algorithm, it verified it with AUC–ROC [46]. Besides, it computed the precision, recall, and F1-scores. In general, the AUC–ROC curve plays a significant role in evaluating the performance of a classification algorithm. The ROC is a probability diagram and the AUC signifies the classification accuracy. A prediction model's strength is represented by its high AUC rate, implying a high prediction accuracy of TPR and true negative rate.

Fig. 7(a) shows the AUC–ROC curve for the random forest algorithm, and it exhibited a mean of 0.917 with ±0.01 standard deviation. It verified the model using the evaluation metrics on all ten folds of the shuffled data. The proposed method exhibited a top precision, recall, and F1-score of 97%, 95%, and 96%, respectively. Table II shows the metrics score on each of the ten folds for the random forest algorithm.

Fig. 7(b) shows the AUC–ROC curve for the gradient boosting algorithm, and it exhibited a mean of 0.924 with ±0.012 standard deviation. It verified the gradient boosting model using the evaluation metrics on all ten folds of the shuffled data such as the random forest algorithm. The proposed method exhibited a maximum precision, recall, and



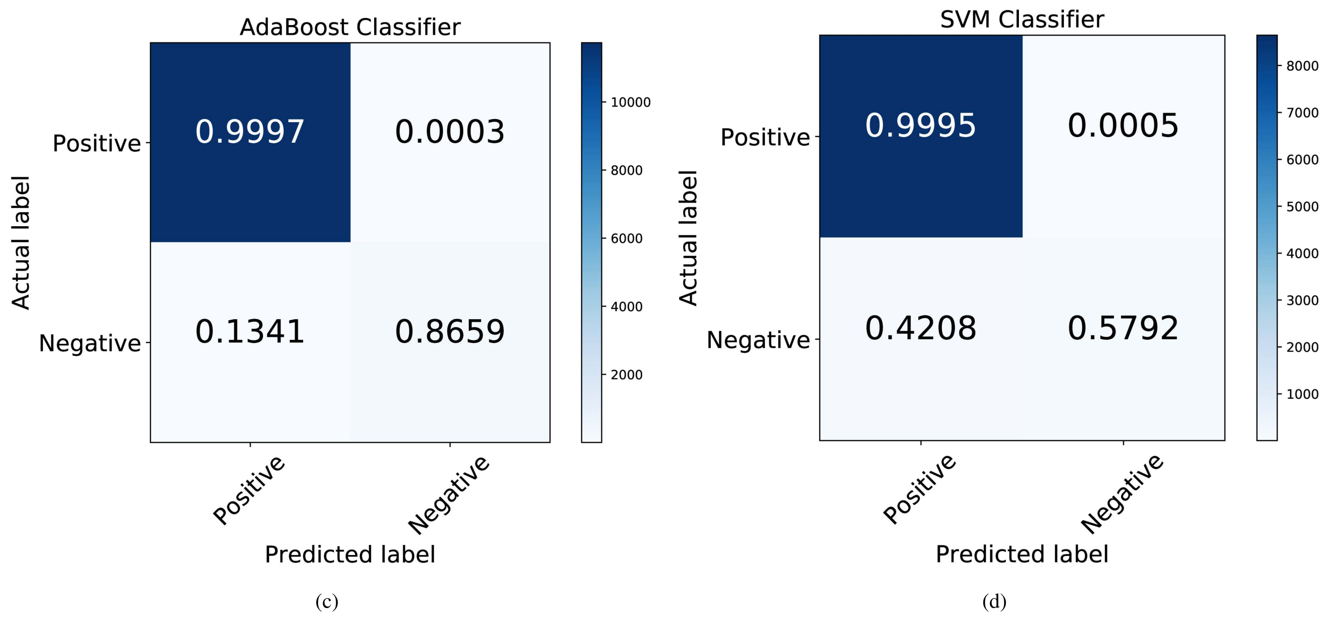


Fig. 8. Proposed gradient boosting and Adaboost algorithms have comparable TPRs to the existing SVM classifiers [2]. In contrast, the proposed Adaboost algorithm has a 28.7% better FPR than the SVM classifiers on the test data. (a) Random forest has a TPR of 95.27%. (b) Gradient boosting has a TPR of 99.83%. (c) Adaboost algorithm has a TPR of 99.97%. (d) SVM classifier has a TPR of 99.95%.

F1-score of 99%, 99%, and 99%, respectively. Table II shows the metrics score on each of the ten folds using the gradient boosting algorithm.

Fig. 7(c) shows the AUC–ROC curve for the Adaboost algorithm, and it exhibited a mean of 0.940 with ± 0.011 standard deviation compared to the existing SVM classifier-based model, which showed a mean of 0.854 with ± 0.01 standard deviation, as shown in Fig. 7(d). Like the other ensemble approaches, it verified the Adaboost model using the evaluation metrics on all ten folds of the shuffled data. The proposed Adaboost method displayed a top precision, recall, and F1-score of 98%, 98%, and 97%, compared to the state-of-the-art SVM classifiers' [2] 98%, 98%, and 97%, respectively. Table II shows the metrics score on every ten folds using the Adaboost algorithm.

The author proposed methodologies were compared by implementing the existing SVM classifiers for the metrics of all ten folds, as shown in Table II. The average AUC (training and test) values for the proposed random forest, gradient boosting, and Adaboost algorithms were (0.9525 with ± 0.0014 standard deviation and 0.9207 with ± 0.0101 standard deviation), (0.9456 with ± 0.0018 standard deviation and 0.9266 with ± 0.0123 standard deviation), and (0.9432 with ± 0.0012 standard deviation and 0.9432 with ± 0.0112 standard deviation), respectively, while the average AUC (training and test) values for the existing SVM classifier were (0.9981 with ± 0.0005 standard deviation and 0.8544 with ± 0.0104 standard deviation), as shown in Table II.

Fig. 8 shows the confusion matrices of the proposed ensemble methods compared to the existing SVM classifiers on

TABLE II

AMONG ALL THE PROPOSED ENSEMBLE ALGORITHMS, THE GRADIENT BOOSTING (GB) AND ADABOOST (ADAB) ALGORITHMS HAVE SIMILAR ACCURACY AND ARE SLIGHTLY BETTER THAN THE RANDOM FOREST (RF) ALGORITHM IN TERMS OF PRECISION, RECALL, AND F1-SCORES

Type	K-fold	AUC training	AUC test	Precision (%)	Recall (%)	F1-score (%)	T_r time (sec)	T_s time (sec)
	Fold 1	0.9521	0.9168	97	94	95	3.90	91.2
	Fold 2	0.9534	0.9194	97	94	95	4.94	63.2
	Fold 3	0.9520	0.9234	97	94	96	3.11	63.1
	Fold 4	0.9499	0.9374	97	94	96	3.13	79.8
	Fold 5	0.9535	0.9242	97	94	95	3.15	77.7
RF	Fold 6	0.9550	0.8975	97	95	96	4.26	73.2
	Fold 7	0.9528	0.9155	97	94	95	3.54	74.4
	Fold 8	0.9518	0.9237	97	94	95	3.30	64.2
	Fold 9	0.9536	0.9259	97	95	96	3.18	62.4
	Fold 10	0.9512	0.9235	97	95	96	3.18	74.7
	Avg.	**0.9525 ± 0.0014**	**0.9207 ± 0.0101**	**97**	**94.3**	**95.5**	**3.57**	**72.4**
	Fold 1	0.9459	0.9182	99	99	98	2.79	22.1
	Fold 2	0.9456	0.9321	99	99	99	2.79	21.3
	Fold 3	0.9430	0.9402	99	99	99	2.76	23.5
	Fold 4	0.9442	0.9404	99	99	99	2.81	19.3
	Fold 5	0.9465	0.9153	98	99	98	2.81	18.5
GB	Fold 6	0.9494	0.9091	99	99	98	2.84	24.1
	Fold 7	0.9459	0.9169	99	99	99	2.81	16.5
	Fold 8	0.9466	0.9179	98	99	98	2.79	17.5
	Fold 9	0.9453	0.9337	99	99	99	2.77	19.3
	Fold 10	0.9433	0.9417	99	99	99	2.85	18.6
	Avg.	**0.9456 ± 0.0018**	**0.9266 ± 0.0123**	**98.8**	**99**	**98.6**	**2.80**	**20.1**
	Fold 1	0.9439	0.9371	98	99	98	4.50	82.3
	Fold 2	0.9416	0.9571	98	98	97	4.57	88.5
	Fold 3	0.9418	0.9559	98	98	97	4.50	81.3
	Fold 4	0.9415	0.9584	98	98	97	4.50	78.1
	Fold 5	0.9439	0.9360	98	98	97	4.54	87.1
AdaB	Fold 6	0.9451	0.9259	98	98	97	4.57	91.2
	Fold 7	0.9440	0.9353	98	98	97	4.56	100.2
	Fold 8	0.9441	0.9356	98	98	97	5.13	90.5
	Fold 9	0.9425	0.9491	98	98	97	4.57	89.6
	Fold 10	0.9433	0.9417	98	98	97	4.63	88.5
	Avg.	**0.9432 ± 0.0012**	**0.9432 ± 0.0112**	**98**	**98**	**97**	**4.61**	**87.7**
	Fold 1	0.9984	0.8613	99	99	98	115.3	952.0
	Fold 2	0.9979	0.8410	98	98	97	94.2	962.0
	Fold 3	0.9973	0.8599	98	98	97	99.2	952.0
	Fold 4	0.9977	0.8727	98	98	97	102.8	1025
	Fold 5	0.9982	0.8444	98	98	97	99.1	982.0
SVM [2]	Fold 6	0.9984	0.8461	98	98	97	103.2	1202
	Fold 7	0.9989	0.8450	98	98	97	142.8	1003
	Fold 8	0.9981	0.8653	98	98	97	102.2	982.0
	Fold 9	0.9977	0.8528	98	98	97	98.8	895.1
	Fold 10	0.9982	0.8557	98	98	97	93.3	986.3
	Avg.	**0.9981 ± 0.0005**	**0.8544 ± 0.0104**	**98**	**98**	**97**	**105.1**	**1002.94**

the test data. Fig. 8(a) and (b) shows the random forest and gradient boosting algorithms results with TPR of 95.27% and 99.83% and FPR of 73.15% and 50.96%, respectively. In this analysis, TPR and FPR are inferred to the accuracy of DRC violation-free data and data with DRC violations. The proposed Adaboost algorithm exhibited TPR and FPR of 99.97%

and 86.59%, while the existing SVM classifier has 99.95% and 57.92%, as shown in Fig. 8(c) and (d), respectively. The proposed random forest, gradient boosting, and Adaboost algorithms exhibited 94.52%, 98.73%, and 98.09% classification accuracy, respectively, compared to the existing supervised machine learning algorithm with 79.8% average accuracy [7]. Besides, the proposed ensembled approaches exhibit better accuracy compared to the NN ensemble approach [34], which has only 90% prediction accuracy.

Among different deep learning models, the PPR model [6] exhibited 92.35% accuracy, which is lower than the proposed Adaboost algorithm's accuracy of 98.09%. RouteNet [29] exhibits an average FPR of 46%, and the proposed Adaboost algorithm showed a much better FPR of 86.59%.

4) Algorithms Training and Test Times: Table II shows the training time (T_r) and testing time (T_s) of each fold of all the ensemble algorithms. According to this research, the gradient boosting algorithm required the lowest average training and testing time, and the AdaBoost algorithm required the highest training and testing time, compared to the other ensemble methods. However, the proposed Adaboost algorithm showed 22.8× and 11.1× lower average training time and testing time, respectively, compared to the SVM classifiers [2]. The proposed gradient boosting algorithm required 37.5× lower average training time and 50× lower average testing time compared to the existing SVM methodologies.

V. Conclusion

In this article, we presented a new 16T SEDU-hardened storage cell and its usage in a latch design. The proposed latch is 81% faster, requires a lower silicon area, and consumes 25% less power compared to a recently reported DNCS latch at 2.5 GHz. In addition, the proposed latch has 86% lower PDP than the DNCS latch. Better yet, the unconstrained sizing architecture of the proposed 16T cell makes it more attractive for dense memory or register file design.

In addition, we presented the first SEDU-hardened flip-flop using our 16T cell. The flip-flop consumes up to 50% lower power and is 29% faster compared to the existing partial SEDU-hardened flip-flop. In addition, the proposed flip-flop consumes 27% lower area than the competing flip-flop. Due to the internal feedback loop, the flip-flop consumes lower power at low data activity. The proposed flip-flop consumes 25% lower static power and has 45% lower setup time compared to the competing flip-flop. Better yet, the negative hold time of the proposed flip-flop makes it more attractive for high-frequency operations.

Acknowledgment

The author would like to thank R. Refat and A. Shahjalal from the University of Michigan Dearborn for initial data collection and analysis.

References

[1] (2017). *DARPA Rolls Out Electronics Resurgence Initiative*. [Online]. Available: https://www.darpa.mil/news-events/2017-09-13

[2] W. J. Chan, Y. Du, A. B. Kahng, S. Nath, and K. Samadi, "BEOL stack-aware routability prediction from placement using data mining techniques," in *Proc. IEEE 34th Int. Conf. Comput. Design*, Oct. 2016, pp. 41–48.

[3] M. Fogaça, A. B. Kahng, R. Reis, and L. Wang, "Finding placement-relevant clusters with fast modularity-based clustering: Part I—Methodology and design strategies," in *Proc. 24th Asia South Pacific Design Autom. Conf.* New York, NY, USA: Association for Computing Machinery, 2019, pp. 569–576.

[4] T. Ajayi *et al.*, "Toward an open-source digital flow: First learnings from the OpenROAD project," in *Proc. 56th Annu. Design Automat. Conf.* New York, NY, USA: Association for Computing Machinery, 2019, pp. 1–4.

[5] A. B. Kahng, S. Kumar, and T. Shah, "A no-human-in-the-loop methodology toward optimal utilization of EDA tools and flows," in *Proc. ACM/EDAC/IEEE Design Autom. Conf. WIP Session*, Jun. 2018, pp. 1–6.

[6] T.-C. Yu *et al.*, "Pin accessibility prediction and optimization with deep learning-based pin pattern recognition," in *Proc. ACM/IEEE 56th Annu. Design Autom. Conf.*, Jun. 2019, pp. 1–6.

[7] Q. Zhou, X. Wang, Z. Qi, Z. Chen, Q. Zhou, and Y. Cai, "An accurate detailed routing routability prediction model in placement," in *Proc. 6th Asia Symp. Quality Electron. Design (ASQED)*, Aug. 2015, pp. 119–122.

[8] A. Kahng, "Opportunities for machine learning in IC physical design," in *Proc. IEEE CASS Seasonal School Phys. Design Autom.*, Aug. 2017, p. 1.

[9] R. Islam, "Feasibility prediction for rapid IC design space exploration," *Electronics*, vol. 11, no. 7, p. 1161, Apr. 2022.

[10] W.-T.-J. Chan, P.-H. Ho, A. B. Kahng, and P. Saxena, "Routability optimization for industrial designs at sub-14 nm process nodes using machine learning," in *Proc. ACM Int. Symp. Phys. Design*, Mar. 2017, pp. 15–21.

[11] R. Islam and M. A. Shahjalal, "Predicting DRC violations using ensemble random forest algorithm," in *Proc. 56th Annu. Design Autom. Conf.*, Jun. 2019, p. 227.

[12] A. Onan, S. Korukoğlu, and H. Bulut, "A multiobjective weighted voting ensemble classifier based on differential evolution algorithm for text sentiment classification," *Expert Syst. Appl.*, vol. 62, pp. 1–16, Nov. 2016.

[13] A. Onan and S. Korukoğlu, "A feature selection model based on genetic rank aggregation for text sentiment classification," *J. Inf. Sci.*, vol. 43, no. 1, pp. 25–38, 2017.

[14] A. Onan, "Sentiment analysis on massive open online course evaluations: A text mining and deep learning approach," *Comput. Appl. Eng. Educ.*, vol. 29, no. 3, pp. 572–589, May 2021.

[15] A. Onan and S. Korukoğlu, "Exploring performance of instance selection methods in text sentiment classification," in *Artificial Intelligence Perspectives in Intelligent Systems*. Cham, Switzerland: Springer, 2016, pp. 167–179.

[16] M. A. Toçoğlu and A. Onan, "Sentiment analysis on students' evaluation of higher educational institutions," in *Proc. Int. Conf. Intell. Fuzzy Syst.*, 2020, pp. 1693–1700.

[17] A. Onan, S. Korukoğlu, and H. Bulut, "A hybrid ensemble pruning approach based on consensus clustering and multi-objective evolutionary algorithm for sentiment classification," *Inf. Process. Manag.*, vol. 53, no. 4, pp. 814–833, Jul. 2017.

[18] A. Onan, "An ensemble scheme based on language function analysis and feature engineering for text genre classification," *J. Inf. Sci.*, vol. 44, no. 1, pp. 28–47, 2018.

[19] A. Onan and M. A. Toçoglu, "A term weighted neural language model and stacked bidirectional LSTM based framework for sarcasm identification," *IEEE Access*, vol. 9, pp. 7701–7722, 2021.

[20] A. Onan, "Hybrid supervised clustering based ensemble scheme for text classification," *Kybernetes*, vol. 46, no. 2, pp. 330–348, 2017.

[21] A. Onan, "Topic-enriched word embeddings for sarcasm identification," in *Software Engineering Methods in Intelligent Algorithms*. Cham, Switzerland: Springer, 2019, pp. 293–304.

[22] K. Jeong, A. B. Kahng, B. Lin, and K. Samadi, "Accurate machine-learning-based on-chip router modeling," *IEEE Embedded Syst. Lett.*, vol. 2, no. 3, pp. 62–66, Sep. 2010.

[23] W.-T.-J. Chan, K. Young Chung, A. B. Kahng, N. D. MacDonald, and S. Nath, "Learning-based prediction of embedded memory timing failures during initial floorplan design," in *Proc. ACM/IEEE/EDAC 21st Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2016, pp. 178–185.

[24] R. Islam and M. R. Guthaus, "HCDN: hybrid-mode clock distribution networks," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 66, no. 1, pp. 251–262, Jan. 2019.

[25] A. B. Kahng, "New directions for learning-based IC design tools and methodologies," in *Proc. ACM/IEEE/EDAC Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2018, pp. 405–410.

[26] Z. Qi, Y. Cai, and Q. Zhou, "Accurate prediction of detailed routing congestion using supervised data learning," in *Proc. IEEE 32nd Int. Conf. Comput. Design (ICCD)*, Oct. 2014, pp. 97–103.

[27] A. F. Tabrizi *et al.*, "A machine learning framework to identify detailed routing short violations from a placed netlist," in *Proc. 55th ACM/ESDA/IEEE Design Autom. Conf. (DAC)*, Jun. 2018, pp. 1–6.

[28] R. Islam and M. A. Shahjalal, "Soft voting-based ensemble approach to predict early stage DRC violations," in *Proc. IEEE 62nd Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2019, pp. 1081–1084.

[29] Z. Xie *et al.*, "RouteNet: Routability prediction for mixed-size designs using convolutional neural network," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Nov. 2018, pp. 1–8.

[30] A. F. Tabrizi, N. K. Darav, L. Rakai, A. Kennings, and L. Behjat, "Detailed routing violation prediction during placement using machine learning," in *Proc. Int. Symp. VLSI Design, Autom. Test (VLSI-DAT)*, Apr. 2017, pp. 1–4.

[31] A. Mirhoseini *et al.*, "Chip placement with deep reinforcement learning," 2020, *arXiv:2004.10746*.

[32] R. Liang *et al.*, "DRC hotspot prediction at sub-10 nm process nodes using customized convolutional network," in *Proc. Int. Symp. Phys. Design*. New York, NY, USA: Association for Computing Machinery, 2020, pp. 135–142.

[33] L. Li, Y. Cai, and Q. Zhou, "An efficient approach for DRC hotspot prediction with convolutional neural network," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2021, pp. 1–5.

[34] W. Zeng, A. Davoodi, and Y. H. Hu, "Design rule violation hotspot prediction based on neural network ensembles," 2018, *arXiv:1811.04151*.

[35] A. B. Kahng, S. Kang, H. Lee, S. Nath, and J. Wadhwani, "Learning-based approximation of interconnect delay and slew in signoff timing tools," in *Proc. ACM/IEEE Int. Workshop Syst. Level Interconnect Predict. (SLIP)*, Jun. 2013, pp. 1–8.

[36] S. S. Kozat and A. C. Singer, "Universal switching linear least squares prediction," *IEEE Trans. Signal Process.*, vol. 56, no. 1, pp. 189–204, Jan. 2008.

[37] S. Bao, "Optimizing leakage power using machine learning," Dept. Comput. Sci., Stanford Univ., Stanford, CA, USA, Final Project CS229, 2010.

[38] M. Gao, Z. Ye, Y. Wang, and Z. Yu, "On modeling the digital gate delay under process variation," *J. Semicond.*, vol. 32, no. 7, Jul. 2011, Art. no. 075010.

[39] S. Ganapathy, R. Canal, A. Gonzalez, and A. Rubio, "Circuit propagation delay estimation through multivariate regression-based modeling under spatio-temporal variability," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2010, pp. 417–422.

[40] B. Leo, "Random forests," *Mach. Learn.*, vol. 45, pp. 5–32, Oct. 2001.

[41] J. Brownlee. (2016). *A Gentle Introduction to the Gradient Boosting Algorithm for Machine Learning*. [Online]. Available: https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/

[42] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Ann. Statist.*, vol. 29, no. 5, pp.1189–1232, Oct. 2001.

[43] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, Aug. 1997.

[44] (2019). *OpenCores: Open Source IP-Cores*. [Online]. Available: http://www.opencores.org

[45] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning* (Springer Series in Statistics). Cham, Switzerland: Springer, 2009, pp. 592–593.

[46] S. H. Park, J. M. Goo, and C.-H. Jo, "Receiver operating characteristic (ROC) curve: Practical review for radiologists," *Korean J. Radiol.*, vol. 5, no. 1, pp. 11–28, 2004.

Riadul Islam (Senior Member, IEEE) is currently an Assistant Professor with the Department of Computer Science and Electrical Engineering, University of Maryland at Baltimore County, Baltimore, MD, USA. In his Ph.D. dissertation work at UCSC, he designed the first current-pulsed flip-flop/register that resulted in the first-ever one-to-many current-mode clock distribution networks for high-performance microprocessors. From 2017 to 2019, he was an Assistant Professor with the University of Michigan, Dearborn MI, USA. He holds two U.S. patents and several IEEE/ACM/MDPI/Springer Nature journal and conference publications in IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR PAPERS, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS, IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTING, IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, IEEE CANADIAN JOURNAL OF ELECTRICAL AND COMPUTER ENGINEERING, Journal of Electronic Testing (JETTA), Design Automation Conference (DAC), International Symposium on Circuits and Systems (ISCAS), Midwest Symposium on Circuits and Systems (MWSCAS), International Symposium on Quality Electronic Design (ISQED), and International Conference on ASIC (ASICON). His current research interests include digital, analog, and mixed-signal CMOS integrated circuits (ICs)/system-on-chips (SOCs) for a variety of applications; verification and testing techniques for analog, digital, and mixed-signal ICs; hardware security; Controller Area Network (CAN); computer-aided design (CAD) tools for design and analysis of microprocessors and field-programmable gate arrays (FPGAs); automobile electronics; and biochips.

Prof. Islam was a Technical Program Committee (TPC) Member of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD 2022), the 30th ACM Great Lakes Symposium on VLSI (GLSVLSI) 2020, 31th ACM GLSVLSI 2021, 32nd ACM GLSVLSI 2022, 57th IEEE/ACM DAC 2020 LBR Session, IEEE Computer Society Annual Symposium on VLSI (ISVLSI) 2021, and IEEE International Conference on Consumer Electronics (ICCE) 2021. He is a member of the ACM, IEEE Circuits and Systems (CAS) Society, the VLSI Systems and Applications Technical Committee (VSA-TC) of the IEEE-CAS, and IEEE Solid-State Circuits (SSC) Society. He was a recipient of the 2021 NSF ERI Award, the 2021 Maryland Industrial Partnerships (MIPS) Award, and the 2021 Maryland Innovation Initiative (MII) Award. He is an Associate Editor of Springer Circuits, Systems and Signal Processing (CSSP) journal.