Check Low-Power Violations by Using Machine Learning Based Classifier

Chi-Ming Lee, Chung-An Wang, Cheok-Yan Goh, Chia-Cheng Tsai,
Chien-Hsin Yeh, Chia-Shun Yeh, Chin-Tang Lai
MediaTek Inc., Hsinchu, Taiwan
{Ross.Lee, CA.Wang, Paddy.Wu, Joseph.Tsai, Chien.Yeh, Jason.Yeh, Citi.Lai}@mediatek.com

***Abstract*- The violation check process for design verification often requires designers to manually inspect and identify design issues. As the complexity of the design increases, the overwhelming workload during the time-limited violation review process may jeopardize the correction for the RTL code and induce Engineering Change Order (ECO). In this paper, we present a practical solution with machine learning techniques that learn to identify highly dubious violations from the historical violation data and reduce the effort of manual review on ignorable violations. We leverage the ensemble XGBoost model with the Bag-of-words (BoW) representation and the well-designed features to optimize learning performance. The empirical study on three different designs demonstrates that we can reduce at least the 12% review effort without the human expert's involvement. Also, we can reduce false alarms, and the largest reduction of total violations reached up to 50%.**

I. Introduction

To verify a design, simulation-based techniques are the most popular approaches. The simulations are iteratively conducted to ensure its correct functionality and behavior before design tape-out. While the simulator runs various checks on the design, the violations are output to flag suspicious conditions. In each simulation iteration, designers manually review the violations which are either waivable violations (false alarms) or true design issues that must be fixed as mentioned in [1]. Fig. 1 shows the violation check flow. The review process requires cross-team expertise of design domain knowledge and lots of time to investigate violations. As the complexity of the design increases, the number of violations increases. The dramatically increased workload of the manual review process usually lets the time-limited violation check process remain unfinished. In the end, design issues may fail to be observed after RTL freeze. Designers need to implement Engineering Change Order (ECO) to fix these design issues and it may postpone the tape-out process.

Inspired by spam filtering problems [2], we formulate the violation check process as a supervised binary classification problem. The spam email filtering is one of the classic Machine Learning (ML) applications. These ML techniques have the capacity to learn and identify spam mails and phishing messages by analyzing the content of the emails or messages throughout a vast collection of computers. The violations in IC design check flow are also composed of text, and the words in the signals usually represent certain physical characteristics. Therefore, we apply ML techniques to learn how to identify highly dubious violations from historical violation data and reduce the effort of manual review on ignorable violations.

In this paper, we take Isolation clamp value check process, one of the low-power check items to inspect correct isolation cell types between two power domains, as an example to demonstrate our solution. As Fig. 2 shows, in the training stage, we implement ML methods to learn an Artificial Intelligence (AI) Checker that inherits and integrates the experiences of design domain experts across time and various design architectures. In the inference stage, we use the AI Checker to classify violations into waivable or high-risk violations to reduce false alarm violations. Finally, the validated labels of inspected violations can be used for incremental learning. Experiments show the model can discover all true design issues in predicted high-risk violations and the number of total violations can be reduced up to 50% to achieve shift left of design issues as Fig. 3 shows.

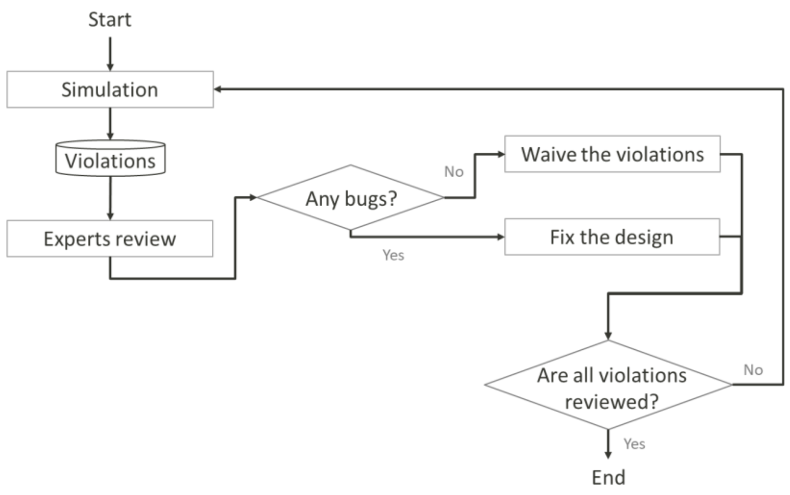

Figure 1. The iterative violation check flow.

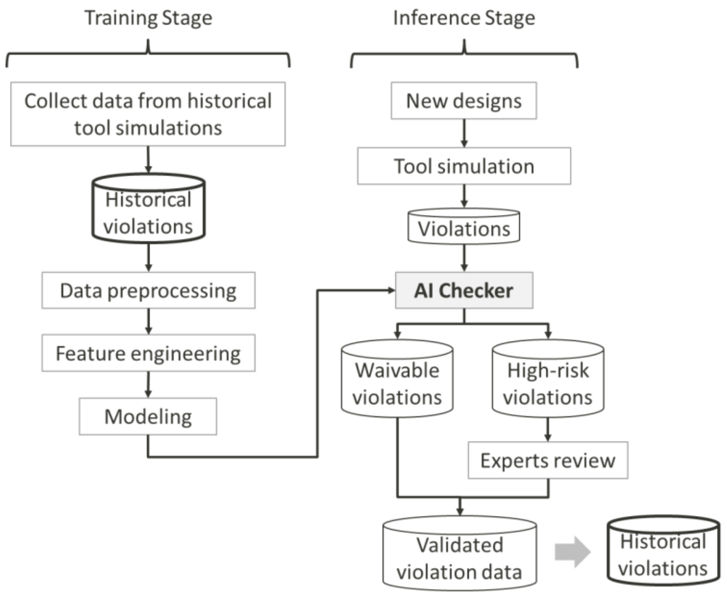

Figure 2. The ML-based violation check flow.

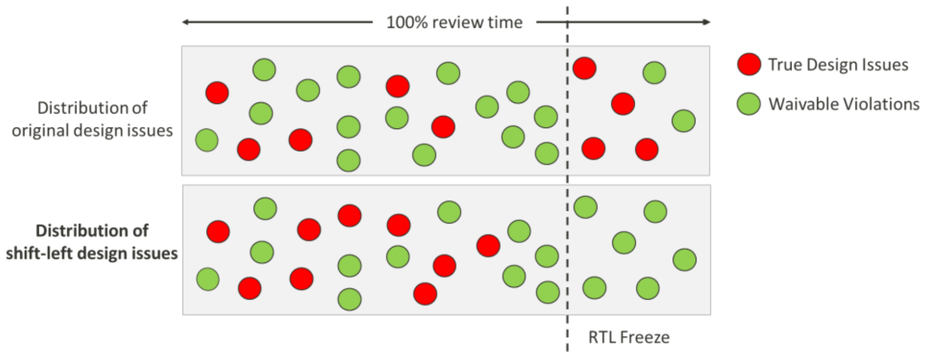
100% review time
Distribution of original design issues
Distribution of shift-left design issues
RTL Freeze
True Design Issues
Waivable Violations

Figure 3. The illustration of design issues shift left.

II. APPROACH

In this section, we introduce our ML-based approach in more details. It consists of three steps, including data preprocessing, feature engineering, and ensemble modeling. To learn an AI Checker, the proposed approach is illustrated in Fig. 4. In the data preprocessing phase, we describe how to represent violation logs using Bag-of-words (BoW) [3] vectors and convert these vectors as a Document-term Matrix (DTM) [4]. During the feature engineering phase, we create generalized features that allow the model to learn the violations of different designs effectively. We illustrate the concept and implementation steps of new feature generation. In the model construction phase, we describe how to optimize model performance through machine learning techniques. We design an ensemble method that splits the data for model training and votes for the highest probability as the final prediction.

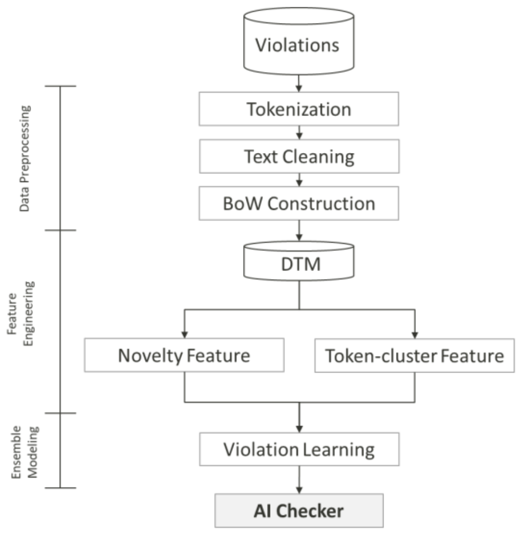

Figure 4. The overall flow of proposed approach.

A. Data Preprocessing

The purpose of this step is to convert the data into a format that can be learned by machine learning models. First, we tokenize violation logs and remove the noise such as word capitals, digits, and special punctuations. Then we represent the unstructured violations with structured BoW features as the input for ML model.

1. Tokenization: Tokenization is a technique that used to segment a string into substrings. We apply word tokenization to separate each violation signal as a set of representative symbols called tokens. The signal log shows an example of word tokenization, where the input of low-power check violation signal is

   “Common1_Common2_A1_B2_[C3]_[D4]_[E5]”,

   and the expected tokenized result is

   {“Common1”, “Common2”, “A1”, “B2”, “[C3]”, “[D4]”, “[E5]”}.

2. Text Cleaning: The goal of this step is to remove irrelevant information from signal tokens. Typically, it is customized based on design domain knowledge. In this paper, we first apply text normalization to lowercase texts. Next, the common prefix tokens are eliminated to prevent the model from learning noises. Also, numbers or special symbols such as square brackets in the token are removed. Some conventions of naming rules should be aligned in case of misleading models. The example of the signal tokens is

   {“Common1”, “Common2”, “A1”, “B2”, “[C3]”, “[D4]”, “[E5]”}.

   The expected result of this example is {“a”, “b”, “c”, “d”, “e”}. First, the common prefix tokens, square brackets and digits are removed. Second, we lowercase all tokens.

3. BoW Construction: Machine learning algorithms can’t work with raw text directly so that the text must be converted into numbers or specifically speaking, vectors of numbers. A Bag-of-words is a representation of text that describes the occurrence of words within a document. We use it to encode each violation log. To construct BoW vectors, we collect a vocabulary of known tokens and measure token occurrence for each violation log. These BoW vectors of violations can be further organized as a DTM for training ML models. The examples of the BoW and the DTM construction are described as follows. Assume that we have 3 lists of preprocessed violation logs:

{[“a”, “b”, “b”, “d”, “e”],
[“a”, “b”, “c”, “f”, “h”],
[“b”, “c”, “d”, “f”, “g”]}.

We first construct the vocabulary which is a list of unique signal tokens:

[“a”, “b”, “c”, “d”, “e”, “f”, “g”, “h”].

Next, we count the occurrences of tokens for each violation log and encode it as a BoW vector. Taking first violation as an example, we can get a vector of token counts in constructed vocabulary:

{“a”:1, “b”:2, “c”:0, “d”:1, “e”:1, “f”:0, “g”:0, “h”:0}.

After encoding all violation logs, we further organize these BoW vectors as a DTM as Table I shows.

TABLE I
THE EXAMPLE OF DOCUMENT-TERM MATRIX

Violation	a	b	c	d	e	f	g	h
1	1	2	0	1	1	0	0	0
2	1	1	1	0	0	1	0	1
3	0	1	1	1	0	1	1	0

B. *Feature Engineering*

The purpose in this step is to help model learn more generalized patterns of violations. For different designs, the token composition and sequence of violations are not identical. Feature engineering extracts meaningful information from data. It makes features more informative and discriminative across different designs and increases ML models' generalization ability and robustness. Based on the observation of Isolation violations, if new signal tokens are observed, it will increase the risk of discovering true design issues. The feature called “Novelty Feature” is produced by comparing the data from new designs and historical waived violations. Furthermore, the feature called “Token-cluster Feature” is generated by performing the clustering analysis on all tokens of design signals as new features.

1. Novelty Feature: We define new tokens which do not appear in historical tokens as novelties. To locate these novelty tokens, we construct the distinct token sets from historical waived violations and new design violations and then compare the differences between them. Only tokens that appear in the new design will be defined as novelty tokens. The illustration is shown in Fig. 5. We create a binary vector that the value is 1 if there are any novelty tokens in the violation; otherwise, 0. As a result, each violation has a value to reveal its novelty status, which is illustrated in Fig.6. On the other hand, in the training data from historical designs, the novelty feature is obtained by comparing the distinct tokens of true design issues and waived violations so that the model can learn how to judge the violations with novelty tokens.

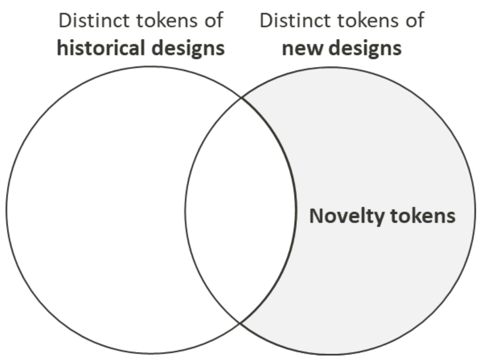
Distinct tokens of **historical designs**
Distinct tokens of **new designs**
Novelty tokens

Figure 5. The defined set of novelty tokens.

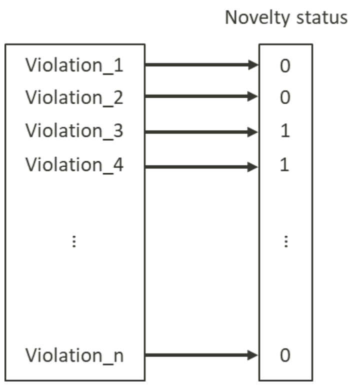

Figure 6. The illustration of novelty-token feature.

2. Token-cluster Feature: When constructing DTM for ML modeling, we use common tokens to align training and testing feature sets, but the tokens that only exist in the training or the testing dataset are invalid for modeling, and they are called non-common tokens. If we don't utilize non-common tokens, during the DTM construction, the non-common tokens with similar word structures are consequently dropped. That means we may lose important information because the tokens shared with similar word structures usually have similar physical significance. Thus, we create token-cluster features to utilize non-common tokens so that the model can learn more patterns in addition to common tokens that exist in the first place. The process of token-cluster feature generation is described as follows.

   We first encode non-common tokens to BoW vectors by segmenting each token to letters. Each dimension represents one character in alphabet. Take the word "apple" as an example, the character-level BoW vector is obtained in Table II.

TABLE II
THE EXAMPLE OF CHARACTER-LEVEL BAG-OF-WORDS VECTOR

Character	a	b	c	…	e	…	l	…	p	…	z
Vector	1	0	0	0	1	0	1	0	2	0	0

We conduct the K-means clustering algorithm [5] on these character-level BoW vectors and group all tokens with similar word structures. Each token is encoded by the cluster it belongs. Finally, we measure the occurrences of clusters for non-common tokens in each violation and generate the cluster-level BoW vectors to construct features. An example is shown in Table III.

TABLE III
AN EXAMPLE OF TOKEN-CLUSTER FEATURES

Violation	Token-cluster 1	Token-cluster 2	Token-cluster 3	Token-cluster 4
1	1	3	1	2
2	2	0	3	2
3	0	2	2	3
4	3	1	3	0
5	0	3	0	4

C. Ensemble Model

This step explains the process of model learning. Different from general natural language, design signals are quite domain-specific which make them no semantics. It's not suitable to enhance model performance via any pre-trained NLP models in current studies. Therefore, we apply XGBoost algorithm [6] as our base models and conduct the ensemble technique to best combine the predictions.

Ensemble modeling is a technique where multiple diverse models are created to predict an outcome. It enhances not only model's performance, but its generalization ability. We conduct the cross-validation to prepare datasets for model training. One round of cross-validation partitions data into complementary subsets that consist of training and testing sets. We perform n-fold cross-validation to train base models using XGBoost algorithm. In the end, there are several inferred candidate probabilities for each violation and the largest one, the violation with highest risk, is chosen to be the final prediction output. The illustration is shown in Fig. 7.

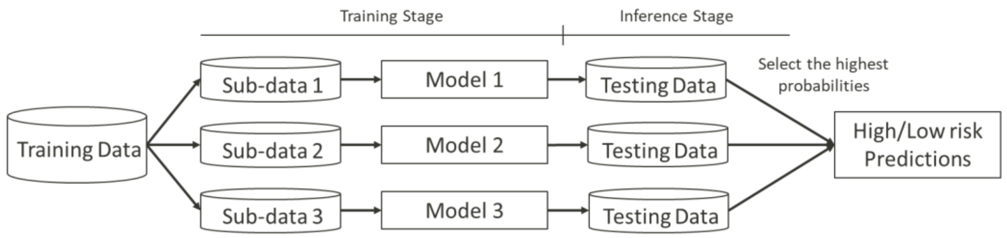

Figure 7. The illustration of ensemble modeling mechanism.

III. EXPERIMENTAL RESULTS

We applied the proposed approach to Isolation clamp value check process of 3 different low-power designs {A, B, C}. Table IV showed the total number of violations and true design issues for testing designs and the training data which were collected from the other design's Isolation check process. These violations were generated by simulation tools during project development. In addition to the Isolation check used in this empirical study, other types of checks that have violation signals consisting of tokens of text are all potentially applicable using our proposed approach.

TABLE IV
TOTAL NUMBER OF VIOLATIONS AND TRUE DESIGN ISSUES OF EXPERIMENT DATASETS

Modeling data	Training data	Testing data		
Design name	Historical designs	A	B	C
Total violations	7000	2002	1001	3177
True design issues	284	166	117	539

D. Train AI Checkers using proposed approach

The training of an AI Checker required the violations of historical and new designs to participate in the process. Three testing designs would share the same training data. In the data preprocessing stage, we compared each testing design with the training data to obtain common tokens. These common tokens were used to constrain the DTM dimensions that would be used for model training and inference. The number of words and common tokens of preprocessed training and testing data were shown in Table V.

TABLE V
NUMBER OF WORDS AND COMMON TOKENS FOR TRAINING AND TESTING DESIGNS

Design	# Words	# Common tokens
Historical designs	900	-
A	359	200
B	429	292
C	956	597

The number of common tokens used in modeling was less than the number of words in the testing data due to differences in the violation of training and testing designs. Therefore, we need feature engineering to extract more information and make efficient use of non-common tokens. In the feature engineering stage, based on the preprocessed DTM, we compared the training and testing data to generate the novelty-token feature, which occupies 1 dimension. The cluster-token features were derived from non-common tokens, and the dimensions of features were all set to 30. In the ensemble modeling stage, we used XGBoost models with the same parameter for each design. We performed 5-fold cross validation on the data and trained 5 models based on the training data subsets. Each test violation was inferred by these models and the highest probability was chosen as the prediction result. We defined 0.5 as the threshold. If the prediction probability was greater than 0.5, the violation was judged as a high-risk violation; otherwise, it was a low-risk violation. For each design, the total time of preprocessing, feature engineering, training and inference is about 3 minutes on a 2-core CPU. We used the same preprocessing procedure and model settings during the training stage.

E. Reviewing Iso Clamp Check using AI Checker

We presented the ablation study on the proposed approach to give an intuition of its behavior and performance. For reproducibility, we run each configuration over five seeds and report the average performance. We trained a Support Vector Machine (SVM) model [7], a classic machine learning algorithm for classification problems, as the benchmark base model. We compared the different performance brought by each step. Table VI shows the experimental configurations.

TABLE VI
THE EXPERIMENTAL CONFIGURATIONS FOR THE ABLATION STUDY

Model	Approach	Configuration
SVM	Data preprocessing	1
XGBoost	Data preprocessing	2
XGBoost	Data preprocessing and novelty feature	3
XGBoost	Data preprocessing, novelty feature and token-cluster feature	4
Ensemble XGBoost	Data preprocessing, novelty feature and token-cluster feature	5

We demonstrated the effectiveness of techniques with the proposed AI Checker. The recall rate denoted the ratio of the number of true design issues in the predicted high-risk violations divided by the number of total true design issues. The recall rate 1 indicated the predicted high-risk violations contain all true design issues. On the other hand, the review reduction rate represented the percentage of eliminated low-risk violations from the review process. Fig. 8 presented the results of recall rate and violation reduction rate in different designs and configurations. The proposed approach (configuration 5) outperformed other configurations and reached recall rate 1 in all testing designs, {A, B, C}. We could observe that as more methods joined in, the recall rate was closer to 1, and the review reduction rate went down. There was a trade-off between the recall and review reduction rate. Thus, we might have more false alarms to capture all true design issues. In real world application, we needed to satisfy a multi-objective scenario that considered the change in both recall and review reduction rate. In particular, the number of to-be-reviewed violations decreased while keeping the constraint that the recall rate was 1. In the end, we could achieve 50%, 12% and 18% review reduction for designs {A, B, C}.

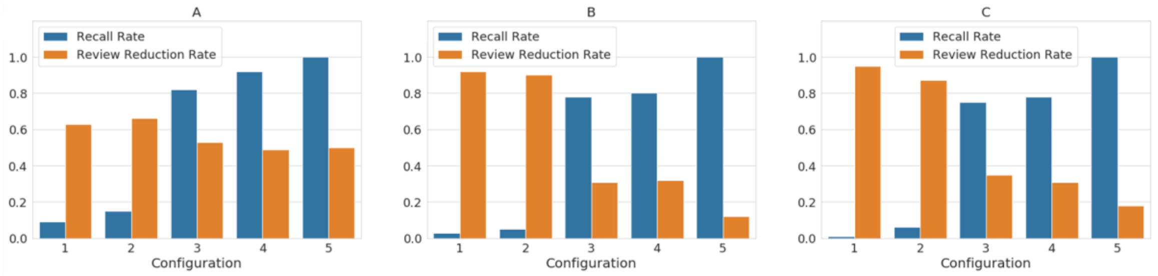

Figure 8. The recall rate and review reduction rate of different configurations for three testing designs.

IV. CONCLUSION

This work proposed an approach that applied ML techniques to assist the violation review process and presented the empirical study on the Isolation check. The result demonstrated that we could reduce at least the 12% review effort without the human expert's involvement while keeping all true design issues detected. Moreover, it helped reduce false alarms and the largest reduction of total violations reached up to 50%. It also achieved the shift left of true design issues, so domain experts could tackle the issues earlier. The result showed that the ML model could learn the experiences and knowledge of reviewing violations by domain experts from historical data. We could also train the model incrementally using accumulated historical data. Hopefully, starting from the promising experiment result of 3 designs, the proposed AI Checker can be generally applied on other design check flows in the future.

REFERENCES

[1] Chang, Norman, et al. "Machine learning based generic violation waiver system with application on electromigration sign-off." 23rd Asia and South Pacific Design Automation Conference (ASP-DAC). IEEE, 2018.

[2] Dada, E. G., Bassi, J. S., Chiroma, H., Adetunmbi, A. O., & Ajibuwa, O. E., “Machine learning for email spam filtering: review, approaches and open research problems.” in Heliyon, 5(6), e01802, 2019.

[3] Harris, Zellig S., "Distributional structure," Word 10.2-3, 1954, pp. 146-162.

[4] Salton, Gerard, Anita Wong, and Chung-Shu Yang, "A vector space model for automatic indexing," in Communications of the ACM 18.11, 1975, pp.613-620.

[5] Hartigan, J. A., & Wong, M. A., Algorithm AS 136: A k-means clustering algorithm, in Journal of the royal statistical society, series c (applied statistics), 28(1), 1979, pp.100-108.

[6] Chen, Tianqi, et al. "Xgboost: extreme gradient boosting," R package version 0.4-2 1.4, 1-4.g, 2015, pp.785-794.

[7] Drucker, H., Wu, D., & Vapnik, V. N., Support vector machines for spam categorization, in IEEE Transactions on Neural networks, 10(5), 1999, pp.1048-1054.