3-4 Automatic Modeling of Logic Device Performance Based on Machine Learning and Explainable AI

Seungju Kim[1], Kwangseok Lee[1], Hyeon-Kyun Noh[1], Youngkyu Shin[1], Kyu-Baik Chang[1], Jaehoon Jeong[1], Sangwon Baek[2], Myunggil Kang[2], Keunhwi Cho[2], Dong-Won Kim[2], Daesin Kim[1]

[1]Data & Information Technology Center, [2]R&D Center / Device Solution Business
Samsung Electronics Co., Ltd., Gyeonggi-do 18448, Republic of Korea.
* Email Address: seungjju.kim@samsung.com (S. Kim).

***Abstract*— In this paper, we propose a machine learning framework for predicting performances of semiconductor devices that can automatically reflect modifications in process conditions. While standard TCAD simulators require intensive modeling and calibration works to capture new process conditions, our proposed framework can learn these conditions from data efficiently and directly. Furthermore, by applying recently attention-getting explainable AI techniques, important factors that affecting device performances can be discovered automatically from the proposed model. Specifically, our model quantifies significance of each process step as the game-theoretic Shapley value, that cannot be achieved by TCAD simulators.**

***Keywords*— Logic device, electrical test, machine learning, explainable AI, SHAP (Shapley Additive explanation)**

I. Introduction

In the sub-10-nm technology era, various process techniques have been introduced to achieve the optimal performance of devices [1-2]. As a result, process conditions that should be estimated have increased explosively. Traditionally, TCAD simulations have been used to predict device performances under numerous process conditions. However, TCAD simulations require time-consuming procedures such as generating device structures, converting process steps as simulation models, and calibrating various parameters. Hence, they suffer from reflecting new device scheme and various modifications in process conditions within practical time scale. To overcome such limitations, in this paper, we propose a machine learning (ML) framework that can predict device performances with automatic reflections of changes in complex process conditions. In addition, the proposed frameworks can deduce important factors that determine device performances, by combining the ML models and recently-developed explainable AI (XAI) technique [3].

Fig. 1 illustratively summarizes the overall procedure of the proposed framework. The proposed framework consists of three principled stage: the data preparation (blue one), pre-processing (orange one) and modeling (green one). In the data preparation stage, raw manufacturing data is collected daily and engineered to extract some useful information for ML model, i.e., features. Then, in the pre-processing stage, the extracted features are further refined. Finally, in the modeling stage, the ML models are (re-)trained and updated with incoming the pre-processed features. Since whole steps mentioned above progress automatically, device engineers can use the model that learns the up-to-date process condition. Details on each stage will be explained in the following Section. Table I shows major differences between traditional TCAD simulations and the proposed framework. There are several merits of the proposed framework over the TCAD: automatic data collection, improved input factors based on process step ID that inaccessible with standard TCADs, e.g., new process scheme or dependency of equipment, and significantly reduced TAT.

TABLE I. Major Differences of TCAD and This Work

	TCAD	This work
Data collection	Fab data & device engineer	Database (automatic extraction)
X Factor	Device structures, process conditions, simulation parameters, etc.	Process step ID based engineering data (device structures, process conditions, equipment, etc.)
Y Factor	Device performances (AC/DC), drain-induced barrier lowering (DIBL), subthreshold swing (SS), threshold volrage (V_t), channel/external resistances ($R_{ch/ext}$), source/drain contact resistances ($R_{cnt\text{-}s/d}$), etc.	
Model	TCAD model	ML model (ensemble methods, e.g., random forest [10], gradient boosting [11], XGBoost [12], Catboost [13])
Tool (S/W)	Commercial TCAD simulator	SQL, Python
TAT	1~2 weeks (include data collection)	~10 min
Explanation	Physical formula	XAI

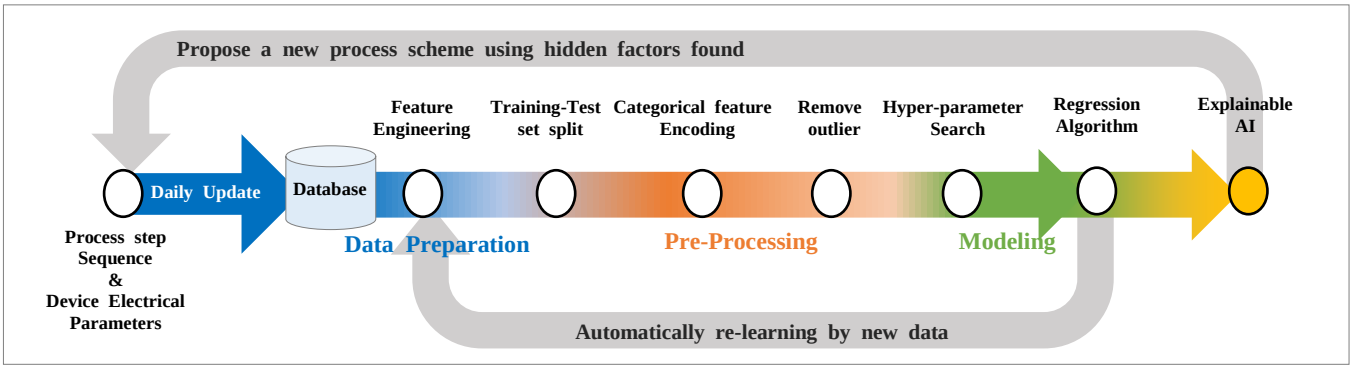

Fig. 1. Main process of automatically analyzing device performance from the process information database

II. Data Preparation

A. Collecting rawdata

Millions of process progress information is generated from thousands of equipment every day. We collect information such as execution time, facility type, facility location, process type, and process conditions for each wafer in a table format and store it in a database once a day. Process conditions are mostly simplified or encrypted for security by engineers, so additional data (feature) engineering work is required to change them to a form for use in learning.

B. Feature Engineering

Logic process information contains hundreds of process progress sequences per wafer, includes various unit process factors, facility information, and structure information. One cannot directly use this raw process information for ML models because of the following reasons: (1) when hundreds of process steps are used as a feature of the model, the model becomes excessively complicated and sometimes the predictability of the model is rather degraded. (2) The process information is encoded as strings, thus it is impossible to recognize numerical correlations between features, such as temperature and thickness changes.

We solve these problems through the feature engineering based on domain knowledge. First, we select 56 of important processes and structure information that should be learned essentially. In addition, we develop a parser that detect numeric process information encoded as string then automatically convert it into numeric form. We compare the correlation matrix to check the effect of this parsing module in Fig. 2. Because the categorical as well as numerical features are exist, we use the Theil's Uncertainty [4] and Pearson's correlation coefficient for former and latter, respectively [5]. Fig. 2 clearly shows that the correlation matrix after the parsing process is more pronounced than the unprocessed one.

III. Data Pre-processing

While the extracted features can be used in its form, they still have several problems in practical. It is mainly due to characteristics of the semiconductor process data. First, there are outliers in electrical measurement result. They distort the test model performance generally, thus should be detected and removed. Next, semiconductor wafers are grouped by lot, and process characteristics in the same lot are almost identical. In this case, dataset should be split into training and test sets appropriately to prevent the overfitting issue. Last, some data contain categorical features. To fully exploit their information, the proper encoding method should be found. For these reasons, we conduct additional data pre-processing with sufficient consideration of such data characteristics. We use the random forest regressor [10] as a baseline model for data pre-processing in this section.

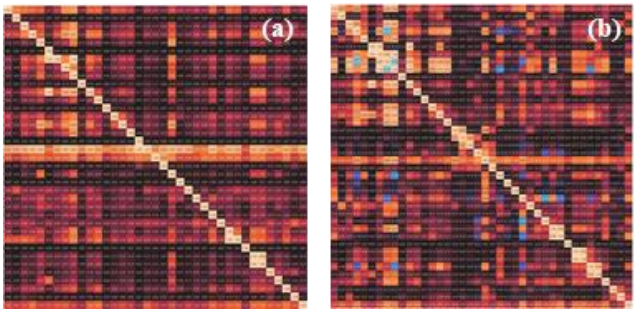

Fig. 2. Comparison of correlation matrix before and after feature engineering. (a) Before feature engineering (only string data) (b) After feature engineering (string data and numeric data)

A. Outlier detection method

In this paper, we evaluate following three outlier detection methods: Tukey method [6], Z-score (standard score) method, and modified Z-score method [7]. Table II shows the equation of each outlier detection method and the corresponding thumb-rule threshold. We summarize the evaluation result in Table III. As a result of the evaluation, the IQR-based Turkey method shows stable outlier detection and removal performances.

B. Dataset splitting method

Semiconductor wafers are grouped into lots. Generally, wafers in the same lot have almost identical process conditions. On the other hand, the process conditions can vary abruptly for wafers from different lots. Thus, if the process conditions for all lots are not regularly distributed in the training set, the predictive performance of the trained model can be degraded significantly. To deal with this problem, we include one or more labels of all types of categorical features in the training set, to ensure stable model performance. In this paper, this method is called a categorical feature-wise splitting.

Fig. 3 compares the performance distributions of models trained with random and proposed splitting methods. To extract the performance distribution, we repeat the experiment 100 times. From the figure, one can find that splitting method used in our paper shows clear improvement over the random one, by checking the mean values and variances of two distributions.

TABLE II. Outlier Detection Methods

Outlier detection methods	Equation	Thumb-rule threshold
Tukey method	IQR(InterQuartile Range) = 3rd Quartile – 1st Quartile	Upper Bound = 3rd Quartile + (IQR * 1.5) Lower Bound = 1st Quartile – (IQR * 1.5)
Z-Score	$Z\text{-score} = (x - \mu)/\sigma$	$\lvert Z\text{-score}\rvert > 3$
Modified Z-Score	$M_i = 0.6745(x_i - \tilde{x})/MAD$ $MAD = \text{median}(\lvert x_i - \tilde{x}\rvert)$	$\lvert \text{Modified Z-score}\rvert > 3.5$

TABLE III. Comparison of Outlier Detection Methods

Outlier detection methods	Model performance [a]		Standard deviation
	Train R^2	Test R^2	
Original data	0.435	-11.41	147.712
Z-Score	0.811	0.333	27.773
Modified Z-score	0.949	0.722	19.577
Tukey method	**0.954**	**0.786**	**19.312**

[a.] Model: Random forest regressor, Target Electrical Test: NMOS Contact Resistant

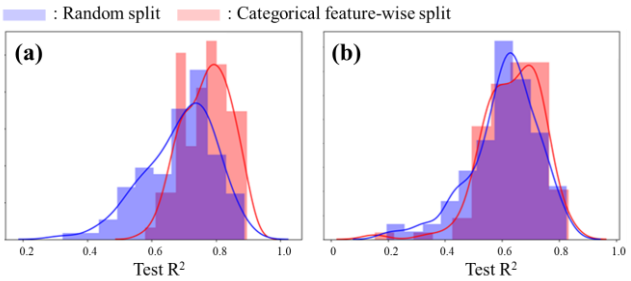

Fig. 3. Comparison of model performance (test R^2) histogram

C. *Categorical data encoding*

Semiconductor data often contains various categorical features such as equipment or material information. It is obvious that for such data the model performance highly depends on the used categorical encoding technique. In ML field, there are six representative encoding methods used frequently [8]. We compare these encoding methods to search the proper one for logic process data.

Table IV summarize the performance of random forest models with respect to the encoding method. We find that the target encoding with the categorical feature-wise data splitting shows the best performance across the various combinations. One notable thing is that the target encoding shows degraded performance than one-hot encoding when the dataset is randomly split. It is because the target encoding converts the categorical labels as a function of the target value y. Thus, in the random splitting, which contains unseen categorical features in the test dataset, the target encoding cannot properly work. However, since the categorical feature-wise splitting can regularly distribute the features, it does a better job. It reveals that the data splitting and categorical encoding methods should be considered simultaneously to find the optimal data preprocessing rule.

IV. Modeling and Results

It is noteworthy that our proposed models not only predict the device performance, but also deduce important factors that critically affect the device performance, with the help of the recently attention-getting XAI technique.

A. *Prediction model*

In this paper, we evaluate the tree-based ensemble models including random forest, gradient boosting [11], XGBoost [12], and Catboost [13]. They have various desirable properties for our target task: first, they show much reliable performance under a small amount of training data. It should be noted that the number of accessible data for semiconductor manufacturing does not exceed ~1,000 generally. In this case, complicated models such as deep neural networks show rather deteriorated performances. Second, they outperform other ML models, e.g., linear model and kernel based model, when the categorical features are included in the input [9]. Last but not least, their prediction results can be easily explained. To help the decision of device engineers, the explainability of the model, i.e., *why the model arrived at a specific prediction*, is highly preferred. The tree-based ensemble models are naturally XAI considering the interpretability of decision trees. Furthermore, some advanced techniques can be directly and efficiently applied to the tree-based ensemble models.

TABLE IV. Evaluation Results of Regression Model using 6 Types of Encoding Methods

Split type + Encoding Method	Train R^2	Test R^2
Random split + One-hot	0.791	0.724
Random split + Sum	0.852	0.603
Random split + Helmert	0.910	0.472
Random split + Target	0.831	0.640
Random split + Leave-One-Out	0.992	-0.139
Random split + Catboost	0.918	0.566
Categorical feature-wise split + One-hot	0.864	0.742
Categorical feature-wise split + target	**0.921**	**0.826**

The experimental conditions are as follows: the number of training data (wafer) and selected features (process steps and measured device structure information) are 323 and 56, respectively. For hyper-parameter tuning, we use the grid search, whose searching space are listed in Table V. We summarize the searched best model's prediction scores for AC, DC, and other electrical test (ET) parameters, in the case of n-type and p-type MOSFETs, in Table VI. It shows that ML model can achieve ~0.7 R^2 for unseen test data.

B. *Explainable AI*

XAI is one of the most active research area in recent ML fields. Frequently used XAI models include decision tree, a simple model-specific method, LIME (Local Interpretable Model-agnostic Explanations) [14], SHAP (Shapley Additive exPlanations) [3], PDP (Partial Dependence Plot), and LRP (Layer-wise Relevance Propagation).

In this paper, we use the Tree SHAP [15], that utilizing the standard kernel-based SHAP and tree-based models.

TABLE V. Hyper-parameter Set

Hyper-parameter	Parameter set
n_estimators	[10, 50, 100, 250]
max_depth	[5, 10, 20, 40]
max_features	[10, 20, 'auto', 'log2']
min_samples_leaf	[1, 0.5, 0.3, 0.1]
loss	['ls', 'lad', 'huber', 'quantile']
subsample	[1.0, 0.8, 0.6]
learning_rate	[0.01, 0.05, 0.1, 0.2]

TABLE VI. Prediction Performance of Regression Model

ET	NMOS		PMOS	
	Train R^2	Test R^2	Train R^2	Test R^2
AC performance	0.915	0.702	0.924	0.605
DC performance	0.941	0.758	0.956	0.803
$R_{cnt\text{-}d}$	0.939	0.752	0.971	0.836
$R_{cnt\text{-}s}$	0.943	0.793	0.968	0.741
R_{ch}	0.934	0.645	0.984	0.783
R_{ext}	0.973	0.683	0.962	0.871
DIBL	0.921	0.854	0.903	0.826
SS	0.931	0.597	0.947	0.649
V_t	0.967	0.831	0.908	0.672

Tree SHAP can calculate the Shapley value [16] of features accurately and efficiently. The Shapley value represent a responsibility of a feature, i.e., a process step, for a change in the model output, i.e., the predicted ET value. More specifically, it is equal to the difference between the predicted output with and without a certain feature, for a given data point. It should be noted that the Shapley value is a function of the feature value, not just a feature itself.

As an example, Fig. 4 (a) shows the Shapley value on ET value with varying the feature value of 10 representative process steps. For TREATMENT #1, as its feature value increases (from blue dot to red dot), its corresponding Shapley value also increases significantly (from -4 to 4). It clearly shows that the predicted ET value is highly depend on the TREATMENT #1. On the other hand, there is no significant variation in the Shapley value with respect to the feature value of ETCH #5. Based on this, one can find that the condition of TREATMENT #1 is relatively important for controlling ET value than that of ETCH #5. Fig. 4 (b) sorts process steps according to the averaged absolute values of the Shapley value on ET value. It plays a similar role with the feature importance, but shows game-theoretically meaningful rank than conventional impurity-based one. We find that the importance ranking derived through the Shapley value matches well with the domain knowledge of device engineers.

Furthermore, with the Shapley value, one can find the optimal process condition that cannot searched well due to its non-linear nature. For example, Fig. 5(a) shows the effect of a certain structure width on Δ Shapley value. It clearly shows that this structure condition shows non-linear effects on ET value. For this process, human engineers predicted that the ET value would decrease as the structure width increased from 12 to 14, assuming it would have linear relationship. However, using the Shapley value plot, they could find optimal ET value could be achieved around 13. Similar analysis can be performed in the case of the categorical feature, as shown in Fig. 5 (b). It shows among 9 different (categorical) conditions for certain etch process, #5, #6, #7, and #8 conditions and the others show respectively positive and negative effects on ET value.

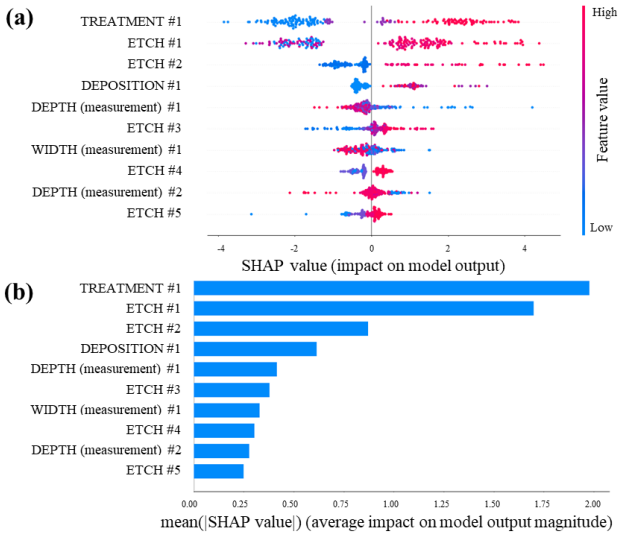

Fig. 4. Rank of process steps and structure measurement values calculated by SHAP value

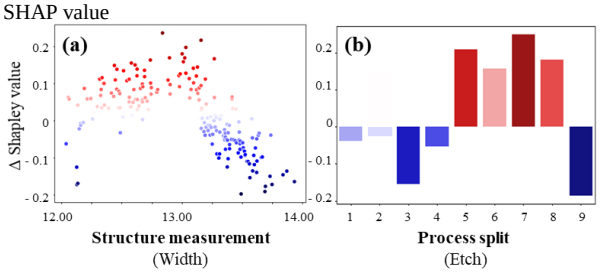

Fig. 5. Analysis of the impact on the ET according to the value of the process step and structure measurement information

V. CONCLUSION

In this paper, we propose the automatic performance modeling framework based on feature engineering, ML and XAI techniques. Our ML model utilizing information from process step ID can achieve prediction score ~0.7 test R^2. Furthermore, by applying the tree SHAP to ML model, hidden key factors that have significant impact on performances can be derived easily.

REFERENCES

[1] Ha, Daewon, et al. "Highly manufacturable 7nm FinFET technology featuring EUV lithography for low power and high performance applications." 2017 Symposium on VLSI Technology. IEEE, 2017.

[2] Bae, Geumjong, et al. "3nm GAA technology featuring multi-bridge-channel FET for low power and high performance applications." 2018 IEEE International Electron Devices Meeting (IEDM). IEEE, 2018..

[3] Lundberg, Scott M., and Su-In Lee. "A unified approach to interpreting model predictions." Advances in neural information processing systems. 2017.

[4] Press, William H., et al. Numerical recipes in Fortran 77: volume 1, volume 1 of Fortran numerical recipes: the art of scientific computing. Cambridge university press, 1992.

[5] Pearson, Karl. "VII. Note on regression and inheritance in the case of two parents." proceedings of the royal society of London 58.347-352 (1895): 240-242.

[6] Hoaglin, David C., Boris Iglewicz, and John W. Tukey. "Performance of some resistant rules for outlier labeling." Journal of the American Statistical Association 81.396 (1986): 991-999.

[7] Rousseeuw, Peter J., and Mia Hubert. "Robust statistics for outlier detection." Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 1.1 (2011): 73-79.

[8] McGinnis, William D., et al. "Category encoders: a scikit-learn-contrib package of transformers for encoding categorical data." *Journal of Open Source Software* 3.21 (2018): 501.

[9] Razi, Muhammad A., and Kuriakose Athappilly. "A comparative predictive analysis of neural networks (NNs), nonlinear regression and classification and regression tree (CART) models." Expert Systems with Applications 29.1 (2005): 65-74.

[10] Liaw, Andy, and Matthew Wiener. "Classification and regression by randomForest." R news 2.3 (2002): 18-22.

[11] Friedman, Jerome H. "Greedy function approximation: a gradient boosting machine." Annals of statistics (2001): 1189-1232.

[12] Chen, Tianqi, and Carlos Guestrin. "Xgboost: A scalable tree boosting system." Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. 2016.

[13] Prokhorenkova, Liudmila, et al. "CatBoost: unbiased boosting with categorical features." Advances in neural information processing systems. 2018.

[14] Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. "'Why should I trust you?' Explaining the predictions of any classifier." Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. 2016.

[15] Lundberg, Scott M., Gabriel G. Erion, and Su-In Lee. "Consistent individualized feature attribution for tree ensembles." arXiv preprint arXiv:1802.03888 (2018).

[16] Shapley, Lloyd S. "A value for n-person games." Contributions to the Theory of Games 2.28 (1953): 307-317.