BOXGB: Design Parameter Optimization with Systematic Integration of Bayesian Optimization and XGBoost

Chanhee Jeon*, Doyeon Won* , Jaewan Yang*, Kyu-Myung Choi† and Taewhan Kim*
*†Department of Electrical and Computer Engineering, Seoul National University, Seoul, Korea
*{cjeon7, dywon, jaewane, tkim}@snucad.snu.ac.kr, †kmchoi@snu.ac.kr

***Abstract*—Finding design flow parameters that ensure a high quality of final chip is a very important task, but requires an excessive amount of effort and time. In this work, we automate this task by proposing a machine learning (ML)-based design space optimization (DSO) framework. Rather than simply applying one ML model exclusively or multiple ones in a naïve manner, we develop a comprehensive chain of ML engines which is able to explore the design parameter space more economically but effectively to make a fast convergence on finding the best parameter set. Specifically, we solve the DSO problem in three steps: (1) random sampling of parameter sets and then performing design evaluation to produce an initial ML training dataset; (2) iteratively, downsizing parameter dimension through Principal Component Analysis (PCA) followed by sampling through *an exploration-centric mechanism* which is internally driven by Bayesian Optimization (BO) and then evaluating the sample; (3) iteratively, sampling through *an exploitation-centric mechanism* driven by XGBoost regression and then checking anomaly by using XGBoost classification followed by evaluating the sample if it's not anomaly. From our experiments with benchmark designs, it is shown that our approach is able to find design parameter sets which are far better than that found by the prior state-of-the-art ML-based approaches, even with fewer number of design evaluations (i.e., EDA tool runs). In addition, in comparison with the designs produced by using the default parameter setting, our DSO framework is able to improve the design PPA metrics by 5∼30%.**

I. Introduction

Finding a good design parameter setting is one of the utmost important tasks that significantly affect the performance, power, and area (PPA) of the final chip. Though engineers spend long time to find a design parameter set (i.e., tuned parameter values) for optimal PPA solution, it is very difficult to find it because a design process is performed in a sequence of multiple steps and each step has lots of its own input parameters to be tuned for PPA optimization. Recently, there have been researches that address this issue by Machine Learning (ML)-based design space optimization (DSO).

The work in [1] utilized the ML tools that have been used for tuning the hyper-parameters of artificial intelligence (AI) applications to tune the input parameters of OpenROAD physical design tools in [2]. In addition, the work in [3] utilized the ML models in AutoML platform to build up the DSO engine in its DTCO framework called PROBE2.0. Though the works in [1], [3] have attempted to apply many sorts of ML models, their performance is unsatisfactory since the models are trained with small size dataset due to the limited number of tool runs.

The other group of research is to develop a domain-specific DSO system that targets the input parameter tuning on physical design. The DSO system called *SynTunSys* (*STS*) in [4] used a genetic algorithm for parameter tuning. The limitation of STS is that it requires the designer's knowledge to construct primitives and scenarios, which are defined as special parameters for assisting the DSO process. The work in [5] proposed a Reinforcement Learning (RL)-driven approach which used synchronous advantage actor-critic agents to transform the parameter tuning change into action space to navigate toward optimal tuning. However, it requires a large number of tool runs for RL model training. The work in [6] proposed XGBoost based DSO system called *FIST*, in which it clustered parameter sets of the samples based on the importance of individual parameters, and then iteratively generated input parameter sets (i.e., new samples) of near-optimal PPA by building a set of cluster-based decision trees in XGBoost. However, acquiring a reliable measure on the parameter importance to be used for selecting pivotal parameters in clustering requires a considerable amount of physical design tool runs. More importantly, since FIST did not consider the interrelation between parameters when assessing their importance, it may produce samples which are almost redundant, thus wasting time in evaluating the samples of almost the same PPA by running EDA tools.

The DSO problem belongs to the design space exploration problem. Two prominent features in the DSO problem are (1) *a large number (≥ 10) of design parameters* and (2) *a long running time of EDA tools to evaluate design per parameter set*. *Feature 1* implies that the design space is enormous. For example, Table I shows a list of 12 distinct parameters together with their possible values, resulting in a design space with more than 4,000,000 parameter sets. On the other hand, *Feature 2* indicates that the number of parameter set samples to evaluate design PPA should be constrained to a certain limit due to the issue of tool licensing and tight implementation schedule. Consequently, it is essential to sample a limited number of design parameters in the course of iterative ML-training and then sampling in a way to quickly converge to an optimal parameter set.

We classify the ML-based samplings of design parameter set

979-8-3503-4859-0/DATE24/© 2024 EDAA

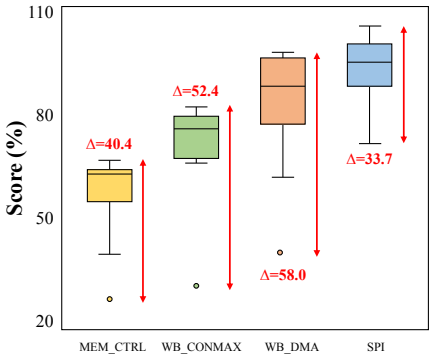

Fig. 1. Box plot of optimization scores in *Eq. 3* for Opencore designs with XGBoost regressor. The box, the horizontal line in the box, and the isolated dot represent the range between the first and third quartiles, median, and outlier for each design, respectively.

into two types: (1) *exploration-centric* and (2) *exploitation-centric*. *Type 1* refers to the samples, predicted by the ML engine, which are likely to be a potential improvement in the regions of high prediction uncertainty while *Type 2* refers to the samples which are likely to be a potential improvement in the regions of high prediction certainty. Our key idea to effectively address the DSO problem is to produce samples by applying different ML engine that is best suited to each type. For example, as shown in Fig. 1, the sole use of XGBoost for optimization with a naïve initial training dataset causes a high variance in the resultant PPA scores, which reveals that it is desirable to separate the sampling process into exploration-centric and exploitation-centric, and apply the most effective ML models individually. In summary, the contributions of this work are:

1. We employ the Bayesian Optimization (BO) technique to sample exploration-centric parameter sets. We internally adjust the weighting parameter in the acquisition function in BO to sample parameter sets in the regions of high prediction uncertainty.
2. Since some of design parameters have a high correlation, we downsize the parameter dimension by applying Principal Component Analysis (PCA) [7] to the sampled design parameter sets before predicting a new sample. This can greatly reduce the design space to explore.
3. We employ the XGBoost regression technique to sample exploitation-centric parameter sets. To this end, we combine the XGBoost regression engine with XGboost classification to iteratively filter the anomaly samples during the sample generation process.
4. Our DSO framework embedding items 1, 2, and 3 is able to produce design implementations with on average 8 percentage points improved PPA over that by the best prior ML-based DSO approach, and with 5∼30% improved PPA over that by the default parameter setting.

II. Preliminary and Motvation

A. XGBoost

Decision tree is commonly used in the tree-based searching algorithms and performs well in both classification and regression. In the DSO field, many of design parameters have discrete values, which makes the decision problem suitable for modeling by decision trees. *Ensemble* is a representative method which is used to improve the performance of decision trees while preventing overfitting. It is usually adopted by the random forest and gradient boosting models. The gradient boosting model (GBM) shows low-biased predictions by sequentially supplementing the tree models with the errors in prior tree models. XGBoost, which is a powerful and popular method in GBM, shows a remarkable performance in both training speed and efficiency [8]. Fig. 2 shows an example of XGBoost tree for regression. Through recursive construction and ensemble of XGBoost trees, an XGBoost model gets trained to be used for prediction.

Index	Age	Height	Ground Truth
1	10	130	80
2	21	158	75
3	28	175	100
4	55	162	89
5	32	151	95

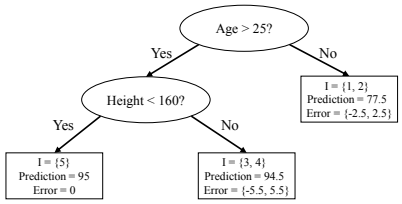

Fig. 2. Single tree structure in XGBoost regression. The predicted score at each leaf node is the mean value of the ground truths of the indices in that node. The error values are used as ground truths for the construction of next trees.

B. Bayesian Optimization

Bayesian optimization (BO) [9] is a method to find a global optimal solution of a given objective function and considered as a principal algorithm for hyper-parameter optimization. As mentioned previously, the design parameter optimization in DSO problem can be characterized by enormous design parameter space and long evaluation time. Thus, BO, which shows high performance in hyper-parameter optimization, can also be adopted in the DSO field.

BO consists of four steps: initial parameter generation, construction of a surrogate model, acquisition function calculation in surrogate model, and evaluation. The main part of BO is the construction and refinement of surrogate model, usually with multivariate Gaussian distribution. After the construction of surrogate model, the expected improvement (EI) function is generally adopted as an acquisition function. The EI function under Gaussian process (GP) model can be expressed as:

$$\mathrm{EI}(\mathbf{x}) = (\mu(\mathbf{x}) - f(\mathbf{x}^*) - \xi)\psi(\mathbf{z}) + \sigma(\mathbf{x})\phi(\mathbf{z}), \quad (1)$$

$$\mathbf{z} = (\mu(\mathbf{x}) - f(\mathbf{x}^*) - \xi)/\sigma(\mathbf{x}) \quad (2)$$

where $\mu(\mathbf{x})$ and $\sigma(\mathbf{x})$ are respectively the mean and standard deviation of GP's posterior prediction model at sample $\mathbf{x}$. $\psi(\mathbf{z})$

and $\phi(\mathbf{z})$ are respectively the CDF and PDF of the standard normal distribution. $f(\mathbf{x}^*)$ is the value of the best sample so far.

It should be noted that ξ in *Eq.1* can be used to trade the amount of exploration-centric samples with that of exploitation-centric ones. By increasing the ξ value, the importance of improvements predicted by the GP posterior mean $\mu(\mathbf{x})$ decreases relative to the importance of potential improvements in regions of high prediction uncertainty, represented by large $\sigma(\mathbf{x})$ value.

C. Principal Component Analysis

When manipulating the physical design tool parameters, many parameters have a small number of options such as binary or ternary values. Each parameter counts up one dimension to the Gaussian distribution constructed during BO, and the large size dimension with a small number of samples lowers the precision of Gaussian distribution. Thus, it enables BO degradation in the prediction accuracy. To overcome this limitation, we try to merge the closely related parameters by using Principal Component Analysis (PCA) [10].

PCA is a powerful technique for dimension reduction by projecting the high-dimensional data onto a small number of principal components. PCA has the advantage of preserving the distribution of original data as much as possible while performing the dimension reduction efficiently when the number of samples is small.

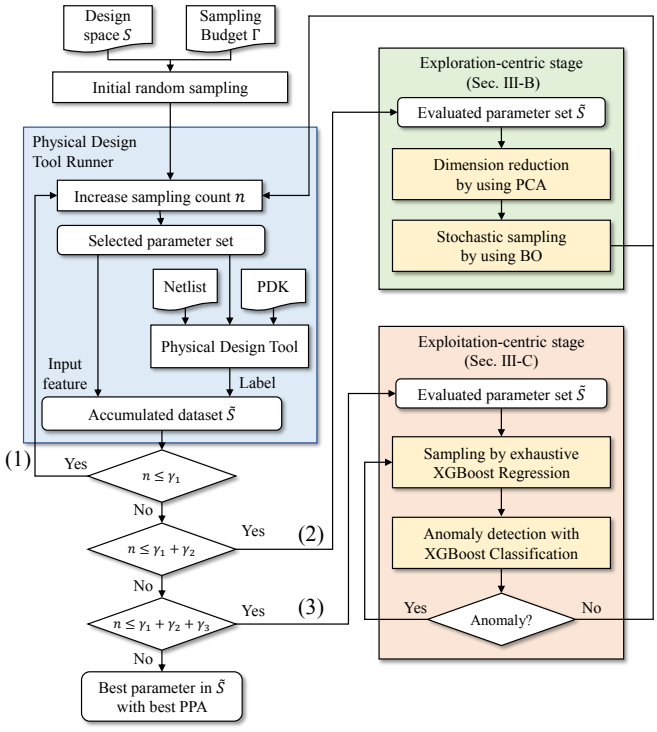

Fig. 3. The overall flow of our proposed DSO framework.

D. Anomaly Detection

Anomalies refer to the samples such that their prediction values by XGBoost regression are significantly different from the actual value obtained from evaluation. Performing regression with a small-size dataset is the major cause of anomalies. Too many anomalies misguide subsequent sampling and optimization. Hence, the iterative XGBoost refinement flow requires a process to stabilize the optimization by eliminating the potential anomalies occurred by the XGBoost regression.

On the other side, it has been shown that XGBoost classification model is very good at anomaly detection [11]. Thus, Our DSO framework combines XGBoost classification model with XGBoost regression model to detect anomaly samples predicted XGBoost regression.

III. The Proposed Design Space Optimization Framework

Fig. 3 shows the overall flow of our DSO framework. It consists of three main modules: (blue box) physical design tool runner, (green box) exploration-centric sampling, and (pink box) exploitation-centric sampling. For a given sampling budget[1], Γ, which is split into γ_1, γ_2, and γ_3 such that $\gamma_1 + \gamma_2 + \gamma_3 = \Gamma$, our framework performs the following. (1) *initial sampling*: performing the blue box γ_1 times and building up initial dataset for ML training; (2) *exploration-centric sampling*: iteratively performing the green box followed by the blue box to sample γ_2 exploration-centric parameter sets and their design evaluation; (3) *exploitation-centric sampling*: iteratively performing the pink box followed by the blue box to sample γ_3 exploitation-centric parameter sets and their evaluation. Consequently, the sampling space $\tilde{S}$ with γ_1 samples obtained in Step 1 is expanded by including γ_2 samples predicted in Step 2 by PCA-BO, and then further expanded by including γ_3 samples predicted by XGBoost regression in Step 3. Whenever a new sample is added in $\tilde{S}$, the corresponding ML engine in Step 2 or 3 is updated accordingly.

A. Objective Function

The goal of our DSO framework is to find a design parameter set that achieves an optimal PPA. The objective function, *Score*, to be maximized is expressed as:

$$Score = \sum_{p \in \mathcal{P}} Score_p \times w_p; \quad Score_p = \frac{\mathcal{C}(p)_{ref} - \mathcal{C}(p)_{current}}{\mathcal{C}(p)_{ref}}$$
$$\mathcal{P} = \{\text{Power, WNS, TNS, Area, WL}\} \tag{3}$$

where $\mathcal{C}(p)_{ref}$ and $\mathcal{C}(p)_{current}$ are the metric p's values of the reference design and the design produced by our current parameter set, respectively. WNS/TNS and WL indicate the metrics of worst/total negative time slacks and wirelength, respectively. w_p is a weighting factor.

[1]We set Γ = 100 for each testcase in our experiments. It means we ran the expensive design flow tool-chain no more than 100 times.

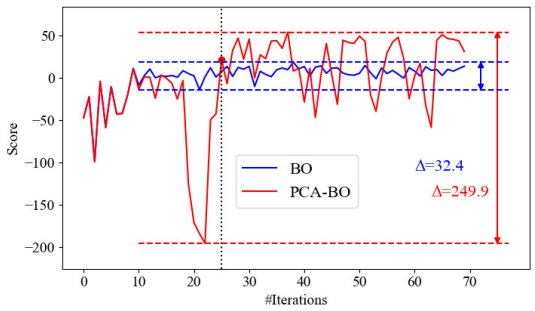

Fig. 4. Score trace of PCA-BO and BO for MEM_CTRL. PCA-BO exceeds the maximum point of BO in iteration 25 and explores the parameter sets with a wider range of scores.

B. Exploration-centric Sampling

We use BO technique for the exploration-centric sampling. Fig. 4 shows the score distributions of the samples predicted by BO alone (blue trace) and by our PCA-assisted BO (red trace), applied to testcase MEM_CTRL. The comparison of the score traces clearly shows that our PCA-assisted BO explores more diverse samples than BO alone does. This effect is caused by the following two factors in our PCA-assisted BO.

1. (*downsizing dimension*) We apply PCA to the sampled dataset to fully or partially merge the design parameters with high correlation. This process enables to reduce the size of exploration space.
2. (*placing more importance on space exploration*) We increase the value of control parameter ξ in *Eq.1* to place more emphasis on predicting samples with potential improvement in the regions of high prediction uncertainty.

Note that in our framework, PCA intervenes twice at each iteration of exploration-centric sampling: before updating surrogate model in BO and before computing EI function in *Eq.1*. Updating the surrogate model requires the accumulated dataset $\tilde{S}$. Since we want to reduce the design space, we update the surrogate model with the PCA-applied dataset to $\tilde{S}$. Likewise, rather than computing EI with the accumulated dataset $\tilde{S}$, we apply PCA to $\tilde{S}$ to reduce the dimension size and then compute EI.

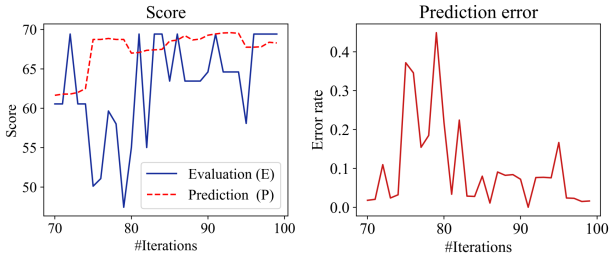

Fig. 5. Score and prediction error ($\frac{|E-P|}{E}$) of XGBoost refinement flow for AC97. The actual score trace fluctuates while XGBoost predicts the optimization process to stably reach high scores.

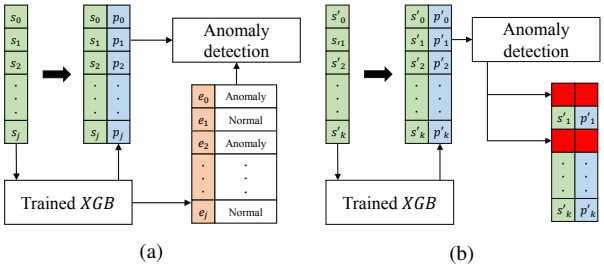

Fig. 6. (a) Training process. (b) Masking process of anomaly detection model with XGBoost regression model. After training the anomaly detection model, we mask the samples (i.e., parameter sets) with potential risk of anomaly (marked with red boxes).

C. Exploitation-centric Sampling

XGBoost regression generally shows a high prediction accuracy, in which the regularization terms play an important role in preventing overfitting. However, XGBoost regression in DSO domain often fails to predict the scores (i.e., very low scores), mainly because of the small training dataset. In addition, the exploitation-centric sampling aims to search for samples of better quality based on prior sampled data, not to enlarge the scope of evaluation, which was done by the exploration-centric sampling. As shown in Fig. 5, the prediction failure of XGBoost regression leads to unnecessary tool execution. Thus, we need to devise a mechanism to filter out the failed samples predicted by XGBoost regression.

Since the prediction failures by XGBoost regression do not occur frequently, they can be considered as anomalies in XGBoost regression model. Despite the sporadic occurrence of anomalies, a small number of meaningless tool executions are critical since the number is also taken out from the sampling budget Γ.

To avoid a potential risk in optimization, we implement an anomaly detection model to preemptively filter out the samples of parameter sets that are likely to cause anomalies in prediction. Fig. 6 shows the training and utilization steps of our anomaly detection model. We construct the anomaly detection model by using XGBoost classification model in which the input features are the input and output of our XGBoost regression. That is, we concatenate the input feature and the predicted score of XGBoost regression to be used as an input feature vector to our anomaly detection model. The labels of anomaly detection model are the prediction error of XGBoost regression. By setting the RMSE of XGBoost regression to a threshold, we classify the predicted errors over the RMSE value as anomalies. For example, if XGBoost regression with RMSE of 0.5 predicts a score for the ground truth of 10, any prediction value out of [9.5, 10.5] is classified as anomaly.

IV. Experimental Results

A. Experiment Setup

We tested our DSO framework on 8 Opencore [12] circuits. Since physical design flow is a very time-consuming process,

TABLE I
DESIGN PARAMETERS. BOLD OPTIONS ARE THE DEFAULT.

Parameter type	Parameter name	Options
Design	Max fanout	4, 8, **16**, 32, 64
	Max transition(%)	5, 10, **15**, 20, 25
	Max capacitance(fF)	5, **10**, 15, 20
Placement	Max density	80, 85, 90, 95, **-1**
	Wirelength opt level	none, **medium**, high
	Clock power driven	**low**, standard, high
	Congestion effort	low, medium, high, **auto**
	Uniform density	true, **false**
	Legalization inst. gap	**0**, 1, 2
Opt-design	DRC margin	-0.2, -0.1, **0**, 0.1, 0.2
	Power effort	**none**, low, high
	Reclaim area	true, false, **default**

we focused on optimizing the placement stage before implementing it through the whole execution of the physical design flow. With ASAP 7nm library [13], we synthesized each design once and repeatedly applied placement to test the optimization capability. The weights in the score function are empirically determined, and the weights of HPWL, WNS, TNS, power, and area are set to 5, 0.05, 0.05, 1, and 1, respectively. To check the improvements in timing with the adjustment of the tool parameters, the clock periods were carefully selected to incur a moderate level of negative slacks. Utilizations for entire designs were set to 80%. The reference HPWL and PPA mentioned in Table IV are the results of placements with default parameter setting.

Table I shows the parameter options we used for optimization. The comprehensive parameter space consists of 4,860,000 parameter sets, and we implemented our DSO system with 100 (= Γ) placements, which is 0.002% of the total parameter space. Starting with 10 (= γ_1) placements for initial random sampling, we allocated 60 (= γ_2) and 30 (= γ_3) placements for the exploration-centric sampling and the exploitation-centric sampling, respectively. We used *Synopsys Design Compiler* to synthesize all benchmarks and *Cadence Innovus* for placement.

B. Incremental Integration Test

Table II summarizes the statistics of the incremental integration test of our proposed modules: BO, PCA-BO, and anomaly detection, respectively. We tested all systems with 10 trials for each design.

When BO is utilized as exploration-centric module before the iterative XGBoost refinement flow, the mean score is decreased due to the decrease in minimum score. On the other hand, when PCA is adopted to assist BO in the exploration, there is a significant improvement in both mean and minimum scores. This clearly shows the implementation by BO only is not effective with the parameters of a small number of options, and PCA becomes an enabling element to overcome the limitation. Lastly, the score improves with the augmentation of XGBoost with the anomaly detection model, showing that the anomaly detection model stabilizes the exploitation-centric sampling. Improving mean score by 8.7% and minimum score by 64.0%, we decided to systematically integrate PCA, BO, XGBoost, and anomaly detection models to build up our DSO framework.

C. Comparison with Prior State-of-the-Art Work

Table III summarizes the score statistics of STS [4], FIST [6], and ours. Due to the variation in the results in FIST and our system, they were tested with 10 trials while STS was tested once due to the deterministic characteristic. The flows utilizing XGBoost, XGB Only, FIST, and ours, were tested with 100 placements while STS was tested with 132 placements. The percentage represents the relative ratio over the tool utilizing XGBoost only, considering it as the baseline.

First, the mean scores of our system outperform those of other systems for all 8 benchmarks as shown in the graph, and the average of mean scores is 8 percentage points higher than that by FIST. Second, due to the non-deterministic nature of iterative XGBoost refinement flow and FIST, their minimum scores are lower than STS score whereas their mean scores are higher than STS score. This means XGB only and FIST have possibility of producing worse results than STS. However, even the minimum score of our system is higher than STS score, which means our system always reduces the fluctuation range and produces higher scores significantly.

D. Design PPA Improvements

Table IV shows design PPA improvements by our DSO system compared to that by the default parameter setting. Our system improves all PPA metrics, which are HPWL, power, WNS, TNS, and area, by 10.1%, 4.7%, 23.0%, 29.8%, and 7.8%, respectively. This indicates that our system can be reliably used as DSO tool.

V. CONCLUSION

In this work, we proposed a new DSO framework which was able to effectively and efficiently solve the two conflicting factors in DSO problem, which are a very limited number of sampling and a huge design space. Specifically, we solved the DSO problem in three sequential steps: (1) random sampling of parameter sets and then performing design evaluation to

TABLE II
SCORE OF INCREMENTAL INTEGRATION TEST. THE PERCENTAGES ARE THE RELATIVE CHANGE COMPARED TO THAT OF THE ITERATIVE XGBOOST REFINEMENT FLOW (XGBOOST ONLY).

Framework name		XGB only	BO / XGB	PCA-BO / XGB	PCA-BO / XGB + Anomaly
Modules	PCA			O	O
	BO		O	O	O
	XGBoost	O	O	O	O
	Filter				O
Designs		Scores			
SPI		91.2	95.5	97.1	99.2
MEM_CTRL		56.0	54.6	60.5	60.9
AC97		65.9	58.8	67.6	67.8
WB_CONMAX		68.9	70.2	77.5	78.5
Average	Mean	70.5	69.8 (1.0%↓)	**75.7 (7.3%↑)**	**76.6 (8.7%↑)**
	Min	43.5	38.0 (12.8%↓)	**69.1 (58.8%↑)**	**71.4 (64.0%↑)**
	Max	80.0	80.5 (0.6%↑)	79.4 (0.8%↓)	81.7 (2.1%↑)

TABLE III
SCORE COMPARISON WITH THE CONVENTIONAL STATE-OF-THE-ART DSO SYSTEMS. THE SCORES OF ALL SYSTEMS ARE COMPARED WITH THOSE OF ITERATIVE XGBOOST REFINEMENT FLOW (XGBOOST ONLY).

Designs		XGB Only (Γ = 100)	STS [4] (Γ = 132)	FIST [6] (Γ = 100)	Ours (Γ = 100)
SPI		91.2	89.4	91.4	99.2
MEM_CTRL		56.0	48.6	59.6	60.9
AC97		65.9	51.1	63.0	67.8
WB_CONMAX		68.9	63.1	74.4	78.5
DES3		55.3	49.3	53.0	55.8
WB_DMA		81.5	79.1	83.7	92.8
SHA256		36.3	26.5	37.2	43.7
AES_CIPHER		21.8	20.6	21.5	23.2
Average	Mean	59.6	53.5	60.5 (1.4%↑)	**65.2 (9.4%↑)**
	Min	37.0		46.8 (26.5%↑)	**59.6 (61.1%↑)**
	Max	68.5		67.3 (1.8%↓)	**70.1 (2.3%↑)**

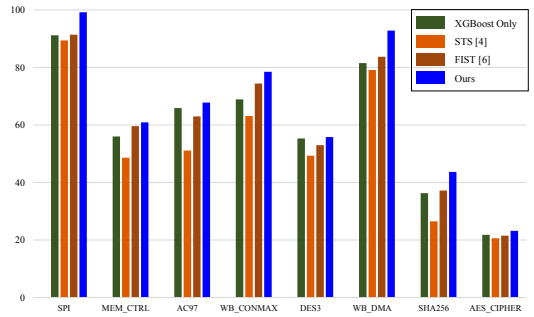

TABLE IV
PPA COMPARISON OF THE DESIGNS BY DEFAULT PARAMETER SETTING IN TABLE I AND OURS. THE PERCENTAGES IN THE LAST ROW ARE THE AVERAGE OF THE IMPROVEMENTS.

Design	Default parameter setting					Optimized parameter setting (Ours)				
	HPWL(μm)	Power(mW)	WNS(ps)	TNS(ps)	Area(μm^2)	HPWL(μm)	Power(mW)	WNS(ps)	TNS(ps)	Area(μm^2)
SPI	20363	1.2	-19	-4540	3450	17220 (15.4% ↓)	1.1 (8.3% ↓)	-4 (78.9%↑)	103 (102.3% ↑)	3222 (6.6% ↓)
MEM_CTRL	67214	3.1	-24	-6440	12187	60255 (10.4% ↓)	3.0 (3.2% ↓)	-19 (20.8% ↑)	-4466 (30.7% ↑)	11608 (4.8% ↓)
AC97	95559	14.5	-24	-5673	20897	86007 (10.0% ↓)	13.8 (4.8% ↓)	-26 (8.3% ↓)	-5846 (3.0% ↓)	18168 (13.1% ↓)
WB_CONMAX	296546	5.3	-25	-5845	30618	265267 (10.5% ↓)	4.8 (9.4% ↓)	-8 (68% ↑)	961 (83.6% ↑)	27523 (10.1% ↓)
DES3	129836	147.0	-27	-7376	59989	117429 (9.6% ↓)	144.5 (1.7% ↓)	-23 (14.8% ↑)	-5505 (23.4% ↑)	57481 (4.2% ↓)
WB_DMA	33715	3.2	-29	-7273	6404	28756 (14.7% ↓)	3.1 (3.1% ↓)	-27 (6.8% ↑)	-6313 (13.2% ↑)	5550 (13.3% ↓)
SHA256	117317	5.4	-20	-4598	19519	108405 (7.6% ↓)	5.2 (3.7% ↓)	-30 (50% ↓)	-7826 (70.2% ↓)	18277 (6.4% ↓)
AES_CIPHER	140028	3.0	-32	-7214	21165	136869 (2.3% ↓)	2.9 (3.3% ↓)	-15 (53.1% ↑)	-3169 (56.1% ↑)	20406 (3.6% ↓)
Improvement	-					**10.1% ↓**	**4.7% ↓**	**23.0% ↑**	**29.8% ↑**	**7.8% ↓**

produce an initial ML training dataset; (2) downsizing parameter dimension through Principal Component Analysis and then sampling through an exploration-centric mechanism which is internally driven by Bayesian Optimization; (3) subsequently, sampling through an exploitation-centric mechanism driven by XGBoost regression and then checking anomaly by using XGBoost classification. Through experiments, it was confirmed that our system outperformed the best prior work, improving PPA by 8 percentage points on average. In addition, our system was able to improve PPA by 5∼30% over that by the default parameter setting.

ACKNOWLEDGMENT

This work was supported in part by Samsung Electronics Company, Ltd. under IO221227-04376-01, in part by Samsung Advanced Institute of Technology (SAIT) under IO230223-05124-01, in part by the National Research Foundation of Korea (NRF) grant funded by the Korea Government (MEST) under 2021-R1A2C2008864, in part by the Institute of Information and communications Technology Planning and Evaluation (IITP) grant funded by Korea government (MSIT) under 2021-0-00754, in part by Software Systems for AI Semiconductor Design and the Artificial Intelligence Semiconductor Support Program to nurture the best talents (IITP-2023-RS-2023-00256081) grant funded by the Korea government (MSIT), in part by National R&D Program through NRF by Ministry of Science and ICT under 2020M3H2A1078119, and in part by the BK21 Four Program of the Education and Research Program for Future ICT Pioneers, Seoul National University. The EDA tool was supported by the IC Design Education.

REFERENCES

[1] J. Jung, A. B. Kahng *et al.*, "Metrics2. 1 and flow tuning in the ieee ceda robust design flow and openroad iccad special session paper," in *ICCAD*, 2021.
[2] T. Ajayi and D. Blaauw, "Openroad: Toward a self-driving, open-source digital layout implementation tool chain," in *Proceedings of Government Microcircuit Applications and Critical Technology Conference*, 2019.
[3] C.-K. Cheng, A. B. Kahng *et al.*, "Probe2. 0: A systematic framework for routability assessment from technology to design in advanced nodes," *IEEE TCAD*, vol. 41, no. 5, pp. 1495–1508, 2021.
[4] M. M. Ziegler *et al.*, "A synthesis-parameter tuning system for autonomous design-space exploration," in *DATE*, 2016.
[5] A. Agnesina, K. Chang, and S. K. Lim, "Vlsi placement parameter optimization using deep reinforcement learning," in *ICCAD*, 2020.
[6] Z. Xie, G.-Q. Fang *et al.*, "Fist: A feature-importance sampling and tree-based method for automatic design flow parameter tuning," in *ASPDAC*, 2020.
[7] E. Raponi, H. Wang *et al.*, "High dimensional bayesian optimization assisted by principal component analysis," in *PPSN 2020*. Springer, 2020, pp. 169–183.
[8] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *KDD*, 2016, pp. 785–794.
[9] J. Bergstra *et al.*, "Algorithms for hyper-parameter optimization," *Advances in neural information processing systems*, vol. 24, 2011.
[10] R. Bro and A. K. Smilde, "Principal component analysis," *Analytical methods*, vol. 6, no. 9, pp. 2812–2831, 2014.
[11] J. Henriques *et al.*, "Combining k-means and xgboost models for anomaly detection using log datasets," *Electronics*, 2020.
[12] C. Albrecht. IWLS 2005 benchmarks. [Online]. Available: :http://www.iwls.org/iwls2005/benchmarks.html
[13] L. T. Clark, V. Vinay *et al.*, "Asap7: A 7-nm finfet predictive process design kit," *Microelectronics Journal*, 2016.