Unveiling the Black-Box: Leveraging Explainable AI for FPGA Design Space Optimization

Jaemin Seo, Sejin Park, and Seokhyeong Kang
Department of Electrical Engineering, Pohang University of Science and Technology
Pohang, Republic of Korea
{seojm, sejinpark, shkang}@postech.ac.kr

***Abstract*—With significant advancements in various design methodologies, modern integrated circuits have experienced noteworthy improvements in power, performance, and area. Among various methodologies, design space optimization (DSO), which automatically explores electronic design automation (EDA) tool parameters for a given design, has been extensively studied in recent years. In this study, we propose an approach to fine-tuning an effective FPGA design space to suit a specific design. By utilizing our ML-based prediction and explainable artificial intelligence (XAI) approach, we quantify parameter contribution scores, which reveal the correlation between each parameter and the final timing results. Using the valuable insights from the parameter contribution scores, we can refine the design space only with effective parameters for subsequent timing optimization. During the optimization, our framework improved the maximum operating frequency by 26% on average in six test designs. To accomplish this, our framework required even 47% fewer FPGA compilations than the baseline, demonstrating its superior capacity for achieving fast convergence.**

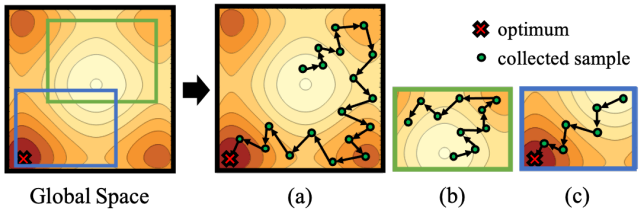
***Index Terms*—FPGA EDA, design space optimization, ML-based prediction, eXplainable Artificial Intelligence (XAI)**

I. Introduction

Field-programmable gate arrays (FPGA) have become an increasingly popular choice for implementing digital systems due to their reconfigurability, high performance, and low cost. However, as the potential of FPGAs has expanded, the complexity of their design process has also increased significantly, mainly attributed to the sheer abundance of available design options. Integrated circuit (IC) designers now face the challenge of fine-tuning tool parameters to meet their desired specifications, leading to an intricate web of decisions. In other words, even after the register-transfer level design is completed, another formidable challenge arises for IC designers: reducing the extremely large synthesis, place, and route (SP&R) design space. Furthermore, the challenge lies not only in the numerous tool options but also in the unpredictable and chaotic behavior exhibited by electronic design automation (EDA) tools [1]. As the SP&R algorithms and EDA tools continue to advance in complexity, their behavior becomes less predictable, leading to prolonged turnaround times for chip production.

To address this issue, design space optimization (DSO) methodologies are actively being studied in recent years [2] - [7]. DSO represents the effort of optimizing the design by exploring various tool parameter configurations and selecting the one that best satisfies the design criteria and constraints. DSO involves performing iterative SP&R runs and analyses in order to improve the quality of results (QoR).

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT). (No. RS-2023-00222085, Development of memory module and memory compiler for non-volatile PIM). Additionally, we thank Tom Spyrou from Precision Innovations Inc. and Scott Whitty and his folks from Intel Corp. for sharing valuable insights and discussions related to this work.



Design Space	(a)	(b)	(c)
Size of space	Large	Small	Small
Existence of optimum	Exist	Not exist	Exist
Convergence	Hard	Easy	Easy

Fig. 1. Comparison of optimization performances of three different design spaces. Design space (a) is too wide to find the optimum in a short time. (b) is small enough to converge easily, but the optimum itself does not exist in it. Only with the design space (c), we can achieve the optimum easily, and it demonstrates the importance of a fine-tuned design space. The gray-highlighted cells indicate the advantages.

Despite the significant advances in the DSO field, they commonly lack a crucial consideration, e.g., the importance of a *fine-tuned design space*. A design space encompasses all the design parameters and their possible combinations that can be tuned to optimize an IC. Although the design space indeed decides the quality of optimization (as illustrated in Fig. 1), many previous studies commonly apply a *general design space* for entirely different designs, which can overlook critical parameters and lead to suboptimal solutions.

For this reason, refining the design space suited for each specific design is the fundamental step before optimization. In this study, we propose a novel approach for determining the effective tool parameters to be included in the design space. At this point, we need to interpret the "chaotic behavior" of EDA tools to figure out the correlation between the parameters and the final QoR. To perform that, we adopt explainable artificial intelligence (XAI), which unveils the black-box in ML-based prediction and translates it into a white-box [8]. Eventually, with a few FPGA compilations, we estimate the contribution score of each parameter and construct an effective design space that is refined for a specific design. As a result, it becomes possible to enhance the performance of optimization and obtain better QoR. Particularly, in our study, we focus on timing optimization of FPGA-based designs among various optimization goals. Our main contributions are as follows:

- We propose a framework that refines a design space for FPGA timing optimization, considering the impact of

979-8-3503-4859-0/DATE24/© 2024 EDAA

various parameters on the final timing result. To the best of our knowledge, this is the first study refining an effective design space.
- We introduce two-stage contribution score estimation to assess the contribution score of each tool parameter. We implement our idea using XAI, which explains the relevance between the input features and the output of an ML-based prediction.
- Leveraging the design space generated by our framework, we improved the maximum operating frequency of six test designs by 26% on average. Furthermore, we demonstrated that not only higher operating frequency but also faster convergence can be achieved using our refined design space, requiring fewer FPGA compilations.
- Our framework can be applied to future DSO studies since it is an intuitive comprehension of the impact of each tool parameter on the quality of optimization. By focusing on the influential parameters, the proposed framework can guide toward promising regions of the design space, improving the overall efficiency of the IC design.

The remainder of this paper is organized as follows. Section II covers the preliminaries of our study. Section III presents our design space refinement framework step by step. Section IV demonstrates the experimental results and analysis. Finally, we conclude our study in Section V.

II. Preliminaries

A. Unveiling the Black-box using XAI

Recent AI models are getting more complex to produce extremely accurate predictions. Their complexity makes it increasingly challenging to interpret how they arrived at those predictions. Therefore, AI models are often characterized as "a black-box problem" since they lack clear explanations for their decision-making process. To address this limitation, XAI has emerged as a field that aims to provide understandable explanations of AI models. Various studies proved their great capability of addressing real-world problems utilizing valuable insights obtained by XAI [9] - [10]. In this study, we utilize XAI to reveal the effective parameters of FPGA timing prediction. Using the design space comprised of these effective parameters, we can achieve fast convergence of optimization without requiring a substantial amount of time and resources.

B. DeepSHAP

DeepSHAP [11] is a promising XAI methodology that exhibits its strong performance in providing explanations of how inputs contribute to the output. In detail, it explains the reasons for ML-based predictions using SHAP (Shapley additive explanation) value as a unified measure of feature contribution score and DeepLIFT [12] as its model architecture. Among various XAI methods, we choose DeepSHAP for its low computational complexity and the capability of assessing the contribution of each feature independently. The detailed descriptions of DeepSHAP are as follows.

1) DeepLIFT: DeepLIFT is a backpropagation-based method for assigning the contribution of a particular input to the output of a model. It compares the activation of each neuron with its reference activation, which serves as a baseline for understanding the impact of the inputs on the output. Then, it assigns the difference to the contribution score, providing a qualitative explanation for how each input feature influences the final output. For a detailed description, let f and g represent the original prediction and explanatory models, respectively. In addition, let $x^0 = \{x_1^0, x_2^0, x_3^0, \cdots\}$ be the reference input and $t_0 = f(x^0)$ be the reference output. If the output for an arbitrary input $x = \{x_1, x_2, x_3, \cdots\}$ is $t = f(x)$, then, the difference in the reference output is $\Delta t = t - t_0$. Subsequently, the contribution scores ($C_{\Delta x_i \Delta t}$) follow the equation below [12].

$$\sum_{i=1}^{n} C_{\Delta x_i \Delta t} = \Delta t \tag{1}$$

If we map $C_{\Delta x_i \Delta t}$ to ϕ_i and $f(x)$ to ϕ_0, then, g satisfies the following additive feature attribution method.

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z_i' \tag{2}$$

where $z' \in \{0,1\}^M$, M is the number of features, and $\phi_i \in \mathbb{R}$. Additive feature attribution method is characterized by an explanation model that can be represented as a linear function of binary variables. Particularly, this method decomposes the output of an ML model into contributions of individual input features. These contributions are represented as a binary variable, indicating the existence of the features in the explanation. In equation (2), by assigning the ϕ to each feature and summing the contributions of all the features, DeepLIFT approximates the output of the original prediction model f.

2) SHAP: A SHAP value is a measure of the contribution of each feature in determining a specific output. This is obtained by calculating the average difference in the output depending on whether a feature is present or absent. The SHAP value is defined as the Shapley values of a conditional expectation function of the learning model, which helps to quantify the contribution of each input feature in defining the simplified input. We calculate the SHAP value using the following equations (3) and (4), where S denotes the set of non zero indices in z' and $\bar{S}$ denotes the set of the features not in S.

$$\phi_i(f, x) = \sum_{z' \subseteq x'} \frac{|z'|!(M - |z'| - 1)!}{M!} [f_x(z') - f_x(z' \backslash i)] \tag{3}$$

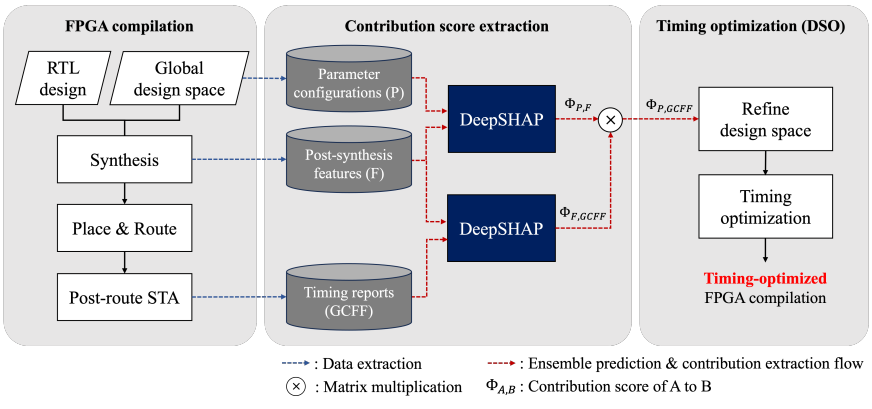
$$f(h_x(z')) = E[f(z)|z_S] = E_{z_{\bar{S}}|z_S}[f(z)] \tag{4}$$

[1] Then, the SHAP value is simplified by calculating the expected value, assuming feature independence and model linearity.

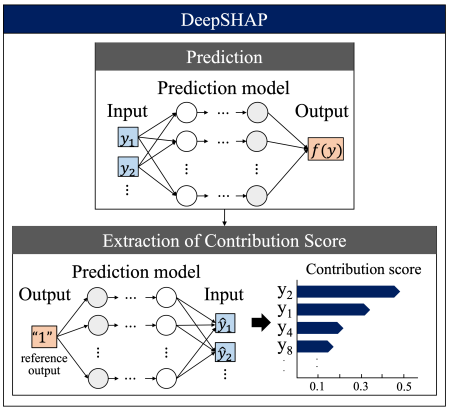
$$\begin{aligned} &E_{z_{\bar{S}}|z_S}[f(z)] \approx E_{z_{\bar{S}}}[f(z)] \quad \textit{(assume feature independence)} \\ &E_{z_{\bar{S}}|z_S}[f(z)] \approx f([z_S, E[z_{\bar{S}}]]) \quad \textit{(assume model linearity)} \end{aligned} \tag{5}$$

3) DeepSHAP (DeepLIFT + SHAP values): Eventually, by combining DeepLIFT and the SHAP values, a robust XAI method called DeepSHAP is defined, which can be applied to obtaining the feature contribution scores for an ML model. To calculate the contribution score using DeepSHAP, the output of the model is set as the reference value (e.g., 1). Then, DeepLIFT determines the values that each node in the layers should have in order to generate that specific output. By

[1] $|z'|$ denotes the cardinality of the set z'.
$z' \backslash i$ indicates the setting $z_i' = \mathbf{0}$. $f_x(z') = f(h_x(z')) = E[f(z)|z_S]$.



(a) Overall flow



(b) DeepSHAP flow

Fig. 2. Overall flow of our design space refinement framework. Fig. 2a illustrates three big parts of our framework: 1) FPGA compilation, 2) two-stage contribution score estimation using XAI, and 3) timing optimization using refined design space. Fig. 2b explains how the contribution score is calculated using the DeepSHAP method.

comparing these node values with their corresponding reference values, DeepSHAP quantifies the contribution of each input feature to the final output. In DeepSHAP, $f_x(z')$ in equation (3) is assumed as the expected value of the input features. By assuming that the input features are independent of each other and that the model is linear, the SHAP values are finally determined. In summary, DeepSHAP derives an effective linearization for each component and computes the contribution scores accurately.

III. Proposed Method

A. Overall Framework

Our proposed framework consists of three big parts, and it is described in Fig. 2a. First, for a given unseen design, we run initial FPGA compilations by randomly selected parameter configurations in the global design space. Then, using the compilation data, we apply DeepSHAP to our pre-trained ensemble prediction [13] model to estimate the contribution score of the parameters to the post-route timing results. In detail, we propose a two-stage contribution score estimation that calculates the parameter contribution score in precision by applying DeepSHAP to two separate stages consecutively. Utilizing the contribution score estimated by our framework, we determine the parameters to be used for the refined design space and subsequent timing optimization. The refined design space consists of the effective parameters that affect the post-route timing results dominantly, indicating that we can perform an efficient exploration during timing optimization. For the timing optimization, we introduce a timing metric called geometric mean of clipped failing F_{max} (GCFF) that encourages all clocks in a multi-clock design to be enhanced.

B. Ensemble Learning

In this study, ML-based predictions for estimating the contribution scores are performed using ensemble learning [13]. Ensemble learning is a technique in which multiple weak models are gathered to provide accurate and robust predictions instead of relying on a single powerful model. By leveraging the strengths of each individual model and compensating for its weaknesses, ensemble learning can achieve a better performance compared to that of a single model. Among various ensemble models, we utilize DRF [14], XGBoost [15], and GBM [16] in parallel, which proved their significant performance in diverse prediction problems. We pre-train each prediction model using SP&R compilation data collected using training designs. After that, when an unseen design is given, we exploit the best-predicting model for each different design for our later step. Since the best-predicting model varies depending on the design, we use different models each time.

Algorithm 1: Two-stage Contribution Score Estimation

inputs : pre-trained synthesis prediction model (M_{syn}), pre-trained P&R prediction model ($M_{P\&R}$)
output : contribution score of parameters to the post-route timing result ($\Phi_{P,t}$)

1 Run FPGA SP&R compilation
2 Extract parameter configuration $P = \{p_1, ..., p_n\}$
3 Extract post-synthesis features $F = \{f_1, ..., f_m\}$
4 Extract post-route timing result t
5 /* $\phi_{a,b}$: contribution score of feature a to output b */
6 /* $\Phi_{A,B} = \{\phi_{a,b}\}, \quad a \in A, b \in B$ */
7 $\Phi_{F,t} \leftarrow$ ExtractContribution$(M_{P\&R}, F, t)$
8 $\Phi_{P,F} \leftarrow$ ExtractContribution(M_{syn}, P, F)
9 $\Phi_{P,t} \leftarrow \Phi_{P,F} \times \Phi_{F,t}$ // matrix multiplication
10 **return** $\Phi_{P,t}$
11 /* Function to apply DeepSHAP and estimate the contribution score of A to B */
12 **Function** ExtractContribution(M, A, B):
13 **foreach** $b \in B$ **do**
14 $\hat{b} \leftarrow$ Inference(M, A)
15 $\Phi_{A,b} \leftarrow$ DeepSHAP$(M, A, b, \hat{b})$
16 **end**
17 **return** $\Phi_{A,B}$

C. Two-stage Contribution Score Estimation

In this section, we introduce a two-stage contribution score estimation, which applies DeepSHAP to two individual ensemble models to estimate the contribution score precisely. The detailed algorithm of our two-stage contribution score extraction is described in Algorithm 1. To obtain the contribution for an unseen design, we first run the FPGA SP&R compilation with random parameter configuration (Line 1). Then, we extract the parameter configuration P, post-synthesis features F, and the final timing result t (Lines 2-4). Thereafter, we employ DeepSHAP *consecutively* to infer the contribution scores of parameters to the final static timing analysis (STA) result. We first employ DeepSHAP for our pre-trained P&R prediction

TABLE I
LIST OF POST-SYNTHESIS FEATURES

	Name of Feature
1	total # of ALM (adaptive logic modules)
2	total # of LUT (look-up tables)
3	maximum depth of LUT
4	average depth of LUT
5	total # of DSP (digital signal processing) blocks
6	total # of registers
7	total # of registers using clock enable
8	total # of fanout
9	maximum # of fanout
10	average # of fanout

model to infer the contribution of post-synthesis features to the final timing results ($\Phi_{F,t}$) (Line 7). Intuitively, $\Phi_{F,t}$ here quantifies the relevance of each post-synthesis feature to the final timing result. Then, we employ DeepSHAP again to obtain the contribution of parameters to each of the post-synthesis features ($\Phi_{P,F}$), which stands for the relevance of each tool parameter to the post-synthesis features (Line 8). Finally, by aggregating the contribution scores determined from each of the two stages, we decide the parameter contribution score to the final timing result ($\Phi_{P,t}$) (Line 9). The matrix multiplication here plays a significant role in combining the useful explanations of the two prediction stages. By multiplying $\Phi_{P,F}$ and $\Phi_{F,t}$, we can infer the overall contribution score of the parameters to the post-route timing result *by observing the influence on each post-synthesis feature.* We adopt this multi-stage method to harvest the parameter contribution score in precision rather than applying DeepSHAP once at the whole SP&R stages. A single prediction model cannot provide accurate predictions of the whole SP&R flow using the parameter configuration as its input and the post-route STA result as its output due to the chaotic behavior of EDA tools [1].

Consequently, according to our idea, a parameter with high $\phi_{p,t}$ is a critical parameter that dominantly influences the post-route STA result during timing optimization. In other words, it suggests that when the design space consists of high $\phi_{p,t}$ parameters, we can perform an efficient exploration while fluctuating the STA results significantly. As a result, by excluding the parameters containing lower $\phi_{p,t}$ than a threshold, we can fine-tune the effective design space, which helps the optimization algorithm prevent the collection of wasteful FPGA compilations. [2] The detailed list of post-synthesis features we used is in Table I.

D. Evaluation: Geometric Mean of Clipped Failing $F_{\max}$

In this section, we discuss how we evaluate the timing performance of each FPGA compilation precisely while conducting the timing optimization. When we measure the timing of a design, we commonly apply a metric called $F_{\max}$, which represents the clock frequency at which the chip can maximally operate. $F_{\max}$ is defined as the inverse of an effective clock period, which is obtained by subtracting the worst negative slack (WNS) from the target clock period. However, many real-world designs are multi-clock designs containing multiple clock signals, so we cannot assess their performance in a unified metric. Therefore, we need to introduce a new metric, which

[2]Via repeated experiments, we set the threshold as 10^{-4}.

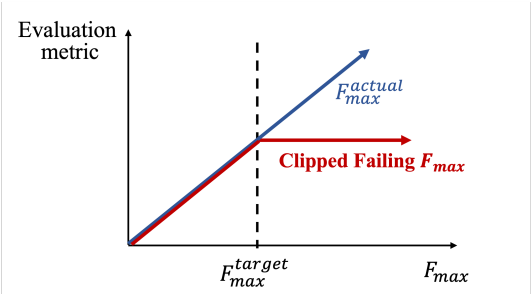


Fig. 3. Concept of clipped failing $F_{\max}$. $F_{\max}$ to be fed to evaluation metric is clipped at $F_{\max}^{target}$.

combines $F_{\max}$ of each clock so that it can be fed into the optimization algorithm to evaluate the result of a compilation.

Although there are various scenarios validating the timing performance of a multi-clock design, we focus on how close an FPGA compilation achieved its timing target. Assuming that the target timing constraint is determined at the earlier stage of IC design, the back-end designer who performs SP&R should provide the results closest to the target constraint. Therefore, we must feed the timing information, which encourages all the clocks to approach the target.

To achieve this, we introduce a metric called clipped failing $F_{\max}$ (CFF), which evaluates the degree of $F_{\max}$ achievement toward the target timing constraints. CFF is a concept that is equal to the actual $F_{\max}$ when a clock is timing-violated ($F_{\max}^{actual} < F_{\max}^{target}$). However, when a clock is timing-met ($F_{\max}^{actual} > F_{\max}^{target}$), the evaluation score is clipped at the target $F_{\max}$ without the incentives surpassing the target (Fig. 3). We adopt this clipping approach to avoid being biased at a specific clock that is easy to optimize during the optimization. Without clipping, an optimization algorithm might prioritize enhancing a clock that is easily improved over the target $F_{\max}$, potentially degrading another clock that is still unsatisfying with the target. This may not be the desired direction for optimizing multi-clock designs. Therefore, CFF allows us to evaluate the $F_{\max}$ of a clock with the amount of how it is close to the target period. Let C be the set of all clocks in a design,

$$CFF_c = \min\{\, F_{\max,c}^{target},\ F_{\max,c}^{actual} \,\}, \qquad c \in C \tag{6}$$

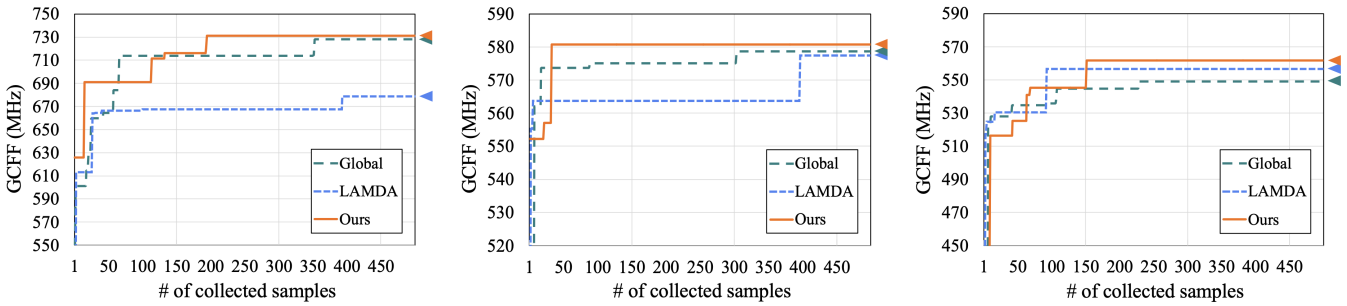
Finally, taking the geometric mean of all the CFF in a design, we obtain GCFF, which is a unified metric for evaluating the performance of a multi-clock design. Since we want to see the relative improvements of the clocks as the optimization iteration goes by, the geometric mean can play an excellent role in evaluating them. In conclusion, using GCFF, the optimization algorithm is expected not to be biased at several specific clocks but to try to improve all the failing clocks evenly.

$$GCFF(C) = \left(\prod_{c \in C} CFF_c \right)^{\frac{1}{|C|}} \tag{7}$$

IV. EXPERIMENTAL SETUP AND RESULTS

A. Experimental Setup

We evaluated our design space refinement framework using *Intel Quartus Prime Pro 22.3.0* targeting the *AGFB014R24B2E2V* FPGA device from the Agilex family. To establish a comparison with our refined design space, we



(a) stereo_vision (b) stap_qrd (c) cholesky_mc

Fig. 4. Comparison of the global-best GCFF based on data samples collected. Our design space was outstanding in converging to the best GCFF, even at the early DSO iterations. We proved that our design space can provide efficient exploration during timing optimization.

TABLE II
LIST OF TOOL PARAMETERS IN GLOBAL DESIGN SPACE

	Name of Parameter	Possible Values
1	*auto_dsp_recognition*	{0, 1}
2	*disable_register_merging_across_hierarchies*	{0, 1, 2}
3	*mux_restructure*	{0, 1, 2}
4	*optimization_technique*	{3, 4, 5}
5	*synthesis_effort*	{2, 6}
6	*allow_register_retiming*	{0, 1}
7	*allow_ram_retiming*	{0, 1}
8	*allow_dsp_retiming*	{0, 1}
9	*adv_netlist_opt_synth*	{0, 1}
10	*state_machine_processing*	{2, 7, 8, 9, 10, 11, 12}
11	*auto_rom_recognition*	{0, 1}
12	*auto_shift_register_recognition*	{0, 2, 13}
13	*qii_auto_packed_registers*	{0, 2, 14, 15, 16, 17, 18}
14-18	*five Quartus Prime internal parameters*	*Quantized Float Values*

*The size of global design space is 1.4×10^{10} scale.
*The possible values of each parameter are represented in abbreviations.
0: off, 1: on, 2: auto, 3: area, 4: balanced, 5: performance, 6: fast, 7: gray, 8: johnson, 9: minimal bits, 10: one-hot, 11: sequential, 12: user-encoded, 13: always, 14: minimize area, 15: minimize area with chains, 16: normal, 17: sparse, 18: sparse auto

define a global design space using, 13 parameters from the *Quartus Prime settings file* (QSF) parameters and five from *Quartus Prime* internal parameters (Table II). In summary, the global design space consists of 18 parameters, resulting in a size on the scale of 1.4×10^{10}. Furthermore, for the timing optimization, we implemented one of the outstanding DSO frameworks, OpenROAD-AutoTuner (OR-AT) [4], which was modified to operate for *Quartus Prime*. Among the six algorithms implemented in OR-AT, we observed that Optuna [17] exhibits the best performance in optimizing *Quartus Prime* design spaces via repeated experiments. Consequently, we selected Optuna-based OR-AT as our baseline optimization methodology for the analysis and comparisons.

As our evaluation benchmark, we adopted Titan23 benchmarks [18]. To build the training dataset for our ensemble prediction model, we used 17 of 23 Titan designs and collected 1,200 data samples for each design. A data sample in this section denotes a single SP&R compilation data, which consists of parameter configuration, post-synthesis features, and post-route STA result (GCFF). Overall, the training dataset consisted of 20,400 samples, and we used the remaining six Titan designs for the test.

As we discussed, to quantify the contribution scores, we need to collect multiple FPGA compilation data for an unseen design. Therefore, for our experiment, we randomly sampled 50 parameter configurations in the global space and executed the SP&R compilation flow in parallel. Meanwhile, during the timing optimization experiments, we collected 500 samples for each experiment. This number looks significantly smaller than those reported in previous studies [2] - [7]. This is because our proposed framework aims to reduce the redundant initial exploration and determine the optimum in a short time; we set the number of samples small.

B. Prediction Results

We first evaluate the accuracy of our pre-trained models using the six test designs to verify if our prediction and contribution scores are reliable. Our P&R prediction model predicted the normalized GCFF of the unseen designs with a 0.69 R2 score on average. Meanwhile, the synthesis prediction model achieved a 0.77 R2 score on average, predicting each of the post-synthesis features in a normalized form.

C. Timing Optimization Results

To evaluate our framework, we compared our design space with the global space and the one defined in LAMDA [7]. We chose LAMDA as our evaluation baseline since it is the latest state-of-the-art DSO study that aims to achieve timing closure in the Intel FPGA design space.

Fig. 4 displays the detailed timing optimization progress of three test designs, and Table III summarizes the optimization results of all test designs. If the global-best GCFF in Fig. 4 increases extensively within a small number of samples, it implies that the design space helps the efficient initial exploration. As a result, our framework showed the greatest performance in finding the highest GCFF in the early optimization iterations.

On the other hand, for some of the test designs (*gsm_swith* and *des90*), ours obtained slightly lower GCFF than the global design. Interestingly, these two designs differed significantly from the other four. That is, the GCFF values of *gsm_swith* and *des90* did not fluctuate much with their parameter configurations compared to the other designs (Fig. 5), which means the two designs obtain less room for improvement *by tuning the tool parameters*. It demonstrated that our framework exhibits remarkable performance when a design obtains enough room for improvement, as ours is concentrated on analyzing how each parameter affects the final output.

Overall, the design space obtained by our framework achieved 25.92% higher GCFF than the default setting, while the global design space and LAMDA exhibited improvements of 23.98% and 21.87%, respectively. In terms of a F_{max} metric, a difference of tens of MHz is significant because it originated from a huge WNS difference. Therefore, this result demonstrates that the general design space defined by LAMDA

TABLE III
OVERALL DSO RESULTS OF ALL TEST DESIGNS

Benchmark	Design Characteristics (Default)			Best-Found GCFF (MHz)				# of Samples to Find Best		
	# ALM	# registers	# LUT	Default	Global	LAMDA [7]	Ours	Global	LAMDA [7]	Ours
stereo_vision	38,380	69,186	45,320	534.37	728.16	678.54	**751.01**	354	395	196
stap_qrd	94,448	165,009	105,318	466.85	578.70	577.37	**580.72**	305	397	33
gsm_switch	72,862	122,667	98,365	373.92	**384.98**	383.00	384.79	238	179	183
des90	47,289	37,876	73,831	217.44	**252.78**	247.16	247.71	391	14	20
cholesky_mc	43,972	78,056	46,328	355.87	549.15	556.48	**561.80**	230	94	153
cholesky_bdti	103,273	184,104	117,095	403.06	443.85	435.73	**467.07**	276	24	364
Average improv.	-	-	-	-	23.98%	21.87%	**25.92%**	-	-38.52%	**-47.10%**

*Design characteristics and default GCFF are obtained by the *Quartus Prime* default setting.

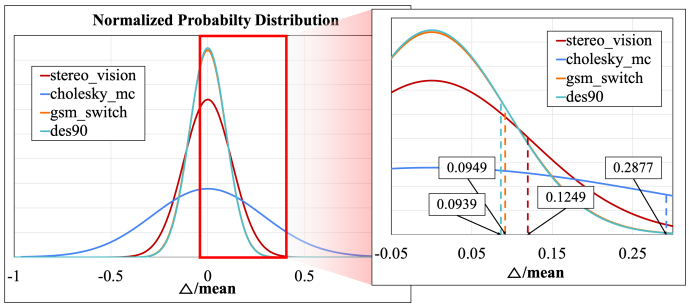


Fig. 5. The probability distributions of normalized GCFF while sweeping parameter configurations. The distributions of *gsm_swith* and *des90* are concentrated near their mean while *stereo_vision* and *cholesky_mc* are widely distributed according to their parameter configurations. Each value represented in the plot on the right denotes the $+1\sigma$ point.

was unsuitable for each different design, making it difficult to achieve acceptable timing results. Additionally, not only finding the best GCFF but also achieving fast convergence was the greatest using our framework. The average samples collected to find the best were the smallest on average among the three design spaces. Our approach collected 47.10% fewer samples to find the best GCFF compared to the global design space while achieving the best average improvement. In summary, ours is outstanding at finding remarkable timing improvements in a short runtime, and it is truly attractive to many IC designers.

Furthermore, our framework required less than a minute to estimate the contribution scores and refine the design space. Therefore, its runtime overhead is the same as a single FPGA compilation time, which is used for estimating the contribution scores. Even considering such runtime overhead, we proved that our framework helps achieve faster convergence of the timing optimization.

V. CONCLUSION

This study proposes an effective and novel framework that fine-tunes the design space suited for a specific design in preparation for subsequent timing optimization. We estimate the parameter contribution scores using the ensemble prediction and DeepSHAP at two separate stages consecutively. Consequently, we produce an effective design space within a single FPGA compilation runtime overhead. Our design space accomplished 25.92% higher GCFF than the *Quartus Prime* default setting, while the global design space and LAMDA achieved 23.98% and 21.87%, respectively. To achieve that, we required 47.10% fewer compilation samples than the global design space, and it was even 14% fewer than LAMDA. We proved that our design space significantly contributes to achieving the best performance over the other design spaces with the smallest number of samples. Our study goes beyond a DSO methodology and provides a comprehensive understanding of effective parameters that influence the quality of optimization. Therefore, it can be applied as a preliminary step for future optimization studies, and we expect that IC designers will take remarkable advantage of obtaining better QoR in less design time. In subsequent studies, we plan to expand our framework to multi-objective optimization, which optimizes power, performance, and resource utilization simultaneously.

REFERENCES

[1] A. B. Kahng and S. Mantik, "Measurement of Inherent Noise in EDA Tools", *Proceedings International Symposium on Quality Electronic Design*, 2002, pp. 206-211.

[2] H. Geng, T. Chen, Y. Ma, B. Zhu and B. Yu, "PTPT: Physical Design Tool Parameter Tuning via Multi-Objective Bayesian Optimization", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 42(1) (2023), pp. 178-189.

[3] Z. Zhang, T. Chen, J. Huang and M. Zhang, "A Fast Parameter Tuning Framework via Transfer Learning and Multi-objective Bayesian Optimization", *DAC '22: Proceedings of the 59th ACM/IEEE Design Automation Conference*, 2022, pp. 133-138.

[4] J. Jung, A. B. Kahng, S. Kim and R. Varadarajan, "METRICS2.1 and Flow Tuning in the IEEE CEDA Robust Design Flow and OpenROAD ICCAD Special Session Paper", *2021 IEEE/ACM International Conference On Computer Aided Design*, 2021, pp. 1-9.

[5] R. Liang, J. Jung, H. Xiang, L. Reddy, A. Lvov, J. Hu and G. J. Nam, "FlowTuner: A Multi-Stage EDA Flow Tuner Exploiting Parameter Knowledge Transfer", *2021 IEEE/ACM International Conference On Computer Aided Design*, 2021, pp. 1-9.

[6] A. Agnesina, K. Chang, and S. K. Lim, "VLSI Placement Parameter Optimization using Deep Reinforcement Learning", *2020 IEEE/ACM International Conference On Computer Aided Design*, 2020, pp. 1-9.

[7] E. Ustun, S. Xiang, J. Gui, C. Yu and Z. Zhang, "LAMDA: Learning-Assisted Multi-stage Autotuning for FPGA Design Closure", *2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines*, 2019, pp. 74-77.

[8] F. Hussain, R. Hussain, E. Hossain, "Explainable Artificial Intelligence (XAI): An Engineering Perspective", *arXiv preprint arXiv:2101.03613*, 2021.

[9] H. Yuan, H. Yu, S. Gui and S. Ji, "Explainability in Graph Neural Networks: A Taxonomic Survey", *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45(5) (2022), pp. 5782-5799.

[10] Y. Xu, X. Yang, L. Gong, H. C. Lin, T. Y. Wu, Y. Li and N. Vasconcelos, "Explainable Object-Induced Action Decision for Autonomous Vehicles", *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 9523-9532.

[11] S. M. Lundberg and S. I. Lee, "A Unified Approach to Interpreting Model Predictions", *31st Conference on Neural Information Processing Systems*, 2017, pp. 4768–4777

[12] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning Important Features Through Propagating Activation Differences", *In: arXiv preprint arXiv:1704.02685*, 2017.

[13] L. K. Hansen and P. Salamon, "Neural network ensembles", *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12(10) (1990), pp. 993-1001.

[14] L. Breiman, "Random Forests", *Machine Learning* 45 (2001), pp. 5-32.

[15] T. Chen and C. E. Guestrin, "XGBoost: A Scalable Tree Boosting System", *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785-794.

[16] J. H. Friedman, "Greedy Function Approximation: A Gradient Boosting Machine", *The Annals of Statistics* 29(5) (2001), pp. 1189-1232.

[17] T. Akiba, S. Sano, T. Yanase, T. Ohta and M. Koyama, "Optuna: A Next-generation Hyperparameter Optimization Framework", *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019, pp. 2623-2631.

[18] K. E. Murray, S. Whitty, S. Liu, J. Luu and V. Betz, "Timing-Driven Titan: Enabling Large Benchmarks and Exploring the Gap Between Academic and Commercial CAD", *ACM Transactions on Reconfigurable Technology and Systems* 8(2) (2015), pp. 1-18.