Boomerang: Physical-aware Design Space Exploration Framework on RISC-V SonicBOOM Microarchitecture

Yen-Fu Liu, Chou-Ying Hsieh and Sy-Yen Kuo

Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan

{*r09921132, f07921043, sykuo*}*@ntu.edu.tw*

***Abstract*—As semiconductor manufacturing technology advances, microarchitecture designers could use the advanced process to create complex microarchitectures, making the design more performant, power-saving, and area efficient. To get better designs for various objectives, designers use hardware description language to deploy a highly parameterizable hardware design generator. However, the more flexible the parameterized design is, the more complex the parameters to choose over the possible parameter space. Traditionally, experienced hardware engineers have to hand-tune each design parameter to achieve better performance, power, and area efficiency. Moreover, the design space is usually too large to practically explore by running place-and-route (PnR) of each design since the electronic design automation (EDA) tool could take days to finish it. Thus, prior works on microarchitecture search only focus on logical results, which leads the search framework to ignore the quality gap between logical and physical design, resulting in inaccurate Pareto frontier prediction. To further address this issue, we use the physical synthesis technique to estimate the physical characteristics and save runtime while exploring the design space simultaneously. In this case, we developed a heuristic idea, Physical-aware Design Space Exploration Framework on RISC-V SonicBOOM Microarchitecture, that optimizes the netlist and physical layout quality. It results in a much more accurate prediction of the Pareto parameter set with a 40% and 28% search time reduction on the three-objective and four-objective problems, respectively, compared to the best baseline model. In addition, our method achieved a seven times smaller hypervolume difference than the best baseline model within the same number of observations.**

***Index Terms*—Microarchitecture search, RISC-V, Multi-objective Bayesian Optimization, Physical Layout Estimation, Physical-aware, Transfer learning.**

I. Introduction

With chips advancing toward the sub-nanometer manufacturing processes, CPU microarchitecture designers create complex designs that enable them to be performant, power-saving, and area efficient. Meanwhile, designers face challenges in creating highly parameterized and modularized microarchitecture generators. Since traditional hardware description languages (HDLs) were born to be used in simulation and verification instead of design, they lack expressiveness on the flexibility of the microarchitecture designs that designers require. Moreover, traditional HDLs cannot provide the powerful high-level abstraction that modern software programming languages can provide. Hardware engineers struggle to construct hardware designs like microarchitectures that require vast design-space exploration. Therefore, Chisel [1], a much more efficient hardware construction language (HCL) based on Scala, comes in handy. Designers can efficiently use Chisel to deploy a highly parameterizable and modularizable hardware design generator leveraging the power of a modern programming language.

Recent studies [2]–[4] utilizing HCL to build their parameterized microprocessor, deep neural network accelerator, and co-processor generator illustrate that HCLs can alleviate the stress of writing flexible generators. Unfortunately, using an HCL escalates the responsibility of hand-tuning each design parameter. Finding the best parameters for the specification with the human-in-the-loop is costly, time-consuming, and prone to human error. Since the search entangles with achieving optimal power, performance and area (PPA) and physical factors [5], [6], which is challenging even for an experienced engineer, designers turned to use a systematic approach to solve the microarchitecture search problem. However, prior works [7]–[9] only considered execution cycles as the objective of the design while neglecting the power, area, and physical factors that are also critical objectives. On top of that, prior works [10], [11] did not consider routing congestion, which significantly impacts quality metrics and remains unknown until the physical design is completed [12], [13]. To address the completeness of objectives and expose the congestion in the early design cycle, we propose a framework to provide early-stage physical-aware optimization based on physical layout estimation (PLE) and simultaneously reduce the long physical design process.

To solve the multi-objective microarchitecture search in our framework, we leverage a register-transfer level (RTL) knowledge dataset to automatically cluster the unsampled design space with adaptive cluster criteria sampling into subregions. To our best knowledge, this paper is the first work to address multi-objective optimization by applying the transfer learning technique to pass knowledge through the physical-aware RTL-inspired neural networks and solve the domain-crossing issue with a pseudo-labeled array technique. The neural network guides multi-objective Bayesian optimization

DOI 10.1109/ASAP57973.2023.00026

with a Pareto forecasting technique to provide a better early exploration quality.

The main contributions can be summarized as follows:

1) We propose using a physical layout estimation flow in microarchitecture search to provide a more true-to-life PPA with physical factors and the congestion score in the early design stage to reduce the hand-off iteration between logic and physical design.
2) We apply the adaptive cluster criteria sampling technique to sample a better initial dataset from the design space with the RTL knowledge dataset.
3) We introduce an pseudo-labeled array technique to solve domain-crossing while applying transfer learning on the model with the knowledge learned from the RTL domain.

The remainder of this paper is organized as follows. In Section II, we discuss the related works that inspire our work. We introduce background in Section III. Section IV describes the overview and implementation of *Boomerang*. Section V shows the results, and Section VI concludes our work with a discussion of future work.

II. RELATED WORKS

Efficient Design Space Exploration via Statistical Sampling and AdaBoost Learning [8] used the orthogonal design sampling and the Adaboost technique to target the microprocessor design space exploration with the gem5 cycle-accurate simulator [14]. They claimed that regression-based methodologies would require a large initial training dataset and a time-consuming software simulation to achieve reliable prediction. *Processor Design Space Exploration via Statistical Sampling and Semi-Supervised Ensemble Learning* [9] proposed using the Latin hypercube sampling (LHS) technique to solve the non-adjustable sizing problem of the orthogonal array (OA) in the orthogonal design sampling. *BOOM-Explorer* [11] argued that neither random nor OA sampling could sample an initial representative dataset. To address this issue, they developed a method called *MicroAL* using the transductive experimental design (TED) [15] method to sample the design space. *BOOM-Explorer* outperformed other techniques in finding the Pareto set. However, they used execution cycles as the performance metric and did not consider the critical path. They also used the same clock speed instead of the maximum clock frequency (Fmax) in the simulation, overlooking the design's real performance and power. Furthermore, the synthesis of the design did not leverage physical estimation, and the best-known Pareto set would not align with the realistic Pareto set after considering the Fmax of each design and physical factors. Hence, our research proposes an optimization framework that uses more genuine physical-aware criteria to address the gap between logic and physical layout performance.

III. BACKGROUND

A. BOOM: Berkeley Out-of-Order RISC-V Processor

BOOM (Berkeley Out-of-Order Machine) was initially created using Chisel for a computer architecture course. SonicBOOM [16], version 3 of the BOOM series, is one of the most performant superscalar out-of-order open-source processors. Based on the lessons learned from the performance and the physical layout of BOOMv2, SonicBOOM improves the bottlenecks within the frond-end, the execution, and the load-store unit of the previous BOOM. To address the hand-off between logic and physical design, *Boomerang* provides an early physical layout estimation while exploring the design space, reducing the number of hand-off iterations between the design and its physical layout.

TABLE I
THE PARAMETER LIST OF SONICBOOM GENERATOR. IF THE CANDIDATE VALUES COLUMN IS A PARAMETER NAME, IT MEANS THE VALUE IS THE SAME AS THAT SPECIFIED PARAMETER VALUE.

Module	Parameter	Description	Candidate values
FrondEnd	fetchWidth	The number of instructions can be fetched at once.	4,8
	numFetchBufferEntries	The number of instructions that are stored between fetch and decode.	decodeWidth * fetchWidth * 2
	IcacheWay	The number of associate sets in the L1 I-Cache.	2, 4, 8
	IcacheTLB	The number of translation lookaside buffer (TLB) ways in I-Cache. The number of TLB entries is equal to the TLB sets times the TLB ways.	8, 16, 32
	IcacheFetchByte	Divide the number of FetchWidth by two and times four.	(FetchWidth / 2)*4
	maxBrCount	The number of branches it speculates simultaneously.	8, 12, 16, 20
	numRasEntry	The number of entries in the Return Address Stack (RAS).	16, 24, 32
	beatBytes	The width of the interface between L1 and outer memory in bytes.	fetchWidth * 2
IDU	decodeWidth	The number of instructions can be decoded at once.	1, 2, 3, 4, 5
	numRobEntries	The number of entries of register reorder-buffer.	32, 64, 96, 128, 130
	numIntPhysRegisters	The number of physical registers for integer.	48, 64, 80, 100, 128
	numFpPhysRegisters	The number of physical registers for floating point.	numIntPhysRegisters
EU	MemIssueWidth	The number of issue widths in the memory issue queue.	1, 2
	MemNumEntries	The number of entries in the memory issue slot.	8, 16, 24, 32, 40
	IntIssueWidth	The number of issue widths in the integer issue queue.	decodeWidth
	IntNumEntries	The number of entries in the integer issue slot.	8, 16, 24, 32, 40
	FpIssueWidth	The number of issue widths in the floating point issue queue.	MemIssueWidth
	FpNumEntries	The number of entries in the floating issue slot.	8, 16, 24, 32, 40
	dispatchWidth	The number of dispatch widths is equal to the number of the DecodeWidth.	decodeWidth
LSU	numLdqEntries	The number of entries in the load queue.	8, 16, 24, 32
	numStqEntries	The number of entries in the store queue.	8, 16, 24, 32
	DcacheWay	The number of associate sets in the L1 D-Cache.	2, 4, 8
	DcacheTLB	The number of TLB ways in D-Cache. The number of TLB entries is equal to the TLB sets times the TLB ways.	8, 16, 32
	DcacheMSHR	The number of entries in the Miss Status Handling Register (MSHR).	2, 4, 8

In the SonicBOOM generator, various parameters can be changed in each module, as shown in Table I. The size of the microarchitecture design space is around 2.9×10^9. For such an astronomical number, searching Pareto set by evaluating every design in the design space is unattainable. In this work, we narrow the search space into a given candidate list that is small enough to get the ground truth Pareto set of the given candidate list. Each method used in evaluation can only choose from the candidate list, and we calculate its hypervolume difference between the solution from our method and the ground truth Pareto set.

B. VLSI Design Flow and Early Physical Layout Estimation

As technology scaling evolves, digital circuit designs and electronic design automation (EDA) tools are getting increasingly complex. The increasingly complex EDA tools enable designers to design cutting-edge hyper-scale processors and thoroughly examine their parameters with advanced technology nodes. Digital design nowadays is a highly automated methodology. As Figure 1(a) shows, the VLSI design flow consists of several steps. Each step in the VLSI design flow

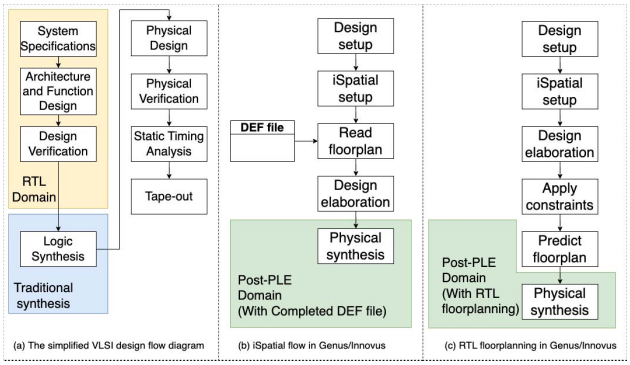


Fig. 1. (a)The simplified VLSI design flow diagram.(b) Genus/Innovus iSpatial flow diagram with floorplan (DEF) file. (c) Genus/Innovus RTL floorplanning flow diagram. The yellow RTL domain means a design's performance only considers its RTL behavior. The traditional synthesis in the blue area can only estimate a design's PPA using a wire-load model for physical factors and netlist information for PPA. The green post-PLE domain means a design's physical layout can be estimated from RTL floorplanning in Genus/Innovus PLE flow, which provides a better PPA correlation between the synthesis and physical design stage.

plays a critical role, and the circuit quality produced by any given stage propagates to later stages.

Traditional synthesis tools often use the wire-load model for early route estimation. The wire-load model does not deliver accurate wire delay information, particularly for scenarios where a substantial amount of the delays are wire delays. As a result, users will notice a huge gap in performance, power, and area of design between logical synthesis and physical layout.

By contrast, physical layout estimation (PLE) leverages technology information to model the impact of placement and routing. It enables designers to analyze and optimize the design in ways that traditional logical synthesis flow cannot provide. In addition, using physical estimation offers a better prediction of PPA in the downstream flow and reduces the number of hand-off iteration cycles from the front end to the backend. Because the PLE can analyze the physical impact in the early synthesis stage, using PLE provides designers with a better experience in agile hardware development. More importantly, physical layout estimation early in the VLSI design flow facilitates designs with superior PPA. In this work, microarchitecture parameter search incorporates early physical design estimation, drastically reducing the iteration cycles and improving the engineers' productivity.

To discover the physical impact during microarchitecture search, Figure 1(b) illustrates the PLE flow in Genus physical synthesis guide. Initially, the early-physical and iSpatial flow in Genus uses placement of synthesized logic, which provides better correlation and prediction of backend quality of results (QoR) and pre-optimizes the netlist before running through the backend. However, in a microarchitecture search, it is not practical to manually place each possible design and provide Genus with the floorplan file (DEF file). To solve the floorplan problem, in Figure 1(c), we use RTL floorplanning, the newest Genus physical synthesis flow feature, and the RTL floorplanning is very suitable for microarchitecture search. Hence, if either no DEF file or an incomplete DEF file has been read, Genus calls the backend engine, Innovus, to create a floorplan for the design with the core aspect ratio and the target core utilization rate after design elaboration.

C. Problem and Terminology Definitions

This section will define the problem and the terminology we used in our research.

*a) **Objective Space**:* Finding a universal index to judge a design is challenging. This research defines performance by the execution time criteria.

- **Execution cycles** Execution cycles define the number of clock cycles that a design requires to finish a benchmark. The execution cycle performance is the same while running the RTL and netlist simulations.
- **Execution time** The execution time is defined by the number of execution cycles multiplied by the clock period. Each design will operate at Fmax frequency. The execution time also is the wall-clock time of finishing a benchmark. The clock period can be obtained through the synthesis result.

Most of the prior works only consider the execution cycles of a design, and neglect the side effect of over-designed parameter specification, particularly with physical impacts, which causes the critical path to be extremely long. Moreover, most prior works did not push the design in the synthesis stage to achieve the high Fmax, which is unrealistic in industry since commercial high-performance computing chips usually operate near the Fmax point.

This research uses execution time with each design's maximum clock frequency as an objective instead of using the execution cycles.

- **Design power consumption** The average power consumption of a benchmark in netlist simulation defines design power consumption.
- **Design area consumption** The core area in the Genus report defines design area consumption. The design area consumption includes macros such as SRAM in the physical synthesis report.
- **Design congestion score** The max hot spot score in the physical synthesis report defines the design congestion score. The congestion score is reflective of the physical QoR after the physical layout. Specifically, the max hot spot score can detect the design's congestion and optimize the specification parameters in advance. In contrast, a traditional synthesis flow could not provide such critical information to the designer.

*b) **Pareto Set of Design Space**:* Pareto optimality, also known as Pareto efficiency, indicates a state of resource allocation in which it is not likely to make any one individual better off without making others worse off. Pareto optimality is a benchmark for efficiency and fairness in resource allocation, and the set of solutions in the Pareto optimality situation is called the Pareto set or the Pareto front.

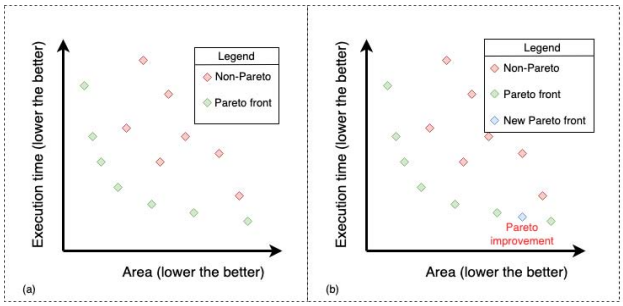


Fig. 2. The Pareto optimality situation. (a)The known solution in the design space with green Pareto front and red non-Pareto front solutions. (b) A new Pareto frontier solution in blue diamond with Pareto improvement.

This concept of Pareto efficiency fits well in the microarchitecture search problem because the objective of CPU design is a resource allocation situation, and a Pareto front exists in the problem. Figure 2 shows the Pareto front of the design space with execution time and area. The green diamonds are the Pareto front in the design space which cannot be improved in the known solution in Figure 2a. Besides, the red diamonds are Pareto-dominated, which means there is a better solution in the design space with better resource allocation in every objective. Moreover, if we find a new solution, the blue diamond in Figure 2b, the improvement of the new solution in Pareto-optimality is Pareo improvement. Overall, the Pareto optimality can be described as the following criteria. For minimizing an n-dimensional objectives problem, an objective function $obj(sample1)$ is said to be the Pareto frontier and dominate $obj(sample2)$ if

$$\begin{aligned} &\forall k \in [0, n-1], obj_k(sample1) \leq obj_k(sample2) \\ &\exists q \in [0, n-1], obj_q(sample1) < obj_q(sample2) \end{aligned} \tag{1}$$

In short, the Pareto frontier like *sample1* in the microarchitecture design space is the target we want to search for in this research.

*c) **Pareto Hypervolume**:* After searching the best-known Pareto set, the Pareto set will form a hypervolume in n-dimensional objective space. A hypervolume can be calculated with a reference point in the objective space. The reference point denotes the lower bound of the objective value. If we want a better solution with a Pareto improvement, we can set our target as maximizing the hypervolume in the design space. This target is also called hypervolume improvement, which means the gain of the hypervolume if we find a new Pareto frontier.

*d) **Microarchitecture Search**:* Microarchitecture search is a specific type of design space exploration that concentrates on searching for an optimal set of hardware parameters for one or more than one objective functions. Mathematically, microarchitecture search can be modeled as an optimization problem. The purpose of the microarchitecture search is to find a set of optimal microarchitectures that minimize the given objective function(s) *obj(d)*, subject to a set of constraints *C(d)* and a set of design parameters *d*. Formally:

$$\begin{aligned} &\text{Minimize } obj(d), \text{ subject to} \\ &d \in D, \text{where } D \text{ is the microarchitecture design space.} \\ &C(d) \leq 0, \text{where } C \text{ is the constraint function.} \end{aligned} \tag{2}$$

In our work, the objective functions *obj(d)* denote the performance, power, area, and congestion score. The constraints *C(d)* ensure that the design satisfies certain requirements, such as design core aspect ratio, design core utilization rate, and timing constraints. To minimize the objective functions, *obj(d)*, and find the Pareto set in the design space *D*, we maximize the Pareto hypervolume with a multi-objective Bayesian optimization technique.

IV. *Boomerang*: Physical-aware Design Space Exploration Framework on RISC-V SonicBOOM Microarchitecture

Boomerang is a physical-design-aware CPU microarchitecture design space exploration and exploitation framework based on Bayesian optimization. The term *Boomerang* illustrates the return of sampling design through the PLE flow. Moreover, the framework learns from experience and tries to throw another shot iteratively with higher and higher accuracy on sampling. Figure 3 shows the *Boomerang* framework overview. The framework's objectives are performance(execution time), power, area, and congestion score(max-hot spot). The framework will perform optimization as following steps:

1. **Generate RTL code and RTL simulation flow** The framework will randomly sample K samples (1000 in our evaluation) from the design space and generate their RTL code with Chisel. After code generation, the framework will perform the RTL simulation in the VCS simulator to collect the execution cycles for each design.

2. **RTL Knowledge Dataset Construction** After the RTL simulation, the framework will construct a dataset with the design parameters and the corresponding execution cycles. The effort of constructing an RTL knowledge dataset is small since the speed of simulation of a *Dhrystone* benchmark in the RTL domain is about 15 times faster than netlist simulation in the post-PLE domain. Furthermore, the netlist simulation can only be done after physical synthesis, which usually takes 10-20 hours. In this case, the RTL simulation is over 100 times faster than the netlist simulation. We can drastically reduce the overall search time if the framework learns some critical information through RTL simulation before the expensive netlist simulation in the post-PLE domain. The framework will leverage the RTL knowledge dataset later to learn some information. Nevertheless, the RTL simulation is not useful as a sole metric, because it only provides execution cycles without other objectives and physical information.

3. **Adaptive Cluster Criteria Sampling** The role of initial sampling in microarchitecture search is like sprinkling seeds on the field. To address the uniform distribution issue of initial sampling, prior works used random sampling, OA sampling, LHS, and TED, as a means to create an improved initial set of samples. While yet, none of those techniques can guarantee

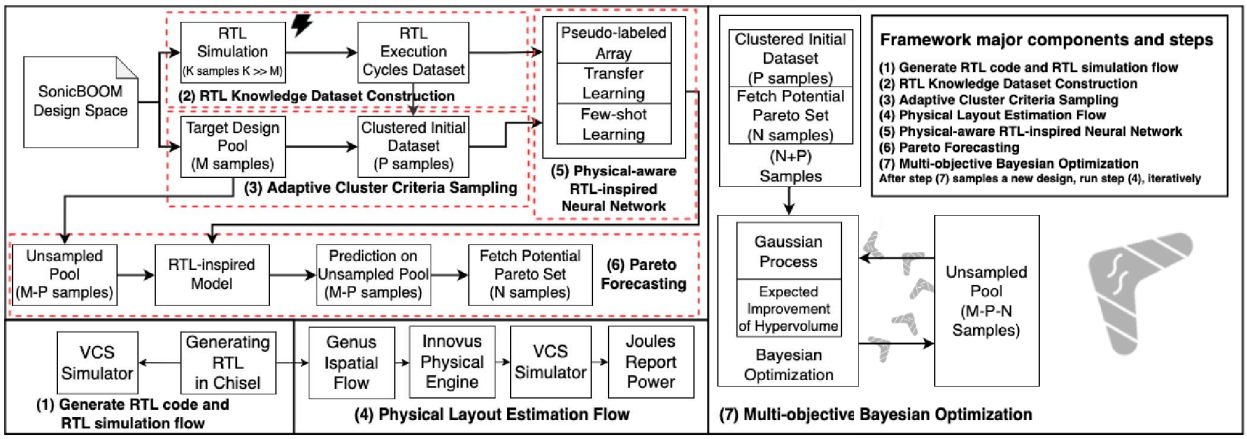


Fig. 3. The *Boomerang* framework overview.

an adaptive result with different data distribution. Since these heuristic methods do not use systematic observation from the design space, the capability of adapting to different design characteristics is missing. Therefore, we introduce a model-based approach in microarchitecture search that can change with data characteristics.

Since RTL simulation can obtain part of the characteristics of the design space, we use a two-level feature selection method on the RTL knowledge dataset to determine the clustering criteria, which acts like a probe to the design space. The feature selection not only can be used to provide a better regression quality but also reveals the importance rank of each feature. However, if a single regression model determines the feature selection, the result will bias toward the specific regressor. Thus, we use an ensemble model to vote on the feature selection. To consider the credibility of each regression model in our ensemble model, we combine the importance rank of features and the regression score of each regressor to calculate the overall feature rank. Furthermore, we use a cross-validation technique in the first-level feature selection in each regression to prevent over-fitting.

Two-level feature selection aims to enable the algorithm that selects the feature in coarse grain by the first-level feature selection and calculates the rank of those coarse grain features individually by the second-level feature selection. This way, we can leverage different regression models to choose the unbiased essential features. In our case, the second-level fine grain feature ranking in order are *decodeWidth*, *fetchWidth*, *issueWidth*, *numPhyRegister*, *numLSEntries*, and *dCacheWay*. The final score of those features is in Figure 4.

The user can choose the final clustering index, and we choose *decodeWidth+fetchWidth* to be our criteria since the number of clusters is suitable for our design pool size. After selecting the clustering criteria, we can apply a randomized transductive experimental design (RTED) to each cluster and sample the initial dataset. The RTED will add one random sample from the first cluster to the initial sample list and then calculate the distance between the initial sample list and

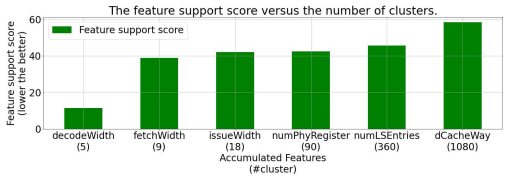


Fig. 4. The feature support score versus the number of clusters after adding this feature into the criteria. To avoid splitting too many clusters and keep the important features simultaneously, we select decodeWidth+fetchWidth as our clustering criteria.

the next cluster. Iteratively, it chooses the sample with the maximum Euclidean distance and adds it to the initial sample list. Overall, the adaptive cluster criteria sampling avoids any pre-defined clustering technique and relies on automatic calculation with feature selection on the RTL simulation dataset.

4. **Physical Layout Estimation Flow** The four objectives, such as performance, power, area, and congestion, that are only known accurately in the post-PLE domain use Physical Estimation Flow to cover the physical factor in design exploration.

In our evaluation, we use the same target core aspect ratio and target core utilization rate to ensure the EDA tool treats each design in the candidate list equally. In this case, the tool will choose the best floorplan prediction concerning the PPA and physical factors such as wire length and routing congestion.

5. **Physical-aware RTL-inspired Neural Network** After sampling the initial dataset, prior works usually feed the data into the regressor model and begin to search the design space. However, the beginning of the search process is explorative, trying bold moves to see what will happen. The quality of exploration is essential to the later-on exploitation stage. Suppose the exploration is too conservative with limited knowledge about the design space in the post-PLE domain. In that case, the fine-tuned exploitation process will likely stick to the best-known local optimal. Hence, we propose constructing a

guide model that can first absorb the knowledge from the RTL domain and transfer the knowledge to the post-PLE domain. To transfer knowledge successfully, we will train the guide model in the RTL domain and fine-tune the guide model with selected initial samples in the post-PLE domain. Our results show that the accuracy and generalizability of the guide model are better than those directly trained by the selected initial samples in the post-PLE domain.

The naive solution is that we pre-trained a neural network (NN) model, an encoder-decoder model, with the RTL knowledge dataset and fine-tuned the model with the initial samples in the post-PLE domain. While the result of the prediction of execution time is quite precise, the other three objectives result in almost random output. They are even worse than training on the data with initial samples in the post-PLE domain. The failure means the domain gap between the execution cycles in the RTL domain and the area, power, and congestion in the post-PLE domain is too large and fails to transfer knowledge from the RTL to the post-PLE domain. To minimize the domain gap between these two domains, we devise a solution using a pseudo-labeled array technique to force the model warm-up in the pre-train stage and finalize with the post-PLE initial samples. The pseudo-labeled array consists of pseudo-labels of area, power, and congestion. The main idea behind the pseudo-labeled array technique is from the pseudo-labeling in a semi-supervised learning algorithm that uses a model trained on the labeled data to generate pseudo labels on the unlabeled data and train our guide model to fit both labeled and pseudo-labeled data.

To generate the pseudo-labeled array, we train an ensemble regression model with feature selection and cross-validation techniques on the initial samples in the post-PLE domain. *Boomerang* will select the best model and its feature selection from the model list and use it to make the pseudo-labeled array in the RTL dataset. The advantages of the pseudo-labeled array enable the NN model to predict better on the unsampled pool than direct training with the initial samples in the post-PLE domain. Thus, we can pre-train our NN model on the RTL dataset with the extra pseudo-labeled array and transfer the knowledge to the post-PLE domain, thereby reducing the domain gap.

6. **Pareto Forecasting** After training the NN model, we can predict the unsampled design space and select the potential Pareto set out in the forecast. However, this approach will not drastically increase the hypervolume from the same number of observations in the baseline model, which does not use the NN model to predict but directly searches the design space. The potential candidate provides a better exploration quality, and the physical-design-aware RTL-inspired neural network guides the search process to find the optimal faster in the later exploitation stage.

7. **Multi-objective Bayesian Optimization** After constructing the initial dataset, we can finally feed the data to the Gaussian process (GP) to learn the objective function in the post-PLE domain. The GP is a supervised learning method that can solve classification or regression problems. Due to the Gaussian process being a non-parametric probabilistic model, we do not have to make any assumption on the data.

Bayesian optimization (BO) is a technique for the global optimization of black-box functions. It is often used in the optimization of expensive-to-evaluate objectives like the VLSI design flow in the microarchitecture search problem. Bayesian optimization provides a principled method based on Bayes' Theorem to solve a global optimization problem efficiently and effectively. The optimization process is guided by a probabilistic model, such as GP, and leveraging the acquisition function to balance the exploration and the exploitation of the objective functions. Since our microarchitecture search is a multi-objective function problem, we use the qExpected-Hypervolume-Improvement (qEHVI) [17] to be our acquisition function for our multi-objective problem. The qEHVI function has the same concept as the expected improvement (EI) function, which calculates the expected hypervolume improvement in the multi-dimensional space.

V. Evaluation

A. Environment Setup

In this research, we use the chipyard [18] framework to generate the design with specific specs, the hammer framework [19] to run the VLSI design flow with our PLE add-on, and use ASAP 7nm predictive PDK [20] and version 28 cell libraries [21]. We use VCS/2021 for simulations, GENUS/21.12 for the physical synthesis flow, INNOVUS/21.12 for the physical back-end engine of physical synthesis, and JOULES/21.12 for the power analysis. To port the design to run through the physical estimation VLSI design flow, we have fixed some minor issues, such as the physical library, timing library, SRAMs macro, and design SDC. For the optimization process, we use Python/3.8, Botorch/0.7.3 [22], and Scikit-learn/1.0.2 with fixed random seed variables for reproducibility. Each optimization method runs 1000 trials and calculates its hypervolume difference mean and 95% confidence bound. To demonstrate our work with ground truth, we set the target design pool size to 50, for which we can get the true Pareto with brutal force, and the RTL dataset size is 1000. The benchmarks are *dhrystone*, *mm*, *mt-matmul*, and *mt-vvadd*. Each design costs 10-20 hours to run through the PLE flow.

B. Physical Layout Estimation with True-to-life Objective Space

The main goal of the microarchitecture design search is to find the most efficient parameter set in the objective space and shorten time-to-market with a competitive design specification. However, suppose the search is not based on the physical factor in a real-world scenario. In that case, the unrealistic Pareto set will misguide the design into the inefficient zone, which is a non-Pareto set and costs extra runtime on the improvement iterations.

Figure 5 shows the different Pareto sets in a non-physical estimation flow and objective space that were used in most prior works and our physical layout estimation flow with more true-to-life objective space. The two different cluster criteria

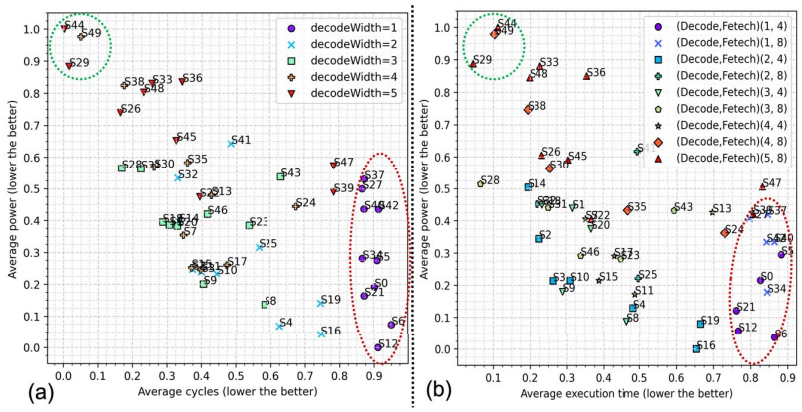


Fig. 5. Pareto sets comparison between (a) non-physical estimation synthesis flow with objective space used in most prior works and (b) PLE flow with more true-to-life objective space in our framework. The criteria of clustering in (a) using prior knowledge and the criteria of clustering in (b) using our adaptive cluster criteria sampling.

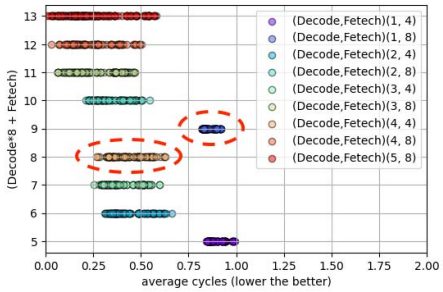


Fig. 6. The hardware cost versus execution cycles. The baseline model *Cluster-inspired-BO* assumes the two red dotted line circles are inefficient clusters.

in figure 5 also show the difference between (a) using prior knowledge and (b) our adaptive cluster criteria sampling. With more realistic objective space, in the green dotted line circle, we can discover that design *S44* has the smallest average execution cycle count, but overall *S29* has the shortest average execution time. On the other hand, using average execution cycles will cause an inaccurate Pareto frontier. Moreover, in the red dotted line circle, using our adaptive cluster criteria sampling can show the hidden difference between *(Decode, Fetch)(1,4)* and *(Decode, Fetch)(1,8)* that *(Decode, Fetch)(1,8)* is likely to be dominated by *(Decode, Fetch)(1,4)* in performance.

Overall, the former could only find 39% of the ground truth Pareto set, which is not ideal for a microarchitecture search. Hence, our physical-design-aware optimization framework will be based on the physical layout estimation flow data, providing a better correlation between specification search and physical layout.

C. Baseline Model

This subsection will introduce our baseline models that compare to our *Boomerang* in the evaluation. The baseline models are *RANDOM-BO*, *Cluster-inspired-BO*, *Reference-inspired-BO*. The *[RTED]* denotes whether the initial sample dataset uses the RTED method to draw samples from the design pool.

The baseline model *RANDOM-BO* uses random sampling on the initial dataset without using any cluster criteria. *RANDOM-BO* will apply the Bayesian optimization technique with the Gaussian optimization and explore the design space.

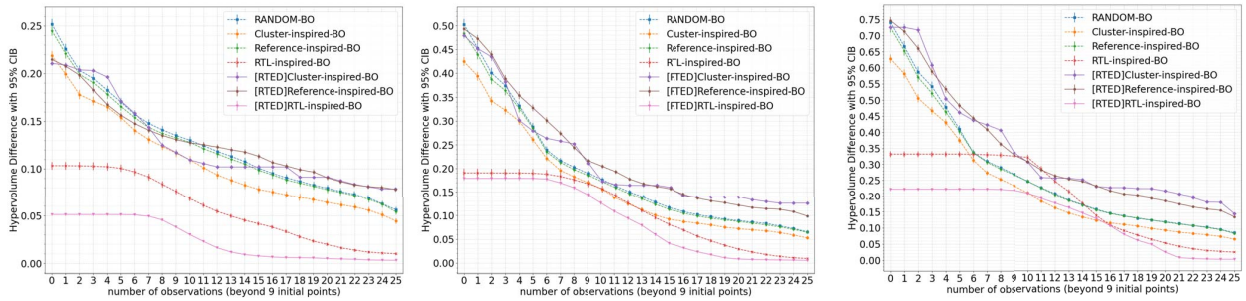
Reference-inspired-BO is motivated by the prior work [11] that uses *decodeWidth* to cluster the design pool and draw initial samples from each cluster. Although the observation of clustering is based on the execution cycles and power space only, the idea of clustering inspires our auto clustering method *Adaptive Cluster Criteria Sampling*.

Cluster-inspired-BO is our initial idea combining several prior works' advantages. *Cluster-inspired-BO* exploits the advantages of the RTL Knowledge Dataset with Adaptive Cluster Criteria Sampling that categorizes the sum of the cluster index parameter and the execution cycles space, as shown in Figure 6. *Cluster-inspired-BO* assumes the two-parameter weight is the same in summation, and the summation stands for the hardware cost, which is the lower, the better. In Figure 6, two clusters circled by a red dotted line exhibits an inconsistent trend of hardware cost that is not scaling with execution cycles compared to other clusters. In this case, *Cluster-inspired-BO* assumes those two red-circled clusters are not ideal for initial sampling since those clusters are inefficient in resource allocation. Therefore, *Cluster-inspired-BO* only randomly draws samples from clusters beside the two low-efficiency clusters into the initial dataset.

D. Microarchitecture Search in Two-objective Space

Figure 7(a) shows the result of each model in two-objective space (execution time and power). As our *Boomerang* model predicts n potential Pareto frontier design points, we waited for other models to obtain the same number of observations for fairness. This explains why the line of hypervolume difference in the first couple of observations is flat. In two-objective space, the possible Pareto frontier prediction gained significant hypervolume relative to other models in the early stage within the same observation count after the end of the flat line. Especially in the RTED version Physical-aware RTL-inspired neural network, the RTED process made the model transfer its knowledge better to the post-PLE domain than the model without RTED.

Furthermore, our method is the only model that benefits from the RTED process since RTED spreads the initial samples in the distance matrix space. Admittedly, maximizing the distance between initial samples can cover the entire parameter space, but it also tends to select over-parameterized or under-



(a) Two-objective: Execution time and power.

(b) Three-objective: Two-objective with area.

(c) Four-objective: Three-objective with congestion score.

Fig. 7. The y-axis is the hypervolume difference between the model and the ground truth, and the x-axis denotes the number of observations beyond nine initial samples. Each optimization method runs 1000 trials and calculates its mean of hypervolume difference and its 95% confidence bound with error bars.

parameterized design points into the initial sample set. The better coverage explains why it could boost the transfer learning quality with the pseudo-labeled array technique. After the Bayesian optimization kicked in, the model with the better-quality initial dataset guided the search for a better solution in the later exploitation stage. At the 25th observation, the best baseline model *Cluster-inspired-BO* achieved 0.45 hypervolume. On the other hand, our method only used 8-9 observations to achieve the same result, which saves 70% of the runtime.

E. Microarchitecture Search in Three-objective Space

As the objective space stacks up the dimensions, the difficulty of the search process escalates with the number of dimensions. For the three-objective space, we use execution time, power consumption, and area consumption as our objectives. In Figure 7(b), the advantage of getting higher hypervolume by Pareto Forecasting disappeared. However, the initial goal of potential Pareto prediction is to provide a better exploration instead of a better hypervolume. The result in the later exploitation stage showed a lower hypervolume difference compared to other methods. To achieve a 0.15 hypervolume difference, using our method saves 47% of the runtime, 7-8 observations, compared to the random initial sample method. At the 25th observation, the best baseline model *Cluster-inspired-BO* achieved 0.05 hypervolume. On the other hand, our method only used 14-15 observations to achieve the same result, which saves 40% of the runtime.

F. Microarchitecture Search in Four-objective Space

The three-objective space, such as execution time, power, and area consumption, can also be approximated through a traditional synthesis flow without the physical estimation technique. However, the fourth objective function *congestion score* can only be calculated with the physical estimation technique since the tool uses the physical library and engine to predict the routing quality.

The congestion score we used is the max hotspot score because such hotspots may contribute to lower design quality after physical design. The congestion affects the QoR of the design heavily, especially in advanced technology nodes. However, most of the design has a small congestion score, which can be safely ignored since the physical engine can eliminate the issue without a significant impact on the QoR. Hence, we are only concerned with the design with a relatively high congestion score. The user can define the threshold for which the congestion score is not an issue. In our case, we set the threshold in raw data to 100. If the raw congestion score is below 100, the congestion score will be overwritten with zero, which is considered as no severe congestion.

After pre-processing the congestion score, Figure 7(c) shows the search process. With the fourth dimension objective function *congestion score*, the search problem is very challenging. Because the prior knowledge of cluster criteria from *Reference-inspired-BO* does not guide the optimization process very well in the exploration stage, the performance in the later exploitation stage is only slightly better than the baseline model *RANDOM-BO*. Especially for baseline models with the RTED sampling technique, the performance was worse than the random initial sampling-based method due to the nature of maximizing the distance between initial samples in the RTED technique.

The benefit from potential Pareto prediction no longer existed in the early exploration stage. However, the Pareto prediction guided our method to lower hypervolume differences in the exploitation stage. At the 25th observation, the best baseline model *Cluster-inspired-BO* reached 0.07 hypervolume difference. However, our method reached 0.07 hypervolume within only 18 observations, which reduces 28% of the runtime. Moreover, our method accomplished a seven times smaller hypervolume difference than the best baseline *Cluster-inspired-BO* at the 25th observation.

VI. Conclusion and Future Work

To the best of our knowledge, the *Boomerang* is the first physical-aware framework for microarchitecture search with four objectives. It is a highly automated method without needing prior knowledge. *Boomerang* uses the adaptive cluster criteria technique to automatically select the cluster criteria

of the design space, and we use the RTED technique to obtain good enough initial samples. Besides, We use the pseudo-labeled array technique to minimize the domain gap issue and transfer the knowledge from the RTL knowledge dataset to adapt the physical-view objectives in the post-PLE domain with a physical-aware NN model. The framework also leverages the PLE flow to provide more accurate QoR and congestion while preventing long runtime place-and-route (PnR) in design space exploration, and *Boomerang* uses true-to-life objective space.

Consequently, in four-objective space, our method saved at least 28% of the runtime while delivering a reliable Pareto set compared to other baselines. Future work concerns deeper integration on RTL-inspired NN and the acquisition function of BO with deep kernel learning and using graph technique to learn deeper from the RTL domain.

REFERENCES

[1] Jonathan Bachrach, Huy Vo, Brian Richards, Yunsup Lee, Andrew Waterman, Rimas Avižienis, John Wawrzynek, and Krste Asanović. Chisel: Constructing hardware in a scala embedded language. In *DAC Design Automation Conference 2012*, pages 1212–1221, 2012.

[2] F. Zaruba and L. Benini. The cost of application-class processing: Energy and performance analysis of a linux-ready 1.7-ghz 64-bit risc-v core in 22-nm fdsoi technology. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 27(11):2629–2640, Nov 2019.

[3] Hasan Genc, Seah Kim, Alon Amid, Ameer Haj-Ali, Vighnesh Iyer, Pranav Prakash, Jerry Zhao, Daniel Grubb, Harrison Liew, Howard Mao, Albert Ou, Colin Schmidt, Samuel Steffl, John Wright, Ion Stoica, Jonathan Ragan-Kelley, Krste Asanovic, Borivoje Nikolic, and Yakun Sophia Shao. Gemmini: Enabling systematic deep-learning architecture evaluation via full-stack integration. In *Proceedings of the 58th Annual Design Automation Conference (DAC)*, 2021.

[4] Abraham Gonzalez, Jerry Zhao, Ben Korpan, Hasan Genc, Colin Schmidt, John Wright, Ayan Biswas, Alon Amid, Farhana Sheikh, Anton Sorokin, Sirisha Kale, Mani Yalamanchi, Ramya Yarlagadda, Mark Flannigan, Larry Abramowitz, Elad Alon, Yakun Sophia Shao, Krste Asanović, and Borivoje Nikolić. A 16mm2 106.1 gops/w heterogeneous risc-v multi-core multi-accelerator soc in low-power 22nm finfet. In *ESSCIRC 2021 - IEEE 47th European Solid State Circuits Conference (ESSCIRC)*, pages 259–262, 2021.

[5] Hadi Esmaeilzadeh, Soroush Ghodrati, Andrew B. Kahng, Joon Kyung Kim, Sean Kinzer, Sayak Kundu, Rohan Mahapatra, Susmita Dey Manasi, Sachin S. Sapatnekar, Zhiang Wang, and Ziqing Zeng. Physically accurate learning-based performance prediction of hardware-accelerated ml algorithms. In *2022 ACM/IEEE 4th Workshop on Machine Learning for CAD (MLCAD)*, pages 119–126, 2022.

[6] Felix Last and Ulf Schlichtmann. Feeding hungry models less: Deep transfer learning for embedded memory ppa models : Special session. In *2021 ACM/IEEE 3rd Workshop on Machine Learning for CAD (MLCAD)*, pages 1–6, 2021.

[7] P.J. Joseph, Kapil Vaswani, and M.J. Thazhuthaveetil. Construction and use of linear regression models for processor performance analysis. In *The Twelfth International Symposium on High-Performance Computer Architecture, 2006.*, pages 99–108, 2006.

[8] Dandan Li, Shuzhen Yao, Yu-Hang Liu, Senzhang Wang, and Xian-He Sun. Efficient design space exploration via statistical sampling and adaboost learning. In *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, 2016.

[9] Dandan Li, Shuzhen Yao, and Ying Wang. Processor design space exploration via statistical sampling and semi-supervised ensemble learning. *IEEE Access*, 6:25495–25505, 2018.

[10] Kaijie Feng, Xiaoya Fan, Jianfeng An, Chuxi Li, Kaiyue Di, and Jiangfei Li. Acdse: A design space exploration method for cnn accelerator based on adaptive compression mechanism. *ACM Trans. Embed. Comput. Syst.*, jun 2022. Just Accepted.

[11] Chen Bai, Qi Sun, Jianwang Zhai, Yuzhe Ma, Bei Yu, and Martin D.F. Wong. Boom-explorer: Risc-v boom microarchitecture design space exploration framework. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pages 1–9, 2021.

[12] Zhonghua Zhou, Sunmeet Chahal, Tsung-Yi Ho, and André Ivanov. Supervised-learning congestion predictor for routability-driven global routing. In *2019 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, pages 1–4, 2019.

[13] Robert Kirby, Saad Godil, Rajarshi Roy, and Bryan Catanzaro. Congestionnet: Routing congestion prediction using deep graph neural networks. In *2019 IFIP/IEEE 27th International Conference on Very Large Scale Integration (VLSI-SoC)*, pages 217–222, 2019.

[14] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna, Somayeh Sardashti, Rathijit Sen, Korey Sewell, Muhammad Shoaib, Nilay Vaish, Mark D. Hill, and David A. Wood. The gem5 simulator. *SIGARCH Comput. Archit. News*, 39(2):1–7, aug 2011.

[15] Kai Yu, Jinbo Bi, and Volker Tresp. Active learning via transductive experimental design. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, page 1081–1088, New York, NY, USA, 2006. Association for Computing Machinery.

[16] Jerry Zhao, Ben Korpan, Abraham Gonzalez, and Krste Asanovic. Sonicboom: The 3rd generation berkeley out-of-order machine. May 2020.

[17] Samuel Daulton, Maximilian Balandat, and Eytan Bakshy. Differentiable expected hypervolume improvement for parallel multi-objective bayesian optimization. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc.

[18] Alon Amid, David Biancolin, Abraham Gonzalez, Daniel Grubb, Sagar Karandikar, Harrison Liew, Albert Magyar, Howard Mao, Albert Ou, Nathan Pemberton, Paul Rigge, Colin Schmidt, John Wright, Jerry Zhao, Yakun Sophia Shao, Krste Asanović, and Borivoje Nikolić. Chipyard: Integrated design, simulation, and implementation framework for custom socs. *IEEE Micro*, 40(4):10–21, 2020.

[19] Harrison Liew, Daniel Grubb, John Wright, Colin Schmidt, Nayiri Krzysztofowicz, Adam Izraelevitz, Edward Wang, Krste Asanović, Jonathan Bachrach, and Borivoje Nikolić. Hammer: A modular and reusable physical design flow tool: Invited. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*, DAC '22, page 1335–1338, New York, NY, USA, 2022. Association for Computing Machinery.

[20] Lawrence T. Clark, Vinay Vashishtha, Lucian Shifren, Aditya Gujja, Saurabh Sinha, Brian Cline, Chandarasekaran Ramamurthy, and Greg Yeric. Asap7: A 7-nm finfet predictive process design kit. *Microelectronics Journal*, 53:105–115, 2016.

[21] Vinay Vashishtha, Manoj Vangala, and Lawrence T. Clark. Asap7 predictive design kit development and cell design technology co-optimization: Invited paper. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 992–998, 2017.

[22] Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. In *Advances in Neural Information Processing Systems 33*, 2020.