BOOM-Explorer: RISC-V BOOM Microarchitecture Design Space Exploration Framework

Chen Bai[1], Qi Sun[1], Jianwang Zhai[2], Yuzhe Ma[1], Bei Yu[1], Martin D.F. Wong[1]
[1]The Chinese University of Hong Kong [2]Tsinghua University

***Abstract*—The microarchitecture design of a processor has been increasingly difficult due to the large design space and time-consuming verification flow. Previously, researchers rely on prior knowledge and cycle-accurate simulators to analyze the performance of different microarchitecture designs but lack sufficient discussions on methodologies to strike a good balance between power and performance. This work proposes an automatic framework to explore microarchitecture designs of the RISC-V Berkeley Out-of-Order Machine (BOOM), termed as BOOM-Explorer, achieving a good trade-off on power and performance. Firstly, the framework utilizes an advanced microarchitecture-aware active learning (MicroAL) algorithm to generate a diverse and representative initial design set. Secondly, a Gaussian process model with deep kernel learning functions (DKL-GP) is built to characterize the design space. Thirdly, correlated multi-objective Bayesian optimization is leveraged to explore Pareto-optimal designs. Experimental results show that BOOM-Explorer can search for designs that dominate previous arts and designs developed by senior engineers in terms of power and performance within a much shorter time.**

I. Introduction

Recently, RISC-V, an open-source instruction set architecture (ISA) gains much attention and also receives strong support from academia and industry. Berkeley Out-of-Order Machine (BOOM) [1], [2], a RISC-V design fully in compliance with RV64GC instructions, is competitive in power and performance against low-power, embedded out-of-order cores in academia. By adopting Chisel hardware construction language [3], BOOM can be parametric, providing great opportunities to explore a series of microarchitecture designs that have a better balance on power and performance for different purposes of use.

Microarchitecture defines the implementation of an ISA in a processor. Due to different organizations and combinations of components inside a processor, microarchitecture designs under a specific technology process can affect power dissipation, performance, die area, *etc.* of a core [4], [5]. Finding a good microarchitecture that can accommodate a good balance between power and performance is a notorious problem because of two restrictions. On the one hand, the design space is extremely large and the size of it can be exponential with more components to be considered, *e.g.*, special queues, buffers, branch predictors, vector execution unit, external co-processors, *etc.* Thus, we cannot traverse and evaluate each microarchitecture to retrieve the best one. On the other hand, it costs a lot of time to acquire metrics, *e.g.*, power, performance, *etc.* when we verify one microarchitecture with diverse benchmarks.

In industry, the traditional solution is based on prior engineering experience from computer architects. However, it lacks scalability for newly emerged processors. In academia, to overcome these two obstacles, researchers proposed various arts, which can be categorized as two kinds of methodologies. First, in view of the difficulty in constructing an analytical model, researchers can otherwise characterize a microarchitecture design space with fewer samples as much as possible by leveraging statistical sampling and predictive black-box models. Li *et al.* [6] proposed AdaBoost Learning with novel sampling algorithms to explore the design space. Second, to search for more designs within a limited time budget, researchers often rely on coarse-grained simulation infrastructure rather than a register-transfer level (RTL) verification flow to accelerate the process [7]–[10]. Moreover, by decreasing redundant overhead, the simulation can be further speed up [11]–[14].

Unfortunately, both of these academic solutions contain several limitations. In the first place, despite the fact that statistical analysis performs well when highly reliable models can be constructed, it fails to embed prior knowledge on microarchitectures to further improve design space exploration. For another, to accelerate the simulation, coarse-grained simulation infrastructure is used widely. Nevertheless, most of them lose sufficient accuracy, especially for distinct processors. The low quality of results is generated often due to the misalignment between simulation and real running behaviors of processors. More importantly, because it is difficult to model the power dissipation of modern processors at the architecture level [15], some infrastructure cannot provide power value, *e.g.*, [8], [10]. In general, because of the aforementioned limitations, academia lacks sufficient discussions on methodologies that can explore microarchitecture designs achieving a good trade-off between power and performance.

In this paper, following the first strategy, we propose BOOM-Explorer address these issues. In BOOM-Explorer, without sacrificing the accuracy of a predictive model, we embed prior knowledge of BOOM to form a microarchitecture-aware active learning (MicroAL) algorithm based on transductive experimental design [16] by utilizing BOOM RTL samples among the entire design space as few as possible. Secondly, a novel Gaussian process model with deep kernel learning functions (DKL-GP) initialized through MicroAL, is proposed to characterize the features of different microarchitectures. The design space is then explored via correlated multi-objective Bayesian optimization flow [17] based on

2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD) | 978-1-6654-4507-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICCAD51958.2021.9643455

DKL-GP. Our framework can not only take advantage of fewer microarchitecture designs as much as possible but also helps us to find superior designs that have a better balance between power and performance.

Our contributions are summarized as follows:

- A microarchitecture-aware active learning methodology based on transductive experimental design is introduced for the first time to attain the most representative designs from an enormous RISC-V BOOM design space.
- A novel Gaussian process model with deep kernel learning and correlated multi-objective Bayesian optimization are leveraged to characterize the microarchitecture design space. With the help of DKL-GP, Pareto optimality is explored between power and performance.
- We verify our framework with BOOM under advanced 7-nm technology. The experimental results demonstrate the outstanding performance of BOOM-Explorer on various BOOM microarchitectures.

The remainder of this paper is organized as follows. Section II introduces the RISC-V BOOM core and the problem formulation. Section III provides detailed explanations on the framework. Section IV conducts several experiments on BOOM core to confirm the outstanding performance of the proposed framework. Finally, Section V concludes this paper.

II. Preliminaries

A. RISC-V BOOM Core

BOOM is an open-source superscalar out-of-order RISC-V processor in academia and it is proved to be industry-competitive in low-power, embedded application scenarios [1], [2].

Fig. 1 demonstrates the organization of BOOM. Consist of four main parametric modules, *i.e.*, `FrontEnd`, `IDU`, `EU`, and `LSU`, BOOM can execute benchmarks in distinct behaviors via choosing different candidate values for each component inside these modules. `FrontEnd` fetches instructions from L2 Cache, packs these instructions as a sequence of fetch packages, and sends them to `IDU`. `IDU` decodes instructions as micro-ops and dispatches these micro-ops *w.r.t.* their categories to issue queues in `EU`, the latter of which, triggered by corresponding micro-ops and related logics, is responsible for manipulating operands in an out-of-order manner. Finally, some memory-related operations, *i.e.*, loading data and storing data, interact with `LSU` after `EU` calculates the results. In addition, BOOM also integrates branch predictors, floating-point execution units, vector execution units, *etc.*

Thanks to the parameterized modules provided by BOOM, various BOOM microarchitectures can be acquired by configuring the core with different parameters. Thus, divergent trade-offs between power dissipation and performance to meet various design requirements can be achieved, *e.g.*, low-power, and embedded applications. However, a satisfying microarchitecture design is non-trivial to be found.

Across all parametric modules, a microarchitecture design space of BOOM is constructed and shown in TABLE I. Each row of TABLE I defines structures of a component inside

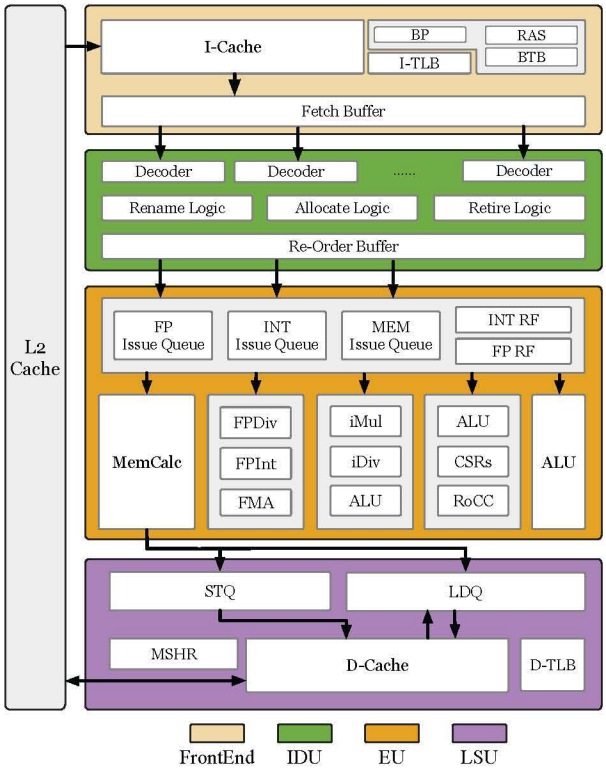

Fig. 1 BOOM implements a ten-stage pipeline, *i.e.*, *Fetch*, *Decode*, *Register Rename*, *Dispatch*, *Issue*, *Register Read*, *Execute*, *Memory*, *Writeback* and *Commit*.

the module. RasEntry and BranchCount in `FrontEnd` are considered since they have great impacts on the behaviors of branch predictions, and thus incur different power and performance. Because caches, *e.g.*, D-Cache in `LSU`, often runs at a lower frequency compared to other modules, the component might be hotspots when many memory-related requests occur in the instructions pipeline. A suitable structure of D-Cache can alleviate the burden, therefore different organizations of D-Cache (*i.e.*, associativity, block width, TLB size, *etc.*) are also included for exploration. Besides, I-Cache is also considered in the design space.

Different BOOM microarchitecture designs can be constructed with various combinations of candidate values. However, some combinations do not observe constraints of BOOM design specifications as shown in TABLE II. Thus they are illegal and cannot be compiled to Verilog. For example, each entry of the reorder buffer traces status of every in-flight but decoded instruction in the pipeline. If a microarchitecture does not obey rule 2, reorder buffer may not reserve enough entries for each decoded instruction or may contain redundant entries that cannot be fully used at all. The last three rules in TABLE II are added to simplify the design space. They require the same number of entries or registers in respective components and their additions will not affect the performance of BOOM-Explorer. After we prune the design space *w.r.t.* rules in TABLE II, the size of the legal microarchitecture design space is approximately 1.6×10^8.

TABLE I Microarchitecture Design Space of BOOM

Module	Component	Descriptions	Candidate values
FrontEnd	FetchWidth	Number of instructions the fetch unit can retrieve once	4, 8
	FetchBufferEntry	Entries of the fetch buffer register	8, 16, 24, 32, 35, 40
	RasEntry	Entries of the Return Address Stack (RAS)	16, 24, 32
	BranchCount	Entries of the Branch Target Buffer (BTB)	8, 12, 16, 20
	ICacheWay	Associate sets of L1 I-Cache	2, 4, 8
	ICacheTLB	Entries of Table Look-aside Buffer (TLB) in L1 I-Cache	8, 16, 32
	ICacheFetchBytes	Unit of line capacity that L1 I-Cache supports	2, 4
IDU	DecodeWidth	Number of instructions the decoding unit can decode once	1, 2, 3, 4, 5
	RobEntry	Entries of the reorder buffer	32, 64, 96, 128, 130
	IntPhyRegister	Number of physical integer registers	48, 64, 80, 96, 112
	FpPhyRegister	Number of physical floating-point registers	48, 64, 80, 96, 112
EU	MemIssueWidth	Number of memory-related instructions that can issue once	1, 2
	IntIssueWidth	Number of integer-related instructions that can issue once	1, 2, 3, 4, 5
	FpIssueWidth	Number of floating-point-related instructions that can issue once	1, 2
LSU	LDQEntry	Entries of the Loading Queue (LDQ)	8, 16, 24, 32
	STQEntry	Entries of the Store Queue (STQ)	8, 16, 24, 32
	DCacheWay	Associate sets of L1 D-Cache	2, 4, 8
	DCacheMSHR	Entries of Miss Status Handling Register (MSHR)	2, 4, 8
	DCacheTLB	Entries of Table Look-aside Buffer (TLB) in L1 D-Cache	8, 16, 32

TABLE II Constraints of BOOM design specifications

Rule	Descriptions
1	FetchWdith $\geq$ DecodeWidth
2	RobEntry | DecodeWidth [+]
3	FetchBufferEntry $>$ FetchWidth
4	FetchBufferEntry | DecodeWidth
5	fetchWidth $= 2\times$ ICacheFetchBytes
6	IntPhyRegister $=$ FpPhyRegister
7	LDQEntry $=$ STQEntry
8	MemIssueWidth $=$ FpIssueWidth

[+] "|" means RobEntry should be divisible by DecodeWidth.

B. Problem Formulation

Definition 1 (Microarchitecture Design)**.** *Microarchitecture design is to define a combination of candidate values given in TABLE I. A microarchitecture design is legal if it satisfies all constraints as referred to in TABLE II. Every legal microarchitecture design to be determined is encoded as a feature vector among the entire design space $\mathcal{D}$. The feature vector is denoted as $\boldsymbol{x}$. For convenience, microarchitecture and microarchitecture design in the following sections are the same.*

Definition 2 (Power)**.** *The power is to be defined as the summation of dynamic power dissipation, short-circuit power dissipation, and leakage power dissipation.*

Definition 3 (Clock Cycle)**.** *The clock cycle is to be defined as the clock cycles consumed when a BOOM microarchitecture design runs a specific benchmark.*

Provided with the same benchmark, power and clock cycle are a pair of trade-off metrics since the lower cycles are, the more power will be dissipated when a design integrates more hardware resources to accelerate instructions execution. Together, They reflect whether a microarchitecture design is good or not. Power and clock cycle are denoted as $\boldsymbol{y}$.

Definition 4 (Pareto Optimality)**.** *For a n-dimensional minimization problem, an objective vector $f(\boldsymbol{x})$ is said to be dominated by $f(\boldsymbol{x'})$ if*

$$\begin{aligned} &\forall i \in [1,n], \quad f_i(\boldsymbol{x}) \leq f_i(\boldsymbol{x'}); \\ &\exists j \in [1,n], \quad f_j(\boldsymbol{x}) < f_j(\boldsymbol{x'}). \end{aligned} \tag{1}$$

In this way, we denote $\boldsymbol{x'} \succcurlyeq \boldsymbol{x}$. In the entire design space, a set of designs that are not dominated by any other is called the Pareto-optimal set and they form the Pareto optimality in this space.

In this paper, our objective is to explore Pareto optimality defined in Definition 4 *w.r.t.* power and clock cycle for various BOOM microarchitectures. Due to the power and clock cycle are a pair of negatively correlated metrics, a microarchitecture belonged to the Pareto-optimal set cannot improve one metric without sacrificing another metric. To guarantee high quality of results, rather than use coarse-grained simulation infrastructure introduced in Section I, we evaluate power and performance using commercial electronic automation (EDA) tools and they are referred to as the VLSI flow. Based on the above definitions, our problem can be formulated.

Problem 1 (BOOM Microarchitecture Design Space Exploration)**.** *Given a search space $\mathcal{D}$, each microarchitecture design inside $\mathcal{D}$ is regarded as a feature vector $\boldsymbol{x}$. Power and clock cycle form the power-performance space $\mathcal{Y}$. Through VLSI flow, the power and cycles $\boldsymbol{y} \in \mathcal{Y}$ can be obtained according to $\boldsymbol{x}$. BOOM microarchitecture design space exploration is to be defined as to find a series of features $\boldsymbol{X}$ that form the Pareto optimality among the corresponding $\boldsymbol{Y} \subset \mathcal{Y}$. Hence, $\boldsymbol{Y} = \{\boldsymbol{y} | \boldsymbol{y'} \not\succcurlyeq \boldsymbol{y}, \forall \boldsymbol{y'} \in \mathcal{Y}\}$, $\boldsymbol{X} = \{\boldsymbol{x} | f(\boldsymbol{x}) \in \boldsymbol{Y}, \forall \boldsymbol{x} \in \mathcal{D}\}$.*

III. BOOM-Explorer

A. Overview of BOOM-Explorer

Fig. 2 shows an overview of BOOM-Explorer. Firstly, the active learning algorithm MicroAL is adopted to sample a set of initial microarchitectures from the large design space. In this step, domain-specific knowledge is used as the prior information to guide the sampling of the initial designs. Then, a Gaussian process model with deep kernel learning functions (DKL-GP) is built on the initial set. To explore the optimal microarchitecture, the multi-objective Bayesian optimization algorithm is used, with the Expected Improvement of Pareto Hypervolume as the acquisition function, and the DKL-GP model as the surrogate model. During this process, BOOM-Explorer interacts with the VLSI flow to get the accurate performance and power values of designs according to different benchmarks. Finally, The outputs of BOOM-Explorer are the set of explored microarchitectures in the iterative optimization process, and the Pareto optimality is gained from the set.

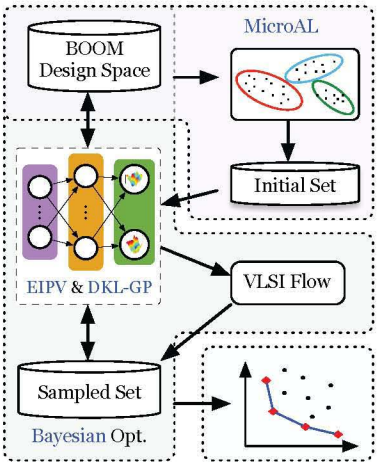

Fig. 2 Overview of the proposed BOOM-Explorer.

B. Microarchitecture-aware Active Learning Algorithm

Due to the time-consuming VLSI flow, to save time, only a limited number of designs will be synthesized practically to obtain power and performance. To guarantee that adequate information is covered in the data set, two principles are considered during the initialization. First, feature vectors should cover the entire design space uniformly. Second, their diversity should fully represent the characteristics of the design space. Within a limited time budget, only push the most representative microarchitecture to VLSI flow can we alleviate the burden to get power and performance.

A naive solution is to sample microarchitectures randomly. In literature, most previous works [18], [19] choose this simple method directly for convenience. In addition, by appraising the importance of each feature vector with suitable distance measurement, greedy sampling [20] can be facilitated to select representative microarchitecture designs.

To further improve the sampling, orthogonal design [6], [21] is also utilized to pick up dissimilar microarchitectures that are distributed orthogonally across the design space.

Algorithm 1 TED$(\mathcal{U}, \mu, b)$

Require: $\mathcal{U}$ is the unsampled microarchitecture design space, μ is a normalization coefficient, and b is the number of samples to draw.

Ensure: $\mathcal{X}$: the sampled set with $|\mathcal{X}| = b$.

1: $\mathcal{X} \leftarrow \emptyset,\ \mathrm{K}_{uu'} \leftarrow f(u, u'),\ \forall u, u' \in \mathcal{U}$;
2: **for** $i = 1 \rightarrow b$ **do**
3: $\quad \boldsymbol{x}_* \leftarrow \arg\max_{\boldsymbol{x} \in \mathcal{U}} \mathrm{Tr}[\mathrm{K}_{\mathcal{U}\boldsymbol{x}}(\mathrm{K}_{\boldsymbol{x}\boldsymbol{x}} + \mu \boldsymbol{I})^{-1}\mathrm{K}_{\boldsymbol{x}\mathcal{U}}]$; ▷ $\mathrm{K}_{\mathcal{U}\boldsymbol{x}}$, $\mathrm{K}_{\boldsymbol{x}\boldsymbol{x}}$ and $\mathrm{K}_{\boldsymbol{x}\mathcal{U}}$ are calculated via f *w.r.t.* corresponding columns in K
4: $\quad \mathcal{X} \leftarrow \mathcal{X} \cup \boldsymbol{x}_*,\ \mathcal{U} \leftarrow \mathcal{U} \setminus \boldsymbol{x}_*$;
5: $\quad \mathrm{K} \leftarrow \mathrm{K} - \mathrm{K}_{\mathcal{U}\boldsymbol{x}_*}(\mathrm{K}_{\boldsymbol{x}_*\boldsymbol{x}_*} + \mu \boldsymbol{I})^{-1}\mathrm{K}_{\boldsymbol{x}_*\mathcal{U}}$;
6: **end for**
7: **return** The sampled set $\mathcal{X}$;

Nevertheless, the aforementioned methodologies are failed to capture the key components of different microarchitectures that bring great impacts to the trade-off in the power-performance space.

Recently, witnessing the great performance improvement attained by transductive experimental design (TED) in the design space exploration of high-level synthesis [22], [23], compilation and deployment of deep neural networks [24], and *etc.*, we introduce this method into the exploration of microarchitectures for the first time.

TED tends to choose microarchitecture that can spread across the feature space to retain most of the information among the whole design space [16]. A pool of representative feature vectors can be acquired with high mutual divergences, by iteratively maximizing the trace of the distance matrix constructed on a newly sampled design and unsampled ones. Algorithm 1 shows the backbone of TED, where f represents the distance function used in computing the distance matrix K. Note that any suitable distance functions can be applied without restrictions.

Unfortunately, TED cannot fully guarantee to generate a good initial data set owing to a lack of prior knowledge of microarchitecture designs. We are motivated to embed the domain knowledge to improve its performance.

DecodeWidth as referred to in TABLE I decides the maximal number of instructions to be decoded as corresponding micro-ops simultaneously. In consequence, it can affect the execution bandwidth of `EU` and `LSU` (*i.e.*, instructions executed per clock cycle). Assigning a larger candidate value to DecodeWidth and allocating a balanced amount of hardware resources, on the one hand, can lead to greater performance improvement. On the other hand, power dissipation will also increase significantly. By clustering *w.r.t.* DecodeWidth, the power-performance space can be separated along the potential Pareto optimality, as shown in Fig. 3. Each cluster in Fig. 3 represents a group of microarchitectures with different candidate values for DecodeWidth in the power-performance space. The entire design space is discrete and non-smooth but nonetheless a large number of microarchitectures with the same DecodeWidth achieve similar power-performance

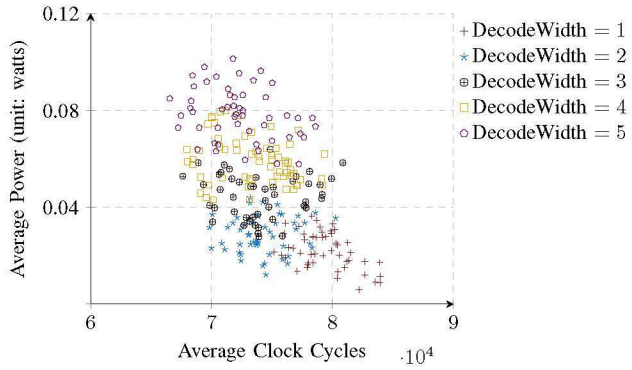

Fig. 3 Clustering *w.r.t.* DecodeWidth

characteristics within their sub-regions respectively. It inspires us that we can select microarchitectures on the possible sub-area from the initial design space, to better cover the entire design space at the same time improve the diversity of samples.

Inside each cluster, Algorithm 1 can be applied instead of choosing the centroid to enlarge the initial data set. The clustering *w.r.t.* DecodeWidth, together with TED, forms MicroAL and the pseudo code is detailed in Algorithm 2.

First, we cluster the entire design space according to Φ, which is the distance function with a higher penalty along the dimension of DecodeWidth. One possible alternative can be $\Phi = (\boldsymbol{x}_i - \boldsymbol{c}_j)^T \boldsymbol{\Lambda} (\boldsymbol{x}_i - \boldsymbol{c}_j)$, with $i \in \{1, \cdots, |\mathcal{U}|\}$ and $j \in \{1, \cdots, k\}$, where $\boldsymbol{\Lambda}$ is a pre-defined diagonal weight matrix. Next, we apply TED for each cluster to sample the most representative feature vectors, *i.e.*, line 9 in Algorithm 2. Finally, containing all of the sampled microarchitectures, the initial data set is formed.

C. Gaussian Process with Deep Kernel Learning

Given the initial data set, it is hard to build a reliable model to fully capture the characteristics of the design space yet.

However, thanks to robustness and non-parametric approximation features reside in Gaussian process (GP) models, they have been applied in various domains [24]–[26]. In view of the success, BOOM-Explorer adopts GP as well.

Assume that we have feature vectors $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ...\boldsymbol{x}_n\}$ and they index a set of corresponding power or clock cycles $\boldsymbol{y} = \{y_1, y_2, ..., y_n\}$. GP provides a prior over the value function f as $f(\boldsymbol{x}) \sim \mathcal{GP}(\mu, k_{\boldsymbol{\theta}})$, where μ is the mean value and the kernel function k is parameterized by $\boldsymbol{\theta}$. Then, Gaussian distributions can be constructed with any collection of value functions $\boldsymbol{f}$ according to Equation (2)

$$\boldsymbol{f} = [f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), ...f(\boldsymbol{x}_n)]^T \sim \mathcal{N}(\boldsymbol{\mu}, \mathrm{K}_{\boldsymbol{XX}|\boldsymbol{\theta}}), \quad (2)$$

where $\mathrm{K}_{\boldsymbol{XX}|\boldsymbol{\theta}}$ is the intra-covariance matrix among all feature vectors and calculated via $\left[\mathrm{K}_{\boldsymbol{XX}|\boldsymbol{\theta}}\right]_{ij} = k_{\boldsymbol{\theta}}(\boldsymbol{x}_i, \boldsymbol{x}_j)$. A Gaussian noise $\mathcal{N}(f(\boldsymbol{x}), \sigma_e^2)$ is necessary to model uncertainties of power or clock cycles generated by different microarchitecture designs. Thus, given a newly sampled feature vector $\boldsymbol{x}_*$, the predictive joint distribution f_* that depends on $\boldsymbol{y}$ can be

Algorithm 2 MicroAL$(\mathcal{U}, \mu, b)$

Require: $\mathcal{U}$ is the unsampled microarchitecture design space, μ is a normalization coefficient, b is the number of samples that to draw.
Ensure: $\mathcal{X}$: the sampled set with $|\mathcal{X}| = b$.
1: $\mathcal{X} \leftarrow \emptyset$;
2: initialize k clusters randomly with the centroids set $\boldsymbol{C} = \{\boldsymbol{c}_1, \boldsymbol{c}_2, ..., \boldsymbol{c}_k\}$ from $\mathcal{U}$;
3: **while** not converged **do**
4: $\quad c^i = \underset{j \in \{1,2,...,k\}}{\arg\min} \ \Phi\,(\boldsymbol{x}_i - \boldsymbol{c}_j), \ \forall \boldsymbol{x}_i \in \mathcal{U}$; ▷ Φ is the distance function considering DecodeWidth
5: $\quad \boldsymbol{c}_j = \frac{\sum_{i=1}^{|\mathcal{U}|} \mathbb{1}_{\{c^i = j\}} \boldsymbol{x}_i}{\sum_{i=1}^{|\mathcal{U}|} \mathbb{1}_{\{c_i = j\}}}, \ \forall j \in \{1, 2, ..., k\}$;
6: **end while**
7: $\mathcal{C} \leftarrow$ neighborhood of $\boldsymbol{c}_i \in \boldsymbol{C}, \ \forall i \in \{1, 2, ..., k\}$;
8: **for** $\mathcal{K}$ in $\mathcal{C}$ **do**
9: $\quad \hat{\mathcal{X}} = \textbf{TED}(\mathcal{K}, \mu, \lfloor \frac{b}{k} \rfloor)$; ▷ Algorithm 1
10: $\quad \mathcal{X} = \mathcal{X} \cup \hat{\mathcal{X}}$;
11: **end for**
12: **return** The sampled set $\mathcal{X}$;

calculated according to Equation (3)

$$f_* | \boldsymbol{y} \sim \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu} \\ \mu_*\end{bmatrix}, \begin{bmatrix}\mathrm{K}_{\boldsymbol{XX}|\boldsymbol{\theta}} + \sigma_e^2 \boldsymbol{I} & \mathrm{K}_{\boldsymbol{X}\boldsymbol{x}_*|\boldsymbol{\theta}} \\ \mathrm{K}_{\boldsymbol{x}_*\boldsymbol{X}|\boldsymbol{\theta}} & k_{\boldsymbol{x}_*\boldsymbol{x}_*|\boldsymbol{\theta}}\end{bmatrix}\right). \quad (3)$$

By maximizing the marginal likelihood of GP, $\boldsymbol{\theta}$ is optimized to sense the entire design space. Nevertheless, the performance of GP normally depends on the expressiveness and hyper-parameters of kernel functions $k_{\boldsymbol{\theta}}$, *e.g.*, radial bias functions, and *etc.* Therefore, a suitable kernel function is necessary to the performance of GP.

In the recent years, deep neural networks (DNN) have shown great potential in various applications and tasks as the black-box model to extract useful features [27]–[29]. Thus, with the help of DNN as a meta-learner for kernel functions, we can relieve workloads in tuning hyper-parameters of kernel functions. By leveraging multi-layer non-linear transformations and weights to calibrate kernel functions with Equation (4) [30], deep kernel functions can provide better performance.

$$k_{\boldsymbol{\theta}}(\boldsymbol{x}_i, \boldsymbol{x}_j) \rightarrow k_{\boldsymbol{w},\boldsymbol{\theta}}(\varphi(\boldsymbol{x}_i, \boldsymbol{w}), \varphi(\boldsymbol{x}_j, \boldsymbol{w})) \quad (4)$$

φ in Equation (4) denotes non-linear transformation layers stacked by DNN and $\boldsymbol{w}$ denotes weights in DNN. Enhanced with the expressive power of DNN, DKL-GP is constructed and then plugged into Bayesian optimization as the surrogate model.

D. Correlated Multi-Objective Design Exploration

Notwithstanding DKL-GP can be used to evaluate a single object (*i.e.*, power or clock cycles) well, to find the Pareto-optimal set still remains an issue, especially for such negatively correlated objectives.

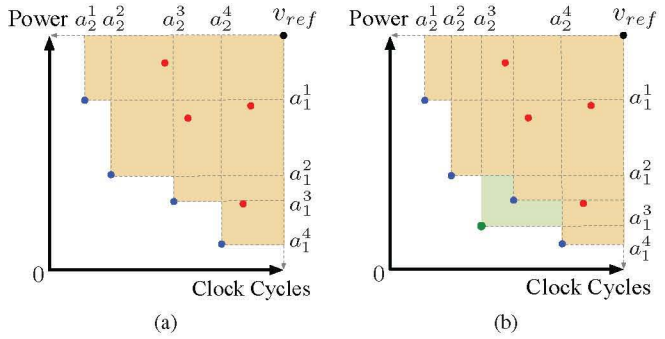

Fig. 4 An example of Pareto hypervolume is shown in the power-performance space. (a) The region covered in orange is dominated by the currently explored Pareto-optimal set denoted as circles in blue. Circles in red denote dominated microarchitecture designs. (b) The circle in green denotes an explored potential candidates of the Pareto-optimal set among the entire design space. EIPV is represented as the area of sub-region colored in light green.

A traditional methodology usually integrates different acquisition functions to solve it. Lyu *et al.* [31] combines Expectation Improvement (EI), Probability Improvement (PI) and Confidence Bound (*i.e.*, UCB and LCB) to form a multi-objective optimization framework in analog circuit design. It still leads to sub-optimal results except that we can select a good composite of acquisition functions for specific problems. To solve the problem more efficiently, we introduce Expected Improvement of Pareto Hypervolume (EIPV) [32] and demonstrate its usability to characterize the trade-off in the power-performance space of different microarchitecture designs.

In our problem, a better microarchitecture can not only run faster (*i.e.*, it gets fewer average clock cycles among all benchmarks) but also dissipate less power. Given a reference point $\boldsymbol{v}_{\text{ref}} \in \mathcal{Y}$, Pareto hypervolume bounded above from $\boldsymbol{v}_{\text{ref}}$ is the *Lebesgue measure* of the space dominated by the Pareto optimality as shown in Fig. 4(a) [33]. The shaded area in orange, indicating Pareto hypervolume *w.r.t.* the current Pareto-optimal set $\mathcal{P}(\mathcal{Y})$ is calculated by Equation (5)

$$\text{PVol}_{\boldsymbol{v}_{\text{ref}}}(\mathcal{P}(\mathcal{Y})) = \int_{\mathcal{Y}} \mathbb{1}[\boldsymbol{y} \succcurlyeq \boldsymbol{v}_{\text{ref}}][1 - \prod_{\boldsymbol{y}_* \in \mathcal{P}(\mathcal{Y})} \mathbb{1}[\boldsymbol{y}_* \not\preceq \boldsymbol{y}]]\mathrm{d}\boldsymbol{y}, \tag{5}$$

where $\mathbb{1}(\cdot)$ is the indicator function, which outputs 1 if its argument is true and 0 otherwise.

$\boldsymbol{v}_{\text{ref}}$ is carefully chosen for convenience of calculation. Ideally, a feature vector $\boldsymbol{x}'$ that can increase the likelihood of DKL-GP maximally should be picked up from the design space $\mathcal{D}$ in every iteration. Thus, a better predictive Pareto-optimal set, enveloping the previous one by improving $\text{PVol}_{\boldsymbol{v}_{\text{ref}}}(\mathcal{P}(\mathcal{Y}))$ is the direct solution according to Equation (6) where $f : \boldsymbol{x} \to \boldsymbol{y} \in \mathcal{Y}$ is denoted as DKL-GP. Then the feature vector $\boldsymbol{x}_* = \arg\max_{\boldsymbol{x}' \in \mathcal{D}} \text{EIPV}(\boldsymbol{x}'|\mathcal{D})$ can be sampled as a new candidate for the predictive Pareto optimality.

Algorithm 3 BOOM Explorer$(\mathcal{D}, T, \mu, b)$

Require: $\mathcal{D}$ is the microarchitecture design space, T is the maximal iteration number, μ is a normalization coefficient and b is the number of samples to draw.

Ensure: Pareto-optimal set $\boldsymbol{X}$ that forms Pareto optimality among $\mathcal{D}$.

```
X_0 ← MicroAL(D, μ, b);                          ▷ Algorithm 2
Push X_0 to VLSI flow to obtan corresonding power and clock cycles Y;
L ← X_0;
U ← D \ L;
for i = 1 ← T do
    Establish and train DKL-GP on L with Y;
    x_* ← arg max_{x∈U} EIPV(x|U);               ▷ Equation (7)
    Push x_* to VLSI flow to obtain corresponding power and clock cycles and add to Y;
    L ← L ∪ x_*, U ← U \ x_*;
end for
Construct Pareto-optimal set X from L;
return Pareto-optimal set X;
```

$$\begin{aligned}\text{EIPV}(\boldsymbol{x}'|\mathcal{D}) = \mathbb{E}_{p(f(\boldsymbol{x}')|\mathcal{D})}[&\text{PVol}_{\boldsymbol{v}_{\text{ref}}}(\mathcal{P}(\mathcal{Y}) \cup f(\boldsymbol{x}')) \\ &- \text{PVol}_{\boldsymbol{v}_{\text{ref}}}(\mathcal{P}(\mathcal{Y}))].\end{aligned} \tag{6}$$

By decomposing the power-performance space as grid cells shown in Fig. 4, Equation (6) can be further simplified as Equation (7)

$$\text{EIPV}(\boldsymbol{x}'|\mathcal{D}) = \sum_{C \in \mathcal{C}_{\text{nd}}} \int_C \text{PVol}_{\boldsymbol{v}_C}(\boldsymbol{y}) p(\boldsymbol{y}, |\mathcal{D}) \mathrm{d}\boldsymbol{y}, \tag{7}$$

where $\mathcal{C}_{\text{nd}}$ denotes non-dominated cells. Region colored in green as referred to Fig. 4(b) shows the improvement of Pareto hypervolume. In Equation (7), $p(\boldsymbol{y}|\mathcal{D})$ is modeled as a multi-objective GP for power and clock cycles where the kernel function in Equation (4) parameterized by DNN can be Matérn 5/2 kernel.

Equipped with all aforementioned methodologies, Algorithm 3 provides the end-to-end flow of BOOM-Explorer. We first leverage Algorithm 2 to sample representative microarchitectures (*e.g.*, different branch prediction capability, cache organization, various structures of issue unit, *etc.*). DKL-GP is then built to characterize the design space. Finally, with Bayesian optimization, the Pareto-optimal set is explored via the maximization of EIPV.

IV. Experiments

We conduct comprehensive experiments to evaluate the proposed BOOM-Explorer. Chipyard framework [34] is leveraged to compile various BOOM RTL designs. We utilize 7-nm ASAP7 PDK [35] for the VLSI flow. Cadence Genus 18.12-e012_1 is used to synthesize every sampled RTL design, and Synopsys VCS M-2017.03 is used to simulate the design running at 2GHz with different benchmarks. PrimeTime PX R-2020.09-SP1 is finally used to get power value for all benchmarks.

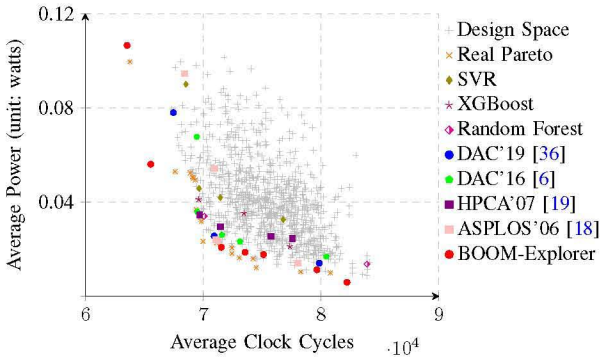

Fig. 5 Learned Pareto optimal set of BOOM microarchitectures.

A. Benchmarks and Baselines

Since it is time-consuming to verify every sampled microarchitecture design online, we construct an offline data set. Consisting of 994 legal microarchitectures, the offline data set is sampled randomly and uniformly from the BOOM design space as referred to in TABLE I. Each design is fed to the VLSI flow to get power and clock cycles with high fidelity for all benchmarks and the corresponding time to conduct the flow is also recorded. The VLSI flow for each design takes approximately from 6 hours to more than 14 hours to finish. All of experiments are conducted on this data set. Several benchmarks are selected to test the performance of microarchitectures, *i.e.*, `median`, `mt-vvadd`, `whetstone`, and `mm` from commonly used CPU benchmark suites. These four benchmarks are complete to all RISC-V instructions, *e.g.*, instructions that transfer data between registers and memory, floating-point manipulations, multi-threading executions, vector instructions, *etc.* The average clock cycles and power on the four benchmarks are denoted as the performance and power value for each design respectively.

Several representative baselines are compared with BOOM-Explorer. The ANN-based method [18] (shorted as ASPLOS'06), stacks ANN to predict the performance of designs, including a complicated chip multiprocessor. The regression-based method [19] (termed HPCA'07), leverages regression models with non-linear transformations to explore the power-performance Pareto curve on POWER4/POWER5 designs. The AdaBoost-RT-based method [6] (abbreviated as DAC'16), utilizes OA sampling and active learning-based AdaBoost regression tree models to explore microarchitectures *w.r.t.* their performance. The aforementioned arts are proved effective in their works of the exploration of microarchitectures respectively. Therefore, it is requisite to compare these methodologies with BOOM-Explorer. The HLS predictive model-based method [36] (named DAC'19), exploring the high-level synthesis design is also chosen as our baseline. Although the starting point is different, their method is proved to be robust and transferable. Moreover, we also compare BOOM-Explorer with traditional machine learning models, *i.e.*, support vector regression (SVR), random forest, and XGBoost [37]. For fair comparisons, experimental settings of the baselines are the same as those mentioned in their papers. Simulated annealing is leveraged for traditional machine learning algorithms, *e.g.*, SVR, Random Forest, and XGBoost.

TABLE III Normalized Experimental Results

Methodologies	Normalized ADRS	Normalized ORT [+]
SVR	0.2399	1.0000
Random Forest	0.2263	0.9763
XGBoost	0.2171	1.010
ASPLOS'06 [18]	0.1948	0.9437
HPCA'07 [19]	0.1907	0.8544
DAC'16 [6]	0.1473	3.0102
DAC'19 [36]	0.1884	0.8973
BOOM-Explorer w/o MicroAL	0.1441	**0.3307**
BOOM-Explorer	**0.1145**	0.3556

[+] ORT: Overall Running Time

B. Experiments Settings

In the settings of BOOM-Explorer, DKL-GP is stacked with three hidden layers, each of which has 1000, 500, and 50 hidden neurons respectively, and it adopts ReLU as the non-linear transformation for deep kernel learning. The Adam optimizer [38] is used, with an initial learning rate equals to 0.001. DKL-GP is initialized with 5 microarchitectures sampled according to MicroAL and then BOOM-Explorer performs Bayesian exploration with 9 rounds sequentially. All experiments together with baselines are repeated 10 times and we report corresponding average results.

Average distance to reference set (ADRS) and overall running time (ORT) are two metrics for performance comparisons. ADRS, as shown in Equation (8), is widely used in design space exploration problems to measure how close a learned Pareto-optimal set to the real Pareto-optimal set of the design space.

$$\mathrm{ADRS}(\Gamma, \Omega) = \frac{1}{|\Gamma|} \sum_{\gamma \in \Gamma} \min_{\omega \in \Omega} f(\gamma, \omega), \tag{8}$$

where f is the Euclidean distance function. Γ is the real Pareto-optimal set and Ω is the learned Pareto-optimal set. ORT measures the total time of algorithms including initialization and exploring.

C. Results Analysis

Fig. 5 shows the learned Pareto-optimal sets obtained by the baselines and BOOM-Explorer. The results show that the Pareto-optimal set learned by BOOM-Explorer is much closer to the real Pareto-optimal set and thus outperforming baselines remarkably.

The normalized results of ADRS and ORT are listed in TABLE III. BOOM-Explorer outperforms ASPLOS'06, HPCA'07, DAC'16, and DAC'19 by 70%, 66%, 29%, and 64% in ADRS, respectively. Meanwhile, it accelerates the exploring by more than 88% compared with DAC'16. Since the prior knowledge of BOOM microarchitecture designs is embedded in our method, DKL-GP can outperform baselines by a large margin in ADRS and ORT. The effectiveness of the proposed MicroAL is demonstrated by conducting comparative experiments of BOOM-Explorer with random sampling instead of MicroAL. The corresponding results are

TABLE IV Comparison with two-wide BOOM

Micro-architecture Design	Design Parameters +	Average Power (unit: watts)	Average Clock Cycles
Two-wide BOOM	[4, 16, 32, 12, 4, 8, 2, 2, 64, 80, 64, 1, 2, 1, 16, 16, 4, 2, 8]	6.0700×10^{-2}	74915.2963
Pareto Design *	[4, 16, 16, 8, 2, 8, 2, 2, 32, 64, 64, 1, 3, 1, 24, 24, 8, 4, 8]	5.8600×10^{-2}	73333.7407

+ The parameters are in the same order as TABLE I
* Pareto Design is found by BOOM-Explorer

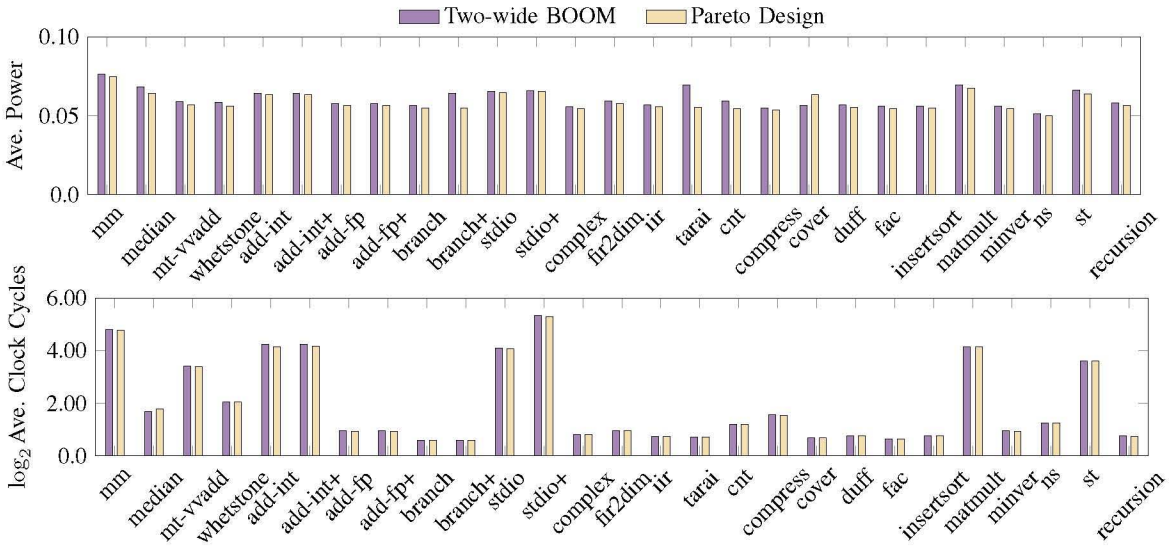

Fig. 6 Comparisons of power and performance between the Pareto design found by BOOM-Explorer and the two-wide BOOM.

listed as BOOM-Explorer w/o MicroAL. The results show that without MicroAL, the performance of BOOM-Explorer would be close to DAC'16.

If a larger initialization set is sampled via MicroAL, BOOM-Explorer will be able to gain a better predictive Pareto-optimal set. Finally, we can achieve different designs to strike good balances between power and performance.

D. The Optimal BOOM Microarchitecture Design

Our Pareto design is chosen from the Pareto-optimal set found by BOOM-Explorer and it is compared with a two-wide BOOM developed by senior engineers [1].

The aforementioned two microarchitectures of BOOM are listed in TABLE IV. Indicated by "Design Parameters" in TABLE IV, our Pareto design has the same DecodeWidth compared with the two-wide BOOM. However, the Pareto design reduces hardware components on the branch predictor (*i.e.*, RasEntry, BranchCount, *etc.*), entries of the reorder buffer, *etc.*, but enlarges instructions issue width, LDQ, STQ, *etc.* Moreover, it has different cache organizations, *e.g.*, different associate sets. Because `LSU` introduced in Section II-A tends to become a bottleneck, the Pareto design increases hardware resources for LDQ, STQ, and meanwhile increases associate sets and MSHR entries for D-Cache to overcome more data conflicts. Furthermore, the Pareto design reduces resources of RAS and BTB since there are not many branches or jump instructions in these benchmarks. Via reducing redundant hardware resources while increasing necessary components, our Pareto design achieves a better trade-off on power and performance.

To demonstrate the superiority of the Pareto design compared with the two-wide BOOM, both of them are evaluated on more benchmarks, and TABLE IV shows the average power and clock cycles of all these benchmarks. These benchmarks are chosen from different application scenarios, *e.g.*, add-int, add-fp, *etc.* are from ISA basic instructions, iir, firdim, *etc.* are from DSP-oriented algorithms [39], compress, duff, *etc.* are from real-time computing applications [40], *etc.* Fig. 6 shows the comparison of power and performance between them. For all of these benchmarks, our Pareto design runs approximately 2.11% faster and at the same time dissipates 3.45% less power than the two-wide BOOM.

V. Conclusions

In this paper, BOOM-Explorer is proposed to search for Pareto optimality among the microarchitecture design space within a short time. To the best of our knowledge, this is the first work introducing automatic design space exploration solution to the RISC-V community. We expect to see a lot of researches in our community to further improve microarchitecture design space explorations of processors.

Acknowledgment

This work is partially supported by HiSilicon and The Research Grants Council of Hong Kong SAR CUHK14209420, CUHK14208021.

REFERENCES

[1] K. Asanovic, D. A. Patterson, and C. Celio, "The berkeley out-of-order machine (BOOM): An industry-competitive, synthesizable, parameterized RISC-V processor," University of California at Berkeley, Tech. Rep., 2015.

[2] C. P. Celio, *A Highly Productive Implementation of an Out-of-Order Processor Generator*. eScholarship, University of California, 2017.

[3] J. Bachrach, H. Vo, B. Richards, Y. Lee, A. Waterman, R. Avižienis, J. Wawrzynek, and K. Asanović, "Chisel: constructing hardware in a scala embedded language," in *ACM/IEEE Design Automation Conference (DAC)*, 2012, pp. 1212–1221.

[4] S. Salamin, M. Rapp, A. Pathania, A. Maity, J. Henkel, T. Mitra, and H. Amrouch, "Power-efficient heterogeneous many-core design with ncfet technology," *IEEE Transactions on Computers*, vol. 70, no. 9, pp. 1484–1497, 2021.

[5] B. Grayson, J. Rupley, G. Z. Zuraski, E. Quinnell, D. A. Jiménez, T. Nakra, P. Kitchin, R. Hensley, E. Brekelbaum, V. Sinha *et al.*, "Evolution of the samsung exynos CPU microarchitecture," in *IEEE/ACM International Symposium on Computer Architecture (ISCA)*, 2020, pp. 40–51.

[6] D. Li, S. Yao, Y.-H. Liu, S. Wang, and X.-H. Sun, "Efficient design space exploration via statistical sampling and adaboost learning," in *ACM/IEEE Design Automation Conference (DAC)*, 2016, pp. 1–6.

[7] M. Moudgill, P. Bose, and J. H. Moreno, "Validation of turandot, a fast processor model for microarchitecture exploration," in *International Performance Computing and Communications Conference (IPCCC)*, 1999, pp. 451–457.

[8] T. Austin, E. Larson, and D. Ernst, "SimpleScalar: An infrastructure for computer system modeling," *Computer*, vol. 35, no. 2, pp. 59–67, 2002.

[9] D. Brooks, P. Bose, V. Srinivasan, M. K. Gschwind, P. G. Emma, and M. G. Rosenfield, "New methodology for early-stage, microarchitecture-level power-performance analysis of microprocessors," *IBM Journal of Research and Development*, vol. 47, no. 5.6, pp. 653–670, 2003.

[10] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, p. 1–7, Aug. 2011. [Online]. Available: https://doi.org/10.1145/2024716.2024718

[11] E. Perelman, G. Hamerly, M. Van Biesbrouck, T. Sherwood, and B. Calder, "Using simpoint for accurate and efficient simulation," *ACM SIGMETRICS Performance Evaluation Review*, vol. 31, no. 1, pp. 318–319, 2003.

[12] Y.-I. Kim and C.-M. Kyung, "Automatic translation of behavioral testbench for fully accelerated simulation," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2004, pp. 218–221.

[13] S. Beamer and D. Donofrio, "Efficiently exploiting low activity factors to accelerate RTL simulation," in *ACM/IEEE Design Automation Conference (DAC)*, 2020, pp. 1–6.

[14] J. Feldmann, K. Kraft, L. Steiner, N. Wehn, and M. Jung, "Fast and accurate DRAM simulation: Can we further accelerate it?" in *IEEE/ACM Proceedings Design, Automation and Test in Eurpoe (DATE)*, 2020, pp. 364–369.

[15] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2009, pp. 469–480.

[16] K. Yu, J. Bi, and V. Tresp, "Active Learning via Transductive Experimental Design," in *International Conference on Machine Learning (ICML)*, 2006, pp. 1081–1088.

[17] Q. Sun, T. Chen, S. Liu, J. Miao, J. Chen, H. Yu, and B. Yu, "Correlated multi-objective multi-fidelity optimization for hls directives design," in *IEEE/ACM Proceedings Design, Automation and Test in Eurpoe (DATE)*, 2021.

[18] E. Ïpek, S. A. McKee, R. Caruana, B. R. de Supinski, and M. Schulz, "Efficiently exploring architectural design spaces via predictive modeling," *ACM SIGOPS Operating Systems Review*, vol. 40, no. 5, pp. 195–206, 2006.

[19] B. C. Lee and D. M. Brooks, "Illustrative design space studies with microarchitectural regression models," in *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2007, pp. 340–351.

[20] D. Wu, C.-T. Lin, and J. Huang, "Active learning for regression using greedy sampling," *Information Sciences*, vol. 474, pp. 90–105, 2019.

[21] D. Li, S. Yao, S. Wang, and Y. Wang, "Cross-program design space exploration by ensemble transfer learning," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2017, pp. 201–208.

[22] H.-Y. Liu and L. P. Carloni, "On learning-based methods for design-space exploration with high-level synthesis," in *ACM/IEEE Design Automation Conference (DAC)*, 2013, pp. 1–7.

[23] S. Liu, F. C. Lau, and B. C. Schafer, "Accelerating FPGA prototyping through predictive model-based HLS design space exploration," in *ACM/IEEE Design Automation Conference (DAC)*, 2019, pp. 1–6.

[24] Q. Sun, C. Bai, H. Geng, and B. Yu, "Deep Neural Network Hardware Deployment Optimization via Advanced Active Learning," in *IEEE/ACM Proceedings Design, Automation and Test in Eurpoe (DATE)*, 2019.

[25] Y. Ma, S. Roy, J. Miao, J. Chen, and B. Yu, "Cross-layer optimization for high speed adders: A pareto driven machine learning approach," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 38, no. 12, pp. 2298–2311, 2018.

[26] Y. Ma, Z. Yu, and B. Yu, "CAD Tool Design Space Exploration via Bayesian Optimization," in *ACM/IEEE Workshop on Machine Learning CAD (MLCAD)*, 2019, pp. 1–6.

[27] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[28] Q. Sun, A. A. Rao, X. Yao, B. Yu, and S. Hu, "Counteracting adversarial attacks in autonomous driving," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2020, pp. 1–7.

[29] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," *arXiv preprint arXiv:2103.14030*, 2021.

[30] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing, "Deep kernel learning," in *Artificial intelligence and statistics*. PMLR, 2016, pp. 370–378.

[31] W. Lyu, F. Yang, C. Yan, D. Zhou, and X. Zeng, "Batch bayesian optimization via multi-objective acquisition ensemble for automated analog circuit design," in *International Conference on Machine Learning (ICML)*. PMLR, 2018, pp. 3306–3314.

[32] A. Shah and Z. Ghahramani, "Pareto frontier learning with expensive correlated objectives," in *International Conference on Machine Learning (ICML)*, 2016, pp. 1919–1927.

[33] S. Daulton, M. Balandat, and E. Bakshy, "Differentiable expected hypervolume improvement for parallel multi-objective bayesian optimization," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[34] A. Amid, D. Biancolin, A. Gonzalez, D. Grubb, S. Karandikar, H. Liew, A. Magyar, H. Mao, A. Ou, N. Pemberton, P. Rigge, C. Schmidt, J. Wright, J. Zhao, Y. S. Shao, K. Asanović, and B. Nikolić, "Chipyard: Integrated design, simulation, and implementation framework for custom socs," *IEEE Micro*, vol. 40, no. 4, pp. 10–21, 2020.

[35] V. Vashishtha, M. Vangala, and L. T. Clark, "Asap7 predictive design kit development and cell design technology co-optimization," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2017, pp. 992–998.

[36] S. Liu, F. C. Lau, and B. C. Schafer, "Accelerating FPGA Prototyping through Predictive Model-Based HLS Design Space Exploration," in *ACM/IEEE Design Automation Conference (DAC)*, 2019, pp. 1–6.

[37] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, 2016, pp. 785–794.

[38] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[39] V. Zivojnovic, J. Martinez, C. Schläger, and H. Meyr, "DSPstone: A DSP-Oriented Benchmarking Methodology," in *Proc. ICSPAT*, 1994.

[40] M. KUZHAN and V. H. ŞAHİN, "MBBench: A WCET benchmark suite," *Sakarya University Journal of Computer and Information Sciences*, vol. 3, no. 1, pp. 40–50, 2020.