Parallel Multi-objective Bayesian Optimization Framework for CGRA Microarchitecture

Bing Li, Wendi Sun, Xiaobing Ni, Kaixuan He, Qi Xu*, Song Chen*, Yi Kang
University of Science and Technology of China

***Abstract*—Recently, due to the flexibility and reconfigurability of Coarse-Grained Reconfigurable Architecture (CGRA), CGRA microarchitecture has become an inevitable trend to accelerate the convolution calculation in diverse deep neural networks. However, since the vast microarchitecture design space and the complicated VLSI verification flow, it is a huge challenge to explore a perfect microarchitecture to compromise between multiple performance metrics. In this paper, we formulate the CGRA microarchitecture design as a design space exploration problem, and propose a parallel multi-objective Bayesian optimization framework (PAMBOF) to automatically explore the CGRA microarchitecture design space. Meanwhile, high-precision performance and area models are built to enable fast design space exploration. To approximate the black-box objective function in the design space, the PAMBOF framework first builds multiple Gaussian processes (GP) with deep regularization kernel learning functions (DRKL-GP). Then a parallel Bayesian optimization algorithm is developed to sample a batch of candidate design points, which are simulated in parallel by the performance and area models. Experimental results demonstrate that compared to the prior arts, the proposed PAMBOF framework can search for a CGRA microarchitecture design with the better area and performance in a shorter runtime.**

I. Introduction

With the growing demand for artificial intelligence applications, there are enormous requirements for computing, storage, and data exchange. As a result, massive data exchange and computation bring tremendous performance and energy challenges for traditional processors. Recently, due to the computational flexibility and reconfigurability, Coarse-Grained Reconfigurable Architecture (CGRA) array has become an inevitable trend to accelerate the calculation task to reduce the computing pressure of the processor.

The CGRA microarchitecture consists of multiple components, each with a diverse organization and allowing reconfiguration at a granularity above gate-level. Besides, different CGRA microarchitectures under a specific technology process demonstrate various areas, performance, power, etc. As a result, finding a microarchitecture design that balances the area and performance is very difficult. First, the entire design space is vast, expanding exponentially as more components are considered. On the other hand, for each CGRA microarchitecture design with a specific benchmark, we need to use the commercia EDA tool to simulate it and obtain metrics such as area and time, requiring plenty of runtime.

In general, design space exploration (DSE) can help customize the trade-off between performance, area, and other metrics in design. However, the large design space associated with CGRA makes DSE very time-consuming and expensive [1]. Several optimization algorithms have been developed to search for designs, including simulated annealing-based heuristic [2], particle swarm optimization (PSO) algorithm [3], and evolutionary algorithm [4]. Although the number of hardware simulations is reduced, most of these algorithms are easily trapped into local optima and have a relatively low coverage rate. In order to quickly obtain an optimal trade-off design, we propose a parallel multi-objective Bayesian optimization framework (PAMBOF) to explore the CGRA microarchitecture design. For each iteration, the PAMBOF requires metrics of the performance and area of the current configuration to make informed choices. For a specific task, performance and area estimations require a complete VLSI flow provided by the commercial EDA tool. Although these estimations are accurate, obtaining these metrics takes dozens of hours, significantly slowing down our exploration process when many iterations must be performed. Therefore, to enhance exploration efficiency, we construct high-precision performance and area models, as discussed in Section III-A and Section III-B. Although the estimation values may not be as accurate as metrics obtained from the commercial EDA tool, they are sufficiently precise for design space exploration. Key technical contributions of this work are listed as follows.

- We propose a parallel multi-objective Bayesian optimization framework (PAMBOF) to explore a CGRA microarchitecture design with the trade-off between area and performance, which contains a correlated multi-objective Gaussian process (GP) model and a parallel GP-based Bayesian optimization algorithm.
- A new high-accuracy area and performance models are built to evaluate the area and performance for CGRAs, enabling fast design space exploration.
- We verify the performance of the CGRA microarchitecture design produced by PAMBOF flow under $28nm$ technology. The results demonstrate the outstanding performance of the CGRA microarchitecture.

The rest of the paper is organized as follows. Section II introduces some knowledge about our CGRA microarchitecture. Section III describes the built area and performance models,

This work is partially supported by the Strategic Priority Research Program of Chinese Academy of Sciences under grant No. XDB44000000, the National Natural Science Foundation of China (NSFC) under grant No. 62141415, 61931008, the CAS Project for Young Scientists in Basic Research under grant No. YSBR-029k, and the National Key R&D Program of China under grant No. 2019YFB2204800.

* The corresponding authors.

979-8-3503-4859-0/DATE24/© 2024 EDAA

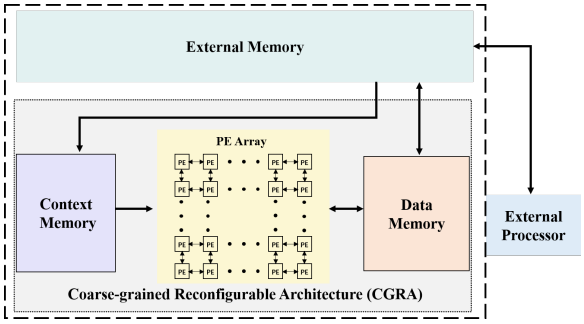

Fig. 1 The overview of the CGRA microarchitecture.

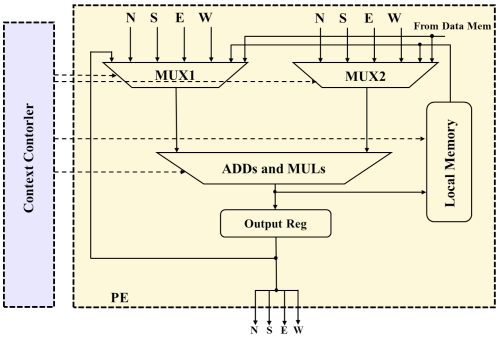

Fig. 2 The overview of the PE architecture.

and then gives the problem definition. Section IV presents the details of the proposed PAMBOF framework. Section V conducts several experiments to validate our methods, followed by conclusions in Section VI.

II. CGRA Microarchitecture

As described above, due to the flexibility and reconfigurability of the CGRA, the CGRA microarchitecture can adapt to different application scenarios. Fig. 1 illustrates an overview of a CGRA module. Generally, a CGRA mainly contains several components, such as a 2D processing element (PE) array, a context memory for configuration and calculation instructions, a data memory for data computation, and an external memory for storing all data required for calculation [5]. The internal structure of the PE in CGRA microarchitecture is depicted in Fig. 2. Each PE consists of several adders (ADDs), several multipliers (MULs), and a local memory for temporary data storage. Additionally, each PE can exchange data with its neighboring PEs located in North (N), South (S), East (E), and West (W). The CGRA primarily utilizes the PE array to execute calculation tasks. After completing the calculation, the results will be stored in the external memory. Therefore, the CGRA has a substantial impact on the performance of specific calculation tasks.

III. Problem Formulation

A. Area Model

The area model is based on $28nm$ technology. Based on the CGRA architectural description, the area model only needs to know the number of PEs and necessary information about components within the PE, such as the depth of internal memory, the type of ADDs, and the type of MUL, to determine the area size. Besides, tiny elements in the microarchitecture design are ignored, i.e., loaders and communication interfaces. Meanwhile, the internal computing resources of CGRA support 8-bit integer (INT8) operations, and the interconnection topology follows a mesh configuration. Based on the information, the entire microarchitecture area is easily calculated as follows.

$$A_{cgra} = \sum_{\forall type} f_{area}(type) \cdot N_{type}, \tag{1}$$

where N_{type} refers to the number of each type ($type$) for each component. The function $f_{area}(\cdot)$ queries the area model database for the area of each component type. Through the calculation model, the area of the CGRA microarchitecture under different hardware configurations can be quickly obtained. The experimental results demonstrate that the area evaluated by the model is comparable to the results performed by the commercial EDA design compiler (DC).

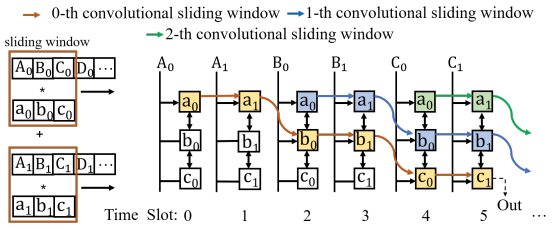

Fig. 3 The overview of row convolutional operations in PE.

B. Performance Model

For a compute-intensive application, we utilize data flow scheduling algorithms to map the data flow graph (DFG) abstracted from the kernel loop of the application to CGRA. In this work, the calculation process is based on the specific scheduling approach and operator design described in [6]. By employing the modulo scheduling method on the operators, adjacent loop iterations start at fixed intervals, thus realizing loop-level parallelism between adjacent iterations. Furthermore, based on the scheduling results, we can determine the computational tasks allocated to each PE during each cycle. Within each computation, the convolution operation is performed for the input vector and the weight vector of each row of the convolutional kernel through a sliding window to obtain the partial sum result, which is then accumulated to derive the final output feature map. Fig. 3 depicts the process of calculating row convolution in the PE through window sliding, where A_i, B_i, and C_i are convolution inputs and a_i, b_i, and c_i are weights of the convolution kernel. As a result, the total calculation time of the task can be easily obtained.

Compared to the accurate processes obtained through RTL simulations using the commercial EDA tool VCS, the evaluation of clock cycles required for a specific task by the built performance model is acceptable.

TABLE I Design space of CGRA microarchitecture.

Parameters	Candidate Values
Number of rows in PE array	4,5,. . .,17,18
Number of columns in PE array	4,5,. . .,17,18
Number of ADDs in each PE	1,2,4,6
Number of MULs in each PE	1,2,3,4
Depth of context memory for configuration instructions	20,24,28,32
Depth of context memory for calculation instructions	20,24,28,32
Depth of data memory	32,48,64,128
Depth of local memory in each PE	1,2,3,4

C. Problem Definition

Definition 1 (CGRA Microarchitecture Design)**.** *A combination of feasible values listed in TABLE I represents a CGRA microarchitecture design. Each design among the entire design space $\mathcal{D}$ is encoded as a feature vector $\boldsymbol{x}$. For convenience, microarchitecture design and microarchitecture design point are the same in the following sections.*

Definition 2 (Area)**.** *The area of the CGRA microarchitecture design is calculated by the proposed area model.*

Definition 3 (Clock Cycle)**.** *The clock cycle is the time the CGRA microarchitecture consumes to execute a calculation task, which is obtained by the built performance model.*

Definition 4 (Pareto-optimal Design)**.** *For a feature vector $\boldsymbol{x}$ in m-dimension minimization problem, an objective vector $f(\boldsymbol{x})$ dominates $f(\boldsymbol{x'})$ if*

$$\begin{aligned} &\forall i \in [1,m], \quad f_i(\boldsymbol{x}) \leq f_i(\boldsymbol{x'}) \text{ and} \\ &\exists j \in [1,m], \quad f_j(\boldsymbol{x}) < f_j(\boldsymbol{x'}). \end{aligned} \tag{2}$$

A point $\boldsymbol{x}$ is Pareto-optimal if there is no other $\boldsymbol{x'}$ in entire design space satisfying that $f(\boldsymbol{x'})$ dominates $f(\boldsymbol{x})$. The set of points that are not dominated by others is called the Pareto-optimal set. For the microarchitecture design space exploration, the Pareto-optimal set is also described as the Pareto-optimal design.

In this work, our objective is to search for the Pareto-optimal design as defined in Definition 4 in consideration of area and execution time (clock cycle). Since these two metrics are negatively correlated, for a Pareto-optimal design, an alternative choice does not exist that can improve each objective without sacrificing others. With the evaluation metrics above, our problem is formulated as follows.

Problem 1 (CGRA Microarchitecture Design Space Exploration)**.** *Given the CGRA design space $\mathcal{D}$, each design is considered as a feature vector $\boldsymbol{x} \in \mathbb{R}^d$. The CGRA microarchitecture design space exploration aims to search for Pareto-optimal designs with a trade-off between area and clock cycles over a vast design space.*

IV. CGRA Design Optimization

A. Overview of PAMBOF

The overall flow of the proposed PAMBOF is shown in Fig. 4. Firstly, given the input microarchitecture design space $\mathcal{D}$, we sample several points from $\mathcal{D}$ by the K-means algorithm [7], and obtain corresponding metrics using the performance and area models. The sampled points should maximize the intra-cluster diversity to get more information from design space. Then, a Gaussian process model with deep regularization kernel learning [8] function (DRKL-GP) is built on the initial dataset. Meanwhile, a parallel multi-objective Bayesian optimization algorithm is performed to explore the Pareto-optimal design. We introduce the parallel expected q-hypervolume improvement (qEHVI) as the acquisition function [9]. Concrete forms of the DRKL-GP model and qEHVI function are described in Sections IV-B and IV-C. At each optimization iteration, we will select new design points from $\mathcal{D}$, maximizing the qEHVI. Afterwards, these points are simulated by the area and performance models to obtain the corresponding reports under a specific task. If sampled designs cannot meet the expectations, we augment the dataset with newly selected designs and retrain our Gaussian model again for the next exploration; otherwise, the exploration process ends, yielding Pareto-optimal designs.

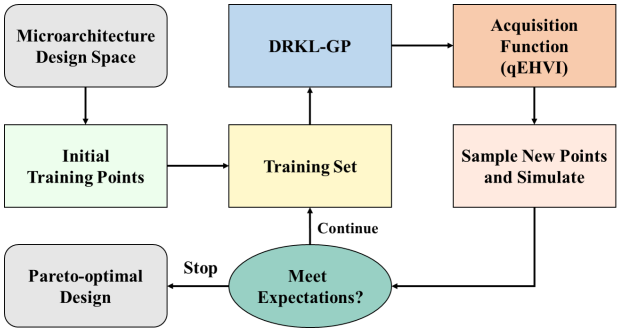

Fig. 4 The illustration of the proposed PAMBOF flow.

B. Gaussian Process

Owing to the robustness and effective prediction of the whole design space with a limited dataset, the GP model has been widely applied for design space exploration problem [10]. In a GP model, for any finite data $\boldsymbol{X} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n\}$, $\boldsymbol{x}_i \in \mathbb{R}^d$, the objective vector $\boldsymbol{y} = (f(\boldsymbol{x}_1), \cdots, f(\boldsymbol{x}_n))^{\mathrm{T}}$ follows a joint multivariate Gaussian distribution according to Equation (3).

$$\left[f(\boldsymbol{x_1}) \cdots f(\boldsymbol{x_n})\right]^{\mathrm{T}} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{K}), \tag{3}$$

where f: $\mathbb{R}^d \to \mathbb{R}$ is a black-box objective function to be inferred. $\boldsymbol{\mu} \in \mathbb{R}^n$ is the mean vector value, and typically the constant mean function is widely used, i.e., $\boldsymbol{\mu} = \mu \cdot \boldsymbol{1}$. $\boldsymbol{K} \in \mathbb{R}^{n \times n}$ is the covariance matrix measured by a kernel function. In this work, we adopt a Matérn 5/2 kernel function to calculate the similarity of any two input points as follows.

$$\begin{aligned} k(\boldsymbol{x_i}, \boldsymbol{x_j}) &= \sigma^2(1 + \sqrt{5}r + \frac{5}{3}r^2)e^{-\sqrt{5}r}, \\ r &= \sum_{k=1}^{d} \frac{(x_{i,k} - x_{j,k})^2}{\lambda_k^2}, \end{aligned} \tag{4}$$

where λ_k and σ are the hyperparameters of the GP model.

For a newly sampled design point $\boldsymbol{x}_*$ and its corresponding objective function $y_* = f(\boldsymbol{x_*})$, the joint Gaussian distribution between y_* and the dataset $\boldsymbol{y}$ is calculated as follows:

$$\begin{bmatrix} \boldsymbol{y} \\ y_* \end{bmatrix} \sim \mathcal{N}(\begin{bmatrix} \boldsymbol{\mu} \\ \mu \end{bmatrix}, \begin{bmatrix} K(\boldsymbol{X}) & k(\boldsymbol{X}, \boldsymbol{x_*}) \\ k(\boldsymbol{X}, \boldsymbol{x_*})^{\mathrm{T}} & k(\boldsymbol{x_*}, \boldsymbol{x_*}) \end{bmatrix}), \tag{5}$$

where $K(\boldsymbol{X})$ is the intra-covariance matrix among microarchitecture design data in $\boldsymbol{X}$, while $k(X, \boldsymbol{x_*})$ is a vector of covariance between x_* and all designs in $\boldsymbol{X}$. Obviously, y_* conditioned by $\boldsymbol{y}$ also follows the normal distribution as $y_* \sim \mathcal{N}(\mu_{y_*|\boldsymbol{y}}, \sigma^2_{y_*|\boldsymbol{y}})$.

To increase the uncertainty of area and clock cycles generated by different CGRA microarchitectures, an uncorrelated Gaussian noise $\epsilon_n \sim \mathcal{N}(0, \sigma_n^2)$ is added to the objective function. The kernel function is then written as $k(\boldsymbol{x_i}, \boldsymbol{x_j}) + \sigma_n^2 \delta_{ij}$, where δ_{ij} indicates the Kronecker delta function, and the σ_n refers to a hyperparameter. Thus, the GP model is updated as follows.

$$\begin{aligned} \mu_{y_*|\boldsymbol{y}} &= \mu + k(\boldsymbol{X}, \boldsymbol{x_*})^{\mathrm{T}} (K(\boldsymbol{X}) + \sigma_n^2 \boldsymbol{I})^{-1} (\boldsymbol{y} - \boldsymbol{\mu}), \\ \sigma^2_{y_*|\boldsymbol{y}} &= k(\boldsymbol{x_*}, \boldsymbol{x_*}) - k(\boldsymbol{X}, \boldsymbol{x_*})^{\mathrm{T}} (K(\boldsymbol{X}) + \sigma_n^2 \boldsymbol{I})^{-1} k(\boldsymbol{X}, \boldsymbol{x_*}). \end{aligned} \tag{6}$$

For simplicity, all hyper-parameters in the GP model are parameterized as $\boldsymbol{\theta}$. Given observed data, the maximum likelihood estimation (MLE) is a common approach for determining $\boldsymbol{\theta}$. However, the MLE method is computationally expensive and cannot ensure learning quality. Recently, deep neural networks (DNNs) have demonstrated great potential in various applications as the black-box model for extracting valuable features [11]. Thus, a Gaussian process model with deep regularization kernel learning function (DRKL-GP) is built to characterize the design space. It first maps the input $\boldsymbol{x}_i$ to intermediate values through a neural network $\varphi(\cdot)$ parameterized by weights and biases $\boldsymbol{w}$. Then these intermediate values are used as inputs to the standard kernel function, which is updated as $k(\boldsymbol{x_i}, \boldsymbol{x_j}) \rightarrow k(\varphi(\boldsymbol{x}_i, \boldsymbol{w}), \varphi(\boldsymbol{x}_j, \boldsymbol{w})))$.

By optimizing the network's weights $\boldsymbol{w}$ instead of adjusting parameters $\boldsymbol{\theta}$ in the GP model, the performance is improved.

C. Parallel Multi-objective Bayesian Optimization

Bayesian optimization is a sequential strategy for optimizing expensive black-box functions, with no explicit form assumptions. In this work, DRKL-GP model defined in Section IV-B is provided as the surrogate model. The acquisition function employs the surrogate model to assign a utility value to a set of candidates to be evaluated, including probability of improvement (PI) [12], expected improvement (EI) [13], and upper confidence bound (UCB) [14]. However, these acquisition functions lead to sub-optimal results. Due to the perfect performance for a trade-off between different metrics, the expected q-hypervolume improvement (qEHVI) [9] is introduced as the acquisition function for our problem. Firstly, we clarify the concept of hypervolume (HV), which is an indicator to evaluate the utility of the candidate points in the design space. For a Pareto-optimal set $\mathcal{P}$ and a reference point

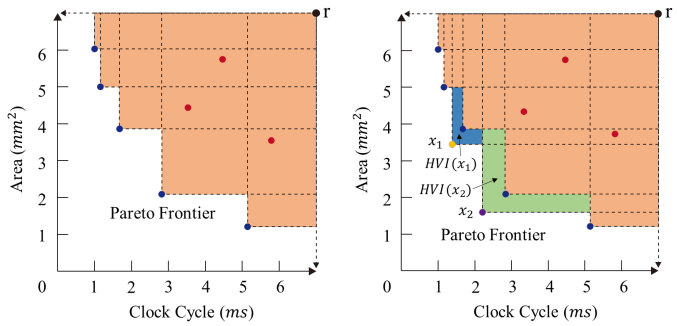

Fig. 5 An example of minimizing area and clock cycles. (a) Blue points are Pareto-optimal points and red points are dominated. Orange cells are dominated while blank regions are not. The volume of orange cells is the current Pareto hypervolume (HV). (b) Purple and yellow points are predicted to be Pareto-optimal designs, and green and blue regions are the corresponding qEHVI.

$\boldsymbol{r} = (r_1, r_2)$ (area and clock cycle metrics), HV measures the size of the region enclosed by $\mathcal{P}$ and $\boldsymbol{r}$ as.

$$\mathrm{HV}(\mathcal{P}, \boldsymbol{r}) = l_m(\bigcup_{\boldsymbol{p} \in \mathcal{P}} [\boldsymbol{p}, \boldsymbol{r}]), \tag{7}$$

where l_m is the Lebesgue measure, and $[\boldsymbol{p}, \boldsymbol{r}]$ refers to the hyper-rectangle bounded by vertices $\boldsymbol{p}$ and $\boldsymbol{r}$. As shown in Fig. 5, the orange-shaded region denotes the current Pareto HV. Based on the Pareto HV information, we further compute the hypervolume improvement (HVI) to evaluate new samples. Given a new candidate point $\boldsymbol{x_c}$ and the corresponding metric vector $\boldsymbol{y_c}$, HVI is calculated as follows.

$$\mathrm{HVI}(\mathcal{P}, \boldsymbol{r}, \boldsymbol{y_c}) = \mathrm{HV}(\mathcal{P} \cup \boldsymbol{y_c}, \boldsymbol{r}) - \mathrm{HV}(\mathcal{P}, \boldsymbol{r}). \tag{8}$$

The surrogate model in Bayesian optimization provides a posterior distribution $p(\boldsymbol{f}(\boldsymbol{x})|\mathcal{D})$ over the objective values for each design point $\boldsymbol{x}$, which can be adopted to compute the EHVI acquisition function.

$$\mathrm{EHVI}(\mathcal{P}, \boldsymbol{r}, \boldsymbol{y_c}) = \mathbb{E}_{p(\boldsymbol{f}(\boldsymbol{x_c})|\mathcal{D})}[\mathrm{HVI}(\mathcal{P}, \boldsymbol{r}, \boldsymbol{y_c})], \tag{9}$$

where f: $\boldsymbol{x} \rightarrow \boldsymbol{y}$ is inferred by DRKL-GP. Assuming the different objectives are independent and modeled with separate GP models, EHVI can be expressed in closed form [15]. In each iteration, the design $\boldsymbol{x}_* = \underset{\boldsymbol{x} \in \mathcal{D}}{\mathrm{argmax}}\ \mathrm{EHVI}(\mathcal{P}, \boldsymbol{r}, \boldsymbol{f}(\boldsymbol{x}))$ is selected as a candidate Pareto point.

In order to query a batch of q points in parallel, $\mathcal{X}_c = \{\boldsymbol{x_1}, \cdots, \boldsymbol{x_q}\}$, we extend the EHVI function in Equation (9) to qEHVI. Using the Monte-Carlo (MC) simulation with samples from the joint posterior distribution $\{\boldsymbol{f}_t(\boldsymbol{x}_i)\}_{i=1}^q \sim p(\boldsymbol{f}(x_1), \cdots, \boldsymbol{f}(x_q)|\mathcal{D}), t = 1, \cdots, M$, the qEHVI function is formulated as Equation (10).

$$q\mathrm{EHVI}(\mathcal{X}_c) = \frac{1}{M} \sum_{t=1}^{M} \mathrm{HVI}(\boldsymbol{f}_t(\boldsymbol{x_c})), \tag{10}$$

where M denotes the number of MC samples.

Algorithm 1 PAMBOF for CGRA Microarchitecture Design Space Exploration

Input: Microarchitecture design space $\mathcal{D}$, initial Pareto design points $\mathcal{X}$ from $\mathcal{D}$, iteration number t.
Output: Pareto-optimal design set $\mathcal{O}$.

1: Push $\mathcal{X}$ into area and performance models to get area and clock cycle $\mathcal{Y}$;
2: $\mathcal{P} \leftarrow \mathcal{X}$, $\mathcal{U} \leftarrow \mathcal{D} \setminus \mathcal{X}$;
3: **for** $i \leftarrow 1$ to t **do**
4: Establish and train DRKL-GP on $\mathcal{P}$ with $\mathcal{Y}$;
5: Query new Pareto designs $\mathcal{X}_c$ from $\mathcal{U}$;▷ Equation 10
6: Put $\mathcal{X}_c$ into area and performance models to achieve corresponding metrics $\mathcal{Y}_c$;
7: $\mathcal{P} \leftarrow \mathcal{P} \cup \mathcal{X}_c$, $\mathcal{U} \leftarrow \mathcal{U} \setminus \mathcal{X}_c$, $\mathcal{Y} \leftarrow \mathcal{Y} \cup \mathcal{Y}_c$;
8: **end for**
9: Achieve Pareto-optimal design set $\mathcal{O}$ from $\mathcal{P}$.

Algorithm 1 summarizes the details of the proposed PAMBOF flow for the CGRA microarchitecture design space exploration. The initial Pareto designs $\mathcal{X}$ are sampled from $\mathcal{D}$ by the K-means algorithm. At each iteration, the DRKL-GP model is firstly built on the current Pareto-optimal design set $\mathcal{P}$ with corresponding metric set $\mathcal{Y}$ (line 4). Then, new Pareto design points $\mathcal{X}_c$ are sampled from unobserved design space $\mathcal{U}$ by maximizing the qEHVI function (line 5). Based on the new set $\mathcal{X}_c$, the accurate metrics $\mathcal{Y}_c$ are simulated through the built area and performance models (line 6). Next, we augment the Pareto-optimal design set $\mathcal{P}$ with $\mathcal{X}_c$, and update metrics set $\mathcal{Y}$ and unsampled design space $\mathcal{U}$ (line 7). Finally, the Pareto-optimal designs are obtained (line 9).

V. Experimental Results

A. Experiment Settings

The proposed PAMBOF framework is implemented in `Python` with `Pytorch` and `Botorch` [16] libraries, and validated on a Linux server with an Intel Xeon Gold 6254 CPU and an Nvidia Tesla V100 GPU. The neural network in the DRKL-GP model consists of three hidden layers, each with 250, 100, and 50 neurons, respectively. A ReLU layer is inserted after the hidden layers as the activation. The adaptive moment estimation (Adam) optimizer is adopted to train the models. In order to evaluate the actual performance of Pareto-optimal CGRA design, we use Synopsys VCS S-2021.09-SP1 to simulate the clock cycle of the CGRA for a specific calculation at 400 MHz, while Synopsys Design Compiler P-2019.03-SP4 is utilized for design synthesis to obtain the area value. In addition, the RTL simulation results in turn validate the prediction accuracy of built area and performance models.

We randomly sample 968 microarchitecture designs from the entire design space. Then, each design description is fed into the area and performance models to achieve the corresponding area and clock cycle metrics for a specific convolution benchmark, which takes approximately one hour. In addition, DRKL-GP is initialized with 12 CGRA samples produced by K-means algorithm from the design space. Based on these sampled data, we perform PAMBOF framework with 12 iterations. Note that two new Pareto design points are chosen in each iteration, and the corresponding metrics are simulated in parallel by using area and performance models.

TABLE II Comparison results.

Approach	Normalized ADRS	Normalized ORT
Random Forest	0.481	0.746
MLP	0.389	1.180
Decision Tree	0.293	0.930
XGBoost	0.206	0.684
BOOM-Explorer [17]	0.165	0.131
PAMBOF	**0.115**	**0.119**

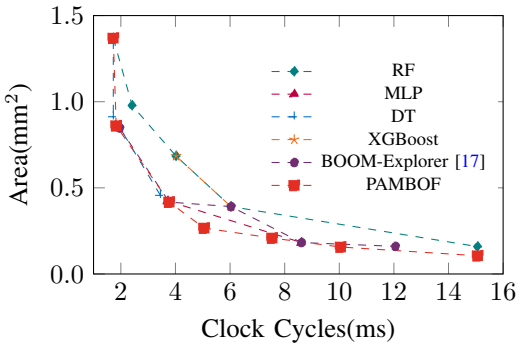

Fig. 6 Learned Pareto Frontier by different explorers.

B. Results and Analysis

To verify the superiority of the proposed framework, we compare our PAMBOF with BOOM-Explorer [17]. Besides, several machine learning algorithms are also executed to find Pareto-optimal design, including Random Forest (RF), Multi-layer Perceptron (MLP), Decision Tree (DT), and XGBoost. The initial dataset and the simulation environment are the same as those set in PAMBOF.

Two indicators are defined to evaluate the performance of different algorithms: the average distance to reference set (ADRS) and the overall running time (ORT). ADRS calculates the average distance between the true Pareto-optimal set and the learned Pareto set, as defined in Equation (11).

$$\mathrm{ADRS}(\Upsilon, \Omega) = \frac{1}{|\Upsilon|} \sum_{\tau \in \Upsilon} \min_{\omega \in \Omega} d(\tau, \omega), \tag{11}$$

where Υ and Ω refer to the true and learned Pareto-optimal sets, respectively. $d(\cdot)$ is the Euclidean distance function. Since the magnitude of values for different components varies greatly as listed in TABLE I, the parameter values are normalized in the calculation of ADRS. To ensure a fair comparison, we also normalize the ORT indicator, which represents the ratio of total runtime for model initialization and iterative explorations to the total iterations. Thus, the normalized ORT considers the discrepancy in the number of iterations among different algorithms.

TABLE II lists the normalized results of ADRS and ORT. As shown in the table, compared with RF, MLP, DT, XG-

TABLE III Experimental results of executing different convolution calculations in VGG and GoogLeNet on the Pareto-optimal CGRA microarchitecture design explored by PAMBOF (time unit: ms).

Input feature map (height×weight×channel)	Kernel (height×weight×channel×number)	Stride	Output feature map (height×weight×channel)	Execution time	Prediction time	Time deviation
58×224×16	3×3×16×16	1	56×224×16	2.498	2.425	2.9%
3×7×832	1×1×832×32	1	3×7×32	0.176	0.173	1.7%
4×14×512	1×1×512×16	1	4×14×16	0.113	0.110	2.7%
13×13×192	3×3×192×32	1	13×13×32	0.859	0.810	4.9%
4×7×48	3×3×48×16	1	3×7×16	0.023	0.022	4.5%
61×229×4	7×7×4×16	2	28×112×16	1.404	1.350	3%
55×55×64	5×5×64×16	2	27×27×16	1.281	1.200	6.7%

TABLE IV Comparison of manual design and Pareto-optimal design (time unit: ms, area unit: mm^2).

Approach	Parameters	Area	Performance time
Manual Design	[5,4,4,4,20,20,32,2]	0.13	4.996
Pareto-optimal Design	[6,4,4,4,20,20,32,2]	0.15	2.425

Boost, and BOOM-Explorer, the proposed PAMBOF achieves 76%, 70%, 60%, 44%, and 30% improvements in ADRS, respectively. Meanwhile, the ORT of the proposed PAMBOF is reduced by 10% compared to BOOM-Explorer.

In addition, Fig. 6 visualizes the corresponding Pareto frontiers discovered by RF, MLP, DT, XGBoost, BOOM-Explorer [17], and our framework in design space, respectively. As shown in the figure, the Pareto frontier searched by our PAMBOF framework is better than other approaches.

Based on the Pareto-optimal design generated by the proposed PAMBOF, the component parameters listed in TABLE I are configured with [6,4,4,4,20,20,32,2]. Compared to the actual area result of 0.1565 mm^2 obtained from Design Compiler on 28nm technology, the proposed area model evaluates an area of 0.15 mm^2 with only 4.5% error. Table III further records the overall time required for various convolution calculations on the explored Pareto-optimal CGRA design. These convolution calculations are in CNNs, i.e., VGG and GoogLeNet. Columns "Execution time" and "Prediction time" represents the time obtained through accurate RTL simulations using the VCS and the performance model, respectively. Column "Time deviation" refers to the estimation deviation for the performance model. The minor time deviation demonstrates that performance model is credible.

Additionally, we also compare the Pareto-optimal design explored by the proposed PAMBOF with a manual design developed by senior engineers. TABLE IV lists the comparison results when performing the convolution calculation listed in the first row of TABLE III. The area and performance are evaluated by the built models. As shown in the table, in contrast to a manual design, our Pareto design achieves 51% performance improvement with only 15% increase in area, demonstrating the effectiveness of our proposed PAMBOF.

VI. Conclusion

In this paper, we have proposed a PAMBOF framework to explore the Pareto-optimal CGRA microarchitecture design among the design space. Experimental results show that compared with the previous methods, the proposed PAMBOF achieves a better CGRA microarchitecture design in a shorter runtime. We expect more research on microarchitecture design space exploration to accelerate the chip design process.

References

[1] J. Li, Y. Qiu, G. Zhu, Q. Zhu, W. Yin, and L. Wang, "Thram: A template-based heterogeneous cgra modeling framework supporting fast dse," in *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2023, pp. 1–5.

[2] B. C. Schafer, T. Takenaka, and K. Wakabayashi, "Adaptive simulated annealer for high level synthesis design space exploration," in *Proc. VLSI-DAT*, 2009, pp. 106–109.

[3] A. Sengupta and V. K. Mishra, "Integrated particle swarm optimization (i-PSO): An adaptive design space exploration framework for power-performance tradeoff in architectural synthesis," in *Proc. ISQED*, 2014, pp. 60–67.

[4] B. Yuan, H. Chen, and X. Yao, "Toward efficient design space exploration for fault-tolerant multiprocessor systems," *IEEE TEVC*, vol. 24, no. 1, pp. 157–169, 2019.

[5] S. Chen, W. Sun, X. Ni, X. Jiang, X. Bai, and Y. Kang, "Deep learning-based coarse-grained reconfigurable array system and computing framework," Oct. 11 2022, China Patent CN202210798554.5.

[6] S. Chen, X. Ni, K. He, Y. Tao, W. Sun, and Y. Kang, "Design and implementation of operators for deep learning-based coarse-grained reconfigurable array," Mar. 3 2023, China Patent 2023102162419.

[7] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*. Pearson, 2005.

[8] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006.

[9] S. Daulton, M. Balandat, and E. Bakshy, "Differentiable expected hypervolume improvement for parallel multi-objective bayesian optimization," in *Proc. NIPS*, 2020, pp. 9851–9864.

[10] Q. Sun, C. Bai, H. Geng, and B. Yu, "Deep neural network hardware deployment optimization via advanced active learning," in *Proc. DATE*, 2021, pp. 1510–1515.

[11] 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.

[12] H. J. Kushner, "A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise," *Journal of Basic Engineering*, vol. 86, no. 1, pp. 97–106, 1964.

[13] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global optimization*, vol. 13, no. 4, pp. 455–492, 1998.

[14] N. Srinivas, A. Krause, S. Kakade, and M. Seeger, "Gaussian process optimization in the bandit setting: No regret and experimental design," in *Proc. ICML*, 2010, pp. 1015–1022.

[15] K. Yang, M. Emmerich *et al.*, "Multi-objective bayesian global optimization using expected hypervolume improvement gradient," *Swarm and evolutionary computation*, vol. 44, pp. 945–956, 2019.

[16] M. Balandat, B. Karrer, D. Jiang, S. Daulton, B. Letham, A. G. Wilson, and E. Bakshy, "BoTorch: a framework for efficient monte-carlo bayesian optimization," in *Proc. NIPS*, 2020, pp. 21 524–21 538.

[17] C. Bai, Q. Sun, J. Zhai, Y. Ma, B. Yu, and M. D. Wong, "BOOM-Explorer: RISC-V BOOM microarchitecture design space exploration framework," in *Proc. ICCAD*, 2021, pp. 1–9.