SenseDSE: Sensitivity-based Performance Evaluation for Design Space Exploration of Microarchitecture

Zheng Wu[1], Xiaoling Yi[1], Li Shang[1,2], and Fan Yang[1]
[1] State Key Lab of Integrated Circuits and Systems, School of Microelectronics, Fudan University.
[2]Shanghai Key Laboratory of Data Science, School of Computer Science, Fudan University.

***Abstract*—The design of modern processors is driven by plenty of benchmarks. As processors evolve and applications expand, the complexity of benchmark programs grows, which increases the computational cost of architecture design space exploration (DSE). To accelerate performance evaluations of processors in DSE, we developed a sensitivity-based framework for performance evaluation of a large set of benchmarks. The framework avoids simulating the insensitive benchmarks to the adjusted parameters during the exploration of designs. We developed a sampling algorithm based on evolutionary strategies to provide learning data for the sensitivity analysis and enhance the performance of the fast performance evaluation algorithm. We integrated this framework into a RISC-V processor architecture exploration framework. Our experiments revealed that we could achieve a significant acceleration in runtime with negligible accuracy loss in DSE.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

As technology continues to evolve at a rapid pace, the demand for more efficient and powerful CPUs has grown exponentially. Various applications, from artificial intelligence to real-time systems and high-performance computing, require optimized CPU architectures to achieve maximum performance. Design space exploration (DSE) involves evaluating different design alternatives concerning various parameters such as power consumption, performance, and area. It is crucial to find the optimal trade-off among these parameters to develop a CPU that meets the specific requirements of an application while minimizing costs.

Architectural exploration faces significant challenges. Firstly, the search space of the architecture itself is a high-dimensional and vast complex space. Both traversing the design space and modeling the relationship between architecture and performances are challenging tasks. The DSE framework in [1] and [2] attempt to use Bayesian optimization (BO) [1] to find higher quality CPU designs in the search space with fewer design simulation iterations. Secondly, during DSE, each CPU performance simulation consumes vast computational resources. As CPUs are general-purpose, they necessitate a comprehensive benchmark suite for performance assessment. With the proliferation of digital scenarios in recent times, the spectrum of CPU applications and the benchmark set size have surged, heightening the simulation pressures in architectural exploration.

Reducing benchmark simulation time is crucial for Design Space Exploration. One approach involves training program-specific models in the factory using large simulation data from part of the benchmarks and then using a linear regression model to predict the performance of new benchmark programs with small data [3]. However, this method requires a significant amount of simulation data (at least 512 simulation results for each program-specific model), and at least five program-specific models are needed to construct the prediction framework, resulting in a substantial time investment and a 20% error rate, leading to low DSE efficiency. Another approach is using subset techniques like SimPoint [4] and SMARTS [5] to select a small but representative portion of the benchmark suite or workload, thereby reducing simulation time while maintaining result accuracy. A statistical framework [6] utilizing principal components analysis (PCA) can also map benchmarks into an independent workload space, cluster them, and avoid executing similar benchmarks, thus accelerating simulation. However, these methods focus mainly on software analysis and overlook the interrelation between software and hardware during simulation, which is a significant limitation.

This paper introduces SenseDSE, a streamlined Design Space Exploration framework featuring rapid simulation techniques. Equipped with a rapid simulation framework based on sensitivity analysis of architecture to benchmarks, a meticulously crafted initial sampling algorithm, and a hierarchical multi-objective Bayesian optimization framework, SenseDSE efficiently explores the Pareto-optimal designs. The pivotal contributions of SenseDSE encapsulate the following aspects:

- We present SenseSimulate, a framework designed to accelerate simulations. It analyses the sensitivity of the benchmark performance to the microarchitectures with machine learning techniques and reduces the redundant re-simulation of insensitive benchmarks to the adjusted architecture parameters during DSE, enhancing the optimization efficiency.
- We propose an initial sampling algorithm based on evolutionary algorithms, aimed at finding a uniformly distributed initial design set. This not only enhances the efficiency of SenseSimulation but also allows machine learning models within the DSE framework to capture the characteristics of the search space more accurately at the onset of the search.
- We incorporate the aforementioned techniques into a design space exploration framework, leveraging an efficient hierarchical multi-objective Bayesian optimization algorithm driven by benchmark sensitivity to expedite the identification of high-quality designs. Results indicate that these rapid simulation techniques result in a 2× speed up to the exploration, with negligible compromising the quality

Corresponding author: yangfan@fudan.edu.cn

979-8-3503-4859-0/DATE24/© 2024 EDAA

of the identified design sets.

The rest of this paper is organized as follows. Section II introduces the BOOM processor, and CPU benchmarks and defines the microarchitecture DSE problem. Section III presents SenseDSE, the proposed fast simulation DSE framework. Section IV demonstrates the experiment results. Section V concludes the work.

II. Preliminary

A. Microarchitecture

A processor's microarchitecture, often termed "μarch", describes the design and organization of its core components. It dictates how the processor executes machine instructions and interacts with memory and caches. While the instruction set architecture (ISA) defines the machine language, microarchitecture determines how a CPU implements that ISA, affecting its performance and efficiency.

The Berkeley-Out-of-Order Machine (BOOM) [7] is a renowned open-source RISC-V microarchitecture that stands out in architecture research due to its superior power efficiency. The BOOM pipeline units are categorized into three functional modules: Frontend, Execute, and Load/Store Units. These modules involve various microarchitecture design parameters like *FetchWidth*, *DecodeWidth*, cache size, reorder buffer entries, and load & store queue length, which significantly affect the processor's Performance, Power, and Area (PPA).

Chisel [8] enables the generation of various BOOM cores with different microarchitecture configurations. However, microarchitecture exploration presents challenges due to the vast design space and the intricate trade-offs between performance, power consumption, and area. Identifying optimal configurations requires navigating this complexity while considering ever-evolving workload demands.

B. Benchmark

Benchmarking is a vital tool in CPU design, used to assess and compare the performance of different processor architectures. It involves standardized tasks that mimic real-world computing, helping designers measure execution speed, power efficiency, and more. These results guide improvements, optimize components, and enable fair CPU comparisons, ensuring processors meet modern computing needs effectively. Common benchmarks include Geekbench, Cinebench, PassMark, SPEC, and so on.

As modern processors consistently evolve in performance, the scope of computer applications broadens, resulting in a growing number of benchmark tests. For CPU architects, simulating all benchmark programs becomes time-consuming. In design space exploration, the repeated simulation of benchmark suites significantly hinders the pace of architectural optimization. In industry, benchmarks are typically simulated by iteratively running program segments using simple equal partitioning. Nevertheless, this approach ignores the software and hardware information that can be obtained during the DSE process and can lead to greater performance evaluation errors.

C. DSE Problem Formulation

Microarchitecture design space exploration (DSE) aims to discover superior designs that achieve an optimized trade-off among PPA values. As such, the microarchitecture DSE problem can be formulated as a multi-objective optimization problem, as follows:

$$\begin{aligned} \text{minimize } & \boldsymbol{f}(\boldsymbol{x}) = \{-Perf(\boldsymbol{x}), Power(\boldsymbol{x}), Area(\boldsymbol{x})\}, \\ \text{s.t. } & \boldsymbol{x} \in \boldsymbol{D}, \end{aligned} \tag{1}$$

where $\boldsymbol{f}(\boldsymbol{x})$ denotes the objective function of DSE. $Perf(\boldsymbol{x})$, $Power(\boldsymbol{x})$, and $Area(\boldsymbol{x})$ represent the performance, power, and area of the design with parameter $\boldsymbol{x}$, respectively. $\boldsymbol{D}$ denotes the whole design space. In multi-objective optimization, the aim is to identify a set of Pareto-optimal solutions, not just a single solution. A Pareto-optimal solution is one where one objective cannot be improved without worsening another objective. These solutions form the Pareto front, a set of points representing trade-offs among conflicting objectives, and provide a comprehensive view of the best possible trade-offs in the design space.

III. Algorithm

A. Overview

To accelerate the simulation of large benchmark sets in design space exploration, we designed an efficient DSE framework based on hardware-software sensitivity analysis. The algorithm structure is illustrated in Fig. 1.

Initially, the processor's design space is fed into the DistanceEA sampling algorithm to produce a uniformly distributed design set. These initial configurations are assessed using RTL-level simulation tools, with results added to the observed data set. A hierarchical multi-objective Bayesian optimization (BO) algorithm then derives a high-quality design suggestion from this data. This hierarchical BO first targets key parameter configurations before embarking on a global search, optimizing the exploration efficiency. Subsequently, a collection of machine learning-driven benchmark-sensitivity models is developed to map out the impact of microarchitecture parameters on benchmark performance. Based on these models, an importance table is constructed, pinpointing benchmarks sensitive to modifications in the suggested design's parameters. Only benchmarks sensitive to these changes are re-simulated, preventing unnecessary repetitions of benchmarks with minor performance shifts. The outcomes of these simulations are then assimilated into the observed data set. This Bayesian optimization process will be repeated multiple times, resulting in a Pareto-optimal design set in the end.

B. SenseSimulation: Efficient Benchmark Simulation Algorithm

Conducting a comprehensive set of benchmark tests is essential to verify the CPU's adaptability under varied conditions. However, when we were exploring the CPU design space, we found that modifying some parameters may lead to minor performance changes to benchmark simulation results. This is because the benchmarks have a testing propensity. For example, some of the benchmark programs are computational-intensive

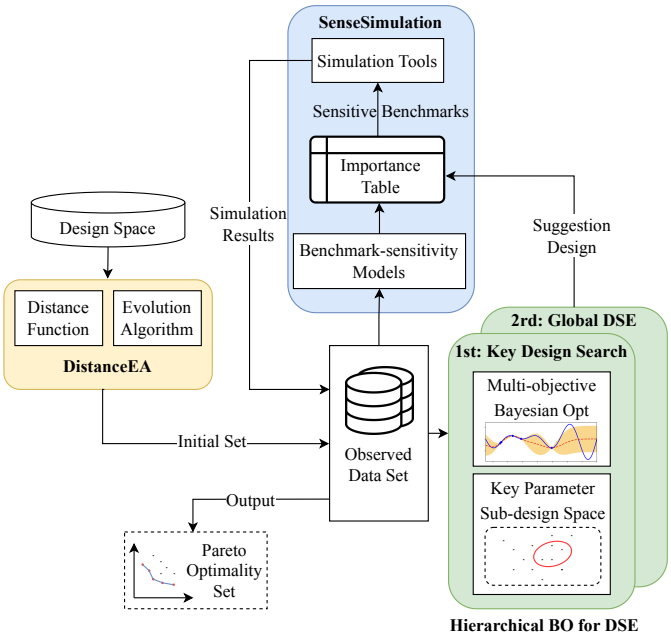


Fig. 1: The structure of the SenseDSE framework.

while others are memory-intensive. Some of the programs test the performance of the Integer calculation performance while others test the float-point calculation performance. During the design space exploration, modifying microarchitecture parameters to which the benchmark program is not sensitive will not bring much difference to the performance simulation result.

Leveraging insights into benchmark sensitivity to microarchitecture parameters, we employ a machine learning model trained on simulation data from DSE. This model aids in analyzing benchmark sensitivity and devising an algorithm to minimize redundant simulations during architectural optimization. The algorithm is listed in Algorithm 1.

Algorithm 1 SenseSimulation Benchmark Selection Algorithm

Require: The sampled design configurations X, the simulated benchmark performance Y, and the suggestion design to be evaluated x.
Ensure: The benchmark performance y_{perf} of x
1: Train *benchmark-sensitivity Models* M with X and Y;
2: Construct *importance table* T with the weight values w of M;
3: Set *neighbor design* $x_n \leftarrow$ the closest design to the suggestion design x in the simulated design set;
4: Get the *adjusted parameters* d_p between x and x_n;
5: **for** each $benchmark \in benchmark\ suite$ **do**
6: **if** $\forall p \in d_p$: $benchmark$ is not sensitive to p **then**
7: Set the performance of x_n to x;
8: **else**
9: Simulate $benchmark$ on the design x;
10: **end if**
11: **end for**
12: **return** Performance vector y_{perf} of x;

In the algorithm, we use the Lasso model [9] to characterize the relationship between the microarchitecture parameters and the benchmark performance. Lasso is a machine learning technique used for feature selection and regression by adding

	Decode	Fetch	Dispatch	Issue	Entries	Registers
Benchmark1	0	0.1	0.2	0	0.2	0
Benchmark2	0	0	0.2	0.3	0.25	0
Benchmark3	0	0.1	0.1	0.1	0	0

Fig. 2: An example of the importance table.

a penalty term to the linear regression equation to encourage sparsity in the model. The optimization target of the Lasso model can be formulated as:

$$\min_{w} \frac{1}{2n_{\text{samples}}} ||Xw - y||_2^2 + \alpha||w||_1, \tag{2}$$

where w denotes the weight of each microarchitecture parameter to the performance result and α denotes the regularization parameter. The simulation results of each benchmark program are used to train a Lasso model which is called *benchmark-sensitivity model* M. Throughout the training phase, certain weight parameters are rendered to zero due to L1 regularization. Benchmark results are insensitive to changes in these architecture parameters with weights equal to 0.

The weights of architecture parameters in all the *benchmark-sensitivity models* are collected in the *Importance Table*. Fig. 2 illustrates an example of the importance table which indicates that the importance table is sparse and the benchmark performance results are sensitive to only parts of microarchitecture parameters. Within the *Importance Table*, the parameter set deemed vital across all benchmark outcomes is designated as the *Importance Zone*.

During the DSE process, the Bayesian optimization algorithm proposes a *suggestion design*. The SenseSimulation attempts to match this *suggestion design* with the parameters of previously simulated designs. The objective is to pinpoint a *neighbor design* that shares the maximum number of architecture parameters with the *suggestion design* beyond the *Importance Zone*. For benchmarks that exhibit insensitivity to all *adjusted parameters* which are the different parameters between the *suggestion* and the *neighbor design*, no further simulation of this benchmark on this *suggestion design* is conducted. This signifies that the modification of these parameters from the *neighbor design* to the *suggested design* is unlikely to yield substantial performance alterations. SenseSimulation will adopt the simulation outcomes derived from the *neighbor design* as the corresponding results for the *suggested design*.

For example, if the *adjusted parameters* are *Entries* and *Registers*, according to the importance table in Table 2, Benchmark 3 will not be needed for simulation because it is not sensitive to the *adjusted parameters*. Therefore, the performance result of Benchmark 3 on *suggestion design* can be approximated as the performance result of the *neighbor design*.

C. DistanceEA Sampling Algorithm

In SenseSimulation, finding a close *neighbor design* to the *suggestion design* is an important process. As benchmarks need to be insensitive to all *adjusted architecture parameters* when opting not to simulate, fewer *adjusted parameters* can offer more opportunities to reduce benchmark simulations.

To sample a design set that can minimize the number of *adjusted parameters* in the *suggestion designs* obtained during the exploration process, we employ a formula to depict our desired average number of adjusted parameters as:

$$distance(X_{sampled}) = \frac{1}{m}\sum_{m}^{i} \min\left(\mathrm{adj}_i(X_{sampled})\right), \quad (3)$$

where m denotes the number of unsimulated designs, adj_i denotes the number of the *adjusted parameters* between design i and the designs in the simulated design set, and $X_{sampled}$ is the initial simulated design set. By minimizing the object function $distance(X_{sampled})$, a design set, which has the minimum number of *adjusted parameters* to the rest of the designs in the search space, is sampled. Moreover, the minimum distance design set distributes uniformly in the search space which will help the benchmark-sensitivity model and the surrogate model of BO capture the data features more comprehensively.

The Evolutionary algorithm [10] is used to optimize the function and get an optimal design set. Evolutionary algorithms are versatile and adaptive, suitable for a broad range of optimization problems. Their bio-inspired nature allows for global optimization, avoiding local optima, while their stochastic approach ensures diverse solution exploration and robustness against problem specifics. After undergoing 100 generations of evolutionary algorithm optimization, an initial design set is obtained.

D. SenseDSE: Hierarchical DSE Framework

1) Sensitivity-driven Design Space Exploration: In order to enhance the efficiency of exploring the design space, a hierarchical multi-objective Bayesian optimization (BO) framework is proposed. The architecture exploration process is divided into two steps according to the importance of the microarchitecture parameters. In the first step, we initially explore within a sub-search space composed of key parameters. In the second step, we conduct global exploration. The structure of the hierarchical DSE framework is illustrated in Fig. 3.

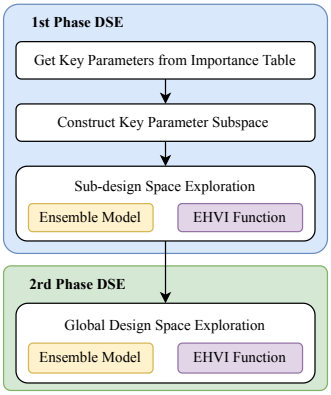


Fig. 3: Hierarchical DSE Framework

In the first optimization phase, we can identify parameters that have a significant impact on processor performance according to the benchmark-sensitivity model. The architecture parameters that are crucial for the performance of approximately 80% of benchmarks according to the importance table will be considered the key architecture parameters. A sub-search space is extracted from the total search space, which includes all possible combinations of the key parameters. Then we conduct design space exploration among the sub-search space.

Prioritizing the search in the key parameter sub-search space improves efficiency by focusing on regions of the parameter space that are more likely to yield optimal outcomes. This approach enables us to sample a wider range of combinations for key parameters. Consequently, when executing the SenseSimulation framework during subsequent searches, it brings more possibility to find designs that share the same key parameters. This, in turn, results in a sparser importance table in adjusted parameter regions, thereby enhancing the acceleration effect of simulations.

In the second optimization phase, we will conduct a comprehensive search across the entire design space to guarantee the inclusion of the global Pareto optimal set in our findings.

2) Efficient Multi-objective Bayesian Optimization Framework for DSE: Bayesian optimization [11] is a model-based method ideal for Design Space Exploration (DSE) due to its efficient targeting of global optimum using prior beliefs, especially when evaluations are costly and the design space is large. For our challenge, considering performance, area, and power trade-offs, the multi-objective Bayesian optimization can identify an optimal Pareto design set.

In Bayesian optimization, ensemble models enhance robustness and predictive accuracy by combining predictions from multiple surrogate models, leading to more efficient and informed sampling of the design space. Random forest [12] and Gaussian process models [13], commonly used as surrogate models in Bayesian optimization, are integrated into our Bayesian optimization framework. The ensemble prediction of the surrogate model can be formulated as:

$$y_{ensemble} = (y_{RF} + y_{GP})/2. \quad (4)$$

In multi-objective optimization, hypervolume quantifies the quality of the Pareto front, measuring solution dominance and diversity. The EHVI (Expected Hypervolume Improvement) acquisition function [14] offers the advantage of effectively guiding multi-objective Bayesian optimization towards Pareto-optimal solutions, which can be formulated as:

$$\begin{aligned}\mathrm{EHVI}(\boldsymbol{x}) &= \mathbb{E}_{p(f(\boldsymbol{x}))}\left[\mathrm{HV}_{\boldsymbol{y}_{\mathrm{ref}}}(\mathcal{P}(\boldsymbol{y}) \cup \boldsymbol{f}(\boldsymbol{x})) \right. \\ &\quad \left. - \mathrm{HV}_{\boldsymbol{y}_{\mathrm{ref}}}(\mathcal{P}(\boldsymbol{y}))\right], \\ \mathrm{HV}_{\boldsymbol{y}_{\mathrm{ref}}}(\mathcal{P}(\mathcal{Y})) &= \int_{\mathcal{Y}} \mathbb{I}\left[\boldsymbol{y} \succcurlyeq \boldsymbol{y}_{\mathrm{ref}}\right]\left[1 - \prod_{y_* \in \mathcal{P}(\mathcal{Y})} \mathbb{I}\left[\boldsymbol{y}_* \not\succcurlyeq \boldsymbol{y}\right]\right]\mathrm{d}\boldsymbol{y},\end{aligned} \quad (5)$$

where $\mathcal{P}(\mathcal{Y})$ is the Pareto-optimal set, $\boldsymbol{f}(\boldsymbol{x})$ means the prediction of $\boldsymbol{x}$ from ensemble surrogate models and $\boldsymbol{y}_{\mathrm{ref}}$ denotes the reference point which is dominated by each element of $\mathcal{P}(\mathcal{Y})$. The expectation $\mathbb{E}_{p(f(\boldsymbol{x}))}$ of the hypervolume improvement is calculated by the Monte-Carlo method which samples the hypervolume improvement values and computes their average value. This function is optimized using traversal techniques, minimizing computational overhead and identifying designs with the highest potential for Pareto optimally for VLSI evaluation. The EHVI function co-related with the ensemble

surrogate model constitutes a basic Bayesian optimization algorithm and serves its exploration job in the hierarchical DSE framework. The suggestion design will be evaluated with the EDA simulation tools which is accelerated by the SenseSimulation algorithm.

IV. Experimental Results

A. Experiment Settings

The efficient simulation framework is applied for the design space exploration of the RISC-V BOOM processor. ICCAD Contest 2022 Problem C [15] focuses on the DSE of BOOM, in which 29 microarchitecture parameters of the core are chosen, which are listed in Table I. Due to certain parameter constraints in the BOOM processor design, the search space does not encompass all possible combinations of these parameter options. Overall, the ICCAD Contest 2022 provides a microarchitecture design space comprising 1.5×10^4 design candidates.

TABLE I: The Microarchitecture Design Space of BOOM processor.

Module	Parameters	Values
Front End	FetchWidth	4, 8
	FetchBufferEntry	8, 16, 24, 32, 35, 40
	FetchTargetQueueEntry	16,40,32
	ICacheSets	32,64,128
	ICacheWays	4,8
	ICacheTLBSets	1, 2
	BranchCount	6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26
	DecodeWidth	1, 2, 3, 4, 5
	RobEntries	32, 64, 96, 128, 130
Execution Core	Int/FpPhysRegisters	48, 64, 80, 96, 112
	Mem/Fp/IntDispatchWidth	1, 2, 3, 4, 5
	Mem/Fp/IntNumEntries	4,8,12,16,20,24,32,40
	Mem/FpIssueWidth	1,2
	IntIssueWidth	1, 2, 3, 4, 5
Load/Store Unit	Ld/StqEntries	8, 16, 24, 32
	DCacheSets	32,64,128
	DCacheWays	2, 4, 8
	DCacheTLBSets	1,2
	DCacheTLBWays	8,16,32
	DCacheMSHR	2, 4, 8
	ReplacementPolicy	0, 1

Since the ICCAD Contest neither discloses PPA data nor provides an option to modify benchmark simulation, we use the Chipyard to generate the BOOM cores of different microarchitecture configurations to evaluate the PPA of the designs. RTL-level simulations are conducted with Synopsys VCS simulator and Design Compiler to get the PPA of the microprocessor. The simulations were executed on a cluster equipped with an Intel Xeon Platinum 8354H CPU@3.10GHz. Due to the time-consuming nature of the VLSI verification flow, we obtained a limited number of PPA metrics of designs, which constructed a search space of 974 processor designs.

As for the benchmarks, we choose the riscv-test [16] benchmarks, which include common benchmark programs to evaluate the performance of the processors. 12 common used benchmark programs are selected, including dhrystone, median, mt-matmul, mt-vvadd, multiply, pmp, qsort, rsort, spmv, towers, vvadd, and mm. These benchmarks encompass a comprehensive set of RISC-V instructions, enabling a thorough assessment of the microarchitecture design's performance.

B. Benchmark-Architecture Sensitivity Validation

In order to validate the idea that the benchmark performance results are sensitive to specific microarchitecture parameters, we trained 12 random forest models and got the Gini importance [12] of microarchitecture parameters to each benchmark performance result. Fig 4 shows the proportion of different architectural parameter weights under various benchmarks. The results indicate that a small portion of the parameters account for the majority of the performance metric weights in the benchmarks.

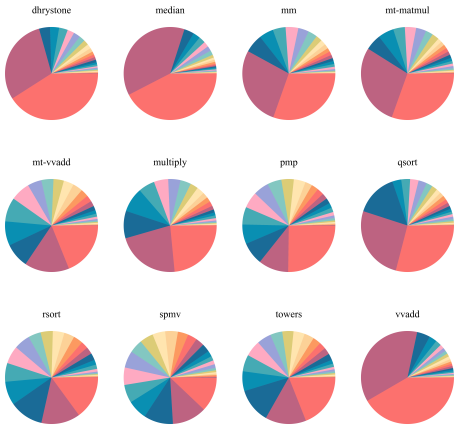


Fig. 4: The importance proportions for each architectural parameter to the benchmark programs. Each segment in the pie chart represents the proportion of sensitivity of benchmark performance to microarchitecture parameters.

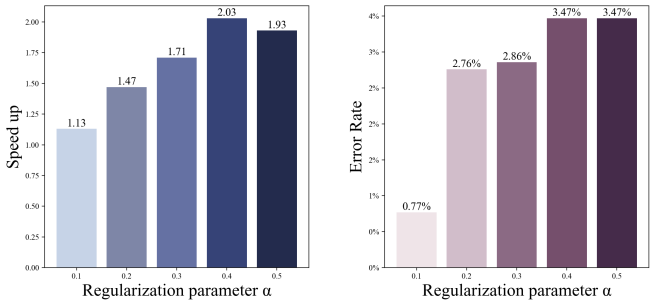
Table II displays the weight proportions affecting performance under each benchmark and the number of architectural parameters that account for more than 80%. The results indicate that only half of the architectural parameters can contribute to 80% of the weight in these benchmark results, while some benchmark results can be covered by the sum of the weight of just 20% of the architectural parameters. This might be the 80-20 rule in the DSE field.

TABLE II: The Number of Architecture Parameters that Cover 80% Importance to Each Benchmark.

Benchmark	Dhrystone	Median	MM	Mt-matmul	Mt-vvadd	Multiply
Num	5	3	8	8	10	8
Benchmark	Pmp	Qsort	Rsort	Spmv	Towers	Vvadd
Num	11	6	10	11	12	3

C. Effectiveness of SenseDSE

In order to validate the efficiency of SenseDSE, design space exploration of the BOOM processor experiments are conducted with SenseDSE. 20 initial designs are sampled by the DistanceEA algorithm. Then 30 and 50 rounds of Bayesian optimization are conducted in the first and the second phases respectively. The overall execution time speed-up rate compared to the BOOM-Explorer [1] and the average approximation simulation error of SenseDSE with different regularization parameters are shown in Fig. 5. The results in Fig. 5a indicate that compared to DSE without using the sensitivity-based approximate simulation algorithm, we can achieve up to 2 times search acceleration. This acceleration is the overall speed-up, including the overhead caused by the initial sampling phase without approximate simulation processing. Fig. 5b shows that the average error caused by the approximate simulation process is 3.47% under 2x acceleration. The experiments demonstrate that a larger regularization parameter will consider fewer important parameters, leading to greater simulation acceleration,



(a) Speed up (b) Approximation error

Fig. 5: The acceleration and error result under different regularization parameters.

but the cost is that the average simulation error will also be larger.

We compared the result of the DSE algorithm with and without SenseSimulation to see whether the approximation simulation framework harmed the result of the DSE problem. We use the Pareto hypervolume and the Generational distance [17] to estimate the quality of the found Pareto optimal design set. The results in Table III suggest that the SenseDSE framework w/ SenseSimulation has a 1.8‰ loss in hypervolume and a 2.2% loss in Generational distance. SenseDSE achieves nearly identical search results as the current best BOOM-Explorer [1] and GRL-DSE [2] framework, yet the simulation time of SenseDSE is only half that of GRL-DSE. The time in Table III represents the sum of benchmark simulation times in the DSE algorithm. During actual architectural exploration, parallel techniques can be used to accelerate the exploration process.

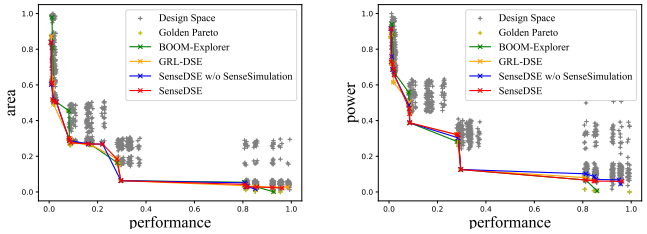
TABLE III: The Exploration Results Comparison of SenseDSE

	HV↑	GD↓	Simulation Time	Speed Up
BOOM-Explorer	0.6966	0.05695	354.33h	1×
GRL-DSE	0.6969	0.05690	340.78h	1.04×
SenseDSE w/o SenseSimulation	0.7042	0.05611	347.22h	1.02×
SenseDSE	0.7029	0.05742	156.70h	2.26×

Fig. 6 displays the Pareto frontiers identified by BOOM-Explorer, GRL-DSE, SenseDSE, and SenseDSE w/o SenseSimulation. The units on the graph represent normalized PPA (Power, Performance, Area), and the performance metric is the average benchmark runtime. Therefore, a smaller value indicates stronger design performance. The results show that the Pareto frontiers identified by the three DSE methods are nearly consistent. This suggests that the approximate simulation algorithm of SenseSimulate has minimal impact on the DSE results. SenseDSE is able to halve the simulation time while maintaining consistency with the state of the art.

V. CONCLUSION

This paper introduces a processor architectural exploration framework using benchmark-architecture sensitivity analysis. It minimizes redundant simulations of insensitive benchmarks, reducing overhead in DSE. Efficient multi-objective Bayesian optimization algorithms are designed to enhance exploration efficiency. In experiments on the BOOM processor, SenseDSE achieves a 2x performance simulation acceleration, with negligible impact on final exploration results. We aim to bridge the gap in rapid simulation technology for software-hardware co-design, fostering swift processor iteration.



(a) (b)

Fig. 6: (a) The Pareto frontier in performance-area space; (b) The Pareto frontier in performance-power space.

ACKNOWLEDGMENT

This research is supported partly by National Key R&D Program of China 2020YFA0711900, 2020YFA0711901, partly by National Natural Science Foundation of China (NSFC) research projects 92373207, 62090025, and 62141408, and partly by Science and Technology Commission of Shanghai Municipality Project (22DZ1101100).

REFERENCES

[1] 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 *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 2021, pp. 1–9.

[2] X. Yi, J. Lu, X. Xiong, D. Xu, L. Shang, and F. Yang, "Graph representation learning for microarchitecture design space exploration," in *2023 IEEE/ACM Design Automation Conference (DAC)*. IEEE, 2023.

[3] C. Dubach, T. Jones, and M. O'Boyle, "Microarchitectural design space exploration using an architecture-centric approach," in *40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 2007)*. IEEE, 2007, pp. 262–271.

[4] E. Perelman, G. Hamerly, and B. Calder, "Picking statistically valid and early simulation points," in *2003 12th International Conference on Parallel Architectures and Compilation Techniques*. IEEE, 2003, pp. 244–255.

[5] R. E. Wunderlich, T. F. Wenisch, B. Falsafi, and J. C. Hoe, "Smarts: Accelerating microarchitecture simulation via rigorous statistical sampling," in *Proceedings of the 30th annual international symposium on Computer architecture*, 2003, pp. 84–97.

[6] L. Eeckhout, H. Vandierendonck, and K. De Bosschere, "Workload design: Selecting representative program-input pairs," in *Proceedings. International Conference on Parallel Architectures and Compilation Techniques*. IEEE, 2002, pp. 83–94.

[7] 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 Berkeley United States, Tech. Rep., 2015.

[8] 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 *DAC Design automation conference 2012*. IEEE, 2012, pp. 1212–1221.

[9] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 58, no. 1, pp. 267–288, 1996.

[10] T. Bäck, D. B. Fogel, and Z. Michalewicz, *Evolutionary computation 1: Basic algorithms and operators*. CRC press, 2018.

[11] J. Mockus, *Bayesian approach to global optimization: theory and applications*. Springer Science & Business Media, 2012, vol. 37.

[12] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in *International conference on learning and intelligent optimization*. Springer, 2011, pp. 507–523.

[13] C. E. Rasmussen, "Gaussian processes in machine learning," in *Summer school on machine learning*. Springer, 2003, pp. 63–71.

[14] A. Shah and Z. Ghahramani, "Pareto frontier learning with expensive correlated objectives," in *International conference on machine learning*. PMLR, 2016, pp. 1919–1927.

[15] X. W. Sicheng Li, Chen Bai, "ICCAD CAD Contest 2022," 2022.

[16] Jun 2023. [Online]. Available: https://github.com/riscv-software-src/riscv-tests

[17] D. A. Van Veldhuizen, *Multiobjective evolutionary algorithms: classifications, analyses, and new innovations*. Air Force Institute of Technology, 1999.