Design space exploration of HPC systems with Random Forest-based Bayesian Optimization

Vincent Fu, Lilia Zaourar, Alix Munier-Kordon, Marc Duranton

Design Space Exploration of HPC Systems with Random Forest-based Bayesian Optimization

Vincent Fu*
Université Paris-Saclay, CEA, List
Palaiseau, France
vincent.fu@cea.fr

Lilia Zaourar
Université Paris-Saclay, CEA, List
Palaiseau, France
lilia.zaourar@cea.fr

Alix Munier-Kordon
Sorbonne Université, CNRS, LIP6
Paris, France
alix.munier@lip6.fr

Marc Duranton
Université Paris-Saclay, CEA, List
Palaiseau, France
marc.duranton@cea.fr

ABSTRACT

Nowadays, High-Performance Computing (HPC) systems need to deliver computational performance by processing complex applications and workloads at high speeds in parallel. To provide computing power, Multiprocessor System-on-Chip, the main design paradigm, is scaled with advanced technology nodes and even with heterogeneity. These improvements open up more design possibilities, leading to an increase in its complexity. Therefore, chip designers are facing unprecedented challenges to find the best Power, Performance, and Area architectural configurations, inducing a Design Space Exploration problem. This work proposes a complete framework to ease the next generation of HPC processor designs. By combining competitive simulators VPSim and McPAT for a realistic estimation of Key Performance Indicators, with a time-consuming simulation-adapted exploration algorithm such as Bayesian Optimization, we leveraged an Automated Design Space Exploration for efficient HPC processor designs based on ARMv8 architecture. We have also demonstrated the potential of Bayesian Optimization to reach a similar or even larger Pareto front than Genetic Algorithm while being around 2× to 5× sample-efficient. Furthermore, the diversity of the obtained Pareto-front enables deep analysis of relevant architectural parameters that significantly impact design performances and thus, empowering architects' knowledge for further targeted design exploration and design choices.

CCS CONCEPTS

• **Hardware → Methodologies for EDA**; • **Computer systems organization → Multicore architectures**; • **Applied computing** → *Computer-aided design*; • **Theory of computation** → *Active learning*; • **Mathematics of computing → Bayesian computation**.

*Also with Sorbonne Université, CNRS, LIP6.

RAPIDO '24, January 18, 2024, Munich, Germany

ACM ISBN 979-8-4007-1791-8/24/01
https://doi.org/10.1145/3642921.3642923

KEYWORDS

Automated Design Space Exploration, High-Performance Computing, Multiprocessor System-on-Chip, Bayesian Optimization, Random Forest

ACM Reference Format:
Vincent Fu, Lilia Zaourar, Alix Munier-Kordon, and Marc Duranton. 2024. Design Space Exploration of HPC Systems with Random Forest-based Bayesian Optimization. In *Rapid Simulation and Performance Evaluation for Design (RAPIDO '24), January 18, 2024, Munich, Germany*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3642921.3642923

1 INTRODUCTION

The emergence of Artificial Intelligence and High-Performance Computing (HPC) applications requires more and more computing power. Various hardware options are arising to support these growing computing needs. From Multi-Processor System-on-Chip (MPSoC) to heterogeneous architectures, processor designers must face a combinatorial explosion of the design space, leading to a major challenge of seeking efficient computing architectures.

With the sophistication of current HPC systems and processor architectures, simulation platforms are widely used to access performance metrics (latency, power, etc.) at early design stages. It allows running applications on virtual prototypes. Thus, the simulation-based design flow enables designers to benchmark a processor architecture given its architectural configuration parameters, such as the number of cores, cache sizes, Network-on-Chip (NoC) placement, etc., and the target applications. Since simulations are time-consuming, the crucial issue for designers is to intelligently explore the design space to point out efficient configurations with as few simulations as possible. Analytical formulations [12, 22] for performance metrics have been proposed as a substitute for simulators but remain industrial data-dependent and unreachable at a higher stage level of design. In addition, they could not evaluate application execution dynamically.

Consequently, with the constraint to simulate, designers commonly employ brute-force methods to tune architectural configuration parameters and refine their search based on observations and expert knowledge. However, this method may be unreliable in a high-dimensional multiobjective optimization scheme, which is a main pattern in processor design problems. As a result, Design Space Exploration (DSE) methodologies [18] have gained a lot of

interest in the literature for providing better exploration scalability and design results.

This work follows the generic method of automating the DSE from the previous work A-DECA [24] (Automated Design space Exploration for Computing Architectures) by analyzing existing state-of-the-art optimization algorithms and their integration into different design flows. Our contributions are as follows:

- We built an automated DSE framework fully adapted for designing efficient HPC ARMv8 multicore processor system-level architectures with Power, Performance, and Area (PPA) multiobjective optimization.
- We used HyperMapper [15], a Bayesian Optimization exploration tool with VPSim [3] simulator and McPAT [13] modeler for a complete early system-level design approach.
- We compared Bayesian Optimization to the Genetic Algorithm of A-DECA framework [24] and showed model-based methods systematically outperform model-free methodologies in an expensive DSE problem with an extended design parameter set.
- As a result, A-DECA framework is enhanced with Bayesian Optimization features inside and provides a much faster new exploration strategy.

The paper is structured as follows. Section II discusses the state-of-the-art in exploration algorithms and their features. Section III introduces the proposed framework in detail, followed by the experimental results in Section IV. Finally, Section V ends with the conclusion and perspectives.

2 RELATED WORK

Design Space Exploration refers to two parts in the literature: a proposal for new simulation tool to evaluate configuration performance, and an efficient strategy to explore the design space.

In [7], the authors introduced MUSA simulator as a DSE enabler for large HPC machine designs. However, it does not provide a particular exploration algorithm to automate the search for optimal designs, which is the ultimate goal of the architect. Most DSE contributions just provide DSE enablers and often neglect the automated search. To move further towards the goal of fully solving architects' design problems, optimization algorithms from the Operational Research and Machine Learning literature must be integrated into the design flow to complete the process of finding optimal designs.

In Platune [6], a platform-tuning environment with cycle-accurate simulators inside for performance-power efficient System-on-Chip architecture design, Pareto Local Search has been used. This technique employs a local improvement heuristic to expand the search iteratively. However, it often exhibits large intermediate Pareto fronts as the number of objectives increases, leading to a time-consuming neighborhood computation for the next iterations. To reduce intermediate Pareto fronts, the authors observed that several architectural parameters can be clustered depending on their impacts on performance and power metrics. They built a parameter dependency model as a smaller Pareto front to reapply local search on it. Nevertheless, identifying parameter interdependencies requires a minimum amount of information about the objective space, which is rarely available.

Local optimality is the second limitation of the previous technique. Indeed, the exploration can be stuck in local optima. Simulated Annealing has been integrated in Cwbexplorer [21] for area-latency efficient micro-architectural design. It is a variant of Local Search that adds a restart mechanism to escape from local optima. Cwbexplorer enables micro-architectural exploration through pragma tuning with the High-Level Synthesis (HLS) tool CyberWorkBench. However, the proposed simulated annealer aggregates area and latency metrics into a linear weighted cost function to evaluate a design as a single objective procedure. Using a standard linear weighted aggregator leads to omitting Pareto-optimal solutions in non-convex search space regions.

Metaheuristics provide other novel approaches based on computational intelligence paradigm. For example, Particle Swarm Optimization has been exploited in the System Tuning Shell optimization framework [16]. It involves performance and energy consumption optimization for a system-level design using Wattch as simulation model. The authors adapted the Swarm approach for discrete design space based on the concepts of random walk theory. On the other hand, an enhanced version of Platune has been suggested with Genetic Algorithm [17]. By mapping the DSE problem into a genetic encoding formalism, the algorithm reproduces evolutionary theory to discover relevant designs.

Despite all the methods mentioned above having been used to build an automated DSE, we observed a significant number of simulations is required to be efficient. It means these methods are only suitable for cheap DSE. Fast simulations enable them to explore more and to use their strength in numbers to empower the search. These algorithms are identified as model-free methodologies since they operate without any specific mathematical properties. In contrast, Model-based methodologies uses mathematical models to learn the relationship between design parameters and performance metrics. By introducing probabilistic and statistical models, these methods can progressively capture the objective space structure during the exploration and subtly guide the search.

In [2], the authors are interested in delay-energy efficient multiprocessor design using ReSP and CACTI simulators with multimedia compression applications. They availed of the Markov Decision Process framework as a decision model to define an exploration policy through Reinforcement Learning. Value Iteration, a dynamic programming-based algorithm, has been used in the paper to identify the optimal policy. Although there are more relevant state-of-the-art Reinforcement Learning algorithms such as Multi-Agent Proximal Policy Optimization [10], which has been exploited for latency-power product efficient DRAM memory controller design with DRAMSys simulator, the reinforcement learning process naturally demands a large number of simulation samples before reaching the optimal policy.

Instead, Bayesian Optimization delivers a sample-efficient method to tackle expensive simulations. It combines a surrogate model that describes the objective function's behavior and an acquisition function that tells the next simulation to explore according to the information captured by the surrogate from the completed simulations. In BOOM-Explorer [1], the authors studied performance-power efficient micro-architecture designs of the RISC-V Berkeley Out-of-Order Machine (BOOM) through a VLSI flow. They have introduced

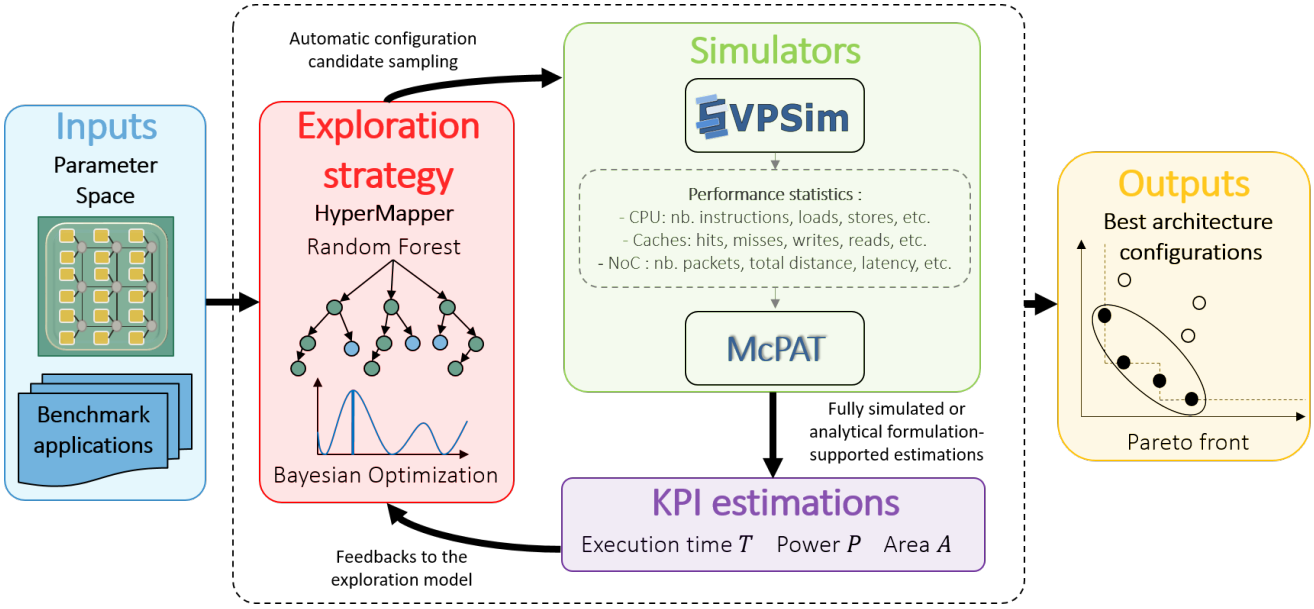


Figure 1: The generic methodology of automated Design Space Exploration with the use case instance.

an enhanced Bayesian Optimization algorithm with two key features: Deep Kernel Learning Gaussian Process (DKL-GP) surrogate model and Microarchitecture-aware Active Learning (MicroAL) initialization. DKL-GP enables better learning of the design space structure using the representational power of neural networks. Then, since surrogate models need to be set up with initial designs, sampling a representative set of designs would highly increase the performance of surrogate-based exploration. Such approaches like Latin Hypercube Sampling (LHS), I-optimal, and D-optimal designs are referred to as Design of Experiments [20] (DoE) techniques. MicroAL, in particular, used Transductive Experimental Design with design space clustering to generate a diverse initial set.

Moreover, Bayesian Optimization offers a powerful modular framework for exploration algorithm design. Although the standard version of Bayesian Optimization uses Gaussian Process (GP), it would be enhanced by adapting the surrogate model to the problem specificity. In [11], Tree-structured Parzen Estimator (TPE) has been exploited as a better surrogate model for dealing with the high-dimensional and discrete design space of Coarse-Grained Reconfigurable Architecture (CGRA). Thus, Bayesian Optimization is a promising global approach for automated DSE that requires time-consuming simulations.

As a result, all the above research demonstrated the potential of automating DSE through optimization algorithms. However, the non-optimization specialist must keep in mind that most of the used optimization algorithms for DSE are well-known baselines in the optimization community. It means that even an average-performance optimization algorithm can significantly improve architects' design tasks by providing unseen solution designs that can't be obtained with brute-force and *a priori* methods. Therefore, the architects should not undervaluate the importance of automating the DSE. The next section details our automated DSE framework adapted to optimize HPC processor designs.

3 AUTOMATED EXPLORATION METHODOLOGY

Figure 1 depicts the automation methodology. Starting from the parameter space and the set of target applications specified by the architect, an input candidate configuration is generated and automatically evaluated by combining high-level simulation results with or without analytical formulations. The exploration strategy will then consider this results information to sample the next interesting design efficiently. By repeating this procedure over and over, a closed-loop process is built and represents the main idea of how both exploration and optimization of complex designs work. Ideally, relevant automated exploration frameworks feature multi-processing simulations to reduce search time drastically.

3.1 Multiobjective Optimization formalism

Achieving relevant MPSoC designs to optimize different Key Performance Indicators (KPI) involves minimizing several metrics simultaneously. Since they are negatively correlated, minimizing one metric comes at the cost of increasing another. It leads to an essential notion of optimality in multiobjective optimization that we define below.

Pareto Optimality: let D be the design space and $n \geq 2$. Given a n-dimensional objective function:

$f(x) = (f_1(x), f_2(x), \ldots, f_n(x))$, a solution $x \in D$ is said to be dominated by $y \in D$ in minimization (resp. maximization) problem if:

$$\begin{aligned} &\forall i \in [\![1, n]\!], \quad f_i(y) \leq f_i(x) \quad (\text{resp. } f_i(y) \geq f_i(x)); \\ &\exists j \in [\![1, n]\!], \quad f_j(y) < f_j(x) \quad (\text{resp. } f_j(y) > f_j(x)). \end{aligned} \tag{1}$$

A solution x is Pareto-optimal if no other solution dominates it. The corresponding image set of Pareto-optimal solutions is called the Pareto front.

In single-objective optimization, the optimality is trivially deduced from the total order of real numbers. On the contrary, multi-objective optimization aims to reveal all possible Pareto-optimal solutions that correspond to different objective trade-offs.

In particular, we want to simultaneously minimize here: the energy consumption E, the execution time T and the area A. This rely on tuning set of architectural parameters $x \in D$ given a target application Γ:

$$\min_{x \in D} \; E(x, \Gamma), T(x, \Gamma), A(x) \tag{2}$$

The design space D is defined by the architect according to his reflection about sensitive parameters that need to be explored. As analytical formulations are not directly exploited here, we leverage a fully simulation-based design flow to compute each objective value accurately.

3.2 VPSim simulator and McPAT modeler

The choice of simulators is important in DSE problems. Indeed, the simulation speed and accuracy directly impact the ability to efficiently perform large exploration. However, both these criteria are antagonists, and the architects must trade-off by choosing selective simulators for their design tasks.

Recently, the HPC industry focused on ARM-based chips [19] as a new opportunity to archive large exascale systems with high computing efficiency while maintaining low power. This turnover pushes the emergence of HPC-dedicated simulators based on ARMv8 architecture. For instance, in the previous work A-DECA [24], VPSim [3] ARM cores customizable simulator and CACTI modeler have been combined for PPA evaluation of HPC processor designs. However, since CACTI only computes power and area for caches, the authors build analytical formulations for the area and power of an entire processor with rough formulations. Therefore, we propose here to improve A-DECA's simulation flow, using VPSim with McPAT [13] modeler for more accurate evaluations.

VPSim [3] is a virtual prototyping tool dedicated to Hardware-Software co-design for system-level architectures. It enables architectural simulations of different CPU models (RISC-V, ARM, etc.) with a good trade-off between simulation speed and accuracy. VPSim runs applications on simulated architectures to estimate the execution time and various hardware performance statistics [23] (cache accesses, misses, writes, NoC counters, etc.).

McPAT [13] is an integrated Power, Area, and Timing Modeling framework for many-core architectures. Built on the top of CACTI, McPAT follows the same analytical approach in the latter to provide power and area metrics for a complete MPSoC. It exploits common physics formulas, existing data, and future projections on processor designs to stand out as a parameterizable numerical model with curve fitting.

The computation of PPA estimations follows these steps:

(1) The architectural configuration candidate is set by instantiating its system-level parameters, such as the number of cores, cache sizes, NoC setting, etc., in VPSim.
(2) VPSim simulates the configuration candidate through an interactive system platform. On this platform, a benchmark is conducted by automatically running the target application. VPSim gives, in return, the execution time of the application, including performance statistics such as read accesses, writes, misses, etc. From this step, we get our first objective value T.
(3) The McPAT entries are fed with the system-level parameters from step 1 and the statistics results from step 2. McPAT then quickly returns the area A, dynamic, and leakage powers. The Energy consumption E corresponds to the sum of powers times the execution time of the application. Thus, the three objective values E, T, A are finally obtained for the candidate configuration.

3.3 Exploration strategy: HyperMapper's search engine

In A-DECA [24], the exploration is based on a genetic algorithm. Although genetic algorithms have been proven to be as efficient as reinforcement learning algorithms, which have made incredible advances in recent years, it often requires a tremendous number of simulations to converge towards efficient design solutions. That's led us to focus on Bayesian Optimization as the most appropriate sample-efficient exploration strategy.

Algorithm 1 HyperMapper's Bayesian Optimization

Input: D design space, X initial designs set, N simulation budget, $\mathcal{R}$ generic random forest, aq acquisition function

Simulate $\forall x \in X$ and get their PPA_x estimations;
Update the Pareto front P;
Initialize $\mathcal{R}_T, \mathcal{R}_P, \mathcal{R}_A$ for each objective by fitting (training) with all couples (x, PPA_x);
while $n \leq N$ **do**
  $x_{new} \leftarrow \text{argmax}_x \; aq$ (aggregation $(\mathcal{R}_T, \mathcal{R}_P, \mathcal{R}_A)$);
  if x_{new} is a valid design **then**
    Simulate x_{new} and $n \leftarrow n + 1$;
    Update the Pareto front P;
    Fit $\mathcal{R}_T, \mathcal{R}_P, \mathcal{R}_A$ with $(x_{new}, PPA_{x_{new}})$;
  else
    Fit $\mathcal{R}_T, \mathcal{R}_P, \mathcal{R}_A$ with x_{new} as invalid;
return P with Pareto-optimal solutions X_P

HyperMapper [15] is a Bayesian Optimization-based open-source library that was specially designed to tackle DSE problems from the computer systems community. Unlike the standard version of Bayesian Optimization, HyperMapper can handle multiobjective optimization with categorical/ordinal variables and design constraints. Its pseudo-code is presented in Algorithm 1. The core idea behind Bayesian Optimization is to map the objective space with already simulated solutions using a specific interpolation function. The latter then provides objective estimation values coupled with uncertainty quantification for unexplored designs.

HyperMapper leverages Random Forest for each objective as a non-linear regressor surrogate to deal with discrete parameters of computer designs. Random Forests use a set of decision trees to output a mean prediction for objective function and its variance value through bootstrap aggregating. Furthermore, the algorithm takes advantage of random forests to predict invalid design candidates. It ensures that only valid design simulations are progressively focused on without enumerating the design space explicitly.

An acquisition function is then used to mathematically define the next design to be explored. Depending on how it is formulated, it characterizes the exploration and exploitation trade-off. Popular acquisition functions [8] involve Expected Improvement, Probability of Improvement, and Upper Confidence Bound. The next design to explore is the one that maximizes it. Since the acquisition function computation is instantaneous, HyperMapper exploits model-free methods to find the maximum. Thus, the strength of Bayesian Optimization lies in transforming an expensive DSE into a cheap prediction problem where simulations are used as an oracle to verify the prediction model.

To extend towards a multiobjective version, the most common approach consists of weighted aggregation. HyperMapper, in particular, uses Augmented Tchebycheff Norm [9] to cover non-convex search space regions for its exploration.

By interfacing HyperMapper with its particular features in our simulation-based design flow, we are able to automatically exhibit Pareto-optimal designs.

4 IMPLEMENTATION AND EXPERIMENTS

The studied architecture in this work focus on ARM Neoverse V1 Reference Design. The processor element contains high-performance ARMv8.4-A cores. It includes various architectural improvements compared to previous ARM architectures with customization possibilities for meeting system performance and area chip requirements. In particular, the cores include two-level private caches with a shared last-level cache from a NoC clusterization.

We focus our experiments on parameters related to the memory hierarchy since it is crucial in the HPC systems. The full design space D explored here is given in Table 1 modulo design constraints. As benchmark applications, we choose: *STREAM*, *DGEMM*, and *WaLBerla* that are commonly used in this field.

Table 1: The explored design space parameters.

Architectural parameters	Candidate values
Number of cores	1, 2, 4, 8, 16, 32, 64
Number of cores per cluster	1, 2, 4, 8
RAM size (GB)	2, 4, 8
L1 Data/Instruction cache size (KB)	8, 16, 32, 64
L2 cache size (KB)	128, 256, 512, 1024
LLC* size (KB)	512, 1024, 2048
L1/L2/LLC line size (B)	16, 32, 64, 128, 256
L1/L2/LLC associativity	1, 2, 4, 8
NoC X/Y dimension	1, 2, 4, 8, 16
Number of memory channels	1, 2, 4, 8, 16, 32

*Last Level Cache (L3)

STREAM [14] is designed to measure the memory bandwidth and latency of computer systems. It consists of four simple vectorized kernels used to evaluate the performance of memory-intensive operations on a given architecture. Thus in HPC, it helps to assess the memory subsystem's efficiency and identify potential bottlenecks in data transfer rates and memory access patterns.

DGEMM [4] is a fundamental linear algebra operation that computes the product of two double-precision matrices, which involves a large number of floating-point operations. Optimizing DGEMM execution is crucial for enhancing the performance of many HPC applications that rely on linear algebra computations.

WaLBerla [5] is an open-source software framework designed for the efficient simulation of fluid dynamics and multi-physics problems on HPC architecture. It is specifically tailored for large-scale simulations that require the utilization of multiple compute nodes in parallel and therefore, well suited for the design of massively-parallel supercomputers. With these three benchmark applications, we are able to investigate the performance of the architecture in each aspect of its diverse possible configurations for HPC.

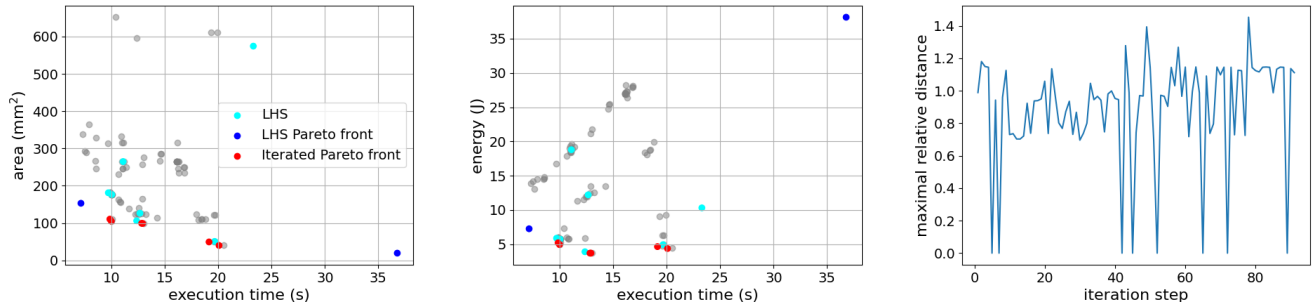
In our experiment, we set HyperMapper with expected improvement acquisition function, 20 initial valid designs from Latin Hypercube Sampling (LHS), and 80 optimization iterations for a total of 100 valid simulations. We spent 9, 49, 174 hours for *STREAM*, *DGEMM*, and *WaLBerla* to run our automated design flow, including simulations and HyperMapper's algorithm. The simulation time of a single application run for *STREAM*, *DGEMM*, and *WaLBerla* ranges from 2 to 15 min, 20 to 50 min and 10 to 200 min respectively. Table 2 gives the Pareto-optimal configuration designs for the three applications.

First, the obtained configurations show a diversity of the Pareto-optimal solutions. With a range from 2 to 64 cores, the designers can choose an optimal configuration according to their metric considerations and design constraints. For instance, the interested designer can select the configuration (6) as the best execution time for *STREAM* or could prefer the configuration (7) for around $\times 2$ less area at the cost of increasing 30% of execution time. Comparing optimal solutions also enables architects to analyze the parameters that have a real impact or inertia on the design performances. For instance, designs (5) and (6) have the same configuration except for a difference in RAM and L1 D/I sizes for similar objectives values. It means that *STREAM* is not RAM and L1 D/I sizes dependent. The results for *WaLBerla* even reinforce this interest. 6 out of 7 optimal configurations have 8 cores, and 4 are nearly identical, with only the number of memory channels differing. Increasing the number of memory channels gives a shorter execution time but with larger energy consumption and area. Despite this observation being obvious since more memory channels enable better parallelization for CPU-RAM communication, it confirms that the automated exploration flow is operational and provides valid solutions. Moreover, it is less apparent that with a decrease in execution time, the energy consumption still grows proportionally, whereas both are correlated. The gain on the execution time compared to the cost of power may not be worth it. In contrast, a higher execution time doesn't necessarily mean increased energy consumption. The configurations (19) and (22) exactly represent it. With an area of 73.98 mm^2, 60.22 mm^2 for the designs (19) and (22) respectively, the light weight of the configuration (22) carries out the gain on energy consumption. The interaction between the objectives is thus very difficult to apprehend, and the automated design flow helps to get insight into it.

Secondly, the leveraged exploration algorithm is another point to consider. Figure 3a, which shows the Pareto front for *STREAM*, presents an interesting pattern of the exploration process. The arrangement of the grey dots on around 600 mm^2 of area level implies that HyperMapper is able to consistently improve one performance

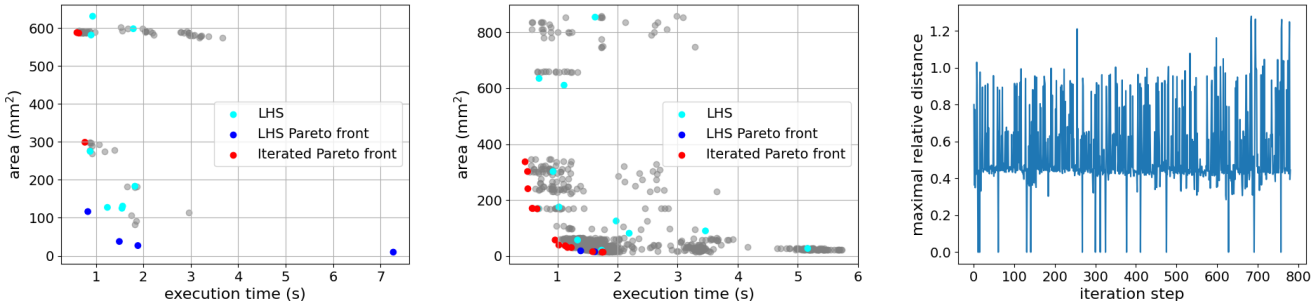
Table 2: The Pareto-optimal solutions automatically generated among explored designs.

	STREAM							DGEMM								WaLBerla						
Config.	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)	(16)	(17)	(18)	(19)	(20)	(21)	(22)
Cores	8	2	16	16	8	8	32	2	32	16	16	32	32	64	64	8	8	8	8	8	8	4
Cores/cluster	1	1	8	2	8	8	8	1	4	4	4	4	4	8	8	2	1	8	8	8	8	1
RAM (GB)	4	8	8	4	4	2	8	4	2	4	4	8	2	2	4	8	4	2	2	2	2	2
L1 D/I (kB)	64	8	64	8	8	32	8	32	16	64	64	8	8	32	64	64	32	64	64	64	64	16
L2 (kB)	256	512	128	1024	128	128	256	128	512	256	256	256	128	128	128	128	512	128	128	128	128	512
L3 (kB)	1024	1024	2048	2048	512	512	1024	512	2048	512	512	512	512	2048	2048	1024	512	1024	1024	1024	1024	2048
Line sizes (B)	32	16	64	128	128	128	256	128	128	32	32	16	16	32	32	128	64	256	256	256	256	256
L1/L2/L3 asso.	1,1,1	2,4,4	8,2,8	4,2,8	1,8,4	1,8,4	2,4,1	1,1,8	1,8,1	8,8,2	4,8,2	1,1,8	1,1,8	8,8,1	8,8,1	2,8,8	4,8,4	2,2,4	2,2,4	2,2,4	2,2,4	2,4,2
NoC X×Y	8x1	1x2	2x1	1x8	1x1	1x1	1x4	2x1	8x1	2x2	2x2	4x2	4x2	1x8	1x8	2x2	8x1	1x1	1x1	1x1	1x1	2x2
Channels	2	4	1	4	32	32	4	8	8	16	16	2	4	4	4	2	2	4	2	8	16	2
Time (s)	0.188	0.726	0.149	0.082	0.063	0.060	0.079	3.678	0.710	1.280	1.289	1.912	2.006	0.995	0.982	17.86	22.48	0.549	0.629	0.510	0.490	0.652
Energy (J)	0.867	4.276	0.877	0.525	2.741	2.845	1.690	38.13	7.248	3.689	3.692	4.639	4.358	4.938	5.200	904.9	564.2	12.40	13.38	12.66	14.74	12.07
Area (mm^2)	26.81	9.513	37.62	116.4	586.7	588.0	299.0	19.48	153.1	99.39	99.03	49.49	40.07	105.2	110.6	35.60	28.58	75.29	73.98	83.74	146.5	60.22



(a) Time-Area Pareto front for *DGEMM* **(b) Time-Energy for *DGEMM*** **(c) Convergence for *DGEMM***

Figure 2: Exploration results for 100 iterations and HyperMapper's algorithm convergence.



(a) Pareto front for 100 iterations **(b) Pareto front for 800 iterations** **(c) Convergence for 800 iterations**

Figure 3: Exploration results for *STREAM* application.

metric without any other metric trade-off. This can't be done without an optimization algorithm. Compared to A-DECA's genetic algorithm results for the population size of 50 with 10 generations, we achieved around 3× larger Pareto-front for 5× less simulations and with bigger design space. Besides, based on analytical formulations and CACTI, A-DECA's model does not account for leakage power, which is known to be an increasing proportion of total power as the technology node progresses. Aside from different energy and area computations between CACTI and McPAT, our configurations (5) and (6) have similar execution times performance in comparison to the solutions among the best ones from A-DECA's exploration algorithm.

As a result, Bayesian Optimization as a model-based technique outperforms genetic algorithms. The surrogate model can fully capture the information of each simulation through active learning and, therefore, surpasses model-free methods when it's about minimizing the number of simulations for the same Pareto-front

performance. The DoE also plays a major part in obtaining Pareto-optimal solutions and disparate information about the design space. In Figures 2a and 3a, 2 to 4 Pareto-optimal solutions are from the initial LHS showing that DoE is certainly as important as exploration iterations. Optimizing both DoE and exploration algorithm is key to reaching neat designs.

To push further the analysis, we plotted the Pareto front and the maximal relative Euclidean distance between the current iteration objective values and the Pareto front (0.0 distance means it is a Pareto-optimal solution) in Figures 3b and 3c for 800 optimization iterations. We notice, this time, a better space coverage of explored solutions spread in different areas of the objective space. Again, compared to 100 individual population with 20 generations of A-DECA's exploration results, we obtained a similar Pareto-front for around 2× less simulations. In addition, the capability of HyperMapper to consistently exhibits new Pareto-optimal solutions during the exploration process is illustrated in figure 3c. We observe that even at the end of the exploration algorithm, between the 600th and 800th iteration, Pareto-optimal solutions can still be found, meaning that designers can even obtain more Pareto-optimal designs at the cost of a higher simulation budget.

5 CONCLUSION AND FUTURE WORKS

Design Space Exploration techniques are fundamental to striking relevant system designs in the early stages. This work used VPSim and McPAT as PPA estimation simulators coupled with HyperMapper, a Bayesian Optimization tool for sample-efficient automated design framework to specifically target HPC ARM cores-based processor designs.

We benchmarked several well-known HPC applications (*STREAM*, *DGEMM*, *WaLBerla*) for large MPSoC design space and achieved diverse Pareto-optimal system-level architectures with only a hundred simulations, enabling large design choices and deep analysis of impactful parameters.

Moreover, we confirmed the efficiency of Bayesian Optimization by comparing it with the Genetic Algorithm of the previous framework A-DECA and highlighted the conclusion that model-based outperforms model-free techniques in expensive DSE. Bayesian Optimization is finally added in A-DECA framework for an enhanced version of it.

In the future, we plan to extend our work to a larger design space and use the results of this work as a baseline. To push further, we will propose our end-to-end optimized exploration algorithm based on features from Machine Learning and Operational Research techniques.

ACKNOWLEDGMENTS

The authors would like to thank Ayoub Mouhagir and Mohamed Benazouz from the French Alternative Energies and Atomic Energy Commission Technology Research Division CEA-List Institute for their valuable support in this work and in particular, on VPSim.

REFERENCES

[1] Chen Bai, Qi Sun, Jianwang Zhai, Yuzhe Ma, Bei Yu, and Martin D.F. Wong. 2021. BOOM-Explorer: RISC-V BOOM Microarchitecture Design Space Exploration Framework. In *2021 IEEE/ACM International Conference On Computer Aided Design.*

[2] Giovanni Beltrame, Luca Fossati, and Donatella Sciuto. 2010. Decision-Theoretic Design Space Exploration of Multiprocessor Platforms. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.*

[3] Amir Charif, Gabriel Busnot, Rania Mameesh, Tanguy Sassolas, and Nicolas Ventroux. 2019. Fast Virtual Prototyping for Embedded Computing Systems Design and Exploration. In *Proceedings of the Rapid Simulation and Performance Evaluation: Methods and Tools.*

[4] J. J. Dongarra, Jeremy Du Croz, Sven Hammarling, and I. S. Duff. 1990. A set of level 3 basic linear algebra subprograms. *ACM Trans. Math. Software.*

[5] C. Feichtinger, S. Donath, H. Köstler, J. Götz, and U. Rüde. 2011. WaLBerla: HPC software design for computational engineering simulations. *Journal of Computational Science.*

[6] T. Givargis, F. Vahid, and J. Henkel. 2001. System-level exploration for Pareto-optimal configurations in parameterized systems-on-a-chip. In *2001 IEEE/ACM International Conference on Computer Aided Design. IEEE/ACM Digest of Technical Papers.*

[7] Constantino Gómez, Francesc Martınez, Adrià Armejach, Miquel Moretó, Filippo Mantovani, and Marc Casas. 2019. Design Space Exploration of Next-Generation HPC Machines. In *2019 IEEE International Parallel and Distributed Processing Symposium.*

[8] Ping Jiang, Qi Zhou, and Xinyu Shao. 2020. Surrogate-Model-Based Design and Optimization. In *Surrogate Model-Based Engineering Design and Optimization.*

[9] J. Knowles. 2006. ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation.*

[10] Srivatsan Krishnan, Natasha Jaques, Shayegan Omidshafiei, Dan Zhang, Izzeddin Gur, Vijay Janapa Reddi, and Aleksandra Faust. 2022. Multi-Agent Reinforcement Learning for Microprocessor Design Space Exploration.

[11] Huizhen Kuang, Su Zheng, and Lingli Wang. 2022. Automated Design Space Exploration of Coarse-Grained Reconfigurable Architecture via Bayesian Optimization. In *2022 IEEE 16th International Conference on Solid-State & Integrated Circuit Technology.*

[12] Jian Li and José F. Martínez. 2005. Power-performance considerations of parallel computing on chip multiprocessors. *ACM Transactions on Architecture and Code Optimization.*

[13] Sheng Li, Jung Ho Ahn, Richard D. Strong, Jay B. Brockman, Dean M. Tullsen, and Norman P. Jouppi. 2009. McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture.*

[14] John McCalpin. 1995. Memory bandwidth and machine balance in high performance computers. *IEEE Technical Committee on Computer Architecture Newsletter.*

[15] Luigi Nardi, David Koeplinger, and Kunle Olukotun. 2019. Practical Design Space Exploration.

[16] Gianluca Palermo, Cristina Silvano, and Vittorio Zaccaria. 2008. Discrete Particle Swarm Optimization for Multi-objective Design Space Exploration. In *2008 11th EUROMICRO Conference on Digital System Design Architectures, Methods and Tools.*

[17] Maurizio Palesi and Tony Givargis. 2002. Multi-objective design space exploration using genetic algorithms. In *Proceedings of the Tenth International Symposium on Hardware/Software Codesign.*

[18] Jacopo Panerati, Donatella Sciuto, and Giovanni Beltrame. 2017. Optimization Strategies in Design Space Exploration. In *Handbook of Hardware/Software Codesign.*

[19] Roxana Rusitoru. 2015. ARMv8 micro-architectural design space exploration for high performance computing using fractional factorial. In *Proceedings of the 6th International Workshop on Performance Modeling, Benchmarking, and Simulation of High Performance Computing Systems.*

[20] Thomas J. Santner, Brian J. Williams, and William I. Notz. 2003. Space-Filling Designs for Computer Experiments. In *The Design and Analysis of Computer Experiments.*

[21] Benjamin Carrion Schafer, Takashi Takenaka, and Kazutoshi Wakabayashi. 2009. Adaptive Simulated Annealer for high level synthesis design space exploration. In *2009 International Symposium on VLSI Design, Automation and Test.*

[22] T. Wolf and M.A. Franklin. 2006. Performance models for network processor design. *IEEE Transactions on Parallel and Distributed Systems.*

[23] Lilia Zaourar, Mohamed Benazouz, Ayoub Mouhagir, Fatma Jebali, Tanguy Sassolas, Jean-Christophe Weill, Carlos Falquez, Nam Ho, Dirk Pleiter, Antoni Portero, Estela Suarez, Polydoros Petrakis, Vassilis Papaefstathiou, Manolis Marazakis, Milan Radulovic, Francesc Martinez, Adrià Armejach, Marc Casas, Alejandro Nocua, and Romain Dolbeau. 2021. Multilevel simulation-based co-design of next generation HPC microprocessors. In *2021 International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems.*

[24] Lilia Zaourar, Alice Chillet, and Jean-Marc Philippe. 2023. A-DECA : an Automated Design space Exploration approach for Computing Architectures to develop efficient high-performance many-core processors. In *26th Euromicro Conference on Digital System Design.*