A-DECA : an Automated Design space Exploration approach for Computing Architectures to develop efficient high-performance many core processors

Lilia Zaourar, Alice Chillet, Jean-Marc Philippe

A-DECA : an Automated Design space Exploration approach for Computing Architectures to develop efficient high-performance many-core processors

Lilia Zaourar, Alice Chillet, Jean-Marc Philippe
Université Paris-Saclay, CEA-List,F-91120, Palaiseau, France
lilia.zaourar, alice.chillet, jean-marc.philippe@cea.fr

***Abstract*—High-performance many-core processors have complex computing architectures with many design parameters related to different levels (CPU macro/micro-architecture, interconnect, memory, specific accelerators, etc.). Design Space Exploration (DSE) is key to tackle the challenges related to the design of such processors, especially in the early stages. This work introduces A-DECA, a highly modular DSE approach for automating the exploration of design parameters. A-DECA combines simulators, models, and exploration strategies to derive relevant objective estimations while preserving a reasonable execution time. Thus, it provides a full methodology enabling the exploration of the design space in an easy-to-use, automatic, and effective way. A-DECA is evaluated in the context of next-generation HPC processors with various applications. We combine simulation tools and analytical formulations to assess PPA (Performance, Power, and Area). Based on an efficient implementation of a multi-objective genetic algorithm for the exploration strategy, current results show a great reduction of design space optimization by around 30% compared to the initial population. A-DECA optimizes the objectives and automatically returns a set of configurations with different characteristics allowing the architect to choose the best design according to the application context.**

***Index Terms*—Automated Design Space Exploration, multi-objective optimization, High Performance Computing.**

I. Introduction

Nowadays, the growing worldwide demand for efficient data processing is driving innovations in High-Performance Computing (HPC). HPC systems are required to meet the high computational needs required for data analysis, weather forecasting, or scientific, economic applications for example. These systems also need to be energy-efficient in order to lower cooling requirements and overall power consumption.

Typically, these systems are very complex since they comprise several tens of thousands of parallel processors interconnected with high-speed and low-latency links. Consequently, designing these systems and their components is a challenging process in which complex architectures with many design parameters related to different levels (CPU macro/micro-architecture, interconnect, memory, accelerators, etc.) need to be chosen and finally tuned. In addition, heterogeneity (various cores, accelerators) is a key element enabling energy-efficiency and new technological perspectives such as chiplets and 3D integration lead to an ever growing number of design parameters. Consequently, processor designers are facing an increasingly difficult challenge of ensuring they make at least good choices among all these opportunities.

Therefore, design space exploration (DSE) is crucial to enable various design options to be explored, especially in the early stages of the design process. This early DSE is essential since early design choices strongly influence the success or failure of the resulting device. Moreover, automatic exploration is paramount to push the performance and efficiency of these systems with reduced effort and a fast time to results. In this context, the development of decision tools for the design of such processors is important but raises many challenges, which makes it a very active research field [15].

The growing importance of Software (SW) in every system and the fact that Hardware (HW) and SW need to be adapted to each other to have an efficient system lead to an increasing demand for exploration strategies taking into account several parameters. It is especially true when designing heterogeneous System-on-Chips based on accelerators (GPUs, eFPGAs, etc.) or even processor extensions (rise of RISC-V CPUs). The main purpose is to answer the so-called "what if" questions concerning design decisions and their impact on both functional and extra-functional aspects, as explained in [11].

The process of system-level DSE is usually divided into two parts [13]. The first one consists in evaluating a single design point in the design space using analytical models as well as simulation, usually called system-level design. The second one is the search mechanism to browse the design space systematically. System-level design methodologies typically urge designers to start with modeling and simulating system components and their interactions in the early design stages. The target is to have an estimation of the performances and a validation of the system. This first part is covered through the development and improvement of various models and simulation tools at different levels for improving the accuracy of the evaluation while not degrading too much evaluation time. The proposed work focuses on the second part and proposes a full methodology to automate design space exploration called A-DECA (**A**utomated **D**esign space **E**xploration for **C**omputing **A**rchitectures). It is a modular DSE approach for automating the investigation of design parameters. A-DECA combines several simulators, models, and exploration strategies to derive relevant objective estimations and best configurations while preserving a reasonable execution time. A-DECA is evaluated

2771-2508/23/$31.00 ©2023 IEEE
DOI 10.1109/DSD60849.2023.00108

in the context of next-generation HPC processors, using both CPU- and memory-bound applications. The main contributions of this article include the following:

- a comprehensive flow for early stages DSE,
- an intuitive and effective way for automatic exploration,
- combination of different simulation tools and analytical formulations to fully estimate PPA
- a modular easy-to-use and updatable approach.

The paper is structured as follows. Section II describes an overview of existing DSE approaches, Section III presents the proposed methodology to perform automatic exploration followed by the implementation of the proposed framework in Section IV. The numerical results are discussed in Section V. Finally, Section V concludes and opens on future works.

II. Related work

Several approaches exist in the literature to accelerate the DSE process. They mainly tackle two challenges: the exploration (i.e. the algorithms) and the evaluation of the objectives.

Platune [10] is an exploration framework for the design of SoCs. It simulates the system using application mapping to capture performance and power metrics. Then, it explores some parameters (e.g., processors, memories, interconnect busses, peripherals) to derive the best configurations for these two objectives. Since the simulation is cycle-accurate, the evaluations are consistent; however, the timing is large, and thus exploration is limited. The approach combines heuristics and parameter inter-dependency models which might lead to an exhaustive investigation, not adapted for very complex architectures as we find currently.

The framework NASA [13] is a modular approach that uses various interfaces to easily integrate different system-level simulation tools and combinations of search strategies. However, it does not combine simulation data outcomes to address a broader set of metrics. It has been evaluated for PowerPC exploration with vision application.

Sesame [7] targets heterogeneous embedded multimedia systems. It focuses on multiprocessor mapping problems under multiple objectives by modeling the application and architecture using mathematical representations. The exploration problem is formulated as a nonlinear mixed integer programming and solved with various optimization approaches. This work targets more application mapping rather than optimization of design parameters at early stages.

The MAGELLAN framework [14] was developed to explore the design space of heterogeneous multi-core architectures using Machine Learning algorithms. This work highlights the potential of Machine Learning algorithms to reduce exploration time without relying on simulators to perform the evaluation phase. Their algorithms aim to find the best configuration for given area and power consumption targets. Hence two objectives are modeled as constraints, which is not enough for current design problems.

The ArchExplorer [6] framework is a website that serves as an implementation platform for design space exploration. The user can choose the simulator and upload it to the platform. He can define the parameters on which the optimization is based and the possible value ranges. The platform explores the design space and then proceeds to the configuration for a set of given applications. Then it simulates each configuration and application. The authors present this framework as flexible. However, the objectives are not specified, and exploration techniques not detailed.

In MULTICUBE project [19], the author presents an automatic exploration framework for the design of multi-core processors with Multicube-Scopefor performance estimation while Multicube-Explorer implements the solution search. To improve convergence speed, the authors propose adding a surface response model to eliminate solutions rather than executing simulations on several configurations

This approach is bound to one simulator that is not[A] easily changed and thus is limited to the latter's feedback while only dealing with two objectives (performance and power).

Also, the work in [8] presents an iterative method based on a meta-model for deriving objective values of various MPSoC configurations from low-level simulations. It prevents expensive multiple evaluations but provides less accurate results.

The articles [3], [4], [9] propose a new FADSE framework based on genetic algorithms. This framework takes advantage of a particularity of genetic algorithms to increase convergence speed. As the population is never completely changed, some configurations are evaluated several times. The results of the simulations are saved in a database allowing the reuse of the results without having to run a new simulation. Moreover, the simulations of a population can be performed in parallel. This study restricted evaluation from a single simulator. Even if this one can change (contrary to other approaches), there is no combination of multiple simulation outputs.

In [18], authors outline the issues involved in DSE, focusing on simulation tools and the flow around them to be as generic as possible and cover large evaluation needs. However, no automatic exploration solutions with optimization algorithms are provided.

In [1], the authors present Boom Explorer, an automatic framework to explore micro-architecture designs for RISC-V Berkeley Out-of-Order Machine (BOOM) to find a good trade-off between power and performance. First, micro-architecture-aware active learning (MicroAL) algorithms generate a diverse and representative initial design set. Then, the Gaussian process model with deep kernel learning functions (DKL-GP) is built to characterize the design space. Moreover, correlated multi-objective Bayesian optimization is leveraged to explore Pareto-optimal designs. Even if the global strategy and vision are effective, their results are within limited application in the many-core context. In addition, various design choices have been already fixed before the exploration.

In [22], the authors present an exploration methodology based on multi-level HW/SW co-design. This exploration is performed using three simulators (runtime trace, architecture, and micro-architecture levels) to refine the exploration parameters. In this flow, the exploration is performed by hand, especially regarding the different configurations, the simulator

executions, and the exploitation of the results. Therefore, no automatic optimization algorithm allows the generation of the best configurations following the different simulations, which may be time consuming.

As written in [11], there is a growing need for efficient and salable DSE approach despite several challenges to be solved.

Thus, we aim here to leverage works in [9], [13], [22] to tackle DSE of future many-core processors for HPC systems with the proposed A-DECA framework. It is a fully automatic approach that does not require the regular intervention of the architect to fix some choices before the exploration and then launch scripts for the exploration. It quickly adapts the various possible configurations to assist the architect in choosing the initial parameters as early as possible in the design process while being fast to perform several iterations and test different configurations rapidly. By combining analytical formulas with several simulators to get the most out of their evolution in evaluations, A-DECA provides good accuracy without necessarily requiring high precision, especially at the beginning of the flow. Next section details the proposed A-DECA flow.

III. Overview of the A-DECA design methodology

Two main challenges have to be addressed in the process of tuning architecture parameters for new complex high-performance computing architectures. The first one is to enable fast and precise automatic exploration of the design space and the second one is to satisfy multiple and possibly conflicting design metrics. Methodologies targeting one or two objectives such as performance and power consumption are no longer sufficient to meet the complexity of modern design flows.

The proposed A-DECA methodology has been developed to automatically solve the problem of designing and optimizing a complex computing architecture by improving multiple criteria. This is done by taking into account coming from various sources. Figure 1 presents the overall A-DECA methodology. The main concept of A-DECA is clearly to separate the problems and thus the flow, into different domains, each with appropriate representations and tools to solve the domain's specific sub-problem. Therefore, A-DECA proposes three different functional blocks allowing to perform the exploration in an automatic and optimized way. Each of these blocks 1) translates one representation domain (e.g. problem description, mathematical modeling, simulation modeling, etc.) into another one and 2) controls the process (e.g. the tools) for solving the particular sub-problem in the overall flow 3) generates HW configurations. These blocks are in blue in Figure 1 and detailed below.

The set of inputs of A-DECA is a template of the computational architecture (e.g. the possible processing cores, their various components, specific accelerators, etc.), the architecture parameters, but also the objectives (e.g. maximize performance, minimize power consumption and/or area, etc.) and the constraints (e.g. memory size, interconnect, etc.).

The first block, called *Modeling the HW problem* allows a translation between the possible input formats and a mathematical model representation to formulate the various constraints and objectives from a mathematical point of view. This functional block thus provides a translation between the internal way of representing the problem from the A-DECA implementation point of view (obtained after parsing the input files) and a mathematical representation adapted to the processing by optimization and exploration algorithms. It therefore separates the architectural vision of the problem from the decision making problem. It enables the use of mathematical algorithms adapted to the problem or different parts of the problem.

The second functional block, called *HW translator* translates the outputs of the mathematical algorithms used for the optimization into a HW representation of the problem, automatically generating the specifications of the various configurations of architectures for the targeted evaluation tools (e.g. simulators). This HW representation allows us to deduce the parameter files to configure the simulation, emulation, or other tools allowing the evaluation. This block also allows launching and/or controlling several tools simultaneously with a single uniform representation (the decision representation coming from the first block). It provides, on the one hand, the independence of the mathematical modeling (adapted to the optimization problem) from the evaluation tools (simulators, etc.). On the other hand, it allows the simultaneous execution of several tools (contrary to other state-of-the-art approaches).

The last block, called *HW simulator & generator* allows to recover the results of the evaluations at the end of the execution of the evaluation tools (simulations, etc.) and to exploit them. Thus, this block makes it possible to combine the results of the evaluation tools at different levels. It allows 1) to drive the functional blocks of translations of the assessment results of some tools to inject them into other tools with multi-level management, 2) to recover the data of the assessment tools to combine them to get back more complex/adapted metrics for exploration. This third block also allows the translation of these evaluation results into a compatible format for optimization algorithm(s) to close the exploration loop.

One key feature of the proposed approach is modularity and genericity. Indeed, the exploration is unrelated to a particular simulator because it is not restricted to a single evaluation tool. Thus, it is fully modular, generic, and therefore not limited to a restricted set of parameters due to the simulator. On the one hand, the three blocks can be integrated into a classical SoC design flow. On the other hand, the methodology is intended to optimize the design and thus allow engineers to quickly create configurations, evaluate them and explore the different combinations. The next section introduces the first implementation of the A-DECA methodology.

IV. Framework implementation

A. Basic notions and definitions

Multi-objective optimization methods are usually applied to get the best solution to a well-defined problem. They are considered as a mathematical process looking for a set of alternatives representing the Pareto optimal solution. Pareto

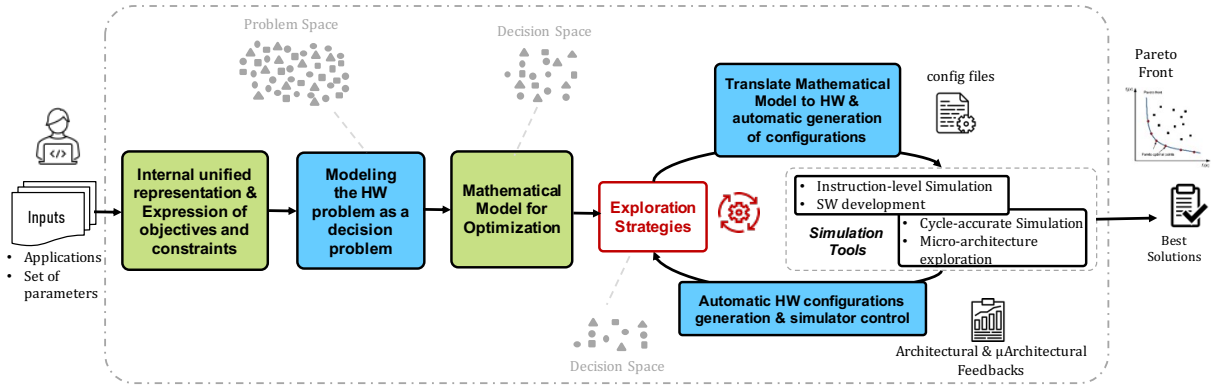


Fig. 1: Illustration of the A-DECA automated DSE methodology.

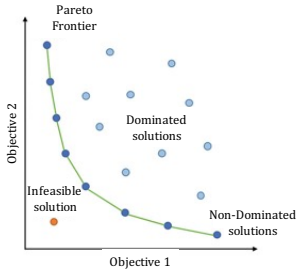


Fig. 2: Multi-objective Pareto frontier with 2 objectives.

optimality is a cornerstone concept in the field of multi-objective optimization. We give here some basic notions.

A multi-objective combinatorial optimization problem can be mathematically defined as follows:

Optimize $F(x) = (f_1(x), f_2(x), ..., f_n(x))$ with $x \in D$, where $(n \geq 2)$ is the number of objectives , $x = (x_1, x_2, ..., x_k)$ is the vector representing the decision variables, D represents the set of feasible solutions and each of the functions $f_i(x)$ is to be optimized (minimized or maximized). In contrast to single-objective optimization, the solution of a multi-objective problem is a set of non-dominated solutions, known as the set of Pareto Optimal solutions.

A feasible solution $x^* \in D$ is Pareto optimal (non-dominated) if and only if there is no solution $x \in D$ such that x dominates x^*. A solution $y = (y_1, y_2, ..., y_k)$ is said to *dominate* a solution $z = (z_1, z_2, ..., z_k)$, in the case of objective minimization, if $\forall i \in [1...n], fi(y) \leq fi(z)$ and $\exists i \in [1...n]$ such that $f_i(y) < f_i(z)$.

Hence, any solution of the Pareto set can be considered optimal since no improvement can be made on an objective without degrading the relative value of another objective. These solutions form the *Pareto front*, also called the set of *non dominated* solutions.

In the case of a bi-objective problem (two objectives to be minimized, for example), the efficient solutions can be identified visually in the as represented in Figure 2.

The DSE problem to determine the best many-core processors parameters is a multi-objective optimization problem. Let's consider a system with several design parameters $(D_1, D_2, \ldots, D_n)$ under the designer's control. The particular values of the parameters result in specific values of criterion functions $(f_1, f_2, \ldots, f_n)$, which are functions of the inputs and measure the system performance. Each choice of parameter values yields a feasible solution (system design that meets the user constraints) and has a value in the objective space.

We aim to find the best design parameter values (many-core architectures configurations) to optimize all imposed objectives. We studied here the three objective functions: timing performances, energy, and area. Next section details how we compute these evaluations.

B. Objectives evaluation

The proposed approach A-DECA is evaluated using a DSE problem focusing on complex memory hierarchy regarding classical PPA key performance indicators. We compute them here through the three objectives: timing performance, power consumption, and chip area. We combine different simulators and analytical models to derive these objectives. We detailed here the models used with analytical formulas as well as simulator tools.

Timing performance of a system configuration is evaluated by simulating the execution of an application on this configuration. Indeed, an analytical formulation will not reflect the real execution and thus is unsuitable. We choose the *VPSim* Virtual Prototyping Simulator [5] for this purpose. *VPSim* ensures fast simulation speed while keeping models accurate. In addition, it provides many performance counters parameters related to memory hierarchy, NoC, peripherals, and SW [12]. Thus, a large set of design parameters could be explored to design efficient many-core architectures.

Evaluation of **Power consumption** concentrates on the caches and is noted P_{cache}. We combine here *CACTI* [21] simulation results to estimate the cache power per access noted P_{RA} for read and P_{WA} for write depending on cache parameters found during the exploration phase. *VPSim* counts the number of cache access noted Nb_{RA} or Nb_{WA} for each cache level for the application. Consequently, different information can be combined to calculate power as follows :

References	[16]	[16]	Wikichip ARM A57	Extrapolation
Size of technology (nm)	65	32	16	1
Size of the chip (nm^2)	18,5	5	1,66	0,013

TABLE I: Chip Area estimation.

$P_{cache} = \sum_{L1D,L2,L3} Nb_{RA_K} \times P_{RA_K} + Nb_{WA_K} \times P_{WA_K}$

Note that we do not consider here static power consumption that should add a fixed parameter to the previous formulation. Since we will have to minimize this value adding a constant is no more needed.

We can also evaluate the energy (denoted here by E_{cache}) that reflects the execution of an application over a given time. This value must also be minimized to limit, for example, the environmental impact of processing an application on the chosen design, $E_{cache} = P_{cache} \times Time_{Execution}$.

To compute chip **Area**, we evaluate the area of the caches and use the estimation of cache dimensions given by *CACTI* noted A_{cache}. Next, the cache's area is added to the estimated value of the core area, evaluated by a model with a factor size in nm noted F and a theoretical area for a $1nm$ core noted Th_{Acore}. Peripherals and NoC areas are not modeled in this example.

Thus assume here that the area equals $Area = Area_{cores} + Area_{cache}$. The core area is computed as follows:

$A_{cores} = Nb_{cores} \times Th_{Acore1nm} \times F^{size_{nm}} + A_{cache}$.

The area of a cache is directly extracted from *CACTI* simulator. Thus the final area is equal to :

$Area = Nb_{cores} \times 0,013 \times 3,4^{size_{nm}} + A_{cache}$

C. Multi-objective exploration based on Genetic Algorithm

The Genetic Algorithm (GA) is a bio-inspired optimization algorithm. It is widely used for multi-objective optimization exploration. It has been studied in the domain of system-level design [17], and has been demonstrated to yield good results.

The principle of the GA is based on individuals' reproduction and natural selection. Figure 3 shows the custom diagram of a GA. The first step is to create an initial population, i.e. a set of individuals (in our case, design configuration). The explored solutions must be feasible, so a verification step is performed (as represented in red). Afterward, the candidates must be evaluated and then selected. This selection is one of the key points of convergence in the algorithm. We will carry out crossings on this new reduced population to obtain children individuals (new configurations). Some children will be mutated in one or more of their genes before being checked again and integrated into the population.

In our case, an individual (configuration) is represented by a vector: this allows a seamless manipulation of individuals and enables the possibility to change or add parameters easily. Thus, any design parameters can be represented to reflect architectural choices.

V. Numerical Results and analysis

The A-DECA framework was evaluated using a DSE problem focusing on memory hierarchy regarding the three objectives: timing performance, energy consumption and area. The studied many-core systems has up to 16 $ARMV8$ processors. The set of explored parameters are presented in Table II. In addition, various HPC applications has been used as described in the next section.

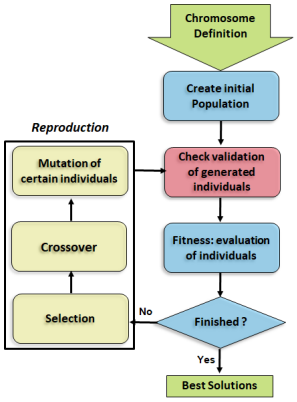


Fig. 3: Costumed multi-objective Genetic Algorithm.

Parameters	Design space
Number of cores	1, 2, 4, 8, 16
Size of L1D (data) cache kB	4, 8, 16, 32, 64
Size of L1I (instruction) cache kB	4, 8, 16, 32, 64
Size of L2 cache kB	128, 256, 512, 1024
Associativity L1 cache	1, 2, 4
Associativity L2 cache	2, 4, 8

TABLE II: Description of design space parameters.

During our experiments, we used three different HPC applications. The first one is *STREAM* which is an application with the primary objective of stressing the memory, and the other two are *Fmm* and *Radiosity* from the *SPLASH* suite.

STREAM aims at measuring the bandwidth of memories, by performing different types of operations such as copies, sums, averages and compositions of multiplication and sums [20]. The *Fmm* application of the SPLASH suite which is widely used in the HPC domain. It handles a problem of size $65,536$ data called particles [2]. The *Radiosity* application computes the equilibrium distribution of light. This is a data-intensive and also time-consuming benchmark.

The A-DECA framework was built to be flexible regarding optimization. Thus, it is possible to choose the number of objectives to optimize and the number of parameters to adjust. In this section, we present the results obtained by optimizing first two objectives: energy consumption and execution time and next with three objectives by adding the area which have to be minimized. We also give GA best parameters.

Indeed, the efficiency of GA relies on the right choice of parameters, such as the number of individuals in the initial population and the number of generations, the proportion of good individuals to keep during the selection, the probability of maintaining an individual among those remaining (the less good ones) and finally the probability that an individual mutates. All these parameters may influence the convergence of

the algorithm. Besides, since the initial population is generated randomly in our case, we have also studied the impact of its distribution. Thus after performing several tests, the GA parameters used here are described in Table III. However, the size of the initial population and the number of generations has been adapted (through intensification and diversification) depending on exploration needs.

Proportion of good individuals retained	0.4
Probability of keeping a less good individual	0.2
Probability of mutate	0.1
Number of generations	10,20,50
Number of individuals	30,50,100

TABLE III: Description of the Genetic Algorithm parameters.

Figure 4 illustrates the results obtained with A-DECA for a population of 50 individuals and 10 generations. The initial population is in blue, and the Pareto front is in red. This figure highlights the advantages of the framework. Indeed, without this automatic exploration, the designer would need to explore the 50 configurations in blue to estimate their fitness. A-DECA reduces the design space to interesting solutions to explore. Only two solutions remain for the designer to consider, which saves time spent on initial analysis.

Config.	Nb of cores (kB)	Size L1.D /L1.I	Size L2 (kB)	Asso. L1 /L2	Energy (nJ)	Time (ns)
1	8	16	1Mo	1 / 2	5503365	61700921
2	8	16	1Mo	1 / 2	4834036	69876511

TABLE IV: Description of the configurations for bi-objectives consumption & timing exploration, with an initial population of 50 individuals, and 10 generations.

Table IV gives the configurations obtained for energy consumption and timing optimization, with an initial population of 50 individuals, and 10 generations. Columns 1 to 6 describe the configurations selected by the framework, and the last columns contain the objective values for each configuration.

Figure 4b shows the bi-objective exploration results for an initial population of 50 configurations and 20 generations. The black dots represent the initial population, the gray dots represent the generations. For example, the darker gray points represent the 4^{th} generation of individuals, the lighter ones represent the last generation. In red, the points belonging to the Pareto front appear, which can be seen in the figure 4c with the three configurations. We can observe that generation after generation; the population converges and minimizes objectives. The three objectives are minimized around 25% or 30% compared to the initial population.

Moreover, the resulting configurations appear interesting. Indeed, thanks to this result, the architect will be able to easily make a decision according to the application context or the given constraints. For instance, in this case, the reasonable choice seems to be the configuration 2 with 16 cores, 8 KB of $L1$ instruction and data cache, 1 MB of $L2$ cache, an associativity of 2 for the $L1$ cache and 4 for the $L2$ cache. Indeed, the gain in energy compared to the configuration 1 is noticeable for a very slight time increase and similarly, the time gain over the configuration 3 is significant.

These figures allow us to conclude on the interest of this approach and the convergence of this one. On the average, the energy consumption of the initial population is 7621034 nJ against 5583864 nJ for the Pareto front. The gain is thus 26%. Similarly, the average temporal performance of the initial population is 154444960 ns against 71704660 ns for the Pareto front. This is a gain of 53%. Thus, these graphs clearly show the interest of the framework and its ability to optimize the objectives. Moreover, the proposed configurations seem interesting. Indeed, thanks to this result, the architect will be able to easily make a decision according to the application context or the given constraints. For example, in this case, the reasonable choice seems to be the orange configuration, 16 cores, 8 KB of $L1$ instruction and data cache, 1 MB of $L2$ cache, an associativity of 2 for the $L1$ cache and 4 for the $L2$ cache. In fact, the gain in energy consumption compared to the blue configuration is notable for a very slight increase in time and similarly, the time gain compared to the green configuration is significant. To complete this part on the bi-objective exploration, the configurations which belong to the Pareto front for two different executions under the same conditions (same initial population and same configuration of the framework parameters) are reported in Table V, along with their corresponding objectives.

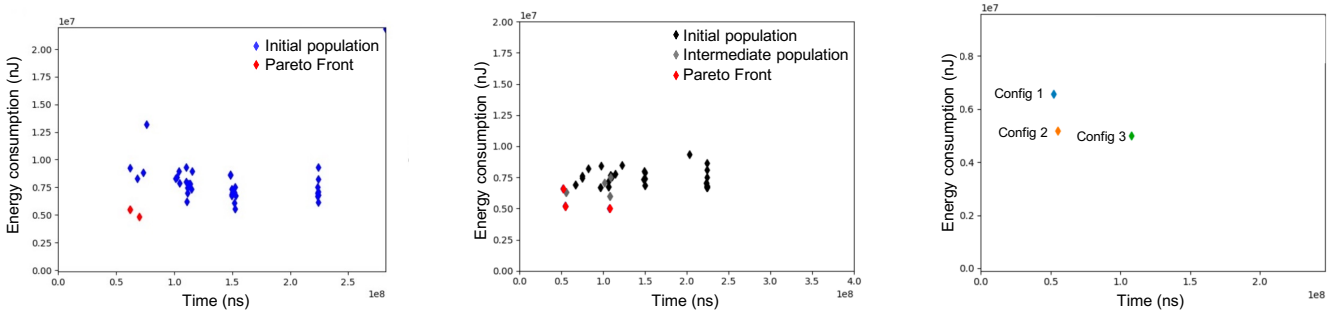
Nb of cores	Size L1.D (ko)	Size L1.I (ko)	Size L2 (ko)	Asso. L1/L2	Energy (nJ)	Time (ns)
First execution						
16	8	8	1Mo	2 / 4	5185089	54936198
16	4	8	1Mo	1 / 4	4031200	71313079
Second execution						
8	16	8	1Mo	2 / 4	6574809	52300955
16	8	8	1Mo	2 / 4	5185089	54936198
16	8	8	1Mo	1 / 4	4991696	107876829

TABLE V: Solutions achieved for a bi-objective exploration: consumption & time, for the same configuration.

Thus, for two executions of the GA with the same parameters and the same initial population, the solutions obtained are different due to the random nature of the crossovers. Indeed, the choice of the individuals to cross is random, but the way of crossing is not, and this is what leads to that the configurations are quite close to each other, while being various.

The flexibility of the framework enabled us to easily add a third objective to optimize. Thus, we chose the area and present here optimization results with three objectives: timing performance, power consumption which is converted into energy and area.

Table VI shows configurations obtained with A-DECA for a multi-objective exploration, with different configurations and populations. Columns 1 to 6 describe the configurations selected by the framework, and the last columns contain the objective values for each configuration. For the first set, the first execution, it is interesting to notice that the single-core



(a) Initial population: 50 individuals and 10 generations.

(b) Initial population: 50 individuals and 20 generations.

(c) initial population: 50 individuals and 20 generations.

Fig. 4: Exploration result for bi-objectives optimisation: Energy consumption and timing.

configurations consume less, but are also less fast. The fastest configurations are those with 4 cores, there is no other solution with more cores proposed here by the framework. Moreover, for the same $L1$ cache configuration and the same number of cores, it seems more interesting in terms of consumption to choose an $L2$ cache of 1 MB rather than 512 KB and an associativity of the $L1$ and $L2$ caches of 2 and 4 respectively instead of 1 and 8.

For the second set, we perform an optimization on the same initial population with a different framework configuration. Here again, the solutions highlight different points.

Considering only consumption and time, the most interesting configurations are those with many cores and a large $L2$ cache. However, these are also the most expensive configurations in terms of area. Therefore, 4 core configurations with 256 KB or 1 MB of $L2$ cache, 8 KB of $L1.D$ cache, and 4 KB of $L1.I$ cache seem better if we want to minimize the set of objectives without favoring one. This is a benefit of this framework, which gives a Pareto front as an output. It allows the architect to decide which configuration among those proposed is the most suitable according to the context. Notice that in this part, a configuration with 16 cores appears in the Pareto front. This can be due to random choices or a mutation making new solutions appear. The third set does not bring any additional information. However, it is possible to compare the results with those of the first set. The modification of initial population allowed the algorithm to explore other configurations with higher associativity values of 4 and 8 or 4 and 4. The consumption of the configurations is globally higher than for the first population. Finally, Figure 5 illustrates Configuration 1 with 100 individuals and 20 generations .

Regarding A-DECA execution time, our process is fast since a database is created and consulted to minimize the number of simulations launched. Each time a new configuration is generated, the database is updated with the values of the three objective functions to avoid unnecessary simulations. This mechanism allows us to speed up the exploration. In addition, a good compromise between the intensification and diversification operations of the search (the size of the initial population and the generation name) is implemented.

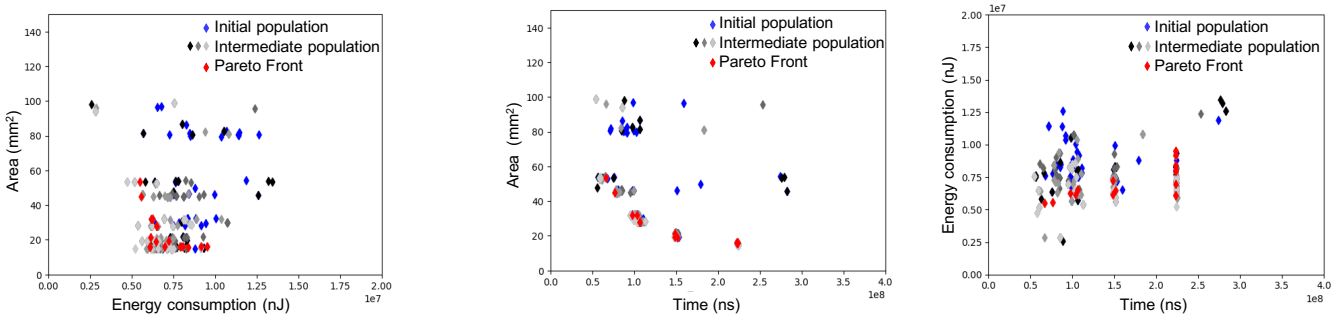
Nb of cores	Size L1.D (ko)	Size L1.I (ko)	Size L2 (ko)	Asso. L1/L2	Energy (nJ)	Area (mm²)	Time (ns)
3 objectives, configuration 1 population 1							
4	8	4	1Mo	2 / 4	7503721	32.07	101671585
2	32	16	1Mo	2 / 4	7541173	21.38	148245725
2	32	16	512	1 / 8	7936920	20.07	149850398
1	4	8	1Mo	2 / 4	7107366	15.94	223392147
1	4	8	512	1 / 8	7230306	15.28	223942955
3 objectives, configuration 2 population 1							
16	8	8	1Mo	2 / 4	5324934	96.69	70470207
4	8	4	256	1 / 4	6453200	28.07	108118854
2	32	8	256	1 / 4	6738560	19.37	149832524
2	32	8	128	4 / 2	7179795	19.16	146628182
4	8	4	1Mo	2 / 4	7503721	32.07	101671585
16	8	8	256	1 / 4	8077873	80.68	87641542
4	8	4	128	4 / 8	8204671	28.02	103547531
4	8	4	1Mo	4 / 2	8407218	32.03	97911504
3 objectives, configuration 1 population 2							
4	64	8	1Mo	4 / 8	10446521	32.12	99850234
4	64	8	256	4 / 4	9397644	28.68	106946821
1	32	8	1Mo	4 / 8	8570587	15.87	223172226
1	8	4	1Mo	4 / 8	8339735	15.81	223238938
1	32	8	256	4 / 4	7574778	15.01	224298406

TABLE VI: A-DECA solutions for a multi-objective exploration, with different configurations & populations.

As an example, for the *STREAM* application design explorations the average time is 2.25 min while it is 30 min for the *Radiosity*.

VI. Conclusion and Future Works

Design Space Exploration (DSE) is key to tackle the challenges related to the design of complex HPC processors, especially in the early stages. This work introduces A-DECA (Automated Design space Exploration for Computing Architectures), a modular approach for automating the exploration of design parameters. A-DECA combines several simulators, models, and exploration strategies to derive relevant objective estimations while preserving a reasonable execution time. It can quickly adapt to different possible architectural needs to assist in the initial choices as early as possible in the design process. Thanks to its fast execution, the designer can quickly perform several iterations and test many configurations. Moreover, it provides relevant feedback to evaluate tendencies



(a) Area Energy consumption. (b) Area timing. (c) Energy consumption Time.

Fig. 5: Numerical Results for three objectives 100 individuals and 20 generations.

between the different designs explored (without necessarily requiring a high level of precision since it is at the beginning of the flow).

We implemented a multi-objective Genetic Algorithm using classical HPC applications to explore many-core architectures concentrating on the number of cores and memory hierarchy tuning parameters. The evaluation relies on combinations of simulators (VPSim and CACTI) and models to compute the objectives (timing performance, energy consumption, and area). We use an evaluation database to increase the speed of our algorithm.A-DECA optimizes the objectives and returns a set of configurations with different characteristics allowing the architect to choose the best design according to the application context. In future works, we plan to propose other optimization strategies, refine objective function evaluations, use more accurate simulators to explore the accuracy versus speed tradeoff, and perform tests on larger architectural design parameters and heterogeneous cores.

Acknowledgment

The authors would like to thank Mohamed Benazouz and Ayoub Mouhagir for their valuable help with this work, especially regarding the simulation tools.

References

[1] Chen Bai, Qi Sun, Jianwang Zhai, et al. Boom-explorer: Risc-v boom microarchitecture design space exploration framework. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*.

[2] Christian Bienia, Sanjeev Kumar, and Kai Li. Parsec vs. splash-2: A quantitative comparison of two multithreaded benchmark suites on chip-multiprocessors. In *2008 IEEE International Symposium on Workload Characterization*, 2008.

[3] Horia Calborean, Ralf Jahr, et al. Optimizing a superscalar system using multi-objective design space exploration. In *Proceedings of the 18th International Conference on Control Systems and Computer Science (CSCS)*, 2011.

[4] Horia Calborean and Lucian Vinţan. An automatic design space exploration framework for multicore architecture optimizations. In *9th RoEduNet IEEE International Conference*, 2010.

[5] Amir Charif, Gabriel Busnot, Rania Mameesh, et al. Fast virtual prototyping for embedded computing systems design and exploration. In *Proceedings of the Rapid Simulation and Performance Evaluation: Methods and Tools*. 2019.

[6] Veerle Desmet, Sylvain Girbal, et al. Archexplorer for automatic design space exploration. *IEEE micro*, 2010.

[7] Cagkan Erbas, Cerav-Erbas, et al. Multiobjective optimization and evolutionary algorithms for the application mapping problem in multiprocessor system-on-chip design. *IEEE Transactions on Evolutionary Computation*, 2006.

[8] G. Mariani et. al. A correlation-based design space exploration methodology for multi-processor systems-on-chip. In *Proceedings of the 47th Design Automation Conference*, 2010.

[9] Arpad Gellert, Horia Calborean, et al. Multi-objective optimisations for a superscalar architecture with selective value prediction. *IET computers & digital techniques*, 2012.

[10] Tony Givargis and Frank Vahid. Platune: A tuning framework for system-on-a-chip platforms. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2002.

[11] Marius Herget, Faezeh Sadat Saadatmand, Martin Bor, et al. Design space exploration for distributed cyber-physical systems: State-of-the-art, challenges, and directions. In *2022 25th IEEE Euromicro Conference on Digital System Design (DSD)*.

[12] Fatma Jebali, Oumaima Matoussi, Arief Wicaksana, Amir Charif, and Lilia Zaourar. Decoupling processor and memory hierarchy simulators for efficient design space exploration. In *System Engineering for constrained embedded systems*, pages 47–52. 2022.

[13] Zai Jian Jia, Andy D Pimentel, et al. Nasa: A generic infrastructure for system-level mp-soc design space exploration. In *2010 8th IEEE Workshop on Embedded Systems for Real-Time Multimedia*.

[14] Sukhun Kang and Rakesh Kumar. Magellan: a search and machine learning-based framework for fast multi-core design space exploration and optimization. In *2008 Design, Automation and Test in Europe*, pages 1432–1437. IEEE, 2008.

[15] Bart Coppens Christian Gamrat Madeleine Gray *et al.* Marc Duranton, Koen De Bosschere. *HiPEAC Vision 2023: High Performance Embedded Architecture And Compilation*. 2023.

[16] Taecheol Oh, Hyunjin Lee, et al. An analytical model to study optimal area breakdown between cores and caches in a chip multiprocessor. In *2009 IEEE Computer Society Annual Symposium on VLSI*.

[17] Jacopo Panerati, Donatella Sciuto, and Giovanni Beltrame. Optimization strategies in design space exploration. In *Handbook of Hardware/Software Codesign*. Springer, 2017.

[18] et.al. R. Giorgi. A design space exploration tool set for future 1k-core high-performance computers. In *Proceedings of the Rapid Simulation and Performance Evaluation: Methods and Tools*. 2019.

[19] Cristina Silvano, William Fornaciari, Gianluca Palermo, et al. Multicube: Multi-objective design space exploration of multi-core architectures. In *VLSI 2010 Annual Symposium*, pages 47–63. Springer, 2011.

[20] Joël Wanza Weloli, Sébastien Bilavarn, et al. Efficiency modeling and analysis of 64-bit arm clusters for hpc. In *2016 Euromicro Conference on Digital System Design (DSD)*. IEEE, 2016.

[21] Steven JE Wilton and Norman P Jouppi. Cacti: An enhanced cache access and cycle time model. *IEEE Journal of solid-state circuits*, 1996.

[22] Lilia Zaourar, Mohamed Benazouz, Ayoub Mouhagir, et al. Multi-level simulation-based co-design of next generation hpc microprocessors. In *2021 IEEE International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS)*.