Information Sciences 476 (2019) 375–391

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Performance and energy optimisation in CPUs through fuzzy knowledge representation

Arpad Gellert[a], Adrian Florea[a,*], Ugo Fiore[b], Paolo Zanetti[b], Lucian Vintan[a]

[a] *Computer Science and Electrical Engineering Department 'Lucian Blaga', University of Sibiu, Romania*
[b] *Department of Management Studies and Quantitative Methods, Parthenope University, Italy*

A R T I C L E I N F O

Article history:
Received 20 November 2017
Revised 3 March 2018
Accepted 11 March 2018
Available online 12 March 2018

Keywords:
Automatic design space exploration
Processor design knowledge
Superscalar microarchitecture
Dynamic value prediction
Energy saving

A B S T R A C T

This paper presents an automatic design space exploration using processor design knowledge for the multi-objective optimisation of a superscalar microarchitecture enhanced with selective load value prediction (SLVP). We introduced new important SLVP parameters and determined their influence regarding performance, energy consumption, and thermal dissipation. We significantly enlarged initial processor design knowledge expressed through fuzzy rules and we analysed its role in the process of automatic design space exploration. The proposed fuzzy rules improve the diversity and quality of solutions, and the convergence speed of the design space exploration process. Experiments show that a set-associative prediction table is more effective than a direct mapped table and that 86% of the configurations in the Pareto front use multiple values per load. In conclusion, our experiments show that integrating an SLVP module into a superscalar microarchitecture is hardware feasible; in comparison with the case without SLVP, performance is better, energy consumption is lower, and the temperatures inside the chip decreases, remaining below 75 °C.

1. Introduction

The main goal of this article is the optimisation of processor architecture, with the aim of containing energy consumption and improving performance, as well as the quantification and control of thermal dissipation in the obtained Pareto-optimal configurations. To this end, we propose an improvement in the selective load value prediction (SLVP) microarchitecture [15] analysing processor design knowledge expressed through fuzzy logic rules (PDK-FLR) over the design space exploration (DSE) process from both the viewpoints of solution quality and convergence speed.

The SLVP unit is a hardware structure which exploits the locality property of load values by anticipating the next output of a certain load instruction based on its previous outputs. The prediction process is applied selectively, only on long-latency loads, resulting in a simpler hardware structure, fewer SLVP accesses, and better performance. In this work, the SLVP unit has been extended with data memory address-based access, set-associative organisation, multiple load values, as well as the possibility to increase the selectivity level by accessing the SLVP solely when a miss in the level 2 (L2) cache occurs. The study in [14] covered 19 architectural parameters with $2.5 \cdot 10^{15}$ possible configurations. As the number of parameters considered in this work is 23, potential configurations grow to more than 121 millions of billions, thus requiring the use of a

[*] Corresponding author.
E-mail address: adrian.florea@ulbsibiu.ro (A. Florea).

https://doi.org/10.1016/j.ins.2018.03.029
0020-0255/© 2018 Elsevier Inc. All rights reserved.

heuristic search. The automatic DSE will be generated with the Framework for Automatic Design Space Exploration (FADSE) presented in [5], applying the NSGA-II multi-objective genetic algorithm [10]. We significantly improve and enlarge initial design knowledge, represented inclusively through fuzzy logic rules, during the optimisation process of the target microarchitecture for faster convergence and better solution diversity and quality. We will analyse how processor design knowledge expressed by experts through fuzzy logic rules might be formally represented and we will evaluate how it could improve the effectiveness of multi-objective DSE algorithms. These improvements lead to a new, original optimisation methodology of micro-architectural design.

We optimise the target microarchitecture with respect to performance, energy, and temperature. Thermal evaluation and control is an essential feature in modern central processing unit (CPU) design. All power consumption is finally dissipated as heat and, in large data centres, heating ventilation and air conditioning (HVAC) systems absorb almost the same amount of energy as the computing resources themselves [34]. Considering that a non-negligible fraction of the worldwide energy production is attributable to information technology equipment (ITE), and that the corresponding greenhouse gases (GHG) emissions are expected to grow relatively faster than in other sectors, energy-efficiency and thermal performance of CPUs are fundamental factors in the quest for a reduction of the global energy footprint, and consequently GHG emissions.

CPU performance can usually be improved by increasing clock frequencies, enlarging and refining hardware resources, or adding new hardware resources. Over the last years, this has however resulted in increased power consumption of servers, stretching the limits of power supply and cooling equipment. In high-performance computing (HPC) systems, when extremely expensive computation is involved, e.g. in simulations for engineering, biochemistry, or finance, thirst for speed has been coupled with consumption of egregious amounts of power.[1] For such supercomputers, ultimate performance depends on complex optimisations involving the manner in which cores interact with memory and their communication paradigms [9]. The effect of architectural evolution on HPC can be appreciated by observing the TOP500[2] list. Whilst the average power consumption of the top 10 (resp., 50, and 500) systems in the list is 1.32 (resp., 0.908, and 0.257) MW, the average power efficiency scores at 248 (resp., 193, and 122) MFLOPS/W. Therefore, the most powerful supercomputers (which also happen to be the most recent) are more energy efficient than their predecessors. In synthesis, although the net effect of an architectural improvement such as the one proposed here on the energy consumed by an HPC system depends on the microarchitecture of its CPUs, even a modest reduction in power requirements will account for significant savings.

Given the conspicuous amounts of money invested in data centres, energy-awareness has also attracted the attention of the security research community. Indeed, recent evolution in denial of service (DoS) attacks contemplates a new facet where, instead of focusing on degrading the performance of a system to make it useless, raiders concentrate on actions whose ultimate purpose is to raise the energy consumption, with potentially devastating effects on the financial side [27]. This is particularly evident with battery-powered devices, where a targeted attack may stealthily deplete the battery, finally leading to service failure (see, e.g. [13]). It should be kept in mind that, as far as next-generation mobile devices are concerned, the requirements for secure communications with the involved computation-intensive cryptographic operations will affect the energy needs of the mobile SoC [6]. In large-scale data centre infrastructures, complex contract structures for energy supply exacerbate the amplitude of this type of attack, because if a threshold billing system (a higher tariff is applied when consumption exceeds a given threshold) is in place, even a small increase in consumption may result in a significant amount of monetary loss [28]. In designing new microarchitectures, then, all the above aspects should be considered.

This paper is structured as follows. Section 2 reviews the existing value prediction based speculative microarchitectures and the DSE concept. Section 3 describes the target SLVP based superscalar microarchitecture. Section 4 presents our developed method to improve the effectiveness of multi-objective DSE algorithms with processor design knowledge expressed through fuzzy logic rules and other restrictions. Section 5 details the simulation methodology and presents the quality metrics applied in this work. Section 6 discusses the simulation results generated on the Alpha AXP 21264 microarchitecture improved with SLVP capabilities. Finally, Section 7 highlights the main contributions and proposes some possible further work.

2. Related work

2.1. Load value prediction

Load value prediction is a data-speculative micro-architectural technique introduced in [23] exploiting value locality and the correlation between the load instruction addresses and their actual values. In this work, we apply a selective approach; prediction will only apply to load instructions with a miss in the data cache. Doing so, we reduce mis-prediction costs, decrease hardware costs and significantly improve the performance-energy ratio of the microarchitecture (a lower hardware complexity and performance increase are combined with lower energy consumption). Data-speculative techniques have also been presented in [37]. Superscalar/SMT microarchitectures with a direct mapped SLVP table have already been presented and investigated in [14–16]. Additionally, in this study, we highly parameterise the SLVP table in a more realistic manner by adopting a set-associative organisation by allowing the SLVP table to be accessed with either the load instruction address

[1] See, for example, the HiPEAC Vision (https://www.hipeac.net/publications/vision/).

[2] The 500 most powerful supercomputers in the world are listed at http://www.top500.org.

(Program Counter) or with the data address, to store multiple distinct values per entry, and to trigger selective access on a miss in the level 1 (L1) data cache or in the L2 unified cache.

In [30], the authors used 3-bit confidence counters, predicting only on a saturated counter and resetting it upon a mis-prediction. In fact, with this highly selective confidence mechanism, prediction accuracy is between 95 and 99%, yet coverage is low. They also introduced the VTAGE context-based value predictor (derived from ITTAGE) which uses global branch history and path information to predict values. Delaying the validation of value prediction until the commit stage is the strategy proposed in [29]. By checking correctness only at commit time, an entire pipeline squash is executed in the case of mis-prediction, significantly simplifying the design. This delay and complete pipeline squash is also applied in [14–16] and in our current work. Additionally, in [29], the authors further reduce the complexity by dynamically classifying instructions into early execution, out-of-order execution, and late execution instructions. Instructions having immediate or predicted operands are executed early and in order in the front end. On the contrary, predicted instructions and highly confident branches are executed in-order late and in a pre-commit pipeline stage. Both the early and late execution instructions avoid out-of-order execution, leading to lower energy consumption. In contrast, we apply the value prediction selectively, by focusing only on critical (high latency) load instructions.

In [26], load value approximation is explored in applications that can tolerate inexactness. Rollbacks are eliminated, as load instructions need not be re-executed when the memory value does not exactly match the value provided by the approximator. In contrast, our technique targets all types of applications by not accepting any inexactness regarding the speculated values.

2.2. Design space exploration

M3Explorer [40] is a valuable DSE tool containing many state-of-the-art DSE algorithms. Another well-known DSE framework is implemented as a website [11] where users can upload target microarchitectures. A similar tool is the non ad-hoc search algorithm (NASA) [20] which allows the integration of DSE algorithms, allowing connection to any simulator.

The work presented in [33] addresses the optimal topological placement of Intellectual Property cores onto the network-on-chip (NoC) tiles, an NP-hard problem. As a result, multi-objective evolutionary algorithms were developed, evaluated and optimised for minimizing the NoC communication energy and for obtaining a thermally balanced NoC design. Specific domain knowledge was coded in the mutation and crossover operators, increasing both solution quality and convergence speed.

Another approach quite similar to ours is the one in [24], where the authors apply domain knowledge in the initialisation and mutation stages. The problem solved is multi-objective and aims at minimizing the total annual cost and CO2 emissions produced by energy systems. In the present study, in addition to changing the mutation operator, we introduce domain-specific fuzzy logic rules by which we limit the search space. These fuzzy logic rules represent qualitative knowledge, largely accepted by CPU designers, regarding the interrelation between the sizes of micro-architectural components.

In [19] the FADSE tool was successfully used to perform a DSE of the grid ALU processor (GAP) together with its GAPtimize static scheduler. A recent multi-objective meta-optimisation method is integrated in FADSE [44], with the main aim of obtaining Pareto-optimal solutions in the same amount of time as a single optimisation algorithm. An interesting DSE methodology for FPGAs is proposed in [32], with a Register Transfer Logic design description under resource constraints using low complexity of input transformations and a greedy algorithm for the exploration process. Unlike our solution, that optimisation method is only single-objective, intending to minimise circuit latency under the given constraints. The quality of solutions is also limited, as greedy-like searches tend to become trapped in local optima. In [41] the authors develop a DSE method for mapping tasks to the micro-architectural resources of an MPSoC. They exploit specific domain knowledge related to task allocation through a genetic algorithm. In contrast to our work, their domain knowledge is not focused on finding the best parameter values for a mono-core/multi-core hardware architecture from a multi-objective point of view, and it does not consider hardware constraints, hierarchical parameters or design rules expressed in fuzzy logic as in our work.

A performance modelling and simulation framework called ABSOLUT reduces the complexity of exploration by applying abstract virtual system models [42]. The authors used the ABSOLUT tool in different applications, with a quite acceptable (12%) average difference between simulation results and measurements obtained on real platforms.

In [39], the authors use the strength Pareto evolutionary algorithm (SPEA II) and non-dominated sorting genetic algorithm (NSGA II) to optimise parallel task allocation to cores, with the same objectives considered here—performance, energy, and temperature. In contrast, our technique applies fuzzy logic domain knowledge over the DSE process, which increases solution spread and quality as well as convergence speed.

In [8] the authors implement PESA-II, NSGAII, SPEA2, and PAES multi-objective algorithms to L2 cache memory optimisation to reduce energy consumption and execution time of applications in the embedded systems. The main drawback is that their work only concerns the second level of cache; neither the entire memory hierarchy nor the entire microarchitecture, as in our holistic approach, is covered. Additionally, they do not consider temperature for optimisation.

In [14], an automatic DSE of a direct mapped SLVP-based superscalar microarchitecture was performed. The goal was to determine optimal configurations with respect to performance and energy consumption. Additionally, in this study, we extend the set of fuzzy logic rules to highly parameterise the SLVP structure allowing important functional improvements, and we also perform thermal analysis and control of some selected Pareto-optimal configurations. These improvements

Table 1
SLVP parameter limits.

Parameter	Lower limit	Upper limit
Entries	16	8192
Values per entry	1	4
Associativity	1	8
Index	Instruction address	Memory address
Access type	On miss in DL1	On miss in UL2

make possible a realistic optimisation of the CPU. As a consequence of the significantly enlarged and improved optimisation methodology, we provide more detailed relevant experimental results showing that dynamic value prediction techniques are feasible to be implemented in realistic processors with important performance-energy advantages. We are unaware of other applications of fuzzy logic rules for decision making in automatic DSE of microarchitectures. To our knowledge, only the FADSE tool allows DSE with fuzzy logic rules.

3. Highly parameterised SLVP

We extended the SLVP-based superscalar architecture [14] with the following new important capabilities:

1. Indexing the SLVP table, also using data memory address (memory-centric approach). As access is performed selectively (only upon a miss in the L1 data cache when the data memory address is already available), the SLVP can be indexed using the data memory address without causing delays in the pipeline structure. This comparative analysis is motivated by the fact that some studies have demonstrated that the locality of values is statistically higher on the data memory address [22].
2. Evaluating set-associative SLVP configurations. Despite complexity and access latency of associative tables, they can improve prediction accuracy by decreasing interferences. We compare the efficacy of set-associative and direct mapped SLVP tables.
3. Exploiting a multiple-value locality. The solution consists of a value prediction table that stores in a line the last N distinct results for a certain load instruction. A simple confidence counter and a LRU field are attached to each of these results. In [45], the authors show that most of dynamic instructions produce up to four distinct values in their recent history. This observation is also confirmed by previous value locality evaluations. Thus, we will vary the value history between 1 and 4.
4. Accessing the SLVP selectively on miss in the L2 cache may be of interest, too. We will compare this approach with the previous one (accessing the SLVP on miss in the L1 data cache) from the performance, energy consumption, and temperature dissipation viewpoints.

We will evaluate the proposed microarchitecture within the M-SIM simulator [38] that implements a state of the art superscalar/SMT microarchitecture [17] by performing an automatic DSE with the FADSE tool [5]. Therefore, besides the already existing SLVP size (given as the number of entries) we introduce the following additional parameters:

- Accessing the SLVP table with the Instruction/Data Memory Address;
- The number of distinct values stored in each entry of the SLVP table (N): 1, 2, 4;
- Associativity degree of the SLVP table (Assoc): 1-way, 2-way, 4-way, 8-way;
- Accessing the SLVP on a miss in the L1 data cache (DL1) or only on a miss in the L2 unified cache (UL2).

We will vary the architectural parameters, as in [14]. The parameter limits of the parameterised SLVP are listed in Table 1. Fig. 1 presents an example of 2-way set-associative SLVP with a history of 2 values, without losing generality. Each SLVP entry has as fields:

- Tag_{ke} - the higher part of the instruction/data address from set k and entry e;
- $eLRU_{ke}$ - used only for set-associative tables to select which entry (e) of a certain set (k) is to be overwritten in case of a miss;
- V_{kej} - the last N results stored in set k and entry $e; j$ ranges between 1 and N;
- C_{kej} - an automaton on 2 bits having two unpredictable states and two predictable states attached to each value V_{kej};
- $vLRU_{kej}$ - a counter attached to each value V_{kej} only in the case of history greater than 1, indicating the order in which the values were last produced.

A miss means that the load instruction is not in the SLVP and therefore it is not predictable. A hit means that the load is in the SLVP, so it can be unpredictable (if all its confidences are in unpredictable states) or predictable, generating a correct or an incorrect prediction.

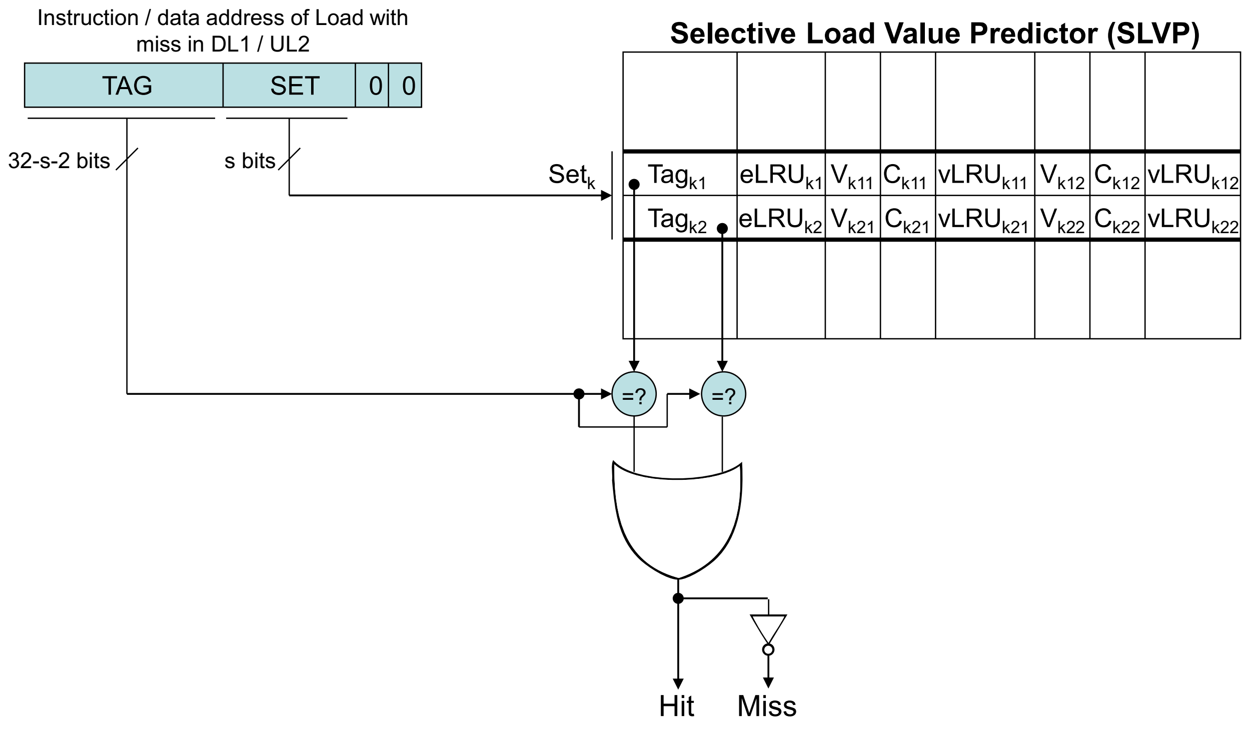

Fig. 1. An example of 2-way associative SLVP with a history of 2 values.

3.1. SLVP

The SLVP table is accessed in the issue stage, only in case of a miss in the L1 data cache in one approach or a miss in the L2 unified cache in a second approach. On a hit in the SLVP table, the highest confidence is checked; if it is in an unpredictable state, the predicted value is ignored, and the current load instruction is processed normally. Otherwise, in the case of a predictable state, the value corresponding to the highest confidence is speculatively forwarded to the in-flight Read After Write dependent instructions.

We check after execution (in the commit stage) the correctness of the predicted value. On mis-prediction, a recovery process is performed which squashes all the speculatively executed instructions and re-executes them using the correct values. We considered in our simulations a realistic average recovery process taking 20 CPU cycles [12].

A malicious, carefully crafted, sequence of instructions may be devised with the specific purpose of triggering this recovery as often as possible. This would cause performance degradation and should be avoided. Although a comprehensive solution is beyond the scope of this work, we briefly discuss possible defences. Monitoring functions at relatively fine granularity throughout the execution of a program could be used to check and prevent the unwanted behaviour. Unfortunately, monitoring can be very expensive to perform, and adding it to code would be detrimental to portability and in software development and debugging. Alternatively, a dedicated monitoring block could be added to the SoC. Such module could monitor the number of cache misses over a given time interval and intervene upon trespassing of some critical threshold. One possible way to mitigate the problem could be to use dynamic selectivity for value prediction. A 3-bit confidence counter could be used for deciding when to trigger value prediction, but in normal operation the most significant bit would be ignored (thus reproducing the case we are focusing on here), whereas it would be considered when responding to critical conditions, effectively reducing misses.

3.2. Updating the SLVP table

The critical load instructions must update the SLVP table in the commit stage: the confidences and vLRU fields on correct prediction, and additionally the history value in case of mis-prediction. The confidence is incremented for the correct value and decremented for incorrect values. If the produced value is not in the history, it overwrites the value having the lowest vLRU. The vLRU fields are set to the maximum for the correct value and decremented for other values.

In the miss case, the entry with the lowest eLRU is selected, the Tag and the first value are introduced into the selected entry, all the confidences are reset (to a strongly unpredictable state), the first vLRU is set to the maximum whereas all the other vLRUs are reset, the eLRU of the selected entry is set to maximum and the eLRUs corresponding to the other entries from the set are decremented.

3.3. Modelling power consumption

For the power consumption model of the SLVP table, we considered the following cache array columns:

- 1 column for Tag;
- 1 column for eLRU (only in the case of set-associative tables);
- N columns for the values;
- N columns for the confidences;
- N columns for vLRUs (only if $N > 1$).

We correspondingly modified the power model of the M-SIM simulator to include the SLVP table.

4. Processor design knowledge expressed through fuzzy rules

The previous section presented our micro-architectural improvements. This section discusses an improved and enlarged processor design knowledge used during the optimisation process of the target microarchitecture for faster convergence and better solution spread and quality.

Fuzzy logic is based on fuzzy set theory [46]. In classical set theory, the characteristic function associated to a set S, for a certain element, is 1 or 0 depending on whether that element belongs or not to S, respectively. In the theory of fuzzy sets this characteristic function can take any real value between 0 and 1, allowing a gradual transition between membership and non-membership. Some human decisions are based on logic inferences that use linguistic variables to model imprecise concepts. Linguistic variables have linguistic values (for example, in the rule 'If John is tall and young he might be good as a basketball player' the linguistic variables and their possible values are size: tall/medium/small, age: young/old, value: good/bad). Fuzzy logic has encountered successful applications in many fields, and we discovered that fuzzy logic rules can also provide a useful representation of knowledge in the CPU design domain.

We will analyse in more depth how a processor design knowledge expressed through fuzzy logic rules might be formally represented. The PDK-FLR must be complete, non-redundant and non-contradictory. Simultaneously fulfilling such requirements represents an old, difficult challenge for mathematics and logic. Some initial developments regarding the contradiction degrees of such fuzzy logic rules were presented in [43]. After understanding such subtle aspects, we must develop a PDK-FLR for our SLVP-based microprocessor and, after that, try to integrate it into our DSE algorithms, which is not a trivial task. The main questions concern the manner in which PDK-FLR affects the convergence of the DSE algorithms and the solutions quality.

There are several operations required to go from a crisp input value to a fuzzy value (fuzzification) and, after the inference system has been solved, to go back (through defuzzification) to a crisp value. In this work we use the Mamdani rule system [25]. This defines how the logical operations are carried out. The general (conjunctive/disjunctive) normal form of a rule system has similar format to the following:

Rule$_1$: IF x_{11} IS A_{11} AND/OR x_{12} IS A_{12} AND/OR $\ldots$ AND/OR x_{1n} IS A_{1n} THEN y_1 IS B_1

...Rule$_m$: IF x_{m1} IS A_{m1} AND/OR x_{m2} IS A_{m2} AND/OR $\ldots$ AND/OR x_{mn} IS A_{mn} THEN y_m IS B_m where A_{ij} are linguistic values of the input whereas B_{ij} are the linguistic values of the output. The part before 'THEN' is called an antecedent and the part after a consequent.

First the AND/OR operations must be solved by using the min (for AND) and max (for OR) functions through a natural analogy with classical (Boolean) logic algebra. The membership function $\mu(\cdot)$ defines the curve associated with a linguistic term. Returning to the AND/OR operators, the AND (the only one we have used in our rules) can be written as:

$$\mu_{\text{AND}}(\mathbf{x}_i) = \min_{j=1}^{n} \left(\mu_{ij}(x_{ij})\right) \tag{1}$$

where $\mathbf{x}_i$ is the vector of the input variables of Rule$_i$ and μ_{ij} is the membership function associated with the linguistic term ij. Then we apply the Mamdani implication:

$$\mu_{A\to B}(x, y) = \min\left(\mu_A(x), \mu_B(y)\right) \tag{2}$$

where $\mu_A(x)$ is the membership function obtained after applying the AND/OR operators and $\mu_B(y)$ is the membership function of the output.

Each rule that has (on the antecedent) a membership value greater than zero will provide as output a membership function. These output functions need to be aggregated into a single output. One of the most used methods is the Mamdani aggregation method [31]. In this method, the output is equal to the maximum (MAX) truth value in the area where the output membership functions overlap:

$$\mu_{\text{AGG}}(x, y) = \max_{i=1}^{m} \left(\mu_{A\to B_i}(x, y)\right) \tag{3}$$

Even if fuzzy sets are used, it is necessary to obtain a crisp output value to make an adequate decision.

Defuzzification represents the conversion of a certain fuzzy quantity to a precise quantity. There are many methods to do this: max membership value (the height method), centroid methods (centre of area, centre of gravity), mean max membership, weighted average method and many others [35]. We have used the centre of gravity method, which is one of the most used methods, and corresponds mathematically to the expected value of probability and is defined as:

$$x_{\mathrm{COG}} = \frac{\int x \cdot \mu(x)dx}{\int \mu(x)dx} \tag{4}$$

Recall that $\mu(x)$ is the membership function.

The integration of the obtained crisp value into the NSGA-II algorithm's mutation operator was performed as follows. The 'fuzzy logic mutation' operator determines the membership of the output variable which is followed by the centre of gravity (COG) estimation of this membership function. For this x_{COG} crisp value the membership $\mu(x_{\mathrm{COG}})$ of the final output function can be calculated. The current output parameter is set to the x_{COG} value with a certain probability (explained below). We used the bit flip mutation fuzzy operator presented in [4].

To define the threshold corresponding to the mutation probability, we use the following Gaussian function:

$$f(x) = a \cdot \exp\left\{-\frac{(x-b)^2}{2c}\right\} \tag{5}$$

The goal is to have a relatively high probability of mutation at the start of the DSE. As the algorithm progresses, the influence of the rules will be less significant. We selected: a = 1-P_M, $b = 0$ and $c = 150$, where P_M denotes the mutation probability. The value of x is incremented for each generated individual. The parameters were selected such that after 500 individuals ($x = 500$) the curve is 'close enough' to the value 0. For a population of 100 individuals after five generations, the influence of the fuzzy rules will be negligible. The maximum of this function is $1 - P_M$. To preserve solution diversity, we will decrease the mutation probability as the algorithm progresses and we also multiply the final value by 0.8.

The membership $\mu(x_{\mathrm{COG}})$ value is used as a measure of confidence. The mutation probability threshold T_{MP}, used in our modified mutation operator, is defined by the following formula:

$$T_{MP} = 0.8 \cdot \mu(x_{\mathrm{COG}}) \cdot \left((1-P_M) \cdot \exp\left\{-\frac{x^2}{2 \cdot 150}\right\} + P_M\right) \tag{6}$$

This number is used in the new mutation genetic operator.

To avoid unfeasible configurations, some constraints regarding the caches were applied, as in [14], where the design space was reduced from $2.5 \cdot 10^{15}$ configurations to 3%, i.e. $7.7 \cdot 10^{13}$ configurations.

Below we present some qualitative design knowledge represented using linguistic variables within fuzzy logic rules in Conjunctive Normal Form (CNF). These fuzzy logic rules represent an extended and improved version of those from the set we previously used in [14]. The rules were implemented in the mutation operator of our DSE algorithms. We have not implemented this type of knowledge into the crossover operator. These CNF fuzzy logic rules represent qualitative knowledge largely accepted by the CPU designers in an empirical manner, fulfilling the usual 'common-sense' requirement:

(R1) IF Number_of_Physical_Register_Sets IS *small/big* THEN
Decode/Issue/Commit_Width IS *small/big*
(R2) IF Decode_Width IS *small/big* AND Issue_Width IS *small/big* AND Commit_Width IS *small/big* THEN
Number_of_Physical_Register_Sets IS *small/big*
(R3) IF SLVP_Size IS *small/big* THEN
L1_Data_Cache IS *big/small*
(R4) IF L1_Data_Cache IS *small/big* THEN
SLVP_Size IS *big/small*
(R5) IF N IS *small/big* AND Assoc IS *small/big* THEN
SLVP_Size IS *big/small*

All the linguistic variables used represent the CPU parameters as they are shown in Table 1. The last rule is justified by the results obtained in [14] where for $N = 1$ and Assoc = 1 (direct mapped last value predictor); the optimal SLVP size was large (4096, 8192 entries).

The gradual membership functions were defined as in Table 2

The rules have been written in Fuzzy Control Language (FCL) which is a domain specific programming language for fuzzy logic. It has been standardised through IEC 61131-7.[3] In FCL membership functions, rules, methods to solve the rule system, etc. can be specified. We use jFuzzyLogic[4] to parse the FCL source file and solve the rule system. The FCL file sent to the jFuzzyLogic library contains all our input and output parameters and their associated membership functions. We also specify in the FCL the functions to be used by the fuzzy operators for solving the rule system. The output value computed by the library is then passed to the fuzzy logic mutation genetic operator and it is used as we have already presented.

Another set of rules (the 'deterministic' constraints) are implemented directly in FADSE. These 'crisp' rules can define relationships such as 'greater', 'greater or equal', and 'less', as well as operations between parameters (multiplication, addition,

[3] http://www.fuzzytech.com/binaries/ieccd1.pdf.
[4] http://jfuzzylogic.sourceforge.net/.

Table 2
Membership functions.

Parameter	Range	Small	Big
SLVP entries	[16, 256]	1	0
	(256, 2048)	Linearly decreases to 0	Linearly increases to 1
	[2048, 8192]	0	1
SLVP history	1	1	0
	(1, 4)	Linearly decreases to 0	Linearly increases to 1
	4	0	1
SLVP associativity	1	1	0
	(1, 8)	Linearly decreases to 0	Linearly increases to 1
	8	0	1
DL1 cache	[16, 2048]	1	0
	(2048, 32768)	Linearly decreases to 0	Linearly increases to 1
	[32768, 8388608]	0	1
Decode, Issue,	[2, 4]	1	0
and Commit	(4,16)	Linearly decreases to 0	Linearly increases to 1
width	[16, 32]	0	1
Physical register	2	1	0
Sets (same number	(2, 8)	Linearly decreases to 0	Linearly increases to 1
for int and fp)	8	0	1

etc.). Rules can also be composed using operators such as ‘and’ and ‘or’. If an individual does not comply with these rules it is automatically marked as infeasible giving it a much lower chance of surviving in the next generation. The rules are specified through an XML file and are parsed by our DSE framework.

5. Simulation methodology

5.1. Tools used in simulations

The FADSE framework integrates the jMetal library[5] which provides many multi-objective heuristics. One of the implemented algorithms used in this work is NSGA-II, a multi-objective genetic algorithm described in [10]. FADSE has as inputs the metaheuristic (NSGA-II in this case) and specific parameters such as the following:

- The name of the genetic algorithm parameters (crossover operator, mutation operator, selection operator, population size, maximum evaluations, mutation probability, crossover probability) and their corresponding values;
- The benchmark to be run;
- The name of the evaluated objectives (the cycles per instruction (CPI) and energy consumption of the Alpha AXP 21264 processor enhanced with SLVP);
- Configurations for the M-SIM v2.0 simulator enhanced with SLVP.

FADSE includes many quality metrics, some of them being inherited from the jMetal library. The outputs of FADSE are:

- Pareto fronts - a set of non-dominated solutions, being chosen as optimal, if no objective can be improved without sacrificing another objective;
- Hypervolume - used to observe the evolution of a single algorithm or to compare multiple runs;
- Coverage - used to compare the Pareto fronts of two populations or of two different runs.

We integrated the jFuzzyLogic library into FADSE and thus it accepts an input file with the fuzzy logic rules written in FCL.

M-SIM v2.0 is an open source multi-threaded architectural simulator developed in the C language by Joseph Sharkey [38]. We enhanced M-SIM with SLVP techniques, using processor design knowledge. We modified the power model of M-SIM to include the SLVP table. The inputs of M-SIM are (see also Table 1):

- DL1/IL1/UL2 cache parameters: set, block size, associativity;
- SLVP: number of entries, associativity, access type;
- Core parameters: decode/issue/commit width, physical register sets, ROB/LSQ/IQ entries.

The outputs of M-SIM are the CPI and power consumption.

CACTI[6] is a cache and memory model developed by Norman Jouppi and other researchers from the Hewlett Packard laboratories. The inputs of CACTI are:

[5] https://jmetal.github.io/jMetal/.
[6] http://www.hpl.hp.com/research/cacti/.

- Integrated circuit technology;
- Cache geometry (size, associativity, block size).

The outputs of CACTI are the time, power and area estimations.

QUILT [1] represents a user-friendly circuit floor planning environment developed in the Java language by David Albonesi (Cornell University, USA) and Greg Briggs (Rochester University, USA). QUILT allows building floorplans for the HotSpot simulator. The inputs of QUILT are:

- The microarchitecture's area;
- Components.

The output of QUILT is a floorplan coordinate text file.

The HotSpot simulator [18] is a thermal model developed in the C language by Kevin Skadron and his colleagues from Virginia University, USA. The inputs of HotSpot are:

- The microarchitecture floorplan;
- A power trace file.

HotSpot outputs the corresponding transient temperatures onto a temperature trace file.

5.2. Performance and energy consumption analysis

The experimental results were obtained on SPEC 2000, skipping the initial 300 million instructions and executing the next 500 million. Simulations were performed on an Intel Xeon powered HPC with 96 cores at 2 GHz. At each generation (we considered 100 individuals per each generation) the algorithm sends a number of individuals for simulation to clients running on each core equal to the number of available processing resources. Once a client finishes a simulation a new individual is sent. At the end of the generation, a join operation is triggered and all the clients finish their processing before moving to the next generation. The granularity of the job is at benchmark level, which means that for a single individual we spawn 12 jobs (equal to the number of benchmarks) to be run by the clients. The simulated microarchitecture is an Alpha AXP 21264 microprocessor enhanced with our SLVP unit considering 32 nm CMOS technology, 1.2 GHz frequency and 1V Vdd.

In this work, we consider the following latencies: DL1 / IL1 / UL2 / MEM latency = 1 / 1 / 10 / 200 CPU cycles. For a 1.2 GHz clock frequency on 32 nm technology (thus, a clock cycle of 0.83 ns), the SLVP's latency will vary depending on its organisation (number of entries, line size influenced by the number of values and associativity). We determined with the CACTI 5.3 tool that the SLVP latency is 1 CPU cycle in most of the configurations; only in the configuration having 8192 entries, 8-way associativity and a history of 4 values the latency is 2 cycles.

As a performance metric, we chose CPI (instead of instructions per cycle (IPC)) so that all objectives are to be minimised and a Pareto front can be obtained. Formula (7) was used to compute CPI reduction:

$$CPI_{reduction} = \frac{CPI_{base} - CPI_{improved}}{CPI_{base}} \tag{7}$$

where CPI_{base} and $CPI_{improved}$ are average cycles per instruction with the baseline and improved architectures, respectively.

The simulator's power model is described in [2]. We applied the 'aggressive non-ideal conditional clocking' model [7] which linearly scales the active units' power with their usage and considers a reduced 10% power dissipation for unused units. The energy consumption [W · cycles] is given by the following formula:

$$E = P \cdot T \tag{8}$$

where P is the instantaneous average power consumption and T is the total simulation time in cycles. The relative energy reduction is described by the following formula:

$$E_{reduction} = \frac{E_{base} - E_{improved}}{E_{base}} \tag{9}$$

where, E_{base} and $E_{improved}$ represent the energy consumption of the baseline and the enhanced microarchitecture, respectively.

To determine the performance and energy consumption of the Alpha AXP 21264 processor enhanced with SLVP, we used the M-SIM v2.0 simulator [38]. The automatic DSE is performed by our FADSE tool which uses a dedicated connector to aggregate the parameter values and the benchmark information in a command line to start the simulation, and at the end, to parse the results for the objectives.

5.3. Thermal analysis

To validate the hardware implementation of a microarchitecture whose optimal parameter values were derived from automated DSE methods, performance and energy consumption analysis should be complemented by a thermal analysis

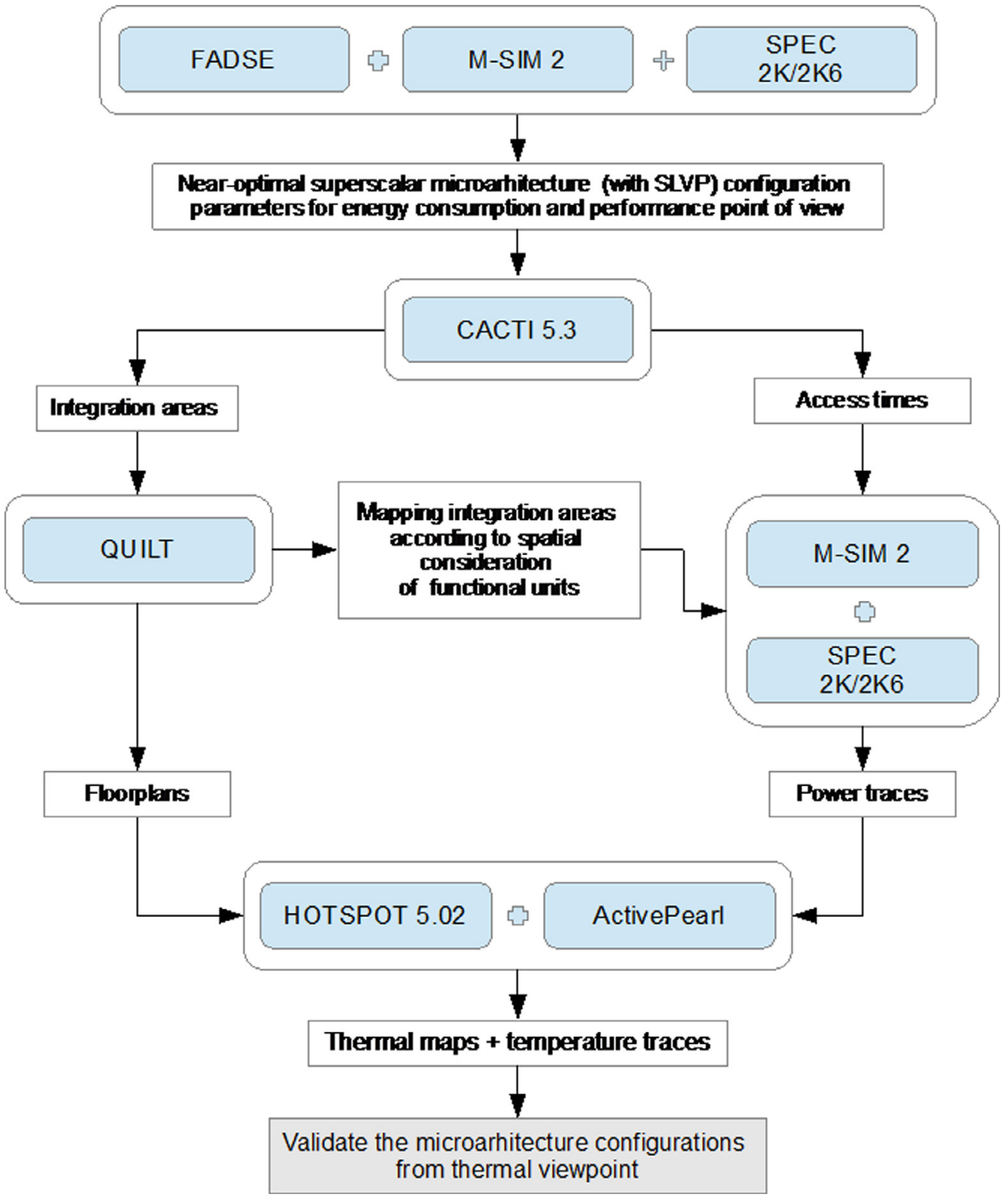
FADSE
M-SIM 2
SPEC 2K/2K6
Near-optimal superscalar microarhitecture (with SLVP) configuration parameters for energy consumption and performance point of view
CACTI 5.3
Integration areas
Access times
QUILT
Mapping integration areas according to spatial consideration of functional units
M-SIM 2
SPEC 2K/2K6
Floorplans
Power traces
HOTSPOT 5.02
ActivePearl
Thermal maps + temperature traces
Validate the microarhitecture configurations from thermal viewpoint

Fig. 2. Workflow for finding a near optimal microarchitecture from the thermal viewpoint.

of the improved superscalar processor, also establishing a maximum permissible temperature which may occur on chip. A processor that reaches peak temperatures over the feasibility threshold of 111.8 °C [36] is very difficult to cool by conventional convection methods in current computer systems (laptop, desktop, portable media devices). In Fig. 2 we present the workflow to validate the microarchitecture from a thermal viewpoint.

With the parameter values corresponding to the configurations selected by FADSE, we used the CACTI 5.3 tool to determine the area of integration and the access time of the following memories: ICache, DCache, UL2Cache and SLVP. Based on the obtained integration areas, we have modified the original ALPHA 21264 floorplan with the QUILT tool [1] to obtain a more realistic approximation of our microarchitecture floorplan. This process has been applied to all the selected configurations; thus, each configuration has its own floorplan.

The power consumption has been determined with the M-SIM simulator [38] (that integrates the Wattch framework) for each configuration with and without our SLVP structure. The power traces are generated collecting power statistics for each functional unit every 500 kilocycles. In the UL2Cache case, due to its U-shape surrounding other components, and because QUILT uses rectangular shapes to model the architecture, we were constrained to split the area of UL2Cache into three separate blocks. For a better approximation in these situations, we reported the computed power from M-SIM for

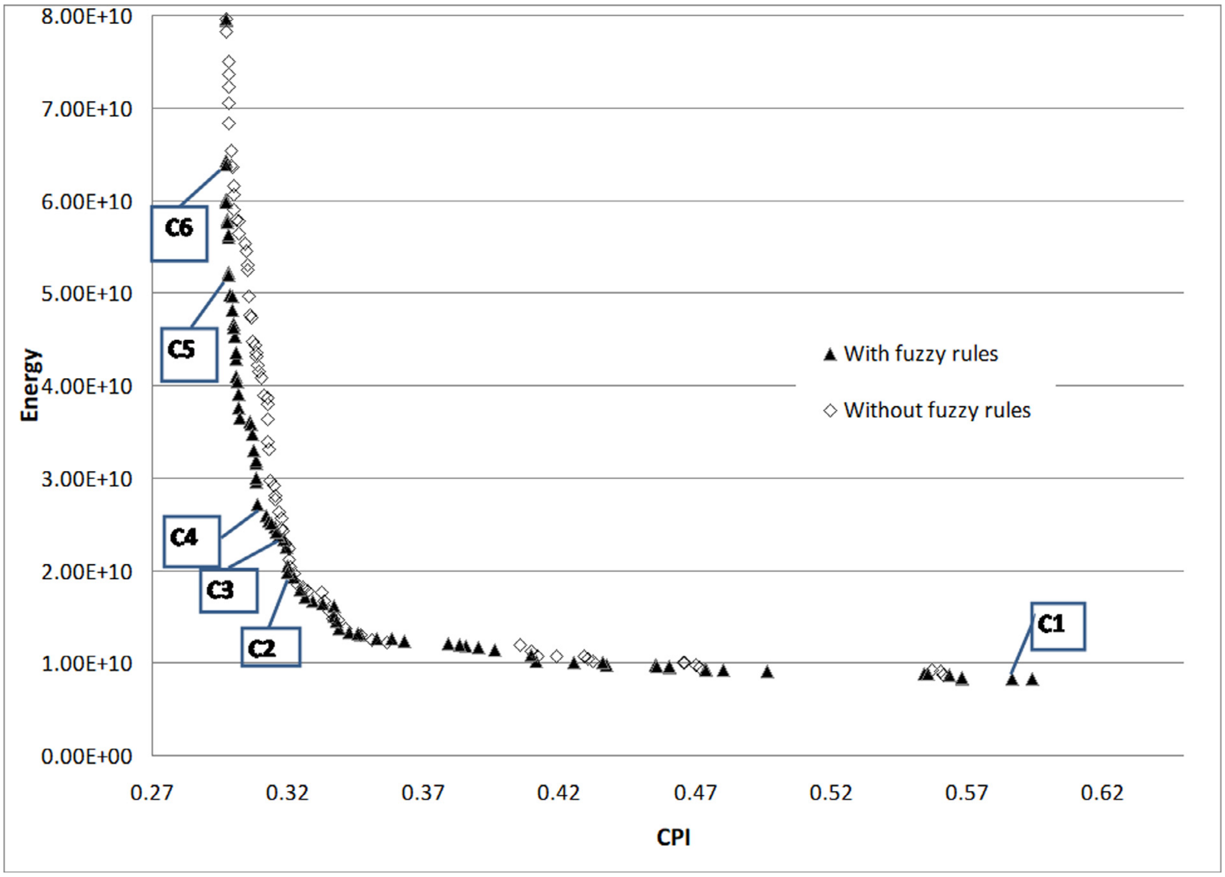

Fig. 3. Pareto front comparison, running NSAG-II.

each block based on its integration area. The next step is to generate the thermal maps and temperature traces with the HotSpot 5.02 simulator [18] to identify possible hotspots (temperatures over 111.8 °C) and validate the microarchitecture. HotSpot 5.02 uses as input files the floorplan and the previously obtained power trace. The simulator has a variety of cooling configuration packages. For these simulations, we used the default cooling package (as a first cooling configuration) that consists of forced air flow over a copper heatsink and, additional to this package, we enabled a secondary heat transfer path (as a secondary configuration).

6. Experimental results

6.1. Performance and energy consumption evaluations

First, we compared the Pareto fronts of our new extended architecture with and without fuzzy logic rules. Further we selected six configurations from the obtained Pareto front for thermal analysis.

Fig. 3 presents the Pareto front approximations discovered by our modified NSGA-II genetic algorithm after 52 generations (at the end of the exploration). Globally, the best results are obtained with PDK-FLR, showing a noticeable gain. We can observe that in the area with high energies, the DSE running with fuzzy logic rules is able to discover significantly better results, which is remarkable. But that is not the only area of interest; for low energies, quality of results is comparable for both Pareto fronts, but the spread of solutions found by the run with fuzzy logic rules is better. This means that a computer architect will have more choices in selecting an optimal architecture from the Pareto front discovered by the optimisation algorithm. It also proves that even if we impose some rules on the algorithm that reflect some biased 'pre-conceptions' of the architect ('a priori design knowledge'), the algorithm does not remain confined to the restricted areas imposed by these rules and can provide diversity in solutions. This is possible because the fuzzy logic rules are applied with Gaussian probability distribution.

All the individuals from the Pareto front have set-associative SLVPs and 86% of them have a value history higher than 1. Interestingly, none of the configurations from the final population belong to our previous optimised microarchitecture [14] with direct mapped SLVP and only one value per entry. Another important observation is that all the individuals from

Table 3
Pareto optimal configurations.

Parameter		C1	C2	C3	C4	C5	C6
DL1 cache	Sets	64	256	256	256	256	256
	Block size	64	64	64	64	256	128
	Associativity	2	2	2	2	2	8
IL1 cache	Sets	512	512	512	512	32	32
	Block size	64	8	8	8	256	256
	Associativity	1	8	8	8	8	8
UL2 cache	Sets	1024	4096	4096	4096	4096	4096
	Block size	128	256	256	256	256	256
	Associativity	8	2	2	2	8	8
SLVP	Entries	8192	128	8192	128	8192	8192
	Values per entry	2	2	2	2	2	2
	Associativity	8	8	8	8	4	4
	Decode width	8	8	32	32	32	32
	Issue width	4	8	8	8	16	16
	Commit width	32	32	32	32	32	32
	ROB size	128	256	256	256	1024	1024
	LSQ size	64	64	64	64	512	1024
	IQ size	128	64	64	64	256	256
	Physical register sets	64	128	128	128	256	256
	int / fp ALU	8/8	8/8	8/8	8/8	8/8	8/8
	int / fp MUL/DIV	2/4	2/4	2/4	8/4	8/8	8/8

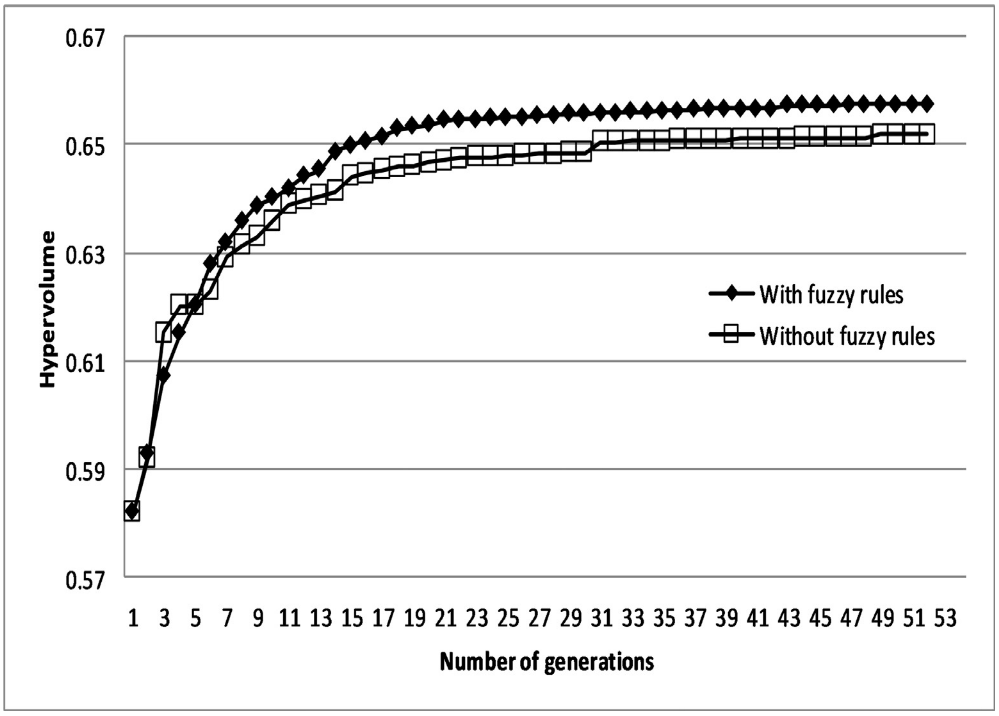

Fig. 4. Hypervolume comparison, running NSGA-II.

the Pareto front are accessing the SLVP only on a miss in the DL1 cache. Configurations which access the SLVP only on a miss in the UL2 cache are performing worse, possibly because only three benchmarks (applu, lucas, and mgrid) have a relatively high number of loads with misses in the L2 cache [16]. An important observation is that all the configurations from the Pareto fronts index the SLVP table with the instruction address. Accessing the SLVP table with the data memory address proved less effective, even though some researchers report that significant value locality exists in both directions: memory-centric and producer-centric [22]. Table 3 presents six configurations (C1-C6 from Fig. 3) selected for thermal evaluation from the Pareto front obtained by the run with fuzzy logic rules.

Fig. 4 compares the hypervolumes of the explorations with and without fuzzy logic rules. Hypervolume evolution shows the convergence speed of the algorithm to a relatively stable Pareto front. A larger hyper-volume means better quality solutions: ‘whenever one approximation completely dominates another approximation, the hypervolume of the former will

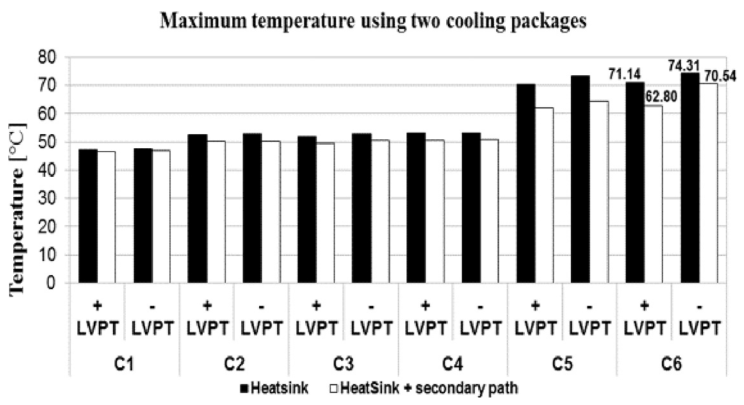

Fig. 5. Maximum temperatures using two cooling packages reached on the 6 selected micro-architectural configurations.

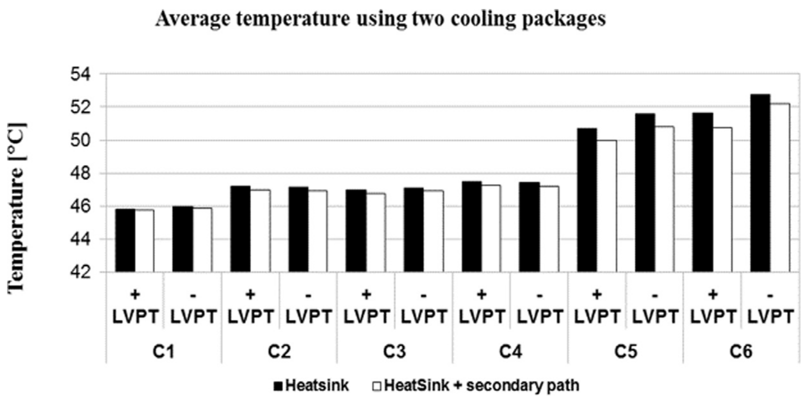

Fig. 6. Average chip temperatures reached on the 6 selected microarchitectures.

be greater than the hypervolume of the latter' [47]. From the hypervolume evolution over the generations we can observe that the results obtained by the run with fuzzy logic rules are generally better than the ones obtained without them. The improved convergence induced by the usage of fuzzy information is significant and can be deduced from the fact that the DSE with fuzzy logic rules obtains a quality of results (from a hypervolume value point of view) in generation 17 which is equivalent with the quality obtained without fuzzy rules in generation 48. Furthermore, the hypervolume value reached in generation 18 is never reached without fuzzy logic rules during all 52 simulated generations. This represents a considerable improvement in convergence speed.

According to Figs. 3 and 4 the best solution quality and convergence speed are given by the exploration with fuzzy logic rules. Running one generation (100 individuals), without PDK-FLR, on 96 cores at 2 GHz on an Intel Xeon based HPC, takes approximately one day. Running all 52 generations, without fuzzy logic rules takes 50 days. With fuzzy logic rules we obtained the same results in only 33 days (thus 34% faster).

6.2. Thermal evaluations and control

The last step of our analysis was to determine how the Pareto optimal microarchitectures found by FADSE behave from a thermal point of view, and we also wanted to quantify the thermal impact achieved by introducing the SLVP module into the M-SIM superscalar microarchitecture. Thus, we analysed six micro-architectural configurations (Fig. 3) from the Pareto front corresponding to maximum energy and the best processing performance (C5 & C6), minimum energy areas with the worst performance (C1) and three configurations situated at the inflexion of the Pareto front approximation (C2 & C3 & C4). The parameters of these selected configurations are presented in Table 3.

As Fig. 3 reveals, the performance of the C2, C3 and C4 configurations is roughly equal to that obtained on C5 and C6. From a thermal point of view, all the studied microarchitectures are feasible in hardware implementations (the maximum temperatures inside chips being under 75 °C according to Fig. 5 and the average temperatures being situated under 53°C according to Fig. 6). In Figs. 5 and 3 note that the maximal temperatures and energy consumption are reached for configurations C5 and C6. According to our initial expectations, we note that the maximum temperatures were obtained for the Instruction Queue (IQ) in all the simulated benchmarks (see Figs. 7 and 5).

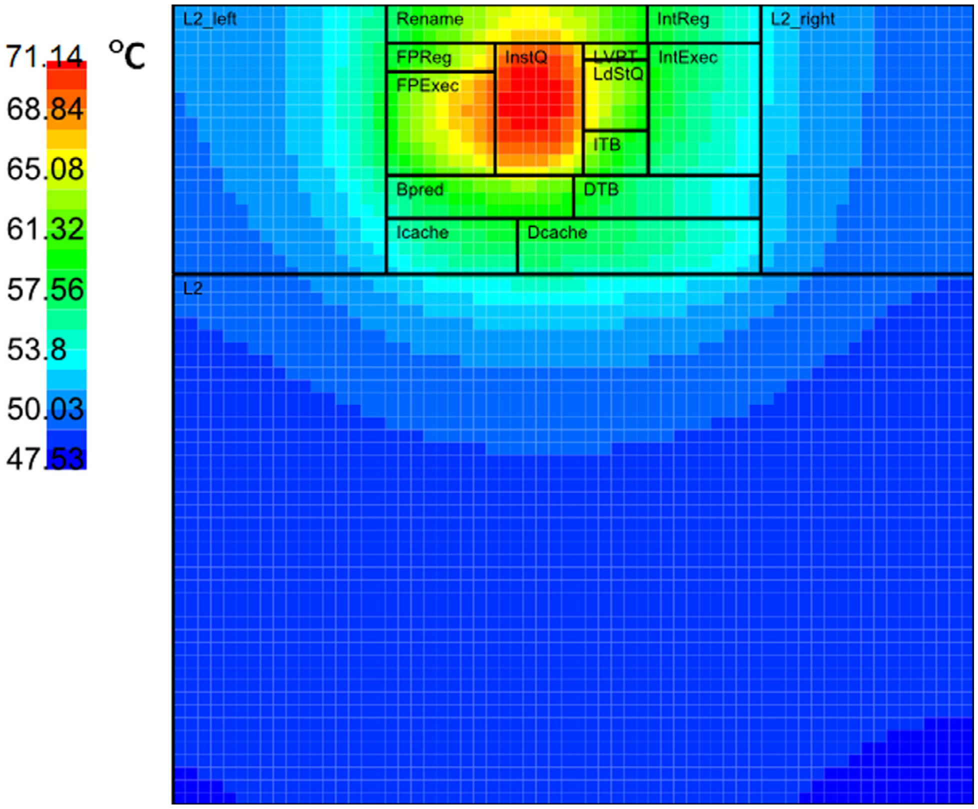

Fig. 7. Temperature map for the C6 configuration of the SLVP-based superscalar architecture.

By introducing a SLVP module into a superscalar microarchitecture, we did not produce a significant thermal impact on the chip. However, we might be tempted to conclude that the pressure and, as a consequence, the higher temperature on the IQ is caused by the load value prediction, which increases the demand for resources as instructions executed with erroneous operands must be re-executed. However, this is not true at all, because in Figs. 5 and 7 we can observe that the SLVP reduces the maximum temperature of the chip by up to 8 °C. A very interesting observation related to Fig. 5 consists of the importance of a secondary heat transfer path located between the processor and the heatsink. The secondary path behaves like a ‘heat buffer’. This reduces the maximum chip temperature to normal values. The secondary cooling package utility improves the C5 and C6 configurations, where the temperature reduction is around 9 °C.

Furthermore, based on previous research presented in [3,21], we chose to implement the following micro-architectural solutions to reduce the overload in the IQ:

- reducing the number of mis-speculated branch instructions using an effective perceptron predictor [21];
- finding the appropriate size of the decode/commit width (DCR);
- issue-aware fetch gating (whenever the number of decoded instructions is higher than those which do commit, it will disable the fetch stage in the next cycle - known as decode/commit rate fetch gating).

Much of the wasted energy from the front-end pipeline stages is due to the mis-speculated control instructions. Mis-predicted branches cause more instructions to be decoded than those that are committed. Mis-speculated instructions will reside in the IQ with no useful purpose until they are squashed after the branch instruction’s resolution. The front-end energy can be decreased by performing better branch predictions. By increasing the accuracy of branch predictions, fewer mis-speculated instructions are executed, resulting in less pressure on the IQ, which increases performance and saves energy. In this sense we have integrated into our simulator a perceptron branch predictor, too.

Table 4 illustrates the simulation results compared with a similar microarchitecture having different (just the branch predictor) structure (Bimodal). As we can see, if the prediction accuracy increases, it will contribute to higher processing performance, lower energy consumption and lower heat. The main cause of IQ overloading can be attributed to the mismatch between decoded and committed instructions. Technically, the producer rate is obtained by tracking the queue occupancy, while for the consumer rate we monitor how many instructions are leaving the reorder buffer.

Fig. 8 illustrates an execution of the gzip benchmark assuming a superscalar architecture with maximum Decode width and Commit width of 32 instructions. We note that in more than 7% of the cycles no instructions are committed while the decoding unit is in use, indicating that the producer is not correlated with the consumer, e.g. the front-end processing is

Table 4
Percentage variation of the Micro-architectural Metrics by Reducing the Number of Mis-speculated Branches.

Metrics	Variation
Prediction accuracy	+4.2%
Processing performance	+9.5%
Energy	−5.9%
Maximum temperatures	−2.3%

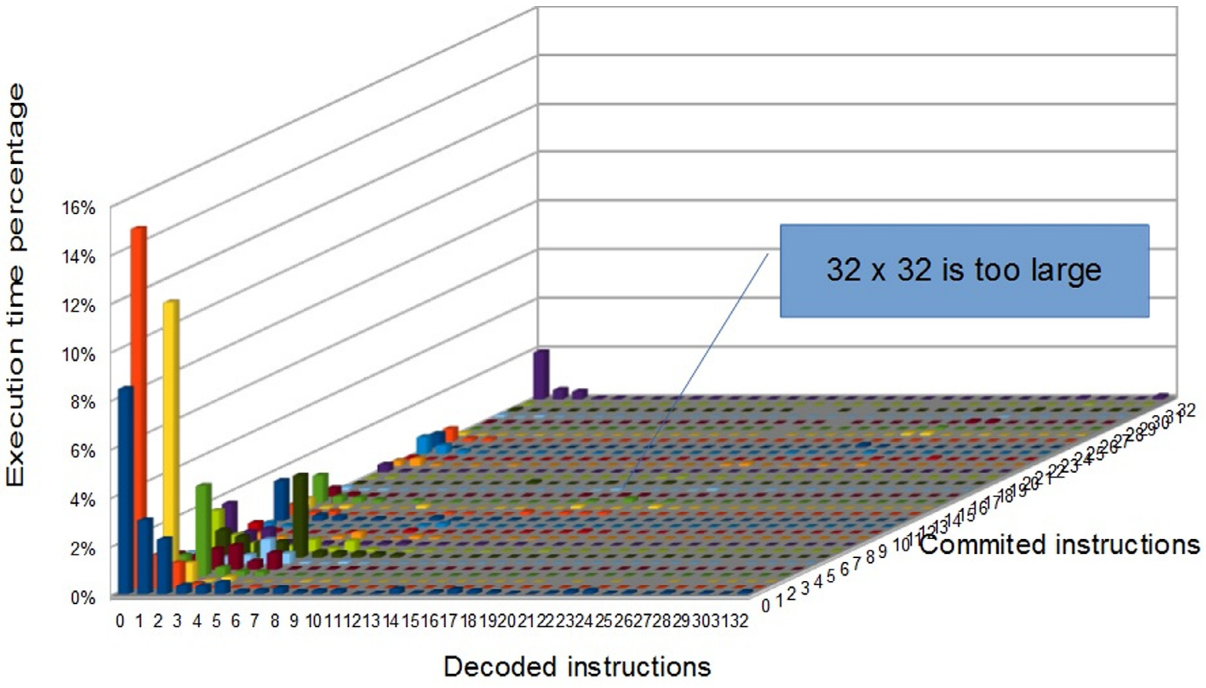

Fig. 8. Mismatches between the producer and consumer rate within the instruction window.

too fast to supply the commit rate. As Fig. 8 illustrates, a superscalar architecture with a DCR of 32 is too large, involving an inefficient exploitation of the IQ and leading sometimes to potential hotspots. By applying issue-aware fetch gating, performance decreases by 4% but the temperature is reduced by 7 °C.

7. Conclusion and further work

In this work, we presented an improved automatic DSE methodology which significantly enlarges an initial domain knowledge represented through fuzzy logic rules and other deterministic restrictions. We highly extended the SLVP microarchitecture and we also performed a thermal analysis and control which verifies the energy efficiency of the design and its temperature optimisation. These improvements led to a new, realistic and original methodology for performance, energy consumption, and temperature optimisation of the CPU.

Evaluations showed that a set-associative SLVP table is more effective than a direct mapped table and that 86% of the configurations from the Pareto front use multiple values per SLVP entry ($N > 1$). The fuzzy logic rules produce a Pareto front which is noticeably better than the front obtained without them, and convergence is 34% faster with our fuzzy logic rules. The spread of the solutions found by the DSE algorithm with fuzzy logic rules is also better, providing computer architects with more choices in selecting one of the Pareto optimal configurations discovered.

We analysed the thermal dissipation of six architectural configurations selected from the Pareto front and all of them were hardware feasible, as their temperatures were below the feasibility threshold of 111.8 °C. It is very important to remark that our added SLVP structure does not negatively impact the maximum chip temperature; on the contrary, its inclusion causes the maximum chip temperature to decrease by up to 8 °C. We also implemented some effective micro-architectural solutions to reduce the IQ overload.

Broadly speaking, we showed that the cross-fertilisation between computer architecture and CPU multi-objective optimisation methods, on one hand, and knowledge representation domain, on the other hand, leads to a more effective processor design. Extending and improving this idea may be extremely promising in computer architecture research. In fact, fuzzy logic rules are just one of the ways for describing domain knowledge; alternatives worth investigating include semantic nets and other knowledge representation methods such as the domain ontologies used in semantic web projects. Automatically computing the degrees of contradiction for such sets of fuzzy logic rules would also be helpful to avoid solution quality degradation.

Although temperature (the HotSpot tool) is not yet integrated into our SLVP based speculative superscalar simulator as a full-fledged third objective, the thermal analysis we performed did indicate the maximum chip temperatures and produced solutions that are both feasible and have desirable properties. We would like to repeat the experiments on SLVP-based multicore microarchitectures and study the effect of adding to our simulator two new objectives: temperature and die size.

Acknowledgements

We would like to thank former M.Sc. student Claudiu Buduleci of the Embedded Systems Specialisation at 'Lucian Blaga' University of Sibiu, Romania, for offering his help in the technical development of the thermal analysis section.

References

[1] G.J. Briggs, E.J. Tan, N.A. Nelson, D.H. Albonesi, QUILT: a GUI-based integrated circuit floorplanning environment for computer architecture research and education, in: Proceedings of the 2005 Workshop on Computer Architecture Education: Held in Conjunction with the 32nd International Symposium on Computer Architecture, Madison, WI, USA, ACM, 2005, p. 5.
[2] D. Brooks, V. Tiwari, M. Martonosi, Wattch: A framework for architectural-level power analysis and optimizations, 28, ACM, 2000.
[3] A. Buyuktosunoglu, T. Karkhanis, D.H. Albonesi, P. Bose, Energy efficient co-adaptive instruction fetch and issue, in: Proceedings of the 30th Annual International Symposium on Computer Architecture, San Diego, CA, USA, IEEE, 2003, pp. 147–156.
[4] H. Calborean, Multi-Objective Optimization of Advanced Computer Architectures using Domain-Knowledge, "Lucian Blaga" University of Sibiu, Romania, 2011 Ph.D. thesis.
[5] H. Calborean, L. Vintan, An automatic design space exploration framework for multicore architecture optimizations, in: 9th RoEduNet International Conference, Sibiu, Romania, IEEE, 2010, pp. 202–207.
[6] A. Castiglione, F. Palmieri, U. Fiore, A. Castiglione, A. De Santis, Modeling energy-efficient secure communications in multi-mode wireless mobile devices, J. Comput. Syst. Sci. 81 (8) (2015) 1464–1478.
[7] J. Chen, M. Dubois, P. Stenstrom, Integrating complete-system and user-level performance/power simulators: the SimWattch approach, in: IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Austin, TX, USA, IEEE, 2003, pp. 1–10.
[8] F.R. Cordeiro, A.G. da Silva-Filho, Multi-objective optimization applied to unified second level cache memory hierarchy tuning aiming at energy and performance optimization, Appl. Soft Comput. 49 (2016) 603–610.
[9] P.L. De Angelis, F. Perla, P. Zanetti, Hybrid MPI/OpenMP application on multicore architectures: the case of profit-sharing life insurance policies valuation, Appl. Math Sci. 7 (102) (2013) 5051–5070.
[10] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput. 6 (2) (2002) 182–197.
[11] V. Desmet, S. Girbal, O. Temam, Archexplorer.org: joint compiler/hardware exploration for fair comparison of architectures, 13th Workshop on Interaction between Compilers and Computer Architecture (Interact-13), Raleigh, NC, USA, 2009.
[12] S. Eyerman, J.E. Smith, L. Eeckhout, Characterizing the branch misprediction penalty, in: IEEE International Symposium on Performance Analysis of Systems and Software, Austin, TX, USA, IEEE, 2006, pp. 48–58.
[13] U. Fiore, A. Castiglione, A. De Santis, F. Palmieri, Exploiting battery-drain vulnerabilities in mobile smart devices, IEEE Trans. Sustainable Comput. 2 (2) (2017) 90–99.
[14] A. Gellert, H. Calborean, L. Vintan, A. Florea, Multi-objective optimisations for a superscalar architecture with selective value prediction, IET Comput. Dig. Tech. 6 (4) (2012) 205–213.
[15] A. Gellert, A. Florea, L. Vintan, Exploiting selective instruction reuse and value prediction in a superscalar architecture, J. Syst. Archit. 55 (3) (2009) 188–195.
[16] A. Gellert, G. Palermo, V. Zaccaria, A. Florea, L. Vintan, C. Silvano, Energy-performance design space exploration in SMT architectures exploiting selective load value predictions, in: Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, Germany, IEEE, 2010, pp. 271–274.
[17] J.L. Hennessy, D.A. Patterson, Computer Architecture: A Quantitative Approach, Morgan Kaufmann, 2017.
[18] W. Huang, K. Skadron, S. Gurumurthi, R.J. Ribando, M.R. Stan, Differentiating the roles of IR measurement and simulation for power and temperature-aware design, in: IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Boston, MA, USA, IEEE, 2009, pp. 1–10.
[19] R. Jahr, T. Ungerer, H. Calborean, L. Vintan, Automatic multi-objective optimization of parameters for hardware and code optimizations, in: International Conference on High Performance Computing and Simulation (HPCS), Istanbul, Turkey, IEEE, 2011, pp. 308–316.
[20] Z.J. Jia, A.D. Pimentel, M. Thompson, T. Bautista, A. Núñez, NASA: A generic infrastructure for system-level MP-SoC design space exploration, in: 8th IEEE Workshop on Embedded Systems for Real-Time Multimedia (ESTIMedia), Scottsdale, AZ, USA, IEEE, 2010, pp. 41–50.
[21] D.A. Jiménez, C. Lin, Dynamic branch prediction with perceptrons, in: 7th International Symposium on High-Performance Computer Architecture (HPCA), Monterrey, Nuevo Leon, Mexico, IEEE, 2001, pp. 197–206.
[22] K.M. Lepak, M.H. Lipasti, Silent stores for free, in: 33rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-33), Monterey, CA, USA, IEEE, 2000, pp. 22–31.
[23] M.H. Lipasti, C.B. Wilkerson, J.P. Shen, Value locality and load value prediction, ACM SIGPLAN Notices 31 (9) (1996) 138–147.
[24] M.S. Mahbub, M. Wagner, L. Crema, Incorporating domain knowledge into the optimization of energy systems, Appl. Soft Comput. 47 (2016) 483–493.
[25] E. Mamdani, Application of fuzzy logic to approximate reasoning using linguistic synthesis, IEEE Trans. Comput. 12 (C-26) (1977) 1182–1191.
[26] J.S. Miguel, M. Badr, N.E. Jerger, Load value approximation, in: 47th Annual IEEE/ACM International Symposium on Microarchitecture, Cambridge, UK, IEEE Computer Society, 2014, pp. 127–139.
[27] F. Palmieri, S. Ricciardi, U. Fiore, Evaluating network-based DoS attacks under the energy consumption perspective: new security issues in the coming green ICT area, in: International Conference on Broadband and Wireless Computing, Communication and Applications (BWCCA), Barcelona, Spain, IEEE, 2011, pp. 374–379.
[28] F. Palmieri, S. Ricciardi, U. Fiore, M. Ficco, A. Castiglione, Energy-oriented denial of service attacks: an emerging menace for large cloud infrastructures, J. Supercomput. 71 (5) (2015) 1620–1641.
[29] A. Perais, A. Seznec, EOLE: Paving the way for an effective implementation of value prediction, in: ACM SIGARCH Computer Architecture News, Orlando, FL, USA, 42, IEEE Press, 2014, pp. 481–492.
[30] A. Perais, A. Seznec, Practical data value speculation for future high-end processors, in: 20th International Symposium on High Performance Computer Architecture (HPCA), Minneapolis, MN, USA, IEEE, 2014, pp. 428–439.
[31] D.T. Pham, M. Castellani, Action aggregation and defuzzification in mamdani-type fuzzy systems, Proc. Inst. Mech. Eng. Part C 216 (7) (2002) 747–759.

[32] A. Prost-Boucle, O. Muller, F. Rousseau, Fast and standalone design space exploration for high-level synthesis under resource constraints, J. Syst. Archit. 60 (1) (2014) 79–93.
[33] C. Radu, M.S. Mahbub, L. Vinţan, Developing domain-knowledge evolutionary algorithms for network-on-chip application mapping, Microprocess. Microsyst. 37 (1) (2013) 65–78.
[34] S. Ricciardi, D. Careglio, G. Santos-boada, J. Solé-pareta, U. Fiore, F. Palmieri, Towards an energy-aware internet: modeling a cross-layer optimization approach, Telecommun. Syst. 52 (2) (2013) 1247–1268.
[35] T.J. Ross, Fuzzy Logic with Engineering Applications, John Wiley & Sons, 2009.
[36] K. Sankaranarayanan, S. Velusamy, M. Stan, K. Skadron, et al., A case for thermal-aware floorplanning at the microarchitectural level, J. Instruct.-Level Parall. 7 (1) (2005) 8–16.
[37] S.R. Sarangi, W.L. Torrellas, Y. Zhou, Reslice: Selective re-execution of long-retired misspeculated instructions using forward slicing, in: 38th Annual IEEE/ACM International Symposium on Microarchitecture, Barcelona, Spain, IEEE Computer Society, 2005, pp. 257–270.
[38] J. Sharkey, D. Ponomarev, K. Ghose, M-sim: A Flexible, Multithreaded Architectural Simulation Environment, Technical Report, Department of Computer Science, State University of New York at Binghamton, 2005. CS-TR-05-DP01.
[39] H.F. Sheikh, I. Ahmad, S.A. Arshad, Performance, energy, and temperature enabled task scheduling using evolutionary techniques, Sustain. Comput. Inf. Syst. (2017).
[40] C. Silvano, W. Fornaciari, G. Palermo, V. Zaccaria, F. Castro, M. Martinez, S. Bocchio, R. Zafalon, P. Avasare, G. Vanmeerbeeck, et al., Multicube: Multi--objective design space exploration of multi-core architectures, in: VLSI 2010 Annual Symposium, Lixouri, Kefalonia, Greece, Springer, 2011, pp. 47–63.
[41] M. Thompson, A.D. Pimentel, Exploiting domain knowledge in system-level MPSoc design space exploration, J. Syst. Archit. 59 (7) (2013) 351–360.
[42] J. Vatjus-Anttila, J. Kreku, J. Korpi, S. Khan, J. Saastamoinen, K. Tiensyrjä, Early-phase performance exploration of embedded systems with ABSOLUT framework, J. Syst. Archit. 59 (10) (2013) 1128–1143.
[43] L. Vintan, Degrees of contradiction for fuzzy logic rules implementing computer architecture ontologies, Revista Română de Informatică şi Automatică 23 (3) (2013) 23–26.
[44] L. Vintan, R. Chis, M.A. Ismail, C. Cotofana, Improving computing systems automatic multiobjective optimization through meta-optimization, IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 35 (7) (2016) 1125–1129.
[45] K. Wang, M. Franklin, Highly accurate data value prediction using hybrid predictors, in: 30th Annual ACM/IEEE International Symposium on Microarchitecture, Research Triangle Park, NC, USA, IEEE Computer Society, 1997, pp. 281–290.
[46] L.A. Zadeh, Information and control, Fuzzy Sets 8 (3) (1965) 338–353.
[47] E. Zitzler, D. Brockhoff, L. Thiele, The hypervolume indicator revisited: On the design of Pareto-compliant indicators via weighted integration, in: Evolutionary Multi-Criterion Optimization, Springer, 2007, pp. 862–876.