Microprocessors and Microsystems xxx (2015) xxx–xxx

Contents lists available at ScienceDirect

Microprocessors and Microsystems

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

# Fuzzy logic based energy and throughput aware design space exploration for MPSoCs

Muhammad Yasir Qadri [a,*], Nadia N. Qadri [b], Klaus D. McDonald Maier [a]

[a] University of Essex, CO4 3SQ Colchester, United Kingdom
[b] COMSATS Institute of Information Technology, Wah Campus, Pakistan

ARTICLE INFO

*Article history:*
Available online xxxx

*Keywords:*
Design space exploration
Multicore processing
Fuzzy logic
Energy efficiency

ABSTRACT

Multicore architectures were introduced to mitigate the issue of increase in power dissipation with clock frequency. Introduction of deeper pipelines, speculative threading etc. for single core systems were not able to bring much increase in performance as compared to their associated power overhead. However for multicore architectures performance scaling with number of cores has always been a challenge. The Amdahl's law shows that the theoretical maximum speedup of a multicore architecture is not even close to the multiple of number of cores. With less amount of code in parallel having more number of cores for an application might just contribute in greater power dissipation instead of bringing some performance advantage. Therefore there is a need of an adaptive multicore architecture that can be tailored for the application in use for higher energy efficiency. In this paper a fuzzy logic based design space exploration technique is presented that is targeted to optimize a multicore architecture according to the workload requirements in order to achieve optimum balance between throughput and energy of the system.

## 1. Introduction

Traditional design flows for architecture exploration rely heavily either on earlier experiences or domain level expertise of the system architect. This can often result in sub-optimal designs, consequently resulting in sub-optimal performance. Since the scaling of technology, and the advent of multicore architectures has resulted in design spaces that are larger and too complex to be handled by intuitive methods. Consequently, there arises a need of automatic and intelligent schemes to systematically explore the design parameters to find an optimum balance in terms of a particular design goal (such as Energy and Throughput) for specific applications [1]. This process is termed as Design Space Exploration (DSE) [2]. A Design Space is composed of two main parts, i.e. (1) Problem Space and (2) Solution Space. The Problem Space is defined as the parameters that are not ultimate design objectives but rather natural characteristics of the design space effecting the performance metric, whereas the Solutions Space represents the primary objectives of the DSE process, e.g. throughput and energy consumption.

Multicore architectures are rapidly emerging as an important design paradigm for both high performance and embedded processing. These architectures have often been investigated and designed in order to achieve a greater throughput combined with reduced energy consumption [3]. However several issues related to resource sharing on the chip can have a negative impact on the performance of an application and therefore may result in decreased performance [4]. A large body of research now focuses on reconfigurable multicore architectures in order to support DSE algorithms to find optimal solutions for improved energy and throughput balance [5–7]. As a result of on-going research several online and offline DSE techniques and algorithm have been proposed for hardware adaptation [8–10].

Generally, offline DSE techniques aim to optimize the system at design time and therefore contain strategies that are not suitable for runtime implementation for one or more of the following interdependent reasons: (1) requirement of a large number of iterations through simulations, (2) higher complexity, and (3) involvement of large amount of calculations. This paper presents a novel fuzzy logic based design space exploration scheme [11–13] suitable for online application, targeted for Multiprocessor System on-Chips (MPSoCs) to find an optimum balance between power and performance. The main contributions of this work are as follows:

* Corresponding author.
*E-mail address:* yasirqadri@acm.org (M.Y. Qadri).

http://dx.doi.org/10.1016/j.micpro.2015.08.001
0141-9331/© 2015 Elsevier B.V. All rights reserved.

- A fuzzy logic-based robust DSE technique that can be implemented at runtime for reconfigurable architectures.
- A proposed expert system based (i.e. using expert knowledge-based) scheme for energy efficiency that is independent of sophisticated analytical models.
- A proposed technique that does not require a large number of iterations, and the solution converges in less than 5 iteration for most of the cases shown.

Overall, this paper offers insights on the use of Fuzzy Logic for MPSoC design space exploration to strike a balance on energy consumption and throughput and validates its robustness through analysis of results using multicore benchmarks. Advantages of applying fuzzy control in DSE process are, that it is parameter insensitive, provides fast convergence, accepts noisy and inaccurate signals, and can produce overall good if not the optimal results [14]. Given the importance of energy efficiency in current and future multicore systems, these contributions provide an alternative and straightforward approach to address the issue.

The remainder of the paper is divided into four sections. The following section provides a comparison and review of related wok. The Section 3 gives an overview of the proposed system architecture including the target MPSoC, the Fuzzy Logic based Design Space Exploration (DSE) Engine, benchmarks used and simulation setup. The Section 4 discusses the results, and Section 5 concludes the paper.

## 2. Related work

In this section, research work related to the proposed design space exploration methodology is discussed and compared. Generally, the DSE techniques can be classified into three main categories according to the design space search criterion [2]: i.e. (1) Exhaustive evaluation of every design point, (2) Random search, and (3) Heuristic search mechanisms involving knowledge of the design space. The result of the above mentioned search techniques, i.e. the candidate configurations are then analyzed using simulations or analytical models.

1. **Exhaustive evaluation of every design point:** This technique involves evaluation of all possible combination of problem space. This method is useful only in situations where the size of design space is very small, for large designs this method is prohibitive due to the latency involved in such unguided search processes. The examples of exhaustive search based DSE are discussed in [15–17].
2. **Random search:** Random search is highly desirable where exhaustive search is not possible due to large design space. Schemes based on Monte Carlo approximations [18], Simulated Annealing [19,20], and Tabu Search [21,22] fall under this category.
3. **Heuristic search mechanisms involving knowledge of the design space:** These strategies involve guided exploration using knowledge of characteristics of design space to improve convergence towards final solution. Examples of such techniques include, but are not limited to, Fuzzy Logic, Markov Decision Process (MDP) [1,23], Genetic [24,25] and Evolutionary Algorithms [26,27]. The fuzzy logic based DSE presented in this paper can be classified into this category. However in order to compare a wider range of techniques with the presented one an overview of other classes of DSE are also discussed as follows.

Exhaustive search mechanisms thoroughly investigate the design space by exploring all the possible configurations. Baghdadi et al. [15] present a DSE technique based on high-level simulation to evaluate the dynamic behaviors of system configurations of a complex architecture. However, due to the higher level of simulation, timing information is not accounted for. Hence additionally, a back-annotation approach based on the RTL (Register Transfer Level) analysis of some implementations allows to extract timing elements needed for performance estimation of all feasible implementations.

Another example of exhaustive search is applied by Monchiero [28] where a target architecture of a configurable number of cores, L2 Cache size, and processor issue width, is analyzed on the basis of power consumption, throughput, and thermal effects. Based on this analysis, further evaluations were performed using various chip floorplans to minimize and study chip temperature effects.

As mentioned above, the random search based schemes provide a viable alternative to typical exhaustive search mechanisms for large design spaces. Bruni et al. [18] present an unsupervised Monte Carlo based design space exploration scheme. In general, the main characteristic of Monte Carlo methods is the use of random sampling techniques to come up with a solution of the target problem. The random sampling technique has been shown to be one of the best techniques to avoid falling into local minima [29].

Another Monte Carlo search based approach for DSE is the use of Simulated Annealing [19,20]. In Simulated Annealing, a new configuration is formed at each iteration by random displacement. If the cost function of this new configuration is less than the previous one, the change is accepted unconditionally; if not, then it is probabilistically analyzed for further exploration [29].

Kreutz et al. [21] present a Tabu search based DSE scheme. The authors propose an optimization algorithm based on the analysis of Network-on-Chip (NoC) topologies and router architectures to find a trade-off between latency and energy constraints. The key concept behind the Tabu search mechanism is a Tabu list that contains moves that are not allowed or prohibitive, and typically includes the recently visited configurations.

Design exploration based on random search techniques such as Monte Carlo, Simulated Annealing and Tabu Search are robust and the literature reviewed suggest their usefulness in general. However, typically these techniques are applied as an off-line process, and require to collect sufficient statistical information before actually starting the design process.

In contrast to exhaustive search and random search schemes, heuristic exploration based techniques involve knowledge of the design space in the search process. One such example of work is presented by Beltrame et al. [1] that employs Markov Decision Processes (MDP). MDP is a reinforcement learning method in which design states are explored in a decision tree probabilistically according to values that have been learned over time. For initial estimates this search typically produces sub-optimal results. However, over a number of trials, the MDP can be trained to produce high-quality architectures [30]. Beltrame et al. [1] have applied MDP to derive custom Very Long Instruction Width (VLIW) processors for various image and video compression algorithms. The authors have adopted a unique approach based on automated guided search by estimating the impact of varying architectural parameters probabilistically and performing simulations only when the estimates are insufficient. Consequently, the number of expensive simulations is reduced. However, this approach requires the availability of an appropriate estimation methodology, which is not generally present in case of arbitrary Chip Multiprocessor (CMP) architectures [31].

Kang et al. [24] present a DSE framework called Magellan. The proposed framework applies a machine-learning approach for the design space optimization problem, by iterative benchmarks simulations on available processor cores. Their work contain a comparison of single objective algorithms such as (genetic algorithms, ant colony optimization, hill climbing, and random search) against an exhaustive simulation. The search space is reduced by fixing many parameters, but the work is not extended to multi-objective optimization, and explores the design space as single objective problem [32].

Evolutionary Algorithms (EAs) are another alternative approach of knowledge based design space search mechanisms. Erbas et al. [26] present a multiobjective optimization process for maximum processing time, power consumption, and cost of the architecture. The proposed scheme provides the system designer with a set of solutions, rather than a single optimal point. Evolutionary algorithms, evolve over a population rather than a single solution. The presented approach showed good performance over the desired objective matrix. However multiobjective evolutionary algorithms (MOEAs) have some limitations by nature: firstly, MOEAs do not perform well in limited search spaces with a few feasible solutions. Secondly, after convergence, MOEAs saturate and the search process result in repetition of already known solutions [33].

In order to take advantage of guided search mechanisms and to avoid the drawbacks of large number of simulations, this paper presents a fuzzy logic based approach for design space exploration. Fuzzy logic is an expert system based reasoning technique that provides a framework to transform imprecise information into a meaningful output [34]. The use of fuzzy logic for the proposed design space exploration process brings the following advantages: firstly, a detailed model of the system that can describe the behavior of the system in various configurations is not present. The implementation of fuzzy inference systems does not require analytical models. Secondly, the problem space covered in this work is large enough so that a linear model may not be able to precisely depict the behavior of the system. This is by the fact that, generally a model's complexity increases when the system parameters begin to interact in a non-linear fashion [35]. Thirdly, Fuzzy Logic linguistically explains the behavior of the system using *if-then* rules. The fuzzy inference engine converts these rules to their mathematical equivalents, and results in much more accurate representations of the behavior of the system.

Furthermore, Fuzzy logic can handle problems with imprecise and incomplete data sets, and it can model non-linear functions of arbitrary complexity. Therefore the use of fuzzy logic in this work provides a robust, adaptive, and flexible approach to implement complex search mechanisms for balanced energy consumption and throughput of the system.

## 3. System architecture and simulation setup

This section provides an overview of the proposed approach in detail: initially the target platform (i.e. the problem space) of the configurable multicore architecture is introduced. The Sections 3.3 and 3.4 overview the fuzzy logic based DSE, and the benchmark applications used. Finally the simulation setup is explained in Section 3.5.

### 3.1. The problem space: a configurable MPSoC architecture

The proposed experimental MPSoC is comprised of a 16-core symmetric chip multiprocessor platform based on the Intel x86 architecture, employing a shared memory architecture. The platform incorporates L1 and L2 caches with configurable size and associativity, shared with the Modified–Exclusive–Shared–Invalid (MESI) coherence protocol. The number of active cores and the processor frequency/voltage can be adjusted both for energy and throughput regulation. The system configuration parameters are described in Table 1, showing various operating points of the system in terms of voltage, frequency and energy consumption. The energy consumption and voltage/frequency information was obtained from the Intel 486 GX embedded processor datasheet [36].

**Table 1**
System parameters: the design space.

| Parameter | Value |
|---|---|
| Processor type | Intel x86 |
| Number of cores | [1⋯16] |
| Operating frequencies | [16, 20, 25, 33] MHz |
| Operating voltages | [2.0, 2.2, 2.4, 2.7] V |
| Dynamic energy consumption per cycle | [13.1, 15.4, 18.7, 22.9] nJ |
| Technology node | 0.8 μm |

The selection of this particular architecture was mainly due to two reasons: 1. a homogeneous and symmetric architecture was required that is supported by cycle accurate simulators such as MARSSx86 [37] in this case, also 2. using such type of architecture reduces the rule base and membership functions of the proposed Fuzzy Logic based Design Space Exploration (DSE) Engine.

Each core of the system is connected to a CMOS power switch, similar to the one proposed by Kim et al. [38], so that when it is turned off, the leakage energy of the core should not contribute to the overall energy consumption of the system. The default size and associativity for L1 and L2 caches are 8 KB, 4-way set associative; and 128 KB, 8-way set associative as per the original Intel 486 GX processor specifications [36]. Due to the possible variance in timing by changing cache size and associativity, the L1 cache miss penalty was assumed to be uniform at 10 cycles and that for L2 cache was set to 30 cycles. The L1 and L2 cache timing and energy data was obtained from CACTI [39] (see Table 2). Although CACTI provides thorough, near accurate memory access time and energy estimates, but it is not a trace driven simulator, so energy consumption resulting in a number of hits or misses is not accounted for a particular application. Therefore detailed cache energy and throughput models accounting for cache miss ratios were presented by Qadri et al. [40]. It should be noted that these models are not part of the proposed fuzzy logic exploration engine. However these models are required to estimate the impact of various cache configuration on energy consumption of the overall system.

### 3.2. Mathematical models for energy estimation of MPSoC

If $E_{ic}$, $E_{dc}$, and $E_{l2c}$ is the energy consumed by instruction, data and level 2 (L2) cache operations, $E_{misc}$ is the energy consumed by the instructions which do not require data memory access, and $E_{leak}$ the leakage energy of the processor, then the total energy consumption of the code $E_{total}$ in Joules [J] can be defined as

$$E_{total} = E_{ic} + E_{dc} + E_{l2c} + E_{misc.} + E_{leak}$$

where,

**Table 2**
Default L1 and L2 cache description.

| Parameters | L1 cache | L2 cache |
|---|---|---|
| Size (Kbytes) | 8 | 128 |
| Associativity | 4 | 8 |
| Number of blocks | 64 | 1024 |
| Block size (bytes) | 128 | 128 |
| Read/write penalty (cycles) | 10 | 30 |
| Access time (s) | 6.11E−09 | 7.92E−09 |
| Cycle time (s) | 3.32E−09 | 4.38E−09 |
| Total dynamic read energy (J) | 1.27E−08 | 8.53E−08 |
| Total dynamic write energy (J) | 1.52E−09 | 7.17E−09 |

L1 Instruction Cache

$$E_{ic} = E_{ic-read} + E_{ic-mp}$$

$$E_{ic-read} = E_{ic-rcycle} \cdot \eta_{ic-read}$$

$$E_{ic-mp} = E_{cycle} \cdot P_{ic-rmiss} \cdot \eta_{ic-rmiss}$$

L1 Data Cache

$$E_{dc} = E_{dc-read} + E_{dc-write} + E_{dc-mp}$$

$$E_{dc-read} = E_{dc-rcycle} \cdot \eta_{dc-read}$$

$$E_{dc-write} = E_{dc-wcycle} \cdot \eta_{dc-write}$$

$$E_{dc-mp} = E_{cycle} \cdot (P_{dc-rmiss} \cdot \eta_{dc-rmiss} + P_{dc-wmiss} \cdot \eta_{dc-wmiss})$$

L2 Cache

$$E_{l2c} = E_{l2c-read} + E_{l2c-write} + E_{dc-mp} + E_{l2c\to m}$$

$$E_{l2c-read} = E_{l2c-rcycle} \cdot (\eta_{l2c-if} + \eta_{l2c-dread})$$

$$E_{l2c-write} = E_{l2c-wcycle} \cdot \eta_{l2c-dwrite}$$

$$E_{l2c-mp} = E_{cycle} \cdot [P_{l2c-rmiss} \cdot (\eta_{l2c-if} + \eta_{l2c-dread}) + P_{l2c-wmiss} \cdot \eta_{l2c-dwrite}]$$

$$E_{l2c\to m} = E_{l2c\to ram} + E_{l2c\to rom}$$

In the above equations, $E_{x-read}$, $E_{x-write}$, and $E_{x-mp}$ denote the read, write and miss penalty energy of the corresponding cache x (i.e. instruction, data or L2 cache). The read and write cycle energy per cache access is denoted by $E_{x-rcycle}$ and $E_{x-wcycle}$. The number of data read and write transactions of the cache (including all hits and misses) is denoted by $\eta_{x-read}$ and $\eta_{x-write}$. Furthermore $\eta_{l2c-if}$, $\eta_{l2c-dread}$, $\eta_{l2c-dwrite}$ denote the L2 cache's instruction fetch, data read and data write transactions respectively. The processor's per cycle energy consumption is denoted by $E_{cycle}$; $P_{x-rmiss}$, $P_{x-wmiss}$, $\eta_{x-rmiss}$ and $\eta_{(}x - wmiss)$ denote the read/write miss penalty (in terms of number of cycles) and their corresponding miss ratios. The energy consumed in the L2 cache to data and code memory is denoted by $E_{l2c\to ram}$ and $E_{l2c\to rom}$ that could also be calculated by multiplying the number of memory accesses with their read and write cycles energy.

## 3.3. The Fuzzy Logic based Design Space Exploration (DSEE) Engine

Fuzzy logic is considered to be one of the most suitable candidates for bridging the gap between computer and human logic. The ability of fuzzy inference systems to interpret linguistic rules and to defuzzify the results to crisp numbers, without the use of sophisticated mathematical models of the system, has made a case for attaining the target of design space exploration of the proposed MPSoC architecture through the application of fuzzy logic. The proposed MPSoC architecture takes advantage of fuzzy logic based reconfiguration in a closed loop as shown in Fig. 1. It may be noted that the proposed system is based on the principles of a typical feedback control system, the advantage of using fuzzy logic here is that one does not need to model the impact of input parameters over the outputs precisely, as fuzzy systems have a natural ability to handle vague information based on a rule base of linguistic terms. The design and implementation of the proposed system involved the following steps:

1. identification of the input variables,
2. partitioning of variables into fuzzy membership functions,
3. formation of rule base,
4. defuzzification, and
5. C language implementation.

### 3.3.1. Identification of the input variables and output parameters

A number of parameters were identified that can be modified dynamically in a reconfigurable MPSoC and were classified as control parameters for the fuzzy logic based DSEE. These parameters are: L1/L2 Cache Size and associativity, CPU Frequency, and Number of active cores. Since the target architecture is symmetric and homogeneous all the computed parameters through fuzzy logic based DSEE were implemented on all the cores universally, and outputs were observed on an aggregate basis. Based on the input parameters certain quantities were identified that receive a clear impact from these parameters, which include L1/L2 aggregate Cache Miss Ratio, aggregate CPU Throughput, and Energy Consumption.

The identification of output parameters of the system was based on the fact that changing the cache size, and associativity effects the cache miss ratio, CPU throughput and energy consumption of the system [41,40]. Similarly, CPU operating frequency and number of active cores effect overall throughput [42] and energy consumption of the system [43].

### 3.3.2. Partitioning of variables into fuzzy membership functions

In order to fuzzify the input and output variables, each variable was partitioned into three fuzzy subsets which were assigned to their respective membership functions namely $\mu_A$, $\mu_B$, and $\mu_C$; classifying lower, middle and upper bounds of the given variable. For example in Table 3, for L1 and L2 the cache miss ratio varies from 0% to 100%, the membership function A is used to classify a low miss ratio which is defined from 0% to 40%, B a moderate miss ratio from 25% to 75%, and C a high miss ratio from 60% to 100%. A similar approach is adopted for the remainder of the input and output variables, and is detailed in Tables 3 and 4 respectively.

### 3.3.3. Formation of rule base

To establish the relationship between the input variables and output parameters of the SoC, fuzzy logic rules were defined. There is no known standard criteria for the definition of membership functions and/or fuzzy rule base, and any comprehensive

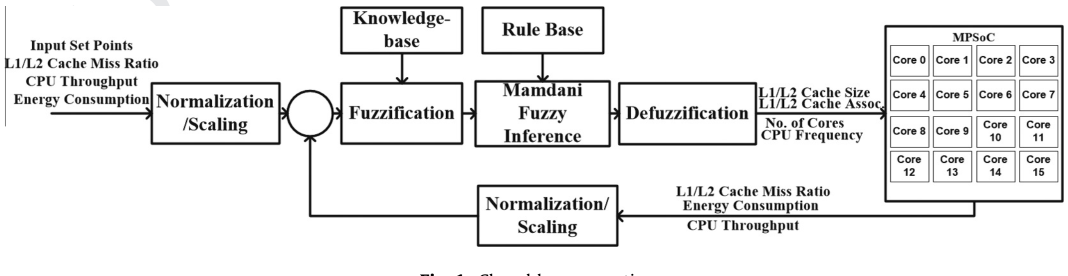
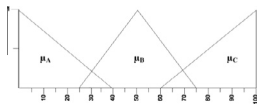
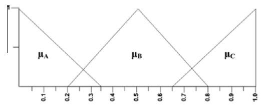
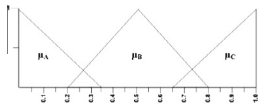


**Fig. 1.** Closed loop operation.

**Table 3**
Fuzzy membership functions for input variables.

| | |
|---|---|
|  L1 and L2 Miss ratio | $\mu_A = \begin{cases} 0 & \textit{if } 40\% \leqslant \text{L1/L2 Miss ratio} \leqslant 0\% \\ \frac{40-x}{40} & \textit{if } 0\% \leqslant \text{L1/L2 Miss ratio} \leqslant 40\% \end{cases}$ <br> $\mu_B = \begin{cases} 0 & \textit{if } 75\% \leqslant \text{L1/L2 Miss ratio} \leqslant 0\% \\ \frac{x-25}{25} & \textit{if } 25\% \leqslant \text{L1/L2 Miss ratio} \leqslant 50\% \\ \frac{75-x}{25} & \textit{if } 50\% \leqslant \text{L1/L2 Miss ratio} \leqslant 75\% \end{cases}$ <br> $\mu_C = \begin{cases} 0 & \textit{if } 100\% \leqslant \textit{if } \text{L1/L2 Miss ratio} \leqslant 60\% \\ \frac{x-60}{40} & \textit{if } 60\% \leqslant \text{L1/L2 Miss ratio} \leqslant 100\% \end{cases}$ |
|  Normalized Throughput | $\mu_A = \begin{cases} 0 & \textit{if } 0.35 \leqslant \text{Throughput} \leqslant 0 \\ \frac{0.35-x}{0.35} & \textit{if } 0 \leqslant \text{Throughput} \leqslant 0.35 \end{cases}$ <br> $\mu_B = \begin{cases} 0 & \textit{if } 0.8 \leqslant \text{Throughput} \leqslant 0 \\ \frac{x-0.2}{0.3} & \textit{if } 0.2 \leqslant \text{Throughput} \leqslant 0.5 \\ \frac{0.8-x}{0.3} & \textit{if } 0.5 \leqslant \text{Throughput} \leqslant 0.8 \end{cases}$ <br> $\mu_C = \begin{cases} 0 & \textit{if } 1.0 \leqslant \text{Throughput} \leqslant 0.65 \\ \frac{x-0.65}{0.35} & \textit{if } 0.65 \leqslant \text{Throughput} \leqslant 1.0 \end{cases}$ |
|  Normalized Energy Consumption | $\mu_A = \begin{cases} 0 & \textit{if } 0.35 \leqslant \text{Energy Consumption} \leqslant 0 \\ \frac{0.35-x}{0.35} & \textit{if } 0 \leqslant \text{Energy Consumption} \leqslant 0.35 \end{cases}$ <br> $\mu_B = \begin{cases} 0 & \textit{if } 0.8 \leqslant \text{Energy Consumption} \leqslant 0 \\ \frac{x-0.2}{0.3} & \textit{if } 0.2 \leqslant \text{Energy Consumption} \leqslant 0.5 \\ \frac{0.8-x}{0.3} & \textit{if } 0.5 \leqslant \text{Energy Consumption} \leqslant 0.8 \end{cases}$ <br> $\mu_C = \begin{cases} 0 & \textit{if } 1.0 \leqslant \text{Energy Consumption} \leqslant 0.65 \\ \frac{x-0.65}{0.35} & \textit{if } 0.65 \leqslant \text{Energy Consumption} \leqslant 1.0 \end{cases}$ |

theory does not exists that explains the method to acquire knowledge for the development of a rule base [44,45]. Also the quality of rule base cannot be evaluated analytically [46]. However one of the widely adopted method for knowledge acquisition is based on human experts [47]. The rules defined by a human expert are conditional statements i.e. *if...then* rules.

The rules for the proposed DSEE were formed in such a way that a balanced throughput and energy consumption ratio can be achieved. For primary or core level configuration the fuzzy logic rules were devised so as to keep track of the average L1 miss ratio, energy consumption and throughput for all the cores and to strive to find an optimum cache size, associativity and operating frequency. The cache size and associativity not only affect the miss ratio but they also have an impact on the throughput and energy consumption of the device. For example first rule states that *if* L1 miss ratio, energy consumption, and throughput are low, it means the L1 cache size and associativity are fine since miss ratio is low, but as throughput is low provided that energy consumption is also low; *then* operating frequency of the cores can be increased in order to improve throughput (see Table B.2).

Similarly for the secondary or system level configuration the fuzzy logic DSEE strives to find an optimal number of cores, L2 cache size and associativity while taking into account the L2 miss ratio and total throughput and energy consumption of the SoC.

### 3.3.4. Defuzzification

The output of the inference system is a fuzzy value and is represented by a membership function. As the MPSoC reconfiguration needs crisp values for the control parameters, the resultant fuzzy values have to be defuzzified. There exist a number of defuzzification methods such as Centroid, Mean of Maxima, and Threshold. The proposed system applies the Centroid method that calculates the centroid or center of gravity (COG) of the area under the membership function, thus the defuzzified value depends on both the size and shape of the membership function, so is a more complete representation of the inference. However due to averaging, the control action is diluted and becomes less sensitive to minor variations. But conversely, this is a very robust process that generates less oscillatory process response [48].

### 3.3.5. C language implementation

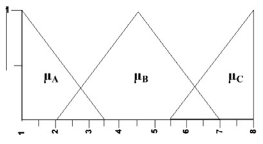
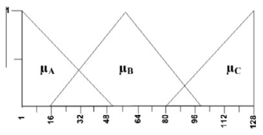
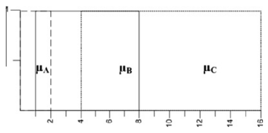
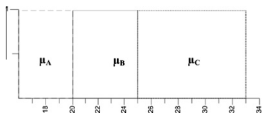
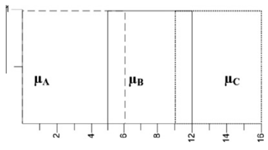
The fuzzy logic engine was implemented using the fuzzy logic API presented in [49] conforming to IEC 61131-7 standard [50]. The Free Fuzzy Logic Library (FFLL) is an open-source fuzzy logic class library which is optimized for faster response and a low processing overhead. The control parameters were fed into the configurable SoC and its response such as L1/L2 cache miss ratios and throughput were observed using MARSSx86 [37] instrumentation, where the energy consumption was calculated using the energy mathematical models presented in Section 3.1 [41,40].

## 3.4. Benchmark applications

As the proposed architecture is comprised of a multicore setup, a set of SPLASH-2 benchmarks [51] is selected to perform the target evaluation. The benchmark applications used for this purpose are described as follows:

- BARNES: The BARNES application simulates the interaction of a system of bodies (galaxies or particles, for example) in three dimensions over a number of time-steps, using the Barnes–Hut hierarchical *N*-body method.
- LU: The LU kernel factors a dense matrix into the product of a lower triangular and an upper triangular matrix. The dense $n \times n$ matrix $A$ is divided into an $N \times N$ array of $B \times B$ blocks ($n = N \times B$) to exploit temporal locality on submatrix elements.
- OCEAN: The OCEAN application studies large-scale ocean movements based on eddy and boundary currents. It uses a red–black Gauss–Seidel multigrid equation solver [52].

**Table 4**
Fuzzy membership functions for output variables.

| | |
|---|---|
|  L1 Cache size | $\mu_A = \begin{cases} 0 & \text{if } 3.5\text{ KB} \leqslant \text{L1 Cache size} \leqslant 1\text{ KB} \\ \frac{3.5-x}{3.5} & \text{if } 1\text{ KB} \leqslant \text{L1 Cache size} \leqslant 3.5\text{ KB} \end{cases}$ <br> $\mu_B = \begin{cases} 0 & \text{if L1 Cache size} \leqslant 2\text{ KB } \mathit{or} \geqslant 7\text{ KB} \\ \frac{x-2}{2.5} & \text{if } 2\text{ KB} \leqslant \text{L1 Cache size} \leqslant 7\text{ KB} \\ \frac{7-x}{2.5} & \text{if } 4.5\text{ KB} \leqslant \text{L1 Cache size} \leqslant 7\text{ KB} \end{cases}$ <br> $\mu_C = \begin{cases} 0 & \text{if L1 Cache size} \leqslant 5.5\text{ KB } \mathit{or} \geqslant 8\text{ KB} \\ \frac{x-5.5}{2.5} & \text{if } 5.5\text{ KB} \leqslant \text{L1 Cache size} \leqslant 8\text{ KB} \end{cases}$ |
|  L2 Cache size | $\mu_A = \begin{cases} 0 & \text{if L2 Cache size} \leqslant 1\text{ KB } \mathit{or} \geqslant 50\text{ KB} \\ \frac{50-x}{50} & \text{if } 1\text{ KB} \leqslant \text{L2 Cache size} \leqslant 50\text{ KB} \end{cases}$ <br> $\mu_B = \begin{cases} 0 & \text{if L2 Cache size} \leqslant 20\text{ KB } \mathit{or} \geqslant 100\text{ KB} \\ \frac{x-20}{40} & \text{if } 20\text{ KB} \leqslant \text{L2 Cache size} \leqslant 60\text{ KB} \\ \frac{100-x}{40} & \text{if } 60\text{ KB} \leqslant \text{L2 Cache size} \leqslant 100\text{ KB} \end{cases}$ <br> $\mu_C = \begin{cases} 0 & \text{if L2 Cache size} \leqslant 80\text{ KB } \mathit{or} \geqslant 128\text{ KB} \\ \frac{x-80}{48} & \text{if } 80\text{ KB} \leqslant \text{L2 Cache size} \leqslant 128\text{ KB} \end{cases}$ |
|  L1/L2 Cache Associativity | $\mu_A = \begin{cases} 0 & \text{if L1/L2 Cache Associativity} \leqslant 0 \mathit{ or} \geqslant 2 \\ 1 & \text{if } 0 \leqslant \text{L1/L2 Cache Associativity} \leqslant 2 \end{cases}$ <br> $\mu_B = \begin{cases} 0 & \text{if L1/L2 Cache Associativity} \leqslant 1 \mathit{ or} \geqslant 8 \\ 1 & \text{if } 1 \leqslant \text{L1/L2 Cache Associativity} \leqslant 8 \end{cases}$ <br> $\mu_C = \begin{cases} 0 & \text{if L1/L2 Cache Associativity} \leqslant 4 \mathit{ or} \geqslant 16 \\ 1 & \text{if } 4 \leqslant \text{L1/L2 Cache Associativity} \leqslant 16 \end{cases}$ |
|  Operating frequency | $\mu_A = \begin{cases} 0 & \text{if Operating frequency} \leqslant 16\text{ MHz}\mathit{or} \geqslant 20\text{ MHz} \\ 1 & \text{if } 16 \leqslant \text{Operating frequency} \leqslant 20\text{ MHz} \end{cases}$ <br> $\mu_B = \begin{cases} 0 & \text{if Operating frequency} \leqslant 20\text{ MHz } \mathit{or} \geqslant 25\text{ MHz} \\ 1 & \text{if } 20\text{ MHz} \leqslant \text{Operating frequency} \leqslant 25\text{ MHz} \end{cases}$ <br> $\mu_C = \begin{cases} 0 & \text{if Operating frequency} \leqslant 25\mathit{MHz\ or} \geqslant 33\text{ MHz} \\ 1 & \text{if } 25\text{ MHz} \leqslant \text{Operating frequency} \leqslant 33\text{ MHz} \end{cases}$ |
|  Number of cores | $\mu_A = \begin{cases} 0 & \text{if Number of cores} \leqslant 1 \mathit{ or} \geqslant 6 \\ 1 & \text{if } 1 \leqslant \text{Number of cores} \leqslant 6 \end{cases}$ <br> $\mu_B = \begin{cases} 0 & \text{if Number of cores} \leqslant 5 \mathit{ or} \geqslant 12 \\ 1 & \text{if } 5 \leqslant \text{Number of cores} \leqslant 12 \end{cases}$ <br> $\mu_C = \begin{cases} 0 & \text{if Number of cores} \leqslant 10 \mathit{ or} \geqslant 16 \\ 1 & \text{if } 10 \leqslant \text{Number of cores} \leqslant 16 \end{cases}$ |

- WATER: This application evaluates forces and potentials that occur over time in a system of water molecules. It imposes a uniform 3-D grid of cells on the problem domain, and uses an $O(n)$ algorithm. The movement of molecules into and out of cells causes cell lists to be updated, resulting in communication.
- FMM: Like Barnes, the FMM application also simulates a system of bodies over a number of time steps. However, it simulates interactions in two dimensions using a different hierarchical *N*-body method called the adaptive Fast Multipole Method [53].

### 3.5. Simulation setup

The proposed architecture was simulated on a cycle accurate full system simulator, the Micro-ARchitectural and System Simulator for x86-based Systems (MARSSx86) [37] which facilitates instruction level simulations and is capable of running unmodified operating systems such as Linux, and Windows XP virtually on the target platforms. The simulator is targeted to provide a fairly accurate timing profile, but at present does not support energy profiling of the target system. MARSSx86 provides a reasonably accurate cache profiling utility, making it well-suited for memory system research. An x86 based 16-core system was defined with each core having private L1 cache and coupled with a single shared L2 cache.

Ubuntu Linux with kernel version 2.6.31.4 was chosen as the target OS due to the inherent multicore support provided in Linux, and availability of its port in Marssx86 Simulator. Also Advanced Configuration and Power Interface (ACPI) enabled operating systems such as Linux support hot-plugging (i.e. turning on/off) of a CPU core on-the-go which is a vital feature for reconfigurable MPSoC scenarios such as the one presented here. The instruction execution, and cache hit/miss information was instrumented through MARSSx86 [37]. Interconnect network energy information was gathered by using Orion [54], cache energy and timing information was gathered by using CACTI [39], and finally MPSoC's total energy was calculated as the sum of interconnect energy, cache energy, and each processor core energy. The processor core energy information was obtained from the Intel 486 GX embedded processor datasheet [36], whereas the cache energy was calculated using the mathematical models presented in [40]. All the applications were sampled for the whole execution cycle of the application and then reconfiguration was carried out based on the decisions made by the fuzzy logic DSE engine. The applications were re-executed for each iteration, so as to observe a clear

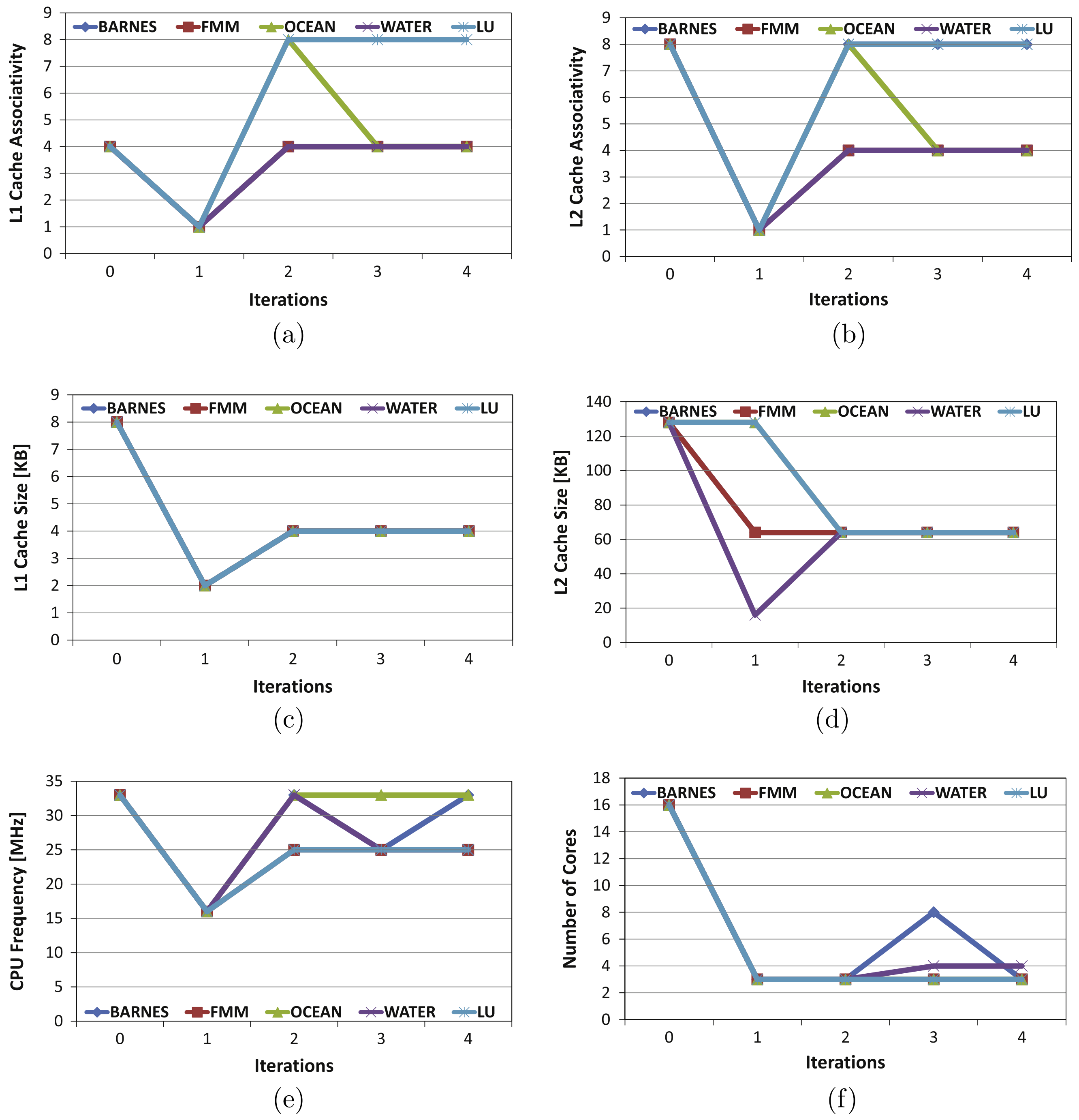


**Fig. 2.** Fuzzy logic based DSEE results for (a) L1 cache associativity, (b) L2 cache associativity, (c) L1 cache size, (d) L2 cache size, (e) CPU frequency, and (f) number of cores for various iterations.

impact of cache reconfiguration and CPU scaling on the energy consumption and throughput of the MPSoC.

## 4. Results

As the main emphasis of the proposed DSE process is to have a balance between throughput and energy consumption of the SoC. In order to achieve this, the DSE Engine based on data of unoptimized cores (iteration 0) starts modifying the configurable parameters, i.e. Number of Cores, Operating Frequency, L1 Cache Size and Associativity (see Fig. 2). The DSE Engine completed the system configuration in five iterations and results were found unvarying for all the subsequent iterations. The impact of these iterative configurations on individual parameters is discussed as follows.

### 4.1. L1 cache

The L1 cache size and associativity play an important role in determining the throughput and energy consumption of an MPSoC. Theoretically, an infinitely large cache with the highest associativity is the best option to bring down the miss ratio, which is the major cause of processor stalls, i.e. the main contributor towards energy wastage and throughput loss. However increasing the size and associativity of the cache not only increases the latency but also the energy consumption.

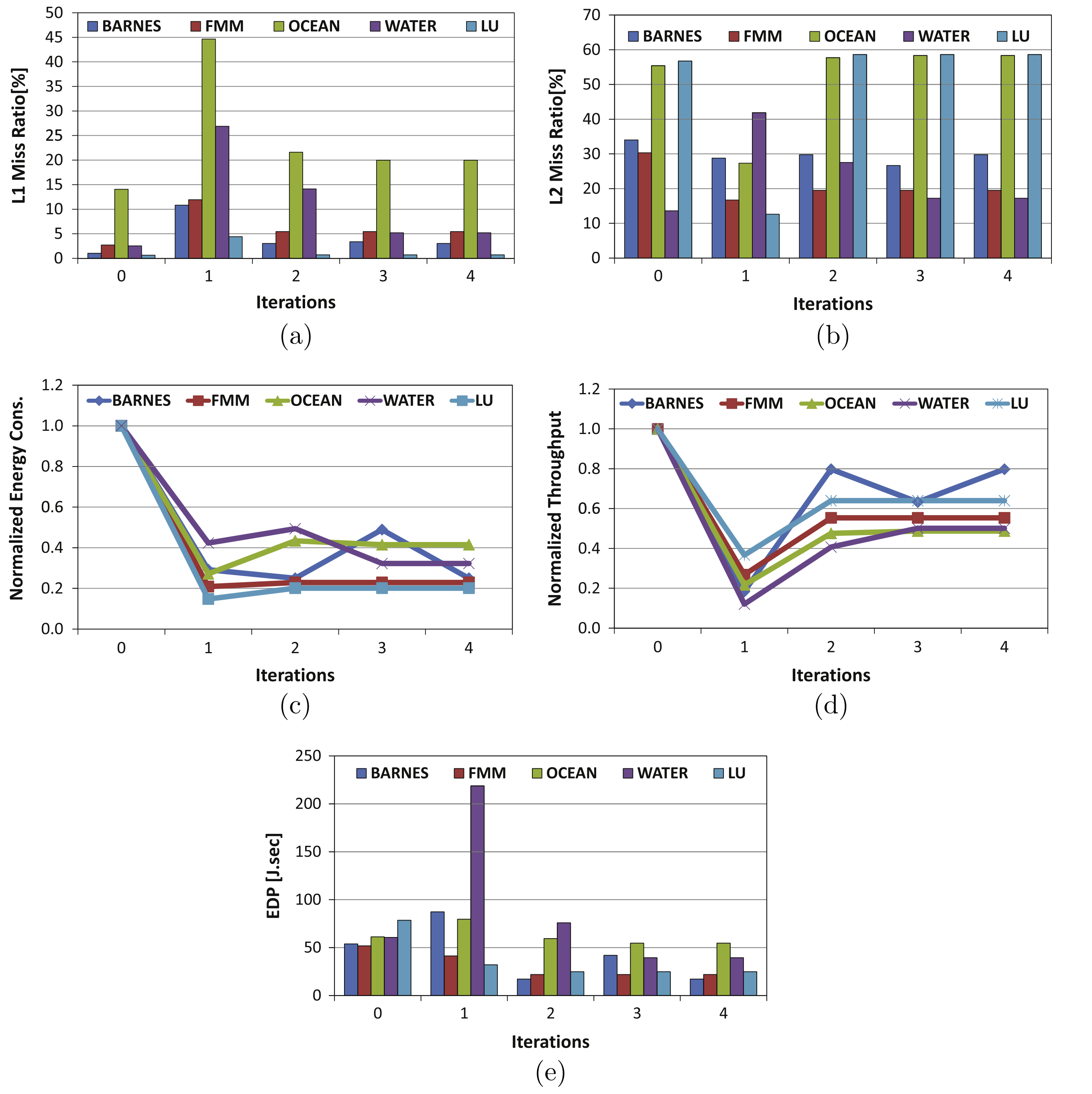


**Fig. 3.** Impact of optimizations by fuzzy logic based DSEE on (a) L1 miss ratio, (b) L2 miss ratio, (c) normalized energy consumption, (d) normalized throughput, and (e) energy delay product for MPSoC.

For the default 16-core configuration, the energy consumption and throughput were maximum, therefore in order to achieve maximum energy savings the fuzzy logic DSE Engine reduced the cache size and associativity to 2 KB and direct-mapped in first iteration for all the benchmark applications, given lower L1 miss ratios (Refer Figs. 2a,c and 3a, and Table 3.

These modifications resulted in a higher miss ratio for the next iteration (i.e. Iteration-1) for OCEAN benchmark and for remainder of the applications a moderately lower raise in miss ratio is observed. Consequently, the L1 cache associativity was increased to 8-way for OCEAN. Also for the LU benchmark, the energy consumption was found to be the lowest thus giving the DSE Engine a chance to gain throughput by increasing the cache associativity to 8-way. For the remainder of the applications cache associativity was increased to 4-way. The cache size for all the applications was also increased to 4 KB. The impact of these change resulted in drop of miss ratio for all the applications as compare to iteration 1 (see Fig. 3a). Particularly for OCEAN the miss ratio dropped from 45% to 22%. For the next two iterations further optimization of cache associativity was undertaken for the OCEAN application and it was set to 4-way, keeping miss ratio, energy and throughput almost the same as for the previous iterations.

It must be noted that the L1 miss ratio is calculated on an aggregate basis for all the CPU cores in the MPSoC by averaging the individual miss ratios for the purpose of simplification. Thus the same average miss ratio for fewer number of cores can be greater if considered on an individual basis. However as the cache size is reduced, the cache throughput is increased and energy

**Table 5**
Timing and energy consumption of various cache configurations.

| Cache size (KB) | Associativity | Blocks | Access time (nsec) | Cycle time (nsec) | Read energy (J) | Write energy (J) |
|---|---|---|---|---|---|---|
| 2 | 1 | 32 | 4.74 | 3.06 | 3.186e–09 | 6.57e–10 |
| 2 | 2 | 16 | 5.92 | 3.00 | 5.81e–09 | 9.5e–10 |
| 4 | 4 | 16 | 6.29 | 3.20 | 1.2e–08 | 1.57e–09 |
| 4 | 8 | 8 | 6.40 | 3.17 | 2.28e–08 | 2.64e–09 |
| 8 | 8 | 16 | 6.44 | 3.39 | 3.367e–08 | 3.04e–9 |
| 16 | 1 | 256 | 5.77 | 3.75 | 9.16e–09 | 2.19e–09 |
| 32 | 1 | 512 | 6.19 | 4.11 | 1.30e–08 | 3.31e–09 |
| 64 | 1 | 1024 | 6.72 | 3.7 | 1.54e–08 | 4.70e–09 |
| 64 | 4 | 256 | 8.61 | 4.25 | 4.18e–08 | 6.98e–09 |
| 64 | 8 | 128 | 8.70 | 4.25 | 7.57e–08 | 8.74e–09 |
| 128 | 1 | 2048 | 7.93 | 4.25 | 2.13e–08 | 6.30e–09 |
| 128 | 8 | 256 | 9.73 | 3.86 | 7.87e–08 | 1.21e–08 |

consumption is significantly decreased. This phenomenon can be observed from Table 5, which contains the different cache configurations used and their timing and energy consumption information based on CACTI [39].

### 4.2. L2 cache

The proposed MPSoC architecture includes a shared level 2 (L2) uniform cache memory. For the L2 cache the Fuzzy logic based DSE Engine reduced the L2 Cache associativity to direct-mapped for all the applications, and the cache sizes for BARNES, FMM and LU were set as 64 KB. For OCEAN, and LU the cache size was kept the same as default 128 KB, as the default configuration has resulted in higher miss ratio of around 55% and 57% respectively. For WATER the default configuration resulted in a lower miss ratio of 15%, therefore the cache size was reduced to 16 KB in the first iteration. These changes in configuration resulted in significant reduction in miss ratios for all the applications, except WATER, where the miss ratio actually increased from 15% to 42%. It must also be noted that for the first iteration, the number of cores was also reduced for all the applications, therefore the reduction in cache size and associativity did not increase the miss ratio for most of the applications, but have a negative impact (see Fig. 3b).

For iteration-2, the size and associativity of L2 cache for WATER was increased to 64 KB and 4-way. The reduction in miss ratios from 55% to 28% for OCEAN, and 57% to 13% for LU application gave a chance to the DSE Engine to reduce L2 size to 64 KB, and to compensate any further increase in miss ratio, L2 associativity was raised to 8-way. For BARNES application L2 associativity was also increased to 8-way while maintaining its size, this is due to the fact that BARNES showed a smaller decrease in L2 miss ratio i.e. from 33% to 22%.

Taking the same approach and keeping track of energy consumption and throughput of the overall system, the DSE Engine maintained the cache sizes for the next two iterations for all the applications and changed the associativity to achieve an optimal value of Energy-Delay Product (EDP). It must be noted that lowering the miss ratio is not the main purpose of the proposed DSE Engine, but a lower EDP for an application is the desired outcome here. Therefore for the OCEAN application, a slight increase of miss ratio from 55% to 58%, and 57% to 59% for LU is observed while comparing the initial and final configuration, but for the remainder of the applications a decrease in miss ratio can be observed.

### 4.3. CPU frequency and number of cores

The MPSoC's operating frequency not only influences the throughput but also its energy consumption, i.e. the higher the frequency, the greater the energy consumption. Therefore the proposed Fuzzy logic based DSE Engine addresses the energy and throughput of the system holistically, both at the same time. To obtain a clearer picture on interdependence of energy and throughput Energy-Delay Product (EDP) is chosen as a metric to analyze the performance of the DSE Engine (see Fig. 3e). The Fuzzy logic based DSE Engine configures the CPU operating frequency to 16 MHz and the Number of Cores to 3 for all benchmarks as compared to the default 33MHZ, 16-Core design (see Fig. 2e and f). These changes resulted in an overall decrease in both energy and throughput for all applications. However if the EDP is analyzed, BARNES, OCEAN, and WATER show an increase in EDP. Therefore in order to bring down the EDP, the DSE Engine increases the Operating Frequency to 25 MHz for FMM and LU; and 33 MHz for remainder of the applications. The number of cores were kept the same for iteration-2. These changes resulted in a decrease in EDP for all applications. It was observed that the OCEAN benchmark showed a nominal decrease in EDP as compared to the default configuration, i.e. 61–59%. therefore the DSE Engine kept the Operating frequency for OCEAN at 33 MHz and for others it was set to 25 MHz. The Number of Cores for BARNES, and WATER were changed to 8 and 4 respectively and for the remainder it was kept constant. When analyzed on the basis of Energy and Throughput for Iteration-3, the BARNES application shows an increase in Normalized Energy Consumption from 0.25 to 0.49, and a decrease in Normalized Throughput i.e. from 0.8 to 0.63. Therefore for the final iteration (i.e. Iteration-4) the DSE Engine increases the CPU Frequency for BARNES to 33 MHz and the Number of Cores were decreased to 4. The impact of the final iteration was an overall decrease of EDP for all applications.

In summary, the application of the proposed Fuzzy Logic based DSE Engine demonstrated the following characteristics:

1. An overall decrease in EDP for all the applications is observed. However, it must be noted that for some benchmarks, the resulting decrease is quite small, if not insignificant.
2. The Energy and Throughput of the system was given a higher priority over the miss ratio while defining the Fuzzy rule base. Therefore after attaining a certain saturation point (i.e. upon achieving lower EDP) Miss Ratio was considered to be the candidate for optimization.
3. The over all reduced priority to miss ratio resulted in its increase for some cases.

Therefore it can be concluded that the proposed Fuzzy Logic based DSE Engine can produce an overall good performance where no detailed mathematical model is present. It should be noted that inherently the fuzzy logic does not necessarily produce optimal solutions but is applied where an overall good performance is required.

## 5. Conclusion

In this paper a novel Fuzzy Logic based Design Space Exploration (DSE) Engine was presented. The proposed DSE Engine was used to find an overall good balance between the energy consumption and performance of the system. To evaluate the proposed scheme, an Intel x86 based multicore SoC with 16 processor cores and a shared memory architecture was simulated using the MARSSx86 full system simulator. A detailed analysis of core, cache, and interconnect power consumption was conducted and an overall decrease in Energy-Delay Product (EDP) was observed. However, as the EDP was given a higher priority while defining the fuzzy rule base, in some cases, an increase in cache miss ratio could be observed.

The system in general showcased the capability and usefulness of the proposed Fuzzy Logic based DSE Engine; therefore this technique can be adapted for a variety of architectures to search for a good compromise for throughput and energy under user defined constraints. The proposed MPSoC architecture and simulation setup can be tailored for use in variety of applications such as NoC research, dynamic thread scheduling, operating system development and high performance computing.

## Appendix A. Supplementary material

Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.micpro.2015.08.001.These data include MOL files and InChiKeys of the most important compounds described in this article.

## References

[1] G. Beltrame, L. Fossati, D. Sciuto, Decision-theoretic design space exploration of multiprocessor platforms, IEEE Trans. Comput. Aid. Des. Integr. Circ. Syst. 29 (7) (2010) 1083–1095.
[2] M. Gries, Methods for evaluating and covering the design space during early design development, VLSI J. Integr. 38 (2) (2004) 131–183.
[3] E. Seo, J. Jeong, S. Park, J. Lee, Energy efficient scheduling of real-time tasks on multicore processors, IEEE Trans. Parallel Distrib. Syst. 19 (11) (2008) 1540–1552.
[4] S. Zhuravlev, S. Blagodurov, A. Fedorova, Addressing shared resource contention in multicore processors via scheduling, ACM SIGARCH Computer Architecture News, vol. 38, ACM, 2010, pp. 129–142.
[5] K. Compton, S. Hauck, Reconfigurable computing: a survey of systems and software, ACM Comput. Surv. (csuR) 34 (2) (2002) 171–210.
[6] T. Marescaux, A. Bartic, D. Verkest, S. Vernalde, R. Lauwereins, Interconnection networks enable fine-grain dynamic multi-tasking on FPGAs, Field-Program. Logic Appl.: Reconfig. Comput. Going Mainstream 2438/2002 (2002) 741–763.
[7] J. Resano, D. Mozos, D. Verkest, F. Catthoor, A reconfigurable manager for dynamically reconfigurable hardware, Des. Test Comput., IEEE 22 (5) (2005) 452–460.
[8] H. Esbensen, E.S. Kuh, Design space exploration using the genetic algorithm, in: Circuits and Systems, 1996. ISCAS'96., Connecting the World., 1996 IEEE International Symposium on, vol. 4, IEEE, 1996, pp. 500–503.
[9] G. Ascia, V. Catania, A.G. Di Nuovo, M. Palesi, D. Patti, Efficient design space exploration for application specific systems-on-a-chip, J. Syst. Architect. 53 (10) (2007) 733–750.
[10] V. Soteriou, L.-S. Peh, Design-space exploration of power-aware on/off interconnection networks, in: Proceedings of IEEE International Conference on Computer Design: VLSI in Computers and Processors, 2004, ICCD 2004, IEEE, 2004, pp. 510–517.
[11] M.Y. Qadri, K.D. McDonald Maier, A fuzzy logic reconfiguration engine for symmetric chip multiprocessors, in: 2010 International Conference on Complex, Intelligent and Software Intensive Systems, IEEE Computer Society, Washington, Krakow, Poland, 2010, pp. 937–943.
[12] M.Y. Qadri, K.D. McDonald Maier, A fuzzy logic based dynamic reconfiguration scheme for optimal energy and throughput in symmetric chip multiprocessors, in: NASA/ESA Conference on Adaptive Hardware and Systems (AHS), 2010, IEEE, Anaheim California, USA, 2010, pp. 333–339.
[13] M.Y. Qadri, K.D. McDonald Maier, N.N. Qadri, Energy and throughput aware fuzzy logic based reconfiguration for MPSoCs, J. Intell. Fuzzy Syst. 26 (1) (2014) 101–113.
[14] M. Simoes, B. Bose, R. Spiegel, Fuzzy logic based intelligent control of a variable speed cage machine wind generation system, IEEE Trans. Power Electron. 12 (1) (1997) 87–95.
[15] A. Baghdadi, N. Zergainoh, W. Cesario, T. Roudier, A. Jerraya, Design space exploration for hardware/software codesign of multiprocessor systems, in: Proceedings of the 11th International Workshop on Rapid System Prototyping, 2000, RSP 2000, IEEE, 2000, pp. 8–13.
[16] S. Blythe, R. Walker, Efficient optimal design space characterization methodologies, ACM Trans. Des. Autom. Electron. Syst. (TODAES) 5 (3) (2000) 322–336.
[17] K. Lahiri, A. Raghunathan, S. Dey, System-level performance analysis for designing on-chip communication architectures, IEEE Trans. Comput. Aid. Des. Integr. Circ. Syst. 20 (6) (2001) 768–783.
[18] D. Bruni, A. Bogliolo, L. Benini, Statistical design space exploration for application-specific unit synthesis, in: Proceedings of Design Automation Conference, 2001, IEEE, 2001, pp. 641–646.
[19] D. Gajski, F. Vahid, S. Narayan, J. Gong, SpecSyn: an environment supporting the specify-explore-refine paradigm for hardware/software system design, IEEE Trans. Very Large Scale Integr. VLSI Syst. 6 (1) (1998) 84–100.
[20] H. Orsila, T. Kangas, E. Salminen, T. Hamalainen, Parameterizing simulated annealing for distributing task graphs on multiprocessor SoCs, in: International Symposium on System-on-Chip, 2006, IEEE, 2006, pp. 1–4.
[21] M. Kreutz, C. Marcon, L. Carro, F. Wagner, A. Susin, Design space exploration comparing homogeneous and heterogeneous network-on-chip architectures, in: Proceedings of the 18th Annual Symposium on Integrated Circuits and System Design, ACM, 2005, pp. 190–195.
[22] B. Xin, J. Chen, J. Zhang, L. Dou, Z. Peng, Efficient decision makings for dynamic weapon-target assignment by virtual permutation and tabu search heuristics, IEEE Trans. Syst. Man Cybern., Part C: Appl. Rev. 40 (6) (2010) 649–662.
[23] G. Shani, Task-based decomposition of factored POMDPs, IEEE Trans. Cybern. 44 (2) (2014) 208–216.
[24] S. Kang, R. Kumar, Magellan: a search and machine learning-based framework for fast multi-core design space exploration and optimization, in: Proceedings of the Conference on Design, Automation and Test in Europe, ACM, 2008, pp. 1432–1437.
[25] K. Nag, T. Pal, N. Pal, ASMiGA: an archive-based steady-state micro genetic algorithm, IEEE Trans. Cybern.
[26] C. Erbas, S. Cerav-Erbas, A. Pimentel, Multiobjective optimization and evolutionary algorithms for the application mapping problem in multiprocessor system-on-chip design, IEEE Trans. Evol. Comput. 10 (3) (2006) 358–374.
[27] J. Liu, W. Zhong, L. Jiao, A multiagent evolutionary algorithm for combinatorial optimization problems, IEEE Trans. Syst. Man Cybern., Part B: Cybern. 40 (1) (2010) 229–240.
[28] M. Monchiero, R. Canal, A. González, Design space exploration for multicore architectures: a power/performance/thermal view, in: Proceedings of the 20th Annual International Conference on Supercomputing, ICS '06, ACM, New York, NY, USA, 2006, pp. 177–186, http://dx.doi.org/10.1145/1183401.1183428.
[29] G. Palermo, C. Silvano, S. Valsecchi, V. Zaccaria, A system-level methodology for fast multi-objective design space exploration, in: Proceedings of the 13th ACM Great Lakes Symposium on VLSI, ACM, 2003, pp. 92–95.
[30] J. Phillips, AC to Register Transfer Level Algorithm Using Structured Circuit Templates: A Case Study with Simulated Annealing, Ph.D. Thesis, Utah State University, 2008.
[31] H. Ishebabi, Architecture Synthesis for Adaptive Multiprocessor Systems on Chip, Ph.D. Thesis, Universitätsbibliothek, 2010.
[32] H. Calborean, R. Jahr, T. Ungerer, L. Vintan, Optimizing a superscalar system using multi-objective design space exploration, in: Proceedings of the 18th International Conference on Control Systems and Computer Science (CSCS), Bucharest, Romania, 2011, pp. 339–346.
[33] M. Lukasiewycz, M. Glaß, C. Haubelt, J. Teich, Efficient symbolic multi-objective design space exploration, in: Proceedings of the 2008 Asia and South Pacific Design Automation Conference, IEEE Computer Society Press, 2008, pp. 691–696.
[34] A. Ibrahim, Fuzzy Logic for Embedded Systems Applications, Butterworth-Heinemann, Newton, MA, USA, 2003.
[35] G.M. Marakas, Modern Data Warehousing, Mining, and Visualization: Core Concepts, Pearson Education, 2002.
[36] Intel, Embedded Ultra-Low Power Intel486? GX Processor Datasheet, Intel Corporation, USA, 1997.
[37] A. Patel, F. Afram, S. Chen, K. Ghose, MARSSx86: a full system simulator for x86 CPUs, in: Design Automation Conference 2011 (DAC'11), 2011.
[38] H.-O. Kim, Y. Shin, H. Kim, I. Eo, Physical design methodology of power gating circuits for standard-cell-based design, in: Proceedings of the 43rd Annual Design Automation Conference, DAC '06, ACM, New York, NY, USA, 2006, pp. 109–112, http://dx.doi.org/10.1145/1146909.1146942.
[39] D. Tarjan, S. Thoziyoor, N. Jouppi, Cacti 4.0, HP Laboratories Palo Alto, Tech. rep. HPL-2006-86 1.
[40] M.Y. Qadri, K.D. McDonald Maier, Analytical evaluation of energy and throughput for multilevel caches, in: 2010 12th International Conference on Computer Modelling and Simulation, IEEE, IEEE Computer Society, Cambridge, UK, 2010, pp. 598–603.
[41] M.Y. Qadri, K.D. McDonald Maier, Data cache-energy and throughput models: design exploration for embedded processors, EURASIP J. Embed. Syst. (2009).
[42] M.D. Hill, M.R. Marty, Amdahl's law in the multicore era, Computer (7) (2008) 33–38.
[43] R. Ge, X. Feng, S. Song, H.-C. Chang, D. Li, K.W. Cameron, Powerpack: Energy profiling and analysis of high-performance systems and applications, IEEE Trans. Parallel Distrib. Syst. 21 (5) (2010) 658–671.
[44] M.H.F. Zarandi, N. Mohammadhasan, S. Bastani, A fuzzy rule-based expert system for evaluating intellectual capital, Adv. Fuzzy Syst. 2012 (2012) 7.
[45] C. Mount, T.W. Liao, Prototype of an intelligent failure analysis system, in: Case-Based Reasoning Research and Development, Springer, 2001, pp. 716–730.
[46] A. Kandel, Fuzzy Expert Systems, CRC press, 1991.
[47] J.E. Buburge, Knowledge Elicitation Tool Classification, Artificial Intelligence Research Group Worcester Polytechnic Institute, 2001.
[48] C. De Silva, Intelligent Control: Fuzzy Logic Applications, CRC Press, USA, 1995.
[49] S. Rabin, AI Game Programming Wisdom, Charles River Media, Inc., Hingham, Massachusetts, 2002.
[50] IEC, International Standard: Programmable Controllers – Part 7: Fuzzy Control Programming, International Electrotechnical Commission, Geneva, Switzerland, iEC Standard, 2000.
[51] S. Woo, M. Ohara, E. Torrie, J. Singh, A. Gupta, The splash-2 programs: characterization and methodological considerations, ACM SIGARCH Computer Architecture News, vol. 23, ACM, 1995, pp. 24–36.
[52] A. Brandt, Multi-level adaptive technique (MLAT) for fast numerical solution to boundary value problems, in: Proceedings of the Third International

Conference on Numerical Methods in Fluid Mechanics, Springer, pp. 82–89.
[53] L. Greengard, The Rapid Evaluation of Potential Fields in Particle Systems, Vol. 1987, MIT Press, 1988.
[54] A.B. Kahng, B. Li, L.-S. Peh, K. Samadi, Orion 2.0: a fast and accurate NoC power and area model for early-stage design space exploration, in: Proceedings of the Conference on Design, Automation and Test in Europe, DATE '09, European Design and Automation Association, 3001 Leuven, Belgium, 2009, pp. 423–428. <http://portal.acm.org/citation.cfm?id=1874620.1874721>.

**Muhammad Yasir Qadri** has obtained his PhD in Electronic Systems Engineering from University of Essex, UK. He has over ten years of practical experience in the development of high-end embedded systems and FPGAs. He is an Approved PhD Supervisor by the Higher Education Commission of Pakistan, and is currently a visiting researcher at University of Essex, UK. He is also a Visiting Faculty Member at HITEC University, Taxila. He is the editor of a recently published book on Multicore Technology by CRC Press, USA, and has two US patent applications in the area of computer architecture. He is also the recipient of a research grant by National ICT R&D Fund, and is the Co-Principal Investigator for the project of development of a reconfigurable multicore architecture. His area of specialization is energy/performance optimization in reconfigurable MPSoC architectures.

**Nadia N. Qadri** is an Associate Professor and Program Coordinator in Department of Electrical Engineering at COMSATS Institute of Information Technology, Wah Campus. He obtained her PhD degree in Electronics Systems Engineering from University of Essex, UK. She has more than 12 years of teaching and research experience at renowned universities of Pakistan. She is also leading a research group of Wireless Networks. She published her research work in more than 34 journals/ conferences and in various international books. She has also edited a book with IGI global. Her research interests includes video streaming, mobile ad hoc and vehicular ad hoc networks, P2P networks, wireless sensor networks, smart grid, self-organized networks, embedded systems, image and video processing. She is also a Professional member of IEEE. She has served on the program committee of more than 20 international conferences.

**Klaus McDonald-Maier** is a Professor in CSEE where he leads the Embedded and Intelligent Systems Research Group. He is actively involved in research in Systems-on-Chip (SoC) architectures and autonomous systems. Specifically, his research concentrates on novel computer and embedded systems techniques and architectures, Embedded Systems and SoC design, development support/debug and technology to increase performance and reliability. He has also worked on hardware and software architectures offering advanced processing power for robotics, image processing and other real-time critical applications under limited power constraints and applied AI techniques for real world problems, robot control and embedded systems. His research work has resulted in more than 150 scientific journal and conference publications as well as 12 pending patents. He has held 5 EPRSC grants with substantial contributions from industry and several EU grants. He is a Fellow of IET, a member of VDE and a senior member of IEEE.