Scalable, energy-aware system modeling and application-specific reconfiguration of MPSoCs with a type-2 fuzzy logic system

Ishfaq Hussain*

Shahid Ali Murtza

Muhammad Yasir Qadri

Martin Fleury

Nadia N. Qadri

*CISTER Research Centre

CISTER-TR-200502

2019/03/01

Scalable, energy-aware system modeling and application-specific reconfiguration of MPSoCs with a type-2 fuzzy logic system

Ishfaq Hussain*, Shahid Ali Murtza, Muhammad Yasir Qadri, Martin Fleury, Nadia N. Qadri

*CISTER Research Centre

Polytechnic Institute of Porto (ISEP P.Porto)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8321159

E-mail: hussa@isep.ipp.pt

https://www.cister-labs.pt

Abstract

This paper demonstrates that interval type-2 Fuzzy Logic Systems (FLS 19s) are suited to Multiprocessor System-on-a-Chip (MPSoC) design modeling because of their ability to handle the uncertainty associated with input parameters. This makes for a scalable modeling process, whereas prior usage of type-1 FLS 19s for modeling inhibited scalability. Specifically, the paper proposes a type-2 Takagi Sugeno Kang (TSK) FLS for modeling MPSoCs built around a multicore processor with shared memory. The paper also presents a TSK FLS reconfiguration methodology in order to dynamically reduce the energy consumption of such a multiprocessor. Modeling of MPSoC 19s by means of the type-2 TSK FLS was found to be effective in terms of a reduction in the Energy Delay Product when applied to five large-scale numerical processing applications on an MPSoC. In fact, experimental results show a significant reduction (by 87 1393% across the applications) in the energy consumption of an MPSoC.

Computers and Electrical Engineering 74 (2019) 292–304

Contents lists available at ScienceDirect

Computers and Electrical Engineering

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

Scalable, energy-aware system modeling and application-specific reconfiguration of MPSocs with a type-2 fuzzy logic system[☆]

Ishfaq Hussain[a], Shahid Ali Murtza[a], Muhammad Yasir Qadri[b], Martin Fleury[b,*], Nadia N. Qadri[c]

[a] HITECH University, Taxila, Pakistan
[b] School of Computer Science and Electronic Engineering, University of Essex, UK
[c] COMSATS Institute of Information Technology, Wah Cantonment, Pakistan

ARTICLE INFO

Article history:
Received 21 November 2017
Revised 27 January 2019
Accepted 28 January 2019
Available online 13 February 2019

Keywords:
Design space exploration
Multiprocessor System-on-a-Chip
Multicore processor
Type 2 Fuzzy Logic System
Reconfiguration

ABSTRACT

This paper demonstrates that interval type-2 Fuzzy Logic Systems (FLS's) are suited to Multiprocessor System-on-a-Chip (MPSoC) design modeling because of their ability to handle the uncertainty associated with input parameters. This makes for a scalable modeling process, whereas prior usage of type-1 FLS's for modeling inhibited scalability. Specifically, the paper proposes a type-2 Takagi Sugeno Kang (TSK) FLS for modeling MPSoCs built around a multicore processor with shared memory. The paper also presents a TSK FLS reconfiguration methodology in order to dynamically reduce the energy consumption of such a multiprocessor. Modeling of MPSoC's by means of the type-2 TSK FLS was found to be effective in terms of a reduction in the Energy Delay Product when applied to five large-scale numerical processing applications on an MPSoC. In fact, experimental results show a significant reduction (by 87–93% across the applications) in the energy consumption of an MPSoC.

1. Introduction

This paper presents a type-2 fuzzy logic based scheme [1] for the run-time reconfiguration of a Multiprocessor System on a Chip (MPSoC) to achieve an optimum balance between throughput and energy. The rule-based Fuzzy Logic System (FLS) is intended to support the dynamic reconfiguration of a multiprocessor platform according to the workload requirements of a particular application. Fuzzy logic is a reasoning mechanism that provides a framework to transform imprecise information arising in an expert system into a meaningful output. Generally, the computational complexity of a system model increases with the interaction of system parameters in a non-linear manner. Therefore, we selected fuzzy logic type-2 rather than type-1 to support a reconfiguration engine, avoiding complex and detailed modeling of the effects on its performance of all the system parameter variations.

In a type-2 FLS, the boundaries of the membership functions are themselves made uncertain, whereas in a type-1 FLS the membership functions' boundaries are fixed. Consequently, the total number of rules in a type-2 FLS rule base is reduced

[☆] Reviews processed and recommended for publication to the Editor-in-Chief by Associate Editor Dr. Tie Qiu.

[*] Corresponding author.
E-mail address: fleum@essex.ac.uk (M. Fleury).

https://doi.org/10.1016/j.compeleceng.2019.01.015

compared to that of a type-1 FLS because the boundaries of the type-2 membership functions are not fixed. Therefore, specific membership functions for every eventuality are no longer required. As a result, the computational overhead during the design space exploration process is reduced with an interval type-2 FLS. Interval type-2 FLS's [2], as considered herein, are a simplification of generalized type-2 FLS's. They make computation more manageable by confining uncertainty to two dimensions. A further advantage of a type-2 FLS is that it is possible to model distortion of the input measurements by time-varying noise (or uncertainty), whereas type-1 FLS's inputs are assumed to only be affected by time-stationary noise. Further consideration of type-2 FLS's can be found in Section 1.

The MPSoC, which is the subject of design exploration with a type-2 FLS (as detailed in Section 6), contains a 16-core multiprocessor with a shared memory. Recently, multicore architectures [3], with an option to reconfigure system resources, have emerged as an important design option for both high-performance and embedded processors. Multicore reconfigurable processor architectures give rise to an improved application-specific performance/power ratio, as they can limit or adapt system resources, according to an application's requirements. These architectures have been designed to achieve a greater throughput with minimal energy consumption. However, due to resource sharing on the processor chip, the performance of an application can suffer. Therefore, researchers have employed several reconfiguration approaches to improve the energy/throughput balance. For example, Givargis et al. [4], proposed an optimization framework named as Platune. The Platune framework employs the concept of parameter independence for characterizing approximate Pareto sets, without the need for an exhaustive search of the entire design space. The drawback of this approach is that the user has to set up parameter independence via a dependency graph because, in the framework, there is no self-tasking for this purpose. Then in another example, Mariani et al. [5] suggested a design space exploration (DSE) technique and a resource management operating system (OS) layer that aims at software/hardware reconfiguration of an industrial multicore platform. However, they used only the number of cores and operating frequency as run-time reconfigurable parameters for an application. Moreover, their multicore platform uses 8-cores, while our suggested platform is a general one, in the sense that it can handle any number of cores.

A number of multicore systems support run-time configuration of voltage and frequency also known as dynamic voltage/frequency scaling (DVFS). To efficiently handle energy consumption and temperature regulation, many multicore systems have been applied several task scheduling, task migration, and core consolidation policies using DVFS. Though an extensive research literature is available on DVFS [6], frequency scaling is not in many cases as efficient as might have been hoped for because, with this methodology, large unused portions of the chip still consume leakage power (unless power gating is employed). In our work, to obtain a better energy and throughput balance, we have employed the idea of level-1 and level-2 cache re-sizing and core switching along with clock or switching frequency scaling. This balance is quantified, herein, by means of the widely-used Energy-Delay Product (EDP) metric [7]. In so doing, the EDP value measures the trade-off between increased delay and lower energy/operation and is not simply a measure of energy consumption.

The proposed scheme has two levels of abstraction: 1) primary level or core level and 2) secondary level or system-on-chip (SoC) level. With this scheme, at core level, the system selects the core frequency/voltage, level-1 (L1) cache size and associativity, depending upon the L1 miss rate, energy consumption and throughput of the core. At SoC level, it reconfigures level-2 (L2) cache size and associativity, along with the number of active cores on the basis of L2 miss rate, energy consumption and throughput of the entire system. Notice that voltage and frequency are interrelated, as reducing the voltage introduces greater circuit delay and, hence, requires a lower switching frequency, and vice versa.

There has also been research in the area of fuzzy-logic-based DSE of multiprocessors [8–11]. Chen et al. [8] employed type-1 fuzzy logic to find suitable cores for a program, and guide program scheduling for Heterogeneous Multicore Processors (HMP's). However, their method is not scalable because the complexity of type-1 fuzzy logic increases exponentially as the number of characteristics increases. Li et al. [9] established fuzzy models for complex MPSoC systems and proposed a real-time type-1 fuzzy scheduling algorithm for this type of system, named as multi-characteristics fuzzy dynamic scheduling (Multi-Fuzzy algorithm). Their algorithm employed linguistic fuzzy sets for modeling various characteristics of tasks and systems. It then evaluated the target priority functions of tasks and resources, based upon the membership rules of those fuzzy sets. For an optimum balance between the power and performance of the system Qadri et al. [10] presented a type-1 fuzzy-logic-based reconfiguration engine targeted at optimizing a symmetric chip multiprocessor (SCMP), according to the workload requirements. Thus, it is apparent that hitherto type-1 fuzzy logic has been the norm, rather than the more flexible type-2 fuzzy logic. This paper demonstrates that the modeling scalability of an existing, traditional Takagi Sugeno Kang (TSK) type-1 FLS [10] can be enhanced by converting to interval type-2 processing.

The rest of the paper is organized as follows. Section 2 is an introduction to type-2 FLSs. In Section 3, we present the type-2 TSK FLS based methodology for DSE. Section 4 discusses the modeling technique utilized for the FLS, whereas Section 5 describes the type-2 FLS membership function modeling. Sections 6 details our simulation methodology with reference to the targeted MPSoC, while Section 7 is an evaluation of the DSE for that MPSoC. Finally, Section 8 highlights the contributions of the research.

2. Fuzzy logic type-2 based reconfiguration engine

Fuzzy logic has been used for configuration optimization and DSE of various platforms. Conceptually, it has the ability to bridge the gap between human thinking, e.g. the design of a computer system by an expert, and computer logic, e.g. automated design actioned by a non-expert. Thus, fuzzy logic was selected by us for the reconfiguration of MPSoC's because

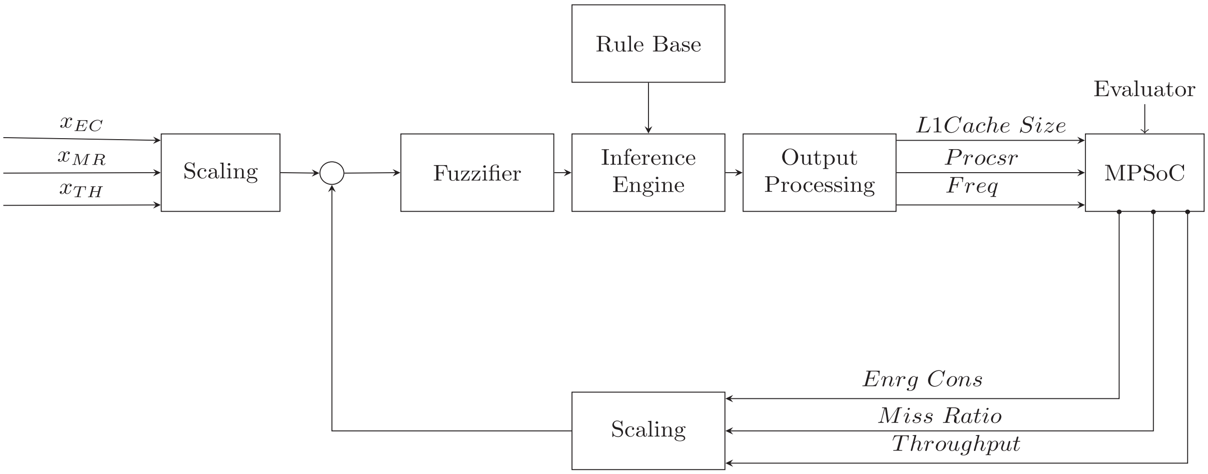

Fig. 1. Closed loop operation of fuzzy type 2 baseddesign space exploration.
Problem Space: • Energy Consumption • Throughput • L1/L2 Miss Ratio
Solution Space: • L1 and L2 Cache Size • L1 and L2 Cache Associativity • Processor frequency • No of Cores (Processors)

it does not require complex mathematical models of the system, as a fuzzy inference system interprets linguistic rules, i.e. the reasoning of an expert. However, uncertainties in those linguistic rules in respect to inputs can occur and these cannot be handled by type-1 fuzzy logic. Therefore, Zadeh, the pioneer of fuzzy logic, as long ago as 1975, introduced type-2 fuzzy sets to handle uncertainties in the rules themselves, as expressed by membership functions (though type-2 FLS's were not developed at the time). Instead of a crisp number as input to type-1 fuzzy logic, the membership value/grade of a type-2 fuzzy set is a fuzzy set having values in the range [0, 1]. Interval type-2 sets are the simplest of type-2 sets, as they restrict the dimensions of the input membership functions to two rather than three dimensions. We have used interval type-2 sets in our reconfiguration engine, as such sets assume equal uncertainty across the range. Interval sets are useful when no firm information about uncertainties in the membership function is known. As such, they have in recent decades grown in importance [12] and spread to computer-engineering applications [13].

Two significant advantages of an interval type-2 fuzzy system [2] for DSE are as follows:

- The strength of a fuzzy type-2 based reconfigurable engine lies in suggesting an optimal output, even when the design dataset is uncertain. This ability to incorporate uncertainty is not possible in a good number of other optimization algorithms (i.e. Genetic Algorithm, Cat Swarm Optimization, Estimation of Distribution Algorithm and so on) or in fuzzy logic type-1. It was also proven by Wu and Mendel [14] that, in order to incorporate the uncertainty region, fuzzy logic type-2 is the most appropriate.
- As compared to fuzzy logic type-1, a fuzzy logic type-2 system can realize the input-output relationship in a more flexible way, as these systems are intrinsically more adaptive. In fact, Karnik et al. [15] showed that a fuzzy type-2 system can be considered to be a collection of many embedded type-1 fuzzy systems, which no longer need to be separately defined.

3. Fuzzy logic type-2 based design space exploration

In order to achieve an efficient trade off between the energy consumption and throughput of an MPSoC, various system parameters were identified that clearly affect other objective metrics. Our objective metrics consist of: normalized energy consumption; throughput; and the L1/L2 cache miss rate.

3.1. Fuzzy logic type-2 processing

Fig. 1 shows the structure of the fuzzy logic type-2 based DSE engine, in which, after normalization of the input values, the fuzzifier converts input values into fuzzy type-2 linguistic variables, using membership functions kept in a knowledge base.

The type-2 fuzzifier is responsible for converting inputs into a form in which those input values of memberships of one or more fuzzy sets are uncertain. This is because every type-2 fuzzy membership function has an uncertainty region or Footprint-of-Uncertainty (FOU), within which the membership functions boundary may fall, though exactly where is not certain. More specifically, in this process, the fuzzifier determines the inner and outer FOU boundary values of a membership function. These are called the primary membership of a value. Given those values, the distribution of possible values across a line joining the boundary values is normally needed, in order to determine the secondary membership. However, in the case of an interval type-2 membership function, as herein, for a given input value, all values are said to be equally uncertain along the line joining the boundary points of the FOU. Therefore, assessing the impact of the uncertainty distribution reduces to simply multiplying by one, thus considerably reducing the overall calculations compared to a generalized fuzzy logic type-2 system. Consequently, the fuzzifier herein, in its conversion process, calculates the two boundary points using values and formulas kept in its knowledge base concerning the uncertainty region or FOU and accessed according to the input value. The design of the uncertainty region herein, leading to the construction of the knowledge base, is described in Section 4.

The next stage of Fig. 1 is to employ **If ... Then** rules in a rule base. The mth rule in a total of M rules of the fuzzy logic type-2 based exploration engine is expressed in (if) antecedent and (then) consequent form as:

$$\mathbf{R}^m : \textbf{If } x_1 \text{ is in } \widetilde{A}_1^m \text{ and } x_2 \text{ is in } \widetilde{A}_2^m \text{ and } \ldots x_n \text{ is in } \widetilde{A}_n^m \textbf{ Then } y \text{ is } \widetilde{w} = \widetilde{p}_0^m + \widetilde{p}_1^m x_1 + \cdots + \widetilde{p}_n^m x_n,$$

where $\widetilde{p}_0^m, \widetilde{p}_1^m, \ldots, \widetilde{p}_n^m$, represent zero, first and nth order coefficients, and $\widetilde{A}_0^m, \widetilde{A}_1^m, \ldots, \widetilde{A}_n^m$, are the linguistic values in the universe of discourse of the fuzzy sets. For example, in our scheme, for each of the three premise linguistic variables (i.e. energy consumption, throughput, cache miss ratios), three linguistic values are defined.

The explicit degree of membership for the mth rule, in general, is determined by:

$$\bar{f}^m(x) = \bar{\mu}_1^m(x_1') \star \bar{\mu}_2^m(x_2') \star \bar{\mu}_3^m(x_3') \star \ldots \star \bar{\mu}_n^m(x_n'), \tag{1}$$

$$\underline{f}^m(x) = \underline{\mu}_1^m(x_1') \star \underline{\mu}_2^m(x_2') \star \underline{\mu}_3^m(x_3') \star \ldots \star \underline{\mu}_n^m(x_n') \tag{2}$$

where $\star$ indicates the composition operator. The composition operator could be a product operator or a minimum operator.

The symbols $\bar{f}^k(x)$ and $\underline{f}^k(x)$ in (1) and (2) indicate the upper and lower membership values; that is to say they denote the membership firing interval.

Returning to Fig. 1, the next stage in the processing is to find the output. The output of the type-2 TSK exploration engine can be calculated by using Eq. (3):

$$\widetilde{w} = [w_l^k, w_r^k] = \int_{w^l \in [w_l^1, w_r^1] \ldots w^n \in [w_l^n, w_r^n]} \int_{f^1 \in [\underline{f}^1, \overline{f}^1] \ldots f^n \in [\underline{f}^n, \overline{f}^n]} \frac{1}{\frac{\sum_{k=1}^M f^k w^k}{\sum_{k=1}^M f^k}} \tag{3}$$

where, $\widetilde{w} = [w_l^k, w_r^k]$ denotes a type-1 set and the symbols w_r^k and w_l^k denote the upper and lower limit of the fuzzy kth rule. For equation (3), two endpoints are calculated by means of Eqs. (4) and (5):

$$w_l = \frac{\sum_{k=1}^M f_l^k w_l^k}{\sum_{k=1}^M f_l^k} \tag{4}$$

$$w_r = \frac{\sum_{k=1}^M f_r^k w_r^k}{\sum_{k=1}^M f_r^k} \tag{5}$$

In a type-2 TSK FLS, the output $\widetilde{w}$ is obtained through the average of the upper and lower limits (i.e. w_r, w_l):

$$\widetilde{w} = \frac{w_r + w_l}{2} \tag{6}$$

In order to design the uncertainty regions for all the input parameters, extensive simulations were performed, as further discussed in Section 4. All the type-2 membership functions associated with the input parameters are summarized in Table 1. That is to say, Table 1 is a representation of the knowledge base referred to by the fuzzifier. (See the description of the fuzzifier's processing at the start of this Section.)

3.2. Type-2 system for MPSOCs

The steps in fuzzy type-2 based exploration procedure for MPSoCs are briefly discussed as follows:

Step 1: Use nature-inspired algorithms, see the authors' [11,16,17], to find the uncertainty values associated with antecedent parameters, which are energy consumption, throughput and cache miss ratio (i.e. $\delta_{E.C}$, δ_T, and $\delta_{M.R}$).

Step 2: Use uncertainty values and Gaussian type-1 membership functions as the principal models for the type-2 membership function by using uncertainty data arrived at by prior simulations, where lower μ_{jk} and upper values μ_{jk} are computed by using $\{\mu_{jk} - c\sigma/2\}$ and $\{\mu_{jk} + c\sigma/2\}$, and where the μ are means and the σ are standard deviations of Gaussian distributions.

Step 3: The consequent parameters are considered as type-1 because no uncertainty is involved with the output parameters of the exploration engine, those being number of cores, operating frequency, size and associativity of L1 and L2 caches.

Step 4: The scaling block of Fig. 1 converts the applied inputs $X = (x_{e.c}, x_{m.r}, x_t) \in X_{E.C} \times X_{M.R} \times X_T$ to normalized values between [0 1].

Step 5: For all the normalized crisp inputs $x_{e.c} \in X_{E.C}$, $x_{m.r} \in X_{M.R}$ and $x_t \in X_T$, fuzzy type-2 membership values (i.e. $\underline{\mu}_j^m, \overline{\mu}_j^m$) are calculated.

Step 6: The mapping from input to the output is calculated using the inference engine $\mu_{R^m}(x, y) = \mu_{F \to W}(x, y) = \mu_{A_1^m}(x) \sqcap \ldots \sqcap \mu_{A_n^m}(x)$.

Table 1
Fuzzy logic type-2 inputs.

		Membership Function	σ	$[\underline{\mu}_m, \overline{\mu}_m]$	δ	c
$\boldsymbol{X_{TH}}$	L	$m_A(x) = \begin{cases} 0 & if\ 0.35 \le x_{TH} \le 0 \\ \frac{0.35-x}{0.35} & if\ 0 \le x_{TH} \le 0.35 \end{cases}$	0.0718	$[m_A$-cδ, m_A+c$\delta]$	0.0359	0.57
	M	$m_B(x) = \begin{cases} 0 & if\ 0.8 \le x_{TH} \le 0 \\ \frac{x-0.2}{0.3} & if\ 0.2 \le x_{TH} \le 0.5 \\ \frac{0.8-x}{0.3} & if\ 0.5 \le x_{TH} \le 0.8 \end{cases}$	0.1184	$[m_B$-cδ, m_B+c$\delta]$	0.0592	0.57
	H	$m_C(x) = \begin{cases} 0 & if\ 1.0 \le x_{TH} \le 0.65 \\ \frac{x-0.65}{0.35} & if\ 0.65 \le x_{TH} \le 1.0 \end{cases}$	0.0446	$[m_C$-cδ, m_C+c$\delta]$	0.0223	0.57
$\boldsymbol{X_{EC}}$	L	$m_A(x) = \begin{cases} 0 & if\ 0.35 \le x_{EC} \le 0 \\ \frac{0.35-x}{0.35} & if\ 0 \le x_{EC} \le 0.35 \end{cases}$	0.0897	$[m_A$-cδ, m_A+c$\delta]$	0.04485	0.57
	M	$m_B(x) = \begin{cases} 0 & if\ 0.8 \le x_{EC} \le 0 \\ \frac{x-0.2}{0.3} & if\ 0.2 \le x_{EC} \le 0.5 \\ \frac{0.8-x}{0.3} & if\ 0.5 \le x_{EC} \le 0.8 \end{cases}$	0.0749	$[m_B$-cδ, m_B+c$\delta]$	0.03745	0.57
	H	$m_C(x) = \begin{cases} 0 & if\ 1.0 \le x_{EC} \le 0.65 \\ \frac{x-0.65}{0.35} & if\ 0.65 \le x_{EC} \le 1.0 \end{cases}$	0.0898	$[m_C$-cδ, m_C+c$\delta]$	0.0449	0.57
$\boldsymbol{X_{MR}}$	L	$m_A(x) = \begin{cases} 0 & if\ 40\% \le x_{MR} \le 0\% \\ 1 & if\ 0\% \le x_{MR} \le 40\% \end{cases}$	0.0768	$[m_A$-cδ, m_A+c$\delta]$	0.0384	0.57
	M	$m_B(x) = \begin{cases} 0 & if\ 75\% \le x_{MR} \le 0\% \\ \frac{x-25}{25} & if\ 25\% \le x_{MR} \le 50\% \\ \frac{75-x}{25} & if\ 50\% \le x_{MR} \le 75\% \end{cases}$	0.1130	$[m_B$-cδ, m_B+c$\delta]$	0.0565	0.57
	H	$m_C(x) = \begin{cases} 0 & if\ 100\% \le x_{MR} \le 60\% \\ \frac{x-60}{40} & if\ 60\% \le x_{MR} \le 100\% \end{cases}$	0.0970	$[m_C$-cδ, m_C+c$\delta]$	0.0485	0.57

X_{TH}: Linguistic variable for throughput X_{EC}: Linguistic variable for energy consumption.
X_{MR}: Linguistic variable for cache miss ratio.

Step 7: In the output processing block of Fig. 1, a type-1 fuzzy set output is first created using Eqs. (3)–(5), which is then converted into crisp outputs using the defuzzifier of (6).

Step 8: Based upon the suggested values in Step 7 (i.e. y_{ca}, y_{co}, y_{fr}) the MPSoC is reconfigured. For the suggested values $x_{E.C}$, $x_{M.R}$ and x_T values are calculated for further comparison via the feedback loop of Fig. 1.

Step 9: If the antecedent parameters value are less than a pre-specified condition, which is $x_{E.C} < x_{D.E.C}$, $x_{T.H} < x_{D.T.H}$ and $x_{M.R} < x_{D.M.R}$, then the loop is terminated, while otherwise processing continues from Step 4.

Thus, fuzzy logic rules were defined in order to establish the relationship between input and output variables, so as to achieve a balance between the energy consumption and throughput of MPSoCs, though the details of the rules are not included for reasons of available space.

4. Designing the uncertainty region for the antecedent

In the development process, initially a fuzzy logic type-1 exploration was designed using membership functions, as shown in the second column of Table 1. Once the fuzzy type-1 exploration engine was designed, uncertainty information about throughput, energy consumption and miss ratio was incorporated for better exploration of MPSoCs.

While designing the fuzzy-logic type-2 exploration engine, only the antecedents were of type-2. In order to design those uncertainty regions, extensive simulations were performed for various configurations of input parameters within the design space. For all the antecedents within the exploration engine (i.e. energy consumption, throughput, and miss ratio) membership functions were arrived at. Based upon these simulations, antecedent parameters were calculated with the simulation methodology discussed in Section 5.

Based upon these parameters, the standard deviations were calculated for energy consumption, throughput and cache miss ratio. These standard deviations are incorporated into Table 1 (as the third column) for energy consumption, throughput, and cache miss ratio. The possible variation within which each antecedent parameter can vary in the range [μ-cδ, μ+cδ], where $\delta = \sigma_{uncertainty}/2$. Thus, the fuzzy type-2 model has been designed using a type-1 set, when the equivalent fuzzy type-2 set has an uncertainty associated with it. Changing the value of constant c changes the uncertainty region of all the antecedents. By experimenting with different values, an appropriate value for this constant was found, as noted in the last column of Table 1.

Fig. 2 shows an inner type-1 set around which the outer boundary of the equivalent type-2 set is plotted. The width of the FOU is marked as $2c\delta$, with the values of c and δ recorded in Table 1. In Fig. 2, input values are found on the $u(k)$ horizontal axis. A vertical line from a given input value will cross the boundary of the inner type-1 set and the outer boundary of the FOU. The membership values for the two crossing points can then be read-off from the vertical membership axis. In practice, these values are found by calculation, using the formulas in the knowledge base, of which Table 1 is a representation.

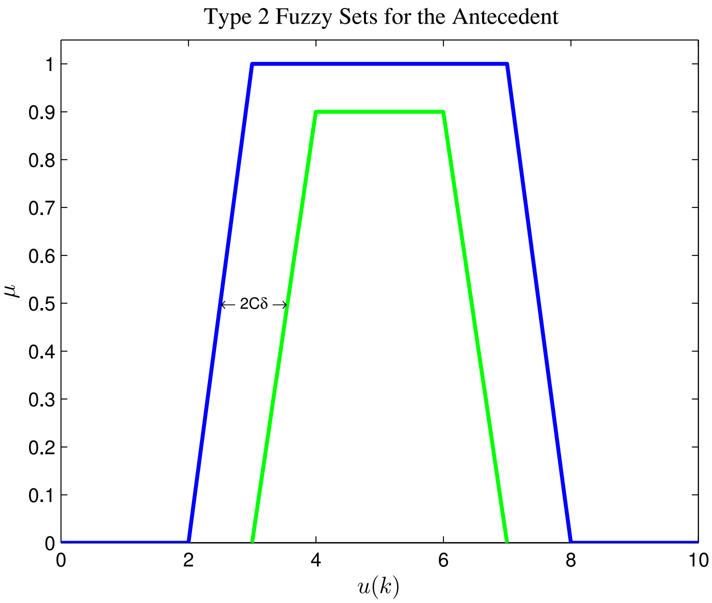

Fig. 2. Type-2 set obtained by displacing the mean of a type-1 set. The uncertainty region or FOU for each type-2 set is assumed to be in the interval of $2c\delta$. Notice that this interval of uncertainty is symmetric about the mean.

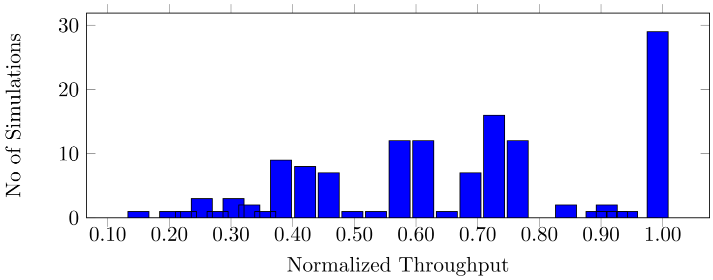

Fig. 3. Simulation data for normalized throughput.

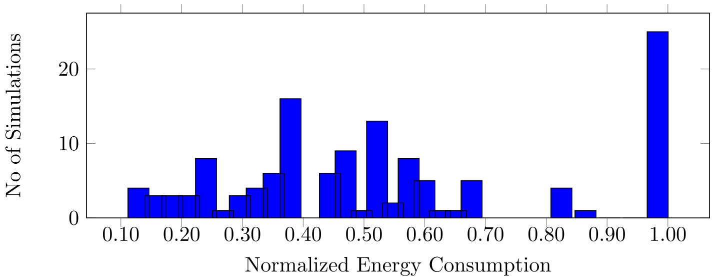

Fig. 4. Simulation data for normalized energy consumption.

Notice that the main strength of the fuzzy logic type-2 based reconfigurable engine lies in suggesting an optimal output, even when the data provided for design purposes are uncertain. To reiterate, the ability to incorporate uncertainty in the input is impossible with other optimization algorithms such as a genetic algorithm, cat swarm optimization, estimation of distribution algorithms or fuzzy logic type-1.

5. Simulation results for type-2 investigations

Figs. 3–5 show the results of extensive simulations that were conducted in order to find the FOUs for the fuzzy type-2 membership functions. Because, for each of the three fuzzy type-2 based exploration engines, three fuzzy membership functions were needed representing lower, upper and medium membership, so, for each of these three membership functions, simulations were required in order to find the uncertainty regions.

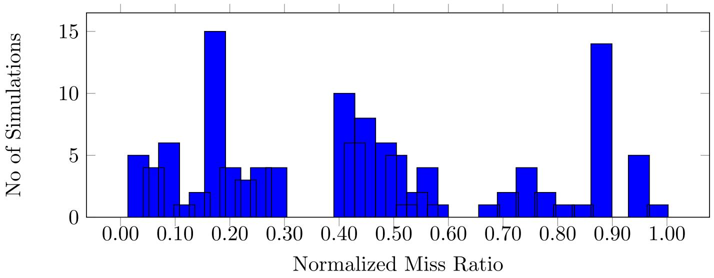

Fig. 5. Simulation data for normalized cache miss ratio.

Fig. 3 shows the normalized throughput results against the number of simulations giving rise to each data point. For each of the three membership functions, δ values were found by halving the standard deviations using this plot. The values are recorded in the last but one column of Table 1.

Normalized energy consumption values against the number of simulations are presented in Fig. 4. Just as for the normalized throughput ratio, for the normalized energy consumption, for lower, medium, and upper membership functions (i.e. μ_A, μ_B and μ_C) their ranges were established and δ values were derived.

The same procedure was followed for the normalized cache miss ratio, with Fig. 5 being used in the delineation of the membership functions. However, different ranges for the membership functions were found to be necessary, as is evident from column 2 of Table 1.

With the value of c determined heuristically, and the value of δ found from the simulations of Figs. 3–5, it is possible to find the formulas for the boundaries of the regions of uncertainty or FOUs. These formulas are given as column 4 of Table 1 for each of the membership functions according to their linguistic variables, i.e. for throughput, energy, and cache miss ratio.

6. System architecture and simulation setup

This Section outlines the proposed approach. Initially the multicore architecture is discussed, followed by the proposed interconnect infrastructure, the fuzzy logic reconfiguration engine, the benchmark applications and finally the simulation setup.

6.1. MPSoC architecture

The proposed MPSoC has a 16-core symmetric chip multiprocessor platform based on the standard Intel x86 architecture. The platform has a shared-memory architecture with L1 and L2 caches having a configurable size and associativity, supported with the Modified-Exclusive-Shared-Invalid (MESI) coherence protocol. Both, the number of active cores and the processor frequency/voltage are adjustable for throughput and energy control. The energy consumption and voltage/frequency information were obtained from the Intel 486 GX embedded processor datasheet. Each core of the system is connected to a CMOS power switch, so that when it is turned off, the leakage energy of the core does 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 respectively, as per the original Intel 486 GX processor specifications [18]. Owing to the possible variance in timing by changing the cache size and associativity, the L1 cache miss penalty was assumed to be uniform at 10 machine cycles and that for the L2 cache was set to 30 cycles. The L1 and L2 cache timing and energy data were obtained from the CACTI cache-modeling software tool [19], with near accurate memory access time and energy estimates. However, as CACTI is not a trace-driven simulator, energy consumption as the result of the number of hits or misses is not accounted for in respect to a particular application. Detailed cache energy and throughput models accounting for cache miss ratios were presented by Qadri et al. [10,11]. These models are required to estimate the impact of various cache configuration on energy consumption of the overall system.

6.2. Simulation setup: Simulator and benchmark applications

For the simulation of the proposed architecture, we chose a cycle accurate, full-system simulator, the Micro-ARchitectural and System Simulator for x86-based Systems (MARSSx86) [20]. MARSSx86 is a fast cycle-accurate simulation platform for single core, multicore, and heterogeneous core configurations with cycle-accurate out-of-order x86 cores. It facilitates instruction-level simulations and is capable of running virtually on the target platforms' unmodified operating systems, such as Linux, and Windows XP. The simulator provides a fairly accurate timing profile but at present does not support energy profiling of the target system. It also gives a reasonably accurate cache profiling utility, making it well-suited to memory system research. It allows users to capture separate statistics for user-mode and kernel-mode activities in a single simula-

Table 2
Selected benchmark applications for simulation on MARSSx86.

Sr.#	Application	Description	Base problem size	Ref
1	Barnes	Uses Barnes-Hut method to solve N-body problems.	16,384 particles	[22]
2	FMM	Uses a parallel adaptive Fast Multipole Method(FMM) to simulate N-body problems.	16,384 particles	[23]
3	Water-Spatial	This application in SPLASH-2 simulates the same molecular dynamics N-body problem as the original Water-NSquared code in SPLASH.	512 molecules	[24]
4	LU	Factorization of a dense matrix into the product of a lower triangular and an upper triangular matrix	512 × 512 matrix	[24]
5	Ocean	Imitates large-scale ocean motility based on eddy and boundary currents.	258 × 258 grid	[25]

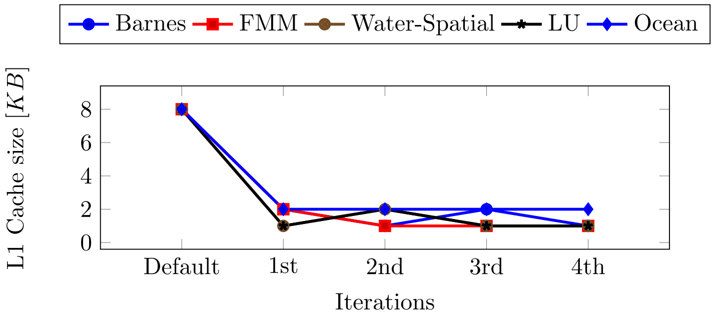

Fig. 6. Optimization results for L1 cache size.

tion run. To evaluate the results suggested by the fuzzy engine for a multicore processor, a number of SPLASH-2 benchmarks [21] were chosen to perform the target evaluation. The benchmark applications used for that purpose are shown in Table 2.[1]

7. Evaluation results for reconfiguration engine

The main objective of the proposed DSE methodology is to achieve a good balance between energy consumption and throughput for MPSoCs. To achieve this goal, the DSE engine optimizes the reconfigurable parameters (i.e. L1/L2 cache size and associativity, operating frequency and number of cores) over a number of iterations. The system reconfiguration process completes within four iterations. The following Sections present the effects of reconfiguration on each parameter.

7.1. L1 cache and associativity

The L1 cache size and associativity have a significant impact upon the throughput and energy consumption of an MPSoC. The larger cache sizes with higher associativity produce a lower miss ratio. (The size of miss ratio is a major cause of energy consumption and throughput reduction.) Though increasing cache size and associativity also increases throughput, it also increases the energy consumption and latency. Therefore, it is important for an MPSoC to use an optimum cache size and choose an associativity appropriate to the application being run. The default 16-core configuration had maximum energy consumption and throughput. In order to save energy, our proposed fuzzy logic type-2 based DSE engine output, after the 4th iteration through the feedback loop of Fig. 1, a reduced cache size from the default 8–1 KB for the Barnes, FMM, Water-Spatial and LU applications, and 2 KB for the Ocean application (see Fig. 6). The DSE engine also changed the L1 cache associativity from default 16-way to 1-way associativity for all benchmarks after the 3rd iteration through the feedback loop of Fig. 1 (see Fig. 7).

7.2. L2 cache and associativity

The targeted MPSoc utilizes a uniform, shared L2 cache memory along with an L1 cache. The proposed DSE engine reduced the cache size from 128 KB to 64 KB for Barnes, FMM and LU applications after the 1st iteration and reduced the size from 32 KB to 16 KB for Water-Spatial and Ocean applications after the 4th iteration (see Fig. 8). The DSE engine also changed the L1 cache associativity from default 16-way to 1-way associativity for all benchmarks except for Water-Spatial. For Water-Spatial the associativity became 2-way associativity after the 4th iteration (see Fig. 9).

[1] For a maximum 64 processors.

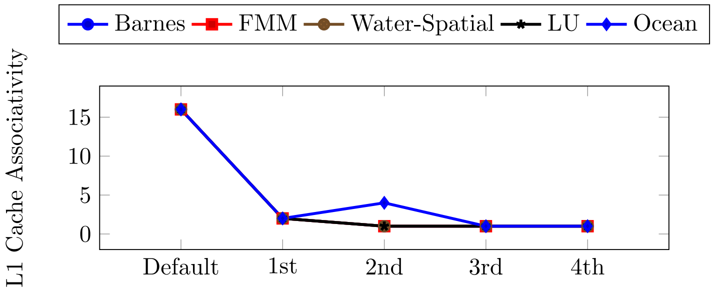

Fig. 7. Optimization results for L1 cache associativity.

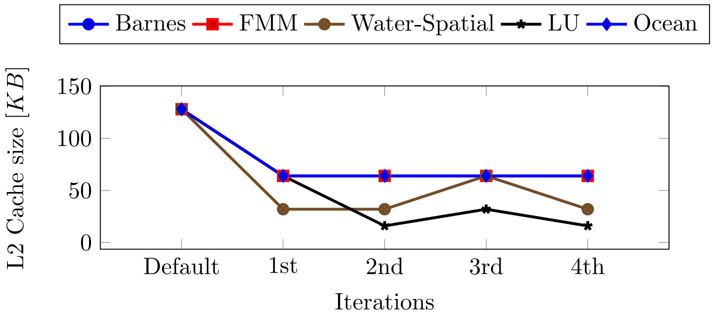

Fig. 8. Optimization results for L2 cache size.

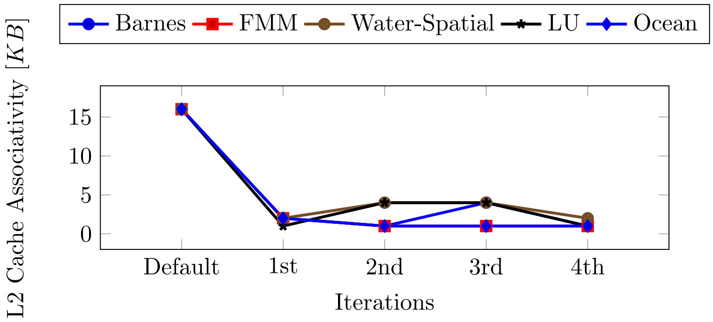

Fig. 9. Optimization results for L2 cache associativity.

7.3. *Operating frequency and number of cores*

Similarly to cache size and associativity, the operating frequency of the MPSoCs also contributes to both their throughput and energy consumption. A higher frequency produces a higher throughput along with a higher energy consumption. Therefore, the proposed fuzzy logic type-2 based DSE engine addresses this issue. Notice that the default frequency was 33 MHz in the simulations The DSE engine suggested 20 MHz as a frequency for all benchmark applications (see Fig. 10).

For MPSoCs, the number of cores significantly affects throughput and energy consumption. The behaviour arising from varying the number of cores upon throughput is governed by Amdahl's Law. According to the law, throughput does not vary linearly with a varying number of cores. In fact, throughput saturates for the most applications after scaling the number of cores beyond a certain limit. Energy consumption, which can be reduced by turning off the excess cores, also increases with an increase in the number of cores. The DSE engine decreased the number of cores from the default number of 16 cores to 1 for all benchmark applications in the 4th iteration (see Fig. 11).

The impact of the iterative results on various metrics such as cache miss ratio, energy consumption, throughput and EDP is observed and discussed in the following Sections.

7.4. *Cache miss ratio*

Cache miss ratio is an important metric to check the performance of cache. L1 cache modifications result in decreasing the L1 miss ratio from 1.83% to 0.53% for Barnes, 2.7% to 0.95% for FMM, 2.08% to 0.24% for Water-Spatial, 8.53% to 0.90%

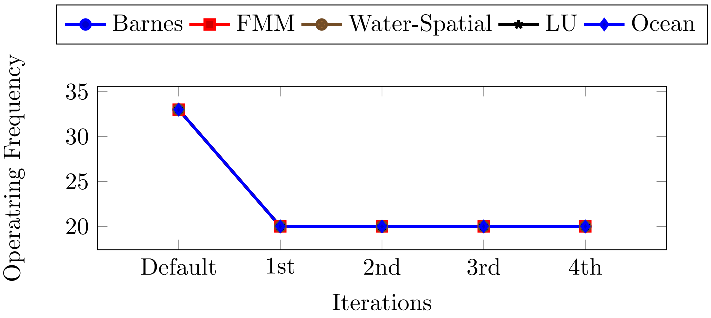

Fig. 10. Optimization results for operating frequency.

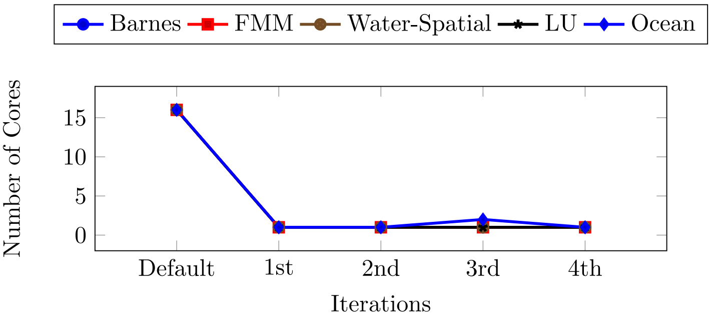

Fig. 11. Optimization results for number of cores.

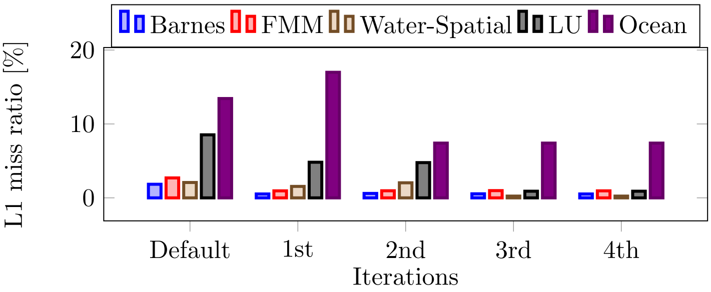

Fig. 12. Impact of optimization on L1 miss ratio.

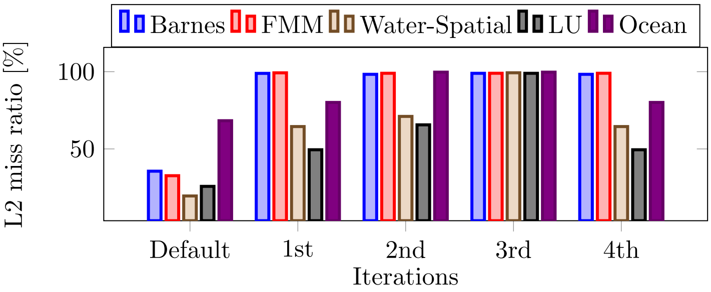

Fig. 13. Impact of optimization on L2 miss ratio.

for LU and from 13.44% to 7.41% for the Ocean benchmark (see Fig. 12). The impact of alteration of the L2 cache size and associativity results in a relative increase in the L2 miss ratio. For Barnes and LU these increases in miss ratios are 35.58–98.32% and 25.74–49.46% respectively. For Water-Spatial, the miss ratio increased from 19.53% to 64.48%, while for Ocean and FMM, the L2 miss ratio increased from 68.27 to 80.11 and 35.58% to 99.01% respectively (see Fig. 13).

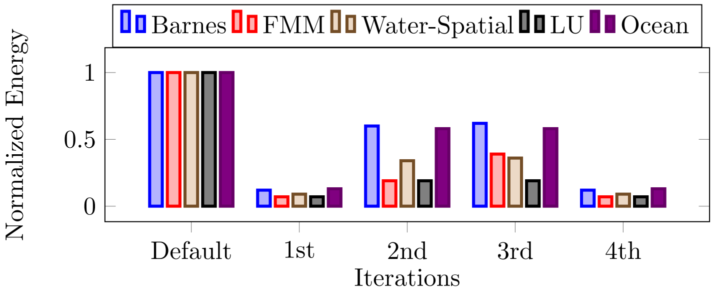

Fig. 14. Impact of optimization on normalized energy.

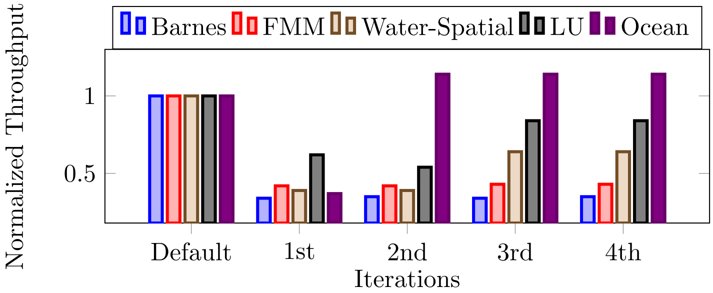

Fig. 15. Impact of optimization on normalized throughput.

7.5. Normalized energy consumption

The energy consumption for the default configuration (i.e., having maximum system resources) was at a maximum. In order to reduce energy consumption, the fuzzy-based technique was applied. Thus, by reconfiguring system resources greater energy saving was achieved (see Fig. 14). Energy consumption for all benchmark applications has been significantly decreased. For Barnes and Ocean, energy consumption was reduced to 12%, being almost 13% of the default configuration. Energy consumption for other benchmark application such as FMM and LU has been decreased to 7%, while for Water-Spatial consumption has been reduced to 9% of the default configuration. That much of a reduction in energy consumption is due to the fact that the number of cores, cache size and frequency has been reduced in the final iteration. In addition, the proper L1 and L2 cache associativity has been assigned. Shutting down a core results in a reduction in energy consumption of MPSoCs. Similarly, energy consumption is significantly decreased by reducing the cache size and operating frequency.

7.6. Normalized throughput

Our main objective was to have a good balance between energy consumption and throughput for MPSoCs (i.e., minimizing energy consumption at an acceptable lower throughput). As in the energy consumption discussion (Section 7.5), we noticed that the resulting energy reduction was due to the smaller resource allocation of the system. It is obvious that smaller resources may well produce a smaller throughput. In this Section, we will discuss how the throughput is affected by the proposed DSE technique, while achieving lower energy consumption. Fig. 15, shows the normalized throughput of the MPSoC at successive iterations. The results show that, after the third iteration, the throughput is reduced to 84%, 64%, 43% and 35% for LU, Water-Spatial, FMM and Barnes applications respectively. However, for the Ocean benchmark application, the throughput is increased to 114%. As our main goal is to reduce energy consumption, in general, throughput is compromised to achieve greater energy efficiency. Another metric, the energy-delay product (discussed in the next Section), gives a collective measure of the impact of the levels of energy consumption and throughput.

7.7. Energy delay product

As mentioned in Section 1, the EDP [7] is an important metric to measure the combined effect of energy consumption and throughput. A configuration with a smaller EDP value is jointly more efficient in terms of energy consumption and throughput. Fig. 16 shows the normalized EDP of different configurations achieved by the proposed fuzzy logic type-2 based DSE engine at successive iterations. At all iterations, the EDP value is smaller than the default EDP value for all benchmark applications, except for the Barnes application. For the Barnes application, it shoots up to 172% and 180% in the second and third iterations respectively before reducing to 35% in the final iteration. The EDP value decreases to 18%, 24% and 13% for Ocean, FMM and Water-Spatial respectively, which are significant decreases. For the LU application, the EDP value is reduced to 60%, which is much better than the default EDP for this application.

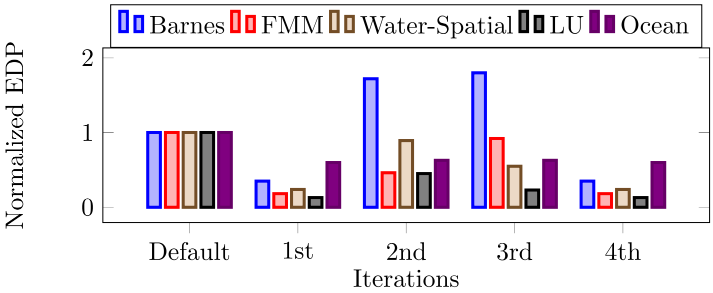

Fig. 16. Impact of optimization on normalized EDP.

8. Conclusion

The authors set-out to demonstrate the effectiveness of modeling a reconfigurable Multiprocessor System-on-a-Chip with fuzzy logic of type-2. The input parameters were: the level-one and level-two cache sizes and their associativities; the processor's clock frequency; and the number of cores employed in the targeted 16-core, symmetric multiprocessor. Many simulations were performed, varying the input parameters and plotting the distributions of the energy consumption, throughput, and cache miss ratios. From these results, it was possible to establish the controlling parameters of the membership functions of the three corresponding type-2 exploration engines. Assuming Gaussian distributions, these parameters were the standard deviations and a heuristically selected constant. Consequentially, following the type-2 logic computational rules, the reconfiguration settings were iteratively arrived at.

For the five large-scale, numerical applications tested, four iterations were enough to arrive at convergence. For the derived configuration parameters, results showed that the energy reduction was considerable, ranging from 93% to 87% compared to a default configuration in which maximum multiprocessor resources were employed. For three of the applications, there was some reduction in throughput, ranging from 35% to 84%. However, as a balanced solution and judged by the Energy-Delay-Product metric, there was a gain over the default ranging from 97% to 40%. Thus, results validate the approach of using a fuzzy logic type-2 based reconfiguration model to achieve a balance between the energy consumption and the throughput of Multiprocessor System-on-a-Chips.

Acknowledgments

This work was fully supported by the National ICT R & D Fund, Ministry of IT, Government of Pakistan, under the grant number: ICTRDF/TR&D/2012/65.

References

[1] Mendel JM. Uncertain rule-based fuzzy logic systems: introduction and new directions. 2nd ed. Berlin, Germany: Springer Verlag; 2017.
[2] Mendel J. Type-2 fuzzy sets and systems: an overview. IEEE Comput Intell Mag 2007;2(1):20–9.
[3] Solihin Y. Fundamentals of parallel multicore architecture. Boca Raton, FL: CRC; 2016.
[4] Givargis T, Vahid F. Platune: a tuning framework for System-on-a-Chip platforms. IEEE Trans Comput Aided Des Integr Circt Syst 2002;21(11):1317–27.
[5] Mariani G, Avasare P, Vanmeerbeeck G, Ykman-Couvreur C, Palermo G, Silvano C, et al. An industrial design space exploration framework for supporting run-time resource management on multi-core systems. In: Design, automation & test in Europe conference & exhibition; 2010. p. 196–201.
[6] Spittal S. A survey of techniques for improving energy efficiency in embedded computing systems. Int J Comput Aided Eng Technol 2014;6(4):440–59.
[7] Ahmed J, Siyal M, Najam S, Najam Z. Energy-delay product and throughput. In: Fuzzy logic based power-efficient real-time multi-core system. Singapore: Springer Verlag; 2017. p. 17–22.
[8] Chen J, John LK. Energy-aware application scheduling on a heterogeneous multi-core system. In: IEEE international symposium on workload characterization; 2008. p. 5–13.
[9] Li D, Hou Y, Huang Z, Xiao C. Multiprocessor system-on-chip design with fuzzy dynamic scheduling for parallel video processing. In: IEEE/ACIS 10th international conference on computer and information science; 2011. p. 65–70.
[10] Qadri MY, McDonald-Maier KD. A fuzzy logic reconfiguration engine for symmetric chip multiprocessors. In: International conference on complex, intelligent and software intensive systems; 2010. p. 937–43.
[11] Qadri MY, McDonald Maier KD, Qadri NN. Energy and throughput aware fuzzy logic based reconfiguration for MPSocs. J Intell Fuzzy Syst 2014;26(1):101–13.
[12] Hagras H, Alghazzawi D, Aldabbagh G. Employing type-2 fuzzy logic systems in the efforts to realize ambient intelligent systems. IEEE Comput Intell 2015;10(1):44–51.
[13] Jammeh E, Fleury M, Wagner C, Hagras H, Ghanbari M. Interval type-2 fuzzy logic congestion control for video streaming across IP networks. IEEE Trans Fuzzy Syst 2009;17(5):1123–42.
[14] Wu D, Mendel JM. Computing with words for hierarchical decision making applied to evaluating a weapon system. IEEE Trans Fuzzy Syst 2010;18(3):441–60.
[15] Karnik NN, Mendel JM, Liang Q. Type-2 fuzzy logic systems. IEEE Trans Fuzzy Syst 1999;7(6):643–58.
[16] Hussain I, Ahmad A, Qadri MY, Qadri NN, Ahmed J. Ant colony optimization for multicore re-configurable architecture. AI Commun 2016;25(5):595–606.
[17] Ahmad A, Qadri MY, Qadri NN. Fuzzy logic based adaptive MPSoc for balanced energy and throughput. Kuwait J Sci 2016;43(3):91–100.
[18] Embedded ultra-low power Intel 486$^{\text{TM}}$ GX processor, datasheet, 48 pages (1997).
[19] Shyamkumar T, Muralimanohar N, Ahn J-H, Jouppi N. Cacti 5.1. Tech. Rep.. HP Labs; 2008.

[20] Patel A, Afram F, Ghose K. Marss-x86: A QEMU-based micro-architectural and systems simulator for x86 multicore processors. In: 1st International QEMU users forum; 2011. p. 29–30.

[21] Woo SC, Ohara M, Torrie E, Singh JP, Gupta A. The SPLASH-2 programs: characterization and methodological considerations. ACM SIGARCH Comput Arch News 1995;23(2):24–36.

[22] Singh JP, Hennessy JL, Gupta A. Implications of hierarchical n-body methods for multiprocessor architectures. ACM Trans Comput Syst 1995;13(2):141–202.

[23] Singh JP, Holt C, Hennessy JL, Gupta A. A parallel adaptive fast multipole method. In: ACM/IEEE conference on supercomputing; 1993. p. 54–65.

[24] Woo SC, Singh JP, Hennessy JL. The performance advantages of integrating block data transfer in cache-coherent multiprocessors. In: 6th international conference on architectural support for programming languages and operating systems; 1994. p. 219–29.

[25] Woo SC, Singh JP, Hennessy JL. The performance advantages of integrating message passing in cache-coherent multiprocessors. Tech. Rep.. Stanford University; 1993.

Ishfaq Hussain received his bachelors and masters degrees in Electrical Engineering, in 2012 and 2015 respectively, from HITEC University Taxila Cantt., Pakistan. He is currently pursuing his Ph.D. degree in Electrical and Computer Engineering at University of Porto/CISTER research center, Portugal. His research interests includes safety critical systems, energy-aware computing, advanced computer architecture, Artificial Intelligence, and embedded systems.

Shahid Ali Murtza received his B.Sc. in Electronic Engineering from the International Islamic University, Islamabad, Pakistan in 2014. Currently, he is pursuing his M.Sc. (Electrical Engineering) from SEECS-NUST, Islamabad and working as an RA at SAVe Lab SEECS-NUST. Previously, he worked in an ICT R&D funded project on multi-core reconfigurable processors. His research interests include Optimization and Formal Reliability Analysis.

Muhammad Yasir Qadri gained his Ph.D. in 2010 at the University of Essex, UK. He has more than seven years experience in embedded system design. Yasir is Co-Principal Investigator at the National ICT R& D Fund, Pakistan for research in MPSoC schemes based on power efficient throughput management. His research interests include low-power processor architectures and cache optimization for reconfigurable MPSoC.

Martin Fleury has authored or co-authored around two hundred and ninety articles and book chapters as well as a number of books. He is an Associate Editor of Electronics Letters. His current research interests are video communication over wired and wireless networks, including video security and video quality of experience. He also publishes on embedded systems and smart meter systems.

Nadia N. Qadri gained her Ph.D. (Electronics Systems Engineering) in 2010 at the University of Essex, UK. Nadia is an Associate Professor and Program Co-coordinator of Electrical Engineering Programs at CIIT, Wah Campus, Pakistan. She also leads a research group Multimedia Wireless Networks and Technology at CIIT. Her research interests include Smart Grid, Internet of Things, and Multicore Reconfigurable Architectures.