Energy and Throughput aware Fuzzy Logic based Reconfiguration for MPSoCs

Muhammad Yasir Qadri, Klaus D. McDonald Maier*

School of Computer Science and Electronic Engineering
University of Essex, CO4 3SQ, Colchester, United Kingdom

Nadia N. Qadri†

Department of Electrical Engineering,
COMSATS Institute of Information Technology, Wah Cantt. Pakistan

Abstract

Multicore architectures offer amount of parallelism that is often underutilized, as a result these underutilized resources became a liability instead of advantage. Inefficient resource sharing on the chip can have a negative impact on the performance of an application and may result in greater energy consumption. A large body of research now focuses on reconfigurable multicore architectures in order to support algorithms to find optimal solutions for improved energy and throughput balance. An ideal system would be able to optimize such reconfigurable system to a level that optimum resources are allocated to a particular workload and all the other underutilized remain inactive for greater energy savings. This paper presents a fuzzy logic based reconfiguration engine targeted to optimize a multicore architecture according to the workload requirements for optimum balance between power and performance of the system. The proposed fuzzy logic reconfiguration engine is designed around a 16-core SCMP architecture comprising of reconfigurable cache memories, power gated cores and adaptive on-chip network routers for minimizing leakage energy effects for inactive components. A coarse grained architecture was selected for being able to reconfigure faster, thus making it feasible to be used for runtime adaptation schemes. The presented architecture is analyzed over a set of OpenMP based parallel benchmarks and results show significant energy savings in all cases.

1 Introduction

Multicore architectures are rapidly emerging as an important design paradigm for both high performance and embedded processors. These architectures have often been investigated and designed in order to achieve a greater throughput combined with minimum energy consumption. However several issues related to resource sharing on the chip can have a negative impact on the performance of an application and therefore result in decreased performance. A large body of research now focuses on reconfigurable multicore architectures in order to support algorithms to find optimal solutions for improved energy and throughput balance. As a result of on-going research several online and offline techniques and algorithms have been proposed for hardware adaptation. This paper presents a novel fuzzy logic reconfiguration scheme called Essex Fuzzy Logic Reconfiguration Engine (E-FLORE) [1, 2] for coarse-grained MPSoC reconfiguration to find an optimum balance between power and performance. The fuzzy logic system is designed to support the dynamic reconfiguration of a multicore platform as per work load requirements. Fuzzy logic is an expert system based reasoning technique that provides a framework to transform imprecise information into a meaningful output

*Electronic address: yasirqadri@acm.org, kdm@essex.ac.uk
†Electronic address: drnadia@ciitwah.edu.pk

[3]. Generally a model's complexity increases when the system parameters begin to interact in a non-linear fashion [4], therefore fuzzy logic is applied as a reconfiguration engine in the proposed system to avoid a detailed modeling of the impact of all the system parameters on its behavior.
The use of fuzzy logic to optimize the proposed hardware design space offers the following advantages: Firstly, detailed modeling of the system is not required which is otherwise necessary to describe the behavior of the system under various configurations. Secondly, if modeling is required in case of a non-fuzzy logic based approach due to the large size of problem space, a linear model may not be possible to depict the behavior of the system and non-linear models are generally complex and compute intensive to be implemented at run-time for reconfiguration. Therefore the use of fuzzy logic provides a robust, adaptive, and light-weight approach to achieve balanced energy consumption and throughput of the system.
The next section gives an overview of the related research in the field of reconfigurable Multiprocessor System on Chip (MPSoC) architectures and allied techniques.

2 Related Work

Most of the research in reconfigurable architectures has been focused on Field Programmable Gate Arrays (FPGAs) as the target platform. As some of the FPGAs also provide a run-time reconfiguration option, this feature is largely being exploited in adaptive hardware scenarios to provide optimized energy consumption and performance, or to support configurations larger than the available area on the device. However FPGAs suffer from higher reconfiguration latency as discussed in [5, 6, 7]. In order to address this issue Resano et al. [6] proposed a pre-emptive task scheduling and replacement scheme over a dynamically reconfigurable hardware (DRH) model that supports task migration and inter-task communication. Another example of such an approach was proposed by Kalte et al. [8] which implemented each task to a single location in FPGA and manipulated the configuration data stream to relocate the task in the FPGA. As a result of their work a tool called REPLICA2Pro was developed to facilitate the reconfiguration task for the Virtex-II/Pro FPGAs. Danne et al. [9] proposed two periodic real-time task scheduling algorithms for full FPGA reconfiguration. The first one is based on the earliest deadline first (EDF) concept, termed EDF-Next Fit (EDF-NF) and the second is based on the concept of servers that reserve area and execution time for other tasks called Merge Server Distribute Load (MSDL). The system utilization of the EDF-NF algorithm was found to be better than MSDL; however MSDL was proven to be more feasible for larger real-time tasks sets. Other examples of scheduling techniques and their applications can be found in [10, 11, 12, 13].
MPSoCs have also been investigated for run-time energy aware scheduling in several research articles. Thread scheduling is generally classified into balanced and unbalanced scheduling categories. Balanced scheduling distributes an equal amount of threads among the cores. DeVuyst et al. [14] have analyzed the performance of various scheduling schemes considering both energy and performance, and have shown that uneven thread scheduling often outperforms balanced scheduling, as greater throughput can be achieved by combining certain threads together on one core rather than by distributing among several cores. Reconfigurable multicore platforms also pose challenges in handling the communication between dynamically changing tasks and their synchronization. Li [15] has performed a detailed analysis of the performance of various task scheduling algorithms for minimizing schedule length combined with an energy consumption constraint. This work also analyzed the algorithms for minimizing energy consumption combined with a schedule length constraint on Dynamic Voltage and Frequency (DVF) supported multiprocessor systems. Yang et al. [16] have proposed a task scheduling method for concurrent tasks on a multicore platform that combines offline and online scheduling to exploit the energy-performance trade-off at run-time. This work is an extension of a proposed framework based on grey box modeling for improved concurrency and lower energy consumption by Prayati et al. [17]. Ma et al. [18] discuss a design time and run-time scheduling scheme for concurrent task management for real-time applications on a heterogeneous multicore platform. At design time, a set of schedules and assignments for each task was defined using Pareto curves, and at run-time a lightweight scheduler was used to select optimal working points exploiting dynamic and non-deterministic behavior of the system. For an MPEG4 texture decoder application their approach has shown significant improvement in performance while maintaining lower energy consumption. Other references to related work in this field can be found in [19, 20, 21].
Considering memory as the best candidate for optimization in an energy constrained multicore scenario,

Ahn et al. [22] presented a simplified approach to group DRAM chips into multiple virtual memory devices that receive separate address and control signals on a shared command path. This approach reduces energy consumption by minimizing the number of bits activated per memory access and by replacing the memory register with a demultiplexer register for routing command signals to the appropriate memory module, instead of mere transmission on the path.
Another candidate for power optimization in an MPSoC is the Network-on-Chip (NoC). [23] have presented a low latency router architecture with a two stage pipeline employing an adaptive routing scheme for congestion aware flow control. The router architecture was termed as path sensitive, as it utilized look-ahead routing for selecting the next route based on the four possible quadrants and routed the packet to the corresponding virtual channels assigned to that quadrant. Additionally, based on this partitioned approach a decomposed crossbar switch was proposed that resulted in a reduction of size for its connections and lower packet conflicts. Their work also includes a complete solution safeguarding against both the traditional link faults and internal router upsets, without incurring any significant latency, area and power overhead. Park et al. [24] have provided a detailed analysis of various logic errors and have proposed data recovery mechanisms. Individual cases were analyzed such as link errors occurring during flit traversal between routers, deadlocks, intra-router errors in the router pipeline such as errors caused by virtual channel allocators, routing units, switch allocators, and crossbars.
The next section provides an overview of the target design space and simulation environment and explores the implementation of proposed E-FLORE architecture in details.

3 System architecture and simulation setup

In this section an overview of the proposed approach is described in detail: initially the reconfigurable multicore architecture is introduced, the proposed interconnect infrastructure, the fuzzy logic reconfiguration engine, the benchmark applications used and finally the simulation setup.

3.1 Reconfigurable MPSoC Architecture

The proposed SoC is comprised of a 16-core symmetric chip multiprocessor platform based on the Intel x86 architecture, holding a shared memory architecture (see Figure 1). The platform incorporates L1 and L2 caches with configurable size and associativity. The number of active cores and processor frequency/voltage can be varied for energy and throughput regulation. The system configuration parameters are given in Table 1, showing various operating points of the system in terms of frequency and energy consumption. The energy consumption and voltage/frequency information was obtained from the Intel 486 GX embedded processor datasheet [25].
Each core of the system is assumed to be is connected to a CMOS power switch, such as the one proposed by Kim et al. [26], so that during the reconfiguration process when it is turned off, the leakage energy of the core should not contribute to the overall energy consumption of the system. An example of power gating of processing units can be found in the work by Zhigang et al. [27] and is depicted in Figure 2. The default size and associativity for L1 and L2 caches are 8KB, 4-way set associative; and 128KB, 8-way set associative as per original Intel 486 GX processor specifications [25]. Due to the possible variance in timing by changing cache size and associativity, the L1 cache miss penalty was assumed to be fixed 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 [28]. However CACTI 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 analytical models presented by Qadri et al. [29] were used to estimate the Cache energy and throughput based on cache hit/miss information.
In order to estimate the energy consumed in inter-processor communication a simulation methodology similar to [30, 31, 32, 33] is adopted. Each core in the MPSoC is assumed to be linked through a 2D mesh Network-on-Chip (NoC). The impact of various MPSoC configurations on the NoC power consumption is calculated using Orion 2.0 [34, 35], which is a fast and accurate NoC power and area simulator.

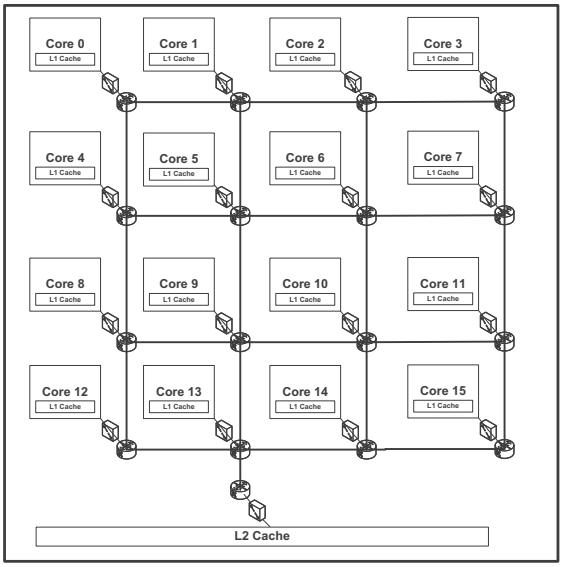

Figure 1: Target MPSoC Architecture

Table 1: System Parameters

Parameter	**Value**
Processor Type	Intel x86
Number of Cores	16
Operating Frequencies	[16, 20, 25, 33] MHz
Energy Consumption per cycle	[13.1,15.4,18.7,22.9]nJ

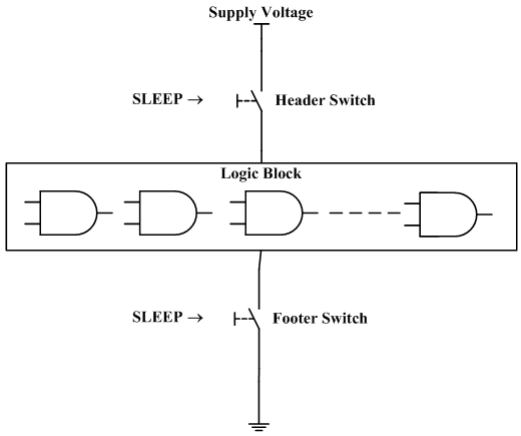

Figure 2: Power Gating using Header/Footer Switches

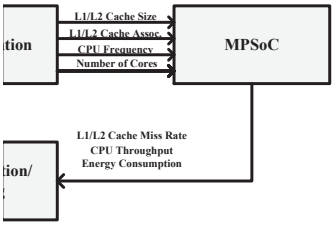

Figure 3: Closed Loop operation

3.2 The Essex-Fuzzy Logic Reconfiguration Engine (E-FLORE)

Fuzzy logic is considered to be one of the most suitable candidates for bridging the gap between computer and human logic. Fuzzy logic has been widely used for design space exploration and finding optimization in various applications [36, 37, 38, 39, 40, 41, 42, 43, 44]. Research has been carried out in-order to use fuzzy logic along with other optimization tools in order to explore optimal solutions [45, 46, 47]. Furthermore 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 reconfiguration 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 Figure 3, and is referred to as Essex Fuzzy Logic Reconfiguration Engine (E-FLORE). It may be noted that the proposed system is based on the principles of a typical feedback control system but 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 the fuzzy systems have a natural ability to handle vague information based on a rule base of linguistic terms.

In order to control the reconfiguration process a number of parameters were identified that can be modified dynamically. These parameters include L1/L2 Cache Size and associativity, CPU Frequency, and Number of active Cores. Based on the input parameters, system variables were identified that receive a clear impact from these parameters, which include L1/L2 aggregate Cache Miss Rate, aggregate CPU throughput, and Energy Consumption.
In order to fuzzify the input and output variables, each variable was partitioned into three fuzzy subsets which were assigned to their respective triangular membership functions namely μ_A, μ_B, and μ_C; classifying lower, middle and upper bounds of the given variable. For example in Table 2, for L1 and L2 the cache miss

rate varies from 0-100%, the membership function A is used to classify a low miss rate which is defined from 0-40%, B a moderate miss rate from 25-75%, and C a high miss rate from 60-100%. A similar approach is adopted for the rest of the input and output variables, and is detailed in Table 2 and 3 respectively.
To establish the relationship between the input variables and output parameters of the SoC, fuzzy logic rules were defined as shown in Appendix A. The rules were formed in such a way that a balanced throughput and energy consumption ratio can be achieved. For primary or core level configuration E-FLORE was devised so as to keep track of the average L1 miss rate, 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 rate but they also have an impact on the throughput and energy consumption of the device. Similarly for the secondary or system level configuration E-FLORE strives to find an optimal number of cores, L2 cache size and associativity while taking into account the L2 miss rate and total throughput and energy consumption of the SoC.
The proposed system applies the Centeroid method that calculates the centeroid 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].
The fuzzy logic engine was implemented using fuzzy logic API presented in [49] conforming to IEC 61131-7 standard [50].

3.3 Benchmark Applications

As the proposed architecture is comprised of a multicore setup, a set of NAS parallel benchmarks [51] based on OpenMP [52] is selected to perform the target evaluation. The benchmark applications used for this purpose are described as follows:

- IS (A): IS stands for Integer Sort. This application sorts small integers using the bucket sort [53] algorithm. The IS class A solves a problem size of 223 integers for 10 iterations.
- CG (A): The conjugate gradient method is used to compute an approximation to the smallest eigenvalue of a large, sparse, symmetric positive definite matrix. The CG class A, executes for a problem size of up to 14,000 for 15 iterations.
- FT(A): A 3-D partial differential equation solution using Fast Fourier Transforms. This kernel performs the essence of many spectral codes. The FT class A solves a problem size of 2562 x128, for 6 iterations.
- MG(A): A simplified multigrid kernel. It approximates a solution to the discrete Poisson problem. The class A problem size is 2563 for 4 iterations.
- EP(S): An *embarrassingly parallel* kernel. It provides an estimate of the achievable upper limits for floating point performance. In order to achieve this, the kernel generates pseudo-random floating point values using the Marsaglia polar method [54]. The class S application generates 33,554,432 random numbers.

3.4 Simulation Setup

The proposed architecture was simulated on a full system simulator Simics [55] which facilitates instruction level simulations and is capable of running unmodified operating systems such as VxWorks, Solaris, Linux, Tru64, 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. Simics 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. The cache memory simulation was carried out using the g-cache model which is the standard cache model that handles one transaction at a time in a flat way i.e. all needed operations (copy-back, fetch, etc.) are performed in order and at once.
Fedora Linux version 10 was chosen as the target OS due to the inherent multicore support provided in

Table 2: Fuzzy Membership Functions for Input Variables

$\mu_A = \begin{cases} 0 & if\ 40\% \leq \text{ L1/L2 Miss rate } \leq 0\% \\ \frac{40-x}{40} & if\ 0\% \leq \text{ L1/L2 Miss rate } \leq 40\% \end{cases}$ $\mu_B = \begin{cases} 0 & if\ 75\% \leq \text{ L1/L2 Miss rate } \leq 0\% \\ \frac{x-25}{25} & if\ 25\% \leq \text{ L1/L2 Miss rate } \leq 50\% \\ \frac{75-x}{25} & if\ 50\% \leq \text{ L1/L2 Miss rate } \leq 75\% \end{cases}$ $\mu_C = \begin{cases} 0 & if\ 100\% \leq \text{ if L1/L2 Miss rate } \leq 60\% \\ \frac{x-60}{40} & if\ 60\% \leq \text{ L1/L2 Miss rate } \leq 100\% \end{cases}$
Ł1 and L2 Miss rate
$\mu_A = \begin{cases} 0 & if\ 0.35 \leq \text{ Throughput } \leq 0 \\ \frac{0.35-x}{0.35} & if\ 0 \leq \text{ Throughput } \leq 0.35 \end{cases}$ $\mu_B = \begin{cases} 0 & if\ 0.8 \leq \text{ Throughput } \leq 0 \\ \frac{x-0.2}{0.3} & if\ 0.2 \leq \text{ Throughput } \leq 0.5 \\ \frac{0.8-x}{0.3} & if\ 0.5 \leq \text{ Throughput } \leq 0.8 \end{cases}$ $\mu_C = \begin{cases} 0 & if\ 1.0 \leq \text{ Throughput } \leq 0.65 \\ \frac{x-0.65}{0.35} & if\ 0.65 \leq \text{ Throughput } \leq 1.0 \end{cases}$
Normalized Throughput
$\mu_A = \begin{cases} 0 & if\ 0.35 \leq \text{ Energy Consumption } \leq 0 \\ \frac{0.35-x}{0.35} & if\ 0 \leq \text{ Energy Consumption } \leq 0.35 \end{cases}$ $\mu_B = \begin{cases} 0 & if\ 0.8 \leq \text{Energy Consumption } \leq 0 \\ \frac{x-0.2}{0.3} & if\ 0.2 \leq \text{ Energy Consumption } \leq 0.5 \\ \frac{0.8-x}{0.3} & if\ 0.5 \leq \text{ Energy Consumption } \leq 0.8 \end{cases}$ $\mu_C = \begin{cases} 0 & if\ 1.0 \leq \text{ Energy Consumption } \leq 0.65 \\ \frac{x-0.65}{0.35} & if\ 0.65 \leq \text{ Energy Consumption } \leq 1.0 \end{cases}$
Normalized Energy Consumption

Table 3: Fuzzy Membership Functions for Output Variables

$\mu_A = \begin{cases} 0 & if\ 3.5KB \leq \text{ L1 Cache size } \leq 1KB \\ \frac{3.5-x}{3.5} & if\ 1KB \leq \text{ L1 Cache size } \leq 3.5KB \end{cases}$ $\mu_B = \begin{cases} 0 & if \text{ L1 Cache size } \leq 2KB \text{ or } \geq 7KB \\ \frac{x-2}{2.5} & if\ 2KB \leq \text{ L1 Cache size } \leq 7KB \\ \frac{7-x}{2.5} & if\ 4.5KB \leq \text{ L1 Cache size } \leq 7KB \end{cases}$ $\mu_C = \begin{cases} 0 & if \text{ L1 Cache size } \leq 5.5KB \text{ or } \geq 8KB \\ \frac{x-5.5}{2.5} & if\ 5.5KB \leq \text{ L1 Cache size } \leq 8KB \end{cases}$
L1 Cache size
$\mu_A = \begin{cases} 0 & if \text{ L2 Cache size } \leq 1KB \text{ or } \geq 50KB \\ \frac{50-x}{50} & if\ 1KB \leq \text{ L2 Cache size } \leq 50KB \end{cases}$ $\mu_B = \begin{cases} 0 & if \text{ L2 Cache size } \leq 20KB \text{ or } \geq 100KB \\ \frac{x-20}{40} & if\ 20KB \leq \text{ L2 Cache size } \leq 60KB \\ \frac{100-x}{40} & if\ 60KB \leq \text{ L2 Cache size } \leq 100KB \end{cases}$ $\mu_C = \begin{cases} 0 & if \text{ L2 Cache size } \leq 80KB \text{ or } \geq 128KB \\ \frac{x-80}{48} & if\ 80KB \leq \text{ L2 Cache size } \leq 128KB \end{cases}$
L2 Cache size
$\mu_A = \begin{cases} 0 & if \text{ L1/L2 Cache Associativity } \leq 0 \text{ or } \geq 2 \\ 1 & if\ 0 \leq \text{ L1/L2 Cache Associativity } \leq 2 \end{cases}$ $\mu_B = \begin{cases} 0 & if \text{ L1/L2 Cache Associativity } \leq 1 \text{ or } \geq 8 \\ 1 & if\ 1 \leq \text{ L1/L2 Cache Associativity } \leq 8 \end{cases}$ $\mu_C = \begin{cases} 0 & if \text{ L1/L2 Cache Associativity } \leq 4 \text{ or } \geq 16 \\ 1 & if\ 4 \leq \text{ L1/L2 Cache Associativity } \leq 16 \end{cases}$
L1/L2 Cache Associativity
$\mu_A = \begin{cases} 0 & if \text{ Operating frequency } \leq 16MHz \text{ or } \geq 20MHz \\ 1 & if\ 16 \leq \text{ Operating frequency } \leq 20MHz \end{cases}$ $\mu_B = \begin{cases} 0 & if \text{ Operating frequency } \leq 20MHz \text{ or } \geq 25MHz \\ 1 & if\ 20MHz \leq \text{ Operating frequency } \leq 25MHz \end{cases}$ $\mu_C = \begin{cases} 0 & if \text{ Operating frequency } \leq 25MHz \text{ or } \geq 33MHz \\ 1 & if\ 25MHz \leq \text{ Operating frequency } \leq 33MHz \end{cases}$
Operating frequency

$$\mu_A = \begin{cases} 0 & if \text{ Number of cores } \leq 1 \text{ } or \geq 6 \\ 1 & if \text{ } 1 \leq \text{ Number of cores } \leq 6 \end{cases}$$

$$\mu_B = \begin{cases} 0 & if \text{ Number of cores } \leq 5 \text{ } or \geq 12 \\ 1 & if \text{ } 5 \leq \text{ Number of cores } \leq 12 \end{cases}$$

$$\mu_C = \begin{cases} 0 & if \text{ Number of cores } \leq 10 \text{ } or \geq 16 \\ 1 & if 10 \leq \text{ Number of cores } \leq 16 \end{cases}$$

Number of cores

Table 4: Timing and Energy Consumption of various Cache Configurations

Cache Size	**Associativity**	**Access Time[nsec]**	**Cycle Time[nsec]**	**Read Energy [J]**	**Write Energy[J]**
2 KB	1	4.74	2.87	1.86E-09	4.04E-10
4 KB	4	6.26	3.14	3.78E-09	6.96E-10
4 KB	8	6.33	3.10	7.03E-09	1.03E-09
8 KB	4	6.11	3.32	1.27E-08	1.52E-09
16 KB	1	5.39	3.48	4.23E-09	7.64E-10
32 KB	4	6.42	4.18	4.06E-08	3.77E-09
64 KB	4	7.52	3.90	2.39E-08	2.70E-09
64 KB	8	7.23	4.24	8.09E-08	6.10E-09
128 KB	8	7.92	4.38	8.53E-08	7.17E-09

Linux. 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 like the one presented here. The instruction execution, and cache hit/miss information was instrumented through Simics [55]. Interconnect network energy information was gathered by using Orion [34], cache energy and timing information was gathered by using CACTI [28], 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 [25], whereas the cache energy was calculated using the mathematical models presented in [29].
To profile thread execution statistics the Intel Concurrency Checker [56] was used, which provided information such as core utilization, thread distribution, percentage of parallelism and timing of the applications. 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 E-FLORE. Simics provides the facility of check-pointing through which, each time the machine parameters such as cache size, and associativity, and operating frequency were modified; and the number of cores were adjusted by using the Linux hotplug feature. The applications were re-executed for each iteration, so as to observe a clear 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 reconfiguration process is to have a balance between throughput and energy consumption of the SoC. In order to achieve this, the reconfiguration engine based on data of un-optimized core (iteration 0) starts modifying the reconfigurable parameters, i.e. Number of Cores, Operating Frequency, L1 Cache Size and Associativity (see Figure 4). The reconfiguration engine completed the system configuration in five iterations and results were found unvarying for all the subsequent iterations. The impact of reconfiguration on individual parameters is discussed as follows.

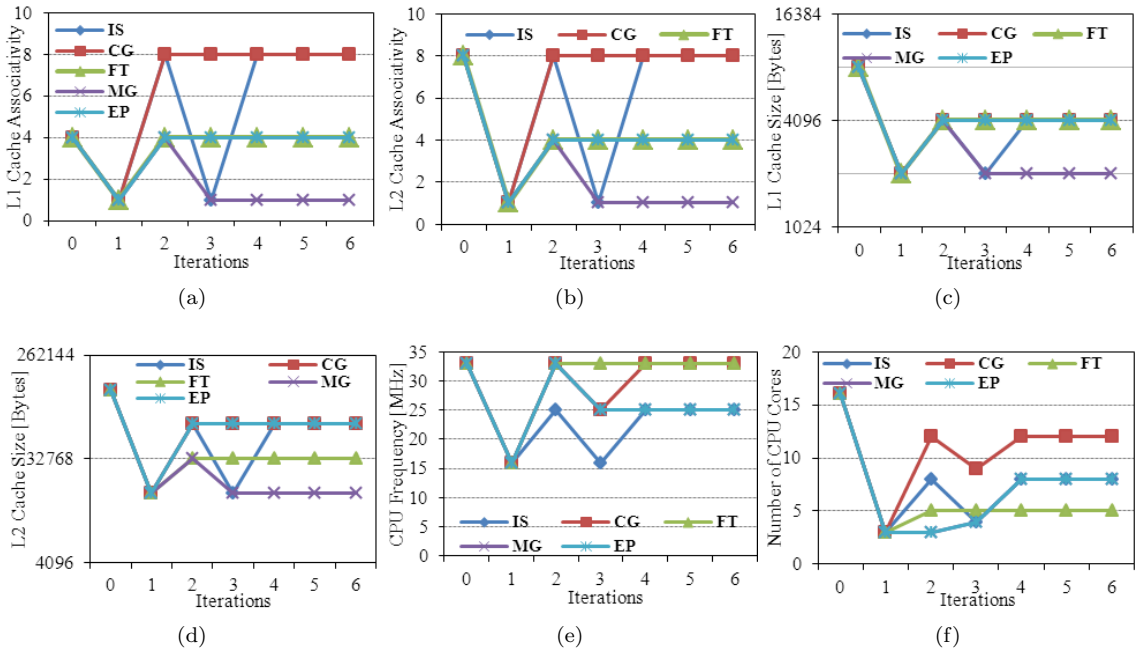

Figure 4: E-FLORE Results for (a) L1 Cache Associativity,(b) L2 Cache Associativity,(c) L1 Cache Size, (d) L2 Cache Size, (e) CPU Frequency, and (f) Number of Cores

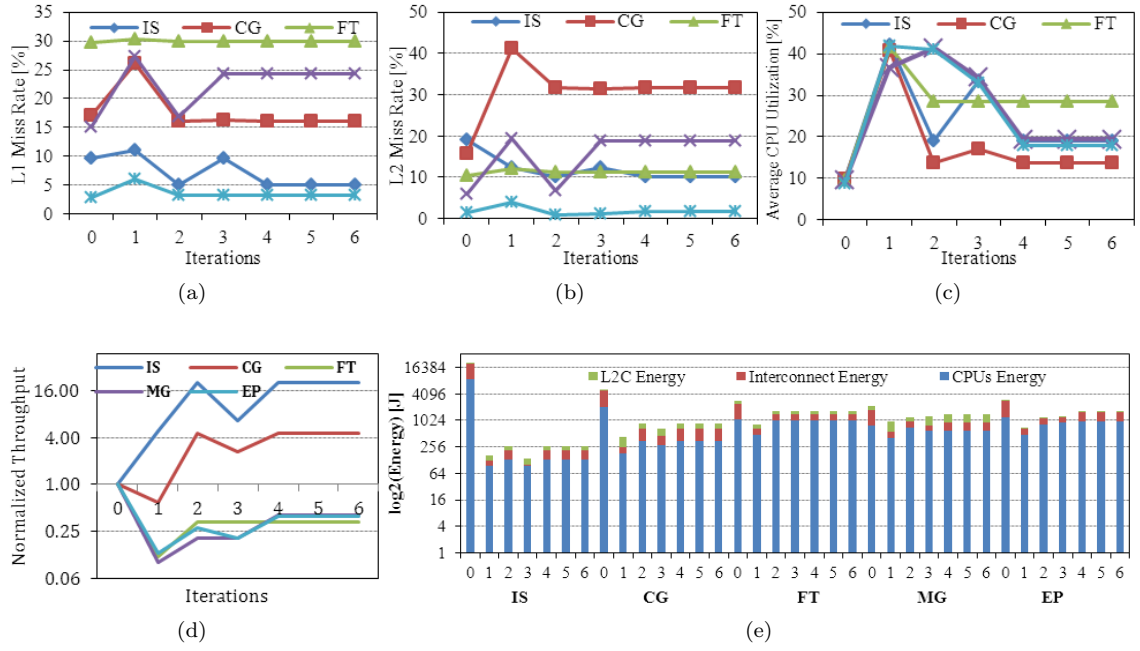

Figure 5: Impact of optimizations by E-FLORE on (a) L1 Miss Rate, (b) L2 Miss Rate, (c) Average CPU Utilization, (d) Normalized Throughput, and (e) Energy Consumption

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 highest the associativity is the best option to bring down the miss rate, which is the major cause of processor stalls i.e. the main contributor to energy wastage and throughput loss. However increasing the size and associativity of the cache not only increases the latency but also the energy consumption. The proposed fuzzy engine modified the associativity of the L1 cache from the default 4-way set associative to the 8-way set associative for IS and CG benchmarks, and 1-way or direct-mapped associativity for MG, and kept the same for the rest of the applications (see Figure 4(a)). The size of the cache for MG was configured to be 2Kbytes, and for the rest of applications 4Kbytes compared to the original 8Kbytes (see Figure 4(c)). These modifications resulted in the aggregate L1 miss rate to increase from 15% to 23% for MG, for IS it was reduced from 10% to 5%, and remained almost constant for all other applications (see Figure 5(a)).
It must be noted that the L1 miss rate is calculated on an aggregate basis for all the CPU cores in the MPSoC by averaging the individual miss rates for the purpose of simplification. Thus the same average miss rate 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 consumption is significantly decreased. This phenomenon can be observed from Table 4, which contains the different cache configurations used and their timing and energy consumption information based on CACTI [28].

4.2 L2 Cache

The proposed MPSoC architecture leverages the use of a shared level 2 (L2), uniform cache memory. For the L2 cache the E-FLORE configured the associativity to be 1-way set associative or direct-mapped for MG, and 4-way associative for EP applications, whereas for the rest of the applications it remains unchanged (see Figure 4(b)). The L2 cache size was decreased to 32KB for FT, 16KB for MG and for the rest of the applications it was changed to 64KB (see Figure 4(d)). The impact of these alterations on the miss rate is almost negligible for all the benchmarks except for CG and MG, where it has been increased to around 30% for CG compared to the default of 15%, and 18% for MG from the default value of 6% (see Figure 5 (b)). However in the case of CG a significant amount of energy savings and throughput increase can be observed (see Figure 5(e)), which is due to the impact of reducing cache size and number of cores by the E-FLORE.

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. As the E-FLORE addresses the energy and throughput of the system holistically at the same time, in final iterations the frequency of operation remains unchanged for all the applications, except for EP and IS where it is decreased to 25MHz (see Figure 4(e)). Whereas the number of cores was decreased to 5 for FT, 8 for MG, IS and EP, and 12 for CG (see Figure 4(f)). The impact of these changes on throughput can be observed in Figure 5(d) where the throughput was increased by almost 20 times for IS and 5 times for CG, whereas for the rest of the benchmarks it was decreased.
The reason for that is the applications are not prioritized statistically by the user but have been prioritized by the kernel itself. Secondly for any application; increasing the number of threads beyond a certain level actually decreases performance since the thread handling overhead will surpass the per thread execution time. This phenomenon is discussed in detail in [48]. For the rest of the applications, throughput had to be compromised in order to achieve greater energy savings and core utilization. This can be observed in Figure 5(c), where the core utilization for all the applications show a significant increase, as in the case of FT where it is almost 3 times greater and for CG there is more than a 40% increase compared to the nominal value of 10%. Similarly the energy consumption for all the benchmarks has been decreased significantly; as in the case of IS it is almost 80 times less than the original configuration and for FT it is around 60% of the default (see Figure 5 (e)). This is due to the fact that almost a third of the energy of the MPSoC is being consumed by the interconnect network. Thus reducing the number of nodes (i.e. CPU cores) significantly decreases the overall energy consumption of the SoC. Also the individual cores are connected to CMOS switches i.e. shutting down a core also eliminates any leakage energy consumption by them. The decrease in

the number of cores also reduces the L2 cache transactions and consequently greater overall energy savings can be achieved.

5 Conclusion

In this chapter a novel fuzzy logic based MPSoC reconfiguration scheme called Essex-Fuzzy Logic Reconfiguration Engine (E-FLORE) was presented. The fuzzy reconfiguration engine was used to find an optimal 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 Simics full system simulator. A detailed analysis of core, cache, and interconnect power consumption was conducted and overall a significant amount of energy saving with increased core utilization was observed. However, due to these optimizations in some cases, the device throughput was reduced with an increase in cache miss rate.
The system in general validated the use of the proposed Fuzzy Logic based technique for MPSoC reconfiguration; 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 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 Fuzzy Logic Rules

The fuzzy logic system provides a mean to form a rule base in linguistic terms. There is no set criteria to form a rule base therefore the overall performance of the system relies on the quality of rules. However the robustness of fuzzy systems does not allow the response to degrade intermittently as the quality of knowledge base degrades [57]. A set of rules is defined for E-FLORE relating input and output variables, and are detailed in Table 5, 6. For further explanation, an example can be taken of the first rule in Table 5, that can be read as
if "L1 Miss Rate" ***is*** LOW ***and*** "Energy Consumption" ***is*** LOW ***and*** "Throughput" ***is*** LOW ***then*** "L1 Cache Associativity" ***is*** NO CHANGE, "L1 Size" ***is*** NO CHANGE ***and*** "Clock Frequency" ***is*** HIGH

References

[1] M. Qadri and K. McDonald-Maier, "A fuzzy logic reconfiguration engine for symmetric chip multiprocessors," in *2010 International Conference on Complex, Intelligent and Software Intensive Systems*, IEEE. Krakow, Poland: IEEE Computer Society Washington, February 2010, pp. 937–943.

[2] ——, "A fuzzy logic based dynamic reconfiguration scheme for optimal energy and throughput in symmetric chip multiprocessors," in *Adaptive Hardware and Systems (AHS), 2010 NASA/ESA Conference on*, IEEE. Anaheim California, USA: IEEE, June 2010, pp. 333–339.

[3] A. Ibrahim, *Fuzzy Logic for Embedded Systems Applications*. Newton, MA, USA: Butterworth-Heinemann, 2003.

[4] G. M. Marakas, *Modern Data Warehousing, Mining, and Visualization: Core Concepts*. Pearson Education, 2002.

[5] T. Marescaux, A. Bartic, D. Verkest, S. Vernalde, and R. Lauwereins, "Interconnection networks enable fine-grain dynamic multi-tasking on fpgas," *Field-Programmable Logic and Applications: Reconfigurable Computing Is Going Mainstream*, vol. 2438/2002, pp. 741–763, 2002.

Table 5: Core level Rules for E-FLORE

L1 Miss Rate	**Energy Cons.**	**Throughput**	**L1 Cache Assoc.**	**L1 Size**	**Clock Freq.**
L	L	L	-	-	H
L	L	M	-	-	M
L	L	H	-	-	-
L	M	L	M	M	H
L	M	M	M	M	M
L	M	H	M	M	M
L	H	L	L	L	M
L	H	M	L	L	L
L	H	H	L	L	L
M	L	L	H	M	H
M	L	M	H	M	M
M	L	H	H	M	-
M	M	L	M	M	H
M	M	M	M	M	M
M	M	H	M	M	M
M	H	L	H	L	M
M	H	M	H	L	L
M	H	H	H	L	L
H	L	L	H	H	H
H	L	M	H	H	M
H	L	H	H	H	-
H	M	L	H	M	H
H	M	M	H	M	M
H	M	H	H	M	M
H	H	L	M	M	M
H	H	M	M	M	L
H	H	H	M	M	L

Table 6: SoC level Rules for E-FLORE

L2 Miss Rate	**Energy Cons.**	**Throughput**	**L2 Cache Assoc.**	**L2 Size**	**No. of Cores**
L	L	L	-	-	L
L	L	M	-	-	L
L	L	H	-	-	M
L	M	L	M	M	L
L	M	M	M	M	L
L	M	H	M	M	M
L	H	L	L	L	L
L	H	M	L	L	M
L	H	H	L	L	L
M	L	L	H	M	L
M	L	M	H	M	L
M	L	H	H	M	M
M	M	L	M	M	L
M	M	M	M	M	L
M	M	H	M	M	M
M	H	L	L	H	L
M	H	M	L	H	M
M	H	H	L	H	L
H	L	L	M	H	L
H	L	M	M	H	L
H	L	H	M	H	M
H	M	L	H	H	L
H	M	M	H	H	L
H	M	H	H	H	M
H	H	L	M	M	L
H	H	M	M	M	M
H	H	H	M	M	L

[6] J. Resano, D. Mozos, D. Verkest, and F. Catthoor, "A reconfigurable manager for dynamically reconfigurable hardware," *Design & Test of Computers, IEEE*, vol. 22, no. 5, pp. 452–460, 2005.

[7] K. Compton and S. Hauck, "Reconfigurable computing: a survey of systems and software," *ACM Computing Surveys (csuR)*, vol. 34, no. 2, pp. 171–210, 2002.

[8] H. Kalte and M. Porrmann, "Replica2pro: task relocation by bitstream manipulation in virtex-ii/pro fpgas," in *Proceedings of the 3rd conference on Computing frontiers*, ser. CF '06. New York, NY, USA: ACM, May 2006, pp. 403–412. [Online]. Available: http://doi.acm.org/10.1145/1128022.1128045

[9] K. Danne and M. Platzner, "A heuristic approach to schedule periodic real-time tasks on reconfigurable hardware," in *Field Programmable Logic and Applications, 2005. International Conference on*, IEEE. Tampere, Finland: IEEE, August 2005, pp. 568–573.

[10] K. Danne, R. Miihlenbernd, and M. Platzner, "Executing hardware tasks on dynamically reconfigurable devices under real-time conditions," in *Field Programmable Logic and Applications, 2006. FPL'06. International Conference on*, IEEE. Madrid, Spain: IEEE, August 2006, pp. 1–6.

[11] K. Danne and M. Platzner, "An edf schedulability test for periodic tasks on reconfigurable hardware devices," in *Proceedings of the 2006 ACM SIGPLAN/SIGBED conference on Language, compilers, and tool support for embedded systems*, ser. LCTES '06. New York, NY, USA: ACM, June 2006, pp. 93–102. [Online]. Available: http://doi.acm.org/10.1145/1134650.1134665

[12] P. Saha and T. El-Ghazawi, "Extending embedded computing scheduling algorithms for reconfigurable computing systems," in *Programmable Logic, 2007. SPL'07. 2007 3rd Southern Conference on*, IEEE. Mar del Plata, Argentina: IEEE, February 2007, pp. 87–92.

[13] ——, "Software/hardware co-scheduling for reconfigurable computing systems," in *Field-Programmable Custom Computing Machines, 2007. FCCM 2007. 15th Annual IEEE Symposium on*, IEEE. Napa Valley, California: IEEE, April 2007, pp. 299–300.

[14] M. DeVuyst, R. Kumar, and D. M. Tullsen, "Exploiting unbalanced thread scheduling for energy and performance on a cmp of smt processors," in *Proceedings of the 20th international conference on Parallel and distributed processing*, ser. IPDPS'06. Washington, DC, USA: IEEE Computer Society, April 2006, pp. 140–140. [Online]. Available: http://portal.acm.org/citation.cfm?id=1898953.1899070

[15] K. Li, "Performance analysis of power-aware task scheduling algorithms on multiprocessor computers with dynamic voltage and speed," *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, pp. 1484–1497, November 2008. [Online]. Available: http://portal.acm.org/citation.cfm?id=1441362.1441371

[16] P. Yang, C. Wong, P. Marchal, F. Catthoor, D. Desmet, D. Verkest, and R. Lauwereins, "Energy-aware runtime scheduling for embedded-multiprocessor socs," *Design & Test of Computers, IEEE*, vol. 18, no. 5, pp. 46–58, 2001.

[17] A. Prayati, C. Wong, P. Marchal, F. Catthoor, H. de Man, N. Cossement, R. Lauwereins, D. Verkest, and A. Birbas, "Task concurrency management experiment for power-efficient speed-up of embedded mpeg4 im1 player," in *Proceedings of the 2000 International Workshop on Parallel Processing*, ser. ICPP '00. Washington, DC, USA: IEEE Computer Society, August 2000, pp. 453–. [Online]. Available: http://portal.acm.org/citation.cfm?id=850942.852969

[18] Z. Ma, C. Wong, P. Yang, J. Vounckx, and F. Catthoor, "Mapping the mpeg-4 visual texture decoder: a system-level design technique based on heterogeneous platforms," *Signal Processing Magazine, IEEE*, vol. 22, no. 3, pp. 65–74, 2005.

[19] F. Bower, D. Sorin, and L. Cox, "The impact of dynamically heterogeneous multicore processors on thread scheduling," *Micro, IEEE*, vol. 28, no. 3, pp. 17–25, 2008.

[20] Y. Jiang, X. Shen, J. Chen, and R. Tripathi, "Analysis and approximation of optimal co-scheduling on chip multiprocessors," in *Proceedings of the 17th international conference on Parallel architectures and compilation techniques*, ser. PACT '08. New York, NY, USA: ACM, October 2008, pp. 220–229. [Online]. Available: http://doi.acm.org/10.1145/1454115.1454146

[21] T. Li, D. Baumberger, D. A. Koufaty, and S. Hahn, "Efficient operating system scheduling for performance-asymmetric multi-core architectures," in *Proceedings of the 2007 ACM/IEEE conference on Supercomputing*, ser. SC '07. New York, NY, USA: ACM, October 2007, pp. 53:1–53:11. [Online]. Available: http://doi.acm.org/10.1145/1362622.1362694

[22] J. H. Ahn, J. Leverich, R. Schreiber, and N. P. Jouppi, "Multicore dimm: an energy efficient memory module with independently controlled drams," *IEEE Comput. Archit. Lett.*, vol. 8, pp. 5–8, January 2009. [Online]. Available: http://portal.acm.org/citation.cfm?id=1591872.1591932

[23] J. Kim, D. Park, T. Theocharides, N. Vijaykrishnan, and C. R. Das, "A low latency router supporting adaptivity for on-chip interconnects," in *Proceedings of the 42nd annual Design Automation Conference*, ser. DAC '05. New York, NY, USA: ACM, December 2005, pp. 559–564. [Online]. Available: http://doi.acm.org/10.1145/1065579.1065726

[24] D. Park, C. Nicopoulos, J. Kim, N. Vijaykrishnan, and C. R. Das, "Exploring fault-tolerant network-on-chip architectures," in *Proceedings of the International Conference on Dependable Systems and Networks*. Washington, DC, USA: IEEE Computer Society, June 2006, pp. 93–104. [Online]. Available: http://portal.acm.org/citation.cfm?id=1135532.1135690

[25] Intel, *Embedded Ultra-Low Power Intel486 GX Processor Datasheet*, Intel Corporation, USA, 1997.

[26] H.-O. Kim, Y. Shin, H. Kim, and I. Eo, "Physical design methodology of power gating circuits for standard-cell-based design," in *Proceedings of the 43rd annual Design Automation Conference*, ser. DAC '06. New York, NY, USA: ACM, July 2006, pp. 109–112. [Online]. Available: http://doi.acm.org/10.1145/1146909.1146942

[27] Z. Hu, A. Buyuktosunoglu, V. Srinivasan, V. Zyuban, H. Jacobson, and P. Bose, "Microarchitectural techniques for power gating of execution units," in *Proceedings of the 2004 international symposium on Low power electronics and design*, ser. ISLPED '04. New York, NY, USA: ACM, August 2004, pp. 32–37. [Online]. Available: http://doi.acm.org/10.1145/1013235.1013249

[28] D. Tarjan, S. Thoziyoor, and N. Jouppi, "Cacti 4.0," *HP Laboratories Palo Alto, Tech. Rep. HPL-2006-86*, vol. 1, 2006.

[29] M. Qadri and K. McDonald-Maier, "Analytical evaluation of energy and throughput for multilevel caches," in *2010 12th International Conference on Computer Modelling and Simulation*, IEEE. Cambridge, UK: IEEE Computer Society, March 2010, pp. 598–603.

[30] L. Shang, L.-S. Peh, and N. K. Jha, "Dynamic voltage scaling with links for power optimization of interconnection networks," in *Proceedings of the 9th International Symposium on High-Performance Computer Architecture*, ser. HPCA '03. Washington, DC, USA: IEEE Computer Society, February 2003, p. 91. [Online]. Available: http://portal.acm.org/citation.cfm?id=822080.822800

[31] M. Monchiero, R. Canal, and A. González, "Design space exploration for multicore architectures: a power/performance/thermal view," in *Proceedings of the 20th annual international conference on Supercomputing*, ser. ICS '06. New York, NY, USA: ACM, May-June 2006, pp. 177–186. [Online]. Available: http://doi.acm.org/10.1145/1183401.1183428

[32] S. Thoziyoor, J. H. Ahn, M. Monchiero, J. B. Brockman, and N. P. Jouppi, "A comprehensive memory modeling tool and its application to the design and analysis of future memory hierarchies," in *Proceedings of the 35th Annual International Symposium on Computer Architecture*, ser. ISCA '08. Washington, DC, USA: IEEE Computer Society, June 2008, pp. 51–62. [Online]. Available: http://dx.doi.org/10.1109/ISCA.2008.16

[33] L. Guang, E. Nigussie, and H. Tenhunen, "System-level exploration of run-time clusterization for energy-efficient on-chip communication," in *Proceedings of the 2nd International Workshop on Network on Chip Architectures*, ser. NoCArc '09. New York, NY, USA: ACM, December 2009, pp. 63–68. [Online]. Available: http://doi.acm.org/10.1145/1645213.1645228

[34] A. B. Kahng, B. Li, L.-S. Peh, and 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*, ser. DATE '09. 3001 Leuven, Belgium, Belgium: European Design and Automation Association, April 2009, pp. 423–428. [Online]. Available: http://portal.acm.org/citation.cfm?id=1874620.1874721

[35] H.-S. Wang, X. Zhu, L.-S. Peh, and S. Malik, "Orion: a power-performance simulator for interconnection networks," in *Proceedings of the 35th annual ACM/IEEE international symposium on Microarchitecture*, ser. MICRO 35. Los Alamitos, CA, USA: IEEE Computer Society Press, November 2002, pp. 294–305. [Online]. Available: http://portal.acm.org/citation.cfm?id=774861.774893

[36] L. Wu, X. Su, P. Shi, and J. Qiu, "Model approximation for discrete-time state-delay systems in the t-s fuzzy framework," *IEEE Transactions on Fuzzy Systems*, vol. 19, no. 2, pp. 366–378, 2011.

[37] ——, "A new approach to stability analysis and stabilization of discrete-time ts fuzzy time-varying delay systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 41, no. 1, pp. 273–286, 2011.

[38] X. Zhang, Z. Zhang, and G. Lu, "Fault detection for state-delay fuzzy systems subject to random communication delay," *International Journal of Innovative Computing, Information and Control (ICIC)*, vol. 8, no. 4, pp. 2439–2451, April 2012.

[39] S.-H. Yang, W.-J. Wang, C.-Y. Chen, C.-H. Hsu, and P.-H. Chou, "The impedance based fuzzy logic control for the cathode air flow of a direct methanol fuel cell system," *International Journal of Innovative Computing, Information and Control (ICIC)*, vol. 7, no. 2, pp. 625–635, February 2011.

[40] M. Khalid, R. Yusof, and H. Mokayed, "Fusion of multi-classifiers for online signature verification using fuzzy logic inference," *International Journal of Innovative Computing, Information and Control (ICIC)*, vol. 7, no. 5(B), pp. 2709–2726, May 2011.

[41] X. Su, L. Wu, P. Shi, and Y.-D. Song, "H8 model reduction of takagisugeno fuzzy stochastic systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. PP, no. 99, pp. 1–12, May 2012.

[42] R. Prakash and R. Anita, "Modeling and simulation of fuzzy logic controller-based model reference adaptive controller," *International Journal of Innovative Computing, Information and Control (ICIC)*, vol. 8, no. 4, pp. 2533–2550, April 2012.

[43] C.-J. Huang, K.-W. Hu, H.-M. Chen, T.-K. Chang, Y.-C. Luo, and Y.-J. Lien, "Application of type-2 fuzzy logic to rule-based intrusion alert correlation detection," *International Journal of Innovative Computing, Information and Control (ICIC)*, vol. 8, no. 4, pp. 2865–2874, April 2012.

[44] X. Su, P. Shi, L. Wu, and Y. Song, "A novel approach to filter design for ts fuzzy discrete-time systems with time-varying delay," vol. PP, no. 99, p. 1, 2012.

[45] T. Chen, "A hybrid fuzzy and neural approach with virtual experts and partial consensus for dram price forecasting," *International Journal of Innovative Computing, Information and Control (ICIC)*, vol. 8, no. 1(B), pp. 583–597, January 2012.

[46] M. H. Bahari, A. Karsaz, and N. Pariz, "High maneuvering target tracking using a novel hybrid kalman filter-fuzzy logic architecture," *International Journal of Innovative Computing, Information and Control (ICIC)*, vol. 7, no. 2, pp. 501–510, February 2011.

[47] I.-J. Ding, "Enhancements of maximum likelihood eigen-decomposition using fuzzy logic control for eigenvoice-based speaker adaptation," *International Journal of Innovative Computing, Information and Control (ICIC)*, vol. 7, no. 7(B), pp. 4207–4222, July 2011.

[48] C. De Silva, *Intelligent control: fuzzy logic applications*. USA: CRC Press, 1995.

[49] S. Rabin, *AI game programming wisdom*. Hingham, Massachusetts: Charles River Media, Inc., 2002.

[50] IEC, *International Standard: Programmable controllers - Part 7: Fuzzy control programming*, International Electrotechnical Commission, Geneva, Switzerland, 2000, iEC Standard.

[51] H. Jin, M. Frumkin, and J. Yan, *The OpenMP implementation of NAS parallel benchmarks and its performance*, 1999.

[52] L. Dagum and R. Menon, "Openmp: an industry standard api for shared-memory programming," *Computational Science & Engineering, IEEE*, vol. 5, no. 1, pp. 46–55, 1998.

[53] W. Saphir, R. Van der Wijngaart, A. Woo, and M. Yarrow, "New implementations and results for the nas parallel benchmarks 2," in *In 8th SIAM Conference on Parallel Processing for Scientific Computing*. Citeseer, MArch 1997.

[54] G. Marsaglia and T. Bray, "A convenient method for generating normal variables," *Siam Review*, vol. 6, no. 3, pp. 260–264, 1964.

[55] P. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, G. Hållberg, J. H
"ogberg, F. Larsson, A. Moestedt, and B. Werner, "Simics: A full system simulation platform," *COMPUTER,*, vol. 35, no. 2, pp. 50–58, 2002.

[56] Intel, *Intel Concurrency Checker*, Intel Corporation, USA, 2008.

[57] A. M. Zalzala and P. J. Fleming, Eds., *Genetic Algorithms in Engineering Systems*. Stevenage, UK, UK: Institution of Electrical Engineers, 1997.