A Fuzzy Logic Reconfiguration Engine for Symmetric Chip Multiprocessors

Muhammad Yasir Qadri, Klaus D. McDonald-Maier
School of Computer Science and Electronic Engineering
University of Essex, CO4 3SQ
Colchester, United Kingdom
yasirqadri@acm.org, kdm@essex.ac.uk

Abstract— **Recent developments in reconfigurable multiprocessor system on chip (MPSoC) have offered system designers a great amount of flexibility to exploit task concurrency with higher throughput and less energy consumption. This paper presents a novel fuzzy logic reconfiguration engine (FLRE) for coarse grain MPSoC reconfiguration that facilitates to identify an optimum balance between power and performance of the system. The FLRE is composed on two levels of abstraction layers. The system selects an optimal configuration of Level 1 / Level 2 cache size and Associativity, processor operating frequency and voltage, the number of cores based on miss rate, and energy and throughput information of the system both at core and SoC level. An 8-core symmetric chip multiprocessor has been used to evaluate the proposed scheme. The results show an overall decrease of energy consumption with not more than 30% decrease in the throughput.**

Keywords- Fuzzy Logic; Reconfigurable Hardware; Symmetric Chip multiprocessors; Energy; Performance

I. INTRODUCTION

Fuzzy logic is a popular soft computing technique that is suited for runtime hardware adaptive systems as it is straightforward and lightweight to implement [1]. Fuzzy based systems are particularly well suited for applications where absolute precision is less important in making a general improvement to the configuration. With the advent of modern reconfigurable multiprocessor architectures it has now become possible to apply such techniques to find a better compromise between energy and throughput of a system.

This paper presents a novel fuzzy logic reconfiguration engine (FLRE) for coarse grain MPSoC reconfiguration to find an optimum balance between power and performance of the system. The FLRE is composed on two levels of abstraction layers. The FLRE at primary level or core level reconfigures core frequency/voltage, level 1 (L1) cache size and Associativity based on the miss rate, throughput and energy consumption of the core. At the secondary level or SoC level, Level 2 (L2) cache Associativity and size, along with number of active cores are selected on the basis of L2 miss rate, throughput and energy consumption of the entire system. This reconfigurable cache architecture can be implemented in a runtime reconfigurable FPGA platform or alternatively by using hardware cache partitioning schemes such as the one proposed by Settle et al. [2].

This paper is organized into five sections. The following section covers the related research in the field of reconfigurable architectures and hardware adaption techniques. The section 3 describes the experimental setup and the FLRE. The subsequent two sections present the results, conclusion and future research directions.

II. RELATED WORK

The majority of research in reconfigurable architectures has been done on field programmable gate arrays (FPGAs) as the target platform. Some of these FPGAs also provide runtime reconfiguration option which is largely being used in adaptive hardware scenarios to provide optimized energy consumption and performance, and to support larger configurations than the available area on the device. However FPGAs suffer from higher reconfiguration latencies as discussed in [3-5]. This issue has largely been addressed by preempting task implementation schemes for all possible locations at design time and then during the runtime a placement algorithm selects proper bitstream to be downloaded. One such approach was presented by Resano et al. in [5] using a dynamically reconfigurable hardware (DRH) model which facilitates task migration and inter-task communication. These authors have also proposed a hardware reconfiguration manager adopting prefetch scheduling and task replacement scheme designed to hide reconfiguration latency by more than 93% even for highly dynamic applications. Another example of such an approach was proposed by Kalte et al. [6] in which they have 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. [7] proposed two periodic real-time tasks 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 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 could be found in [8-11].

MPSoCs have been investigated for runtime energy aware scheduling in several research articles. Thread scheduling is generally classified into 1) balanced and 2) unbalanced scheduling categories. Balanced scheduling has an equal amount of threads distributed among the cores. DeVuyst et al. [12] have analyzed 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 these among several cores. Reconfigurable multicore platforms also pose challenges in handling the communication between dynamically changing tasks and their synchronization. Li [13] has performed a detailed analysis of the performance of various task scheduling algorithms for minimizing schedule length combined with an energy consumption constraint and for minimizing energy consumption combined with a schedule length constraint on Dynamic Voltage and Frequency (DVF) supported multiprocessor systems. Yang et al. [14] 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 runtime. 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. [15]. Ma et al. [16] discuss a design time and runtime 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 runtime a lightweight scheduler was used to select optimal working points exploiting dynamic and nondeterministic behavior of the system. For an MPEG4 texture decoder application their approach has shown significant improvement in performance while maintaining lower energy consumption. Other reference to related work in this field can be found in [17-19].

Considering memory as the best candidate for optimization in an energy constrained multicore scenario, Ahn et al. [20] presented a simplified approach of grouping 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 demultiplexor 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 Networks-on-Chip (NoC). Kim et al. [21] have presented a novel low latency router architecture with two stage pipeline employing adaptive routing scheme for congestion aware flow control. The router architecture was termed as path sensitive, as it utilizes look-ahead routing for selecting the next route based on the four possible quadrants and routes the packet to the corresponding virtual channels assigned to that quadrant. Additionally, based on this portioned approach a decomposed crossbar switch was proposed that results 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. [22] 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 router pipeline such as errors caused by virtual channel allocators, routing units, switch allocators, and crossbars.

III. System Description and Experimental Setup

An 8-core symmetric chip multiprocessor (SCMP) platform was designed to be evaluated on the proposed reconfiguration scheme. The SCMP is based on the Intel x86 architecture, with a customised shared memory architecture. The platform is comprised of L1 and L2 caches with configurable size and Associativity. The number of cores and processor frequency/voltage could be adjusted for energy and throughput regulation. The system configuration and parameters are described in Table 1. The energy consumption and voltage/frequency information was obtained from the Intel 486 GX embedded processor datasheet [23]. Each core of the system is connected to a CMOS switch, which allows to turn off the core, and prevent the leakage energy of the core from contributing to the overall energy consumption of the system. The default size and Associativity for L1 and L2 caches are 8KB, and 4-way set associative; and 128KB and 8-way set associative. The miss rate for L1 cache was assumed to be 10 cycles and that for L2 cache was set as 30 cycles. Each core in the SCMP is linked through a router that provides a seamless communication interface to the other members of this NoC. The router architecture was selected as a virtual channel arrangement with 5-stage pipeline similar to the one proposed by Peh et al. [24].

TABLE I. System Parameters

Parameter	Value
Processor Type	Intel x86
Number of Cores	8
Operating Frequencies	[16, 20, 25, 33] MHz
Operating Voltages	[2, 2.2, 2.4, 2.7]V
Energy Consumption per cycle	[13.1,15.4,18.7,22.9]nJ

In order to search for an optimal configuration, a fuzzy logic reconfiguration engine based on Mamdani's [25] inference technique was employed. A detailed description of each membership function for input variables such as L1 and L2 miss rate, normalized energy consumption, and throughput; and output parameters such as L1 and L2 Cache

Associativity and size, operating frequency and number of cores is given in Table 2.

To establish the relationship between the input variables and output parameters of the SoC, fuzzy logic rules were defined as shown in Appendix B. The rules were formed in way that a balanced throughput and energy consumption ratio could be achieved. For primary or core level configuration the FLRE keeps track of the average L1 miss rate, energy consumption and throughput for all the cores and strives to find an optimum cache size, Associativity and operating frequency. The cache size and Associativity does not only affect the miss rate but also have an impact on the throughput and energy consumption of the device. Similarly for the secondary or system level configuration the FLRE strives to find an optimal number of cores and L2 cache size and Associativity while taking into account the L2 miss rate and total throughput and energy consumption of the SoC.

TABLE II. INPUT AND OUTPUT MEMBERSHIP FUNCTIONS OF FLRE

$$\mu_A = \begin{cases} 0 & \text{if } L1/L2 \text{ Miss rate} \le 0\% \text{ or } \ge 40\% \\ \dfrac{40-x}{40} & \text{if } 0\% \le L1/L2 \text{ Miss rate} \le 40\% \end{cases}$$

$$\mu_B = \begin{cases} 0 & \text{if } L1/L2 \text{ Miss rate} \le 0\% \text{ or } \ge 75\% \\ \dfrac{x-25}{25} & \text{if } 25\% \le L1/L2 \text{ Miss rate} \le 50\% \\ \dfrac{75-x}{25} & \text{if } 50\% \le L1/L2 \text{ Miss rate} \le 75\% \end{cases}$$

$$\mu_c = \begin{cases} 0 & \text{if } L1/L2 \text{ Miss rate } \le 60\% \text{ or} \ge 100\% \\ \dfrac{x-60}{40} & \text{if } 60\% \le L1/L2 \text{ Miss rate} \le 100\% \end{cases}$$

(a) L1 and L2 Miss rate

$$\mu_A = \begin{cases} 0 & \text{if Energy cons.} \le 0 \text{ or} \ge 0.35 \\ \dfrac{0.35-x}{0.35} & \text{if } 0 \le \text{Energy cons.} \le 0.35 \end{cases}$$

$$\mu_B = \begin{cases} 0 & \text{if Energy cons.} \le 0 \text{ or } \ge 0.8 \\ \dfrac{x-0.2}{0.3} & \text{if } 0.2 \le \text{Energy cons.} \le 0.5 \\ \dfrac{0.8-x}{0.3} & \text{if } 0.5 \le \text{Energy cons.} \le 0.8 \end{cases}$$

$$\mu_c = \begin{cases} 0 & \text{if Energy cons.} \le 0.65 \text{ or } \ge 1.0 \\ \dfrac{x-0.65}{0.35} & \text{if } 0.65 \le \text{Energy cons.} \le 1.0 \end{cases}$$

(b) Normalized Energy Consumption

$$\mu_A = \begin{cases} 0 & \text{if Throughput} \le 0 \text{ or } \ge 0.35 \\ \dfrac{0.35-x}{0.35} & \text{if } 0 \le \text{Throughput} \le 0.35 \end{cases} \quad \mu_B = \begin{cases} 0 & \text{if Throughput} \le 0 \text{ or} \ge 0.8 \\ \dfrac{x-0.2}{0.3} & \text{if } 0.2 \le \text{Throughput} \le 0.5 \\ \dfrac{0.8-x}{0.3} & \text{if } 0.5 \le \text{Throughput} \le 0.8 \end{cases}$$

$$\mu_c = \begin{cases} 0 & \text{if Throughput} \le 0.65 \text{ or} \ge 1.0 \\ \dfrac{x-0.65}{0.35} & \text{if } 0.65 \le \text{Throughput} \le 1.0 \end{cases}$$

(c) Normalized Throughput

$$\mu_A = \begin{cases} 0 & \text{if } L1/L2 \text{ Cache Assoc.} \le 0 \text{ or } \ge 2 \\ 1 & \text{if } 0 \le L1/L2 \text{ Cache Assoc.} \le 2 \end{cases} \quad \mu_B = \begin{cases} 0 & \text{if } L1/L2 \text{ Cache Assoc.} \le 1 \text{ or } \ge 8 \\ 1 & \text{if } 1 \le L1/L2 \text{ Cache Assoc.} \le 8 \end{cases}$$

$$\mu_C = \begin{cases} 0 & \text{if } L1/L2 \text{ Cache Assoc.} \le 4 \text{ or } \ge 16 \\ 1 & \text{if } 4 \le L1/L2 \text{ Cache Assoc.} \le 16 \end{cases}$$

(d) L1 and L2 Cache Associativity

$$\mu_A = \begin{cases} 0 & \text{if } L1 \text{ Cache size} \le 0KB \text{ or} \ge 3.5KB \\ \dfrac{3.5-x}{3.5} & \text{if } 0KB \le L1 \text{ Cache size} \le 3.5\,KB \end{cases}$$

$$\mu_B = \begin{cases} 0 & \text{if } L1 \text{ Cache size} \le 0KB \text{ or } \ge 7KB \\ \dfrac{x-2}{2.5} & \text{if } 2KB \le L1 \text{ Cache size} \le 4.5\,KB \\ \dfrac{7-x}{2.5} & \text{if } 4.5KB \le L1 \text{ Cache size} \le 7KB \end{cases}$$

$$\mu_c = \begin{cases} 0 & \text{if } L1 \text{ Cache size} \le 5.5KB \text{ or } \ge 8KB \\ \dfrac{x-0.65}{0.35} & \text{if } 5.5KB \le L1 \text{ Cache size} \le 8KB \end{cases}$$

(e) L1 Cache Size

$$\mu_A = \begin{cases} 0 & \text{if Operating freq.} \le 16MHZ \text{ or } \ge 20MHz \\ 1 & 16MHz \le \text{Operating freq.} \le 20MHz \end{cases}$$

$$\mu_B = \begin{cases} 0 & \text{if Operating freq.} \le 20MHz \text{ or } \ge 25MHz \\ 1 & \text{if } 20MHz \le \text{Operating freq.} \le 25MHz \end{cases}$$

$$\mu_C = \begin{cases} 0 & \text{if Operating freq.} \le 25MHz \text{ or } \ge 33MHz \\ 1 & \text{if } 25MHz \le \text{Operating freq.} \le 33MHz \end{cases}$$

(f) Operating Frequency

$$\mu_A = \begin{cases} 0 & \text{if } L2 \text{ Cache size} \le 0KB \text{ or} \ge 50KB \\ \dfrac{50-x}{50} & \text{if } 0KB \le L2 \text{ Cache size} \le 50KB \end{cases}$$

$$\mu_B = \begin{cases} 0 & \text{if } L2 \text{ Cache size} \le 20KB \text{ or } \ge 100KB \\ \dfrac{x-20}{40} & \text{if } 20KB \le L2 \text{ Cache size} \le 60KB \\ \dfrac{100-x}{40} & \text{if } 60KB \le L2 \text{ Cache size} \le 100KB \end{cases}$$

$$\mu_c = \begin{cases} 0 & \text{if } L2 \text{ Cache size} \le 80KB \text{ or } \ge 128KB \\ \dfrac{x-80}{48} & \text{if } 80KB \le L2 \text{ Cache size} \le 128KB \end{cases}$$

(g) L2 Cache size

$$\mu_A = \begin{cases} 0 & \text{if No.of cores} \le 1 \text{ or } \ge 3 \\ 1 & 1 \le \text{No.of cores } \le 3 \end{cases} \quad \mu_B = \begin{cases} 0 & \text{if No.of cores} \le 2 \text{ or } \ge 6 \\ 1 & \text{if } 2 \le \text{No.of cores} \le 6 \end{cases}$$

$$\mu_C = \begin{cases} 0 & \text{if No.of cores} \le 5 \text{ or } \ge 8 \\ 1 & \text{if } 5 \le \text{No.of cores} \le 8 \end{cases}$$

(h) Number of Cores

The fuzzy logic reconfiguration scheme was tested on the proposed MPSoC using Simics full system simulator. Simics [26] facilitates instruction level simulations and is capable to run unmodified OS such as VxWorks, Solaris, Linux, Tru64, and Windows XP virtually on the target platforms. The simulator is targeted to provide fairly accurate timing profile, but at present does not support energy profiling of a target system. Simics also provides a fairly accurate cache profiling utility making it ideal for memory system research. Fedora v10 Linux operating system was used on the target platform, as it supports Advanced Configuration and Power Interface (ACPI) for hot-plugging (i.e. turning on/off) a CPU core on the go which is a vital feature for reconfigurable MPSoC scenarios such as the one presented here. Cache energy information was obtained from CACTI [27] which is an open source standard tool for highly accurate cache energy and timing analysis. However it is not a trace driven simulator, so energy consumption resulting in number of hits or misses is not accounted for a particular application. The information obtained from CACTI along with cache hit and miss rate profile provided by Simics was used to find the total cache energy consumption for the sample interval. The core interconnect energy consumption was estimated from Orion a power-performance simulator for interconnection networks [28-30]. Three OpenMP [31] based Class B, benchmark applications namely BT (Block Tridiagonal), CG (Conjugate Gradient), and LU (Lower-Upper symmetric Gauss-Seidel algorithm [32]) from the NAS parallel benchmark suite were executed on the target platform to evaluate the performance of the proposed scheme. The thread scheduling was done statically, and the thread profile was collected using Intel Concurrency Checker [33] which provided data such as core utilization, thread distribution, percentage of parallelism and timing of the applications. All applications were sampled for the first five seconds and then reconfiguration was done through decisions made by the fuzzy logic engine. Simics provides a facility of check-pointing through which the current state of the system can be saved and then machine parameters can be modified in the checkpoint file. When this file is reloaded into the simulator, the simulation resumes from the state at which it was previously saved, however with new parameters. This feature was exploited for each iteration to modify parameters such as cache size, and Associativity, and operating frequency of the target platform. The number of cores in the SoC was adjusted by using the Linux hotplug feature. The applications were re-executed for each iteration, since, as the application proceeds in execution, the sampled average cache miss rate keeps on changing, so a clear impact of cache reconfiguration could not be judged and the same is the case for dynamic thread scheduling.

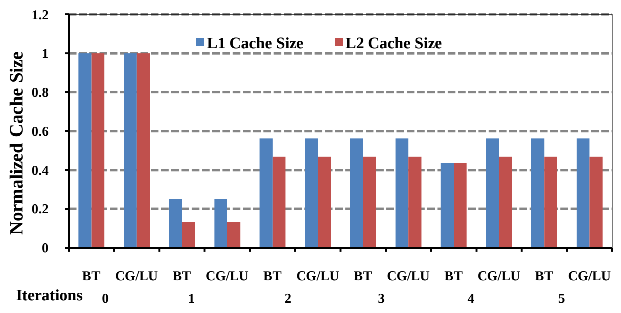


Figure 1. FLRE Results for Cache Sizing

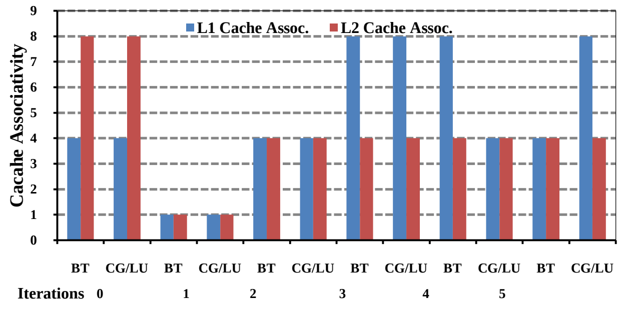


Figure 2. FLRE Results for Cache Associativity

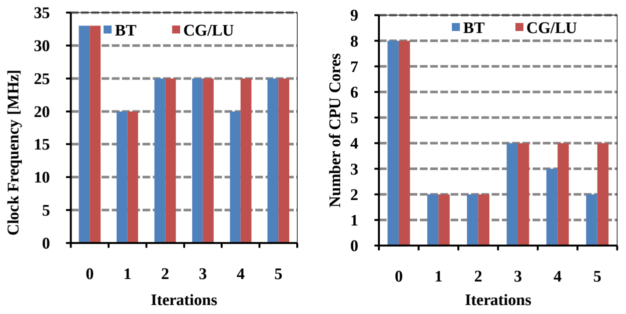


Figure 3. FLRE Results for Clock Frequency and Number of Cores

IV. Results

The FLRE started optimizing the configuration parameters by first taking the data from the un-optimized core (Iteration 0). For each iteration, core and system level optimizations were passed to the target system and the benchmark applications were re-executed for a consistent miss rate profiling. The main objective of the system is to search for an optimum solution for performance and energy of the MPSoC. The inference engine completed the system configuration in five iterations and results were found to be invariant for all the subsequent iterations. The operation of the FLRE is shown in Figures 1, 2, and 3; where for input variables such as core utilization, throughput, and energy consumption; parameters such as L1 and L2 cache size (Figure 1) and Associativity (Figure 2), processor frequency and number of cores (Figure 3) are being optimized. For each of the iterations, the resultant core utilization, throughput, L1 and L2 miss rate along with the total energy consumption are shown in Figures 4, 5, and 6. The L1 cache size was finally configured as 4.5KB, for all applications while the Associativity was selected as 4-way and 8-way set Associative for BT and CG/LU benchmarks respectively, as compared to the original 8KB, 4-way set associative cache. The L2 cache size and Associativity was optimized as 60KB, 4-way set associative cache; while the original configuration was 128KB, 8-way set associative cache. The number of cores has been reduced from 8 to 2 for BT, and for CG/LU to 4, while the frequency of operation has been selected as 25MHz where the default was 33MHz. The optimizations resulted in an overall 100% increase in core utilization for BT, and around 38% increase for CG, and 67% for LU applications. The energy consumption has been reduced by more than 6 times for BT, and by approximately two times for CG and LU applications. The system throughput has been decreased to 74% for the BT, 91% for the CG, and 82% for the LU benchmarks when compared with the one for default configuration. A significant increase in the miss rate in L1 and L2 could be observed but that is a result of finding an optimal balance in the system's performance and energy consumption. An infinitely large cache with highest Associativity is an ideal solution for the least possible miss rate. However the cache energy and throughput greatly varies with its size and Associativity. Therefore the miss rate was compromised to an extent in order to permit greater energy savings for the overall system.

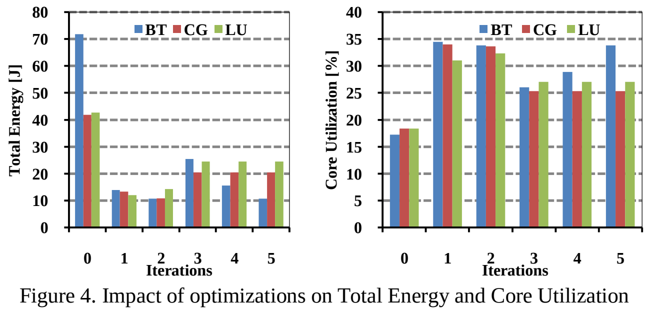


Figure 4. Impact of optimizations on Total Energy and Core Utilization

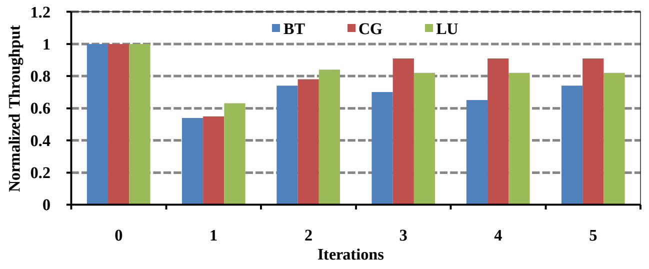


Figure 5. Impact of optimizations on Throughput

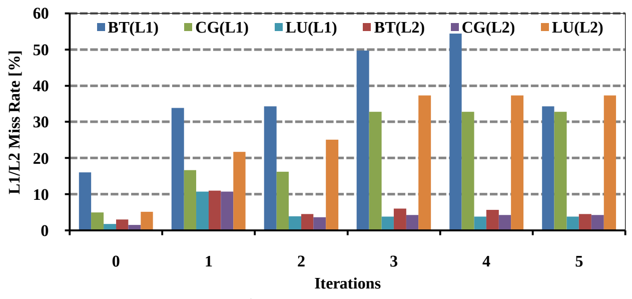


Figure 6. Impact of optimizations on L1 and L2 Miss Rates

V. Conclusion and Future Directions

This paper has presented a novel fuzzy logic based MPSoC reconfiguration scheme. The fuzzy reconfiguration engine was used to find an optimal balance between energy consumption and performance of the system. To evaluate the proposed scheme an Intel x86 based multicore SoC with 8 processor cores and a shared memory architecture, was simulated using Simics full system simulator. The SoC architecture included power gated cores for minimal energy consumption whilst not in use. A detailed analysis of core, cache, and interconnect power consumption was conducted and a significant amount of energy saving with increased core utilization has been observed. However due to these optimizations, 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 a good compromise for throughput and energy under user defined constraints. The proposed MPSoC architecture can be tailored to be used in variety of applications such as NoC research, dynamic thread scheduling, operating system development and high performance computing. Future work will see dynamic thread scheduling applied to the system for it to be able to reconfigure while executing a task.

References

[1] V. Kadirkamanathan, "Fuzzy Logic and Control: Software and Hardware Applications. Mohammad Jamshidi, Nader Vadiee and Timothy J. Ross (eds.)," *Artificial Intelligence Review,* vol. 13, pp. 337-339 1999.

[2] A. Settle, D. Connors, E. Gibert, and A. Gonzalez, "A dynamically reconfigurable cache for multithreaded processors," *Journal of Embedded Computing,* vol. 2, pp. 221-233, 2006.

[3] K. Compton and S. Hauck, "Reconfigurable computing: a survey of systems and software," *ACM Computing Surveys (CSUR),* vol. 34, pp. 171-210, 2002.

[4] T. Marescaux, A. Bartic, D. Verkest, S. Vernalde, and R. Lauwereins, "Interconnection networks enable fine-grain dynamic multi-tasking on FPGAs," in *Lecture Notes in Computer Science.* vol. 2438 Montpellier, France, 2002, pp. 795-805.

[5] J. Resano, D. Mozos, D. Verkest, and F. Catthoor, "A reconfigurable manager for dynamically reconfigurable hardware," *IEEE Design & Test of Computers,* vol. 22, pp. 452-460, 2005.

[6] 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* Ischia, Italy: ACM, 2006, pp. 403-412.

[7] K. Danne and M. Platzner, "A Heuristic Approach to Schedule Periodic Real-Time Tasks on Reconfigurable Hardware," in *Proceedings of the International Conference on Field Programmable Logic and Applications (FPL) 2005* Tempere, Finland, 2005, pp. 568-573.

[8] K. Danne, R. Miihlenbernd, and M. Platzner, "Executing Hardware Tasks on Dynamically Reconfigurable Devices Under Real-Time Conditions," in *Processdings of The International Conference on Field Programmable Logic and Applications, 2006. FPL '06.* MAdrid, Spain, 2006, pp. 1-6.

[9] 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* Ottawa, Ontario, Canada: ACM, 2006, pp. 93-102.

[10] P. Saha and T. El-Ghazawi, "Extending Embedded Computing Scheduling Algorithms for Reconfigurable Computing Systems," in *Proceedings of 3rd Southern Conference on Programmable Logic, 2007. SPL '07. 2007* Mar del Plata, Argentina, 2007, pp. 87-92.

[11] P. Saha and T. El-Ghazawi, "Software/Hardware Co-Scheduling for Reconfigurable Computing Systems," in *Proceedings of 15th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, 2007. FCCM 2007.* Napa Valley, California, 2007, pp. 299-300.

[12] 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 IEEE/ACM International Parallel and Distributed Processing Symposium* Rhodes Island, Greece: IEEE/ACM, 2006.

[13] 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, 2008.

[14] P. Yang, C. Wong, P. Marchal, F. Catthoor, D. Desmet, D. Verkest, and R. Lauwereins, "Energy-aware runtime scheduling for embedded-multiprocessor SOCs," *IEEE Design & Test of Computers,* vol. 18, pp. 46-58, 2001.

[15] A. Prayati, W. Chun, P. Marchal, N. Cossement, F. Catthoor, R. Lauwereins, D. Verkest, H. De Man, and A. Birbas, "Task concurrency management experiment for power-efficient speed-up of embedded MPEG4 IM1 player," in *Proceedings of International Workshops on Parallel Processing, 2000.* Toronto, Canada, 2000, pp. 453-460.

[16] Z. Ma, C. Wong, P. Yang, J. Vounckx, F. Catthoor, I. M. Center, and B. Leuven, "Mapping the MPEG-4 visual texture decoder: a system-level design technique based on heterogeneous platforms," *IEEE Signal Processing Magazine,* vol. 22, pp. 65-74, 2005.

[17] F. A. Bower, D. J. Sorin, and L. P. Cox, "The Impact of Dynamically Heterogeneous Multicore Processors on Thread Scheduling," *IEEE Micro,* vol. 28, pp. 17-25, 2008.

[18] 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* Toronto, Ontario, Canada: ACM, 2008, pp. 220-229.

[19] 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* Reno, Nevada: ACM, 2007, pp. 1-11.

[20] J. H. Ahn, J. Leverich, R. Schreiber, and N. P. Jouppi, "Multicore DIMM: an Energy Efficient Memory Module with Independently Controlled DRAMs," *Computer Architecture Letters,* vol. 8, pp. 5-8, 2009.

[21] 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* Anaheim, California, USA: ACM, 2005, pp. 559-564.
[22] D. Park, C. Nicopoulos, J. Kim, N. Vijaykrishnan, and C. R. Das, "Exploring Fault-Tolerant Network-on-Chip Architectures," in *Dependable Systems and Networks, International Conference on*. vol. 0 Los Alamitos, CA, USA: IEEE Computer Society, 2006, pp. 93-104.
[23] Intel, "EMBEDDED ULTRA-LOW POWER Intel486™ GX PROCESSOR," in *Datasheet*: Intel Corporation, 1997, p. 48.
[24] L.-S. Peh, N. Agarwal, N. Jha, and T. Krishna, "GARNET: A Detailed On-Chip Network Model inside a Full-System Simulator," in *International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2009.
[25] E. H. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," *International Journal of Man-Machine Studies*, vol. 7, pp. 1-13, 1975.
[26] P. S. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, G. Hallberg, J. Hogberg, F. Larsson, A. Moestedt, and B. Werner, "Simics: A full system simulation platform," *IEEE Computer*, vol. 35, pp. 50-58, 2002.
[27] S. J. E. Wilton and N. P. Jouppi, "An enhanced access and cycle time model for on-chip caches," HP Labs %L WRL Research Report 93/5, 1994.
[28] X. Chen, L.-S. Peh, and S. Malik, "Leakage Power Modeling and Optimization in Interconnection Networks," in *Proceedings of the International Symposium on Low Power and Electronics Design (ISLPED)* Seoul, Korea, 2003.
[29] H. Wang, L.-S. Peh, and S. Malik, "Orion: A Power-Performance Simulator for Interconnection Networks," in *Proceedings of MICRO 35* Istanbul, Turkey, 2002.
[30] H. Wang, L.-S. Peh, and S. Malik, "A Power Model for Routers: Modeling Alpha 21364 and InfiniBand Routers," *IEEE Micro*, vol. 23, pp. 26-35, 2003.
[31] H. Jin, M. Frumkin, and J. Yan, "The OpenMP Implementation of NAS Parallel Benchmarks and Its Performance," NASA Ames Research Center, 1999.
[32] R. V. d. Wijngaart, "NAS Parallel Benchmarks Version 2.4," NASA Advanced Supercomputing (NAS) Division, NASA Ames Research Center, Moffett Field, CA 2002.
[33] Intel, "Intel Concurrency Checker v2.1," Intel Corporation, 2008.
[34] J. S. R. Jang and N. Gulley, "Fuzzy Logic Toolbox for use with MATLAB," *The Math Works Inc*, 1995.
[35] L. A. Zadeh, *Fuzzy sets, fuzzy logic, and fuzzy systems: selected papers by Lotfi A. Zadeh* vol. 6: World Scientific, 1996.

Appendix A: Fuzzy Logic Introduction

Fuzzy logic uses a collection of membership functions defining input and output variables and specifies their corresponding relationship by IF-THEN based conditional statements called rules.

a) Membership Functions

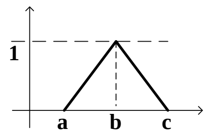
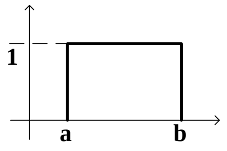
In contrast to a crisp-set, in which an element can belong to a set or not (i.e. having membership value of 1 or 0), a fuzzy logic membership function is a curve that defines the mapping of input values to a membership value between 0 and 1 [34]. This, in turn makes it convenient to represent lingusitic lables such as slow, fast, medium, heavy etc. Although there are many types of fuzzy membership functions such as pi, bell, trapezoidal etc, we only decribe in Table A1, rectangular or discrete, and triangular functions as they are used in this article.

b) Logical Operations

In fuzzy logic, logical operations such as AND, OR, NOT have corresponding equivalents such as min, max, and complement and are defined as following [35]

$$\mu_{A\cup B}(x) = max[\mu_A(x), \mu_B(x)]$$
$$\mu_{A\cap B}(x) = min[\mu_A(x), \mu_B(x)]$$
$$\mu_{\bar{A}}(x) = 1 - \mu_A(x)$$

TABLE A1. Triangular and Rectangular Membership Functions

Membership Function	**Definition**
	$\mu(x;a,b,c) = \begin{cases} 0 & for\ x < a \\ \frac{x-a}{b-a} & for\ a \le x < b \\ \frac{c-x}{c-b} & for\ b \le x \le c \\ 0 & for\ x > c \end{cases}$
	$\mu(x;a,b) = \begin{cases} 0 & for\ x < a\ or\ x > b \\ 1 & for\ a \le x \le b \end{cases}$

c) If-Then Rules

The fuzzy logic rules comprise of if-then statements operating on the fuzzy sets using fuzzy operators e.g.
If temperature is **high *then*** put the fan at **high** speed
If temperature is **very low** and humidity is **high *then*** put the heater at **high** temperature.

Generally, a single rule cannot specify the relationship among the inputs and outputs so two or more rules are required. The output of each rule is a fuzzy set, which are aggregated to find a single output fuzzy set. The resulting fuzzy set is then defuzzified to get a crisp number output which could be applied to the physical world.

d) Fuzzy Infernce Systems

The process of formulating the mapping from input to output using fuzzy logic is the fuction of fuzzy inference systems. This article has used Mamdani's fuzzy inference system [25] which was among the earliest implementations of fuzzy logic in a control system. For defuzzification centeroid of the curve method was used to find the crisp output.

Appendix B. Rules for Fuzzy Logic Inference Engine

TABLE B1. Core Level Rules for FLRE

L1 Miss Rate	**Energy Cons.**	**Thput.**	**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

Legend: L = Low, M = Medium, and H = High

TABLE B2. SoC Level Rules for FLRE

L2 Miss Rate	Energy Cons.	Thput.	L2 Cache Assoc.	Cache 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

Legend: L = Low, M = Medium, and H = High