Level-2 Shared Cache versus Level-2 Dedicated Cache for Homogeneous Multicore Embedded Systems

Abu ASADUZZAMAN, Manira RANI
Computer Science and Engineering Department, Florida Atlantic University
Boca Raton, Florida 33431, USA

and

Darryl KOIVISTO
Architecture Modeling Group, Mirabilis Design, Inc., 3000 Scott Blvd, Suite 201
Santa Clara, CA 95054, USA

ABSTRACT

Multicore brings tremendous amount of processing speed. On the contrary, it offers challenges for embedded systems as embedded systems suffer from limited resources. Various cache memory hierarchies are proposed to satisfy the requirements of different systems. Traditionally, level-1 cache memory is dedicated to each core. However, level-2 cache can be shared (like Intel Xenon) or dedicated (like AMD Athlon). Level-2 shared cache enables each core to dynamically use up to 100% of available CL2. Level-2 dedicated caches help each core to reduce latency when there is data / code sharing between the cores. However, it is not clear how level-2 shared and dedicated cache hierarchies should impact on the performance and total energy consumption for a set of applications. In this paper, we evaluate the impact of level-2 cache hierarchies (shared versus dedicated) on the performance and total energy consumption for homogeneous multicore embedded systems. We use VisualSim simulation tool to model the target architectures (one with shared CL2 and the other one with dedicated CL2). We use FFT, MI, and DFT workload (generated using Heptane package) to run the simulation program. Experimental results show that for negligible interconnection delay, level-2 shared cache hierarchy outperforms level-2 dedicated cache hierarchy.

Keywords—Multicore Embedded System, Level-2 Shared Cache, Level-2 Dedicated Cache, Performance Modeling, and Power-Aware Design

1. INTRODUCTION

Cache memory is first appeared in the IBM System/360 Model 85 computer in 1968 to improve performance by reducing the speed gap between the CPU and the main memory. Almost immediately after that all gigantic chip-vendors introduced cache to their processors [1][2]. Today, processors are having multiple processing cores and most processors have level-1 cache (CL1) and level-2 cache (CL2) [3][4][5][6]. The demand of multicore embedded systems is increasing and billions of transistors are possible in a single chip. As a result, the trend of using multicore systems is expected to increase for the next few decades. In a multicore processor, two or more independent cores are combined into a die. Usually, each core has its own CL1 – CL1 may be split into instruction (I1) and data (D1) caches. Most processors have unified CL2 – CL2 may be shared by the cores or distributed and dedicated to each core [7][8][9]. Intel's Advanced Smart Cache works by sharing CL2 among the cores. It is optimized for multicore processors to improve performance. AMD's multicore processors have dedicated CL2. With cache memory, the system consumes more energy and cache makes the system more unpredictable [10][11][12][13][14][15][16]. It can be argued that neither of these two level-2 cache hierarchies is better for all workloads. The shared hierarchy outperforming the dedicated one on workloads with high level of data or code sharing as it simplifies cache coherence and eliminates coherency traffic between multiple dedicated caches at the same level. The opposite may be true for workloads composed of independent threads with little sharing [18].

Multicore is a new direction for modern computing. In multicore embedded systems, performance and energy consumption are significantly affected by the cache memory hierarchy and applications. More cache miss means decrease in performance and increase in energy consumption. Various techniques are being used to reduce cache miss rate. Recently published articles show that multicore design improves the performance/energy ratio [4][5][7][13][15] by executing more number of instructions per cycle at a lower frequency. However, it is unknown how the level-2 cache hierarchies (shared and dedicated) impact on the performance and energy consumption. In this work, we focus on the impact of level-2 cache memory hierarchies (shared versus dedicated) on performance and total energy consumption.

The outline of his paper is as follows. Contemporary level-2 cache memory hierarchies used in popular multicore processors are described in Section 2. Some articles related to cache modeling in multicore systems are presented in Section 3. Section 4 explains experimental details used in this work. In Section 5, the simulation results are discussed. Finally, we conclude our work in Section 6.

2. LEVEL-2 CACHE IN MULTICORE PROCESSORS

Almost every PC built since the first cache memory appeared in IBM System/360 computer in 1968 has some sort of cache memory. In early 1990s, Intel 486DX4 and Pentium included off-chip level-2 cache. Contemporary multicore processors usually have dedicated level-1 cache and shared (example: Intel Xenon) or dedicated (example: AMD Athlon) level-2 cache (see Figure 1). Usually, level-2 cache resides on the motherboard. However, level-2 cache is also seen on the microprocessor itself. Some manufacturers connect the microprocessor and the level-2 cache with a backside bus to improve the performance. Recently, most manufacturers are adopting multicore processor or chip-level multiprocessor (CMP) for their future embedded systems to acquire additional processing speed and to save (battery) energy.

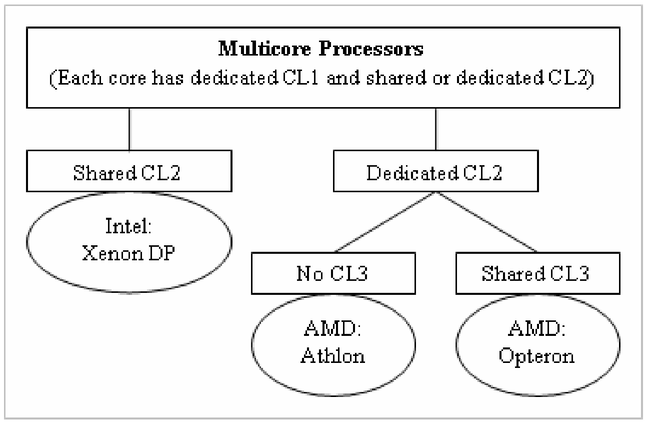

Figure 1. Various contemporary multicore CPUs

Intel dual-core has a shared CL2 (for example, dual-core Xenon has 64 KB I1, 64 KB D1, and one 4 MB CL2) while AMD dual-core employs distributed and dedicated CL2s (for example, dual-core Athlon Classic has 64 KB I1, 64 KB D1, and two 512 KB CL2s) [8]. Shared CL2 enables each core to dynamically use up to 100% of available CL2. Dedicated CL2s help each core to reduce latency when there is data/code sharing between the cores.

As shown in Figure 2, Intel quad-core (example, Xenon DP: 128 KB I1, 128 KB D1, one 8 MB CL2) has one shared CL2 [7]. In this work, we simulate a Xenon-like quad core with shared CL2.

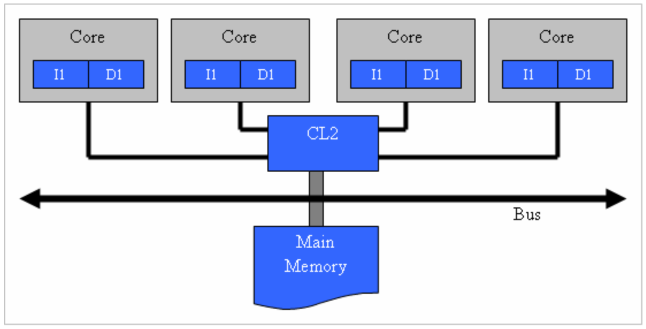

Figure 2. Quad-core architectures (Intel – Kentsfield XE)

However, AMD quad-core (example, Opteron: 256 KB I1, 256 KB D1, four 2 MB CL2s) has distributed and dedicated CL2s and a shared CL3 [5] as shown in Figure 3. CL3 of AMD Opteron processors may be 2 MB (Santa Rosa) or 4 MB (Deerhound). In this work, we simulate an Opteron-like quad core with dedicated CL2 (excluding the CL3).

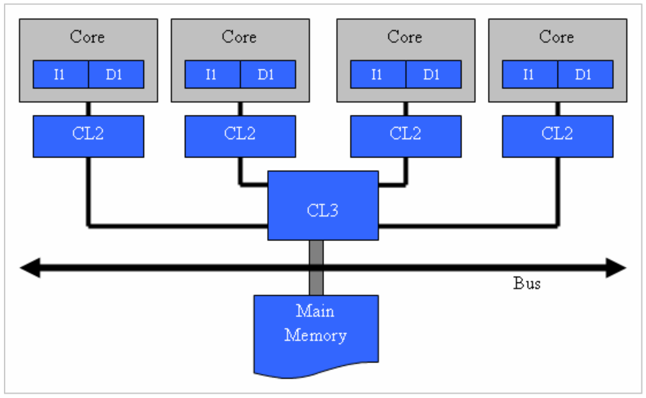

Figure 3. Quad-core architectures (AMD - Opteron)

IBM, with a joint project with Sony and Toshiba, has introduced the Cell multicore architecture, in a joint venture with Sony and Toshiba, to boost up the processing speed demanded by the 3D electronic games. The Cell chip may have a number of different configurations. The basic configuration is a multicore chip composed of one threaded Primary Processing Element (PPE) and multiple Synergistic Processing Elements (SPEs) [17][19][20][21]. In a typical Cell processor, CL1 is dedicated to the PPE and CL2 may be shared by the PPE and SPEs. A SPE is called a “Cell”. Each cell may have 256 KB SRAM and a 4x128 bit Arithmetic Logical Unit (ALU) which does the math in a processor and 128 of 128-bit registers. The Element Interconnect Bus (EIB) is the communication bus internal to the Cell processor which connects the various on-chip system elements: PPE processor, memory controller (MIC), SPE coprocessors, and off-chip I/O interfaces.

3. RELATED WORK: CACHE MODELING IN MULTICORE SYSTEMS

Various cache memory hierarchies have been proposed to improve the performance and to decrease the total power consumption of multicore embedded systems. Some selected work relating to cache modeling in multicore systems are discussed in this section.

In [18], the performance of shared level-2 and distributed level-2 cache organizations has been studied using various kind of workload. Experimental results show that the shared level-2 cache organization outperforms the dedicated one on stressful workloads which increase the interconnect latencies between dedicated caches; while the dedicated private level-2 cache organization is superior on lighter workloads with smaller interconnect latencies. This work does not perform energy consumption analysis which is important for embedded systems.

In [22], a comparative performance and energy analysis is provided for cache-coherence support schemes in multi-processor system-on-a-chip (MPSOC). Experimental results show that hardware based solution needs more power when traffic grows. This work does not provide any good solutions to improve performance and to decrease the power consumption in MPSOC.

In [23], a hardware/software methodology is proposed to make the caches coherent in heterogeneous multiprocessor platforms with shared memory. This experiment shoes that the performance improvement can be achieved with low miss penalty at the expense of adding simple hardware, compared to a pure software solution. Speedup can be improved even further as the miss penalty increases. This approach provides embedded system programmers a transparent view of shared data, removing the burden of software synchronization. This methodology is neither applicable for dedicated CL2 architecture nor suitable for analyzing energy consumption.

In [24][25], two approaches are presented to cope with the predictability problem due to cache in real-time systems. According to these approaches, cache contents are statically locked so as to make memory access time and cache-related preemption delay predictable. However, more study is needed to see the applicability of static cache locking techniques on various level-2 cache schemes and the impact of these approaches on performance and energy consumption for larger real benchmarks.

In [26], we model and simulate a multicore system using VisualSim where level-1 cache is dedicated to each core and level-2 cache is shared. Experimental results show that the execution time predictability of applications running on multicore systems can be improved with negligible impact on the ratio of performance to energy consumption by using cache optimization technique.

4. EXPERIMENTAL SETUP

Assumptions

The following assumptions are made to model the target architectures (with shared CL2 and dedicated CL2) and to run the simulation program.

1) For simplicity, both multicore systems (one with shared CL2 and the other one with dedicated CL2) are considered to be homogenous.
2) In both shared and dedicated CL2 architectures, CL1 size (I1 size + D1 size) is the same.
3) Interconnection delay is negligible.
4) Total CL2 sizes in both shared and dedicated CL2 architectures are the same. For shared CL2 architecture, CL2 size = k x Number of cores x CL1 size; where k = 1, 2, 4, and 8. For dedicated CL2 architecture, each CL2 size = k x CL1 size; where k = 1, 2, 4, and 8.

Simulated Architectures

In this work, our goal is to investigate the impact of level-2 cache memory hierarchies (shared and dedicated) on performance and energy consumption. Based on contemporary multicore processor design trends from Intel, AMD, and IBM we simulate two similar multicore embedded systems, one with one shared CL2 and the other one with four dedicated CL2s.

As shown in Figure 4, we use Intel-like Advanced Smart Cache where CL1s are dedicated to the cores and one CL2 is shared among the cores so that data is stored in one place that each core can access. Here CL2 enables each core to dynamically use up to 100 percent of available CL2.

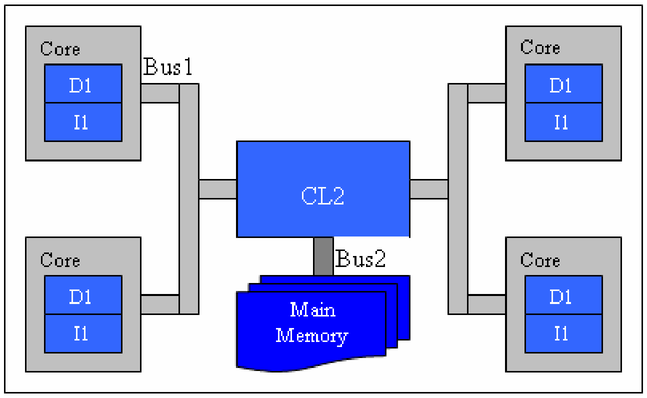

Figure 4. Simulated quad-core processors – shared CL2

Also shown in Figure 5, we use AMD Opteron-like processor where both a CL1 and a CL2 are dedicated to each core (unlike AMD Opteron, we exclude CL3 in our architecture). In this architecture, CL2 helps each core to reduce latency when there is little data and/or code sharing between the threads running on each core.

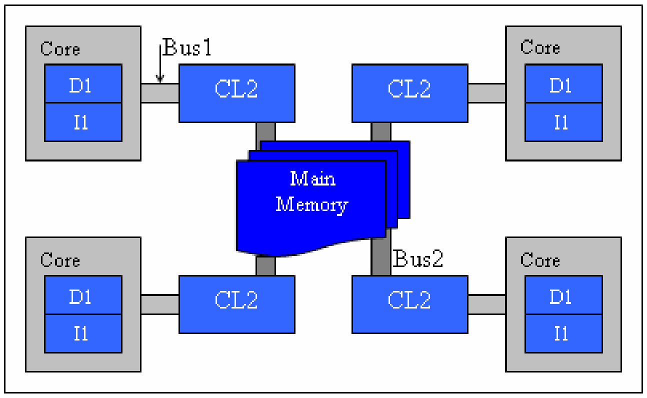

Figure 5. Simulated quad-core processors – dedicated CL2

Workload

In this work, we use workloads that are generated from Fast Fourier Transform (FFT), Matrix Inversion (MI), and Discrete Fourier Transform (DFT) applications. Useful information about FFT, MI, and DFT is shown in Table 1.

Table 1. Information about FFT, MI, and DFT applications

Application Name	Code Size (B)	Number of Instructions	Proc. Cycles Needed
FFT	2335	365184	16224629
MI	1468	227518	9801353
DFT	1158	171307	7996174

We use Heptane [27] and VisualSim [28] simulation tools. Heptane takes C code (application) as the input and generates tree-graph showing the blocks that cause cache misses. After post-processing the tree-graph a Miss Table showing the number of misses caused by the blocks is generated. Using VisualSim, a simulation platform is developed to model and simulate multicore embedded systems. Miss Table is used to run the VisualSim simulation program.

5. RESULTS AND DISCUSSION

In this work, we explore the impact of the level-2 cache hierarchies (shared and dedicated) on the performance and energy consumption of homogeneous multicore embedded systems. We use mean delay to express the performance (decrease in average delay means improve in performance). We keep level-1 cache size fixed at I1 = 2KB and D1 = 2KB. Throughout the experiment we use random cache replacement policy and write-back write miss policy. Assuming negligible interconnection delay and using FFT, MI, and DFT workloads, we obtain results for embedded systems with 4 cores. We define delay as the number of processor cycles between the start of execution of a task and the end. Mean delay is the average delay of all the tasks used in VisualSim simulation. In order to weight the impact of various CL2 sizes, we keep the total CL2 size identical for both shared and dedicated CL2 architectures.

The average delay per core Vs total CL2 size for shared CL2 hierarchy is shown in Figure 6. Experimental results show that the mean delay per core decreases with the increase of CL2 size for all three applications. The decrease is significant for smaller CL2 size (16 to 32 KB). Results also show that the mean delay for FFT workload is higher than that of MI or DFT.

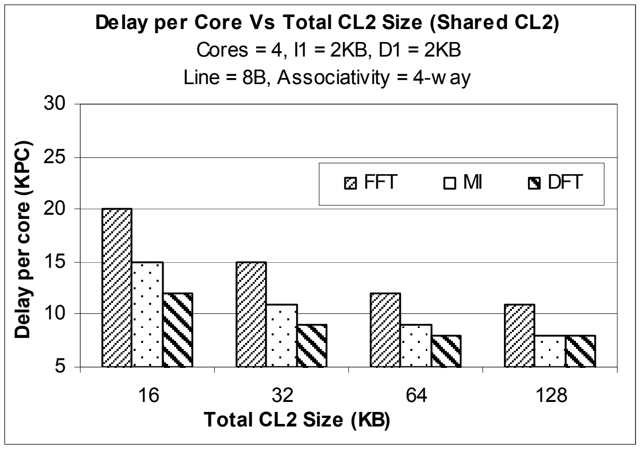

Figure 6. Mean delay per core for shared CL2 architecture Vs CL2 size

For dedicated CL2 architecture, total CL2 size = number of cores x size of each CL2 (and each CL2 size = k x CL1 size; where k = 1, 2, 4, and 8). In Figure 7, the average delay per core Vs total CL2 size for dedicated CL2 hierarchy is shown. For all three applications, the decrease in mean delay is significant for smaller CL2 size (16 to 32 KB). It is also noticed that the mean delay for FFT workload is higher than that of MI or DFT.

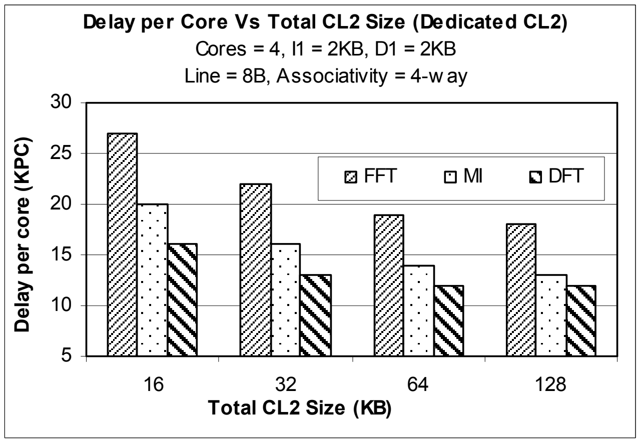

Figure 7. Mean delay per core for dedicated CL2 architecture Vs CL2 size

Based on the results shown in Figures 6 and 7, for both shared and dedicated CL2 hierarchies the mean delay per core decreases with the increase of CL2 size for all three applications. It is also observed that for all three applications and all CL2 size considered the FFT workload takes more memory access time when compared with that of MI and DFT workload. Next, we compare the mean delay per core and total energy consumption Vs total CL2 size for shared and dedicated CL2 hierarchies using FFT workload (MI and DFT are expected to produce similar impact on delay and total energy consumption).

Experimental results show that mean delay per core decreases with the increase in CL2 size for both shared and dedicated CL2 hierarchies [see Figure 8]. However, the delay for shared CL2 hierarchy is significantly smaller than that of dedicated CL2 hierarchy.

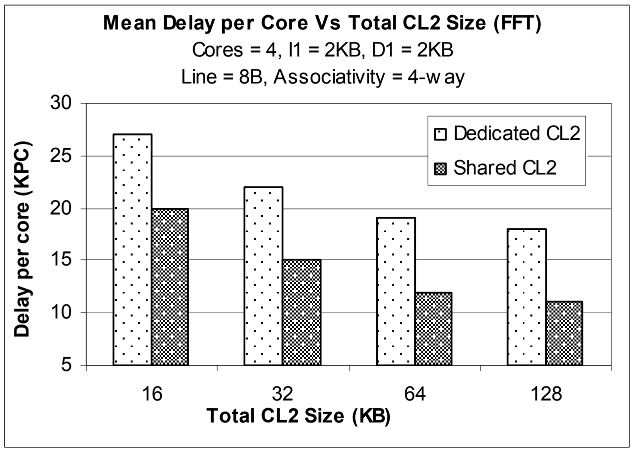

Figure 8. Mean delay per core Vs CL2 size

Similarly, the total energy consumption decreases with the increase of CL2 size for both shared and dedicated CL2 hierarchies (see Figure 9). But, the decrease in energy consumption for shared CL2 hierarchy is more significant than that of dedicated CL2 hierarchy.

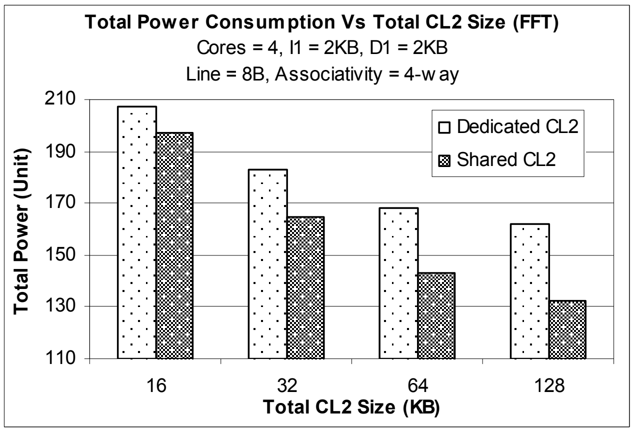

Figure 9. Total energy consumption Vs CL2 size

In summary, when we change the total CL2 size from 16 KB to 128 KB, the shared CL2 hierarchy reduces delay by 45% and total energy consumption by 33% but the dedicated CL2 hierarchy reduces delay by 33% and total energy consumption by 22%.

6. CONCLUSIONS

From recent studies we know that multicore design improves the ratio of performance to energy consumption. Typically in a multicore architecture, level-1 cache is dedicated to each core. However, level-2 cache can be shared or dedicated. Level-2 shared cache enables each core to dynamically use up to 100% of available CL2. Level-2 dedicated caches help each core to reduce latency when there is data and/or code sharing between the cores. It is not very clear how level-2 shared and dedicated cache hierarchies impact on the performance and total energy consumption for a target set of applications. In this paper, we explore the impact of level-2 cache hierarchies (shared versus dedicated) on the performance and total energy consumption for homogeneous multicore embedded systems. Using VisualSim, we model and simulate two 4-core architectures (one with shared CL2 and the other one with dedicated CL2). We use FFT, MI, and DFT workloads (generated by Heptane package) to run the simulation programs. For simplicity, we assume that the interconnection delay is negligible. Experimental results show that for any size of CL2, the mean delay and the total energy consumption due to the shared CL2 architecture is smaller when compared with those of dedicated CL2 architecture. This is because the workload used is lighter from the level-2 cache's perspective. The impact may be different if the interconnection delay is significant and heavier workloads (like MPEG-4 application) are used.

We plan to repeat our experiments with significant interconnection delay and running real-time applications including MPEG-4 and H.264/AVC in the future.

7. REFERENCES

[1] **Cache - Smart Computing Encyclopedia**, 2008. http://www.smartcomputing.com/editorial/dictionary/detail.asp?guid= &searchtype=&DicID=16600&RefType=Encyclopedia

[2] L. Schoeb, G. Darnell, "Large Processor L2 Cache Sizes in Dell PowerEdge Servers", **Dell**, 1999.

[3] R. Cook, J. Linn, C. Linn, T. Walker, "Cache memories: A Tutorial and Survey of Current Research Directions", **ACM**, 1982, pp. 99–110.

[4] R.M. Ramanathan, "Intel Multi-Core Processors: Making the Move to Quad-Core and Beyond", **White Paper**, 2006.

[5] V. Romanchenko, "Quad-Core Opteron: architecture and roadmaps", **Digital-Daily.com**, 2006.

[6] G. Torres, "Inside Pentium 4 Architecture", 2005. http://www.hardwaresecrets.com/printpage/235/1

[7] V. Romanchenko, "Evaluation of the multi-core processor architecture Intel core: Conroe, Kentsfield…", **Digital-Daily.com**, 2006.

[8] "Multi-core (computing)", **Wikipedia**, 2008. http://en.wikipedia.org/wiki/Xeon; http://en.wikipedia.org/wiki/Athlon

[9] D.K. Every, "IBM's Cell Processor: The next generation of computing?", **Shareware Press**, 2005. http://www.mymac.com/fileupload/CellProcessor.pdf

[10] D. Lenoski, J. Laudon, M.S. Lam, et al., "The Stanford Dash Multiprocessor", **IEEE**, 1992.

[11] E. Tamura, F. Rodriguez, J.V. Busquets-Mataix, A.M. Campoy, "High Performance Memory Architectures with Dynamic Locking Cache for real-Time Systems", **Proceedings of the 16th Euromicro Conference on Real-Time Systems**, Italy, 2004.

[12] L. Thiele, R. Wilhelm, "Design for Timing Predictability", **Real-Time Systems**, 2004, Vol. 28, Issue 2-3, pp. 157-177.

[13] "Improving cache performance", **UMBC**, 2008. ODI=http://www.csee.umbc.edu/help/architecture/611-5b.ps

[14] X. Vera, B. Lisper, "Data Cache Locking for Higher Program Predictability", **SIGMETRICS'03**, CA, USA, 2003.

[15] R. Wilhelm, J. Engblom, S. Thesing, D. Whalley, "The Determination of Worst-Case Execution Times", **ARTIST**, 2003.

[16] R. Wilhelm, L. Thiele, "Timing Predictability — a Must for Avionics Systems", **ARTIST**, 2006.

[17] A.L. Shimpi, Understanding the Cell Microprocessor", **CPU & Chipset**, 2005.

[18] F. Sibai, "On the Performance Benefits of Sharing and Privatizing Second and Third Level Cache Memories in Homogeneous Multi-Core Architectures", **Microprocessors and Microsystems**, Elsevier, 2008.

[19] N. Blachford, "Cell Architecture Explained Ver2.", 2006. ODI= http://www.blachford.info/computer/Cell/Cell0_v2.html

[20] J. Stokes, "Introducing the IBM/Sony/Toshiba Cell Processor – Part II: The Cell Architecture", 2005. DOI=http://arstechnica.com/articles/paedia/cpu/cell-2.ars

[21] A. Vance, "Cell processor goes commando", **Mountain View**, 2006. ODI= http://www.theregister.co.uk/2006/01/22/cell_mecury_army/

[22] M. Loghi, M. Poncino, L. Benini, "Cache coherence tradeoffs in sharedmemory MPSOCs", **ACM Transaction on Embedded Computing Systems**. Vol. 5, No. 2, 2006, pp. 383-407.

[23] T. Suh, D. Kim, H.-H.S. Lee, "Cache Coherence Support for Non-Shared Bus Architecture on Heterogeneous MPSOCs", **Proceedings of the 42nd annual conference on Design automation (CA-2005)**, 2005, pp. 553-558.

[24] I. Puaut, D. Decotigny, "Low-Complexity Algorithms for Static Cache Locking in Multitasking Hard Real-Time Systems", **IEEE**, 2002.

[25] I. Puaut, "Cache Analysis Vs Static Cache Locking for Schedulability Analysis in Multitasking Real-Time Systems", 2006. http://citeseer.ist.psu.edu/534615.html

[26] A. Asaduzzaman, N. Limbachiya, I. Mahgoub, F. Sibai, "Evaluation of I-Cache Locking Technique for Real-Time Embedded Systems", **IEEE-IIT'07**, UAE, 2007.

[27] Heptane, **Heptane – A tree-based WCET analysis tool**, 2008. ODI=http://ralyx.inria.fr/2004/Raweb/aces/uid43.html

[28] VisualSim (VisualSim Architecture), **Mirabilis Design, Inc.**, 2008. ODI=http://www.mirabilisdesign.com/