

## Cache Architecture Limitations in Multicore Processors

Wael Mohamed, Maher Mansour Abd El-Aziz  
Faculty of Engineering,  
Helwan University, Cairo, Egypt.

---

### Abstract

Today supercomputers play a big role in our life for complex applications. Multicore processors represent the most significant part in supercomputers. A multicore processor consists of several cores which can execute different tasks independently. Due to the budget and chip area limit, the last level cache is usually shared among cores. Running tasks on different cores access the shared cache intensively and concurrently. This may lead to cache miss rate and significant performance degradation. In this paper, we study the effect of cache architectures on the performance of multicore processors for multi-threading applications and their limitations on increasing the number of processor cores. The study focus on two design issues: cache architecture and configuration parameters. It also based on a cache simulator that models the functionality of a multicore cache hierarchy with arbitrary levels and various organizations. Our evaluations also help in determining the best cache architecture and configuration for a given number of cores to obtain the good performance.

---

**Keywords:** cache memory, multicore processing, performance analysis, simulation architecture

---

### INTRODUCTION

Since a large main memory cannot provide instructions and data as fast as the clock speed of the processors. This increases the access time of instructions and data from the main memory to processors. Therefore most of computer architecture research focuses on innovative ideas to reduce the access time between processors and main memories.

In order to minimize this access time a temporary storage space called cache memory is used to fetch instructions and data from a main memory which is accessed frequently by processors. Due to improving in chip manufacturing technology, more transistors can be placed on one chip. This enables hardware designers to place more processors and a hierarchy of bigger caches on one chip while sharing a common external main memory. The simplest way for processors to access the shared external memory is using multi-level cache memory hierarchy with one of the levels shared by all the processors. Sharing level 1 cache (L1) is difficult because its response time must be fast enough to keep up with the processors clock speed. Sharing level 2 caches (L2) among processors is more desirable because it enables processors to communicate with each other in a fairly short amount of time but without slowing the processors clock speed.

Since 1990s until now, researchers have been trying to determine cache architecture delivers better overall performance with higher hit rate and lower miss rate for multiprocessors [Q. Yang et al, J.S Vitter et al, Lioupis et al, J. Bertoni et al, F. N. Sibai et al and Jin Young et al, 1990,1994,1997, 1992,2008,2009], [B. Nitzberg et al and G. Dewan et al, 1991], [Anderson et al, 1993, 1995], [J. L. Baer et al and M. K. Vernon et al, 1989]. The cache architecture becomes very

complex as the number of processors increases. Cache architectures such as single shared L2, L3 and L4 bus, hierarchical bus, and ring-shaped architecture are widely known and studied independently.

### RELATED WORKS

Research work addresses on the cache design of this novel architecture is increased with the trend of microprocessor towards multicore.

Chun Liu et al, 2004 proposed a new architecture called Shared Processor-Based Split L2. With this design, the shared L2 cache is comprised of multiple smaller units which can be selectively allocated to the processor cores. This architecture has the advantage of configuring L2 caches based on application workload characteristics. The architecture is evaluated on the SIMICS simulation environment. Experimentally, the private organization improves the IPC by an 11.52%

D. Benitez et al, 2006 designed a Field-Programmable Cache Array that can be dynamically reconfigures. A runtime algorithm was also developed to manage this specific architecture. This algorithm can compute the best cache configuration for each program phase, where program phase is detected using data working-set signatures.

M. Modarressi et al, 2006 designed reconfigurable cache architecture for object-oriented application-specific instruction set processors (ASIP). With this design, cache architecture is virtually divided into a number of partitions and the partition sizes can be dynamically changed depending on the run-time behavior of the application. The goal of cache partitioning is to both provide the concurrent memory

access for multiple functional units and to reduce the number of tag comparisons per cache access. In addition, a simple and energy-efficient cache consistency mechanism was also developed. The impact of the proposed cache architecture on the cache energy consumption has been evaluated. Experimentally, a 39%.

Xuemei Zhao et al, 2007 proposed new cache architecture SPS2 with split private and shared L2 caches. He also proposed a corresponding SPS2 cache coherence protocol which is described by means of new state transition graphs in which each node has two states to indicate the states of private L1 or private L2, and shared L2 respectively. Using the state transition graphs, the functional correctness of coherence protocol is proven. The use of formal design verification methods helps identify coherence problems in the early stage, and provide assurance of the correctness of the protocol before commencing on the hardware development.

In summary, researchers are investigating various cache designs for enhancing the performance of this component towards a high performance of the overall system. Our goal is to evaluate different traditional designs available on current microprocessors and thereby find potential combination or improvement for the emerging multicore systems. The self-developed cache simulator allows us to comprehensively understand the cache access behavior of applications with various cache designs.

**SIMULATION FRAMEWORK**

The used simulation framework contains a multicore Simulation tool and a subset of benchmark programs to evaluate the architectural enhancement of multicore by workload all threads of multicore using one benchmark program executed [Ubal R et al and Woo et al, 2007, 1995].

**A- Multicore Simulation Description**

Multi2Sim is a simulation framework for heterogeneous computing, including models for super-scalar, multithreaded, multicore, and graphics processors. Multi2sim uses three different models, embodies in three different modules as shown in Fig. 1. Also we will use two terms: context and threads, in which context refers to a software entity defined by status of virtual memory image and a logical register \_le while thread refers to a processor hardware entity represented as Physical register file, a set of physical memory pages, a set of pipeline queues, etc.

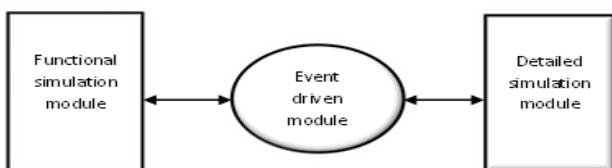


Figure 1. Simulator module

**A functional simulation:** It behaves as a library and provides an interface to the rest of the simulator. It owns the functions to create/destroy software contexts, perform program loading, enumerate existing contexts, execute machine instructions and handle speculative execution. It also has checked pointing capability of implemented memory module and register file. This capability leads to save both register file and memory status and reload them when a wrong execution path starts.

**Detailed Simulation:** It uses as a function engine to perform a timing-first simulator simulation. This means that in each cycle, a sequence of calls to kernel updates the states of existing contexts. It also analyses the nature of the executed machine instructions and calculates the operation latencies incurred by hardware structure.

**Event Driven Simulation:** It implements the function calls between functional simulation module and detailed simulation module for increasing the simulator modularity. It is also an interface to calculate the latency.

**B- Benchmark Suite Description**

SPLASH-2 is a suite consists of eleven parallel benchmarks provide configuration files to specify the input data size. All the benchmarks perform computations, synchronizations, communication, stressing processor cores, memory hierarchy, and interconnection networks for evaluating the baseline and proposed multicore architectures. In addition, they provide arguments to specify the number of contexts created at runtime for the evaluation of systems with different number of cores. SPLASH-2 is a suite of parallel scientific work-loads. Each benchmark executes the same number of cores.

**PROPOSED METHODOLOGY**

Multi2sim has been developed integrating some significant characteristics of popular simulators such as separate functional and timing simulation, Simultaneous multithreading (SMT) [Marco Paolieri et al, 2013] and multiprocessor support and cache coherence. Multi2sim is an application tool intended to simulate x86 binary executable files.

Table 1 and table 2 show the cache memory architectures and configurations.

**TABLE I. CACHE MEMORY ARCHITECTURES**

Item	Structure
Architecture 1 (A1)	Shared-L2 cache memory
Architecture 2 (A2)	Shared-L2-H cache memory
Architecture 3 (A3)	Shared-L3 cache memory
Architecture 4 (A4)	Shared-L3-H cache memory
Architecture 5 (A4)	Shared-L4 cache memory
Architecture 6 (A6)	Shared-L4-H cache memory

**TABLE 2. CACHE MEMORY CONFIGURATIONS**

Item	Configuration
Configuration 1 (C1)	L1(32KB+32KB),L2 2MB,8-Way,64B
Configuration2 (C2)	L1(64KB+64KB),L2 2MB,8-Way,64B
Configuration3 (C3)	L1(32KB+32KB),L2 4MB,8-Way,64B
Configuration4 (C4)	L1(32KB+32KB),L2 4MB,16-Way,64B
Configuration5 (C5)	L1(64KB+64KB),L2 4MB,8-Way,64B
Configuration6 (C6)	L1(64KB+64KB),L2 4MB,16-Way,64B
Cache memory L3 : 8MB	
Cache memory L4 : 16 MB	

Table 3 shows the detail information about the machine used in the simulation. All configurations are running in parallel and the result data is gathered and analyzed.

**TABLE 3.MACHINE CONFIGURATION**

Item	Specification
Cores	Intel 2.13GHz(Dual core)
L1 cache	128 KB
L2 cache	512 KB
L3 cache	3.0 MB

**SIMULATION RESULTS**

**A. CPU performance: Instruction per Cycle**

One of the most common and widely used performance metrics is IPC (Instruction per Cycle) and it is measured and analyzed in all simulation cases because it is an indicator speed for the processor [Wu, 2011].

The following figures show the performance (IPC) of multicore architectures with different cache configurations for Barnes benchmark:

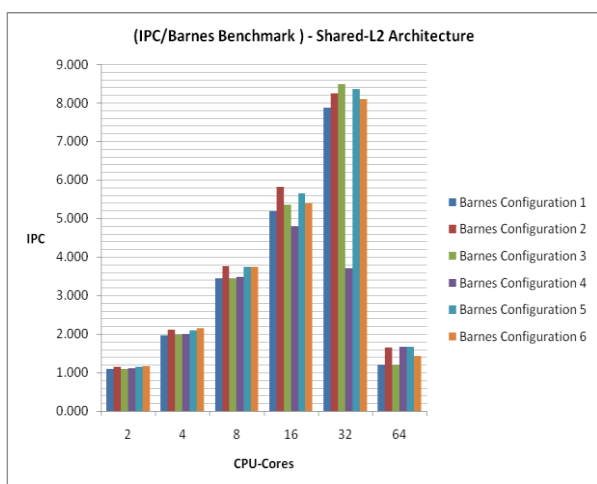


Figure 2. (IPC/Barnes Benchmark) shared-L2-H Architecture.

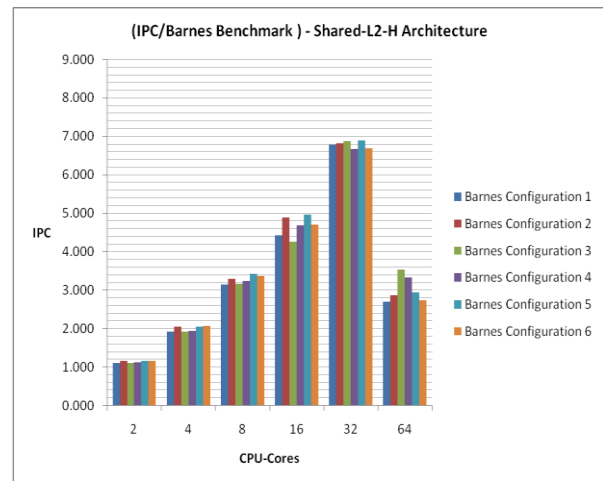


Figure3. (IPC/Barnes Benchmark) shared-L2-H Architecture

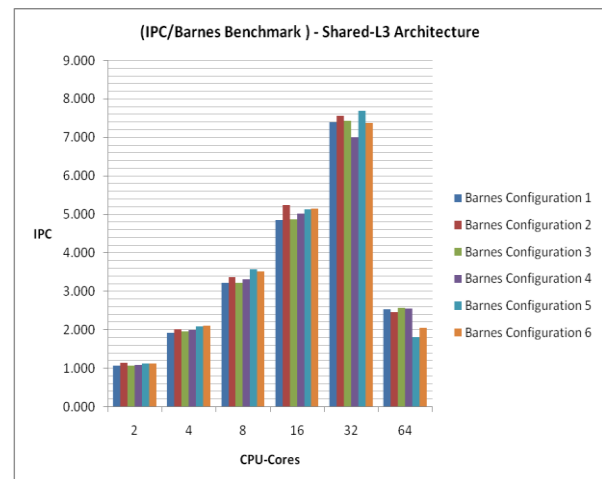


FIGURE 4. (IPC/BARNES BENCHMARK) SHARED-L3 ARCHITECTURE

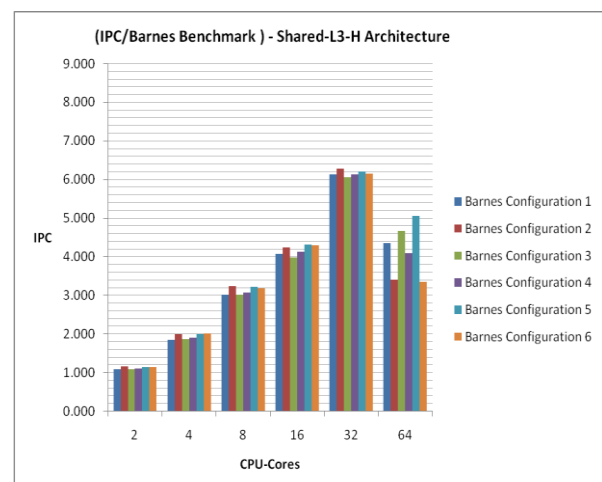


Figure 5. (IPC/Barnes Benchmark) cache memory architectures versus cache configurations (for 32 cores).

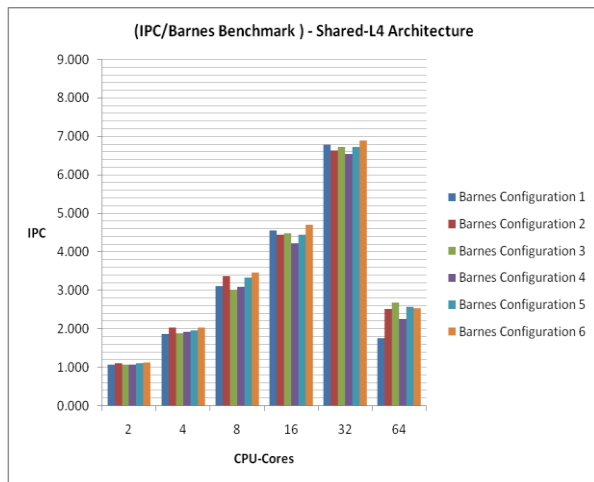


Figure 6. (IPC/Barnes Benchmark) shared-L4 Architecture.

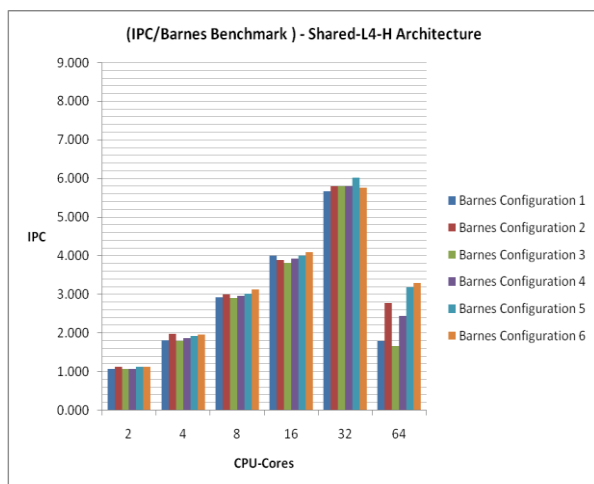


Figure 7. (IPC/Barnes Benchmark) shared-L4-H Architecture.

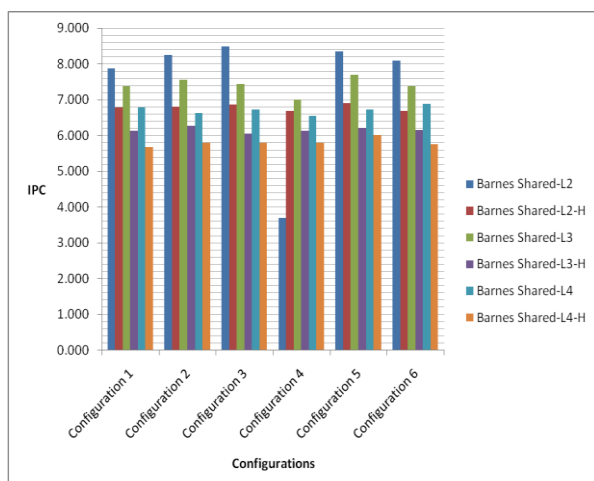


Figure 8. (IPC/Barnes Benchmark) cache memory Architectures versus cache configurations (for 32 cores).

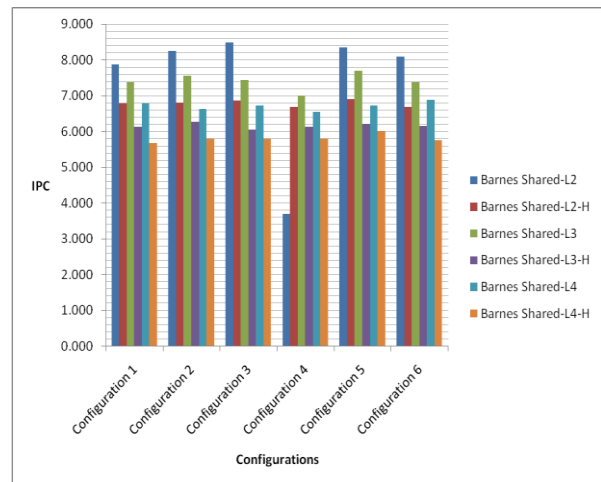


Figure 9. (IPC/Barnes Benchmark) cache memory configurations versus cache memory architectures (for 32 cores).

The first study addresses the organization of the cache level closest to the main memory. We executed the applications on systems with 2, 4,8,16, 32 and 64 cores separately and with both private and shared L2, L3 and L4 organization where in the private case each processor core has a local second and third level cache and in the shared case a combined L2, L3 and L4 is available for all cores. We examine also how different cache configurations influence the performance. For this, we simulated all applications again, but with various associativity and cache size.

As shown in Fig.2 to Fig.7, we note that the average IPC is increased as the number of processor cores is increased except for 64 cores the average IPC is decreased. Also, duplicating the size of L1 or L2 cache does not significantly increase the IPC. As shown in Fig.8 and Fig.9, the IPC gain is achieved with shared L2 for all configurations except configuration 4 preferred to shared L3.

### B. CPU performance: Cache Hit Ratio

In computing, a cache is a component that transparently stores data so that future requests for that data can be served faster. The data that is stored within a cache might be values that have been computed earlier or duplicates of original values that are stored elsewhere. If requested data is contained in the cache (cache hit), this request can be served by simply reading the cache, which is comparatively faster. Otherwise (cache misses), the data has to be recomputed or fetched from its original storage location, which is comparatively slower. Hence, the greater the number of requests that can be served from the cache, the faster the overall system performance becomes.

The following figures show the performance (Cache Hit Ratio) of multicore architectures with different cache configurations for Barnes benchmark:

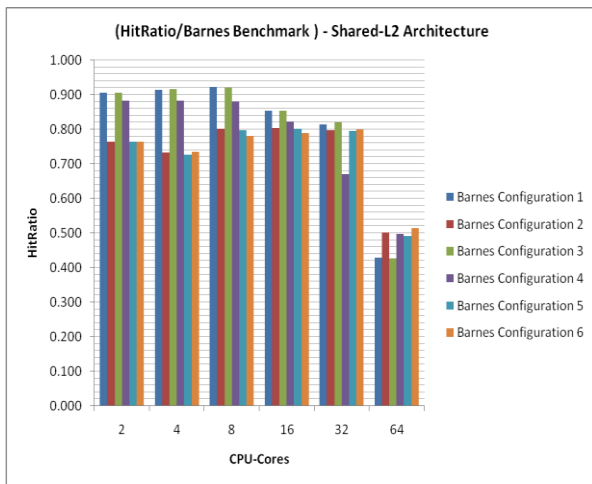


Figure 10. (HitRatio/Barnes Benchmark) shared-L2 Architecture

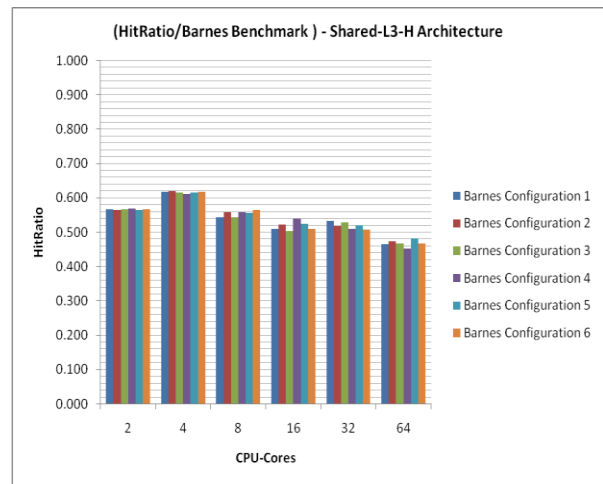


Figure 13. (HitRatio/Barnes Benchmark) shared-L3-H Architecture

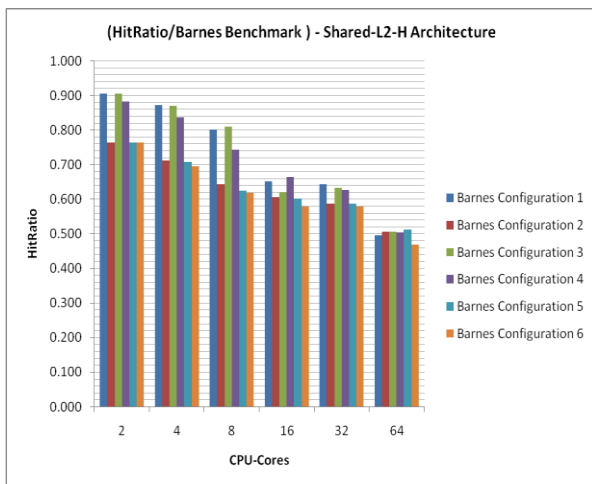


Figure 11. (HitRatio/Barnes Benchmark) shared-L2-H Architecture

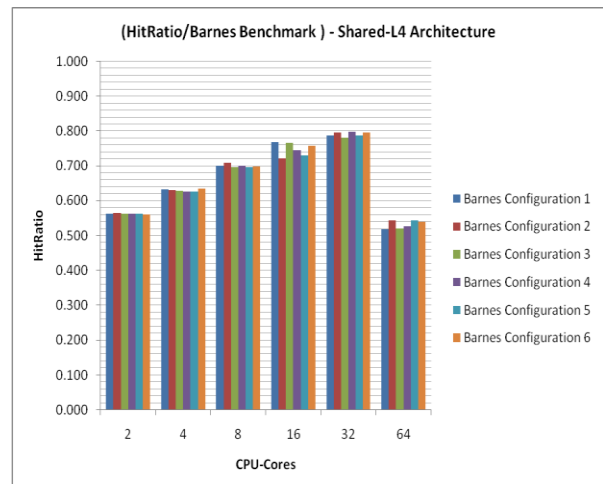


Figure 14. (HitRatio/Barnes Benchmark) shared-L4 Architecture

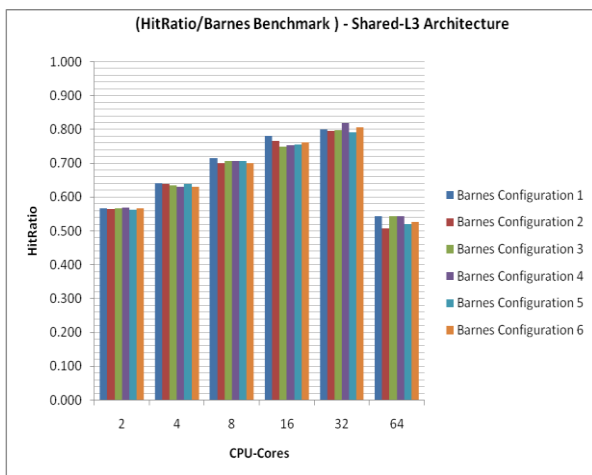


Figure 12. (HitRatio/Barnes Benchmark) shared-L3 Architecture

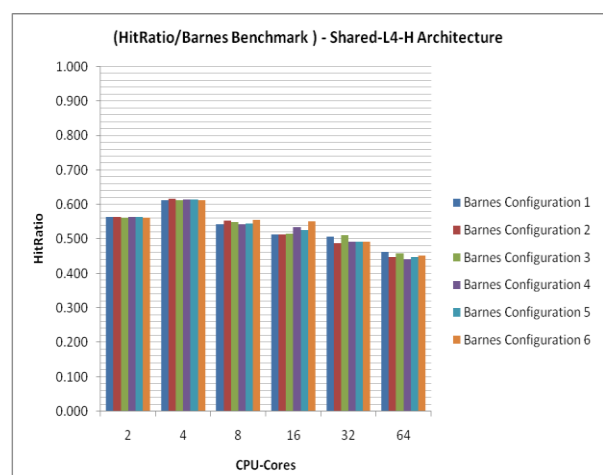


Figure 15. (HitRatio/Barnes Benchmark) shared-L4-H Architecture

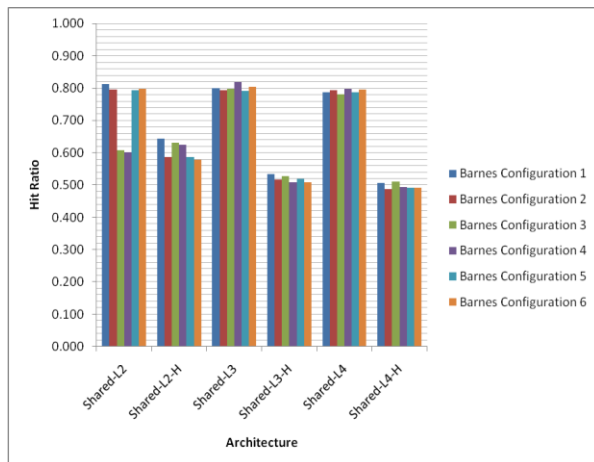


Figure 16. (HitRatio/Barnes Benchmark) cache memory architectures versus cache configurations (for 32 cores).

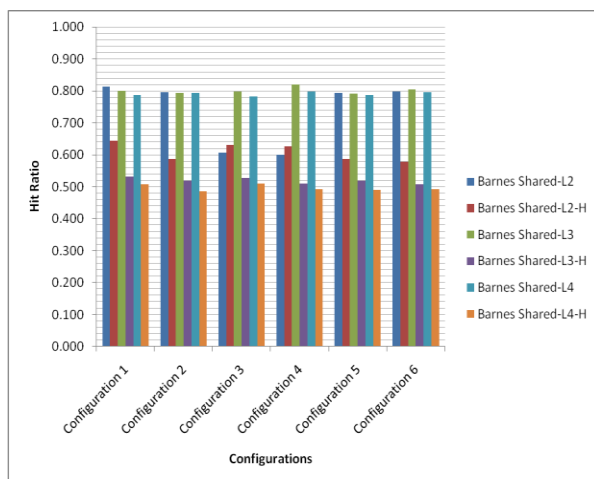


Figure 17. (HitRatio/Barnes Benchmark) cache memory configurations versus cache memory architectures (for 32 cores).

As shown in Fig.10 to Fig.15, we note that the Shared Cache Hit Ratio does not significantly decrease as the number of processor cores is increased. Also, duplicating the size of L1 or L2 cache does not significantly decrease the Shared Cache Hit Ratio. As shown in Fig.16 and Fig.17, it can also be seen that the highest Hit Ratio gain achieved with shared L2 is primarily for all configurations except configurations (3 and 4) preferred to shared L3 and L4.

### CONCLUSION

This paper introduces our research work of applying a simulator to evaluate the cache architectures in multicore processors, with a focus on L2, L3 and L4 architectures and configuration parameters. We found that the IPC and shared cache hit ratio are increased as the number of processor cores is increased till 32 cores, while for 64 cores or more the performance will be decreased. Therefore the cache architecture limitation in multicore processors is 32 cores.

Also, our evaluations help in determining the best cache architecture and configuration for a given number of cores to obtain the good performance, as shown below for 32 cores:

1. The best architecture for (both IPC and Hit Ratio) with (all configurations) is A3 (i.e. Shared L3-archit.) and then A5 (i.e. Shared L4- archit.).
2. The best configuration for (both IPC and Hit Ratio) with all architectures is C5 (i.e. increasing L1 and L2).
3. Architectures A3 and A5 improve (both IPC and Hit Ratio) for configuration C4 (i.e. increasing L2 and Way).
4. Architectures A3 and A5 improve (Hit Ratio only) for configuration C3 (i.e. increasing L2).
5. Configuration C4 (i.e. increasing L2 and Way) decreases (both IPC and Hit Ratio) for architecture A1.
6. Configuration C3 (i.e. increasing L2 only) improves only the IPC for architecture A1, but decreases the Hit Ratio.

### REFERENCES

Anderson, C. and J. L. Baer., (1993) A multi-level hierarchical cache coherence protocol for multiprocessors. In Parallel Processing Symposium, 1993., Proceedings of Seventh International, Newport, CA , 13-16 Apr 1993, pp.142 - 148. IEEE.

Anderson and J. L. Baer., (1995) Two techniques for improving performance on bus-based multiprocessors, High-Performance Computer Architecture, 1995. Proceedings. First IEEE Symposium on, Raleigh, NC, pp.264-275.IEE.

B. Nitzberg and V. Lo., (1991) Distributed shared memory: a survey of issues and algorithms. Computer, vol. 24, pp. 52-60.

Chun Liu; Anand Sivasubramaniam; Kandemir, M.,(2004) Organizing the Last Line of Defense Before Hitting the Memory Wall for CMPs., In Proceedings of the International Symposium on High-Performance Computer Architecture Madrid, Spain, February 2004, pp 176185,IEEE.

D. Benitez, J. C. Moure, D. I. Rexachs, and E. Luque.,(2006) Evaluation of the Field-programmable Cache: Performance and Energy Consumption. In Proceedings of the 3rd conference on Computing frontiers Ischia, Italy, May 2006.,pp 361372 ,ACM,New York, NY, USA.

F. N. Sibai.,(2008) On the performance benefits of sharing and privatizing second and third-level cache memories in homogeneous multi-core architectures, Microprocess. Microsystems. vol. 32, pp. 405-412.

- G. Dewan and P. Biswas., (1991) a snooping cache coherency protocol for hierarchically organized multiprocessors, *Microprocess, Microprogram*, vol. 31, pp.105-111.
- J.S. Vitter and E.A.M. Shriver., (1994) Algorithms for Parallel Memory II: Hierarchical Multilevel memories. *Algorithmica*, vol. 12, pp. 148-169.
- J.-L. Baer and W.-H. Wang. (1989) multilevel cache hierarchies: organizations, protocols, and performance. *J. Parallel Distrib. Comput.* vol. 6, pp. 451-476
- J. Bertoni, J.-L. Baer and W.-H. Wang. (1992) Scaling shared-bus multi-processors with multiple buses and shared caches: a performance study, *Microprocess. Microsystems.*, vol. 16, pp. 339-350.
- Jin Young Park and L. Choi., (2009) RING-DATA ORDER: A new cache coherence protocol for ring based multicores., In *High Performance Computing and Simulation, 2009. HPCS '09. International Conference on, Leipzig, 21-24 June 2009*, pp. 82-88.
- Lioupis, D., and Milios, S., (1997) Exploring cache performance in multithreaded processors, *Microprocessors and Microsystems, Microprogram*, vol. 12, pp. 148-169.
- M. K. Vernon, R. Jog, and G. S. Sohi., (1989) Performance analysis of hierarchical cache-consistent multiprocessors, *perform. Eval*, vol. 9, pp. 287-302.
- M. Modarressi, S. Hessabi, and M. Goudarzi.,(2006) A Reconfigurable Cache Architecture for Object Oriented Embedded Systems. In *Proceedings of the Canadian Conference on Electrical and Computer Engineering, Ottawa, Ont, May 2006*,pp 959962,IEEE.
- Marco Paolieri, Jorg Mische, Stefan Metzlaf, Eduardo Quinones, Sascha Uhrig, Sascha Uhrig, Theo Ungerer, Theo Ungerer, Francisco J. Cazorla.(2013) A Hard Real-Time Capable Multi-Core SMT Processor, *ACM Transactions on Embedded Computing Systems (TECS)* , vol. 12 Issue 3, March 2013, Article No. 79,ACM New York, NY, USA.
- Q. Yang. (1990) Performance analysis of a cache coherent multiprocessor based on hierarchical multiple buses. *Databases, Parallel Architectures and Their Applications. PARBASE-90, International Conference on, Miami Beach, FL, 7-9 Mar 1990*, pp. 248 - 257. IEEE.
- Ubal, R, Sahuquillo, J.; Petit, S.; Lopez, P., (2007) Multi2Sim: A Simulation Framework to Evaluate Multicore-Multithreaded Processors. *Computer Architecture and High Performance Computing, 2007. SBAC-PAD 2007. 19th International Symposium on,Rio Grande do Sul,24-27 Oct. 2007* ,pp 62 – 68 ,IEEE.
- Woo, S.C.; Ohara, M.; Torrie, E.; Singh, J.P.; Gupta, A. (1995) the SPLASH-2 Programs: Characterization and Methodological Considerations. *Proceeding ISCA '95 Proceedings of the 22nd annual international symposium on Computer architecture.* vol. 23, pp. 24-36.
- Wu, C.-J.; Martonosi, M.(2011) Characterization and dynamic mitigation of intra-application cache interference. *Performance Analysis of Systems and Software (ISPASS), IEEE International Symposium on.*, Austin, TX ,10-12 April 2011 , pp 2 11,IEEE
- Xuemei Zhao, Karl Sammut, Fangpo He, Shaowen Qin., (2007) Split Private and Shared L2 Cache Architecture for Snooping-based CMP. *Computer and Information Science, 2007. ICIS 2007. 6th IEEE/ACIS International Conference on, Melbourne, Qld, 11-13 July 2007*, pp 900 - 905, IEEE.