STUDY OF VARIOUS FACTORS AFFECTING PERFORMANCE OF MULTI-CORE PROCESSORS

Nitin Chaturvedi[1] S Gurunarayanan[2]

[1]Department of Electrical Electronics Engineering, BITS, Pilani, India
nitin80@bits-pilani.ac.in
[2]Department of Electrical Electronics Engineering, BITS, Pilani, India
sgru@bits-pilani.ac.in

ABSTRACT

Advances in Integrated Circuit processing allow for more microprocessor design options. As Chip Multiprocessor system (CMP) become the predominant topology for leading microprocessors, critical components of the system are now integrated on a single chip. This enables sharing of computation resources that was not previously possible. In addition the virtualization of these computation resources exposes the system to a mix of diverse and competing workloads. On chip Cache memory is a resource of primary concern as it can be dominant in controlling overall throughput. This Paper presents analysis of various parameters affecting the performance of Multi-core Architectures like varying the number of cores, changes L2 cache size, further we have varied directory size from 64 to 2048 entries on a 4 node, 8 node 16 node and 64 node Chip multiprocessor which in turn presents an open area of research on multi-core processors with private/shared last level cache as the future trend seems to be towards tiled architecture executing multiple parallel applications with optimized silicon area utilization and excellent performance.

KEYWORDS

Chip Multiprocessor (CMP), Multiple-Chip Multiprocessor (M-CMP), Tiled Architecture.

1. INTRODUCTION

Present sub-micron integrated circuit technologies have fueled microprocessor performance growth [3, 4]. Each new process technology increases the integration density allows for higher clock rates and offers new opportunities for micro-architectural innovation. Both of these are required to maintain microprocessor performance growth. Micro-architectural innovations employed by recent microprocessors include multiple instruction issue, dynamic scheduling, speculative execution, instruction level parallelism [6] and non-blocking caches. In the past, we have seen the trend towards CPUs with wider instruction issue and support for larger amounts of speculative execution but due to fundamental circuit limitations and limited amounts of instruction level parallelism, the superscalar execution model provides diminishing returns in performance for increasing issue width. Faced with this situation, building further a more complex wide issue superscalar processor was not at all the efficient use of silicon resources and a better utilization of silicon area. So researchers came up with a novel architecture which was constructed from simpler processors then super-scalar and multiple such processors are integrated on a single chip popularly known as chip multiprocessor or multi-core processors. To understand the performance trade-offs between wide-issue processors and single chip multiprocessors in a more quantitative way, researchers had compared performance of a six-issue dynamically scheduled superscalar processor with a 4 × two-issue multiprocessor. Comparison has a number of unique features. The results show that on applications that cannot be parallelized, the

DOI : 10.5121/ijdps.2013.4404

superscalar micro-architecture performs better than one processor of the multiprocessor architecture. For applications with fine grained thread-level parallelism the multiprocessor micro-architecture can exploit this parallelism so that the super-scalar micro-architecture is at most 10% better. For applications with large grained thread-level parallelism and multiprogramming workloads the multiprocessor micro-architecture performs 50–100% better than the wide super-scalar micro-architecture. Today data centres powering web and transaction services face new requirements as they attempt to deliver higher levels of content-rich and high-bandwidth services to increasing numbers of users. These workloads share certain common characteristics. They exhibit high Thread-Level Parallelism (TLP) with multiple and independent processes running concurrently. Moreover, servers equipped with even more powerful and power-hungry microprocessors to meet higher computational demands are pushing the power and cooling capabilities of these servers to their limits, resulting in increased operating costs and decreased system reliability. Therefore for achieving high performance while maintaining existing power and thermal envelopes requires novel microprocessor designs that not only focuses on performance but rather on the aggregate performance per watt. The rest of the paper is organized as follows: In Section 2, the performance limits of superscalar design are discussed and it also presents the case for a single chip multiprocessor from an applications perspective and extended to M-CMP. Section 3 gives details of Simulation methodology. Section 4 presented the obtained results and finally Section 5 concludes.

2. LITERATURE REVIEW

2.1. Super-Scalar Processors

A trend in the microprocessor industry has been the design of CPUs with multiple instruction issue and the ability to execute instructions out of program order. This ability, called dynamic scheduling uses hardware to track register dependencies between instructions; an instruction is executed, possibly out of program order, as soon as all of its dependencies are satisfied. The register dependency checking was done with a hardware structure called the *scoreboard*. IBM used register renaming to improve the efficiency of dynamic scheduling using hardware structures called reservation stations [2]. It is possible to design a dynamically scheduled superscalar microprocessor using reservation stations, the most recent implementations of dynamic superscalar processors have used a structure similar to the one shown in Figure 1, which shows three major phases of instruction execution in a dynamic superscalar machine they are fetch, issue and execute.

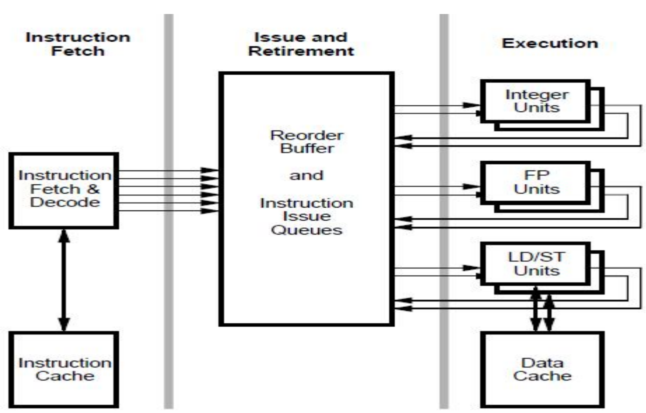

Figure1. Dynamic Super Scalar Microprocessor

The goal of the fetch phase is to present the rest of the CPU with a large and accurate window of decoded instructions. Three factors constrain instruction fetch: miss predicted branches, instruction misalignment, and cache misses. The ability to predict branches correctly is crucial to establishing a large, accurate window of instructions. However, good branch prediction is not enough. As previous work has pointed out, it is also necessary to align a *packet* of instructions for the decoder. Even with good branch prediction and alignment a significant cache miss rate will limit the ability of the fetcher to maintain an adequate window of instructions. Previous research have shown that over 60% of the instruction cache miss latency can be hidden on a database benchmark with a 64KB two way set associative instruction cache. In the issue phase, a packet of renamed instructions is inserted into the instruction issue queue. An instruction is issued for execution once all of its operands are ready. The results have shown a quadratic increase in the size of the instruction issue queue and researchers believe that the instruction issue queue will fundamentally limit the performance of wide issue superscalar machines. In the execution phase, operand values are fetched from the register file or bypassed from earlier instructions to execute on the functional units. The wide superscalar execution model will encounter performance limits in the register file, in the bypass logic and in the functional units. Wider instruction issue requires a larger window of instructions, which implies more register renaming. Not only must the register file be larger to accommodate more renamed registers, but the number of ports required to satisfy the full instruction issue bandwidth also grows with issue width. Again, this causes a quadratic increase in the complexity of the register file with increases in issue width.

2.2. Single Chip Multiprocessor

In today's information era, commercial servers constitute the backbone of the global information and communication system infrastructure. Such servers run useful commercial applications that are essential to many aspects of everyday life such as banking, airline reservations, web searching and web browsing. As more people depend on these multi-threaded throughput-oriented applications, demand for more throughputs is likely to increase for the foreseeable future. Commercial servers must therefore improve their performance by providing more throughputs to keep up with the application demand.

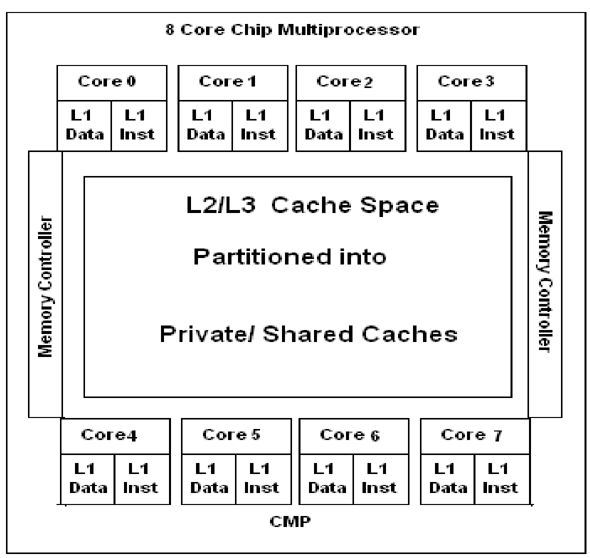

Figure2. Single Chip Multiprocessor (CMP)

Since commercial applications have abundant thread-level parallelism, commercial servers were designed as multiprocessor systems—or clusters of multiprocessors—to provide sufficient throughput. While traditional symmetric multiprocessors (SMPs) can exploit thread-level parallelism, they also suffer from a performance penalty caused by memory stalls due to cache misses and cache-to-cache transfers, both of which require waiting for long off-chip delays. Several researchers have shown that the performance of commercial applications and database applications in particular, is often dominated by sharing misses that require cache-to-cache transfers. To avoid these overheads, architects proposed several schemes to integrate more resources on a single chip. Researchers have shown that chip-level integration of caches, memory controllers, cache coherence hardware and routers can improve performance of online transaction processing workloads by a factor of 1.5. Simultaneous multithreading designs allow the processor to execute several contexts (or threads) simultaneously by adding per-thread processor resources. This approach also improves the performance of database applications compared to a superscalar processor with comparable resources. The trend towards more integration of resources on a single chip is becoming more apparent in CMP designs where multiprocessor systems are built on a single chip. Chip multiprocessor (CMP) systems can provide the increased throughput required by multi-threaded applications while reducing the overhead incurred due to sharing misses in traditional shared-memory multiprocessors. A chip multiprocessor design is typically composed of two or more processor cores (with private level-one caches) sharing a second- level cache. CMPs in various forms are becoming popular building blocks for many current and future commercial servers. The increasing number of processor cores on a single chip increases the demand on two critical resources: the shared cache capacity and the off-chip pin bandwidth.

2.3. Multiple-Chip Multiprocessor (M-CMP)

The increasing number of transistors per chip now enables Chip Multiprocessors (CMPs), which implement multiple processor cores on a chip. CMP-based designs provide high-performance, cost-effective computing for workloads with abundant thread-level parallelism, such as commercial server workloads. Smaller-scale Single-CMP (S-CMP) systems, such as Stanford Hydra and Sun UltraSparc T1 [1], use a single CMP along with DRAM and support chips. Larger-scale Multiple-CMP (M-CMP) systems, such as Piranha [7] and IBM Power4, combine multiple CMPs to further increase performance.

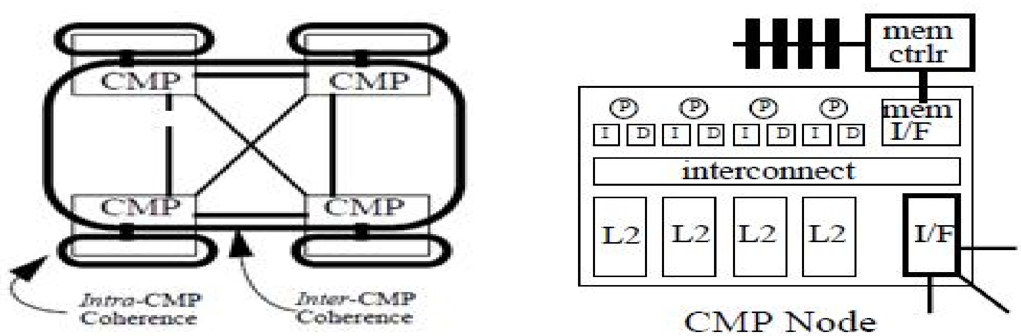

Figure3. Multiple-CMP

Because all of these systems use shared memory (to preserve operating system and application investment), a key challenge for M-CMP systems is implementing correct and high performance cache coherence protocols. These protocols keep caches transparent to software, usually by maintaining the coherence invariant that each block may have either one writer or multiple readers. S-CMP systems are conceptually straightforward and maintain coherence with traditional non-hierarchical snooping protocols (which rely on a logical bus) or directory protocols (which track cached copies at memory). M-CMPs present a greater challenge, because they must maintain both intra-CMP coherence and inter- CMP coherence.

3. SIMULATION METHODOLOGY AND APPLICATION

For evaluating the performance of the novel CMP or M-CMP micro-architectures requires a way of simulating the environment in which we would expect these architectures to be used in real systems. We have used GEM5 [8] full system functional simulator extended with Multi-facet GEMS which is popularly used in the research community. The heart of GEM5 is the Ruby memory simulator.

1. Execution time: The Ruby Cycle is our basic metric of simulated time used in the Ruby module of the GEMS simulator. The Ruby module is the basic module for the memory system configuration and interconnection network design The Ruby cycles are the recommended performance metric in GEMS.

2. L1 cache misses: As the name says it represents the misses of L1 cache. It's calculated by dividing request missed by number of requests (Instruction + Data). It's an important metric for cache hierarchy.

3. L2 miss/miss rate: This represents the total misses and miss rate of the L2 cache. It is calculated from the number of requests issued to the L2 and the misses of all banks of L2.

4. L2/Dir replacement: Number of replacements of L2/Directory entries. It's caused by capacity misses and conflict misses.

5. Miss latency average: Average of the L1 miss latency in Ruby cycles. It is measured from the moment a memory request is issued to the moment when the date is retrieved.

6. Memory requests: Number of reads and writes issued to main memory.

7. Applications: The benchmark used is a multithreaded implementation of a simulation of 3D Lattice-Boltzmann magneto-hydrodynamics (in other words, plasma turbulence). Pthreads are used to implement threading.

4. RESULTS

In order to analyze the impact of various parameters on the performance of Multi-core Architectures, we have varied the number of cores, which in turn changes L2 cache size further we have varied directory size from 64 to 2048 entries on a four node, 8 node 16 node and 64 node Chip multiprocessor.

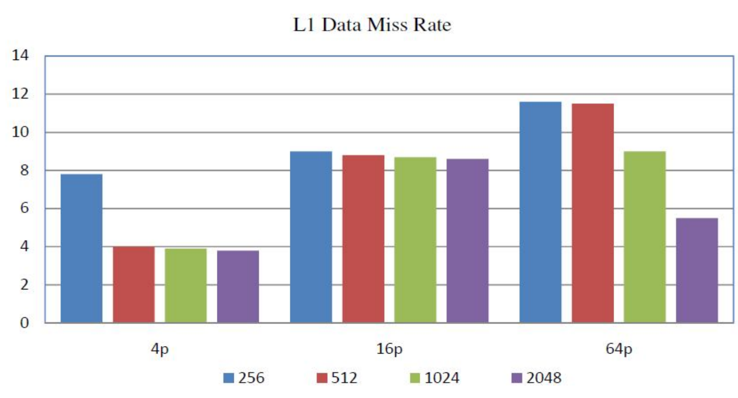

Figure4. L1 Data Miss Rate

All simulations in this section are configured with same cache size: 32KB L1I + 32KB L1D and 512KB L2 cache per core. The L1 data cache miss rate was calculated as dividing number of L1D misses by data requests. It was observed that L1 miss was greatly affected by directory si ze. It again influences L2 requests and on-chip traffics. Therefore, it's an important factor that determines CMP performance.

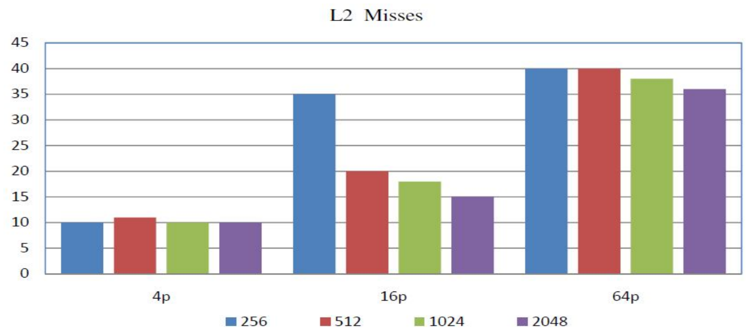

Figure5. L2 Misses

The number of L2 misses increases as no of cores within the system increases, nevertheless the miss rate decrease to 40%-80% because of the larger total L2 cache on chip.

Figure6. L2 Miss Rate

It was observed that L2 miss rate decreases with increase in director size. L2 misses is mainly determined by L2 cache size and more importantly application working set.

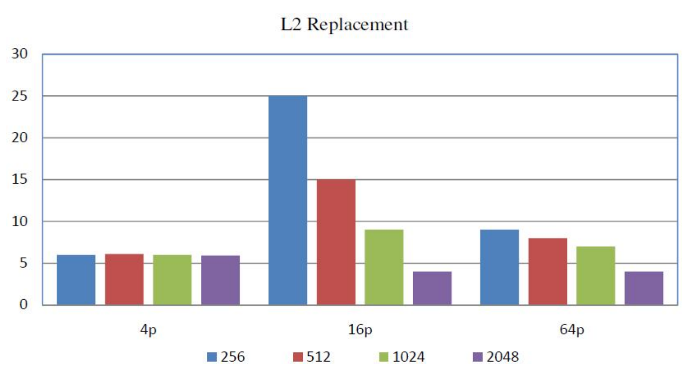

Figure7. L2 Replacement

L2 Replacement occurs when the L2 cache is full and another allocation is required. According to LRU policy, a block will be chosen and replaced by a new one. If this block is clean just ignore it and process allocation without pause. Otherwise the data block needed to be written back to main memory. It was observed that the replacements are influence by directory and L2 size and on the applications.

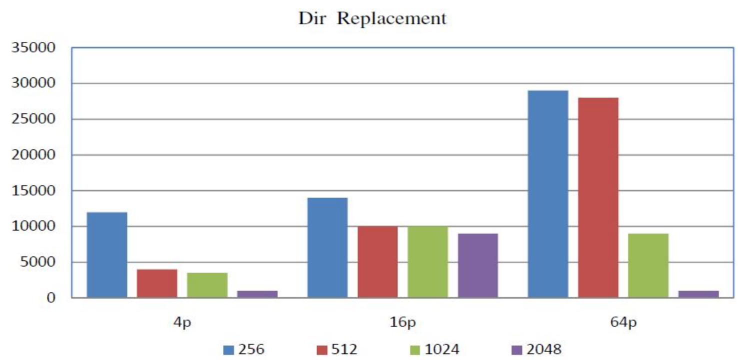

Figure8. Directory Replacements

On 16 core node we have observed that larger directory size does not improve the directory replacements. The reason is that so many data are mapped to the same location resulting in many conflicts. One possible solution is increase set associativity of directory to avoid the conflicts. The trend seems normal in other two cases.

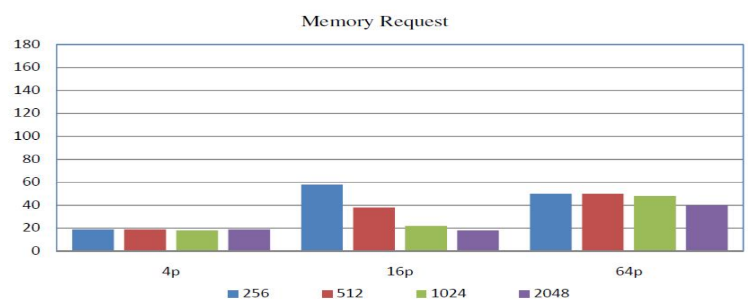

Figure9. Memory request

Memory request to some extent represents the requirement of memory bandwidth of applications. Comparing with L2 misses in previous figure it is observed that the memory read requests caused by L2 miss is the dominant faction of total memory requests.

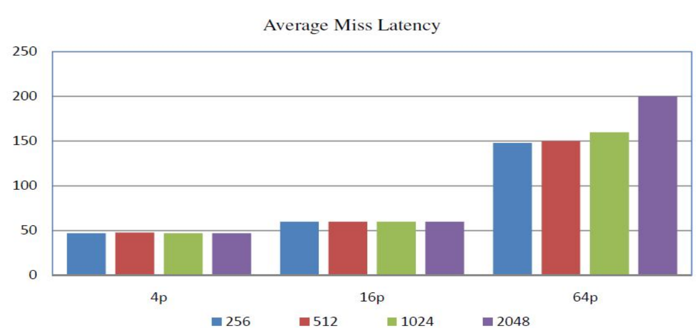

Figure10. Average Miss Latency

The miss latency increases 50% from 4-core to 16- core and 150%-230% from 16-core to 64-core. On a L1 miss, there are up to 3 nodes involved to fulfill the miss: local node, home node and remote node. The 3- way communication aggravates the latency of on-chip communication.

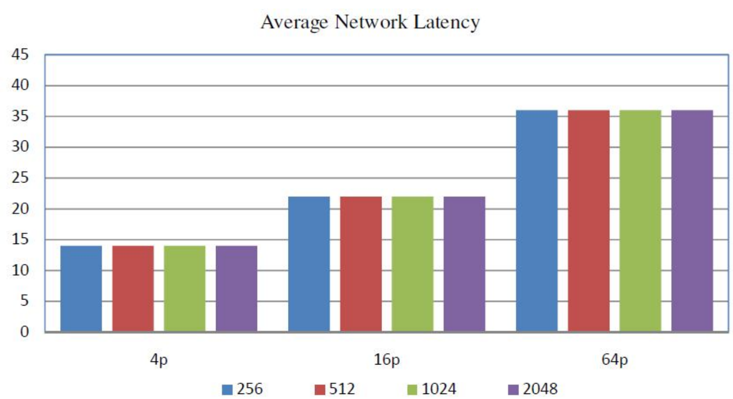

Figure11. Average Network Latency

With Hierarchical network model, we can measure the average network latency in detail. It goes from 14, 22 to 36 in there configurations. For all structure, the latency looks almost the same, which depends on network topology and on-chip link latency.

4. CONCLUSIONS

The current advanced submicron integrated circuit technologies require us to look for new ways for designing novel microprocessors architectures to utilize large numbers of gates and mitigate the effects of high interconnect delays. In this paper we have discussed the details of implementing both a wide, dynamically scheduled superscalar processor and a single chip multiprocessor. The alternative S-CMP is composed of simpler processors and can be implemented in approximately the same area. We believe that the S-CMP will be easier to implement and will reach a higher clock rate. Results show that on applications that cannot be parallelized the superscalar processor performs marginally better than one processor of the multiprocessor architecture. On applications with large grained thread-level parallelism and multiprogramming workloads the single chip multiprocessor performs 50–100% better than the wide superscalar processor. Novel architectures like M-CMP that exploits large bandwidth instead of extremely high frequency to achieve the target throughput performance by executing multiple threads concurrently in a shallow with simple pipeline effectively hides instruction and memory latency while utilizing resources. Results shows that the performance of these S-CMP and M-CMP depends on L2 cache size, no of directory entry and no of processors integrated on the single chip. Therefore, paper presents an open area of research on CMP to achieving high performance while maintaining existing power and thermal envelopes requirements and the designs must focus not only on performance but rather on the aggregate performance per watt, optimized interconnect topology, novel intra and inter CMP cache coherence protocol. Further performance gap between processors and memory require adaptive novel techniques to manage on chip cache memory judiciously.

REFERENCES

[1] A. S. Leon et al., "The UltraSPARC T1 processor: CMT reliability," in Proc. IEEE Custom Integrated Circuits Conf., Sep. 2006, pp. 555–562.

[2] D. W. Anderson, F. J. Sparacio, and R. M. Tomasulo, "The IBM System/360 model 91: Machine philosophy and instruction-handling," IBM Journal of Research and Development, vol. 11, pp. 8–24, 1967.

[3] T. C. Chen, "Where CMOS is going: trendy hype vs. real technology," presented at the IEEE Int. Solid-State Circuits Conf. (ISSCC), Sanrancisco, CA, Feb. 2006, Plenary Session 1.1.

[4] M. Horowitz and W. Dally, "How scaling will change processor architecture," in IEEE ISSCC Dig. Tech. Papers, Feb. 2004, pp. 132–133.

[5] K. Farkas, N. Jouppi, and P. Chow, "Register file considerations in dynamically scheduled processors," Proceedings of the 2nd Int. Symp. on High-Performance Computer Architecture, pp. 40–51, San Jose, CA, Feb 96.

[6] D. W. Wall, "Limits of Instruction-Level Parallelism," Digital Western Research Laboratory, WRL Research Report 93/6, November 1993.

[7] L. A. Barroso, K. Gharachorloo, R. McNamara, A. Nowatzyk, S. Qadeer, B. Sano, S. Smith,R. Stets, and B. Verghese. Piranha: A Scalable Architecture Based on Single-Chip Multiprocessing.In Proceedings of the 27th Annual International Symposium on ComputerArchitecture, pages 282–293, June 2000.

[8] Milo M.K. Martin, Daniel J. Sorin, Bradford M. Beckmann, Michael R. Marty, Min Xu, Alaa R. Alameldeen, Kevin E. Moore, Mark D. Hill, and David A. Wood, "Multifacet's General Execution-driven Multiprocessor Simulator (GEMS) Toolset" Computer Architecture News (CAN), pages 92-99, November 2005.

Authors

Nitin Chaturvedi

Received M.Sc (Electronics) from Devi Ahilya University Indore in 2002 and M.Tech from Panjab University Chandigarh in 2005. Presently working as a Lecturer and pursuing Ph.D. from Birla Institute of Technology and Science, Pilani, India. His field of interest is Computer Architectures, Digital CMOS Circuits.

S Gurunarayanan

Received B.E, M.E. and Ph.D. from Birla Institute of Technology & Science Pilani. Presently he is working as Professor having more than 22 years of teaching experience in EEE Group, BITS, Pilani, India. His area of interest includes Computer Architecture, VLSI System Design.