Next Generation Embedded Processor Architecture for Personal Information Devices

In-Pyo Hong, Yong-Joo Lee and Yong-Surk Lee

The School of Electrical and Electronic Engineering, Yonsei University,

134 Shinchon-dong, Seodaemoon-gu, 120-749, Seoul, Korea

{necross, leemann}@dubiki.yonsei.ac.kr, yonglee@yonsei.ac.kr

Abstract. In this paper, we proposed a processor architecture that is suitable for next generation embedded applications, especially for personal information devices such as smart phones, PDAs, and handheld computers. Latest high performance embedded processors are developed to achieve high clock speed. Because increasing performance makes design more difficult and induces large overhead, architectural evolution in embedded processor field is necessary. Among more enhanced processor types, out-of-order superscalar cannot be a candidate for embedded applications due to its excessive complexity and relatively low performance gain compared to its overhead. Therefore, new architecture with moderate complexity must be designed. In this paper, we developed a low-cost SMT architecture model and compared its performance to other architectures including scalar, superscalar and multiprocessor. Because current personal information devices have a tendency to execute multiple tasks simultaneously, SMT or CMP can be a good choice. And our simulation result shows that the efficiency of SMT is the best among the architectures considered.

1. Introduction

Using simple RISC processors as CPUs for personal information devices such as smart phones, PDAs, and handheld computers is one of the major applications of embedded processors. The characteristics of application programs are similar to that of desktop computers and the need for high performance is getting stronger. However, the limitation in chip size and power consumption prevent the handheld computers from applying more powerful processor cores as their CPUs. The latest commercial processor, ARM11 core, adopts single-issue in-order scalar architecture which is the simplest form among current processor types. Current architectural evolutions in this kind of processors concentrate on increasing clock speed through deeper pipelines [1]. Some embedded processors employ high performance out-of-order superscalar technique that induces large chip area and design complexity. However, they are mainly used for network/communication equipments or home game machines that are neither mobile nor battery-powered. Due to their complexity, these processors cannot be used for handheld devices. Therefore, new

processor architecture that boosts architectural performance while suppressing the complexity under the affordable level is needed.

This paper is organized as follows. In section 2, previous works are described. In section 3, architecture models that we simulated are presented in detail. Section 4 shows our simulation methodology and workloads. And after we present simulation results in section 5, section 6 concludes.

2. **Previous Works**

Most conventional embedded processors adopt simple scalar architecture with 3 to 5 stages of pipeline. These processors can work well when applications are limited to simple personal data management and when multi-tasking is not necessary. However, because mobile internet and multimedia applications are getting popular, personal information devices are requested to cover the application domain of desktop computers. Architectural evolution in this field is concentrated on increasing clock speed while minimizing performance penalty of longer pipeline and suppressing power consumption overhead by using intelligent clock speed and voltage control [1]. Although various techniques are used to prevent negative effects, super-pipelining induces decrease in IPC and fast clock speed requires more power [2][3]. Therefore architectural enhancement that can fundamentally overcome these problems is needed.

Some researchers tried to adopt multithreading to embedded processors [4][5]. They added multiple register sets for multiple hardware contexts and made instructions from multiple threads issue-able cycle by cycle while keeping the other part of the processor unchanged. This architecture type is a kind of fine-grain multithreading. With this simple multithreading, they intended to improve response of processors to randomly triggered events because multithreading architecture does not require context switching on interrupt handling. However, the throughput of these processors is limited to 1 IPC even in ideal cases. Although they invest the largest overhead of multithreading, multiple register sets, the potential of multithreading cannot be fully exploited in the fine-grain multithreading architecture. It is mainly because issue width is restricted to one instruction while more TLP (Thread Level Parallelism) exist. As a result, previous multithreading processors improve response of the processor when multiple events are pending and make real performance closer to the ideal level by switching threads dynamically in wasted cycles.

Enhanced architectures that can improve architectural performance are superscalar and SMT (Simultaneous MultiThreading) [6]. However, it is impossible to use these gigantic processor architectures for our target application that requires tiny and simple hardware, unless it is simplified dramatically. A few researches tried to adopt the SMT technique to embedded applications. However, they are very limited to the applications that require massive parallel data processing such as network processors. In most cases, the complexity is not a matter of concern because the equipments have sufficient space and stable power supply [7][8].

3. **Architecture Models**

3.1. Baseline Architecture

Our baseline architecture is a reference for comparison against proposed architecture. It resembles current embedded processors which has longer pipeline and single instruction issue mechanism. The instruction set architecture (ISA) is compatible with ARM ISA version 5.

The pipeline has 7 stages and designed to achieve higher clock frequency. Fig. 1 shows the microarchitecture and the pipeline. In fetch stage, a single-ported 128-entry branch target buffer with bimodal direction prediction fields is installed. To track dependencies, a scoreboard is employed To access the scoreboard array and calculate the availability of source operands, separated issue stage is needed. The ARM ISA requires a maximum of three source registers for shifter operand feature. Increasing register read path to three ports incurs additional register read delay and the shifter that must be serially attached to ALU induces great timing overhead to the functional unit. Therefore we divided execution stage to two stages, that is, a register read stage and an execution stage. The architecture described above fetches and executes a single instruction each cycle. We call this architecture scalar architecture in the rest of this paper.

The scalar architecture can be easily extended to execute two instructions per cycle in program order. Although out-of-order scheduling of instructions is needed to maximize ILP, it induces tremendous overhead. Therefore we restrict the issue order to strict program order. First of all, the fetch and decode bandwidth is increased to two instructions. Secondly, to check dependencies successfully, the number of scoreboard array ports is doubled. To fetch the increased number of operands and execute multiple instructions, additional ports of register file and a supplementary ALU is also employed. Finally, a write port of register file is added to support writeback of two instructions in the same cycle. We call this architecture as superscalar in the rest of this paper.

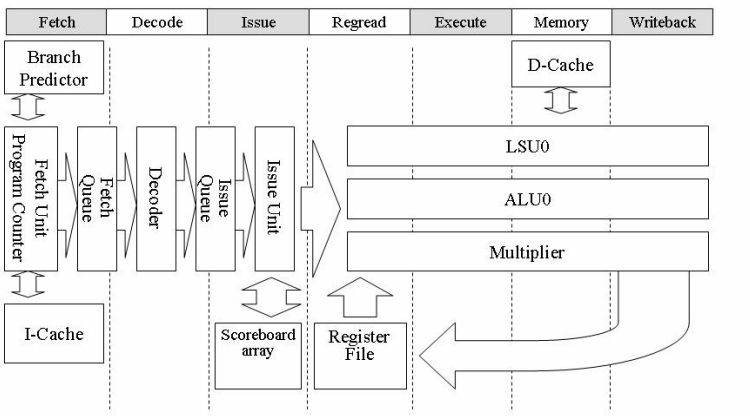

Fig. 1. The microarchitecture and pipeline structure of the baseline processor

3.2. Multiprocessor Architecture

Multiple scalar or superscalar processors can be organized to form a multiprocessor model. We organized two or four of our scalar and superscalar model into multiprocessor models. Each processor core has its own level-1 caches because the level-1 cache must be tightly coupled with the processor core and sharing a cache induces timing overhead. Because there is no level-2 cache in embedded processors, the cores communicate with each other through the main memory bus.

3.3. In-order SMT Architecture

We developed an SMT architecture that executes instructions in program order. Although we used the term, SMT, the architecture does not resemble the original SMT except for the fact that multiple threads share execution resources. Our multithreading architecture is from our in-order baseline processor models.

To fetch instructions from multiple threads, thread selection stage is added. In select stage, the hardware calculates priorities of each thread and determines which thread is to participate in fetching next cycle. The number of threads that fetches instructions each cycle is the same with the number of I-cache ports. It is known that it is desirable for the I-cache to have multiple ports to maintain good instruction mixture in instruction queue [9]. After decoding the fetched instructions, a decoder puts them into per-thread instruction queues. In these queues, dependency checking and selection for issue take place. In scalar architecture, issue unit checks whether the oldest instruction in the queue can be issued or not using scoreboard. If the result is positive, the instruction is sent to a functional unit. Our SMT architecture extended this decision procedure. If the issue bandwidth is represented as *n* instructions per cycle, per-thread dependency check logic determines whether the preceding *n* instructions can be issued or not. After the decision is over, select logic chooses *n* instructions among all issue-able instructions across the threads, as shown in fig. 3.

An essential part that needs to be modified to support multithreading is the register file. Because the

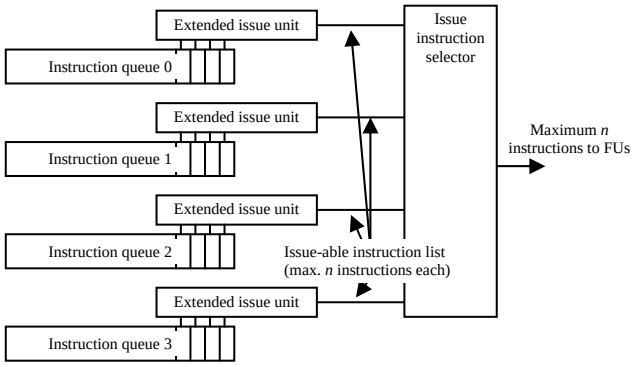

Fig. 2. The Issue Unit of the SMT architecture

hardware must maintain contexts of multiple threads, register file, program counter and status registers of every thread must be duplicated. This is the largest overhead of our multithreading architecture. While the size of register file must be increased and renaming unit must be changed to accommodate the new register file in the out-of-order SMT architecture, the register file in in-order SMT is merely duplicated and multiplexed. Moreover, because the number of architectural registers is the same with that of physical registers in our SMT model, absolute complexity of our register file is much lower than that of the original SMT architecture. Another modification is in dependency tracking mechanism. Our baseline architecture adopts a dependency tracking method in which every single register needs a corresponding scoreboard entry. Therefore, each thread must have its own scoreboard array.

Our multithreading architecture models are divided into two categories. The first is a fine-grain multithreading architecture with issue bandwidth of one instruction. And the second is SMT architecture that can issue multiple instructions from any thread simultaneously. Fig. 4 shows the two architectures. In this figure, the number of supported threads is set to two and maximum two instructions can be issued per cycle in the SMT architecture model.

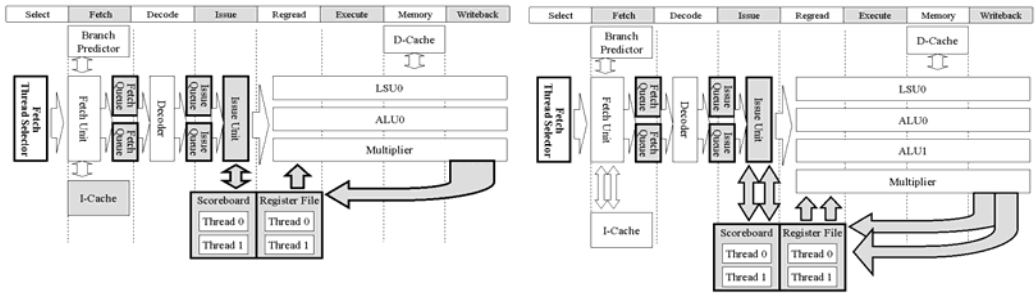

Fig. 3. Multithreading architecture models; single issue fine-grain multithreading (left), in-order SMT (right)

4. Methodology

A cycle-based execution-driven simulator is used in this research. The core part of the simulator is from our previous research [10]. ARM ELF binary loader and system call handling routine that is compatible with ARM Linux is derived from Simplescalar ARM port [11]. Multiple ARM binaries are read simultaneously and participate in the simulation on multithreading architecture. By setting the number of supported threads to one, the simulator operates as a scalar or a superscalar architecture model.

When performing simulation, we set initial options as described in table 1.

In this research, most workloads are derived from MiBench embedded benchmark suite [12] and we programmed an application that performs the IEEE 802.11 WLAN WEP (Wired Equivalent Privacy) algorithm [13]. In personal information devices, internet and streaming multimedia applications are getting as important as traditional personal data management applications that replace a pocket diary.

Therefore, workloads are organized to represent the trend that requires multi-tasking of multimedia and networking applications. Detailed information about the workloads is as follows.

- Web browsing: While the processor is receiving packets from a secured WLAN channel, it decodes a html text and a jpeg image.
- Streaming audio and e-book: While the processor is decoding a streaming MP3 audio sequence via WLAN channel, it searches a specific word in an e-book text.
- Smart phone (Voice phone call): While encoding a raw voice data into ADPCM signals and decoding reversely, the processor handles a GSM communication channel.
- Smart phone (Web browsing via GSM channel): While receiving data from a GSM channel, the processor decodes an html text and a jpeg image.
- Secured file transmission: The processor encrypts a plain text file into AES encrypted text and transmits it through a WLAN channel with enabled WEP.

Table 1. Brief descriptions about simulated architecture models

	Scalar	Superscalar	FGMT	SMT_sm	SMT_lg	CMP_is*N*_pr*M*
Thread	1	1	4	4	4	*M*
Pipeline	7-stage	7-stage	8-stage	8-stage	8-stage	7-stage
Issue width	1	2	1	2	4	*N*
Functional units	1 ALU 1 LSU 1 Multiplier	2 ALU 1 LSU 1 Multiplier	1 ALU 1 LSU 1 Multiplier	2 ALU 1 LSU 1 Multiplier	4 ALU 2 LSU 1 Multiplier	(N ALU)*M (N/2 LSU)*M (1 Multiplier)*M
I cache	8KB / 8-way 1 port	8KB / 8-way 1 port	8KB / 8-way 2 ports	8KB / 8-way 2 ports	8KB / 8-way 2 ports	(8KB / 8-way 1 port)*M
D cache	8KB / 8-way 1 port	8KB / 8-way 1 port	8KB / 8-way 1 port	8KB / 8-way 1 port	8KB / 8-way 2 ports	(8KB / 8-way 1 port)*M
BTB	128	128	128	128	128	128*M

5. Simulation Results and Optimization

5.1. Overall Performance Comparison

We have simulated eight architecture variations. In Fig. 5, the performance evaluation results are shown. Because in-order instruction scheduling is employed, superscalar architecture cannot increase performance dramatically. It shows only 19% of speedup which is not sufficient for next-generation processors. On the other side, architectures that utilize thread-level parallelism raise performance greatly. Even the simplest single-issue FGMT achieves 51% increased performance. Two-issue processor models, SMT_sm and CMP_is1_pr2, nearly double the performance. The largest four-issue processor models, SMT_lg, CMP_is2_pr2 and CMP_is1_pr4, increase performance by the ratio of 120% to 280%. In the absolute performance aspect, CMP with four individual scalar cores is the best. However, it means that four individual processors with their own caches on a chip are needed. While threads in SMT architecture

share many hardware resources, such as caches, branch predictors, TLBs, and functional units, the CMP must have those individually. Because most part of the chip area is devoted to memory blocks including cache memory and MMU, sharing capability of the SMT has great influence on overhead. Therefore, cost-performance efficiency of the SMT is better than that of the CMP. Moreover, the CMP_is1_pr4 shows the same performance with the scalar architecture when there is only one thread to execute. On the other side, the SMT can execute the thread as fast as the superscalar architecture, that is, 19% faster than the CMP.

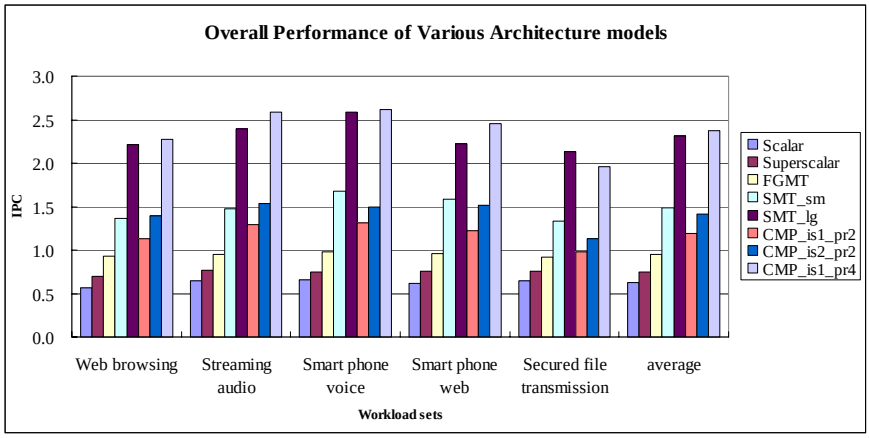

Fig. 4. Performance evaluation results of various architecture models

5.2. More Practical SMT Architecture

To exploit TLP (Thread Level Parallelism) well, instructions from multiple threads must be fairly distributed across the per-thread instruction queues. For this reason, fetching instructions from multiple threads every cycle has positive effect on performance. Therefore, our multithreading architecture also has multiple instruction cache ports, one port for two threads. In our simulation, limiting the number of I-cache ports to one induces average 14.8% of performance penalty in four-thread SMT_lg architecture when thread selection algorithm is round-robin. However, it can be recovered by changing the thread selection algorithm to a more intelligent one. We adopted a selection algorithm which calculates the number of total instructions in instruction queues separately according to threads and selects the thread that has the smallest number of instructions in the queues. By doing so, the simulation result with one I-cache port can be the same as that with two I-cache ports in the SMT_lg architecture model. We used ECACTI model to estimate the effect of the number of cache ports [14]. In a cache memory with 8KB of capacity, 32-byte of line size and 8-way set associative configuration, reducing two read ports to a single port decreases access time from 1.28ns to 0.96ns when 0.10um fabrication process is supposed. Total area is diminished by about 41%, in detail, from 0.017cm^2 to 0.010cm^2. We can achieve 33% of increase in clock speed by simplifying the I-cache and employing the intelligent thread selection algorithm.

Another impractical part is in the register file. In our SMT architecture, a maximum of four instructions from a thread can be issued every cycle. This incurs serious overhead in register file. As stated before, ARM ISA requires maximum three source operands per instruction and four instructions uses maximum 12 read ports and four write ports in worst case. This means that each per-thread register set must have 12 read ports and four write ports to supply source operands of four issued instructions successfully. This excessive number of register file ports is a problem. To mitigate this overhead, we restrict the maximum number of issue-able instructions per thread to two instructions per cycle. By doing so, the number of read ports can be diminished to six, and write ports to two as depicted in Fig. 6. This simplification also affects the number of scoreboard array ports. Before the issue restriction, the issue unit must read the dependency information of four instructions per thread. However, it is reduced to two instructions per thread in a similar way of register file. The architectural performance is decreased by 3% in SMT_lg model because it limits the ability of hardware to use ILP. However, this reduces per-thread register file access time by 41%, from 1.12ns to 0.65ns.

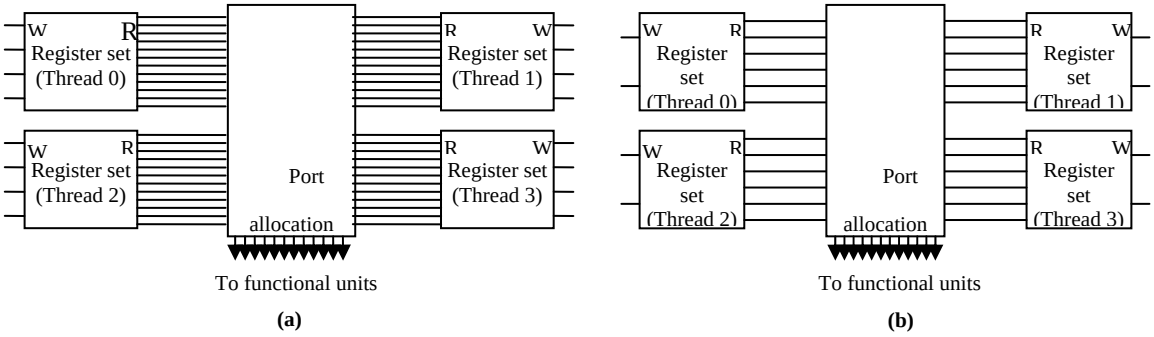

Fig. 5. Reducing the number of register file ports (a) before limiting issue width (b) after limiting issue width to two instructions per thread

5.3. Rough Estimation on Complexity

To evaluate cost-performance relations, we estimated gate counts of all the architectures simulated. Instead of designing all processor models into hardware, we manipulate the approximated area overhead based on the gate count of real ARM9 processor hardware. Total gate counts of a new architecture can be estimated by manipulating gate counts of individual sub-blocks. Though it is not accurate, it can be used as data for comparison. Therefore, meaningful data is not absolute gate counts but relative size. The result is summarized in table 2. To compare the architectures, the unit of IPC per million gates (IPC/area field in the table) is used in the table. Although IPC values of SMT_lg and CMP_is1_pr4 are the best, cost-performance of SMT_sm and SMT_lg is better. Therefore, the in-order SMT architecture can be a strong candidate of next generation processors for personal information devices.

Table 2. Estimated area overhead and cost-performance

	Scalar (ARM9)	Super-scalar	FGMT	SMT_sm	SMT_lg	CMP *is1_pr4*
ALU	8.1	16.2	8.1	16.2	32.4	8.1*4
Multiplier	9.8	9.8	9.8	9.8	9.8	9.8*4
Register file	22	44	88	176	176	22*4
Control	4.2	6.3	6.3	6.3	8.4	4.2*4
Decode	2	4	2	4	8	2*4
Forwarding	2	4	2	4	8	2*4
MMU	87	87	87	87	87	87*4
BUS i/f	12	12	12	12	12	12*4
Cache	278	278	278	278	278	278*4
Total area	425.1	461.3	493.2	593.3	619.6	1748.4
Area overhead	-	8.52%	16.02%	39.57%	45.75%	311.29%
IPC/area	1.49	1.62	1.92	2.50	3.73	1.36

6. Conclusions

In this paper, a new processor architecture that is suitable for the CPU of personal information devices is proposed. Simple architectures that are focused on using TLP rather than ILP can achieve high performance without complex out-of-order scheduling of instructions. It is due to the application characteristics of latest personal information devices. Current applications of personal computing machines are getting more network and multimedia oriented than ever while traditional applications merely managing personal information that consists of text data. This inclination requires the processor core to perform multi-tasking.

Under the assumption above, we set up several workload sets that represent applications of the personal information devices, and various processor architectures that have complexity within affordable level are simulated by using the workloads. To avoid excessive overhead, all processor models employ in-order instruction scheduling policy. And some optimizations that reduce hardware overhead of in-order SMT model is proposed. Because the processor cannot exploit ILP well in in-order issue mechanism, superscalar shows the speedup under 20%. On the other hand, SMT and CMP architectures that uses TLP as their performance source outperforms superscalar model with large gap. Although IPC values of the two architectures in case of multithreaded workloads are similar, SMT occupies less area and complexity. Moreover, the SMT can achieve similar performance with superscalar when executing single-threaded workload while the CMP cannot. As a result, proposed simple SMT shows the best cost-performance efficiency.

Acknowledgements

The work presented in this paper is supported by Korea Sanhak Foundation.

References

1. David Cormie, *The ARM11^{TM} Microarchitecture*, ARM Ltd, 2002
2. V. Agarwal, M. S. Hrishikesh, S. W. Keckler and D. Burger,"Clock Rate versus IPC: The End of the Road for Conventional Microarchitectures", Proc. of the 27th Annual International Symposium on Computer Architectures, pp. 248 - 259, 2000
3. Claasen, "High speed: not the only way to exploit the intrinsic computational power of silicon," Digest of Technical Papers, Solid-state Circuits Conference, pp.22~25, 1999
4. U. Brinkschulte , C. Krakowski , J. Kreuzinger , Th. Ungerer, "A Multithreaded Java Microcontroller for Thread-Oriented Real-Time Event Handling," Proceedings of the 1999 International Conference on Parallel Architectures and Compilation Techniques, p.34, October 12-16, 1999
5. J. Kreuzinger, A. Schulz, M. Pfeffer, T. Ungerer, U. Brinkschulte, C. Krakowski. "Real-time scheduling on multithreaded processors," the proceedings of seventh international conference on Real-Time Systems and Applications, pp. 155, 2000
6. Dean M. Tullsen, Susan J. Eggers, and Henry M. Levy, "Simultaenous Multithreading: Maximizing On-Chip Parallelism," Proceedings of 22nd Annual International Symposium on Computer Architecture, pp. 392-403, Santa Margherita Ligure, Italy, May 1995
7. Patrick Crowley, Marc E. Fiuczynski Jean_Loup Baer, and Brian N. Bershad, "Characterizing Processor Architectures for Programmable Network Interfaces," Proceedings of the 2000 International Conference on Supercomputing, May 2000.
8. Peter N. Glaskowsky, "Networking Gets XStream," Microprocessor Report, November 13, 2000
9. Dean M. Tullsen, Susan J. Eggers, Joel S. Emer, and Henry M. Levy, "Exploiting Choice: Instruction Fetch and Issue on an Implementable Simultaneous Multithreading Processor," Proceedings of 23rd Annual International Symposium on Computer Architecture, pp. 191-202, Philadelphia, Pennsylvania, May 1996
10. Byung In Moon, Moon Gyung Kim, In Pyo Hong, Ki Chang Kim, and Yong Surk Lee, "Study of an In-order SMT Architecture and Grouping Schemes," International Journal of Control, Automation, and Systems, Volume 1, Number 3, pp.339~350, September 2003
11. *SimpleScalar Version 4.0 Test Releases*, SimpleScalarLLC, http://www.simplescalar.com/v4test.html
12. Matthew R. Guthaus, Jeffrey S. Ringenberg, Dan Ernst, Todd M. Austin, Trevor Mudge, Richard B. Brown, "MiBench: A free, commercially representative embedded benchmark suite," IEEE 4th Annual Workshop on Workload Characterization, Austin, Texas, December 2001
13. ANSI/IEEE Std 802.11-1999, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications
14. Mahesh Mamidipaka and Nikil Dutt, "eCACTI: An Enhanced Power Estimation Model for On-chip Caches," Center for Embedded Computer Systems (CECS) Technical Report TR-04-28, Sept. 2004