Impact of Warp Formation on GPU Performance

Hong Jun Choi, Dong Oh Son, and Cheol Hong Kim

***Abstract*—As computing power of GPU increases dramatically, the GPU is widely used for general-purpose parallel applications as well as graphics applications. Especially, programmers using the GPU can easily create multiple threads with the help of APIs provided by GPU vendors. In GPU architecture, threads are grouped into a warp to run on the SIMD pipeline, leading to high performance. However, computational resources of GPU are not fully utilized in executing general-purpose applications due to control-flow instructions, resulting in performance degradation. To improve the GPU performance, several warp formations for handling branch divergence due to control-flow instructions have been proposed. In this work, we analyze the GPU performance according to warp formations with real GPU hardware configuration. Our simulation results show that the warp formation providing high hardware utilization does not guarantee high performance if hardware resources are not fully supported. Therefore, hardware configuration should be considered together with hardware utilization to improve the GPU performance by using warp formation.**

***Index Terms*—Control-flow instruction, GPU, warp formation, utilization.**

I. Introduction

Continuing advances in semiconductor technology enables increasing chip density, leading to improved computing power of the chip. Unfortunately, increased chip density causes high power consumption [1]. Therefore, architecture-level approaches for improving the performance of the chip with power constraints have become one of major challenges. Exploiting parallelism can be a good solution to improve the performance of the chip with power constraints. Exploiting parallelism with ILP (Instruction Level Parallelism) has limitations due to high complexity of scheduling logic and large caches [2]. For this reason, researches to exploit parallelism with TLP (Thread Level Parallelism) have been widely proposed. In the TLP, hardware designer as well as software designer should focus on improving the parallelism [3]. Many-core processors such as GPU(Graphics Processing Unit), which are composed of many processing units, can provide high computational power to improve the TLP[4].

Typically, the GPU is used as a hardware accelerator for graphics rendering operations in common computer systems. To support real-time graphics rendering operations requiring high computational power, up-to-date GPUs provide programmable hardware for graphics algorithms, whereas previous GPUs are consisted of fixed number of functional rendering pipeline. Programmable hardware inside the GPU is named as shader core, which can run diverse graphics operations, enables high flexibility and high throughput of the GPU. To take advantage of high flexibility and high throughput of recent GPUs, general-purpose parallel applications as well as graphics applications are executed on the GPU [5].

In order to utilize the GPU for executing general-purpose computations, GPU vendors provide various APIs (Application Programming Interface) such as CUDA, OpenCL, Cg, HLSL and GLSL [6]–[10]. Contrary to graphics applications, general-purpose applications tend to have branch divergence due to control-flow instructions [11]. For this reason, recent GPU architecture allows different program paths caused by branch divergence. However, branch instructions incur serious performance degradation of the GPU, because GPU has fine-grained multi-threading on a SIMD (Single Instruction Multiple Data) pipeline[12], [13]. To overcome the performance degradation due to control-flow instructions, recent GPUs support solutions for branch divergence, called warp formation. Several warp formations for GPUs have been studied: SIMD serialization, SIMD reconvergence, SIMD dynamic warp formation [14]–[19].

In general, high-utilization of hardware resources in ideal SIMD architecture leads to high performance. However, computational and peripheral hardware resources in real SIMD architecture are restricted. In this work, we investigate the impact of warp formation and hardware configuration on GPU performance. Especially, we present the guideline for improving the performance of SIMD architecture by analyzing the correlation between hardware configuration and warp formation. The rest of this paper is organized as follows: Section II describes background of our work. Section III presents the simulation infrastructure and detailed experimental results. Finally, Section IV concludes this paper.

II. Background

A. GPU Architecture

Fig. 1 illustrates the GPU architecture of our baseline, similar to NVIDIA Quadro FX5800 [20].

The GPU consists of 30 shader cores which have 32 SIMT (Single Instruction Multiple Thread) lanes, interconnection network and DRAM. In a shader core, 32 SIMT lanes can

Manuscript received October 4, 2012; revised January 21, 2013. This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology(2012R1A1B4003492) and following are results of a study on the "Leaders Industry-university Cooperation" Project, supported by the Ministry of Education, Science & Technology(MEST).

The authors are with the School of Electronics and Computer Engineering, Chonnam National University, Gwangju, 500-757, South Korea (e-mail: chj6083@gmail.com, sdo1127@gmail.com, chkim22@chonnam.ac.kr).

simultaneously execute up to 32 threads per cycle. To support the concurrent execution, threads are combined into group called warp, which executes together using a single program counter. Quadro FX5800 supports 32 active warps for a total of 1,024 active threads per shader core.

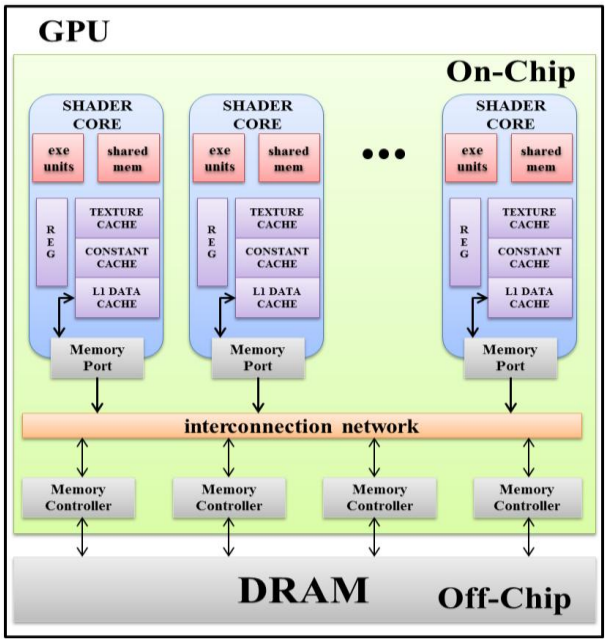

Fig. 1. Baseline architecture.

To accommodate many threads, it has huge on-chip register file resources. Threads within a warp can communicate via a shared memory which is implemented within each shader core as an SRAM. Data from off-chip DRAM is cached on-chip in the local memory (Texture Cache, Constant Cache, and L2 Data Cache) [21].

B. Warp Formation

After a branch instruction is executed, a warp is divided into two (or multiple) warps since a SIMD pipeline executes one single instruction in the same cycle, resulting in performance degradation due to underutilization. To solve this problem, several warp formations have been proposed for handling branch divergence. In this section, we describe three techniques for branch divergence.

1) *SIMD serialization* [14]–[15]

SIMD serialization is to handle branch divergence through serializing the threads within a warp when a branch instruction is executed. SIMD serialization is simple and can be implemented easily, while it cannot guarantee high performance. After serializing the threads in a warp with branch divergence, a single warp is separated into multiple warps. Threads in each separated warp follow the same program path. Separated warps are executed independently and never come back into a single warp. In other words, threads within a warp will be diverged until execution of threads in a warp is finished, since SIMD serialization does not support branch reconvergence. Therefore, SIMD serialization degrades the utilization of computational resources significantly. In this work, *NRCV (no reconvergence)* represents the SIMD serialization.

2) *SIMD reconvergence* [16]–[18]

When a branch instruction is executed, one path is taken and the other is not-taken. In SIMD reconvergence, threads in a warp separated by the branch instruction are combined together after divergent paths have been completed. All separated threads reach a merging point called reconvergence point in SIMD reconvergence. Therefore, program path following the reconvergence point is always precise, even though control-flow behavior is unclear.

Post-dominator, which is a popular SIMD reconvergence approach, is employed in recent commercial GPUs to handle branch divergence [18]. Dominance in post-dominator is identified by compiler for code optimization. In this approach, A post-dominates B ("A *pdom* B") means that every path from B to the exit node goes through A. To accomplish the reconvergence in post-dominator, GPU employs the divergence stack. An entry of the divergence stack is consisted of three fields: PC, active mask, and re-convergence point (or return PC). In this work, *PDOM* represents the SIMD post-dominator.

3) *SIMD dynamic warp formation* [19]

In *PDOM*, after a warp is generated, threads in a warp are processed in lock-step with a single instruction until the warp is completed. Similar to *PDOM*, threads in a warp generated by dynamic warp formation are processed in parallel with a single instruction. However, when a single instruction in a warp is finished, a warp in dynamic warp formation is broken while a warp in post-dominator is sustained until the warp is completed. In other words, after a single instruction is finished, threads in a warp are divided into the new warp which has same PC. Then, new warp is processed in lock-step with a single instruction if it is selected by the scheduler. Therefore, threads in a warp generated by dynamic warp formation always have a single PC, leading to full warp occupancy. In this work, *DWF* represents the SIMD dynamic warp formation.

III. Experiments

In this section, we briefly describe the simulation infrastructure. In order to evaluate the performance according to various warp formations and hardware configurations, we perform applications selected from NVIDIA CUDA SDK [22] using GPGPU-sim[23]. GPGPU-sim is constructed from SimpleScalar[24] to model various aspects of the parallel architecture such as GPU.

A. Experimental Methodology

In this paper, our baseline GPU is modeled by NVIDIA Quadro FX5800 [20]. Table I shows the GPU system configuration and memory hierarchy parameters used in our simulations. In Table I and Table II, bold values show our baseline GPU architecture.

B. Experimental Results

In this section, we analyze the performance according to warp formation, computational resource, memory resource. In the results, *NRCV*, *PDOM* and *DWF* represent SIMD serialization, post-dominator and dynamic warp formation, respectively. In this work, we execute diverse benchmarks (BFS, fastWalshTransform, scanLargeArray, scan, MersenneTwister, RAY) [21] to analyze the characteristics

of the GPU with various general-purpose applications. But, we present average value due to limited page.

TABLE I: HARDWARE CONFIGURATION

Parameter	Value
Number of Shader Cores	9/15/**30**
Warp Size	8/16/**32**
SIMD Pipeline Width	8/16/**32**
Number of Threads/Core	**1024**
Shared Memory/Core	**16KB**
Constant Cache/Core	**8KB, 2-way 64byte lines, Read-only**
Texture Cache/Core	**8KB, 2-way 64byte lines, Read-only**
L1 Data Cache	**32KB, 4-way, 64byte lines**
Number of L1 Cache Bank	4/2/**1**
Clock (Core: Interconnection: DRAM)	**325MHz: 650MHz: 800MHz**
Number of Memory Controller	**12**
Number of Memory Chip/Controller	**2**
Memory Channel Bandwidth	16/8/**4 bytes**
DRAM Request Queue Size	**32**
GDDR3 Memory Timing	**tCL=10, tRP =10, tRC=35, tRAS=25, tRCD=12, tRRD=8**

TABLE II: INTERCONNECTION NETWORK CONFIGURATION

Parameter	Value
Topology	Fly
Routing Mechanism	Destination Tag
Routing Delay	0
Virtual Channels	1
Virtual Channel Delay	0
Virtual Channel Buffers	8
Virtual Channel Allocator	iSLIP
Input Speedup	2
Output Speedup	1
Internal Speedup	1
Flit Size	32 bytes

Fig. 2 shows the normalized IPC of the GPU in executing general-purpose applications with multiple control-flow instructions, based on the assumption that the memory system is idealized. *MIMD* (Multiple Instruction, Multiple Data) [26] represents the ideal case architecture providing full-utilization of computational resources. As shown in the graph, *DWF* shows highest IPC among warp formation methods (*NRCV*, *PDOM*, *and DWF*).

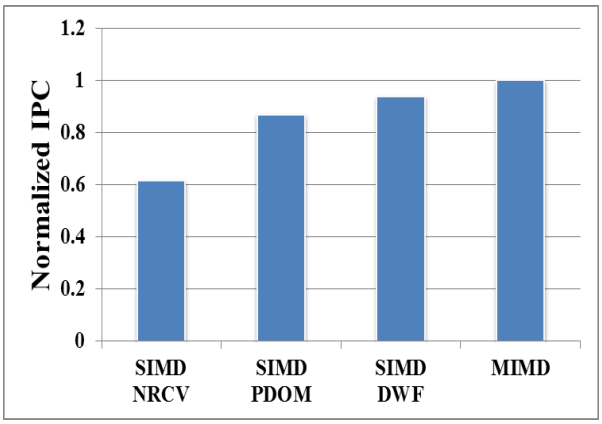

Fig. 2. Performance (with idealized memory) according to warp formation.

PDOM shows better IPC than *NRCV*. From the results shown in Fig. 2, we can know that *NRCV* incurs serious performance degradation of the GPU compared to the ideal case (*MIMD*) due to severe hardware underutilization whereas *DWF* and *PDOM* show less performance degradation. However, note that the results in the graph are obtained from the assumption that the memory system is idealized. As you know, memory resources in commercial GPU architecture are restricted. Therefore, our baseline GPU architecture in this work is modeled by QuadroFX5800 [20], because we focus on the performance of commercial GPUs with non-ideal memory system according to hardware configuration and warp formation.

Fig. 3 shows the normalized performance of different warp formations where memory resources are configured as shown in Table I. As you can see, the results in Fig. 3 are different to the results in Fig. 2. *PDOM* shows highest performance, while *NRCV* still incurs serious performance degradation of the GPU. Even though *DWF* and *MIMD* have the potential to utilize the computational hardware resources fully, they degrade the performance due to the delay caused by non-ideal memory system. Therefore, the correlation between hardware configuration and warp formation should be considered to improve the performance of SIMD architecture. For this reason, we analyze the performance of the GPU varying computational hardware resources, memory system configuration and warp formation.

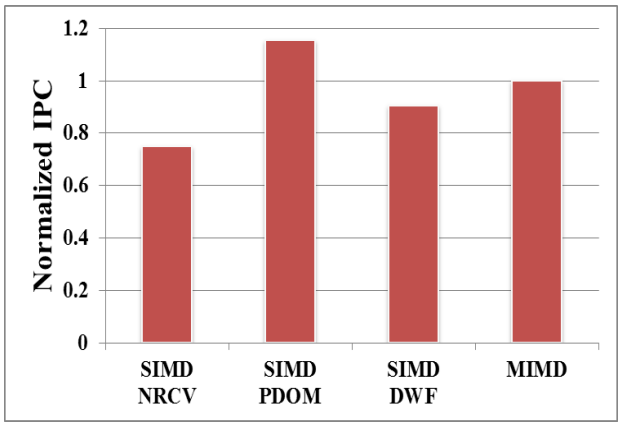

Fig. 3. Performance (in real GPU) according to warp formation.

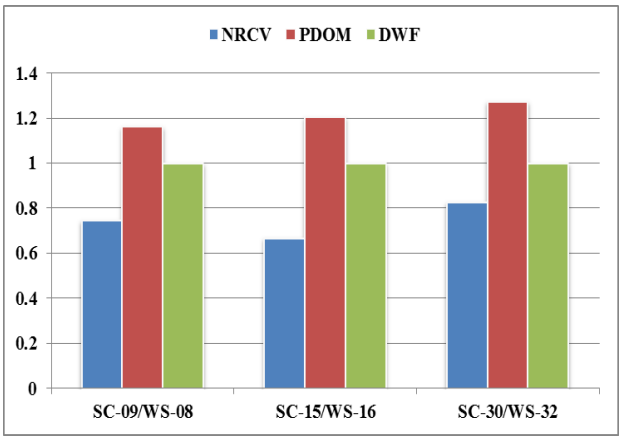

Fig. 4. Performance of warp formation according to computational hardware configuration.

Fig. 4 shows the performance of three branch handling methods according to various computational hardware configurations. In the graph, SC-Number and WS-Number

denote the number of shader cores and warp size, respectively. For an example, SC-09/WS-08 means that target GPU architecture has 9 shader cores and executes 8 threads in a warp on the shader core. Each bar in the graph is normalized to the IPC of *DWF*. As depicted in the graph, performance gap between *PDOM* and *DWF* is decreased by 9.6 %(SC-09/WS-08 compared to SC-30/WS-32) and 5.7 %(SC-15/WS-16 compared to SC-30/WS-32). The performance gap between *NRCV* and *DWF* is increased by 9.8 %(SC-09/WS-08 compared to SC-30/WS-32) and 19.3 %(SC-15/WS-16 compared to SC-30/WS-32). The performance gap between *PDOM/NRCV* and *DWF* is continuously decreased as computational resources are reduced. As the warp size is decreased, memory bottleneck is reduced; because the number of memory requests at the same time is reduced as the warp size is decreased, resulting in decreased memory conflicts. And, reduced computational resources have negative impact on the parallelism.

Fig. 5 shows the normalized IPC of three branch handling methods according to the memory resources. In the graph, CB-Number and MB-Number denote the number of cache bank and memory bandwidth, respectively. For an example, CB-04/MB-16 means that the number of cache bank is 4 and memory bandwidth is 16Bytes/controller. As shown in the graph, the performance gap between *PDOM* and *DWF* is decreased by 19.6 %(CB-04/MB-16 compared to CB-01/MB-04) and 17.2 %(CB-02/MB-08 compared to CB-01/MB-04). The performance gap between *NRCV* and *DWF* is increased by 26.9 %(CB-04/MB-16 compared to CB-01/MB-04) and by 20.6 %(CB-02/MB-08 compared to CB-01/MB-04). Experimental results show that the performance of *DWF* improves as the memory resources increase, because improved memory system alleviate the problem of warp formation due to high-utilization of computational resources. However, note that memory resources are expensive.

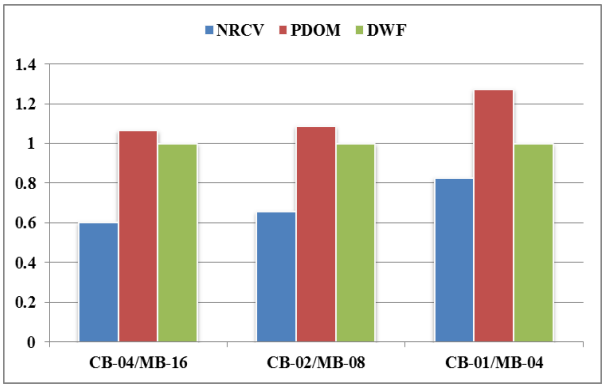

Fig. 5. Performance of warp formation according to memory system configuration.

IV. Conclusion

Hardware utilization of the GPU is limited by control-flow behavior when general-purpose applications are executed. In this work, we analyzed the efficiency of warp formations, which have been proposed to improve the hardware utilization. Contrary to previous expectations, our simulation results show that the performance of the warp formation providing better hardware utilization is lower than that of the warp formation providing worse hardware utilization due to limited hardware resources such as memory system. Severe memory bottleneck due to random memory accesses degrades the GPU performance even though computational resources are highly utilized by warp formation methods. To analyze the correlation between the performance and hardware configuration according to branch divergence methods, we presented the results with various computational resources and memory resources. According to our simulation results, increased memory resources are required to solve the problems of warp formation methods providing high-utilization of computational resources. Consequently, we can know that warp formation for high-utilization of computational resources cannot be practical without proper hardware support. Unfortunately, additional hardware resources require high costs. Therefore, we'll research on the new handling methods for branch divergence with limited memory resources for future work.

References

[1] V. Agarwal, M. S. Hrishikesh, S. W. Keckler, and D. Burger, "Clock rate versus IPC: the end of the road for conventional microArchitectures," in *Proc. 27th International Symposium on Computer Architecture, Vancouver*, 2000, pp. 248-259.

[2] N. P. Jouppi and D. W. Wall, "Available instruction-level parallelism for superscalar and superpipelined machines," in *Proc. 3th International Conference on Architectural Support for Programming Languages and Operating Systems*, Boston, 1989, pp. 272-282.

[3] D. M. Tullsen, S. J. Eggers, and H. M. Levy, "Simultaneous multithreading: maximizing on-chip parallelism," in *Proc. 22th International Symposium on Computer Architecture*, Seattle, 1995, pp. 392-403.

[4] I. Buck, T. Foley, D. Horn, J. Sugerman, K. Fatahalian, M. Houston, and P. Hanrahan, "Brook for GPUs: stream computing on graphics hardware," in *Proc. 31th Annual Conference on Computer Graphics*, Los Angeles, 2004, pp. 777-786.

[5] E. Lindholm, M. J. Kligard, and H. P. Moreton, "A user-programmable vertex engine," in *Proc. 28th Annual Conference on Computer Graphics (SIGGRAPH)*, Los Angeles, 2001, pp. 149–158.

[6] NVIDIA CUDA Programming Guide. [Online]. Available: http://www.developer.nvidia.com/object/cuda_3_1_downloads.html

[7] OpenCL. [Online]. Available: http://www.khronos.org/opencl/

[8] NVIDIA Cg Toolkit. [Online]. Available: https://www.developer.nvidia.com/cg-toolkit

[9] Microsoft HLSL. [Online]. Available: http://www.msdn2.microsoft.com/en-us/library/bb509638.aspx

[10] OpenGL Shading Language. [Online]. Available: http://www.opengl.org/registry/doc/GLSLangSpec.Full.1.20.8.pdf

[11] S. Che, J. Meng, J. Sheaffer, and K. Skadron, "A performance study of general purpose applications on graphics processors using CUDA," *Journal of Parallel and Distributed Computing*, vol. 68, no. 10, pp. 1370-1380, Oct. 2008.

[12] J. Helin, "Performance analysis of the CM-2, a massively parallel SIMD computer," in *Proc. 6th International Conference on Supercomputing*, Washington, 1992, pp. 45-52.

[13] A. Levinthal and T. Porter, "Chap - a SIMD graphics processor," in *Proc. 11th Annual Conference on Computer Graphics (SIGGRAPH)*, Minneapolis, 1984, pp. 77-82.

[14] R. A. Lorie and H. R. Strong, "Method for conditional branch execution in SIMD vector processors," US Patent 4435758, Mar. 6, 1984.

[15] S. Moy and E. Lindholm, "Method and system for programmable pipelined graphics processing with branching instructions," US Patent 6947047, Sep. 20, 2005.

[16] E. Rotenberg, Q. Jacobson, and J. E. Smith, "A study of control independence in superscalar processors," in *Proc. 5th International Symposium on High-Performance Computer Architecture, Orlando*, 1999, pp. 115-124.

[17] B. W. Coon and J. E. Lindholm, "System and method for managing divergent threads in SIMD architecture," US Patent 7353369, Apr. 1, 2008.

[18] E. Rotenberg, Q. Jacobson, and J. Smith, "A study of control independence in superscalar processors," in *Proc. 5th International Symposium on High-Performance Computer Architecture*, *Orlando*, 1999, pp. 115-124.
[19] W. W. L. Fung, I. Sham, G. Yuan, and T. M. Aamodt, "Dynamic Warp Formation and Scheduling for Efficient GPU Control Flow," in *Proc. 40th Microarchitecture*, Vancouver, 2007, pp. 407-420.
[20] QuadroFX5800. [Online]. Available: http://www.nvidia.com/object/product_quadro_fx_5800_us.html
[21] W. W. L. Fung, I. Singh, A. Brownsword, and T. M. Aamodt, "Hardware Transactional Memory for GPU Architectures," in *Proc. 44th Microarchitecture*, *Porto Alegre*, 2011, pp. 296-307.
[22] CUDA SDK. [Online]. Available: http://developer.download.nvidia.com/compute/cuda/sdk/website/samples.html
[23] A. Bakhoda, G. L. Yuan, W. W. L. Fung, H. Wong, and T. M. Aamodt, "Analyzing CUDA Workloads Using a Detailed GPU Simulator," in *Proc. 9th International Symposium on Performance Analysis of Systems and Software*, Boston, 2009, pp. 163-174.
[24] SimpleScalar Tool Set. [Online]. Available: http://www.simplescalar.com
[25] Booksim. [Online]. Available: http://www.nocs.stanford.edu/booksim.html
[26] M. J. Flynn, "Very high-speed computing systems," in *Proc.the IEEE*, vol. 54, no. 12, pp. 1901-1909, Dec. 1966.

Hong Jun Choi was born in 1985, in South Korea. He obtained Bachelors (2009) and Masters(2011) in School of Electronics and Computer Engineering, Chonnam National University, Gwangju, South Korea. He is a Ph.D. student in School of Electronics and Computer Engineering, Chonnam National University since September 2011. His research interests include computer architecture, energy/thermal-aware processor architecture, and parallel computing.

Dong Oh Son was born in 1986, in South Korea. He received the B.S. and M.S. degree in School of Electronics and Computer Engineering from Chonnam National University in 2010 and 2012, respectively. He is the Ph.D. student in School of Electronics and Computer Engineering, Chonnam National University since 2012. His research interests include embedded system, heterogeneous system, computer architecture.

Cheol Hong Kim was born in 1975, in South Korea. He received the B.S., M.S, and Ph.D. degrees in Computer Engineering from Seoul National University, Seoul, Korea, in 1998, 2000, and 2006, respectively. He worked as a senior engineer in Samsung Electronics, Korea from 2005 to 2007. In 2007, he joined the faculty of the School of Electronics and Computer Engineering from Chonnam National University, Gwangju, Korea. His research interests include high-performance multicore architecture, power-aware processor architecture, embedded systems design, and GPU architecture design.