International Journal of Control and Automation
Vol.6, No.5 (2013), pp.429-438
http://dx.doi.org/10.14257/ijca.2013.6.5.38

Synthesizable Manycore Processor Designs with FPGA in Teaching Computer Architecture

Hana Park, Young-Woong Ko, Jungmin So and Jeong-Gun Lee

*Dept. of Computer Science, Hallym University,
Chuncheon, South Korea*

{hnpark, yuko, jso1, jeonggun.lee}@hallym.ac.kr

Abstract

Manycore processor architectures are currently very widely used in almost all computer systems. Furthermore, many computer science department have started to teach parallel programming on manycore system platforms. To best understand the parallel programming on those manycore processors, the classes on manycore processor architecture also have to be provided. However, most current undergraduate computer architecture classes have not yet provided. In this paper, we have design a manycore processor design using a conventional MIPS processor design that is considered as a standard processor architecture model in a traditional undergraduate computer architecture class. Synthesize manycore processor models are developed in 850 lines of Verilog-HDL together with simple parallel applications so students can understand the entire manycore processor design and can learn the basics of the manycore system including underlying communication architecture.

Keywords: *Maycore Processor. Computer Architecture, Teaching*

1. Introduction

Engineering education at Universities are always suffering from the mismatches between rapidly advancing technology at the side of industry and slowly evolving course materials at the side of University. In the major of computer science and computer engineering, such mismatches seem to be more frequently observed than other science and technology department since the IT technologies have been advanced faster than others. To solve the IT education problems, IEEE *Computer Education Society* and other IT related society have worked on the education issues from few decade ago.

In a current commercial computing system, manycore processors are major products of many chip manufacturing vendors (*e.g.*, Intel, AMD, nVidia) [1, 2, 3, 4]. Quad or hexa-core processors are not only available for high-performance computing but also available for general computing in desktop PC computers. Moreover, those manycore systems are also used in mobile phones. However, in the undergraduate course work of computer science (CS) or computer engineering (CE), there is not yet enough classes for manycore system architecture and programming. In consequence, a manycore processor paradigm is one topic of such mismatches happen between academia and industry. Recently, there have been parallel programming classes in which the lectures on practical CUDA or OpenCL parallel programming languages are provided [5, 6]. However, it seems that there is no enough class on the manycore processor at the viewpoint of a computer architecture. Some advanced chapters of textbooks in computer architecture classes cover the theoretical points of modern high-performance manycore processor architectures, but they do not provide actual design or experimental platform for a manycore processor because its design is more complex compared with a single cycle processor, a multi-cycle processor and a pipelined processor [7, 8].

In this paper, we develop a synthesizable manycore processor design that can be used in undergraduate-level computer architecture classes. Our manycore processor design is described in Verilog-HDL and it is based on a well-known MIPS processor core [7,8]. In particular, our synthesizable manycore design is simplified as much as possible for giving students better understanding on manycore processor internals, while the communication architectures, such as "shared memory structure", are being described. The communication architecture are currently a core design issue in a camera usb modern manycore design. The manycore model has been synthesized successfully with the Altera Quartus II CAD tool and implemented and tested on a commercial field programmable gate array (FPGA) development platform called a DE2-70 FPGA board that is developed by the Altera Corporation.

In Section 2, we briefly address previous work on computer architecture education. Detailed issues on our synthesizable manycore processor architecture is described in Section 3. Verification and implementation issues are addressed in Section 4. Finally, Section 5 summarize this work with future work.

2. Previous Work

In the literatures of CS and CE educations, there have been many work so far with long history for supporting teaching materials and for establishing course contents that are efficiently deliverable to CS and CE majored university students with better understanding [9].

In [10], the authors have develop a computer-aided teaching package for teaching a processor architecture. The CAT is composed of an assembler and a graphics simulator. In the study, a simple Z80 model is used as a processor model. Similar to [10], a simple processor simulator has been developed as a teaching tool for first-year undergraduates [11]. As a program run on the simulator, the tool shows a snapshot of the processor internals such as register values and program counter. In [12], simple micro-architectures are used as a processor education model for first-year students of computer science with a graphic tool that visualize the operations of the micro-architectures. The authors in [13] have presented a graphical and interactive tools for reduced instruction set computer processor and memory simulator. Through those visualizing tools, undergraduate students can actively learn theoretical concepts covered in computer architecture classes. It is a unique feature of [13] that the simulator can be configured into processors of having many different levels of complexity from a simple processor without caches or pipelines to a highly complex one with caches and superscalar execution. The core idea behind [10, 11, 12, 13] is supporting visualization tools for better understanding of computer architecture operations. In addition to [10, 11, 12, 13], many of similar visualization approaches have been performed so far in the computer science education society.

Another research trend of the processor education in CS / CE is employing an FPGA devices. The programmability of the FPGA is a very fascinating feature in processor design educations because students can make real working processor chip by downloading their designs to the FPGA as opposed to just experimenting software-based simulations. Furthermore, unlike ASIC designs that require at least few months for their implementations, the FPGA device provide a quick verification/test cycle and easy modification capability. Consequently, students can experience all the process of designing a actual product by working on designing, implementing, testing, and debugging processors using a commercial FPGA development boards [14]. Many educational purpose FPGA development boards are available at the University Programs provided by main FPGA chip vendors such as Altera and

Xilinx [15, 16]. In [17, 18, 19, 20], an FPGA device has been used as their educational teaching platforms for demonstrating real working processor to students. In [21], the authors have discussed some usage of a FPGA device in teaching processor design class.

Most importantly, students can have much motivations toward their learning for computer architecture because developing a real working processor can be a unique experiences. Our goal of work is to provide the educational processor architecture model that follows the recent trend of microprocessor market so that students can feel much higher motivation for their learning.

3. A Manycore Processor Model for Teaching

A manycore processor architecture is now becoming the mainstream of a commercial processor architecture in the market. Consequently, it is strongly required for the University education to provide students theoretical and practical issues of the manycore processor since the trend of increasing the number of cores in a processor will continue further. In this section, we describe the details of the proposed manycore processor architecture for teaching.

2.1. Single-core MIPS model

In a traditional computer architecture education [8, 22, 23], A MIPS processor architecture is almost standard teaching model for a processor architecture. The course materials on the MIPS have been developed for undergraduate and graduate students with long history. From a single-cycle MIPS, multi-cycle MIPS, pipelined MIPS and a superscalar version of the MIPS have been developed for an educational purpose and those models have been used widely and frequently over the world of the computer architecture education society.

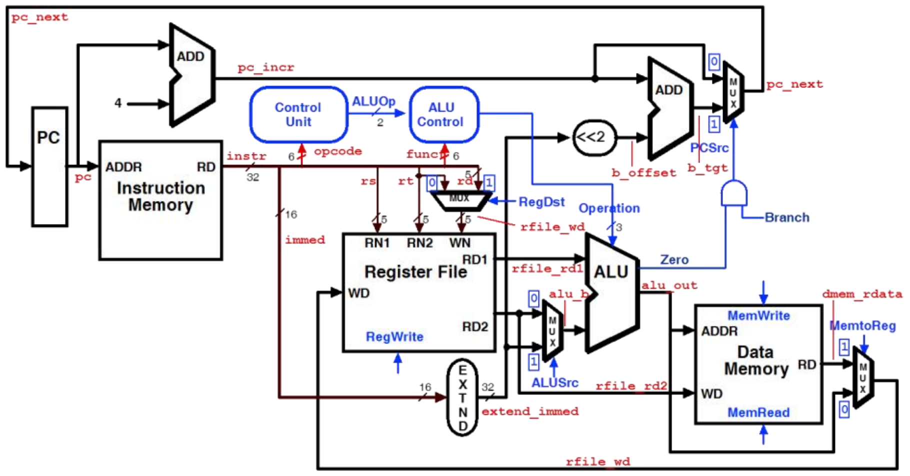


Figure 1. A traditional MIPS processor architecture that is implemented in a *single-cycle* micro-architecture [8, 22, 23]

The MIPS processor architecture follows the design principle of a RISC architecture and internal structures are simpler than other processor architecture such as x86 processor models. Register numbering is easy to understand and only small types of instruction formats (*i.e.*, R-type, I-type, J-type) are used for simplifying instruction decoding. Students can extend it by adding more instruction and such a modifications are straightforward with a Verilog-HDL based design.

We use the 32-bit MIPS processor as shown in Figure 1 that is employing only 11 instructions as shown in Table 1. This is a minimal instruction set for describing applications. Further extension is possible by modifying few of control logic and datapath. Those modification can be performed by adding few lines of a Verilog-HDL code to the original MIPS design source code.

Table 1. Supported instructions for our MIPS-based manycore processor

Instruction Class	Instructions
Arithmetic Logical Instructions	add, sub, and, or, slt, sll, srl
Memory Access Instructions	lw (load word), sw (store word)
Branch Instructions	beq
Processor Halt Instruction	halt

For branch control, only one instruction is supported so that other types of branch instructions can be emulated with the mixed sequence of two instructions: slt and beq.

It is noteworthy that our MIPS processors that is used as core in a manycore processor model are different from the standard single-core version. The differences are coming from the memory access behaviors. Our MIPS core can access two different types of data memories: *local memory* and *shared memory* while The conventional single-core MIPS has one data memory and accesses it. It leads to the change of control circuits and datapath for the memory accesses. The details on the selection circuits for data memories are described the following sections.

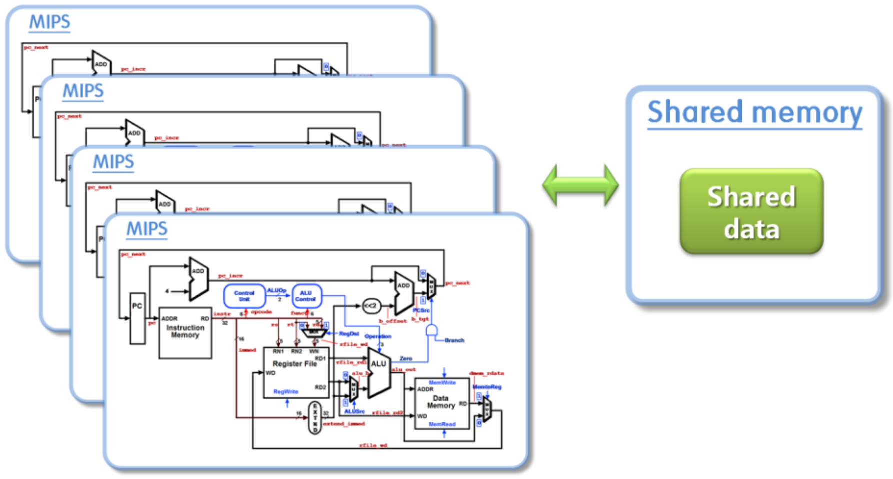
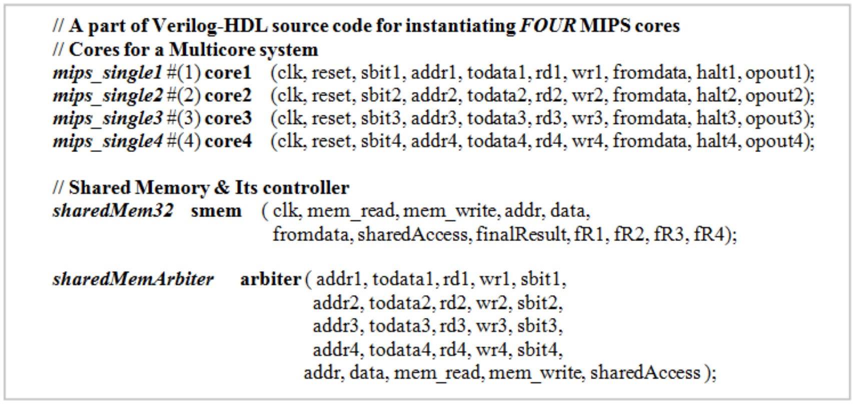


Figure 2. The proposed MIPS-based manycore processor: Four MIPS cores are integrated with a shared memory for communications

2.2. Manycore Extension with single-core MIPS

Figure 2 shows a quad-core processor configuration in which the multiple cores are communicating with each other by sharing a memory. Figure 3 show a part of Verilog-HDL description for instantiating four multiple MIPS cores, a shared memory and a shared memory controller. The modules, *mips_single1*, *mips_single2*, *mips_single3* and *mips_single4*, are all single-cycle MIPS processor modules, but they have differently coded instruction memories so that different codes are working on each cores.



```
// A part of Verilog-HDL source code for instantiating FOUR MIPS cores
// Cores for a Multicore system
mips_single1 #(1) core1   (clk, reset, sbit1, addr1, todata1, rd1, wr1, fromdata, halt1, opout1);
mips_single2 #(2) core2   (clk, reset, sbit2, addr2, todata2, rd2, wr2, fromdata, halt2, opout2);
mips_single3 #(3) core3   (clk, reset, sbit3, addr3, todata3, rd3, wr3, fromdata, halt3, opout3);
mips_single4 #(4) core4   (clk, reset, sbit4, addr4, todata4, rd4, wr4, fromdata, halt4, opout4);

// Shared Memory & Its controller
sharedMem32   smem    ( clk, mem_read, mem_write, addr, data,
                        fromdata, sharedAccess, finalResult, fR1, fR2, fR3, fR4);

sharedMemArbiter    arbiter ( addr1, todata1, rd1, wr1, sbit1,
                              addr2, todata2, rd2, wr2, sbit2,
                              addr3, todata3, rd3, wr3, sbit3,
                              addr4, todata4, rd4, wr4, sbit4,
                             addr, data, mem_read, mem_write, sharedAccess );
```

Figure 3. A part of Verilog-HDL description for instantiating four MIPS cores, a shared memory and a shared memory controller

The first and second arguments of a MIPS module are clk and reset, respectively. The third argument is used for identifying the shared memory request from a core. If a core needs a shared memory access then the argument is set to a HIGH value. The fourth, fifth, sixth and seventh arguments are used for denoting a shared memory address, write-data to the shared memory, read-control signal and write-control signal, respectively. The eighth, ninth, tenth arguments are used for denoting read-data from the shared memory, halt status of the core and instruction's opcode running at a current cycle, respectively.

2.3. Core-to-Core Communication

In a manycore processor architecture, it is important for students to understand the structure of providing communication mechanism among multiple cores. There are two main communication architectures for manycore processors: a *shared memory architecture* and a *message-passing architecture* [23]. More recently, incorporating a concept of a network router (*network-on-chip*, NoC) has been introduced as a communication structure for manycore processors [24]. In this paper, we adopt a shared memory architecture for the communication structure because it is easiest way to design and implement. However, even though only a shared memory model is provided, theoretical pros and cons on the different communication structures have to be addressed to students in a course.

Figure 4 shows a interconnection between two MIPS cores and one shared memory together with a shared memory controller that is arbitrating multiple requests from multiple cores. sharedMem1 and shareMem2 are signals for denoting shared memory requests from core 1 and core 2. The shared memory controller takes sharedMem1 and shareMem2 and it determines which core request a shared memory access.

Figure 5(a) shows a shared memory controller and its input and output signals. From each core, a shared memory request signal, address to access, read/write control signals and data bus signals are going to the controller. If a request is granted, then address, read/write control signals and data bus signals are connected to a shared memory.

In this paper, to make the architecture model more easy to understand, no concurrent requests is assumed, which means that all the cores are aware of a global schedule of shared memory accesses from multiple cores. Consequently, together with this lab experiments, more abundant theories on synchronization methods among manycore processors have to be provided in a lecture.

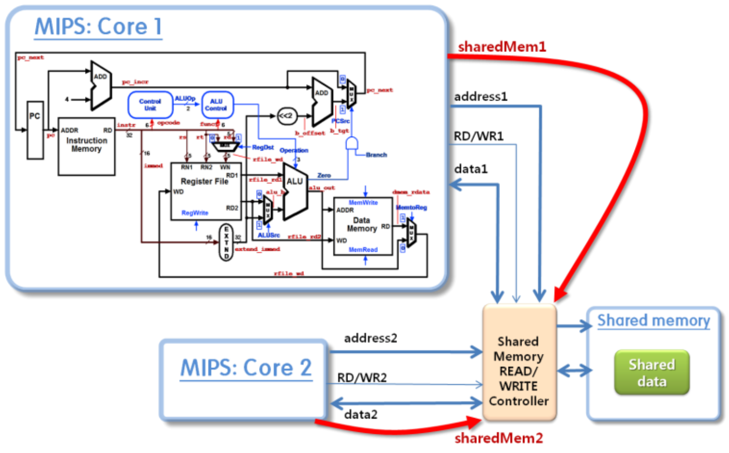
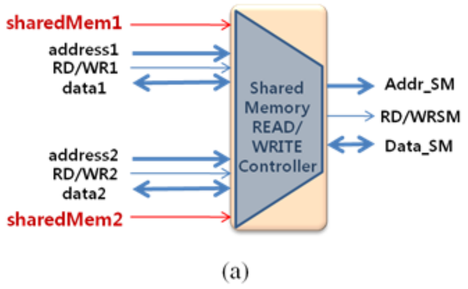
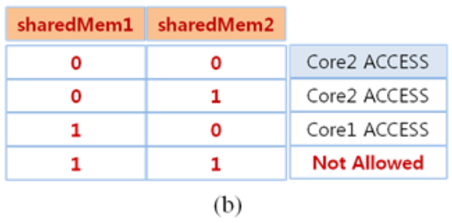


Figure 4. Shared memory accesses from multiple cores





sharedMem1	sharedMem2	
0	0	Core2 ACCESS
0	1	Core2 ACCESS
1	0	Core1 ACCESS
1	1	Not Allowed

(b)

Figure 5. (a) A block diagram of a shared memory controller with input and output signals; (b) A truth table denoting an arbitration logic for granting request signals from cores

Figure 5(b) shows a truth table denoting an arbitration circuit for granting request signals from cores. Even though we show the shared memory controller and truth table only for a dual-core processor. The extension to the higher number of cores is easily derived by the similar fashion.

Our manycore processor architecture has been designed with around 850 lines of Verilog-HDL so the student can easily understand entire processor architecture.

4. Verification and Implementation

We choose an application performing "1 to *n* summation" in parallel since it has high degree of parallelism and it is intuitively easy to partition the workload to multiple cores. The source codes that are running at a dual-core MIPS processor are developed only with 11 instructions that is available on our MIPS core. Figure 6 shows the verification of dual-core MIPS processor performing parallel summation from 1 to *n*. The verification has been performed with a ModelSim-Altera simulation tool.

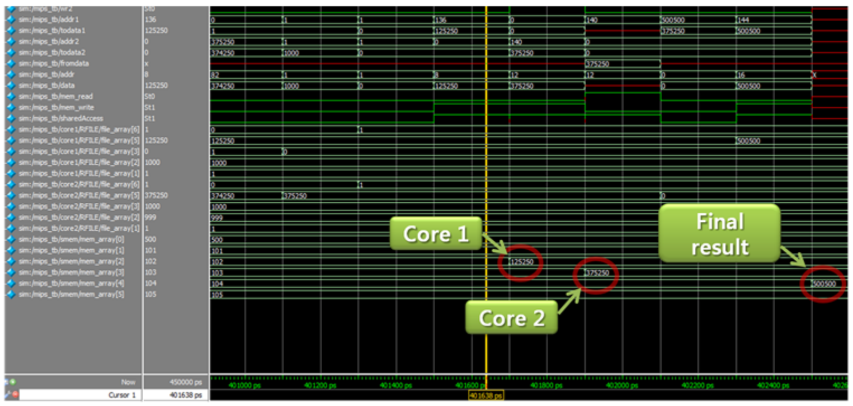


Figure 6. Shared memory accesses from multiple cores

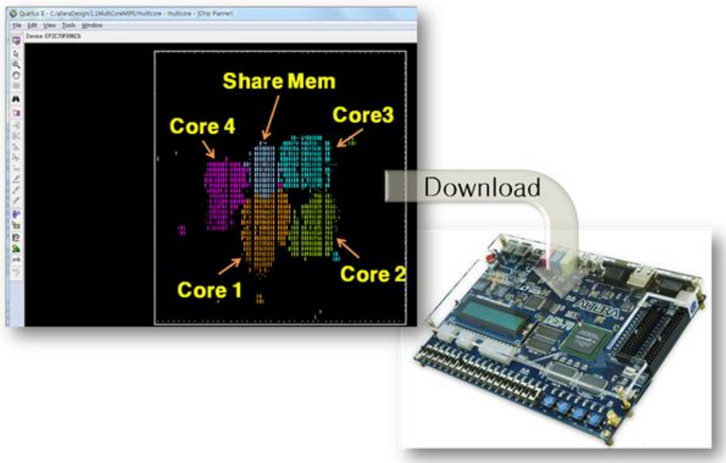


Figure 7. Manycore processor chip placement and its downloading to an FPGA

The workload is evenly partitioned into two cores: core 1 performs 1 to *n*/2 and while core 2 sum up from *n*/2+1 to n. The partial summation results are shown in the Figure 6. Finally, the final result is calculated by collecting the partial results from both cores through the shared memory.

In our course, Altera Quartus II tool and Altera Cyclone-II FPGA based DE2-70 development board has been used for synthesizing and implementing the HDL code, respectively [25, 26]. Finally, to show how the multiple cores are mapped onto the FPGA, chip placement view are presented in Figure 7 so that students can visually see the actual implementation of a manycore processor design.

5. Conclusion

Contemporary commercial manycore processor architectures are currently very widely used in almost all computer systems from supercomputers to mobile handheld devices. To educate engineers who can work with the prevailing computing devices, many computer science department have started to teach parallel programming on manycore system platforms. In order to have best understand the parallel programming on those manycore processors, the classes on manycore processor architecture also have to be provided together with the programming language course. However, most current undergraduate computer architecture classes have not yet provided. In this paper, we have design a manycore processor design using a conventional MIPS processor design that is considered as a standard processor architecture model in a traditional undergraduate computer architecture class. Synthesize manycore processor models are developed in Verilog-HDL together with simple parallel applications. We expect that the manycore processor architecture model is used as educational platform for teaching practical experiments of manycore processor designs at the undergraduate computer architecture classes.

Acknowledgements

This research was supported by Hallym University Research Fund, HRF-201209-025.

References

[1] Multi-core processor, http://en.wikipedia.org/wiki/Multi-core_processor.
[2] Intel MIC, http://en.wikipedia.org/wiki/Intel_MIC.
[3] Bulldozer (microarchitecture), http://en.wikipedia.org.
[4] Comparison of Nvidia graphics processing units, http://en.wikipedia.org/wiki/Comparison_of_Nvidia_graphics_processing_units.
[5] nVidia Corporation, CUDA C Programming Guide, http://docs.nvidia.com/cuda/cuda-c-programming-guide/.
[6] Khronos Group, OpenCL Overview, http://www.khronos.org/opencl/.
[7] D. A. Patterson and J. L. Hennessy, "Computer Organization and Design: The Hardware/Software Interface", Morgan Kaufmann, **(1994)**.
[8] D. Harris and S. Harris, "Digital Design and Computer Architecture", 2st Edition, Morgan Kaufmann, **(2012)**.
[9] IEEE Education Society, http://www.ewh.ieee.org/soc/es/esinfo.html.
[10] H. B. Diab and I. Demashkieh, "A computer-aided teaching package for microprocessor systems education", IEEE Transactions on Education, vol. 34, no. 2, **(1991)**.
[11] W. D. Henderson, "Animated models for teaching aspects of computer systems organization", IEEE Transactions on Education, vol. 37, no. 3, **(1994)**.
[12] J. R. Arias and D. F. Garcia, "Introducing computer architecture education in the first course of computer science career", IEEE Computer Society, **(1999)**, pp. 37-40.
[13] M. I. Garcia, S. Rodriguez, A. Perez and A. Garcia, "p88110: A Graphical Simulator for Computer Architecture and Organization Courses", IEEE Transaction on Education, vol. 52, no. 2, **(2009)**, pp. 248-256.
[14] Y. Zhu, T. Weng and C. Cheng, "Enhancing learning effectiveness in digital design courses through the use of programmable logic boards", IEEE TRAN. Education, vol. 52, no. 1, **(2009)** February, pp. 151-156.
[15] Altera University Program, http://www.altera.com/education/univ/unv-index.html.
[16] Xilinx University Program, http://www.xilinx.com/university/.
[17] H. Ochi, "ASAver.1: An FPGA-Based Education Board for Computer Architecture/System Design", Design Automation Conference, **(1997)**, pp. 157-165.
[18] I. Mazei and V. Malbasa, "Using VHDL to Improve an FPGA Based Educational Microcomputer", EUROCON, **(2005)**, pp. 799-802.

[19] R. de J. Romero-Troncoso, A. Ordaz-Moreno, J. A. Vite-Frias and A. Garcia-Perez, "8-bit CISC Microprocessor Core for Teaching Applications in the Digital Systems Laboratory", IEEE International Conference on Reconfigurable Computing FPGA's, **(2006)**, pp. 1-5.
[20] K. Nakano and Y. Ito, "Processor, Assembler and Compiler Design Education using an FPGA", 14th IEEE International Conference on Parallel and Distributed Systems, **(2008)**, pp. 723-728.
[21] R. D. Williams, R. H. Klenke and J. H. Aylor, "Teaching computer design using virtual prototyping", IEEE Trans. Education, vol. 46, no. 2, **(2003)**, pp. 296–301.
[22] D. A. Patterson and J. L. Hennessy, "Computer Organization and Design, The Hardware software Interface", Fourth Edition, Morgan Kaufmann Publishers, Inc., USA, **(2011)**.
[23] J. L. Hennessy and D. A. Patterson, "Computer Architecture: A Quantitative Approach", Fourth Edition, Morgan Kaufmann Publishers, Inc., USA, **(2006)**.
[24] W. J. Dally and B. Towles, "Principles and Practices of Interconnection Networks", Morgan Kaufmann, **(2004)**.
[25] Altera Coporation, "Altera DE2-70 Development and Education Board User Manual", http://www.altera.com/education/univ/unv-index.html, **(2009)**.
[26] Altera Coporation, "Altera Quartus II Introduction Using Verilog Designs: For Quartus II 12.0", http://www.altera.com/education/univ/unv-index.html, **(2012)**.

Authors

Hana Park

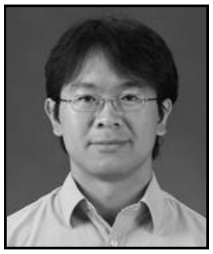
She received the B.S. degree in computer engineering from Hallym University in 2011. Currently she is pursuing a master's degree and teaching assistant for digital logic design and computer architecture classes in the same department.

Young-Woong Ko

He received both a M.S. and Ph.D. in computer science from Korea University, Seoul, Korea, in 1999 and 2003, respectively. He is now a professor in Department of Computer engineering, Hallym University, Korea. His research interests include operating system, embedded system and multimedia system..

Jungmin So

He received the B.S. degree in computer engineering from Seoul National University in 2001, and Ph.D. degree in Computer Science from University of Illinois at Urbana-Champaign in 2006. He is currently an assistant professor in Department of Computer Engineering, Hallym University. His research interests include wireless networking and mobile computing.



Jeong-Gun Lee

He received the B.S. degree in computer engineering from Hallym University in 1996, and M.S. and Ph.D degree from Gwangju Institute of Science and Technology (GIST), Korea, in 1998 and 2005. He is currently an assistant professor in the Computer Engineering department at Hallym university. Prior to joining the faculty of Hallym University in 2008, he was a postdoctoral researcher of the Computer Lab. at the University of Cambridge, UK.