Design of a DMA Controller for Loss-less Image Processing

Seong Mo Lee, Ji Hoon Jang, Sang Muk Lee, Jung Hwan Oh, and Seung Eun Lee[a]
Department of Electronic Engineering, Seoul National University of Science and Technology
E-mail : seung.lee@seoultech.ac.kr

***Abstract* – In this paper, we present a DMA (Direct Memory Access) controller to access data from the system memory on image processing system. The image data, which are processed in real time, have to be stored or read without loss of data in the memory. However, an embedded processor has a limit to directly process large amounts of the image data from camera sensor in real time. In order to resolve this limitation, DMAs are used in various systems.**

In this paper, we implement the DMA controller by using an FPGA and verify the functionality of the DMA controller in the system based on Avalon Memory Mapped interface. Also, the DMA controller is fabricated by using Magnachip/Hynix 0.35 um CMOS technology and operations of the fabricated DMA is verified.

I. INTRODUCTION

As electronic devices such as smartphone and assistant systems in automobile have been developed, performance of a camera sensor has been increased dramatically as well. This improvement leads to increase in size of image data received from the camera sensor. When a processor directly processes large amounts of the image data, performance of the system decreases due to overhead of processing. In order to solve this limitation, a DMA, which processes data of I/O peripheral instead of the processor, is widely used in image processing systems. Especially, various DMA controllers supporting PCI express have been developed [1][2]. In [1], a DMA controller supporting a Gen 2 4-lane PCIe link was proposed in order to solve a bottleneck between FPGA based processing board and the host computer. In [2], Zazo et al. presented a PCIe DMA engine which accelerates performance of virtual network appliance. The proposed PCIe DMA engine transfers data from the FPGA to the host computer at rates greater than 40Gbps. In both [1] and [2], proposed DMA controllers are connected to PCIe endpoint IP and specific hardware logic in target system. Therefore, these DMA controllers have a limitation in the system including a processor because bus interface such as AXI4 memory mapped interface, AHB (Advanced High-performance Bus), and Avalon interface is not supported. In addition, various DMA controllers are used to improve performance of system [3][4]. In [3], the DMA engine supporting 4K@30fps is proposed, focusing on improvement for DDR access efficiency. Main ideas are converting 2D 4×4(bytes) data into one single 16bytes and virtual alignment to optimize DDR access. In [4], Wang et al. presented a flexible DMA controller which supports block address-conversion and prefetching. However, the proposed DMA controllers in [3] and [4] focus on improving performance with memory access efficiency.

In this paper, we present a DMA controller for loss-less image processing. The proposed the DMA controller accesses image data from a memory in the system and displays them on the monitor without loss of data. Our DMA controller focuses on processing image data without any loss of them in accordance with video timing specifications.

The rest of the paper is organized as follows. In section 2, we introduce the architecture of the DMA controller in detail. Section 3 shows simulation results for verifying the functionality of the DMA controller. In section 4, we present FPGA prototyping and chip verification for the DMA controller in section 5, respectively. Finally, we conclude in section 6 by summarizing this study.

II. ARCHITECTURE OF THE DMA CONTOLLER

Our DMA controller conducts the operation for accessing the image data from a memory that is a SDRAM and displays the image data without any loss in accordance with VGA (Video Graphics Array) specifications. In case of a bus interface, the DMA controller supports Avalon MM interface (Avalon Memory Mapped interface), which is one of the Avalon interfaces, in order to easily connect components in an Altera FPGA [4]. The DMA controller is a master as well as a slave in Avalon MM interface for connecting between master and salve components. When the DMA controller is the slave in the interface fabric, the processor as the master controls operations of the DMA controller. When the DMA controller accesses the image data from the SDRAM, it is the master in the interface fabric. The DMA controller is composed of a DMA control module, a FIFO control module, and a VGA control module. Figure 1 illustrates the architecture of the DMA controller.

A. DMA Control Module

The DMA control module generates control signals for Avalon MM interface such as address, read, and a

a. Corresponding author; seung.lee@seoultech.ac.kr

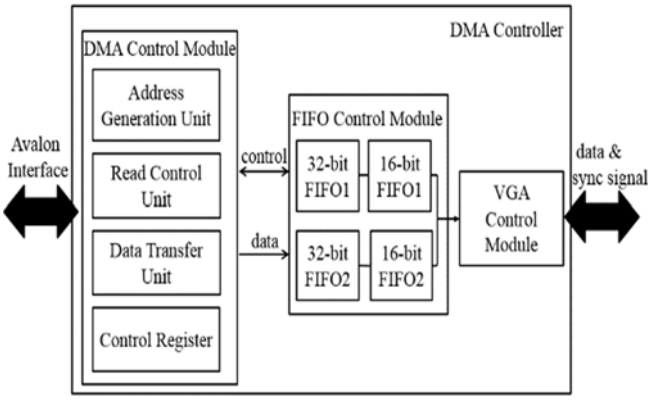

Fig. 1. The architecture of the DMA controller

wait_request signal. The DMA control module consists of an address generation unit, a read control unit, a data transfer unit, and a control register. The address, which is from master to slave, represents a byte address in Avalon interface specifications. When a master reads a word of data from a slave, an address increases from 0 to 4. The role of the address generation unit is to increase the address up to reading the image data from the SDRAM. The address generation unit also controls the address in accordance with an image data size and the number of frames configured in the control register. The read signal of Avalon MM interface is asserted to indicate a read transfer from the master to the slave. The read control unit allows the read signal to set high or low in accordance with the *wait_request* signal for arbitration between components on the interface fabric. Because the read signal is active high in our design, the read control unit asserts one to the interface fabric while read operation. The *wait_request* signal is used in order to identify whether or not the interface fabric is ready to be used by the master. When the slave is unable to respond to a read or write request, the *wait_request* signal is asserted by the slave. Therefore, the master has to wait until the interface fabric is ready to proceed with the read or write request. Along with the read signal, the *wait_request* is active high in our design. When the *wait_request* is high, the interface fabric is wait state. On the other hand, when the *wait_request* is low, the interface fabric is ready state to respond request by the master. The DMA control module has a *read_data_valid* signal, which is a 1-bit input, in order to receive the valid data from the SDRAM. When the *read_data_valid* signal is asserted by slave, read data signal contains valid data. Thus, when the *read_data_valid* signal is available, the data transfer unit transmits the image data to the FIFO control module. The data transfer unit also controls where to transfer the image data to one of the two 32-bit FIFOs. The control register includes information for start operation, the number of frame, and the burst count. The burst count indicates the number of transfers in each burst. In Avalon interface specifications, the value of the maximum burst count has to be a power of 2. The DMA control module supports burst count up to four in order to access the image data from the SDRAM. As a result, when the DMA control module requests a read operation at a time, it accesses 16 bytes of the image data in total from the SDRAM.

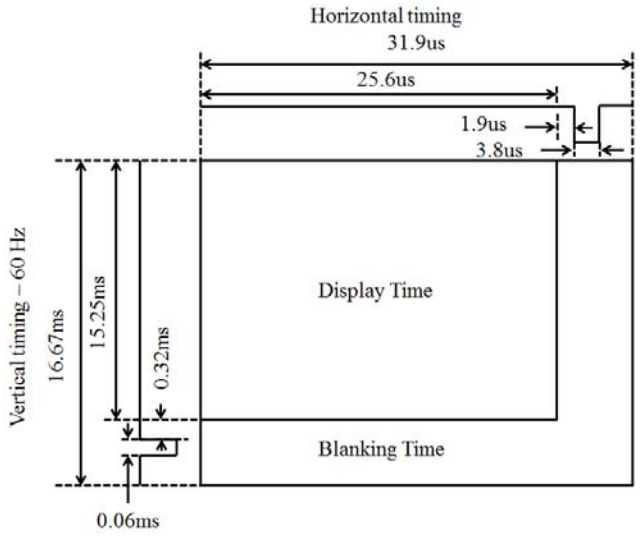

Fig. 2. The VGA timing specifications of a 60Hz refresh rate

B. FIFO Control Module

The FIFO control module has two 32-bit FIFOs and two 16-bit FIFOs. The 32-bit FIFO stores the image data from the DMA control module. Because Avalon MM interface fabric is 32-bit, these image data has 32-bit data width. The 16-bit FIFO is used to transfer the image data of the RGB565 color format from the 32-bit FIFO to the VGA control module. When the 16-bit FIFO receives the image data from the 32-bit FIFO, lower 16-bit is first stored to the 16-bit FIFO and then upper 16-bit is stored to the 16-bit FIFO. In our design, two FIFO sets are used in order to display the image data in real time. One FIFO set is composed of one 32-bit FIFO and one 16-bit FIFO. Also, the size of the 32-bit FIFO and the 16-bit FIFO are 2048 bytes.

C. VGA Control Module

The VGA control module displays the image data to the monitor in real time, supporting the RGB565 color format. The RGB565 color format represents 16bits per pixel. In this format, Red is 5bits, Green is 6bits, and Blue is 5bits respectively. The resolution of one frame is 640×480 and one row of the one frame is 1280 bytes in our design. When the resoultion of one frame is 640×480, the refresh rate is 60Hz and this is approximately 40*ns* per pixel. Therefore, the VGA control module displays one pixel at a time from the top left corner to the bottom right corner until it redraws the entire frame 60 times per second. In addition, the VGA control module generates active signals for synchronisation in order to display the image data in accordance with VGA standard. These active signals are horizontal and vertical sync signal. The horizontal sync signal indicates to refresh another row of 640 pixels. The vertical sync signal indicates to start displaying a new frame. Figure 2 shows the VGA timing specifications of a 60Hz refresh rate. The VGA control module generates the horizontal and the vertical signal and outputs the image data in accordance with VGA timing specifications.

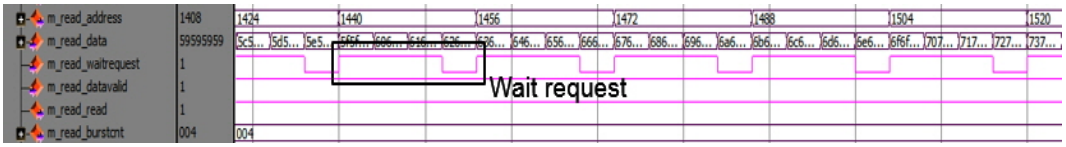

Fig. 3. Simulation result for the wait_request signal when burst count is four

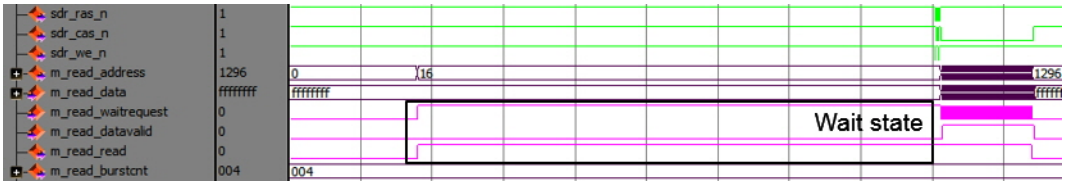

Fig. 4. Wait state during initialization of the SDRAM

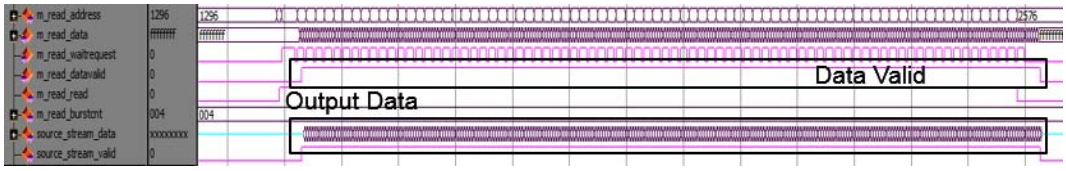

Fig. 5. Output of the image data from the DMA control module

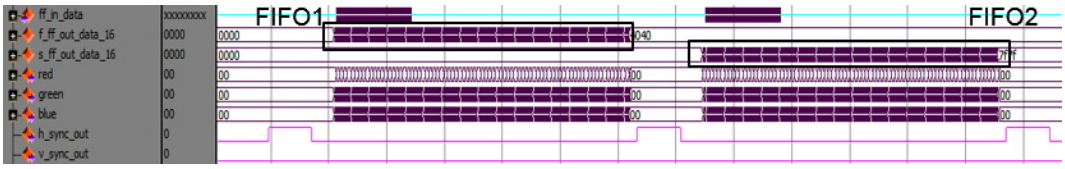

Fig. 6. The image data which is transmitted from 16-bit FIFOs to the VGA control module

III. SIMULATION RESULTS

We implemented the DMA controller in Verilog HDL and verified the functionality with simulation. First case is a result related in the *wait_requset* signal. The *wait_requset* signal is important signal to avoid conflict between the masters in Avalon MM interface. When the one master is using the interface fabric, the others have to wait until the interface fabric is available. In fact, the slave deasserts the *wait_request* to the interface fabric when the slave is able to respond a request of the master. Because the DMA control module is the master in the interface fabric, the *wait_request* signal is input signal to the DMA control module. In this simulation, burst count is four. Therefore, the DMA control module accesses 16 bytes from the SDRAM at a time while reading operations. Figure 3 shows a simulation result for the *wait_request* signal when burst count is set to four. When the *wait_request* signal is low, the DMA control module requests a read operation to the SDRAM controller and then the *wait_request* signal becomes high by the SDRAM controller. The waitrequest signal maintains high while reading 12 bytes at first, indicating a wait state that the SDRAM controller is unable to respond different requests. When the DMA control module accesses 4 bytes lastly, the *wait_request* signal becomes low by the SDRAM controller. At this time, the DMA control module is able to request another read operation to the SDRAM controller. Figure 4 shows wait state during initialization of the SDRAM. At the start of a read request, the SDRAM has to be initialized by the SDRAM controller for implementing power-up sequence and setting the mode register of the SDRAM. The SDRAM controller asserts the *wait_request* signal to the Avalon MM interface fabric during initialization of the SDRAM until initialization sequence finishes. After initialization of the SDRAM, the DMA control module accesses the image data up to 1280 bytes, incrementing the address. Figure 5 shows output of the image data from the DMA control module. The source stream data signal indicates the image data from the DMA control module to the FIFO control module. When the *read_data_valid* signal is available, the image data is transmitted to the FIFO control module. Figure 6 shows the image data which is transmitted from 16-bit FIFOs to the VGA control module. When the first horizontal synchronization signal is low active, the VGA control module accesses the image data from the first 16-bit FIFO and then displays them to the monitor. At this time, the second 16-bit FIFO receives the image data from the second 32-bit FIFO in advance. After the VGA control module displays the image data in accordance with the first horizontal synchronization signal, it reads the image data from the second 16-bit FIFO when next horizontal synchronization signal is available.

IV. FPGA PROTOTYPING

A DE2-115 development board is exploited to implement FPGA prototyping for the DMA controller [5]. Because the DE2-115 development board has the SDRAM and supports VGA port, it is suitable for implementing our design. In

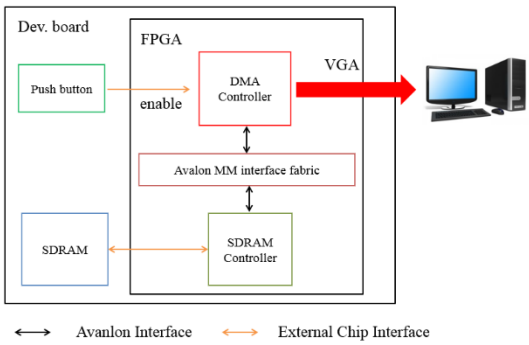

Fig. 7. A verification environment of FPGA prototyping

addition, we used Qsys tool in order to establish demonstration environment based on Avalon MM interface fabric [6]. The Qsys provides users with convenience by automatically generating interconnect logic to connect both IPs and custom logics based on Avalon interface specifications [7]. Also, we exploited the SDRAM controller which is generated by using the Qsys. Because the image data have to be stored to the SDRAM at the first time, we used the control panel to store the image data to the SDRAM on the DE2-115 development board. The control panel allows users to access various components such as, LEDs, an LCD, an SRAM, a Flash, SDRAMs, and a VGA on the DE2-115 development board from a host computer. We used Quartus II 13.1 for synthesizing Avalon MM interface system including the DMA controller. A procedure of the verification with FPGA prototyping is as follows:

- First step is to store the image data to the SDRAM on the DE2-115 development board by using the control panel.
- Second step is to download synthesized file, which includes Avalon MM interface system, to the DE2-115 development board.
- Final step is to enable the DMA controller to start operations by using push button.

We verified the functionality of the DMA controller through above steps [8]. Figure 7 illustrates a demonstration environment of FPGA prototyping.

V. CHIP IMPLEMENTATION AND RESULTS

The DMA controller was fabricated by using Magnachip/Hynix 0.35 *um* CMOS technology. A chip photograph of the STLSM, which is the fabricated DMA controller, is shown in figure 8. Core size of the STLSM is $3mm \times 2mm$ and operating frequency of the STLSM is 50MHz with 3.3V.

In order to verify the functionality of the STLSM, we designed a PCB including necessary components such as the SDRAM and an Altera FPGA. An Altera FPGA, which is Cyclone III, is used to implement subsystem based on Avalon MM interface. The FPGA and the STLSM are connected through Avalon MM interface signals. Figure 9 shows the PCB for verification of the STLSM. The

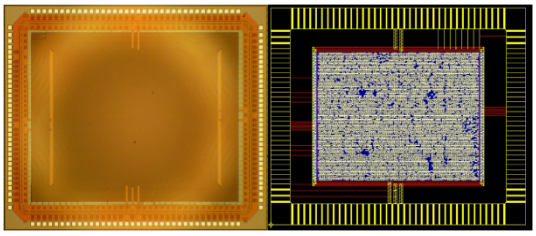

Fig. 8. A chip photograph of the STLSM

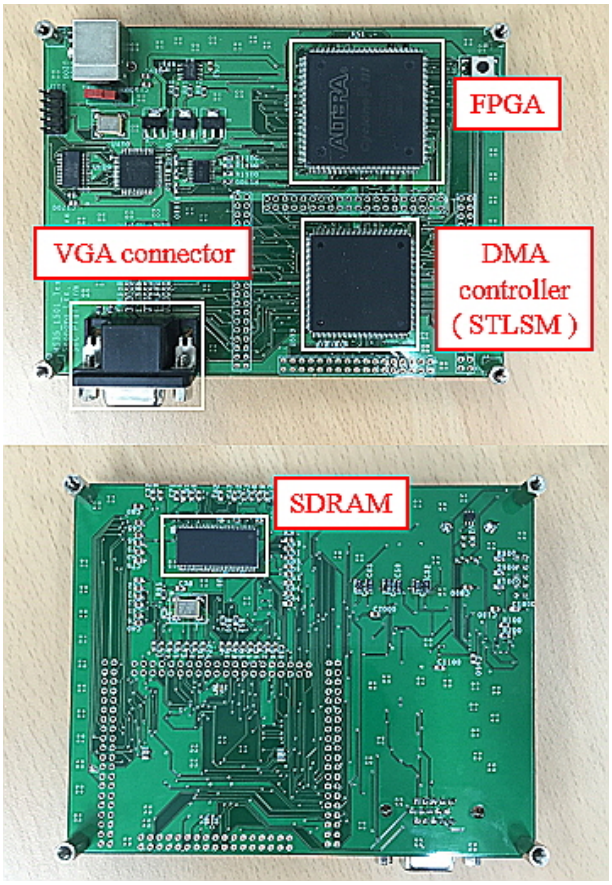

Fig. 9. The PCB for verification of the STLSM

subsystem which is implemented on the FPGA consists of JTAG-Avalon master bridge, Avalon MM interface fabric, and the SDRAM controller. The JTAG-Avalon master bridge is a master on Avalon MM interface fabric. Also, the JTAG-Avalon master bridge controls the DMA controller by configuring control register and enables the DMA controller to start operation. In addition, the JTAG-Avalon master bridge initializes data in the SDRAM with the image data by using the System Console. The System Console provides read and write access to the specific IPs generated by the Qsys [9]. Because Control Panel is dependent on the hardware, it is not exploited for verification using the PCB. Therefore, we performed initialization of the data in the SDRAM using the System Console. A procedure of verification with the PCB is as follows:

- First step is to generate the subsystem by using the Qsys and then download synthesized file to the FPGA on the PCB.

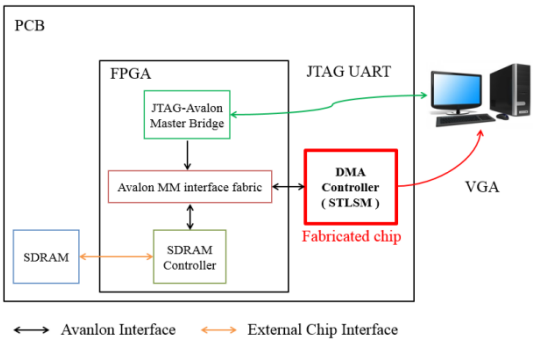

Fig. 10. A verification environment of the STLSM

• Second step is to initialize data of the SDRAM with the image data by using the System Console.

• In final step, we configure the control register of the fabricated DMA controller and enable it to start operation with the JTAG-Avalon master bridge by using the System Console.

As a result, we verified the operations of STLSM through above steps. Figure 10 illustrates a verification environment of the STLSM.

VI. CONCLUSIONS

In this paper, we presented the DMA controller for loss-less image processing. In order to process large amounts of the image data from camera sensor, the DMA is required to process these data instead of the processor. Proposed DMA controller accesses image data from the SDRAM and displays the image data without any loss in accordance with the RGB 565 color format. Simulation results showed appropriate operations in accordance with Avalon interface specifications and VGA timing specifications. In addition, we established a demonstration environment using FPGA development board in order to verify operations of the DMA controller. We also designed a chip of the DMA controller using Magnachip/Hynix 0.35 *um* CMOS technology and verified operations of the fabricated DMA controller. Implementation results show the feasibility of the DMA controller. We plan to adopt different bus interface instead of Avalon MM interface such as AMBA and AXI.

ACKNOWLEDGMENT

This work was supported in part by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government (MSIP) [NRF-2014R1A1A1004150]. This work was also supported by the IDEC (IC Design Education Center).

REFERENCES

[1] Hossein Kavianipour, and Christian Bohm, "High performance FPGA-based scatter/gather DMA interface for PCIe", *IEEE Nuclear Science Symposium and Medical Imaging Conference Record*, pp.1517-1520, Anaheim, CA, Oct. 27-Nov. 3 2012.

[2] J. F Zazo, S. Lopez-Buedo, Y. Audzevich, and A. W. Moore, "A PCIe DMA engine to support the virtualization of 40Gbps FPGA-accelerated network appliances", *2015 International Conference on ReConFigurable Computing and FPGAs*, pp.1-6, Mexico City, Mexico, Dec 2015.

[3] Niraj Nandan, "High performance DMA controller for ultra HDTV video codecs", *IEEE International Conference on Consumer Electronics*, pp.65-66, Las Vegas, NV, Jan 2013.

[4] Yinhui Wang, Teng Wang, Pan Zhou, and Xin′an Wang, "Design and Implementation of a Flexible DMA Controller in Video Codec System", *IEEE International Conference on Digital Signal Processing*, pp.78-82, Hong Kong, Aug 2014.

[5] "DE2-115 board User Manual", Terasic and Altera Corporation. [Online]. Available: http://www.terasic.com.tw/cgi-bin/page/archive_download.pl?Language=English&No=502&FID=cd9c7c1feaa2467c58c9aa4cc02131af.

[6] "Introduction to the Altera Qsys System Integration Tool", Altera Corporation. [Online]. Available: ftp://ftp.altera.com/up/pub/Altera_Material/12.0/Tutorials/Introduction_to_the_Altera_Qsys_Tool.pdf.

[7] "Avalon Interface Specifications", Altera Corporation. [Online]. Available: https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/manual/mnl_avalon_spec.pdf.

[8] Seong Mo Lee, Seon Young Lee, Kyung Won Min, and Seung Eun Lee, "Design of DMA controller for image data processing on image recognition system", *IEIE SoC Conference*, pp.99-100, Seoul, Korea, May 2014.

[9] "System Console User Guide", Altera Corporation. [Online]. Available: https://www.altera.co.jp/ja_JP/pdfs/literature/ug/ug_system_console.pdf.

Seong Mo Lee received the B.S. and M.S. degree in Electronic Engineering from the Seoul National University of Science and Technology, Seoul, Korea, in 2014 and 2016, respectively. His research interests include computer architecture, System-on-Chip, and fault tolerant system design.

Ji Hoon Jang received the B.S. and M.S. degree in Electronic Engineering from the Seoul National University of Science and Technology, Seoul, Korea, in 2014 and 2016, respectively. His research interests include computer architecture, System-on-Chip, Network-on-Chip, and hardware accelerator design.

Sang Muk Lee received the B.S. degree in Electronic Engineering from the Seoul National University of Science and Technology, Seoul, Korea, in 2014. He is M.S. student in the Department of Electronic Engineering at the Seoul National University of Science and Technology, Seoul, Korea. His research interests include computer architecture, System-on-Chip, and hardware multi-core scheduler design.

Jung Hwan Oh is B.S. student in the Department of Electronic Engineering at the Seoul National University of Science and Technology, Seoul, Korea. His research interests include computer architecture and System-on-Chip Design.

Seung Eun Lee received the Ph.D. degree in electrical and computer engineering from the University of California, Irvine (UC Irvine) in 2008 and the B.S. and M.S. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon in 1998 and 2000, respectively. After graduating, he had been with Intel Labs., Hillsboro, OR, where he worked as Platform Architect. In 2010, he joined the faculty of the Seoul National University of Science and Technology, Seoul. His current research interests include computer architecture, multi-processor system-on-chip, low-power and resilient VLSI, and hardware acceleration for emerging applications.