Hardware Platforms for Flash Memory/NVRAM Software Development

Eyee Hyun Nam

School of Computer Science and Engineering, Seoul National University, Seoul, Korea
ehnam@archi.snu.ac.kr

Ki Seok Choi

H Project Team, SK Energy, Daejeon, Korea
kiseok.choi@skenergy.com

Jin-yong Choi, Hang Jun Min, and Sang Lyul Min

School of Computer Science and Engineering, Seoul National University, Seoul, Korea
{jychoi, hjmin, symin}@archi.snu.ac.kr

Received 12 September 2009; Accepted 23 September 2009

Flash memory is increasingly being used in a wide range of storage applications because of its low power consumption, low access latency, small form factor, and high shock resistance. However, the current platforms for flash memory software development do not meet the ever-increasing requirements of flash memory applications. This paper presents three different hardware platforms for flash memory/NVRAM (non-volatile RAM) software development that overcome the limitations of the current platforms. The three platforms target different types of host system and provide various features that facilitate the development and verification of flash memory/NVRAM software. In this paper, we also demonstrate the usefulness of the three platforms by implementing three different types of storage system (one for each platform) based on them.

Categories and Subject Descriptors: Embedded systems [Systems & Architecture]

General Terms: Design, Experimentation

Additional Key Words and Phrases: Flash memory, Storage system, Platform

1. INTRODUCTION

Flash memory is being widely used as a storage medium in mobile devices because

Copyright(c)2009 by The Korean Institute of Information Scientists and Engineers (KIISE). Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Permission to post author-prepared versions of the work on author's personal web pages or on the noncommercial servers of their employer is granted without fee provided that the KIISE citation and notice of the copyright are included. Copyrights for components of this work owned by authors other than KIISE must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires an explicit prior permission and/or a fee. Request permission to republish from: JCSE Editorial Office, KIISE. FAX +82 2 521 1352 or email office@kiise.org. The Office must receive a signed hard copy of the Copyright form.

of its low power consumption, small form factor, and high resistance to shock and vibration [Douglass et al. 1994; Lawton 2006]. As the density of a flash memory chip increases and the price continues to drop, the flash memory is being adopted in more diverse storage applications. For example, flash memory solid-state disks (SSDs) that provide the same interface as hard disk drives (HDDs) are replacing HDDs in mobile and general-purpose computers.

Flash memory has characteristics that are different from conventional storage devices such as HDDs. Thus, specialized hardware and software are required to use flash memory as a storage device. The role of flash memory software is particularly important because it has to deal with the peculiarities of flash memory. It also needs to be diverse because flash memory is used in a wide spectrum of applications ranging from micro-embedded systems (e.g., a sensor-node) to large-scale servers.

However, the current development platforms for flash memory software are limited to simulators or embedded flash memory controllers, which are not sufficient to meet the performance and verification requirements of recent flash memory applications. Furthermore, they do not have provisions for the next generation non-volatile RAM (NVRAM) such as ferroelectric RAM (FRAM) [Sheikholeslami and Gulak 2000], magnetoresistive RAM (MRAM) [Tehrani et al. 1999], and phase-change RAM (PRAM) [Ha et al. 2003].

This paper describes three flash memory software development platforms that overcome the limitations of the current platforms. The three platforms provide development environment for different classes of flash memory application. The first platform (referred to as the legacy platform) is targeted for legacy embedded systems that do not have an embedded flash memory controller. The second platform (referred to as the ASIC platform) is for developing flash memory software and hardware that are to be integrated into an ASIC chip. Finally, the last platform (referred to as the SSD platform) is specialized for developing flash memory SSDs.

The rest of this paper is organized as follows. The next section explains the basics of flash memory and NVRAM, and also flash memory software. We then present the organization of the three development platforms and their applications in Section 3. Finally, Section 4 gives conclusions.

2. FLASH MEMORY, NVRAM, AND FLASH MEMORY SOFTWARE

2.1 Flash Memory and NVRAM

Flash memory is a non-volatile memory that can be electrically erased and reprogrammed. There are two major types of flash memory depending on the structure of the flash memory cell: NOR flash memory and NAND flash memory. NOR flash memory has an SRAM-like interface and provides fast random reads (<20 ns), which make it suitable for code storage of embedded systems. Compared to NOR flash, NAND flash memory has higher density, and provides a faster write speed and lower power consumption. Therefore it is more suitable for bulk data storage. With a scale-down of cell size and the introduction of a higher density cell structure called MLC (Multi-Level Cell), the density of NAND flash memory has doubled every year for the past ten years [Hwang 2003] and now 32Gbit chip is available. NAND flash memory also allows a limited number of bad blocks, which significantly increases the yield.

These features make NAND flash memory more widely used as data storage than NOR flash memory.

The internal organization and interface of NAND flash memory are different from conventional memories. A NAND flash memory chip is organized into physical blocks, each containing a set of pages that are accessed by read and program operations [Samsung Electronics]. There is also asymmetry in the read and program speeds - the read operation is much faster than the program operation (20 ms vs. 200 ms). Unlike conventional memories, NAND flash memory does not allow in-place updating of data. Instead, writing is performed by a page-program operation, which must be preceded by an erase-block operation that takes about 2 ms and sets all the bits in the target block to 1. The number of P/E (program/erase) cycles allowed for each block is limited to 10,000 for MLC and 100,000 for SLC (Single-Level Cell). NAND flash memory is also subject to bit-flipping errors, causing one or more bits in a block to be reversed, which necessitates the use of external error correction logic.

One variation of NAND flash memory is OneNAND [Kim *et al.* 2004] that combines the SRAM interface whose read/write accesses take less than 10 ns with the high density of NAND flash memory. The SRAM interface of OneNAND is provided by the internal SRAM (<16 KB) that is used for data buffering and boot-up purposes. Since it provides an efficient booting support and obviates the need for a specialized NAND flash memory interface, OneNAND is increasingly being used in mobile embedded devices such as cellular phones. Recently, there also has been active research on the next generation NVRAMs such as FRAM, MRAM, and PRAM, all of which provide the SRAM interface and low-latency read/write accesses. They also allow in-place updating and the maximum number of P/E cycles allowed for those memories is much larger than that of flash memory.

2.2 Flash Memory Software

As the cell size of NAND flash memory shrinks down and the MLC technique is more widely used, new restrictions are imposed in using NAND flash memory. For example, the number of partial programmings (referred to as NOP) allowed for a page before erasure is limited to one. Moreover, the pages within a block must be programmed in an ascending order in recent NAND flash memory chips. Despite these new restrictions, NAND flash memory is increasingly being adopted as the storage medium for a wide spectrum of applications because of its advantages in density, power consumption, and access latency. Flash memory applications usually need specialized software support whose examples include the flash translation layer (FTL) [Intel 1998], flash file systems [Ban 1995; Woodhouse 2001; Aleph One 2002], and flash-aware virtual memory systems.

The FTL is a software layer between flash memory and the rest of the operating system and it emulates the functionality of an HDD while hiding the peculiarities of flash memory. To emulate the in-place update semantics of the HDD, the FTL maintains a mapping between the logical sector address used by the host system and the physical flash memory address. With this re-mapping, legacy system software modules such as file systems and virtual memory systems can run on top of the FTL without any modification. This approach is simple but misses the opportunities to

optimize the performance and reliability of the system software modules by exploiting the unique characteristics of underlying flash memory.

There are however approaches where system software modules are re-implemented assuming flash memory as the underlying storage device. For example, the so-called flash file systems, such as JFFS [Woodhouse 2001] and YAFFS [Aleph One 2002], do not use the FTL and directly access the flash memory so as to allow optimizations by exploiting the unique characteristics of the flash memory. Likewise, the so-called flash-aware virtual memory systems exploit the characteristics of flash memory in providing demand paging using flash memory [Jo et al. 2006; Park et al. 2006].

As flash memory applications become more diverse, flash memory software developers need more powerful development platforms. The next section explains the problems of the current development platforms in dealing with the ever-increasing demands of flash memory software developers. We then discuss the requirements of flash memory software development platforms and present the three platforms that we designed and implemented with the requirements in mind.

3. FLASH MEMORY DEVELOPMENT PLATFORMS

Previously, development platforms for flash memory software were either based on simulators or embedded flash memory controllers hard-wired in commercial micro-controllers. When simulators are used to develop flash memory software, it is difficult to assess the accurate timing and the correct functioning of the software on actual flash memory chips. On the other hand, when hard-wired flash memory controllers are used, it is difficult to meet the ever-changing restrictions of NAND flash memory. Also, the current development platforms do not support software development for the next-generation NVRAMs such as FRAM, MRAM, and PRAM that are becoming increasingly important. From the discussions above, the requirements of flash memory development platforms can be summarized as follows:

1. They should support diverse interface styles to accommodate as many host systems as possible.
2. They should be flexible enough to allow the development of diverse flash memory controllers.
3. They should support flash memories as well as diverse next-generation NVRAMs.
4. They should allow accurate measurements of performance and power consumption.
5. They should provide additional resources such as DRAM and SRAM to provide more flexibility in designing flash memory HW/SW.

To meet the first requirement, the three platforms presented in this paper provide three different host system interfaces: (1) SRAM interface (in the legacy platform), (2) the AMBA system bus interface (in the ASIC platform), and (3) an HDD-compatible interface (in the SSD platform). To meet the second requirement, all the three platforms have an FPGA (Field Programmable Logic Array) that allows the construction of any custom-made flash memory controllers to address the increasing restrictions of NAND flash memory as well as to facilitate the exploitation of a trade-off between the cost and performance in the flash memory controller design. Hardware description languages, such as Verilog and VHDL, can be used to implement the flash memory

Figure 1. NVRAM Modules.

controller in the FPGA. To meet the third requirement, each platform has standard DIMM (Dual In-line Memory Module) sockets that can accommodate not only flash memory modules but also diverse next-generation NVRAM modules. To meet the fourth requirement, each platform has an interface to a DAQ (Data Acquisition) system [National Instrument] that provides accurate measurements of performance and power consumption of the developed system. To meet the final requirement, the platforms include various types of volatile memory that give more flexibility to the designers of the flash memory HW/SW. The next subsection explains NVRAM modules that are used by all the three platforms in common. Sections 3.2-4 describe the details of the three platforms along with their applications. Finally, we explain the data acquisition system in Section 3.5.

3.1 NVRAM Modules

Figure 1 shows various NVRAM modules that are currently available for the development platforms. They include modules for NOR flash memory, SLC/MLC NAND flash memory, OneNAND, FRAM, MRAM, and PRAM. The NVRAM modules are connected to a development platform through a standard DIMM socket interface. We augmented the original DIMM socket interface to define the signal and power lines between the NVRAM modules and the DIMM socket. Any NVRAM module that adheres to this newly defined interface can be used in all the three development platforms. This abstraction decouples the design of NVRAM module from the rest of the development platform and thus facilitates the accommodation of yet-to-come non-volatile memories.

3.2 Legacy Platform

Mobile embedded systems, such as cellular phones and MP3 players, have one or more micro-controllers that execute application programs and operating system code. These micro-controllers contain processor cores and auxiliary components such as memory controllers, interrupt controllers, and timer circuits, but many of them do not include a flash memory controller. The legacy platform targets the micro-controllers

Figure 2. Legacy Platform.

without an embedded flash memory controller and provides an external flash memory controller using the SRAM interface that is available in almost all micro-controllers.

The host system for the legacy platform is EDB9315A [Cirrus Logic 2006] shown in Figure 2(a). The EDB9315A board has an EP9315 micro-controller based on the ARM920T processor core, a 16MB NOR flash memory for code storage, and a 64MB SDRAM for data storage. As a typical single-chip micro-controller, EP9315 also has various I/O interfaces such as Ethernet, USB, SPI, UART, I2C, and IDE for communicating with external devices. As shown in Figure 2(b), the EDB9315A board is connected to the legacy platform using the two extension connectors in Figure 2(a). Those connectors provide pins not only for the SRAM interface but also for interrupts and DMA. The SRAM interface is used to access the command/status registers of the external flash memory controller implemented by the FPGA in the platform while the pins for interrupts and DMA are for high-speed data transfers between the flash memory controller and the micro-controller. With this organization, the flash memory

Figure 3. The Development of a Flash Memory Storage System Using the Legacy Platform.

software runs on the micro-controller in EDB9315A that accesses NAND flash memory modules through the flash memory controller, both located in the legacy platform.

The FPGA used in the legacy platform is Sparatan3-XC3S500 [Xilinx Sparatan-3] containing 75 K logic cells that are sufficient for implementing any type of flash memory controller. The logic cells in the FPGA can also be used to implement soft processor cores such as MicroBlaze [Xilinx MicroBlaze], which allows the legacy platform to be used as a stand-alone embedded system. The DIMM sockets, also shown in Figure 2(b), provide connection to the NVRAM modules. The DAQ connector is for data acquisition purposes (cf. Section 3.5) and allows accurate measurements of performance and power consumption of various components in the legacy platform.. The legacy platform also includes a 64MB SDRAM that can be used as high-speed data buffers between the flash memory controller and the micro-controller.

We used the legacy platform to implement a flash memory storage system based on an in-house FTL, as shown in Figure 3. The FTL runs as a kernel module of the Linux operating system ported to EBD9315A and provides a block device interface so that legacy file systems, such as ext2 and vfat, can be used without any modification. The device driver below the FTL communicates with the flash memory controller in the FPGA of the legacy platform for commands, status checks, and data transfers. Some glue logic implemented in the FPGA handles discrepancies between EP9315-specific signals and the signals used in the FPGA. The flash memory controller in the FPGA is divided into NAND-type dependent and independent parts, enhancing the portability.

3.3 ASIC Platform

As applications increasingly require higher performance and lower power consumption, the demand for micro-controllers that embed a flash memory controller in a single ASIC chip is increasing as well. Unlike FPGA, an ASIC chip does not provide reconfigurability and its manufacturing process takes a long turn-around time. Therefore, the ASIC design requires much more efforts in the verification. The second platform, called the ASIC platform, targets for such environment and uses as the host system the PB926EJ-S board that is considered as the de facto environment for developing

Figure 4. ASIC Platform.

Figure 5. Development of an NVRAM file System Using the ASIC Platform.

an ASIC chip based on the ARM926EJ-S processor core.

The PB926EJ-S board, shown in Figure 4, has an ARM926EJ-S processor, a 128MB SDRAM, a 2MB SRAM, and a 64MB NOR flash memory. It also has interfaces to SD, MMC, Ethernet, PCI, and USB that are intended to provide a wide range of options in implementing a micro-controller as an ASIC chip. The PB926EJ-S board is connected to the ASIC platform through three extension connectors (not shown in the figure) that collectively expose the AMBA (Advanced Microcontroller Bus Architecture) interface, which is a widely adopted on-chip system bus. As in the legacy platform, the ASIC platform is used to implement a flash memory controller. However, since the flash memory controller in this ASIC platform is to be integrated into an ASIC micro-controller, we use a logically on-chip (albeit physically off-chip) bus interface to PB926EJ-S.

The FPGA in the ASIC platform, shown in Figure 4, is Virtex 4 XC4VLX25 [Xilinx Virtex-4] containing 24 K logic cells, which are sufficient for implementing not only the flash memory controller but also the slave interface logic for the AMBA. The two DIMM sockets and a 64MB SDRAM are provided as in the legacy platform for the same purpose. There is also a 32-pin debug port that can be used as the interface to the data acquisition system.

Since the implementation of the FTL using a flash memory controller has been demonstrated by the legacy platform, we used the ASIC platform to implement a file system based on the next-generation NVRAMs, as shown in Figure 5. The AMBA slave interface logic and the FRAM/MRAM/PRAM controllers are implemented in the FPGA in the ASIC platform. The FRAM/MRAM/PRAM controllers map the non-volatile memories into the system memory space so that regular load/store instructions from ARM926EJ-S can operate on them. A file system for NVRAM called PRAMFS [Sourceforge.net] uses this interface to manipulate non-volatile data structures in the file system.

3.4 SSD Platform

As the price of NAND flash memory continues to drop, the flash memory SSD is increasingly being adopted in laptop computers and UMPCs. On the enterprise side, power-hungry HDDs are being replaced by more energy-efficient flash memory SSDs in high-end servers. As compared to embedded flash memory storage solutions, flash memory SSDs should provide not only a higher performance but also a larger capacity. These requirements and other practical considerations lead to the following features that an SSD development platform should have:

1. Multiple processor cores for a multi-threaded FTL.
2. Larger FPGA to implement an intelligent flash memory controller with parallel data-paths.
3. HDD-compatible host interfaces such as Parallel ATA (P-ATA) and Serial ATA (S-ATA).
4. Multi-channel flash memory interface for parallelism and extensibility.
5. Larger SRAM and DRAM for extensive caching and write-buffering.
6. Interface to a higher bandwidth data acquisition system.

We designed and implemented an SSD platform that provides all of the features above. Figure 6 shows various components are incorporated into our SSD platform. First, the Virtex4 XC4VFX100 FPGA [Xilinx Virtex-4] contains two PowerPC405 processors and 96K logic cells that should be sufficient to implement an intelligent flash memory controller, parallel data-paths, and multi-channel flash memory interface. The SSD platform also provides HDD-compatible interfaces such as P-ATA and S-ATA through connectors as shown in Figure 6. The device-side ATA protocol

Figure 6. SSD Platform.

Figure 7. Development of an SSD Using the SSD Platform.

can be implemented in the FPGA. The Multi-channel flash memory interface is connected to the four DIMM sockets in Figure 6. Each socket supports two channels, each of which can have up to eight chips. This multi-channel/multi-chip structure allows us to exploit parallelism in multiple flash memory chips for higher performance and also for a larger capacity. There is also one DIMM socket reserved for an FRAM/MRAM/PRAM module. This provision is made to exploit a hybrid SSD that combines flash memory with the next-generation NVRAMs [Yoon et al. 2008]. The SSD platform also has an 8MB synchronous SRAM, a 64MB SDR SDRAM, and a 64MB DDR SDRAM to enable SSD designers to explore a wider range of caching and write-buffering options. Finally, the SSD platform has two DAQ connectors to allow a larger amount of performance and power consumption data to be collected and also an Ethernet interface to enable a future extension as a development platform for network-based storage devices.

We implemented a flash memory SSD using the SSD platform as shown in Figure 7. The intelligent flash memory controller in the SSD controller extensively automates the management work of the FTL and effectively utilizes the parallelism of multiple flash memory chips. Fast prototyping with FPGA reduced the design turn-around time and facilitated the system tuning (in terms of performance and power consumption) and debugging. The SSD platform was seamlessly integrated into the hardware/software development environment provided by Xilinx, which further improved the productivity of the SSD development. More details about this prototyping of a flash memory SSD using the SSD platform can be found in [Seong et al. 2009].

3.5 Data Acquisition System

The data acquisition system is comprised of a development platform whose performance and power consumption is to be measured, a DAQ device that collects data in real-time, and a host computer that stores the data received from the DAQ device and performs a post-mortem analysis. Figure 8 shows the connections between these three components within the data acquisition system. The DAQ connector is used as a link

Figure 8. Data Acquisition System.

Figure 9. Data Acquisition and Analysis Example.

between the DAQ device and the development platform and the PCI bus as a link between the DAQ device and the host computer. In-house Labview programs run on the host computer to control the DAQ device and also to analyze the collected data.

There are two types of measured data: digital and analog. The former is mainly used for measuring the performance while the latter for measuring the power consumption. The beginning and the end of the data collection can be controlled either by a Labview program or by signals from the development platform.

Figure 9(a) shows a Labview program sampling the digital and analog data through their respective ports in the DAQ device. The collected data is later analyzed by another Labview program to assess the performance and the power consumption of the measured system. Figure 9(b) shows such a Labview program that calculates the energy consumption for each sector read or written by the FTL based on the collected data by the DAQ device.

4. CONCLUSION

Flash memory has become a viable storage solution not only for mobile embedded applications but also for general computing applications. To overcome the inherent limitations of flash memory while providing fast and reliable flash memory storage solutions, developers need powerful and flexible development platforms where they can experiment on various design options and pursue architectural innovations. In this paper, we presented three different types of development platform (the legacy platform, the ASIC platform, and the SSD platform), each targeted for a different application scenario. They all provide an FPGA to facilitate the implementation of a wide range of flash memory controllers; a DIMM socket interface to accommodate a wide range of NVRAMs including not only SLC/MLC flash memories but also various types of the next-generation NVRAM; a DAQ interface to allow accurate measurements of the performance and the power consumption of the developed solution; and additional resources such as DRAM and SRAM to provide more options to the designers of flash memory HW/SW. This paper also demonstrated the usefulness of the development platforms by implementing several experimental flash/NVRAM storage systems.

ACKNOWLEDGEMENTS

This work was partly supported by the IT R&D program of MKE/ITA [2006-S-040-03, Development of Flash Memory-based Embedded Multimedia Software]. An earlier version of this paper was published in the Korean language in the Communications of the KIISE, Vol. 27, No. 5, pages 38-46.

REFERENCES

- BAN, A. 1995. Flash file system, US Patent, no. 5,404,485.
- Cirrus Logic. 2006. EDB9315A engineering development board, http://www.cirrus.com/en/pubs/manual/EDB9315A_Tech_Ref_Manual.pdf.
- DOUGLIS, F., R. CACERES, F. KAASHOEK, K. LI, B. MARSH AND J. TAUBER. 1994. Storage alternatives for mobile computers. In *Proceedings of the 1st Symposium on Operating Systems Design and Implementation (OSDI)*:25–37.
- HA, Y., J. YI, H. HORII, J. PARK, S. JOO, S. PARK, U. CHUNG AND J. MOON. 2003. An edge contact type cell for phase change RAM featuring very low power consumption. *2003 Symposium on VLSI Technology*: 175–176.
- HWANG, C. 2003. Nanotechnology enables a new memory growth model. In *Proceedings of the IEEE* 91(11):1765–1771.
- Intel. 1998. Understanding the flash translation layer (FTL) specification. <http://developer.intel.com>.
- JO, H., J. KANG, S. PARK, J. KIM AND J. LEE. 2006. FAB: flash-aware buffer management policy for portable media players. *IEEE Transactions on Consumer Electronics* 52(2):485–493.
- KIM, B., S. CHO AND Y. CHOI. 2004. OneNAND (TM): a high performance and low power memory solution for code and data storage. In *Proceedings of the 20th Non-Volatile Semiconductor Memory Workshop*.
- LAWTON, G. 2006. Improved flash memory grows in popularity. *Computer* 39(1):16–18.
- National Instruments. What is data acquisition. <http://www.ni.com/dataacquisition/whatis.htm>.
- Aleph One. 2002. YAFFS: yet another flash file system. <http://www.yaffs.net>.
- PARK, S., D. JUNG, J. KANG, J. KIM AND J. LEE. 2006. CFLRU: a replacement algorithm for flash memory. In *Proceedings of the 2006 international conference on compilers, architecture and synthesis for embedded systems*. Seoul, Korea: 234–241.
- Sourceforge.net. PRAMFS: protected and persistent RAM file system. <http://pramfs.sourceforge.net>.

- Samsung Electronics. NAND flash memory data sheets. <http://www.samsung.com>.
- SEONG, Y., E. NAM, J. YOON, H. KIM, J. CHOI, S. LEE, Y. BAE, J. LEE, Y. CHO AND S. MIN. 2009. Hydra: a block-mapped parallel flash memory solid-state disk architecture. Accepted for publication in the *IEEE Transactions on Computers*.
- SHEIKHOLESAMI, A. AND P. GULAK. 2000. A survey of circuit innovations in ferroelectric random-access memories. In *Proceedings of the IEEE* 88(5):667–689.
- TEHRANI, S., J. SLAUGHTER, E. CHEN, M. DURLAM, J. SHI AND M. DEHERREN. 1999. Progress and outlook for MRAM technology. *IEEE Transactions on Magnetics* 35(5):2814–2819.
- WOODHOUSE, D. 2001. JFFS: the journalling flash file system. In *Proceedings of Ottawa Linux Symposium*.
- Xilinx. MicroBlaze processor reference guide. http://www.xilinx.com/support/documentation/sw_manuals/mb_ref_guide.pdf.
- Xilinx. Virtex-4 family overview. http://direct.xilinx.com/support/documentation/data_sheets/ds112.pdf.
- Xilinx. Spartan-3 FPGA family data sheet. http://direct.xilinx.com/support/documentation/data_sheets/ds099.pdf.
- YOON, J., E. NAM, Y. SEONG, H. KIM, B. KIM, S. MIN AND Y. CHO. 2008. Chameleon: a high performance flash/FRAM hybrid solid state disk architecture. *IEEE Computer Architecture Letters* 7(1):17–20

Eyee Hyun Nam received his BS degree in electrical engineering from Seoul National University in 1998. He worked at Comtec System from 1999 to 2002 and Future Systems from 2003 to 2004. He is currently a PhD candidate in computer science and engineering at Seoul National University. His research interests include computer architecture, operating systems, embedded systems and flash memory based storage systems.

Ki Seok Choi received his BS and MS degrees in computer science and engineering both from Seoul National University, in 2006 and 2008 respectively. He is currently working as a researcher at SK energy. His research interests include computer architecture, embedded systems, and flash memory storage systems.

Jin-Yong Choi received his BS degree in electrical engineering from Seoul National University in 2005. He is currently a PhD candidate in computer science and engineering at Seoul National University. His research interests include flash memory based storage systems, embedded systems, computer architecture, and host interface systems.

Hang Jun Min received his BS degree in computer science and engineering from Sung Kyun Kwan University in 2009. He is currently a master student in computer science and engineering at Seoul National University. His research interests include computer architecture, operating systems, embedded systems, and flash memory storage systems.

Sang Lyul Min received his BS and MS degrees in computer engineering, both from Seoul National University, Seoul, Korea, in 1983 and 1985, respectively. In 1985, he was awarded a Fulbright scholarship to pursue further graduate studies at the University of Washington. He received a PhD degree in computer science from the University of Washington, Seattle, in 1989. He is currently a professor in the School of Computer Science and Engineering, Seoul National University, Seoul, Korea. Previously, he was an assistant professor in the Department of Computer Engineering, Pusan National University, Pusan, Korea, from 1989 to 1992 and a visiting scientist at the IBM T.J. Watson Research Center, Yorktown Heights, New York, from 1989 to 1990. He has served on a number of program committees of technical conferences and workshops, including the International Conference on Embedded Software (EMSOFT), the Real-Time Systems Symposium (RTSS), and the Real-Time Technology and Applications Symposium (RTAS). He was also a member of the editorial board of the IEEE Transactions on Computers. Currently, he is serving as the editor-in-chief of the Journal of KIISE: Computer Systems and Theory. His research interests include embedded systems, computer architecture, real-time computing, and parallel processing. Dr. Min is a life-time member of the KIISE.