LETTER

Solid-State Disk with Double Data Rate DRAM Interface for High-Performance PCs

Dong KIM[†], **Kwanhu BANG**[†], **Seung-Hwan HA**[†], **Chanik PARK**[††], **Sung Woo CHUNG**[†††], ***Nonmembers***, ***and*** **Eui-Young CHUNG**[†a)], ***Member***

SUMMARY We propose a Solid-State Disk (SSD) with a Double Data Rate (DDR) DRAM interface for high-performance PCs. Traditional SSDs simply inherit the interface protocol of Hard Disk Drives (HDD) such as Parallel Advanced Technology Attachment (PATA) or Serial-ATA (SATA) for maintaining the compatibility. However, SSD itself provides much higher performance than HDD, hence the interface also needs to be enhanced. Unlike the traditional SSDs, the proposed SSD with DDR DRAM interface is placed in the North Bridge which provides two or more DDR DRAM interface ports in high-performance PCs. The novelty of our work is on DQS signaling scheme which allows arbitrary Column Address Strobe (CAS) latency unlike typical DDR DRAM interface scheme. The experimental results show that the proposed SSD maximally outperforms the traditional SSD by 8.7 times in read mode, by 1.5 times in write mode. Also, for synthetic workloads, the proposed scheme shows performance improvement over the conventional architecture by a factor of 1.6 times.

key words: *SSD (Solid-State Disk), NAND flash, North Bridge, DRAM interface*

1. Introduction

SSD is a massive storage device based on NAND Flash memories. It is gradually replacing mechanical HDDs thanks to its non-volatile, low-power, silent, and shock-resistant features in mobile and desktop PCs. Compared to HDDs, the major disadvantage of SSD is its cost. However, its cost is rapidly decreasing, since NAND Flash memory technology is continuously evolving. Over the past decade, the density of NAND Flash memory has doubled on every 12 month according to Hwang's law, hence SSD will lead the massive storage market in the near future. Even though modern SSDs outperform HDDs, many research works continue to pursue further increases in their performance from the architectural perspective. Lee et al. proposed a new NAND Flash memory package with smart buffer cache for exploiting the spatial and temporal localities [1]. Park et al. proposed an energy aware demand paging using Clean First LRU algorithm which minimizes the number of write or erase operations [2]. Kang et al. proposed a multi-channel architecture using I/O parallelism such as striping, interleaving and pipelining [3]. The authors in [4] introduced a flash memory controller having a dedicated datapath between the host interface and NAND Flash memory interface. Also, the authors in [5] enhanced data throughput using multi-channel and way interleaving schemes. However, all of these works focused on the internal architecture of SSD. On the contrary, we focus on the interface scheme of SSD, since it would be a performance bottleneck according to Amdahl's law when the internal architecture of SSD is continuously improved. Table 1 shows a summary of high-performance interface schemes which are recognized as future SSD interface candidates in industry. Among those interface schemes, DDR DRAM interface delivers the highest bandwidth in the same generation. For this reason, we investigate an SSD with DDR DRAM interface for high-performance PCs in this work. The interface is well tailored to an SSD for arbitrary length of burst data transfer with arbitrary CAS latency which are not necessary for conventional DDR DRAMs. Such an enhanced SSD greatly improves the PC performance when it transfers massive data to and from the SSD.

2. Internal Architecture of SSD

A typical SSD architecture is shown in Fig. 1. It consists of a

Table 1 Storage interfaces (Unit: MB/s).

Interface	Organization	2001	2005	2007	2009~	Ref.
SATA	SATA-IO	150	300	-	600	[6]
DDR DRAM (x64)	JEDEC	3200	8500	16000	25600	[7]
PCI-E (x16)	PCI SIG	4000	-	8000	16000	[8]
USB	USB-IF	1.5	60	-	600	[9]
SAS	SCSITA	150	300	600	1200	[10]
Fibre Channel (B2)	FCIA	400	800	1600	3200	[11]

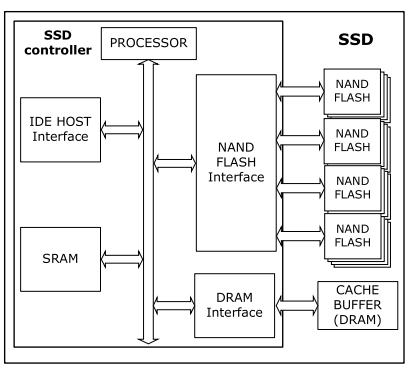

Fig. 1 SSD internal architecture.

Manuscript received July 15, 2008.
Manuscript revised November 28, 2008.
[†]The authors are with Yonsei University, Seoul, Korea.
[††]The author is with Samsung Electronics Co. Ltd., Kyeonggi, Korea.
[†††]The author is with Korea University, Seoul, Korea.
a) E-mail: eychung@yonsei.ac.kr
DOI: 10.1587/transinf.E92.D.727

processor, NAND Flash interface, NAND Flash memories, SRAM, DRAM interface, DRAM and a host interface. A processor controls the overall system and a firmware called Flash Translation Layer (FTL) is running on it for wear-leveling. The NAND Flash interface can support multiple channels and ways to increase read/write throughput. The DRAM is utilized as a cache buffer in order to achieve a higher performance. In other words, the data read time from the SSD critically depends on whether the cache hit occurs or not. Finally, the host interface is a communication path between the PC and the SSD.

3. Conventional PC Architecture with SSD

3.1 Overview

Figure 2 (a) shows the block diagram of a conventional 80x86 PC architecture. High-speed peripherals including DRAMs are connected to North Bridge, while slow peripherals including SSD are connected to South Bridge. Data transfer to (from) an SSD is initiated by the CPU and the source (destination) of data is DRAM located in North Bridge. Read and write operations for an SSD consists of several sub-operations defined in Table 2. The details of read and write operations are described in the following sections.

3.2 Read Operation

Read operation is defined as the data transfer from an SSD to the main memory (DRAM). For read, the CPU issues a DMA read command to the SATA controller in South Bridge (Operation A). Then the DMA in SATA controller receives the first packet including data ID from the SSD through the SSD interface (Operation C and D). The SATA controller analyzes the first packet to fetch the data transfer size and the starting address from the Physical Region Descriptor (PRD) table in the main memory (Operation B). According to the information, the DMA in SATA controller transfers data appropriately from the SSD to main memory (Operation E). The operation sequence 'C→D→B→E' is repeated until when the requested size of data is completely transferred. At the end of the DMA transfer, the SSD signals an interrupt and then read operation is completed. Note that the DMA read operation is performed in a pipelined fashion. For instance, when there are two consecutive read operations, the second read operation is started right after the data requested by the first read operation is delivered to South Bridge. Hence the delay due to the bridges is hidden by the next read operation.

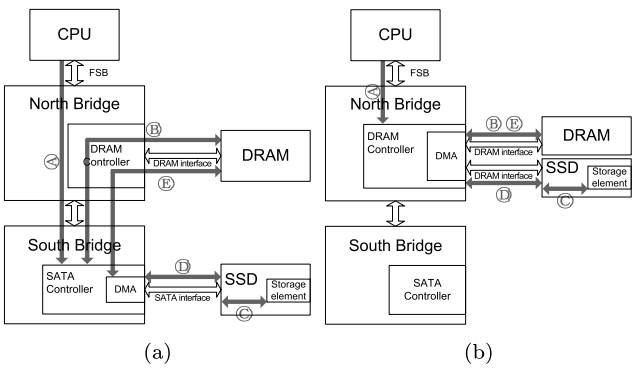

Fig. 2 PC architecture with an SSD: (a) Conventional architecture. (b) Enhanced architecture.

Table 2 DMA transfer time.

	Operation	Description
A	DMA command	CPU issues a DMA command
B	PRD fetch	Read the physical memory region
C	SSD internal	SSD internal read/write transfer
D	SSD interface	SSD interface read/write transfer
E	Memory access	Memory access to or from SSD

3.3 Write Operation

Write operation is also initiated by a DMA command issued from CPU (Operation A). Then the SATA controller fetches the starting address and the size of the data to be transferred from the PRD table in main memory (Operation B). With the information, the DMA in SATA controller fetches the data from the main memory (Operation E) and then transfers data appropriately from the main memory to the SSD in the order of Operation D and C. The operation sequence 'B→E→D→C' is repeated until when the requested size of data is completely transferred. At the end of the DMA transfer, the SSD signals an interrupt and then write operation is completed. Unlike read operation, the write operation is performed in a sequential manner in our design. When there are two consecutive write operations, the second write operation is started only after the first write operation completely writes the data to the NAND Flash memories in the SSD. Thus, the delay due to bridges is transparently reflected to the write performance. Note that the sequential operation in write mode is for protecting write failure. The SSD may lose the first data when it is written to a bad block, if the write operation is performed in a pipelined fashion.

3.4 Performance Limiting Factors

A traditional SSD replaces a HDD while keeping the same interface protocol for compatibility. Apparently, this architecture is not suitable to provide the expected performance when massive data is written to or read from the SSD for two reasons. First, every access to SSD has to pass through two bridges (North and South Bridges) meaning that the request for SSD needs to be arbitrated twice. Second, the maximum bandwidth of PATA (SATA2) is 133 (300) MB/s, hence the interface may become a bottleneck due to its insufficient bandwidth. The enhanced PC architecture in the next section resolves such issues for higher data throughput.

4. Enhanced PC Architecture with SSD

4.1 Overview

The block diagram of the enhanced PC architecture is shown in Fig. 2 (b). The SSD is connected to North Bridge through the DDR DRAM interface. The major advantages of this architecture are two folds. First, DDR DRAM interface provides higher bandwidth than SATA interface. Second, South Bridge is eliminated in a data path between main memory and SSD, hence the arbitration overhead is reduced. By eliminating the South Bridge, the following changes are required. The SATA controller is no longer necessary, while the DRAM controller is extended for supporting the proposed SSD. CPU issues a DMA command to the DRAM controller in North Bridge, PRD fetching and the data transfer between DRAM and SSD are also performed by DMA in DRAM controller. In case of read operation, transfer operation sequence is 'A→C→D→B→E'. And the operation sequence 'C→D→B→E' is repeated to complete data transfer. In write operation, the operation sequence is 'A→B→E→D→C'. The operation sequence 'B→E→D→C' is repeated as in read operation.

4.2 DDR DRAM Interface Tailored to SSD

The major challenge in this architecture is to tune the typical DDR DRAM interface to SSD from two aspects. First, typical DDR DRAMs have a fixed CAS latency meaning that the data is available on the data pins after a fixed number of clock cycles in read mode. On the contrary, the data response time of SSD is not a constant, since it has a cache buffer which greatly reduces the response time when the cache hit occurs. Second, it is necessary to convert the size of data to be transferred in the unit of burst length which is the basic unit of data transfer in DRAM.

To tackle the CAS latency issue, we introduce a DQS signaling scheme. DQS is a feedback clock defined in DDR DRAM standard protocol for the easy synchronization of data from the storage. In our scheme, DQS is asserted by SSD to inform the host machine (PC) of the data availability. Figure 3 shows how DQS signaling scheme controls the variability of data response time. Figures 3 (a) and (b) show DQS behavior when the cache hit occurs and when the cache miss occurs, respectively. More precisely, the assertion of DQS is delayed until when the data is fetched from the NAND Flash memories if the cache miss occurs.

As far as data size conversion is concerned, the DMA in DRAM controller imitates the DMA in the SATA controller of the conventional architecture. In conventional architecture, the DMA divides the size of the data to be transferred by the sector size yielding the quotient representing the number of data transfers. Similarly, the DMA in DRAM controller of the enhanced architecture uses the same method except replacing the sector size by the burst length. Burst length 4 and 8 are commonly supported by DDR1, DDR2, and DDR3 standard protocols, hence either burst length can be used as the basic data transfer unit in the enhanced architecture.

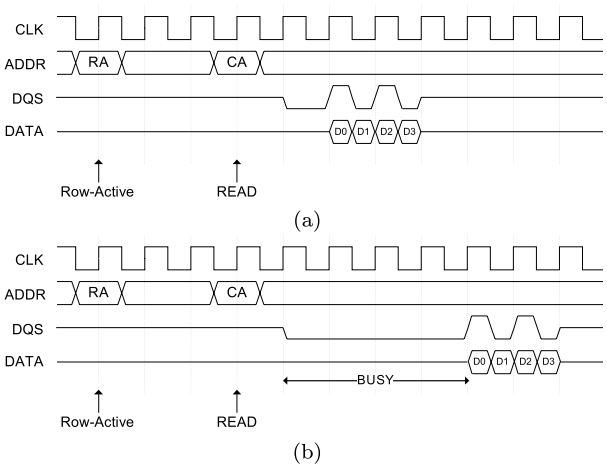

Fig. 3 Timing diagrams of an SSD: (a) When cache buffer read hit occurs. (b) When cache buffer read miss occurs.

5. Experimental Results

To show the effectiveness of our method, we modeled two transaction-level PC architectures with SystemC [15] based on the specification in [12] and [13], respectively. One is a conventional architecture with a SATA interfaced SSD as shown in Fig. 2 (a) and the other is the PC architecture with the proposed SSD as shown in Fig. 2 (b). To maximally exploit the performance of each interface, we set the interface bandwidth of SATA and DDR DRAM interfaces as 300 MB/s (SATA2) and 6400 MB/s (DDR2-800), respectively. Note that these values are the maximum bandwidth allowed in their specifications. Other system specifications of both architectures are identical for a fair comparison. Finally, we modeled the SSD according to the specification in [14]. We measured the data response time of both systems when the host transfers a page of 64 KB to and from an SSD by simulating two models.

In read mode, we performed the experiments for two extreme cases. The first case represents that every access to an SSD causes a cache miss, while the other case represents 100% of cache hit. As shown in Figs. 4 (a) and (b), the performance improvement ratios of both cases are 1.16 and 8.7, respectively. Such difference comes from the source of data. In case of 100% of read miss, every data is read from NAND Flash memory which is much slower than the cache buffer. Thus, the impact of interface improvement is marginal compared to the case of 100% of cache hit. Even in the worst case (100% of cache miss), the proposed architecture outperforms the conventional architecture by 16.20%, which is mainly due to the reduction of the SSD interface overhead.

In write mode, the performance improvement ratio of the proposed architecture is 1.50 as shown in Fig. 4 (c), which is greater than the performance improvement ratio in

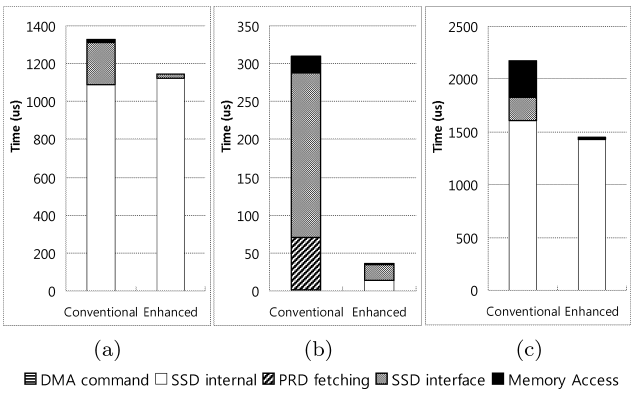

Fig. 4 64 KB DMA transfer: (a) Cache buffer read miss. (b) Cache buffer read hit. (c) DMA write.

Table 3 Utilized bandwidth of each interface.

Operation	Conventional I/F	Enhanced I/F
Write	35.96 MB/s	45.68 MB/s
Cache read miss	50.03 MB/s	57.97 MB/s
Cache read hit	296.53 MB/s	3175.19 MB/s

read mode with 100% of cache miss. The result is closely related to the elimination of South Bridge in the proposed scheme. In write mode, the data is fetched from the DRAM and then written to the NAND Flash memories inside an SSD. The reverse action is performed in read mode. However, there is a big difference such that the write operations are performed in a sequential manner, whereas the read operations are performed in a pipelined fashion as mentioned in Sect. 3. To summarize, the elimination of South Bridge is more beneficiary to the write operation due to its sequential operational property.

We also measured the utilized bandwidth of each interface as shown in Table 3. In case of write or cache read miss, both SSDs show similar utilized bandwidth. However, when a read cache hit occurs, the SSD with DDR DRAM interface utilizes much higher bandwidth than the SSD with SATA interface by a factor of 11. More importantly, SATA interface is almost saturated, while DDR DRAM interface can still manage the required bandwidth. In other words, SATA interface shows 98.84% of bandwidth utilization meaning that the system performance is limited due to its insufficient bandwidth. On the other hand, DDR DRAM interface utilizes 49.61% of its maximum bandwidth. This result show that the host interface of the conventional SSD can be a performance bottleneck even in contemporary PC architecture. Also, other interface schemes shown in Table 1 except PCI-E (×16) do not satisfy the maximum bandwidth (when 100% cache read hit) required by SSD even in contemporary PC architecture. Hence, we expect that DDR interface and PCI-E (×16) would be strong candidates of future SSD interface, since the internal architecture of SSD is continuously improved to maximize its internal throughput.

Finally, we evaluated the proposed interface scheme for real workload which was popularly used for SSD performance evaluation [16] and the trace can be downloaded from [17]. It is a trace collected by executing several applications (web surfing, email transfer, movie playing, downloading, document typing and gaming) on Windows XP. As shown in Fig. 5, the proposed architecture outperformed the conventional architecture by a factor of 1.6. Such improvement ratio is strongly correlated to the transaction mix of the trace. More specifically, 52% of the transactions in the trace is write, while the remaining portion (48%) is read. Also, 97.65% of read transactions incurs cache misses, hence the performance improvement ratio of the proposed architecture was limited to 60% over the conventional architecture. It is expected that our architecture can achieve larger gain by increasing the cache buffer size with more sophisticated replacement policy.

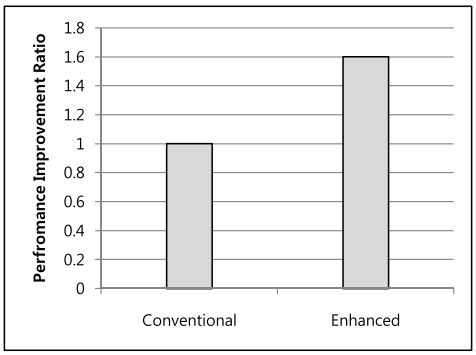

Fig. 5 Evaluation of the typical programs.

6. Conclusions

As the price of NAND Flash memories is decreasing, the demand for high performance SSDs will increase. However, the conventional SSD interface scheme can be a performance bottleneck of a high performance SSD. The proposed interface scheme shows the significant performance improvements in both write and read modes of an SSD. The experimental results showed that the proposed SSD maximally outperforms the traditional SSD by 8.7 times in read mode, by 1.5 times in write mode. Also, for synthetic workloads, the proposed scheme shows performance improvement over the conventional architecture by a factor of 1.6 times. Furthermore, the impact of our improvement will become larger as SSD itself becomes faster as shown in the experimental results of two extreme read cases.

Acknowledgement

This work was supported in part by Samsung Electronics Company and Korea Research Foundation Grant funded by the Korean Government (MOEHRD) KRF-2007-313-D00578.

References

[1] J.-H. Lee, G.-H. Park, and S.-D. Kim, “A new NAND-type flash

memory package with smart buffer system for spatial and temporal," J. Syst. Archit., vol.51, pp.111–123, Feb. 2005.
[2] C. Park, J. Kang, S.Y. Park, and J. Kim, "Energy-aware demand paging on NAND flash-based embedded storages," Proc. 2004 International Symposium on Low Power Electronics and Design (ISLPED'04), pp.338–343, 2004.
[3] J.-U. Kang, J.-S. Kim, C. Park, H. Park, and J. Lee, "A multi-channel architecture for high-performance NAND flash-based storage system," J. Syst. Archit., vol.53, pp.644–658, Sept. 2007.
[4] S.-L. Min and E.-H. Nam, "Current trends in flash memory technology," Proc. Asia and South Pacific Design Automation Conference (ASP-DAC), pp.332–333, Jan. 2006.
[5] C. Park, P. Talawar, D. Won, M. Jung, J. Im, S. Kim, and Y. Choi, "A high performance controller for NAND flash-based solid state disk," 21st IEEE Non-Volatile Semiconductor Memory Workshop (NVSMW), pp.17–20, Feb. 2006.
[6] SATA Specification (Rev.1.0a), http://www.serialata.org
[7] http://www.samsung.com
[8] http://www.pcisig.org
[9] http://www.usb.org
[10] http://www.scsita.org
[11] http://www.fibrechannel.org
[12] Intel corporation, "Intel 965 express chipset family," data sheet, July 2006.
[13] Intel corporation, "Intel I/O Controller Hub 8 (ICH8) family," data sheet, July 2006.
[14] Samsung Electronics Company, "Nand flash based SSD preliminary specification," data sheet, July 2007.
[15] IEEE Standard SystemC Language Reference Manual, http://www.systemc.org
[16] L.-P. Chang and T.-W. Kuo, "An adaptive stripping architecture for flash memory storage systems of embedded systems," Proc. IEEE Eighth Real-Time and Embedded Technology and Applications Symposium (RTAS), pp.187–196, San Jose, USA, Sept. 2002.
[17] http://newslab.csie.ntu.edu.tw/~flash