Reliable Hydra SSD Architecture for General Purpose Controllers

Renjith G.[1], Prof. R. Bijukumar[2]

[1]*PG Scholar, Dept. of Electronics and Communication Engg, TKM Institute of Technology, Kollam, Kerala, India*

[2]*Head of the Dept., Dept. of Electronics and Communication Engg, TKM Institute of Technology, Kollam, Kerala, India.*

Abstract: *The Solid State Disks (SSDs) had almost replaced the traditional Hard Disk Drives (HDDs) in modern computing systems. SSDs possess advanced features such as low power consumption, faster random access and greater shock resistance. In general, NAND Flash memories are used for bulk storage applications. This project focus on an advanced SSD architecture, called Reliable Hydra to enhance the SSD performance. Hydra SSDs overcomes the discrepancy between slow flash memory bus and fast host interfaces like SATA, SCSI, USB etc. It uses multiple high level memory controllers to execute flash memory operation without the intervention of FTL (Flash Translation Layer). It accelerates the processing of host write requests by aggressive write buffering. Memories are subjected to bit flipping errors, so this project also considers the incorporation of matrix code for increasing the reliability and hence yield of the system. The highly sophisticated controllers for real time systems in the industrial, robotics, medical and scientific applications require high performance and reliable memories. The aim of this project is to design Reliable Hydra SSD architecture for controller applications to enhance the performance. The design architecture is to be coded in VHDL using Xilinx ISE tools.*

I. Introduction

Flash memories are widely used as a storage medium in modern computing devices because of its low power consumption, fast random access, and high shock resistance. Most of the conventional Hard Disk Drives (HDDs) are replaced with the Solid State Disk Drives (SSDs). Solid State Disk Drives (SSDs) are much better suited to embedded computers and the cost of flash memory still going decreases. The widely used Flash memories are NAND flash based solid state disks (SSDs) [2] because of flash memories are nonvolatile, less power consumable and shock resistant. A basic NAND flash memory chip consists of a set of blocks, each sub divided with a set of pages. Each page has a provision for data that stores the user data and a separate provision that stores metadata associated with the user data. HDDs can be overwritten data directly, but in flash memory it is difficult to perform in place updating. The flash memory to get wide acceptance as a storage device, it has to imitate the functionality of HDDs. The software layer that provides this imitation is called the flash translation layer (FTL) [1]. It hides the anomalies of flash memory and gives the illusion of an HDD. The reliability of the data is very much important while saving or reading, because the application may be scientific, real time or in medical field. Here this project designs a new model of SSD architecture called Reliable Hydra SSD, which exploits the parallelism of multiple NAND flash memory chips to enhance storage system performance and the matrix code for increasing the reliability of the data.

Discrepancy between the slow flash memory bus and the fast host interface is overcome by interleaving enough flash memory buses so that their effective bandwidth meets that that of the host interfaces. In addition to this bus-level interleaving, chip-level interleaving hides the flash read latency. Multiple high level flash memory controllers execute sequences of high level flash memory operations without any intervention by the FTL. One of these controllers is named as the foreground unit and the other is called background units, which have less priority. The foreground unit is used to accelerate the processing of host read requests for which processes in the host system are waiting. Aggressive write buffering accelerates.

The bus level and chip level interleaving of flash memory provides the parallelism and increase the transfer rate of the data. Most of the host controller transfers data in very high rate and a single chip memory cannot follow the transfer rate of the interface. So this superchip increases the fastness of the data read and write operations and it bring the speed up to the host controller. The flash translation layer (FTL) hides the anomalies of flash memory and imitates the functionality of an HDD. The most important role of the FTL is to maintain a mapping between the logical sector address used by the host system and the physical flash memory address.

In this project, a high level method for detection and correction of multiple faults using Matrix [3] code is presented. This code is a combination of Hamming codes and Parity codes in a matrix format which can correct multiple faults in memory. A memory chip protected with the Matrix Code technique, it possess high reliability than the Cyclic Redundancy Check (CRC), Hamming [3] or Reed-Muller codes. The overhead of Matrix Code is less than the overhead imposed by the Reed-Muller code. Note that, although Cyclic Redundancy Check (CRC), Hamming code has less overhead compared to the matrix Code method and Reed-Muller [3], it can detect only two errors and correct one error.

II. Related Works

Most of the modern computing devices have recently started replacing hard drives with NAND [2][8] flash based solid state disks (SSDs). An SSD communicates with the host computer through a standard interface, such as PATA (Parallel Advanced Technology Attachment), SATA (Serial Advanced Technology Attachment), SCSI (Small Computer System Interface) or USB (Universal Serial Bus), and behaves much like a standard hard drive. An SSD incorporates a controller and a flash memory array and the SSD controller firmware performs disk emulation. The read/write/erase behavior of flash memory is differing than that of other programmable memories, such as magnetic disks and volatile RAM (Random Access Memory).

In fact, flash memories come in two flavors, NOR and NAND, that are also entirely different from each other. In both types, write operations can only clear bits (change their value from 1 to 0). The only way to set bits (change their value from 0 to 1) is to erase an entire region of memory. These regions have fixed size in a given device, typically ranging from several kilobytes to hundreds of kilobytes and are called erase units. NOR flash memory, the older type for code storage, is a random access device that is directly addressable by the processor. NOR flash memory suffers from high erase times. NAND flash memory [9], the newer type for data storage, enjoys much faster erase times, but it is not directly addressable, access is by page (a fraction of an erase unit, typically 512 bytes) not by bit or byte, and each page can be modified only a small number of times in each erase cycle. That is, after a few writes to a page, subsequent writes cannot reliably clear additional bits in the page; the entire erase unit must be erased before further modifications of the page are possible. Because of these peculiarities, storage management techniques that were designed for other types of memory devices, such as magnetic disks, are not always appropriate for flash memory.

Since most operating systems does not support flash memory directly but it is possible with a thin software layer called FTL (Flash Translation Layer) is usually employed between OS and flash memory. The main role of FTL is to imitate the functionality of HDD, hiding the latency of erase operation as much as possible. FTL achieves this by redirecting each write request from OS to an empty location in flash memory that has been erased in advance, and by maintaining an internal mapping table to record the mapping information from the logical sector number to the physical location. The mapping schemes of FTL are classified into a page level mapping scheme into a block level or into a hybrid mapping scheme. In page level mapping, a logical page can be mapped to any physical page in flash memory. In block level mapping, only the mapping information between the logical block number (LBN) and the physical block number (PBN) is maintained. Hybrid mapping is a compromise between page level mapping and block-level mapping. Many FTL schemes are proposed to reduce the number of block erase and the data page copying in block level mapping.

The use of block level mapping in Hydra considerably simplifies wear leveling since block level mapping, unlike page level mapping, does not suffer from the complications that arise if wear leveling is coupled to garbage. Instead, Hydra uses two simple techniques borrowed from wear leveling in page mapping FTLs: one is implicit and the other explicit. In implicit wear-leveling [1] [10], when a merge operation is performed, the free physical superblock with the smallest erase count is used as the destination of the copy back superchip operation. In explicit wear leveling, when the SSD is idle, the physical superblock with the smallest erase count (among those mapped to logical superblocks) is swapped with the free physical superblock with the largest erase count, provided that the difference between the two counts is above a certain threshold.

Matrix Code is the combination of Hamming Code and the Parity Code. It can correct multiple faults in the memory. Concurrent Error Detection (CED) [3] is one of the major approaches for transient errors mitigation. In its simpler forms, CED allows only the detection of errors, requiring the use of additional techniques for error correction. Nevertheless, the implementation of CED usually requires sometimes the duplication of the area of circuit to be protected. One of the simpler examples of CED is called duplication with comparison, which duplicates the circuit to be protected and compares the results generated by both copies to check for errors. This technique imposes an area overhead higher than 100%, and when an error is detected the outputs of the circuit must be recomputed.

Hamming codes and odd weight codes are largely used to protect memories against SEU because of their efficient ability to correct single upsets with a reduced area and performance overhead. The Hamming code implementation is composed by a combinational block responsible to code the data (encoder block), inclusion of extra bits in the word that indicate the Parity (extra latches or flip flops) and another combinational block responsible for decoding the data (decoder block). The encoder block calculates the parity bit, and it can be implemented by a set of two input XOR gates. The decoder block is more complex than the encoder block, because it needs not only to detect the fault, but it must also correct it.

The Reed–Muller [3] code is another protection code that is able to detect and correct more errors than a Hamming code and is suitable for tolerating multiple upsets in SRAM memories. Although this type of protection code is more efficient than Hamming code in terms of multiple error detection and correction, the main drawback of this protection code is its high area and power penalties (since encoding and decoding correcting circuitry in this code is more complex than for a Hamming code). Hence, a low overhead error detection and correction code for tolerating multiple upset is required.

III. System Architecture

The basic NAND flash based memories have single bus chip architecture and are constructed from an array of flash packages. Each SSD must contain host interface logic to support some form of physical host interface connection (USB [4], Fiber Channel, PCI Express, and SATA) and logical disk imitation, like a flash translation layer mechanism to enable the SSD [2] to mimic a hard disk drive. The bandwidth of the host interface is often a critical parameter on the performance of the device as a whole, and it must be matched to the performance available to and from the flash array.

A. Basic NAND flash Architecture

The basic NAND flash based memory architecture as the following figure 1. The architecture contains the NAND based flash memory chip, NAND flash controller, Cache buffer controller and SRAM etc.

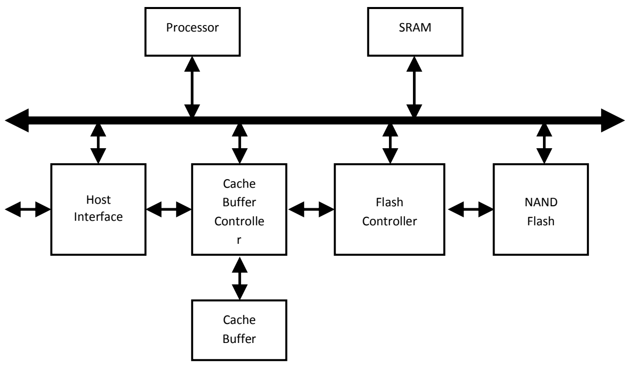

Figure: 1 Basic NAND flash Architecture.

One of the major disadvantages of the basic NAND flash based memory is the discrepancy between the transfer rate of the host interface and the flash memory. Most of the host controllers (SATA, SCSI, and USB [4] etc.) are operating at very high transfer rate because of the technology advantage. Operating systems use storage devices to provide file systems and virtual memory, and it is usually assumed that these devices have an HDD like interface. The software layer that provides this imitation is called the flash translation layer (FTL). It hides the anomalies of flash memory and gives the illusion of an HDD. The FTL needs a device driver installation for the particular storage volume. In the case of firmware it may not be feasible.

Another important parameter is the reliability of the data. In the basic NAND [10] flash based memories only a simple error check and correction mechanism like CRC. It only detects and corrects single bit errors. It cannot do the bulk data error check and correction. The disadvantages of basic NAND flash based memories are at a glance is

1 The discrepancy between slow flash memory bus and the fast host interface.
2 The intervention of Flash Translation Layer (FTL).
3 There is no efficient mechanism for the reliability of the data.

B. Reliable Hydra SSD Architecture

The Reliable Hydra [1] Solid State Disk (SSD) architecture is a new proposed model which over comes all the disadvantages of the conventional NAND flash based memories. The Hydra SSD architecture uses various techniques to achieve this goal. The discrepancy between the slow flash memory bus and the fast host is overcome by interleaving enough flash memory buses so that their effective bandwidth meets that of the host interface. In addition to this bus level interleaving, chip level interleaving hides the flash read latency.

Multiple high level flash memory controllers execute the operations of high level flash memory without any intervention by the FTL. One of these controllers is designated as the foreground unit and has priority over the remaining controllers, called background units. Aggressive write buffering expedites the processing of host write requests. More importantly, it also allows the parallelism in multiple flash memory chips to be exploited by multiple background units that perform materialization to flash memory in parallel on different interleaved units. By introducing Matrix Code [3] error detection and correction mechanism, which provides the reliability of the data while reading or writing it to the flash memory without make high overheads. In Figure 2, block MCE represents Matrix code encoder and block MCD represents Matrix code decoder.

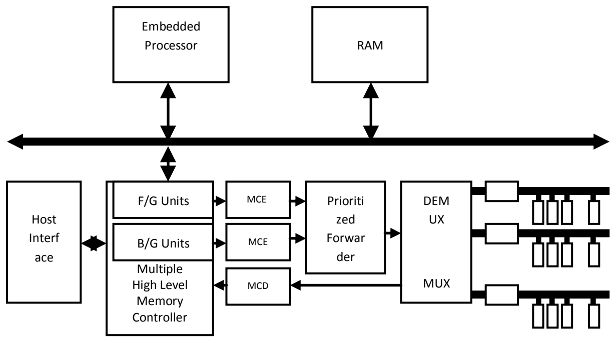

Figure: 2 Reliable Hydra SSD Architecture.

C. Chip level and bus level interleaving

The Hydra SSD uses interleaving over multiple flash memory buses to overcome the bandwidth limitation of the flash memory bus. In the bus level interleaving, each memory location within a superblock are distributed in a round robin manner. In Hydra, the set of flash memory chips that are related to each other by the bus level and chip level interleaving is called a superchip.

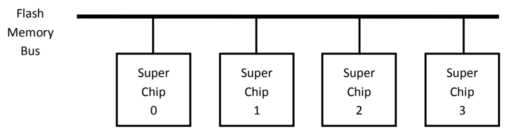

Figure: 3 Superchips.

D. Multiple high level memory controllers

The Hydra SSD architecture uses multiple high level flash memory controllers, consisting of one foreground unit and several background units. Each controller is capable of executing a sequence of high level flash memory operations, as the descriptors. In Hydra, a high level flash [9] memory operation is directed to a superchip, and it is to be designed to perform the read, write, erase and program the chip. The operation of the MHL memory controller is based on a counter FSM algorithm. Here two counters operate in parallel for the operation of the MHL counter algorithm, read count FSM and write count FSM.

E. Matrix Code error check and correction mechanism

Matrix Code [3] is the combination of Hamming Code and the Parity Code. It can correct multiple faults in the memory. By introducing Matrix Code error detection and correction mechanism, which provides the reliability of the data while reading or writing it to the flash memory without make high overheads.
The Reed–Muller code is another protection code that is able to detect and correct more errors than a Hamming code and is suitable for tolerating multiple upsets in SRAM memories. Although this type of protection code is more efficient than Hamming code in terms of multiple error detection and correction, the main drawback of this protection code is its high area and power penalties (since encoding circuitry ,decoding and correcting circuitry in this code is more complex than for a Hamming code). Hence, a low-overhead error detection and correction code for tolerating multiple upset is required.

In order to improve the efficiency of repairing embedded memories, a novel redundancy mechanism to cope not only with defects but as well as with transients is required. The proposed technique, in comparison with the previous techniques, improves the overall systems reliability.

IV. Simulation Results

The design entry is modeled using VHDL in Xilinx ISE Design Suite 12.1 and the simulation of the design is performed using ModelSim SE 6.2c from Mentor Graphics to validate the functionality of the design. Chip structure for Super Chip flash memory module is designed as follows

1 Word	=	32 bit
1 Page	=	4 Words (Word 0, Word 1, Word 2, Word 3)
1 Block	=	2 Pages (Page 0, Page 1)
1 Chip	=	4 Blocks (Bus 0, Bus 1, Bus 2, Bus 3)
1 Super Chip	=	2 Chips (Chip 0, Chip 1)

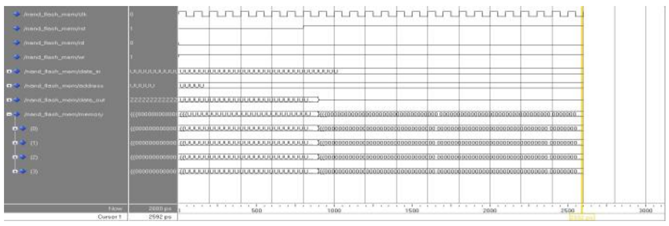
Figure: 4 Synchronous reset.

The input signals to the memory are the global clk, rst, rd, wr, address and data_in. Applying the global clk signal and run the simulation we can see that the flash memory is in high impedance state. After applying the rst signal high, rd low, wr high it can be see that the entire memory is reset to zero. Data out is also at high impedance state.

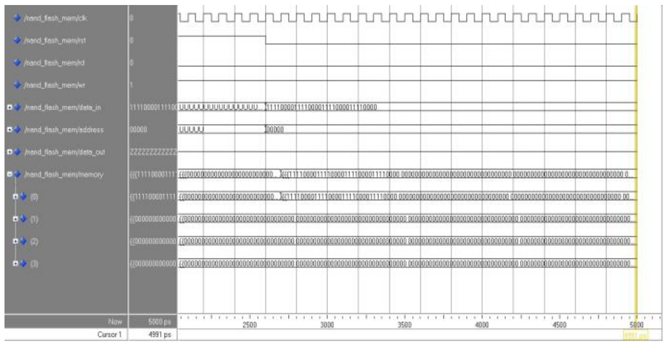
Figure: 5 Memory write operation.

For memory write operation the rst signal to low and make a particular data force to the data_in and also force the address signal to a particular address. Make the rd signal is low and wr signal is high then run the simulation it can see that the particular data is write to the particular memory.

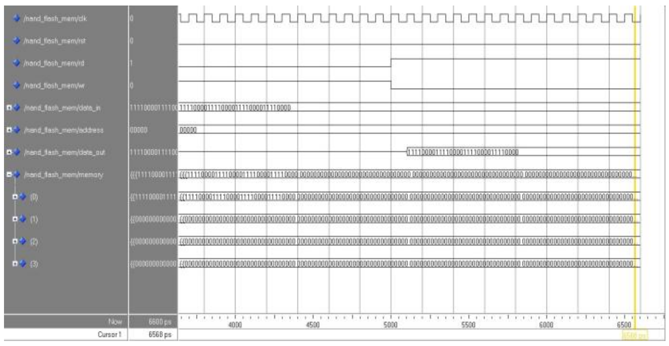
Figure: 6 Memory read operation.

For memory read operation the rst signal to low and make a particular data force to the data_in and also force the address signal to a particular address. Make the rd signal is high and wr signal is low then run the simulation it can see that the particular data is read from the particular memory address to the data out.

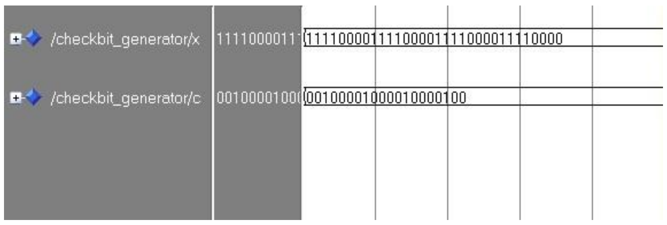

Figure: 7 Check bit generation.

Check bit generator is the part of Matrix encoder and that module calculates according to the check bit generation formula and generates the check bits for the particular input data to the encoder, here data is given to the check bit generator and corresponding check bits are generated.

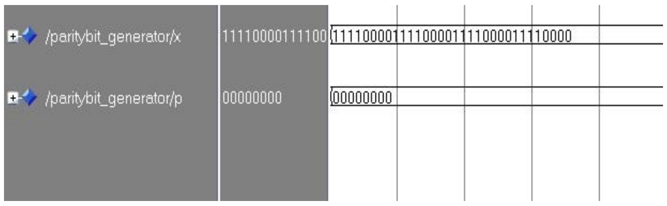

Figure: 8 Parity bit generation.

Parity bit generator is the part of Matrix encoder and that module calculates according to the parity bit generation formula and generates the parity bits for the particular input data to the encoder. Parity bit is calculated in the eight columns of the data separately and here two data is given to the parity bit generator and corresponding parity bits are generated.

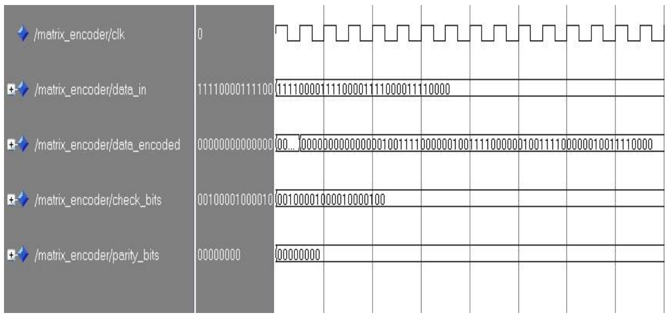

Figure: 9 Matrix code encoder.

Matrix encoder is the combined module of check bit generator and the parity bit generator modules and this module encodes the data given to the memory as per the algorithm and the encoded data is to be writes to the flash memory as the super chip sequence reading method. It is in a round robin manner.

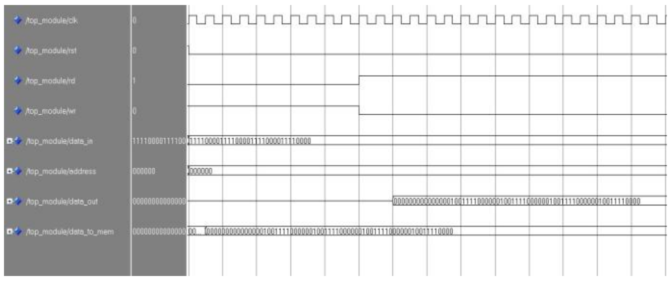

Figure: 10 Matrix code encoder with flash memory.

The simulation result in the figure 10 shows the combination of the matrix code encoder and the super chip flash memory. By applying the global clk and the rst signal the entire chip reset and ready for a read or write operation. Data is given to the particular address as the input of the top module which first encode the data by using the matrix code and write to the memory that the address of the memory is given.

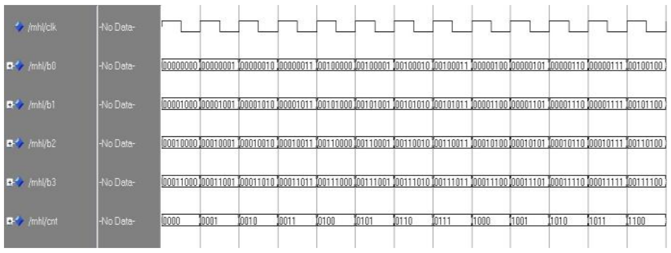
/mhl/clk -No Data-
/mhl/b0 -No Data- 00000000 00000001 00000010 00000011 00100000 00100001 00100010 00100011 00000100 00000101 00000110 00000111 00100100
/mhl/b1 -No Data- 00001000 00001001 00001010 00001011 00101000 00101001 00101010 00101011 00001100 00001101 00001110 00001111 00101100
/mhl/b2 -No Data- 00010000 00010001 00010010 00010011 00110000 00110001 00110010 00110011 00010100 00010101 00010110 00010111 00110100
/mhl/b3 -No Data- 00011000 00011001 00011010 00011011 00111000 00111001 00111010 00111011 00011100 00011101 00011110 00011111 00111100
/mhl/cnt -No Data- 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100

Figure: 11 MHL memory controller counter FSM.

REFERENCES

[1] Yoon Jae Seong, Eyee Hyun Nam, Jin Hyuk Yoon, Hongseok Kim, Jin-Yong Choi, Sookwan Lee, Young Hyun Bae, Jaejin Lee, Member, *IEEE*, Yookun Cho, Member, *IEEE*, and Sang Lyul Min, Member, *IEEE*, "Hydra: A Block-Mapped Parallel Flash Memory Solid-State Disk Architecture", *IEEE Transactions on Computers*, Vol. 59, No. 7, July 2010, pp 905-921.

[2] Li-Pin Chang, "A Hybrid Approach to NAND-Flash-Based Solid-State Disks", *IEEE Transactions on Computers*, Vol. 59, No. 10, October 2010, pp 1337-1349.

[3] Costas Argyrides, Member, *IEEE*, Dhiraj K. Pradhan, Fellow, *IEEE*, and Taskin Kocak, "Matrix Codes for Reliable and Cost Efficient Memory Chips", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 19, No. 3, March 2011, pp 420-428.

[4] Chung-Ping Young, Michael J. Devaney, Member, *IEEE*, and Shyh-Chyang Wang, "Universal Serial Bus Enhances Virtual Instrument-Based Distributed Power Monitoring", *IEEE Transaction on Instrumentation and Measurement*, Vol. 50, No. 6, December 2001, pp 1692-1696.

[5] Jin Hyuk Yoon1, Eyee Hyun Nam2, Yoon Jae Seong2, Hongseok Kim2, Bryan S. Kim2, Sang Lyul Min2, and Yookun Cho3 School of Computer Science and Engineering, Seoul National University, Seoul, Korea, "Chameleon: A High Performance Flash/FRAM Hybrid Solid State Disk Architecture", *IEEE Computer Architecture Letters*, Vol. 7, No. 1, January-June 2008, pp 17-20.

[6] Tao Xie, Member, *IEEE*, and Yao Sun, "Dynamic Data Reallocation in Hybrid Disk Arrays", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 21, No. 9, September 2010, pp 1330-1341.

[7] Dong Kim, Kwanhu Bang, Student Member, *IEEE*, Seung-Hwan Ha, Sungroh Yoon, Member, *IEEE*, and Eui-Young Chung, Member, *IEEE*, "Architecture Exploration of High-Performance PCs with a Solid-State Disk", *IEEE Transactions on Computers*, Vol. 59, No. 7, July 2010, pp 878-890.

[8] Soojun Im and Dongkun Shin, Member, *IEEE*, "Flash-Aware RAID Techniques for Dependable and High-Performance Flash Memory SSD*", IEEE Transactions on Computers*, Vol. 60, No. 1, January 2011, pp 80-92.

[9] Seung-Ho Park, Jung-Wook Park, Shin-Dug Kim, and Charles C. Weems, "Brief Contributions A Pattern Adaptive NAND Flash Memory Storage Structure", *IEEE Transactions on Computers*, Vol. 61, No. 1, January 2012, pp 134-138.

[10] Sarah Boyd, Arpad Horvath, and David Dornfeld, "Life-Cycle Assessment of NAND Flash Memory", *IEEE Transactions on Semiconductor manufacturing*, Vol. 24, No. 1, February 2011, pp 117-12.