특집

NAND Flash와 Software

권문상·이상훈·정승진·임승호 (삼성전자)

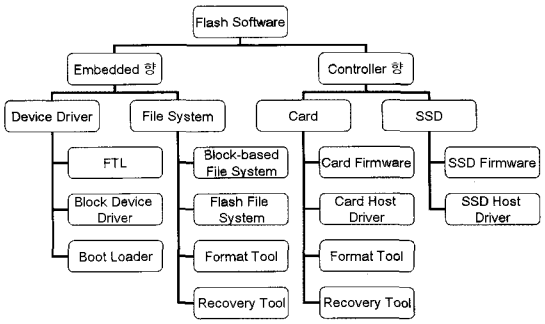
I. 서 론

NAND Flash를 저장장치로 사용하는 가전 기기들이 더욱 일반화되고 있다. 특히, 옴니아II나 iPhone 같은 Smart Phone이 확산되면서 수십 GB 크기의 NAND Flash를 지니고 다니는 시대가 되었다. 삼성전자는 NAND Flash 시장 1위 업체로서 NAND Flash의 제조뿐만 아니라 NAND Flash를 제어하는 여러 가지 Software들을 연구 개발하여 NAND Flash와 함께 제공하고 있다. NAND Flash는 여러 가지 물리적인 특성 때문에 이를 어떻게 제어하느냐에 따라서 성능과 수명 차이가 많이 있을 수 있다.

<그림 1>은 NAND Flash Software를 적용 제품군에 따라 분류한 것이다. NAND Flash Software는 크게 Embedded 향과 Controller 향으로 나뉜다. Embedded 향은 휴대폰 같이 Host System에 NAND가 직접 연결되는 환경을 위한 것으로, NAND를 제어하는 Software가 운영체제의 일부분으로 동작한다. Controller 향은 Card나 SSD 같이 저장장치 전용의 Embedded Computer System을 가지면서 외부 Host와는 SATAII 같은 Storage Interface를 통해서 연결되는 환경이다. 이 경우 NAND Flash Software는 Firmware의 형태로 Storage 제품 내부에서 각종 Hardware Resource들을 제어하고 Host와의 통신을 처리한다.

NAND Flash를 직접 제어하는 Software 이외에도 NAND Flash Storage를 각 Embedded System에서 사용할 수 있도록 Adaptation 해주는 Block Device Driver, Application의 IO Pattern을 NAND Flash Storage가 선호하는 IO Pattern으로 만들어 주는 NAND 향 File System과 Application Library, 유지 보수 및 관리를 위한 Utility 등이 있다.

본 문서에서는 <그림 1>에서 분류하고 있는



〈그림 1〉 적용 제품군에 따른 NAND Flash Software의 분류

각 Software 분야 가운데 중요한 몇 가지 NAND Flash Software에 대한 연구개발 경험을 소개한다.

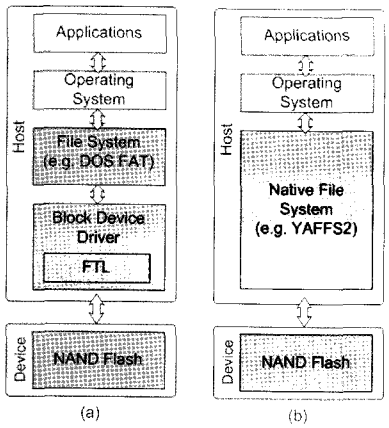
II. Embedded 향 Flash Software

Embedded 향 Storage System의 경우 Data용 Storage의 요구량이 커지면서 Code용 NOR의 크기는 줄이고 NAND의 크기를 늘리는 쪽으로 발전하였으며, Booting까지 지원하는 OneNAND와 Flex-OneNAND의 등장으로 NOR를 사용하지 않는 구성도 가능하게 되었다[1,2].

<그림 2>는 Embedded System에서 NAND Device를 제어하는 NAND Flash Software의 구성방식을 나타낸 것이다. Operating System은 File System을 통해서 File System API Service를 Application에게 제공한다. (a) 방식에서 File System은 저장장치를 '**Sector[1] 단위로 읽기와 쓰기가 가능한 Block Device**'로 가정하고 동작한다. Block Device Driver의 내부에 있는 FTL은 Sector 단위의 읽기와 쓰기 요청을 적절한 NAND Flash Operation으로 변환하여 수행함으로써 NAND Flash를 일반적인 Block Device Storage인 것처럼 사용할 수 있게 한다. 이 때, FTL은 단순한 Logical 주소의 재매핑(re-mapping) 뿐만 아니라 Wear-leveling이나 Bad Block 관리, Bit Error 보정 같은 부가적인 작업을 통해서 NAND Flash Storage의 신뢰성을 향상시킨다.

(b) 방식의 경우 File System을 설계할 때부터 NAND Flash의 특성을 고려한 것으로, 성능이나 신뢰성 측면에서 좀 더 유리한 구현이 가능한 것으로 알려져 있다. (b) 방식으로 개발된 최초의 File System은 YAFFS인 것으로 알려져 있으며 Android에서 Root File System으로 YAFFS2를 채택하고 있다[2].

NAND Flash는 H/W 특성상 아래와 같이 선호하는 IO Pattern이 있으며, Embedded 향 Flash Software의 성능을 최적화하려면 응용프로그램을 작성하는 단계부터 이에 대해 고려해야 한다.



<그림 2> Embedded 향 Flash Software 구성

- Page에 Align된 IO: Logical Sector 주소를 기준으로 IO의 시작 주소와 버퍼의 길이가 NAND의 page 또는 Mapping Unit[3] 크기에 align 되어야 불필요한 Data padding이나 program을 피할 수 있다.
- {조금씩, 자주} 보다는 {많이, 연속된 주소로}: IO 횟수와 연관된 것으로, 같은 양의 Data를 IO할 때, 가능한 연속되게 긴 단위로 IO를 수행하는 것이 유리하다.

1) 보통 512-Byte 크기이다.

2) http://en.wikipedia.org/wiki/YAFFS

3) FTL이 관리하는 최소 IO 단위로 보통 NAND의 page크기의 배수이다.

- 가능한 쓰지 않음: 보통 Storage에 Data를 기록할 때, RAM 상에 두었다가 적당한 시점에 non-volatile 하게 저장한다. 이 때, non- volatile 하게 Storage에 저장하는 것을 최대한 줄이는 IO Pattern이 유리하다. Application이나 File System이 가정하는 Synchronous IO 요구 조건을 만족하는 범위 안에서 가능한 IO를 유발하지 않도록 할 수 있다.
- 더 이상 유효하지 않은 영역에 대한 정보 제공: Application이나 File System이 Storage에 기록되었던 데이터 중에서 더 이상 유효하지 않은 영역에 대한 정보를 FTL에게 전달하는 경우, FTL은 이를 고려하여 좀 더 효율적으로 동작할 수 있다.

1. Flash Translation Layer

FTL(Flash Translation Layer)은 상위 계층 소프트웨어가 NAND Flash 고유의 특징들을 별도로 고려하지 않아도 기본적인 Block Device 관련 연산이 가능하도록 별도의 추상화를 제공한다. FTL이 고려해야 할 NAND Flash의 특성들은 여러 가지가 있겠지만, 그 중 기본적으로 고려해야 할 특성들을 세 가지만 언급하면 다음과 같다.

첫째, NAND Flash는 program 단위(page)와 erase 단위(block)가 다르며, overwrite를 허용하지 않고, 지정된 page를 program하기 위해서는 해당 page가 속한 block이 사전에 erase 되어 있어야 한다. 둘째, program/erase가 정상적으로 수행될 수 없는 소위 bad(or invalid) block이라 불리는 비정상 block들이 존재한다. Bad block은 동작 중에도 발생할 수 있다. 셋째, page program 도중 power가 off되면 해당 page 또는 주변 page의 Data가 망가질 수 있다.

첫 번째와 두 번째 특성은 Block Device Driver가 사용하는 Logical Address와 실제로 NAND에 저장되는 Physical Address가 달라지는 문제가 생기는데, FTL은 Logical Address로부터 Physical Address를 얻어낼 수 있는 mapping 메커니즘을 구현하여 이러한 문제를 해결한다. FTL의 mapping 메커니즘은 전체 Computer System의 read와 write 성능에 큰 영향을 미치게 되므로, FTL을 설계할 때 여러 가지 선택 사항들을 신중하게 결정해야 한다. Mapping 단위와 방식을 어떻게 할 것인가? 각 방식의 Mapping 정보의 양은 어떻게 차이 나는가? Mapping 정보를 non-volatile하게 저장할 때, 그 공간을 얼마나 할애할 것이며 위치는 어디로 할 것인가? Non-volatile하게 저장되어 있는 mapping 정보를 RAM 공간으로 loading 하는 시점은 언제로 할 것이며, 빈도는 어떻게 될 것인가? 등의 설계 결정 사항들이 있다. Mapping 단위는 block 별로 관리하는 방법, sector 별로 관리하는 방법, 이 둘을 조합하는 방법 등 그동안 여러 기법들을 발전시켜왔으며, mapping 정보를 저장하는 위치에 있어서도 block 마다 저장하는 방식과 특정 영역에 몰아두는 방식으로 구분할 수 있다.

FTL의 내부 mapping 메커니즘 설계 방식은 Embedded System 전체의 성능에 큰 영향을 줄 뿐만 아니라, mapping 정보를 읽고 저장하는 빈도와 그 양에 따라 NAND 메모리의 수명에도 영향을 크게 준다. 최근 반도체 공정이 미세화 되면서 NAND의 신뢰성이 취약해지고 있는 추세이므로, mapping 메커니즘 설계시 NAND Flash 메모리의 수명도 아울러 고려하는 것이 NAND

device 자체의 성능 특성 개선 못지않게 매우 중요한 이슈가 되고 있다.

NAND Flash의 신뢰성과 관련된 특성 항목으로는 block당 program과 erase 횟수를 규정하는 endurance와 read 횟수를 규정하는 read disturbance 특성을 들 수 있는데, 이러한 NAND Flash의 특성을 개선시키기 위해 FTL은 가급적 NAND Flash로의 program/erase 횟수를 최소화하고, read disturbance로 인한 read error를 사전에 미리 예방하는 기법들을 고안할 수 있다.

삼성전자는 Embedded 향 FTL로 OneNAND 향인 XSR과 Flex-OneNAND 향인 FSR을 제공한다[3].

2. Block Device Driver

Block Device Driver는 각 OS마다 구조와 제공하는 API가 다르기 때문에, 이에 맞추어 개발해야 한다. 삼성전자는 Embedded 제품군을 위하여 RTOS, Symbian, Linux, Windows CE/Mobile 용 Block Device Driver를 제공하고 있는데, 각각의 Package 이름은 <표 1>과 같다. 특히, Linux의 경우 Open Source의 형태로 MTD Driver를 제공하므로 Open Source에서도 OneNAND나 Flex-OneNAND를 사용할 수 있다[3].

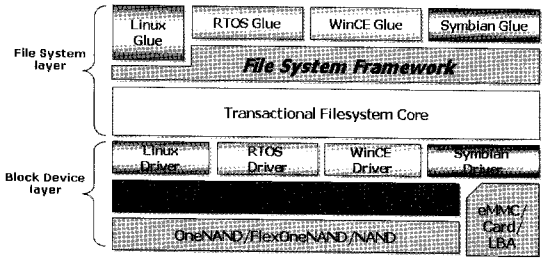
〈표 1〉 Embedded 향 OS별로 제공하는 NAND Flash Block Device Driver

OS	제공 Block Device Driver
RTOS	각 RTOS 마다 Porting 가능하도록 Porting Layer 제공
Symbian	UniStoreII (for OneNAND) UniStoreIII (for Flex-OneNAND)
Linux	LinuStoreII (for OneNAND) LinuStoreIII (for Flex-OneNAND) OneNAND/Flex-OneNAND MTD
Windows CE/Mobile	PocketStoreII (for OneNAND) PocketStoreIII (for Flex-OneNAND)

3. File System

삼성전자는 Embedded 향의 각 OS별로 Transactional File System을 제공하고 있다. Transactional File System은 File System의 Meta Data 변경을 유발하는 각 File System API를 Transaction으로 처리한다. 만일 File System Meta Data를 수정하는 도중 전원이 꺼진다면, 다시 전원이 인가되어 해당 File System 볼륨을 Mount할 때 Transaction Recovery 과정을 통해서 File System Meta Data의 Integrity를 복구한다. Transaction 지원은 특히 배터리 분리형 휴대폰에서 필수적으로 지원해야 한다. Transaction을 지원하려면 추가적인 IO가 유발되기 때문에 Transaction을 지원하면서도 전체적인 쓰기 성능이 떨어지지 않도록 하는 것이 중요하다. 개발 초기에는 하나의 File System API를 Transaction으로 처리하기 위해 여러 개의 Log Write가 필요했으나 현재는 여러 개의 Transaction을 처리하는데 1개의 Log Write만 필요한 수준까지 발전하였다.

<그림 3>은 여러 가지 Embedded 향 OS들을 하나의 File System Core로 지원할 수 있는 File System의 Architecture이다. 초기에는 각 OS별로 File System을 각각 개발하였으나, OS별 Block Device Driver와 마찬가지로 하나의 File System Core를 개발하고 이를 각 OS별로 Adaptation하는 방식으로 발전하였다. Glue 모듈은 각 OS별로 존재하는 Virtual File System 구조에 맞게 Filesystem Core를 Adaptation 한



〈그림 3〉 여러 OS를 동시에 지원하는 File System 구조

다. 이렇게 함으로써 초기 개발비용은 증가하지만 유지보수나 새로운 File System을 지원하려는 경우 개발 기간을 단축할 수 있다.

File System은 Application이 요청하는 IO Pattern을 NAND Flash가 선호하는 IO Pattern으로 변환하는데 매우 유리한 위치에 있다. 사용자의 Data Write 요청을 실제 Storage에 기록하는 시점을 File System이 결정할 수 있기 때문에, 적절한 시점에 적절한 IO 형태로 Block Device Driver에게 IO 요청을 할 수 있다. <그림 4>는 Application이 File System API를 사용하는 일반적인 방식을 C 언어로 작성한 것이다. <그림 4>의 1)이나 2)의 위치에서 File System은 Data를 Storage에 non-volatile 하게 저장하지 않아도 된다. 즉, Application이 명시적으로 sync 요청을 하기 전까지는 해당 Data를 Storage에 기록하지 않아도 된다[4]. 단, fsync()가 호출되고 난 3)의 시점이나 해당 file의 descriptor를 close 하는 4)의 시점에서는 해당 Data를 non-volatile 하게 Storage에 저장해야 한다. 따라서 Application의 입장에서 명확하게 Data가 Storage에 기록되었음을 보장받고자

```
int fd = open("file.txt", "rw");

write(fd, "this is test data1", 10);
1)
write(fd, "this is test data2", 10);
2)
...
fsync(fd);
3)
close(fd);
4)
```

Where is data at 1), 2), 3), 4) ?

〈그림 4〉 File System 최적화

하는 부분에서만 sync 요청을 하도록 Application을 작성해야 NAND Flash를 사용하는 시스템에서 최대의 성능을 얻을 수 있다. <표 2>는 각 OS별로 제공하는 File System Solution이다[3].

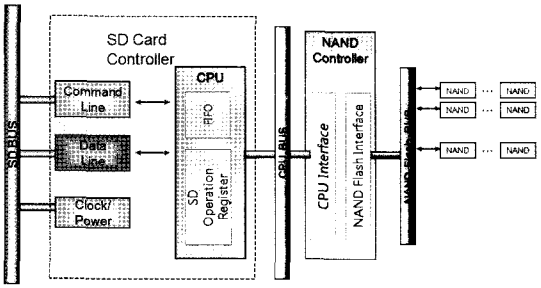
〈표 2〉 Embedded 향 OS별 File System Solution

OS	삼성전자 제공 File System
Windows CE/ Mobile	PocketFS
Linux	RFS
Symbian	UniFS
RTOS	TFS4

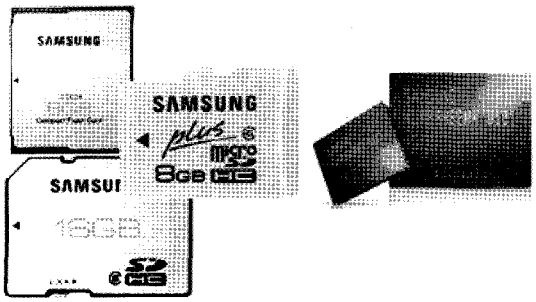
IV. Card 향 Flash Software

Card 타입의 Storage에는 Compact Flash, Memory Stick, eMMC, SD Card 등 여러 가지가 있다. 이들 모두는 Card Host와 연결되어 사용되면서, Card 안에 CPU와 SRAM, BUS, NAND Controller 등 독립적인 Computer System을 내장하고 있다는 공통점을 가지고 있다. <그림 5>는 SD Card의 내부를 Block Diagram으로 나타낸 예이다[5]. Host와는 SD Bus로 연결되어 있다. 내부적으로는 NAND

4) File System으로 하여금 Memory 상의 Data를 Non-volatile 하게 Storage에 저장하도록 하는 API는 여러 가지가 있다. 보통 이런 명령을 flush나 sync 명령이라고 한다.



〈그림 5〉 SD Controller Block Diagram



〈그림 6〉 삼성전자 SD Card와 eMMC Card

Controller를 통해서 NAND Flash를 제어한다. Card 내부에 있는 CPU에서 수행되는 Card Firmware는 Command를 해석하여 처리하고, Data의 송수신이 필요한 경우 NAND Flash와 FIFO 사이의 Data 송수신을 제어한다.

현재 외장형 Card 시장은 SD 계열이, 내장형 Card 시장은 eMMC 계열이 널리 사용되고 있다. SD Card는 Host에 착탈식으로 연결되어 사용하고, eMMC는 Host 내부에 고정된 형태로 사용한다[4~6]. Card의 기본 용도는 사진이나 동영상 등 Data를 저장하는 것이지만, 최근에는 Application을 저장하고 수행하는 **'Code 저장용'**으로까지 확대되고 있다. 이 것은 NAND Flash의 공정 미세화와 MLC 사용이 일반화 되면서 NAND Flash를 효과적으로 사용하기 위한 H/W와 S/W 기술이 점점 더 중요해 짐에 따른 자연스러운 진화이다.

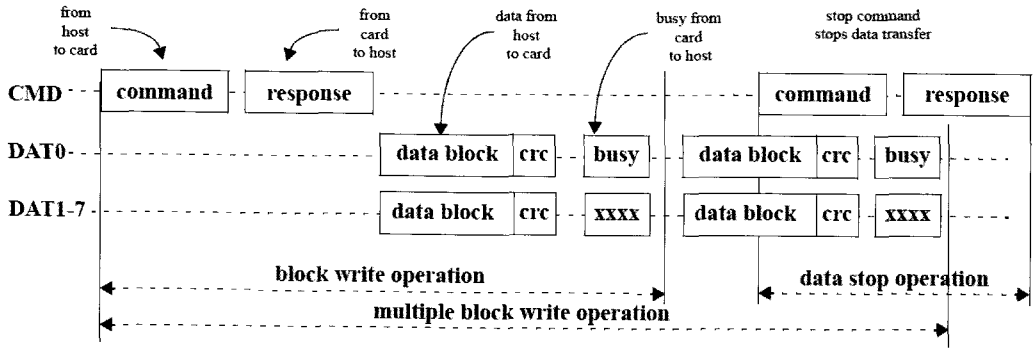
삼성전자는 eMMC(= moviNAND)와 SD Card를 양산[5]하고 있는데, 이 장에서는 Card Protocol과 Firmware에 대해 간략히 설명한다 <그림 6>.

1. Card용 Firmware

eMMC와 SD Card는 Sector 단위의 읽기와 쓰기를 지원한다는 공통점이 있지만, 전자는 내장형 Card로 발전하면서 Code 까지 저장할 수 있도록 Booting, Multiple 파티션, High Priority Read 등의 부가적인 기능을, 후자는 Data를 안전하게 저장하기 위해 필요한 Copyright Protection 기능을 지원하는 차이가 있다. 자세한 Spec.은 각각 관련된 Site에서 확인할 수 있다[5,6]. 현재 eMMC는 v4.4까지, SD는 v3.0까지 표준화가 된 상태이고, 차기 Spec으로 각각 UFS와 UHS가 준비 중이다.

eMMC나 SD Card IO의 특징은 "open-ended"라는 것이다. 즉, Sector 단위로 Data를 읽거나 쓸 때, 시작 주소와 IO Type 만 알려 주고, IO 동작을 시작하며, 언제 끝날지 모르는 상태에서 지속적인 IO를 처리해야 한다. <그림 7>은 eMMC에서 Multiple Block (= Sector)을 Write 하는 것을 그림으로 표현한 것이다. CMD line을 통해서 Command와 인자 (= Multiple Block Write, 시작 Block Address)를 전달하면, Card는 response를 보낸다. 이어서, Data Line (0 ~ 7)을 통해서 Data Block이 Host로부터 Card로 전달된다. Card는 해당 Data Block

5) 현재 32nm 32GB microSD, 32nm 64GB eMMC (= moviNAND)까지 양산하고 있다. http://www.kbench.com/hardware/?no=78629 참고.



〈그림 7〉 eMMC Multiple Block Write Operation

을 Storage에 저장하고 있음을 나타내기 위해 DAT0 라인을 busy 상태로 유지하여 Host가 추가적인 Data Block을 보내지 않도록 한다. Busy 상태가 풀리면 Host는 다음 Data Block을 전송한다. 이렇게 Data Block을 연속해서 Host에서 Card로 전송하며, 맨 마지막 Data Block을 전송하면 STOP command를 Card로 보내서 Data Block의 전송이 완료되었음을 나타낸다. 이러한 동작방식은 제한된 환경에서 최고의 성능을 달성하기 위한 것이지만, Software의 입장에서는 여러 가지 예외적인 상황들을 고려하여야 하는 어려움이 있다. 특히, Random IO를 처리해야 하는 경우 가용 SRAM Size나 Mapping 단위 등에 따라서 성능과 수명에 많은 차이가 있을 수 있다.

2. Card Format Tool

Host에서 Card에 Data를 저장할 수 있는 가장 작은 단위는 Sector지만, Card 내부적으로는 Mapping Unit 이라고 하는 최소 단위로 Mapping을 관리한다. 따라서 Host로부터 Card로 Data를 전송할 때, Mapping Unit의 크기에 Align 되도록 시작 Address나 Data의 길이를 맞추어 주는 것이 성능 면에서 유리하다. Application에서도 이를 고려하면 좋겠지만 Card에 일반적으로 많이 사용하는 FAT File System으로 Format을 할 때, Data Cluster의 시작 위치와 크기를 최소 Mapping Unit 크기의 배수가 되도록 맞추어 주는 것도 한 가지 방법이다. Windows XP 등에서 FAT으로 Format을 할 때 이를 고려하기 시작했으며, 별도로 Format Tool을 제공하기도 한다[6].

3. Card Recovery Tool

NAND의 특성이 악화되면서 Data의 손상 가능성이 높아짐에 따라 File System의 Meta Data가 일부 손상되더라도 파일은 복구할 수 있는 Utility의 필요성이 증가하게 되었다[7]. 또한, 사용자가 실수로 Card를 Format 하는 경우에도 다시 복구하려는 경우가 있다. Card의 용량이 수 GB에서 수십 GB로 커짐에 따라서, 복구 Tool의 기능뿐만 아니라 속도 또한 중요하게 되었다.

6) http://www.sdcard.org/consumers/formatter/eula
7) 기존의 SLC를 사용하던 Card에 비해서 Meta Data의 손상 확률이 상대적으로 높아졌다는 의미이다.

V. SSD 향 Flash Software

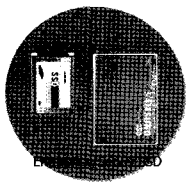
1. SSD 기술 개요 및 삼성 SSD 제품군

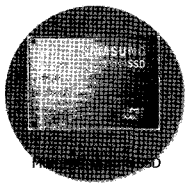
지난 수십 년 간 컴퓨터 시스템 성능의 가장 큰 문제점이었던 저장장치의 성능이 NAND Flash 기반 SSD의 출현으로 인해서 많은 발전을 가져오고 있다. 최근 개발되고 있는 SSD는 기존의 HDD에 비해서 매우 빠른 Sequential 및 Random IO 성능을 보여주고 있으며, 전력 소모량이 HDD에 비해서 월등히 적고, 기계적인 부분이 없어지면서 물리적인 결함성도 해소되었다. 이로 인해 머지않아 컴퓨터 저장장치 시장의 많은 영역에서 SSD가 폭발적으로 사용될 것으로 전망되고 있다. 따라서 많은 NAND Flash 공급자들은 NAND Flash의 대량 수요처로서 SSD를 고려하고 있으며 SSD의 개발을 전략적으로 진행 중이다.

SSD의 시장은 크게 노트북 및 데스크톱을 위한 PC 스토리지 시장과 대용량 서비스를 위한 서버 스토리지 시장으로 나뉜다. PC 스토리지 시장에서 SSD는 HDD와 비교하여 높은 성능 대비 가격 경쟁력을 유지하기 위하여 주로 저가형의 MLC NAND Flash Memory를 이용한 SSD 개발이 진행되며, Random 성능보다는 Sequential 성능에 초점을 맞추어 개발된다. PC 향 SSD의 Interface는 주로 SATAII Interface를 지원함으로써 HDD를 대체하는 저장장치로 사용된다.

서버 스토리지를 위한 SSD는 높은 Sequential Bandwidth 뿐만 아니라, 높은 Random IOPS (Input/Output Operations Per Second)까지 요구하기 때문에 SLC NAND Flash Memory를 이용한 고가의 고성능 SSD개발이 주로 진행되어 왔다. 그러나 서버 향 SSD 역시 가격 경쟁력을 유지하기 위하여 최근에는 MLC NAND Flash를 사용한 고성능 SSD의 개발로 확장되고 있다. 서버 향 SSD의 Interface는 SATAII 뿐만 아니라 SAS(Serial Attached SCSI), PCIe 등 고성능 Interface의 채택이 늘어가고 있다.

<그림 8>은 삼성전자에서 개발하고 있는 SSD 제품군을 응용처 별로 구분하여 나타낸 것이다. 삼성은 Consumer, Server, Industrial Storage 제품군 등 응용에 맞추어 다양한 SSD 개발 제품군을 내놓으며 SSD 제품 업계를 선도





Major Target	Consumer/Laptop Storage	Enterprise/Server Storage	Military/ Industrial Storage
Optimized For	Lowest $$/GB, thin/small FF size	Performance, Power, Reliability	Reliability, Durability, Rugged, Security
Applications	Mobility	OLTP, Office Productivity, Data Mining	Data Recording, Mission Media
Trend	Moving to MLC	Higher MTBF	Encryption

〈그림 8〉 삼성 SSD Product Categories

해 나가고 있다.

SSD의 성능을 결정하는 요소는 크게 SSD Controller와 SSD Firmware라 할 수 있으며, 삼성전자는 세계 1위인 삼성 NAND Flash Memory를 바탕으로, 독자적으로 개발한 고성능 SSD Controller와 이에 최적화된 SSD Firmware를 개발하여 세계 최고 수준의 SSD를 선보이고 있다.

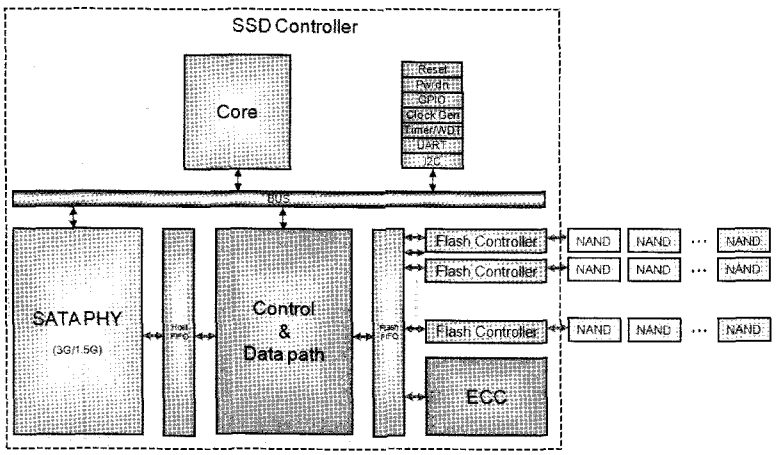
2. SSD Controller

SSD의 성능을 결정하는 가장 중요한 요소 중 하나는 Controller 기술이라 할 수 있다. <그림 9>는 SSD 내부를 Block Diagram으로 나타낸 예이다[8]. SSD Controller는 Multi-Chip과 Multi-Way Interleaving 구조를 위해서 각 Channel 당 전용 Flash Memory Controller가 존재한다. SSD Controller의 Host Interface로는 SATA I, II를 지원하는 SATA Interface가 주로 사용되나, SAS, PCIe 등 다양한 Interface를 가지는 SSD도 존재한다. 이렇게 Host Interface를 통해서 전송된 데이터는 Buffer Manager를 이용해서 DRAM에 저장된 후, Flash Controller로 보내져 실제 NAND Flash Memory에 써지게 된다.

다수의 Host Request를 다수 개의 NAND Flash Controller가 동시에 처리할 수 있기 때문에 Flash Controller의 병렬성을 이용한 성능 향상이 가능하다. 최근의 SSD Controller는 단일 CPU의 계산량 제약으로 인한 Controller의 성능 제약을 극복하기 위하여 Multi-Core를 사용하고 있다.

3. SSD Firmware

SSD Firmware는 주로 HIL (Host Interface Layer), FTL (Flash Translation Layer), FIL (Flash Interface Layer)의 세 가지 Layer로 구성되어 있다. HIL은 Host Command parsing, Host와 Device 사이의 Data 전송, Data Caching 등을 담당하며, FIL은 NAND Flash Memory와의 Interface를 담당하는 모듈로써 각각의 Channel에 dedicated된 NAND Flash Controller에 Interleaving을 고려하여 Read,



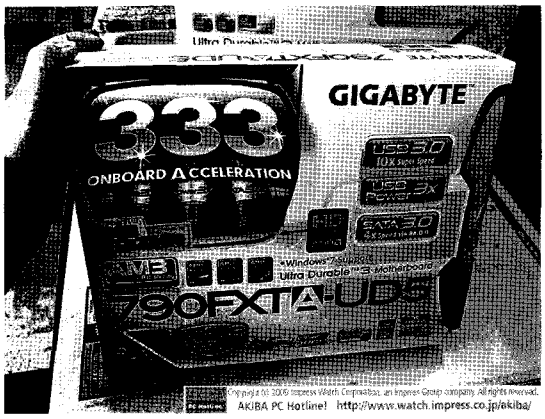
〈그림 9〉 SSD Controller Block Diagram

Write, Erase Operation을 효율적으로 인가하고 처리하는 역할을 담당한다.

FTL은 Flash 기반의 Storage system에서 성능을 결정짓는 중요한 소프트웨어 모듈이며, 이는 SSD에서도 마찬가지이다. FTL은 Wear Leveling, Mapping Management, Garbage Collection, Bad Block Management 등의 알고리즘을 담당한다. SSD의 FTL이 기존의 다른 NAND Flash Storage System의 FTL과 다른 점은 기존의 FTL 디자인에서 최적으로 고려했었던 효율적인 매핑 관리 방법 이외에도 Multi-Channel과 Multi-Way구조를 효율적으로 사용할 수 있는 구조를 가져야 한다는 것이다. 삼성전자 SSD의 FTL은 다중 채널의 Flash Memory를 효율적으로 Control할 수 있는 병렬 매핑 관리, 병렬 NAND Flash 연산 방법 등을 통하여 성능을 최적화하는 구조로 디자인 되어 있다. 이와 더불어 향후 SSD를 위한 NAND Flash Software의 디자인 방향은 컴퓨터의 주 저장장치로서의 역할을 하기 위해서 신뢰성과 긴 수명을 보장하는 방향으로 나아가고 있으며, 이를 달성하기 위한 NAND Flash Software 기술이 다양하게 연구되고 있다.

VI. 향후 NAND Flash Software 기술

Storage Interface의 속도는 빨라지고 있고, NAND Flash의 시장은 커지고 있으며, NAND Flash의 물리적인 특성은 악화되고 있다. NAND Flash Software는 NAND Flash의 물리적인 특성 변화를 최대한 숨기면서 새롭게 요구되는 성능과 기능을 지원해야 하는 숙제를 가지고 있다 <그림 10>.



〈그림 10〉 Storage Interface 변화: 이미 3.x의 시대가 시작되었다

우선, 공정 미세화 대응을 위한 에러보정 기술과 성능 향상 기술이 필요하다. File System의 경우 Microsoft의 exFAT이 기존의 FAT을 서서히 대체할 것으로 예상된다. 따라서 기존에 FAT에 대해 개발되었던 Software들을 exFAT에 대해서도 개발해야 한다[7].

Card의 경우 UHS나 UFS로 스펙이 변하면서 최대 Interface 속도가 300MB/s로 증가함에 따라 이를 지원하기 위한 NAND와 Controller, Software 기술이 필요하다. 특히, UFS는 Code 저장까지 지원해야 하므로 Random Read/Write 성능이 더욱 중요하다. 기존의 OneNAND나 Flex-OneNAND가 지원하던 수준의 Random IO 성능을 Card 형태의 Storage에서 지원하는 기술이 필요하다[2,6].

SSD의 경우 SATAII에서 SAS로의 Host Interface 변화라는 큰 흐름 안에서 다중 Channel, Multi-Core 등 병렬성을 최대한 지원함으로써 높은 IOPS를 달성할 수 있는 SSD Controller Architecture와 운영 Software의 개발이 필요하다.

그 외에 PRAM, FRAM, MRAM 등 차세대 메

모리를 이용한 성능 및 수명 향상방안에 대한 연구 또한 NAND Flash 응용 전반에 걸쳐서 필요하다.

참고문헌

[1] Samsung OneNAND, http://www.samsung.com/global/business/semiconductor/products/fusionmemory/Products_OneNAND.html

[2] Joo Y.S, Choi Y.S, Park C.I, Chung S.W, Chung E.Y, Chang N.H, Demand paging for OneNAND Flash eXecute-in-place, International Conference on Hardware Software Codesign, pp.229-234, 2006.

[3] Samsung Flash Software, http://www.samsung.com/global/business/semiconductor/products/flash/Products_FlashSoftware.html

[4] JEDEC eMMC Card Product Std V4.4

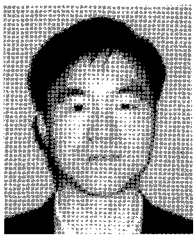
[5] www.sdcard.org

[6] www.jedec.org

[7] http://en.wikipedia.org/wiki/ExFAT

[8] Park C.I., Talawar, P., Won D.S., Jung M.J., Kim S.S., Choi Y.J., A High Performance Controller for NAND Flash-based Solid State Disk (NSSD), Non-Volatile Semiconductor Memory Workshop, pp.17-29, 2006.

저자소개



권 문 상

1995년 2월 서울대학교 컴퓨터공학 학사
1997년 2월 서울대학교 컴퓨터공학 석사
2003년 8월 서울대학교 컴퓨터공학 박사
2003년 4월~2010년 2월 삼성전자 메모리사업부 Flash 개발실 책임연구원
2010년 3월~현재 삼성전자 메모리사업부 Flash 개발실 수석연구원

주관심 분야 : 운영체제, File System, Flash Software

이 상 훈

2004년 2월 홍익대학교 전자전기공학부 학사
2003년 12월~현재 삼성전자 메모리사업부 Flash 개발실 선임연구원

주관심 분야 : Flash Software, Embedded Software

저자소개

정 승 진

1997년 8월 서울대학교 컴퓨터공학 학사
1999년 2월 서울대학교 컴퓨터공학 석사
1999년 3월~2005년 Koscom 전산기술 연구소 연구원
2003년 12월~2005년 9월 삼성전자 메모리사업부 Flash 개발실 선임연구원
2005년 10월~현재 삼성전자 메모리사업부 Flash 개발실 책임연구원

주관심 분야 : Flash Software, Embedded Software

임 승 호

2001년 2월 한국과학기술원 학사
2003년 2월 한국과학기술원 석사
2008년 2월 한국과학기술원 박사
2008년 3월~현재 삼성전자 메모리사업부 Flash 개발실 책임연구원

주관심 분야 : Flash Software, Embedded Software, Operating System