Understanding SSDs with the OpenSSD Platform

Sang-Won Lee & Jin-Soo Kim
Sungkyunkwan University
{swlee, jinsookim}@skku.edu
http://www.openssd-project.org
Outline

- Introduction to flash memory & SSD
- The OpenSSD project
- Jasmine Hardware
- Jasmine Firmware
Introduction to Flash Memory and SSD
Storage Device Metrics

- Capacity ($/GB) : Harddisk >> Flash SSD
- Bandwidth (MB/sec): Harddisk < Flash SSD
- Latency (IOPS): Harddisk << Flash SSD
- Weight/energy/shock resistance/heat & cooling ....
  - Harddisk << Flash SSD

- e.g. Harddisk
  - 7.2K HDD: 50$ / 1TB / 100MB/s / 100 IOPS
  - 15K HDD: 250$/ 72GB / 200MB/s / 500 IOPS
  - The HDD price is said to be proportional to IOPS, not capacity.
NAND Applications

- **USB**
- **Flash Cards**
  - CompactFlash, MMC, SD/miniSD, xD, ...
- **Ubiquitous CE**
  - MP3, Smartphone, Navigator, DTV, Set-Up
- **Hybrid HDD, Intel Turbo Memory, e-MMC**
- **Flash SSDs for PC/Laptop, Enterprise**
NAND Flash Device Organization

Source: Micron Technology, Inc.
HDD vs. Flash Memory Chip

- Erase-before-overwrite
- No mechanical latency
- Asymmetric read/write speed
NAND Flash Types (1)

- **SLC NAND flash**
  - Small block (≤ 1Gb)
  - Large block (≥ 1Gb)

- **MLC NAND flash**
  - 2 bits/cell

- **TLC NAND flash**
  - 3 bits/cell

Source: Micron Technology, Inc.
# NAND Flash Types (2)

<table>
<thead>
<tr>
<th></th>
<th>SLC NAND¹ (small block)</th>
<th>SLC NAND² (large block)</th>
<th>MLC NAND³</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page size (Bytes)</td>
<td>512+16</td>
<td>2,048+64</td>
<td>4,096+128</td>
</tr>
<tr>
<td>Pages / Block</td>
<td>32</td>
<td>64</td>
<td>128</td>
</tr>
<tr>
<td>Block size</td>
<td>16KB</td>
<td>128KB</td>
<td>512KB</td>
</tr>
<tr>
<td>t_R (read)</td>
<td>15 μs (max)</td>
<td>20 μs (max)</td>
<td>50 μs (max)</td>
</tr>
<tr>
<td>t_PROG (program)</td>
<td>200 μs (typ) 500 μs (max)</td>
<td>200 μs (typ) 700 μs (max)</td>
<td>600 μs (typ) 1,200 μs (max)</td>
</tr>
<tr>
<td>tBERS (erase)</td>
<td>2 ms (typ) 3 ms (max)</td>
<td>1.5 ms (typ) 2 ms (max)</td>
<td>3 ms (typ)</td>
</tr>
<tr>
<td>NOP</td>
<td>1 (main), 2 (spare)</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>Endurance Cycles</td>
<td>100K</td>
<td>100K</td>
<td>10K</td>
</tr>
<tr>
<td>ECC (per 512Bytes)</td>
<td>1 bit ECC 2 bits EDC</td>
<td>1 bit ECC 2 bits EDC</td>
<td>4 bits ECC 5 bits EDC</td>
</tr>
</tbody>
</table>

¹ Samsung K9F1208X0C (512Mb) ² Samsung K9K8G08U0A (8Gb) ³ Micron Technology Inc.
HDDs vs. SSDs (1)

2.5” HDD    Flash SSD
(101x70x9.3mm)

1.8” HDD    Flash SSD
(78.5x54x4.15mm)
## HDDs vs. SSDs (2)

<table>
<thead>
<tr>
<th>Feature</th>
<th>SSD (Samsung)</th>
<th>HDD (Seagate)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>MMDOE56G5MXP (PM800)</td>
<td>ST9500420AS (Momentus 7200.4)</td>
</tr>
<tr>
<td>Capacity</td>
<td>256GB (16Gb MLC x 128, 8 channels)</td>
<td>500GB (2 Discs, 4 Heads, 7200RPM)</td>
</tr>
<tr>
<td>Form factor</td>
<td>2.5” Weight: 84g</td>
<td>2.5” Weight: 110g</td>
</tr>
<tr>
<td>Host interface</td>
<td>Serial ATA-2 (3.0 Gbps)</td>
<td>Serial ATA-2 (3.0 Gbps)</td>
</tr>
<tr>
<td></td>
<td>Host transfer rate: 300MB</td>
<td>Host transfer rate: 300MB</td>
</tr>
<tr>
<td>Power consumption</td>
<td>Active: 0.26W</td>
<td>Active: 2.1W (Read), 2.2W (Write)</td>
</tr>
<tr>
<td></td>
<td>Idle/Standby/Sleep: 0.15W</td>
<td>Idle: 0.69W, Standby/Sleep: 0.2W</td>
</tr>
<tr>
<td>Performance</td>
<td>Sequential read: Up to 220 MB/s</td>
<td>Power-on to ready: 4.5 sec</td>
</tr>
<tr>
<td></td>
<td>Sequential write: Up to 185 MB/s</td>
<td>Average latency: 4.17 msec</td>
</tr>
<tr>
<td>Measured performance¹</td>
<td>Sequential read: 176.73 MB/s</td>
<td>Sequential read: 86.07 MB/s</td>
</tr>
<tr>
<td></td>
<td>Sequential write: 159.98 MB/s</td>
<td>Sequential write: 84.64 MB/s</td>
</tr>
<tr>
<td></td>
<td>Random read: 10.56 MB/s</td>
<td>Random read: 0.61 MB/s</td>
</tr>
<tr>
<td></td>
<td>Random write: 2.93 MB/s</td>
<td>Random write: 1.28 MB/s</td>
</tr>
<tr>
<td>Price²</td>
<td>539,190 won</td>
<td>80,400 won</td>
</tr>
</tbody>
</table>

Flash SSD Block Diagram

- Pata/ Sata/ Scsi/ SAS
- More Powerful CPU, Bigger SDRAM
- Parallelism / Interleaving for Large Bandwidth
Storage Abstraction

- Abstraction given by block device drivers:

  - Identify(): returns $N$
  - Read (start sector #, # of sectors)
  - Write (start sector #, # of sectors, data)

Source: Sang Lyul Min (Seoul National Univ.)
What is FTL?

▪ A software layer to make NAND flash fully emulate traditional block devices (or disks)

Source: Zeen Info. Tech.
Roles of FTL

- **For performance**
  - Indirect mapping (address translation)
  - Garbage collection
  - Over-provisioning
  - Hot/cold separation
  - Interleaving over multiple channels/flash chips/planes
  - Request scheduling
  - Buffer management
  - ...

- **For Reliability**
  - Bad block management
  - Wear-leveling
  - Power-off recovery
  - Error correction code (ECC)
  - ...

- **Other Features**
  - Encryption
  - Compression
  - Deduplication
  - ...
Overwrites in Flash Memory

- Naïve approach

  - New write (2K): 0.2ms
  - Overwrite (2K)
    - 63 2K-reads = 6.3ms
    - 63 2K-writes = 12.6ms
    - 1 2K-write = 0.2ms
    - 1 erase = 1.5ms
    - Total = 20.6ms
Various Mapping Techniques

▪ Block mapping
  • Logical block vs. physical block

▪ Page mapping
  • Logical page vs. physical block

▪ Hybrid mapping
  • Block mapping + page mapping
Block Mapping

- Each table entry maps one block
- Small RAM usage
- Inefficient handling of small writes
Page Mapping

- Most flexible
- Efficient handling of small writes
- Large memory footprint
  - 32MB for 32GB MLC (4KB page)
- Sensitive to the amount of reserved blocks
- Performance affected as the system ages
Hybrid Mapping: An Example

Based on “Locality” and “Log Structured File System” idea

- Every 2K writes can be buffered in 0.2 ms, not in 20.6 ms
- When a log block is full, it should be merged
  - “Merge” operation
- BAST: Block Associative Sector Translation
- Log block thrashing
- Other hybrid scheme: FAST, Superblock, LAST, .....
Roles of SSDs (in Enterprise)

▪ Special purpose disk
  • Swap device, redo log device, temporary tablespace

▪ Complementing disk
  • Extended {buffer / disk} cache
  • A new component in memory hierarchy
  • E.g. SSD as faster disk, separate layer between ram and disk

▪ Replacing disk
  • IOPS Booster
  • “Flash is disk, disk is tape, and tape is dead” (Gray)
Current SSD Usage Trends

**Oracle for TPC-C (2010 Dec.)**
- Oracle Exadata
- **Total cost:** 49M
  - Storage: 23M
  - Sun Flash Array: 22M
  - 720 * 2TB 7.2K HDD: 0.7M

- IBM SSD Buffer (VLDB 10)
- MS SQL Server (SIGMOD 11)
Some Future Trends

- **FlashSSD based In-Storage Processing (ISP)**
  - Promising in data-intensive applications: OLAP, search, map/reduce, scientific data
    - E.g. Scan/filtering, hashing, sorting

- **vs. Disk based ISP History**
  - Database machine (Boral and DeWitt)
  - Active disk (Eric Riedel et al.)
    - Processing power in storage
    - CMU, {HP, Sun} + Oracle: Oracle Exadata
The OpenSSD Project
What’s the OpenSSD Project?

- An initiative to promote research and education on the SSD technology
- Provides an “OpenSSD platform” for developing open-source SSD firmware
- Started as a collaborative work between SKKU and Indilinx
Why OpenSSD?

- No more simulations
- Broaden research horizon
- Educate people with a real system
- Share expertise
- Just for fun!
- ...

http://www.openssd-project.org
Jasmine OpenSSD Platform

- A reference implementation of SSD based on the Indilinx Barefoot controller
- Sample FTL source codes
- Technical documents
Jasmine Users

- 31 sets shipped to 10 institutions (16 labs)

- 1 set shipped abroad
  - RecoverMyFlashDrive.com

- Inquiries from US, China, Netherlands, ...

- Currently preparing more sets
Jasmine Firmware

- The firmware source code v1.0.0 released on April 7, 2011 by Indilinux under the GPL
- The latest version: v1.0.6 (June 25, 2011)
- Available at http://www.openssd-project.org/wiki/Downloads

- Total firmware downloads: 337 times (As of June 28, 2011)
Jasmine Resources

- FAQs
- Forums
- Board schematics
- FTL Developer’s Guide
- Barefoot Controller Technical Reference
- Contributions from community
  - Developer Information (by Jeremy Brock), ...
OpenSSD Forum
1st OpenSSD Workshop

- May 11, 2011 @ Sungkyunkwan Univ.
- 53 participants from 12 institutions
Jasmine Hardware
A Real SSD

Source: benchmarkreviews.com
Jasmine Board

- Indilinx Barefoot SSD controller
- SATA 2.0 interface
- 64MB SDRAM
- Current Measurement points
- JTAG port
- UART port
- NAND flash module (32GB/module)
- Mictor connectors for logic analyzers
Indilinx Barefoot Controller
Barefoot Features

- ARM7TDMI-S running up to 87.5MHz
- 96KB SRAM
- SATA 2.0 (3Gbps)
- Mobile SDRAM controller up to 64MB
- NAND flash 8/12/16-bit BCH ECC per sector
- SDRAM 2-byte RS ECC per 128 +4 bytes
- Maximum 64CE’s (4 channels, 8 banks/ch)
- System bus running up to 175MHz
SDRAM & Flash

- Mobile SDRAM
  - Samsung 64MB (subject to change)

- NAND Flash
  - Samsung 64GB (subject to change) in two NAND flash modules
  - Four K9LCG08U1M (8GB) packages / module
  - 32Gb (4GB) per die, 2 CE signals / package (Dual Die Package)
Debugging/Monitoring Aids

- JTAG
- UART
- 1 LED and 6 GPIO pins
- Mictor connectors to NAND flash signals for logic analyzers
- Separate current measurement points for core, IO, SDRAM, and NAND
# NAND Flash Configuration

<table>
<thead>
<tr>
<th>Ch. A</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Ch. B</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bank B0</td>
<td>Bank B1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ch. C</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bank C0</td>
<td>Bank C1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ch. D</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bank D0</td>
<td>Bank D1</td>
</tr>
</tbody>
</table>
Bank Configuration

- 16-bit IO/bank
- Virtual page size  
  = Page size * 2
- Virtual page size  
  = Page size * 4  
  (for 2-plane mode)
SDRAM Layout

- The location & size of each region fixed at init time
- Buffer is segmented in fixed size (4~32KB)
- Read/write buffers in circular buffer scheme
- 4-byte ECC parity added to every 128B
NAND Flash Controller

Flash Command Port

<table>
<thead>
<tr>
<th>CMD</th>
<th>BANK</th>
<th>OPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMA_ADDR</td>
<td>DMA_CNT</td>
<td>COLUMN</td>
</tr>
<tr>
<td>ROW_0_L</td>
<td>ROW_0_H</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROW_31_L</td>
<td>ROW_31_H</td>
<td>DST_COL</td>
</tr>
<tr>
<td>DST_ROW</td>
<td>ISSUE</td>
<td></td>
</tr>
</tbody>
</table>

Waiting Room

<table>
<thead>
<tr>
<th>CMD</th>
<th>BANK</th>
<th>OPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bank Status Ports

<table>
<thead>
<tr>
<th>CMD</th>
<th>OPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

(cf) Set BANK = 0x3F for Autoselect Mode
Example: Reading a VPage

```
SETREG (FCP_CMD, FC_COL_ROW_READ_OUT);
SETREG (FCP_DMA_CNT, SECTORS_PER_PAGE * BYTES_PER_SECTOR);
SETREG (FCP_COL, 0);
SETREG (FCP_DMA_ADDR, RD_BUF_PTR(g_ftl_read_buf_id));
SETREG (FCP_OPTION, FO_P | FO_E | FO_B_SATA_R);
SETREG (FCP_ROW_L(bank), row);
SETREG (FCP_ROW_H(bank), row);
SETREG (FCP_BANK, bank);
while ((GETREG(WR_STAT) & 0x00000001) != 0);
SETREG (FCP_ISSUE, NULL);
```
Flash Operation Parallelism

command issued to A1

A1 accepts the command

A0 completes the command

command issued to A0

A0 accepts the command

A1 completes the command

BSP_FSM of Bank A0

<table>
<thead>
<tr>
<th>IDLE</th>
<th>BUSY</th>
<th>IDLE</th>
</tr>
</thead>
</table>

RB signal of Bank A0

<table>
<thead>
<tr>
<th>READY</th>
<th>BUSY (cell operation)</th>
<th>READY</th>
</tr>
</thead>
</table>

BSP_FSM of Bank A1

<table>
<thead>
<tr>
<th>IDLE</th>
<th>BUSY</th>
<th>IDLE</th>
</tr>
</thead>
</table>

RB signal of Bank A1

<table>
<thead>
<tr>
<th>READY</th>
<th>BUSY (cell operation)</th>
<th>READY</th>
</tr>
</thead>
</table>

Owner of Channel A (= who is doing an interface operation)

<table>
<thead>
<tr>
<th>FREE</th>
<th>A0</th>
<th>A1</th>
<th>FREE</th>
<th>A1</th>
<th>FREE</th>
<th>A0</th>
<th>FREE</th>
</tr>
</thead>
</table>
Buffer Management

- Buffer management and flow control in hardware

Write buffer

BM write limit (HW register)

FTL write pointer (FW global variable)

SATA write pointer (HW register)

being written to flash

Read buffer

SATA read pointer (HW register)

BM read limit (HW register)

FTL read pointer (FW global variable)
Memory Utility

- Hardware accelerator for memory operations
  - Initializing a memory region with a given value
  - Copying a memory between SRAM & DRAM
  - Finding a bit pattern
  - Finding a given value
  - Finding a min/max value
  - Reading/writing DRAM with ECC handling
  - ...

Jasmine Firmware
Firmware Source Tree

- Build directory for Sourcery G++ toolchain
- Build directory for ARM RVDS toolchain
- Sample FTL: DummyFTL
- Sample FTL: GreedyFTL
- Sample FTL: TutorialFTL
- Header files
- Firmware installer utility (installer.exe)
- SATA protocol handling
- Initialization, flash control, etc.
- Testing code
Toolchain

- For building firmware
  - ARM RealView Development Suite (RVDS) 3.0 or higher ($$$$$)
  - CodeSourcery G++ Lite Edition for ARM EABI
  - CrossWorks for ARM from Rowley Associates ($$$)

- For building installer utility (install.exe)
  - Microsoft Visual Studio 2005 or Microsoft Visual C++ 2010 Express Edition
Development Setup
Building Firmware (1)

- Set compile options (.\include\jasmine.h)

```c
#define OPTION_2_PLANE 1 // 1: use 2-plane mode
#define OPTION_ENABLE_ASSERT 0 // 1: enable ASSERT()
#define OPTION_FTL_TEST 0 // 1: FTL test w/o SATA
#define OPTION_UART_DEBUG 0 // 1: enable UART msgs
#define OPTION_SLOW_SATA 0 // 1: SATA1 (1.5Gbps)
#define OPTION_SUPPORT_NCQ 1 // 1: NCQ support
#define OPTION_REDUCED_CAPACITY 0 // 1: for testing
```
Building Firmware (2)

- Specify the target FTL (.build_gnu/Makefile)

```
FTL   = [dummy|tutorial|greedy]
...
```

- Compile the firmware source

```
> cd build_gnu
> build.bat
```

- Firmware binary: .build_gnu/firmware.bin
Building Firmware (3)

- Compile the installer utility
  - Open ./installer/installer.sln in MS Visual Studio
  - Build the project
  - Move ./installer/install.exe to ./build_gnu
Installing Firmware (1)

▪ Boot the Jasmine board in “Factory mode”
  • J2 jumper need to be set

▪ Install firmware

> cd build_gnu
> install.exe
Installing Firmware (2)
Available FTLs

- **TutorialFTL** (by Indilinx)
  - Page-mapping FTL
  - No garbage collection
  - NAND flash initialized at power-on
- **GreedyFTL** (by SKKU VLDB Lab.)
  - Page-mapping FTL with garbage collection
  - Data survive a normal power-off
- **DummyFTL** (by Indilinx)
  - For measuring SATA and DRAM speed
Overall Flow

1. Board Initialization
2. Call ftl_open()

- EventQ?
  - Empty
    - No
      - Process slow cmd
    - Yes
      - Slow cmd?
        - No
          - Read?
            - Yes
              - Call ftl_read()
            - No
              - Call ftl_write()
        - Yes
          - Not Empty
            - EventQ?
              - Empty
                - No
                  - Process slow cmd
              - Not Empty
                - EventQ?
                  - Empty
                    - No
                      - Process slow cmd
                  - Not Empty
                    - EventQ?
Debugging Firmware

▪ On-board LED
▪ UART
▪ ARM JTAG ICE + ARM RVDS
▪ USB ARM JTAG devices + CodeWorks
▪ Monitoring Signals with Logic Analyzers
Call For Participation

- Welcome any contributions from
  - SSD manufacturers
  - NAND flash vendors
  - Research groups
  - Individual developers
  - ...

- You can create and edit most of pages after registration
Thank You!