Tutorial FTL

Joohyung Park
Computer Systems Laboratory
Sungkyunkwan University
http://csl.skku.edu
NAND Flash Physical Layout
NAND Flash Logical Layout

8 Way

4 Channel
NAND Flash Logical Layout

• Banks share the same IO bus
• Each bank can perform cell operation in parallel
  • Cell to Page Buffer operation, Block erase
• Barefoot has only 4 R/B signal inputs per channel
  • (0,4) (1,5) (2,6) (3,7) chip binding in a same channel
  • Maximum 4 way interleaving
NAND Flash Memory

- **Page Buffer**: 8KB + 640B
- **Page**: 8KB + 640B
- **Block**: 1MB + 80KB
- **Data**: 8KB
- **Spare**: 640B

Die 0 (“Low”) and Die 1 (“High”) with two planes (Plane 0 and Plane 1) per die.
NAND Flash Memory

- K9LCG08U1M (Dual die)
  - Samsung 35 nm 2-bit MLC flash
  - 16 sectors per page (8 KB + 640 B)
  - 128 pages per block (1 MB + 80 KB)
  - 4096 + 56 blocks per die
  - Page read : 250 us, Page program : 1.3 ms, Block erase : 1.5 ms

<table>
<thead>
<tr>
<th>K</th>
<th>9</th>
<th>X</th>
<th>X</th>
<th>X</th>
<th>X</th>
<th>X</th>
<th>X</th>
<th>X</th>
<th>X</th>
<th>X</th>
<th>X</th>
<th>X</th>
<th>X</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
</tr>
</tbody>
</table>

3. Small Classification
(SLC : Single Level Cell, MLC : Multi Level Cell, SM : SmartMedia, S/B : Small Block)

- K : SLC Die Stack
- L : MLC DDP
- M : MLC DSP
- N : SLC DSP
- O : 3bit MLC ODP

4-5. Density
- 12 : 512M
- 32 : 32M
- 64 : 64M
- 2G : 2G
- 1G : 1G
- 8G : 8G
- 16G : 16G
- 40 : 4M
- 80 : 8M
- 4G : 4G
- 32G : 32G
- 256G : 256G
- 512G : 512G
- 64G : 64G
- 128G : 128G
- 24G : 24G
- 8G : 8G
- 00 : NONE
NAND Flash Operations

• Page read
  • Cell -> Page Buffer -> RAM

• Page program
  • RAM -> Page Buffer -> Cell

• Page copy-back
  • Cell (Src) -> Page Buffer -> Cell (Dst)

• Block erase
Bank/Row Abstraction

- 2 Plane operation, 16 bit IO program
  - 33 44 55 66 ............ 77 88 99 AA ............

![Bank/Row Abstraction Diagram](image)
Quiz

• Answer those questions with evidence
• The size of pages managed by mapping table
  • Hint: SECTORS_PER_PAGE
• No channel, no way, but banks. How could it be?
  • Hint: include/bank.h
• Page allocation hierarchy in CWDP
  • Hint: include/bank.h + virtual page
Tutorial FTL

• ./ftl_tutorial
  • ftl.c, ftl.h

• Page Mapping FTL
  • Write data from DRAM to NAND
  • Read data from NAND to DRAM
  • But no garbage collection
Page Mapping Table

• LPN to PPN map
  • LPN: Logical Page Number
    • LPN = LBA / Sectors Per Page
  • PPN: Physical Page Number

<table>
<thead>
<tr>
<th>Index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>...</th>
<th>2097151</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>100</td>
<td>256</td>
<td>0</td>
<td>INVALID</td>
<td>...</td>
<td>20000</td>
</tr>
</tbody>
</table>

```c
static UINT32 get_physical_address(UINT32 const lpage_addr)
{
    // Page mapping table entry size is 4 byte.
    return read_dram_32(PAGE_MAP_ADDR + lpage_addr * sizeof(UINT32));
}

static void update_physical_address(UINT32 const lpage_addr, UINT32 const new_bank, UINT32 const new_row)
{
    write_dram_32(PAGE_MAP_ADDR + lpage_addr * sizeof(UINT32), new_bank * PAGES_PER_BANK + new_row);
}
NAND Flash Controller

- To issue NAND Flash operation
Flash Command Port Registers

- Defined in ./include/flash.h

<table>
<thead>
<tr>
<th>Registers</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCP_ISSUE</td>
<td>87</td>
</tr>
<tr>
<td>FCP_CMD_ID</td>
<td>86</td>
</tr>
<tr>
<td>FCP_DST_ROW_H</td>
<td>85</td>
</tr>
<tr>
<td>FCP_DST_ROW_L</td>
<td>84</td>
</tr>
<tr>
<td>FCP_DST_COL</td>
<td>83</td>
</tr>
<tr>
<td>FCP_ROW31_H</td>
<td>82</td>
</tr>
<tr>
<td>FCP_ROW31_L</td>
<td>81</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>FCP_ROW0_H</td>
<td>20</td>
</tr>
<tr>
<td>FCP_ROW0_L</td>
<td>19</td>
</tr>
<tr>
<td>FCP_COL</td>
<td>18</td>
</tr>
<tr>
<td>FCP_DMA_CNT</td>
<td>17</td>
</tr>
<tr>
<td>FCP_DMA_ADDR</td>
<td>16</td>
</tr>
<tr>
<td>FCP_OPTION</td>
<td>15</td>
</tr>
<tr>
<td>FCP_BANK</td>
<td>14</td>
</tr>
<tr>
<td>FCP_CMD(FCP_BASE)</td>
<td>13</td>
</tr>
<tr>
<td>WR_BANK</td>
<td>12</td>
</tr>
<tr>
<td>WR_STAT</td>
<td>11</td>
</tr>
</tbody>
</table>
FCP Setting

```c
void nand_page_ptread_to_host(UINT32 const bank, UINT32 const vblock,
 UINT32 const page_num, UINT32 const sect_offset, UINT32 const num_sectors)
{
    ...
    row = (vblock * PAGES_PER_BLK) + page_num;

    SETREG(FCP_CMD, FC_COL_ROW_READ_OUT);
    SETREG(FCP_DMA_ADDR, RD_BUF_PTR(g_ftl_read_buf_id));
    SETREG(FCP_DMA_CNT, num_sectors * BYTES_PER_SECTOR);

    SETREG(FCP_COL, sect_offset);
#if OPTION_FTL_TEST == TRUE
    SETREG(FCP_OPTION, FO_P | FO_E);
#endif
    ...
    SETREG(FCP_ROW_L(bank), row);
    SETREG(FCP_ROW_H(bank), row);

    g_ftl_read_buf_id = (g_ftl_read_buf_id + 1) % NUM_RD_BUFFERS;

    ...
    flash_issue_cmd(bank, RETURN_ON_ISSUE);
}
```
void flash_issue_cmd(UINT32 const bank, UINT32 const sync)
{
    UINT32 rbank = REAL_BANK(bank);

    SETREG(FCP_BANK, rbank);

    while (((GETREG(WR_STAT) & 0x00000001) != 0);

    SETREG(FCP_ISSUE, NULL);

    if (sync == RETURN_ON_ISSUE)
        return;

    while (((GETREG(WR_STAT) & 0x00000001) != 0);

    if (sync == RETURN_ON_ACCEPT)
        return;

    while (_BSP_FSM(rbank) != BANK_IDLE);
}
Quiz

• Replace some part of tutorial ftl with flash wrapper functions in target_spw/flash_wrapper.c as possible as you can.
• Capture the result of mounting modified openssd
• Send me one compacted file(GroupId.zip) including
  • Answer sheet(txt) of upper 4 quizzes
  • Modified code(ftl.c)
  • Success of mount(jpg)
Assignment

• Read the linked paper from 36:4(1.4. Characteristics) to 36:6(2.1. Functionalities)
  • Only 3 pages, not the whole paper

• E-mail me(joohyungpark@csl.skku.edu) with the answers within one sentence for each question below
  • Why is ‘in-place update’ unavailable?
  • What is the functionality of FTL to solve the ‘in-place update’?
  • Why is ‘garbage collection’ needed?
  • Guess the reason why there is a speed difference between read and write

• Until 23h 59m 59s, Fri 3/25
Any Questions?