Project #1: NAND Simulator

Prof. Jinkyu Jeong (jinkyu@skku.edu)
TA -- Minwoo Ahn (minwoo.ahn@csl.skku.edu)
TA -- Donghyun Kim (donghyun.kim@csl.skku.edu)

Computer Systems Laboratory
Sungkyunkwan University
http://csl.skku.edu
NAND Flash Memory

• **Erase-before-write**

• **Bulk erase**
  – Program unit: page
  – Erase unit: block

• **Sequential write in a block**
Flash Translation Layer

• A software layer to make NAND flash fully emulate traditional block devices
Project Outline (without board)

- Project #1: NAND simulator
- Project #2: Greedy Page mapping FTL on simulator
- Project #3: Other Page mapping FTLs on simulator
- Project #4: Multi-Stream SSD

Diagram:
- NAND Storage
  - Controller
    - FTL
      - Sector Translation
      - Block Management
      - Low Level Driver
  - Flash Memory
Project 1

• Develop a NAND simulator
  – Simulate NAND flash memory using host DRAM
  – 4B for data / 4B for spare
  – Functions to implement:
    • nand_init()
    • nand_read()
    • nand_write()
    • nand erase()
    • nand_blockdump()
  – The skeleton code is available at course homepage
nand_init()

• nand_init(nblks, npages)
  – Description
    • Initialize your own NAND flash memory using DRAM
    • Initial state of the flash memory is ‘all-blks-erased’
    • If success, print initialized information of the flash memory
    • If not, print appropriate error message (reason for the error)
  – Argument
    • nblks : the total # of blocks (should be > 0)
    • npages : # of pages per block (should be > 0)
  – Return value
    • Return 0 on success
    • Return -1 on errors
nand_write()

- nand_write(blk, page, data, spare)
  - Description
    - Write ‘data’ and ‘spare’ to the memory pointed by ‘blk’ and ‘page’
    - If success, print written data and spare
    - If not, print appropriate error message (reason for the error)
  - Argument
    - blk, page : address of the flash memory
    - data, spare : data to store
  - Return value
    - Return 0 on success
    - Return -1 on errors
nand_read()

- nand_read(blk, page, data, spare)
  - Description
    • Read ‘data’ and ‘spare’ from the memory pointed by ‘blk’ and ‘page’
    • If success, print read data and spare
    • If not, print appropriate error message (reason for the error)
  - Argument
    • blk, page : address of the flash memory
    • data, spare : data to load
  - Return value
    • Return 0 on success
    • Return -1 on errors
nand_erase()

• nand_erase(blk)
  – Description
    • Erase the NAND memory block ‘blk’
    • If success, print appropriate message with ‘blk’
    • If not, print appropriate error message (reason for the error)
  – Argument
    • blk : Address of the NAND flash memory
  – Return value
    • Return 0 on success
    • Return -1 on errors
nand_blkdump()

• nand_blkdump(blk)
  – Description
    • Dump the contents of the flash memory block ‘blk’ (for debug)
    • If success, print appropriate message with ‘blk’
    • If not, print appropriate error message (reason for the error)
  – Argument
    • blk : Address of the NAND flash memory
  – Return value
    • Return 0 on success
    • Return -1 on errors
Sample Output

```plaintext
sally@dylee:~/nandsim$ ./nandsim
NAND: 8 blocks, 8 pages per block, 64 pages
write(3,0): data = 0x00000000, spare = 0x00000000
write(3,1): data = 0x00000001, spare = 0x00010000
write(3,2): data = 0x00000002, spare = 0x00020000
write(3,3): data = 0x00000003, spare = 0x00030000
write(3,4): data = 0x00000004, spare = 0x00040000
write(3,5): data = 0x00000005, spare = 0x00050000
write(3,6): data = 0x00000006, spare = 0x00060000
write(3,7): data = 0x00000007, spare = 0x00070000
read(3,7): data = 0x00000007, spare = 0x00070000
read(3,6): data = 0x00000006, spare = 0x00060000
read(3,5): data = 0x00000005, spare = 0x00050000
read(3,4): data = 0x00000004, spare = 0x00040000
read(3,3): data = 0x00000003, spare = 0x00030000
read(3,2): data = 0x00000002, spare = 0x00020000
read(3,1): data = 0x00000001, spare = 0x00010000
read(3,0): data = 0x00000000, spare = 0x00000000
write(4,0): data = 0x00000000, spare = 0x00000000
write(4,0): failed, the page was already written
write(4,0): failed, the page is not being sequentially written
read(4,3): failed, trying to read an empty page
read(0,0): failed, trying to read an empty page
read(7,0): failed, trying to read an empty page
erase(3): block erased
erase(4): block erased
erase(0): failed, trying to erase a free block
write(3,0): data = 0x00000000, spare = 0x00000000
write(3,1): data = 0x00100000, spare = 0x00000001
write(3,2): data = 0x00200000, spare = 0x00000002
write(3,3): data = 0x00300000, spare = 0x00000003
write(3,4): data = 0x00400000, spare = 0x00000004
write(3,5): data = 0x00500000, spare = 0x00000005
write(3,6): data = 0x00600000, spare = 0x00000006
write(4,1): data = 0x00000000, spare = 0x00000000
read(4,1): data = 0x00000000, spare = 0x00000000
read(4,0): data = 0x00000000, spare = 0x00000000
write(4,-1): failed, invalid page number
read(4,-1): failed, invalid page number
write(4,-1): failed, invalid page number
read(4,-1): failed, invalid page number
write(-1,0): failed, invalid block number
read(-1,0): failed, invalid block number
write(8,0): failed, invalid block number
read(8,0): failed, invalid block number
erase(1): failed, invalid block number
erase(8): failed, invalid block number
erase(1000000): failed, invalid block number
Blk 0: FREE
Blk 1: FREE
Blk 2: FREE
Blk 3: Total 8 page(s) written
Page 0: data = 0x00000000, spare = 0x00000000
Page 1: data = 0x01000000, spare = 0x00000000
Page 2: data = 0x02000000, spare = 0x00000000
Page 3: data = 0x03000000, spare = 0x00000000
Page 4: data = 0x04000000, spare = 0x00000000
Page 5: data = 0x05000000, spare = 0x00000000
Page 6: data = 0x06000000, spare = 0x00000000
Page 7: data = 0x07000000, spare = 0x00000000
Blk 4: Total 2 page(s) written
Page 0: data = 0x00000000, spare = 0x00000000
Page 1: data = 0x00000000, spare = 0x00000000
Blk 5: FREE
Blk 6: FREE
Blk 7: FREE
```
Miscellaneous

• Recommended environment: Linux (Ubuntu is ok!)
  – You can do it in Windows, but be sure that your work also runs in Linux (I’ll score all the works only in Linux)

• Personal Project

• Submit to the e-mail (minwoo.ahn@csl.skku.edu)
  – Your submission status will be noticed in course homepage

• Late penalty: -20% / day
Any Questions?