Reliability Issues/Test

Sejun Kwon(sejun000@csl.skku.edu)
Computer Systems Laboratory
Sungkyunkwan University
http://csl.skku.edu
Contents

- ECC check
- Power-Off Recovery
- Bad Block
- FTL Test
DRAM Controller

- **DRAM ECC**
  - Check and correct DRAM bit errors
  - 128 Byte + 4 Byte ECC Parity

- **Memory Utility**
  - Support data transmission from/to DRAM
    - SRAM to DRAM, DRAM to SRAM, DRAM to DRAM
  - `./include/mem_util.h`
DRAM Controller

- Write 8 Byte data from SRAM to DRAM
Power-Off Recovery

- Power-off leads the data loss
  - Userdata that resides in write buffer
  - Metadata that resides in SRAM and DRAM
    - Page-level mapping table

- Recover mapping table on next power-on
  - On program operation, store LPN into spare area
  - By scanning spare area, mapping table can be reconstructed
Power-Off Recovery

- Store and recover metadata

* Power off

SRAM

DRAM

* Power on

SRAM

DRAM

Flash

Meta block
Bad Block

- Bad block cannot be used
  - Data in bad block is crashed
  - Any operation cannot be operated
Bad Block

- Initial bad block
  - Bad block caused by problem in manufacturing process

- Runtime bad block
  - Bad block caused by wearing out
Bad Block

- **Bad block detecting**
  - Initial bad block
    - Scan block #0
  - Runtime bad block
    - Invoke interrupt service

- **Bad block handling**
  - Avoid using bad block
  - Remapping to reserved block
Bad Block

- **ftl_isr()**

```c
// BSP interrupt service routine
void ftl_isr(void)
{
    UINT32 bank;
    UINT32 bsp_intr_flag;

    uart_print("BSP interrupt occurred...");
    // interrupt pending clear (ICU)
    SETREG(APB_INT_STS, INTR_FLASH);

    for (bank = 0; bank < NUM_BANKS; bank++) {
        while (BSP_PSM(bank) != BANK_IDLE);
        // get interrupt flag from BSP
        bsp_intr_flag = BSP_INTR(bank);

        if (bsp_intr_flag == 0) {
            continue;
        }

        UINT32 fc = GETREG(BSP_CMD(bank));
        // BSP clear
        CLR_BSP_INTR(bank, bsp_intr_flag);

        // interrupt handling
        if (bsp_intr_flag & FIRQ_DATA_CORRUPT) {
            uart_print("BSP interrupt at bank: 0x%X", bank);
            uart_print("FIRQ_DATA_CORRUPT occurred..."));
        }

        if (bsp_intr_flag & (FIRQ_BADBLK_H | FIRQ_BADBLK_L)) {
            uart_print("BSP interrupt at bank: 0x%X", bank);
            if (fc == FC_COL_ROW_IN_PROG || fc == FC_IN_PROG || fc == FC_PROG) {
                uart_print("Find runtime bad block when block program...");
            }
        } else {
            uart_print("Find runtime bad block when block erase...block #: %d", GETREG(BSP_ROW_H(bank)) / PAGES_PER_BLK);
            ASSERT(fc == FC_ERASE);
        }
    }
}
```
FTL test

▪ In the host PC
  • Send read/write command to jasmine board.

▪ In the Firmware(Next Week)
  • Use ftl_test()

▪ Real task
  • File Copy and Read
  • Kernel Compile
  • Compression
  • Sort
FTL test

- include <sys/types.h>
- include <sys/stat.h>
- include <fcntl.h>
- include <unistd.h>

int main()

  int fd = open("/dev/sdb", O_RDWR|O_SYNC);
  int i;

  if(fd == -1) return 0;

  char ch[256]="I am Sejun Kwon, Computer Systems Laboratory");
  char ch_out[256];

  lseek(fd, 10, SEEK_SET);
  write(fd, ch, 10);
  lseek(fd, 13, SEEK_SET);
  read(fd, ch_out, 9);
  for(i=0;i<9;i++)
    putchar(ch_out[i]);
  return 0;
FTL test

- We can open device as file.
  - Confirm device name with “fdisk –l”

- We can select LBA(logical block address) with lseek.
FTL test

- In-Firmware Test
  - Not communicate SATA
  - DRAM <-> NAND
  - In jasmine.h
    - #define OPTION_FTL_TEST 1
    - #define OPTION_UART_DEBUG 1
FTL test

- **Kernel Compile**
  - Get from [www.kernel.org](http://www.kernel.org)
  - make menuconfig
  - make –j2

- **Compression**
  - Random Read/Sequential Write
  - Data Compare
  - Zlib, LZO, LZ4
FTL test

- Sort
  - Quick Sort
    - Swap -> In place update
    - Occur Random Read/Write
  - Merge Sort
    - Sequential Read, Sequential Write
Any Questions?