Dummy FTL

Dong-Yun Lee(dongyun.lee@csl.skku.edu)
Computer Systems Laboratory
Sungkyunkwan University
http://csl.skku.edu
Troubleshooting

- Q. Compile errors occurred in eabi-related libraries
  - A. Modify all the lines in uart_printf

- Q. ERROR: cannot open firmware.bin for reading
  - A. You have not done build yet, or build is failed
Contents

▪ Deep dive to the Jasmine & codes
  • Read / write command flow
  • Host interface layer
    - SATA controller
    - Buffer manager
  • Flash translation layer

▪ Intro. to DummyFTL
  • Request handling in Dummy FTL (code-level)

▪ Appendix: Measuring performance with IOmeter
Hardware Architecture
Read Command Flow

[Diagram of system components and connections]

ICE3028: Embedded Systems Design (Spring 2017) – Jin-Soo Kim (jinsookim@skku.edu)
Write Command Flow

INDILINX
Barefoot™ Controller

SRAM (96KB)
Controller

ROM
Controller

ARM7TDMI-S Core

Clock Generator

APB Bridge

UART

GPIO

Timer

WDT

PMU

ICU

JTAG

SRAM

NAND Flash

NAND Controller

Buffer Manager

SATA Device

DRAM Controller

Memory Utility

DRAM Access Bus

SATA Host interface

JTAG debug port

DRAM

ICE3028: Embedded Systems Design (Spring 2017) – Jin-Soo Kim (jinsookim@skku.edu)
SATA Controller

- SATA event queue
  - 128 slots for SATA command
  - Inserted by ISR
  - Deleted by FTL

- Queue policy
  - FIFO
    - Read latency suffers
  - Read first
    - RAW hazard
    - History log by H/W

Disabled in Jasmine
Buffer Manager

- SATA data is buffered in DRAM
- Memory map of Jasmine board

![Memory Map Diagram]

0xFFFF_FFFF
0x5000_0000
0x4000_0000
0x1000_0000
0x0000_0000

(omitted)
DRAM (FTL metadata)
DRAM (buffer)
(omitted)
SRAM

Copy buffer

SATA write buffer
SATA read buffer
Buffer Manager (cont’d)

- `ftl_read_ptr, ftl_write_ptr`
  - Transfer data from / to NAND

- `sata_read_ptr / sata_write_ptr`
  - Transfer data to / from host

Q. Why the order of `bm_ptr` is different in read and write?

![Diagram of SATA Read Buffer and SATA Write Buffer](image)
Triggering & Initializing FTL

- ./target_spw/init_gnu.s
  - Call init_jasmine()
  - Call Main()

- init_jasmine()
  - Initialize H/W configurations

- Main()
  - FTL top level loop
  - ./sata/sata_main.c
Dummy FTL

- ./ftl_dummy
  - ftl.c, ftl.h

- Dummy FTL is not a real FTL
  - No access to NAND flash
  - Neither stores nor retrieves any data

- Why Dummy FTL?
  - To simply measure the SATA & DRAM speed
How to Enable Dummy FTL

- `./build_gnu/Makefile`

```makefile
FTL   = tutorial
PREFIX = arm-none-ebi-
CC     = $(PREFIX)gcc
AS     = $(PREFIX)as
LD     = $(PREFIX)ld
OBJCOPY = $(PREFIX)objcopy
FW     = del

INCLUDES = -I../include -I../ftl=$(FTL) -I../sata -I../target_spv
CFLAGS = -mcpu=arm7tdmi-s -mthummb-interwork -ffreestanding -nostdlib -std=c99 -O2 -g -DPROGRAM_MAIN_FW -Wall
ASFLAGS = -R -mcpu=arm7tdmi-s
LDFLAGS = -static -nostartfiles -ffreestanding -T ld_script -Wl,-O1,-Map=list.txt
LIBS = -Igcc
VPATH  = ../ftl=$(FTL):../sata:../..:target_spv

SOURCES  = ftt.c sata_identify.c sata_cmd.c sata_isr.c sata_main.c sata_table.c initialize.c mem_util.c flash.c flash_wrapper.c misc.c uart.c
OBJECTS  = $(SOURCES:.c=.o)
DEPS     = $(SOURCES:.c=.d)
TARGET   = firmware
TARGETELF = $(TARGET).elf
TARGETBIN = $(TARGET).bin
```
Dummy FTL: Read Handling

- ./ftl_dummy/ftl.c

```c
void ftl_read(UINT32 const lba, UINT32 const total_sectors)
{
    UINT32 num_sectors_to_read;

    UINT32 lpage_addr = lba / SECTORS_PER_PAGE;       // logical page address
    UINT32 sect_offset = lba % SECTORS_PER_PAGE;       // sector offset within the page
    UINT32 sectors_remain = total_sectors;

    while (sectors_remain != 0) // one page per iteration
    {
        if (sect_offset + sectors_remain < SECTORS_PER_PAGE)
        {
            num_sectors_to_read = sectors_remain;
        }
        else
        {
            num_sectors_to_read = SECTORS_PER_PAGE - sect_offset;
        }

        UINT32 next_read_buf_id = (g_ftl_read_buf_id + 1) % NUM_RD_BUFFERS;

        while (next_read_buf_id == GETREG(SATA_RBUF_PTR));  // wait if the read buffer is full (slow host)

        SETREG(BM_STACK_RDSET, next_read_buf_id);   // change bm_read_limit
        SETREG(BM_STACK_RESET, 0x02);           // change bm_read_limit

        g_ftl_read_buf_id = next_read_buf_id;

        sect_offset = 0;
        sectors_remain -= num_sectors_to_read;
        lpage_addr++;
    }
} // end while sectors_remain! = 0
```
Dummy FTL: Write Handling

- `./ftl_dummy/ftl.c`

```c
void ftl_write(UINT32 const lba, UINT32 const total_sectors)
{
    UINT32 num_sectors_to_write;
    UINT32 sect_offset = lba % SECTORS_PER_PAGE;
    UINT32 remain_sectors = total_sectors;

    while (remain_sectors != 0)
    {
        if (sect_offset + remain_sectors >= SECTORS_PER_PAGE)
        {
            num_sectors_to_write = SECTORS_PER_PAGE - sect_offset;
        }
        else
        {
            num_sectors_to_write = remain_sectors;
        }

        while (g_ftl_write_buf_id == GETREG(SATA_WBUF_PTR)); // bm_write_limit should not outpace SATA_WBUF_PTR
        g_ftl_write_buf_id = (g_ftl_write_buf_id + 1) % NUM_WR_BUFFERS; // Circular buffer
        SETREG(BM_STACK_WRSET, g_ftl_write_buf_id); // change bm_write_limit
        SETREG(BM_STACK_RESET, 0x01); // change bm_write_limit
        sect_offset = 0;
        remain_sectors -= num_sectors_to_write;
    }
}
```
To End up Today Class

- Q. Why the order of bm_ptr is different in read and write?

- Q. What is for ‘copy buffer’ of buffer region in DRAM? (#8 slide)

- Answer each question in a single sentence

- Email me with your own answers in each group
  - Due: Next lab class
Any Questions?
IOmeter

- Performance measurement tool for storage
  - [http://www.iometer.org](http://www.iometer.org)

- Performance factors
  - IOPS (IOs Per Second)
  - Bandwidth (MB/s)
  - Response time (also known as latency)
Example

- Install & Run Iometer
- Select disk target
Example

- Make new access specification
  - Access Specifications -> New
Example

- Assign access specification
  - Select access specification and “Add”
Example

- **Start Tests**
  - Results Display -> Click ‘Flag Icon’