Homework Ⅲ 說明

Instructor：Lih-Yih Chiou

TA : Gale

Date : 2023/10/25

Outline

Homework to do:

- Hardware
  - **ISA**
    - New instructions(CSR instructions, mret, WFI)
  - **IPs**
    - DRAM Wrapper
    - ROM Wrapper (storing booting code)
    - Sensor control Wrapper
    - Watch Dog timer (clocked by 10 MHz)
- Software & Bootloader
  - Sorting, Gray Scale
  - Timer interrupt
  - Sensor interrupt
  - Booting(from DRAM to IM & DM)
- EDA tool verification
  - Spyglass CDC check
- Submission rule

Problem 1 - Hardware

Specification

New instructions - Overview

Interrupt:

A signal notifying CPU that there is some event triggered.

- ➔ Hardware interrupt examples: DMA finish signal, timeout signal from timer
- ➔ Software interrupt example: system call in OS

Interrupt Service Flow:

1. Some hardware or software interrupts CPU when a specific event is triggered
2. CPU enters a specific program to handle the interrupt
3. CPU returns from the interrupt service program to the original program.

- ➔ CPU need some control registers and status registers to know whether it can serve for the asserted interrupts, the address of interrupt service program, and the return address… etc.

New instructions - Overview

To support interrupt, there are 8 new instructions in HW3:

- CSR Instructions (6) – see unprivileged ISA
  - Register operand
    - CSRRW (Atomic Read/Write CSR)
    - CSRRS (Atomic Read and Set Bits in CSR)
    - CSRRC (Atomic Read and Clear Bits in CSR)
  - Immediate operand
    - CSRRWI (Atomic Read/Write CSR)
    - CSRRSI (Atomic Read and Set Bits in CSR)
    - CSRRCI (Atomic Read and Clear Bits in CSR)
- Trap-Return Instructions (1) – see privileged ISA
  - Machine Mode return from trap
    - MRET
- Interrupt-Management Instructions (1) – see privileged ISA
  - Wait for Interrupt
    - WFI

New instructions – Corresponding registers

- CSRs to be implemented – see privileged ISA
- There are 3 privilege levels defined by RISCV
  - Machine level – highest priority, which is to be implemented in HW3
  - Supervisor level – OS usually operates in this level
  - User level – user applications usually run on this level
- 6 machine mode CSRs to be implemented:

Address	Privilege	Name	Description
0x300	M	mstatus	Machine status register
0x304	M	mie	Machine interrupt-enable register
0x305	M	mtvec	Machine Trap-Vector Base-Address register
0x341	M	mepc	Machine exception program counter
0x344	M	mip	Machine interrupt pending register

- See CSR appendix for details of m-mode CSRs

New instructions - Description

- Description of CSR instructions in unprivileged ISA

31 20	19 15	14 12	11 7	6 0		
imm[11:0]	rs1	funct3	rd	opcode	Mnemonic	Description

csr	rs1	001	rd	1110011	CSRRW	rd = csr, if #rd != 0
						csr = rs1

csr	rs1	010	rd	1110011	CSRRS	rd = csr, if #rd != 0
						csr = csr | rs1, if that csr bit is writable and #rs1 != 0

csr	rs1	011	rd	1110011	CSRRC	rd = csr, if #rd != 0
						csr = csr & (~rs1), if that csr bit is writable and #rs1 != 0

csr	uimm[4:0]	101	rd	1110011	CSRRWI	rd = csr, if #rd != 0
						csr = uimm(zero-extend)

csr	uimm[4:0]	110	rd	1110011	CSRRSI	rd = csr, if #rd != 0
						csr = csr | uimm(zero-extend), if that csr bit is writable and uimm != 0

csr	uimm[4:0]	111	rd	1110011	CSRRCI	rd = csr, if #rd != 0
						csr = csr & (~uimm(zero-extend)), if that csr bit is writable and uimm != 0

New instructions - Description

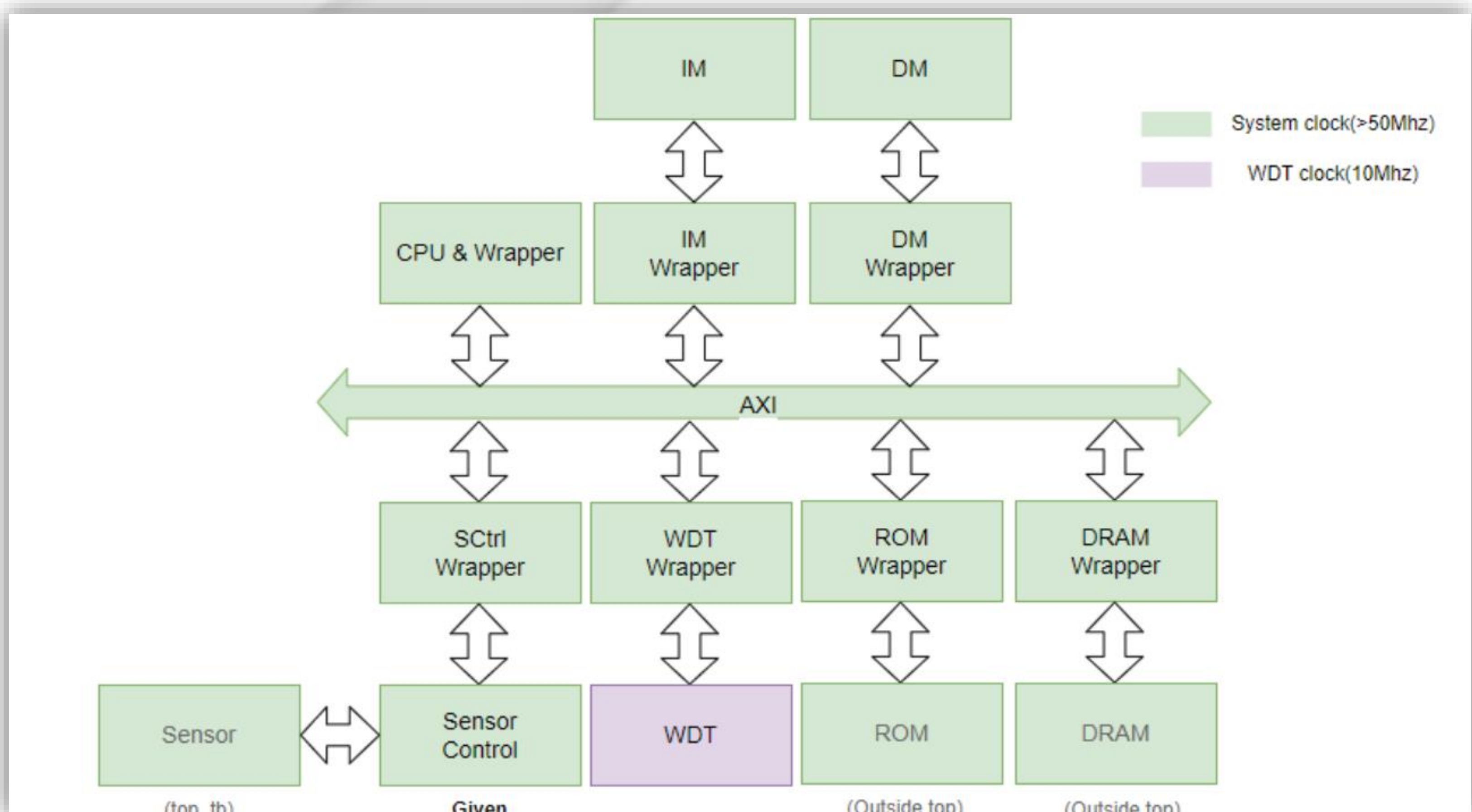
- Description of instructions in privileged ISA

31	20	19 15	14 12	11 7	6 0		
imm[11:0]		rs1	funct3	rd	opcode	Mnemonic	Description
0011000	00010	00000	000	00000	1110011	MRET	Return from traps in Machine Mode

31	20	19 15	14 12	11 7	6 0		
imm[11:0]		rs1	funct3	rd	opcode	Mnemonic	Description
0001000	00101	00000	000	00000	1110011	WFI	Wait for interrupt

New IPs

- System Architecture in HW3



New IPs - Slave Configuration

Table 1-5 : Slave configuration

NAME	Number	Start address	End address	Property
ROM	Slave 0	0x0000_0000	0x0000_1FFF	memory
IM	Slave 1	0x0001_0000	0x0001_FFFF	memory
DM	Slave 2	0x0002_0000	0x0002_FFFF	memory
sensor_ctrl	Slave 3	0x1000_0000	0x1000_03FF	I/O
WDT	Slave 4	0x1001_0000	0x1001_03FF	I/O
DRAM	Slave 5	0x2000_0000	0x201F_FFFF	memory

New IPs - DRAM

- Memory slower then SRAM, storing programs to load in HW3
- Excluded outside of top module, need to write FSM to control

DRAM	System signals			
	CK	input	1	System clock
	RST	input	1	System reset (active high)
	Memory ports			
	CSn	input	1	DRAM Chip Select (active low)
	WEn	input	4	DRAM Write Enable (active low)
	RASn	input	1	DRAM Row Access Strobe (active low)
	CASn	input	1	DRAM Column Access Strobe (active low)
	A	input	11	DRAM Address input
	D	input	32	DRAM data input
	Q	output	32	DRAM data output
	VALID	output	1	DRAM data output valid
	Memory space			
	Memory_byte0	reg	8	Size: [0:2097151]
	Memory_byte1	reg	8	Size: [0:2097151]
	Memory_byte2	reg	8	Size: [0:2097151]
	Memory_byte3	reg	8	Size: [0:2097151]

See DRAM appendix for more details

New IPs – ROM

- Read only memory, excluded outside of top module
- Used to store booting program

ROM	System signals			
	CK	input	1	System clock
	Memory ports			
	DO	output	32	ROM data output
	OE	input	1	Output enable (active high)
	CS	input	1	Chip select (active high)
	A	input	12	ROM address input
	Memory Space			
	Memory_byte0	reg	8	Size: [0:4095]
	Memory_byte1	reg	8	Size: [0:4095]
	Memory_byte2	reg	8	Size: [0:4095]
	Memory_byte3	reg	8	Size: [0:4095]

New IPs – Sensor control(1/2)

- I/O device that will fetch data from sensor
- When memory in sensor control is full, it will send sctrl_interrupt to CPU

sensor_ctrl	System signals			
	clk	input	1	System clock
	rst	input	1	System reset (active high)
	sctrl_en	input	1	Sensor controller enable (active high)
	sctrl_clear	input	1	Sensor controller clear (active high)
	sctrl_addr	input	6	Sensor controller address
	sctrl_interrupt	output	1	Sensor controller interrupt
	sctrl_out	output	32	Sensor controller data output
	sensor_ready	input	1	Sensor data ready
	sensor_out	input	32	Data from sensor
	sensor_en	output	1	Sensor enable (active high)
	Memory space			
	mem	logic	32	Size: [0:63]

New IPs – Sensor control(2/2)

- Sensor generates a new data every 1024 cycles
- Sensor control stores data to its local memory
- When local memory is full, sensor control will stop requesting data (sensor_en = 0) and assert interrupt (sctrl_interrupt = 1)
- CPU load data from sensor controller and store it to DM
- Write non-zero value in 0x1000_0100 or 0x1000_0200 to enable stcrl_en or stcrl_clear

Address	Mapping
0x1000_0300 – 0x1000_03FF	mem[0] – mem[63]
0x1000_0100	stcrl_en
0x1000_0200	stcrl_clear

New IPs – Watch Dog Timer(1/2)

- Watch Dog timer is set by CPU and is used to count for a given time.
- Once time up, WDT will send timeout signal to interrupt CPU.
- CPU control signals come from “clk” domain, and counting register is in “clk2” domain.

Module	Specifications			
	Name	**Signal**	**Bits**	**Function explanation**
WDT	clk	input	1	System clock
	rst	input	1	System reset (active high)
	clk2	input	1	WDT clock
	rst2	input	1	WDT reset (active high)
	WDEN	input	1	Enable the watchdog timer
	WDLIVE	input	1	Restart the watchdog timer
	WTOCNT	input	32	Watchdog timeout count
	WTO	output	1	watchdog timeout

New IPs – Watch Dog Timer(2/2)

- CPU can write WTOCNT to set timer value.
- When WDT is enabled(WDEN = 1), the “clk2” domain counter starts counting.
- When WDT get restart signal (WDLIVE = 1), it will reset counter to 0.
- When counter value exceeds WTOCNT, WDT will assert timeout interrupt(WTO = 1).
- Write non-zero value in 0x1001_0100 or 0x1001_0200 to enable WDEN or WDLIVE

Address	Mapping
0x1001_0100	WDEN
0x1001_0200	WDLIVE
0x1001_0300	WTOCNT

Problem 2 – Software & Bootloader

Specification

Verification(1/9)

Software list:

- prog0
  - 測試45個instruction (助教提供)
- prog1
  - Sort algorithm of half-word
- prog2
  - Gray scale
- prog3
  - Timer interrupt (助教提供)
- Prog4
  - Timer interrupt with clock uncertainty (助教提供)
- Prog5
  - Sensor interrupt (助教提供)

Firmware:

- Bootloader
  - Load software from DRAM to IM/DM

Verification (2/9)

Bootloader introduction:

- The first program that CPU will execute after reset state
- Used to load main programs or OS to faster memories.

Booting in HW3

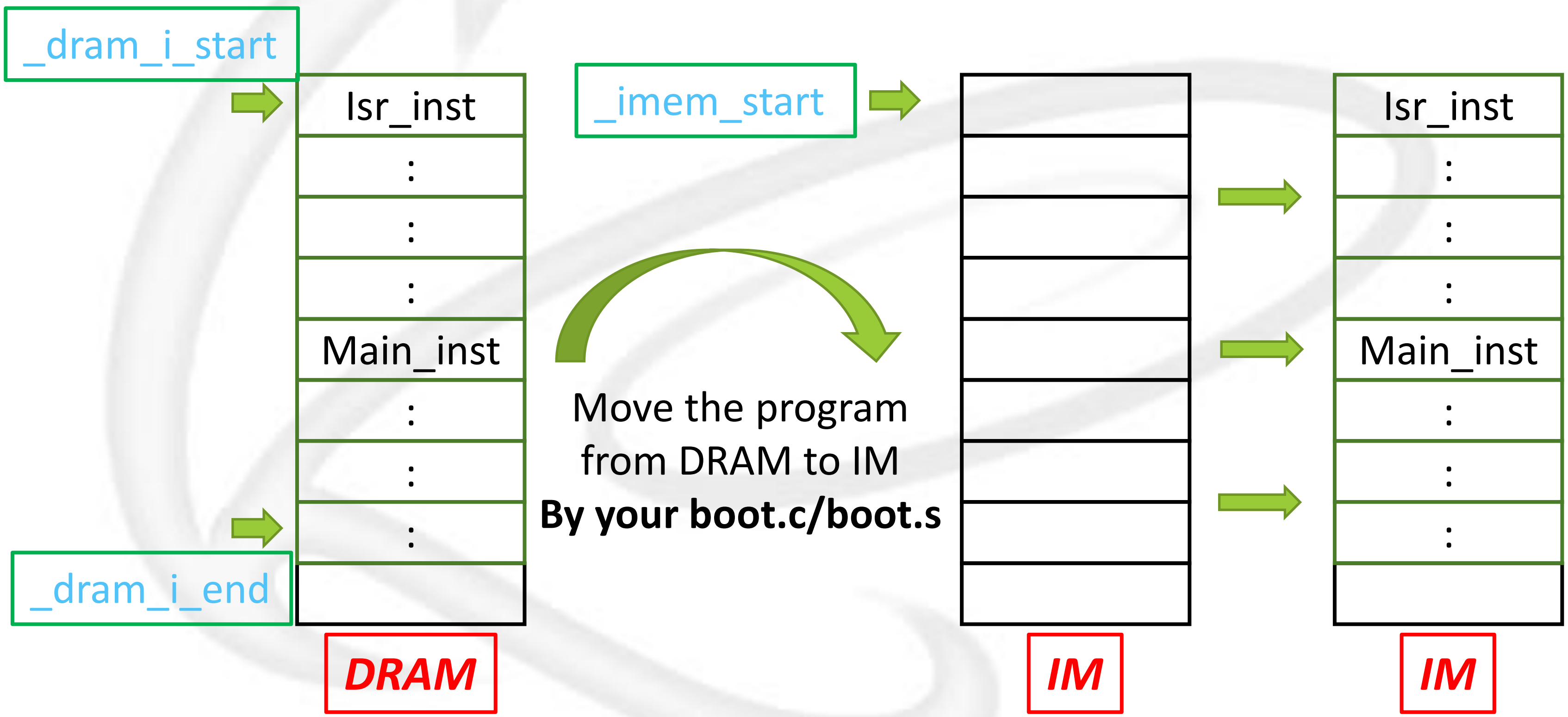
- The booting program is stored in ROM
- Booting program will move instr & data from DRAM to IM and DM

```
extern unsigned int _dram_i_start;
extern unsigned int _dram_i_end;
extern unsigned int _imem_start;

extern unsigned int __sdata_start;
extern unsigned int __sdata_end;
extern unsigned int __sdata_paddr_start;

extern unsigned int __data_start;
extern unsigned int __data_end;
extern unsigned int __data_paddr_start;
```

Verification (3/9)

Booting(1/3) - Move instructions from DRAM to IM:

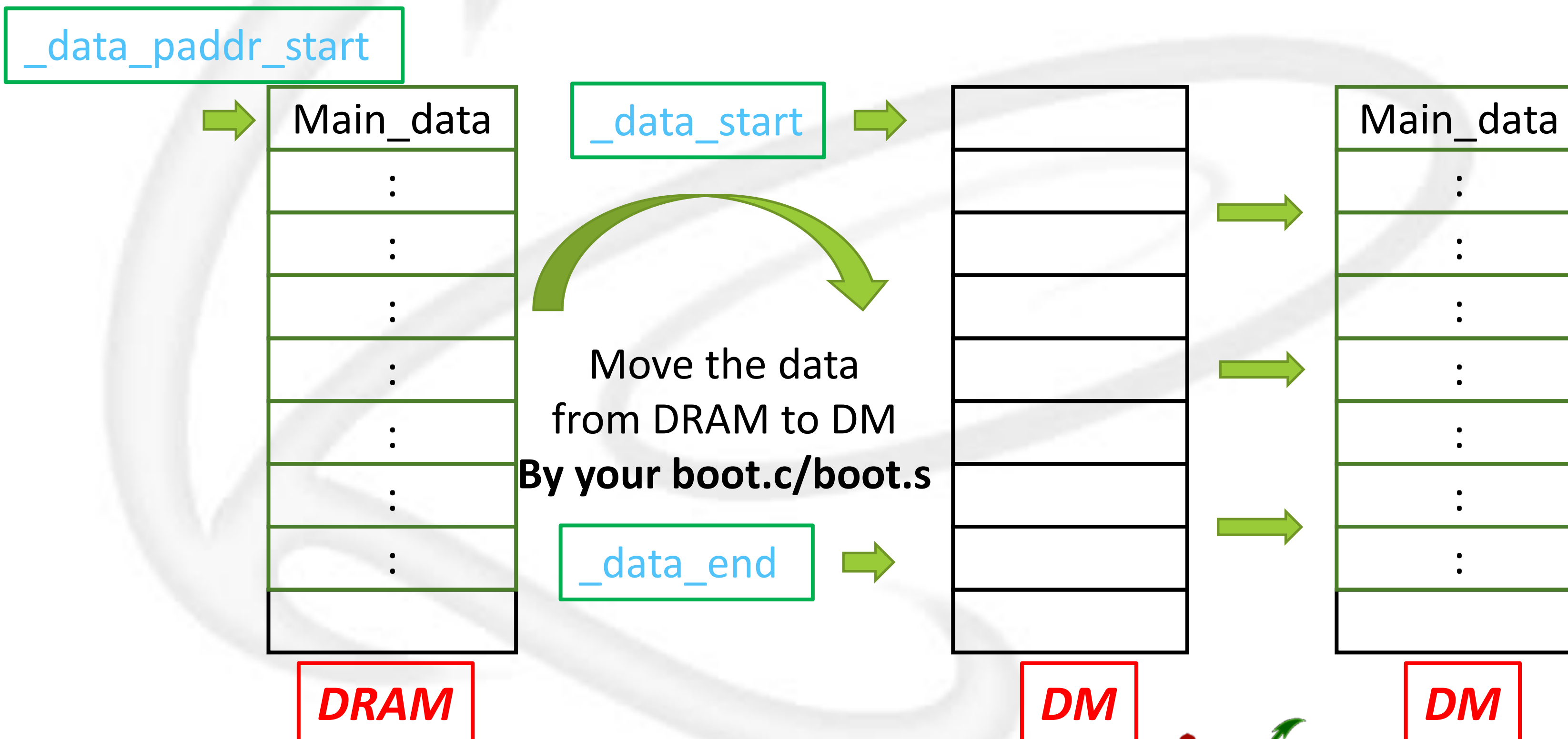
- _dram_i_start = instruction start address in DRAM.
- _dram_i_end = instruction end address in DRAM.
- _imem_start = instruction start address in IM



Verification (4/9)

Booting(2/3) - Move data from DRAM to DM:

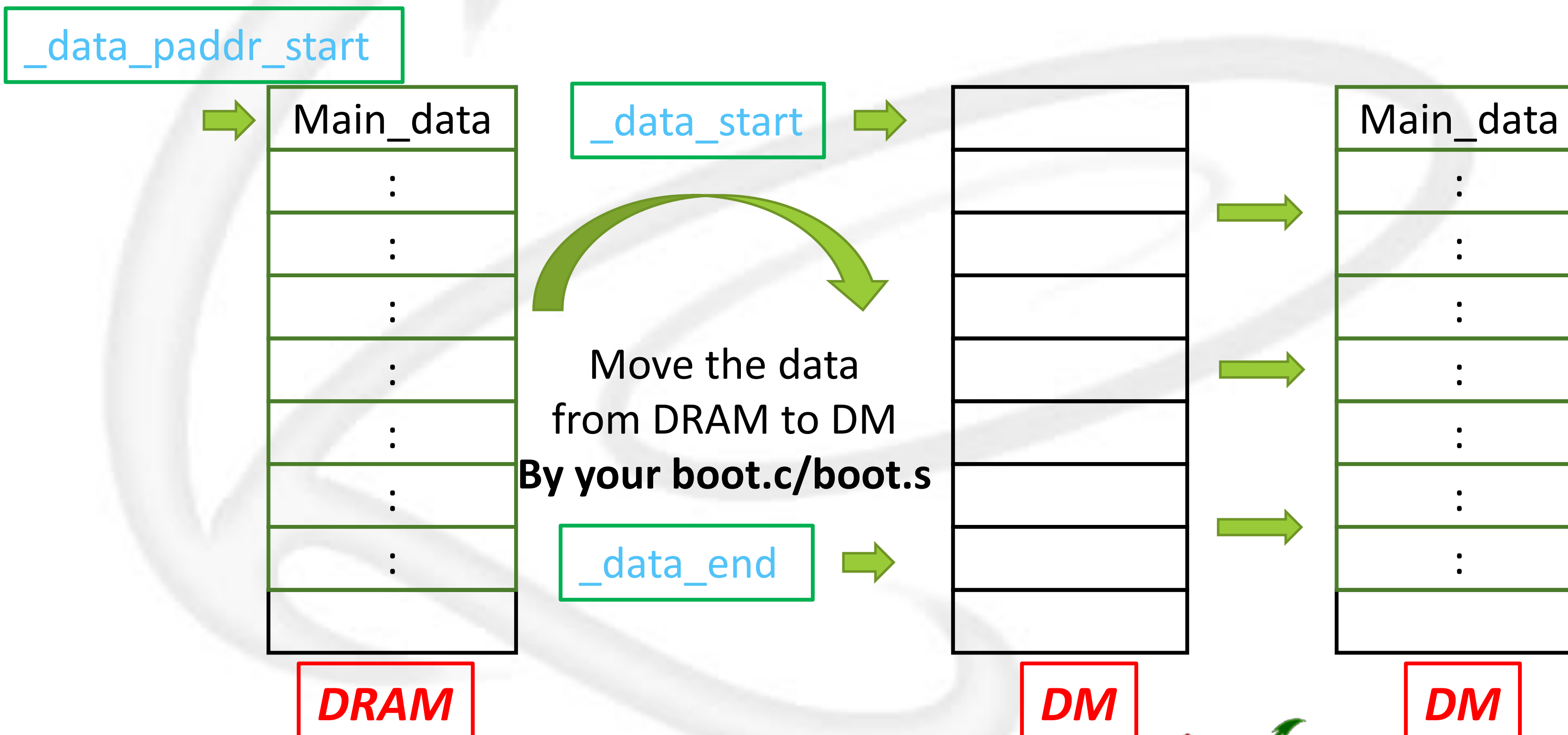
- _data_start = main data start address in DRAM.
- _data_end = main data end address in DRAM.
- _data_paddr_start = main data start address in DM



Verification (5/9)

Booting(3/3) - Move data from DRAM to DM:

- _sdata_start = main data start address in DRAM.
- _sdata_end = main data end address in DRAM.
- _sdata_paddr_start = main data start address in DM



Verification(6/9)

Prog2: gray scale algorithm – gray = 0.11b + 0.59g + 0.3r

- Overwrite b, g, r with the grayscale result.
  - _binary_image_bmp_start: start addr of colored img
  - _binary_image_bmp_end: end addr of colored img
  - _binary_image_bmp_size: total length of colored img

- …
- 12
- 1f
- 25
- 12
- 1f
- 25
- Header (54 bytes)

_binary_image_bmp_start

x = **0.11*b + 0.59*g + 0.3*r**
= 0.11*25 + 0.59*1f + 0.3* 12

y = **0.11*b + 0.59*g + 0.3*r**
= 0.11*25 + 0.59*1f + 0.3* 12

- …
- x
- x
- x
- y
- y
- y
- Header (54 bytes)

Verification(7/9)

There are 54 bytes headers

- You won't need to apply grayscale on headers.

…
12
1f
25
12
1f
25
Header (54 bytes)

← _binary_image_bmp_start

x = **0.11*b + 0.59*g + 0.3*r**
= 0.11*25 + 0.59*1f + 0.3* 12

y = **0.11*b + 0.59*g + 0.3*r**
= 0.11*25 + 0.59*1f + 0.3* 12

…
x
x
x
y
y
y
Header (54 bytes)

B M Size of BMP file (byte) The number of bits per pixel

從檔案頭到點陣圖資料須偏移的byte數

Address	0	1	2	3	4	5	6	7	8	9	a	b	Dump
00000000	42	4d	36	00	24	00	00	00	00	00	36	00	BM6.$.....6.
0000000c	00	00	28	00	00	00	00	04	00	00	00	03	....
00000018	00	00	01	00	18	00	00	00	00	00	00	00	....
00000024	24	00	c4	0e	00	00	c4	0e	00	00	00	00	$.?..?....
00000030	00	00	00	00	00	00	25	1f	12	25	1f	12	......%..%..
0000003c	25	1f	12	25	1f	12	25	1f	12	25	1f	12	%..%..%..%..
00000048	25	1f	12	25	1f	12	25	1f	12	25	1f	12	%..%..%..%..
00000054	25	1f	12	25	1f	12	25	1f	12	25	1f	12	%..%..%..%..
00000060	25	1f	12	25	1f	12	25	1f	12	25	1f	12	%..%..%..%..
0000006c	25	1f	12	25	1f	12	25	1f	12	25	1f	12	%..%..%..%..
00000078	25	1f	12	25	1f	12	25	1f	12	25	1f	12	%..%..%..%..

Verification(8/9)

- Prog3 will set CSRs and test interrupt functions
- Isr.S – interrupt service routine
  - When trap happens, CPU will jump ISR.
  - ISR will do context switch to user defined trap handler.
- Sensor interrupt:
  1. When sensor memory is full, it will interrupt CPU.
  2. The trap handler copy() will copy data to DM.
  3. Then, reset the counter of sensor controller
  4. When copy() is done, ISR return to main program
  5. After 4 groups of data are copied, the copied data will be sorted.

  This process loops for twice.

Verification(9/9)

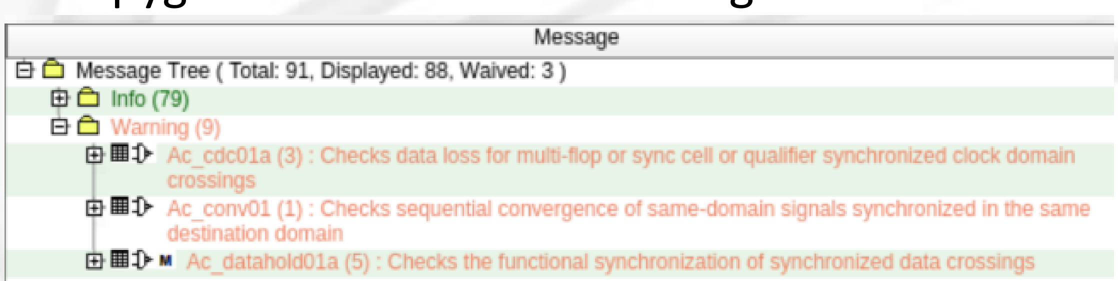
- Prog4 & prog5 will test timer interrupt
- Prog5 will test CDC functions
  - Prog4: “clk” domain is not delayed
  - Prog5: “clk” domain is delayed
- WDT interrupt:
  1. CPU will start executing program and enter dead loop
  2. When WDT assert timer interrupt, CPU will be reset
  3. After CPU is reset, it will periodically reset timer

Timer interrupt will reset CPU, make CPU reboot

Problem 3 – Spyglass CDC check

Spyglass CDC check

- Use Spyglass to do CDC check on the SoC
- Steps:
  1. Modify Spyglass.sgdc
     - Set initial value of CDC flip-flops, fifo memory…etc
  2. Open GUI
     - Type “make spyglass” in HW3
  3. Modify SoC according to the advice given by Spyglass
     - The detail description of warnings and errors can be found in Spyglass CDC Rules reference guide.



- See Spyglass CDC appendix for more details

Report Requirements

- Proper explanation of your design is required for full credits.
- Block diagrams shall be drawn to depict your designs.
- Show your screenshots of the waveforms and the simulation results on the terminal for the different test cases in your report and illustrate the correctness of your results
- Explain your codes of prog1, prog2.
- Explain your codes of boot.c.
- Report your Superlint coverage

Report Requirements

- Report and show screenshots of your prog0 to prog5 simulation time after synthesis and total cell area of your design.
- Explain how the interrupt mechanism work
- Show your screenshots of the Spyglass CDC reports and explain why your CDC circuit can work correctly

Submission rule

Report

- 請使用附在檔案內的Submission Cover
- 請勿將code貼在.docx內 (program的程式可截圖說明)
  - 請將.sv包在壓縮檔內，不可截圖於.docx中
- 需要Summary及Lessons learned(Summary table請放在第二頁，清楚列出有完成以及沒完成的部分)
- 若兩人為一組，須寫出貢獻度(貢獻度請放第二頁)
  - Ex: A(N26071234) 55%, B(N26075678) 45%
  - Total 100%
  - 自己一組則不用寫

Specification

- Module name須符合下表要求

Category	Name		
	File	Module	Instance
RTL	top.sv	top	TOP
Gate-Level	top_syn.v	top	TOP
RTL	SRAM_wrapper.sv	SRAM_wrapper	IM1
RTL	SRAM_wrapper.sv	SRAM_wrapper	DM1
RTL	SRAM_rtl.sv	SRAM	i_SRAM
Behavior	ROM.v	ROM	i_ROM
Behavior	DRAM.v	DRAM	i_DRAM

- 需按照要求命名，以免testbench抓不到正確的名稱

繳交檔案 (1/2)

- 依照檔案結構壓縮成 “.tar” 格式
  - 在Homework主資料夾(N260XXXXX)使用make tar產生的tar檔即可符合要求
- 檔案結構請依照作業說明
- 請勿附上檔案結構內未要求繳交的檔案
  - 在Homework主資料夾(N260XXXXX)使用make clean即可刪除不必要的檔案
- 請務必確認繳交檔案可以在SoC實驗室的工作站下compile，且功能正常
- 無法compile將直接以0分計算
- 請勿使用generator產生code再修改
- 禁止抄襲

繳交檔案 (2/2)

- 一組只需一個人上傳作業到Moodle
  - ➔ 兩人以上都上傳會斟酌扣分
- 壓縮檔、主資料夾名稱、Report名稱、StudentID檔案內的學號都要為上傳者的學號，其他人則在Submission Cover內寫上自己的學號。
  - ➔ Ex: A(N26101234)負責上傳，組員為B(N26105678)
  - ➔ N26101234.tar (壓縮檔)
    - N26101234 (主資料夾)
      - N26101234.docx (Report，Cover寫上兩者的學號)

繳交期限

- 2023/11/22 (三) 14:00前上傳
  - 不接受遲交，請務必注意時間
  - Moodle只會留存你最後一次上傳的檔案，檔名只要是「*N26XXXXXX.tar*」即可，不需要加上版本號

Thanks for your participation and attendance !!