Explanation for Homework IV

Speaker: June-Sheng Cheng (Jason)
Advisor: Lih-Yih Chiou
Date: 2021/12/01

Outline

- New Instructions
- Control and Status Register
- System
- Verification
- Simulation Commands
- Submission rule

New Instructions

New Instructions(1/3)

- CSR Instructions
  - Register
    - CSRRW (Atomic Read/Write CSR)
    - CSRRS (Atomic Read and Set Bits in CSR)
    - CSRRC (Atomic Read and Clear Bits in CSR)
  - Immediate
    - CSRRWI (Atomic Read/Write CSR)
    - CSRRSI (Atomic Read and Set Bits in CSR)
    - CSRRCI (Atomic Read and Clear Bits in CSR)
- Trap-Return Instructions
  - MRET (Return from traps in Machine Mode)
- Interrupt-Management Instructions
  - WFI (Wait for Interrupt)

New Instructions(2/3)

Control and Status Register Instructions

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(3/3)

Trap-Return Instructions

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

Interrupt-Management Instructions

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

Control and Status Register

CSR

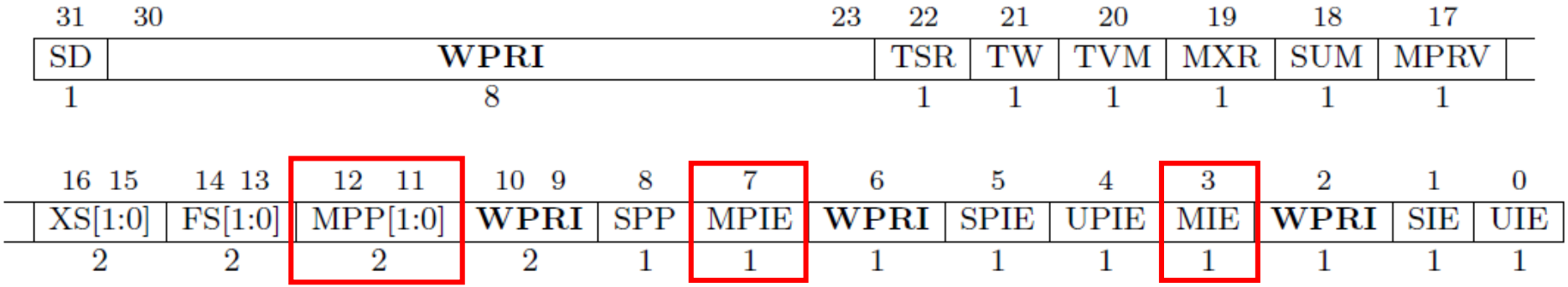
- Three level
  - User-level
  - Supervisor-level
  - Machine-level

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

Table 1-4: CSR in machine-level

CSR - mstatus

- Machine status register
- Keep track of and controls the current operating state.

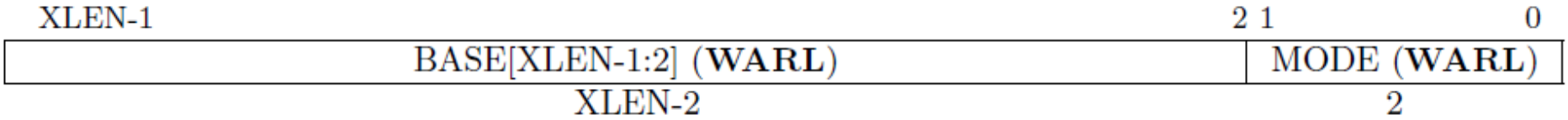


- MIE: Global Interrupt-enable bits
- MPIE: Holds the value of the interrupt-enable bit active prior to the trap
- MPP: Holds the previous privilege mode
- Other bits can hardwire to 0
- MIE will be written by instruction initially
- WFI should be unaffected by MIE

- When interrupt is taken
  - MPIE <= MIE
  - MIE <= 0
  - MPP <= 2'b11(machine mode)
- When interrupt is return
  - MPIE <= 1
  - MIE <= MPIE
  - MPP <= 2'b11(machine mode)

CSR - mtvec

- Machine Trap-Vector Base-Address Register
- Store the address where ISR start(Trap)



- Mode
  - 0: Direct: The pc to be set to the address in the BASE field.
  - 1: Vectored: The pc to be set to the address in the BASE field plus 4*the interrupt cause number
- In this homework, only implement the Direct mode, so
  - PC <= { mtvec[31:2],2'b00 }
- mtvec is hardwire to 0x0001_0000
  - The address where the trap is set.

CSR – mip & mie

- Machine Interrupt Registers
- mip: Machine interrupt-pending register

XLEN-1 12	11	10	9	8	7	6	5	4	3	2	1	0
WIRI	MEIP	WIRI	SEIP	UEIP	MTIP	WIRI	STIP	UTIP	MSIP	WIRI	SSIP	USIP
XLEN-12	1	1	1	1	1	1	1	1	1	1	1	1

- MEIP/MTIP: Indicates a machine-mode external/timer interrupt is pending
  - Connect to the external/timer interrupt signal
  - Need check MEIE/MTIE value
- Other bits hardwire to 0

- mie: Machine interrupt-enable register

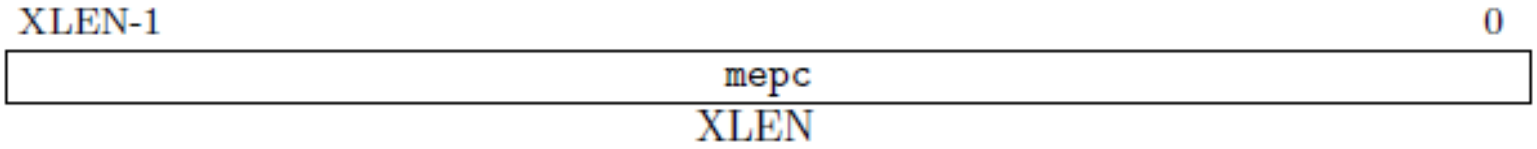
XLEN-1 12	11	10	9	8	7	6	5	4	3	2	1	0
WPRI	MEIE	WPRI	SEIE	UEIE	MTIE	WPRI	STIE	UTIE	MSIE	WPRI	SSIE	USIE
XLEN-12	1	1	1	1	1	1	1	1	1	1	1	1

- MEIE/MTIE: External/timer interrupt enable
  - MEIE is set by CSR instructions
- Other bits hardwire to 0

- WFI shouldn't ignore the MEIE/MTIE

CSR - mepc

Machine Exception Program Counter

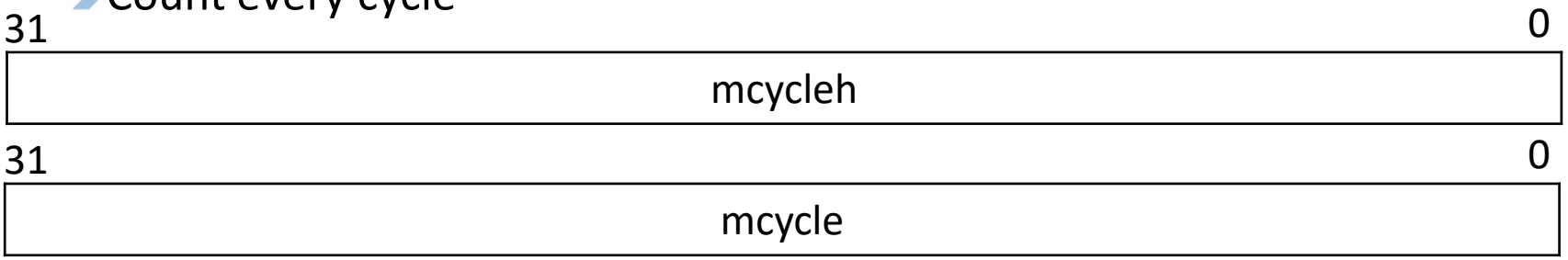


- When interrupt is taken
  - If the interrupt is taken when WFI is currently executed, store the following instruction
    - mepc <= pc+4
  - Otherwise, store the address of the instruction that encountered the interrupt
    - mepc <= pc
- When interrupt return
  - pc <= mepc

CSR – Hardware Performance Monitor

mcycleh & mcycle

- Holds the number of cycles that hardware has executed
- Divide **64-bit mcycle** into mcycleh and mcycle
- Count every cycle



minstreth & minstret

- Holds the number of instructions that CPU has completed
- Divide **64-bit minstret** into minstreth and minstret
- Count when instruction retired

System

Slave Configuration

Table 1-5 : Slave configuration

NAME	Number	Start address	End address
ROM	Slave 0	0x0000_0000	0x0000_3FFF
IM	Slave 1	0x0001_0000	0x0001_FFFF
DM	Slave 2	0x0002_0000	0x0002_FFFF
sensor_ctrl	Slave 3	0x1000_0000	0x1000_03FF
WDT	Slave 4	0x1001_0000	0x1001_03FF
DRAM	Slave 5	0x2000_0000	0x207F_FFFF

- You should design all slave wrappers !!

ROM (1/2)

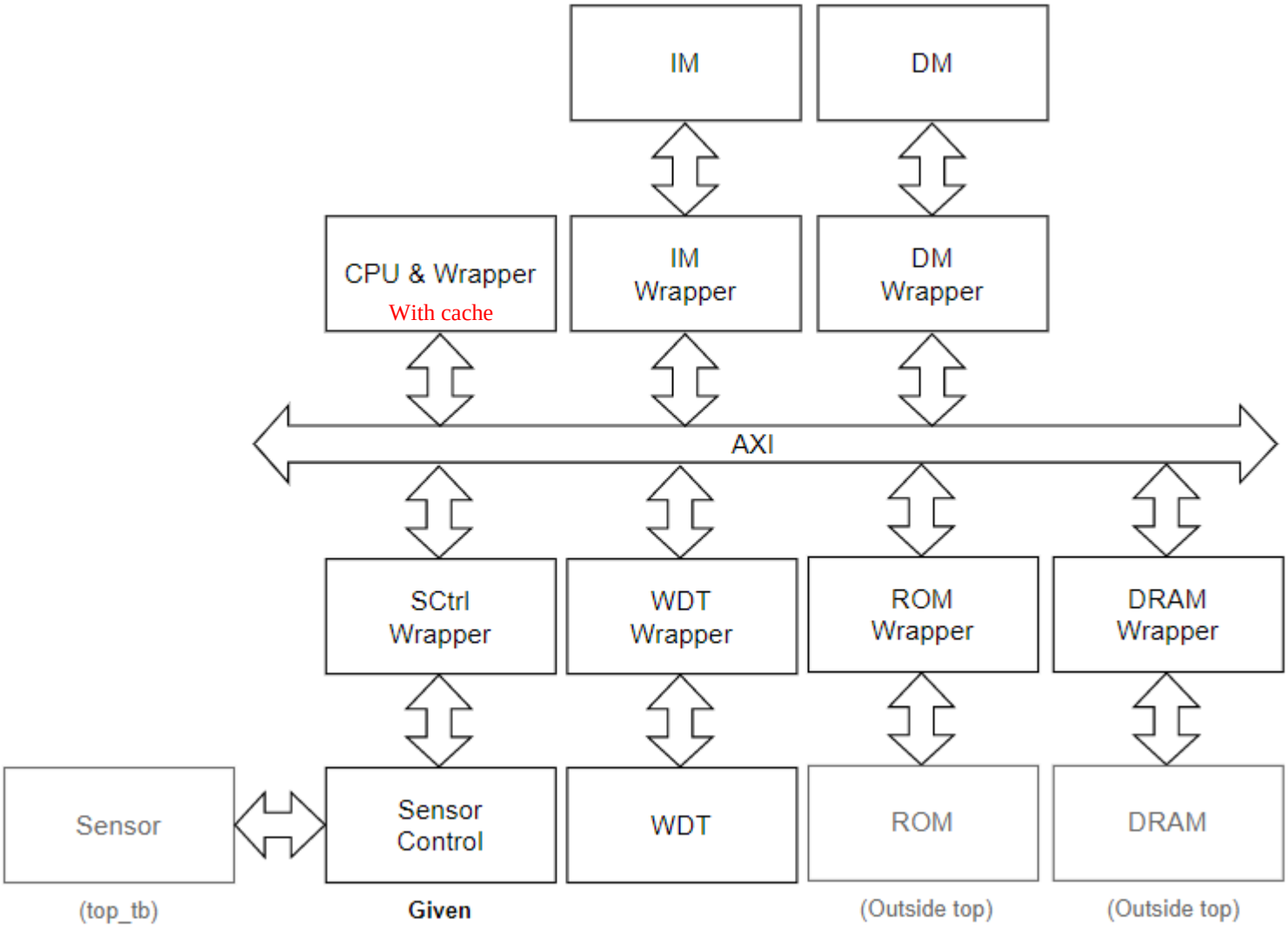
ROM	System signals			
	CK	input	1	System clock
	Memory ports			
	DO	output	32	ROM data output
	OE	input	1	Output enable (active high)
	CS	input	1	Chip select (active high)
	A	input	12	ROM address input
	Memory Space			
	Memory_byte0	reg	8	Size: [0:4095]
	Memory_byte1	reg	8	Size: [0:4095]
	Memory_byte2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

DRAM (1/3)

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]

★ Row address is 11-bit and column address is 10-bit

System


IM
DM
CPU & Wrapper
With cache
IM Wrapper
DM Wrapper
AXI
SCtrl Wrapper
WDT Wrapper
ROM Wrapper
DRAM Wrapper
Sensor
Sensor Control
WDT
ROM
DRAM
(top_tb)
Given
(Outside top)
(Outside top)

Sensor Controller (1/3)

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]

Sensor Controller (2/3)

- Sensor generates a new data every 1024 cycles
- Sensor controller stores data to its local memory
- When local memory is full (64 data), sensor controller 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

Sensor Controller (3/3)

- Any access from address 0x1000_0000 to 0x1000_03FF should be uncacheable, which means that D-cache should pass data between CPU and CPU wrapper without writing it into cache memory.
- Add following code in L1C_data.sv and use this logic to decide whether to write valid bit, tag array, and data array in L1C_data when read miss.

```
logic cacheable;
always_comb cacheable =  (core_addr[31:16] != 16'h1000);
```

Watchdog Timer(1/2)

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

Watchdog Timer(2/2)

- WDT has a real-time counter(affected by clk2) for the watchdog timeout period.
- When WDT is enabled(WDEN = 1), the real-time counter starts counting.
  - CPU need to repeatedly send a restart signal (WDLIVE = 1) to WDT for a period of time to prevent CPU restarting.
  - If WDT receive a restart signal (WDLIVE = 1), WDT need to restart the real-time counter.
  - When the real-time counter value exceeds WTOCNT, WDT will assert interrupt(WTO = 1).
- Write non-zero value in 0x1001_0100 or 0x1001_0200 to enable WDEN or WDLIVE

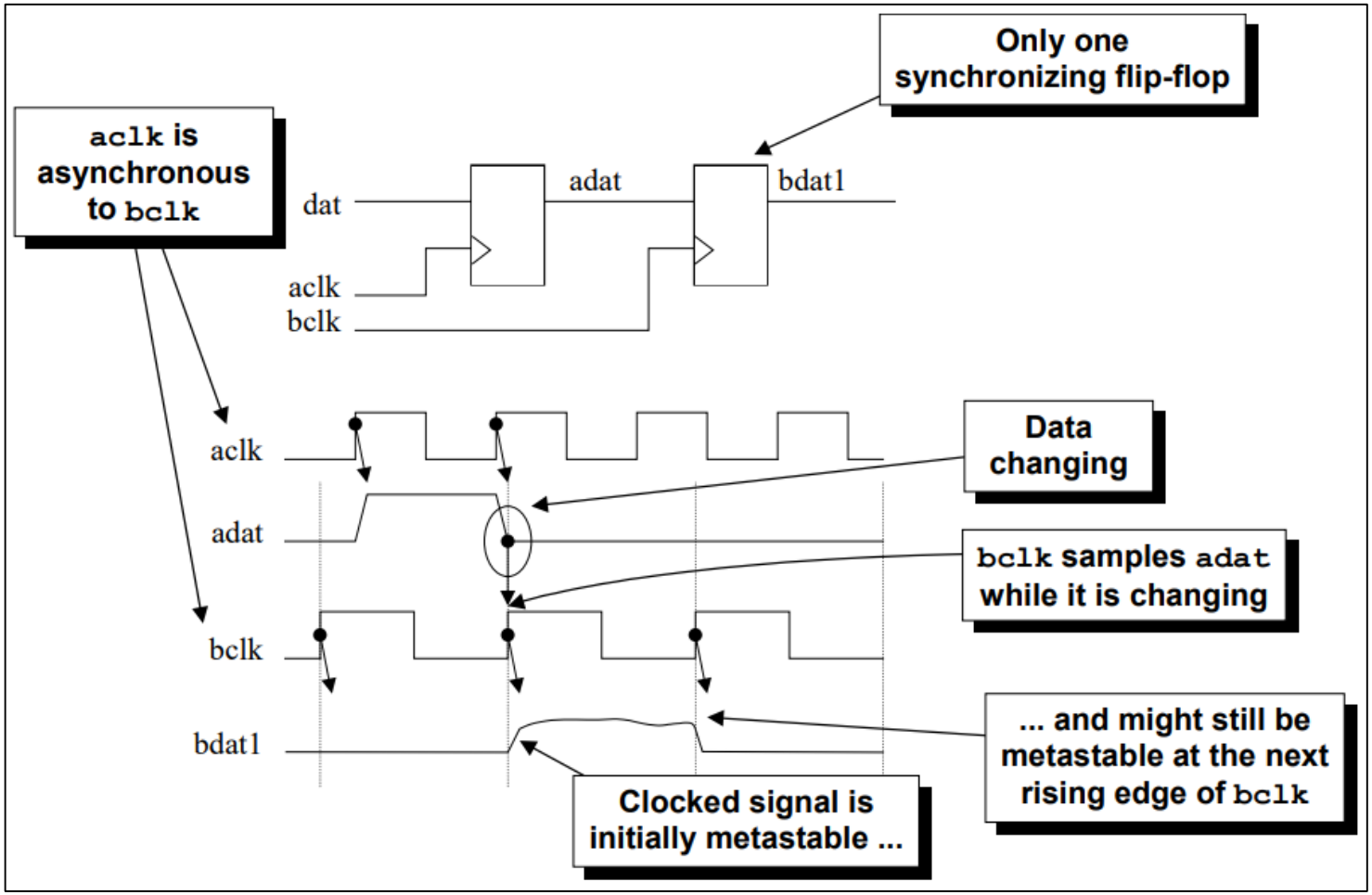
Address	Mapping
0x1001_0100	WDEN
0x1001_0200	WDLIVE
0x1001_0300	WTOCNT

Clock Domain Crossing(1/3)

- Clock Domain Crossing
  - Two or more clock domains
  - A signal or vector from one clock domain to another clock domain
- Metastability
  - It occurs when the sampling takes place during the transition of signal.
  - Setup and hold time violation

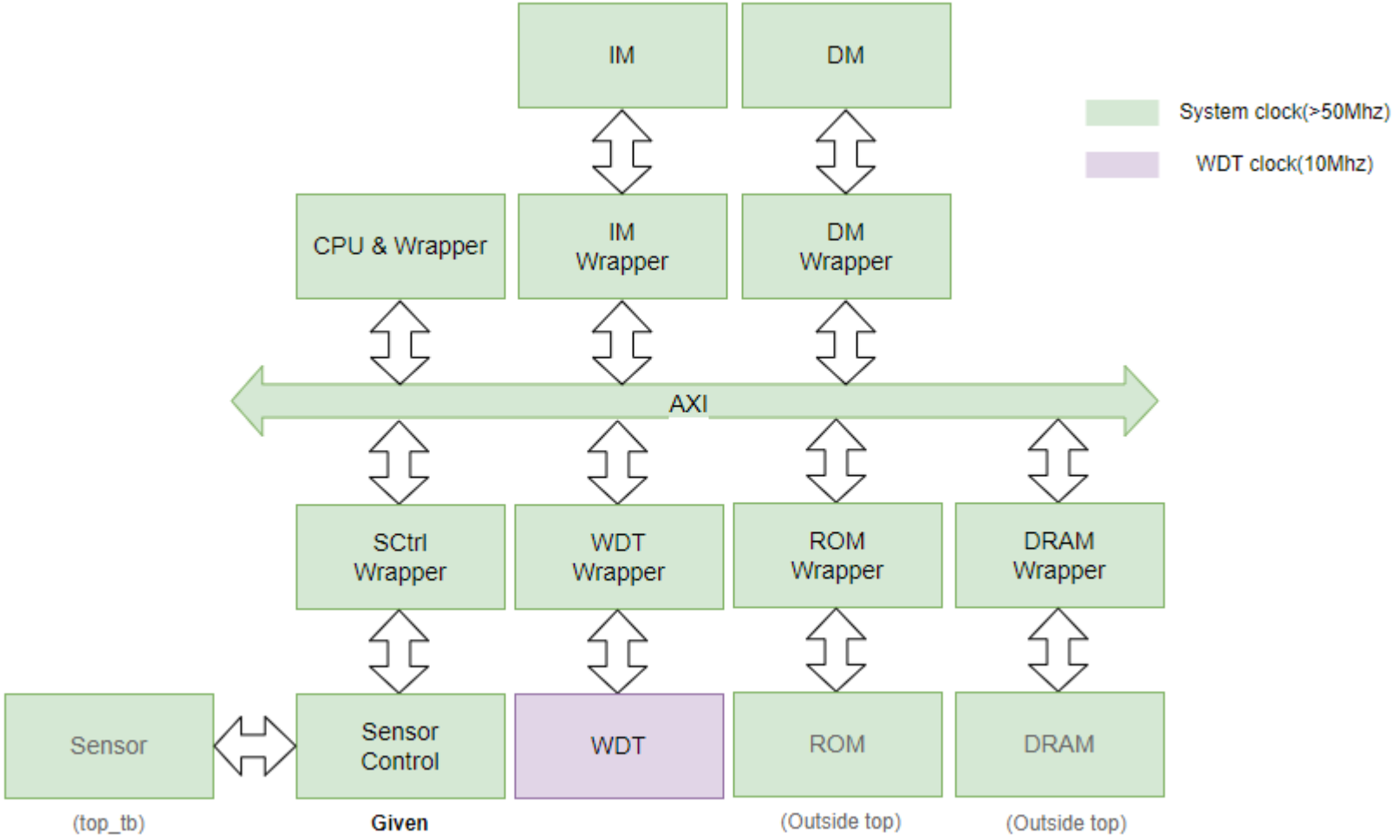
Clock Domain Crossing(2/3)

- Clock domain is sampled too close
  - Generate a metastable output



Clock Domain Crossing(3/3)

- Design CDC circuit between two domains
- And all resets need to be design in synchronous reset.



Verification

Verification (1/8)

- Prog0
  - Booting + Testing 45 instructions (Provided by TA)
- Prog1
  - Booting + Interrupt mechanism verification
- Prog2
  - Matrix multiplication
- Prog3
  - CPU malfunction + CPU operate normally
- Prog4
  - CPU malfunction + CPU operate normally (with clock delay)

Verification (2/8)

- Booting
  - The booting program is stored in ROM
  - Moves data from DRAM to IM and DM

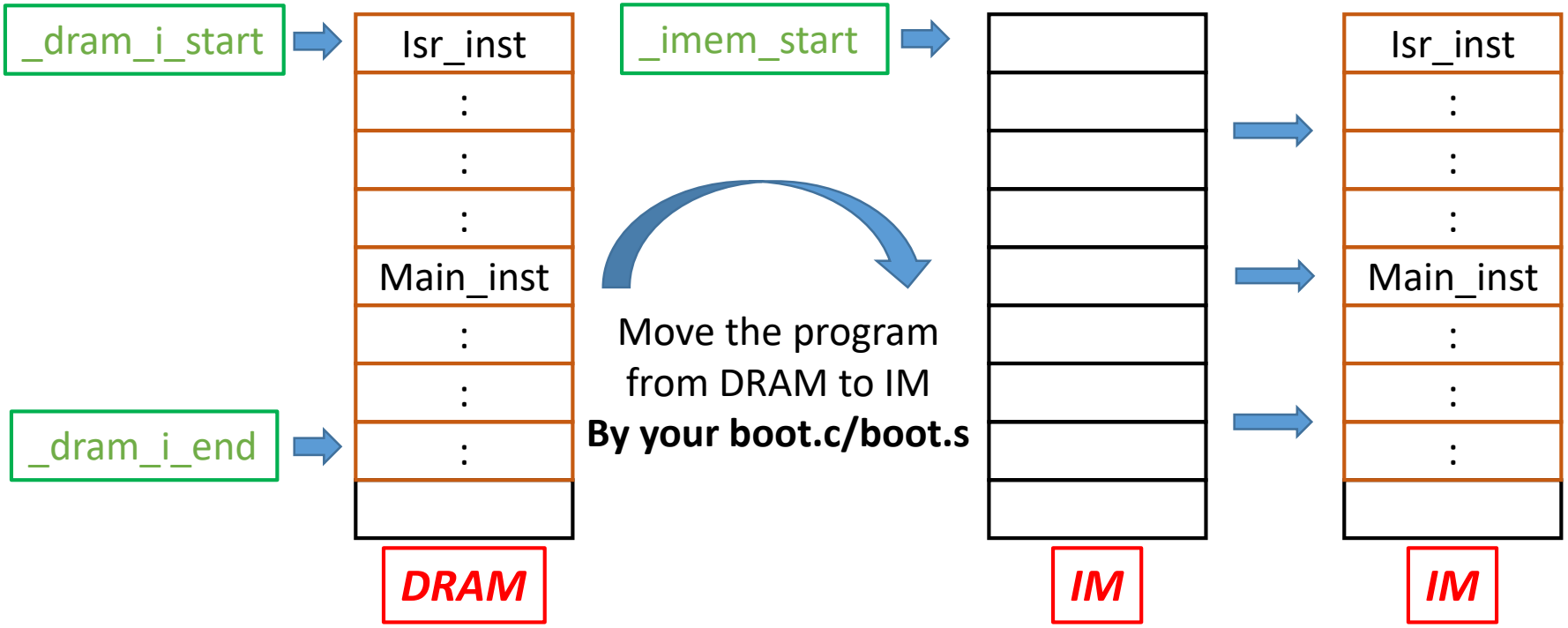
```
extern unsigned int _dram_i_start;
extern unsigned int _dram_i_end;
extern unsigned int _imem_start;

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;
```

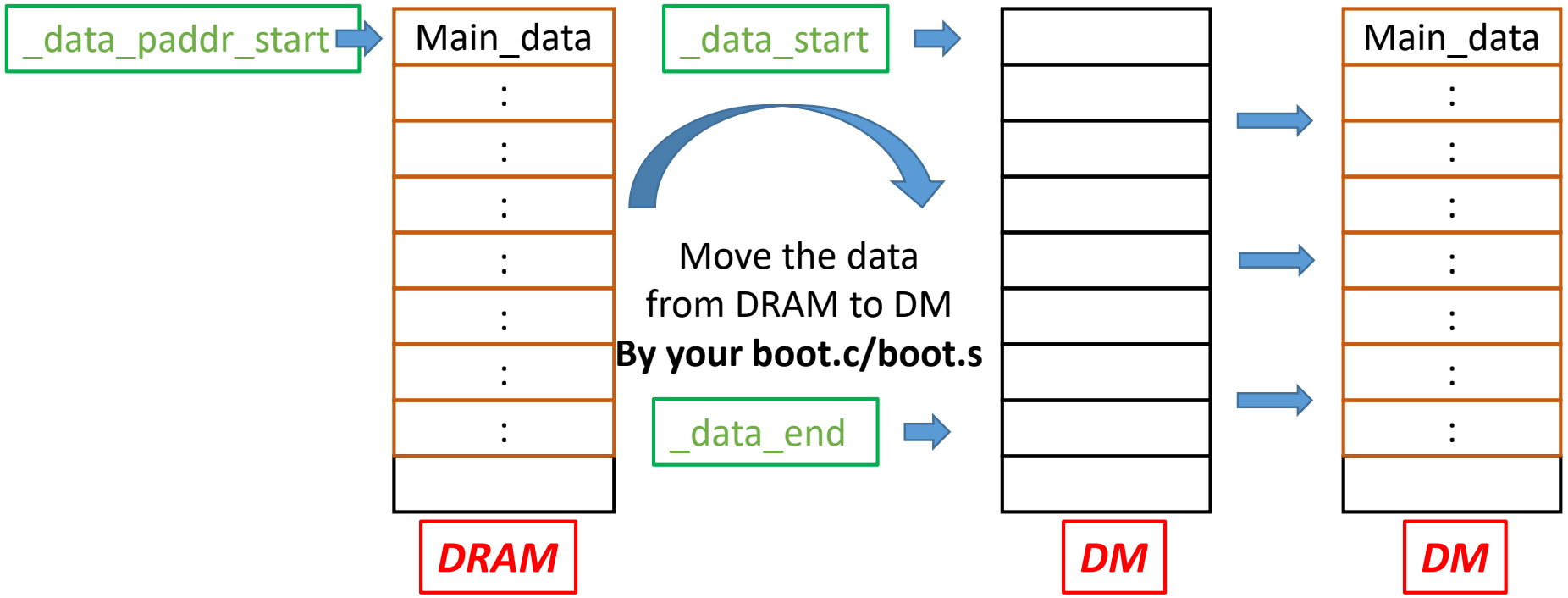
Verification (3/8)

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



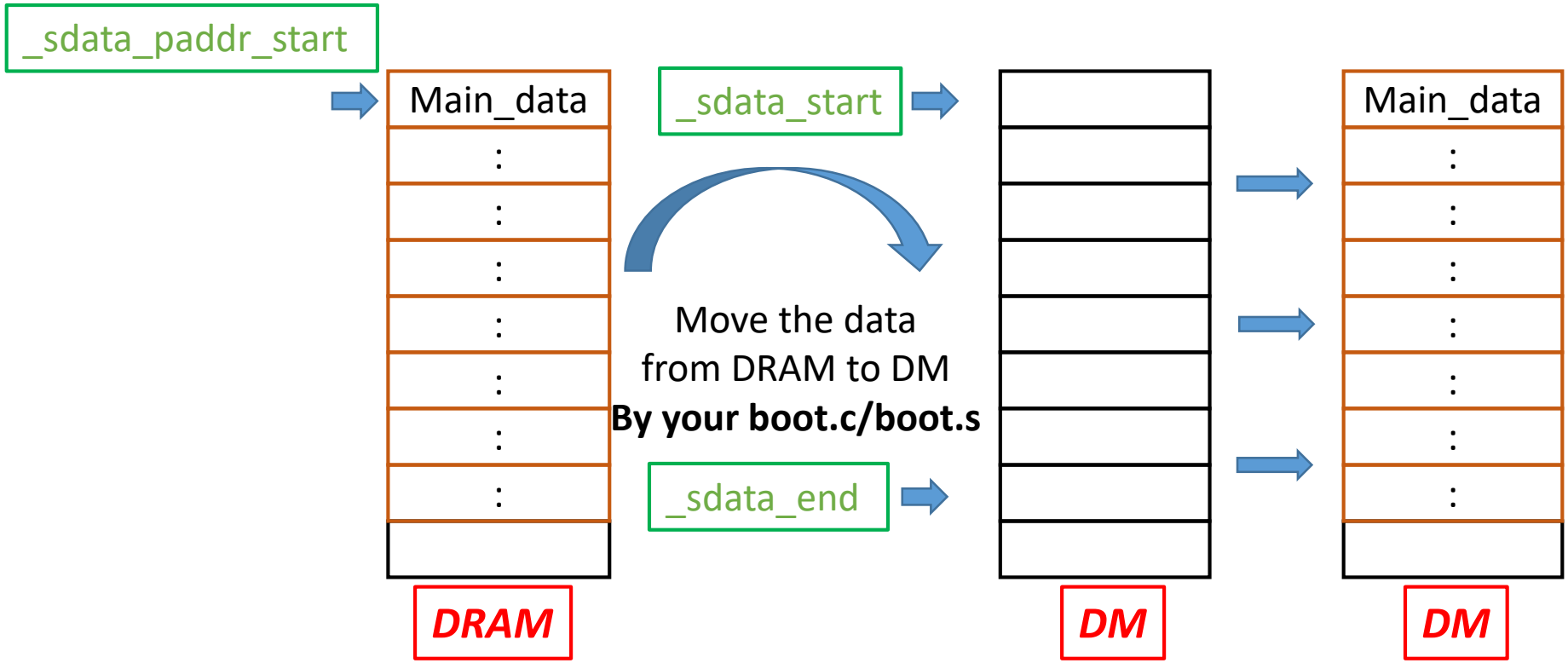
Verification (4/8)

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



Verification (5/8)

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

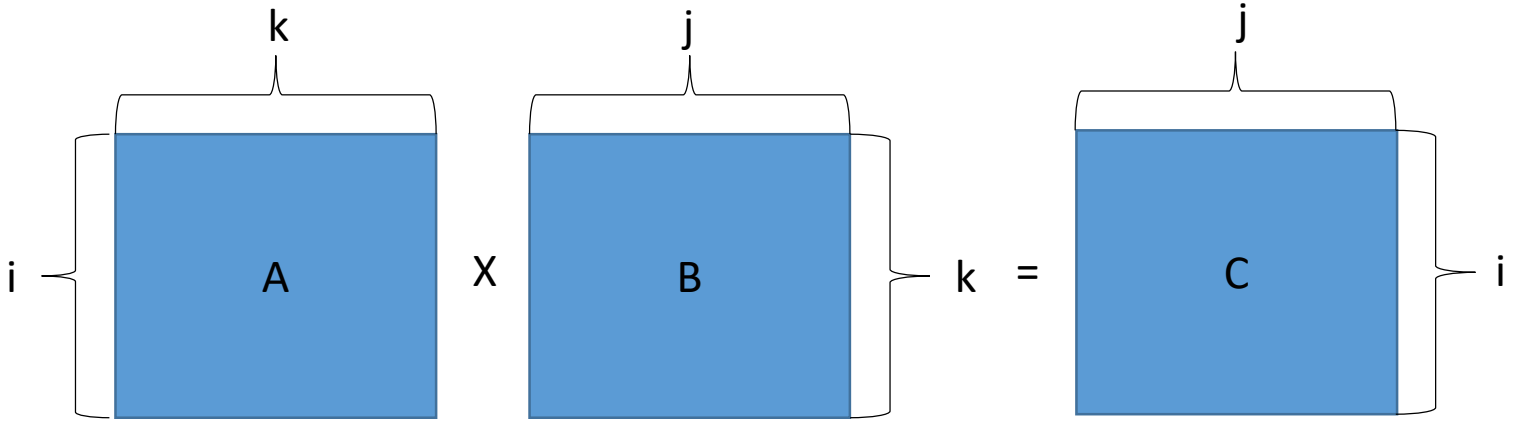


Verification (6/8)

- Sensor interrupt
  - When sensor controller is full, it will interrupt CPU
  - ISR (isr.S)will be activated and copy data to DM. Then, reset the counter of sensor controller
  - When copy is done, ISR return to main program
  - After 4 groups of data are copied, the copied data will be sorted.
    - This process executes 2 times.
- WDT interrupt
  - Reset CPU, jump to 0x0

Verification(7/8)

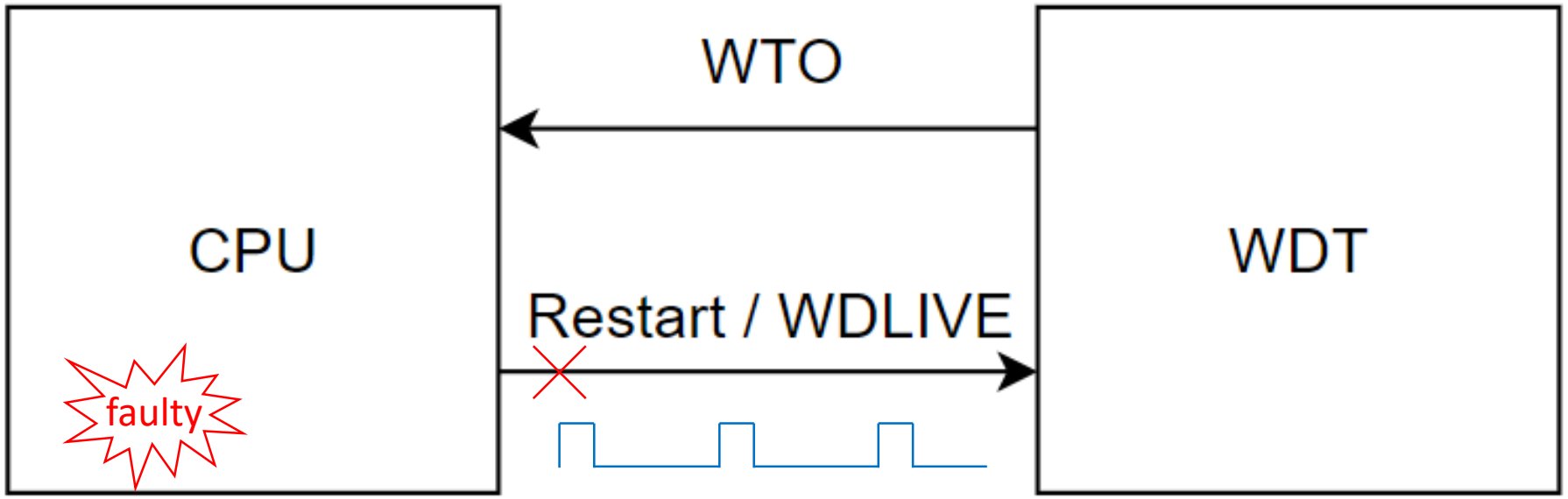
Matrix Multiplication



$$\begin{pmatrix} data\cdot 1 & data\cdot 2 & data\cdot 3 \\ data\cdot 4 & data\cdot 5 & data\cdot 6 \end{pmatrix} \cdot \begin{pmatrix} data\cdot 7 & data\cdot 8 & data\cdot 9 & data\cdot 10 \\ data\cdot 11 & data\cdot 12 & data\cdot 13 & data\cdot 14 \\ data\cdot 15 & data\cdot 16 & data\cdot 17 & data\cdot 18 \end{pmatrix} = \begin{pmatrix} result\cdot 1 & result\cdot 2 & result\cdot 3 & result\cdot 4 \\ result\cdot 5 & result\cdot 6 & result\cdot 7 & result\cdot 8 \end{pmatrix}$$

Verification(8/8)

- CPU malfunction + CPU operate normally



Simulation command

Simulation (1/2)

Table B-1: Simulation commands (Partial)

Simulation Level	Command
Problem1	
RTL	make rtl_all
Post-synthesis (optional)	make syn_all
Post-layout Gate-level	make pr_all

Table B-2: Makefile macros (Partial)

Situation	Command	Example
RTL simulation for progX	make rtlX	make rtl0
Post-synthesis simulation for progX	make synX	make syn1
Post-layout simulation for progX	make prX	make pr2
Dump waveform (no array)	make {rtlX,synX,prX} FSDB=1	make rtl2 FSDB=1
Dump waveform (with array)	make {rtlX,synX,prX} FSDB=2	make syn3 FSDB=2
Open nWave without file pollution	make nWave	
Open Superlint without file pollution	make superlint	
Open Spyglass without file pollution	make spyglass	
Open DesignVision without file pollution	make dv	
Synthesize your RTL code (You need write *synthesis.tcl* in *script* folder by yourself)	make synthesize	
Delete built files for simulation, synthesis or verification	make clean	
Check correctness of your file structure	make check	
Compress your homework to *tar* format	make tar	

Simulation (2/2)

Table B-3: Simulation commands

Situation	Command
Run JasperGold VIP on AXI bridge without file pollution (RTL only)	make vip_b
Open Innovus without file pollution	make innovus

Submission rule

Credit

- Report
  - 32%
- Simulation
  - Program0~2 has 4%. Program3~4 has 2%. Get full credit if not error.
  - Each design has 16% (4%*4)
  - Total is 48% (16%*3)
- Performance & Area
  - 20%

Report Requirements

- Report and show screenshots of your prog0 to prog2 simulation time after APR and total cell area of your design.
- 20% homework credit will be given based on your design performance & area
  - The result of PA will be ranked into 5 groups, each group has its credit point.
  - Only those who pass all APR simulation will have the chance to get PA credit

Rank	Credit	
Tier1	20	Top 20% of PA result
Tier2	16	Top 40% of PA result
Tier3	12	Top 60% of PA result
Tier4	8	Top 80% of PA result
Tier5	4	
Tier6	0	Fail in APR

$$\text{Performace(P)} = \sum_{i=0}^{2} \text{simulation time of prog}i$$

$$\text{Area(A)} = \text{ total cell area of your design}$$

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 C code of booting process.
- Explain your C code of program2.
- Explain how the interrupt mechanism work
- Show your screenshots of the Spyglass CDC reports and explain why your CDC circuit can work correctly.
- Report the number of lines of your RTL code, the results of running Superlint and 3~5 most frequent warning/errors in your code. Describe how you modify your code to comply with Superlint.

Report Requirements

- Please use the Submission Cover
- Please don’t paste your code in the report
  - The code of program is allowed for the explanation
- Summary and Lessons learned
  - Please put the Summary table in page2, and clearly list the completed and unfinished parts
- If two people are in a group, the contribution must be written (please put the contribution on the second page)
  - Ex: A(N26071234) 55%, B(N26075678) 45%
  - Total 100%
  - You don’t need to write if you own a group

File Upload(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

File Upload(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)

Deadline

- 2022/12/28 (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「*N26OXXXXX.tar*」即可，不需要加上版本號

Thanks for your participation and attendance !!