Homework I 說明

Instructor：Lih-Yih Chiou

TA : 陳志芳

Date : 09/11/2024

Outline

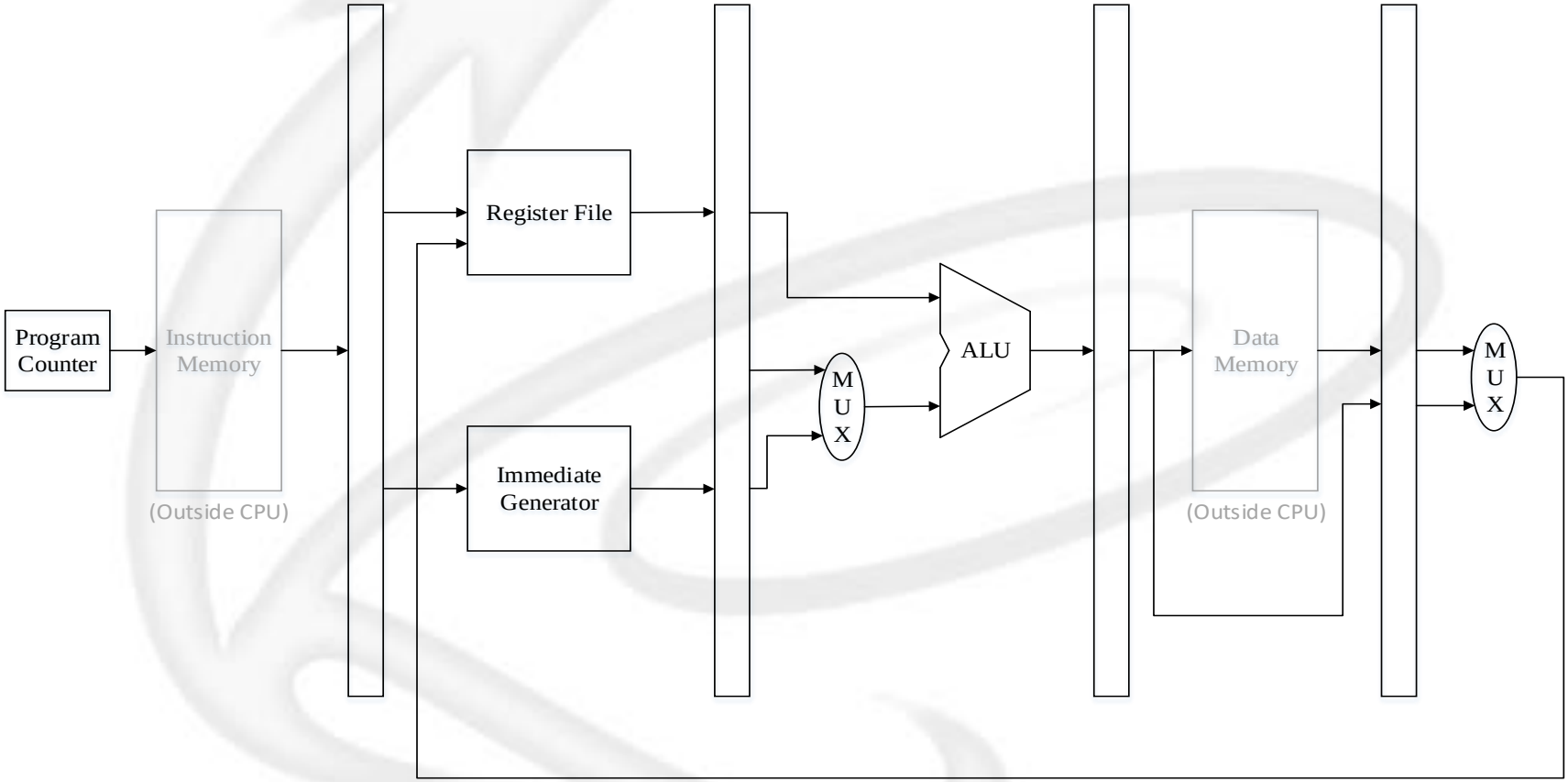
- 作業內容說明
- 作業驗證說明
- 作業繳交注意事項

VLSI Design LAB

作業內容說明

Problem 1

- 根據作業中附的RISC-V ISA，完成一個pipelined CPU



Problem 1 Specification

- Implement the 49 instructions as listed. The number of pipeline stage is 5.
- General register File size: 32x32-bit
  - ➔ x0 is read only 0.
- Floating point register File size: 32x32-bit
- Instruction memory size: 16Kx32-bit
- Data memory size: 16Kx32-bit
- Timescale: 1ns/10ps
- Maximum Clock period: 10ns (100MHz)
- Control and Status Register: 2x64-bit

Problem 1 – Instructions (1/6)

- R-type

31 25	24 20	19 15	14 12	11 7	6 0		
funct7	rs2	rs1	funct3	rd	opcode	Mnemonic	Description
0000000	rs2	rs1	000	rd	0110011	ADD	rd = rs1 + rs2
0100000	rs2	rs1	000	rd	0110011	SUB	rd = rs1 - rs2
0000000	rs2	rs1	001	rd	0110011	SLL	rd = $rs1_u$ << rs2[4:0]
0000000	rs2	rs1	010	rd	0110011	SLT	rd = ($rs1_s$ < $rs2_s$)? 1:0
0000000	rs2	rs1	011	rd	0110011	SLTU	rd = ($rs1_u$ < $rs2_u$)? 1:0
0000000	rs2	rs1	100	rd	0110011	XOR	rd = rs1 ^ rs2
0000000	rs2	rs1	101	rd	0110011	SRL	rd = $rs1_u$ >> rs2[4:0]
0100000	rs2	rs1	101	rd	0110011	SRA	rd = $rs1_s$ >> rs2[4:0]
0000000	rs2	rs1	110	rd	0110011	OR	rd = rs1 | rs2
0000000	rs2	rs1	111	rd	0110011	AND	rd = rs1 & rs2

- ➔ “M” standard extension for integer multiplication

31 25	24 20	19 15	14 12	11 7	6 0		
funct7	rs2	rs1	funct3	rd	opcode	Mnemonic	Description
0000001	rs2	rs1	000	rd	0110011	MUL	rd = lower 32 bits of (rs1 * rs2)
0000001	rs2	rs1	001	rd	0110011	MULH	rd = upper 32 bits of (rs1 * rs2) signed * signed
0000001	rs2	rs1	010	rd	0110011	MULHSU	rd = upper 32 bits of (rs1 * rs2) signed * unsigned
0000001	rs2	rs1	011	rd	0110011	MULHU	rd = upper 32 bits of (rs1 * rs2) unsigned * unsigned

p.s. Be careful of the timing of multiplier, you can use pipeline or other skills to adjust the path per cycle.

Problem 1 – Instructions (2/6)

- I-type

31 20	19 15	14 12	11 7	6 0		
imm[11:0]	**rs1**	**funct3**	**rd**	**opcode**	**Mnemonic**	**Description**
imm[11:0]	rs1	010	rd	0000011	LW	rd = M[rs1+imm]
imm[11:0]	rs1	000	rd	0010011	ADDI	rd = rs1 + imm
imm[11:0]	rs1	010	rd	0010011	SLTI	rd = ($rs1_s$ < imm_s)? 1:0
imm[11:0]	rs1	011	rd	0010011	SLTIU	rd = ($rs1_u$ < imm_u)? 1:0
imm[11:0]	rs1	100	rd	0010011	XORI	rd = rs1 ^ imm
imm[11:0]	rs1	110	rd	0010011	ORI	rd = rs1 | imm
imm[11:0]	rs1	111	rd	0010011	ANDI	rd = rs1 & imm
imm[11:0]	rs1	000	rd	0000011	LB	rd = $M[rs1+imm]_{bs}$
0000000 shamt	rs1	001	rd	0010011	SLLI	rd = $rs1_u$ << shamt
0000000 shamt	rs1	101	rd	0010011	SRLI	rd = $rs1_u$ >> shamt
0100000 shamt	rs1	101	rd	0010011	SRAI	rd = $rs1_s$ >> shamt
imm[11:0]	rs1	000	rd	1100111	JALR	rd = PC + 4 PC = imm + rs1 (Set LSB of PC to 0)
imm[11:0]	rs1	001	rd	0000011	LH	rd = $M[rs1+imm]_{hs}$
imm[11:0]	rs1	100	rd	0000011	LBU	rd = $M[rs1+imm]_{bu}$
imm[11:0]	rs1	101	rd	0000011	LHU	rd = $M[rs1+imm]_{hu}$

- S-type

31 25	24 20	19 15	14 12	11 7	6 0		
imm[11:5]	**rs2**	**rs1**	**funct3**	**imm[4:0]**	**opcode**	**Mnemonic**	**Description**
imm[11:5]	rs2	rs1	010	imm[4:0]	0100011	SW	M[rs1+imm] = rs2
imm[11:5]	rs2	rs1	000	imm[4:0]	0100011	SB	M[rs1+imm]b = $rs2_b$
imm[11:5]	rs2	rs1	001	imm[4:0]	0100011	SH	$M[rs1+imm]_h$ = $rs2_h$

Problem 1 – Instructions (3/6)

- B-type

31 25	24 20	19 15	14 12	11 7	6 0		
imm[12|10:5]	rs2	rs1	funct3	imm[4:1|11]	opcode	Mnemonic	Description
imm[12|10:5]	rs2	rs1	000	imm[4:1|11]	1100011	BEQ	PC = (rs1 == rs2)? PC + imm: PC + 4
imm[12|10:5]	rs2	rs1	001	imm[4:1|11]	1100011	BNE	PC = (rs1 != rs2)? PC + imm: PC + 4
imm[12|10:5]	rs2	rs1	100	imm[4:1|11]	1100011	BLT	PC = ($rs1_s$ < $rs2_s$)? PC + imm: PC + 4
imm[12|10:5]	rs2	rs1	101	imm[4:1|11]	1100011	BGE	PC = ($rs1_s$ ≥ $rs2_s$)? PC + imm: PC + 4
imm[12|10:5]	rs2	rs1	110	imm[4:1|11]	1100011	BLTU	PC = ($rs1_u$ < $rs2_u$)? PC + imm: PC + 4
imm[12|10:5]	rs2	rs1	111	imm[4:1|11]	1100011	BGEU	PC = ($rs1_u$ ≥ $rs2_u$)? PC + imm: PC + 4

- U-type

31 12	11 7	6 0		
imm[31:12]	rd	opcode	Mnemonic	Description
imm[31:12]	rd	0010111	AUIPC	rd = PC + imm
imm[31:12]	rd	0110111	LUI	rd = imm

- J-type

31 12	11 7	6 0		
imm[20|10:1|11|19:12]	rd	opcode	Mnemonic	Description
imm[20|10:1|11|19:12]	rd	1101111	JAL	rd = PC + 4 PC = PC + imm

Problem 1 – Instructions (4/6)

- F-type

31 20	19 15	14 12	11 7	6 0		
imm[11:0]	rs1	funct3	frd	opcode	Mnemonic	Description
imm[11:0]	rs1	010	frd	0000111	FLW	frd = M[rs1+imm]

31 25	24 20	19 15	14 12	11 7	6 0		
imm[11:5]	frs2	rs1	funct3	imm[4:0]	opcode	Mnemonic	Description
imm[11:5]	frs2	rs1	010	imm[4:0]	0100111	FSW	M[rs1+imm] = frs2

31 27	26 25	24 20	19 15	14 12	11 7	6 0		
funct5	fmt	frs2	frs1	rm	frd	opcode	Mnemonic	Description
00000	00	frs2	frs1	111	frd	1010011	FADD.S	frd = frs1+frs2
00001	00	frs2	frs1	111	frd	1010011	FSUB.S	frd = frs1 - frs2

- Note1 : The fmt field represents floating-point format, and in this lab, we only use **32-bit single-precision**.
- Note2 : The rm field represents rounding mode, and in this lab, we only use **the Round to Nearest, ties to Even** mode.
- Note3 : You don't need to consider overflow or underflow situations in this lab.

Problem 1 – Instructions (5/6)

- Control and Status Register (CSR)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

31 20	19 15	14 12	11 7	6 0		
imm[11:0]	rs1	funct3	rd	opcode	Mnemonic	Description
110010000010	00000	010	rd	1110011	RDINSTRETH	rd = instret[63:32]
110000000010	00000	010	rd	1110011	RDINSTRET	rd = instret [31:0]
110010000000	00000	010	rd	1110011	RDCYCLEH	rd = cycle[63:32]
110000000000	00000	010	rd	1110011	RDCYCLE	rd = cycle[31:0]

Problem 1 – Instructions (6/6)

- Control and Status Register (CSR)instructions
  - RDINSTRET/RDINSTRETH
    - Read CSR instret , which is the CSR to count how many instructions processed before

instret = 4

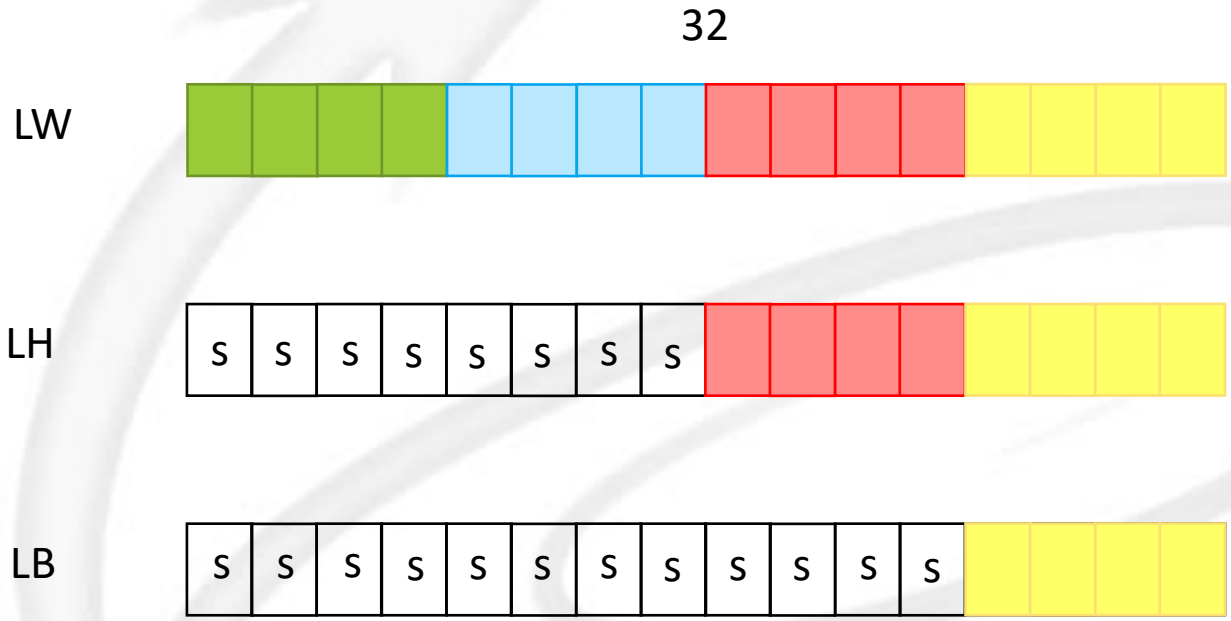
```
auipc a0,0x8
addi  a0,a0,124 # 8100 <__sbss_end>
auipc a1,0x8
addi  a1,a1,112 # 80fc <_test_start+0xfc>
li  a2,0
jal ra,104 <fill_block>
```

  - RDCYCLE/RDCYCLEH
    - Read CSR cycle, which is the CSR to count how many cycles processed before

31 20	19 15	14 12	11 7	6 0		
imm[11:0]	**rs1**	**funct3**	**rd**	**opcode**	**Mnemonic**	**Description**
110010000010	00000	010	rd	1110011	RDINSTRETH	rd = instret[63:32]
110000000010	00000	010	rd	1110011	RDINSTRET	rd = instret [31:0]
110010000000	00000	010	rd	1110011	RDCYCLEH	rd = cycle[63:32]
110000000000	00000	010	rd	1110011	RDCYCLE	rd = cycle[31:0]

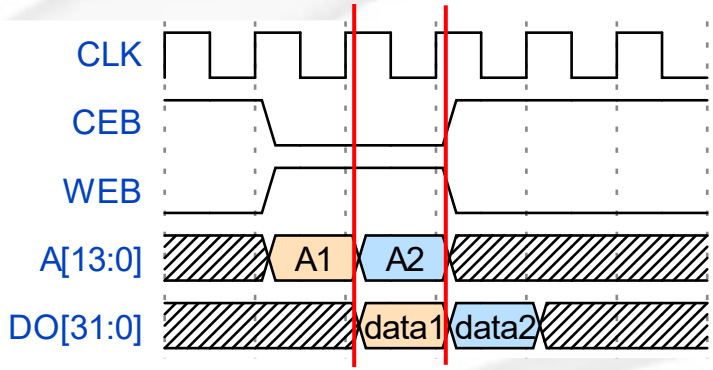
Problem 1 – Read/Write Word/Half-word/Byte

- If the instruction is used to read half-word or bytes, all other bytes left must be filled with 0s or sign bits.

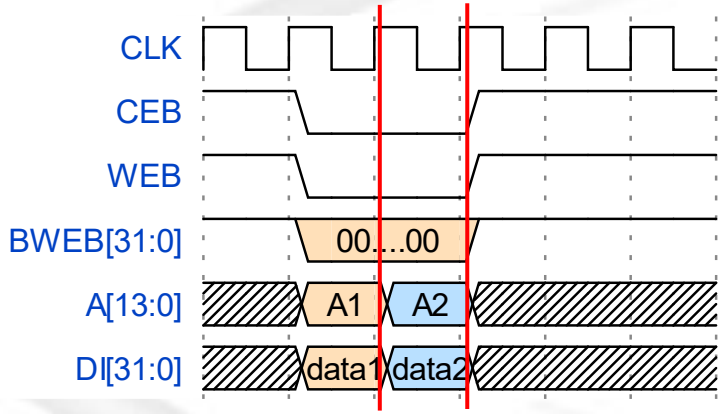


Problem 2 – SRAM

- Read operation



- Write operation



NOTE: DO delays 1 clock after Address is imported, be aware of the pipeline registers delay

VLSI Design LAB

作業驗證說明

Program

- prog0
  - 測試49個instruction (助教提供)
- prog1
  - Sort Algorithm
- prog2
  - 不使用MUL/MUL[[S]U] instructions 完成Multiplication (助教提供)
- prog3
  - Greatest common divisor
- prog4
  - 使用階乘c code測試rdinstret, rdinstreth, rdcycle, rdcycleh(助教提供)
- prog5
  - 使用MUL/MUL[[S]U] instructions實現Multiplication (助教提供)
- prog6
  - 使用floating point instructions進行運算(助教提供)

Simulation

Table B-1: Simulation commands

Simulation Level	Command
Problem1	
RTL	make rtl_all
Post-synthesis	make syn_all

Table B-2: Makefile macros

Situation	Command	Example
RTL simulation for progX	make rtlX	make rtl0
Post-synthesis simulation for progX	make synX	make syn1
Dump waveform (no array)	make {rtlX,synX} FSDB=1	make rtl2 FSDB=1
Dump waveform (with array)	make {rtlX,synX} FSDB=2	make syn3 FSDB=2
Open nWave without file pollution	make nWave	
Open Superlint without file pollution	make superlint	
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	

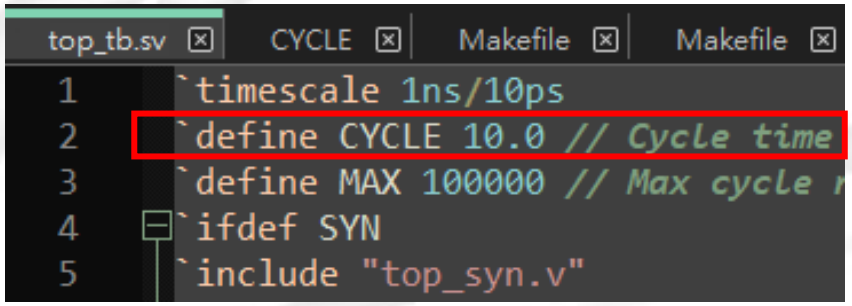
Script/DC.sdc

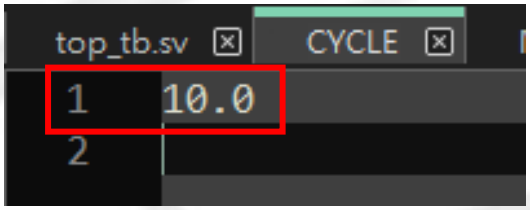
- You can modify those red box in Script/DC.sdc during synthesize.
- The maximum allowed value for the clk_period is **10.0**

```
 6
 7  set clk_period 10
 8  set input_max   [expr {double(round(1000
 9  set input_min   [expr {double(round(1000
10  set output_max  [expr {double(round(1000
11  set output_min  [expr {double(round(1000
12
```

Simulation

- You can only modify those red box in sim/top_tb.sv and sim/CYCLE file during post-synthesize simulation.



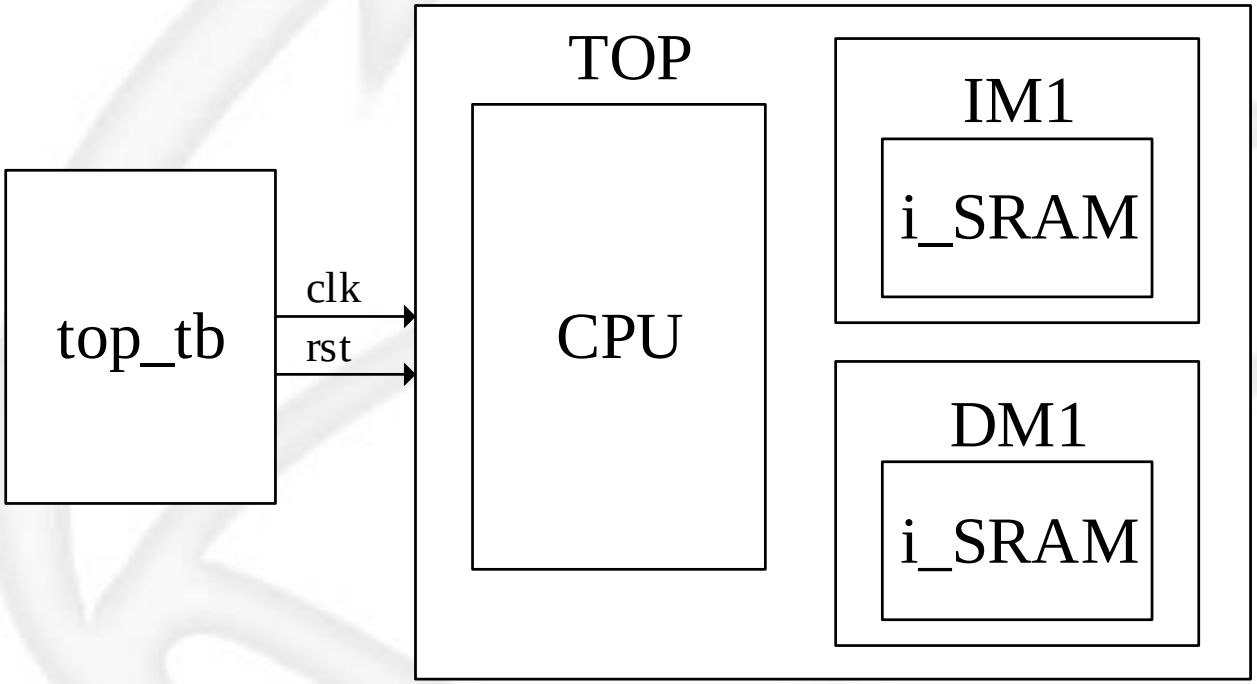


- Ensure the clock period in the sim/CYCLE file matches the value defined in sim/top_tb.sv

VLSI Design LAB

作業繳交注意事項

Testbench Structure



Module (1/2)

- Module name須符合下表要求

Category	Name			
	File	Module	Instance	SDF
RTL	top.sv	top	TOP	
RTL	SRAM_wrapper.sv	SRAM_wrapper	IM1	
RTL	SRAM_wrapper.sv	SRAM_wrapper	DM1	
RTL	TS1N16ADFPCLLLVTA512X45M4SWSHOD.sv	TS1N16ADFPCLLLVTA512X45M4SWSHOD	i_SRAM	

- 紫色部分為助教已提供或已定義好，請勿任意更改
- 其餘部分需按照要求命名，以免testbench抓不到正確的名稱

Module (2/2)

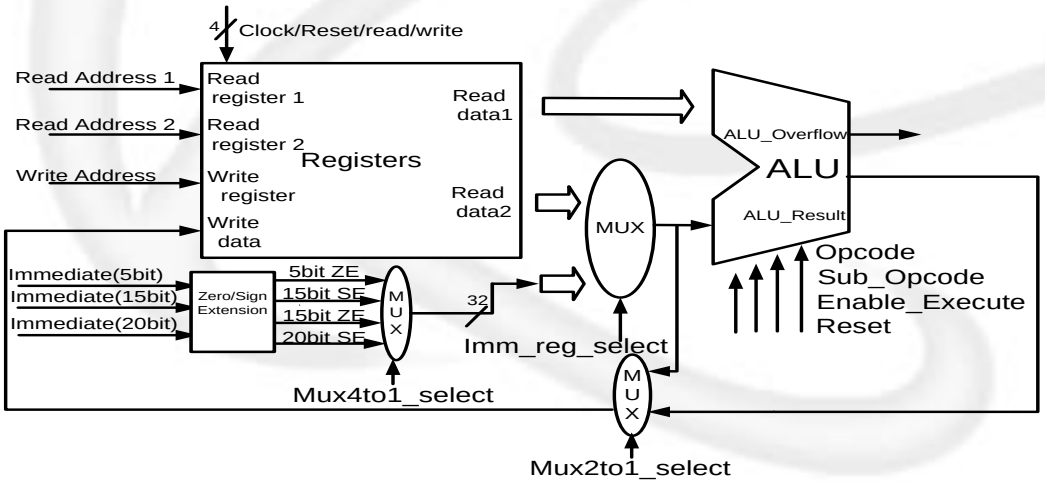
- Module port須符合下表要求

Module	Specifications			
top	**Name**	**Signal**	**Bits**	**Function explanation**
	clk	input	1	System clock
	rst	input	1	System reset (active high)
SRAM_wrapper	CLK	input	1	System clock
	CEB	input	1	Chip enable (active low)
	WEB	input	1	read:active high , write:active low
	BWEB	input	32	Bit write enable (active low)
	A	input	14	Address
	DI	input	32	Data input
	DO	output	32	Data output
TS1N16ADFPCLLLVTA512X45M4SWSHOD	**Memory Space**			
	MEMORY	logic	32	Size: [512][32]

- 紫色部分為助教已提供或已定義好，請勿任意更改
- 其餘部分需按照要求命名，以免testbench抓不到正確的名稱

Report (1/2)

- 請勿將code貼在.docx內
  - 請將.sv包在壓縮檔內，不可截圖於.docx中
- 需要Summary及Lessons learned
- Block diagram
  - 不必畫到gate level，除非該處的邏輯對於設計上有重要意義
  - 可用一個矩形標上名稱以及I/O代表一個 functional block
  - 呈現要點在於讓人較容易理解你的設計的架構
  - 可以使用Visio、Open Office Draw，或其他繪圖軟體



Report (2/2)

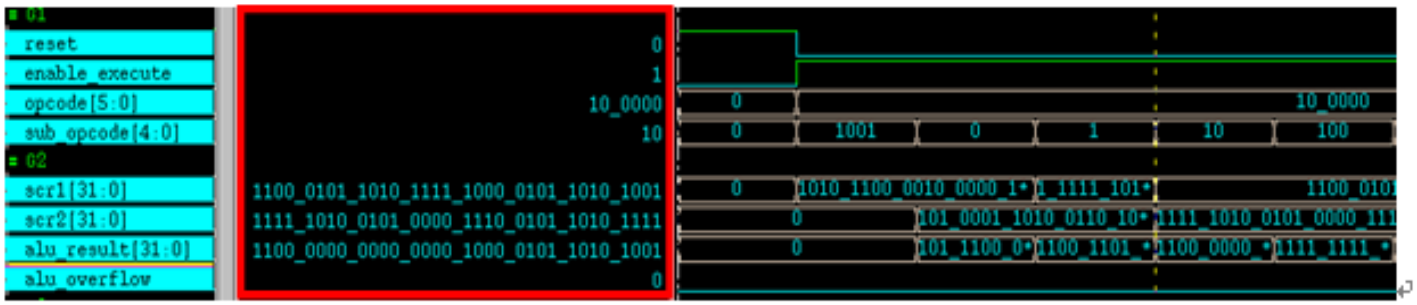
- 驗證波形圖
  - 保留完整訊號名稱以及訊號值
  - 輔以文字解釋該波形圖的操作
  - 可在波形圖加上標示輔助了解
  - 截圖裁減至合適大小

v. 測試AND的功能。當enable_execute為1時，表示alu開始做運算，opcode為100000，sub_opcode為00010，動作為AND。

scr1=32'b1100_0101_1010_1111_1000_0101_1010_1001，

scr2=32'b1111_1010_0101_0000_1110_0101_1010_1111，

結果為=32'b1100_0000_0000_0000_1000_0101_1010_1001。



繳交檔案

- 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

檔案結構 (1/2)

- N26XXXXXX.docx
  - ➔ Your report file
- src
  - ➔ Your source code (*.sv)
- include
  - ➔ Your definition code (*.svh)
- StudentID
  - ➔ Specify your Student ID number
- sim/CYCLE
  - ➔ Specify your clock cycle time
- sim/MAX
  - ➔ Specify max clock cycle number

- *N260XXXXX.tar* (**Don't** add version text in filename, e.g. *N260XXXXX_v1.tar*)
  - *N260XXXXX* (Main folder of this homework)
    - *N260XXXXX.docx* (Your homework report)
    - *StudentID* (Specify your student ID number in this file)
    - *Makefile* (You shouldn't modify it)
    - *src* (Your RTL code with *sv* format)
      - *top.sv*
      - *SRAM_wrapper.sv*
      - Other submodules (*.sv)
    - *include* (Your RTL definition with *svh* format, optional)
      - Definition files (*.svh)
    - *syn* (Your synthesized code and timing file, optional)
      - *top_syn.v*
      - *top_syn.sdf*
    - *script* (Any scripts of verification, synthesis or place and route)
      - Script files (*.sdc, *.tcl or *.setup)
    - *sim* (Testbenches and memory libraries)
      - *top_tb.sv* (Testbench. You can only modify CYCLE in tb)
      - *CYCLE* (Specify your clock cycle time in this file)
      - *MAX* (Specify max clock cycle number in this file)
      - *SRAM* (SRAM libraries and behavior models)
        - Library files (*.lib, *.db, *.lef or *.gds)
        - *TS1N16ADFPCLLLVTA512X45M4SWSHOD.sv (SRAM behavior model)*

檔案結構 (2/2)

- sim/prog0、prog2、prog4、prog5 、prog6
  - → Don't modify contents
- sim/prog1、prog3
  - → main.S
  - → main.c
    - Submit one of these
- Don't modify Makefile

- *prog0* (Subfolder for Program 0)
  - *Makefile* (Compile and generate memory content)
  - *main.S* (Assembly code for verification)
  - *setup.S* (Assembly code for testing environment setup)
  - *link.ld* (Linker script for testing environment)
  - *golden.hex* (Golden hexadecimal data)
- *prog1* (Subfolder for Program 1)
  - *Makefile* (Compile and generate memory content)
  - *main.S* * (Assembly code for verification)
  - *main.c* * (C code for verification)
  - *data.S* (Assembly code for testing data)
  - *setup.S* (Assembly code for testing environment setup)
  - *link.ld* (Linker script for testing environment)
  - *golden.hex* (Golden hexadecimal data)
- *prog2* (Subfolder for Program 2)
  - *Makefile* (Compile and generate memory content)
  - *main.c* (C code for verification)
  - *data.S* (Assembly code for testing data)
  - *setup.S* (Assembly code for testing environment setup)
  - *link.ld* (Linker script for testing environment)
  - *golden.hex* (Golden hexadecimal data)
- *prog3* (Subfolder for Program 3)
  - *Makefile* (Compile and generate memory content)
  - *main.S* * (Assembly code for verification)
  - *main.c* * (C code for verification)
  - *data.S* (Assembly code for testing data)
  - *setup.S* (Assembly code for testing environment setup)
  - *link.ld* (Linker script for testing environment)
  - *golden.hex* (Golden hexadecimal data)
- *prog4* (Subfolder for Program 4)
  - *Makefile* (Compile and generate memory content)
  - *main.c* (C code for verification)
  - *data.S* (Assembly code for testing data)
  - *setup.S* (Assembly code for testing environment setup)
  - *link.ld* (Linker script for testing environment)
  - *golden.hex* (Golden hexadecimal data)
- *prog5* (Subfolder for Program 5)
  - *Makefile* (Compile and generate memory content)
  - *main.c* (C code for verification)
  - *data.S* (Assembly code for testing data)
  - *setup.S* (Assembly code for testing environment setup)
  - *link.ld* (Linker script for testing environment)
  - *golden.hex* (Golden hexadecimal data)
- *prog6* (Subfolder for Program 6)
  - *Makefile* (Compile and generate memory content)
  - *main.c* (C code for verification)
  - *data.S* (Assembly code for testing data)
  - *setup.S* (Assembly code for testing environment setup)
  - *link.ld* (Linker script for testing environment)
  - *golden.hex* (Golden hexadecimal data)

繳交期限

- 2024/10/09 (三) 15:00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
- 作業二預計在2024/10/02 (三)會上傳，請務必加快作業一的設計時間，以免壓縮到作業二的時間

注意事項

- 本次作業合成的部分有計入PA(Performance & Area)，表現較為優異者將有較高的評分(30%)
- 作業部分有任何問題請在moodle上的作業討論區發問並可參考其他人是否有類似的問題，助教信箱恕不回覆。
- 在Script/DC.sdc中，可以更改clk period範圍，請注意最大的接受值為10.0，超過此範圍者恕不納入計分。
- 本次作業需在SoC環境下模擬，請同學務必在SoC教室執行模擬，若是助教在評分作業時於SoC無法成功模擬則以0分計算。

VLSI Design LAB

Thanks for your participation and attendance !!