Homework 1 Explanation

Instructor：Lih-Yih Chiou

TA : 陳正崙 Liam

Date : 09/10/2025

Outline

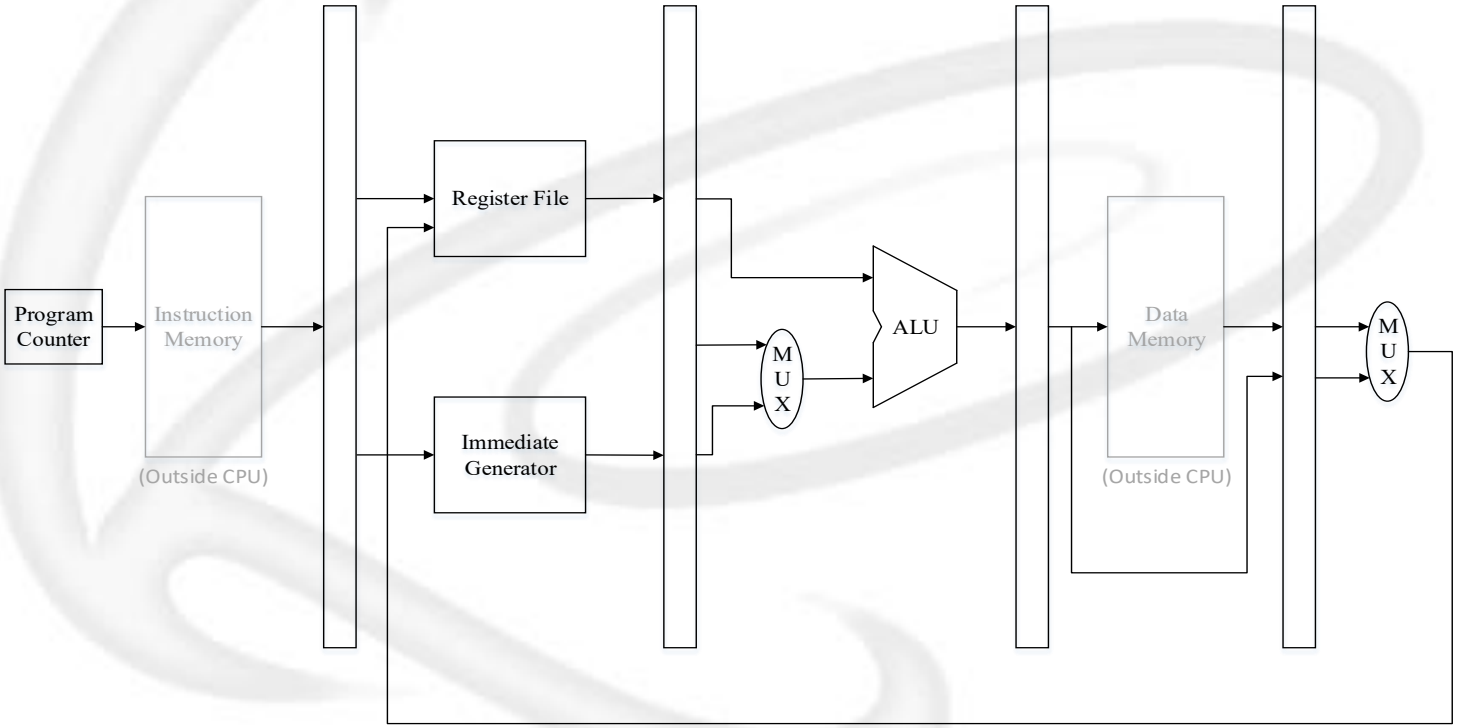
- Design Spec & Behavior
- Validation Programs
- Grading
- Submission Guidelines

LPHPLAB
VLSI Design LAB

Design Spec & Behavior

Target Design

- According to the design specification provided in the PDF, design and implement a 5-stage pipeline CPU based on the RISC-V ISA, with an integrated branch predictor.



Design Specifications

- Implement the 49 instructions as listed. The number of pipeline stage is 5.
- General Register File size: 32x32-bit
  - ➔ x0 is hardwired to 0.
- Floating point Register File size: 32x32-bit
- Memory Size: (64KB)
  - ➔ Depth: 16K
  - ➔ Width/Entry: 32bit
- Timescale: 1ns/10ps
- Maximum Clock period: 10ns (100MHz)
- Control and Status Register: 2x64-bit

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

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	$M[rs1+imm]_h = rs2_h$

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

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.

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]

In this lab, only these 4 instructions need to be implemented.

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

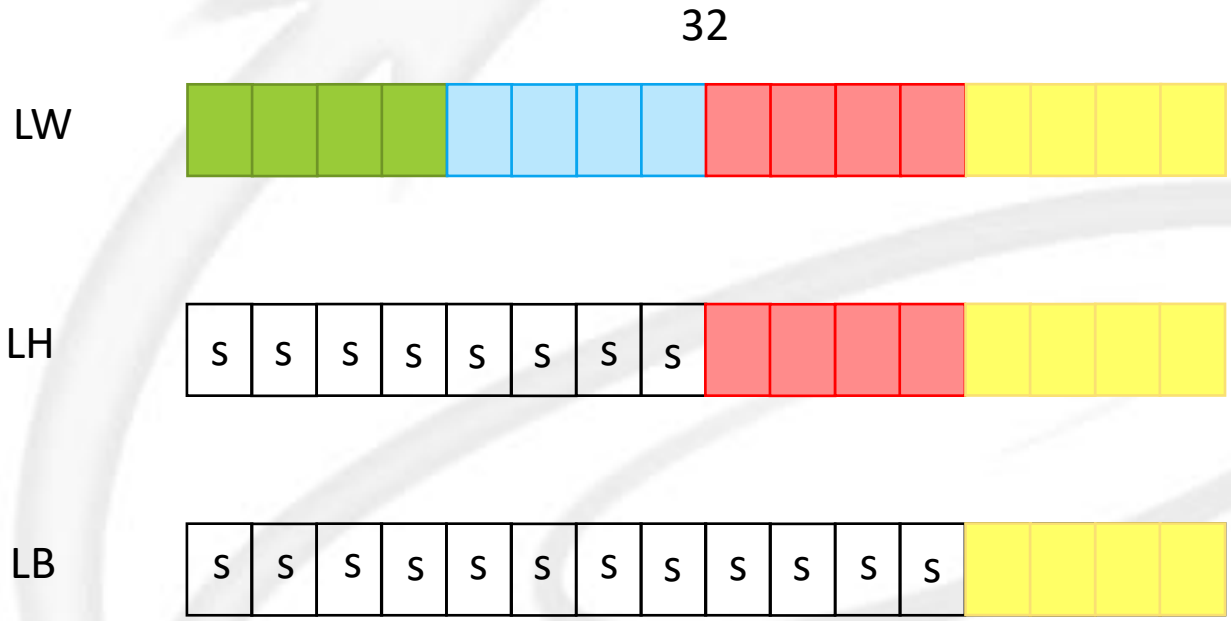
```
auipc a0,0x8
addi  a0,a0,124 # 8100 <__sbss_end>
auipc a1,0x8
addi  a1,a1,112 # 80fc <_test_start+0xfc>
li  a2,0
jal ra,104 <fill_block>
```

  - 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]

Read/Write: Word/Half-word/Byte

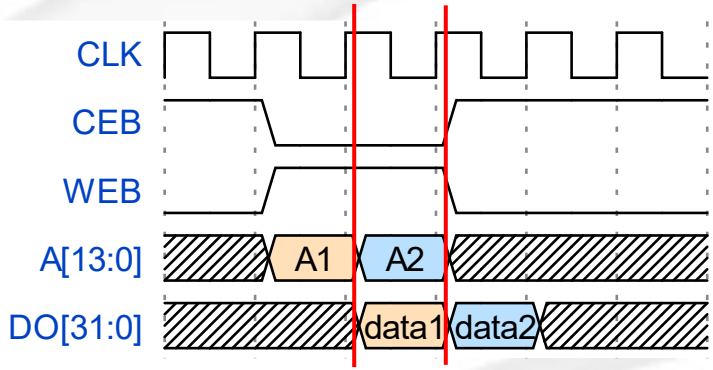
- If the instruction is used to read half-word or bytes, all other bytes left must be filled with 0s or sign bits.



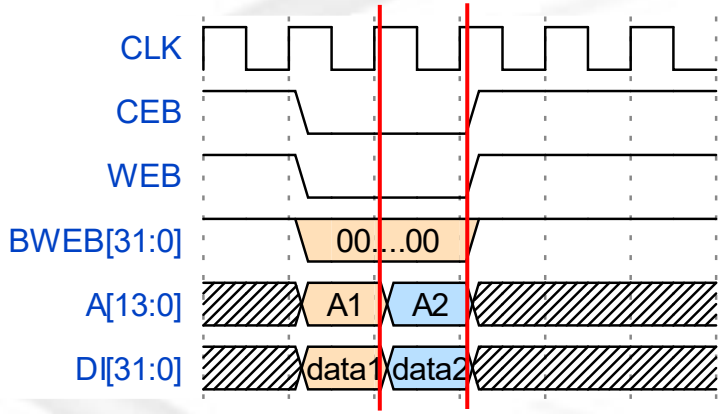
Note: BWEB specifies the bits to be masked or bypass.

SRAM Behavior

- Read operation



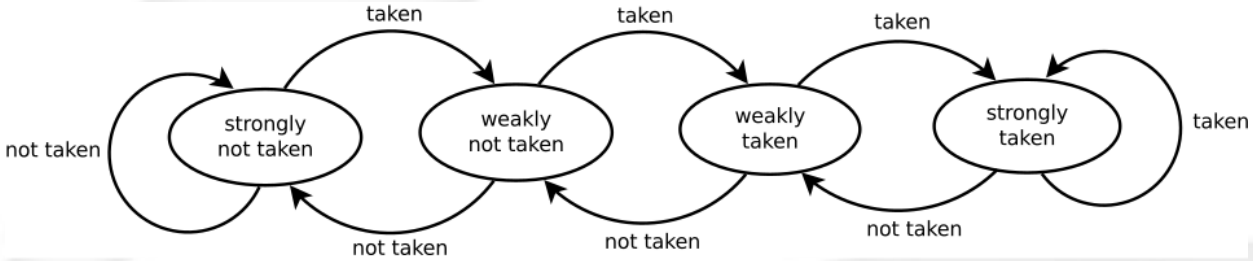
- Write operation



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

Branch Predictor

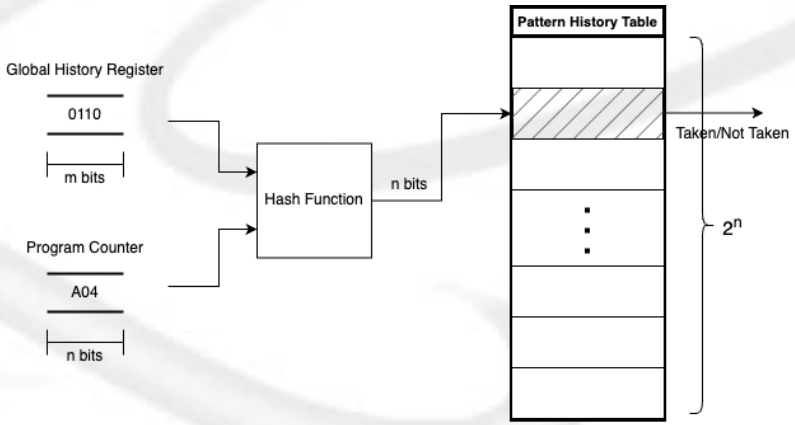
- Bimodal Predictor (Basic)



- 4 state FSM.
- When a branch instruction is evaluated, the corresponding state machine is updated.
- Based on the current state, predict whether the current instruction will branch.

Branch Predictor

- Global Branch Correlation (Intermediate)
  - Determine an m-bit register to store the last m branch outcomes (Global History Register GHR).
  - Construct a Pattern History Table (PHT) with 2^n entries, which is indexed by potential values of the GHR hashed with PC.
  - Each entry of the PHT is a bimodal predictor.
    - The hash methodology can be designed by yourself. Ex.
      - 1. Take N low PC bits that vary by branch.
      - 2. fold GHR(if M > N): split GHR into N-bit chunks and XOR them together. If M < N: zero-extend.
      - 3. XOR the N bits GHR & PC to select which index PHT to use



Branch Predictor

- Other Algorithm (Advanced)
  - You may adopt any other algorithm to implement the branch predictor. However, its complexity must be at least at the level of Global Branch Correlation.
  - Provide a logical analysis explaining the advantages, disadvantages, and application scenarios of your chosen algorithm. The **more comprehensive** your explanation, the **higher the score**.
- Report
  - In your branch predictor report, clearly explain how the predictor is integrated into your existing CPU architecture.
  - Analyze the TA-provided test program (Prog0 is sufficient) and provide insights into the scenarios where the branch predictor offers the most benefit.
  - Include a rough estimation of the performance improvement—specifically, the reduction in execution cycles compared to a baseline CPU without branch prediction.
  - The grading will be heavily based on the clarity and depth of your explanations. You may also include simulation result comparisons between your CPU with & without the branch predictor to further support your analysis, but this is only optional.

LPHPLAB
VLSI Design LAB

Validation Programs

Programs

- prog0
  - Testing 49 instructions (Provided by TA)
- prog1
  - Sort Algorithm (Complete on your own)
- prog2
  - Perform multiplication without using the MUL instructions. (Provided by TA)
- prog3
  - Greatest common divisor (Complete on your own)
- prog4
  - Test rdinstret, rdinstreth, rdcycle, and rdcycleh using a factorial C program. (Provided by TA)
- prog5
  - Perform multiplication using the MUL instructions. (Provided by TA)
- prog6
  - Perform computations using floating-point instructions.(Provided by TA)

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	

Simulation

- You can only modify those red box in sim/top_tb.sv and script/DC.sdc file during post-synthesize simulation. The maximum allowed value for the clk_period is **10.0**

top_tb.sv | CYCLE | Makefile | Makefile

```
1   `timescale 1ns/10ps
2   `define CYCLE 10.0 // Cycle time
3   `define MAX 100000 // Max cycle
4   `ifdef SYN
5   `include "top_syn.v"
```

```
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
```

- Ensure the clock period in the script/DC.sdc file matches the value defined in sim/top_tb.sv

LPHPLAB
VLSI Design LAB

Grading

Grading

- Simulation: 56%

Program	RTL	SYN
Prog0	3%	5%
Prog1	3%	5%
Prog2	3%	5%
Prog3	3%	5%
Prog4	3%	5%
Prog5	3%	5%
Prog6	3%	5%

- Report: 13% + 11%(Branch Prediction)
  - 13%: Design report, explanations, summary, lesson learned…
  - 11%: Include explanations of your branch predictor algorithm

Algorithm	Credit
Bimodal Predictor	1%~3%
Global Branch Correlation	3% ~ 7%
Other Algorithm	7~11% (Base on comprehensive & complexity)

Grading

- Performance & Area: 20%
  - PA score = Prog0 Simulation Time * Total Cell Area (Synthesized)
  - Note: Only those who passes all SYN programs are allowed to compete in the PA ranking.

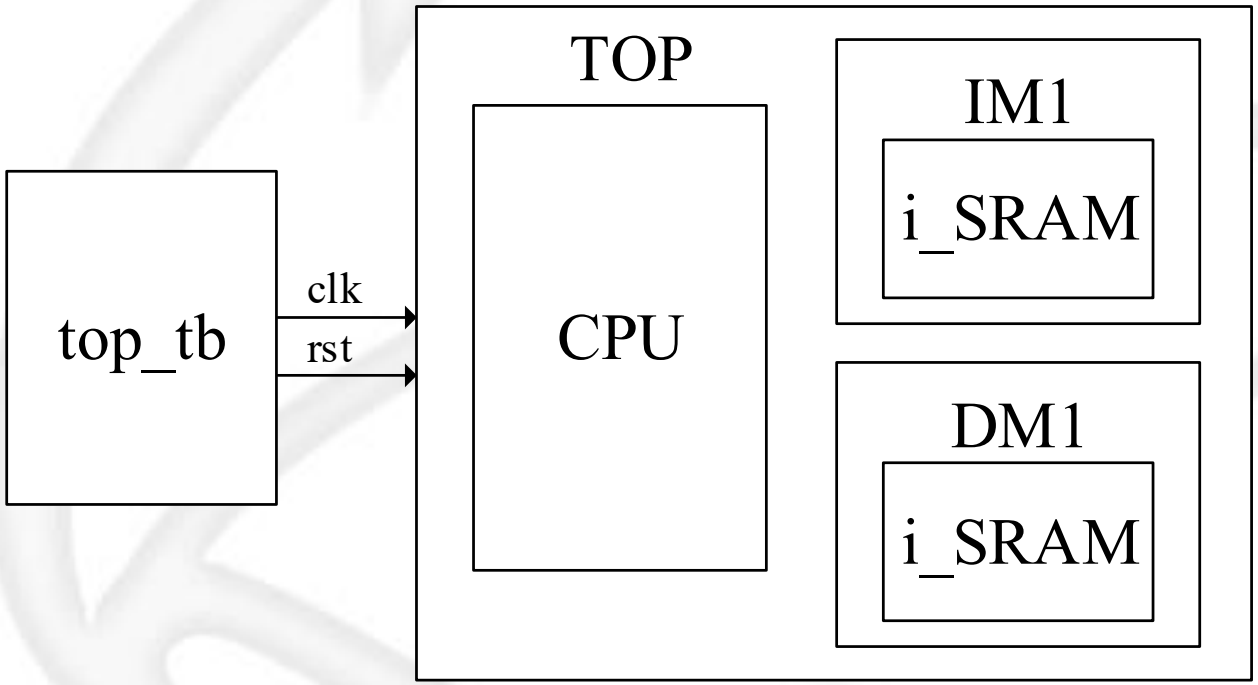
Rank	Credit
Top 5%	20
Top 10%	18
Top 20%	16
Top 30%	14
Top 40%	12
Top 50%	10
Top 60%	8
Top 70%	6
Top 80%	4
Under 80%	2

No pa
wr

LPHPLAB
VLSI Design LAB

Submission Guidelines

Testbench Structure



Module (1/2)

- Module name must meet the spec below

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	

- The purple parts are provided or already defined by the TA. Please do not modify them.
- The remaining parts must be named according to the requirements to ensure the testbench can detect the correct names.

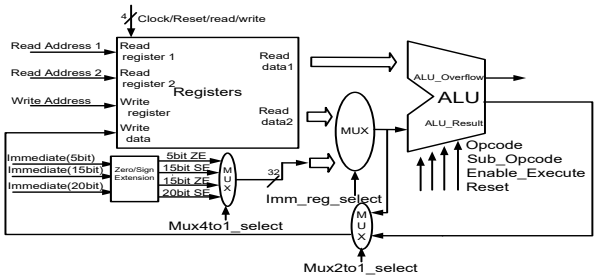
Module (2/2)

- Module ports must comply with the requirements listed in the table below.

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]

- The purple parts are already defined by the TA. Please do not modify them.
- The remaining parts must be named according to the requirements to ensure the testbench can detect the correct names.

Report (1/2)



- Do not paste the code in a .docx file.
  - Please include the .sv file in the compressed archive; screenshots in the .docx file are not allowed.
- **Summary & Lessons learned**
- Block diagram
  - There is no need to draw down to the gate level unless the logic at that point is of significant importance to the design.
  - You may use a single rectangle with labels and I/O to represent a functional block.
  - The key is to present your design in a way that makes the architecture easier to understand.
  - You may use Visio, OpenOffice Draw, draw.io, or other drawing software.
    - Hand-drawn figures are not allowed, including drawings made on an iPad.

Report (2/2)

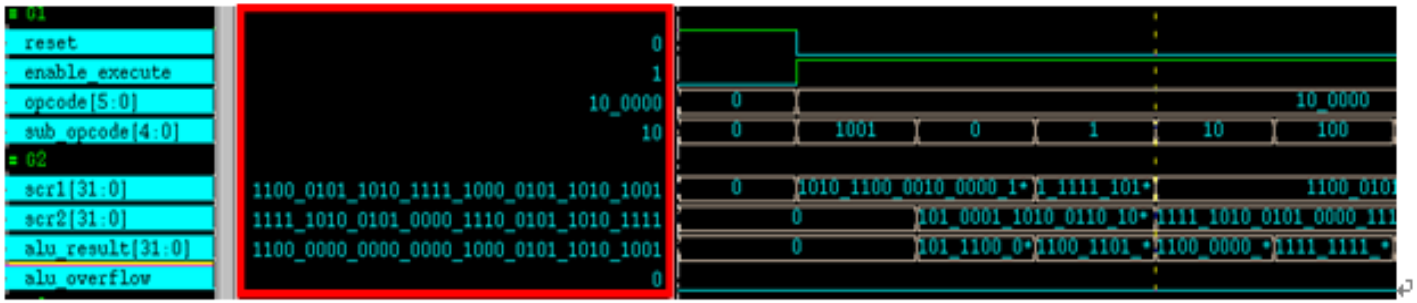
- Validation Waveform
  - Keep the complete signal names and signal values.
  - Provide a textual explanation of the operation shown in the waveform diagram.
  - You may add annotations to the waveform diagram to aid understanding.

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。



Submission Files

- Compress the files according to the directory structure into a ".tar" format.
  - ➔ In the Homework main folder (N2XXXXXX), use make tar to generate the tar file will meet the requirements.
- Follow the file hierarchy specified in the assignment instructions.
- Do not include any files that are not required for submission
  - ➔ In the Homework main folder (N26XXXXXX), use make clean to remove unnecessary files.
- Be sure to confirm that the submitted files can compile and function correctly in the SoC workstations.
  - ➔ Failure to compile will result in a score of zero.
- Do not use any generator to produce code and then modify it.
- Plagiarism is prohibited. (No copy)

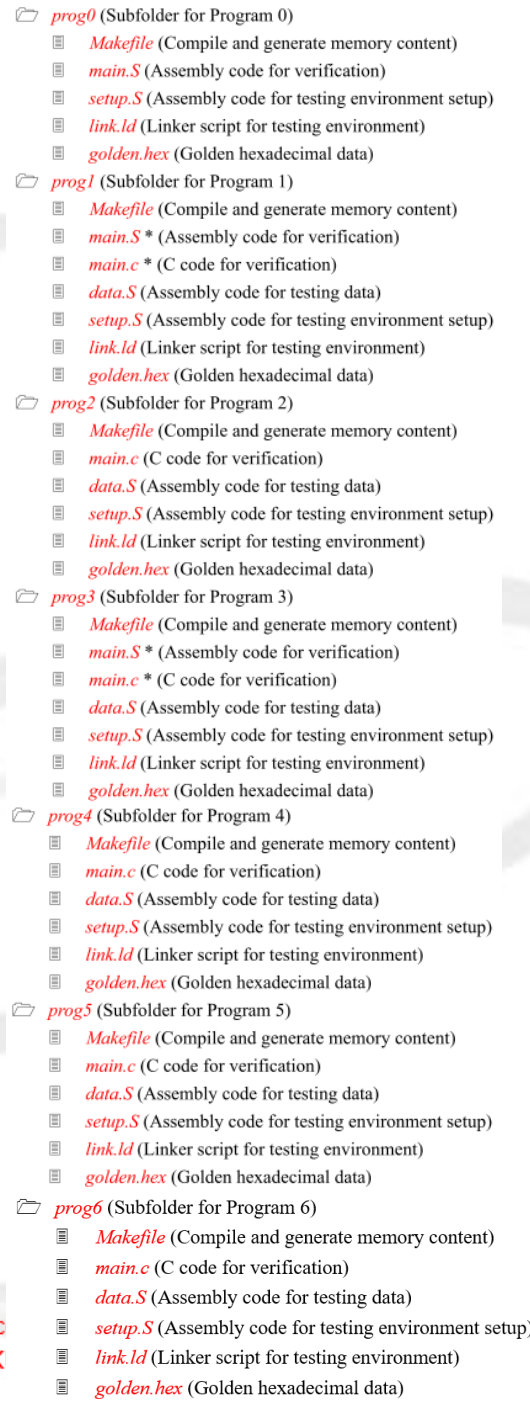
File Hierarchy (1/2)

- StudentID.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)*

File Hierarchy (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



Submission deadline

- Upload to Moodle before 2025/10/01 (Wed) 15:00
- Late submissions will not be accepted, so please be mindful of the deadline.
  - ➔ Moodle will only retain your latest uploaded file. The filename should simply be "StudentID.tar" without any version numbers.
- Homework 2 is scheduled to be uploaded on 2025/10/01(Wed).

Notes

- The synthesis portion of this assignment will be included in the PA (Performance & Area) evaluation, with higher-performing designs receiving a higher score.
- If you have any questions about the assignment, please post them in the assignment discussion on Moodle and check whether others have encountered similar issues. Questions sent to the TA's email will not be answered.
- In Script/DC.sdc, you may adjust the clock period range; however, please note that the maximum acceptable value is **10.0**. Any submission exceeding this limit will not be counted toward the score.
- This assignment must be simulated in the SoC environment. Please ensure that you perform the simulation in the SoC classroom. If the TA is unable to successfully run the simulation in the SoC environment during grading, the score will be recorded as zero.

LPHPLAB
VLSI Design LAB

Thanks for your participation and attendance !!