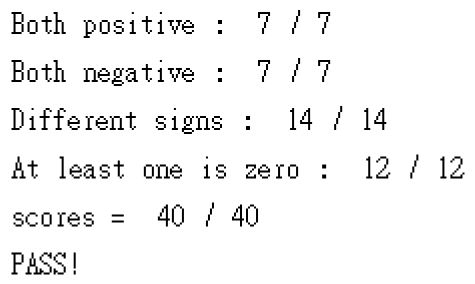
Lab 2 Basic

1. Sign Bit Detector with 1-Cycle Delay (40%)

Design a simple circuit that checks the sign of two signed 8-bit inputs (A, B) and outputs a **2-bit code** based on their signs.

1. To prevent potential timing issues, the output must be stored in a **flip-flop (FF),** introducing a **1-cycle delay**.
2. The output is updated at the **positive edge of the clock**.
3. The reset is **synchronous, active-high**. When reset is asserted, the output should be **0**.
4. Complete the Verilog template lab2_1.v with the provided testbench and pattern file (lab2_1_t.v, pattern_A.dat).
   Refer to the Appendix for instructions on adding the pattern files to simulation sources.
5. Your design must pass simulation, which will display the following PASS message:



```
Both positive :  7 / 7
Both negative :  7 / 7
Different signs :  14 / 14
At least one is zero :  12 / 12
scores =  40 / 40
PASS!
```

IO List and Specification:

Signal	I/O	Bit Width	Description
clk	input	1	Clock (positive-edge triggered)
rst	input	1	Synchronous active-high reset
A	input	8	Signed input
B	input	8	Signed input
out	output	2	Encoded output

Input (A , B)	**Output Code**
Both positive	2’b01
Both negative	2’b10
Different signs	2’b11
At least one is zero	2’b00

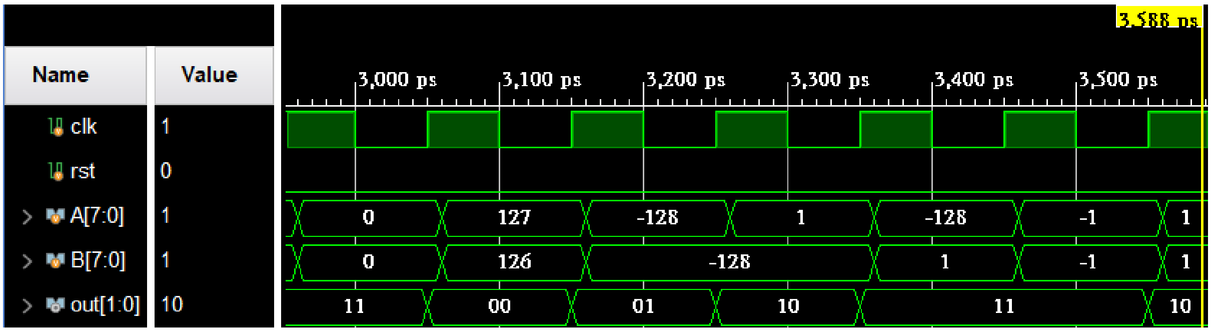
The output must appear **1 cycle after inputs are given**.

Example

Cycle	**A**	**B**	**out (after 1 cycle)**
1	8’d0	8’d0	x
2	8’d127	8’d126	2’b00
3	-8’d128	-8’d128	2’b01
4	8’d1	-8’d128	2’b10
5	-8’d128	8’d1	2’b11

x indicates “don’t care”.

Refer to the following waveform for a correct output example:



The radix of **A** and **B** is signed decimal; the radix of **out** is binary.

Grading:

Name	**Score**
Both positive	7%
Both negative	7%
Different signs	14%
At least one is zero	12%

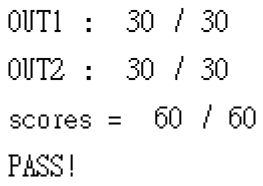
Notes

- **Do not modify** the testbench or patterns.
- The TA will use the same testbench with additional hidden patterns to test your design for the basic lab. The grading is based on correctness. Pass/Fail messages of the testbench are only to assist the debugging.
- Pay attention to **signed number handling**.

2. Impulse-Triggered Arithmetic with Multi-Cycle Delay (60%)

Design a circuit that captures two signed 8-bit inputs (A, B) when impulse is asserted (i.e., impulse == 1), and produces two results with different delays.

1. The output is updated at the **positive edge of the clock**.
2. The reset is **synchronous, active-high**. When reset is asserted, the output should be **0**.
3. Complete the Verilog template lab2_2.v with the given testbench and pattern files (lab2_2_t.v, pattern_B.dat).
   Refer to the Appendix for instructions on adding the pattern files to simulation sources.
4. Your design must pass simulation, which will display the following PASS message:



```
OUT1 :  30 / 30
OUT2 :  30 / 30
scores =  60 / 60
PASS!
```

IO List and Specification:

Signal	I/O	Bit Width	Description
clk	input	1	Clock (positive-edge triggered)
rst	input	1	Synchronous active-high reset
impulse	input	1	Triggers a calculation
A	input	8	Signed operand (2’s complement)
B	input	8	Signed operand (2’s complement)
OUT1	output	16	Concatenation of A and B
OUT2	output	8	(ABS(A * B)) % 128

Once impulse is 1, the circuit captures A and B and performs the following operations:

Timing	**Operation Description**
After 1 cycle	OUT1 = Concatenation of A and B
After 3 cycles	OUT2= (ABS(A * B)) % 128

Each trigger (impulse==1) starts a new calculation.

- On every positive edge of clk, when impulse == 1, store the current values of A and B.
- After 1 clock cycle, output OUT1 = Concatenation of A and B

  at all other cycles, OUT1 = 0.

- After 3 clock cycles, output OUT2 = (ABS(A * B)) % 128;

  at all other cycles, OUT2 = 0.

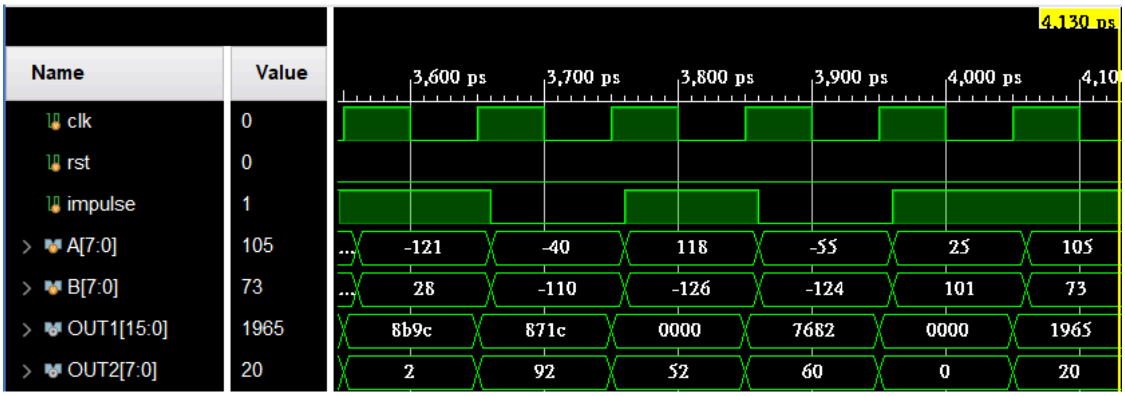
- If rst == 1, both OUT1 and OUT2 are synchronously reset to 0.
- All operations use signed (2’s complement) values.

Example

Cycle	impulse	A	B	OUT1 (after 1 cycle)	OUT2 (after 3 cycles)
1	1	-8’d121	8’d28	x	x
2	0	-8’d40	-8’d110	16’h871c	x
3	1	8’d118	-8’d126	16’h0000	x
4	0	-8’d55	-8’d124	16’h7682	8’d60
5	1	8’d25	8’d101	16’h0000	8’d0
6	1	8’d105	8’d73	16’h1965	8’d20

x indicates “don’t care”.

Refer to the following waveform for a correct output example:



The radix of **A** , **B** and **OUT2** is signed decimal; the radix of **OUT1** is hexadecimal.

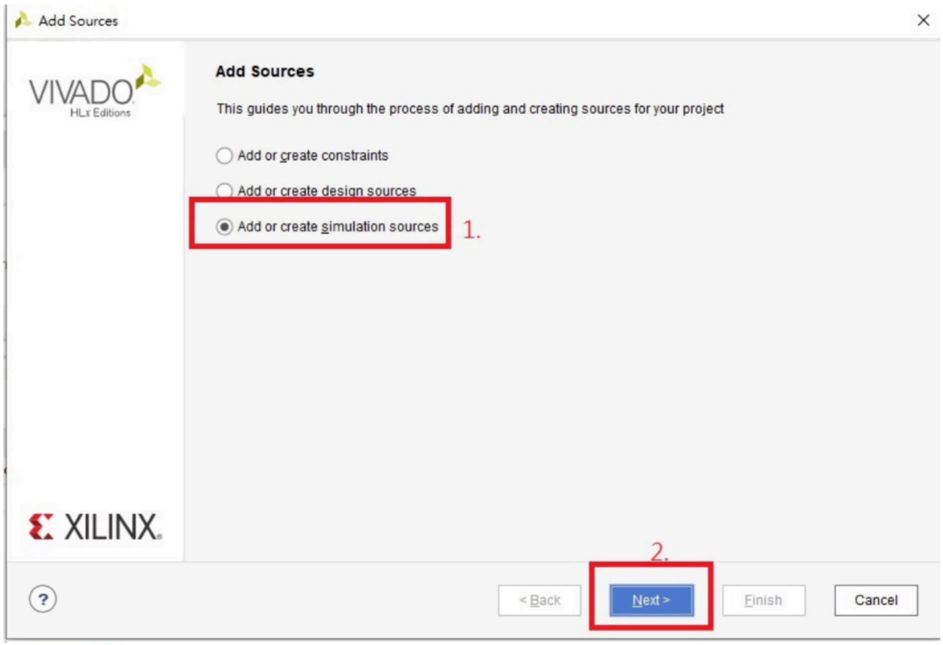
Grading:

Name	Score
OUT1	30%
OUT2	30%

Notes

- **Do not modify** the testbench or patterns.
- The TA will use the same testbench with additional hidden patterns to test your design for the basic lab. The grading is based on correctness. Pass/Fail messages of the testbench are only to assist the debugging.
- Ensure proper **signed arithmetic handling** and timing.

Appendix: Adding Pattern Files to Vivado Simulation



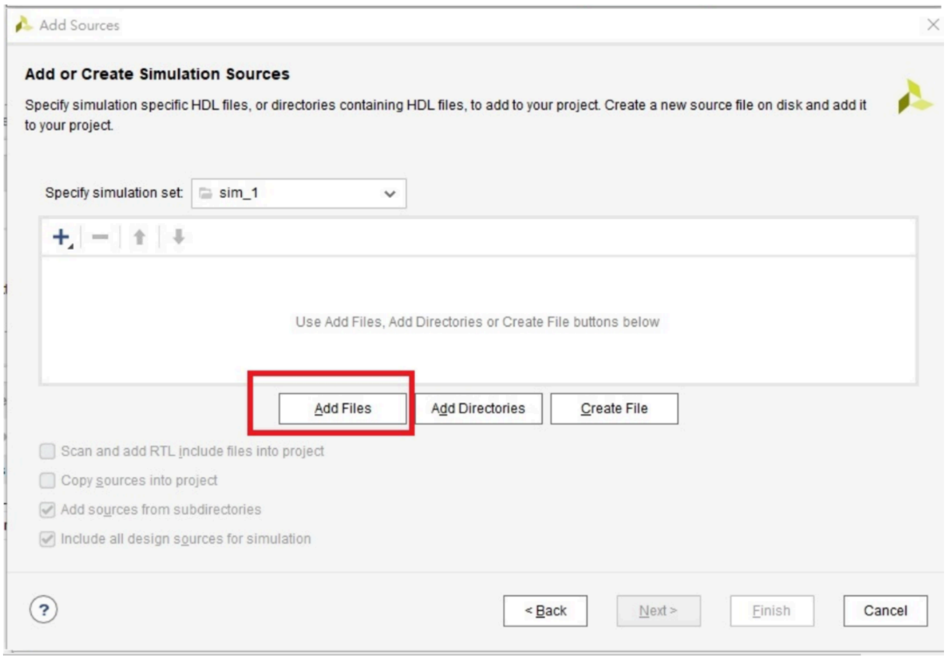
Add Sources

This guides you through the process of adding and creating sources for your project

○ Add or create constraints

○ Add or create design sources

● Add or create simulation sources 1.



Add or Create Simulation Sources

Specify simulation specific HDL files, or directories containing HDL files, to add to your project. Create a new source file on disk and add it to your project.

Specify simulation set: sim_1

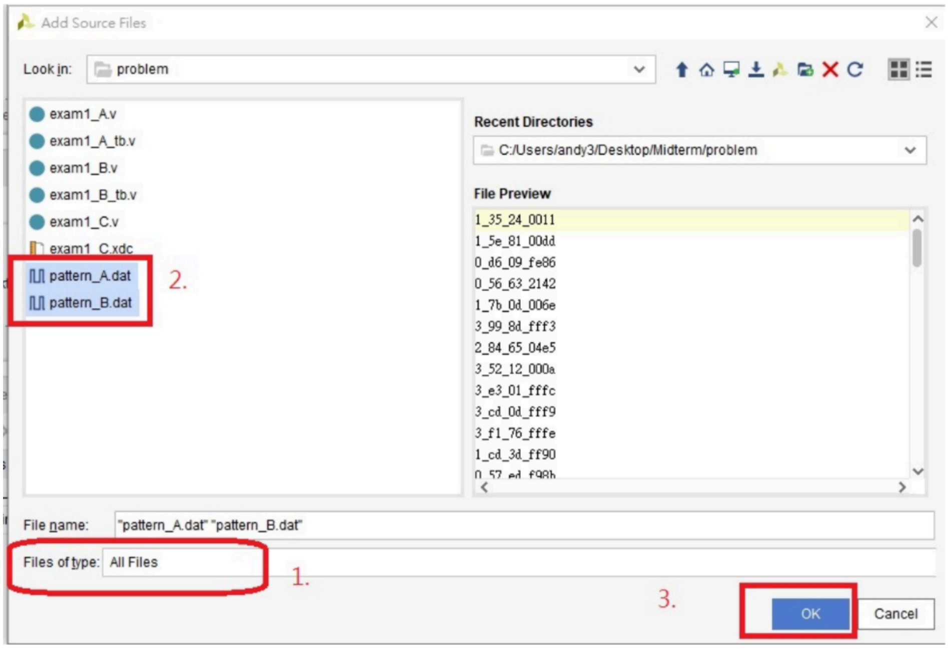
Use Add Files, Add Directories or Create File buttons below

□ Scan and add RTL include files into project

□ Copy sources into project

☑ Add sources from subdirectories

☑ Include all design sources for simulation



C:/Users/andy3/Desktop/Midterm/problem

```
1_35_24_0011
1_5e_81_00dd
0_d6_09_fe86
0_56_63_2142
1_7b_0d_006e
3_99_8d_fff3
2_84_65_04e5
3_52_12_000a
3_e3_01_fffc
3_cd_0d_fff9
3_f1_76_fffe
1_cd_3d_ff90
0_57_ed_f98b
```

File name: "pattern_A.dat" "pattern_B.dat"

Files of type: All Files