Lab 6
Introduction to Verilog – 3
Exercise & Report

Summary

- In this lab, you are required to modify your multiplier in Lab 4 to support pipeline.
- In this lab, you are required to design a 4-input special calculator, with the following function :
  - Calculate the Summation of 4 inputs.
  - Calculate the Multiplication of 4 inputs with your multiplier in Lab 4.
  - Find the Max and Minimum number of 4 inputs.
- You are required to write your own testbench to verify your design.

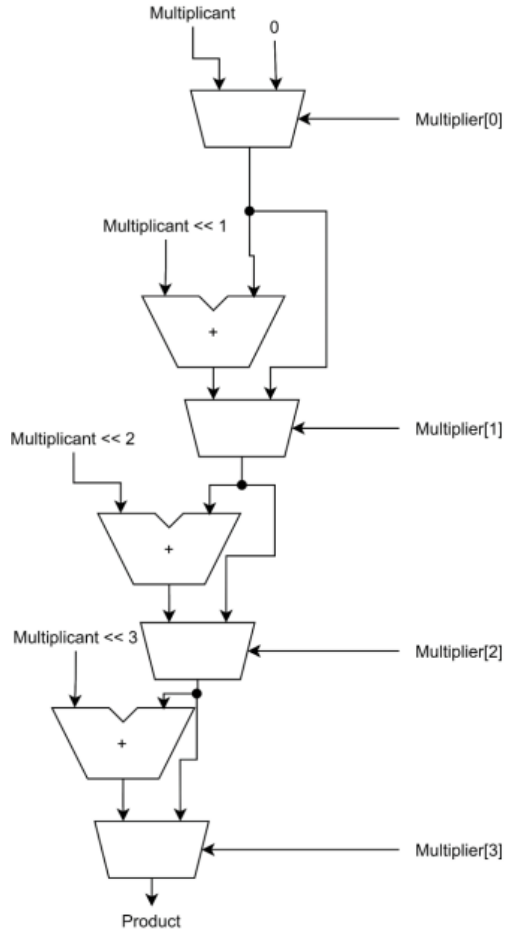
Exercise 1 - Multiplier

- In exercise 1, you are required to :
  - Modify your multiplier in Lab 4 to support

    8bit multiplicand and 8bit multiplier. The product

    should be 16 bits
  - Put a delay of 5ns to your adder.
  - Modify the architecture to support pipeline operation.

    (Insert registers between each adders)



Exercise 2 - Special Calculator

- In exercise 2, you need to design an special calculator which can computer summation, multiplication, max and min.
- After reset, the testbench will assert *data_valid* signal, *op* signal and the first data simultaneously. Then, sequentially pass the remaining data, until all 4 data are passed.
- After the last data, testbench will drop all control signals and wait for results.

Exercise 2 - Special Calculator

- Operation of calculator are defined as follow :

op	*Function*
2'b00	Summation, i.e., a+b+c+d
2'b01	Multiplication, i.e., a*b*c*d
2'b10	Max value
2'b11	Minimum value

Exercise 2 – Special Calculator

- After passing all 4 data, your design should start computing the result.
- When all computations are completed, your design should output the result as well as *result_valid* signal.
- Your design should be able to accept next group of data in next cycles.

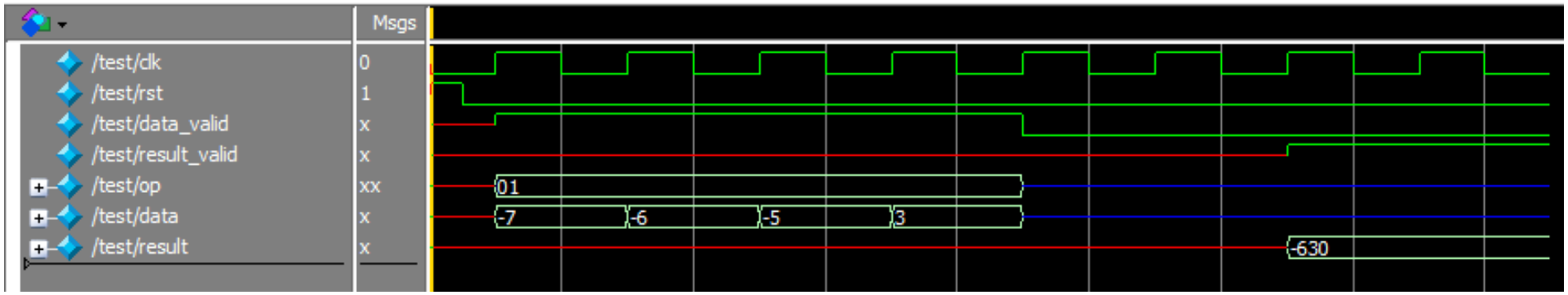
Exercise 2 – Special Calculator

- I/O interface

Name	I/O	Width	Description
clk	I	1	System clock signal. This system is synchronized with the positive edge of the clock.
rst	I	1	Active-high asynchronous reset signal.
data_valid	I	1	Input data is valid.
op	I	2	Operation
data	I	4	Input data.
result_valid	O	1	Result data is valid
result	O	16	Result data.

Exercise 2 – Special Calculator

- Timing diagram



Exercise 2 - Limitation

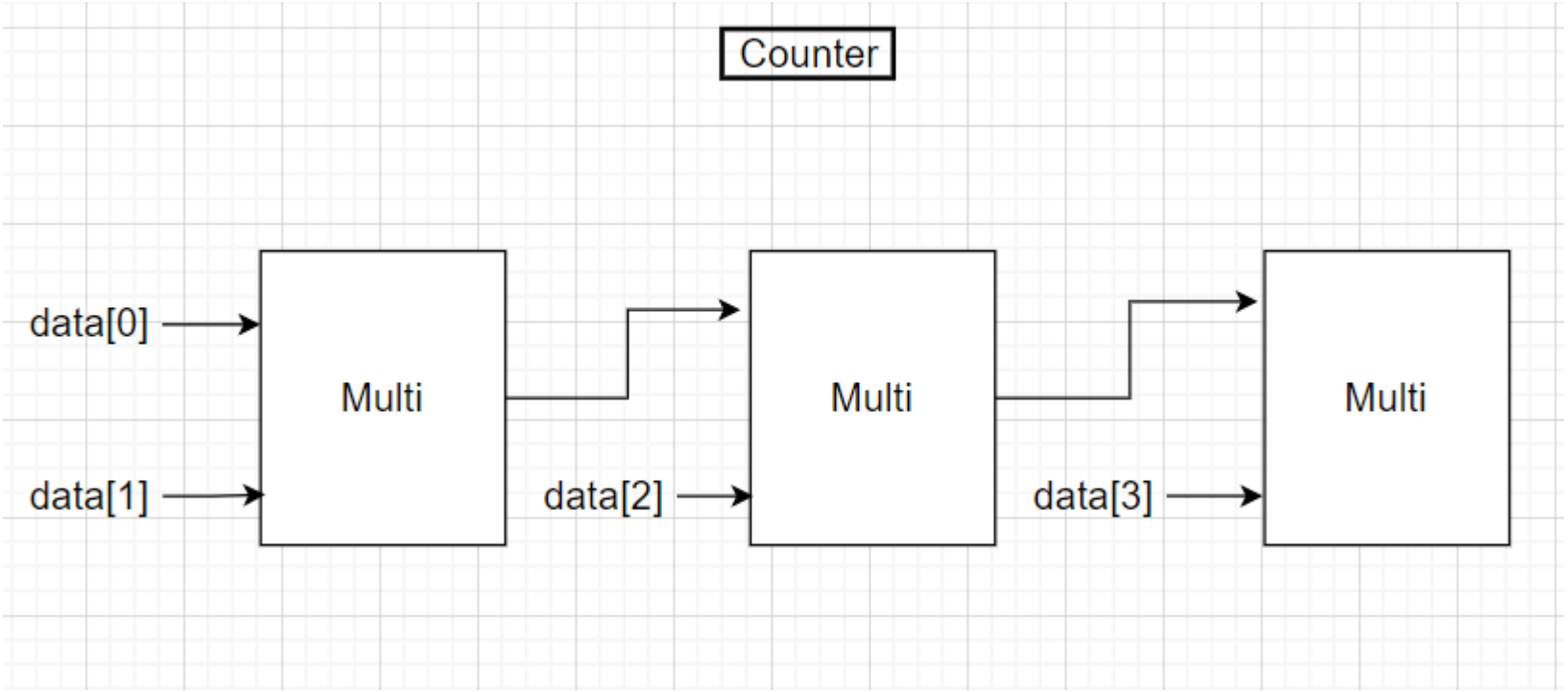
- In exercise 2, you should reuse your design in exercise 1. If you simply use behavioral statement, i.e., a*b*c*d, you can only receive partial credits.
- 就算exercise1的pipeline沒有做出來，也還是儘可能用自己lab4的電路，TA還是會給部份分數。
- In this lab, all calculations should be considered as signed operation.

Exercise 2 – coding style

- In this exercise, coding styles will be consider in grading(not much, but you should try to follow it).
- Here are some coding conventions you should follow :
  - Use macro(define) to name your state machine
  - Do not use unsynthesizable syntax/semantic in your design(except tb).
  - Only one module in a file.

Hints

- Use a counter to let your FSM know when to move.
- Insert multiplier into your design.



Exercise 2 - testbench

- In this Lab, you need your own testbench to verify your design.
- The testbench should generate clock with a period of 10ns .
- 4 input patterns are provided, use each input pattern to run over 4 kinds of operation, so there will be totally 16 times of calculation.
- After one calculation, verify the results, then apply next calculation.

Exercise 2 - testbench

- Use ASCII Art to create your special testbench result ~ (not mandatory)
  - https://www.asciiart.eu/

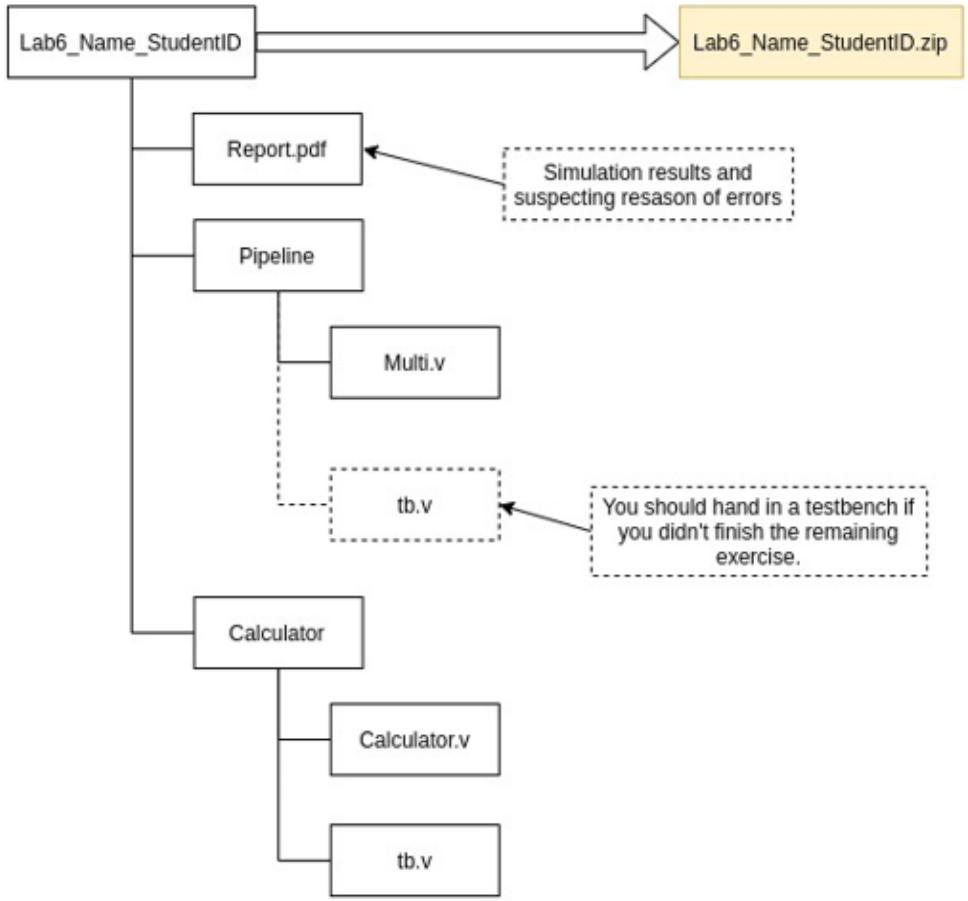
Report requirement

- Design explanation. If there is a state machine in your design, use a graph to illustrate it.
- Simulation result (terminal & waveform)

Exercise 1 - Multiplier

- If you didn't get to finish the remaining part, please provide a testbench that can show what you have done, and explain it in your report.
- You can reuse the testbench we provided in Lab 4.

File


Lab6_Name_StudentID
Lab6_Name_StudentID.zip
Report.pdf
Simulation results and suspecting resason of errors
Pipeline
Multi.v
tb.v
You should hand in a testbench if you didn't finish the remaining exercise.
Calculator
Calculator.v
tb.v

File format

File name	Module name	Instance name
tb.v	*tb*	
Calculator.v	*Calculator*	*cal*
Multi.v	*Multi*	*multi*