NCTU-EE DCS – 2017

Lab06 Exercise

Design: Arithmetic Logic Unit (ALU)

Data Preparation

1. Extract test data from TA's directory:

   % tar xvf ~dcsta02/LAB06.tar

2. The extracted LAB directory contains:
   a. Exercise/ :your design

Design Description and Example Wave

An arithmetic logic unit (ALU) is a digital electronic circuit that performs arithmetic and bitwise logical operations on integer binary numbers. In this exercise, a string of instructions will be given. Your job is to decode these instructions and execute.

The following is the definition of the instruction.

instruction [21:20]	**instruction** [19:10]	**instruction** [9:0]
Opcode	Data of Rs	Data of Rt
2-bit	10-bit	10-bit

Register (Rs) and Register (Rt) represent the data of registers. The instruction takes 2 bits to store the opcode bit and takes 20 bits to store the data. Rs and Rt reserve 10 bits to store data.

e.g. Rs = 10'b0000000000 means the data of Rs is 0.
Rt = 10'b0000000001 means the data of Rt is 1, and so on.

The following is the definition of the mode.

Opcode[1:0]	Operation	Note
2’b00	|Rs – Rt|	Absolute value after Subtraction
2’b01	Rs * Rt	Multiplication
2’b10	(Rs , Rt)	Greatest Common Factor (G.C.D)
2’b11	(Rs[9:5]+Rs[4:0])^2 + (Rt[9:5]+Rt[4:0])^2	Addition after square

EX1:
Mode = 2’d0、Rs = 10’d30 and Rt = 10’d10，
the result is out = 20’d20

EX2:
Mode = 2’d1、Rs = 10’d30 and Rt = 10’d10，
the result is out = 20’d300

Inputs

1. All input signals will be **synchronized** at **negative edge** of the clock.

I/O	Signal name	Description
Input	**clk**	Clock
Input	**rst_n**	Asynchronous active-low reset. You should set all your outputs to 0 when rst_n is low
Input	**in_valid**	instruction[21:0] is valid when in_valid is high.
Input	**instruction[21:0]**	The unsigned **instruction[21:0]** is valid only when **in_valid** is high, and is delivered 1 cycle.

Outputs

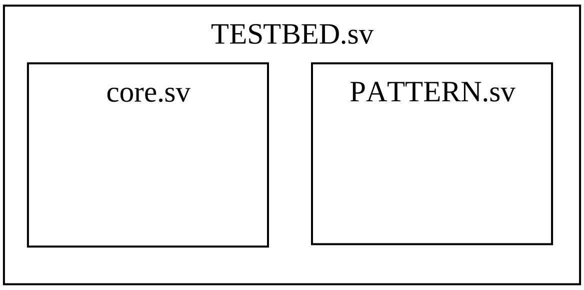
1. Output signals are synchronized at clock positive edge.
2. The TA's pattern will capture your output for checking at clock negative edge.

I/O	Signal name	Description
Output	**out[19:0]**	**out** should print the answer for 1 cycle.
Output	**out_valid**	**out_valid** should be low after initial reset and not be raised when **in_valid is high**. **out_valid** should be set to high when your output value is ready, and will be high for **1** cycle.

Specifications

1. Top module name： **alu** (Filename: **alu.sv**)
2. Input ports: **clk, rst_n, in_valid, instruction[21:0]**
3. Output ports: **out[19:0], out_valid**
4. It is an **asynchronous active-low reset** and **positive edge clk** architecture
5. The clock period of the design is **3ns**.
6. The input delay is set to **1.5ns**.
7. The output delay is set to **1.5ns**, and the output loading is set to **0.05**.
8. The next group of inputs will come in **1~9** cycles after your **out_valid** is over.
9. After synthesis, please check **syn.log** file that can not include any **Latch & Error**.
10. After synthesis, you can check alu.area and alu.timing. The area report is valid when the slack in the end of **timing report** is **non-negative**.
11. The latency for one case should be smaller than **1000 cycles**.

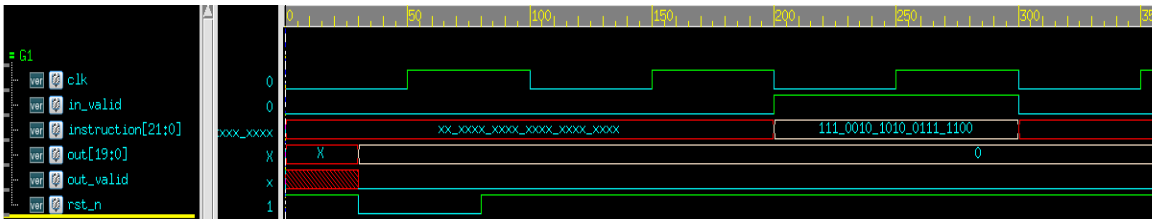
Block Diagram

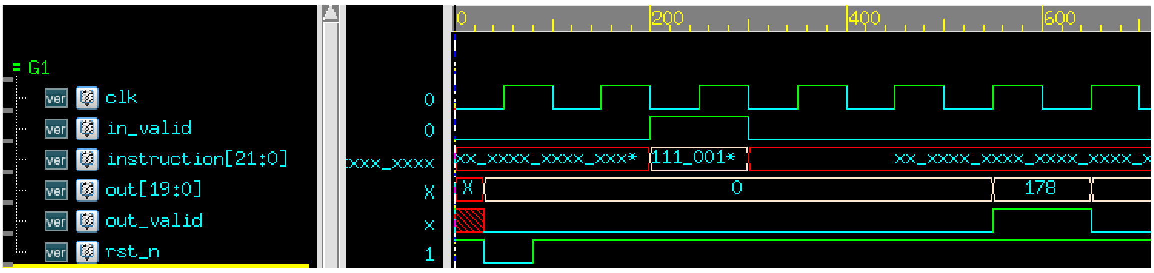


Note

1. Grading policy:
   a. Pass the RTL simulation
   b. Synthesis successful
      (**no Latch and Error**)
      (**slack in the timing report is non-negative**)
   c. Using For loop is not allowed.
   d. Plagiarism is not allowed.
2. Template folders and reference commands:
   a. 01_RTL/ (RTL simulation) **./01_run**
   b. 02_SYN/ (Synthesis) **./01_run_dc**
      (Check the design which contain **latch** or not in **syn.log**
      (Check the design's timing in /Report/**core.timing**)

Sample Waveform


G1
clk
in_valid
instruction[21:0]
out[19:0]
out_valid
rst_n
0
0
xxx_xxxx
X
x
1
0
50
100
150
200
250
300
xx_xxxx_xxxx_xxxx_xxxx_xxxx
111_0010_1010_0111_1100
X
0


G1
clk
in_valid
instruction[21:0]
out[19:0]
out_valid
rst_n
0
0
xxx_xxxx
X
x
1
0
200
400
600
xx_xxxx_xxxx_xxx*
111_001*
xx_xxxx_xxxx_xxxx_xxxx_x
X
0
178