NCTU-CS Digital System Lab.

Lab06 Decode ALU(DALU)

Data Preparation

Use and learn ISE tool and ChipScope Pro to accomplish this design, and make sure you are familiar with tool and setting:

1. Download Lab06.zip from e3 platform.
2. Use ISE tool to run Simulation.
3. Program your code (DALU.v)on Spartan 3E to view Wave by ChipScope Pro.

Design Description and Examples

An arithmetic logic unit (ALU) is a digital electronic circuit that performs arithmetic and bitwise logical operations on integer binary numbers. In this Lab, a string of instructions will be given. Your job is to decode these instructions and execute. You should try to design a simple ALU. The basic required instruction are following:

Type	- Format – (19-bits)			
L	opcode (3-bits)	ssssss (6-bits)	tttttt (6-bits)	func (4-bits)
O	opcode (3-bits)	ssssss (6-bits)	tttttt (6-bits)	llll (4-bits)
I	opcode (3-bits)	ssssss (6-bits)	iiiiiiiii (10-bits)	

L: logic O: operation I: 10bits number operation

A instruction[18:0] can be divided into few parts as above.

ssssss, tttttt, llll, iiiiiiiii represent the 6bits and 4bits and 10bits signed numbers, respectively. Decode the opcode & func by the following rule to compute answer.

In this Lab, there are 19bits instructions meaning 19bits numbers. TA will offer testbend containing a counter to count 0~2^{19}-1 and assign in **instruction[18:0]** to check your design.

Function Name	Meaning	Type	Instruction Binary Encode
AND	ssssss & tttttt (bit-wise)	L	000-ssssss-tttttt -0000
OR	ssssss | tttttt (bit-wise)	L	000-ssssss-tttttt -0001
XOR	ssssss ^ tttttt (bit-wise)	L	000-ssssss-tttttt -0010
ADD	ssssss + tttttt	O	000-ssssss-tttttt -0011
SUB	ssssss – tttttt	O	000-ssssss-tttttt -0100
Mult	ssssss * tttttt * llll	O	001-ssssss-tttttt -llll
Square	(ssssss + tttttt + llll)^2	O	010-ssssss-tttttt -llll
ADDI	ssssss + iiiiiiiii	I	011-ssssss -iiiiii-iiii
SUBI	ssssss – iiiiiiiii	I	100-ssssss -iiiiii-iiii

P.s.: llll are LLLL not 1111

Input：

instruction[18:0]: 001 000011 000001 0011

Mult: 3*1*3 = 9

Output：

9

You should output 9 in 1 cycle with out_valid high.

Your goal is to compute these operations by above rules and output the correct answer.

Inputs

1. **instruction[18:0]** is valid while **in_ valid** is high.
2. All inputs will be changed at clock ***negative*** edge.
3. All operations are signed.

Input Signals	**Bit Width**	**Description**
clk	1	Positive edge trigger clock
rst	1	active-high **synchronous** reset
instruction[18:0]	19	19 bits input
in_valid	1	high when **instruction[18:0]** is valid

Outputs

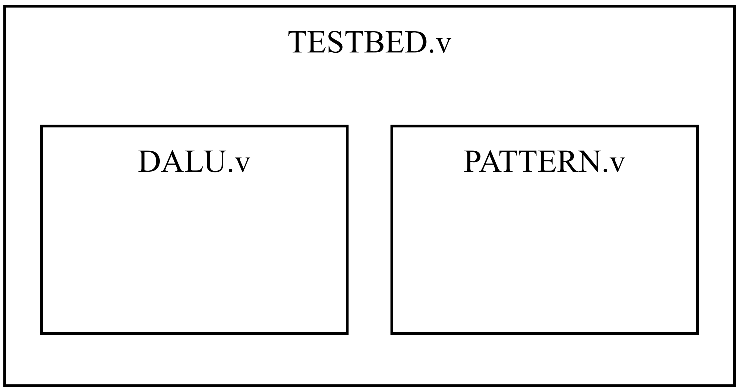
1. Your answer should be output at **out[15:0]** in only **1 cycle**
2. **out [15:0]** is signed integer.
3. **out_valid** should be low and **out** should be set to zero after initial reset.
4. **out_valid** should be set to high when output value is valid.
5. All outputs are synchronized at clock ***positive*** edge.
6. Test pattern will check whether your answer is correct or not at clock **negative edge** when **out_valid** is high.

Output Signals	**Bit Width**	**Description**
out [15:0]	16	**out[15:0] are synchronous to the positive edge**
out_valid	1	high when **out[15:0]** is valid

Specifications

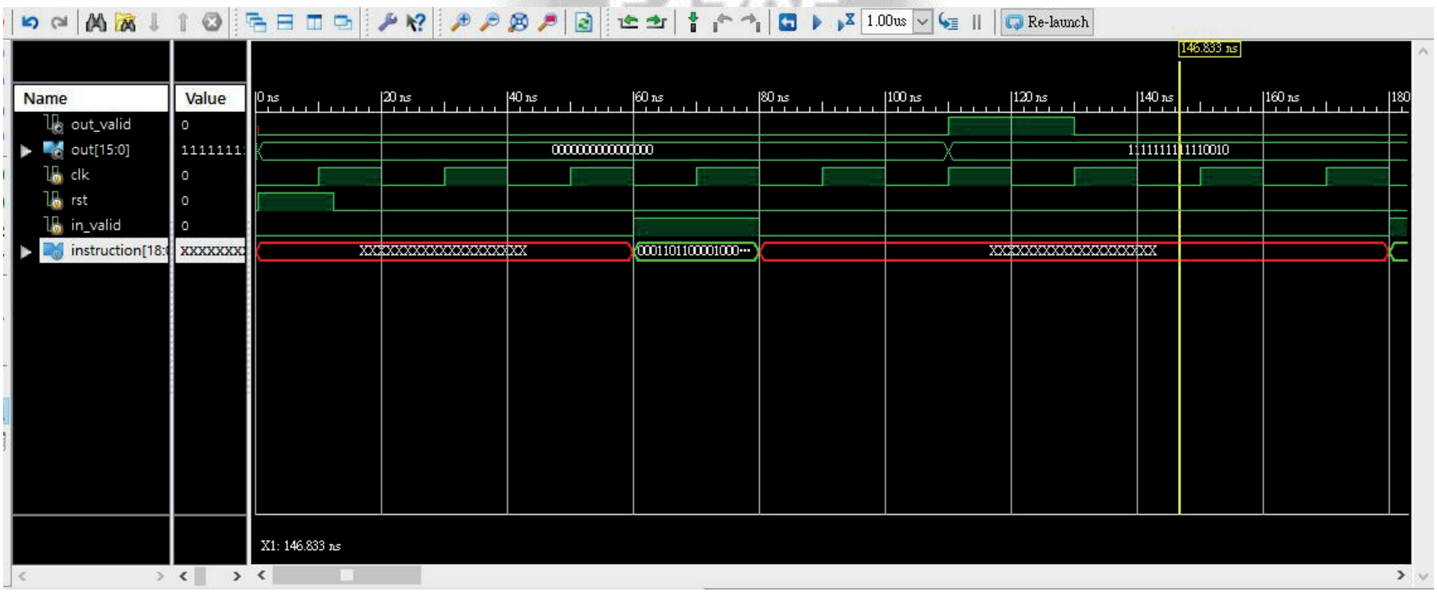
4. Top module name : **DALU** (File name : **DALU.v**)
5. Input pins: **clk, rst, in_valid, instruction[18:0]**.
6. Output pins: **out_valid, out[15:0]**.
7. **out_valid** should not be raised when **in_valid** is high (when **in** data is transferring).
8. It is **active-high synchronous** reset.
9. All operations are signed.
10. Grading policy:
    1. RTL(40%)
    2. Program your code on Spartan 3E(20%)
    3. Use tool at TA demo time(30%)
    4. Question(10%)

Block Diagram



Note

1. Use ISE tool to run Simulation.
2. Program your code on Spartan 3E to view Wave by ChipScope Pro.
3. Please add your student ID and name to the file name of .v file before upload file on e3 platform:
   DALU_0556123_陳小明.v
4. Sample waveform:


1.00us
Re-launch
146.833 ns
Name
Value
out_valid
0
out[15:0]
1111111
clk
0
rst
0
in_valid
0
instruction[18:
xxxxxxx
0 ns
20 ns
40 ns
60 ns
80 ns
100 ns
120 ns
140 ns
160 ns
180
0000000000000000
1111111111110010
XXXXXXXXXXXXXXXXXXXX
0001101100001000···
XXXXXXXXXXXXXXXXXXXX
X1: 146.833 ns