Computer-Aided VLSI System Design
Homework 1: Arithmetic Logic Unit

TA: 李諭奇 d06943027@ntu.edu.tw **Due Tuesday, Oct. 26, 14:00**

TA: 羅宇呈 f08943129@ntu.edu.tw

Data Preparation

1. Decompress 1101_hw1.tar with following command

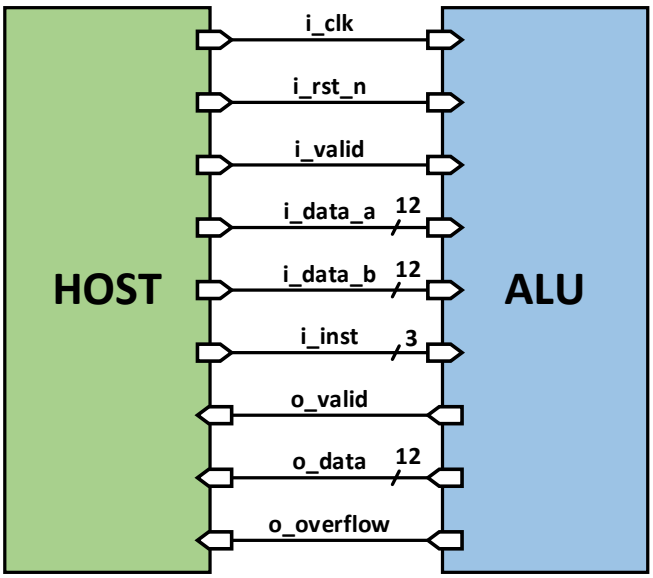
```
tar -xvf 1101_hw1.tar
```

Files/Folder	Description
alu.v	Your design
testbench.v	File to test your design
pattern/	9 instruction tests for verification
01_run	NCVerilog commad
99_clean	Command for cleaning temporal files

Introduction

The Arithmetic logic unit (ALU) is one of the components of a computer processor. The ALU has math, logic, and some designed operations in the computer. In this homework, you are going to design an ALU with some special instructions, and use the ALU to compute input data to get the correct results.

Block Diagram



Specifications

1. Top module name: alu
2. Input/output description:

Signal Name	I/O	Width	Simple Description
i_clk	I	1	Clock signal in the system.
i_rst_n	I	1	Active **low** asynchronous reset.
i_valid	I	1	The signal is **high** if input data is ready
i_data_a	I	12	Signed input data with 2's complement representation (7-bit integer + 5-bit fraction).
i_data_b	I	12	
i_inst	I	3	Instruction for ALU to operate.
o_valid	O	1	Set **high** if ready to output result.
o_data	O	12	Result after ALU processing with 2's complement representation (7-bit integer + 5-bit fraction).
o_overflow	O	1	Set **high** if overflow happened.

3. All inputs are synchronized with the negative edge clock.
4. All outputs should be synchronized at clock **rising** edge. (Flip-flops are added before outputs)
5. You should set all your outputs to be zero when i_rst_n is low. Active low asynchronous reset is used and only once.
6. The i_valid will turn to **high** in one cycle for ALU to get i_data_a, i_data_b and i_inst.
7. The i_valid will turn to **high** in random.
8. Your o_valid should be turned to high for only **one cycle** for every o_data and o_overflow.
9. The testbench will get your output at negative clock edge to check the answer if your o_valid is **high**.
10. If an overflow result is generated after ALU processing, the testbench will not check your o_data when o_valid is **high**.
11. You can raise your o_valid at any moment. TA will check your answer when o_valid is high.

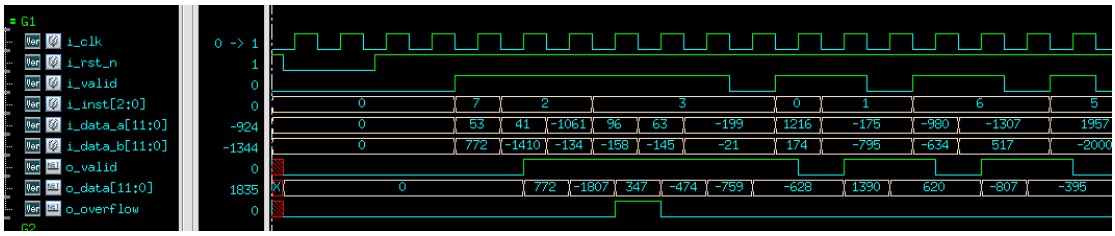
Design Description

1. The followings are the functions of instructions you need to design for this homework:

i_inst [2:0]	Operation	Description
000	Signed Addition	o_data = i_data_a + i_data_b
001	Signed Subtraction	o_data = i_data_a - i_data_b
010	Signed Multiplication	o_data = i_data_a * i_data_b
011	MAC	o_mult = i_data_a * i_data_b $o\_data_{new} = o\_mult + o\_data_{old}$
100	XNOR	o_data =(i_data_a $\oplus$ i_data_b)'
101	ReLU	If i_data_a > 0, o_data = i_data_a Otherwise, o_data = 0
110	Mean	o_data = (i_data_a + i_data_b)/2
111	Absolute Max	o_data = absmax (i_data_a, i_data_b) If abs(i_data_a) = abs(i_data_b), o_data = abs(i_data_a)

2. Considerations between digital systems and signals:
   a. Bit-wise operation for instruction 100.
   b. For instruction 011, only need to accumulate data with continuous MAC instructions. Every continuous MAC instruction set start with 0 to accumulate.
   c. Overflow may be happened for operating instruction 000, 001, 010 and 011. If the results overflow, turn o_overflow to high. For instruction 011, turn every o_overflow to high once the overflow is detected for the continuous MAC instruction set (Overflow: The result of the operation is greater than the maximum range that can be represented by the number of bits, resulting in an incorrect output result.)
   d. Before and after the operation, if the total number of bits of the variable (reg or wire) is inconsistent, the system presets to complement the zero extension. Please use sign extension for the signal at the appropriate time.
   e. For fixed-point operation, please pay attention to the position of separation between integer and fraction after ALU operation.
   f. For instructions 010 and 011, the result need to be rounded before output.
   g. For instruction 110, chop off the least significant bit after ALU operation.

Sample Waveform



Submission

1. Create a folder named **studentID_hw1**, and put all below files into the folder
   - alu.v

   Note: Use **lower case** for the letter in your student ID. (Ex. d06943027_hw1)

2. Compress the folder **studentID_hw1** in a **tar file** named **studentID_hw1_v*k*.tar** (***k* is the number of version, *k* =1,2,…)**

```
tar -cvf studentID_hw1_vk.tar studentID_hw1
```

   TA will only check the last version of your homework.

   Note: Use **lower case** for the letter in your student ID. (Ex. d06943027_hw1_v1)

3. Submit to FTP
   - IP: 140.112.175.68
   - Port: 21
   - Account: 1101cvsd_student
   - Password: ilovecvsd

Grading Policy

1. TA will run your code with following format of command. Make sure to run this command with no error message.

```
ncverilog testbench.v alu.v +define+I0 +access+rw
```

2. Pass all the instruction test to get full score.
   - Released pattern **80%**
     - I0, I1: 5%
     - I2~I8: 10%
   - Hidden pattern **20%**
     - Score with pass ratio of 10000 instructions
3. Delay submission
   - In one day: **(original score)*0.7**
   - In two days: **(original score)*0.4**
   - More than two days: **0 point** for this homework
4. Lose **3 point** for any wrong naming rule.

Note

1. Reference for 2'complement:
   https://en.wikipedia.org/wiki/Two%27s_complement