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

TA: 李其祐 r11943123@ntu.edu.tw
TA: 蔡宇軒 f07943171@ntu.edu.tw

Due Tuesday, Mar. 21st, 13:59

Data Preparation

1. Decompress 1112_hw1.tar with following command

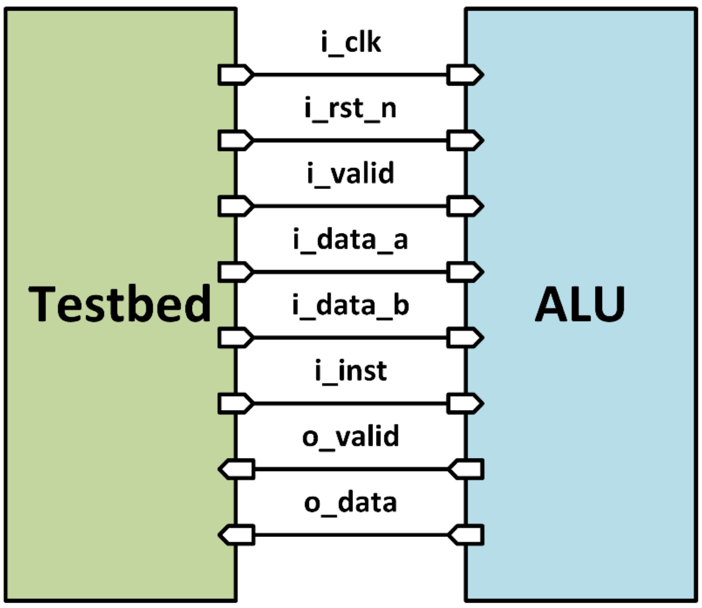
```
tar -xvf 1112_hw1.tar
```

Files/Folder	Description
alu.v	Your design
testbench.v	File to test your design
pattern/	11 instruction tests for verification
01_run	NCverilog simulation command
99_clean	Command for cleaning temporal files

Introduction

The Arithmetic logic unit (ALU) is one of the components of a computer processor. In this homework, you are going to design an ALU, which is able to compute the special operations from RISC-V.

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	10	1. For instruction 0000~0100, signed input data with 2's complement representation. (4-bit integer + 6-bit fraction) 2. For instruction 0101~1001, no fractional part (10-bit number)
i_data_b	I	10	
i_inst	I	4	Instruction for ALU to operate
o_valid	O	1	Set **high** if ready to output result
o_data	O	10	1. For instruction 0000~0100, result after ALU processing with 2's complement representation (4-bit integer + 6-bit fraction) 2. For instruction 0101~1001, no fractional part (10-bit number)

3. All inputs are synchronized with the negative clock edge.
4. All outputs should be synchronized at clock **rising** edge. (Flip-flops are added before outputs)
5. Active low asynchronous reset is used and only once. (You should set all your outputs to be zero when i_rst_n is low)
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 pulled high in random.
8. Your o_valid should be pulled high for only **one cycle** for every o_data.
9. The testbench will get your output at negative clock edge to check the answer when o_valid is **high**.
10. You can raise your o_valid at any moment. TA will check your answer when o_valid is high.

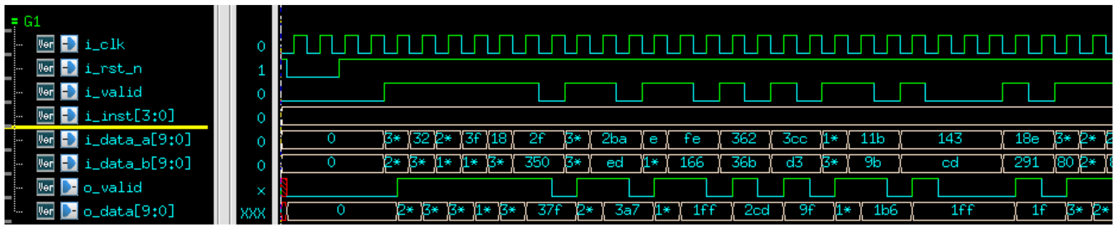
Design Description

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

i_inst [3:0]	Operation	Description
4’b0000	Signed Addition	o_data = i_data_a + i_data_b
4’b0001	Signed Subtraction	o_data = i_data_a - i_data_b
4’b0010	Signed Multiplication	o_data = i_data_a * i_data_b
4’b0011	MAC	o_mult = i_data_a * i_data_b $o\_data_{new} = o\_mult + o\_data_{old}$
4’b0100	Tanh	o_data = tanh(i_data_a)
4’b0101	ORN	o_data = i_data_a | i_data_b’
4’b0110	CLZ	Count leading zero bits
4’b0111	CTZ	Count trailing zero bits
4’b1000	CPOP	Count set bits
4’b1001	ROL	Rotate left

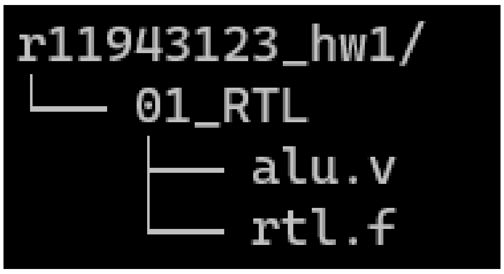
2. Considerations between digital systems and signals:
   a. For instruction 0011, you need to accumulate data with continuous MAC instructions.
   b. For instructions 0000, 0001, 0010, and 0011. If the output value exceeds the maximum value of 10b representation, use the maximum value as output, and vice versa.
   c. For instructions 0010, 0011 and 0100, the result needs to be rounded to the nearest number.
   d. For instructions 0100, you need to implement a tanh function. To have an easier implementation, we use piecewise linear approximation to compute the tanh function. You can check 111-2_HW1_note.pdf for details.
   e. Bit-wise operation for instruction 0101.
   f. For instructions 0110, 0111, and 1000, only one input i_data_a. Note you have to avoid combinational loop or the instruction will not be scored.
   g. For instruction 1001, view i_data_b as the shift amount.
   h. For fixed-point operation, please pay attention to the position of separation between integer and fraction after ALU operation.

Sample Waveform



Submission

1. Create a folder named **studentID_hw1** and follow the hierarchy below.



```
r11943123_hw1/
└── 01_RTL
    ├── alu.v
    └── rtl.f
```

Note: Use **lower case** for the letter in your student ID. (Ex. r11943123_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** in your student ID. (Ex. r11943123_hw1_v1.tar)

3. Submit to NTU Cool

Grading Policy

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

```
ncverilog -f rtl.f +define+I0 +access+rw
```

2. Pass all the instruction test to get full score.
   - Released pattern **70%**
     - I0~I4: 40%
     - I5~I9: 30%
   - Hidden pattern **30%**
     - Only if you pass all patterns will you get the full 30% score.
     - I0~I4: 10000
     - I5~I9: 10000
3. No late submission
   - 0 point for this homework
4. Lose **3 point** for any wrong naming rule.

References

1. Reference for 2'complement:
   https://en.wikipedia.org/wiki/Two%27s_complement
2. Reference for fixed-point representation
   Fixed-Point Representation: The Q Format and Addition Examples
3. Reference for rounding to the nearest:
   Rounding - MATLAB & Simulink (mathworks.com)