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

TA: 李其祐 r11943123@ntu.edu.tw Due Tuesday, Oct. 3rd, 13:59

TA: 駱奕霖 f06943176@ntu.edu.tw

Data Preparation

1. Decompress 1121_hw1.tar with following command

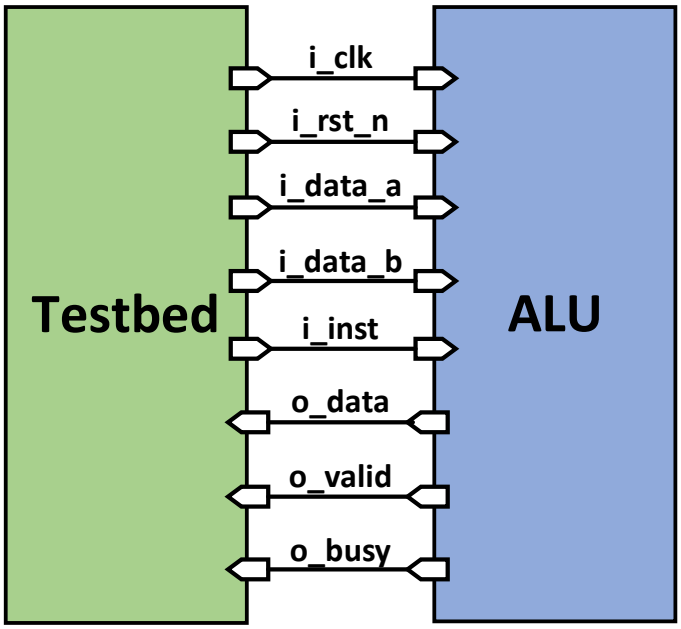
```
tar -xvf 1121_hw1.tar
```

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
02_lint	SpyGlass lint check
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.

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_inst	I	4	Instruction for ALU to operate
i_data_a	I	16	1. For instructions 0000~0100, signed input data with 2's complement representation (6-bit signed integer + 10-bit fraction) 2. For instructions 0101~0111, 16-bit number 3. For instructions 1000, 1001, Floating point (FP16)
i_data_b	I	16	
o_valid	O	1	Set **high** if ready to output result
o_data	O	16	1. For instructions 0000~0100, signed input data with 2's complement representation (6-bit signed integer + 10-bit fraction) 2. For instructions 0101~0111, 16-bit number 3. For instructions 1000, 1001, Floating point (FP16)
o_busy	O	1	Set **low** for one cycle, if ready for new input data. Set **high**, the input data remain unchanged.

3. All inputs are synchronized with the negative clock edge.
4. Active low asynchronous reset is used only once.
5. All outputs should be synchronized at clock rising edge (Flip-flops are added before outputs).
6. i_data_a, i_data_b and i_inst will be sent while o_busy is low.
7. o_valid should be pulled high for only one cycle for each o_data.
8. The testbed will get your output at negative clock edge and check the answer when your o_valid is high.
9. You can raise the o_valid at any moment.

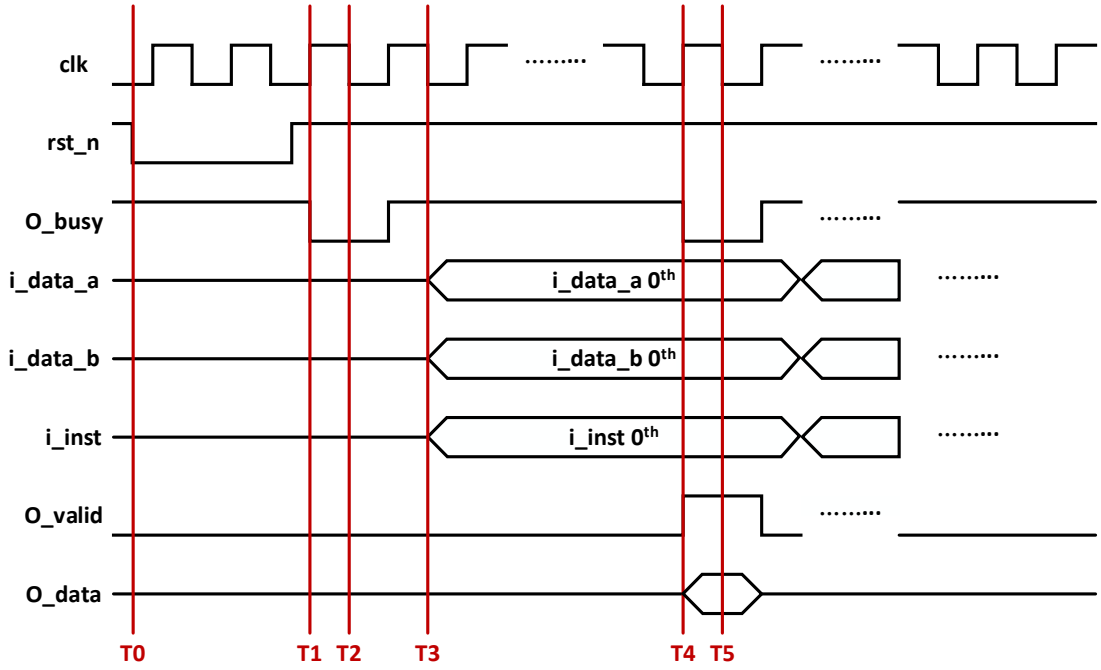
Design Description

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 (FX)	o_data = i_data_a + i_data_b
4'b0001	Signed Subtraction (FX)	o_data = i_data_a - i_data_b
4'b0010	Signed Multiplication (FX)	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	GELU	o_data = GELU(i_data_a)
4'b0101	CLZ	Count leading zero bits
4'b0110	LRCW	Encode the CPOP result
4'b0111	LFSR	Generate pseudo random number
4'b1000	Signed Addition (FP)	o_data = i_data_a + i_data_b
4'b1001	Signed Subtraction (FP)	o_data = i_data_a - i_data_b

2. Considerations between digital systems and signals:
   a. For instruction 0011, you need to accumulate data with continuous MAC instructions (Multiply-accumulate first, then rounding and saturation).
   b. For instructions I4, you need to implement a GELU function. To have an easier implementation, we approximate the GELU with tanh and use piecewise linear approximation to compute the tanh function. You can check 112-1_HW1_note.pdf for details.
   c. For instructions I0~I4. If the output value exceeds the maximum value of 16b representation, use the maximum value as output, and vice versa.
   d. For instructions I2~I4, the result needs to be rounded to the nearest number.
   e. For instructions I5, only one input i_data_a.
   f. For instruction I6, view i_data_a as the bit mask and i_data_b as the initial value
   g. For instruction I7, view i_data_a as the initial value (seed) and i_data_b as the number of the iterations.
   h. For instructions I8, I9, floating-point operations, remember to do rounding. You can check 112-1_HW1_note.pdf for details.

Sample Waveform



Submission

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

```
r11943123_hw1/
└── 01_RTL
    ├── alu.v (and other verilog files)
    ├── flist.v
    └── rtl.f
```

Note: Use **lowercase** 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:

A. Use **lowercase** in your student ID. (Ex. r11943123_hw1_v1.tar)

B. Remember to put all the files in the studentID_hw1

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.

```
vcs -f rtl.f -full64 -R -debug_access+all +v2k +define+$1
```

2. Pass all the instruction test to get full score.
   - Released pattern **70%**
     - I4, I8, I9: 8% for each instruction
     - I0~I3: 7 % for each instruction
     - I5~I7: 6% for each instruction
   - Hidden pattern **30%**
     - Only if you pass all patterns will you get the full 30% score.
     - I0~I3: 8000 (mix)
     - I4~I9: 2000 for each instruction
3. Spyglass check with error (-20).
4. Do not use DesignWare for floating point operations.
5. No late submission
   - 0 point for this homework
6. Lose **5 points** for any wrong naming rule.

References

1. Reference for fixed-point representation
   https://www.allaboutcircuits.com/technical-articles/fixed-point-representation-the-q-format-and-addition-examples/
2. Reference for rounding to the nearest
   https://www.mathworks.com/help/fixedpoint/ug/rounding.html
3. Reference for FP16 representation
   https://en.wikipedia.org/wiki/Half-precision_floating-point_format
4. CMU Lecture – Floating point
   https://www.cs.cmu.edu/afs/cs/academic/class/15213-s16/www/lectures/04-float.pdf