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

TA: 徐以帆 r13943005@ntu.edu.tw **Due Tuesday, Oct. 1st, 13:59**

TA: 林祐丞 d10943005@ntu.edu.tw

Data Preparation

1. Unpack 1131_hw1.tar with the following command

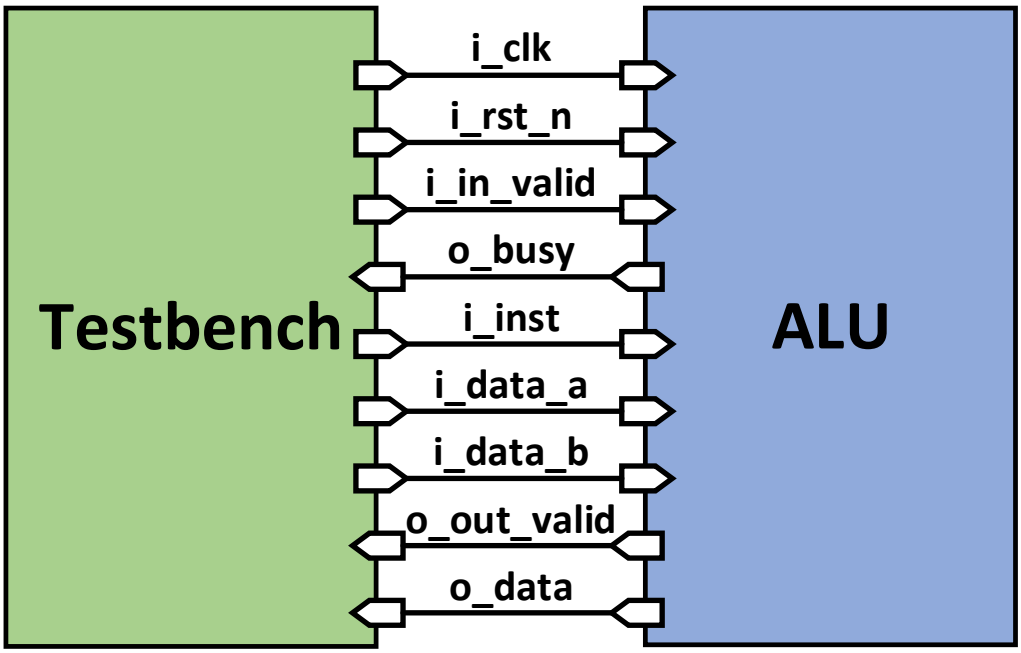
```
tar -xvf 1131_hw1.tar
```

Folder	File	Description
00_TESTBED	testbench.v	File to test your design
00_TESTBED /pattern	INST*_I.dat	Input instruction patterns
	INST*_O.dat	Output golden patterns
01_RTL	alu.v	Your design
	rtl.f	File list for RTL simulation
	01_run	VCS command for simulation
	99_clean	Command for cleaning temporal files

Introduction

An arithmetic logic unit (ALU) is one of the components of a computer processor. It performs arithmetic and bit-level logical operations in a computer. In this homework, you are going to design an ALU with some special instructions.

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_in_valid	I	1	The signal is **high** if input data is ready
o_busy	O	1	Set **low** if ready for next input data. Set **high** to pause input sequence.
i_inst	I	4	Instruction for ALU to perform
i_data_a	I	16	Signed input data with 2's complement representation 1. For instructions 0000~0100, fixed-point number (6-bit signed integer + 10-bit fraction) 2. For instructions 0101~1001, integer
i_data_b	I	16	
o_out_valid	O	1	Set **high** if ready to output result
o_data	O	16	Signed output data with 2's complement representation 1. For instructions 0000~0100, fixed-point number (6-bit signed integer + 10-bit fraction) 2. For instructions 0101~1001, integer

3. Active low asynchronous reset is used only once.
4. All inputs are synchronized with the **negative** clock edge.

5. All outputs should be synchronized with the **positive** clock edge. That is, flip-flops should be added before all outputs.
6. New pattern (i_inst, i_data_a and i_data_b) is ready only when i_in_valid is high.
7. At each negative clock edge, i_in_valid will be randomly pulled high only if o_busy is low.
8. o_out_valid should be pulled high for only one cycle for each o_data.
9. The testbench will sample o_data at **negative** clock edge if o_out_valid is high.
10. You can raise o_out_valid at any moment.

Design Description

1. The following table shows the operations you need to implement on your ALU:

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	Signed Accumulation	idx = i_data_a o_data = $data\_acc[idx]_{new}$ = $data\_acc[idx]_{old}$ + i_data_b
4'b0100	Softplus	o_data = softplus(i_data_a) (Use piecewise linear approximation)
4'b0101	XOR	o_data = i_data_a ⊕ i_data_b
4'b0110	Arithmetic Right Shift	Arithmetically shift i_data_a right by i_data_b bits
4'b0111	Left Rotation	Rotate i_data_a left by i_data_b bits
4'b1000	Count Leading Zeros	Count leading 0's in i_data_a
4'b1001	Reverse Match4	Custom bit-level operation

2. Specifications on number representation:
   a. For instructions 0000~0100, saturation must be applied to the final result. That is, if the final result exceeds the maximum (minimum) representable value of 16-bit representation (6-bit integer + 10-bit fraction), use the maximum (minimum) value as output.
   b. For instructions 0010 and 0100, rounding must be applied to the final result before saturation. The rounding mode used is **rounding to the nearest**[2]. That is, if the value of the remaining bits under LSB is greater than or equal to half the value representable by LSB, it should be rounded up.
   c. For instruction 0011, intermediate values are guaranteed not to exceed 20 bits.
3. For instruction 0011, all data_acc[x] must be reset to 0 when i_rst_n is low, and i_data_a is guaranteed to be from 0 to 15 (inclusive).

4. For instruction 0100, please use the following piecewise linear approximation to compute softplus[3], a non-linear function. It is not allowed to use the division operator (/) in Verilog code.

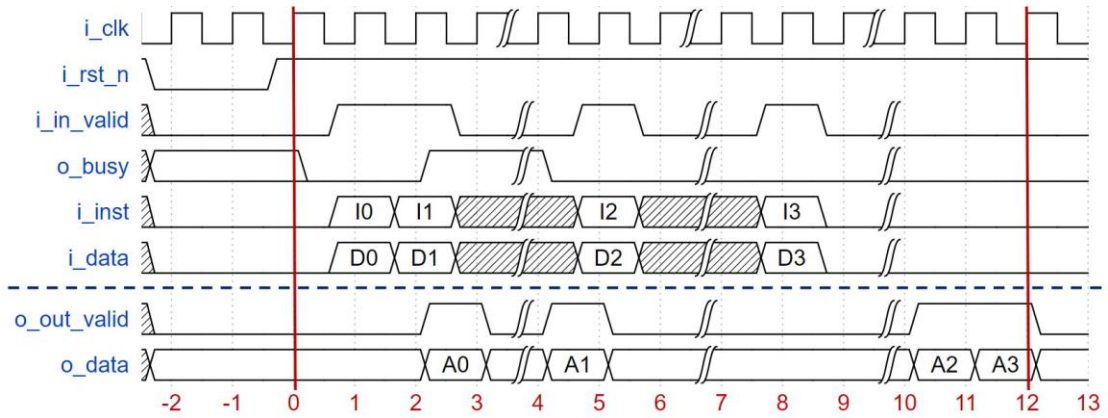
$$softplus(x) = ln(1+e^x) \approx f(x) = \begin{cases} x, & x \geq 2 \\ (2x+2)/3, & 0 \leq x \leq 2 \\ (x+2)/3, & -1 \leq x \leq 0 \\ (2x+5)/9, & -2 \leq x \leq -1 \\ (x+3)/9, & -3 \leq x \leq -2 \\ 0, & x \leq -3 \end{cases}$$

5. For instructions 0110 and 0111, i_data_b is guaranteed to be from 0 to 16 (inclusive).
6. For instructions 1000 and 1001, it is recommended to use for loops.
7. For instruction 1000, there cannot be any combinational loop. Otherwise, the instruction will not be scored.
8. For instruction 1001, implement the following custom bit-level operation.

$$\text{o\_data[i]} = \begin{cases} (\text{i\_data\_a}[i+3:i] == \text{i\_data\_b}[15-i:12-i]), & i = 0 \sim 12 \\ 0, & i = 13 \sim 15 \end{cases}$$

9. You **CANNOT** implement any operation except piecewise linear approximation by look up tables, unless there are good reasons and you have checked with TA.
10. You are **NOT** allowed to use DesignWare.

Sample Waveform



Submission

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

```
r13943000_hw1
    └── 01_RTL
        ├── alu.v
        ├── xxx.v (other verilog files you wrote)
        └── rtl.f
```

   Note: Use **lowercase** for all the letters. (e.g. r13943000_hw1)

2. Pack the folder **studentID_hw1** into a **tar file** named **studentID_hw1_v*k*.tar (*k* is the number of version, *k* =1,2,…)**. TA will only check the last version.

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

   Note:

   a. Use **lowercase** for all the letters. (e.g. r13943000_hw1_v1.tar)

   b. Pack the folder on IC Design LAB server to avoid OS related problems.

3. Submit to NTU Cool

Grading Policy

1. TA will run your code with the following format of command. Make sure to run this command with no error message on IC Design LAB server.

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

2. Pass all the instruction tests to get full score.

   - Released patterns: **75%**

i_inst[3:0]	Operation	Score
4'b0000	Signed Addition	5%
4'b0001	Signed Subtraction	5%
4'b0010	Signed Multiplication	10%
4'b0011	Signed Accumulation	10%
4'b0100	Softplus	10%
4'b0101	XOR	5%
4'b0110	Arithmetic Right Shift	5%
4'b0111	Left Rotation	5%
4'b1000	Count Leading Zeros	10%
4'b1001	Reverse Match4	10%

   - Hidden patterns: **25%**
     - Mixture of all instructions
     - Only if you pass all patterns will you get the score.

3. SpyGlass check (goal: lint_rtl and lint_rtl_enhanced) with error: -20%
4. Lose **5 points** for any incorrect naming or format.
   - It is your responsibility to ensure that the files can be correctly unpacked and executed on IC Design LAB server.
5. No late submission
   - 0 point for this homework
6. No plagiarism
   - Plagiarism in any form, including copying from online sources, is strictly prohibited.

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 softplus function
   https://neuralthreads.medium.com/softplus-function-smooth-approximation-of-the-relu-function-6a85f92a98e6
4. Reference for reciprocal multiplication
   https://homepage.divms.uiowa.edu/~jones/bcd/divide.html