

Computer-Aided VLSI System Design

Homework 1: Arithmetic Logic Unit

Graduate Institute of Electronics Engineering, National Taiwan University



NTU GIEE



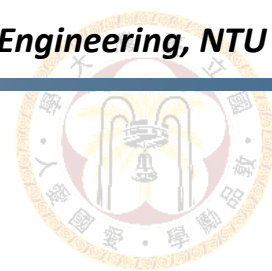
Goal

- In this homework, you will learn
 - How to design ALU with simple operations
 - Differences between combinational circuit and sequential circuit
 - How to define registers and wires
 - How to read spec

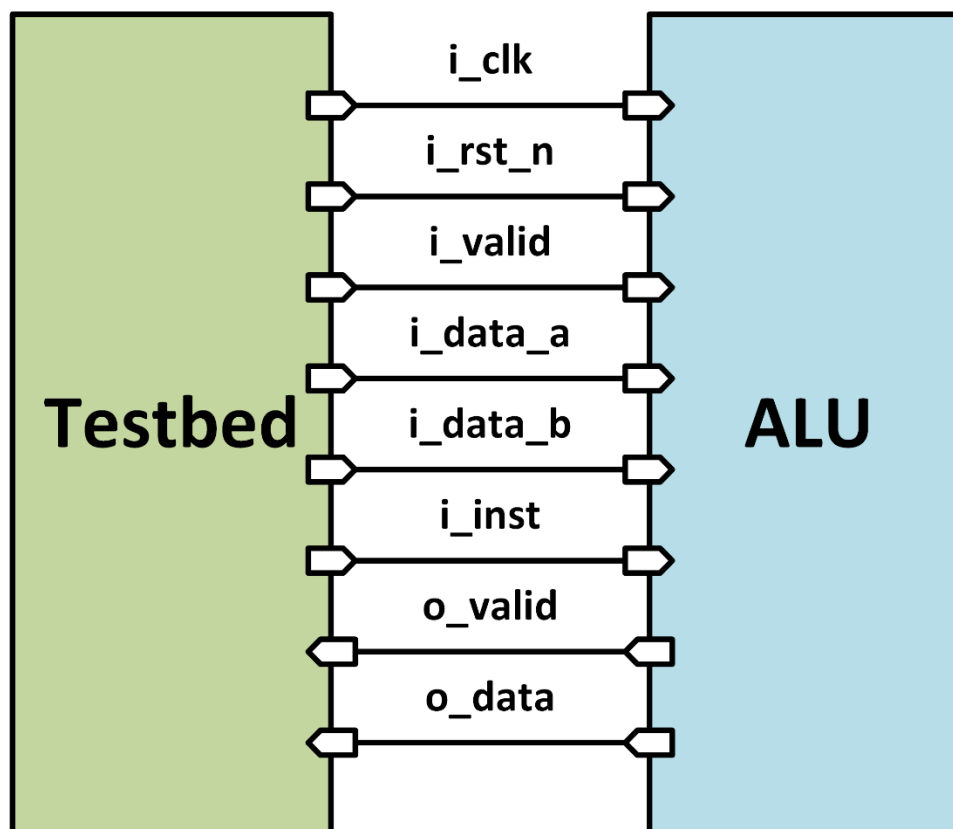


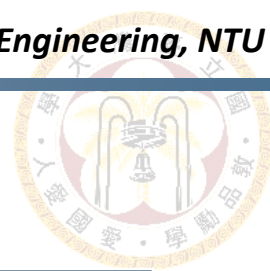
Introduction

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



Block Diagram





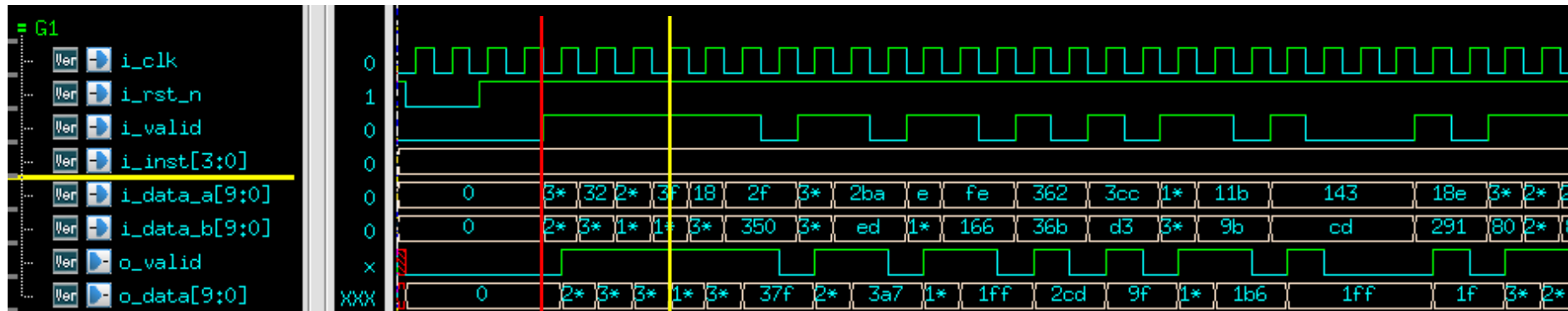
Input/Output

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)



Specification (1)

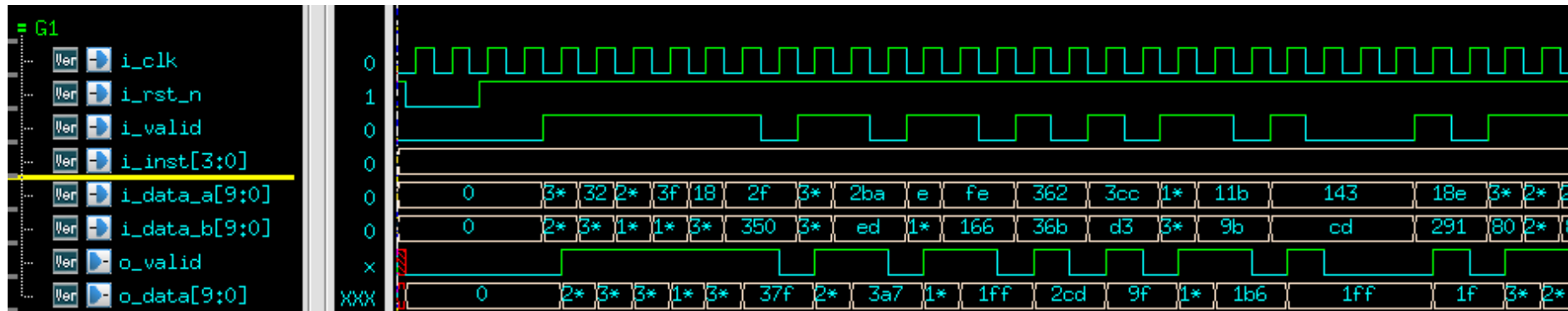
- All inputs are synchronized with the negative clock edge
- All outputs should be synchronized at clock **rising** edge (Flip-flops are added before outputs)
- Active low asynchronous reset is used and only once (You should set all your outputs to be zero when i_rst_n is **low**)





Specification (2)

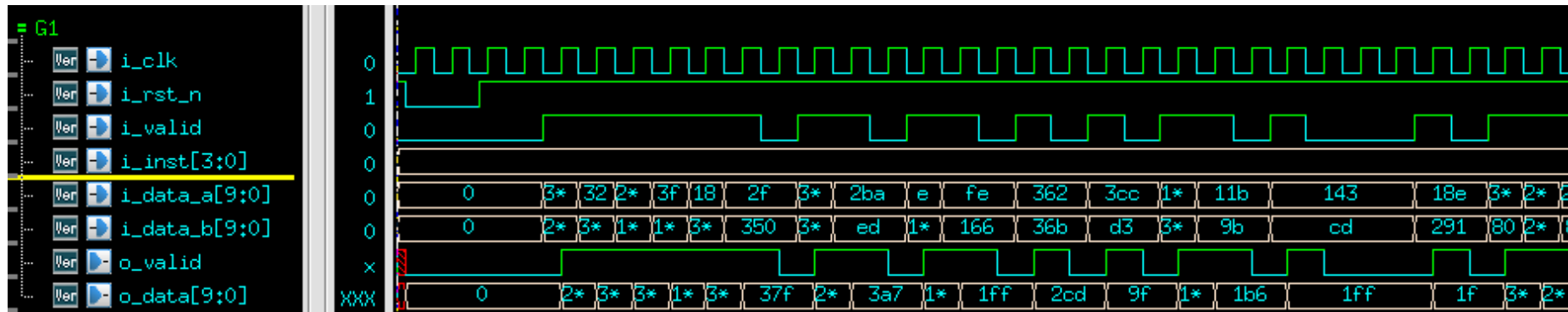
- The i_valid will turn to **high** in one cycle for ALU to get i_data_a, i_data_b and i_inst
- The i_valid will pulled **high** in random
- Your o_valid should be pulled **high** for only one cycle for every o_data

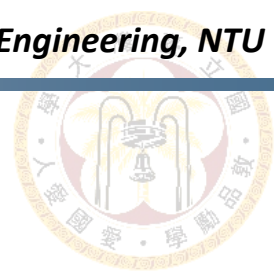




Specification (3)

- The testbed will get your output at negative clock edge to check the answer when your o_valid is **high**
- You can raise your o_valid at any moment.





Instruction

Operation	i_inst [3:0]	Meaning	Note
Signed Addition	4'b0000	$o_data = i_data_a + i_data_b$	Overflow may be happened
Signed Subtraction	4'b0001	$o_data = i_data_a - i_data_b$	
Signed Multiplication	4'b0010	$o_data = i_data_a * i_data_b$	
MAC	4'b0011	$o_mult = i_data_a * i_data_b$ $o_data_{new} = o_mult + o_data_{old}$	
Tanh	4'b0100	$o_data = \tanh(i_data_a)$	Piecewise linear approximation
ORN	4'b0101	$o_data = i_data_a i_data_b'$	Bit-wise operation
CLZ	4'b0110	Count leading zero bits	Only i_data_a is used
CTZ	4'b0111	Count trailing zero bits	
CPOP	4'b1000	Count set bits	
ROL	4'b1001	Rotate left	View i_data_b as shift amount



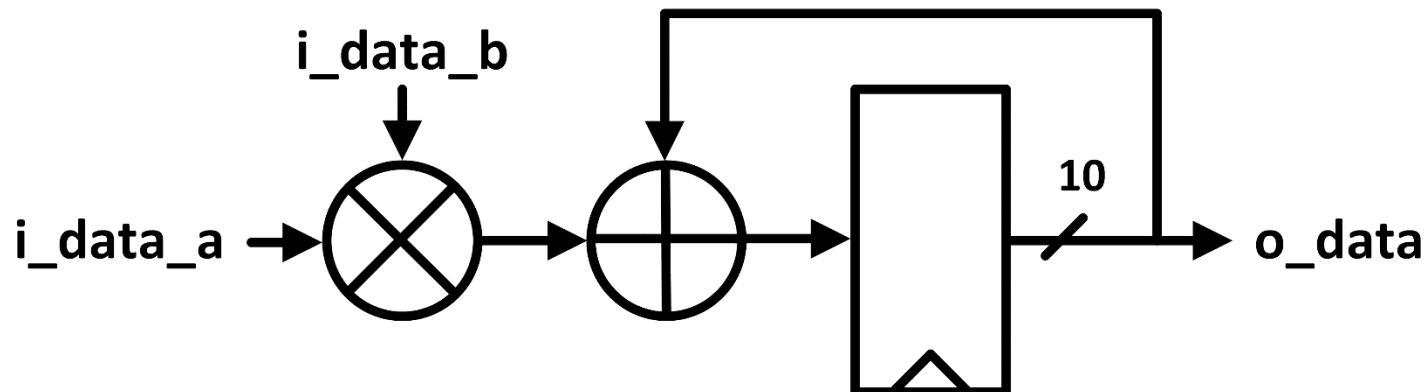
Output Saturation & Rounding

- For instructions 0000, 0001, 0010, and 0011, If the output value exceeds the maximum value of 10-bit representation, use the maximum value as output, and vice versa
- For instructions 0010, 0011 and 0100, the result needs to be **rounded to the nearest number**
 - Check reference [3]



MAC

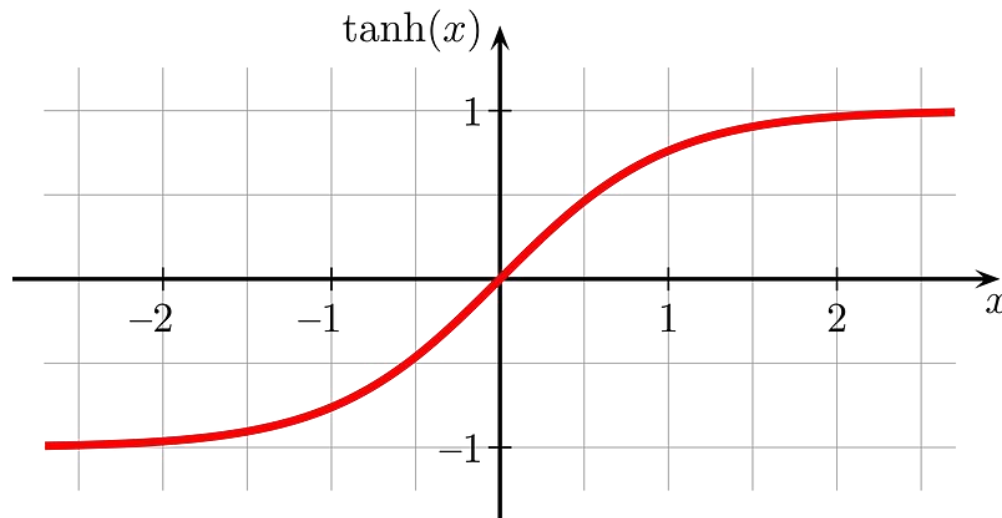
- For instruction 0011, you have to implement Multiply–accumulate operation (MAC) function
- The output of the Flip-flop must be rounded to the nearest number





Tanh Function

- For instruction 0100, you need to implement a popular activation function, a hyperbolic tangent function. It becomes preferred over the sigmoid function as it gives better performance for multi-layer neural networks
- However, it's not intuitive to implement an exponential operation on hardware

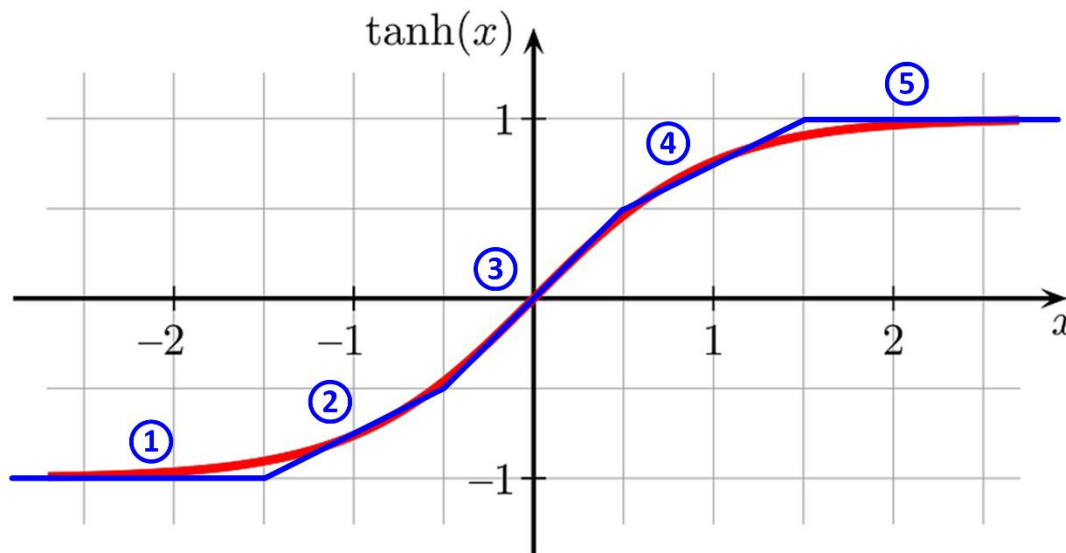


$$\tanh(x) = \frac{2}{1 + e^{-2x}} - 1$$



Tanh Function Approximation

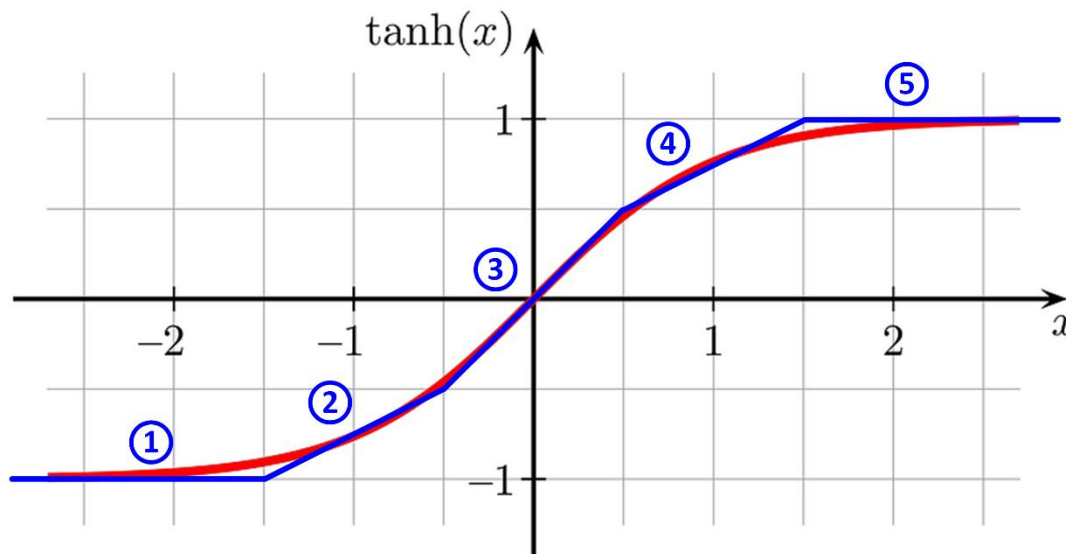
- In order to simplify the implementation, we use **piecewise linear approximation** to compute tanh function
- We divide the curve into **5 segments** to compute the output





Tanh Function Approximation

- Choose the slope of the 2nd, 3rd, 4th segment appropriately
- The 5 segments have 4 intersections
 - $(-1.5, -1)$, $(-0.5, -0.5)$, $(0.5, 0.5)$, $(1.5, 1)$
- Output must be **rounded to the nearest number**





CLZ

- For instruction 0110, count leading zero
 - Count the number of zero bits from MSB end
- For example, if $a = 8'b0010_0000$, then $CLZ(a)=2$

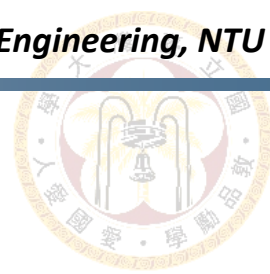
0010_0000
MSB →



CTZ

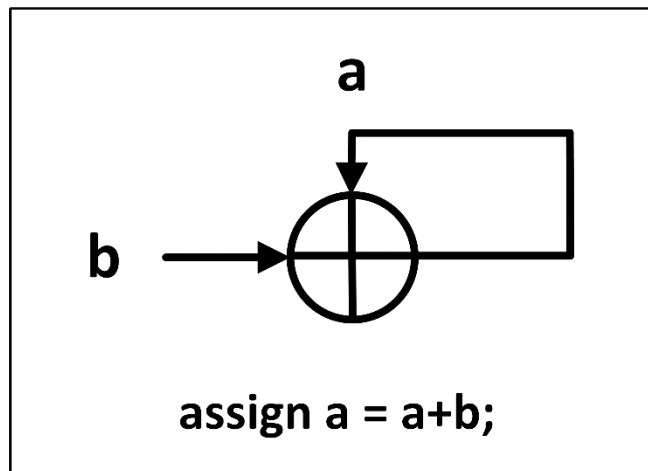
- For instruction 0111, count trailing zero
 - Count the number of zero bits from LSB end
- For example, if $a = 8'b0010_0000$, then $CTZ(a)=5$

0010_0000
 _ _ _ _ _ _ _
 ← LSB



CPOP

- For instruction 1000, count the number of set bits
- For example, if $a = 8'b0010_0001$, then $CPOP(a)=2$
- Note you have to **avoid** the **combinational loop**, where static timing analysis (STA) cannot be applied

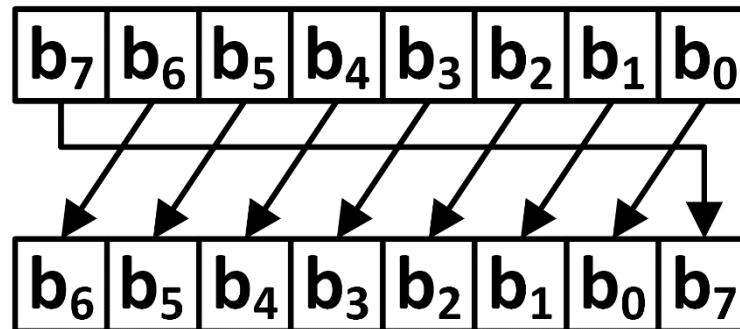


An error example

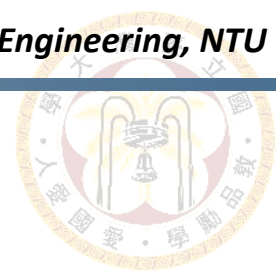


ROL

- For instruction 1001, you need to implement **rotate left** and view `i_data_b` as the **rotation amount**
 - The range of `i_data_b`: 0~9
- In rotation, the shifted bits will be filled into the vacant position



Left rotate 1 bit



alu.v

```
module alu #(
    parameter INT_W  = 4,
    parameter FRAC_W = 6,
    parameter INST_W = 4,
    parameter DATA_W = INT_W + FRAC_W
)(
    input          i_clk,
    input          i_rst_n,
    input          i_valid,
    input signed [DATA_W-1:0] i_data_a,
    input signed [DATA_W-1:0] i_data_b,
    input          [INST_W-1:0] i_inst,
    output         o_valid,
    output [DATA_W-1:0] o_data
); // Do not modify

// -----
// Wires and Registers
// -----
reg [DATA_W-1:0] o_data_w, o_data_r;
reg             o_valid_w, o_valid_r;
// ---- Add your own wires and registers here if needed ---- //
```



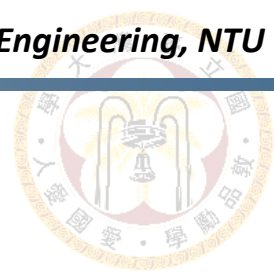
alu.v

```
// -----  
// Continuous Assignment  
// -----  
assign o_valid = o_valid_r;  
assign o_data = o_data_r;  
// ---- Add your own wire data assignments here if needed ---- //  
  
// -----  
// Combinational Blocks  
// -----  
// ---- Write your combinational block design here ---- //  
always@(*) begin  
    o_data_w = ;  
    o_valid_w = ;  
end
```



alu.v

```
// -----  
// Sequential Block  
// -----  
// ---- Write your sequential block design here ---- //  
always@(posedge i_clk or negedge i_rst_n) begin  
    if (!i_rst_n) begin  
        o_data_r <= 0;  
        o_valid_r <= 0;  
    end else begin  
        o_data_r <= o_data_w;  
        o_valid_r <= o_valid_w;  
    end  
end  
  
endmodule
```



testbench.v

```

`timescale 1ns/1ps
`define CYCLE      10.0
`define RST_DELAY  2
`define MAX_CYCLE  100000

`ifdef I0
  `define Inst_I   "../00_TESTBED/pattern/INST0_I.dat"
  `define Inst_O   "../00_TESTBED/pattern/INST0_O.dat"
  `define PAT_NUM  40
`elsif I1
  `define Inst_I   "../00_TESTBED/pattern/INST1_I.dat"
  `define Inst_O   "../00_TESTBED/pattern/INST1_O.dat"
  `define PAT_NUM  40
`elsif I2
  `define Inst_I   "../00_TESTBED/pattern/INST2_I.dat"
  `define Inst_O   "../00_TESTBED/pattern/INST2_O.dat"
  `define PAT_NUM  40
`elsif I3
  `define Inst_I   "../00_TESTBED/pattern/INST3_I.dat"
  `define Inst_O   "../00_TESTBED/pattern/INST3_O.dat"
  `define PAT_NUM  40
`elsif I4
  `define Inst_I   "../00_TESTBED/pattern/INST4_I.dat"
  `define Inst_O   "../00_TESTBED/pattern/INST4_O.dat"
  `define PAT_NUM  40

```

```

`elsif I5
  `define Inst_I   "../00_TESTBED/pattern/INST5_I.dat"
  `define Inst_O   "../00_TESTBED/pattern/INST5_O.dat"
  `define PAT_NUM  40
`elsif I6
  `define Inst_I   "../00_TESTBED/pattern/INST6_I.dat"
  `define Inst_O   "../00_TESTBED/pattern/INST6_O.dat"
  `define PAT_NUM  40
`elsif I7
  `define Inst_I   "../00_TESTBED/pattern/INST7_I.dat"
  `define Inst_O   "../00_TESTBED/pattern/INST7_O.dat"
  `define PAT_NUM  40
`elsif I8
  `define Inst_I   "../00_TESTBED/pattern/INST8_I.dat"
  `define Inst_O   "../00_TESTBED/pattern/INST8_O.dat"
  `define PAT_NUM  40
`elsif I9
  `define Inst_I   "../00_TESTBED/pattern/INST9_I.dat"
  `define Inst_O   "../00_TESTBED/pattern/INST9_O.dat"
  `define PAT_NUM  40
`endif

```



Commands

- ./01_run arg1

```
ncverilog -f rtl.f +define+<arg1> +access+rw
```

– For example:

```
./01_run I0
```

- ./99_clean

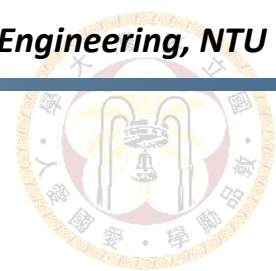
```
rm -rf *.history *.key *.log
```

```
rm -rf novas.rc novas.fsdb novas.conf
```

```
rm -rf INCA_libs nWaveLog BSSLib.lib++
```

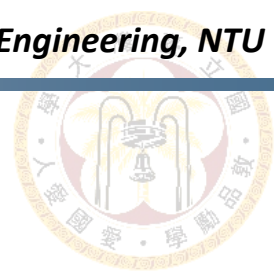
- Note before you execute the shell script, change the permission of the file by

```
chmod +x 01_run
```



Pattern (Input Data)

i_inst	i_data_a	i_data_b
0000	110100111001	000100010101
0000	1010010000	1110011101
0000	0000110011	0101010100
0000	1011111001	101111000
0000	1000101011	0011100010
0000	0011010101	1101111101
0000	1011101001	0100001000



Pattern (Golden Output)

o_data

```
0001100011
1000101101
0110000111
1001110100
1100001101
0001010010
1111110001
0001111110
```



Grading Policy (1)

- Released pattern **70%**
 - I0~I4: 40%
 - I5~I9: 30%
- Hidden pattern: **30%**
 - Hidden pattern contains all instructions
 - I0~I4: 10000
 - I5~I9: 10000
 - Only if you pass all patterns will you get the full 30% score



Grading Policy (2)

- No late submission
 - **0 point** for this homework
- Lose **3 points** for any wrong naming rule or format for submission
- No plagiarize

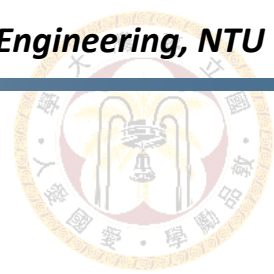


Submission

- Create a folder named **studentID_hw1** and follow the hierarchy below

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

- Compress the folder **studentID_hw1** in a tar file named **studentID_hw1_vk.tar** (k is the number of version, $k = 1, 2, \dots$)
 - Note: Use lower case in your student ID.
(Ex. r11943123_hw1_v1.tar)
- Submit to NTU Cool



Discussion

三 電腦輔助積體電路系統設計 (EEE5022) > 討論 > [HW1]Discussion

111-2

首頁

課程資訊

課程內容

公告

作業

成績

討論

文件

頁面

成員

線上測驗

設定

[HW1]Discussion

所有班別

HW1相關問題在此討論，並請以下列格式發問，方便助教按照每個問題回答

1. 問題一

2. 問題二

...

另外，若需要截圖，請勿把自己的code截圖或code文字上傳，變成大家的參考答案，若違反將扣本次作業總分10分。

[提醒]

1. ...

2. ...

3. ...

祝同學們學習順心

TA



References

- [1] Reference for 2's 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 \(mathworks.com\)](#)