cad 29 不能用

W16 4b：SpyGlass 可免 兩年

Computer-Aided VLSI System Design

Homework 1: Arithmetic Logic Unit

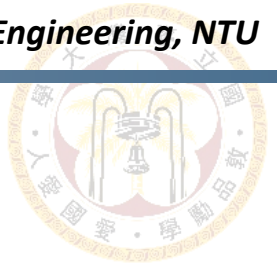
Graduate Institute of Electronics Engineering, National Taiwan University

NTU GIEE

Goal

- In this homework, you will learn
  - How to read spec
  - How to design ALU with simple operations
  - How to implement various operations by Verilog
  - How to separate combinational circuit and sequential circuit
  - How to write testbench (Optional)

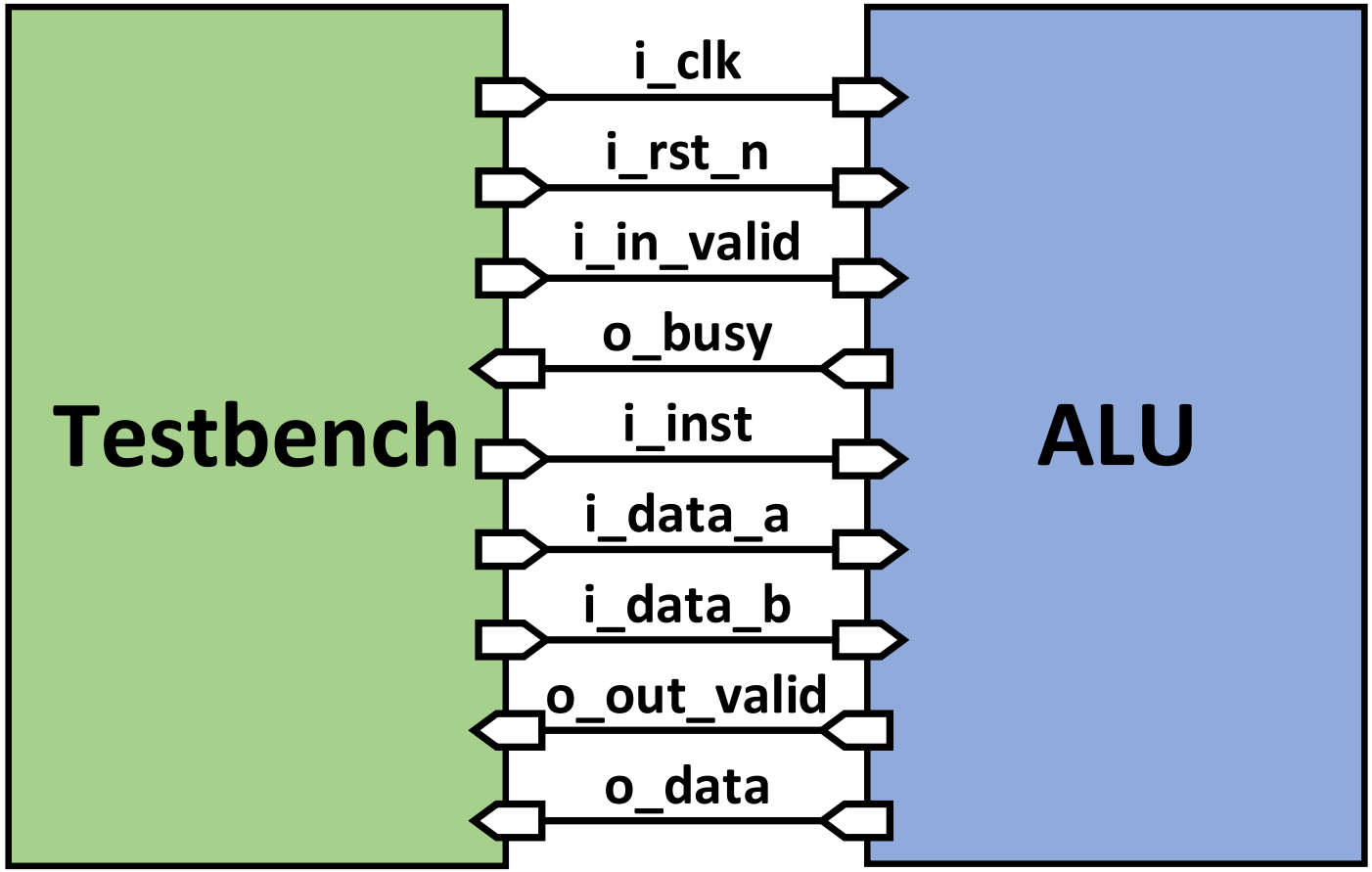
HW1有给tb



Introduction

- An arithmetic logic unit (ALU) is one of the components of a computer processor
- ALU 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

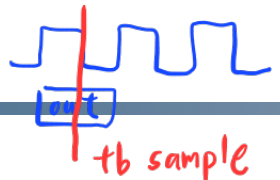


Input/Output

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_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

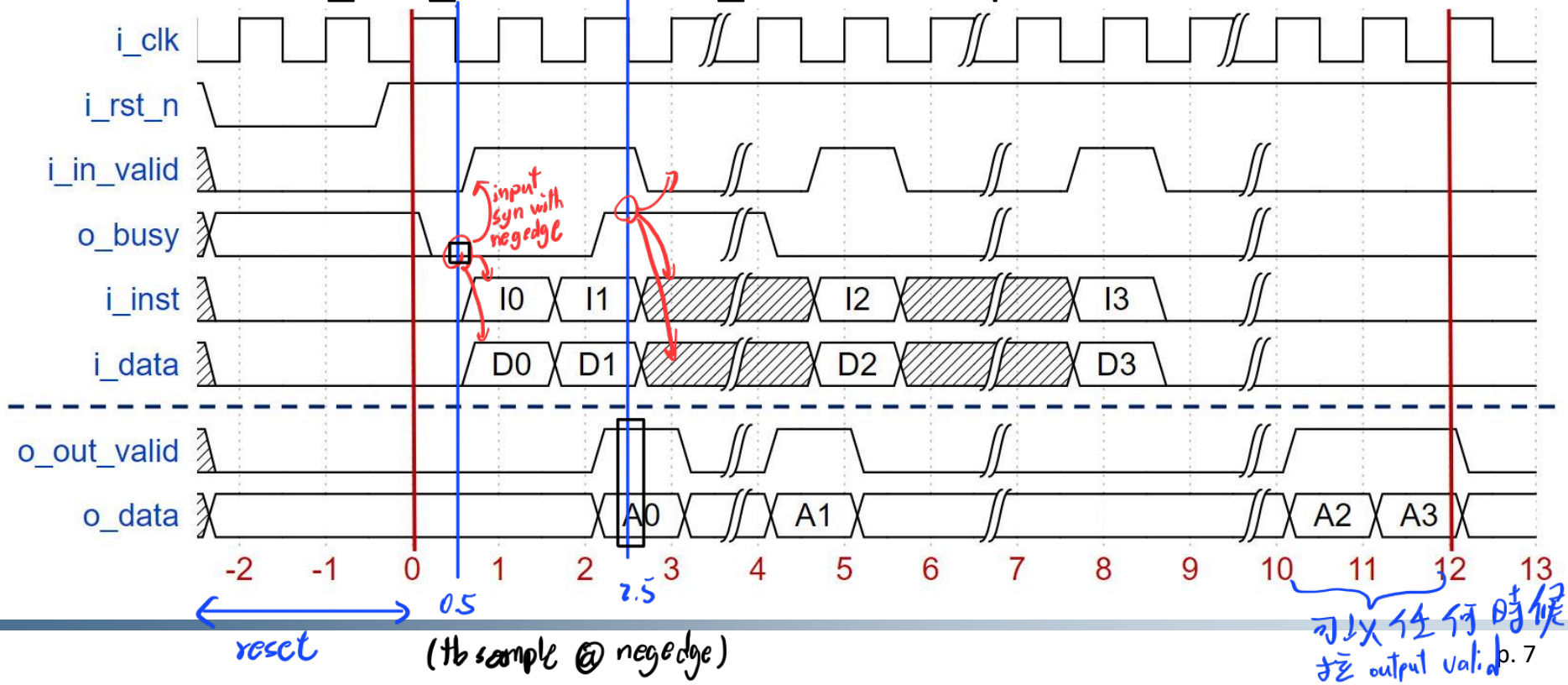
Specification (1/2)

- Active low asynchronous reset is used only once.
- All inputs are synchronized with the **negative** clock edge.
- All outputs should be synchronized with the **positive** clock edge.
  - Flip-flops should be added before all outputs.
- New pattern (i_inst, i_data_a and i_data_b) is ready only when i_in_valid is high.
- i_in_valid will be randomly pulled high only if o_busy is low.
- o_out_valid should be high for only one cycle for each o_data.
- The testbench will sample o_data at **negative** clock edge if o_out_valid is high.
- You can raise o_out_valid at any moment.



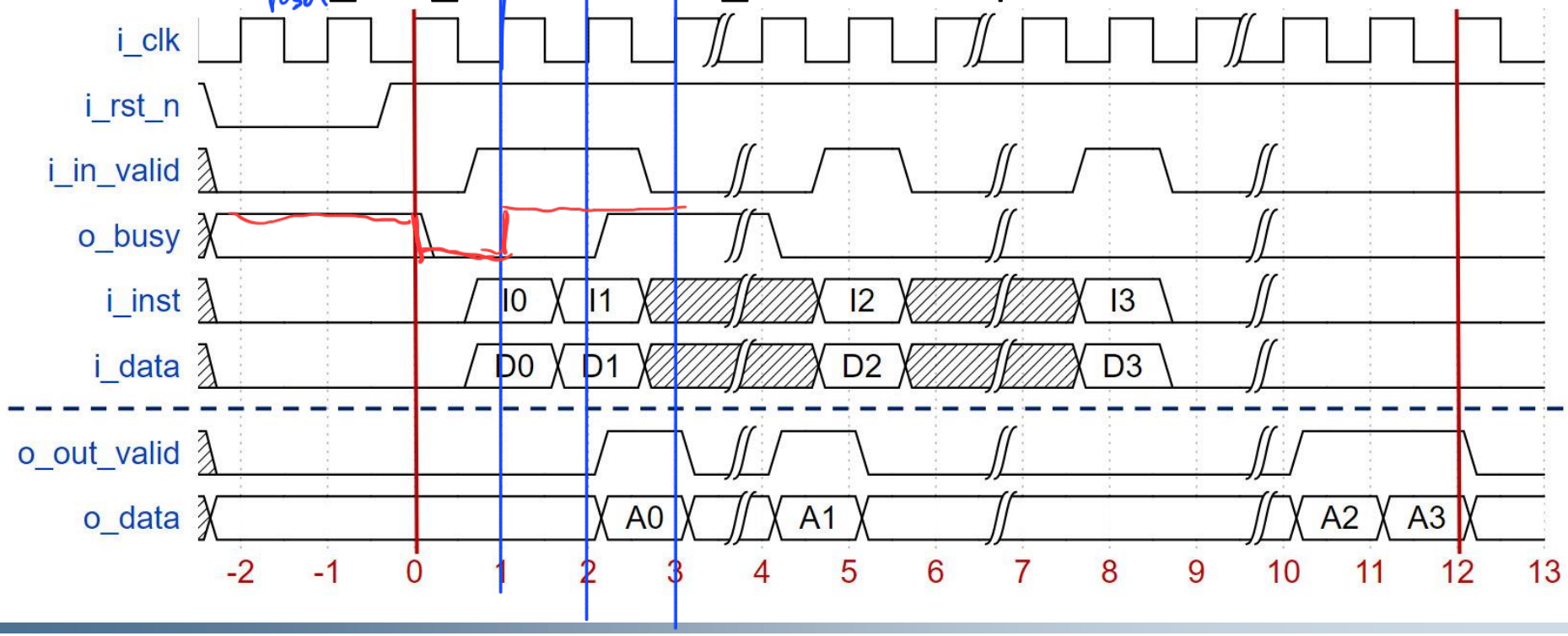
Specification (2/2)

- $t < 0$: ALU reset
- $t = 0.5$: o_busy=0 → new pattern is presented, i_in_valid=1
- $t = 2.5$: o_busy=1 → no pattern is presented, i_in_valid=0
- $t = 2.5$: o_out_valid=1 → o_data is sampled



Specification (2/2)

- $t < 0$: ALU reset
- $t = 0.5$: o_busy=0 → new pattern is presented, i_in_valid=1
- $t = 2.5$: o_busy=1 → no pattern is presented, i_in_valid=0
- $t = 2.5$: o_out_valid=1 → o_data is sampled



Instructions

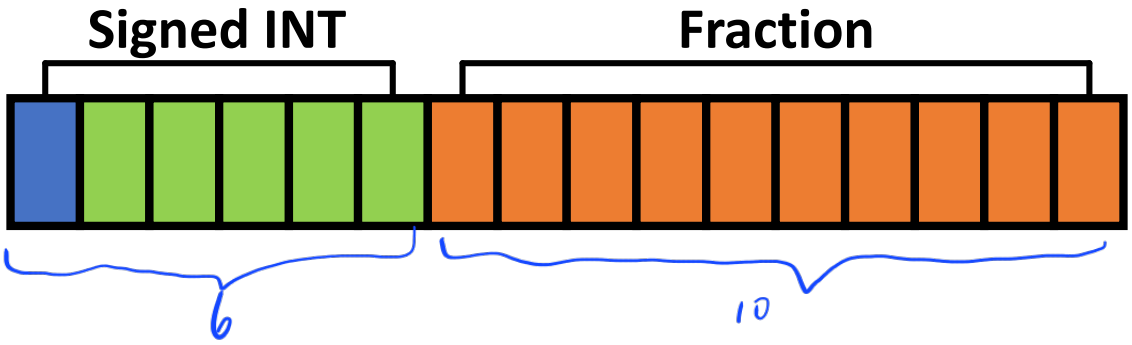
i_inst[3:0]	Operation	Description
4'b0000	Signed Addition	o_data = i_data_a + i_data_b
4'b0001	Signed Subtraction	o_data = i_data_a - i_data_b
4'b0010	Signed Multiplication	o_data = i_data_a * i_data_b
4'b0011	Signed Accumulation	idx = i_data_a o_data = $data\_acc[idx]_{old}$ + i_data_b $data\_acc[idx]_{new}$ = o_data
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

Fixed-Point Number (1/2)

- Representation used for instruction 0000~0100
  - 6-bit signed integer + 10-bit fraction = 16-bit fixed-point

fixed-point 小數

- Saturation (for instruction 0000~0100)
  - If the final result exceeds the maximum (minimum) representable value of 16-bit representation, use the maximum (minimum) value as output.



6.10 × 6.10 → 12.20 → 12.10

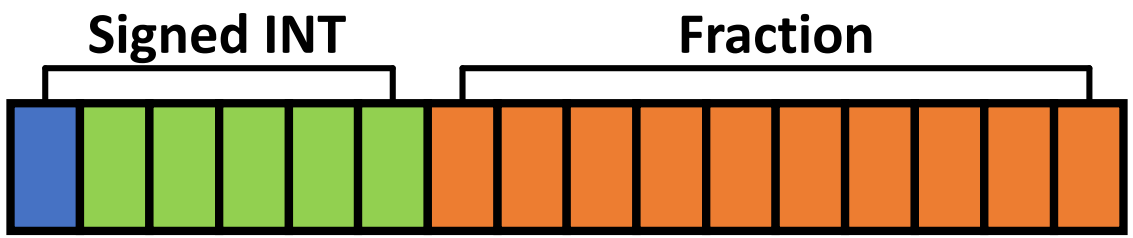
-8~7 × -8~7

4 × 4 → 64 → -56 → 8

-128~127

Fixed-Point Number (2/2)

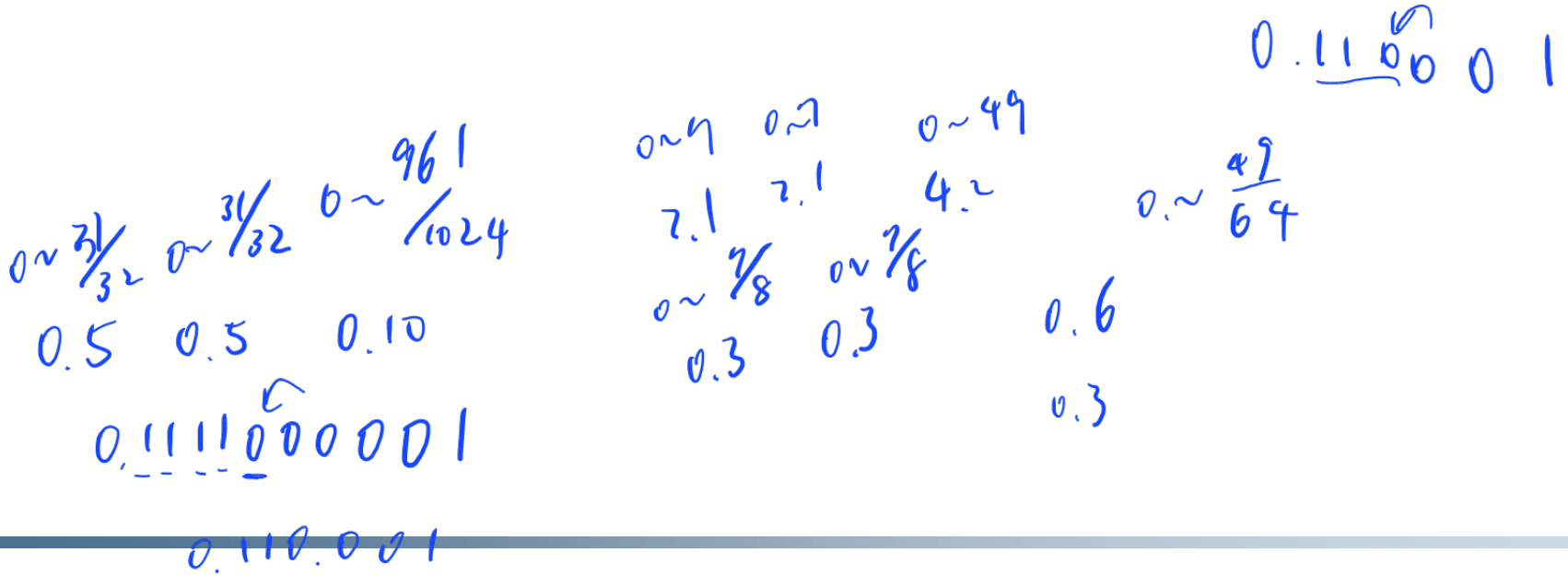
- Rounding (for instruction 0010 and 0100)
  - The result must be **rounded to the nearest**[2] representable number with 10-bit fraction first
  - For tie-breaking, round half toward positive infinity
  - Then, apply saturation to ensure the final output is a valid 16-bit fixed-point number



Signed Add/Sub/Mul

- Topic: basic operator, sizing and signing
- i_data_a is the first operand 左
- i_data_b is the second operand 右2
- o_data is the final result
- Rounding and saturation must be applied to the output

先 再



Signed Accumulation

- Topic: vector array
- Implement an accumulator with 16 independent memory units (initialized to 0 during reset)
- i_data_a is the index of the chosen memory unit, and is guaranteed to be from 0 to 15 (inclusive)
- i_data_b is the value to be accumulated
- o_data is the current accumulated result
- Intermediate values are guaranteed not to exceed 20 bits
- Saturation must be applied to the output

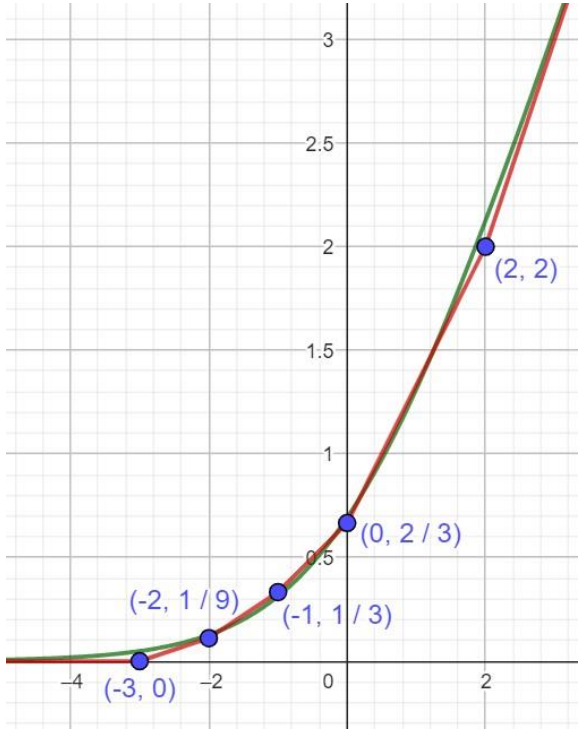
$$\begin{aligned} \text{index} &= \text{i\_data\_a} \\ \text{data\_acc}[\text{index}]_{new} &= \text{data\_acc}[\text{index}]_{old} + \text{i\_data\_b} \\ \text{o\_data} &= \text{SAT}(\text{data\_acc}[\text{index}]_{new}) \end{aligned}$$

Softplus Function (1/2)

- Topic: piecewise linear approximation, constant division
- Implement an activation function, softplus[3], which is a smooth approximation of ReLU
- Use the following **piecewise linear approximation** to compute $softplus$(i_data_a):

$$softplus(x) = ln(1 + e^x)$$

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



Softplus Function (2/2)

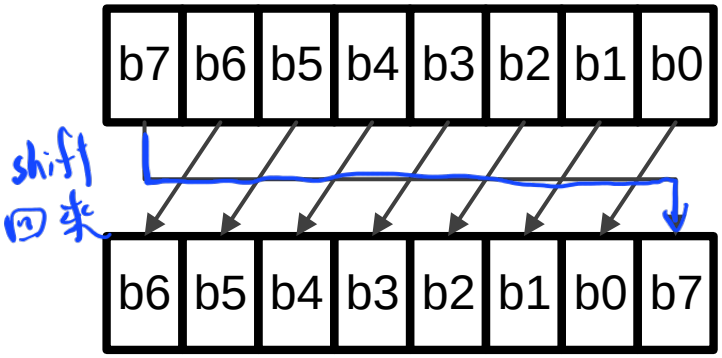
- Topic: piecewise linear approximation, constant division
- You are not allowed to use the division operator (/) in Verilog code.
- Fixed-point constant division algorithms
  - Long division → Too slow
  - DesignWare → Banned in this homework
  - **By multiplication?**
  - …
- The output (after rounding and saturation) must be **exactly the same as the golden** for any possible input value

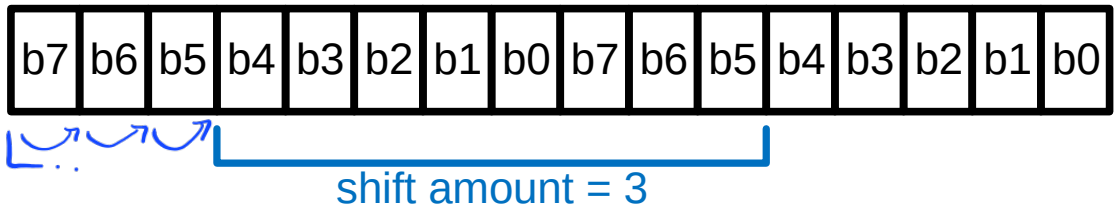
XOR, Arithmetic Shift Right

- Topic: bit-level operation, shifting
- XOR: o_data = i_data_a ⊕ i_data_b bit-wise XOR
- ASR: Arithmetically shift i_data_a right by i_data_b bits
  - i_data_b is guaranteed to be from 0 to 16 (inclusive)

Left Rotation

- Topic: vector concatenation, vector part select (optional)
- Left rotation, also called left circular shift, inserts the bit that got shifted out at one end back to the other end
- i_data_a is the original pattern
- i_data_b is the shift amount, and is guaranteed to be from 0 to 16 (inclusive)



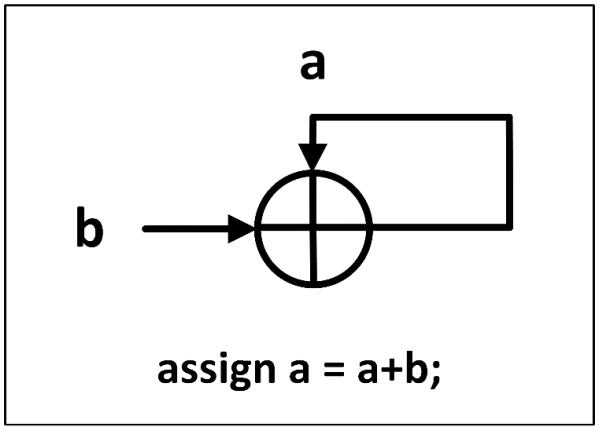


Count Leading Zeros

- Topic: for loop, combinational loop, generate block (optional)
- Count the number of consecutive 0's from MSB 有多少連續的0
- For example, if a = 8'b0010_0000, then CLZ(a)=2
- It is recommended to use **for loops** instead of hand crafting everything
- Be aware of combinational loop, where static timing analysis (STA) cannot be applied

0010_0000

MSB →



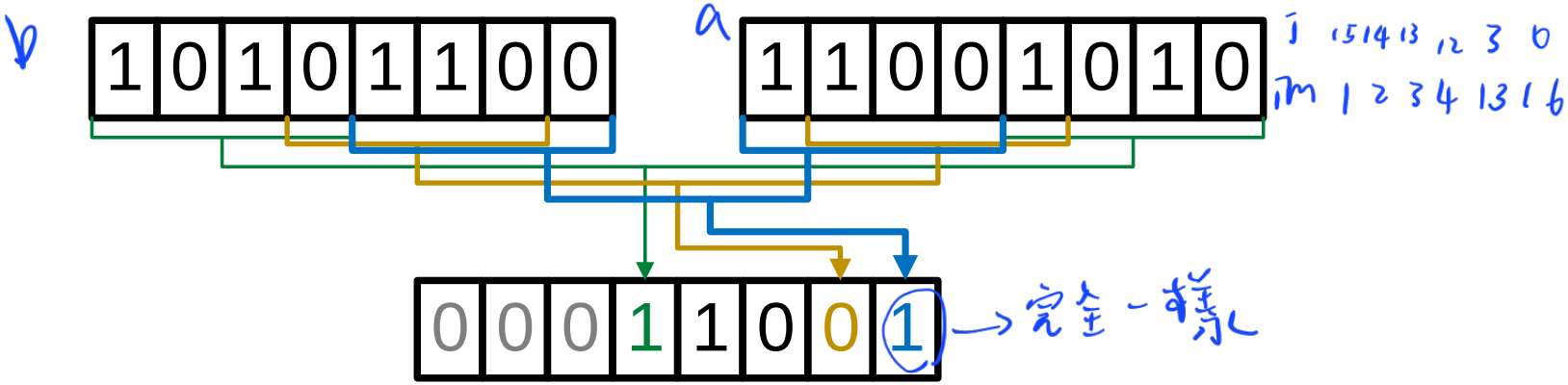
An error example

Reverse Match4

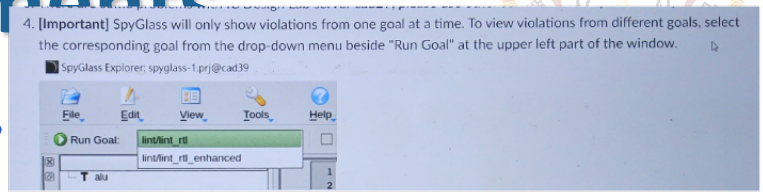
- Topic: for loop, vector part select, generate block (optional)
- This is a custom bit-level operation that matches 4 bits of i_data_a and i_data_b at a time in reverse order

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

- For example:



Other Requirements



- Check your code with SpyGlass
  - Goal setup: lint_rtl and lint_rtl_enhanced
  - List of waivable errors will be updated on NTU Cool
  - If you encounter any error that seems waivable, check with TA on NTU Cool

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

不要改

alu.v

```
module alu #(
    parameter INST_W = 4,
    parameter INT_W  = 6,
    parameter FRAC_W = 10,
    parameter DATA_W = INT_W + FRAC_W
)(
    input                       i_clk,
    input                       i_rst_n,

    input                       i_in_valid,
    output                      o_busy,
    input          [INST_W-1:0] i_inst,
    input  signed  [DATA_W-1:0] i_data_a,
    input  signed  [DATA_W-1:0] i_data_b,

    output                      o_out_valid,
    output         [DATA_W-1:0] o_data
);
```

```
);

    // Local Parameters


    // Wires and Regs


    // Continuous Assignments


    // Combinatorial Blocks


    // Sequential Blocks



endmodule
```

testbench.v

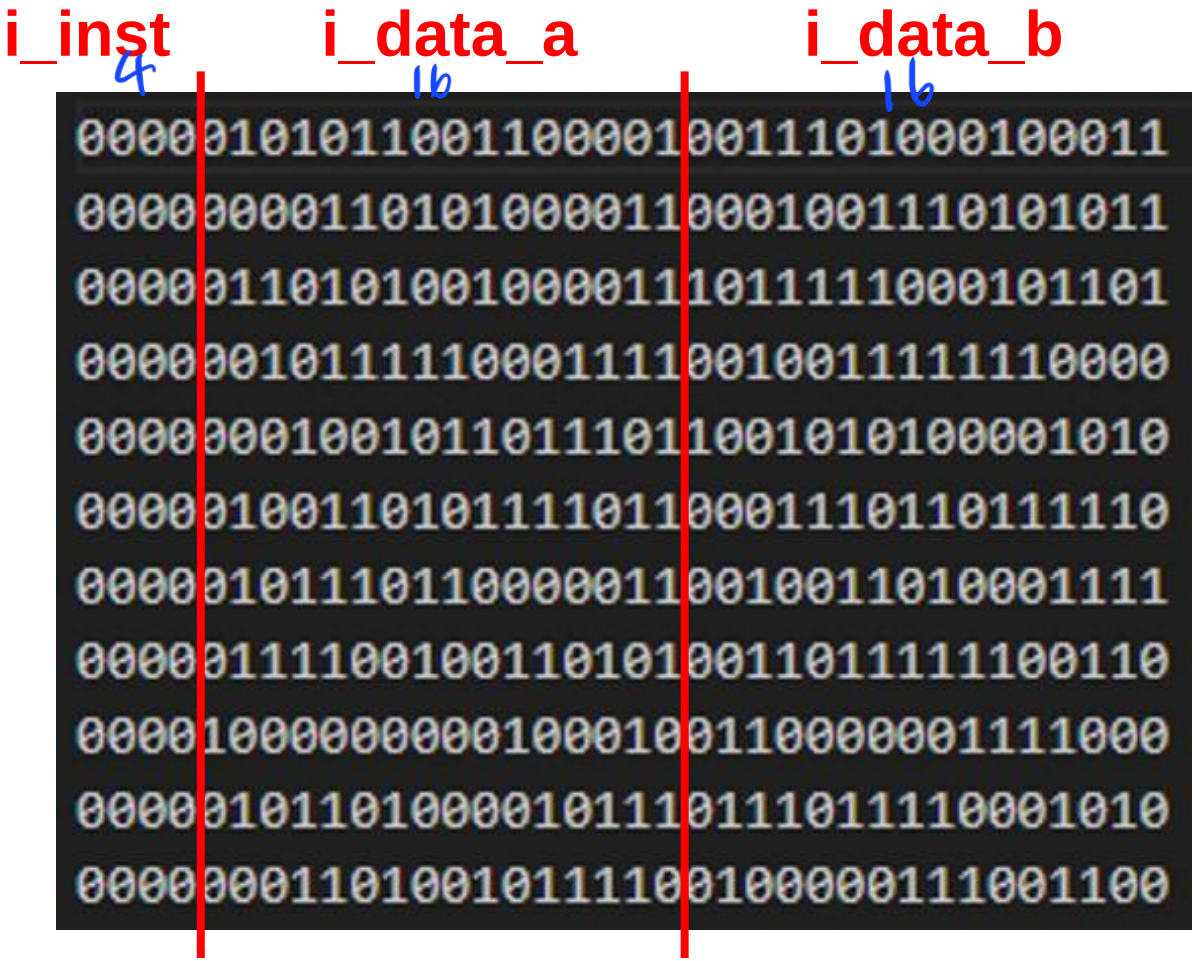
```
`timescale 1ns/10ps
`define PERIOD    10.0
`define MAX_CYCLE 100000
`define RST_DELAY 2.0

`define SEQ_LEN 60
`ifdef I0
    `define IDATA  "../00_TESTBED/pattern/INST0_I.dat"
    `define ODATA  "../00_TESTBED/pattern/INST0_O.dat"
    `define PAT_LEN 40
`elsif I1
    `define IDATA  "../00_TESTBED/pattern/INST1_I.dat"
    `define ODATA  "../00_TESTBED/pattern/INST1_O.dat"
    `define PAT_LEN 40
`elsif I2
    `define IDATA  "../00_TESTBED/pattern/INST2_I.dat"
    `define ODATA  "../00_TESTBED/pattern/INST2_O.dat"
    `define PAT_LEN 40
`elsif I3
    `define IDATA  "../00_TESTBED/pattern/INST3_I.dat"
    `define ODATA  "../00_TESTBED/pattern/INST3_O.dat"
    `define PAT_LEN 40
`elsif I4
    `define IDATA  "../00_TESTBED/pattern/INST4_I.dat"
    `define ODATA  "../00_TESTBED/pattern/INST4_O.dat"
    `define PAT_LEN 40
`elsif I5
    `define IDATA  "../00_TESTBED/pattern/INST5_I.dat"
    `define ODATA  "../00_TESTBED/pattern/INST5_O.dat"
    `define PAT_LEN 40
`elsif I6
    `define IDATA  "../00_TESTBED/pattern/INST6_I.dat"
    `define ODATA  "../00_TESTBED/pattern/INST6_O.dat"
    `define PAT_LEN 40
`elsif I7
    `define IDATA  "../00_TESTBED/pattern/INST7_I.dat"
    `define ODATA  "../00_TESTBED/pattern/INST7_O.dat"
    `define PAT_LEN 40
`elsif I8
    `define IDATA  "../00_TESTBED/pattern/INST8_I.dat"
    `define ODATA  "../00_TESTBED/pattern/INST8_O.dat"
    `define PAT_LEN 40
`elsif I9
    `define IDATA  "../00_TESTBED/pattern/INST9_I.dat"
    `define ODATA  "../00_TESTBED/pattern/INST9_O.dat"
    `define PAT_LEN 40
`else
    `define IDATA  "../00_TESTBED/pattern/INST0_I.dat"
    `define ODATA  "../00_TESTBED/pattern/INST0_O.dat"
    `define PAT_LEN 40
`endif
```

Commands

- `./01_run <arg1>`
  - `vcs -full64 -R -f rtl.f +v2k -sverilog -debug_access+all +define+$1`
  - For example: `./01_run I0` (arg1 = I0~I9)
- `./99_clean`
  - Remove all temporary files
- Before you execute the shell script, change the permission of the file by `chmod +x <script filename>`

Pattern (Input Data)



0111
1000
$-8 \sim 7$

Pattern (Golden Output)

o_data 16

```
0111111111111111
0010000011101110
0010100001110000
0101011101111111
1010011111100111
0110101100111001
0111111111111111
0111111111111111
1110000010011010
0111111111111111
0101110000101010
```

Submission

- Create a folder named **studentID_hw1** and follow the hierarchy below (*.sv is allowed if you use SystemVerilog)

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

- Pack the folder **studentID_hw1** into 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`
  - Use **lowercase** for all the letters. (e.g. r13943000_hw1_v1.tar)
  - Pack the folder on IC Design LAB server to avoid OS related problems
- Submit to NTU Cool

Grading Policy (1/3)

- Grading command
  - `vcs -full64 -R -f rtl.f +v2k -sverilog -debug_access+all +define+$1`
- Released patterns: **75%** public

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%

→ 全對打10

Grading Policy (2/3)

- Grading command
  - `vcs -full64 -R -f rtl.f +v2k -sverilog -debug_access+all +define+$1`
- Hidden patterns: **25%**
  - Mixture of all instructions
  - Only if you pass all patterns will you get the score
- SpyGlass check with error: **-20%**
  - Check **Discussion** on NTU Cool for waivable errors
- All your code has to be synthesizable or you will get **0 point**
- Lose **5 points** for any incorrect naming or format
  - Make sure all your files can be correctly unpacked and executed on IC Design LAB server

Grading Policy (3/3)

- No late submission
  - **0 point** for this homework
- No plagiarism
  - Plagiarism in any form, including copying from online sources, is strictly prohibited

Discussion

- **NTU Cool Discussion Forum**
  - For any questions not related to assignment answers or privacy concerns, please use the NTU Cool discussion forum.
  - **TAs will prioritize answering questions on the NTU Cool discussion forum**

- **Email:** r13943005@ntu.edu.tw
  - Title should start with **[CVSD 2024 Fall HW1]**
  - Email with wrong title will be moved to trash automatically

Discussion


電腦輔助積體電路系統設計 (EEE5022) > 討論 > [HW1]Discussion
課程內容
課程資訊
公告
作業
討論
Gradescope
成績
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。
祝同學們學習順心
by TA
[提醒]
1. ...
2. ...
3. ...

References

- [1] Reference for fixed-point representation
  - Fixed-Point Representation
- [2] Reference for rounding to the nearest
  - Rounding - MATLAB & Simulink
- [3] Reference for softplus function
  - Softplus Function
- [4] Reference for reciprocal multiplication
  - Reciprocal Multiplication

5
↔ 8.10