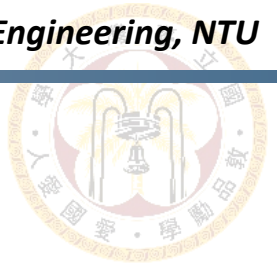
Computer-Aided VLSI System Design

Homework 1: Arithmetic Logic Unit

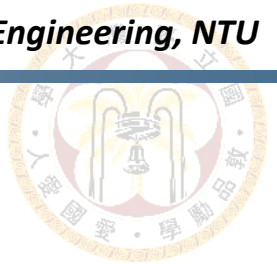
Graduate Institute of Electronics Engineering, National Taiwan University

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

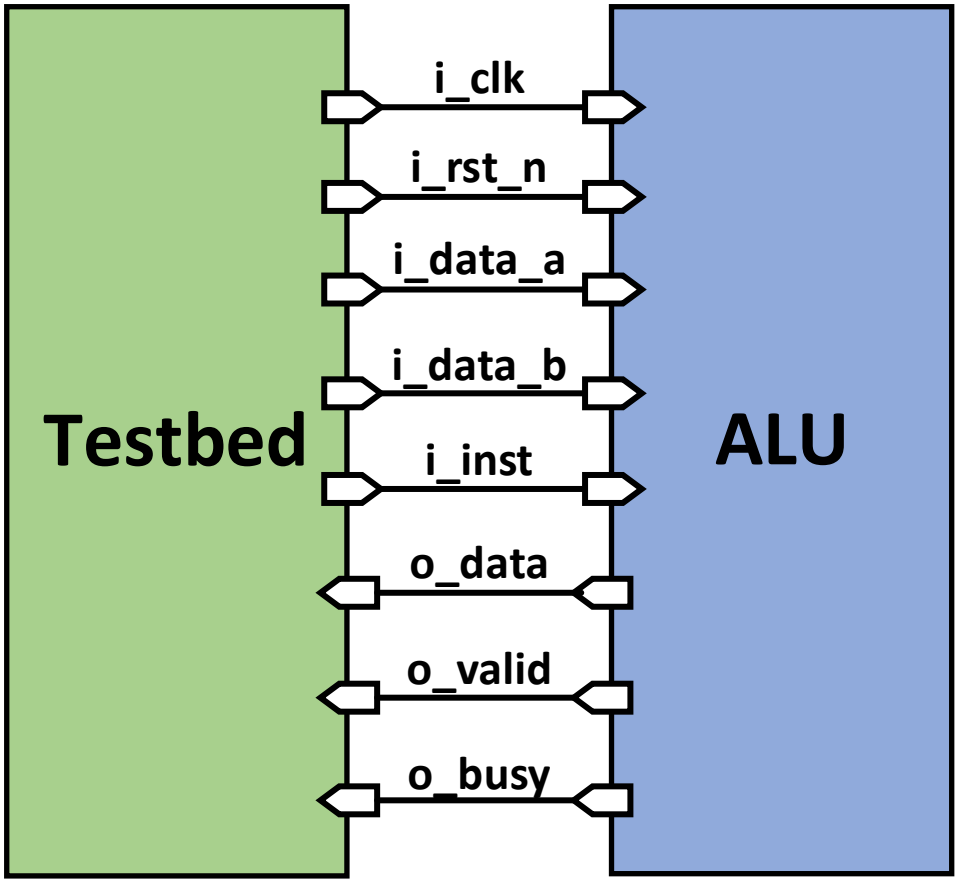


Introduction

- The Arithmetic logic unit (ALU) is one of the components of a computer processor

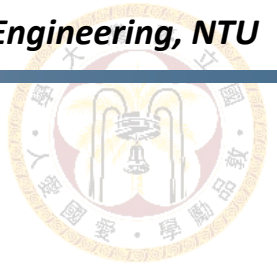
- In this homework, you are going to design an ALU with some special instructions, and use the ALU to compute input data to get the correct results

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_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	
i_inst	I	4	Instruction for ALU to operate
o_valid	O	1	Set **high** if ready to output result
o_data	O	16	1. For instructions 0000~0100, result after ALU processing 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.

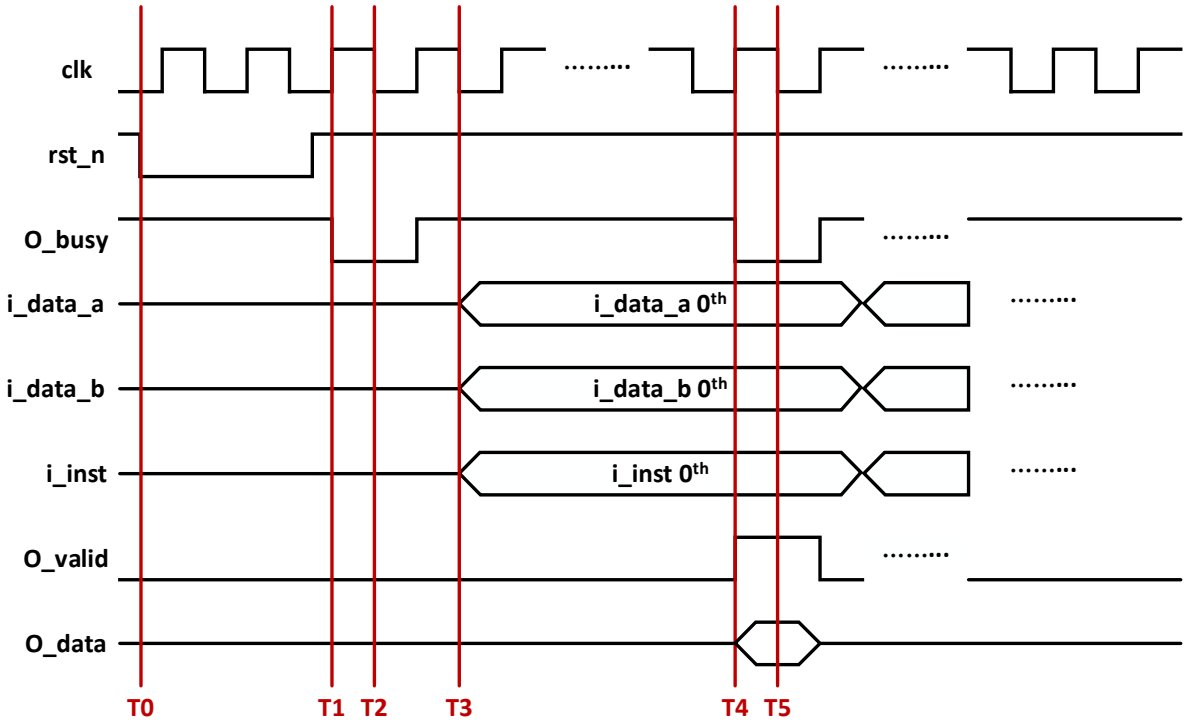


Specification (1)

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

Specification (2)

- T0~T1, ALU circuit reset.
- T2, o_busy is low, testbed send the 0^{th} input data at T3.
- T4, ALU computation is done. Set o_valid high.
- T5, testbed check the computation result.

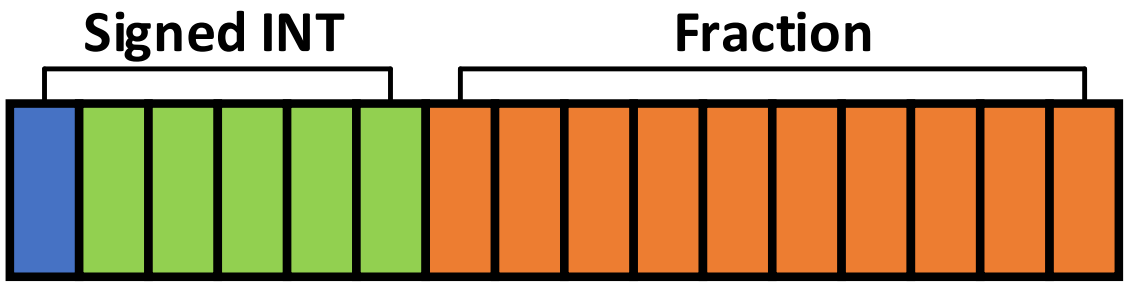


Instruction

Operation	i_inst [3:0]	Description	Note
Signed Addition (FX)	4'b0000	o_data = i_data_a + i_data_b	Output saturation (FX) and rounding is needed (FX/FP)
Signed Addition (FP)	4'b1000		
Signed Subtraction (FX)	4'b0001	o_data = i_data_a - i_data_b	
Signed Subtraction (FP)	4'b1001		
Signed Multiplication (FX)	4'b0010	o_data = i_data_a * i_data_b	
MAC	4'b0011	o_mult = i_data_a * i_data_b $o\_data_{new} = o\_mult + o\_data_{old}$	
GELU	4'b0100	o_data = GELU(i_data_a)	Piecewise linear approximation
CLZ	4'b0101	Count leading zero bits	Only i_data_a is used
LRCW	4'b0110	Encode the CPOP result	
LFSR	4'b0111	Generate pseudo-random number	

Data Format (I)

- In this homework, for instruction 0000~0100, the input data is in fixed-point format.
- Fixed point (6-bit signed integer + 10-bit fraction)
  - Check reference [1]
  - Output Saturation & Rounding



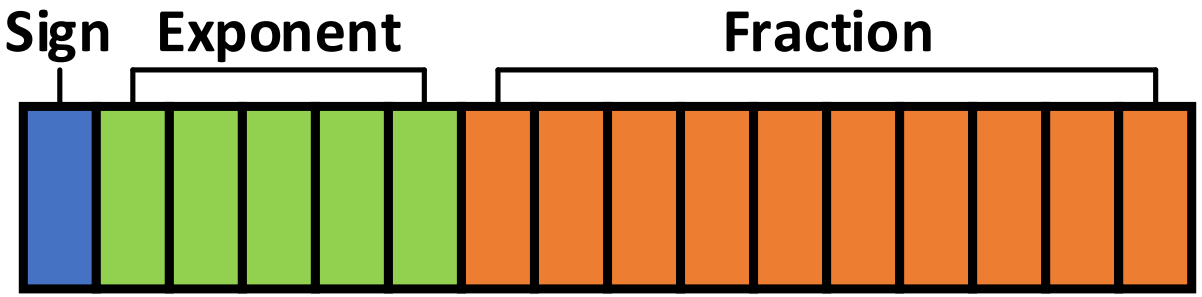
Output Saturation & Rounding

- For instructions I0~I4, If the output value exceeds the maximum value of 16-bit representation, use the maximum value as output, and vice versa.
- For instructions I2~I4, the result needs to be **rounded to the nearest number**.
  - Check reference [2] (Mode=Nearest)

Fixed-Point Designer Rounding Mode	Description	Tie Handling
Ceiling	Rounds to the nearest representable number in the direction of positive infinity.	N/A
Convergent	Rounds to the nearest representable number.	Ties are rounded to nearest even number.
Floor	Rounds to the nearest representable number in the direction of negative infinity. Equivalent to two's complement truncation.	N/A
Nearest	Rounds to the nearest representable number.	Ties are rounded to the closest representable number in the direction of positive infinity.
Round	Rounds to the nearest representable number.	• For positive numbers, ties are rounded to the nearest representable number in the direction of positive infinity. • For negative numbers, ties are rounded to the nearest representable number in the direction of negative infinity.

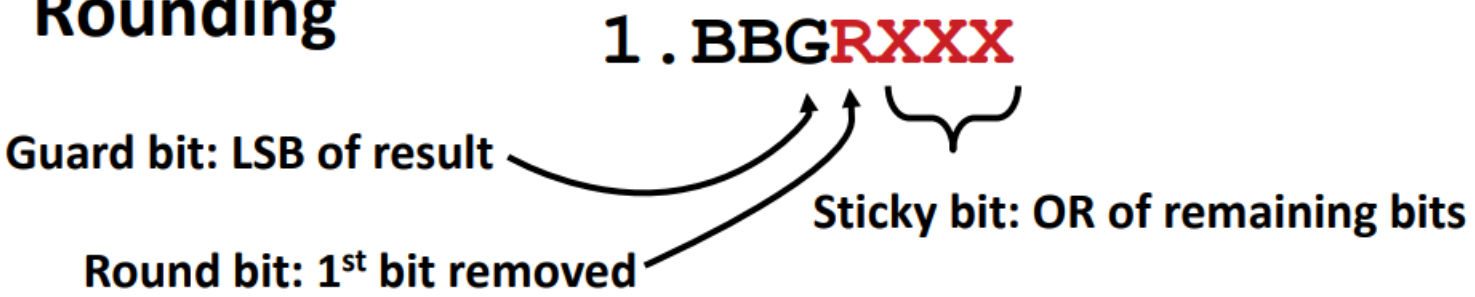
Data Format (II)

- In this homework, for instruction 1000~1001, the input data is in floating point format.
- The input and output will be valid number.
  - You do not need to consider the exception.
  - The difference between the two exponents is less than 11.
- Floating point (FP16) [3]
  - Round to nearest, ties to even [4]



Round to Nearest, Ties to Even [4]

Rounding

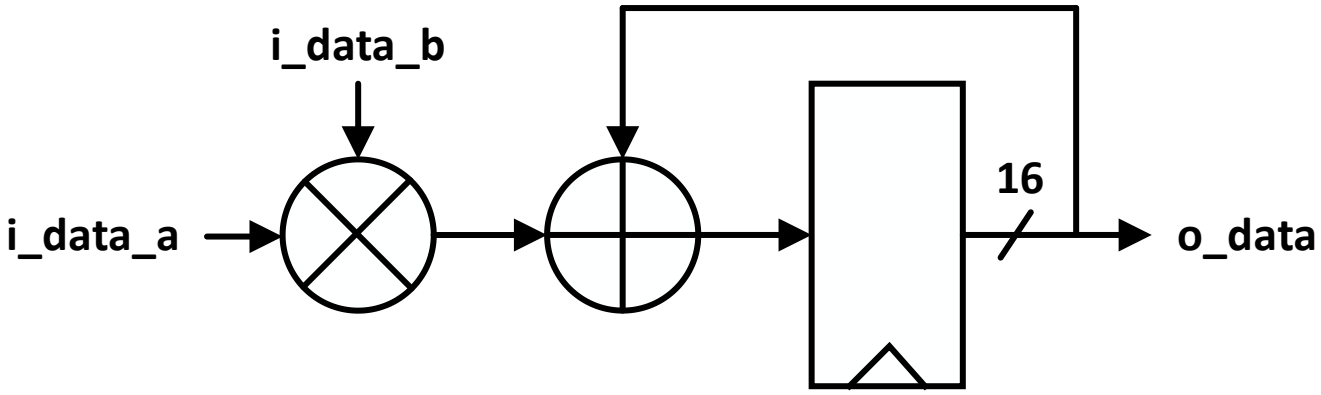


- **Round up conditions**
  - Round = 1, Sticky = 1 → > 0.5
  - Guard = 1, Round = 1, Sticky = 0 → Round to even

Value	*Fraction*	*GRS*	*Incr?*	*Rounded*
128	1.0000000	000	N	1.000
15	1.1010000	100	N	1.101
17	1.0001000	010	N	1.000
19	1.0011000	110	Y	1.010
138	1.0001010	011	Y	1.001
63	1.1111100	111	Y	10.000

MAC

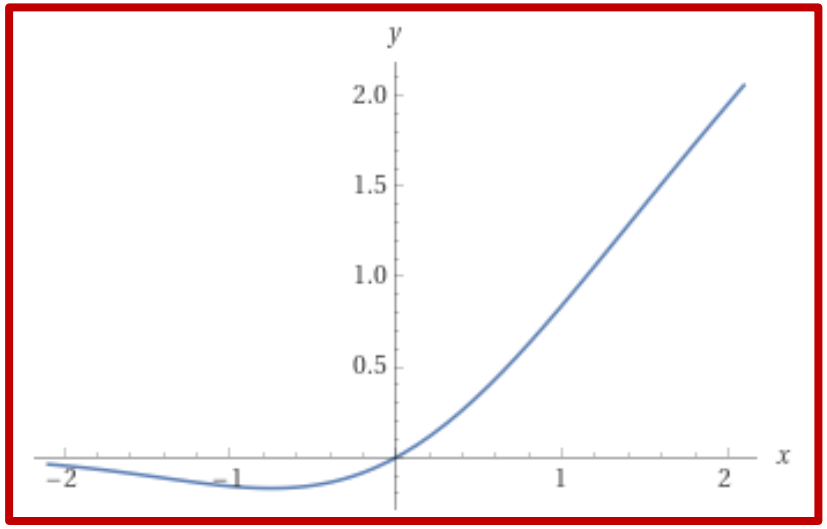
- For instruction 0011, you have to implement the multiply–accumulate operation (MAC) function.
- The output of the Flip-flop must be rounded to the nearest number.
- Multiply-accumulate first, then rounding and saturation.
- In hidden pattern, MAC operation will test with other instructions (I0~I3).

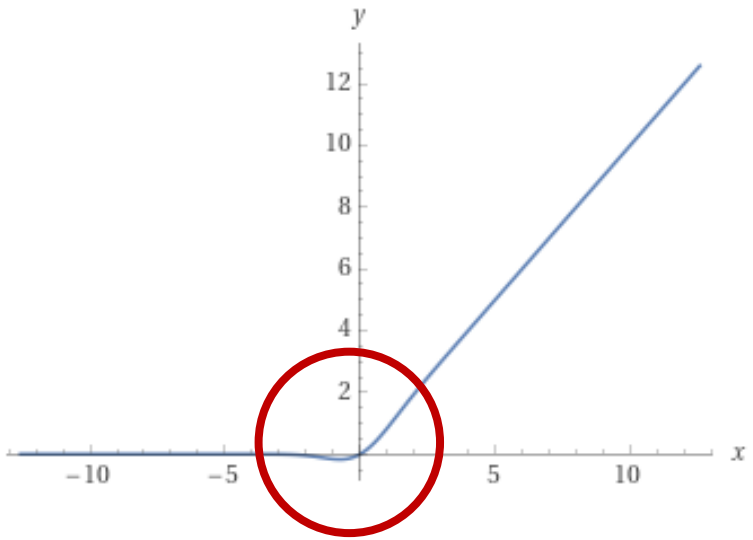


GELU Function

- For instruction 0100, you need to implement an activation function, Gaussian Error Linear Units function [5], which is used in the transformer model recently.
- We approximate the GELU by tanh:

$$\mathrm{GELU}(x) = 0.5x \cdot (1 + \tanh(0.7978515625x \times (1 + 0.044921875x^2)))$$

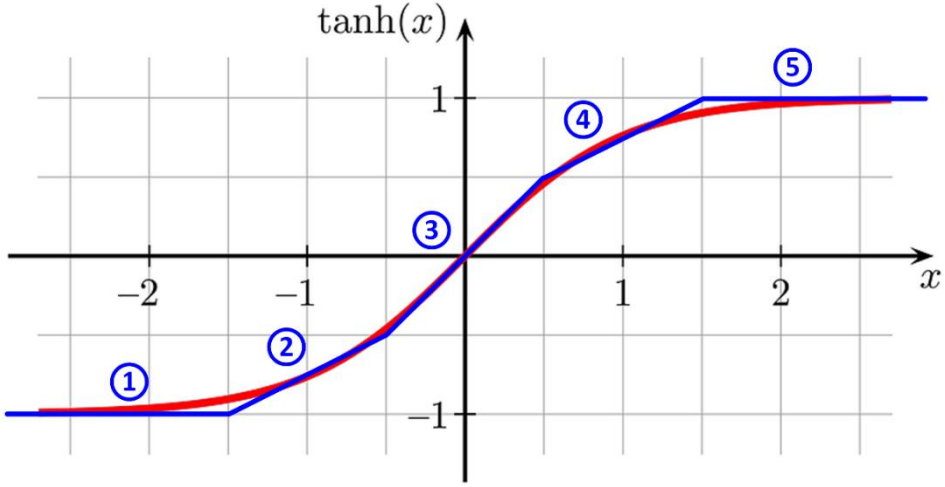




Tanh Function Approximation

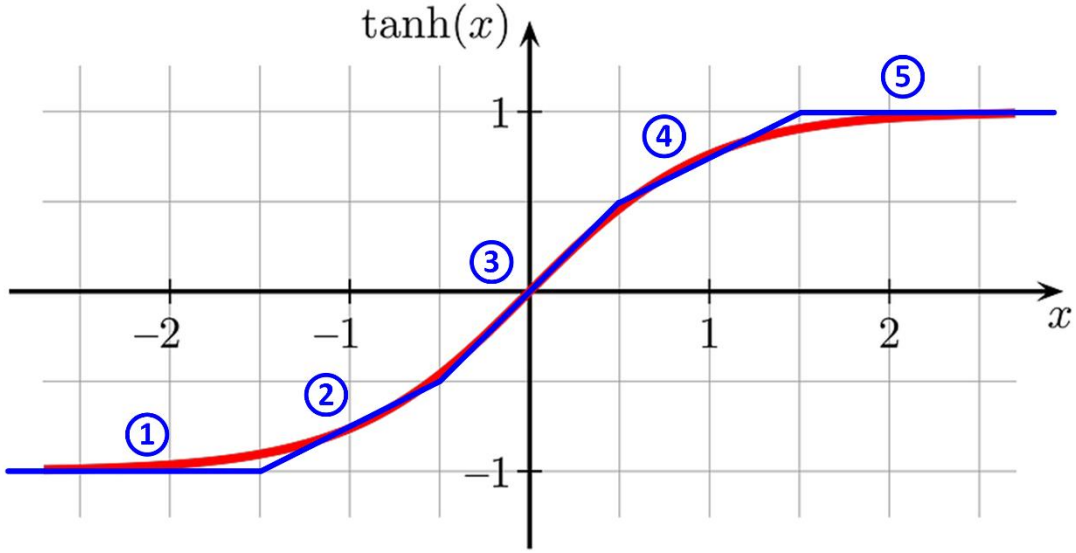
- However, it's not intuitive to implement an exponential operation on hardware for tanh function.
- In order to simplify the implementation, we use **piecewise linear approximation** to compute tanh function.
- We divide the curve into **5 segments** to compute the output.

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



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 of tanh function must be **rounded to the nearest number**.



GELU Function Approximation

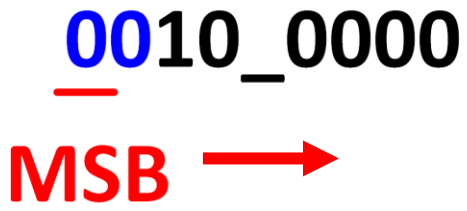
- **Round to the nearest number:**
  - Round the input value before tanh function (1)
  - Round the output value after tanh function (2)
  - Round the final GELU value (3)

(3) (2) (1)

$$\mathrm{GELU}(x) = 0.5x \cdot (1 + \tanh(0.7978515625x \times (1 + 0.044921875x^2)))$$

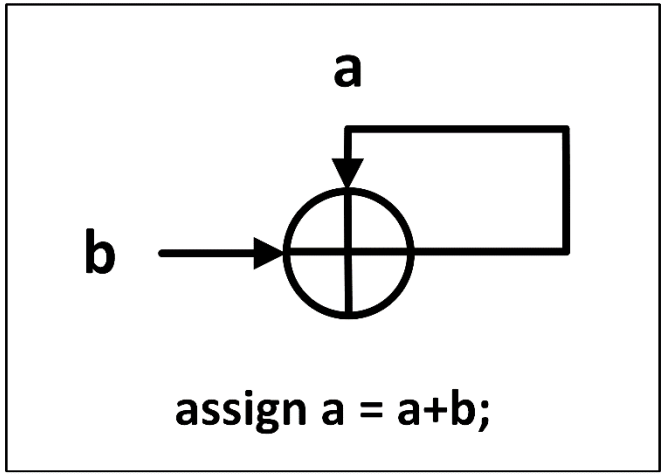
CLZ

- For instruction 0101, count leading zero.
  - Count the number of zero bits from MSB end
  - If the input is zero (no bits set), return the number of bits of the data
- For example, if a = 8'b0010_0000, then CLZ(a)=2.



CPOP

- 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

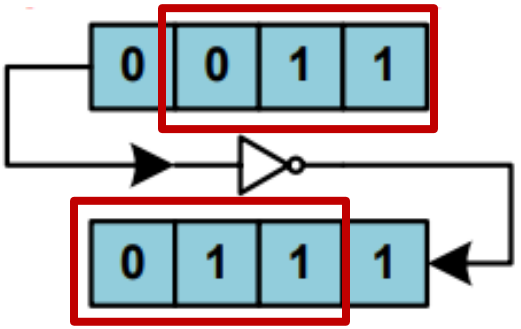
LRCW

- For instruction 0110, you need to implement an unary encoding of the pop count by Left rotation and complement-on-warp [6].
- View i_data_a as the bit mask.
- View i_data_b as the initial value.
- Calculate the CPOP of the bit_mask (i_data_a) first, and encode it by LRCW, which is initialized by i_data_b.
  - The number of the set bits determines the iteration times.
- 4-bit example:

i_data_a: 0001

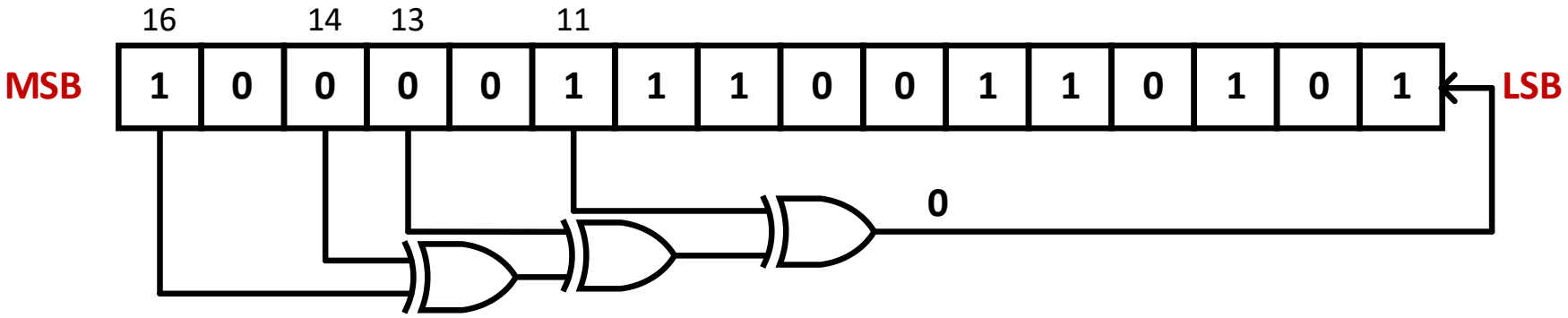
i_data_b: 0011

CPOP(a) = 1



LFSR

- Linear feedback shift register is often used to generate pseudo-random numbers.
- View i_data_a as the initial value (seed).
- View i_data_b as the number of the iterations (max = 8).



- After one iteration:

1000_0111_0011_0101 → 0000_1110_0110_1010

testbench.v

```
`ifdef I0
    `define Inst_I  "../00_TESTBED/pattern/INST0_I.dat"
    `define Inst_O  "../00_TESTBED/pattern/INST0_O.dat"
    `define PAT_NUM 40
    `define FLAG 0
`elsif I1
    `define Inst_I  "../00_TESTBED/pattern/INST1_I.dat"
    `define Inst_O  "../00_TESTBED/pattern/INST1_O.dat"
    `define PAT_NUM 40
    `define FLAG 0
`elsif I2
    `define Inst_I  "../00_TESTBED/pattern/INST2_I.dat"
    `define Inst_O  "../00_TESTBED/pattern/INST2_O.dat"
    `define PAT_NUM 40
    `define FLAG 0
`elsif I3
    `define Inst_I  "../00_TESTBED/pattern/INST3_I.dat"
    `define Inst_O  "../00_TESTBED/pattern/INST3_O.dat"
    `define PAT_NUM 40
    `define FLAG 0
`elsif I4
    `define Inst_I  "../00_TESTBED/pattern/INST4_I.dat"
    `define Inst_O  "../00_TESTBED/pattern/INST4_O.dat"
    `define PAT_NUM 40
    `define FLAG 0
```

```
`elsif I5
    `define Inst_I  "../00_TESTBED/pattern/INST5_I.dat"
    `define Inst_O  "../00_TESTBED/pattern/INST5_O.dat"
    `define PAT_NUM 40
    `define FLAG 0
`elsif I6
    `define Inst_I  "../00_TESTBED/pattern/INST6_I.dat"
    `define Inst_O  "../00_TESTBED/pattern/INST6_O.dat"
    `define PAT_NUM 40
    `define FLAG 0
`elsif I7
    `define Inst_I  "../00_TESTBED/pattern/INST7_I.dat"
    `define Inst_O  "../00_TESTBED/pattern/INST7_O.dat"
    `define PAT_NUM 40
    `define FLAG 0
`elsif I8
    `define Inst_I  "../00_TESTBED/pattern/INST8_I.dat"
    `define Inst_O  "../00_TESTBED/pattern/INST8_O.dat"
    `define PAT_NUM 40
    `define FLAG 1
`elsif I9
    `define Inst_I  "../00_TESTBED/pattern/INST9_I.dat"
    `define Inst_O  "../00_TESTBED/pattern/INST9_O.dat"
    `define PAT_NUM 40
    `define FLAG 1
`endif
```

Commands

- ./01_run arg1

  `vcs -f rtl.f -full64 -R -debug_access+all +v2k +define+$1 | tee sim.log`

  - Modify the rtl.f if you have multiple .v file
  - For example: `./01_run I0` (The argument will be I0~I9)
- ./02_lint
  - SpyGlass lint check
  - Modify the flist.v if you have multiple .v file
- ./99_clean
  - Command for cleaning temporal files
- 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_01010110011000010011101000100011
0000_00001101010000110001001110101011
0000_01101010010000111011111000101101
0000_00101111110001111001001111111110000
0000_00010010110111011001010100001010
0000_01001101011110110001110110111110
0000_01011101100000110010011010001111
0000_01111100100110101001101111111100110
0000_10000000000100010011000000001111000
0000_01011010000101111011101111100001010
0000_00011010010111110100000111001100
```

Pattern (Golden Output)

o_data

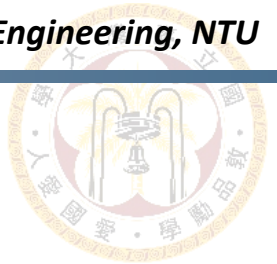
```
0111111111111111
0010000011101110
0010100001110000
0101011101111111
1010011111100111
0110101100111001
0111111111111111
0111111111111111
1110000010011010
0111111111111111
0101110000101010
```

Grading Policy (1)

- 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%**
  - Hidden pattern contains all instructions
    - I0~I3: 8000 (mix)
    - I4~I9: 2000 for each instruction
  - Only if you pass all patterns will you get the full 30% score.
- Spyglass check with error (-20).
- **Do not** use DesignWare for floating point operations.

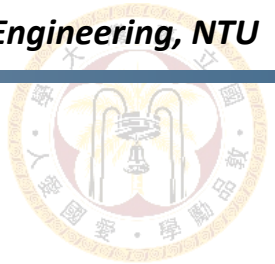
Grading Policy (2)

- No late submission
  - **0 point** for this homework
- Lose 5 **points** for any wrong naming rule or format for submission
- Make sure the code you upload can be unzipped and executed
- No plagiarize



Non-synthesizable Code

- Refer to the lecture note
- Use SpyGlass to check your code
  - Goal setup: lint_rtl and lint_rtl_enhanced
- Combinational loop
- Non-synthesizable code
  - Delay
  - initial block
  - Concurrent block (fork-join)
  - forever/while/repeat loop
  - etc.
- Without default value in case/if statement will infer latches



Lint.tcl

- 02_lint will run the lint.tcl to check your code
- Check the linting reports in the following path
  - spyglass-1/alu/lint/lint_rtl/spyglass_reports/spyglass_violations.rpt
  - spyglass-1/alu/lint/lint_rtl_enhanced/spyglass_reports/spyglass_violations.rpt
- Add all the .v files in the flist.v

```
read_file -type verilog {flist.v}
set_option top alu
current_goal Design_Read -top alu
current_goal lint/lint_rtl -top alu
run_goal
capture ./spyglass-1/alu/lint/lint_rtl/spyglass_reports/spyglass_violations.rpt {write_report spyglass_violations}
current_goal lint/lint_rtl_enhanced -top alu
run_goal
capture ./spyglass-1/alu/lint/lint_rtl_enhanced/spyglass_reports/spyglass_violations.rpt {write_report spyglass_violations}

exit -force
```

Submission

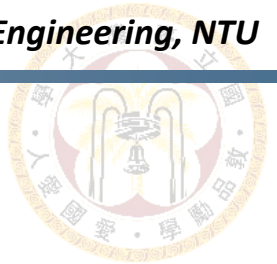
- Create a folder named **studentID_hw1** and follow the hierarchy below

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

- Compress the folder **studentID_hw1** in a tar file named **studentID_hw1_v*k*.tar** (*k* is the number of version, *k* =1,2,…)
  - Use lowercase for student ID. (Ex. r11943123_hw1_v1.tar)
  - Remember to put all the files in the **studentID_hw1**
- Submit to NTU Cool

Discussion


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 fixed-point representation
  - Fixed-Point Representation
- [2] Reference for rounding to the nearest
  - Rounding - MATLAB & Simulink
- [3] Reference for FP16 representation
  - Wikipedia/Half-precision_floating-point_format
- [4] CMU Lecture – Floating point
  - CMU Lecture
- [5] Reference for GELU
  - https://arxiv.org/pdf/1606.08415v3.pdf
- [6] Reference for LRCW
  - Comparing fast implementations of bit permutation instructions