

## Lab2: Design of Arithmetic Logic Unit(ALU)

**Instructor: Lih-Yih Chiou** 

Speaker: Jay

Date: 2024/02/29





#### **Outline**

- Introduction
- Hardware description
  - Block diagram
  - Instruction
  - I/O Information
- Lab2 Implementation
  - Data format
  - Saturation & Rounding
  - Piecewise linear approximate for the tanh function
- Criteria
  - Grading policy
  - Simulation Result
  - Requirement & file format



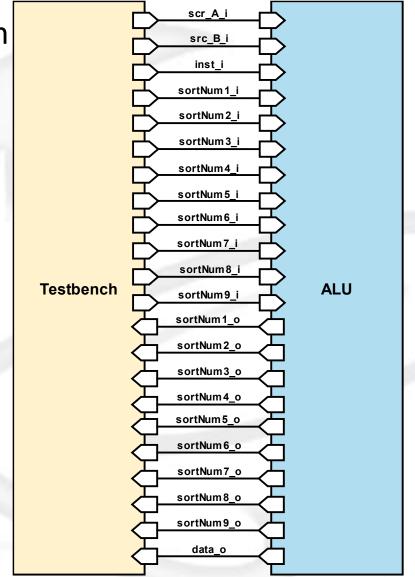
#### Introduction

- Arithmetic logic unit (ALU)
  - → Is one of the components of a computer processor.
- In this lab, you're going to design an ALU with some special instructions. Use the ALU to compute input data to get the correct results.





Block Diagram







#### **Hardware description**

Instruction

Operation	inst_i	Desc.	Note
Signed Addition (Fixed Point)	3′b000	data_o = scr_A_i + scr_B_i	Output saturation and rounding is
Signed Subtraction (Fixed Point)	3'b001	data_o = scr_A_i - scr_B_i	needed. (Rounding policy: round to the nearest, ties to even)
Signed Multiplication (Fixed Point)	3'b010	data_o = scr_A_i * scr_B_i	nearest, ties to eveny
GeLU	3'b011	data_o = GeLU(scr_A_i)	Output is needed; only scr_A_i is used. (Rounding policy: round to the nearest, ties to even)
CLZ	CLZ 3'b100 Count leading 2		Only scr_A_i is used.
Sort nine numbers	3'b101	Sort nine numbers in ascending order.	



#### **Hardware description**

□ I/O Information

Signal	I/O	width	Desc.
scr_A_i	I	16	For instruction 000~011, signed input data with 2's complement representation.
src_B_i	I	16	(6-bit signed integer + 10-bit fraction) For instruction 100, 16-bit number.
inst_i	- 1	3	Operation code select which operation to be executed
sortNum*_i	Ι	8	Unsigned 8-bit number
sortNum*_o	0	8	Unsigned 8-bit number
data_o	0	16	For instruction 000~011, signed input data with 2's complement representation.  (6-bit signed integer + 10-bit fraction)  For instruction 100, 16-bit number.  For instruction 101, don't care.



- Why Fixed point?
  - → Floating-point format in Verilog is not synthesizable; you need to use fixed-point or IEEE 754 floating-point to represent it.
    - Ex: "a = b \* 0.123;" is not synthesizable.
- Fixed point
  - Designer can decide the format.
  - → It often requires less hardware resources compared to floating-point arithmetic.
  - → In this lab, for instruction 000~011, the input data is in fixed-point format.

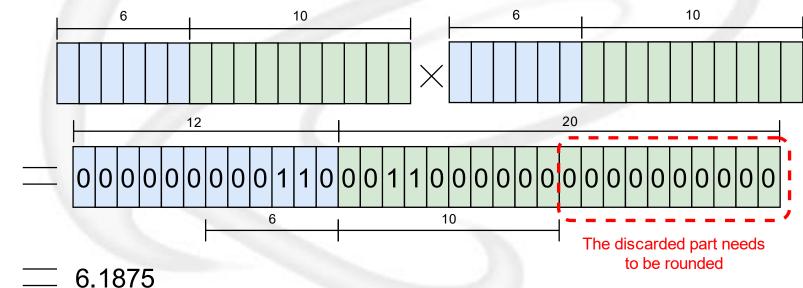


## LPHPLHR VLSI Design LAB

### **Lab2 Implementation**

- Fixed point multiplication

  - → We only want the 6-bit integer and the 10-bit fractional part of the answer.







- Saturation & Rounding
  - → For instructions 3'b000~3'b010, If the output value exceeds the maximum, value of 16-bit representation, use the maximum value as output, and vice versa.
  - → For instructions 3'b010 to 3'b011, you need to round the result to the nearest value.
    - Rounding policy: Round to the Nearest, ties to Even (RNE)

Ex: To round to the nearest, to the second decimal place.

- 1. Greater than 5, round up
- 2. less than 5, round down
- 3. equal to 5, look at the preceding digit if it's odd, round up; if it's even, round down





Here are some examples of output saturation

#### Addition

— input —								
■-  /tb_ALU/UUT/inst_i	0	0						
■-  // /tb_ALU/UUT/scr_A_i	-11.408	-1.5518	5.3564	16	.289	<u> </u>	16.725	
<b>■</b> /tb_ALU/UUT/src_B_i	-13.312	17.707	-20.818	8.	7832		20.144	
— output ———						<u>,                                    </u>		
	-24.720	16.155	-15.462	25	.072	<u>\</u>	31.999	

#### Subtraction

— input —					
■ / tb_ALU/UUT/inst_i	1	1			
•- /tb_ALU/UUT/scr_A_i	15.721	-14.387	-20.232	-8.9678	-18.771
■-4/ /tb_ALU/UUT/src_B_i	15.519	-10.752	2.9141	-12.251	18.904
- output -					
<b>II</b> - <b>1</b> / <b>L</b> /tb_ALU/UUT/data_o	0.20215	-3.6348	-23.146	3.2832	-32.000

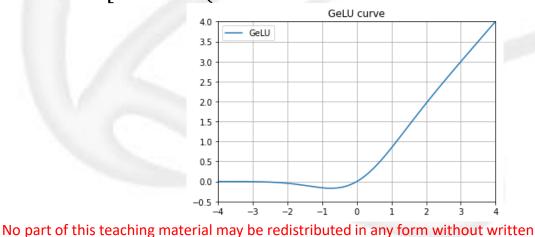
#### Multiplication

input						
■- <pre>// tb_ALU/UUT/inst_i</pre>	2	2			,	
■-  // tb_ALU/UUT/scr_A_i	0.0019531	-0.055664	-19.867	0.086914	-12.851	
■-  /tb_ALU/UUT/src_B_i	0.013672	0.026367	11.367	0.083984	-7.4326	
- output -						
<b>■</b> - <b></b> /tb_ALU/UUT/data_o	0.00000	-0.0019531	-32.000	0.0068359	31.999	

- ☐ GeLU (Gaussian Error Linear Unit)
  - → Compared with ReLU, the GELU function has a non-zero gradient at x = 0, allowing the network to learn in this region.
  - → We approximate the GeLU function using the Tanh function.

GeLU(x)

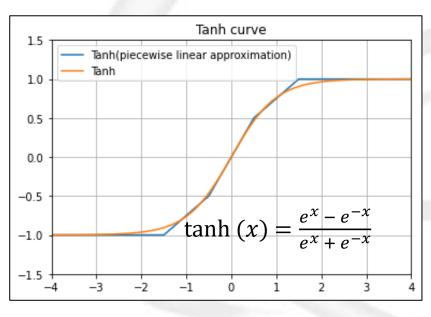
$$\approx 0.5 * x * \left[ 1 + \tanh \left( \sqrt{\frac{2}{\pi}} * x * (1 + 0.044715 * x^2) \right) \right]$$
  
 
$$\approx 0.5 * x * \left[ 1 + \tanh \left( 0.7978515625 * x * (1 + 0.044921875 * x^2) \right) \right]$$

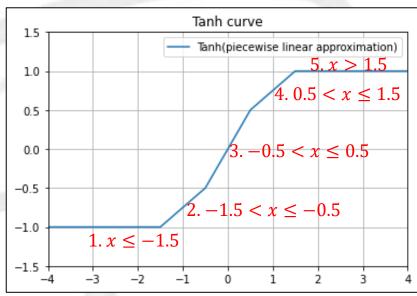






- Piecewise linear approximation
  - → We use piecewise linear approximation to implement the tanh function.
  - We divide the curve into 5 segments to compute the output.







# LPHPLHB VLSI Design LAB

### **Lab2 Implementation**

- Rounding the result of the GELU function
  - → Round to the nearest number with ties rounding to the even number (16-bits, 6-bit integer + 10-bit fraction).
    - Round the value before the tanh...1
    - Round the value after the tanh ... 2
    - Round the final value ... 3

GeLU(x)

$$\approx 0.5 * x * [1 + \tanh(0.7978515625 * x * (1 + 0.044921875 * x^2))]$$





- CLZ (Count Leading Zero bits)
  - → Ex: if input = 8'b0001\_1101, then output = 3

	input											
8	<b>II-</b>	4	4									
2	<b>II-</b>	01101011	01101	00000	101110	10011	000	10000010	000010	00000000000	001	000
	<b>II-</b>	0	0									
	output											
3	<b>■-</b> /tb_ALU/UUT/data_o	1	1	5			3			12		





- Sort 9 numbers
  - → The inputs are 8-bit unsigned numbers.
  - → Avoid the use of the "for-loop" in Verilog to solve this problem.

insul						
— input —						
<b>I →</b> /tb_ALU/UUT/sortNum1_i	207	207	181	112	110	
→ /tb_ALU/UUT/sortNum2_i	14	14	240	80	138	\
<u>+</u>	98	98	56	146	249	ľ
■-  /tb_ALU/UUT/sortNum4_i	10	10	67	222	111	
<u>+</u> √ /tb_ALU/UUT/sortNum5_i	25	25	228	153	76	
/tb_ALU/UUT/sortNum6_i	62	62	78	132	146	
■-  // tb_ALU/UUT/sortNum7_i	207	207	185	62	88	*
■-  // tb_ALU/UUT/sortNum8_i  // page 1.5 in the second of the seco	8	8	198	196	237	
<u>+</u> √ /tb_ALU/UUT/sortNum9_i	242	242	155	19	236	*
— output ———						
small =- / /tb_ALU/UUT/sortNum1_o	8	8	56	19	76	
	10	10	67	62	88	<b> </b>
/tb_ALU/UUT/sortNum3_o	14	14	78	80	110	*
■-  /tb_ALU/UUT/sortNum4_o	25	25	155	112	111	
/tb_ALU/UUT/sortNum5_o	62	62	181	132	138	
■-  /tb_ALU/UUT/sortNum6_o	98	98	185	146		
	207	207	198	153	236	*
<b>II</b> - ✓ /tb_ALU/UUT/sortNum8_o	207	207	228	196	237	
large #-4 /tb_ALU/UUT/sortNum9_o	242	242	240	222	249	ľ







#### Criteria

- ☐ Grading policy(100%)
  - → Lab2
    - Simulation pass (90%)
      - ➤ Signed Addition (16%)
      - ➤ Signed Subtraction (16%)
      - ➤ Signed Multiplication (16%)
      - ➤ GeLU (10%)
      - ➤ CLZ (16%)
      - ➤ Sort nine numbers (16%)
    - Report (10%)



#### **Criteria**

- Simulation result
  - Pass

```
VSIM 52> run -all
# Instruction 0 ALL PASS !!!
# Instruction 1 ALL PASS !!!
# Instruction 2 ALL PASS !!!
# Instruction 3 ALL PASS !!!
# Instruction 4 ALL PASS !!!
 Instruction 5 ALL PASS !!!
      Congratulations !!
      Simulation PASS!!
 ===== Your score : 90 / 90 =====
 ** Note: $stop : E:/HDL course prepare/Lab2 ALU/tb ALU.sv(188)
    Time: 60 us Iteration: 0 Instance: /tb ALU
# Break in Module tb ALU at E:/HDL course prepare/Lab2 ALU/tb ALU.sv line 188
```



- Simulation result
  - → Failed

```
/SIM 54> run -all
       Simulation Start
Instruction 0 ALL PASS !!!
Instruction 1 ALL PASS !!!
Instruction 2 ALL PASS !!!
Instruction 4 ALL PASS !!!
 Instruction 5 ALL PASS !!!
    OOPS!!
                             / x,x
    Simulation Failed!!
 ===== Your score : 80 / 90 =====
** Note: $stop : E:/HDL course prepare/Lab2 ALU/tb ALU.sv(188)
   Time: 60 us Iteration: 0 Instance: /tb ALU
Break in Module tb ALU at E:/HDL course prepare/Lab2 ALU/tb ALU.sv line 188
```



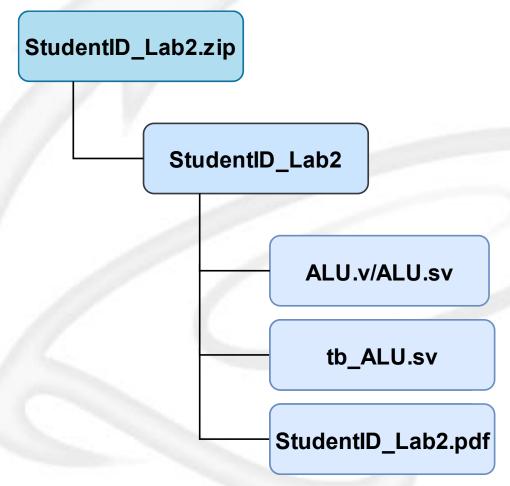
#### Lab2 Requirement & file format

- You must finish ALU.v/.sv and pass all patterns
- ☐ For Lab2, you need to submit
  - → ALU.v / ALU.sv
  - tb\_ALU.sv
  - StudentID\_Lab2.pdf
- □ Deadline:2024/03/06 23:59 (No late submission)



### Lab2 Requirement & file format

Homework submission must follow the file hierarchy!



## Thanks for listening



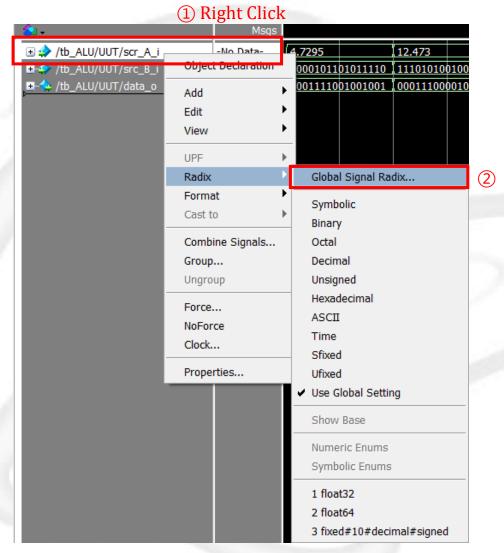
## How to view fixed-point format in ModelSim



# LPHPLER VLSI Design LAB

#### How to view fixed-point format in ModelSim

Step1

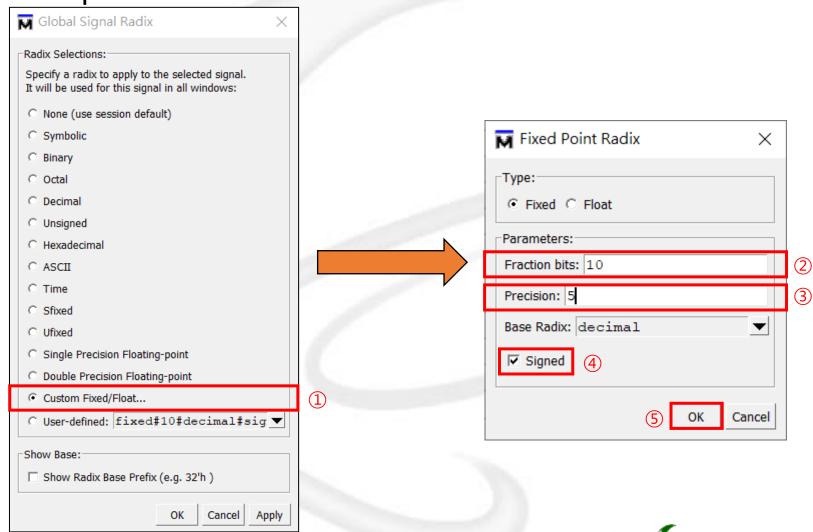




# **LPHPLMB** VLSI Design LAB

#### How to view fixed-point format in ModelSim

☐ Step2





## **LPHPLME** VLSI Design LAB

#### How to view fixed-point format in ModelSim

☐ Step3

