University of Florida **EEL 3701C: Digital Logic & Computer Systems** 

Department of Electrical & Computer Engineering

Revision 0

Lab 4 Report: ALU and RALU

PI Name: Jaiden Magnan

March 13, 2025

Poche, Natalie

Class #: 11198

# REQUIREMENTS NOT MET

N/A

Page 1/13

## VIDEO FILE LINK

https://youtu.be/TXu9-nCmeNc?si=O6ZkBewK-NffY66y

#### PROBLEMS ENCOUNTERED

None, this just takes a long time.

## **FUTURE WORK/APPLICATIONS**

ALUs and RALUs efficiently process complex operations, and with the flexibility of RALUs, they can be valuable in hardware design for anything that involves automated computing. This will be useful in future labs where we have to design a traffic light that involves specific actions being taken based on specific patterns of inputs.

Class #: 11198 PI Name: Jaiden Magnan

Poche, Natalie

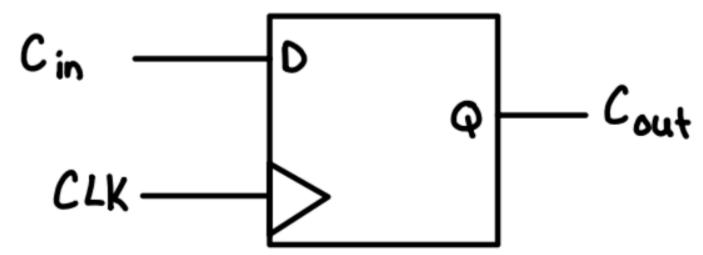
March 13, 2025

Page 2/13

Lab 4 Report: **ALU and RALU** 

## PRE-LAB QUESTIONS OR EXERCISES

1. Draw the single simple device that can be added to your circuit design to "remember" the last carry output. Specify the inputs and outputs for this device.



A D Flip-Flop would be useful to remember the last carry output. For this device, the inputs would be the carry input and clock signa, the output would be the carry output.

- 2. Will a divide by two work for all 4-bit 2's complement numbers? Explain.

  Under the assumption that they are unsigned, dividing by 2 would work for all 4-bit 2's compliment numbers, as all positive numbers are able to be divided by 2, however this changes when you get to negative numbers since a sign extension would need to be considered.
- 3. Describe how you can take the 2's complement of a number, i.e., if A is loaded with a number, get the 2's complement of A into B.
  - First, I would use the compliment of A in order to get 1's compliment at the same time storing the number 1 in B, then I would add B to the 1's compliment in order to increment it by 1 and store the result in B.
- 4. Describe how you subtract with your RALU. Hint: See the previous question. First, I would store the subtracting in A and the number being subtracted from in B (EX: 1-2, A = 2, B = 1). Then I would take the 2's compliment of the subtracting number in A, keep it stored in A, and then add the 2's compliment number to the input value number that is being subtracted from in B.
- 5. Suppose you're not allowed to use a flip-flop that has an asynchronous CLR or SET, how can you add a function that clears the contents of either A or B?
  - Then you could have designed something with a clear signal. Additionally, an AND gate would be most effective to determine when the clear signal is triggered. The AND gate would be the register's current value and the control signal and the resulting output would be put into the register.

University of Florida

Page 3/13 Lab 4 Report: **ALU and RALU** 

PI Name: Jaiden Magnan March 13, 2025

Poche, Natalie

Class #: 11198

# PRE-LAB REQUIREMENTS (Design, Schematic, ASM Chart, VHDL, etc.)

Each section of the pre-lab requirements should be completed separately, and in order. Include each of the following items in order. Note that some of these items will not apply to every lab. Anything scanned or copied *must be clear and legible*.

- Logic equations. (Note that logic equations do not contain activation levels.)
- All tables and figures should have captions with Figure/Table numbers and a description of for which part of the lab it references, e.g., *Table 3: Truth Table for Part B*.
- Truth tables and voltage tables (and/or next-state truth tables).
- When applicable, include Karnaugh Maps (i.e., K-Maps).
- Include hand-drawn circuits (when required). Label all input and output activation-levels and intermediate equations in the circuits.
- Include screenshots of the BDF designs of circuits.
  - Label all input and output activation-levels, i.e., use \_L suffix for active-low signals and no suffix for active-high signals. Add chip and pin numbers to any schematic that will be constructed.
  - o Images should be large enough so that inputs, outputs, labels, and parts are clearly visible and distinguishable to any reader.
  - o Each BDF should have the following info on the top left corner (similar to the top right of this page):

Last Name, First Name

Lab #, Part #

Class #

PI Name:

Description: (short description of what is to be accomplished in the design; perhaps an equation)

- o In Windows, I use the *Snipping Tool*, which is now built into Windows. Just type "snip" in the Windows search box and then select *Snipping Tool*.
- When necessary, include ASM Charts. These can be hand-drawn, but clear and legible. We recommend that you use resources like <a href="https://www.draw.io/">https://www.draw.io/</a> to create computer-generated ASMs.
- Truth tables or next-state truth tables should have the following characteristics.
  - o Can be either typed or hand-written and scanned (must be clear and legible)
  - o Must be in **counting order** (i.e., inputs of 000, 001, 010, 011, ..., 111)
  - o Clearly distinguish inputs from outputs (see the example below that uses a thick line)
  - o If you are designing a state machine or a controller, clearly indicate and separate signal values both before the clock and after the clock (i.e., Q1 and Q1<sup>+</sup>, respectively)
  - o Tip: divide rows into consecutive groups of 4 (or 2 or 8) to make it easier for both you and your PI to read.
  - o Example

A	В	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Table: Caption for above table. It should reference the part of the lab, e.g.,

University of Florida **EEL 3701C: Digital Logic & Computer Systems** 

Department of Electrical & Computer Engineering

Revision 0

Lab / Report: ALLI and RALI

Page 4/13 Lab 4 Report: **ALU and RALU** PI Name: Jaiden Magnan March 13, 2025

Poche, Natalie

Class #: 11198

• Voltage tables should have the following characteristics.

- O Must be in counting order (i.e., inputs of LLL, LLH, LHL, ..., HHH)
- Use similar formatting to truth tables (described above)

**EEL 3701C: Digital Logic & Computer Systems** University of Florida Revision 0

Department of Electrical & Computer Engineering

Lab 4 Report: **ALU and RALU** 

PI Name: Jaiden Magnan March 13, 2025

Poche, Natalie

Class #: 11198

#### o Example

Page 5/13

A(H)	B(H)	C(L)	Y(L)
L	L	L	Н
L	L	Н	Н
L	Н	L	L
L	Н	Н	Н
Н	L	L	Н
Н	L	Н	Н
Н	Н	L	Н
Н	Н	Н	L

- Include **meaningfully annotated** functional simulations.
  - Using the grouping tool, *group as many signals together as possible* (when it makes sense to group them)! If you are simulating a basic logic equation, group all the inputs together. If you are simulating a circuit that includes MSI elements, group signals of the form  $X_{N-0}$ . The most-significant bit should appear first, ending with the least-significant bit. If you are simulating an ALU, group the buses together as just described.
  - Not every row of your voltage table must be annotated in the waveform simulation, but your choices of rows that you annotate must be encompassing
  - o If you are designing a state machine or a controller, your CLK signal should appear at the top of your inputs and outputs. The general order is CLK -> Reset -> state bits -> inputs -> outputs.
  - o *Tip*: Use Microsoft Paint to annotate your waveforms. An alternative is to print out the waveforms, annotate them by hand, then scan and upload
  - O Hint: Consider a truth table where the output is true in significantly less cases than it is false (or vice versa). If the output signal is active-high, it would be wise to annotate only the cases where the output voltage is HIGH.
- Include every line of VHDL programs, including both *architecture* and *behavior* sections.
- Include every line of any MIF files. If these are associated with assembly language programs, you can either put the assembly code as comments or separately include assembly language programs

Page 6/13

#### Lab 4 Report: ALU and RALU

inst15

PI Name: Jaiden Magnan March 13, 2025

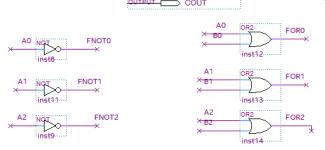
Poche, Natalie

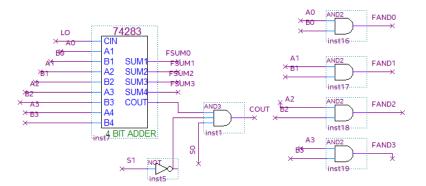
Class #: 11198

# 1. Arithmetic Logic Unit (ALU) Lab 4 Part A Name: Natalie Poche Class #: 11198 Pl Name: Jaiden Magnan

inst10







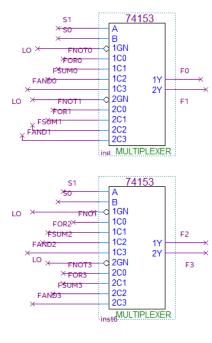


Figure 1: Lab 4 Part 1.1 - BDF of ALU with specifications

Maste	r Time Bar: 0	ps		•	Pointer: 0 ps		Interv	ıl: O ps		Start: 0 ps		End	0 ps	
	Name	Valu	0 ps 100	0 ns 200	0 ns 300	0 ns 400	00 ns 500	0 ns 600	,0 ns 700,	0 ns 800	O ns 900	00 ns 1.	us 1.1	us 1.2 us
<u> </u>	▼ S	B 00	00	01	10	11	00	01	10	11	00	01	_10	11
in	-	B 0	compliment	A or B			complimentA	A or B			complimentA			
in	5[0]	BO	of A	7.01.0	SUM A+B+Cin,	A AND 6		A OI U	SUM A+B+Cin,	A AND 6		7,010	SUM A+B+Cin,	A AND B
<u></u>	▼ A		0000000100100011	010(010)(011(0111		1100110101110(111	000000000000100013	010(0103011(0111	100(100)(101(101)	11001101/1110111	00000000000000001	010001030110011		1100(1101(1110)1111
in	A[3]		0 0 0 0			0 0 0 0	0000			0 0 0		0 0 0 0	1 1 1 1	0 0 0 0
in	A[2]	во	0 0 0 0	9 9 9 9	0 0 0 0		0000	0 0 0 0	0000		0 0 0	9 9 9 9	0 0 0	
in	A[1]	во	0 0	0 6 0	0 0 1 1	0 0 1 0	0011	0 0 0	0 0 1 1	0 0 1 0	0011	0 6 0	0011	0 0 1 0
in	A[0]	во	0 1 0 1	0 0 0	0 1 0 1	0 1 0 1	0 1 0 1		0 1 0 1	0 1 0 1	0 1 0 1	0 0 0	0 1	0 1 0 1
-	▼ B	B 1111	111300000000000000000000000000000000000	0017010(01070110	011100000000000000000000000000000000000	101 110(110 1110	111 000 000 001	001*\010(\010*\0110	011100000000000000000000000000000000000	101 X 100 10 X 1110	(111)000(00)001	001/010(010)(010)	011 X100(X100 X101(	101 (100)110 (1110
in	B[3]	B 1		0 0 0 0	0 1 1 1	0 7 00		0 0 0 0	0 1 1 1	0 7 0 0		0 0 0 0	0 1 1 1	0 0 0 0
in_	B[2]	B 1		0 9 0 0	1000	0 0 0		0 0 0	1000	0 0 0		0 0 0	1000	0 0 0
in_	B[1]	B 1		000	, 00,	1000		9000	, 00,	1000		000	, 00,	1000
in_	B[0]	B 1			7070	1010			1010	, 0 , 0		0 0 0	, 0, 0	ه ۱ ه ۱
*	▼ F	B 1111	(11)(11)(110)(10)(10)	01130103( 0111	111 000 001 0101	1000 1100 (110	111 (111 (110 (110 (110 (	01110010 0111	(111)()00)()01)()10)	1000 1100 (1110	(111)(111)(110)(110)	01110101 0111	111 000 001 0101	1000 1100 1110
<u>sut</u>	F[3]	B 1	1 1 1 1	0 0 0 0	. 000	1 1 1 1	1 1 1 1	0 0 0 0	. 0 0 0	1 1 1 1	1 1 1 1	0 0 0 0	. 0 0 0	1 1 1 1
out	F[2]	B 1	1 1 1 1	1 1 1 1	1001	0 1 1 1	1 1 1 1	1 1 1 1	1001	0 1 1 1	1 1 1 1	1 1 1 1	1 0 0 1	0 1 1 1
out	F[1]	B 1	1 0 0	1011	1010	0 0 0 1	1100	1011	1010	0 0 0 1	1100	1011	1010	0001
out	F[0]	B 1	1010	1 1 1 1	1 1 1 1	0000	1010	1 1 1 1	1 1 1 1	0000	1010	1 1 1 1	1 1 1 1	0000
out	COUT	B 0			<b>D O O</b>				<b>D</b> 0 0				TD 0 0	
*	FNOT	B 1111			011/011(010)(0100			101 (101 (100 (1000	01170110010100	00170010007000			)11%)11 <u>%)10</u> %100	001*\01(\000*\000
*	FOR	B 1111				111 110 111		01130103 0111		1117(110) 1111	X)001X 0011 (			111 10 1111
*	FSUM	B 1111							(111)000)001(010)					
*	▶ FAND	B 0000	0000 0010	0000 0100 X110	0000 <u>1000</u> 0010	1000X 1100 X1110	0000 0010	0100 0110	0000 <u>1000</u> 0010	1000 1100 XI110	0000 0010	0100 X)110	0000 1000 XIO10	1000 1100 XI110
4			000 → 111 011 ← 1000 101 ← 0100 0011 ← 1100		A:1000 B: <sub>0</sub> 011 1 TTT A:1001 B:1000	A: 1010 B: 1001 (1) 0 011 A: 1011 B: 1010	← 0000         ←		A:1000 B: <sub>0</sub> 011 1 1111 A:1001 B:1000	A: 1010 B: 1001 (1) 0 011 A: 1011 B: 1010	← 0000   0    ←   000   10    ← 0  00   00    ←   100		A:1000 B: <sub>011</sub> (111 A:1001 B:1000	A: 1010 8: 1001 (1) 0 011 A: 1011 B: 1010
4		•	1		(1)0 00 (	(0010)			(1) 0 0 0 (1)	(0010)			(I) 0 00 (	(0010)

Figure 2: Lab 4 Part 1.2 - Functional Simulation of ALU BDF

Page 7/13

Department of Electrical & Computer Engineering

Revision 0

Lab 4 Report: ALU and RALU PI Name: Jaiden Magnan

March 13, 2025

Poche, Natalie

Class #: 11198

## 2. Registered ALU

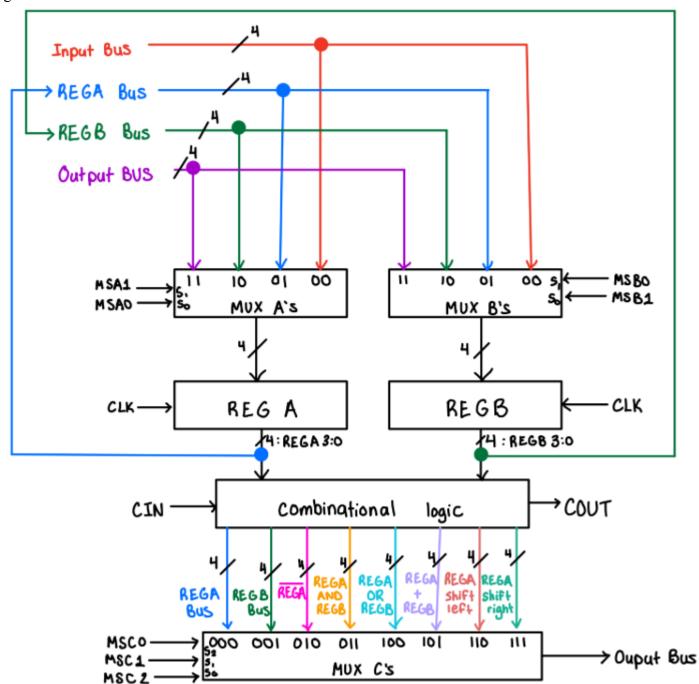


Figure 3: Lab 4 Part 2.1 - Functional Block Diagram of RALU

Department of Electrical & Computer Engineering

Revision 0

Page 8/13 Lab 4 Report: ALU and RALU

Class #: 11198 PI Name: Jaiden Magnan

Poche, Natalie

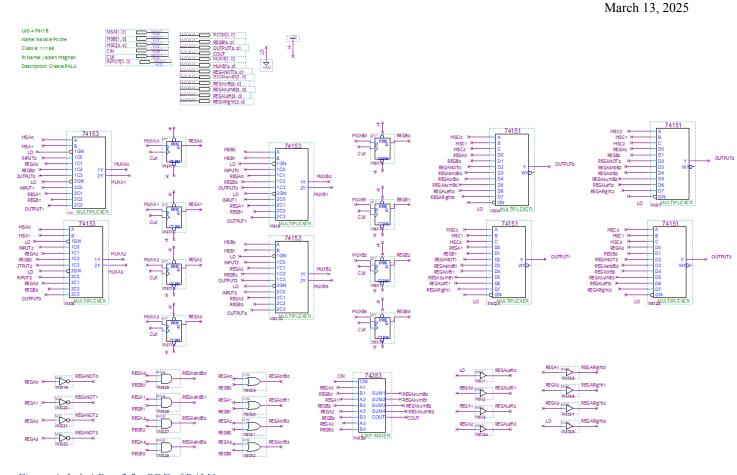


Figure 4: Lab 4 Part 2.2 - BDF of RALU

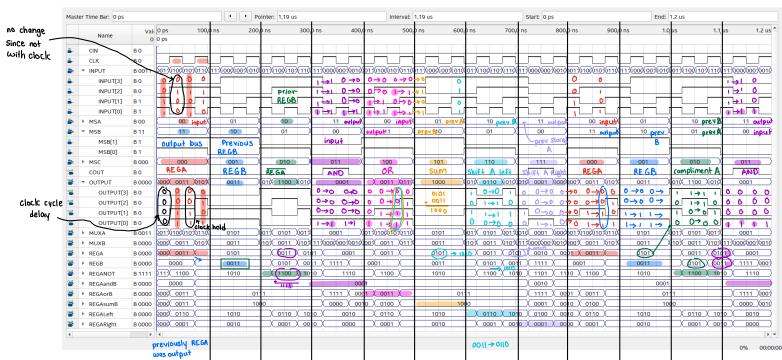


Figure 5: Lab 4 Part 2.3 - Functional Simulation of RALU

Page 9/13

Department of Electrical & Computer Engineering

Revision 0

Lab 4 Report: ALU and RALU

Class #: 11198 PI Name: Jaiden Magnan

Poche, Natalie

March 13, 2025

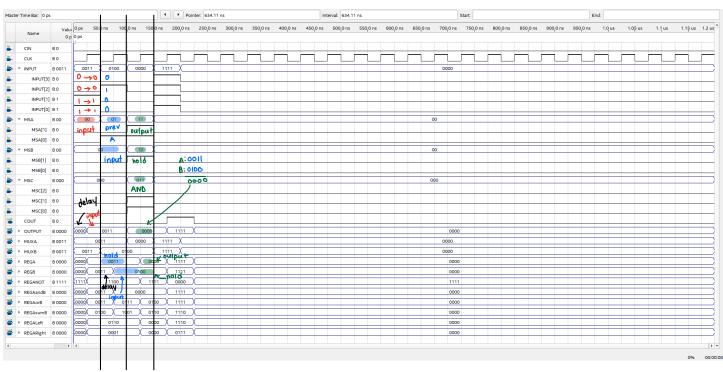


Figure 6: Lab 4 Part 2.4.a - Bit Wise AND store result in A and preserve contents in B

MSA	MSB	MSC	Input	$C_{IN}$	RegA	RegB	Output	RegA <sup>+</sup>	RegB <sup>+</sup>	Output <sup>+</sup>	$C_{out}^+$	Description
00	XX	XXX	0011	0	XXXX	XXXX	XXXX	0011	XXXX	XXXX	0	A <- 3
01	00	XXX	0100	0	0011	XXXX	XXXX	0011	0100	XXXX	0	B <- 4
11	10	011	XXXX	0	0011	0100	0000	0000	0100	0100	0	A <- A
												AND B

aster	Time Bar	: 0 ps				Pointer	178.46 ns		Interval: 178.46	ns	Start			End:		
	Name	Value at		100	) ns	200 <sub>1</sub> 0 ns	300 <sub>i</sub> 0 ns	400 <sub>t</sub> 0 ns	500 <sub>i</sub> 0 ns	600 <sub>1</sub> 0 ns	700 <sub>1</sub> 0 ns	800 <sub>i</sub> 0 ns	900 <sub>t</sub> 0 ns	1.Q us	1.1, us	1.2 us
	rume	0 ps	0 ps													
-		B 0														
		B 0		$\vdash$												Щ
•	INPUT		0011	0100	0000	1111 X					0000					++-
-	IN		0->0													
-	IN		0 > 0	1 → 1												
	IN		1 -> 1	0 -> 0												
	IN		1 → 1	0 + 0												
*	MSA		0	input	hold						00					
	MS			Input	110.0											
	MS		00													
*		B 00		10	<u> </u>	A: 0100					00					
	MS		input	hold	output	B: 0011										
	MS		0		400	0111					000					
Ť	MSC MS		0	ľ	100 )	Ojtt					000					
	MS				OK											
	MS															
	COUT															
	OUTP		2000	11 ()100	0111	10¢ 1111 X					0000					
, }	MIIYA	B 0000	CKRAN	mout	00	1111 X					0000					
·	MUXB	в 0011 <b>с\</b>	N 160	lingut	0111	111110					0000					
			0000000		0100	Loutbut					0000					
			000C	0011	X 01	/ Output					0000					
	REGA		1113 11		1011	X 0000 X					1111					
	REGA				000 X 01						0000					
-	REGA		000000		0111	X 1111 X					0000					
	REGA				01 \( 10	11 X 1110 X					0000					
	REGA		0000 01		1000	X 1110 X					0000					
	REGA		00000		0010	X 0111 X					0000					
		<b></b>	4													-

Figure 8: Lab 4 Part 2.4.b - Bit Wise OR, store result in B, preserve in A

University of Florida

#### **EEL 3701C: Digital Logic & Computer Systems**

Department of Electrical & Computer Engineering

Revision 0

PI Name: Jaiden Magnan

Poche, Natalie

Class #: 11198

March 13, 2025

Page 10/13

#### Lab 4 Report: **ALU and RALU**

Output Output<sup>+</sup> Description MSA MSB MSC Input  $C_{IN}$ RegA RegB RegA<sup>+</sup> RegB<sup>+</sup>  $C_{out}^+$ XX 00 XXX 0011 0 XXXX XXXX XXXX XXXX 0011 XXXX B <- 3 00 10 XXX0100 0 XXXX0011 XXXX0100 0011 XXXX0 A < -4B<-3 0100 01 1 100 XXXX0011 0111 0100 0111 1011  $A \leq -A OR$ В

Figure 9: Lab 4 Part 2.4.b - Bit Wise OR, store result in B, preserve in A

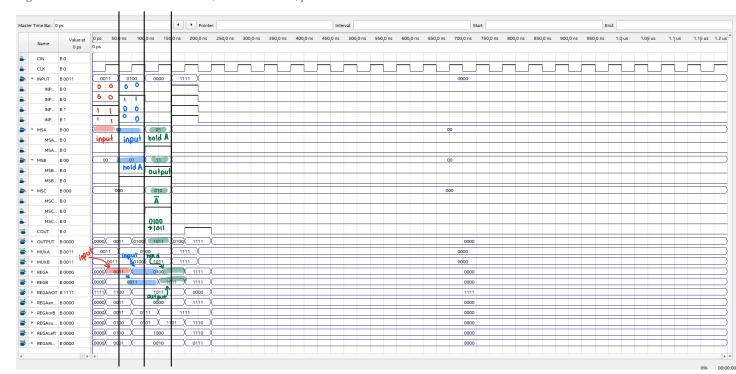


Figure 10: Lab 4 Part 2.4.c - Compliment A, store in B, preserve A

MSA	MSB	MSC	Input	$C_{IN}$	RegA	RegB	Output	RegA <sup>+</sup>	$RegB^+$	Output <sup>+</sup>	$C_{out}^+$	Description
00	XX	XXX	0011	0	XXXX	XXXX	XXXX	0011	XXXX	XXXX	0	A <- 3
00	01	XXX	0100	0	0011	XXXX	XXXX	0100	0011	XXXX	0	A<-4, B<-
												3
01	11	010	XXXX	0	0100	0011	1011	0100	1011	1111	0	A <- /A

Figure 11: Lab 4 Part 2.4.c - Compliment A, store in B, preserve A

Page 11/13

Department of Electrical & Computer Engineering

Revision 0

Lab 4 Report: ALU and RALU

Class #: 11198 PI Name: Jaiden Magnan

March 13, 2025

Poche, Natalie

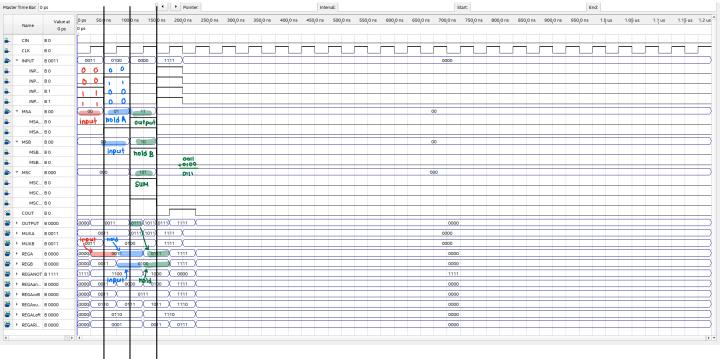


Figure 12: Lab 4 Part 2.4.d - SUM A and B, store A, preserve B

MSA	MSB	MSC	Input	$C_{IN}$	RegA	RegB	Output	RegA <sup>+</sup>	RegB <sup>+</sup>	Output <sup>+</sup>	$C_{out}^+$	Description
00	XX	XXX	0011	0	XXXX	XXXX	XXXX	0011	XXXX	XXXX	0	A <- 3
01	00	XXX	0100	0	0011	XXXX	XXXX	0011	0100	XXXX	0	B <- 4
11	10	101	XXXX	0	0011	0100	0111	0111	0100	1011	0	A <- A+B

Figure 13: Lab 4 Part 2.4.d - SUM A and B, store A, preserve B

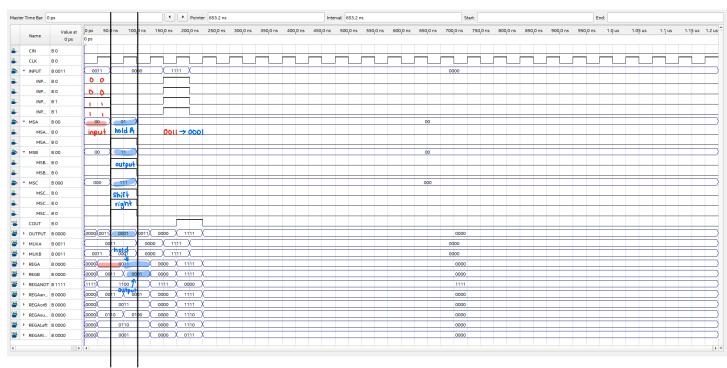


Figure 14: Lab 4 Part 2.4.e - Shift A right one bit, store in B

PI Name: Jaiden Magnan

Page 12/13

Lab 4 Report: **ALU and RALU** 

March 13, 2025

Poche, Natalie

Class #: 11198

MSA	MSB	MSC	Input	$C_{IN}$	RegA	RegB	Output	RegA <sup>+</sup>	RegB <sup>+</sup>	Output <sup>+</sup>	$C_{out}^+$	Description
<b>-00</b>	XX	XXX	0011	0	XXXX	XXXX	XXXX	0011	XXX	XXXX	0	A <- 3
01	-11	<b>11</b> 1	XXXX	0	0011	XXXX	0001	-0011	0001	0100	0	Shift A
												right, store
												in B

Figure 15: Lab 4 Part 2.4.e - Shift A right one bit, store in B

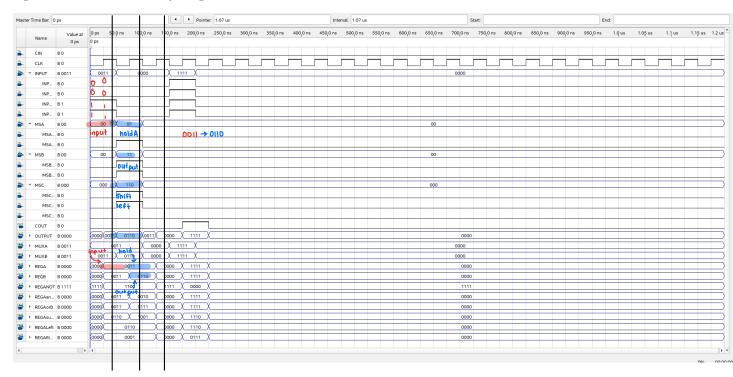


Figure 16: Figure 14: Lab 4 Part 2.4.f - Shift A left one bit, store in B

MSA	MSB	MSC	Input	$C_{IN}$	RegA	RegB	Output	RegA <sup>+</sup>	RegB <sup>+</sup>	Output <sup>+</sup>	$C_{out}^+$	Description
00	XX	XXX	0011	0	XXXX	XXXX	XXXX	0011	XXXX	XXXX	0	A <- 3
-01	<del>1</del> 1	1100	XXXX	0	0011	XXXX	0110	0011	1000	1011	0	A <- Shift
												A left 1 bit

Figure 17: Lab 4 Part 2.4.f - Shift A left one bit, store in B

University of Florida

#### **EEL 3701C: Digital Logic & Computer Systems**

Department of Electrical & Computer Engineering

ВО

INPUT[3] B 0

INPUT[2] B 0 INPUT[1] B 1 INPUT[0] B 1

MSA[1] B 0 MSA[0] B 0 SB B 00 MSB[1] B 0

MSB[0] B 0 6C B 00 MSC[2] B 0

MSC[0]
COUT
OUTPUT

► MUXA

▶ MUXB

▶ REGA

▶ REGB

► REGAandB

▶ REGAsumB

▶ REGALeft

▶ REGARight B 00 0000

B 00 0000

B 00 0000

в оо ооо оо

B 00 0000 00

B 00 0000 00

B 00 0000 01

B 00 0000

B 00

B 00

B 00 0011

0 0

1100 X 0101

hold A

input input nold B

0110

0001

Sum

1110100

11

0101

01

0 X 11 1 X 0100 X 10 0 X 11 1 X 11

0 X 01

X 01 1 X 00 0 X 0101 X 001

X 1110 X 10

11 11 0101 11 00 11 1111 Revision 0

Page 13/13

Name

▼ INPUT

Lab 4 Report: ALU and RALU

600<sub>1</sub>0 ns

0000 - 1000

700<sub>1</sub>0 ns

0000

0000

0000

0000

0000

400,

hold B

07 11

001/110

1110

1110

1110

0 1100 00

0 ( 011

1010

0 1000 101

0100 (1000

1010

Class #: 11198 PI Name: Jaiden Magnan

Poche, Natalie

Figure 18: Lab 4 Part 2.4.g - Add \$3 and \$C, AND \$5 to result, multiply by 4, OR \$A to result, compliment new result, and divide by 2

1010 X

MSA	MSB	MSC	Input	$C_{IN}$	RegA	RegB	Output	RegA <sup>+</sup>	RegB <sup>+</sup>	Output <sup>+</sup>	$C_{out}^+$	Description
00	XX	XXX	0011	0	XXXX	XXXX	XXXX	0011	XXXX	XXXX	0	A <- 3
01	00	XXX	1100	0	0011	XXXX	XXXX	0011	1100	1111	0	B <- 12
11	00	101	0101	0	0011	1100	1111	1111	0101	0100	1	A <- A+B, set B
												to 5
11	10	011	XXXX	0	1111	0101	0101	0101	0101	1010	0	AND(F, 5)
11	10	110	XXXX	0	0101	0101	1010	1010	0101	1111	0	ShiftLeft one bit
11	00	110	1010	0	1010	0101	0100	0100	1010	1110	0	ShiftLeft one bit
11	10	100	XXXX	0	0100	1010	1110	1110	1010	1000	1	OR
110	10	010	XXXX	0	1110	1010	0001	0001	1010	1011	1	COMPLIMENT
11	10	111	XXXX	0	0001	1010	0000	0000	1010	1010	0	DIVIDE(2)

Figure 19: Lab 4 Part 2.4.g - Add \$3 and \$C, AND \$5 to result, multiply by 4, OR \$A to result, compliment new result, and divide by 2