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

Lab 4 Report: **ALU and RALU**

Poche, Natalie
Class #: 11198
PI Name: Jaiden Magnan
March 13, 2025

REQUIREMENTS NOT MET

N/A

VIDEO FILE LINK

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

PROBLEMS ENCOUNTERED

None, this just takes a long time.

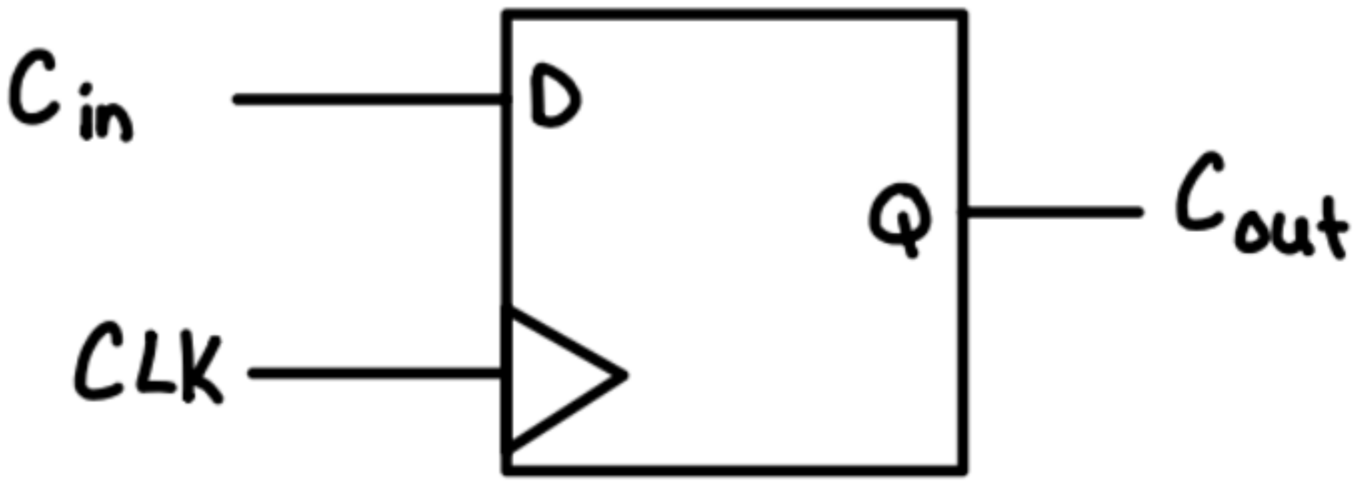
FUTURE WORK/APPLICATIONS

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.

Poche, Natalie
Class #: 11198
PI Name: Jaiden Magnan
March 13, 2025

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.

EEL 3701C: Digital Logic & Computer Systems
Revision **0**
Lab 4 Report: **ALU and RALU**

Poche, Natalie
Class #: 11198
PI Name: Jaiden Magnan
March 13, 2025

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.**
    - Images should be large enough so that inputs, outputs, labels, and parts are clearly visible and distinguishable to any reader.
    - 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)

    - 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 https://www.draw.io/ to create computer-generated ASMs.
- Truth tables or next-state truth tables should have the following characteristics.
    - Can be either typed or hand-written and scanned (must be clear and legible)
    - Must be in **counting order** (i.e., inputs of 000, 001, 010, 011, …, 111)
    - Clearly distinguish inputs from outputs (see the example below that uses a thick line)
    - 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 $Q1^+$, respectively)
    - Tip: divide rows into consecutive groups of 4 (or 2 or 8) to make it easier for both you and your PI to read.
    - **Example**

A	B	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., “Table 3: Truth Table for Part B.”

- Voltage tables should have the following characteristics.
  - Must be in counting order (i.e., inputs of LLL, LLH, LHL, … , HHH)
  - Use similar formatting to truth tables (described above)

- **Example**

A(H)	B(H)	C(L)	Y(L)
L	L	L	H
L	L	H	H
L	H	L	L
L	H	H	H
H	L	L	H
H	L	H	H
H	H	L	H
H	H	H	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***
  - 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.
  - ***Tip***: Use Microsoft Paint to annotate your waveforms. An alternative is to print out the waveforms, annotate them by hand, then scan and upload
  - ***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

1. Arithmetic Logic Unit (ALU)

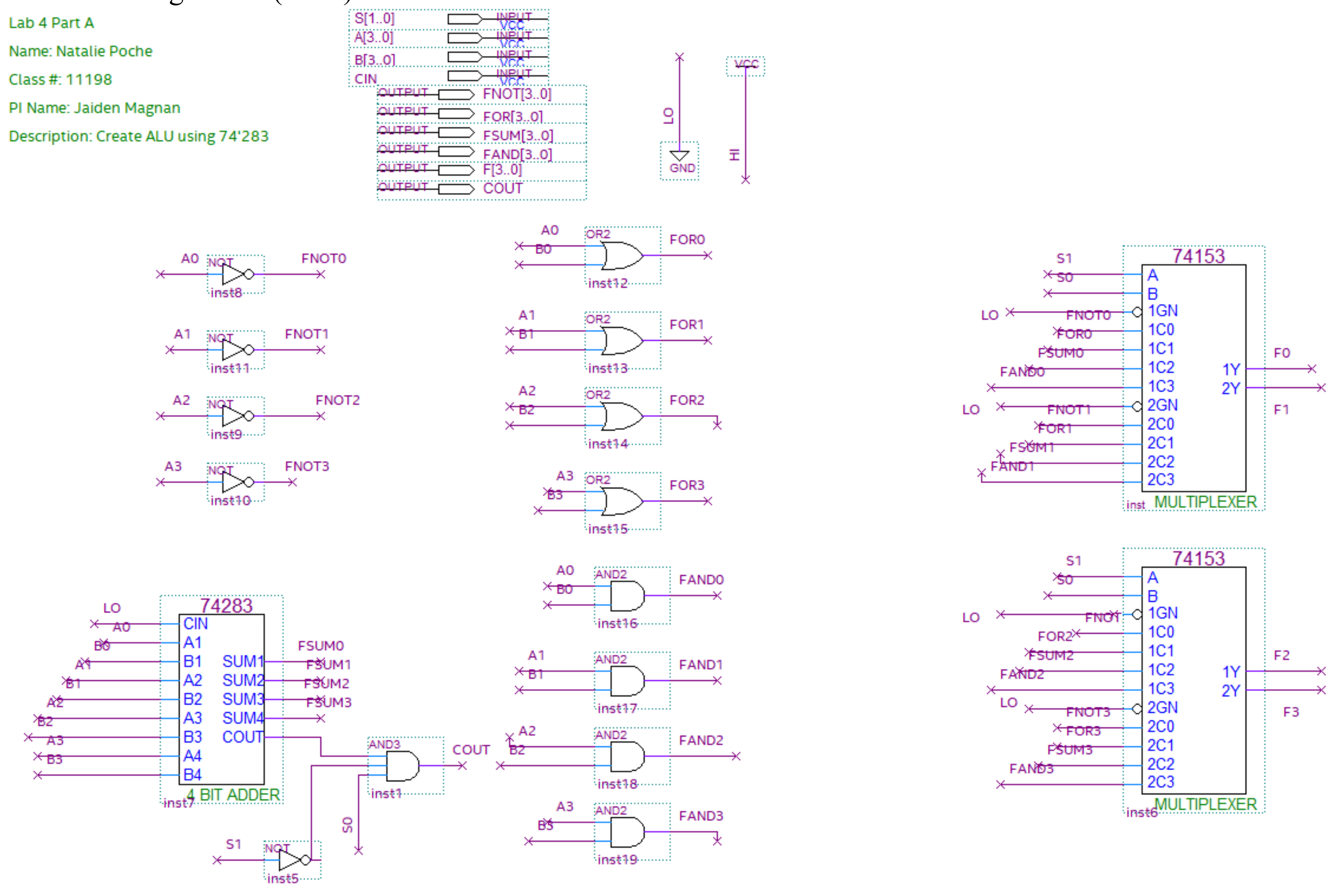


Figure 1: Lab 4 Part 1.1 - BDF of ALU with specifications

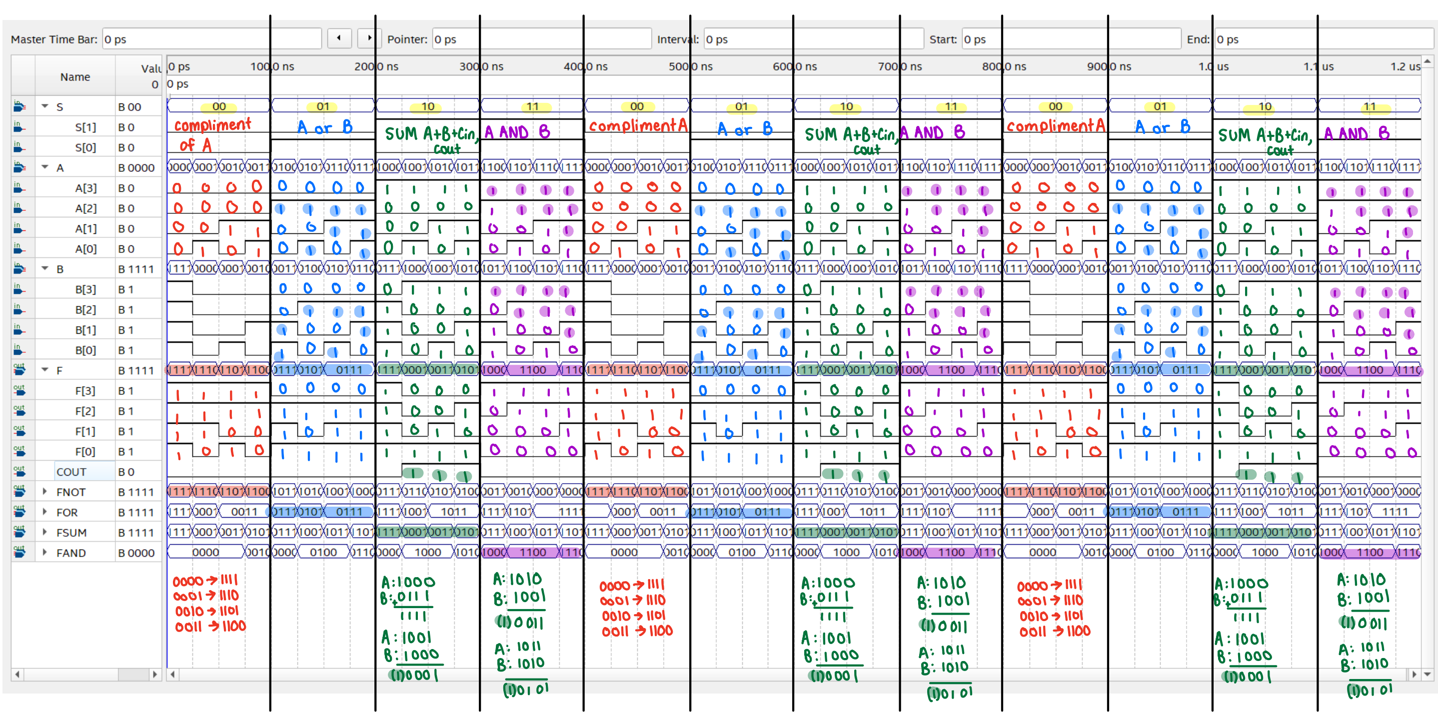


Figure 2: Lab 4 Part 1.2 - Functional Simulation of ALU BDF

2. Registered ALU

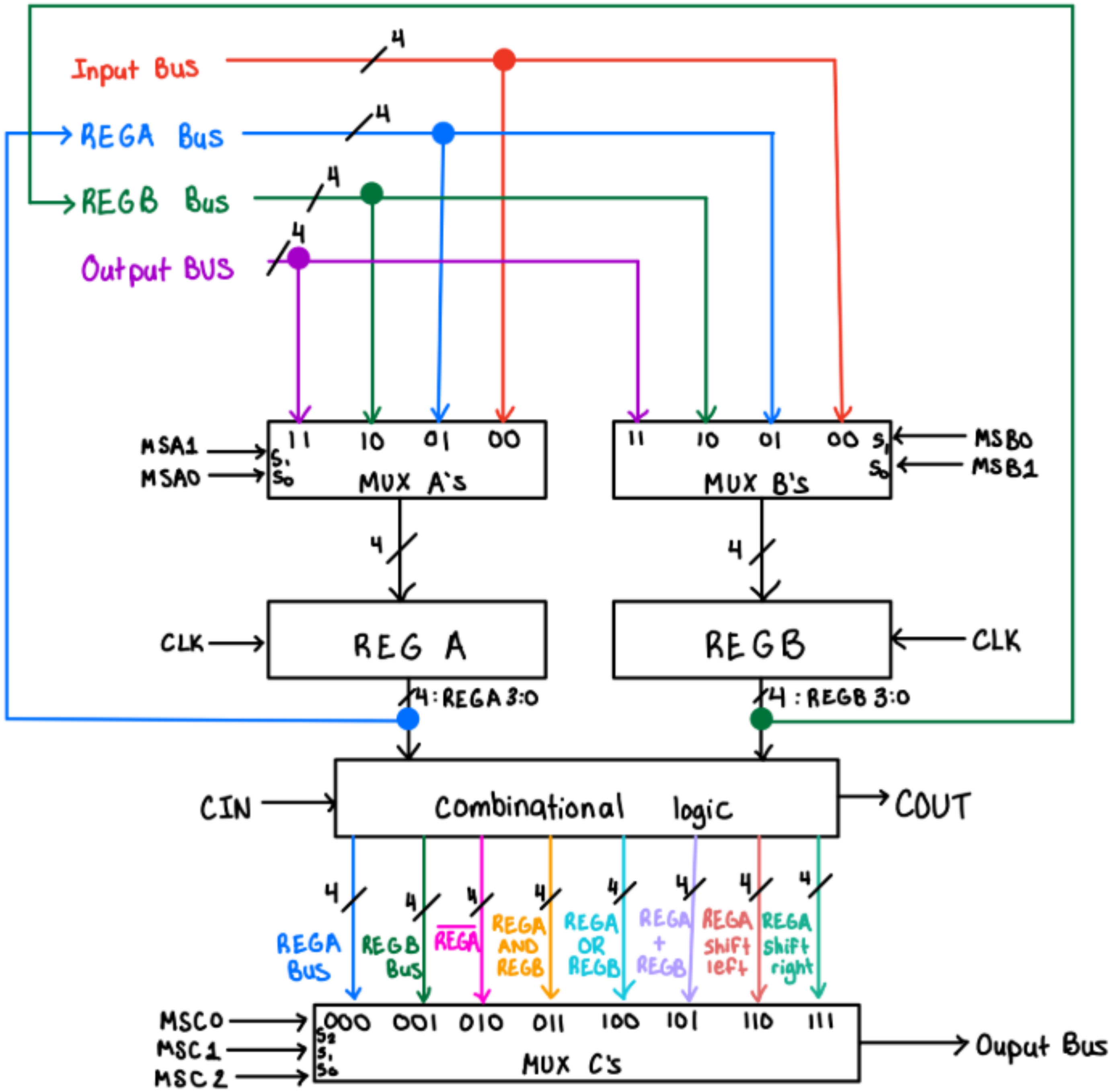


Figure 3: Lab 4 Part 2.1 - Functional Block Diagram of RALU

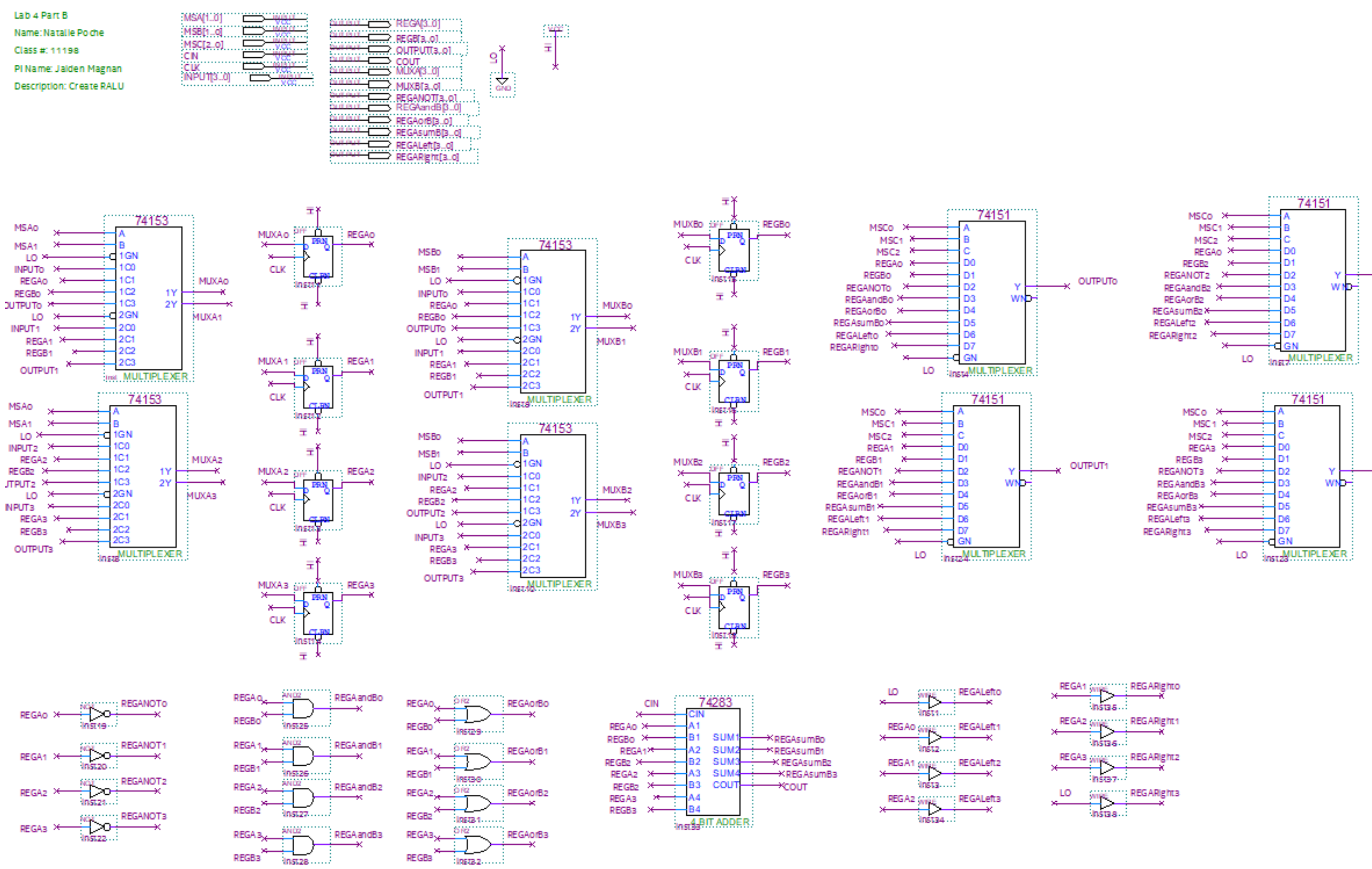


Figure 4: Lab 4 Part 2.2 - BDF of RALU

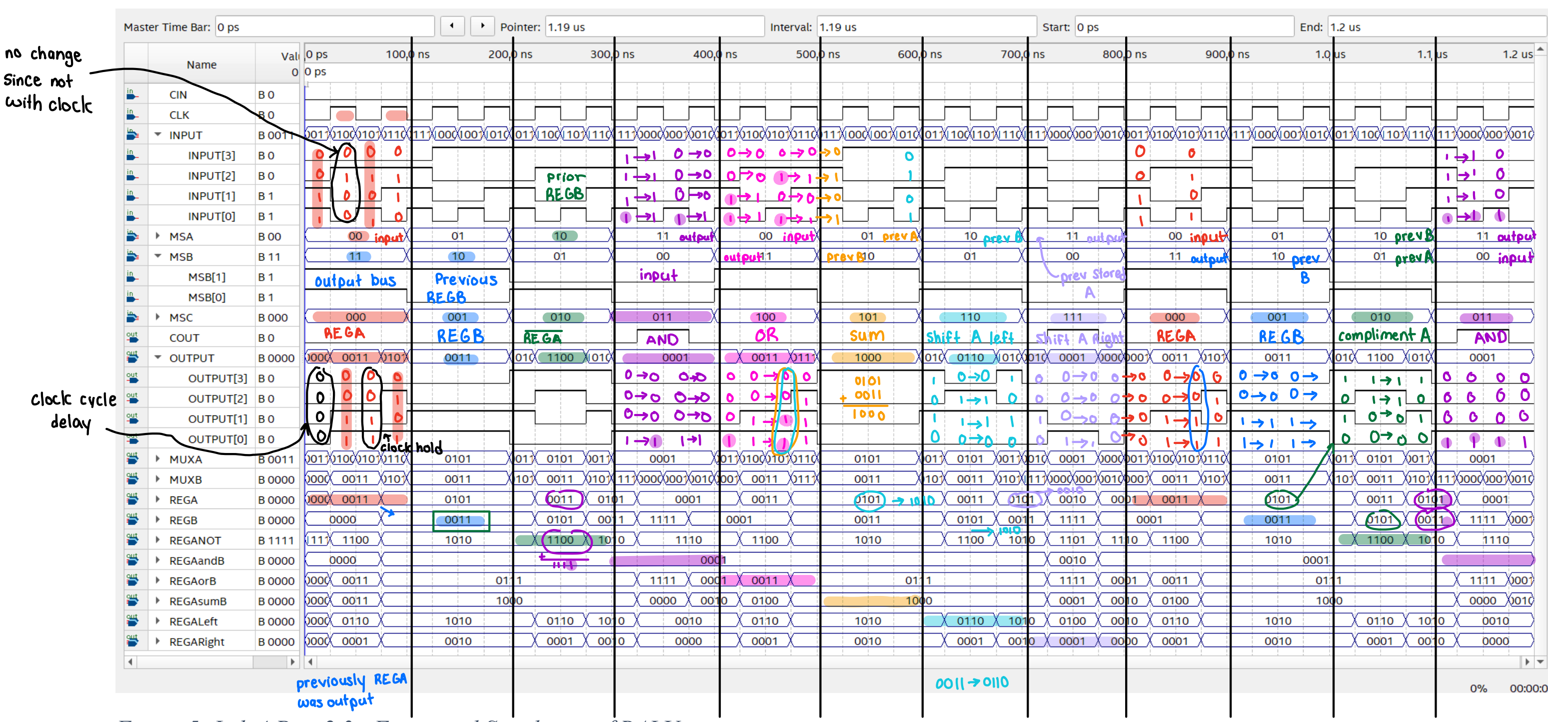


Figure 5: Lab 4 Part 2.3 - Functional Simulation of RALU

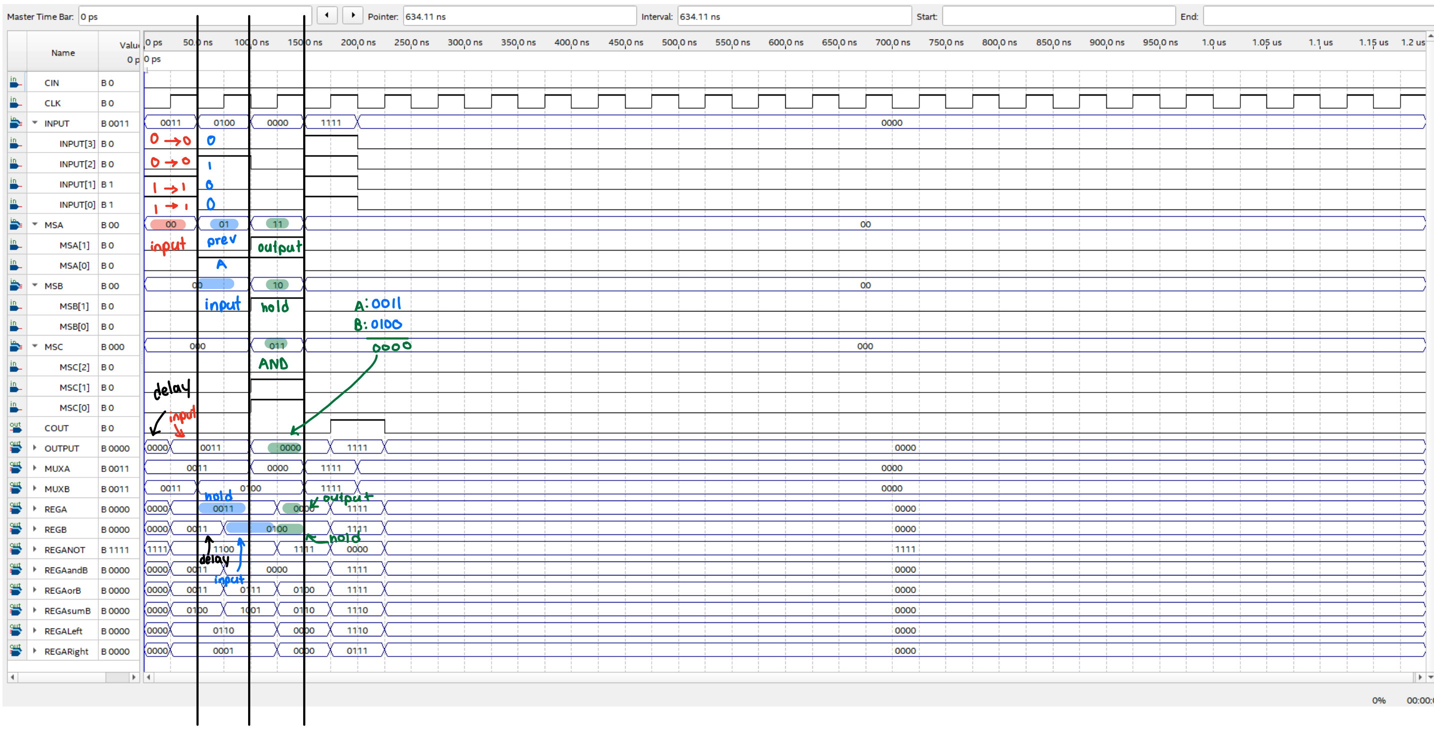


Figure 6: Lab 4 Part 2.4.a - Bit Wise AND store result in A and preserve contents in B

MSA	MSB	MSC	Input	C_{IN}	RegA	RegB	Output	$RegA^{+}$	$RegB^{+}$	$Output^{+}$	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

Figure 7: Lab 4 Part 2.4.a - Bit Wise AND store result in A and preserve contents in B

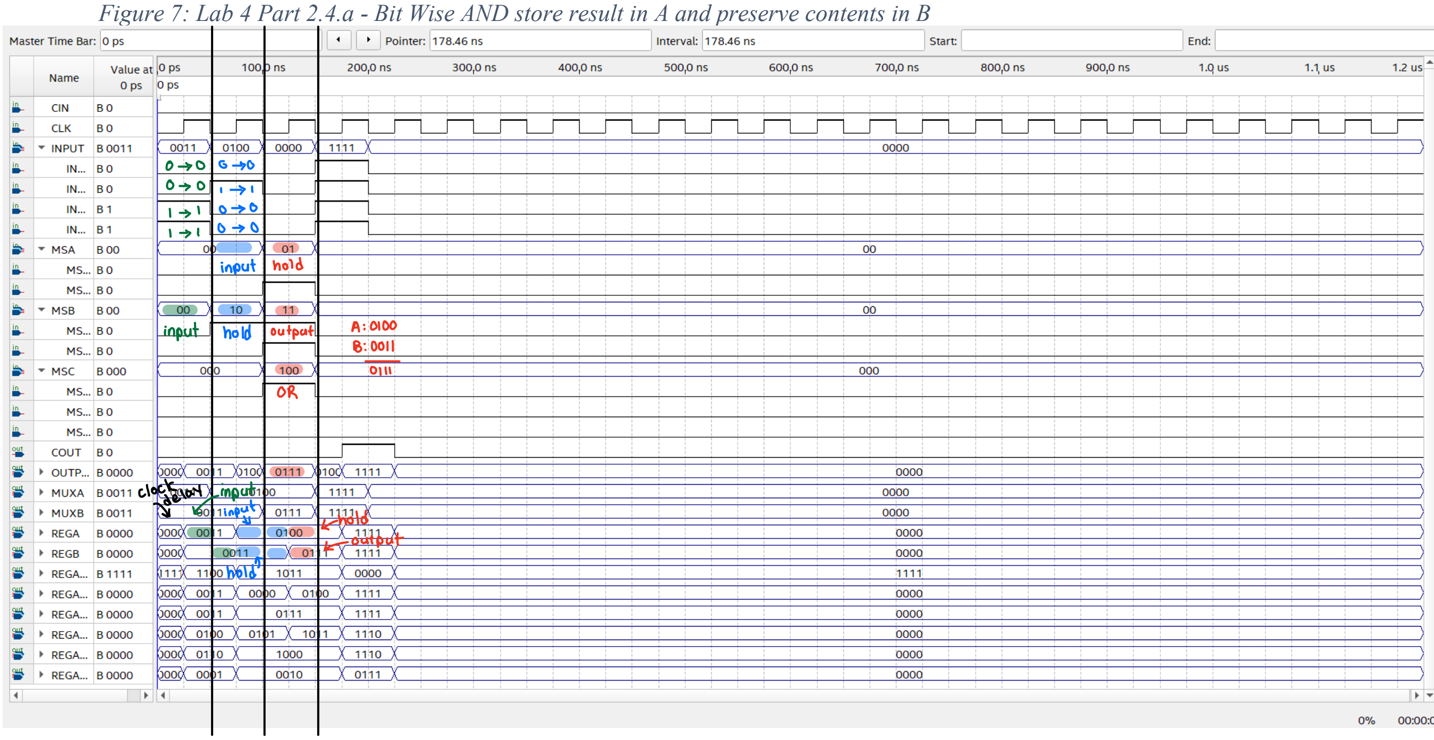


Figure 8: Lab 4 Part 2.4.b - Bit Wise OR, store result in B, preserve in A

MSA	MSB	MSC	Input	C_{IN}	RegA	RegB	Output	$RegA^{+}$	$RegB^{+}$	$Output^{+}$	C_{out}^{+}	Description
XX	00	XXX	0011	0	XXXX	XXXX	XXXX	XXXX	0011	XXXX	0	B <- 3
00	10	XXX	0100	0	XXXX	0011	XXXX	0100	0011	XXXX	0	A <- 4, B<-3
01	11	100	XXXX	0	0100	0011	0111	0100	0111	1011	0	A <- A OR B

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

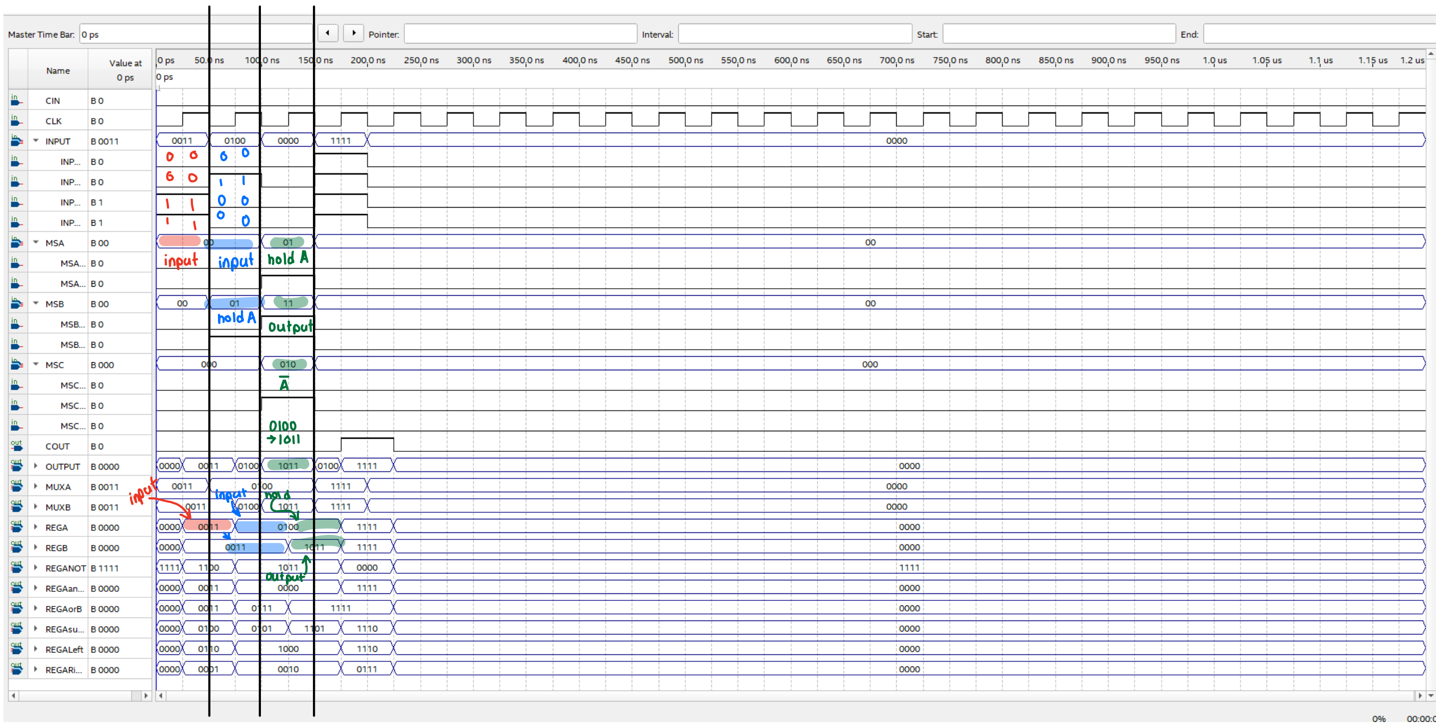


Figure 10: Lab 4 Part 2.4.c - Compliment A, store in B, preserve A

MSA	MSB	MSC	Input	C_{IN}	RegA	RegB	Output	$RegA^{+}$	$RegB^{+}$	$Output^{+}$	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

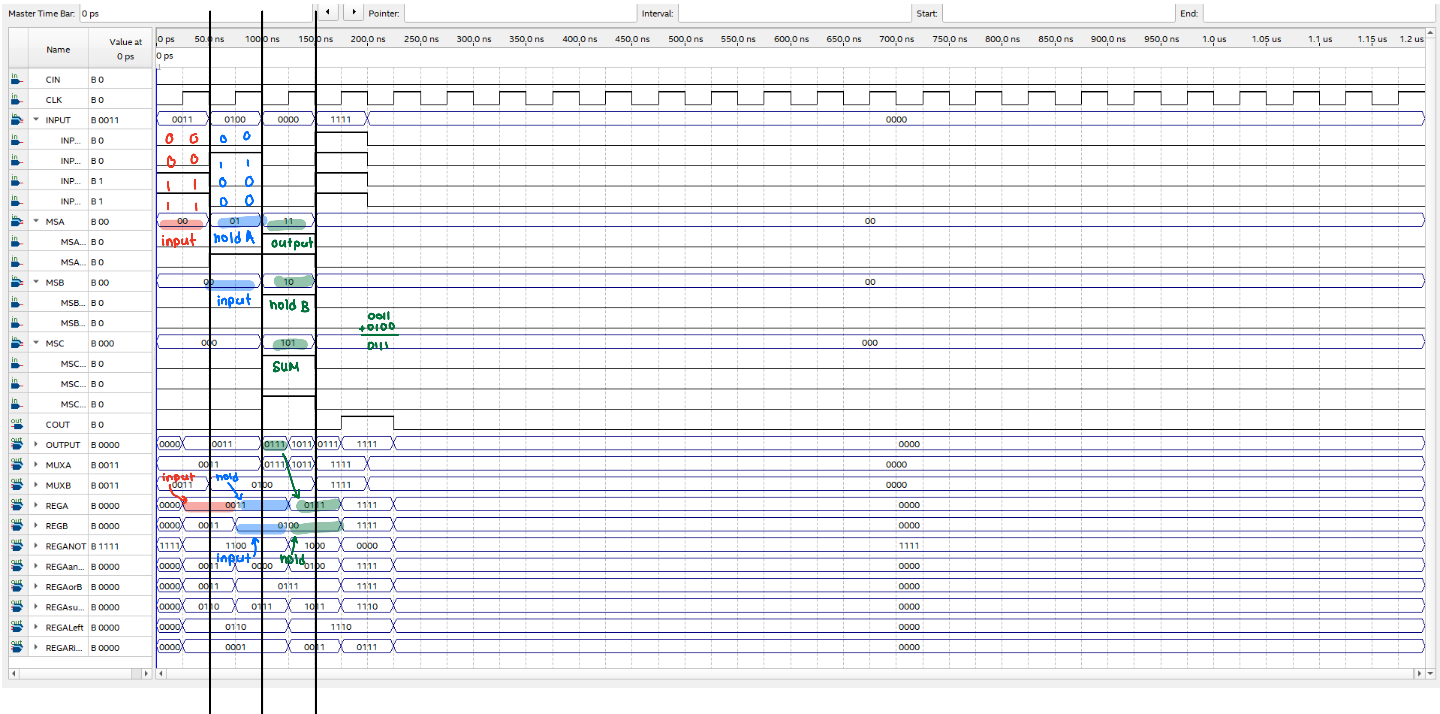


Figure 12: Lab 4 Part 2.4.d - SUM A and B, store A, preserve B

MSA	MSB	MSC	Input	C_{IN}	RegA	RegB	Output	$RegA^+$	$RegB^+$	$Output^+$	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

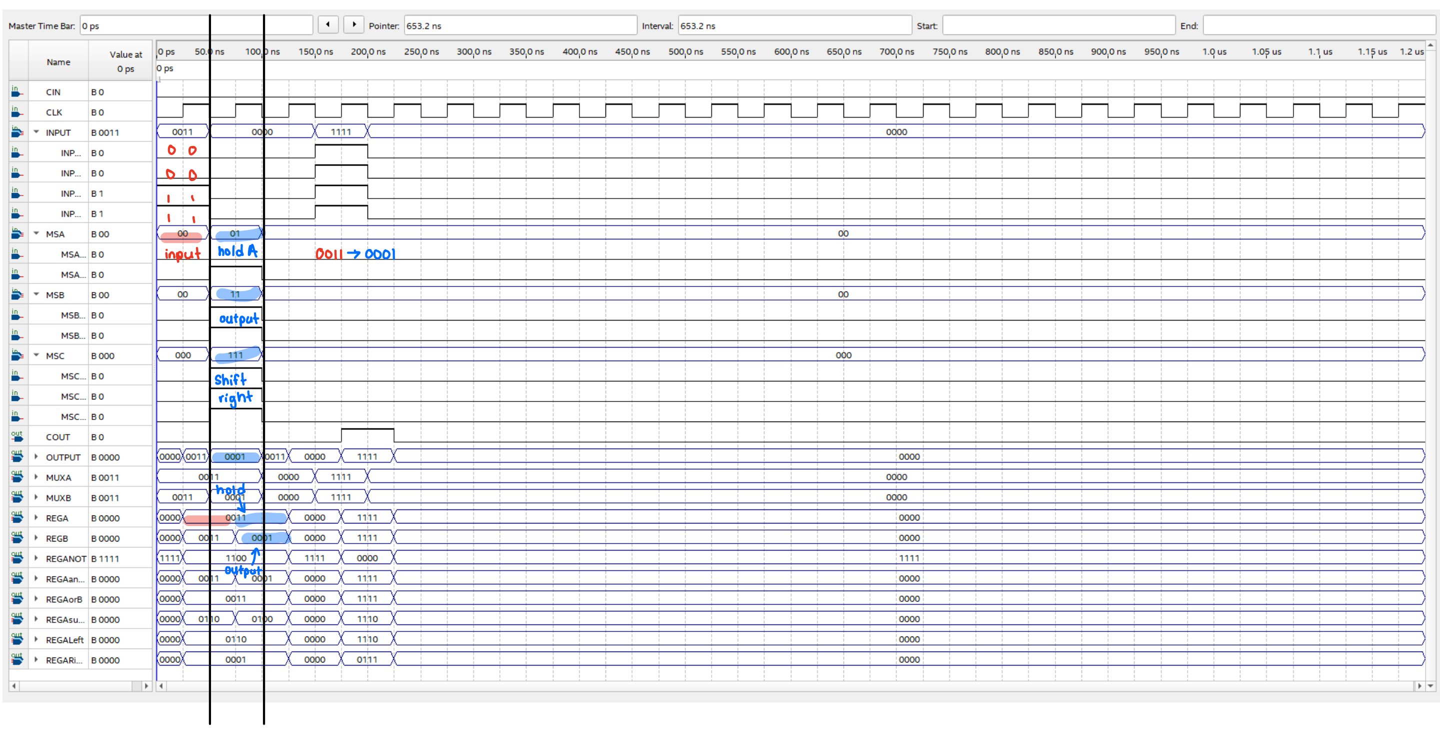


Figure 14: Lab 4 Part 2.4.e - Shift A right one bit, store in B

Poche, Natalie
Class #: 11198
PI Name: Jaiden Magnan
March 13, 2025

MSA	MSB	MSC	Input	C_{IN}	RegA	RegB	Output	RegA$^+$	RegB$^+$	Output$^+$	C_{out}^+	Description
00	XX	XXX	0011	0	XXXX	XXXX	XXXX	0011	XXX	XXXX	0	A <- 3
01	11	111	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

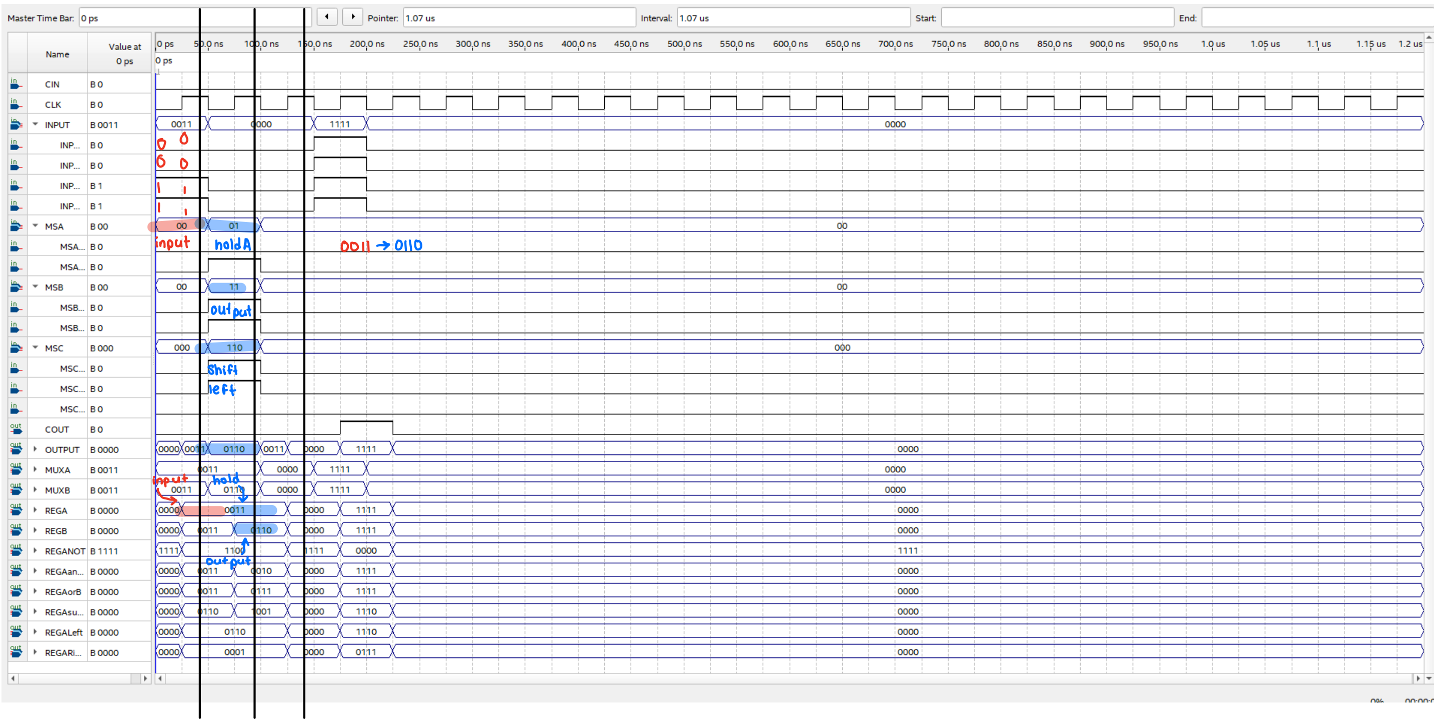


Figure 16: Figure 14: Lab 4 Part 2.4.f - Shift A left one bit, store in B

MSA	MSB	MSC	Input	C_{IN}	RegA	RegB	Output	RegA$^+$	RegB$^+$	Output$^+$	C_{out}^+	Description
00	XX	XXX	0011	0	XXXX	XXXX	XXXX	0011	XXXX	XXXX	0	A <- 3
01	11	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

EEL 3701C: Digital Logic & Computer Systems
Revision 0

Lab 4 Report: **ALU and RALU**

Poche, Natalie
Class #: 11198
PI Name: Jaiden Magnan
March 13, 2025

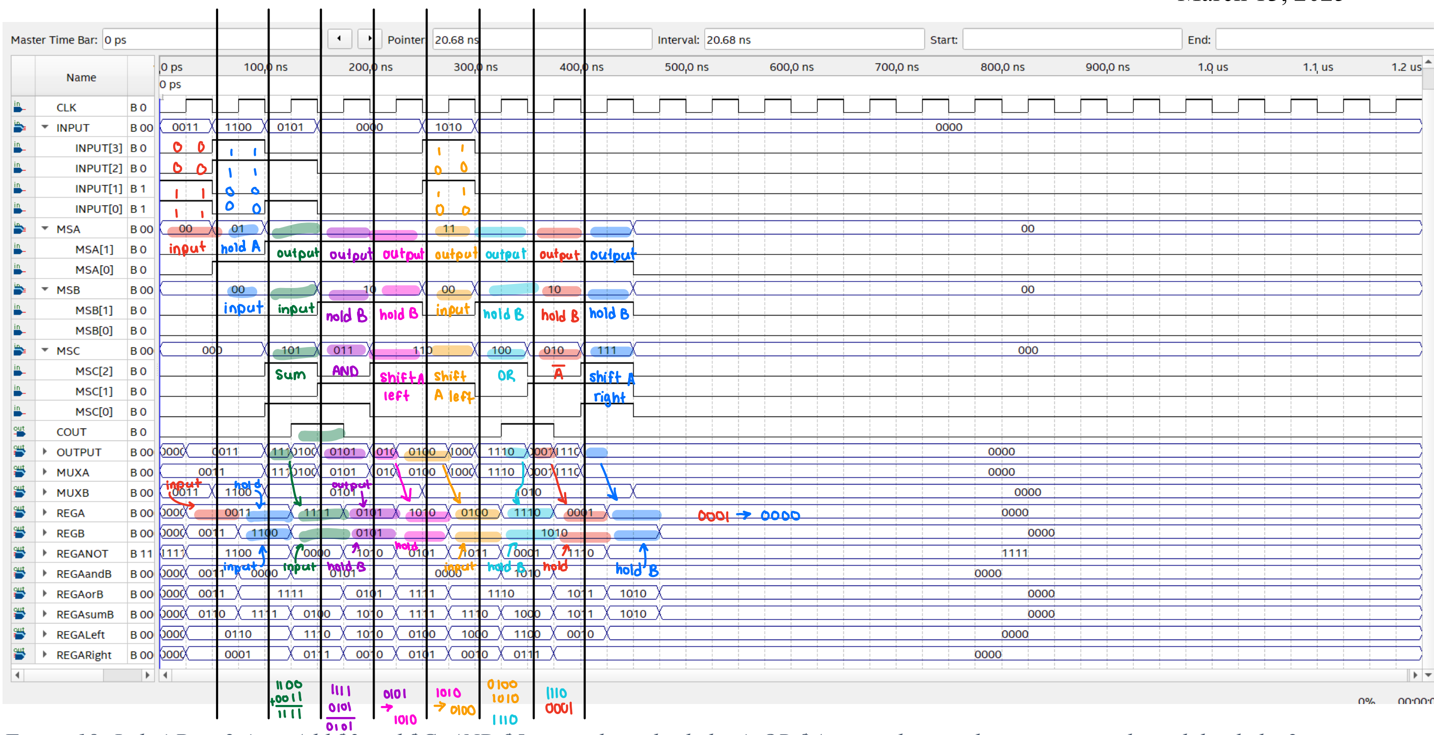


Figure 18: Lab 4 Part 2.4.g - Add \$3 and \$C, AND \$5 to result, multiply by 4, OR \$A to result, compliment new result, and divide by 2

MSA	MSB	MSC	Input	C_{IN}	RegA	RegB	Output	RegA$^+$	RegB$^+$	Output$^+$	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
11	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