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

Lab 4 Report: **ALU and RALU**

Stern, Arion
10844
Erick Zayas Ramos
July 11, 2025

REQUIREMENTS NOT MET

N/A

VIDEO FILE LINK

https://youtube.com/shorts/aZj2Mr-ewOI?feature=share

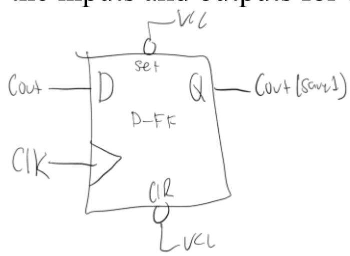
PROBLEMS ENCOUNTERED

I had a lot of problems debugging the RALU with Quartus simulations, but I had to keep tweaking the timing of each the inputs with the clock cycles until it worked. I was also confused about the DAD portion and spent a great amount of time getting that to work with the DE-10 7-segment display.

FUTURE WORK/APPLICATIONS

The RALU created in this lab serves as the foundation for future CPU designs. In Lab 6, we will integrate this RALU with a controller circuit to build a simple 4-bit CPU capable of executing programmed instructions. This design will later be expanded in Lab 7 into the 8-bit Gator CPU (G-CPU), enabling more advanced processing tasks and demonstrating how basic data path components scale into fully functional processors.

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. 

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

    a. No, dividing by two will not work correctly for all 4-bit 2's complement numbers because right shifts can introduce errors for negative numbers. In 2's complement, dividing by two requires arithmetic shifting (preserving the sign bit), but the RALU uses a logical shift, which fills the most significant bit with zero. This causes negative numbers to become incorrect, since the sign is not properly extended during the shift.

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.
    a. To take the 2's complement of a number in REGA and store it in REGB, first load REGA's value into the ALU and use the complement function to invert all bits. Then add 1 to the inverted value by doing an addition with Cin set to 1. Finally, store the result from the OUTPUT bus into REGB.

4. Describe how you subtract with your RALU. Hint: See the previous question.
    a. To subtract using the RALU, compute the 2's complement of the subtrahend as mentioned in the previous problem and then add it to the minuend.

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?
    a. If asynchronous CLR or SET is not available, you can clear the contents of REGA or REGB by loading the value 0 from the INPUT bus into the register. Set the INPUT bus to 0000, configure MSA or MSB to select the INPUT bus, and then clock the value into the desired register.

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

1. ARITHMETIC LOGIC UNIT (ALU):

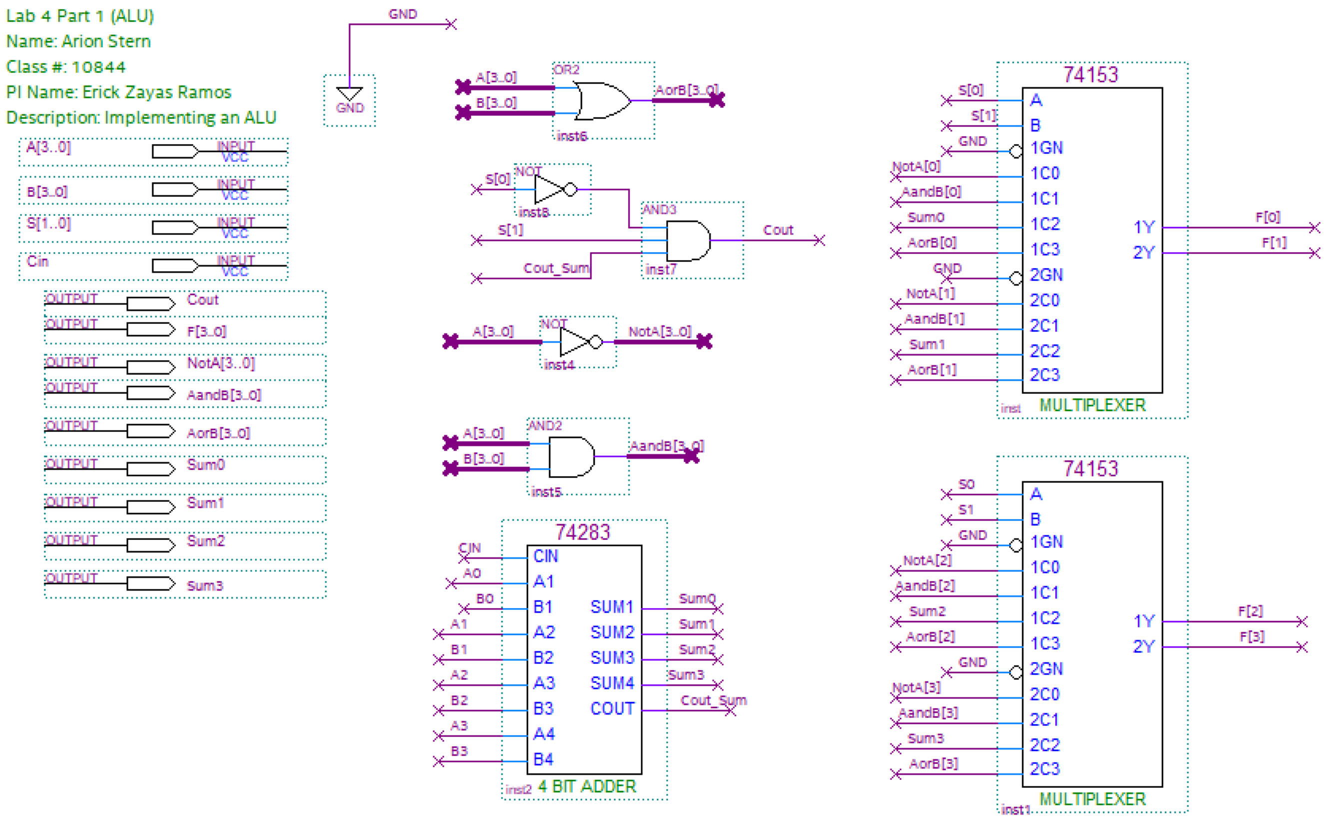
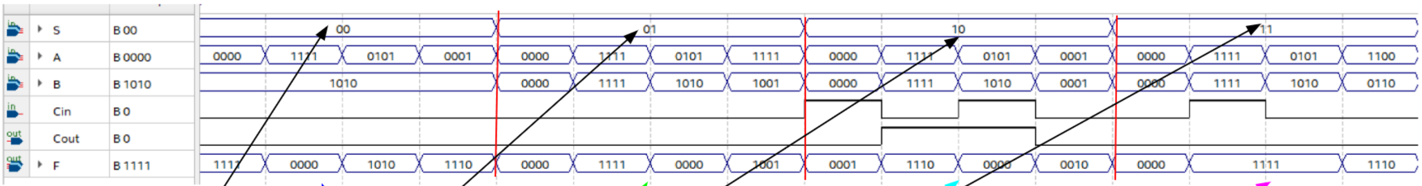


Figure 1: ALU Quartus Schematic (Part 1)



All of the actions work as expected when their select lines are picked.

S1:S0	Action	Equation
00	complement of A	F = /A
01	bit-wise AND	F = A and B
10	Sum (w/ Cin), Cout	F = A + B + Cin , Cout
11	bit-wise OR	F= A or B

Figure 2: ALU Annotated Simulation: (Part 1)

2. REGISTERED ALU:

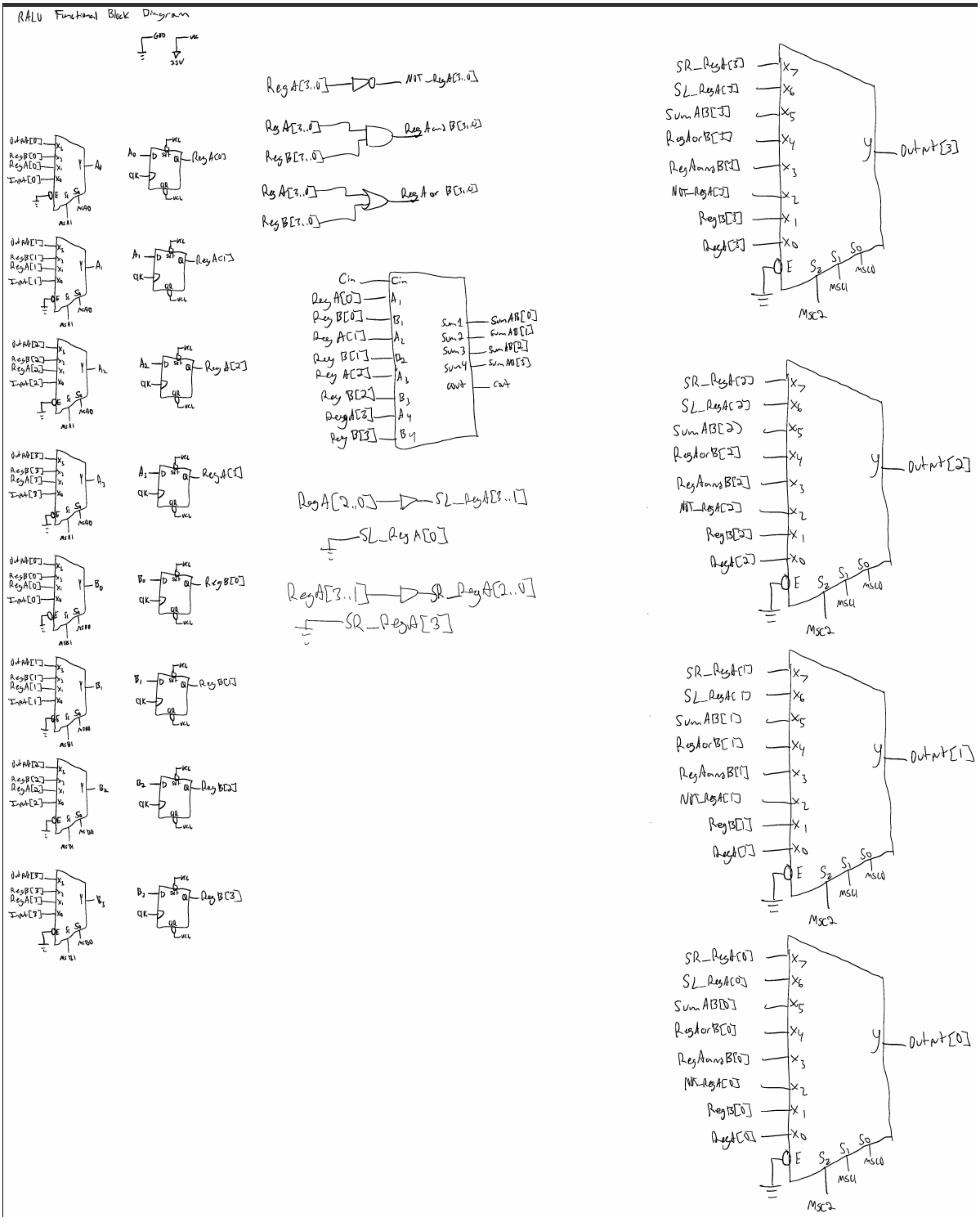


Figure 3: RALU Functional Block Diagram (Part 2)

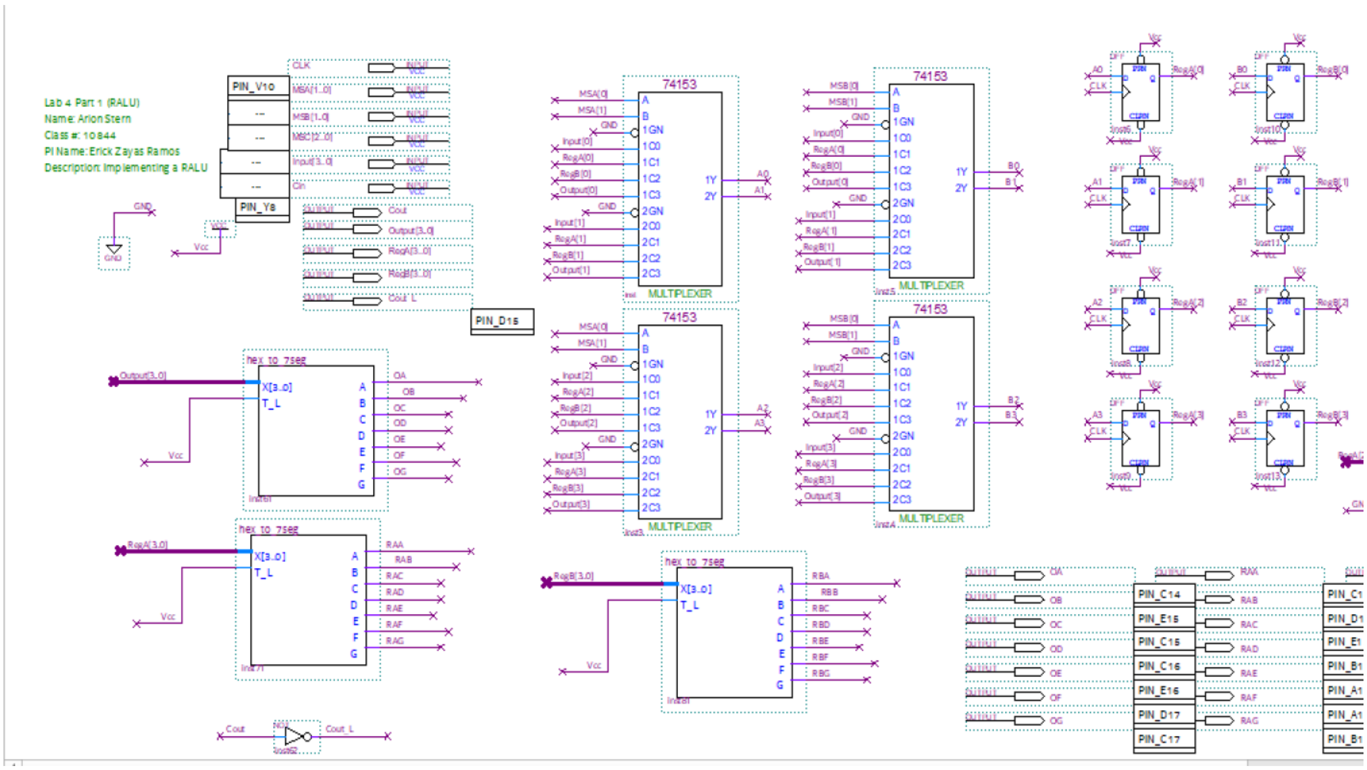


Figure 4: RALU Quartus Schematic 1/2 (Part 2)

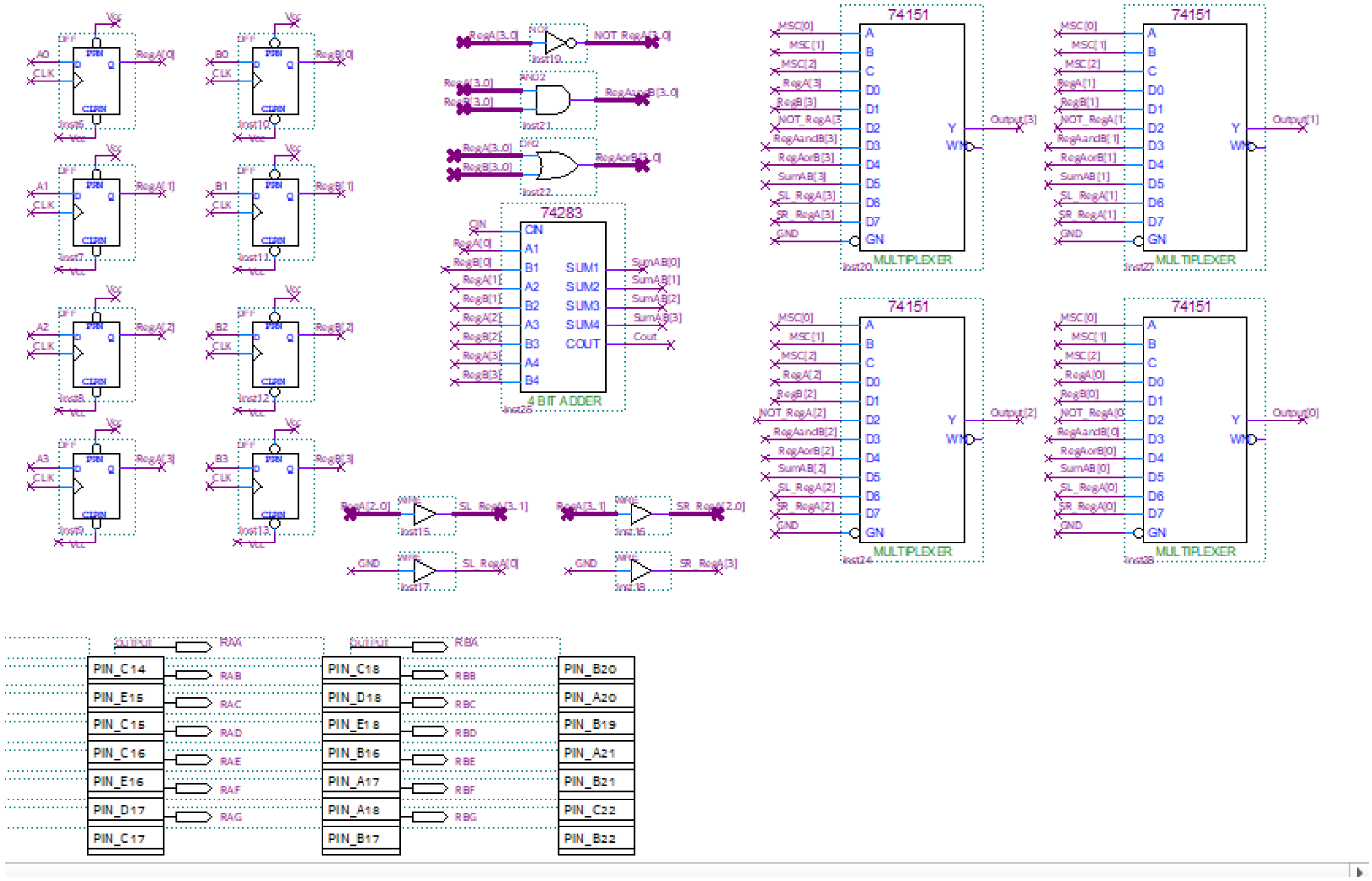
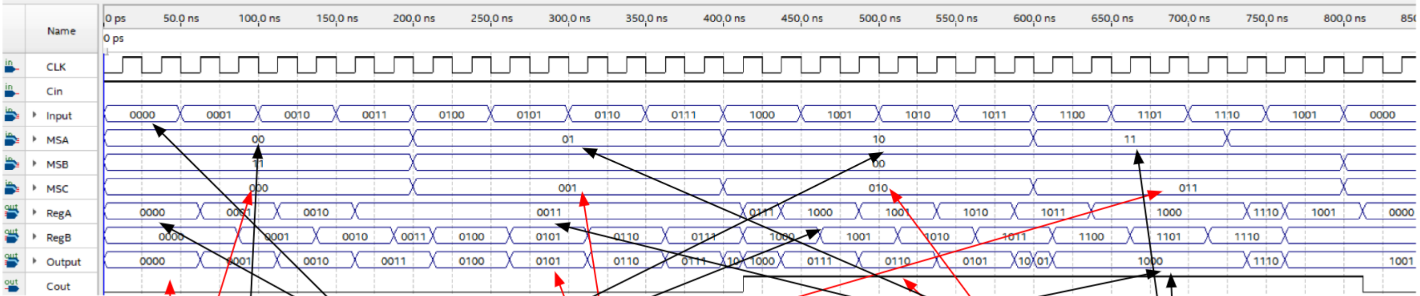


Figure 5: RALU Quartus Schematic 2/2 (Part 2)



When MSA/B is 00, the INPUT bus gets loaded into REGA/B. When MSA/B is 01, the contents of REGA go into REGA/B. When MSA/B is 10, the contents of REGB go into REGA/B. When MSA/B is 11, the OUTPUT bus gets loaded onto REGA/B.
MSC == 000 -> REGA to OUTPUT. MSC == 001 -> REGB to OUTPUT. MSC == 010 -> CompA to OUT. MSC == 011 -> REGA AND REGB to OUT.

MSC == 100 -> REGA OR REGB to OUT. MSC == 101 -> REGA plus REGB and Cin to OUT. MSC == 110 -> Left shift REGA OUT. MSC == 111 -> Right shift REGA to OUT.

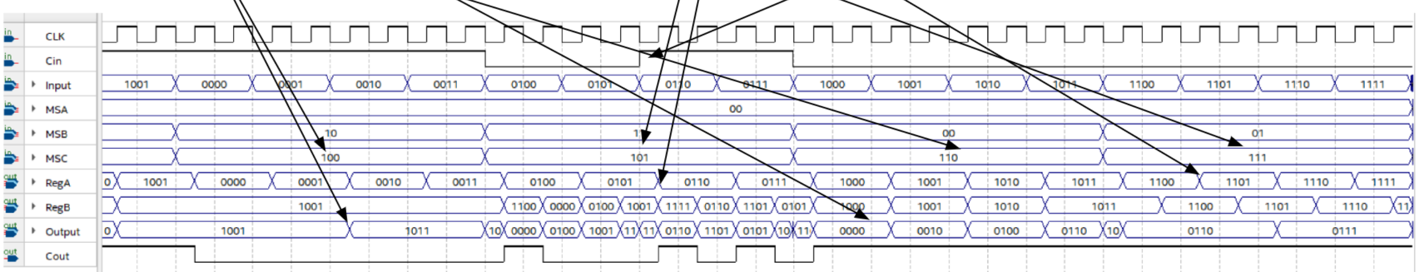


Figure 6: Annotated Simulation of RALU (Part 2)

Stern, Arion
10844
Erick Zayas Ramos
July 11, 2025

MSA	MSB	MSC	Input	Cin	RegA	RegB	Output	RegA+	RegB+	Output+	Cout+	Description
00	10	000	0001	0	X	X	X	0001	X	0001	X	Load A with 0001
01	00	000	0010	0	0001	X	0001	0001	0010	0001	0	Load B with 0010
01	11	100	X	0	0001	0010	0011	0001	0011	0011	0	OR A+B (Store in B/ Preserve A)
00	10	000	0011	0	0001	0011	0001	0011	0011	0011	0	Load A with 0011
01	00	000	0100	0	0011	0011	0011	0011	0100	0011	0	Load B with 0100
11	10	011	X	0	0011	0100	0000	0000	0100	0000	0	AND A•B (Store in A/ Preserve B)
00	10	000	0101	0	0000	0100	0000	0101	0100	0101	0	Load A with 0101
01	11	010	X	0	0101	0100	1010	0101	1010	1010	0	Complement A (Store in B / Preserve A)
00	00	000	0110	0	0101	1010	0101	0110	0110	0110	0	Load A and B with 0110
11	10	101	X	0	0110	0110	1100	1100	0110	0010	1	Sum A+B (Store in A/ Preserve B)
00	10	000	0111	0	1100	0110	1100	0111	0110	0111	0	Load A with 0111
01	11	111	X	0	0111	0110	0011	0111	0011	0011	0	Shift R one bit and Store in B
00	10	000	0011	0	0111	0011	0111	0011	0011	0011	0	Load A with 0011
01	11	110	X	0	0011	0011	0110	0011	0110	0110	0	Shift L one bit and Store in B
00	00	000	0011	0	0011	0110	0011	0011	0011	0011	0	Load A and B with 3
01	00	000	1100	0	0011	0011	0011	0011	1100	0011	0	Load B with $C
00	11	100	0101	0	0011	1100	1111	0101	1111	1111	1	OR 3+C / Store in B Load 5 to A
11	10	011	X	0	0101	1111	0101	0101	1111	0101	1	AND Previous Result with 5 Store in A
11	10	110	X	0	0101	1111	1010	1010	1111	0100	1	Shift Left (multiply by 2) Store in A
11	00	110	1001	0	1010	1111	0100	0100	1001	1000	0	SL (mult by 2) Load 9 in B
11	10	100	X	0	0100	1001	1101	1101	1001	1101	1	OR 9 to the result Store in A
11	10	010	X	0	1101	1001	0010	0010	1001	1101	0	Complement Result Store in A
11	10	111	X	0	0010	1001	0001	0001	1001	0000	0	Shift Right (divide by 2) Store in A

Figure 7: Filled Out Table 4 of RALU functions (Part 2)

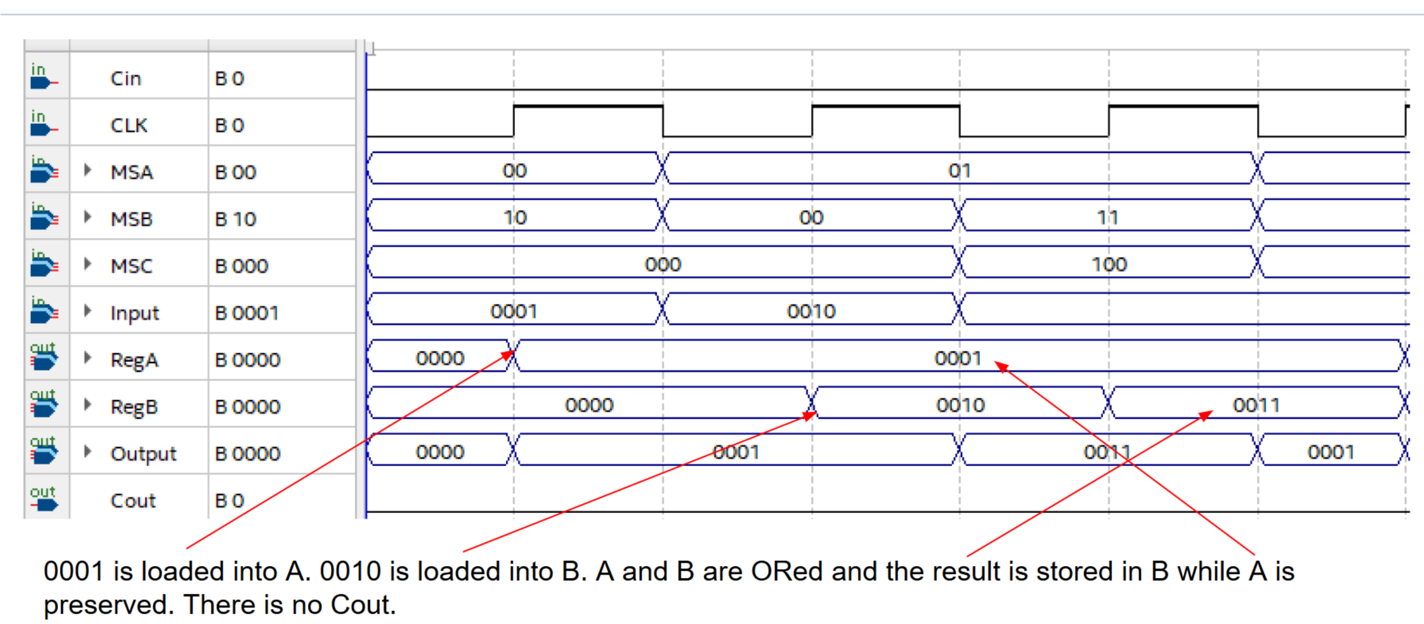


Figure 8: Annotated Simulation of function A (Part 2)

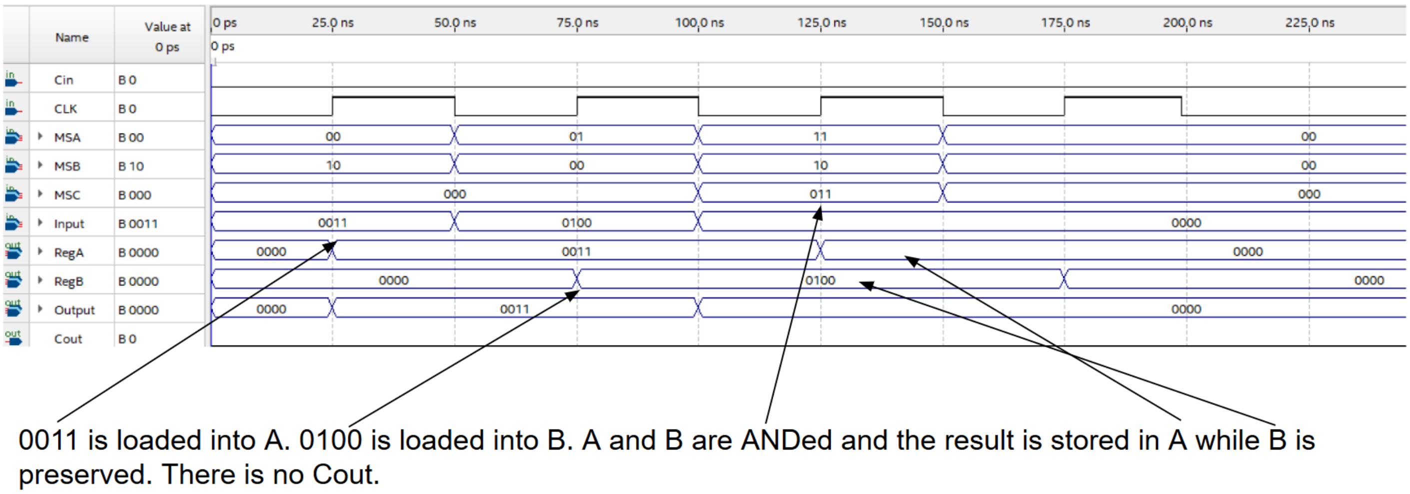


Figure 9: Annotated Simulation of function B (Part 2)

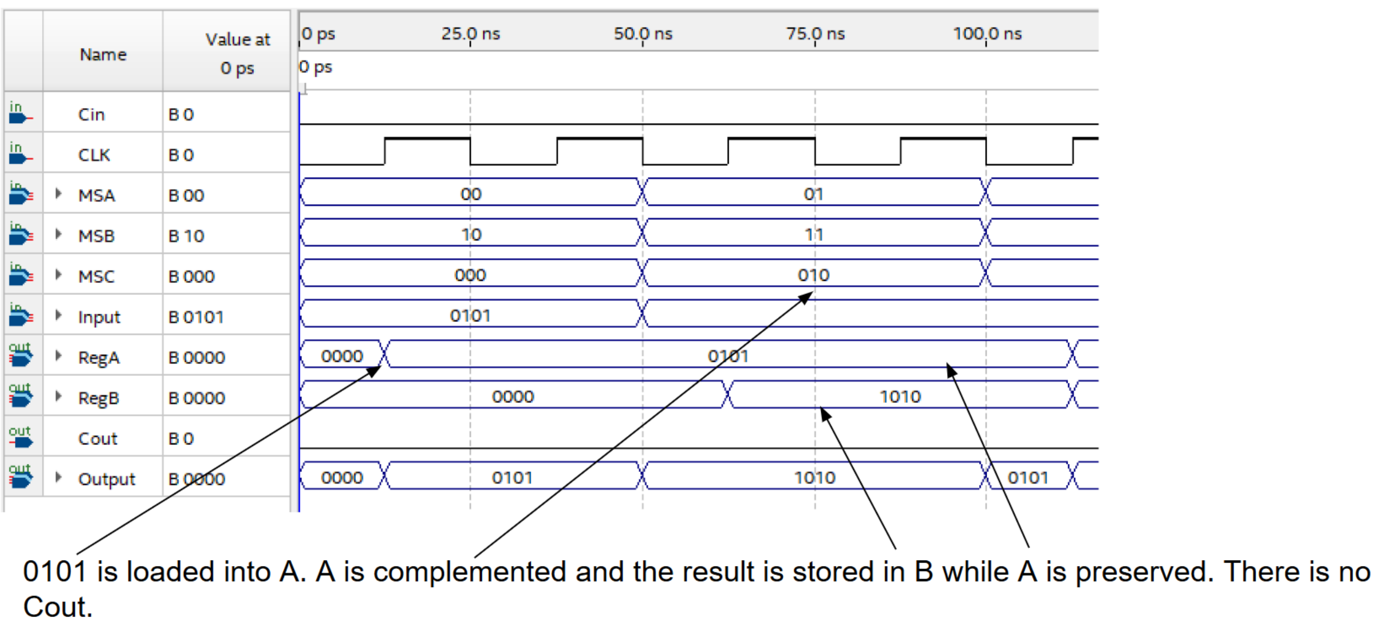


Figure 10: Annotated Simulation of function C (Part 2)

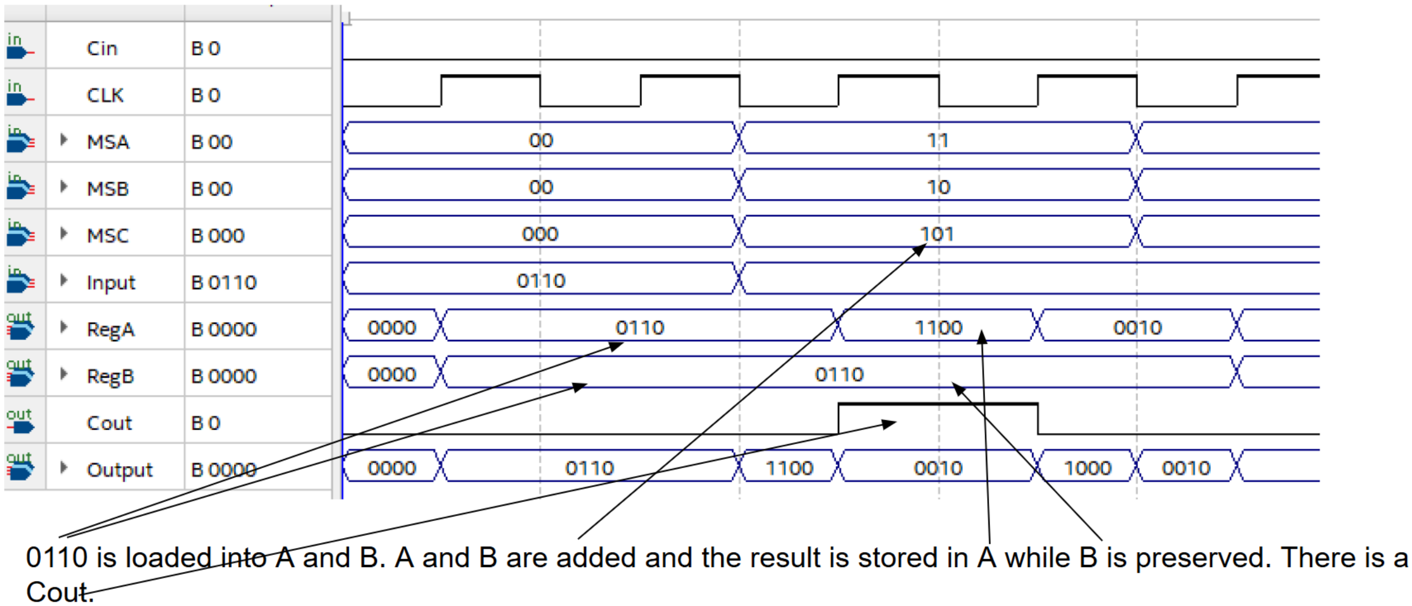


Figure 11 : Annotated Simulation of function D (Part 2)

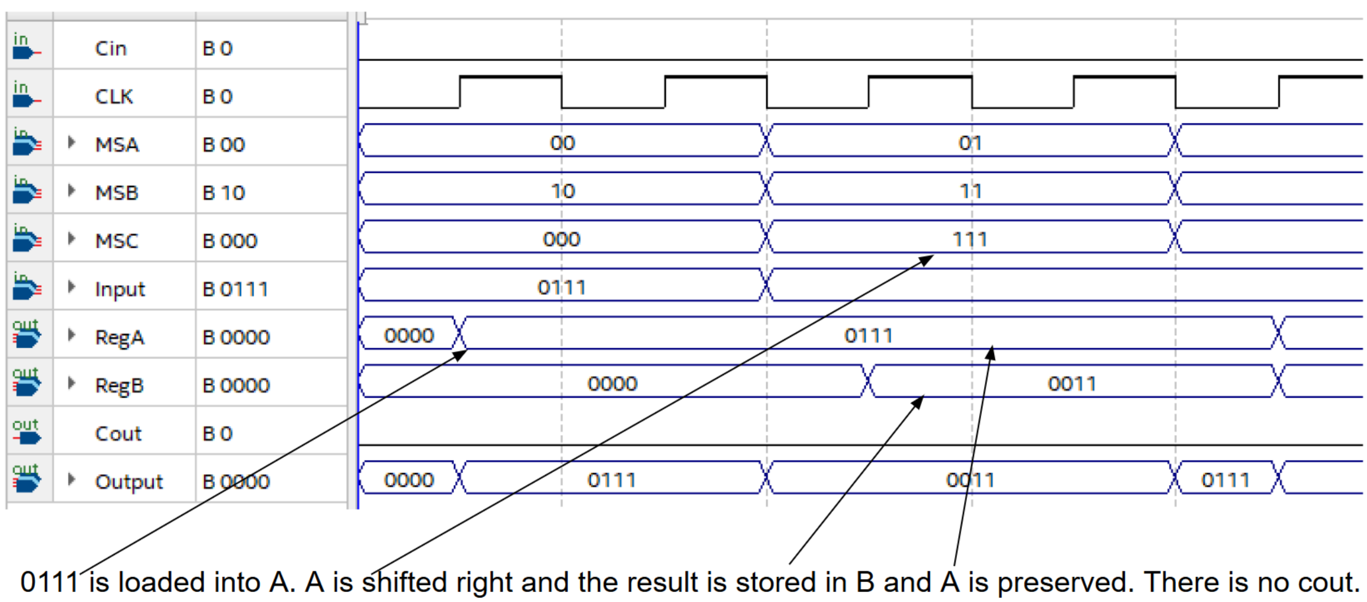


Figure 12: Annotated Simulation of function E (Part 2)

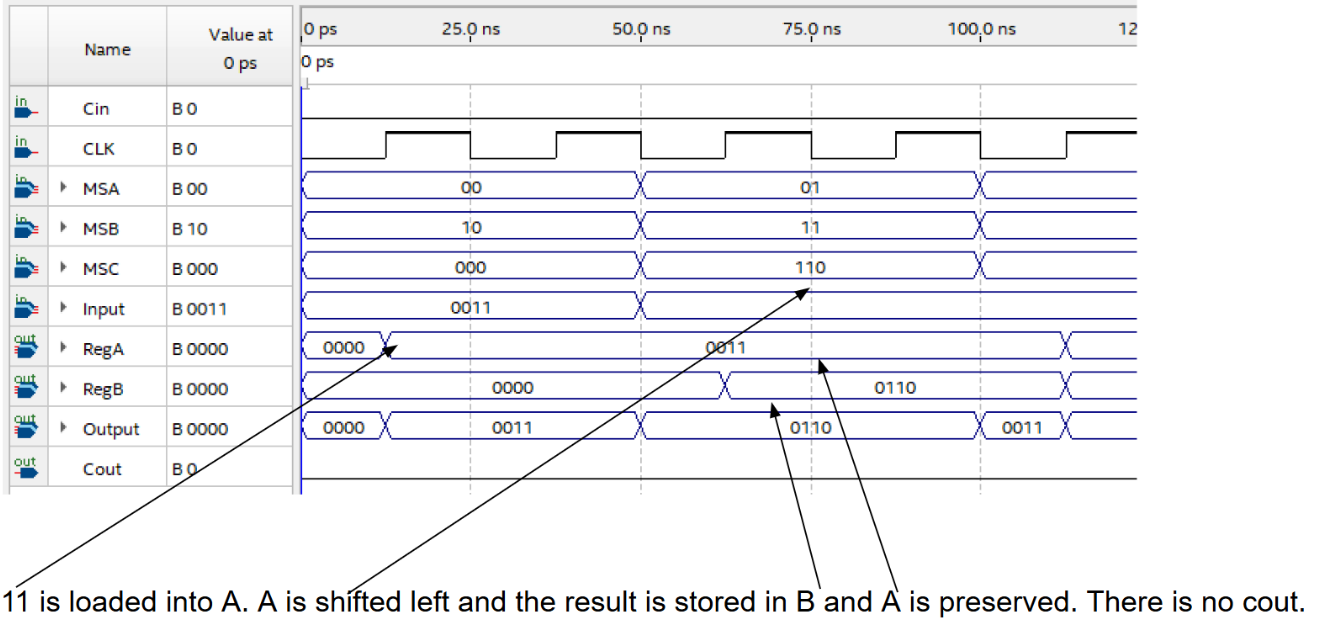


Figure 13: Annotated Simulation of function F (Part 2)

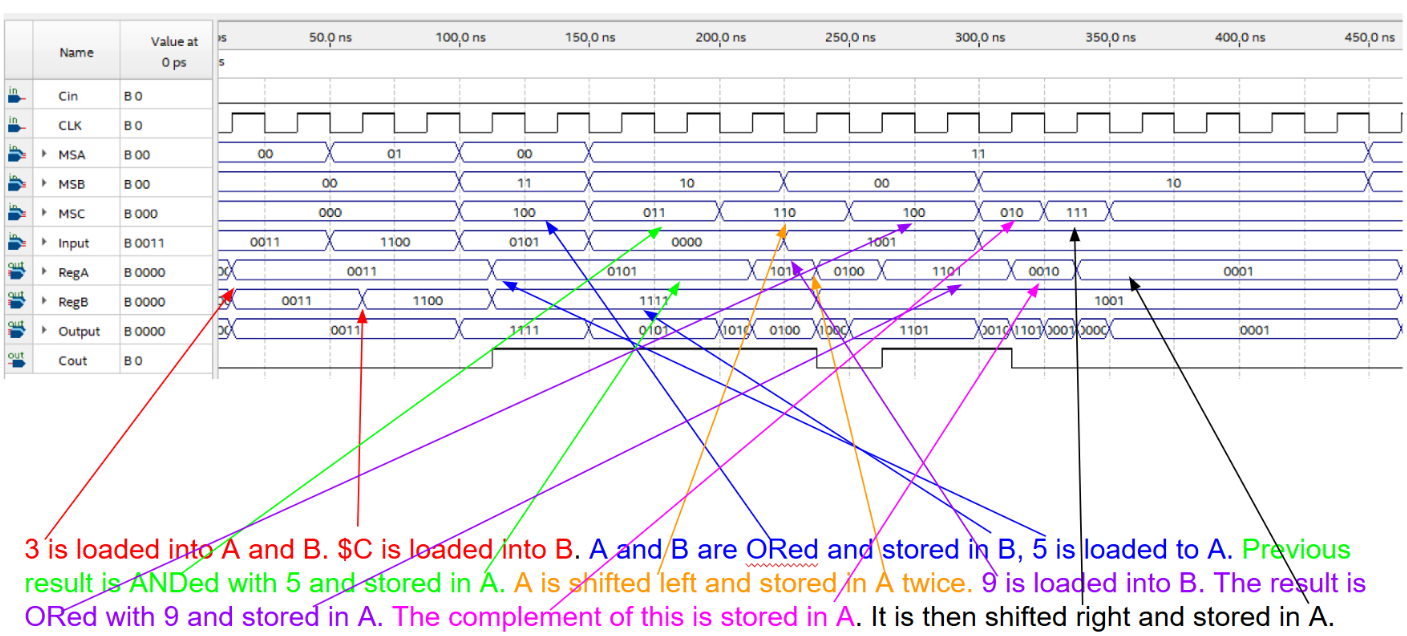


Figure 14: Annotated Simulation of Program G (Part 2)