Experiment #3: A Logic Processor

Name: Chuangy Zhang

NetID: czhan30

Name: Homero Vazquez

NetID: hvazqu6

1.Introduction

a. In this laboratory we design and construct a 4-Bit Serial Logic Processor. The circuit has 8 operations that take 4 cycles to be performed. The logical operations consist of NAND, AND, NOR, OR, XOR, XNOR, Set to 1 and Set to 0. There are 4 major components of the circuit, the control unit, the logic unit, the register file and the routing unit. The Control Unit is based on a Mealy finite state machine that controls the computation of the registers for 1-bit at a time for 4 clock cycles. The Register File contains the data of registers A and B. The Logic Unit has three input signals to select which logical operation will be performed. Finally, the Routing Unit controls the location to where the output of the logic unit will go.

b. Answers to pre-lab questions:

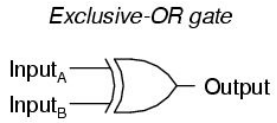
A. The XOR gate is the simplest circuit that can optionally invert a signal.

If InputA = 1(Optionally inverting a signal):

- InputB = 1 then Output = 0
- InputB = 0 then Output = 1

If InputA = 0(Not inverting the signal):

- InputB = 1 then Output = 1
- InputB = 0 then Output = 0



A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0

B. A modular design like the one from above improves testability because with one simple circuit we can test all the different outputs and inputs of our signal and that is why we can easily debug without the need of extra testing components like multiple switches, multiple signals, multiple gates. It cuts down on development time because we will not need to waste our time in building and designing a more complicated circuit.

2.Operation of the Logic Processor:

a. **In order to load data into registers A and B the following sequence of switches must be flipped:**

- D3,D2,D1,D0 are switches that will contain the data that you desire to load into the registers. So insert the data into those switches.
- The Control Unit has 3 inputs: LoadA, LoadB and Execute. When you want to load register A the LoadA signal should be high and the other 2 signals should be low then it will parallel load D3,D2,D1,D0 into register A. The same thing can be done to load register B but instead LoadB should be high and the rest should be low.

b. **Inorder to initiate a computation and routing operation the following sequence of switches must be flipped:**

- First LoadA and LoadB should be set to low.
- F2,F1,F0 are the signals that control the logical operations so set them to the desired operation.
- R1,R0 control the routing unit so you will set them to store data in registers A and B.
- The final step will be to make sure that the Execute signal is high so that the computation is executed.

3.Written Description, Block Diagram and State Machine Diagram of Logic Processor:

a. **Written Description: Our design has four main components, The Control Unit, the Routing Unit, the Register File, and the Computational Logic Unit.**

- **Control Unit:**

  The Control Unit enables the user to load data parallely into Registers A and B by assigning the signals LoadA and LoadB to high respectively. It also takes the Execute signal to start a computation and ends after 4 cycles based on the Mealy State Machine design.

- **Routing Unit:**

  The Routing unit controls the destination of the output of the Computational Logic Unit. It has 2 signals for selection and it writes back to registers A and B.

- **Register File:**

  The Register File contains two shift registers represented by registers A and B. The two registers can be accessed parallely to load data from the switches

D3-D0. The output of the Routing Unit will write back to the most significant bit of each register and the output of each register file will be the least significant bit which is the input of the Computational Logic Unit.

- **Computational Logic Unit:**

  The Computational Logic Unit has three input signals F2-F0 which allows the user to select 1 out of the 8 operations to be computed. It takes the least significant bit of registers A and B. The output goes through the Routing Unit.

b. High-Level Block Diagram:

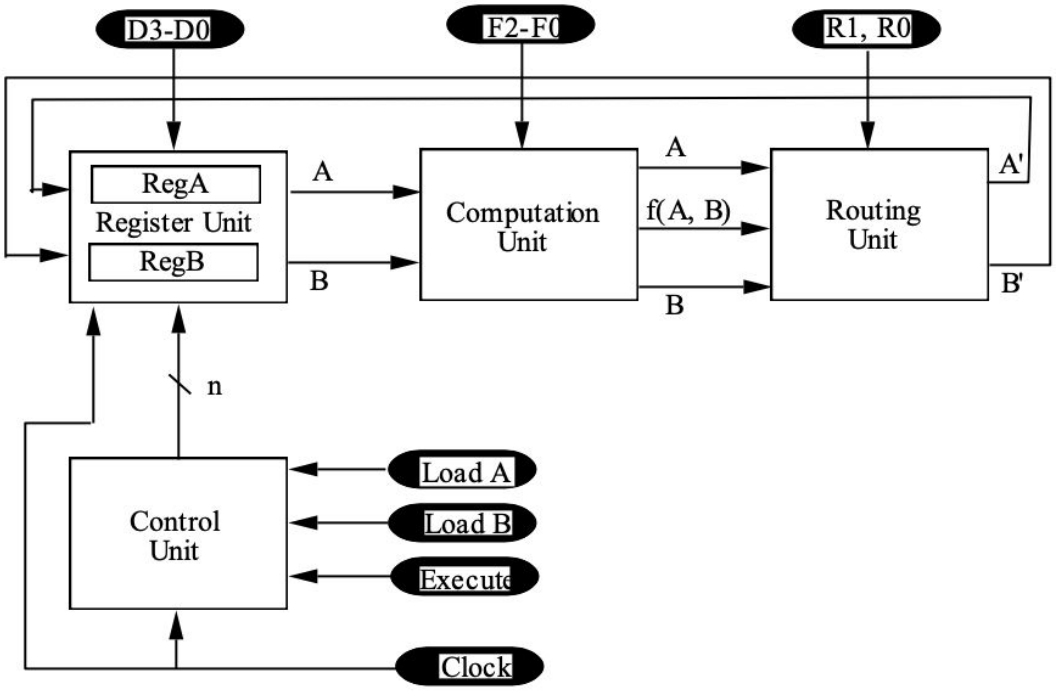
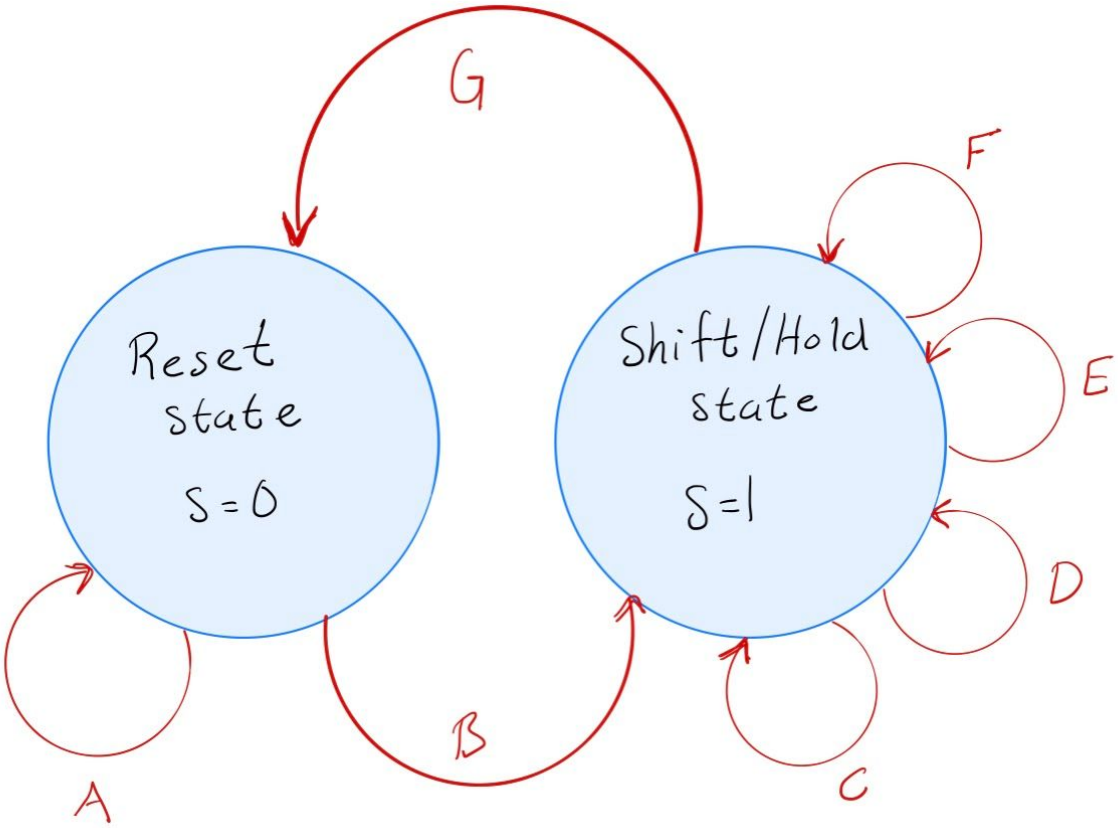


Figure 1: Block Diagram

c. State Machine Diagram:

For our design we utilized a Mealy State Machine, in this type of state machine the input signals are connected to the output of our combinational logic circuit. By using a Mealy State Machine design we reduced the number of states to only 2 states.



Arc label	Cout/Execute/shift	Meaning
A	00/0/0	if (execute != 1), then Reset
B	00/1/1	if(execute == 1) then start shift
C	01/d/1	if(cout != 00), then go back to shift state
D	10/d/1	if(cout != 00), then go back to shift state
E	11/d/1	if(cout != 00), then go back to shift state
F	00/1/0	if(cout == 00 && execute == 1), S = 0, stay at the hold state
G	00/0/0	if(cout == 00 && execute == 0), then go to Reset

4.Design steps taken and detailed circuit schematic diagram:

Exec. Switch ('E')	Q	C1	C0	Reg. Shift ('S')	Q^+	$C1^+$	$C0^+$
0	0	0	0	0	0	0	0
0	0	0	1	d	d	d	D
0	0	1	0	d	d	d	D
0	0	1	1	d	d	d	D
0	1	0	0	0	0	0	0
0	1	0	1	1	1	1	0
0	1	1	0	1	1	1	1
0	1	1	1	1	1	0	0
1	0	0	0	1	1	0	1
1	0	0	1	d	d	d	D
1	0	1	0	d	d	d	D
1	0	1	1	d	d	d	D
1	1	0	0	0	1	0	0
1	1	0	1	1	1	1	0
1	1	1	0	1	1	1	1
1	1	1	1	1	1	0	0

C1C0

EQ

	00	**01**	**11**	**10**
00	0	x	x	x
01	0	1	1	1
11	0	1	1	1
10	1	x	x	x

	00	**01**	**11**	**10**
00	0	x	x	x
01	0	1	1	1
11	1	1	1	1
10	1	x	x	x

$S = EQ' + C0 + C1 = [(EQ)'C0'C1']'$

$Q+ = E + C0 + C1 = (E'C0'C1')'$

C1C0

EQ

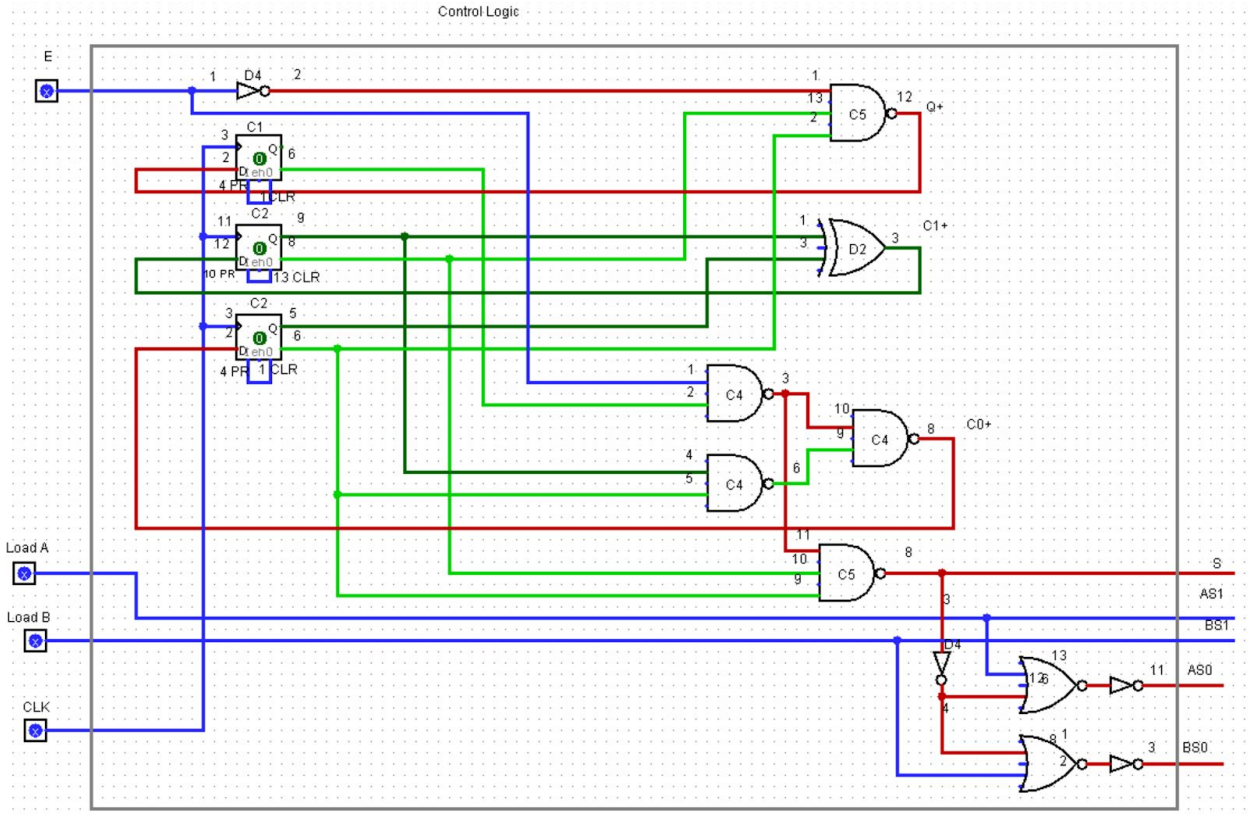
	00	01	11	10
00	0	x	x	x
01	0	1	0	1
11	0	1	0	1
10	0	x	x	x

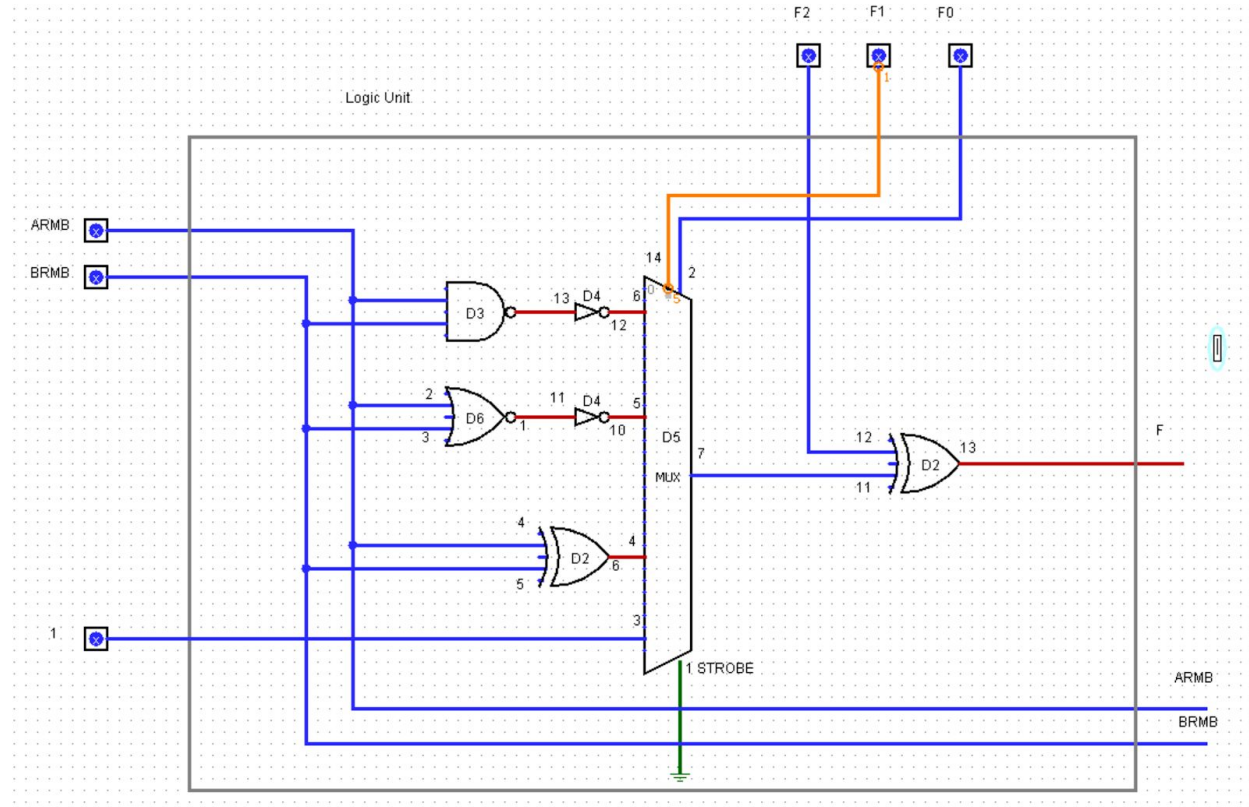
	00	01	11	10
00	0	x	x	x
01	0	0	0	1
11	0	0	0	1
10	1	x	x	x

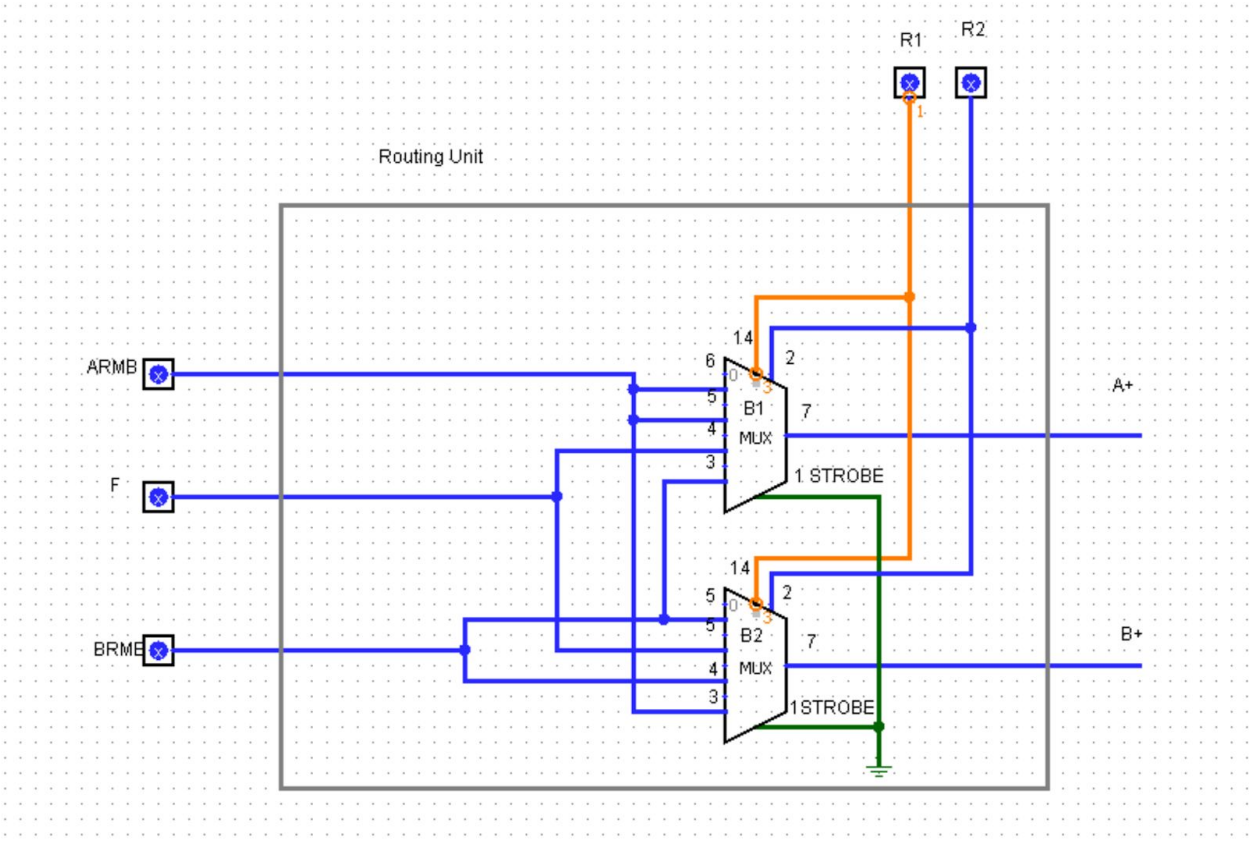
C1+ = C1'C0 + C1C0' = C1⊕ C0 C0 += EQ' + C1C0' = [(EQ')'(C1C0')']'

When working on our design, there were many decisions that we had to make and so we had to consider different approaches to our design and how it would affect the complexity of our circuit. But most of the time our decisions were answered by going for the design with the least number of gates. An example of this is when working on the Computational Logic Unit we could use 8:1 MUX and some logic gates to implement it but we realized that if we used an XOR gate to invert the output intentionally we could reduce the complexity of our circuit.

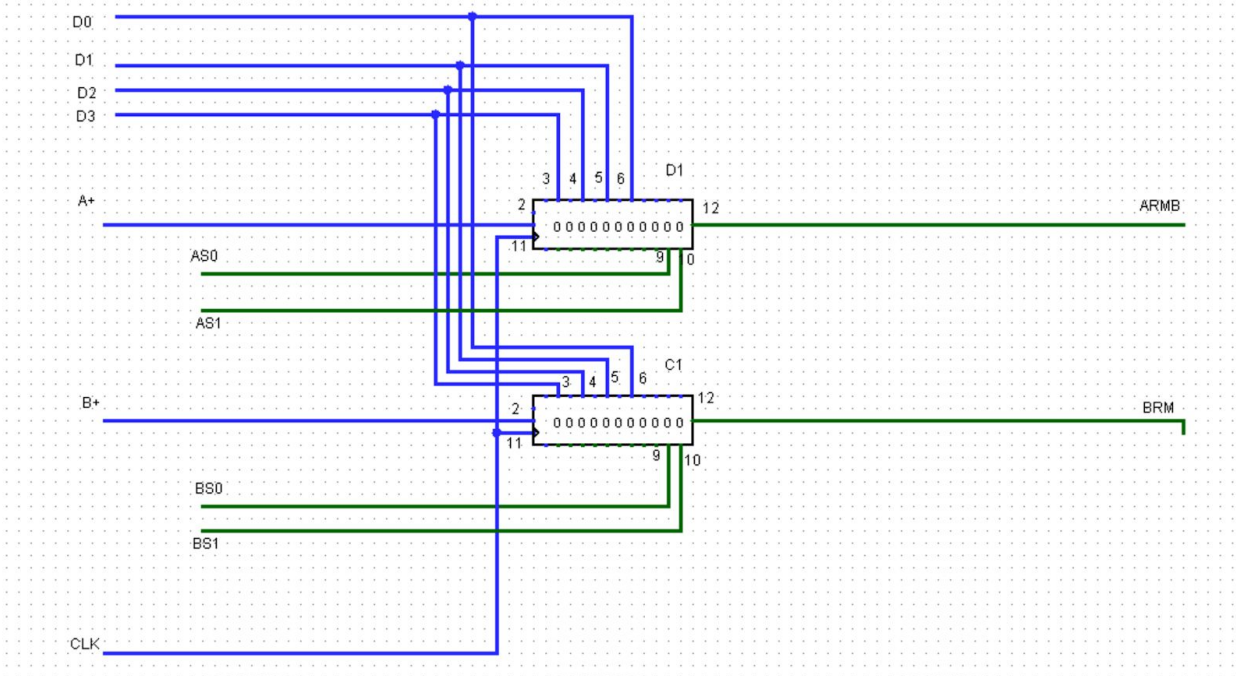
Detailed Circuit Schematic:


Control Logic
E
D4
C1
C2
C2
C5
Q+
D2
C1+
C4
C4
C0+
C4
C5
Load A
Load B
CLK
S
AS1
BS1
D4
AS0
BS0

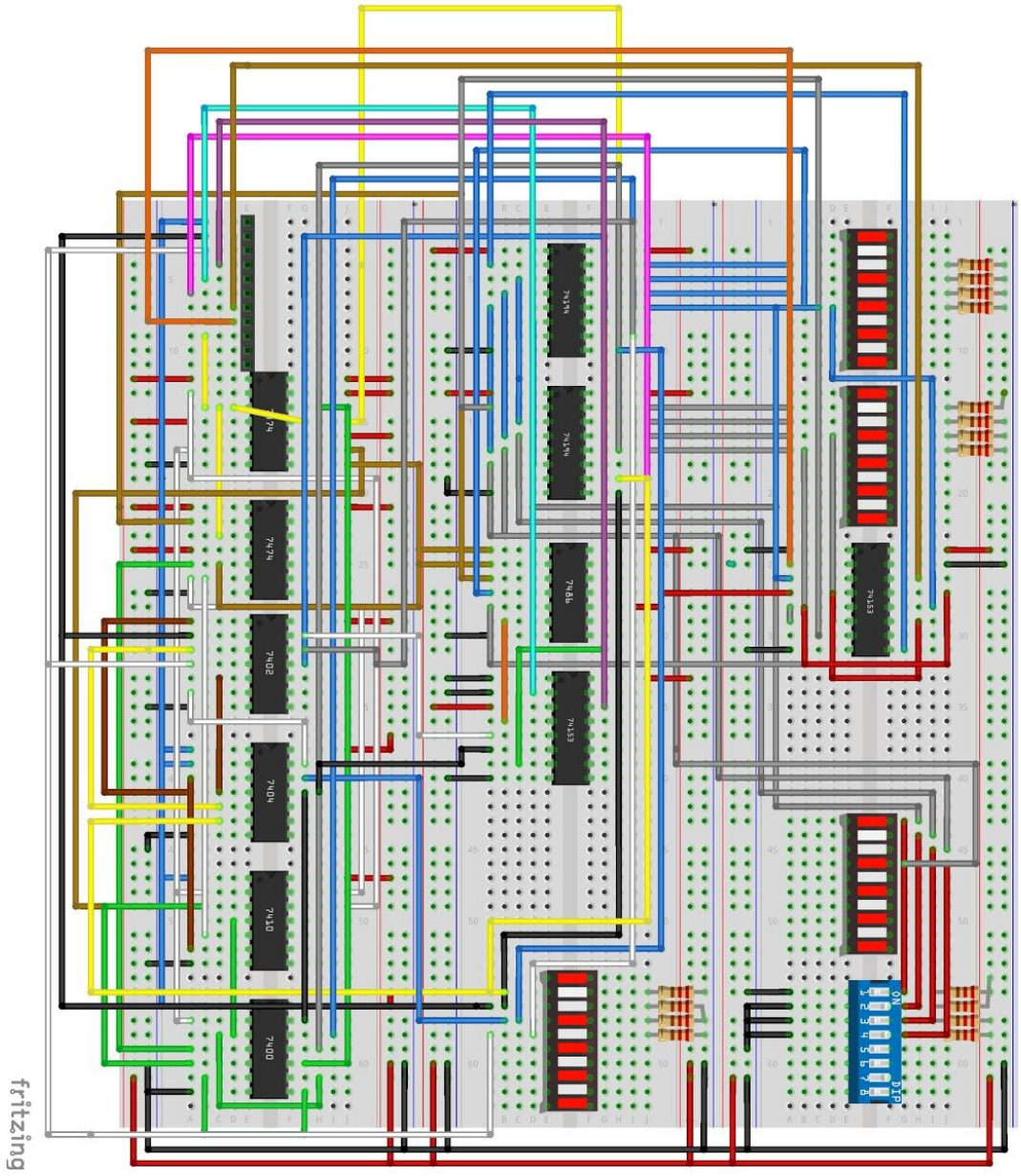

F2
F1
F0
Logic Unit
ARMB
BRMB
D3
D4
D6
D4
D5
MUX
D2
F
D2
1
1 STROBE
ARMB
BRMB



Register File



5.Breadboard View



6.Description of all bugs encountered, and corrective measures taken:
The bug that we encountered was using the counter for our finite state machine, we were unable to halt the execution of instruction after it shifted 4 times. We realized that we should build our finite states machine by using D-flip-flop and it can halt the execution. The other bug that we encountered was our router unit, the routing instruction was different for 01 and 10, but we were able to fix it by changing the R1 and R0.

7.Conclusion:

a. **Summary:**
   This experiment was a great experience to understand better the functionality of a processor. It was really interesting to be able to utilize a complex finite state machine to create a control unit. It was great to see the Logic Processor perform all of the 8 operations and how their solutions were written back to memory. I believe that to enhance our design we could figure out how to implement a counter into our design and maybe that way we can reduce the number of chips in our circuit.

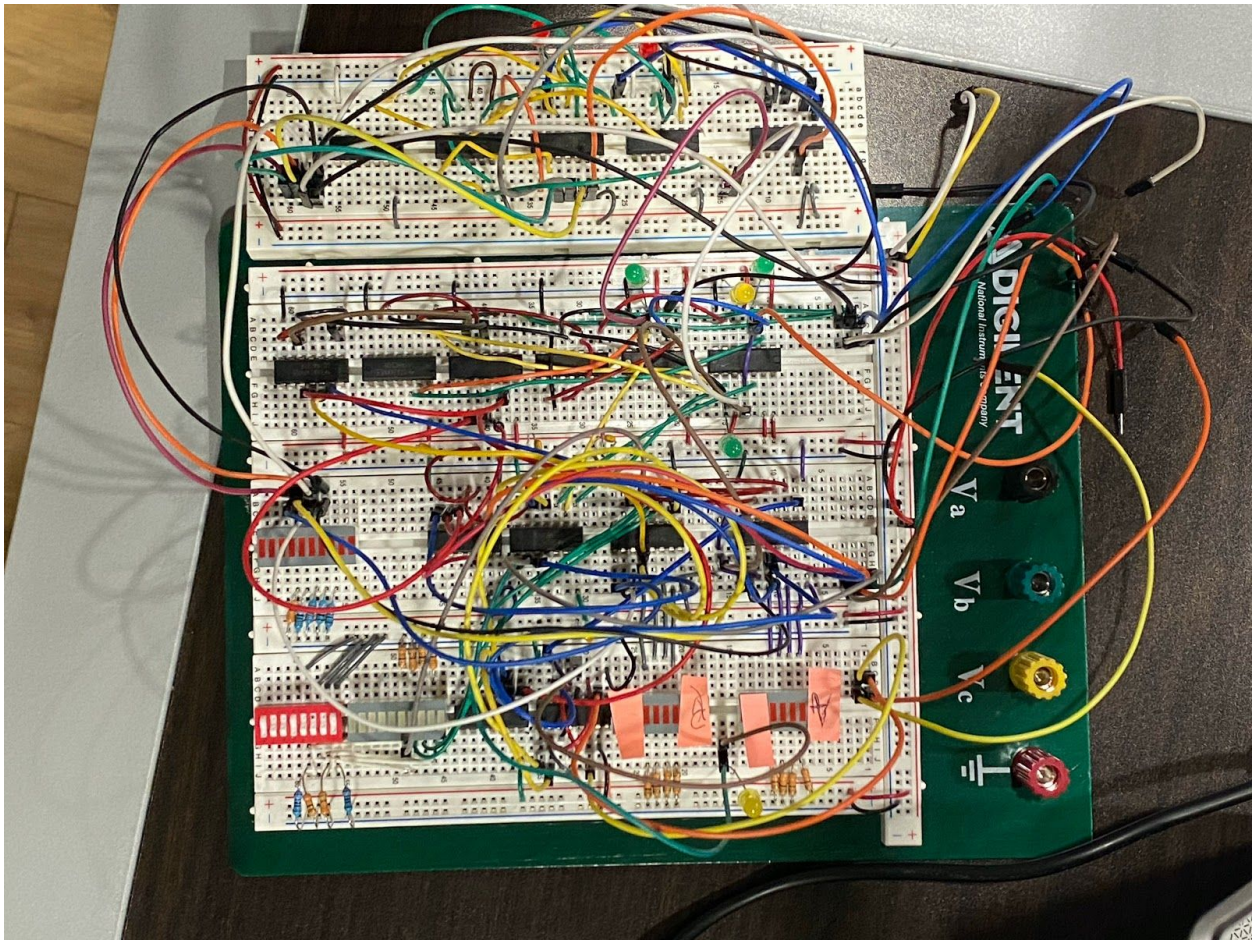
b. **Post Lab Questions:**

   - **Discuss the design process of your state machine:**

     When designing the state machine we combined the outputs of our current states with our current inputs that combination reduced the number of states which makes the process of the computations simpler.

   - **What are the tradeoffs of a Mealy machine vs a Moore machine?**

     On a Mealy State Machine the output depends on its current state and its current input. A Moore State Machine's output only depends on its current state. Because of that a Mealy State Machine will have less states and react facter to it's input. But a Moore State Machine can be slower because the output has to be decoded and therefore there might be more gate delays for that decoding process.


DIGILENT
National Instruments Company
Va
Vb
Vc