A Logic Processor

Hongbo Zheng, Yuhao Yuan

Summer 2021

ECE 385

Section AB1

TA: Hanfei Wang

Introduction

The purpose of this lab is to design and build a 4-bit version of serial logic operation processor which is capable of calculating one of 8 functions: AND, OR, XOR, set all bits to 1, NAND, NOR, XNOR, and set all bits to 0. The circuit first parallelly loads two different sets of 4-bit data into two 4-bit registers (register A and register B). Then 1 computation cycle (4 clock cycles) after Execute switch is triggered, register A and register B will store one of the three 4-bit values: A, B, or operational result F(A,B). This lab is an attempt at building a very elementary version of a CPU.

Operation of the logic processor

Load data into Register A

Quartus Testbench

1. Set D[3:0] signal to some 4-bit binary value (For example: 4'b1010)
2. Set LoadA signal to logic '1' temporarily for about 5ns (#5) and set it back to logic '0'
   For example: LoadA=1'b1
   #5 LoadA=1'b0

Extra Credit

1. Flip switch SW[3:0] to some 4-bit binary value
2. Click the LoadA pushbutton

Load data into Register B

Quartus Testbench

1. Set D[3:0] signal to some 4-bit binary value (For example: 4'b0101)
2. Set LoadB signal to logic '1' temporarily for about 5ns (#5) and set it back to logic '0'
   For example: LoadB=1'b1
   #5 LoadB=1'b0

Extra Credit

1. Flip switch SW[3:0] to some 4-bit binary value
2. Click the LoadB pushbutton

Initiate a computation and routing operation

Quartus Testbench

1. Set the 3-bit Select signal F[2:0] of the Computational Unit to some 3-bit binary value
   For example: F=3'b010 XOR
2. Set the 2-bit Select signal R[1:0] of the Routing Unit to some 2-bit binary value
   For example: R=2'b01 A=A, B=F(A,B)
3. Set Execute signal to logic '1' temporarily for about 5ns (#5) or constantly logic '1'
   For example: Execute=1'b1
   #5 Execute=1'b0
   The operation of the logic processor will start when Execute signal changes from LOW to HIGH and will last only for 1 computation cycle (4 clock cycles). Therefore, the length of the operation will not be affected if the Execute signal changes to LOW during the operation or the Execute signal stays HIGH after the operation finishes.

Extra Credit

1. Flip switch SW[8:6] (3-bit Select signal of the Computational Unit) to some 3-bit binary value
2. Flip switch SW[5:4] (2-bit Select signal of the Routing Unit) to some 2-bit binary value
3. Flip switch SW[9] (Execute signal) to logic '1' temporarily or constantly

Written Description of the Circuit

Overview

The 4-bit logic processor can be broken down into 4 parts which are designed individually: Register Unit, Computational Unit (ALU), Routing Unit, and Control Unit.

The register unit is composed of two 4-bit bidirectional universal shift register (SN74LS194E chip). They are used to store and shift the input data D[3:0]. The parallel load, shift right, and hold functionalities of the register are utilized.

The computational unit is also known as the arithmetic logic unit. It is designed by implementing AND, OR, XOR, and 1111 in combinational logic using only NAND gates, XOR gates, and Vcc. Then the 4 outputs and 2 select bits F1 and F0 are connected to a 4-to-1 MUX. The output of the MUX is feed through an XOR gate. The other input of the XOR gate is the select signal F2, therefore the circuit can perform the rest 4 operations: NAND, NOR, XNOR, and 0000.

The routing unit is built with two 4-to-1 MUXs. The 2 routing output overwrite A and B each with one of the three values: its original value A, the value of B, or the result F(A,B) from the ALU. The select signal for both MUXs are R1 and R0.

The most complicated and challenging part of the logic processor is the control unit because it contains both combinational logic and sequential logic. According to the truth table in Lab 1 document, one SOP equation for shift signal and another SOP equation for Q next state are derived after drawing the K-map of each output. The two equations are: $S = E\bar{Q} + C_1 + C_0$ and $Q^+ = E + C_1 + C_0$. A D-Flip-Flop is required because some special edge cases need to be taken care. More specifically, one case is that Execute is changed from HIGH to LOW during the computation cycle, and the other case is that Execute stays HIGH after the operation is finished. Therefore, the D-Flip-Flop serves to detect a LOW and a subsequent HIGH

of the Execute signal and then restart the process after the operation is completed. In addition, a 4-bit synchronous binary counter (SN74163E chip) is used in the design in order to detect when 1 computation cycle (4 clock cycles) have passed after Execute is triggered. Then, the machine can halt after the task is completed. Moreover, it also simplifies the Mealy machine because the counter can combine the 4 shift states into 1 state.

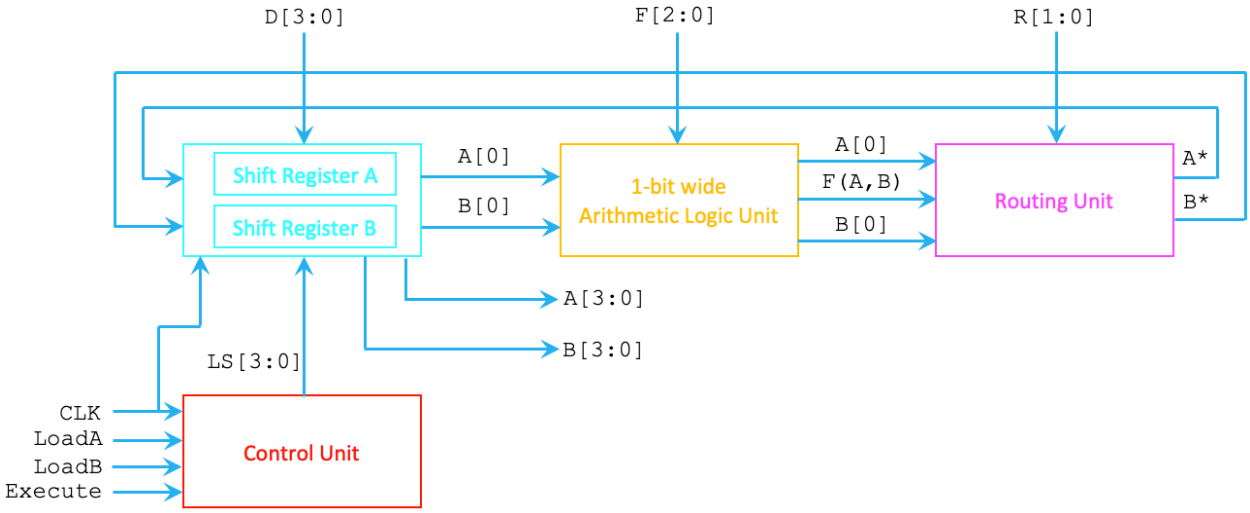
Block Diagrams

High Level Block Diagram

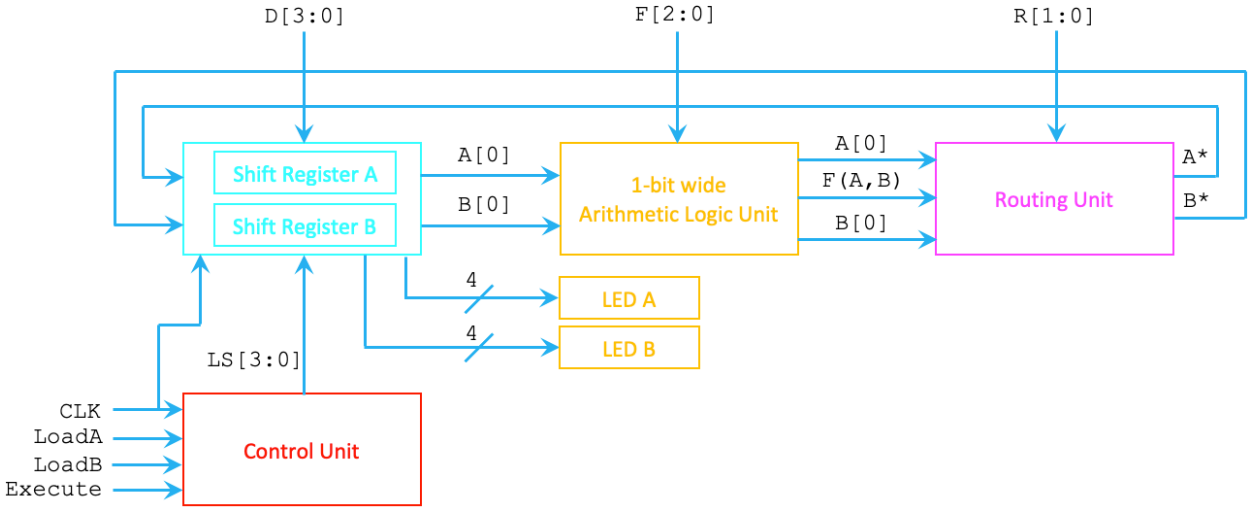
High Level Block Diagram of the Physical Circuit (Extra Credit)

The high-level block diagram shows all of the inputs to the logic processor as a whole and shows the accompanying output. It treats the internal components all as a black box. Overall, there are 6 categories of user selected inputs and 1 CLK input, and the outputs are the 4-bit binary value of A and the 4-bit binary value of B from corresponding register A and register B. For the extra credit, the 4-bit binary value of A and the 4-bit binary value of B are displayed on LED A and LED B for better illustration. There are 4 possible outputs: A and B have their original values, A and B are swapped, keep A and replace B with F (output from ALU), or keep B and replace A with F (the output from ALU).

4-bit Logic Processor Block Diagram



4-bit Logic Processor Block Diagram (Extra Credit)



The more detailed version of the block diagram shows how each of the 4 components interact with each other through I/O paths. The inputs to the shift registers are parallelly loaded from the 4-bit binary value assigned in the testbench. Two 1-bit binary value are also loaded into the register during execution. For extra credit, the inputs are parallelly loaded from the switches SW[3:0], and the contents of these registers are displayed on the LEDs to the user.

The ALU has a 3-bit user input specifying one of 8 possible functions it can apply to the two shift-in inputs from the shift registers. The inputs to the ALU are serial with respect to each shift register. During each clock cycle while executing, only 1 bit from each of the shift register enters the ALU. The outputs of the ALU are the original value of A, original value of B, and the result of the operation specified by the select signals of the computational unit.

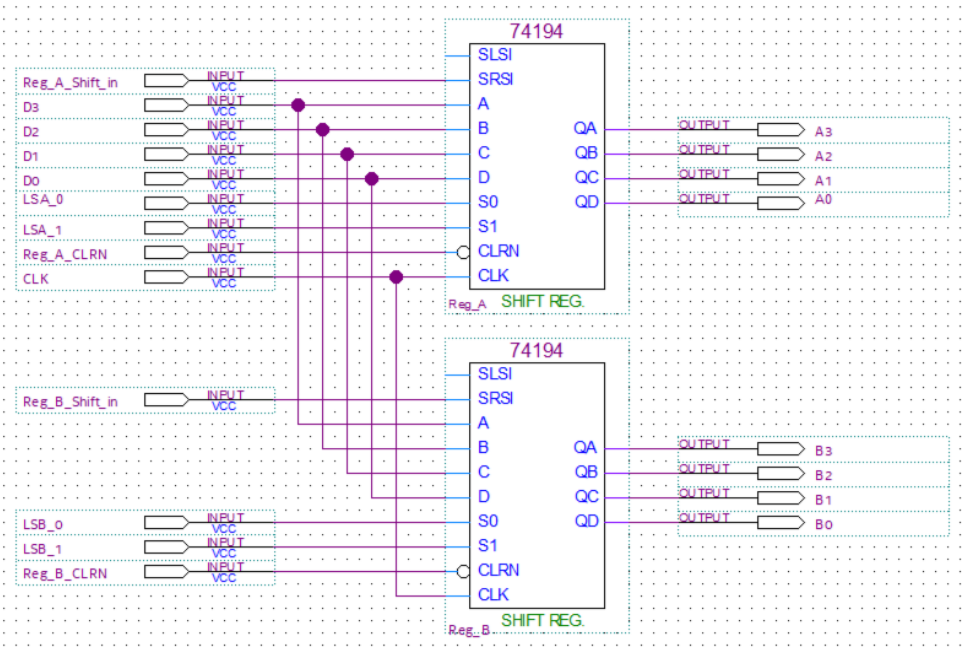
The routing unit contains 2-bit user input which encode for 4 possible routing instructions. The outputs of the computational unit are consolidated into 2 outputs A* and B* representing what should finally be stored in the shift registers once the operation is completed.

The control unit makes sure the correct number of shifts are done in order for 1 computational cycle (4 clock cycles) to be completed. It also controls the operating modes of register A and register B.

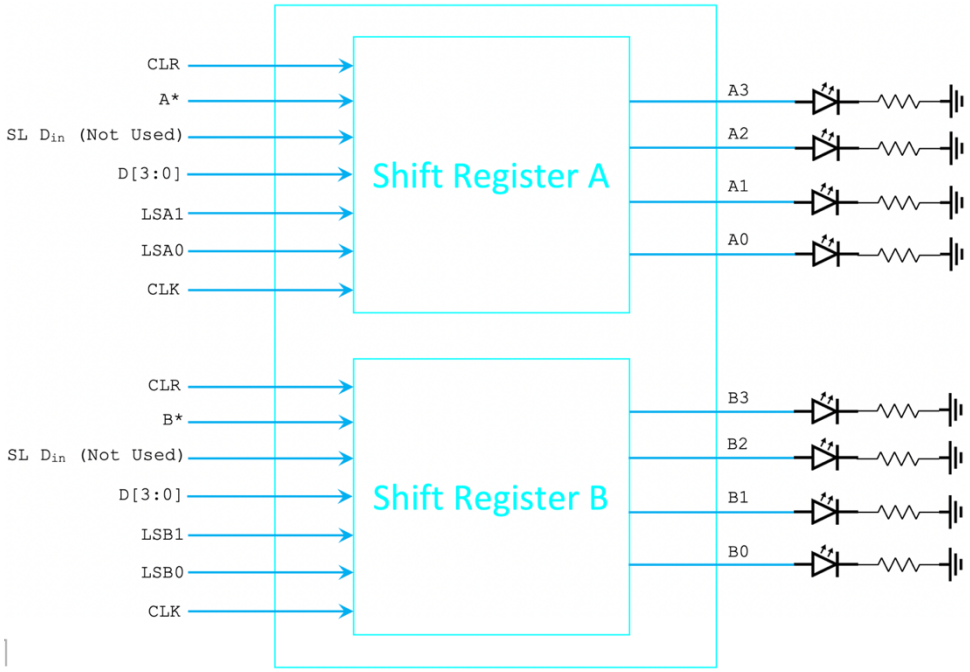
Lab Documentations

Register Unit

Detailed Circuit Schematic (Quartus)



Detailed Circuit Schematic (Extra Credit)



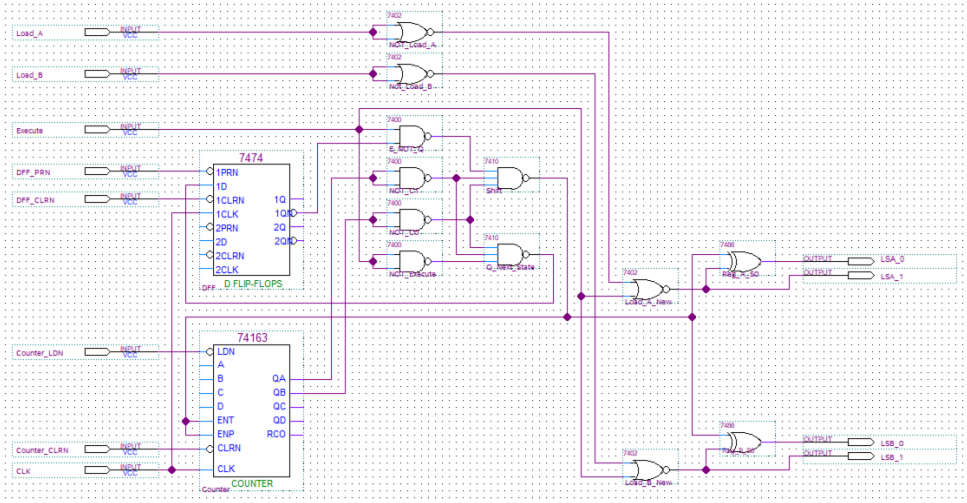
LS1	LS0	Mode
0	0	Hold Current Value
0	1	Shift Right
1	0	Shift Left
1	1	Parallel Load

Table 1. Select signals and corresponding operational modes of the Shift Register

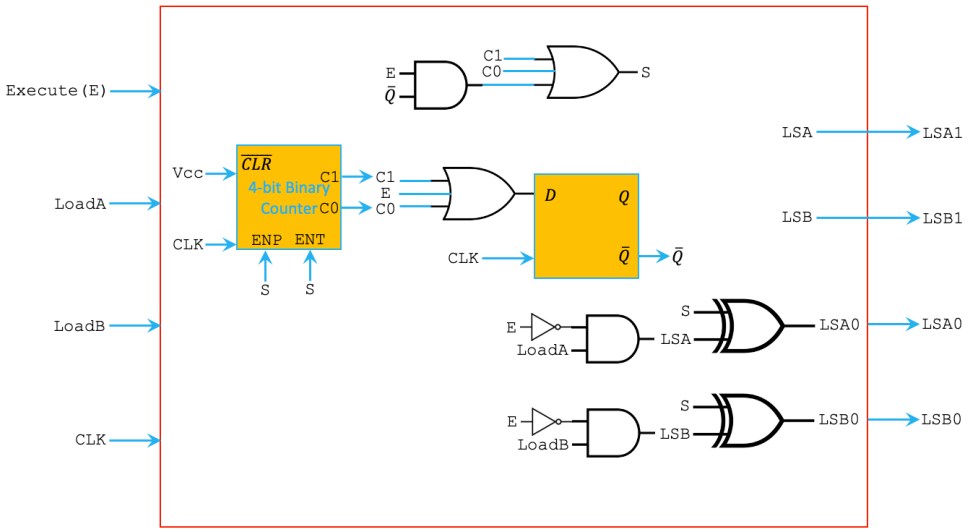
Both shift registers are identical in design. According to the datasheet of the 4-bit bidirectional universal shift register (SN74LS194E chip), the CLR pin has to connected to power (Vcc) in order for the chip to operate, CLK is connected to the clock signal generated by the testbench (For Extra Credit, CLK is connected to the clock signal generated by the DE10-Lite board), the 1-bit A* and 1-bit B* are used as shift-in input while the registers are in shift right mode, the 2-bit LS signal is connected to the outputs of the control unit in order to control the operating mode of the register, and both registers are connected to the same 4-bit binary value D[3:0] as inputs. For extra credit, the outputs of each register are connected to 4 LED's (each is connected with an 1kΩ resistors in series) to display the contents for better illustration.

Control Unit

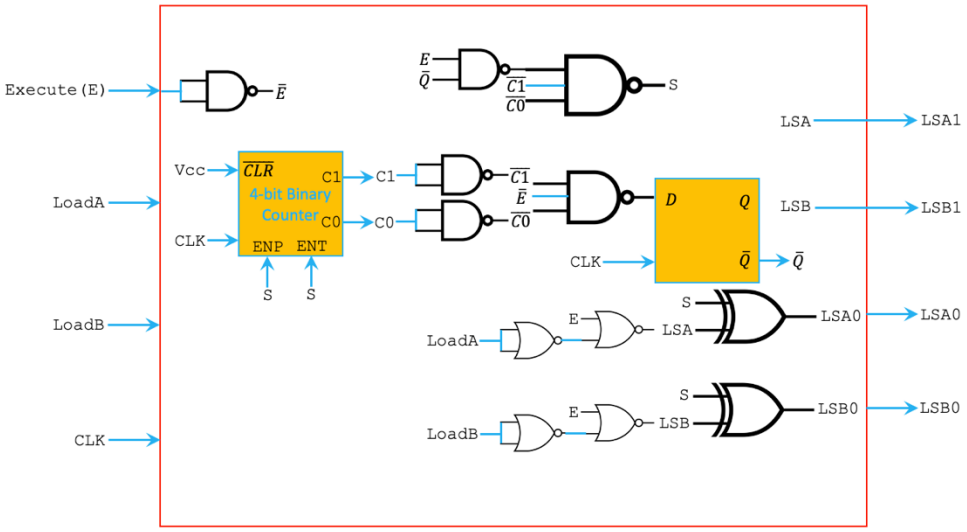
Detailed Circuit Schematic (Quartus)



Detailed Circuit Schematic (Extra Credit)



Replace AND OR NOT with NAND NOR



(Some inputs of the counter are not shown because they are left unconnected)

Since execute and load cannot be high at the same time, some combinational logic is needed to prevent this from happening. (Implemented for both LoadA and LoadB)

Execute (E)	Load (original Load signal)	Load* (new Load signal)	Function
0	0	0	Not Loading
0	1	1	Loading Data
1	0	0	Not Loading
1	1	0	Not Loading

Table 2. Truth Table of Load* signal

$$Load^* = \bar{E}Load$$

According to the data sheet of the SN74LS194E (4-bit bidirectional universal shift register), the LS1 and LS0 signals should have the following functions based on Load* and Shift signals. (Implemented for both Register A and Register B)

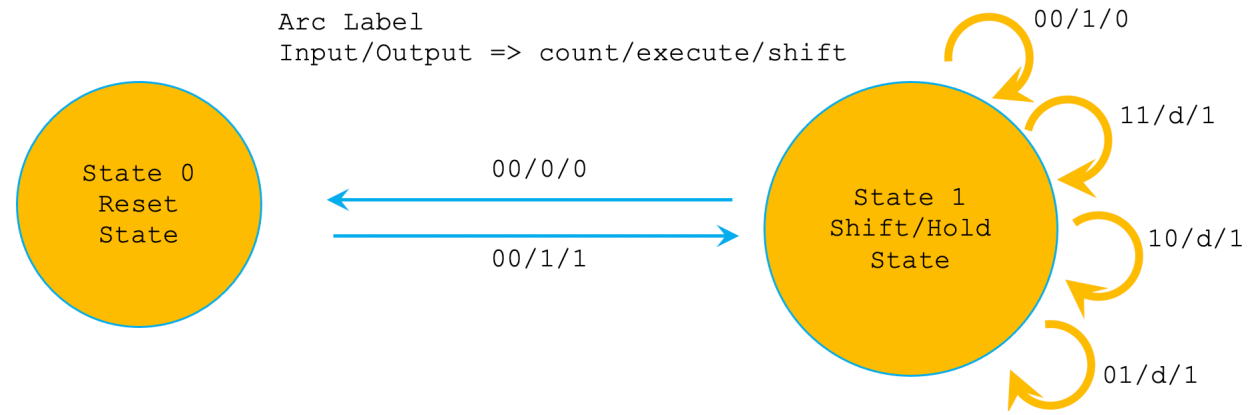
Load*	Shift (S)	LS1 (Load Shift 1)	LS0 (Load Shift 0)	Function
0	0	0	0	Hold Current values of A B
0	1	0	1	Shift Right
1	0	1	1	Parallel Load values into A B
1	1	X(1)	X(0)	Impossible

Table 3. Truth Table for LS1 and LS0 based on inputs Load* and Shift

$$LS1 = Load^*$$

$$LS0 = Load^* \oplus Shift$$

FSM Diagram



Execute (E)	Q	C1	C0	Register Shift (S)	Q^+	C1	C0
0	0	0	0	0	0	0	0
0	0	0	1	X	X	X	X
0	0	1	0	X	X	X	X
0	0	1	1	X	X	X	X
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	X	X	X	X
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
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

Table 2. FSM Truth Table

S		C1C0			
		00	01	11	10
EQ	00	0	X	X	X
	01	0	1	1	1
	11	0	1	1	1
	10	1	X	X	X

$$S|_{sop} = E\bar{Q} + C1 + C0$$

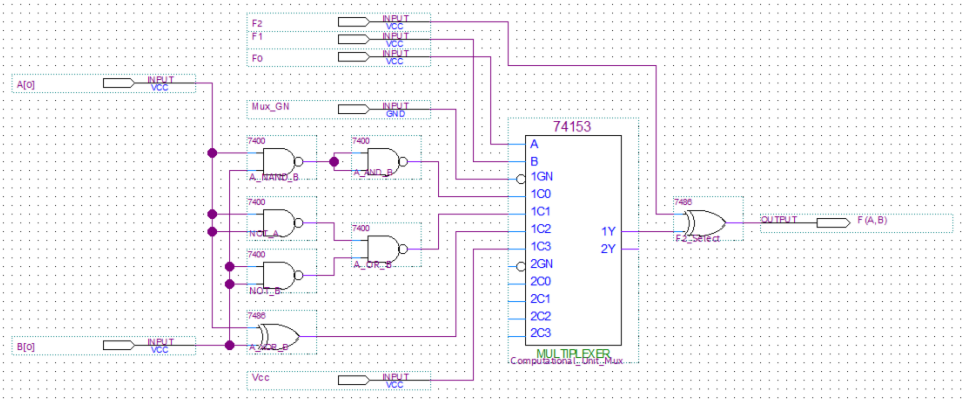
Q^+		C1C0			
		00	01	11	10
EQ	00	0	X	X	X
	01	0	1	1	1
	11	1	1	1	1
	10	1	X	X	X

$$Q^+|_{sop} = E + C1 + C0$$

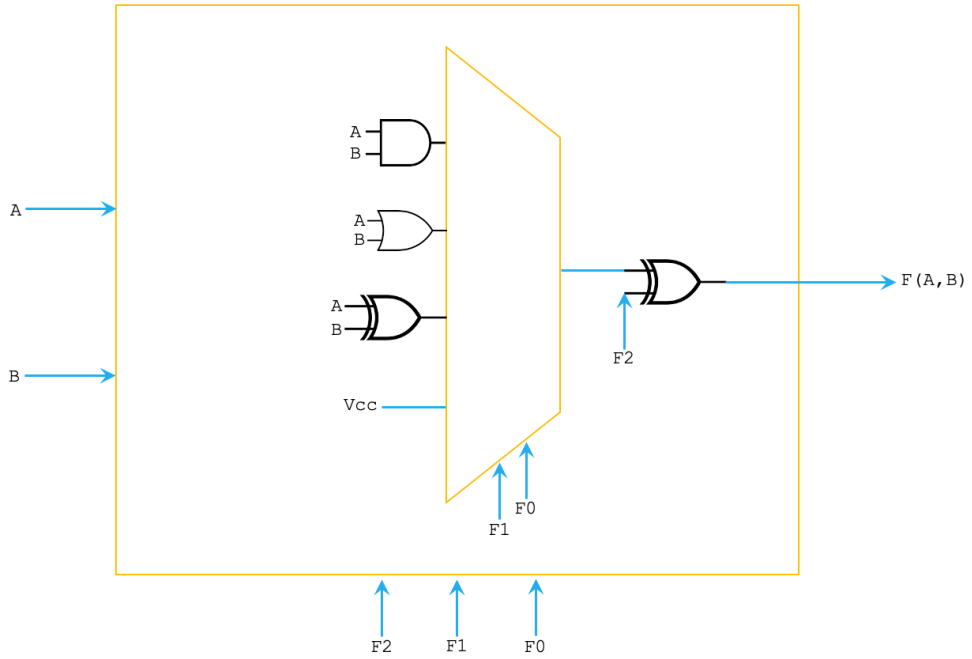
There are many Xs in C_0 and C_1 because execution is primarily dependent on the state of the execute signal. Thus, it seemed more efficient to use a counter in our design in accordance with the mealy machine. Since the registers need to shift 4 times during 1 computation cycle (4 clock cycles), only the 2 least significant bits of the counter are needed. The other 2 bits will count up till 15 which is unnecessary. According to the datasheet of the 4-bit synchronous binary counter (SN74163E chip), C_1 and C_0 are the 2 least significant bits of the counter, and ENP and ENT have to be HIGH for the counter to start counting. Therefore, ENP and ENT are connected to shift signal so that the counter only counts when the shift signal is HIGH; this enables us to precisely count 4 shifts.

Computational Unit

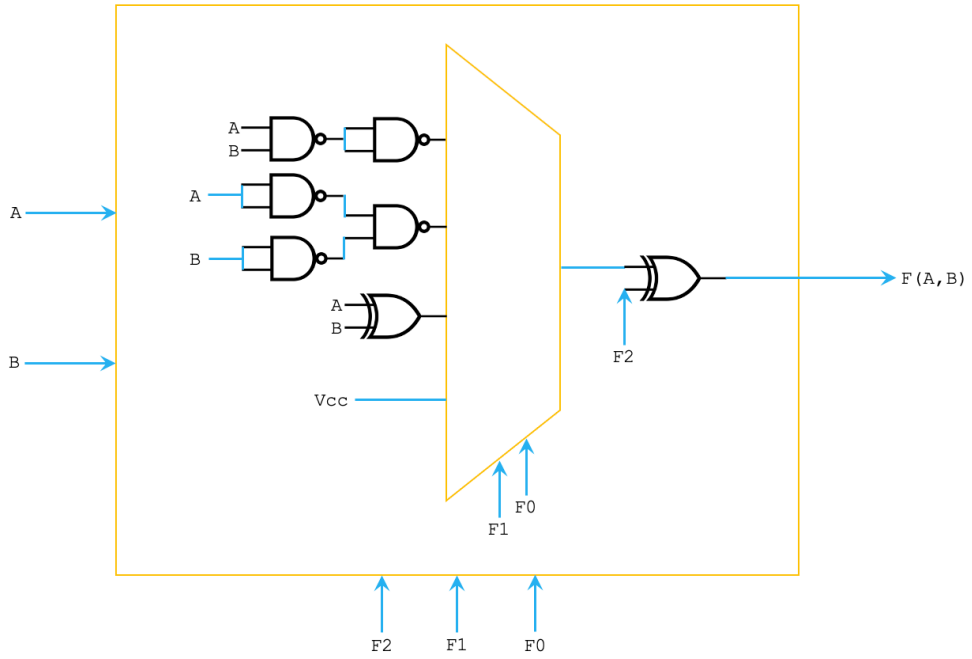
Detailed Circuit Schematic (Quartus)



Detailed Circuit Schematic (Extra Credit)



Replace AND, OR gates with NAND gates



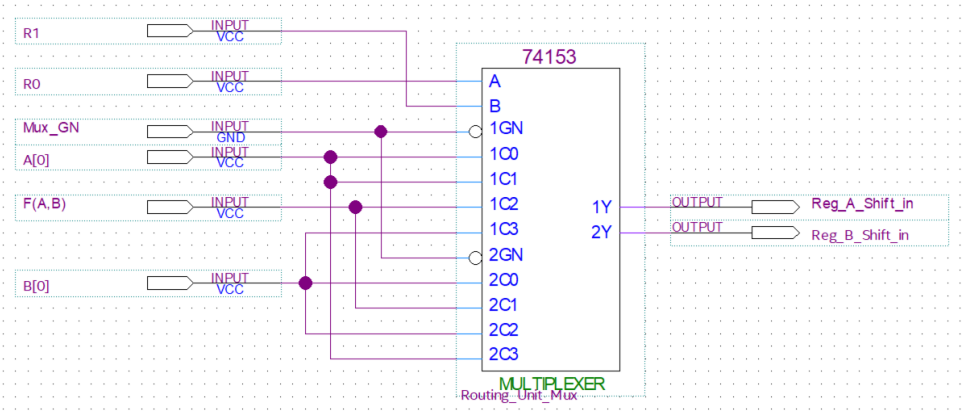
F2	F1	F0	F(A,B)
0	0	0	A AND B
0	0	1	A OR B
0	1	0	A XOR B
0	1	1	1
1	0	0	A NAND B
1	0	1	A NOR B
1	1	0	A XNOR B
1	1	1	0

Table 3. ALU functions and their corresponding select signal F2 F1 F0

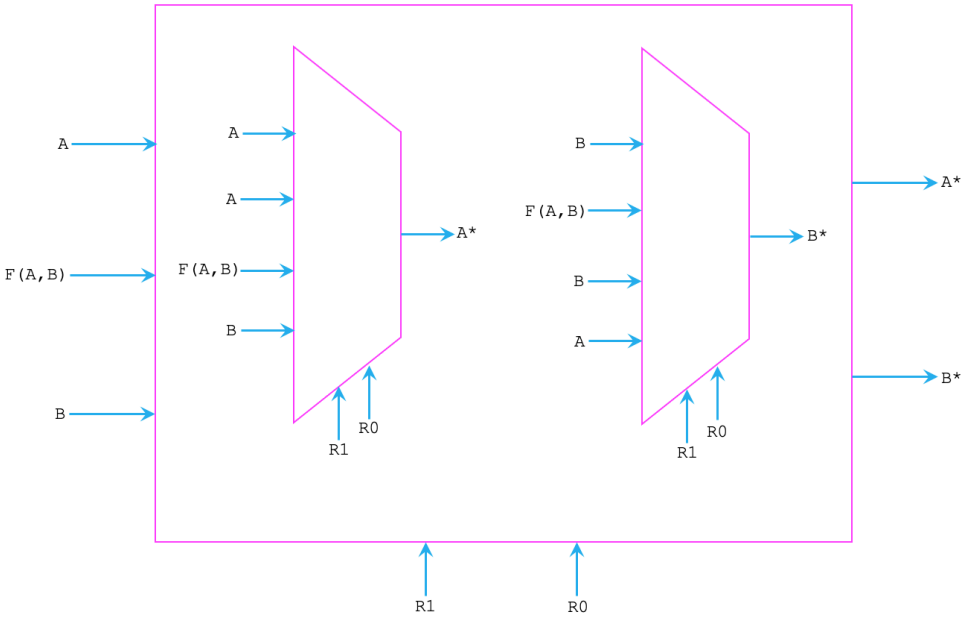
The computation unit has the first 4 of the 8 functions implemented in logic using a combination of NAND and XOR gates, as well as the power source (Vcc). These 4 functions are connected to a 4x1 MUX whose output is connected to one input of the XOR gate. The other input of the XOR is the most significant select bit F2. The XOR gate is not required, but it could save resources by replacing an 8-to-1 MUX with a 4-to-1 MUX. It is critical because it is a 2 input 1 output element with one of the inputs having the potential to invert.

Routing Unit

Detailed Circuit Schematic (Quartus)



Detailed Circuit Schematic (Extra Credit)

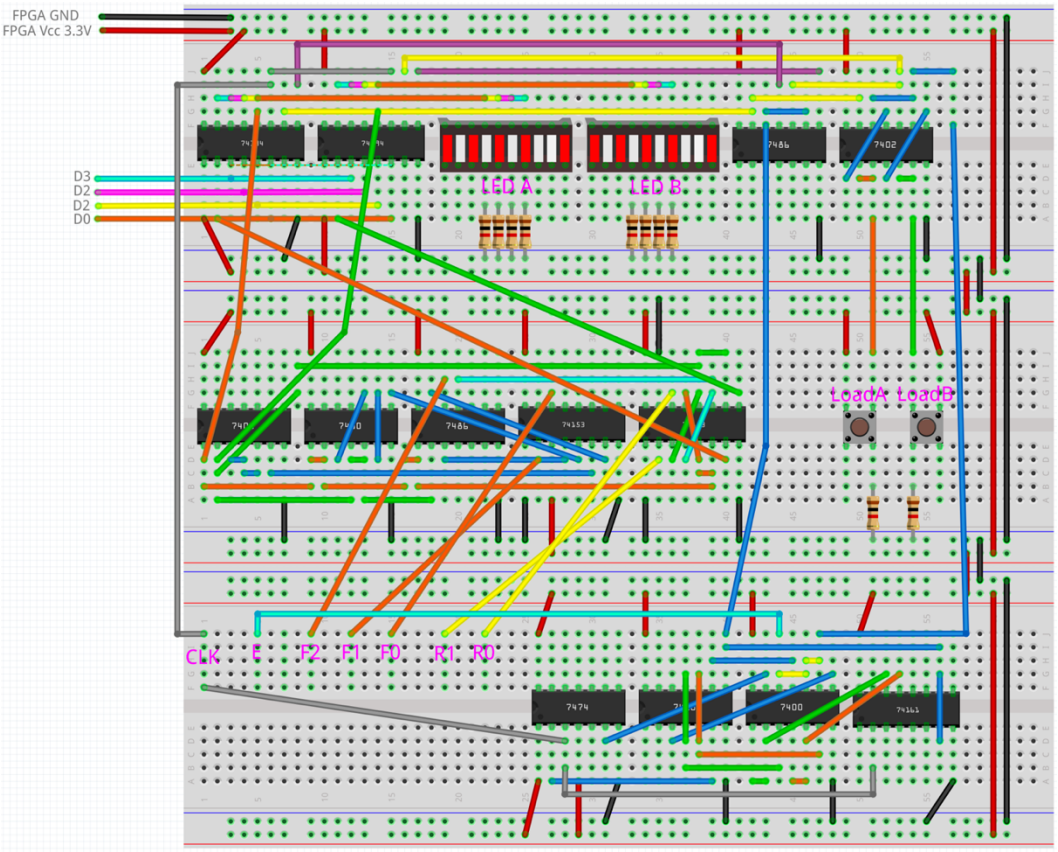


R1	R0	Destination A*	Destination B*
0	0	A	B
0	1	A	F(A,B)
1	0	F(A,B)	B
1	1	B	A

Table 4. Destination A and B based on select signal R1 R0

Lab Documentations (Extra Credit)

Breadboard View



DE10-Lite board Layout (Extra Credit)

Signal	Switch	Function
E	DE-10Lite Switch 9	Execute (Start shifting 4 clock cycles)
F2	DE-10Lite Switch 8	ALU Select
F1	DE-10Lite Switch 7	
F0	DE-10Lite Switch 6	
R1	DE-10Lite Switch 5	Routing Select
R0	DE-10Lite Switch 4	
D3	DE-10Lite Switch 3	4-bit Input for Register A and Register B
D2	DE-10Lite Switch 2	
D1	DE-10Lite Switch 1	
D0	DE-10Lite Switch 0	
LoadA	Left PushButton on breadboard	Load 4-bit Input into Register A
LoadB	Right PushButton on breadboard	Load 4-bit Input into Register B
CLK	SCL on DE-10Lite	Clock signal for sequential components
Vcc	VCC3P3 on DE-10Lite	Power TTL chips and logic ‘1’
GND	GND on DE-10Lite	GND TTL chips and logic ‘0’

Table 5. Signals and their corresponding switches and functions

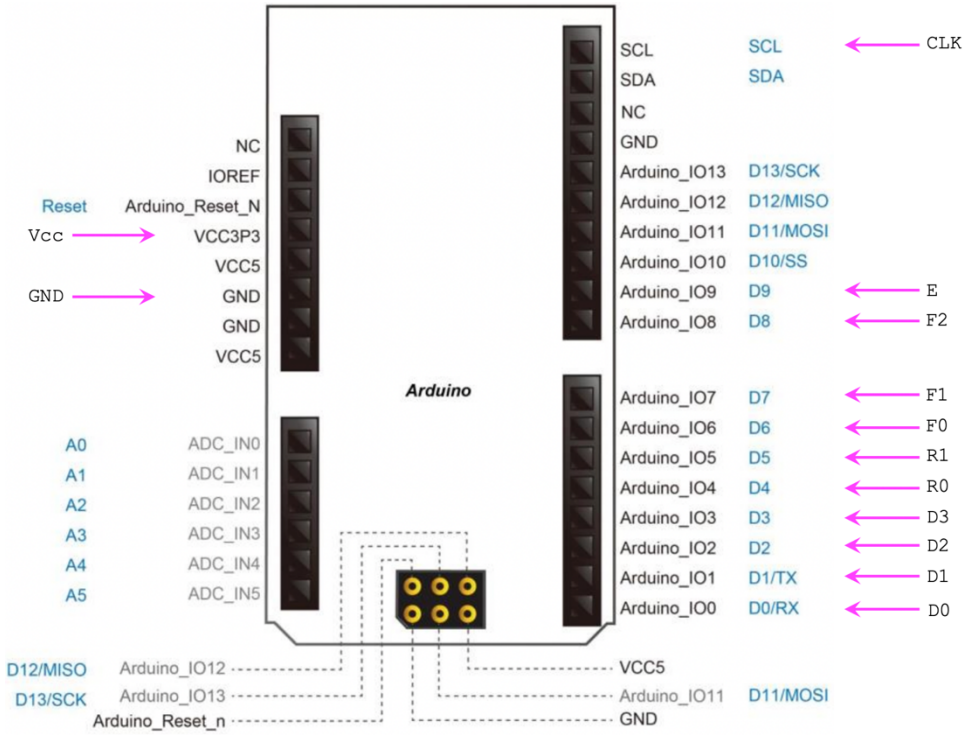


Figure 1. DE10-Lite Board Pin Connections

Component Layout (Extra Credit)

Reference Designator	Type	Description	Place used
U3, U4, U12	SN7400	Quad 2-Input NAND	U3 ALU U4 ALU U12 Control Unit
U9	SN7402	Quad 2-Input NOR	Control Unit
U5, U8	SN7486	Quad 2-Input Exclusive-OR	U5 ALU U8 Control Unit
U11	SN7410	Triple 3-Input NAND	Control Unit
U6, U7	SN74153N	Dual 4-to-1 Multiplexer	U6 ALU U7 Routing Unit
U1, U2	SN74LS194E	4-bit Bidirectional Universal Shift Register	Register Unit
U10	SN7474	Dual D Positive Edge Triggered Flip Flop	Control Unit
U13	SN74163E	4-bit Synchronous Binary Counter	Control Unit

Table 6. TTL chips used to build the physical circuit of the 4-bit Logic Processor

Signal	Function
A3	Display on LED A
A2	Display on LED A
A1	Display on LED A
A0	Display on LED A
B3	Display on LED B
B2	Display on LED B
B1	Display on LED B
B0	Display on LED B
A	Register A Shift Right Output
B	Register B Shift Right Output
C1	Output from Counter Chip
C0	Output from Counter Chip

Table 7. Signal references of Component Layout

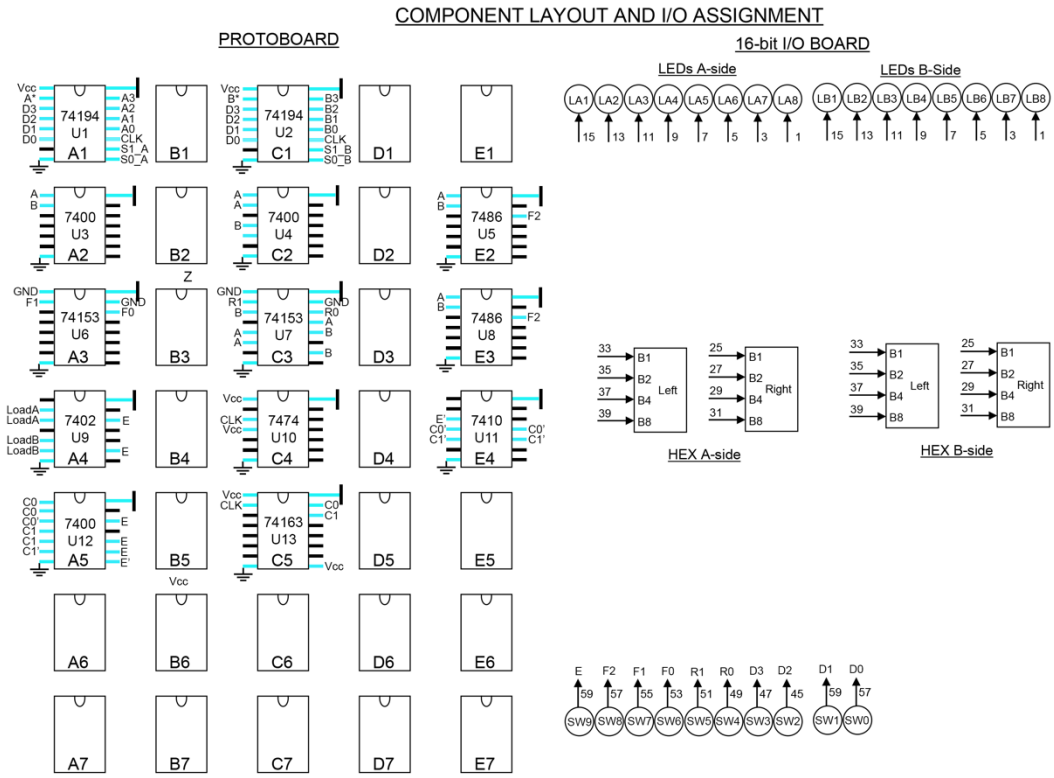


Figure 2. Component Layout and I/O Assignment of the 4-bit Logic Processor physical circuit

Bugs encountered and corrective measures taken

Using the modular design method mentioned in the lab manual, the register unit, ALU, and the routing unit are all confirmed to work individually. Therefore, when these three modules are combined together, they are still working properly which eliminates the time for debugging. However, when the control unit is finished and connected to the entire circuit, the circuit does not work properly as expected. Since the register unit, ALU, and the routing unit are all confirmed working, the problem is suspected to be the wrong wiring of the outputs of the control unit. After carefully checking the wring of the control unit, it seems like all the wirings are correct. Thus, several extra output pins, such as Shift, C_1, C_0, and Q, are added to the output port of the control unit in order to check which signal has the wrong output. After rerunning the testbench, the waveform shows that the signal Q is unknow (X) the whole time. This makes sense because Q signal will then cause Shift, C_1, and C_0 signals unknown shortly after the simulation begins which will further make the entire circuit not functioning as expected. Therefore, the problem is quickly found to be not clearing the register in the testbench. When the register is cleaned before running the simulation, the entire circuit works as expected.

Post-Lab

The initial design for this lab is to build the ALU with an 8-to-1 MUX. However, the above design (4-to-1 MUX and an XOR gate) is optimal than an 8-to-1 MUX because it minimizes the number of IC needed. By designing the circuit modularly, the entire 4-bit logic processor can be simplified to 4 independent modules, and each module can be treated as a black box with inputs and outputs. This way, it was easier to conceptualize the design. In addition, the modular design method can save time in debugging and pinpoint problems quickly when they appear. For example, the ALU is the first module to be completed, and it is then verified to work properly in Quartus and on the breadboard (Extra Credit). Then, the routing unit, the register unit, and chips for hardware design are then also verified to work individually. When all the modules beside the control unit are combined together, everything works properly. Therefore, debugging for the combinational circuit is not necessary which saves lots of time. However, when the control unit is finished and connected to the entire circuit, the shift signal is always displaying unknow (X) in the waveform. Since all other units are already verified to work properly, this is quickly identified to be the problem that the register is not cleared before using it (detailed debugging process in the previous section). If the circuit has been put together without testing and verifying each module, it will cost lots of time for debugging.

The simplest 2 input 1 output circuit that can optionally invert a signal is an XOR gate. The functionality of the XOR gate is shown below in the truth table. According to the ALU functionality table in the above section, the ALU is capable of calculating one of 8 functions: AND, OR, XOR, set all bits to 1, NAND, NOR, XNOR, and set all bits to 0. The last 4 rows are just the inverted version of the first 4 rows. Therefore, the XOR gate is useful for this lab because the ALU can be then constructed with a 4-to-1 MUX and an XOR gate. In this way, the design enables us to avoid the usage of an 8-to-1 MUX which minimizes the number of IC needed because 8-to-1 MUX has more gates than a 4-to-1 MUX.

Select	Input	Output
0	0	0
0	1	1
1	0	1
1	1	0

Input is kept when Select = 0

Table 6. Truth Table of an XOR gate

Select	Input	Output
0	0	0
0	1	1
1	0	1
1	1	0

Input is inverted when Select = 1

Table 7. Truth Table of an XOR gate

A Moore machines can be very lengthy because the output is dependent on only the current state. However, the output of a Mealy machine is dependent not only on the current state, but also on the inputs. Therefore, a Mealy machine usually has a more compact design. Using the Mealy machine in the control unit is a better choice because it requires fewer states which decrease the complexity of the circuit. Moreover, using a counter is also a better solution because it combines the 4 shift states in the Mealy machine into 1 state which further decrease the circuit complexity. Generally, the Mealy machine has less states. Moreover, the Mealy machine requires less logic to decode the outputs which results in less circuit delay because the output is dependent on both the current state and the input. In addition, the Mealy machine reacts faster to inputs which makes it faster than the Moore machine. The Moore machine includes easier traceability due to the output being depended only on the current state. Moreover, the Moore machine is safer to use because it changes states on the clock edge. However, the Moore machine requires more logic to decode the outputs which results in more circuit delay.

Conclusion

Designing the 4-bit logic processor using modular design method makes the lab easier to finish. As stated in previous sections, the most difficult part of this lab is the design of the control unit because both combinational and sequential logics are required in this section. Specifically, designing the FSM and the combinational logic for switching the modes of the registers are two challenging parts of the control unit. Moreover, using the last two bit of the 4-bit Synchronous Binary Counter is a smart trick because it decreases the circuit complexity. For the extra credit, although unit testing each individual chip and figuring out how to use the chip through the datasheet are time consuming, it is really helpful to minimize future debugging time. A particular conundrum faced is choosing between the 74194 and the 74195 chips as the shift registers. After carefully reading the datasheet of both chips, the SN74LS194E chip is more optimal because it has an option to HALT which keeps the original values stored in the registers. In addition, since so many chips are used in designing the 4-bit logic processor, cable(wire) management served to make the design look nice, but more importantly made it easier to understand and as a consequence easier to debug.