ECE 385

Fall 2021

Experiment 3

A Logic Processor

Guanshujie Fu 3190110666
Xiaomin Qiu 3190110717
Lab Section D231 18:00 – 21:00
TA: Yuchuan Zhu

1 Introduction

1.1 Basic Intro

Our circuit implements several logical operations with the use of shift registers, several multiplexers, and some type of counter. The processor will be capable of calculating eight different logical functions, which are AND, OR, XOR and $TRUE$ as well as the inverse of these four functions. The processor also routes the results of those operations in four different ways with appropriate selection signals. A finite state machine, Mearly machine, is implemented for our circuit to serve as the control unit of the design.

1.2 Answers to Pre-lab questions

I. *Describe the simplest (two-input one-output) circuit that can optionally invert a signal (i.e., one input determines if the output is equal to the other input or equal to the other input inverted). Sketch your circuit.*

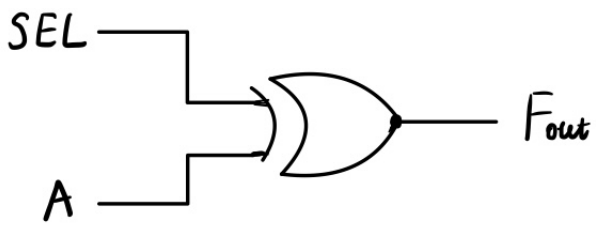


Figure 1: XOR

We can use XOR function to realize this simple circuit, $F_{out} = SEL\ XOR\ A$, where SEL represents selection input and A represents the signal expected to be inverted. The circuit is shown in Figure 1.

II. *Explain how a modular design such as that presented above improves testability and cuts down development time. Propose an approach that could be used to troubleshoot the modular circuit above if it appeared to be completing the computation cycle correctly but was not giving the correct output. (Be specific.)*

With the simplified design shown above, we can make the circuit simpler. Specifically, with the XOR function above, we can cut down some of the components like multiplexer since this circuit can select the input needed.

III. *Discuss the design process of your state machine, what are the tradeoffs of a Mealy machine vs a Moore machine?*

The discussion of State Machine is presented in Conclusion Section.

2 Operation of the logic processor

I. The Control Unit would output S0 and S1 to control the shift of the Registers. First, when the Execute signal becomes 1, the Counter start to count and the flip-flop would output Q', then the value of the value of S can be calculated through some gated based on Q',E,C0, C1 and the value of Q+ can be calculated based on E,C0, C1. Next, as S have been determined, together with the inputs $LOAD_A$ and $LOAD_B$, the mode signal of register S0,S1 can be calculated with S, $LOAD_A$ and $LOAD_B$. $S0 = S + LOAD_A/B$, $S1 = LOAD_A/B$. As $S1S0 = 00$, the registers would do nothing; As S1S0= 01, the registers would Shift right;As S1S0= 11, the registers would load PARALLEL, with means load Dins; As S1S0= 10, the registers would shift left(impossible).

II. For the computation unit, after the output A, B of the registers flip into the unit, they would be calculated with AND, OR, XOR. At the same time, the inputs F2-F0 become the select signals to determine which outcome of the three operations should be chosen. F1,F0 are the select signal of the 4-to-1 multiplexer and F2 is the select signal of the second multiplexer, together they becomes a 3-to-1 multiplexer. The output A, B, F would be flip into the ROUTINE UNIT.

For the ROUTINE UNIT, the select signal R0,R1 would choose A* and B* from the input A,B,F through a multiplexer. As R1R0 = 00, the unit would output A*=A,B*=B ;As R1R0 = 01, the unit would output A*=A,B*=F ;As R1R0 = 11, the unit would output A*=B,B*=A ;As R1R0 = 10, the unit would output A*=F,B*=B. The the A*B*would flip into the register unit again.

3 Description, Block diagram and State machine diagram

I. Written Description:

Control Unit:

For the control unit, it is majorly built with SN74169 Counter, SN7474 Flip-Flops and some Gates. The function of this unit is to generate Mode signal for the Register unit so it could shift the data in the wanted way. It will accept the following inputs: Load A, Load B, Execute, and the clock signal. The Load A and Load B inputs will perform parallel loads from the INPUT port (D3-D0) into the A and B registers. Execute tells the control unit that the select switches and the register contents are ready for execution and that the control unit should begin the computation cycle. The control unit then shifts the register unit the required number of times and halts until the next execution is requested.

Register Unit:

For the Register unit, it is built with two 74LS194A Shift-Registers. Each register could hold 4-bits data and shift the data based on the shift signal. The register has four modes such as parallel shift and serial shift. The contents of these registers should be displayed

so that the contents before and after execution can be inspected. The control of these registers will be provided from the Control Unit while the serial input will be provided from the routing unit.

Computation Unit:

For the Computation unit, it is built with AND, OR, XOR Gates, SN74153 4-to-1 MULTIPLEXER, SN74157 2-to-1 MULTIPLEXER. It will accept as inputs the contents of RegA and RegB, and the function selection inputs F2, F1, F0. The unit could perform A and B, A or B, A xor B operations, and the kind of operation could be chosen by the MULTIPLEXERS. As we need to choose one operation from three, the unit need both 4-to-1 and 2-to-1 MULTIPLEXER to perform this function. The unit will output the logical function $f(A, B)$ specified by $< F2, F1, F0 >$ and will also output the A and B inputs unchanged. The final result $f(A, B)$ will be output as F together with A, B signals. The detailed function of Computation Unit can be seen in the figure below.

Function Selection Inputs			Computation Unit Output
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	1111
1	0	0	A NAND B
1	0	1	A NOR B
1	1	0	A XNOR B
1	1	1	0000

Figure 2: Function Selection

Routine Unit:

For the Routine unit, it is build with a $SN74193$ 4-to-1 MULTIPLEXER. It accepts inputs A, B and $f(A, B)$. Based on the routing selection inputs R1 and R0, which selects the input needed to be stored and the registers that accept the input. The detailed function of Routine Unit can be seen in the figure below.

Routing Selection		Router Output	
R1	R0	A*	B*
0	0	A	B
0	1	A	F
1	0	F	B
1	1	B	A

Figure 3: Routine Selection

II. Block Diagram:

The block diagram is shown below, which is the one in manual. We think it is detailed and contains the information needed for this section.

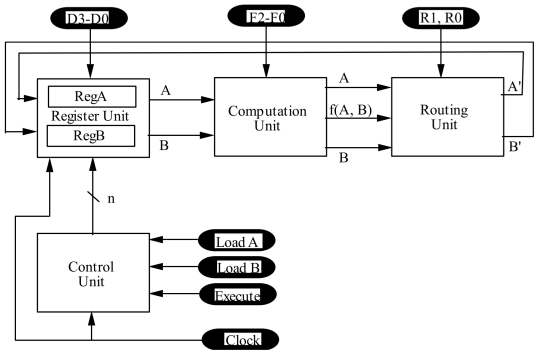


Figure 4: Block diagram

III. State Machine Diagram:

We designed a Mealy machine for this lab as shown below. We use the same picture in the lab manual and make some detailed explanations to indicate the lable of states and arcs.

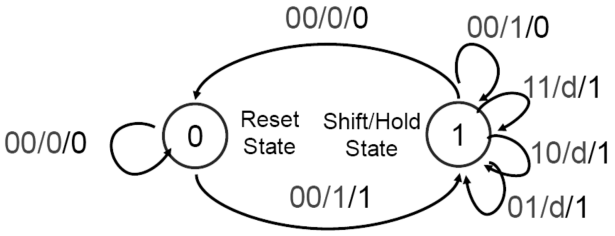


Figure 5: Mealy State Machine

The State 0 represents *Reset* state while the State 1 represents $Shift$ state (or $Halt$ State). In the case, $Q = 0$ indicates *Reset* state and $Q = 1$ indicate $Shift$ or $Halt$ states in flip_flop. As for the arc, the input and output for each state is determined by $COUNTER/EXECUTE/SHIFT$ signals.

4 Design steps and Detailed circuit schematic diagram

I. K-maps or Truth tables

The K-maps for S, Q can be seen from figures (a) and (b) in page 5. The logical functions for $C1^+$ and $C0^+$ are redundant since they are implemented in the counter.

II. Design Steps Taken:

For this circuit, the part in which I consider multiple implementations is the Computation Unit. In our circuit, we use two multiplexers to realize the functions. The computation part actually can be simplified with the use of logical functions. As we

S		EQ			
		00	01	11	10
C1C0	00	0	0	0	1
	01	d	1	1	d
	11	d	1	1	d
	10	d	1	1	1
S = EQ' + C0 + C1					

(a) S

Q+		EQ			
		00	01	11	10
C1C0	00	0	0	1	1
	01	d	1	1	d
	11	d	1	1	d
	10	d	1	1	d
Q+ = E + C0 + C1					

(b) Q^+

C1+		EQ			
		00	01	11	10
C1C0	00	0	0	0	0
	01	d	1	1	d
	11	d	0	0	d
	10	d	1	1	d
C1+ = C1'C0 + C1C0'					

(c) $C1^+$

C0+		EQ			
		00	01	11	10
C1C0	00	0	0	0	0
	01	d	0	0	d
	11	d	0	0	d
	10	d	1	1	d
C0+ = C1C0'					

(d) $C0^+$

have shown in Figure 2, the function selection inputs are F_2, F_1 and F_0. However, since the last four functions are just the inverse of the first four functions, it is possible to use only one multiplexer to generate the correct output. Assume that the output of the first four functions is F_{prim}, we can then have the expected output $F_{out} = F_{prim}\ XOR\ F_2$.

We do not use the simplified circuit due to the debug consideration when design our circuit.

III. Detailed Circuit Schematic

The designed circuit schematic of our circuit can be seen in III. The gate level schematics are presented in the figures in page 7.

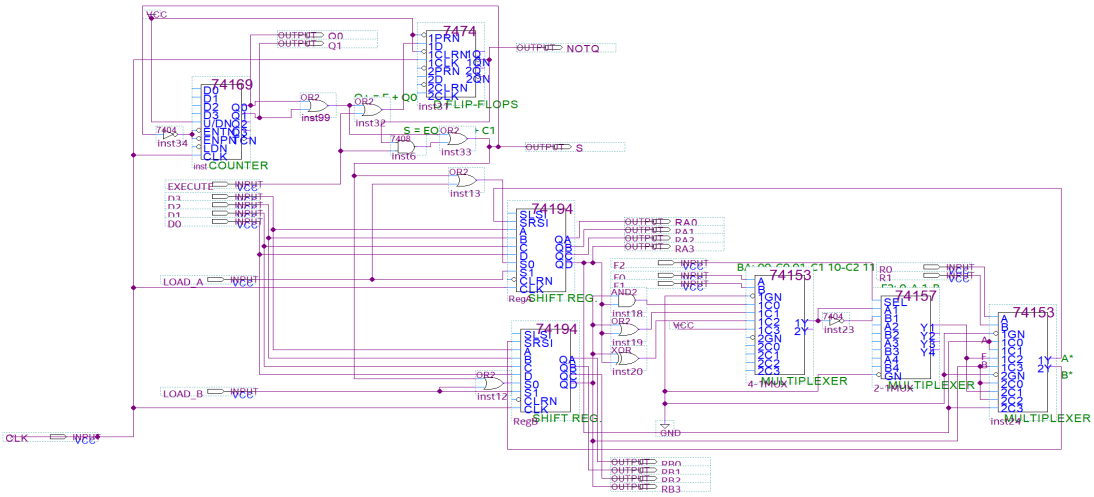
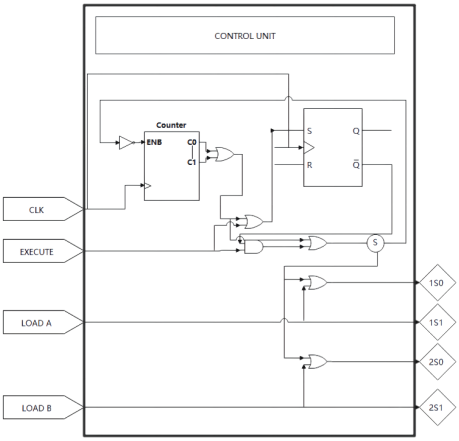


Figure 6: Circuit diagram

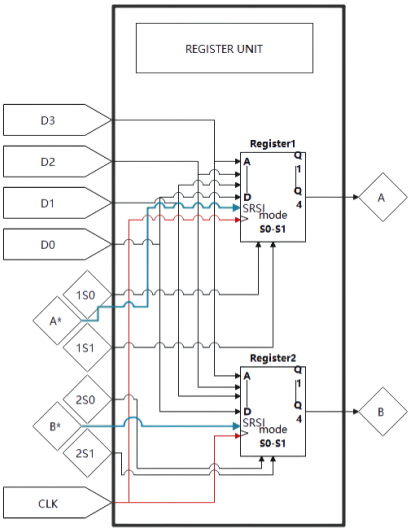
5 Description of Bugs encountered and Corrective measures

I. The first bug we encountered occurs in the counter part. According to the Moore Machine we design, we need the counter to pause and stay in 00 at the *Reset* state. We then check the waveform of the counter to find the signals needed to pause the counter. The waveform is shown in Figure 7. It is clear that when the signals $\overline{P}$ and $\overline{T}$ are triggered, the counter will be paused and stayed in the previous state. We then need to figure out which signal in our circuit should be fetched into $\overline{P}$ and $\overline{T}$.

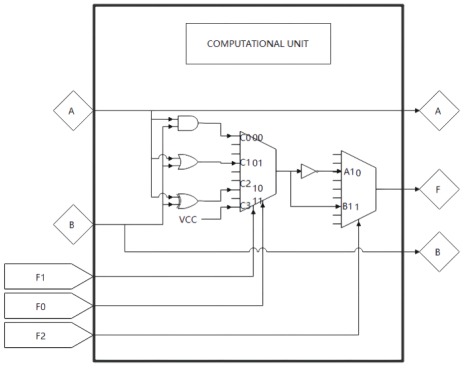
We first connect signal $EXECUTE$ directly to the $\overline{P}$ and $\overline{T}$. The circuit works correctly at first but the counter does not pause when the instruction is finished. To solve the bug, we checked the Moore machine several times and changed the $EXECUTE$ to $SHIFT$ as the counter should be controlled by the $SHIFT$ signal rather than $EXECUTE$.



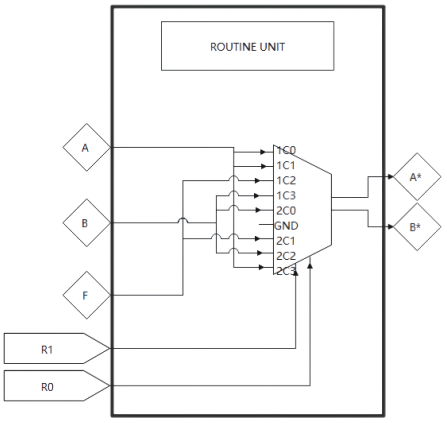
(a) Control Unit



(b) Register Unit



(c) Computation Unit



(d) Routine Unit

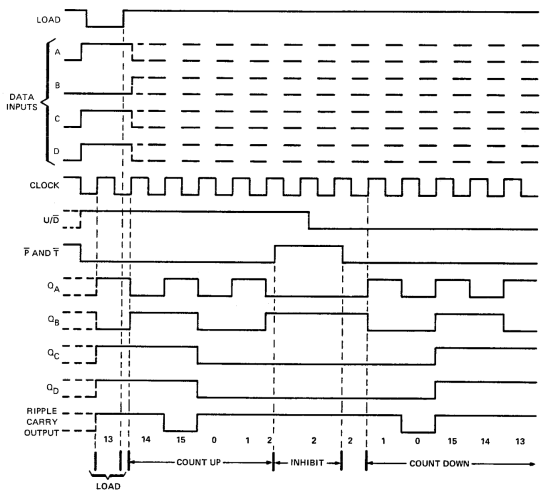


Figure 7: Waveform of Counter $SN74LS169A$

6 Conclusion

6.1 Summary of Lab

In this lab, we designed a circuit based on a Moore Machine which could implement several logical operations. All functions of our circuit work successfully and we generate the correct waveform as shown below. This experiment gives us an insight into the Finite State Machine and the differences between Mealy and Moore Machine. The detailed comparison will be presented in the next section. We also learnt that with the use of logical function like XOR, we can simplify our design and improve testability.

6.2 Answer to Post-lab Questions

The main difference of Moore and Mealy machines falls on how they determine the next state. Specifically, the state of Moore machine is only determined by its previous state while the state of Mealy machine is related to previous state and the input signals.

In our circuit for this experiment, we need to control the execution of the counter and it indicates that we need to use both current state and input signals to determine the next state. Hence, we designed a Mealy machine. It requires less states compared to the Moore Machine and thus less gates will be needed for the circuit. Though Moore Machine is more straightforward and its logical design is simpler , the control unit of our circuit is simplified with Mealy Machine, which is more important in complex circuit design.

7 Waveform Generated

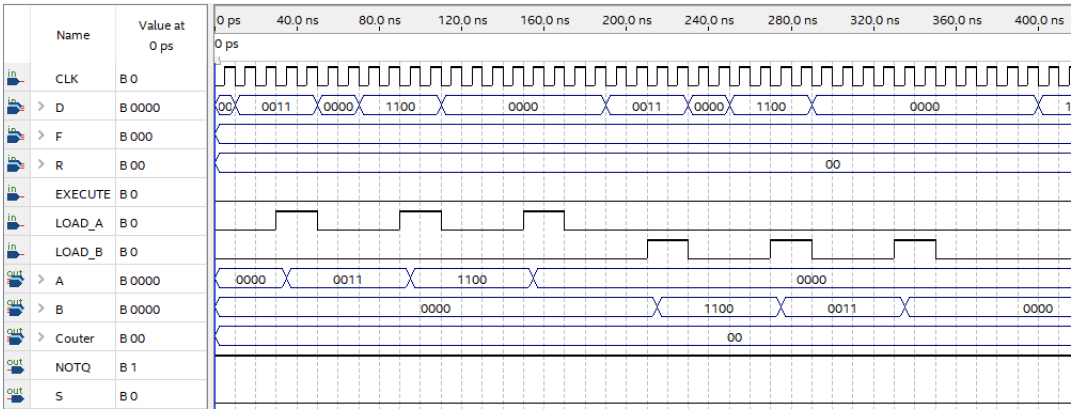


Figure 8: Waveform $0ns$ – $400ns$

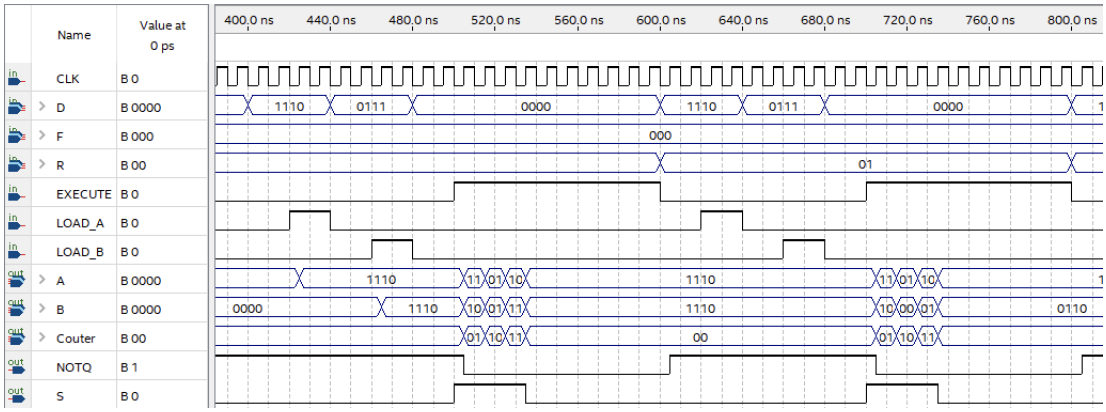


Figure 9: Waveform $400ns$ – $800ns$

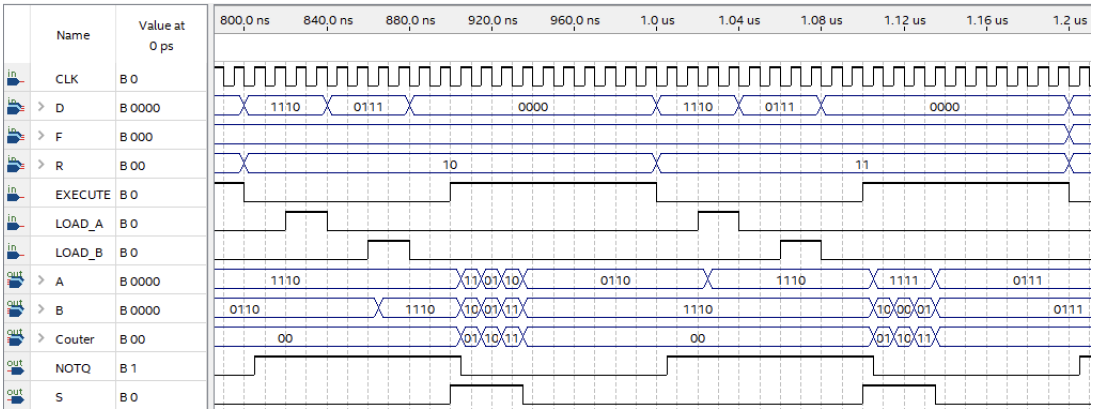


Figure 10: Waveform $800ns$ – $1200ns$

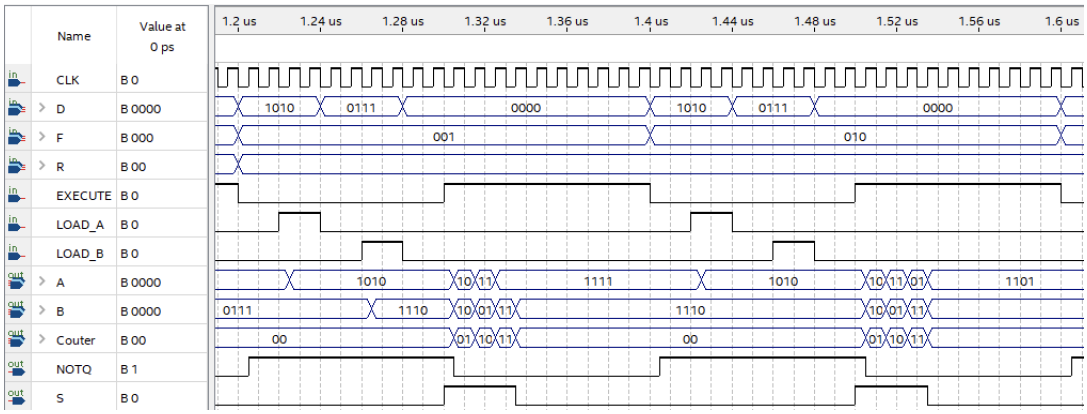


Figure 11: Waveform $1200ns$ – $1600ns$

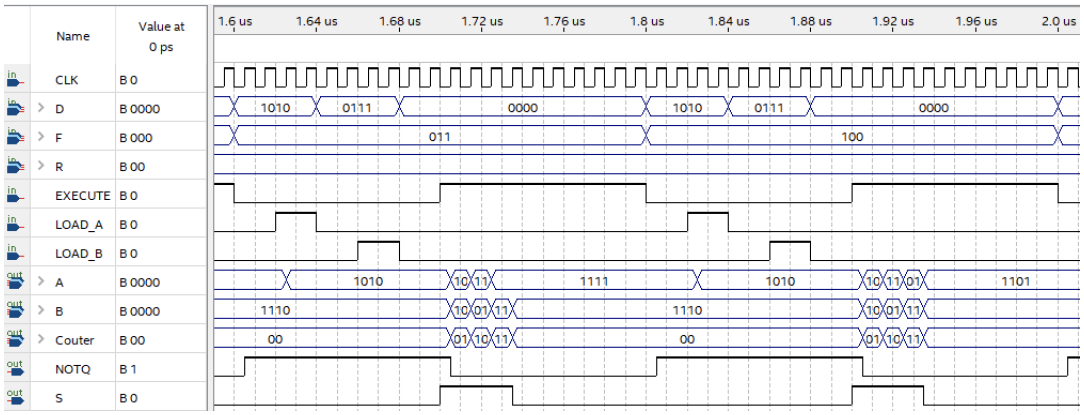


Figure 12: Waveform $1600ns$ – $2000ns$