Liam Rohrer

Wednesday lab session 12:00-3:00

Experiment #3: Flip-Flops Used in Digital Logic Design

Aim: Practice the implementation of clocked master-slave, SR, JK, and D flip-flops and multiplexors.

Results:

1. First, we wired the NOR latch (SR flip-flip), shown below in figure 1, using the 74LS02 IC package.

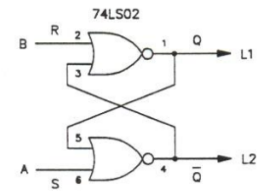
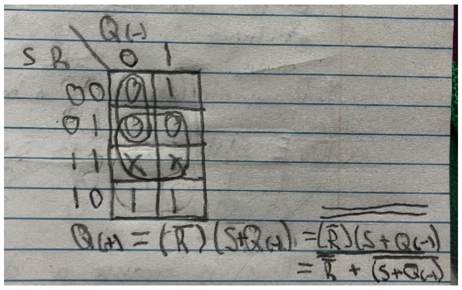


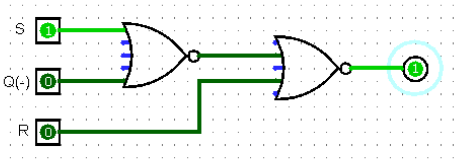
Fig. 1

2. We then applied power to the switches and recorded our results in the truth table below. When we applied the illegal state of (AB) we observed that both outputs (Q and Q') went low. We proceeded to make a Karnaugh map and extract the minimal Product of Sums (POS) expression for Q. We also converted the minimal POS expression to use only NOR gates and connected up the circuit. We finished by checking that it functions correctly with a truth table.

S	R	Q(-)	Q(+)	Q'(+)
0	0	0	0	1
0	0	1	1	0
0	1	0	0	1
0	1	1	0	1
1	0	0	1	0
1	0	1	1	0
1	1	0	X	0
1	1	1	X	0



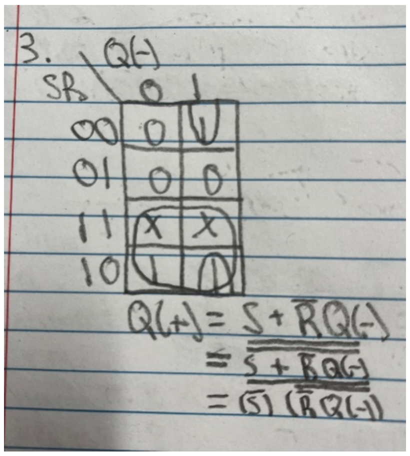
Q(-)
S R
0 1
00 0 1
01 0 0
11 X X
10 1 1
$Q(+) = (\overline{R})(S+Q(-)) = \overline{\overline{(\overline{R})(S+Q(-))}}$
$= \overline{\overline{\overline{R}}} + \overline{(S+Q(-))}$



R	(S + Q(-))'	(R + (S+Q(-))')'
0	1	0
0	0	1
1	1	0
1	0	0
0	0	1

0	0	1
1	0	X
1	0	X

3. We used the same Karnaugh map from the step above but this time extracted the minimal Sum of Products (SOP) expression, converted it to use only NAND gates, connected up the circuit using the 74LS00 IC package, and checked that it functions as intended with a truth table.



3.

SR \ Q(-)	0	1
00	0	1
01	0	0
11	X	X
10	1	1

$$Q(+) = S + \bar{R}Q(-)$$
$$= \overline{\overline{S + \bar{R}Q(-)}}$$
$$= \overline{(\bar{S})(\overline{\bar{R}Q(-)})}$$

S	R	Q(-)	S’	(R’Q(-))’	(S’(R’Q(-)’)’
0	0	0	1	1	0
0	0	1	1	0	1
0	1	0	1	1	0
0	1	1	1	1	0
1	0	0	0	1	1
1	0	1	0	0	1
1	1	0	0	1	X
1	1	1	0	1	X

4. For this step we connected up the circuit, show below in figure 2, with the 74LS00 IC package.

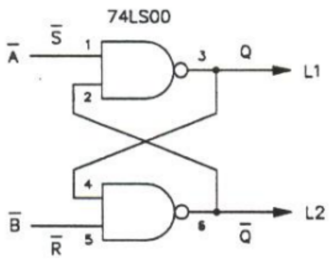


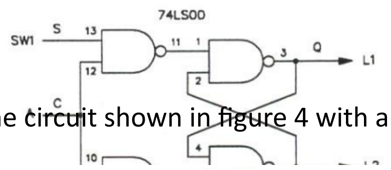
Fig. 2

5. We then supplied power to the circuit and kept track of its outputs with a truth table.

S	R	S’	R’	Q(-)	Q(+)	Q’(+)
1	1	0	0	0	0	1
1	1	0	0	1	1	0
1	0	0	1	0	0	1
1	0	0	1	1	0	1
0	1	1	0	0	1	0
0	1	1	0	1	1	0
0	0	1	1	0	X	X
0	0	1	1	1	X	X

With inverted inputs, the circuit functions the same as the NOR latch from above.

6. For the following step we wired up the circuit shown in figure 3. When the C input is set to 0 the outputs don’t change no matter what the inputs are. When C is set to 1 the outputs can be changed and the circuit functions as a basic NOR SR latch as a result of the inputs. This allows it to store 1 bit of data.



7. Next we connected up the circuit shown in figure 4 with a 74LS00 IC package.

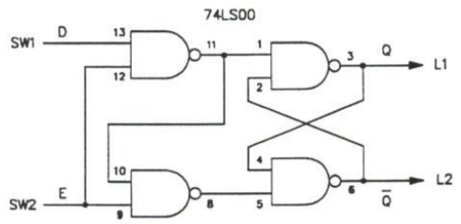


Fig. 4

8. When E is high, the output is set to the save value as the input. When E is low the output does not change regardless of the input value.

9. We then connected up the circuit shown to the right with a 74LS75 IC package in figure 5.

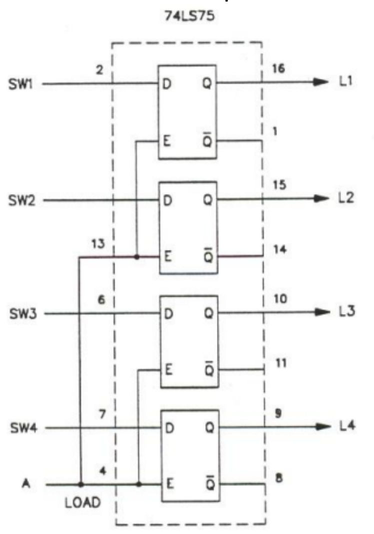


Fig. 5

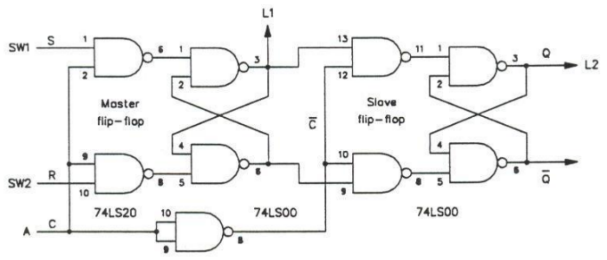
10. To demonstrate how this circuit can be used to store 4 bits of data, recorded the states of L1, L2, L3, L4 for while Enable was high, then changed Enable to low, changed the input of the 4 switches, and demonstrated how the output stays the same.

E	SW1	SW2	SW3	SW4	L1	L2	L3	L4
1	1	0	1	0	1	0	1	0
0	0	0	0	0	1	0	1	0

11. For this step we connected up the circuit shown below in figure 6 which uses two 74LS00 IC packages and one 74LS20 IC package to create a Master-Slave RS flip-flop. The inputs of the master flip-flop only effect the slave flip-flop on the transition of the clock input from high to low because the inputs of the slave flip-flop come from the outputs of the last time the master flip-flop was enabled. This makes it so when the master flip-flop output changes as E is high, the slave input will then be affected by that change when E goes low, making E' high and enabling the slave flip-flop.

Master Input			Master Output/Slave Input			Slave Output	
S	R	C	L1	L1'	C'	L2	L2'
0	0	0	0	1	1	0	1
0	0	1	1	0	0	1	0
0	1	0	0	1	1	0	1
0	1	1	0	1	0	0	1
1	0	0	1	0	1	1	0
1	0	1	1	0	0	1	0

1	1	0	X	X	1	X	X
1	1	1	X	X	0	X	X



12. We then converted the Master-Slave RS flip-flop to a Master-Slave JK flip-flop by connecting the outputs of the circuit to the inputs as shown below. When J and K are high, the output of the circuit changes only when the clock input goes from high to low because the master flip-flop toggles when the clock goes high, which changes the J and K inputs of the slave flip-flop. Then when the clock input goes back low, the slave flip-flop is enabled and the new inputs go into effect, toggling the output of the circuit.

13. We proceeded to create a truth table displaying the outputs of the circuit based on the J input, K input, and previous output. This is shown below.

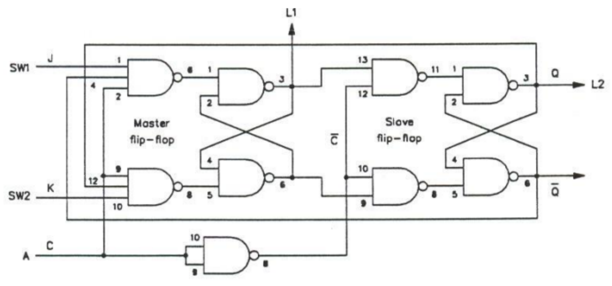
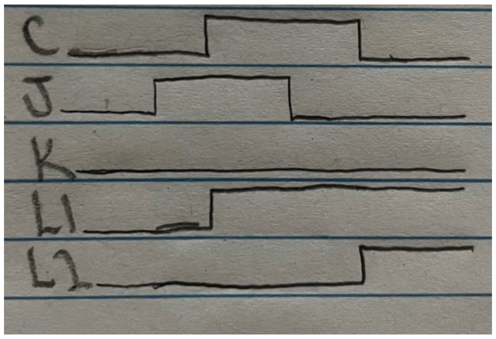


Fig. 7

J	K	Q(-)	Q(+)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	X
1	1	1	X

14. Using the circuit above we created a waveform diagram. The diagram demonstrates starting at J=0, K=0, changing J to 1, pressing the clock button, changing J back to 0, then releasing the clock button.



15. We connected up the circuit shown below using the 74LS109A IC package. This circuit has an active high J input and an active low K input along with PRE' and CLR' inputs.

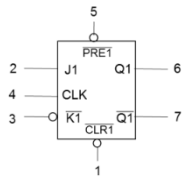


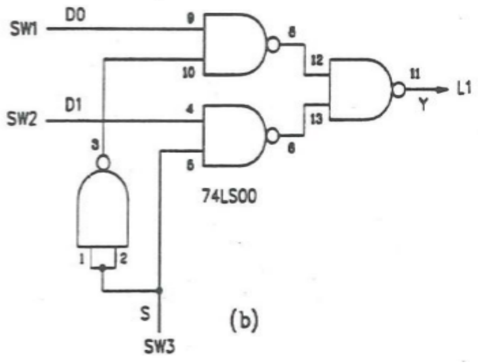
Fig 8

16. In the circuit above, the J and K inputs do not affect the asynchronous operation of the flip-flop because the PRE' and CLR' inputs have priority over the J and K inputs. This makes it so even if J is set high and K is set low, the circuit will output low if CLR' is set low.

17. For this step, we connected up the 2-line-to-1-line multiplexor shown below using the 74LS00 IC package. We then found the Boolean expression for output Y as a function of S, D0, and D1, and checked the function of the circuit with a truth table.

Y = S'D0 + SD1

D0	D1	S	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1



18. We then considered an 8-to-1 multiplexor and found the Boolean expression for which output (D0-D7) is selected based on inputs S2,S1, and S0.

Y = D0S2'S1'S0' + D1S2'S1'S0 + D2S2'S1S0' + D3S2'S1S0 + D4S2S1'S0' + D5S2S1'S0 + D6S2S1S0' + D7S2S1S0

19. Next, we connected up the circuit shown below in figure 11 using the 74LS151 IC package. We then checked all 8 outputs for each of the 8 combinations of the 3 selector inputs. This allowed us to find which input line (D0-D7) was being selected from each combination of selector lines. The results are displayed in the table below.

C	B	A	Line Selected
0	0	0	D0
0	0	1	D1
0	1	0	D2
0	1	1	D3
1	0	0	D4
1	0	1	D5
1	1	0	D6
1	1	1	D7

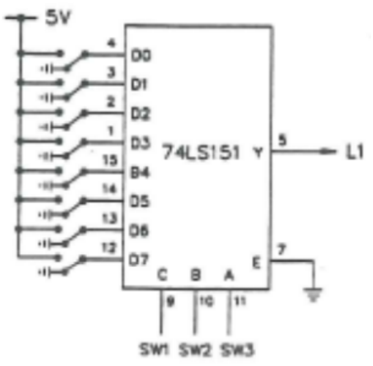


Fig 11

20. For the final step, we modified our circuit from step 19 to appear as shown below in figure 12. We then checked each combination of selector inputs and recorded the outputs in the table below.

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

0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

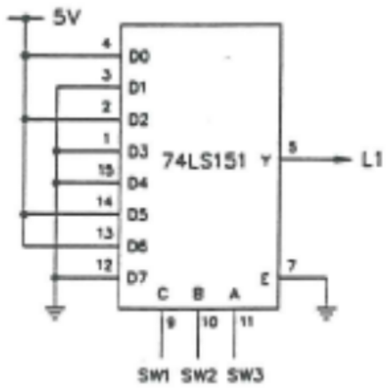


Fig 12

Discussion: Our experiments went smoothly and no errors are expected. Any errors would most likely be a result of small mistakes in wiring circuits or issues with wire connections.

Conclusion: In this experiment we gained practice with basic elements of memory and how they can be implemented. The subtle differences in the operation of different flip-flops and latches, such as the master-slave SR flip-flop and the master-slave JK flip-flop, became more clear as we put them into practice. Potential issues that can arise with such operations were also made more apparent, such as accidentally creating an oscillating circuit. Finally, we learned about multiplexors and saw clearly how they operate to select one out of a number of input values.