CDA 3201L - Thursday (3:30 - 4:45PM) Section 005
Lab #05 - Sequential Logic Circuits (1)
10/22/2016 - Steven Romeiro, Jennyfer Munoz

Purpose & Objectives:

Lab 5 is the first lab that introduced us to Sequential Circuits. The goal behind the following builds served to familiarize ourselves with sequential logic that relies on a clock. Combinational logic is still the basis of the components used, however, the addition of a clock allows this logic to be carried out in a sequence. The following builds introduces us to memory. Part A allowed us to implement a J/K flip-flop and learn the IC layout. Part B required us to implement a D flip-flop. Part C asked us to configure a J/K as a D flip-flop. Part D had us use a D flip-flop as a Toggle. And lastly, Part E had us build a modified version of the SR Latch.

Components Used:

Name	Type	Quantity
74LS08	AND IC	1
74LS02	NOR IC	1
74LS04	Inverter IC	1
74LS86	XOR	1
74LS74	DFF	1
74LS109	JKFF	1
470 Ω Resistor	Resistor	10
LED	Red LED	10
Power Supply	5v	1
Frequency Generator	Wavetek 4MHz	1
Wire Kit	Assorted	1

Description:

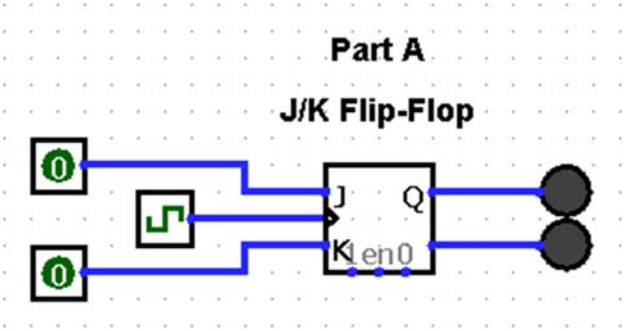
Part A:

The implementation of a JK flip-flop was different than what we learned from theoretical practice in lecture. For starters, the JK chip has K inverted (K'). There were also lines for "Clear" and "Preset" that required a constant high input to allow proper operation of the IC. A function generator was used at 2MHz to allow for a stable rising edge trigger clock.

Truth Table:

J	K	Q	Q'
0	0	Q	Q'
0	1	0	1
1	0	1	0
1	1	Q'	Q

Diagram:



Test Cases:

J = 0, K = 1 → Q = 0, Q' = 1

J = 1, K = 0 → Q = 1, Q' = 0

Part B:

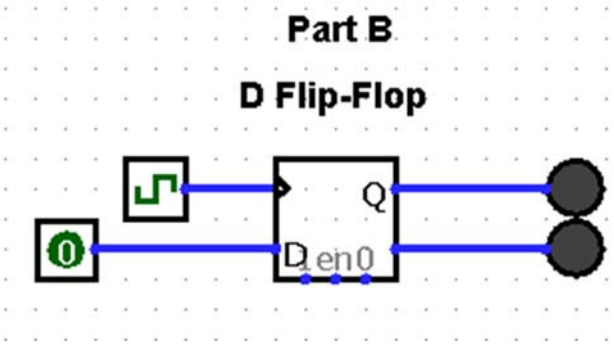
The implementation of a D flip-flop, unlike part A, wasn't very different from what we learned in theory. A function generator was used at 2MHz to keep a steady rising edge trigger clock while different inputs were tested. In each case, the desired outcome was stored as predicted. As with the previous IC, this chip also required a constant high input for "Preset" and "Clear" for normal operations.

Truth Table:

DFF

Clock	D	Q	Q'
0	0	Q	Q'
0	1	Q	Q'
1	0	0	1
1	1	1	0

Diagram:



Test Cases:

D = 0 → Q = 0, Q' = 1

D = 1 → Q = 1, Q' = 0

Part C:

Implementing a D flip-flop from a JK flip-flop required some work. We began with truth tables for both flip-flops. We then compared the outputs of both tables and found that similar results occurred for a slight modification of the inputs. We noticed that the matching outputs were achieved when we took J as one input and the complement of J as the other. From here it was simply a matter of adding an inverter to the branch of J that lead to the K input on the IC.

Truth Table:

JKK

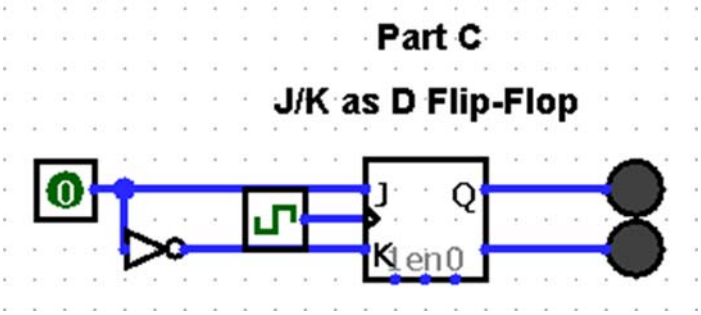
Clock	J	K	Q	Q'
0	0	0	Q	Q'
0	0	1	Q	Q'
0	1	0	Q	Q'
0	1	1	Q	Q'
1	0	0	Q	Q'
1	0	1	0	1
1	1	0	1	0
1	1	1	Q	Q'

DFF

Clock	D	Q	Q'
0	0	Q	Q'
0	1	Q	Q'
1	0	0	1
1	1	1	0

Clock	J	J'	Q	Q'
0	0	1	Q	Q'
0	1	0	Q	Q'
1	0	1	0	1
1	1	0	1	0

Diagram:



Test Cases:

J = 1, J'= 0 → Q = 1, Q' = 0

J = 0, J'= 1 → Q = 0, Q' = 1

Part D:

For this section we again began with building truth tables for each flip-flop. We compared the the DFF inputs with the TFF previous state Q. We also compared DFF output Q' (not Q) with the

next state of the TFF Q+ and found them to be similar. Q' (not Q) produced outputs very similar to the next state in the TFF. By routing Q' (not Q) back to the input D we noticed that you could toggle the output with every clock tick. What we needed now was a way to control this toggle. We added gates and an input that allows one to allow or disallow toggling to happen.

Truth Table:

DFF

Clock	D	Q	Q'
0	0	Q	Q'
0	1	Q	Q'
1	0	0	1
1	1	1	0

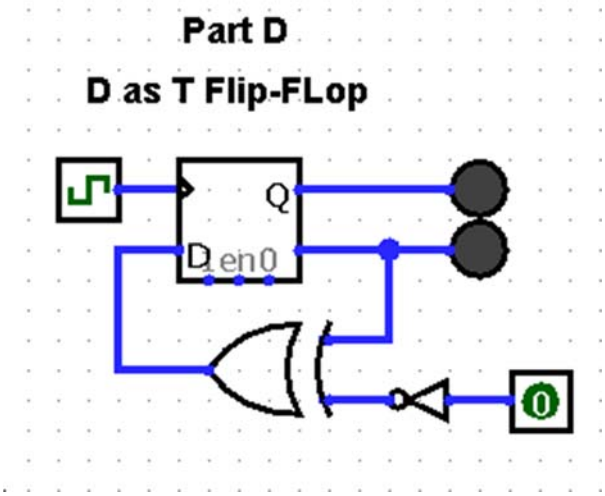
TFF

Q	T	Q_+
0	0	0
0	1	1
1	0	1
1	1	0

Prev

Next

Diagram:



Test Cases:

D = 0, Q = 0, Q' = 1, Q+ = 1
D = 1, Q = 1, Q' = 0, Q+ = 0

Part E:

Building a typical SR Latch was the first step of part E. Once the default SR Latch was built we were asked to modify it to produced a certain output. Once again we constructed truth tables for the regular SR Latch and the Modified SR Latch. We made comparisons between inputs and

outputs to locate the changes. The changes were with inputs 01, 10 and 11 respectively. The modification required us to invert the outputs of regular SR on inputs 01 and 10. Once the inverter was added to inputs S and R it produced the desired modified/inverted outputs. The last modification to tackle was with input 11. The modified SR Latch needed to behave the same whether the inputs were 00 or 11. We added a XOR gate after the inverted SR inputs to detect this. If detected, the XOR gate will produce a 0 output and force a low output on the AND gates that SR inputs were attached to. After this final modification was made, the new modded SR Latch produced the desired outputs displayed on the table.

Truth Table:

SRFF

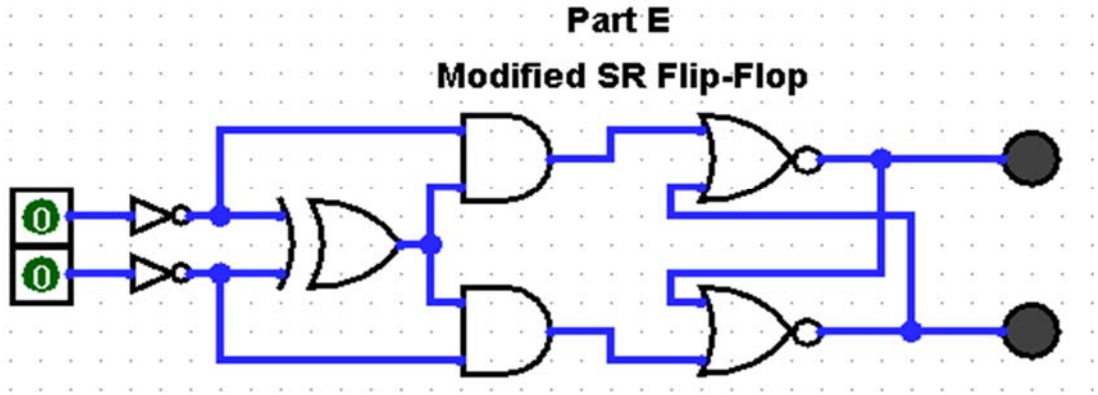
S	R	Q	Q'
0	0	Q	Q'
0	1	0	1
1	0	1	0
1	1	?	?

Mod SRFF

S	R	Q	Q'
0	0	Q	Q'
0	1	1	0
1	0	0	1
1	1	Q	Q'

'

Diagram:



Test Cases:

S = 0, R = 0 → Q = Q, Q' = Q'

S = 1, R = 0 → Q = 1, Q' = 0

S = 0, R = 1 → Q = 0, Q' = 1

Discussion & Conclusion:

Sequential circuits now enable us to store binary information. This ability to store information is the building blocks of memory. With access to stored information, we can now build much more

complex systems that won’t simply perform a single function. The transformation of different flip-flops to other flip-flops means we can rely fewer chips to build a system. The J/K flip-flop has shown to be quite a universal storage device. It can be made to operate like other flip-flops by simply modifying its inputs. This allows us to create any storage device using simply JK flip-flops. The ability to rely on stored information means we can now use stored states as inputs to new systems. This also opens the door for conditional operations that require a certain state to be detected and stored before performing the next function. Sequential loops are now a possibility as well, where the previous state is stored and added with the next state to perform more sophisticated functionality.