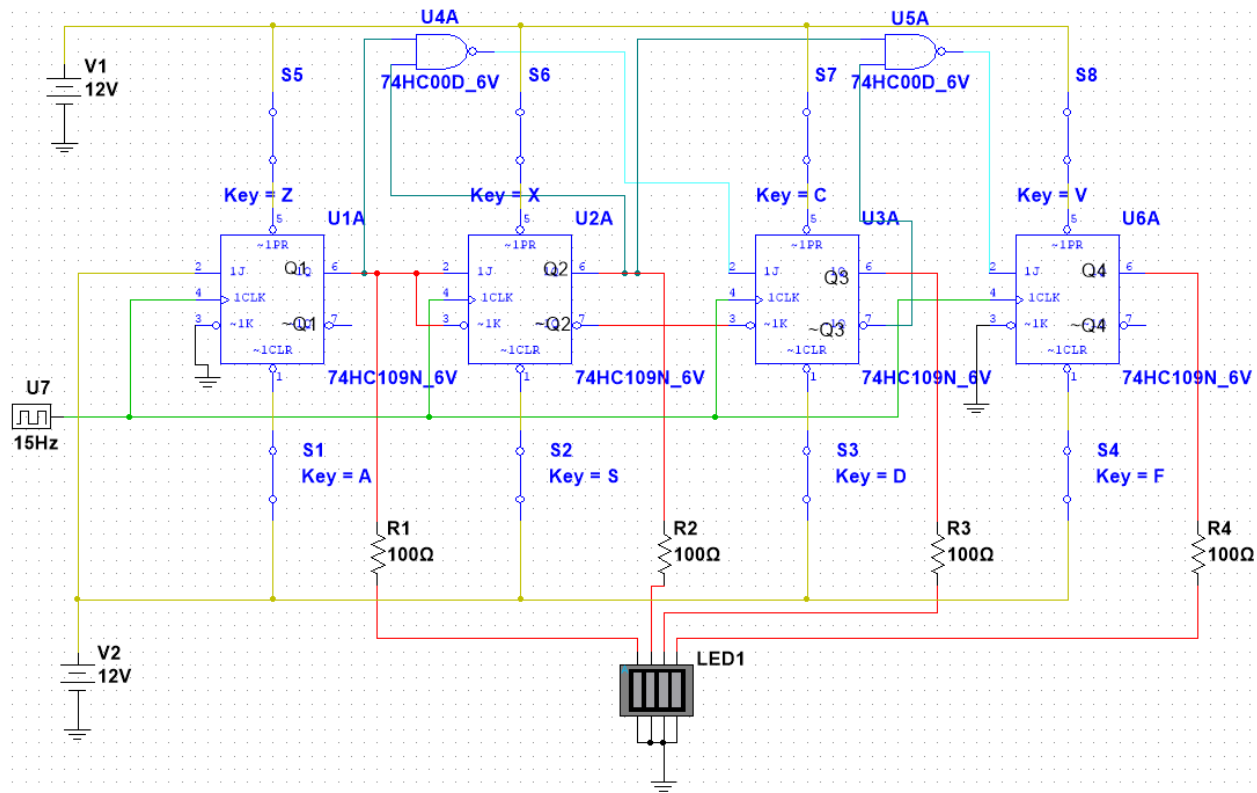


Lab 7

Introduction

In this lab, I will be demonstrating an analytical, digital, and physical solution to a circuit involving the use of JK flip-flops to create a finite state machine. My analytical solution involves determining the outputs in using the J and K inputs and concluding the bit value by the JK flip-flop truth table. The digital solution includes replicating the circuit in Multisim and using the logic analyzer to get the timing diagram for each input. Lastly, for the physical solution, I created a clock signal generator which formed the output to the main circuit. The output of each flip-flop could be seen in the form of the four LEDs turning on and off.



Analytical Solution

In order to know what the outputs of each JK flip-flop is going to be, we shall first derive a value or formula for each input.

The first JK flip-flop has relatively simple inputs; J1 is connected to the high voltage and $\sim K1$ is connected to the ground which gives us:

$$J_1 = 1$$
$$\sim K_1 = 0$$

The second flip-flop has both the J and K inputs connected to the output of the first flip-flop.

$$J_2 = Q1$$
$$\sim K_2 = Q1$$

Notice that there is a NAND gate which accepts the outputs of the first and second flip-flop as its inputs and gives an output which is the input of the third flip-flop.

$$J_3 = \overline{Q1 * Q2}$$
$$\sim K_3 = \overline{Q2}$$

Finally, the fourth flip-flop takes in the output of another NAND gate which is connected to the second flip-flop output and NOT of the third flip-flop output. Its other input is connected to the ground.

$$J_4 = \overline{Q2 * \overline{Q3}}$$
$$\sim K_4 = 0$$

We shall now use a “truth table” for the JK flip-flop which tells its response to all possible inputs.

J_n	$\sim K_n$	Q_{n+1}
0	1	Q_n (hold)
0	0	0 (reset)
1	1	1 (set)
1	0	$\sim Q_n$ (toggle)

I have divided every output of the flip-flops into separate tables in order to show the method I used to determine the outputs and to avoid the mistake of using the current states as an input. For example, it would be wrong to use Q1 to determine Q2. Note that q represents the current state, and Q represents the new state.

The first flip-flop has constant inputs, so it results in a toggle state every time.

q1	q2	q3	q4	J1	~K1	Q1
0	0	0	0	1	0	1
0	0	0	1	1	0	1
0	0	1	0	1	0	1
0	0	1	1	1	0	1
0	1	0	0	1	0	1
0	1	0	1	1	0	1
0	1	1	0	1	0	1
0	1	1	1	1	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	1	0	0
1	0	1	1	1	0	0
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	1	0	0

The second flip-flop has both inputs as Q1 so it will have half of the values as set and half the values as reset.

q1	q2	q3	q4	Q1	J2	~K2	Q2
0	0	0	0	1	1	1	1
0	0	0	1	1	1	1	1
0	0	1	0	1	1	1	1
0	0	1	1	1	1	1	1
0	1	0	0	1	1	1	1
0	1	0	1	1	1	1	1
0	1	1	0	1	1	1	1
0	1	1	1	1	1	1	1
1	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0
1	0	1	0	0	0	0	0
1	0	1	1	0	0	0	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	0	0
1	1	1	0	0	0	0	0
1	1	1	1	0	0	0	0

The third flip-flop has the NAND of Q1 and Q2 and the NOT of Q2, so it gives us half reset and half set outputs.

q1	q2	q3	q4	Q1	Q2	J3	~K3	Q3
0	0	0	0	1	1	0	0	0
0	0	0	1	1	1	0	0	0
0	0	1	0	1	1	0	0	0
0	0	1	1	1	1	0	0	0
0	1	0	0	1	1	0	0	0
0	1	0	1	1	1	0	0	0
0	1	1	0	1	1	0	0	0
0	1	1	1	1	1	0	0	0
1	0	0	0	0	0	1	1	1
1	0	0	1	0	0	1	1	1
1	0	1	0	0	0	1	1	1
1	0	1	1	0	0	1	1	1
1	1	0	0	0	0	1	1	1
1	1	0	1	0	0	1	1	1
1	1	1	0	0	0	1	1	1
1	1	1	1	0	0	1	1	1

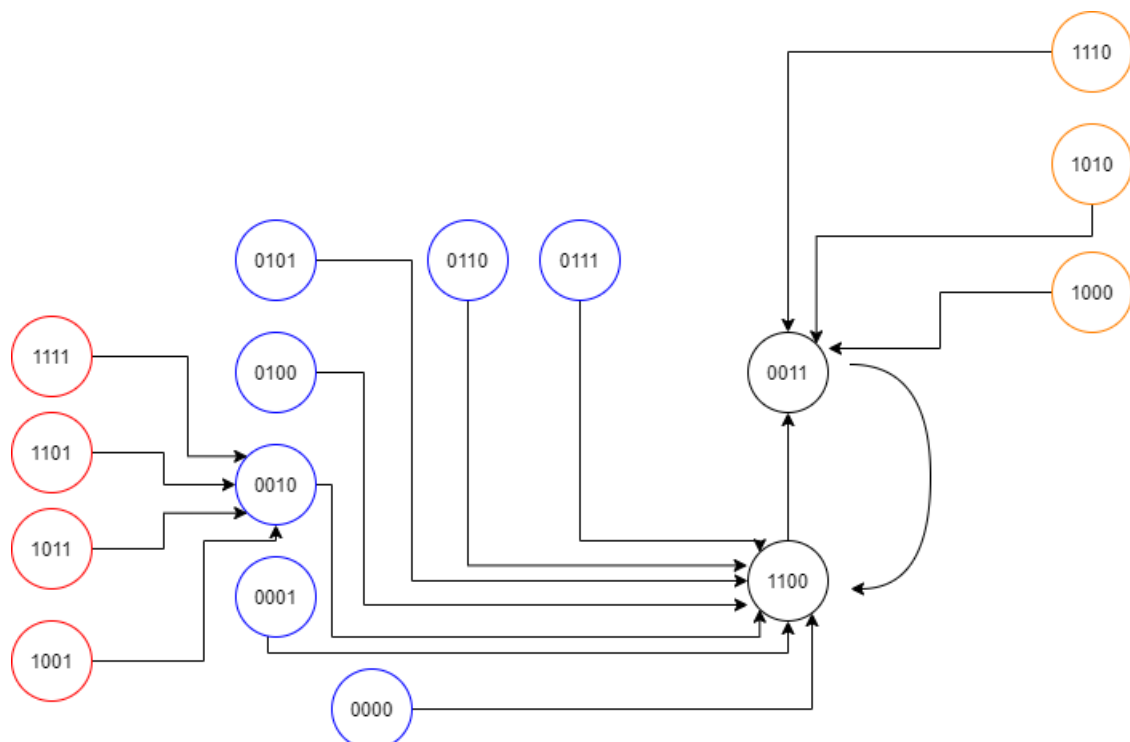
The fourth flip-flop has the NAND of Q2 and NOT of Q3 as its first input and a constant value of 0 as its second input which results in a mixture of reset and toggle states.

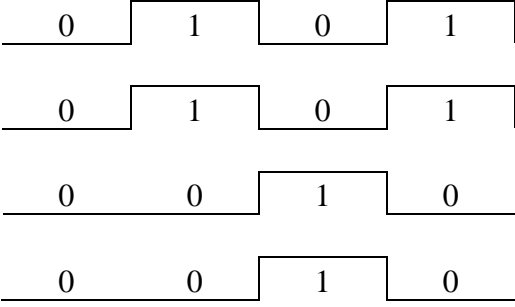
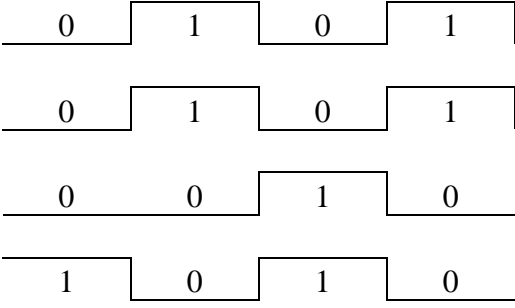
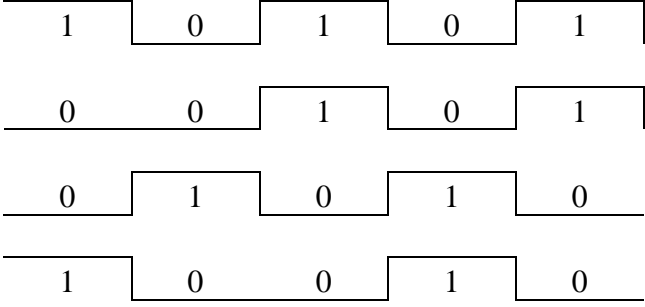
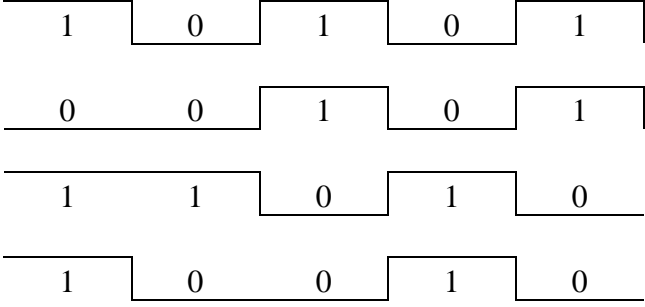
q1	q2	q3	q4	Q2	~Q3	J4	~K4	Q4
0	0	0	0	1	1	0	0	0
0	0	0	1	1	1	0	0	0
0	0	1	0	1	1	0	0	0
0	0	1	1	1	1	0	0	0
0	1	0	0	1	1	0	0	0
0	1	0	1	1	1	0	0	0
0	1	1	0	1	1	0	0	0
0	1	1	1	1	1	0	0	0
1	0	0	0	0	0	1	0	1
1	0	0	1	0	0	1	0	0
1	0	1	0	0	0	1	0	1
1	0	1	1	0	0	1	0	0
1	1	0	0	0	0	1	0	1
1	1	0	1	0	0	1	0	0
1	1	1	0	0	0	1	0	1
1	1	1	1	0	0	1	0	0

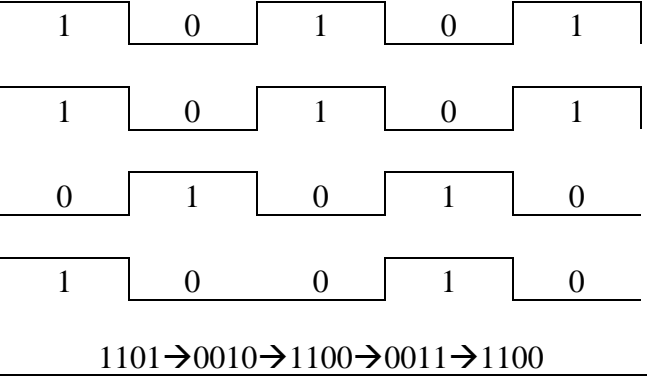
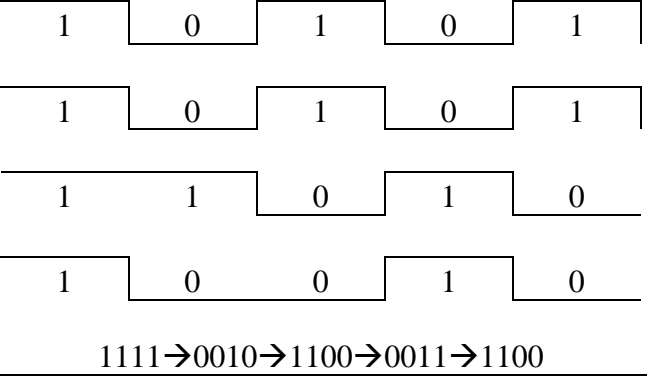
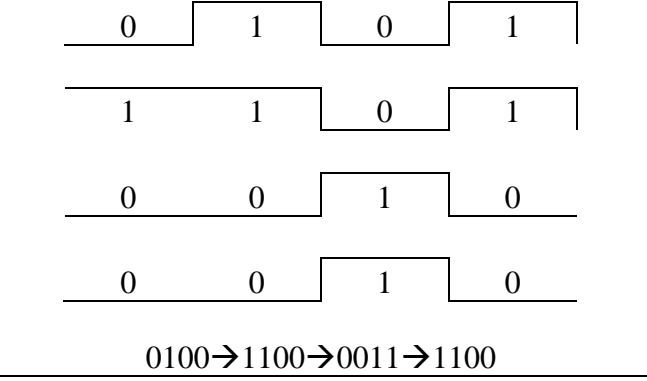
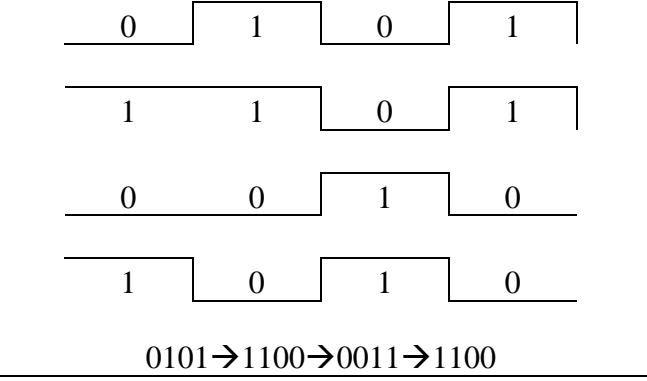
We shall now summarize the results into one table which is shown below.

q1	q2	q3	q4	Q1	Q2	Q3	Q4
0	0	0	0	1	1	0	0
0	0	0	1	1	1	0	0
0	0	1	0	1	1	0	0
0	0	1	1	1	1	0	0
0	1	0	0	1	1	0	0
0	1	0	1	1	1	0	0
0	1	1	0	1	1	0	0
0	1	1	1	1	1	0	0
1	0	0	0	0	0	1	1
1	0	0	1	0	0	1	0
1	0	1	0	0	0	1	1
1	0	1	1	0	0	1	0
1	1	0	0	0	0	1	1
1	1	0	1	0	0	1	0
1	1	1	0	0	0	1	1
1	1	1	1	0	0	1	0

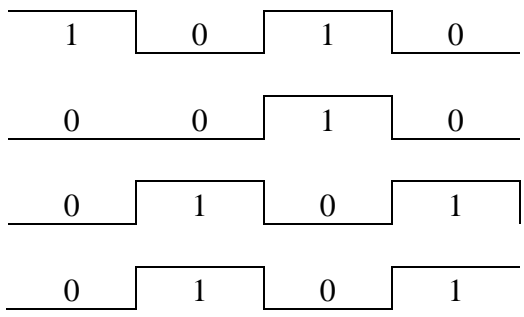
I have made a flowchart diagram which shows the path taken by every possible input.



Starting State	Path
0000	 <p>0000→1100→0011→1100</p>
0001	 <p>0001→1100→0011→1100</p>
1001	 <p>1001→0010→1100→0011→1100</p>
1011	 <p>1011→0010→1100→0011→1100</p>

1101	 <p>1101→0010→1100→0011→1100</p>
1111	 <p>1111→0010→1100→0011→1100</p>
0100	 <p>0100→1100→0011→1100</p>
0101	 <p>0101→1100→0011→1100</p>

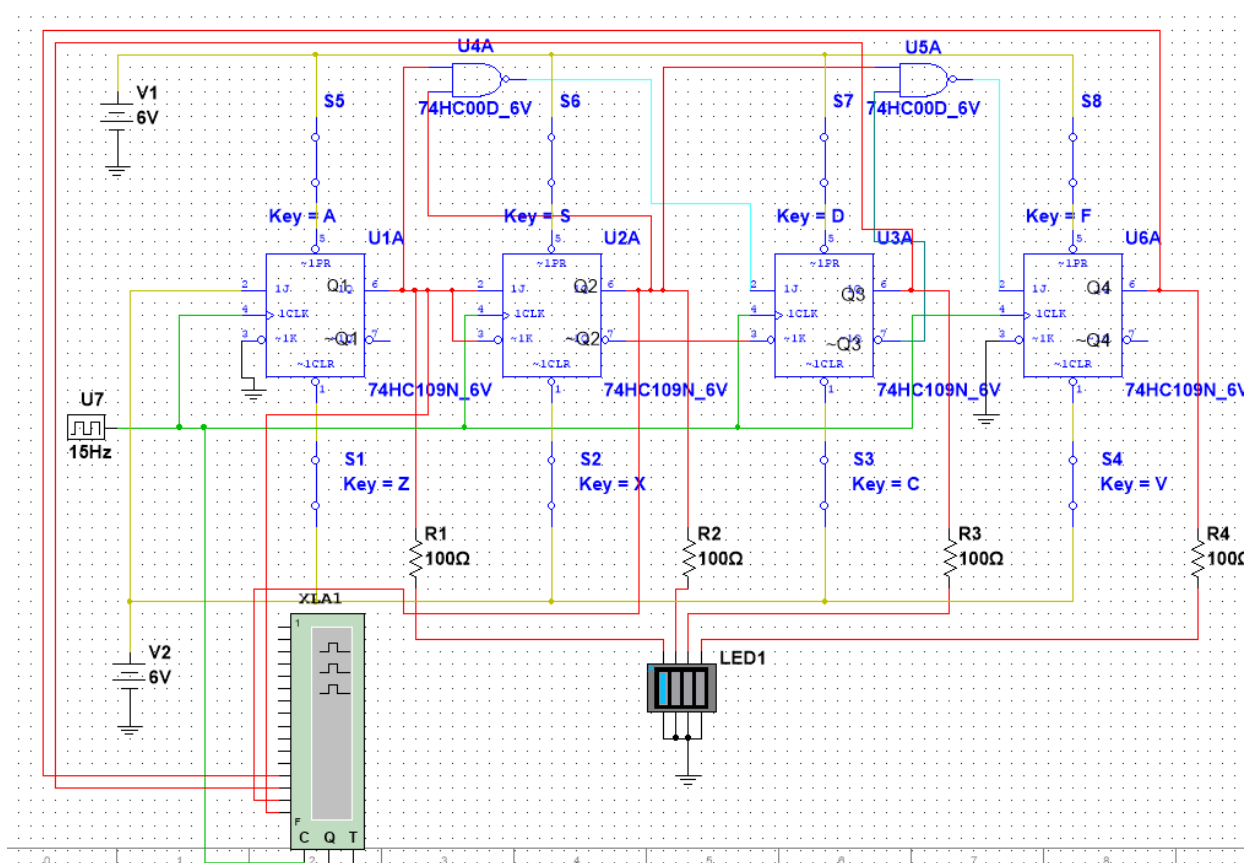
0110	<div><div>0101</div><div>1101</div><div>1010</div><div>0010</div><div>0110→1100→0011→1100</div></div>
0111	<div><div>0101</div><div>1101</div><div>1010</div><div>1010</div><div>0111→1100→0011→1100</div></div>
1110	<div><div>1010</div><div>1010</div><div>1101</div><div>0101</div><div>1110→0011→1100→0011</div></div>
1010	<div><div>1010</div><div>0010</div><div>1101</div><div>0101</div><div>1010→0011→1100→0011</div></div>

1000	 <p>1000→0011→1100→0011</p>
------	---

Digital Solution

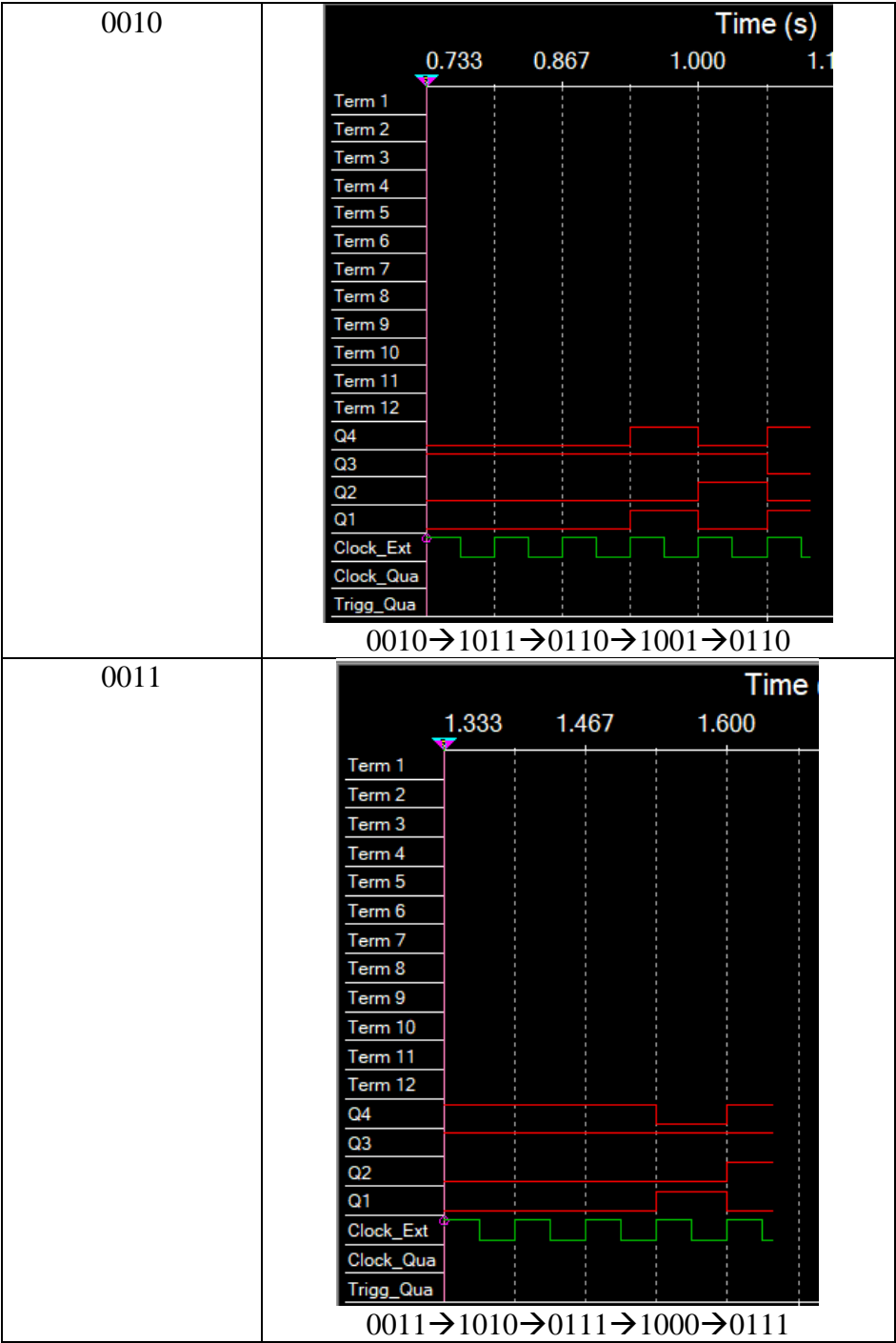
For my digital solution, I used the circuit shown above and connected a logic analyzer to the outputs (Q1, Q2, Q3 and Q4) and the clock signal. I also set the analyzer to read external readings as it was not producing a clock signal of its own.

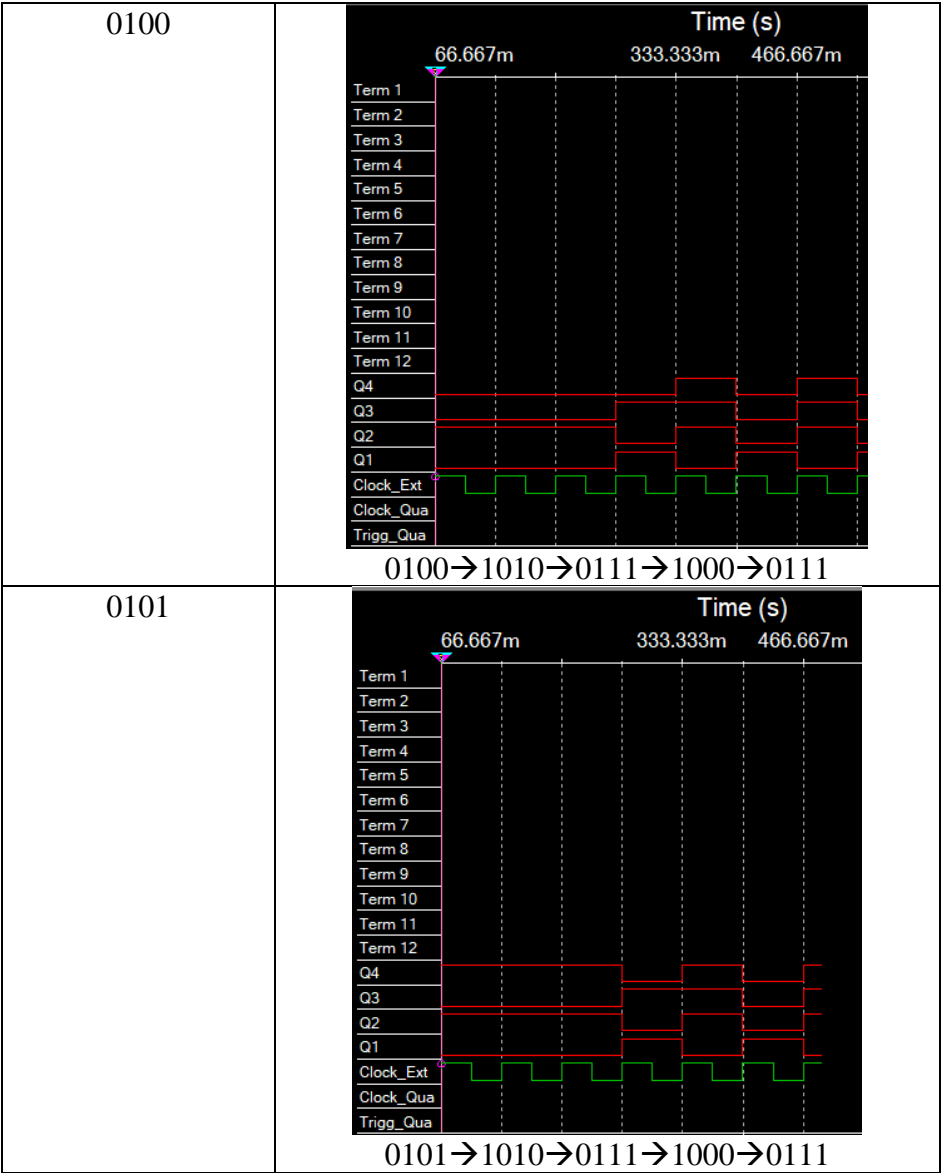
The inputs of each JK flip-flop are discussed above in the analytical solution. In addition, the pre-set and clear inputs play a major role in the digital and physical solutions. The pre-set forces the output to a 1 regardless of the input whereas the clear output sets the output to a 0 regardless of the input. To explain this in detail let's consider an example. In order to produce a starting value of 0101, the first and fourth flip-flop will have the pre-set on and the clear off. The second and the third will have a pre-set off and a clear on. Also note that the pre-set and set pins are NOT of the input hence the opposite input.

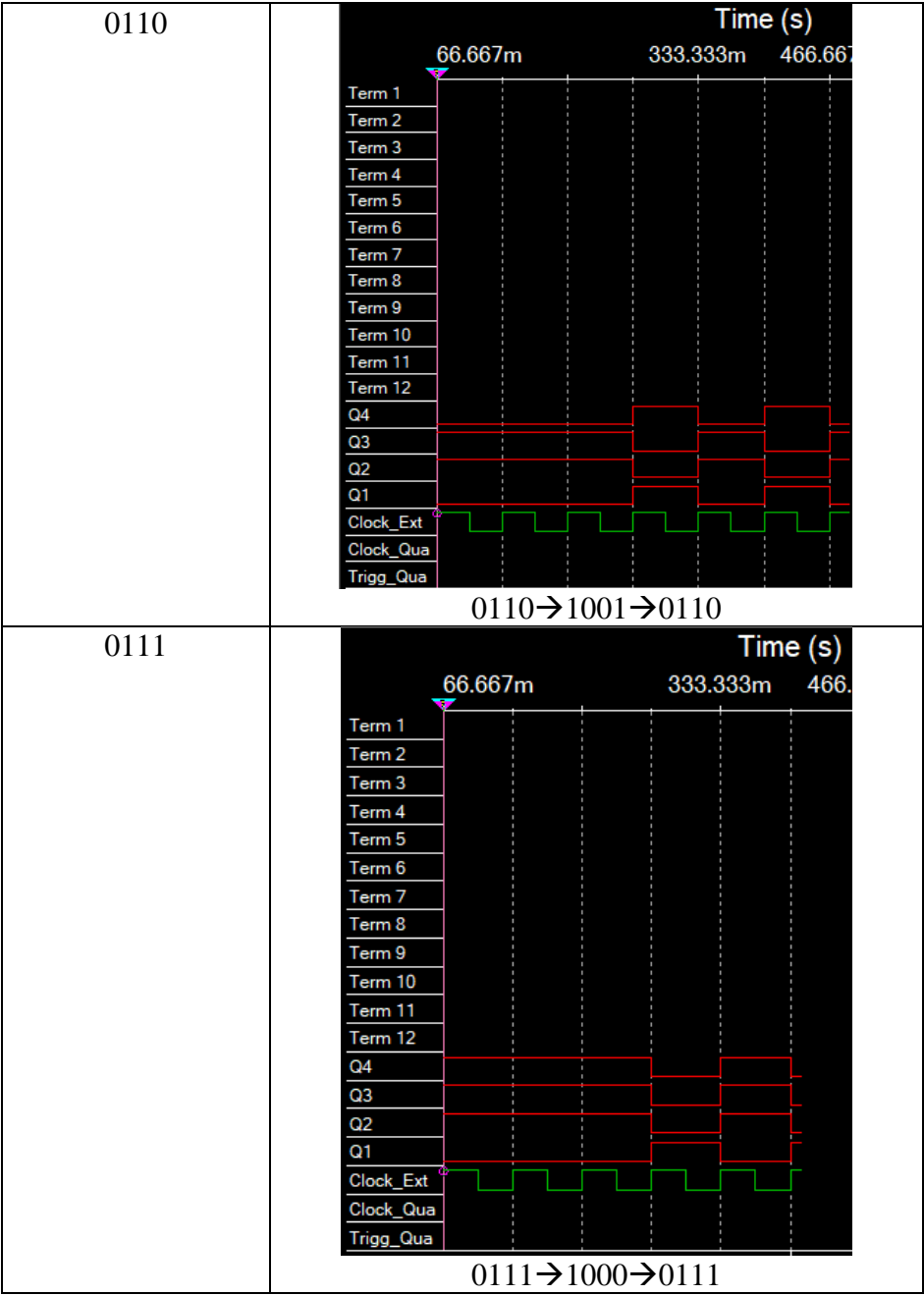


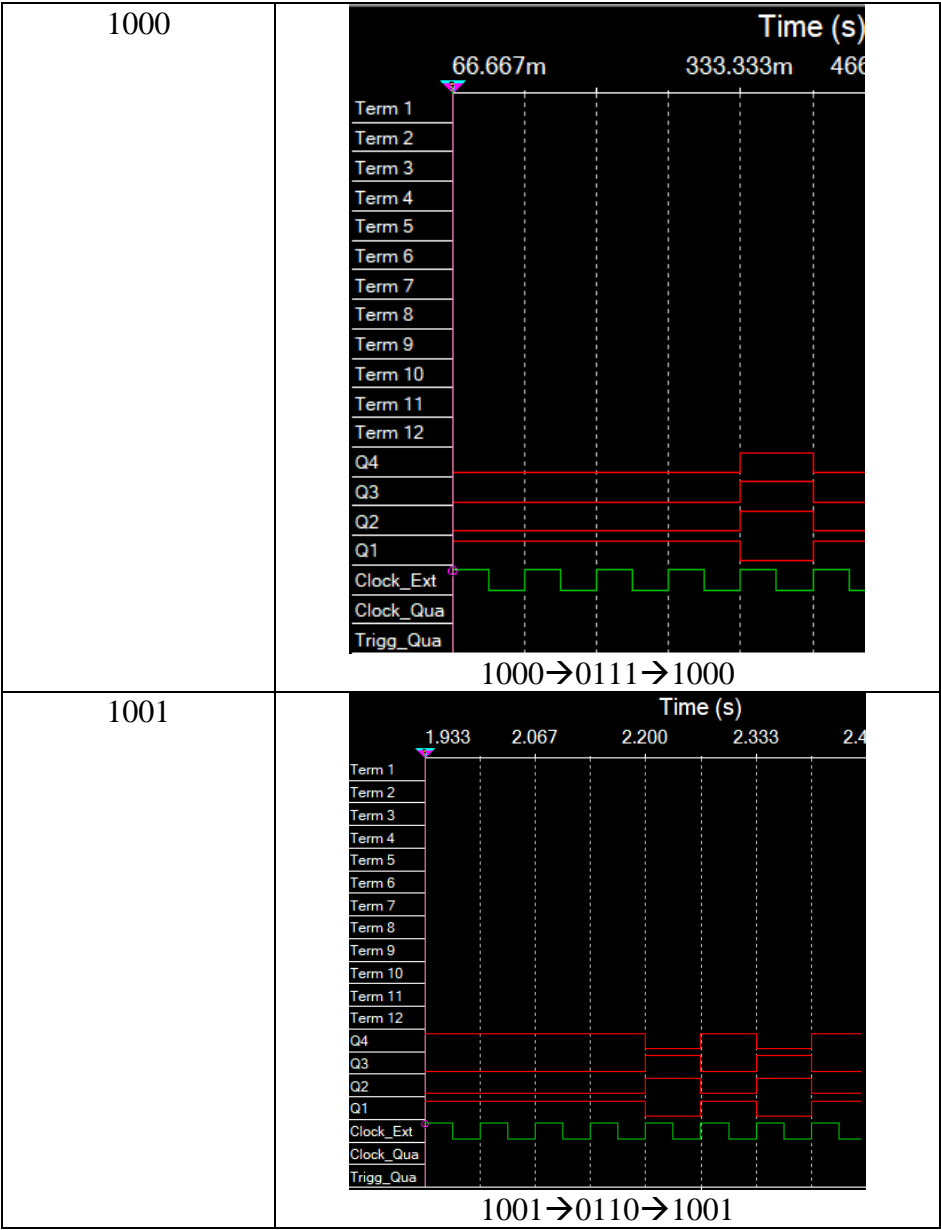
The timing diagram for every possible initial state is shown below.

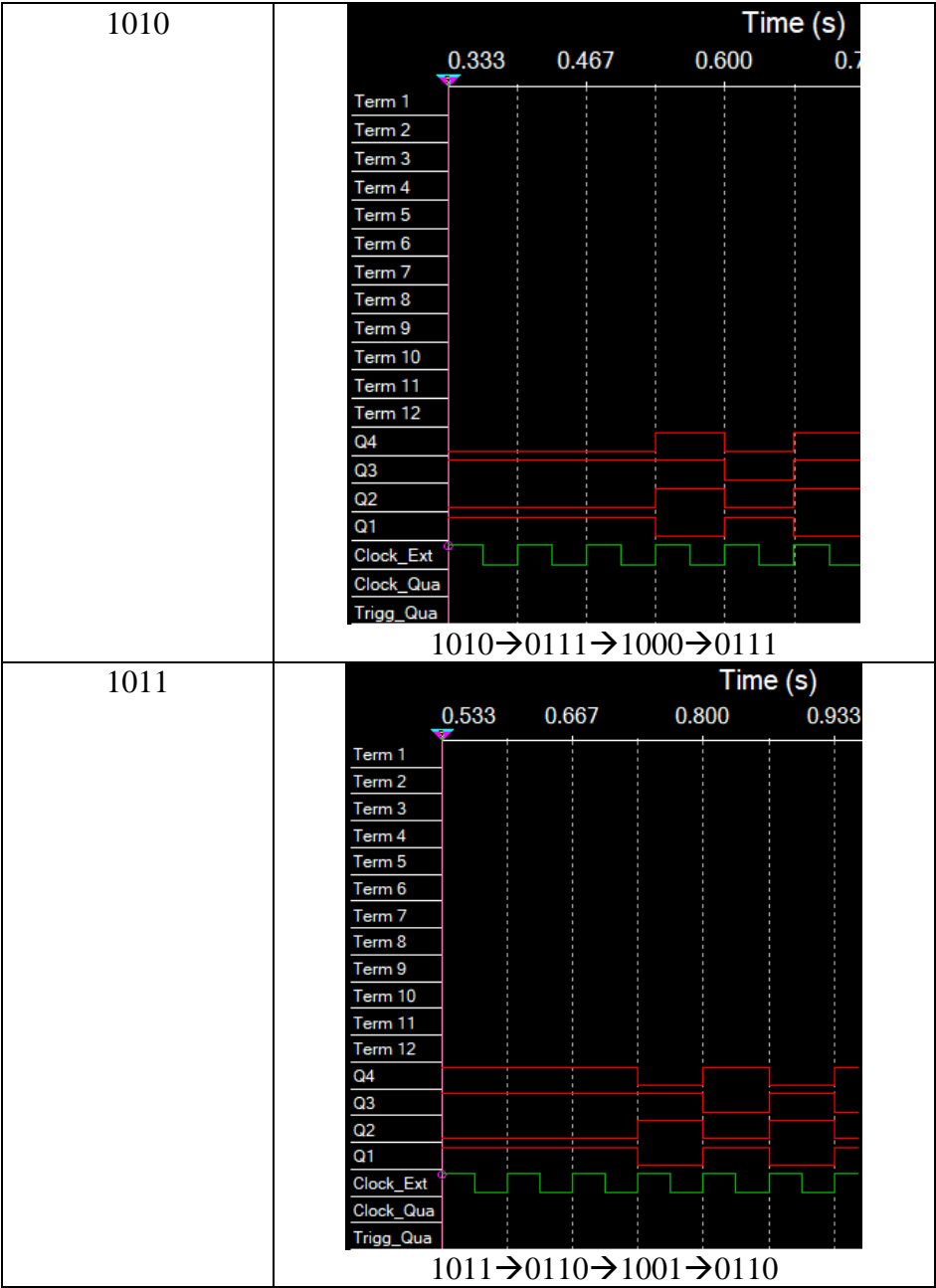
Initial State	Timing Diagram
0000	<div><p>Time (s)</p><p>66.667m 333.333m 466.667m</p><p>Term 1 Term 2 Term 3 Term 4 Term 5 Term 6 Term 7 Term 8 Term 9 Term 10 Term 11 Term 12 Q4 Q3 Q2 Q1 Clock_Ext Clock_Qua Trigg_Qua</p><p>0000→1011→0110→1001→0110</p></div>
0001	<div><p>Time (s)</p><p>66.667m 333.333m 466.667m</p><p>Term 1 Term 2 Term 3 Term 4 Term 5 Term 6 Term 7 Term 8 Term 9 Term 10 Term 11 Term 12 Q4 Q3 Q2 Q1 Clock_Ext Clock_Qua Trigg_Qua</p><p>0001→1010→0111→1000→0111</p></div>

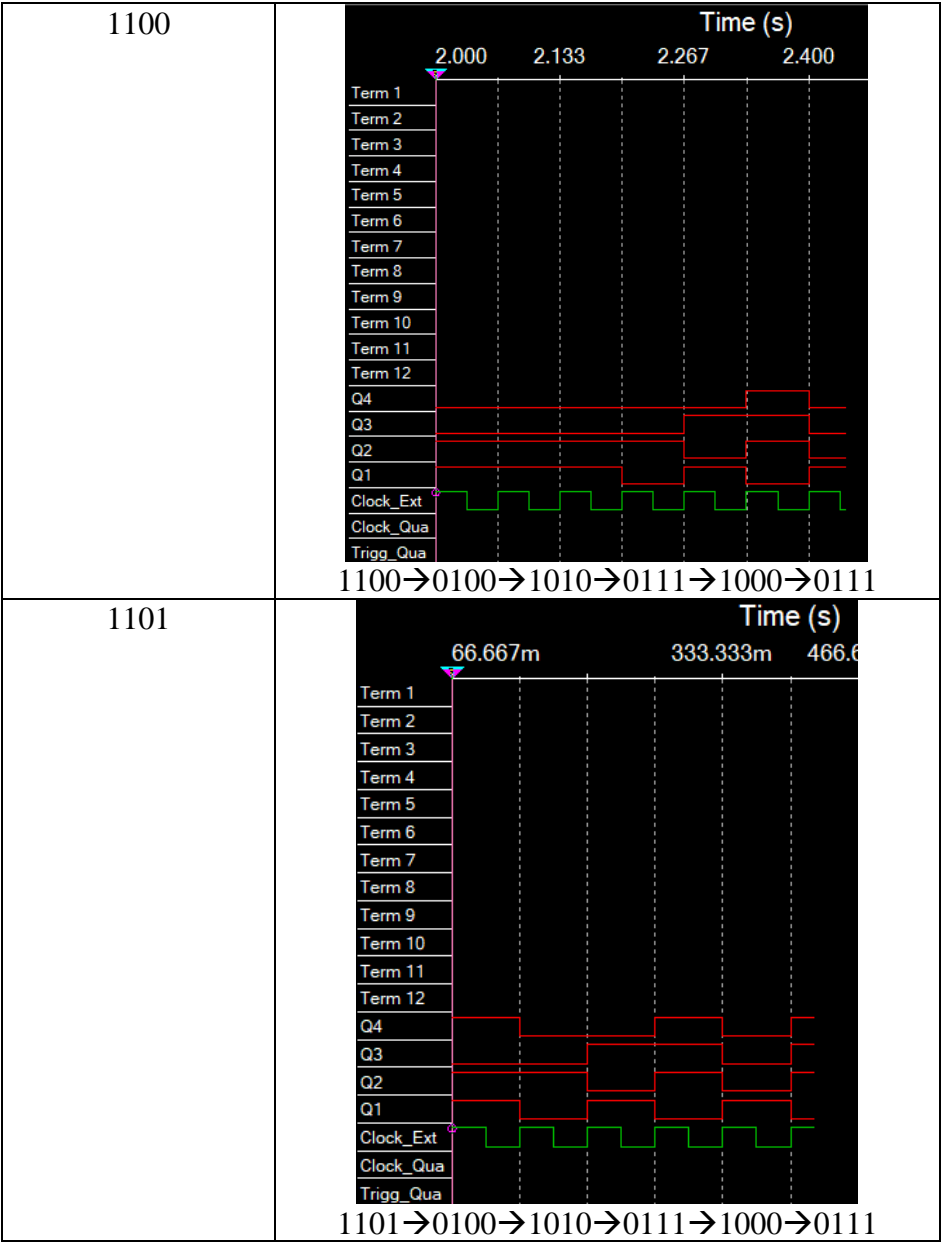


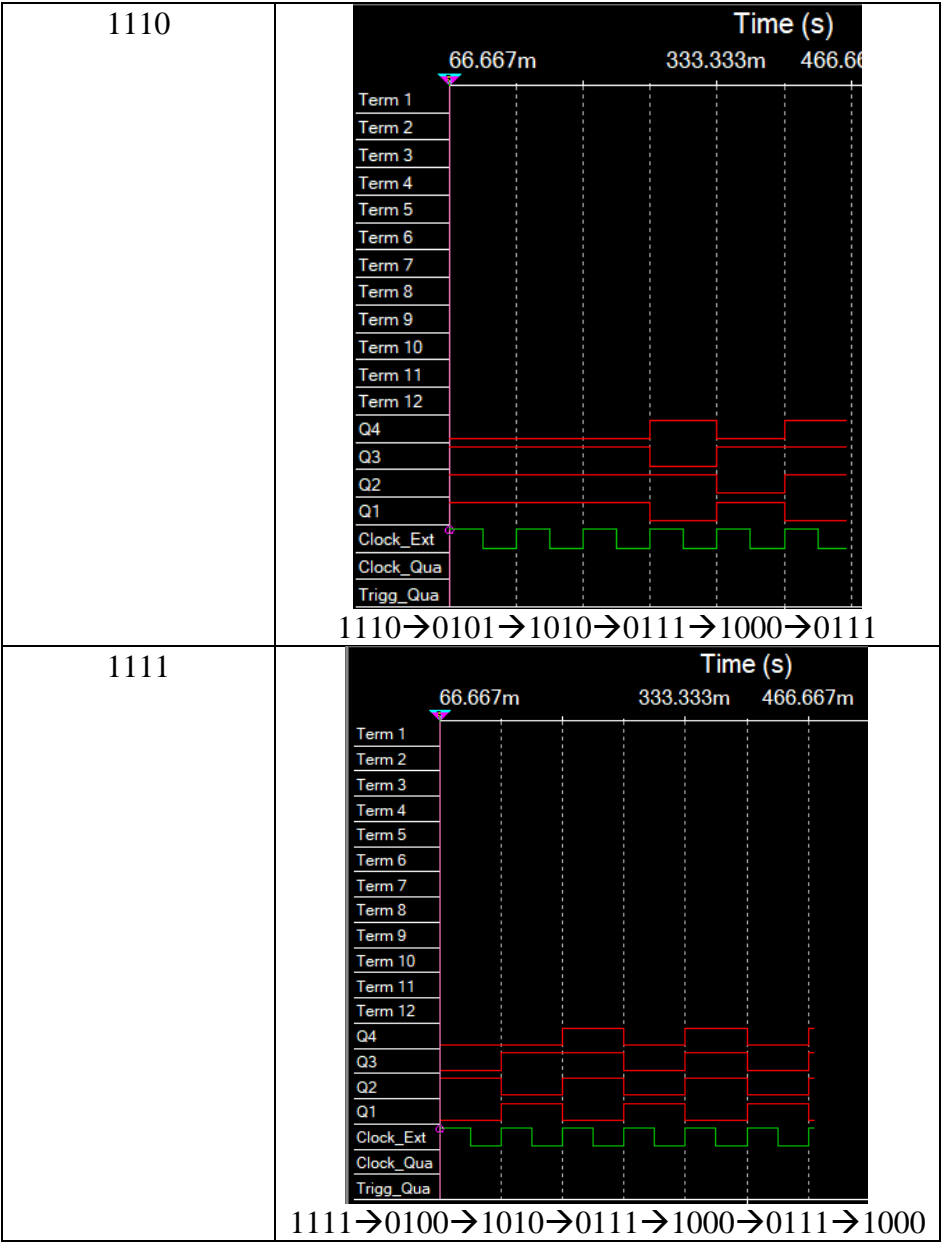












q1	q2	q3	q4	Q1	Q2	Q3	Q4
0	0	0	0	1	0	1	1
0	0	0	1	1	0	1	0
0	0	1	0	1	0	1	1
0	0	1	1	1	0	1	0
0	1	0	0	1	0	1	0
0	1	0	1	1	0	1	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	0	0
1	0	0	0	0	1	1	1
1	0	0	1	0	1	1	0
1	0	1	0	0	1	1	1
1	0	1	1	0	1	1	0
1	1	0	0	0	1	0	0
1	1	0	1	0	1	0	0
1	1	1	0	0	1	0	1
1	1	1	1	0	1	0	0

Debugging Process

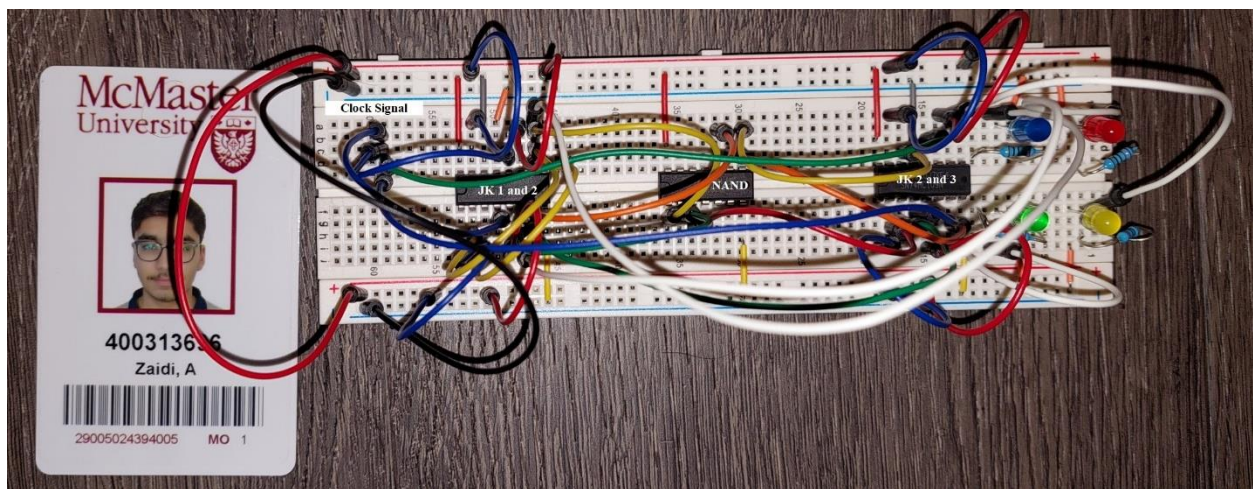
As seen above, my analytical solution results do not match up with my digital solution. My physical solution, however, lines up perfectly with my digital solution. This means that there is something wrong with my reasoning which might have led to the wrong conclusions. For confirming that my method was correct, I reread the lecture notes and rewatched the first three lectures (which cover everything) along with some other videos on YouTube. I did this because I thought that there might be a mistake in my comprehension of the coursework. I also rewired everything in a new Multisim environment, but this was of no use as my physical solution confirmed that the analytical solution was correct. Furthermore, I checked my solution row by row but could not find any mistake.

Because of these results, I made a table which included every 16 possible states in order to account for any mistakes made in the analytical portion. Even though this was very time-consuming to do for both the digital and physical solution, I wanted to do this in order to have error-free solutions.

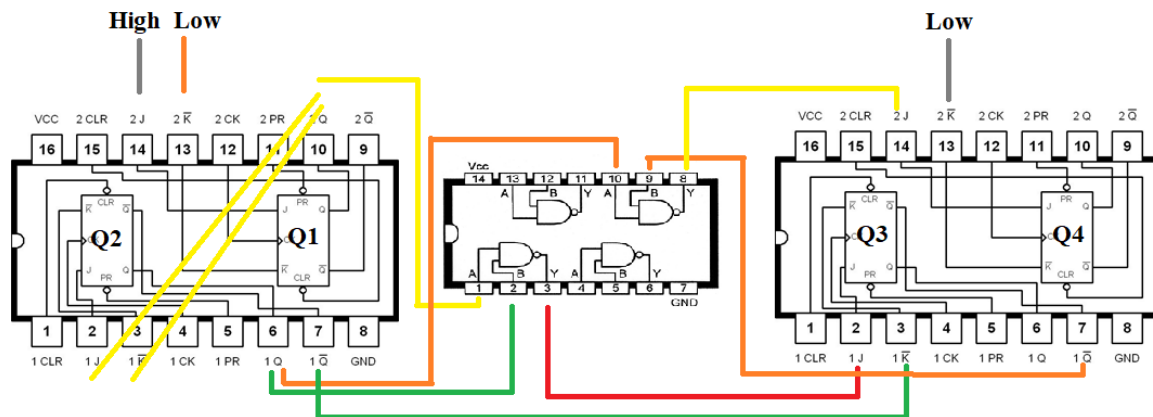
Physical Solution

Below is a picture of my physical solution. The positive and negative rails are connected to each other and take input from the other circuit (shown below). The blue and red wires from each JK flip-flop represent the pre-set and clear of each one. The fixed red and yellow wires are the vcc and ground of each chip. The LEDs are in the order of the top two ones being the left side of the four bits and the bottom one the last two bits. So, if the output is 0110, the order is 0→Blue, 1→Red, 1→Green, 0→ Yellow. This was done as there was a space constraint.

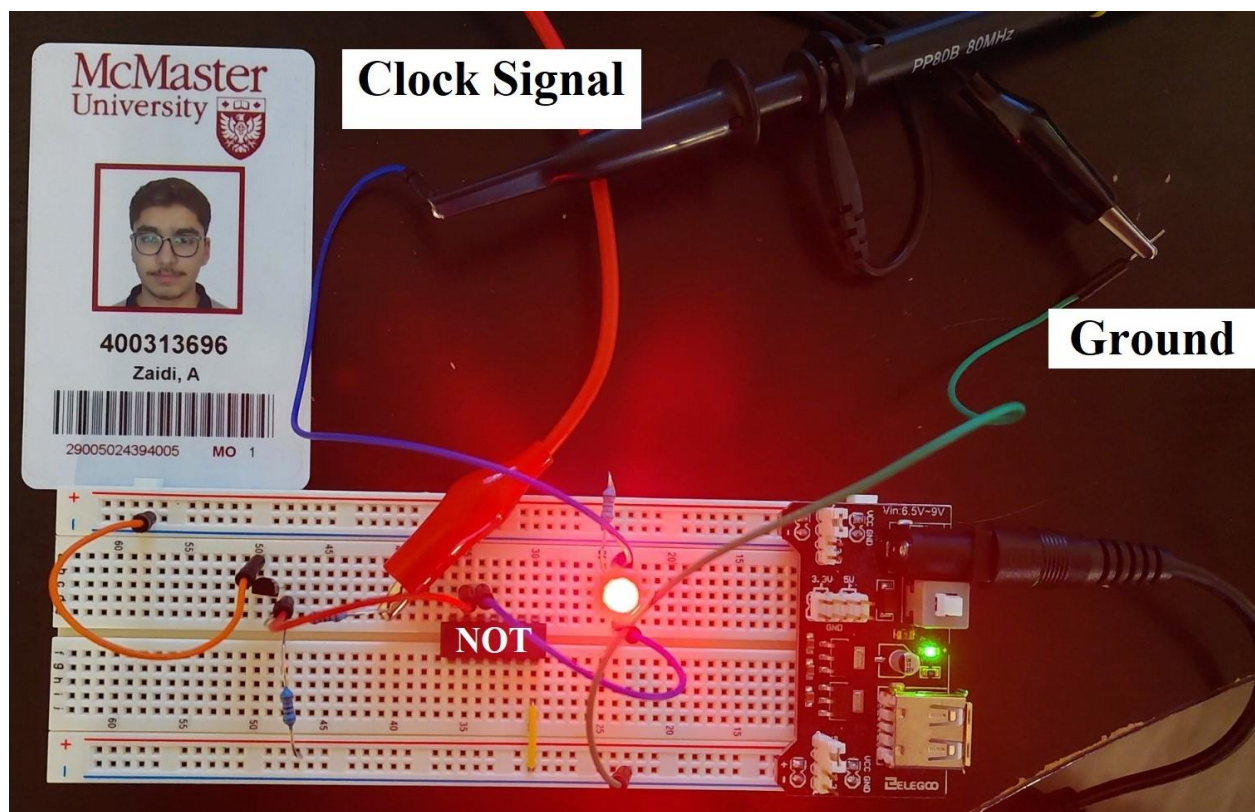
Moreover, the circuit that produces the clock signal, transfers its output to the main circuit as labelled below.



This diagram represents a better view of the working of the circuit. I have purposely left out the pre-set, clear, clock, vcc and ground wires for an easier grasp of the circuit and have explained them above.

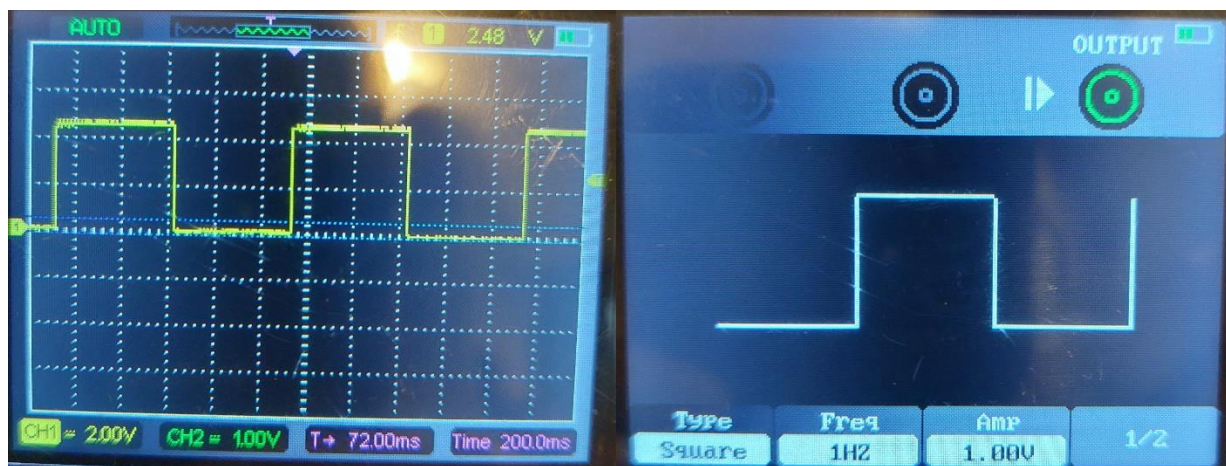


Below is the clock signal produced from a transistor and a NOT gate connected.



Clock signal video: <https://youtu.be/h0G4T8X8VWw>

The clock signal's output was read on the Hantek Oscilloscope.



I also made a circuit which acts as a control experiment to verify if the LEDs are working or not.

Control Experiment: <https://youtu.be/pQL1orM6Sdc>

Initial State	Timing Diagram
0000	https://youtu.be/zROnXC7nunE 0000→1011→0110→1001→0110
0001	https://youtu.be/zny3-xyVs-c 0001→1010→0111→1000→0111
0010	https://youtu.be/zROnXC7nunE 0010→1011→0110→1001→0110
0011	https://youtu.be/zny3-xyVs-c 0011→1010→0111→1000→0111
0100	https://youtu.be/zny3-xyVs-c 0100→1010→0111→1000→0111
0101	https://youtu.be/zny3-xyVs-c 0101→1010→0111→1000→0111

0110	https://youtu.be/zROnXC7nunE 0110→1001→0110
0111	https://youtu.be/zny3-xyVs-c 0111→1000→0111
1000	https://youtu.be/zny3-xyVs-c 1000→0111→1000
1001	https://youtu.be/zROnXC7nunE 1001→0110→1001
1010	https://youtu.be/zny3-xyVs-c 1010→0111→1000→0111
1011	https://youtu.be/zROnXC7nunE 1011→0110→1001→0110
1100	https://youtu.be/zny3-xyVs-c 1100→0100→1010→0111→1000→0111
1101	https://youtu.be/zny3-xyVs-c 1101→0100→1010→0111→1000→0111
1110	https://youtu.be/zny3-xyVs-c 1110→0101→1010→0111→1000→0111
1111	https://youtu.be/zny3-xyVs-c 1111→0100→1010→0111→1000→0111→1000

Discussion

As seen below, my analytical solution had some error (see Debugging Process). My digital and physical solution, however, agreed with each other. The two loops that were seen were 0111→1000→0111 and 0110→1001→0110. They were the same in the digital and physical solution, so they were confirmed to be true values.

One type of experimental constraint that I encountered was the lack of space on the breadboard which caused me to fit the LEDs into a very tight space. Because there were so many components and wiring involved, it was very easy to wire incorrectly. It was also very difficult to spot the error which was why I first made the circuit without the pre-set, clear and clock signals plugged in and then I connected them after confirming with the lab technician.

Analytical Solution	
Initial State	Path
0000	0000→1100→0011→1100
0001	0001→1100→0011→1100
1001	1001→0010→1100→0011→1100
1011	1011→0010→1100→0011→1100
1101	1101→0010→1100→0011→1100
1111	1111→0010→1100→0011→1100
0100	0100→1100→0011→1100
0101	0101→1100→0011→1100
0110	0110→1100→0011→1100
0111	0111→1100→0011→1100
1110	1110→0011→1100→0011
1010	1010→0011→1100→0011
1000	1000→0011→1100→0011

Digital Solution	
Initial State	Path
0000	0000→1011→0110→1001→0110
0001	0001→1010→0111→1000→0111
0010	0010→1011→0110→1001→0110
0011	0011→1010→0111→1000→0111
0100	0100→1010→0111→1000→0111
0101	0101→1010→0111→1000→0111
0110	0110→1001→0110
0111	0111→1000→0111
1000	1000→0111→1000
1001	1001→0110→1001
1010	1010→0111→1000→0111
1011	1011→0110→1001→0110
1100	1100→0100→1010→0111→1000→0111
1101	1101→0100→1010→0111→1000→0111
1110	1110→0101→1010→0111→1000→0111
1111	1111→0100→1010→0111→1000→0111→1000

Physical Solution	
Initial State	Path
0000	0000→1011→0110→1001→0110
0001	0001→1010→0111→1000→0111
0010	0010→1011→0110→1001→0110
0011	0011→1010→0111→1000→0111
0100	0100→1010→0111→1000→0111
0101	0101→1010→0111→1000→0111
0110	0110→1001→0110
0111	0111→1000→0111
1000	1000→0111→1000
1001	1001→0110→1001
1010	1010→0111→1000→0111
1011	1011→0110→1001→0110
1100	1100→0100→1010→0111→1000→0111
1101	1101→0100→1010→0111→1000→0111
1110	1110→0101→1010→0111→1000→0111
1111	1111→0100→1010→0111→1000→0111→1000

Reflection

Overall, this lab showed me a different side to electronics by first touching on a difficult concept of latches and then moving on to the finite state machines. One mistake that I did when I was first learning this material was to look at JK flip-flops in terms of the logic gates inside of it. I was then advised that if I look the flip-flop as a whole then it would make much more sense. Moreover, the content covered in this lab would be very useful for the design project along with having relevant practical applications. We, at the start, learnt that the latches and finite state machines are used to store memory, thus because they form the building blocks of computer memory, it is necessary to understand the working of these devices in order to understand the computer as a whole.