SCHOOL OF INFORMATION, COMPUTER AND COMMUNICATION TECHNOLOGY
SIRINDHORN INTERNATIONAL INSTITUTE OF TECHNOLOGY
THAMMASAT UNIVERSITY

LAB REPORT

EES 370 DIGITAL CIRCUIT LABORATORY

Lab 06 Sequential Logic Circuit I
Lab 07 Sequential Logic Circuit II (Running LEDs)
Lab 08 Running LEDs on Tinkercad

By

Mr. Makhapoom Euaumpon	**ID: 6222780833**
Mr. Paphana Yiwsiw	**ID: 6222780379**

Group No. 12 Section 2

Date:
08 March 2021, Time: 13:00-16:00 (Lab 06)
15 March 2021, Time: 13:00-16:00 (Lab 07)
22 March 2021, Time: 13:00-16:00 (Lab 08)

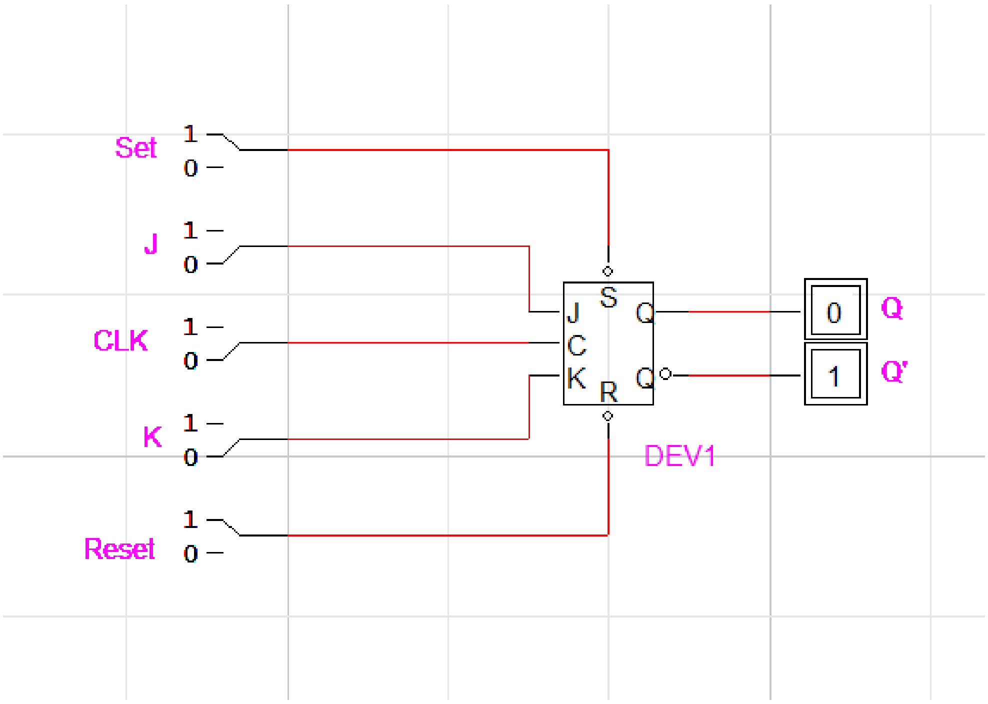
Objectives

1. To know how to use and create D and J-K flip flop, understand its truth table and how to implement to a circuit.
2. To know how to create and design a state diagram and state table of finite state machine.
3. To know how to obtain Boolean expressions of a finite state machine by using K-map and transition table of D and J-K flip flop.
4. To know how to implement finite state machine circuit in LogicWorks 5.
5. To know how to implement circuit of finite state machine of running LEDs in Tinkercad.
6. To create a vending machine and running LEDs circuit in LogicWorks 5 and Tinkercad.

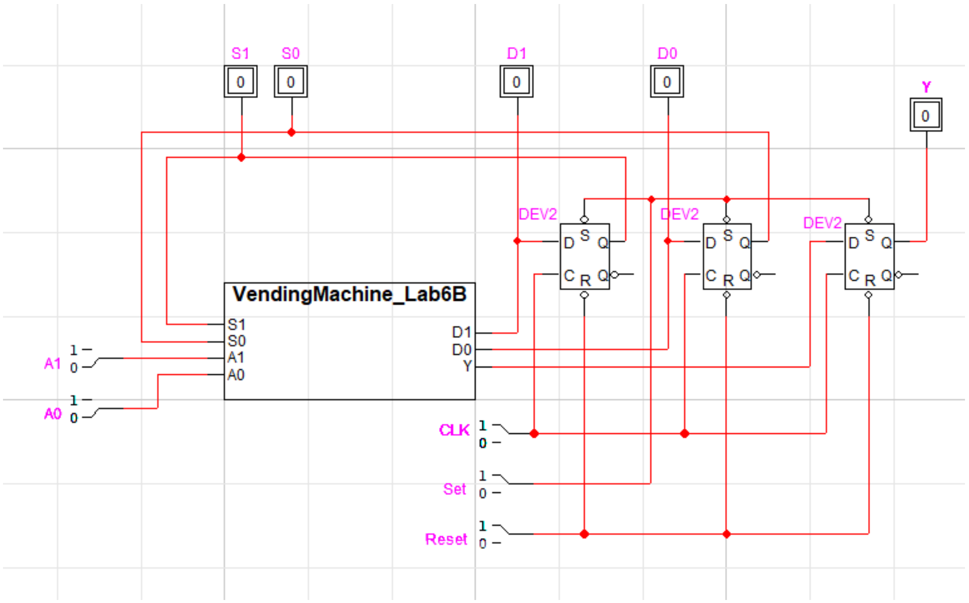
Lab Result

Result of Lab 06: Sequential Logic Circuit I

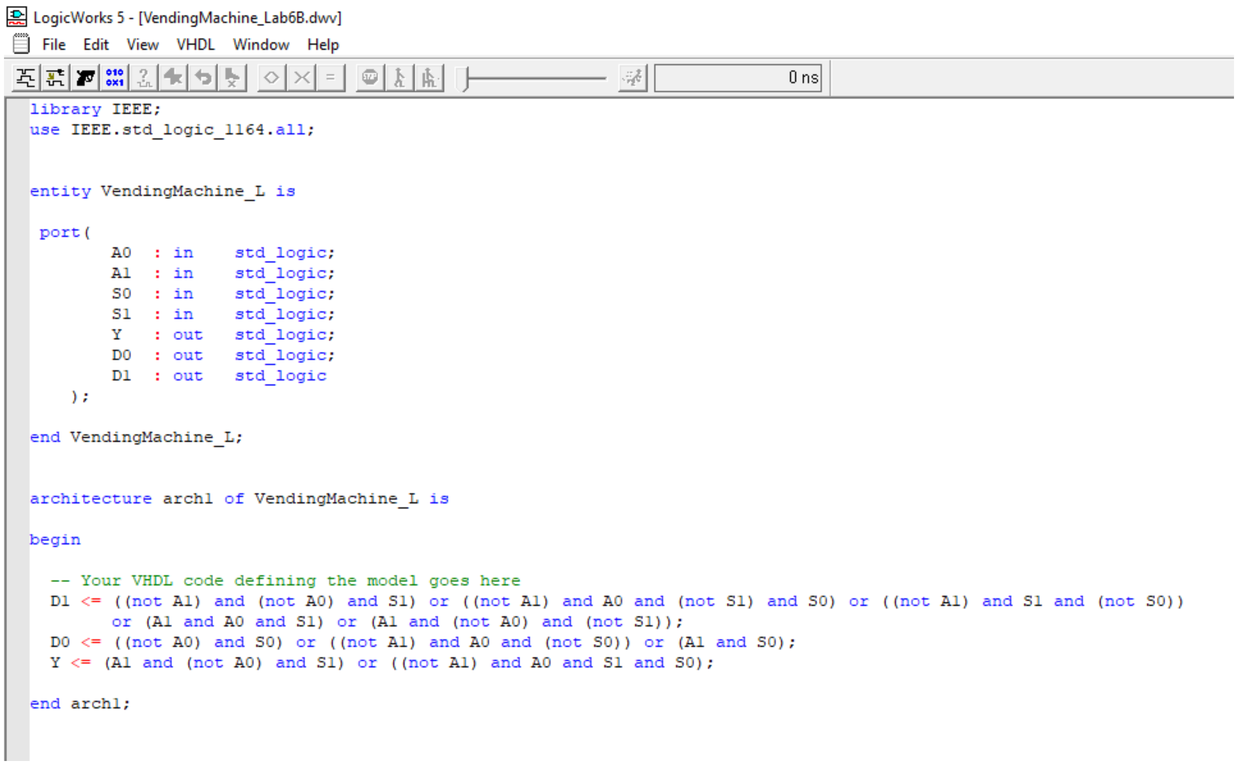
Part A



Part B



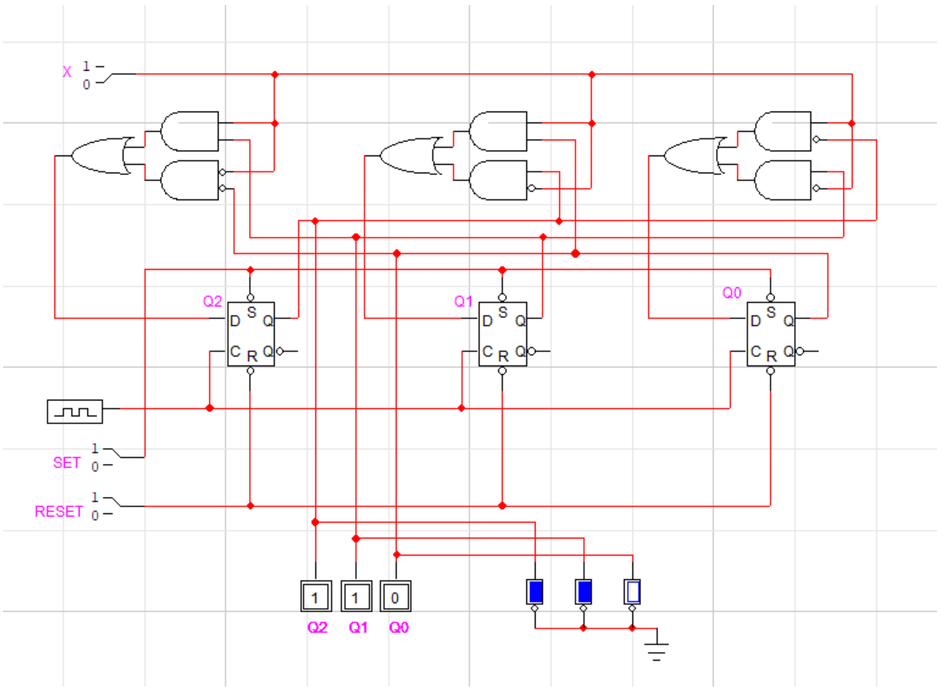
Part B VHDL Code


LogicWorks 5 - [VendingMachine_Lab6B.dwv]
File Edit View VHDL Window Help
0 ns

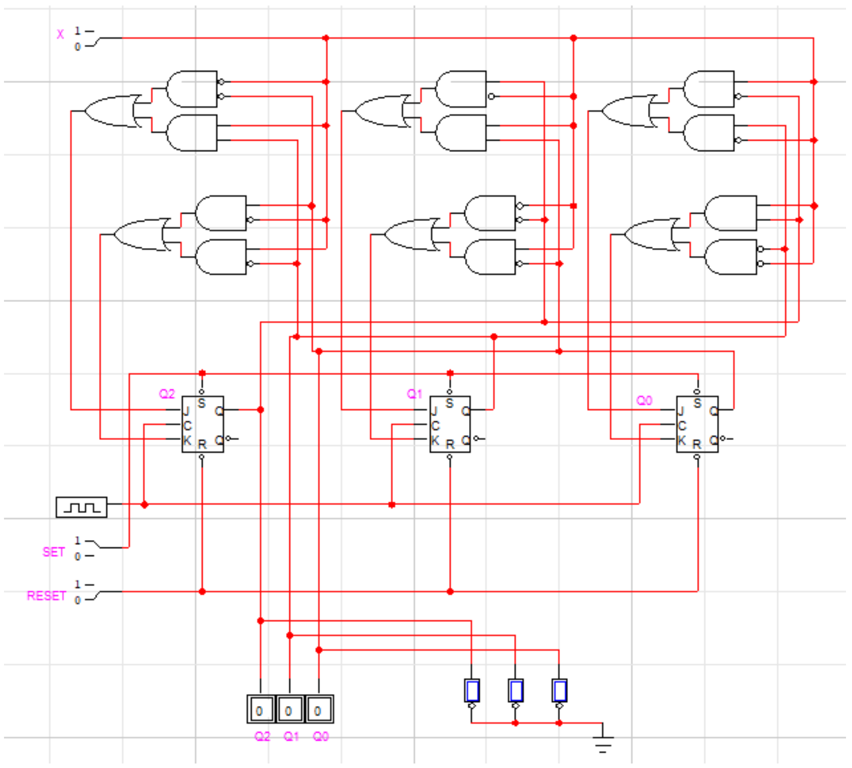
```
library IEEE;
use IEEE.std_logic_1164.all;


entity VendingMachine_L is

 port(
        A0  : in    std_logic;
        A1  : in    std_logic;
        S0  : in    std_logic;
        S1  : in    std_logic;
        Y   : out   std_logic;
        D0  : out   std_logic;
        D1  : out   std_logic
    );

end VendingMachine_L;


architecture arch1 of VendingMachine_L is

begin

  -- Your VHDL code defining the model goes here
  D1 <= ((not A1) and (not A0) and S1) or ((not A1) and A0 and (not S1) and S0) or ((not A1) and S1 and (not S0))
        or (A1 and A0 and S1) or (A1 and (not A0) and (not S1));
  D0 <= ((not A0) and S0) or ((not A1) and A0 and (not S0)) or (A1 and S0);
  Y <= (A1 and (not A0) and S1) or ((not A1) and A0 and S1 and S0);

end arch1;
```

Result of Lab 07: Sequential Logic Circuit II (Running LEDs)

Part A (D Flip-Flop)

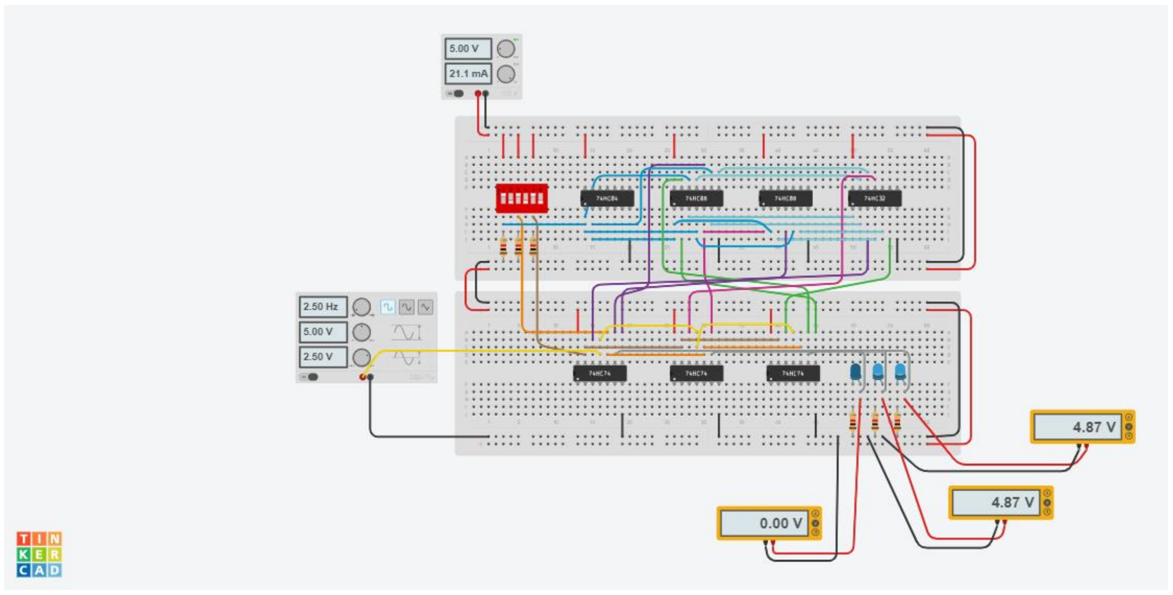


Part B (JK Flip-Flop)



Result of Lab 08: Running LEDs on Tinkercad

https://www.tinkercad.com/things/frMKfjpj7ot-lab08-6222780379/editel?sharecode=m10i3EPvMb6u4sexO_V35zy5rHeJBtDRbr8C8MkoL-s



Discussion

> Lab 06: Sequential Logic Circuit I

In part A, we get to explore and observe output of J-K Flip-Flop by simulate it on LogicWorks. Flip-Flops are parts of sequential logic circuit which the circuit of this kind will produce the output based on both current input and past input. It basically means that it has ability to memorize the input. Flip-flops also required timing, also called clock (CLK), as it is a synchronous device which will change only the transition of the clock signal. If it is positive/rising edge FF, it will change when CLK is goes from LOW to HIGH. If it is negative/falling edge FF, it will change when CLK is goes from HIGH to LOW. In flip-flops, there are also a preset (PRE') and clear (CLR') signal as asynchronous signal to use it to set the output to 1 and 0, respectively. Keep in mind that the preset and clear signal are active-low signal which means that it will activate when the signal is LOW only. There are many types of flip-flops such as S-R flip-flop (2 inputs: S and R), D flip-flop (1 input: D) and J-K flip-flop (2 inputs: J and K). Flip-flops will produce 2 outputs, Q and Q' which is basically the complement of Q. Different type of flip-flop will generate output differently. Focusing on J-K flip-flops, we can begin to observe from the asynchronous signal. These signal will set or reset the output with no regards to the CLK signal at all. As mentioned above, PRE' and CLR' are active-low signal. If both signal are LOW, the flip-flop enters prohibited state as the signal cannot be LOW at the same time. As observed from LogicWorks, the output Q and Q' are HIGH. If the PRE' is LOW, output Q is HIGH. If the CLR' is LOW, output Q is LOW. If both signals are LOW, now you can determine the output Q by the input J and K. As J and K are synchronous signal, the output Q will change only when the CLK is changed from LOW to HIGH (As this part, we used positive edge J-K flip-flop). If both J and K are LOW, the output Q will remain the same as the previous output. If J is HIGH and K is LOW, the output Q is HIGH. If J is LOW and K is HIGH, the output Q is LOW. Finally, if both J and K are HIGH, the output Q will be inverted of previous output, also called "toggle". These possible inputs of J-K flip-flops can be illustrated in the following table:

J-K flip-flop							
Input					Output		Mode
Asynchronous		Synchronous					
PRE'	CLR'	CLK	J	K	Q	Q'	
0	0	X	X	X	1	1	Prohibited
0	1	X	X	X	1	0	Preset
1	0	X	X	X	0	1	Clear
1	1	↑	0	0	Q_0	Q_0'	Hold
1	1	↑	0	1	0	1	Reset
1	1	↑	1	0	1	0	Set
1	1	↑	1	1	Q_0'	Q_0	Toggle

Figure 1: Truth table of J-K flip-flop in Lab06 Part A (Previous output denoted as Q_0)

In part B, we have to design the vending machine circuit by using the sequential logic circuit and flip-flop knowledge from the previous part. This type of circuit is also called as finite state machine. The requirement of this vending machine are as follows: this machine will accept 5 Baht and 10 Baht coin only, served item at 20 Baht, the machine keeps accumulate the total amount until there is 20 Baht or more in the machine then the machine will deduct 20 Baht from it and will serve the item.

As from the requirement, there can be 3 different input, 0, 5 and 10 Baht, which can map to binary number as 00, 01 and 10, respectively. Input will be represented as A_1A_0. There is no input of 15 Baht or 11, because 15 Baht coin does not existing. Therefore, this input is deemed as invalid input and we will design this input of the state machine to have the output as same as output from the input of 0 Baht coin (No coin inserted). Thus, there will be no arrow in the transition diagram that have input 11. The number of state of this machine is 4 which are 0, 5, 10 and 15 Baht. The current machine state (S_1S_0) can be represented in binary as 00, 01, 10 and 11, respectively. The output as Y to represent the serve (1) or not served (0) an item.

First step to begin the design of this vending machine is to draw the transition diagram of this machine. The transition diagram will consist of circle represented as current state and arrow represented as state transition. Input and output can be written above the arrow. For example, if current state is 0 Baht (00), when the input is 5 Baht (01), the next state will be 5 Baht (01) and the served output will remain as 0. In this case, the arrow will point from 00 toward 01 and on the arrow wrote 5/0 or 01/0. Another case that, if current state is 15 Baht (11), when the input is 10 Baht (10), the next state will be 5 Baht (01) and the output Y will be 1 as the machine currently have 25 Baht in it. Then, it will deduct 20 Baht from it and return to the remainder state as mentioned. The final transition diagram of this machine are as follows:

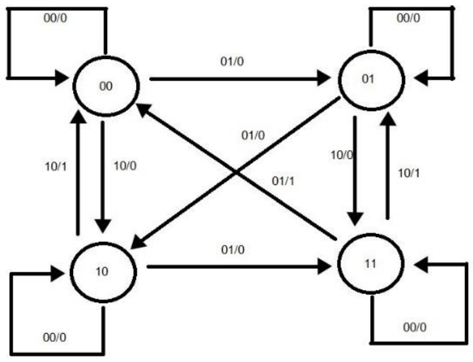


Figure 2: Transition diagram of vending machine.

Second step is to create state table of this vending machine. The state table can be filled easily when there is a transition diagram in place. The column will be the input of machine. The row will be the current state of the machine. And in each cell will be the next state of machine and the output of machine. For example, if the current state is 00 and input is 10, the cell at row 00 and column 10 will be written as 10/0. If the current state is 10 and input is 10, the cell at row 10 and column 10 will be written as 00/1. After filled information from transition diagram into the state table, the state table of this machine are as follows:

Current State (S_1S_0)	**Input (A_1A_0)**			
	00	01	10	11
00	00/0	01/0	10/0	00/0
01	01/0	10/0	11/0	01/0
10	10/0	11/0	00/1	10/0
11	11/0	00/1	01/1	11/0

Figure 3: State table of vending machine

Third step is to create the Boolean expression of each input in the flip-flop. In this vending machine, we will use D flip-flop to create the finite state machine. To create the Boolean expression of this machine, you have to look at the transition table for D flip flop which is as follows:

Current State (Q)	**Next State (Q*)**	**Flip flop input (D)**
0	0	0
0	1	1
1	0	0
1	1	1

Figure 4: Transition table of D flip flop

We can use K-map to obtain the Boolean expression. To fill in each cell of K-map, you must use the information obtained from the D flip-flop transition table which you have to look for current state and next state. In this operation, you have to do in bit-by-bit. As for example, current state 10 and input 01, the next state is 11. In K-map of D_1, the state is from 1 to 1, then wrote 1 on the cell at column (A_1A_0) 01 and row (S_1S_0) 10. In K-map of D_0, the state is from 0 to 1, then wrote 1 on the same position. In K-map of Y, wrote 0 in the same cell location. After filled all cells in K-map, the obtained Boolean expression are as follows:

$$D_1 = A_1'A_0S_1 + A_1'A_0S_1'S_0 + A_1'S_1'S_0' + A_1A_0S_1 + A_1A_0'S_1'$$
$$D_0 = A_0'S_0 + A_1'A_0S_0' + A_1S_0$$
$$Y = A_1A_0'S_1 + A_1'A_0S_1S_0$$

Lastly, you can implement the circuit in LogicWorks by using the obtained Boolean expression in the previous step. First, place the input and flip-flop. Then, connect flip-flop to the CLK signal, PRE' and CLR' switch. Then, connect the output of the flip-flop to logic gates that will corresponding to the Boolean expression (or write a VHDL code to implement it and connect all the output pin to the corresponding flip-flop). Then, connect it into the input of the flip-flop. Then, connect the binary probe to observe the output. Therefore, the vending machine circuit will look like as the figure below (also shown in the lab result part).

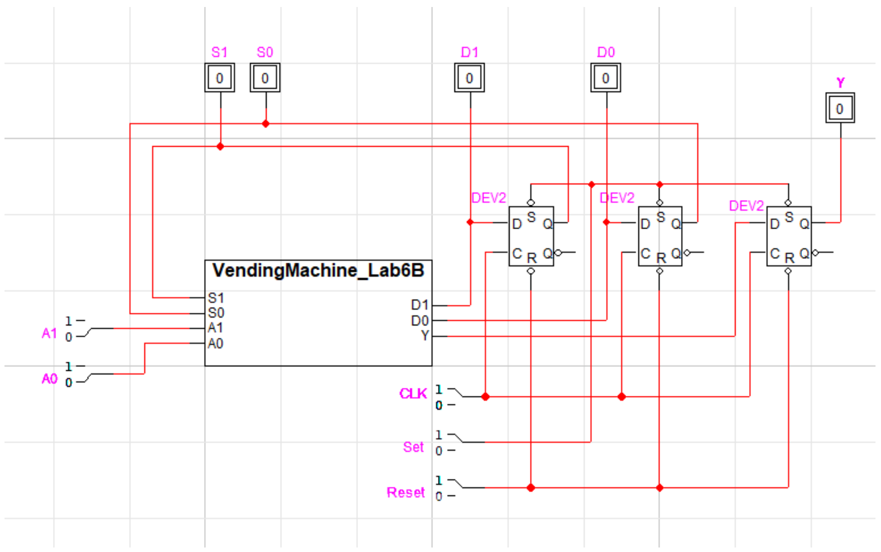


Figure 5: Vending machine circuit using VHDL in LogicWorks 5

> Lab 07: Sequential Logic Circuit II (Running LEDs)

As for this lab, we will implement the circuit of running LEDs. The running LEDs circuit is also using design step as same as the last lab, complete the circuit by using the knowledge of sequential logic circuit and flip-flop from the last lab. In addition to the last lab, the state machine can be separated further into 2 types: Mealy machine and Moore machine. Mealy machine will have the output of the machine depends on the current state and the input. Moore machine will have its output decided solely on the current state only but still receive the input.

In this running LEDs circuit, we will be using only 3 LEDs and received 1 input X to determine the direction of running LEDs. The state of this running LEDs machine is from 000 to 111, for each bit will represent the position of LED. Noted that the state 010 and 101 will not be used in the transition diagram thus both states deemed as invalid state. The direction of running is left to right when X is equal to LOW and right to left when X is equal to HIGH. In each clock, the LED will turn off or on one by one. Thus, we can create a transition diagram as follows:

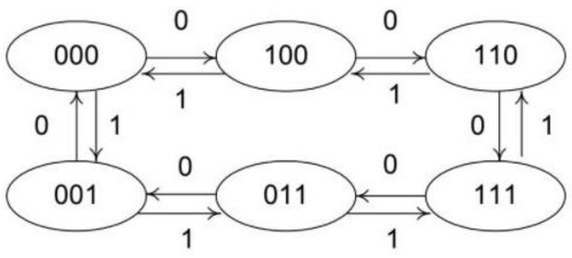
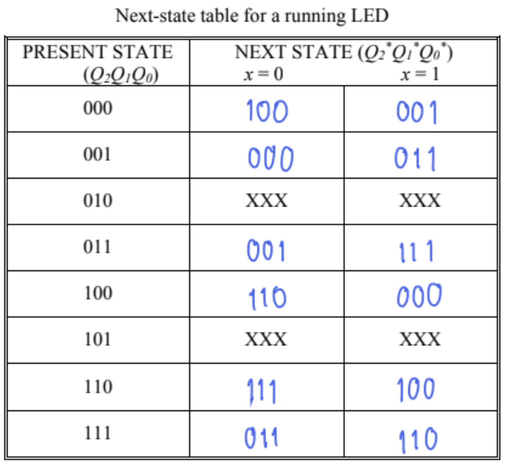


Figure 6: Transition diagram of running LEDs with input to select the direction of running.

From the transition diagram, we can say that when X is LOW, After the transition diagram, we can derive the state table of this machine as the figure below. As we noted above, the state 010 and 101, we will write the next state as XXX (don’t care) into the table as we don’t care about this state at all.



Next-state table for a running LED

PRESENT STATE ($Q_2Q_1Q_0$)	NEXT STATE ($Q_2^*Q_1^*Q_0^*$) $x = 0$	$x = 1$
000	100	001
001	000	011
010	XXX	XXX
011	001	111
100	110	000
101	XXX	XXX
110	111	100
111	011	110

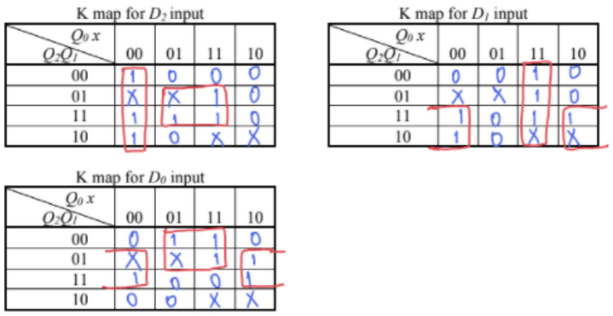
Figure 7: State table of running LEDs

In the next step, we will find Boolean expression of the running LEDs circuit by using D flip flop and J-K flip flop. We will find the expression by using K-map and fill each box in K-map by looking at the transition table. D flip flop transition table is already mentioned in the previous lab discussion part. J-K flip flop transition table is as follows.

Current State (Q)	**Next State (Q*)**	**Flip flop input**	
		J	**K**
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Figure 8: Transition table of J-K flip flop

After filled all K-map, we will have the K-map and the boolean expression are as follow:



K map for D_2 input

Q_2Q_1 \ Q_0x	00	01	11	10
00	1	0	0	0
01	X	X	1	0
11	1	1	1	0
10	1	0	X	X

K map for D_1 input

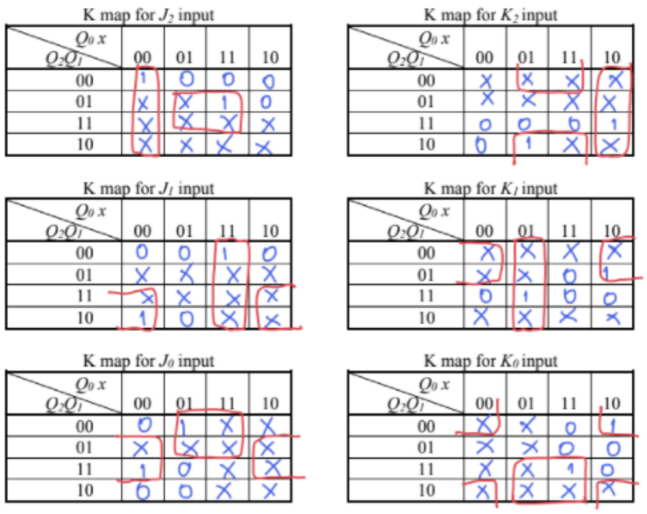
Q_2Q_1 \ Q_0x	00	01	11	10
00	0	0	1	0
01	X	X	1	0
11	1	0	1	1
10	1	0	X	X

K map for D_0 input

Q_2Q_1 \ Q_0x	00	01	11	10
00	0	1	1	0
01	X	X	1	1
11	1	0	0	1
10	0	0	X	X

Figure 9: K-Maps of running LEDs using D flip flop

$D_2 = Q_0'X' + Q_1X$ / $D_1 = Q_0X + Q_2X'$ / $D_0 = Q_1X' + Q_2'X$



K map for J_2 input

Q_2Q_1 \ Q_0x	00	01	11	10
00	1	0	0	0
01	X	X	1	0
11	X	X	X	X
10	X	X	X	X

K map for K_2 input

Q_2Q_1 \ Q_0x	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	0	0	0	1
10	0	1	X	X

K map for J_1 input

Q_2Q_1 \ Q_0x	00	01	11	10
00	0	0	1	0
01	X	X	X	X
11	X	X	X	X
10	1	0	X	X

K map for K_1 input

Q_2Q_1 \ Q_0x	00	01	11	10
00	X	X	X	X
01	X	X	0	1
11	0	1	0	0
10	X	X	X	X

K map for J_0 input

Q_2Q_1 \ Q_0x	00	01	11	10
00	0	1	X	X
01	X	X	X	X
11	1	0	X	X
10	0	0	X	X

K map for K_0 input

Q_2Q_1 \ Q_0x	00	01	11	10
00	X	X	0	1
01	X	X	0	0
11	X	X	1	0
10	X	X	X	X

Figure 10: K-Maps of running LEDs using J-K flip flop

$J_2 = Q_0'X' + Q_1X$ / $K_2 = Q_0X' + Q_1'X$
$J_1 = Q_0X + Q_2X'$ / $K_1 = Q_0'X + Q_2'X'$
$J_0 = Q_1X' + Q_2'X$ / $K_0 = Q_1'X' + Q_2X$

Finally, we can implement both circuit of using D flip flop and J-K flip fop in LogicWorks 5. Connect the circuit with the flip flop, input X, LEDs and GND, PRE' and CLR' switch and connect to CLK signal. Thus, circuits will look as follow (Noted that it can be also implemented by using VDHL but for the sake of continuing on the next lab to be easier to do, we decided to implement it with logic gates. Also, the final circuit are shown in the lab result part):

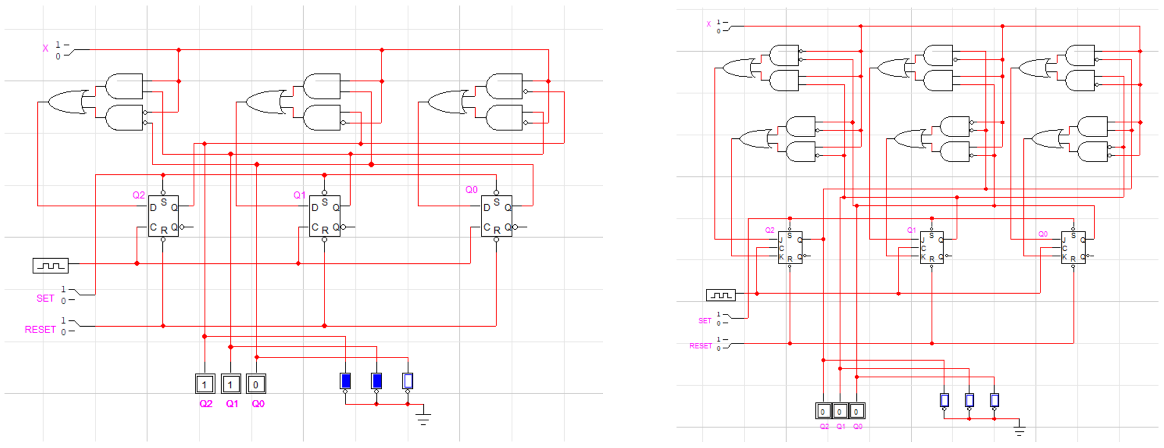


Figure 11: Running LEDs circuit using D flip flop (left) and using J-K flip flop (right)

> Lab 08: Running LEDs on Tinkercad

In this lab, we will continue from the previous lab by implementing the running LEDs circuit using D flip flop on TinkerCad. To implement the circuit, we can take a look by using the circuit and boolean expression that we already design in the previous lab and follow it.

First, to (maybe) make the process of creating a circuit easier, we will connect the circuit only the boolean expression part and connect its output to each LED. This will look like a combinational logic circuit. First, we place breadboards and hook them up to the power supply. Then, place DIP switch and connect the power and resistor to the switch. This is to simulate the input of and X. You may use the 6 DIP switch to prepare the CLK switch in the next step of circuit implementation. After that, we place ICs After that, we place Inverter (74HC04 in Tinkercad) as NOT gate, 74HC08 as AND gate and 74HC32 as OR gate. Connect the power and ground to all IC. Then, connect each input to each gate following the boolean expression we designed before. Lastly, we connect the output $Q_2Q_1Q_0$ from the combinational logic circuit to each LED. Connect LEDs with resistors. Then, test the output with LEDs.

After that, we will implement the circuit more by using it with D flip flop. D flip flop D flThe IC of D flip flop is 74HC74. In IC, it consist of 2 D flip flop which each flip flop has its inputs pin on one side of IC. To use 74HC74 in the running LEDs circuit, connect the power and ground as normally do with other ICs. Then, connect the CLK of each flip flop to the switch (we will connect to the CLK function generator later). Connect the PRE' and CLR' pin of each flip flop to the switch. Then, connect the output of each flip flop to the previous location of corresponding location in combinational circuit part. Then, connect the output of combinational circuit part to the input pin of each flip flop. Test the circuit once again and look at the LEDs whether the result display correct or not. After that, we place the device called function generator to generate clock signal for our flip flop. To use the CLK signal, connect the ground to the board and connect the function output to the CLK pin to all flip flops. Set function generator parameters as amplitude is 5 V, frequency as 2 Hertz (if want running LEDs to go faster, set the frequency higher than we set), and mode to digital signal. After all of that, test the output of the circuit. The result should follow the state diagram as we discuss and design in the previous lab. The final circuit should be as follows:

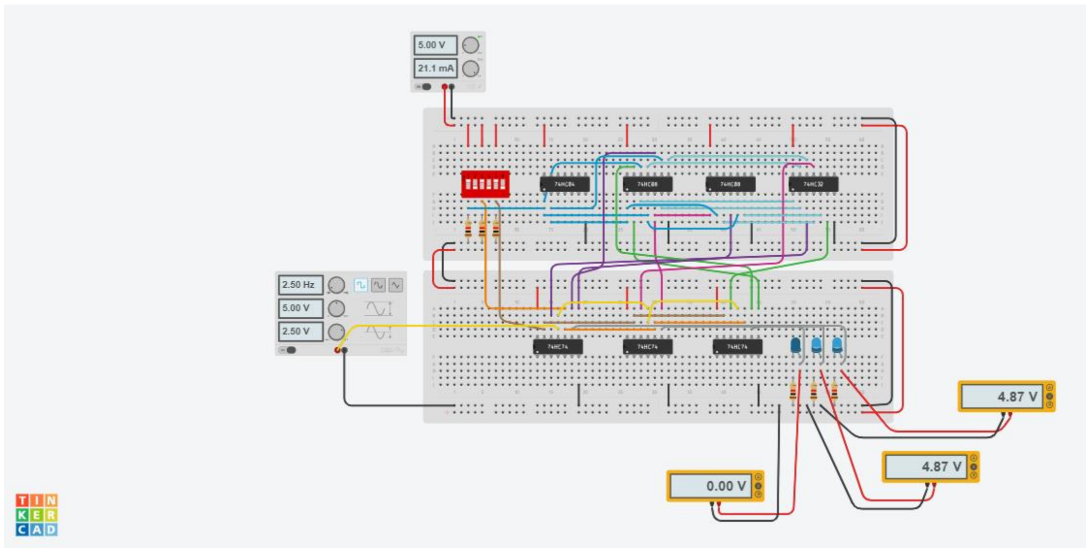


Figure 12: Running LEDs circuit using D flip flop (74HC74) in Tinkercad.

Conclusion

> Lab 06: Sequential Logic Circuit I

In this lab, we learn how to use each flip flop and understand its output, how to implement the flip flop into the circuit and observe output of J-K flip flop from circuit that we create in LogicWorks 5. Also, we learn to use create a vending machine by using D flip flop and VDHL in LogicWorks 5 by designing a state diagram and state table of finite state machine. Then, find Boolean expressions and implement it in the LogicWorks 5. This process can be used to design finite state machine as you want.

> Lab 07: Sequential Logic Circuit II (Running LEDs)

In this lab, we learn how to make state diagram and state table and use that information to fill out K-Maps corresponding to the transition table of D and J-K flip flop in order to find Boolean expressions for creating a circuit in LogicWorks 5. Then, in LogicWorks 5, In order to create a running LEDs circuit, we need to combine with previous lab knowledge about how to use D and J-K flip flop and how to connect LEDs to make the running LEDs work successfully.

> Lab 08: Running LEDs on TinkerCad

In this lab, we learned how create a running LEDs circuit using D flip flop (IC 74HC74) in Tinkercad by using all knowledge from previous labs and understand D flip flop pin configuration in order to create and implement this circuit. If you want to create finite state machine circuit, you can use this lab knowledge to create yours.