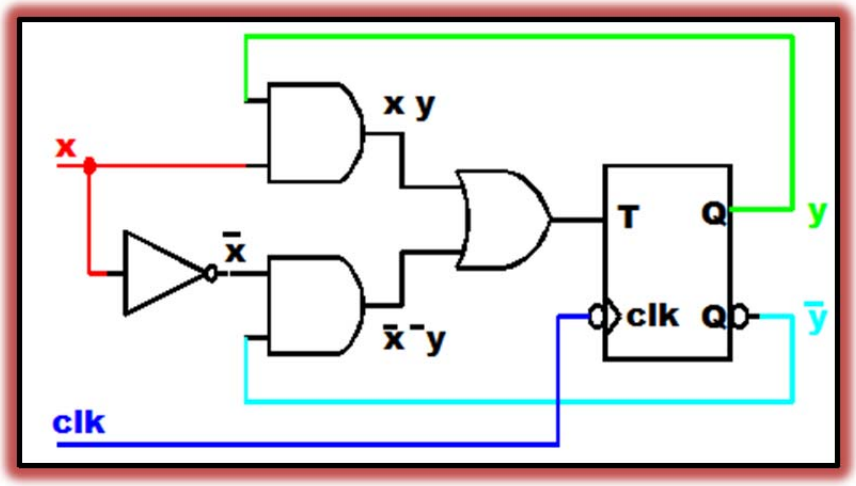
Next, let's reverse engineer a T-Flip flop Prob. (Pg 529)

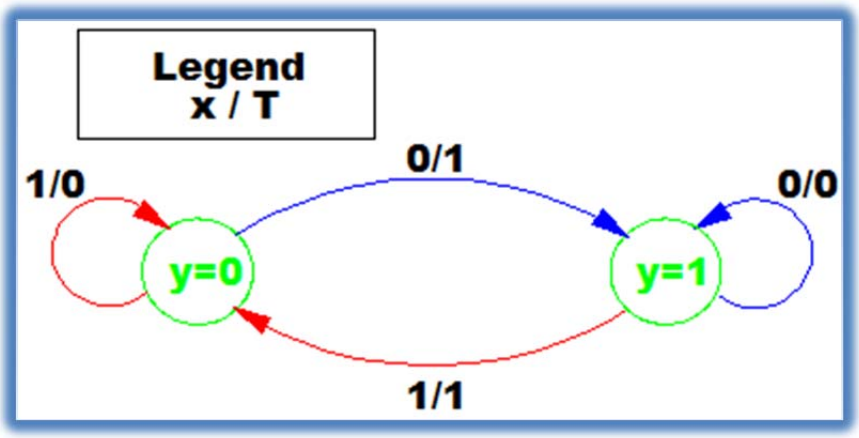


x	y	Y	T
0	0	1	1
0	1	1	0
1	0	0	0
1	1	0	1

Note that whenever T is equal to 1, there is a state change, otherwise, there isn't. In this circuit, **(x) determines whether the output sets or clears.**

$$T = x\ y + \bar{x}\ \bar{y} = \sum m(0,3)$$

From the table, we can now create the state diagram.



Let's make sure that the diagram is correct. **1 input => 2^1 = 2 exits** which is what you have.

Q: Examine the relationship between X and the Next state. How would you describe this circuit?

A: An inverter which is synched with a clock signal.

Example: Let's now design a **synchronous circuit** using **clocked T Flip-flops**

and the switching equations:

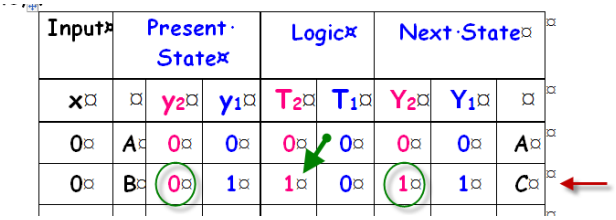
$$z = T_1 = x \bullet \bar{y}_2$$
$$T_2 = x \oplus y_1$$

		y_2	y_1
0	A	0	0
1	B	0	1
3	C	1	1
2	D	1	0

1st, find the state table using the state assignments shown to the right:

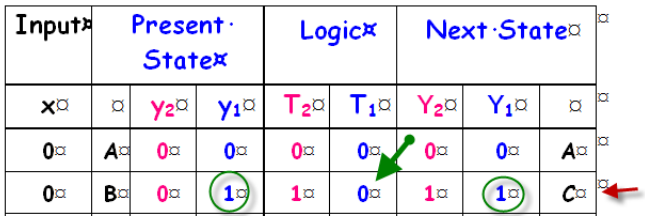
Note: When designing digital systems, if in doubt, start with a state table,

Input	Present State			Logic		Next State		
x		y_2	y_1	T_2	T_1	Y_2	Y_1	
0	A	0	0	0	0	0	0	A
0	B	0	1	1	0	1	1	C
0	D	1	0	0	0	1	0	D
0	C	1	1	1	0	0	1	B
1	A	0	0	1	1	1	1	C
1	B	0	1	0	1	0	0	A
1	D	1	0	1	0	0	0	A
1	C	1	1	0	0	1	1	C

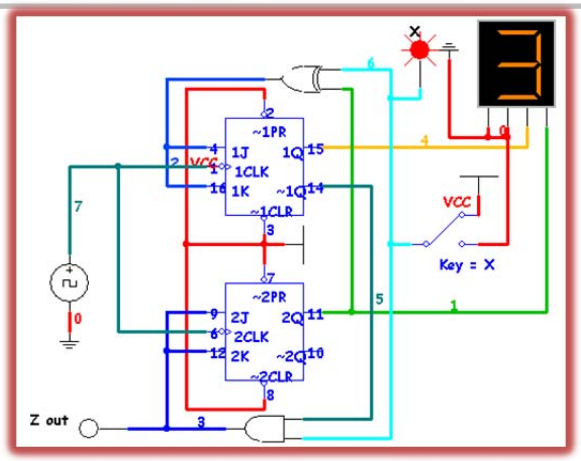


Let's examine a very small portion of the table. Note the '0' in the **MSB Present State** to the left. The associated **MSB Next State** value is a '1'. The only way that could happen is if the **T-FF** to which the states belong, had a '1' on its input to allow the toggle (see the **T-FF** transition table presented earlier).

Now look at the **LSB Present** and Next States. There isn't a change in state (they are both **1**'s), and the only way that could have happened is if the associated **T-FF** had a '0' on its **T** input (which would have forced the '**HOLD**' condition.



Finally, let's see what the circuit (the logic diagram) would look like.



The circuit would naturally change if different FF's are chosen. Whatever creates the minimum, most reliable, circuit should be what the designer should strive for!

Example: Design a standard, synchronous state machine which meets the following specifications:

When		
x=00	⇨	Hold in current state
x=01	⇨	Count 0-1-2-0-1-2 etc
x=10	⇨	Count 2-1-0-2-1-0 etc
x=11	⇨	Go to state 3

Quite often a **state diagram** can help make the leap from a set of count sequences to the **state table** much easier and more accurate. So, the 1st step this time is to create the **State Diagram** which demonstrates the specified state sequences from the specs. (NEXT PAGE)

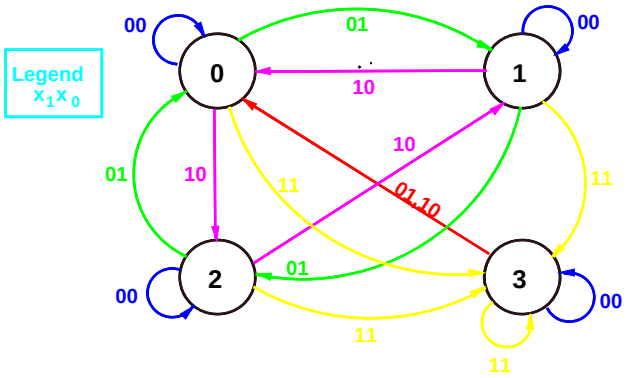
(**Note:** This is obviously the first design we worked with but we are going to take it further.)

- **Remember that we had to add two extra transitions from state 3 to state 0 (Could have chosen state 1 or 2 instead). The reason is that state 3 failed the # of exit arrows test.**

$$2^{\#\,inputs} = 2^2 = 4\ exits/state$$

- **Note that the legend indicates that the input is 2 bits wide.**
- **Note** It is not a good idea to use yellow in your diagrams, as can be seen here!)

Using the state assignment table and the state diagram, create the State Design Table. This is where the logic needed to implement the design in a circuit is determined. For this design a 'T-excitation table' as well as a 'JK-excitation table' are helpful.

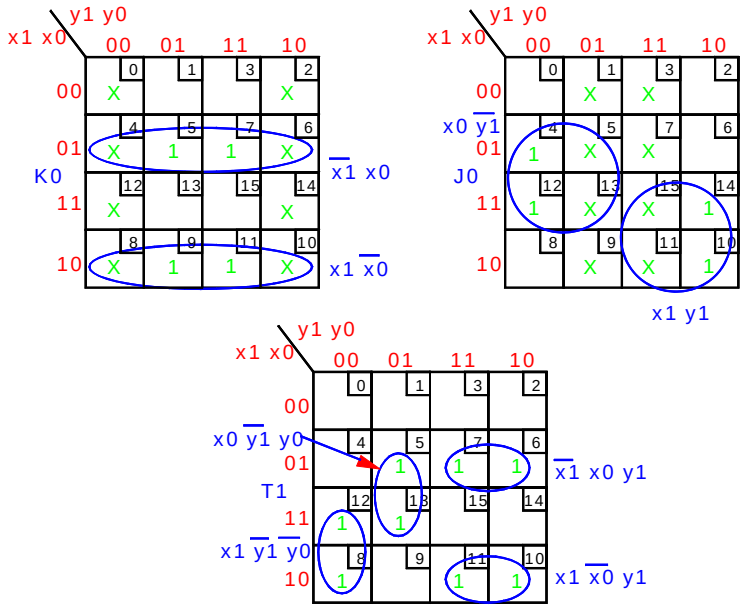


	Input		Present State		Next State		Logic		
	x_1	x_0	y_1	y_0	Y_1	Y_0	T_1	J_0	K_0
Hold	0	0	0	0	0	0	0	0	X
	0	0	0	1	0	1	0	X	0
	0	0	1	0	1	0	0	0	X
	0	0	1	1	1	1	0	X	0
Count up	0	1	0	0	0	1	0	1	X
	0	1	0	1	1	0	1	X	1
	0	1	1	0	0	0	1	0	X
	0	1	1	1	0	0	1	X	1
Count Down	1	0	0	0	1	0	1	0	X
	1	0	0	1	0	0	0	X	1
	1	0	1	0	0	1	1	1	X
	1	0	1	1	0	0	1	X	1
Goto state 3	1	1	0	0	1	1	1	1	X
	1	1	0	1	1	1	1	X	0
	1	1	1	0	1	1	0	1	X
	1	1	1	1	1	1	0	X	0

JK Transition Table

Q_p	Q_n	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

With the **State Design Table** completed, it is necessary to **K-map** each of the logic columns (**T, J,** and **K**) so that the gate structure necessary to control each **FF** can be determined.

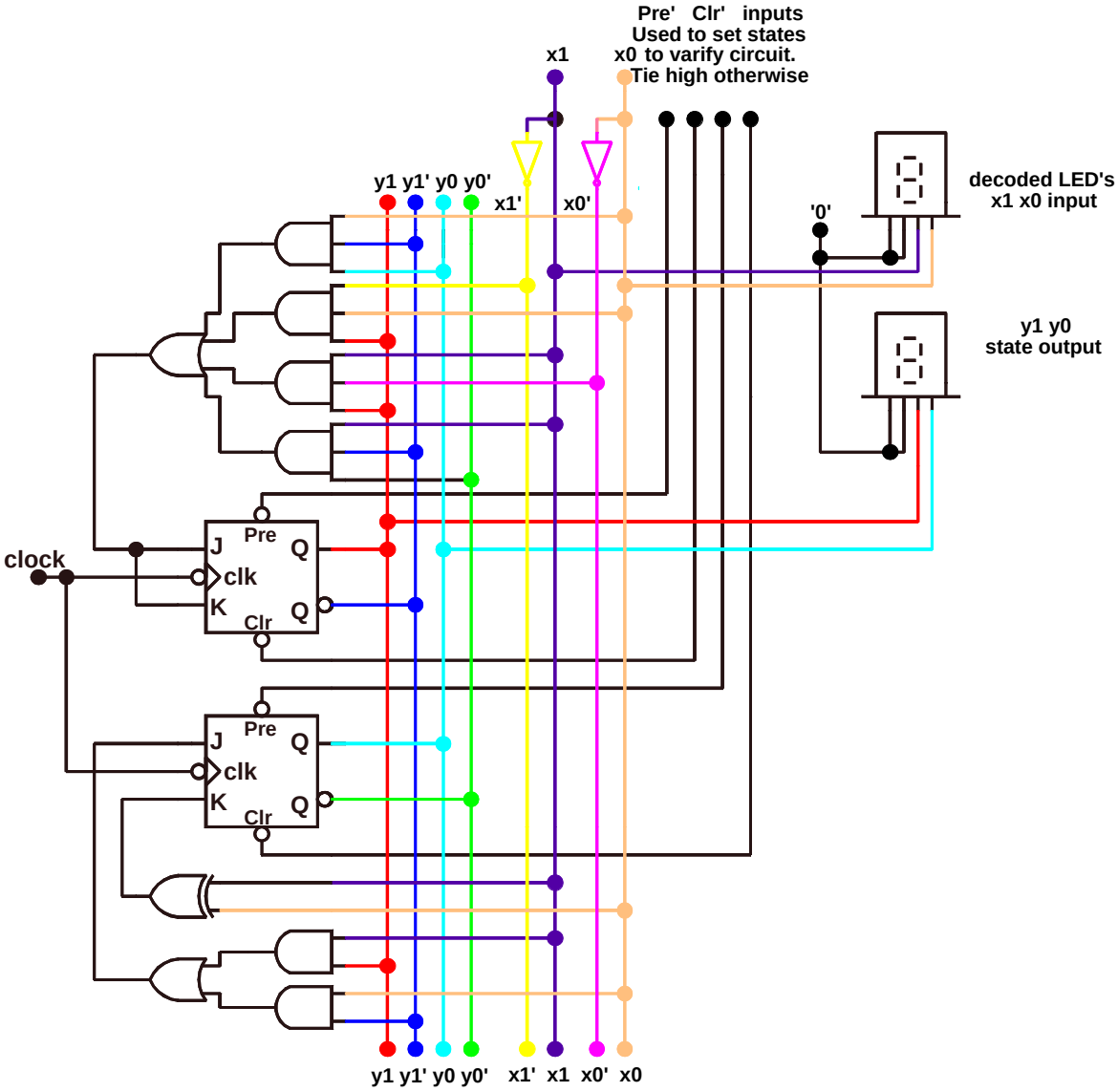


$$K_0 = \bar{x}_1 x_0 + x_1 \bar{x}_0 = x_1 \oplus x_0$$

$$J_0 = x_1 y_1 + x_0 \bar{y}_1$$

$$T_1 = x_0 \bar{y}_1 y_0 + \bar{x}_1 x_0 y_1 + x_1 \bar{x}_0 y_1 + x_1 \bar{y}_1 \bar{y}_0$$

The resulting circuit follows on the next page.


Pre' Clr' inputs
Used to set states
x0 to varify circuit.
Tie high otherwise
x1
y1 y1' y0 y0'
x1'
x0'
'0'
decoded LED's
x1 x0 input
y1 y0
state output
clock
J
Pre
Q
clk
K
Clr
Q
J
Pre
Q
clk
K
Clr
Q
y1 y1' y0 y0'
x1'
x1
x0'
x0

What a monster of a circuit! Can it be made easier? It's possible. We can make two changes.

- First, we can see what will happen if we use a JK FF vice a T FF for the MSB.
- The other thing is that we can use the illegal states which we noted (rows 7 and 11 in the table) to try and simplify the controlling expressions.

	Input		Present State		Next State		Logic			
	x_1	x_0	y_1	y_0	Y_1	Y_0	J_1	K_1	J_0	K_0
Hold	0	0	0	0	0	0	0	x	0	X
	0	0	0	1	0	1	0	x	X	0
	0	0	1	0	1	0	x	0	0	X
	0	0	1	1	1	1	x	0	X	0
Count Up	0	1	0	0	0	1	0	x	1	X
	0	1	0	1	1	0	1	x	X	1
	0	1	1	0	0	0	x	1	0	X
	0	1	1	1						
Count Down	1	0	0	0	1	0	1	x	0	X
	1	0	0	1	0	0	0	x	X	1
	1	0	1	0	0	1	x	1	1	X
	1	0	1	1						
Goto state	1	1	0	0	1	1	1	x	1	X
	1	1	0	1	1	1	1	x	X	0
	1	1	1	0	1	1	x	0	1	X
	1	1	1	1	1	1	x	0	X	0

JK Transition Table

Q_p	Q_n	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

We will leave the completion of this task to the student. Have Fun!!!