State Machine Design Example

Simplification by "Observation" - LONG HAND Method

Design Specifications:

- 3 bit counter
- Count Sequence: 6 - 3 - 1 - 4 - 6 - 3 - .....
- Begin the design with sending all illegal states to state 011 (3). You allowed to send any illegal to an intermediate illegal state as long as it ends up in a legal state within 2 clock cycles.
- Start the design with a D ff (MSB), a T ff, and a JK ff (LSB). Only the JK is mandatory after the first design attempt.

	Present State				Next State		
	A	B	C		A	B	C
0	0	0	0				
1	0	0	1	4	1	0	0
2	0	1	0				
3	0	1	1	1	0	0	1
4	1	0	0	6	1	1	0
5	1	0	1				
6	1	1	0	3	0	1	1
7	1	1	1				

Table 1

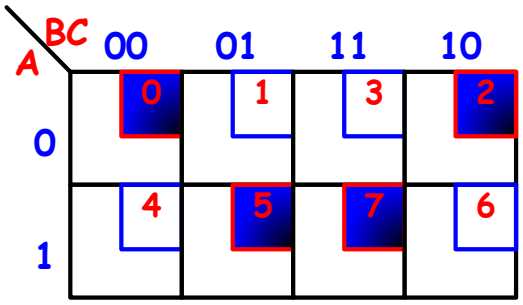
The first step is to set up the Present/Next State table. The Present State is always in binary order. The Next State side is dependent on the count sequence. Note that there are 4 blank rows in Table 1 to the left. These are illegal states. We must assign these states a pathway which will lead them to a legal state if by some chance the machine ends up in this illegal state. The most common reason for being in an illegal state is power up default conditions. This may be as simple as assigning them to go directly to a legal state. Or we might choose to send them to a legal state via an illegal state. Usually you start a design by picking one of the legal states to send them to. Later on you can change them if needed to simplify the circuit.

In the long hand method, each illegal state is initially sent to some legal state. Which state they are sent to is really not important since the assignments will change in the design process. As per the specs in this example, each illegal state is initially sent to state 3 as demonstrated in Table 2.

	Present State				Next State		
	A	B	C		A	B	C
0	0	0	0	3	0	1	1
1	0	0	1	4	1	0	0
2	0	1	0	3	0	1	1
3	0	1	1	1	0	0	1
4	1	0	0	6	1	1	0
5	1	0	1	3	0	1	1
6	1	1	0	3	0	1	1
7	1	1	1	3	0	1	1

Table 2

Table 2 can also be represented in the form of a K-map. Note that the K-map below does not have anything plotted in it since the table does not currently have any output columns.



You should also note that the illegal states, 0, 2, 5, and 7 have been identified in a manner which will make them easy to use to simplify any expressions which may result by reassigning illegal states.

The next step is to start designing for the flip-flop stages.

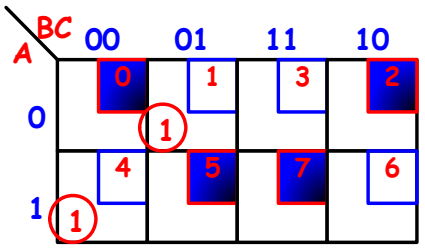
Designing the MSB flip-flop:

Let's start with the D_A stage. Remember that a D flip-flop will emulate the Next State column. The DA column is created by remembering that the Next State of a D flip-flop will follow whatever value is on the D input when the clock occurs. Thus, the D_A and Q_A columns are the same in Table 3 below.

	Present State				Next State			
	A	B	C		A	B	C	D_A
0	0	0	0	3	0	1	1	0
1	0	0	1	4	1	0	0	1
2	0	1	0	3	0	1	1	0
3	0	1	1	1	0	0	1	0
4	1	0	0	6	1	1	0	1
5	1	0	1	3	0	1	1	0
6	1	1	0	3	0	1	1	0
7	1	1	1	3	0	1	1	0

Table 3

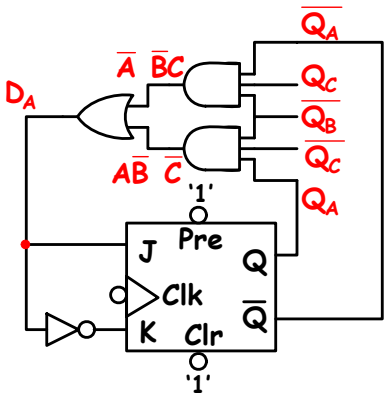
The next step is to K-MAP the D_A column and attempt to simplify it.

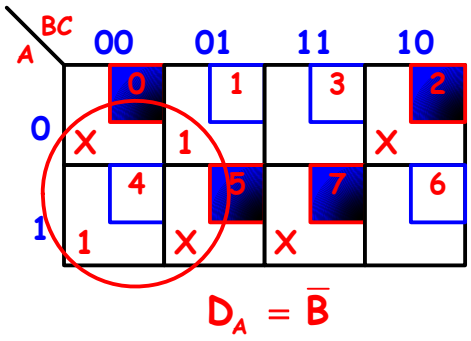


$$D_A = A\bar{B}\,\bar{C} + \bar{A}\,\bar{B}C$$

The simplification means that we can get the D flip-flop to act the way we want it to act to create the required sequence if we connect the following control circuitry to it.

A major goal of design is to keep the complexity of the control circuitry to a minimum. It is obvious that this circuit is a bit complicated. It would be nice if it would simplify to an equation with fewer gates. Let's add in the don't cares due to the illegal states.





Looking at the K-map, we note that if we include the don't cares in states 0 and 5, we can get a much simpler expression. Note that $\overline{B}$ is nothing more than a wire connecting the D input of the A flip-flop (the MSB) to the $\overline{Q}$ output of the B flip-flop (or $\overline{B}$)(the middle bit).

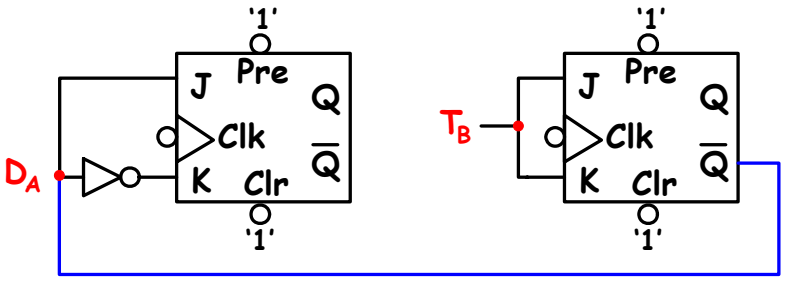
It is within our capability to make these substitutions since both 0 and 5 min-terms are illegal states. However, if we made these substitutions, we would be sending min-terms 0 and 5 to state 7 which is an illegal state as demonstrated in Table 4. Note that the D column and Next state column A have been corrected for the new situation.

	Present State			Next State				
	A	B	C		A	B	C	D_A
0	0	0	0	7	1	1	1	1
1	0	0	1	4	1	0	0	1
2	0	1	0	3	0	1	1	0
3	0	1	1	1	0	0	1	0
4	1	0	0	6	1	1	0	1
5	1	0	1	7	1	1	1	1
6	1	1	0	3	0	1	1	0
7	1	1	1	3	0	1	1	0

Table 4

It is reasonable to send an illegal state to a second illegal state which then goes to a legal state in most cases. In addition, who knows, the other bits in these two rows may change later causing them to be legal states. Note that the specifications for this design allows this to occur as long as it corrects itself to a legal state after two clock pulses.

As can be seen in the circuit below, we have exchanged an equation which required a 2-input OR gate and two 3-input AND gates with an equation consisting of a WIRE. Big savings!



Let's take a minute and discuss why the designer has chosen to use JK emulations of D and T flip-flops. There are several reasons. One could be the easy availability of JK's but the major reason is that it is desired to have all the FF's have the same timing and edge triggering. D ff's in particular are more likely to be found with leading edge triggering but even if a trailing edge trigger was found it still might be a faster or slower FF then the others. And finally, the JK is easier to design with because of the ability to use don't cares.

As a another side note, when is the most likely time for an illegal state to occur? The answer would be on circuit power-up. There are other times such as sun spot activity as well as "things just happen" activity. Whatever the cause, the system needs to have a path from any illegal state to a valid legal state or the circuit could become locked up!

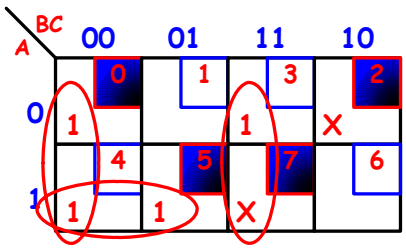
Designing the Middle Bit

Next, let's add in the T flip-flop column. Remember that if there is a change in Q_p to Q_n, T must have been a "1", otherwise it must have been a "0." See Table 5. When creating the T_B column, you are comparing the Present State B column with the Next State B column. If the state changes, T had to be a 1 for it to have happened. Otherwise, T had to be a 0.

As before, the next step is to plot the column into a K-MAP. This time we will go ahead and include the don't cares for the illegal states.

	Present State				Next State				
	A	B	C		A	B	C	D_A	T_B
0	0	0	0	7	1	1	1	1	1
1	0	0	1	4	1	0	0	1	0
2	0	1	0	3	0	1	1	0	0
3	0	1	1	1	0	0	1	0	1
4	1	0	0	6	1	1	0	1	1
5	1	0	1	7	1	1	1	1	1
6	1	1	0	3	0	1	1	0	0
7	1	1	1	3	0	1	1	0	0

Table 5

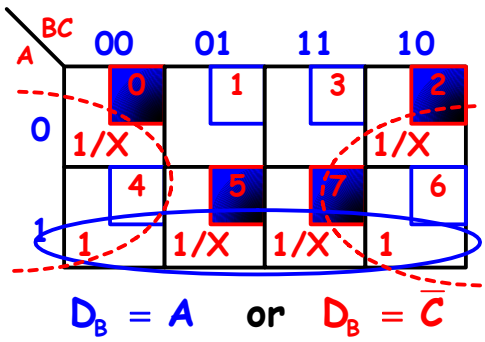


$$T_B = \overline{B}\,\overline{C} + BC + A\overline{B}$$

The resulting solution is one of several 3 term expressions which would describe the required control circuitry for the T input. Note that it isn't very simple and there isn't any way to use illegal states to simplify things. Since the specifications allow us to switch to a different flip-flop type in order to simply control circuitry, why not replace the T with a D ff?

A D_B column replaces the T_B column in Table 6. Again, remember that the D column and the Next State B column will be identical.

Again, plot the D_B column into a K-MAP, include the illegal state don't cares and simplify.



	Present State				Next State				
	A	B	C		A	B	C	D_A	D_B
0	0	0	0	7	1	1	1	1	1
1	0	0	1	4	1	0	0	1	0
2	0	1	0	3	0	1	1	0	1
3	0	1	1	1	0	0	1	0	0
4	1	0	0	6	1	1	0	1	1
5	1	0	1	7	1	1	1	1	1
6	1	1	0	3	0	1	1	0	1
7	1	1	1	3	0	1	1	0	1

Table 6

Note that simplification shows two different but equally simple answers. Both are just wires between different FF outputs to the D input. We will choose the $D_B = A$ answer for this design but the other answer should be recorded in the design journal just in case it is needed by other design processes later. Note that if we had not used the don't cares we would have ended up with an OR gate. I'll leave it to you to figure out what the OR expression would have been.

This stage isn't finished yet. Table 6 still needs to be modified to account for the new D and next state B columns and it needs to be checked to see if we have made a valid choice. (See Table 7.)

	Present State				Next State				
	A	B	C		A	B	C	D_A	D_B
0	0	0	0	5	1	0	1	1	0
1	0	0	1	4	1	0	0	1	0
2	0	1	0	1	0	0	1	0	0
3	0	1	1	1	0	0	1	0	0
4	1	0	0	6	1	1	0	1	1
5	1	0	1	7	1	1	1	1	1
6	1	1	0	3	0	1	1	0	1
7	1	1	1	3	0	1	1	0	1

Table 7

Rows 0 and 2 have been modified in Table 7 from 1's to 0's. This causes the next state for a present state 0 to become a 5 (an illegal state) while next state for a present state 2 is now a 1 (a legal state). Rows 5 and 7 were already 1's and therefore did not need to be modified.

We will hold our decision on if this is ok till bit *C* has been worked on. If we can't get row 0 to go to a legal state, we will have to step back and see what other choices we can make.

Designing the LSB bit (Bit *C*):

The only flip-flop left is the JK FF. In order to design with the JK, it is best to review the JK's transition table as shown in Table 8:

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

Table 8

With this transition table and present state and next state columns *C*, we can now complete the J_C and K_C columns in Table 9 on the next page.

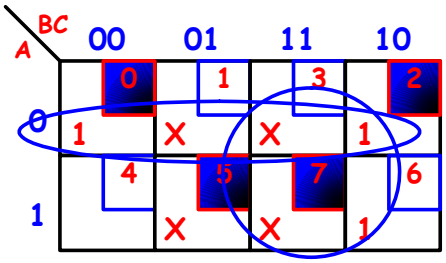
	Present State				Next State						
	A	B	C		A	B	C	D_A	D_B	J_C	K_C
0	0	0	0	5	1	0	1	1	0	1	X
1	0	0	1	4	1	0	0	1	0	X	1
2	0	1	0	1	0	0	1	0	0	1	X
3	0	1	1	1	0	0	1	0	0	X	0
4	1	0	0	6	1	1	0	1	1	0	X
5	1	0	1	7	1	1	1	1	1	X	0
6	1	1	0	3	0	1	1	0	1	1	X
7	1	1	1	3	0	1	1	0	1	X	0

Table 9

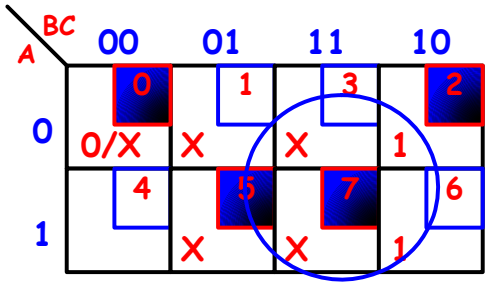
Q_p	→	Q_n	J	K
0	→	0	0	X
0	→	1	1	X
1	→	0	X	1
1	→	1	X	0

Table 10

The K-map to the right results in an OR gate which isn't to bad for a control circuit, however, we can do better. The illegal states are shown by the filled cells (0, 2, 5, 7). If we would change the min-term/illegal state cell 0 to a 0 or an X we could simplify the equation to a single wire again.

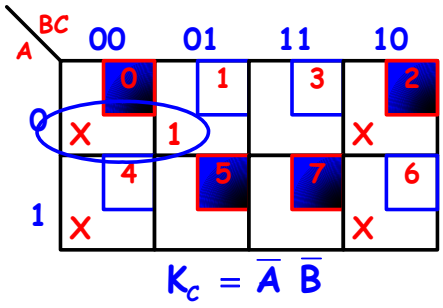


$J_C = \overline{A} + B$



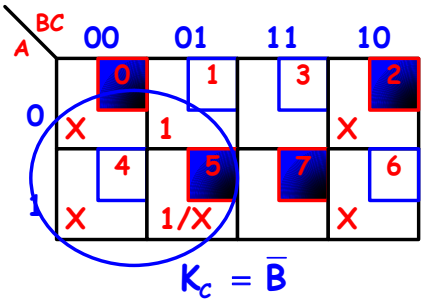
$J_C = B$

Note that we had to specify replacing it with a 0 or an X since this is a JK which could actually have X's in the columns. Before we update the table we can now simply K_C.



Again, we can take a look at this and see that we can use the don't care states to our advantage. If we change cell 5 from a 0 to a 1 or an X we get a big improvement.

Again we were lucky enough to simplify to a wire. Now let's update the table and check to see if the changes are valid.



	Present State			Next State							
	A	B	C		A	B	C	D_A	D_B	J_C	K_C
0	0	0	0	4	1	0	0	1	0	0	X
1	0	0	1	4	1	0	0	1	0	X	1
2	0	1	0	1	0	0	1	0	0	1	X
3	0	1	1	1	0	0	1	0	0	X	0
4	1	0	0	6	1	1	0	1	1	0	X
5	1	0	1	6	1	1	0	1	1	X	1
6	1	1	0	3	0	1	1	0	1	1	X
7	1	1	1	3	0	1	1	0	1	X	0

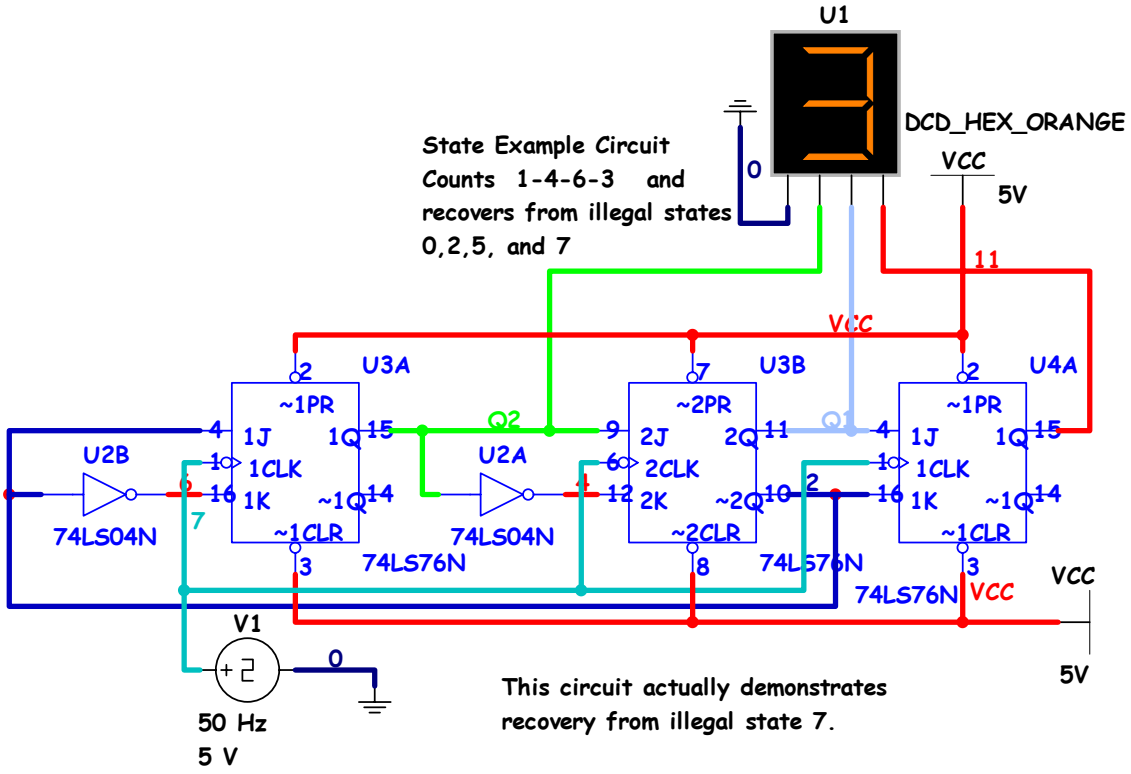
Table 12

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

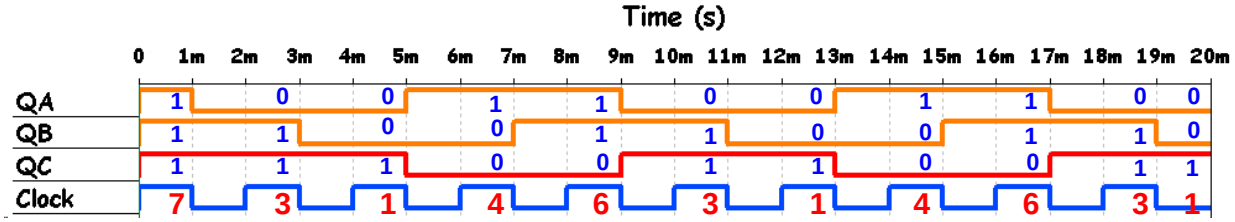
Table 11

Both changes cause state changes to legal states. We have a successful design!!!

Let's now simulate the circuit and test.



N



ote that not only does the graph demonstrate the 6314 sequence but it also demonstrates the predicted recovery path for the illegal state 7.

If further proof to recovery paths from other illegal states is desired, then the use of MultiSim's Word Generator will be useful. Just program in a "Jam Load" of an illegal state into the State Machine on the first program step and then make all the rest of the steps allow the state machine to count from the "Jam Loaded" value. The circuits CLR's and PRE's would be connected to the bit outputs of the Word Generator.

State Machine Design Example

Simplification by "Observation" SHORT HAND Method

Let's now reperform the last example. This time however, instead of assigning the illegal states to specific states, we are going to make then "don't cares" and use the states on the KMAPs directly to minimize our solution. Since we already know that the T ff is not going to work we will go directly to a second D-FF.

Design Specifications:

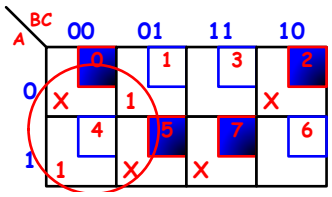
- 3 bit counter
- Count Sequence: 6 - 3 - 1 - 4 - 6 - 3 - .....
- Begin the design with sending all illegal states to state 011 (3). You allowed to send any illegal to an intermediate illegal state as long as it ends up in a legal state within 2 clock cycles.
- Start the design with a D ff (MSB), a D ff, and a JK ff (LSB).

Design the MSB:

	Present State				Next State			
	A	B	C		A	B	C	D_A
0	0	0	0					X
1	0	0	1	4	1	0	0	1
2	0	1	0					X
3	0	1	1	1	0	0	1	0
4	1	0	0	6	1	1	0	1
5	1	0	1					X
6	1	1	0	3	0	1	1	0
7	1	1	1					X

Table 13

Note that this time the illegal states are empty and "Don't Cares" have been placed in the DA column. Lets go straight to the K-map.



It can now be easily seen that the simplest result will be $D_A = \overline{B}$

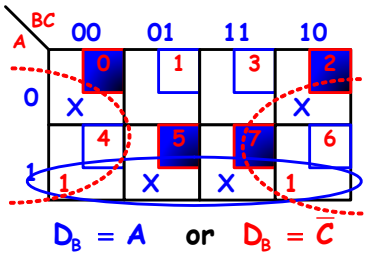
This also means that Cells 0 and 5 in the D column will be considered to be 1's while Cells 2 and 7 will be considered 0's. See Table 14 on the next page.

	Present State				Next State			
	A	B	C		A	B	C	D_A
0	0	0	0		1			X(1)
1	0	0	1	4	1	0	0	1
2	0	1	0		0			X(0)
3	0	1	1	1	0	0	1	0
4	1	0	0	6	1	1	0	1
5	1	0	1		1			X(1)
6	1	1	0	3	0	1	1	X(0)
7	1	1	1		0			0

Table 14

This table shows the Table after what I call "**Rationalizing the 'Don't Cares'** " occurs. Note that the **X's** are maintained in the D column for record purposes. **The chosen value for this X is in the parenthesis.**

The Middle Bit (again a D-ff):



$D_B = A$ or $D_B = \overline{C}$

As before, we end up with two choices and the $D_B = A$ choice is the one that gets used. See Table 16.

	Present State				Next State				
	A	B	C		A	B	C	D_A	D_B
0	0	0	0		1			X(1)	X
1	0	0	1	4	1	0	0	1	0
2	0	1	0		0			X(0)	X
3	0	1	1	1	0	0	1	0	0
4	1	0	0	6	1	1	0	1	1
5	1	0	1		1			X(1)	X
6	1	1	0	3	0	1	1	0	1
7	1	1	1		0			X(0)	X

Table 15

	Present State				Next State				
	A	B	C		A	B	C	D_A	D_B
0	0	0	0		1	0		X(1)	X(0)
1	0	0	1	4	1	0	0	1	0
2	0	1	0		0	0		X(0)	X(0)
3	0	1	1	1	0	0	1	0	0
4	1	0	0	6	1	1	0	1	1
5	1	0	1		1	1		X(1)	X(1)
6	1	1	0	3	0	1	1	0	1
7	1	1	1		0	1		X(0)	X(1)

Table 16

Again, the "Don't Cares" have been rationalized.

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

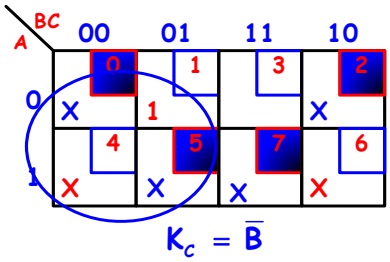
Table 17

Design the LSB (JK-FF):

The J and K columns are worked out a bit different. We will look at the **J** and **K** columns at the same time:

	Present State				Next State						
	A	B	C		A	B	C	D_A	D_B	J_C	K_C
0	0	0	0		1	0		X(1)	X(0)	X	X
1	0	0	1	4	1	0	0	1	0	X	1
2	0	1	0		0	0		X(0)	X(0)	X	X
3	0	1	1	1	0	0	1	0	0	X	0
4	1	0	0	6	1	1	0	1	1	0	X
5	1	0	1		1	1		X(1)	X(1)	X	X
6	1	1	0	3	0	1	1	0	1	1	X
7	1	1	1		0	1		X(0)	X(1)	X	X

Table 18



$K_C = \overline{B}$

Note that as before, $J_C = B$ **and** $K_C = \overline{B}$ works out as the answers.

From these K-maps, we now know that the X's will be:

J Column - either be **kept as an X** or **turned into a 1** for **cells 2 and 7** "**X(1)=?**"
- either be **kept as an X** or **turned into a 0** for **cells 0 and 5** "**X(0)=?**"

K Column - either be **kept as an X** or **turned into a 1** for **cells 0 and 5** "**X(1)=?**"
- either be **kept as an X** or **turned into a 0** for **cells 2 and 7** "**X(0)=?**"

	Present State				Next State						
	A	B	C		A	B	C	D_A	D_B	J_C	K_C
0	0	0	0		1	0		X(1)	X(0)	X(0)=?	X(1)=?
1	0	0	1	4	1	0	0	1	0	X	1
2	0	1	0		0	0		X(0)	X(0)	X(1)=?	X(0)=?
3	0	1	1	1	0	0	1	0	0	X	0
4	1	0	0	6	1	1	0	1	1	0	X
5	1	0	1		1	1		X(1)	X(1)	X(0)=?	X(1)=?
6	1	1	0	3	0	1	1	0	1	1	X
7	1	1	1		0	1		X(0)	X(1)	X(1)=?	X(0)=?

Table 19

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

Table 20

Lets see how we accomplish this.

Look at **illegal state 0**: **Bit C** has a **Present State of 0**. If we look at Table 20 above, that means we are limited to the top two rows. Next we note that for those two rows, **K HAS to be an X**, therefore the **K** for this row has to be an **X**. Then we note that **J** for this minterm is either an **X** or a **0**. Since the **X is already used by K** then we are now left with a **0 for J**. So, for that row, we get (Table 21):

Present State				Next State							
	A	B	C		A	B	C	D_A	D_B	J_C	K_C
0	0	0	0	4	1	0	0	X(1)	X(0)	X(0)=0	X(1)=X

Table 21

Note that from Table 20, a **J/K of 0/X** will cause a **transition from 0 to 0**. So, for this row, minterm **0 will transition to minterm 4**, a "**LEGAL**" state.

Present State				Next State							
	A	B	C		A	B	C	D_A	D_B	J_C	K_C
0	0	0	0	4	1	0	0	X(1)	X(0)	X(0)=0	X(1)=X
2	0	1	0	1	0	0	1	X(0)	X(0)	X(1)=1	X(0)=X

Table 23

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

Table 22

Now we will work out **minterm 2's transition**. Again, **Bit C's Present State** is a **0** so we are still limited to the first two rows of Table 23 above. Just as before, **K is required to be an X** but now **J can either be an X or a 1**. Since K has X taken care of, then **J must be a 1** and minterm **2 will now transition to minterm 1**, a "**LEGAL**" State.

Present State				Next State							
	A	B	C		A	B	C	D_A	D_B	J_C	K_C
5	1	0	1	6	1	1	0	X(1)	X(1)	X(0)=X	X(1)=1

Table 24

It is now **minterm 5's turn**. **Bit C's Present State is a 1** so we are limited to Table 23's bottom two rows. Note from these last two rows that **J now has to be an X**. With that in mind we note from Table 24 that **K is shown as either an X or**

a 1. Since J is already an X, that leaves us **K as a 1.** Then again observing Table 23, a J/K of **X/1 provides for a transition from 1 to 0**. With that entered into Table 24 we note that **minterm 5 will now transition to minterm 6**, a "**Legal**" State.

	Present State				Next State						
	A	B	C		A	B	C	D_A	D_B	J_C	K_C
7	1	1	1	3	0	1	1	X(0)	X(1)	X(1)=X	X(0)=0

Table 25

Finally we are left with the last illegal state transition (minterm 7). **Bit C's Present State is a 1** so we are limited to Table 23's bottom two rows. Note from these last two rows that **J again has to be an X.** With that in mind we note from Table 24 that **K is shown as either an X or a 0**. Since J is already an X, that leaves us **K as a 0.** Observing Table 23, a J/K of **X/0 provides for a transition from 1 to 1**. With that entered into Table 24 we note that **minterm 7 will now transition to minterm 3**, a "**Legal**" State. The table is now completed and we can show the results in Table 26 on the next page.

Present State				Next State							
	A	B	C		A	B	C	D_A	D_B	J_C	K_C
0	0	0	0	4	1	0	0	X(1)	X(0)	X(0)=0	X(1)=X
1	0	0	1	4	1	0	0	1	0	X	1
2	0	1	0	1	0	0	1	X(0)	X(0)	X(1)=1	X(0)=X
3	0	1	1	1	0	0	1	0	0	X	0
4	1	0	0	6	1	1	0	1	1	0	X
5	1	0	1	6	1	1	0	X(1)	X(1)	X(0)=X	X(1)=1
6	1	1	0	3	0	1	1	0	1	1	X
7	1	1	1	3	0	1	1	X(0)	X(1)	X(1)=X	X(0)=0

Table 26

If you compare Table 26 with Table 12 you will find that the two methods result in the same answer. While this method might look longer, it is only because it has been broken up into small pieces to make discussion easier. It is my hope that you will actually find this method to be easier and more methodical then the first method.

Both methods require one final "dummy check". Make sure that all illegal states now have some pathway to a legal state. If not, its back to the drawing board for at least one of the bits for that state.

Happy Designing!!