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 by sending all illegal states to state 011 (3). You are allowed to send any illegal state to an intermediate illegal state as long as it ends up in a legal state within two 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 four blank rows in **Table 1** to the left. These are **illegal states (not defined by the specifications). 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. Or, the more advanced designer with designate them with 'Don't Cares' and take care of the level assignment "on-the-fly" as the design goes along (Short-Hand Method notes).

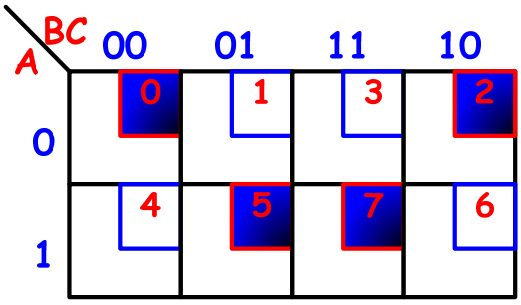
	Present State				Next State		
	A	B	C		A	B	C
0	0	0	0	3	0	1	1
1	0	0	1	4	1	0	0
2	0	1	0	3	0	1	1
3	0	1	1	1	0	0	1
4	1	0	0	6	1	1	0
5	1	0	1	3	0	1	1
6	1	1	0	3	0	1	1
7	1	1	1	3	0	1	1

Table 2

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.**

The table as shown is the 1st step towards designing the required **control circuitry** for each FF in the design. Each set **of control circuitry** will be based upon an output column which results from the relationship between a **Present State** column and its associated **Next State** column.

Each output column will be able to be **k-mapped** using the k-map form shown to the right



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 the logic which will control the actions of each flip-flop in the state machine.

Designing the MSB flip-flop:

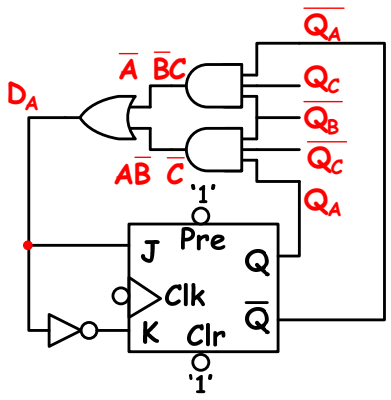
	Present State				Next State			
	A	B	C		A	B	C	D_A
0	0	0	0	3	0	1	1	0
1	0	0	1	4	1	0	0	1
2	0	1	0	3	0	1	1	0
3	0	1	1	1	0	0	1	0
4	1	0	0	6	1	1	0	1
5	1	0	1	3	0	1	1	0
6	1	1	0	3	0	1	1	0
7	1	1	1	3	0	1	1	0

Table 3

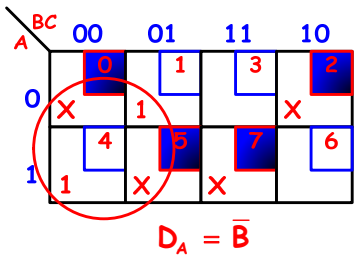
Let's start with the D_A stage. The D_A column is created by remembering that the **Next State** of a **D flip-flop** will follow whatever value is on the **D input** when the clock occurs. Thus, the D_A and Q_A columns are the same in **Table 3.**

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

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. Currently, the illegal states are all **0**'s. Let's change them to **"don't-cares"**.



Looking at the new K-map, we note that the included "don't-cares" in states **0** and **5**, provide a simpler control 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 ($\overline{B}$) **(the middle bit).**

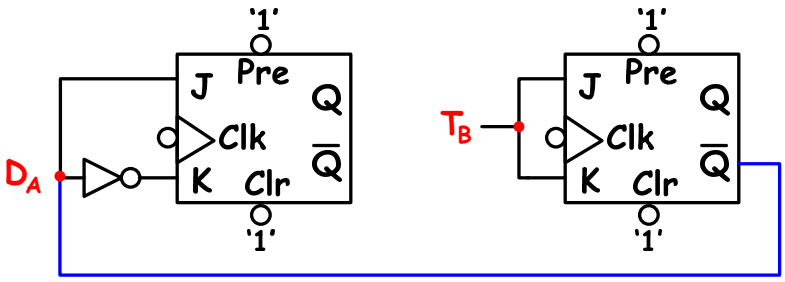
It is within our capability to make these substitutions since both **0** and **5 min-terms** are **illegal states**. However, if we made these substitutions, we would be sending **min-terms** 0 and 5 to state 7 which is an illegal state as demonstrated in **Table 4.** Note that the **D column** and **Next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

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 as well, resulting in legal states. Note that the specifications for this particular design allow this to occur as long as it corrects itself to a legal state after two clock pulses.

	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

As can be seen in the circuit shown, 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 lower cost 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 another side note:

Question: When is the most likely time for an illegal state to occur?

Answer: 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!

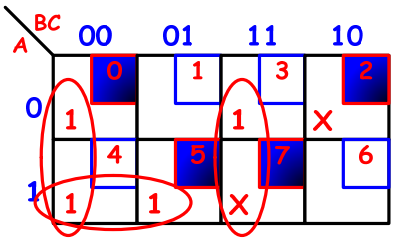
Designing the Middle Bit

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



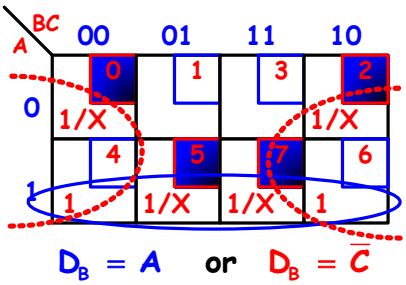
$T_B = \bar{B}\,\bar{C} + BC + A\bar{B}$

The resulting solution is one of several three 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	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 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 8 below)

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 7:

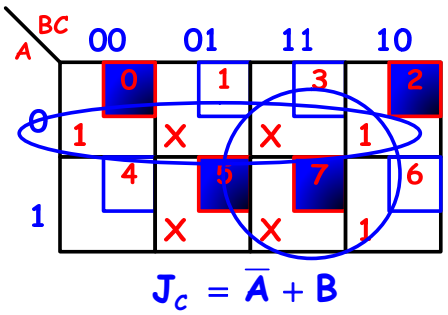
Q_p	→	Q_n	J	K
0	→	0	0	X
0	→	1	1	X
1	→	0	X	1
1	→	1	X	0

Table 7

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 8

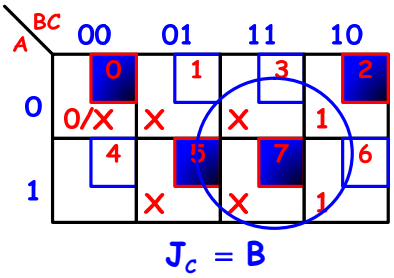
	Present State				Next State						
	A	B	C		A	B	C	D_A	D_B	J_C	K_C
0	0	0	0	5	1	0	1	1	0	1	X
1	0	0	1	4	1	0	0	1	0	X	1
2	0	1	0	1	0	0	1	0	0	1	X
3	0	1	1	1	0	0	1	0	0	X	0
4	1	0	0	6	1	1	0	1	1	0	X
5	1	0	1	7	1	1	1	1	1	X	0
6	1	1	0	3	0	1	1	0	1	1	X
7	1	1	1	3	0	1	1	0	1	X	0

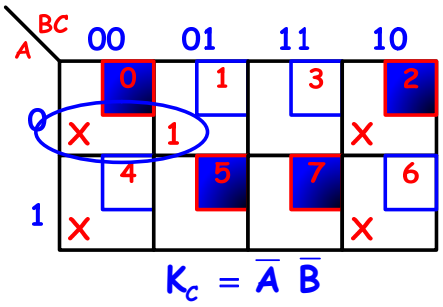
Table 8



This k-map results from the direct k-mapping of the JC column without taking into account the illegal state don't-cares.

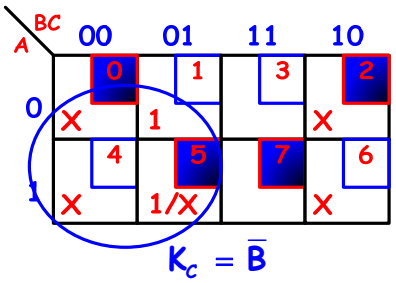
The J_C K-map to the right results another wire when the don't-cares are used.





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 simplify K_C.

Again, we can take a look at the "don't care" states to our 5 from a 0 to a 1 or an X we



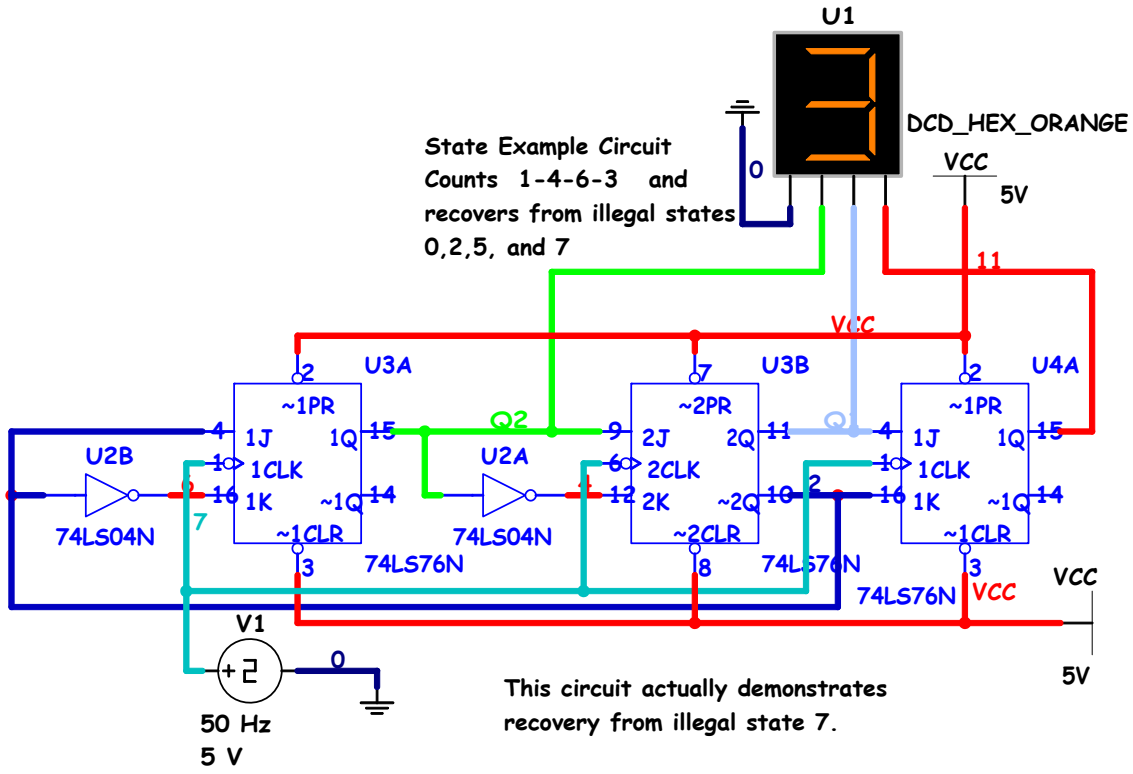
this and see that we can use advantage. If we change cell get a big improvement.

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.

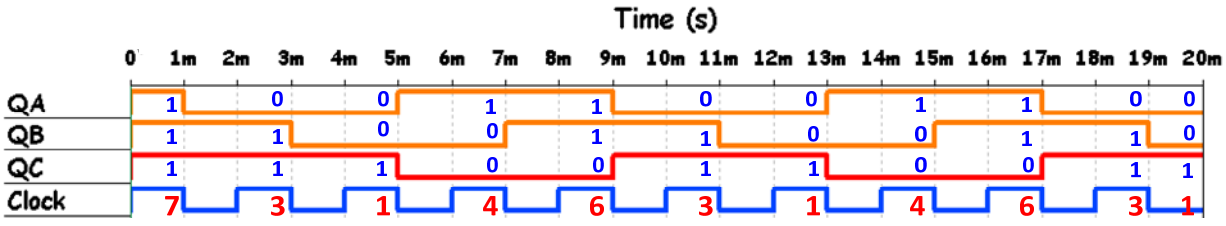
	Present State				Next State			Control Logic			
	A	B	C		A	B	C	D_A	D_B	J_C	K_C
0	0	0	0	4	1	0	0	1	0	0	X
1	0	0	1	4	1	0	0	1	0	X	1
2	0	1	0	1	0	0	1	0	0	1	X
3	0	1	1	1	0	0	1	0	0	X	0
4	1	0	0	6	1	1	0	1	1	0	X
5	1	0	1	6	1	1	0	1	1	X	1
6	1	1	0	3	0	1	1	0	1	1	X
7	1	1	1	3	0	1	1	0	1	X	0

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

Let's now simulate the circuit and test.



Note 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 required to allow the state machine to count from the "Jam Loaded" value. The circuit's CLR's and PRE's would be connected to the bit outputs of the Word Generator.