

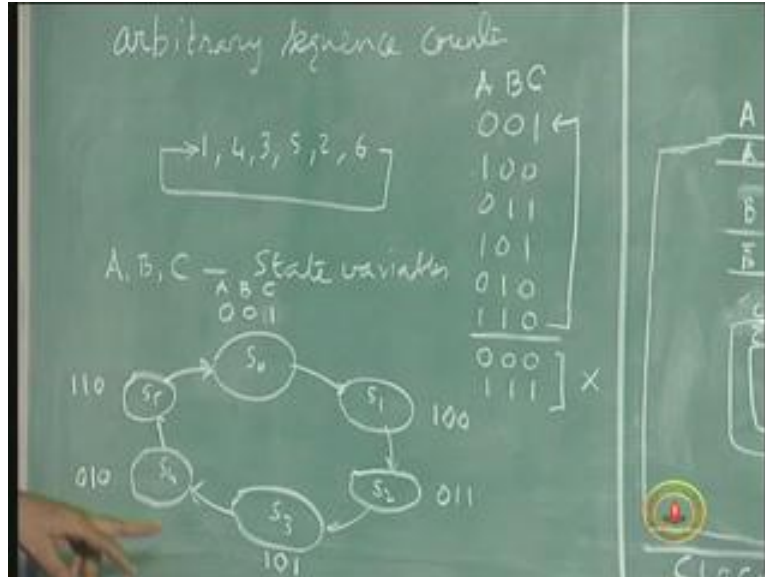
**Digital Circuits and Systems**  
**Prof. S. Srinivasan**  
**Department of Electrical Engineering**  
**Indian Institute of Technology, Madras**  
**Lecture # 25**  
**Design Using J-K Flip-Flop**

(Refer Slide Time: 2:11)



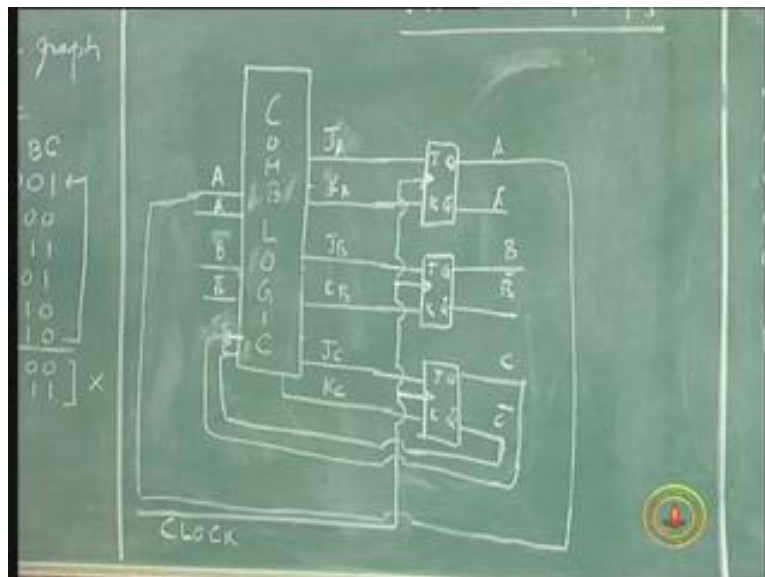
We are discussing the implementation of state graph so go through a state table and from there to a combinational logic which will drive the flip-flops so that the state transitions will occur as per the state graph. So basically any circuit can be represented as a state graph in which several states are defined and the transition from any state to the next state is determined by the present state as well as the input variables.

(Refer Slide Time: 3:25)



Now as an example we took the case of a counter with no external input. A counter is not a sequential counter but an arbitrary sequence counter at six states  $S_0 S_1 S_2 S_3 S_4 S_5$  and it had to go through this sequence and since there are no external inputs it is assumed that the circuit remains in each state for one clock period and at the end of that clock period it goes to next state. So what we have to decide is from this state to this state transition what inputs are to be given to the flip-flops there are six states so it requires a maximum of three flip-flops and with three flip-flops we can have eight states and these three flip-flops have to be steered from state to state based on the present state values and the sequence required.

(Refer Slide Time: 4:15)



First exercise we did was using D flip-flops where the next state variables are same as the D input. But if you want a J-K Flip-Flop to be used we need to find out what J and what K has to be given for each flip-flop in order to change it from a given present state to a given next state.

(Refer Slide Time: 6:55)

STATE TABLE			
J	K	Q	Q <sup>+</sup>
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Q	Q <sup>+</sup>	J	K
0	0	0	0
0	1	1	0
1	0	0	1
1	1	0	0

Characteristic Table 2  
J-K FF

Excitation Table

We introduced a concept called excitation table. if we take the J-K Flip-Flop this is the table of behavioral of flip-flops called characteristic table, the flip-flop behaves in a memory state, reset state, set state and toggle state given the input J and K and a value of Q what will be the next value of Q, this is called the characteristic table or similar to a truth table. But what we want is an inverse of this, what we want is if a flip-flop stays in a particular state and if it has to go the next state what inputs have to be given to J and what input to be given to K in order to steer a flip-flop from a given present state to a given next state. So this is an inverse table of course the information is here in this table,

I have to rewrite it in a way such that what values of J and K will steer the flip-flop from a given present state to a given next state. There are only two present states possible 0 and 1 and 2 next states are possible 0 and 1 so there are only four transitions possible. The present state can be 0 next state can be 0 present state can be 0 next state can be 1 present state can be 1 next state can be 0 present state is 1 and next state is also 1.

In each of these four transitions what j and what k will make this possible so such a table is called an excitation table. How do you excite the flip-flop or the inputs in the flip-flop, how do you steer the flip-flop from state to state so that's what we set about doing for a J-K Flip-Flop taking this as the characteristic table of the flip-flop what change of values of J and K will remain in state 0 will make it go from 0 to 1 will make it go from 1 to 0 will make it remain in 1 that can be obtained from here.

For example; the present state is 0 the next state is 0 (Refer Slide Time: 6:55) so these two rows of the truth table or the characteristic table behavioral table tells you that if the

present state is 0 and next state is 0 what inputs are required input has to be 0 0 or 0 1 so J has to be 0 and K can be either 0 or 1. If J is equal to 0 and K can be 0 or 1 that means k is don't care, a J-K Flip-Flop will make a 0 to 0 transition. So the next is 0 to 1. This is 0 to 1, 0 to 1 these two rows correspond to the transition of a present state 0 to next state 1. In both these cases J is equal to 1 K is equal to 0 here K is 1 here so J has to be 1 and K is don't care which makes a transition of 0 to 1 possible, next is 1 0, present state is 1 next state is 0 this row and this row, present state is 1 next state is 0, J is 0 here and 1 here (Refer Slide Time: 8:25) whereas K is equal to 1 in both cases so k has to be 1 necessarily and J can be either 0 or 1. And finally a one to one transition occurs. The present state is 1 and next state is also 1 will happen if both J and K are 0 or if J is equal to 1 and K is equal to 0 that means J is don't care and K is equal to 0.

(Refer Slide Time: 9:12)

The image shows a chalkboard with two tables. The left table is the State Table, and the right table is the Excitation Table.

**STATE TABLE**

J	K	Q	Q <sup>+</sup>
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

**Excitation Table of J-K Flip**

Q	Q <sup>+</sup>	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Characteristics Table of J-K FF

So this table is called the excitation table of J-K Flip-Flop. So, we are going to design a steering logic in any state machine which is represented by a state graph, we need to design a combinational logic called steering logic which will take these circuits from one state to the next to the next to the next as given by the requirement, the sequence required. So I need to design a steering logic for these flip-flops so that given a present state value what will be the next state so what should be given to the inputs of each of these flip-flops so that the next state will be as desired. So my steering logic will have to necessarily take the excitation information of the flip-flops into account before I can write the combinational logic table. This is the combinational logic table which is also called state table because this is going from state graph to state table.

(Refer Slide Time: 10:19)

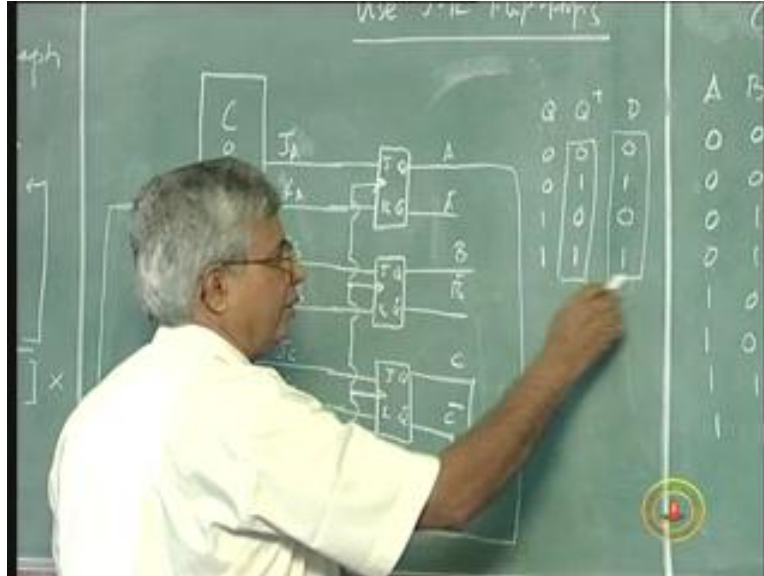


In D flip-flop we didn't have to do this because D flip-flop the output is same as the input next state is same as the present input. If D is 1 the next state is 1 and D is 0 the next state is 0. So it was indirectly done we didn't do explicitly. we did for the next states, we did the map for next state for flip-flop A, next state for flip-flop B, next state for flip-flop C for the previous example where we used D flip-flops we wrote corner maps for DA DB DC because the next states is same as the inputs, it was built into that mechanism so we didn't have to explicitly write the excitation table and then get a steering logic. for a J-K Flip-Flop its not the case in J-K Flip-Flop I am going to use the same example that we used earlier of this transition 0 0 1 to 1 0 0 to 0 1 1 to 1 0 1 0 1 0 to 1 1 0 back to 0 0 1 this is the transition required and from the present state 0 0 1 to next state 1 0 0 flip-flop A has to change from 0 to 1, B has to remain in 0, C has to change from 1 to 0 so what excitation  $J_A$  and  $K_A$  will give in the present state 0 0 1 so that next state of this will be 1, next state of this will be 0 and the next state of this will be 0.

**For what** excitation I will give for  $J_A$   $K_A$   $J_B$   $K_B$   $J_C$  so that the next state of A will become 1, the next state of B will remain 0, the next state of C will become 0 for 1. So my table has to have that information here  $J_A$   $K_A$   $J_B$   $K_B$   $J_C$   $K_C$  and that will be simplified and that will be the steering logic now, this combinational logic which is going to steer the flip-flop from state to state. Earlier we had only DA DB DC the three outputs corresponding to three present states and these outputs were directly got from the map because D and Q were the same because the excitation table of a D flip-flop is....., **I am going to draw the excitation**, the present state is Q, next state is Q power plus so what should be D, so if this is 0 (Refer Slide Time: 12:39) the next state is 0 so D has to be 0 and if the present state is 0 next state is 1 so D has to be 1 only then it will change from 0 to 1, and if the present state is 1 and the next state is 0 then D has to become 0. Only when D is 0 Q power plus will become 0.

So what is the need for writing a table separately because this is same as Q power plus. The next state value of Q is same as the D input required.

(Refer Slide Time: 13:06)



Because of this nature I didn't have to explicitly tell you about an excitation table for a D flip-flop, I have directly taken that into account in my table and simplified the table to get the logic. Here I cannot do that, here it's  $J_A$   $K_A$   $J_B$   $K_B$   $J_C$   $K_C$  they are different for each state now transition to transition so I have to fill this table with the input values of  $J_A$   $K_A$   $J_B$   $K_B$   $J_C$   $K_C$  and then simplify each one of them and draw my steering logic which will take the flip-flops from each state to different states as required in the problem. So let us complete that exercise here.

(Refer Slide time: 14:00)

The image shows a handwritten table on a chalkboard. The table is divided into two main sections: 'COMB LOGIC Truth Table' and 'STATE TABLE'. The 'COMB LOGIC Truth Table' has columns for inputs A, B, and C, and outputs J<sub>A</sub>, K<sub>A</sub>, J<sub>B</sub>, K<sub>B</sub>, J<sub>C</sub>, and K<sub>C</sub>. The 'STATE TABLE' has columns for the next state values A, B, and C. The table is filled with 'x' marks, indicating that the next state is not defined for certain combinations of the present state.

COMB LOGIC Truth Table						STATE TABLE		
A	B	C	J <sub>A</sub>	K <sub>A</sub>	J <sub>B</sub>	K <sub>B</sub>	J <sub>C</sub>	K <sub>C</sub>
0	0	0	x	x	x	x	x	x
0	0	1						
0	1	0						
0	1	1						
1	0	0						
1	0	1						
1	1	0						
1	1	1	x	x	x	x	x	x

For 0 0 0 and 1 1 1 the states are not defined in our sequence so next state is not defined so next state information is don't care. For each one of the other things I need to know that now. The present state is 0 the next state is 0 for A so what should be J<sub>A</sub> K<sub>A</sub>? Look at 0 0, 0 0 entry for a J-K Flip-Flop 0 don't care. So if the present state is 0 next state is 0 I need 0, 0 to 0, 0 don't care.

I need to actually write the next states. So I will write the next states here because otherwise I may make some mistakes as I did just now. The next state value is not defined for the first one, for 0 0 1 the next state is 1 0 0 so I have to see this state and this state. If the present state value of A is 0 and the next state value of A is equal to 1 what transition should happen. In 0 1 0 what is the next state? 1 1 0, 0 1 1 next state is 1 0 1, 1 0 0 next state is 0 1 1, 0 1 0, 1 0 0, 1 0 1 the next state is 0 1 0 (Refer Slide time: 15:57) and 1 1 0 next state is 0 0 1 and for this state there is no next state defined.



(Refer Slide Time: 16:55)

Present state			F-F input					
A	B	C	$J_A$	$K_A$	$J_B$	$K_B$	$J_C$	$K_C$
0	0	0	x	x	x	x	x	x
0	0	1	0					
0	1	0						
0	1	1						
1	0	0						
1	0	1						
1	1	0						
1	1	1	x	x	x	x	x	x

Next State		
A	B	C
1	0	0
1	1	0
1	0	1
0	1	1
0	1	0
0	0	1
x	x	x

Actually I should have put it here (Refer Slide Time: 16:12) next to this to make it convenient. Since I use the same table from the D flip-flop I just changed the inputs from DA DB DC to  $J_A J_B J_C K_A K_B K_C$  I am putting it here. Actually it will be nice to write the present state, the next state.

This is present state values of the variables, the next state values of the variables A B C this is really A power plus, B power plus, C power plus and then the corresponding flip-flop inputs. A state table is simply from state to state variation. If I write a table for this instead of a diagram it's called a state table. A table which contains not only the state changes but also the inputs of the flip-flop is called a transition table. This is the state graph or the state map and instead of this if I put it like this (Refer Slide Time: 17:30) where the present state 0 0 1 and the next state is 1 0 0, the present state is 1 0 0 and the next state is 0 1 1 if I write like that, that is instead of writing the map I write a table so such a table is called a state table.

A state table where the present state value, the next state value along with the flip-flop input is given such a table is called the transition table. Transition table is required to draw the Karnaugh Maps for  $J_A J_B J_C$  and  $K_A K_B K_C$  and in order to get the transition table I need the excitation table of the given flip-flop. In this case we are using JK Flip-Flop so I need JK excitation table. If you want D I also have present state, next state and D values but Q power plus is same as D so we are not writing a separate excitation table and separate transition values. Hence the next state table alone can help me to design D flip-flop steering logic. For D flip-flop steering logic I don't need extra input columns because D and Q are the same so I don't write a separate excitation table and I don't write a separate transition table but the next state table can be directly used for Karnaugh Maps but not so for J-K Flip-Flops.



So now we know that if the present state is 0 and the next state is 1 what should be  $J_A$   $K_A$ ? It should be 1 don't care, 0 to 1 again 1 don't care, 0 to 1 again 1 don't care, 1 to 0 don't care 1, 1 to 0 don't care 1, 1 to 0 don't care 1 this is for  $J_A$   $K_A$ . For  $J_B$   $K_B$  look at the b value in the present state and b value in the next state, 0 next state 0, 0 to 0 is 0 don't care, 1 to 1 is don't care 0, 1 to 0 is don't care 1, 0 to 1 1 don't care, 0 to 1 1 don't care, 1 to 0 don't care 1. You can see one pattern here, fifty percent of the rows are don't cares.

(Refer Slide Time: 21:55)

The image shows a handwritten table on a chalkboard. The left side is titled 'COMB LOGIC Truth Table' and the right side is 'STATE TABLE'. The 'COMB LOGIC Truth Table' has columns for 'Transition' (A, B, C) and 'F-F input' ( $J_A$ ,  $K_A$ ,  $J_B$ ,  $K_B$ ,  $J_C$ ,  $K_C$ ). The 'STATE TABLE' has columns for 'Next State' (A, B, C) and 'J, K' (J, K). The table contains 8 rows of data with many 'x' marks representing don't cares.

COMB LOGIC Truth Table						STATE TABLE							
Transition			F-F input			Next State							
A	B	C	$J_A$	$K_A$	$J_B$	$K_B$	$J_C$	$K_C$	A	B	C	J	K
0	0	0	x	x	x	x	x	x	x	x	x	0	0
0	0	1	1	x	0	x	x	1	1	0	0	0	0
0	1	0	1	x	x	0	0	x	1	1	0	1	1
0	1	1	1	x	x	1	1	0	1	0	1	1	1
1	0	0	x	1	1	x	1	x	0	0	1	1	1
1	0	1	x	1	1	x	1	x	x	0	1	0	0
1	1	0	x	1	x	1	1	x	0	0	1	0	0
1	1	1	x	x	x	x	x	x	x	x	x	1	1

These four are don't cares (Refer Slide Time: 20:20), these four are don't cares in addition to the don't cares for the next state not defined. Not defined don't cares are also there. In addition to them fifty percent of the entries will be don't cares. So our map will be simpler, when we have lot of don't cares our maps will be simpler the hardware will be simpler. So  $J_A$   $K_A$  maps will be very simple compared to the D flip-flop maps.

Now let us see for  $J_C$   $K_C$ , the present state is 1, next state is 0, 1 to 0 is don't care 1, 0 0 is 0 don't care, 1 to 1 is don't care 0, 0 to 1 to..... 1 to 0 is don't care 1, 0 to 0 is 0 don't care, 1 to 1 is don't care 0, 0 to 1 would be 1 don't care, 1 to 0 should be don't care 1, 0 to 1 would be 1 don't care. So you can again see that for every column at least four entries are don't cares and more entries are don't cares because some of them are not defined states so there also you have don't care.

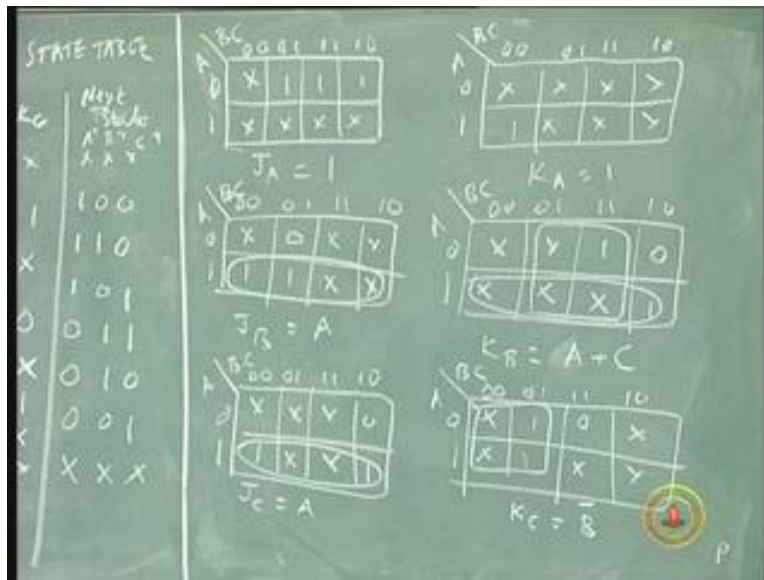
Even if all the states are defined fifty percent of them will be don't cares. So from here the next step is to draw the Karnaugh Map for  $J_A$   $K_A$   $J_B$   $K_B$   $J_C$   $K_C$  and then draw the steering logic which is a fairly straight forward affair. So we will map this  $J_A$  don't care 1 1 this is  $J_A$  map, what is the value of  $J_A$  1? All don't cares can be assumed as 1 so  $J_A$  is 1.  $K_A$  map will be all don't cares, all 1s. This is what I meant when I said in J-K Flip-Flop the steering logic will be simpler even though there are two inputs each time because at least fifty percent of them are don't cares. So  $J_B$   $K_B$  maps,  $J_B$  don't care 0 don't care don't care and 1 1 don't care don't care this is  $J_B$  and  $K_B$  will be don't care don't care 1 0 and

don't care don't care don't care 1 so what is  $J_B$ ? This is  $J_B$  is equal to  $A$  and  $K_B$  will be  $A$  plus  $C$  that is this (Refer Slide Time: 25:23).

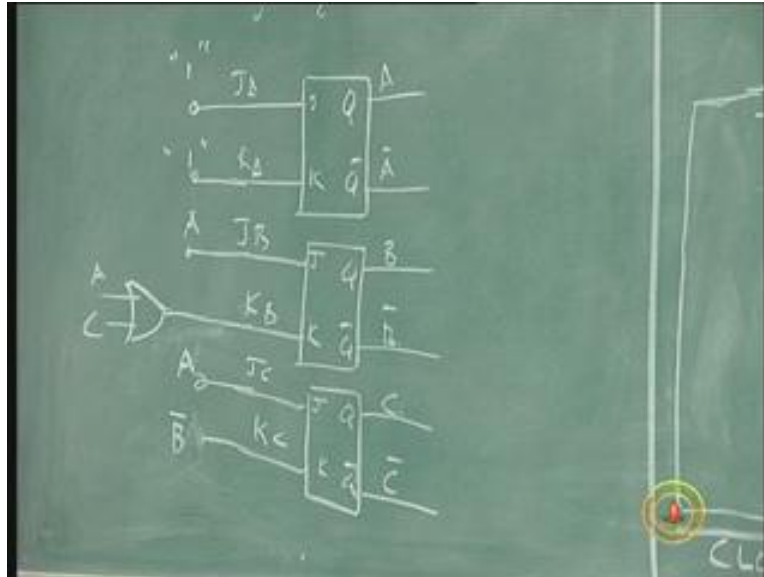
Then the last set of maps  $J_C$  and  $K_C$  will be don't care don't care don't care 0 and 1 don't care don't care 1 so this is  $J_C$  is equal to  $A$  and  $K_C$  would be don't care 1 don't care 0 don't care 1 don't care don't care and  $K_C$  would be  $B$  bar. So now  $J_A$   $K_A$   $J_B$   $K_B$   $J_C$   $K_C$  everything is known so this steering logic can be now defined. In this case it so happens that  $J_A$  is simply 1, this is 1,  $J_B$  is  $A$ ,  $K_B$  is  $A$  plus  $C$   $B$  bar, **I will rewrite this logic here** (Refer Slide Time: 27:16).

**Take** three flip-flops and call them  $A$   $B$   $C$   $Q$   $Q$  bar, this flip-flop we will call  $A$ , this flip-flop is called  $B$ , this flip-flop is called  $C$  so  $J_K$   $J_K$   $J_K$  and this is called  $J_A$  so  $J_A$  is equal to 1 so I will have to connect it to 1 and  $J_B$  is also 1.  $J_B$  is  $A$  and  $A$  is here so from here it comes,  $K_B$   $A$  bar  $A$  or  $C$ ,  $A$  is here (Refer Slide Time: 28:25)  $C$  is here so combine it. This is  $J_C$  which is  $A$ ,  $K_C$  is  $B$  bar, this is also available here, it is a simple design.

(Refer Slide Time: 26:30)



(Refer Slide Time: 29:05)



So the logic is even though we are using three flip-flops in the earlier case of D flip-flops also we used three flip-flops, number of gates used there was more than this case. Even though there are two inputs in each flip-flop the reduction is so much the hardware. We are able to use a small number of, and only one OR gate is extra because all the inverters are available in the flip-flops, there is only one extra two input OR gate along with the three J-K Flip-Flops they will suffice and it gives you the entire logic. Therefore that is in principle the design of sequence circuits or state machines if you want to call that.

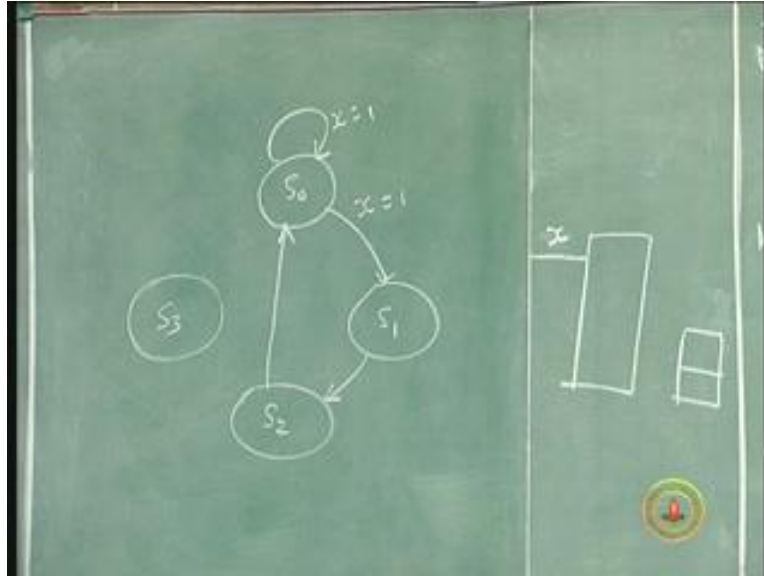
First define the various states the machine is expected to be in circuit. If you don't want to call it a machine call it a circuit or a system or a subsystem or whatever. Define clearly the various states the circuit is supposed to be in and the condition in which the transition from one state to the next will happen so we have to define a logic which will take into account the present state and the inputs so that the next state inputs will be given, the steering logic is defined such that the flip-flops are steered from one state to the next state as per the requirement which again takes into account the present states and the external inputs.

The two circuits we discussed are very simple circuits. I have taken circuits which does not have an external input. That means at every clock cycle the circuit has to change its state. At each state the circuit will remain for only one duration of the clock one period of the clock, duration of one clock period and then it will go to the next state and then it will go to the next state and so on. Of course I can complicate this by giving external inputs then the circuit will depend on the present state as the steering logic will also take into account the external inputs in addition to the present states. That is the generalized model we drew in the beginning of this module of lectures on state graph implementation.

You have external inputs along with the present state inputs which is taken into account, design the external outputs and the next state outputs. So you can always modify this

state graph with external inputs. Let us draw a simple example of a state graph which will also have external inputs. I think we drew this state graph, a similar one in the beginning.

(Refer Slide Time: 32:50)



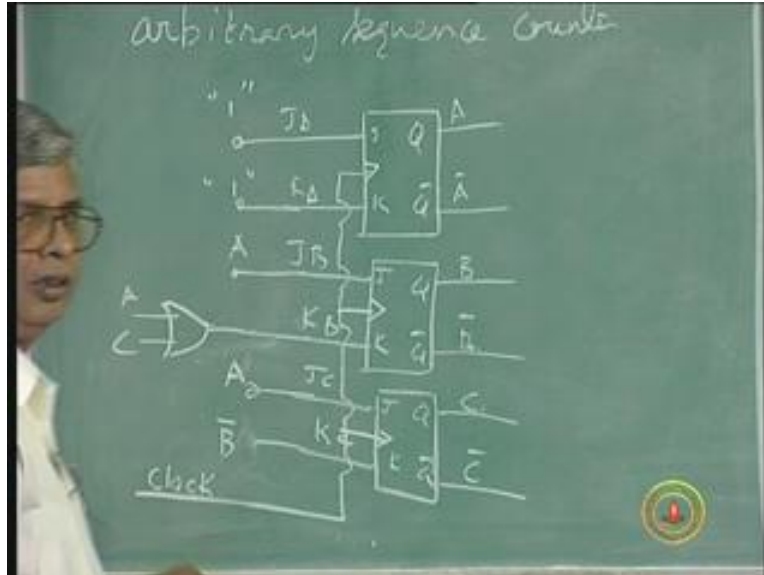
Suppose you have four states call this  $S_0$   $S_1$   $S_2$   $S_3$  the circuit will go from here to here and from here to here so something arbitrary, this is  $x$  is equal to 1  $x$  is an external input so the circuit will have  $x$  as external input and two flip-flops because there are four states, of course I have to draw the clock here, remember a clock is required. The flip-flops need clock to be steered from state to state. We will simply have D flip-flops here (refer Slide time: 33:10) these are the present state values which are fed back so that next state values will be obtained along with the clock here and  $x$  is 0 so it can stay here or it can go here so maybe  $x$  is 0 will take it here and if  $x$  is equal to 1 it will remain here and then in this state let us say  $x$  is equal to 0,  $x$  is equal to 1 and let us say  $x$  is equal to 0 and so on.

(Refer

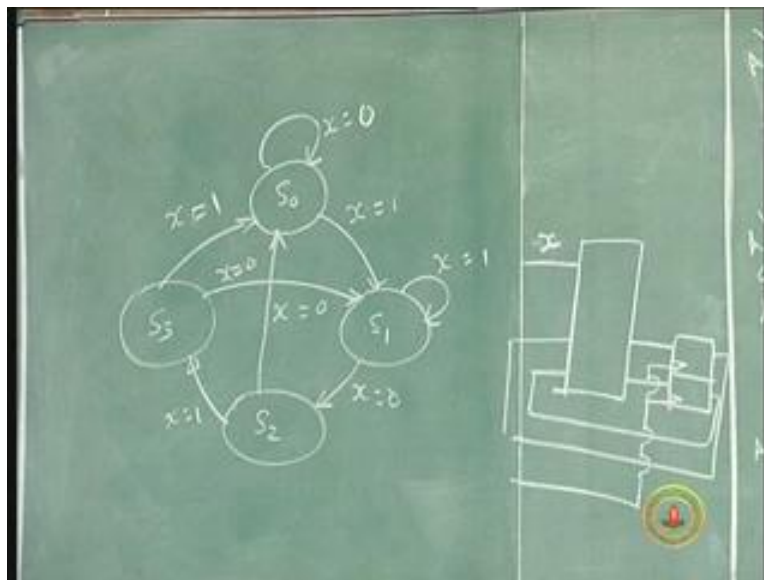
Slide

Time:

32:58)



(Refer Slide Time: 33:58)



So each state you have to make sure that..... because  $x$  is equal to 1 is an input variable it can take two values, for a given state what are the two next states corresponding to two values of  $x$  you have to say. You will have to make sure that every state has two arrows going out of it so  $x$  is equal to 0 arrow  $x$  is equal to 1 arrow, 0 arrow  $x$  is equal to 1,  $x$  is equal to 0  $x$  is equal to 1, and  $x$  is equal to 0  $x$  is equal to 1. So now in that case my state table has to be written with two inputs for present state and one extra input for the external input  $x$ . So my state table will look like this:

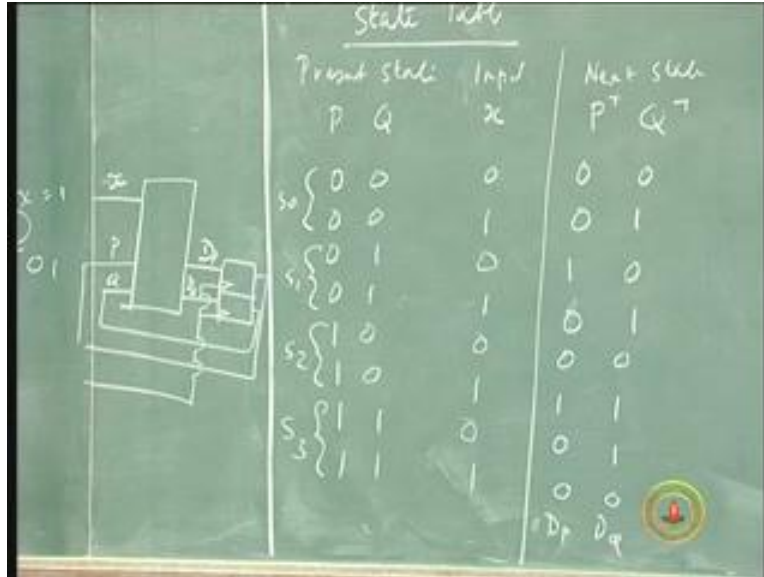
(Refer Slide Time: 35:50)

Present state		Input	Next state	
P	Q	x	P <sup>+</sup>	Q <sup>+</sup>
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	1
1	0	0	1	0
1	0	1	1	0
1	1	0	1	1
1	1	1	1	1

Let me define these states arbitrarily as 0 0 state, 0 1 state, 1 0 state, and 1 1 state and I call these state variables X and Y or P and Q. So present state variables are P and Q and input x corresponding to the next state P and Q. Let us assume D flip-flops so next state is same as D so the transition table will be same as the state table. D flip-flops are used and the state table is same as the transition table. If flip-flops other than D is used the transition table and state table will be different. So for present state P Q 0 0 x is equal to 0, 0 0 x is equal to 1 what happens? It is 0 1 0, 0 1 1, 1 0 0, 1 0 1, 1 1 0, 1 1 1. So if present state is 0 0 x is equal to 0 it stays in 0 0, if present state is 1 0 0 input is 1 it goes to next state 0 1 this is how you write the state table for this with an external input.

The next is, from 0 1 x is equal to 0 it goes to 1 0, if x is equal to 1 it remains in 0 1 this is S<sub>0</sub> state (Refer Slide Time: 36:28), this is S<sub>1</sub> state, this is S<sub>2</sub> state. In S<sub>2</sub> if input is 0 it goes to 0 0, if input is 1 it goes to 1 1. From S<sub>3</sub> if x is equal to 0 it goes to 0 1 and if input is 1 it goes to 0 0.

(Refer Slide Time: 37:25)



The image shows a handwritten diagram of a D flip-flop circuit on the left and its state table on the right. The circuit diagram includes inputs P, Q, and x, and outputs P<sup>+</sup> and Q<sup>+</sup>. The state table is titled 'State table' and lists the next state (P<sup>+</sup>, Q<sup>+</sup>) for various combinations of present state (P, Q) and input (x).

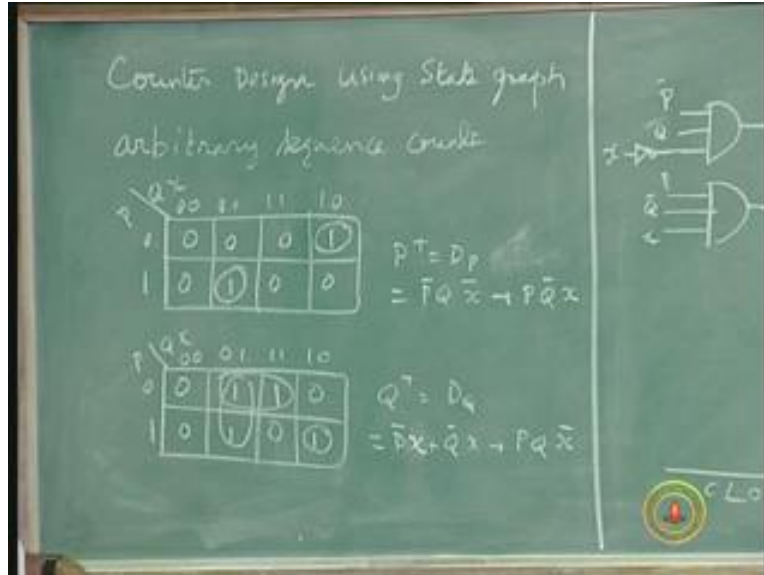
State table			Next state	
Present state		Input	P <sup>+</sup>	Q <sup>+</sup>
P	Q	x		
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
0	1	1	0	1
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	0	0

So this is the transition table as well as the state table since we are using D flip-flops in this model, this is D (Refer Slide Time: 37:06) this is PQ so I call this DP for P flip-flop and D of Q flip-flop DQ. So this is same as the DP map, this is also same as DP, this is also same as DQ. So what will be logic required? The logic required will be that we have to draw the Karnaugh Maps using these three inputs P Q and x for P power plus which is same as DP and Q power plus which is same as DQ and then get the steering logic so that circuit will be completed, **we will do that here** (Refer Slide Time: 37:57)

P power plus is equal to DP map is present state is p, next state is q and input is x. Use those values it becomes 0 0 0 1 and 0 1 0 0, this is P bar Q x bar or P Q bar x.



(Refer Slide Time: 42:24)

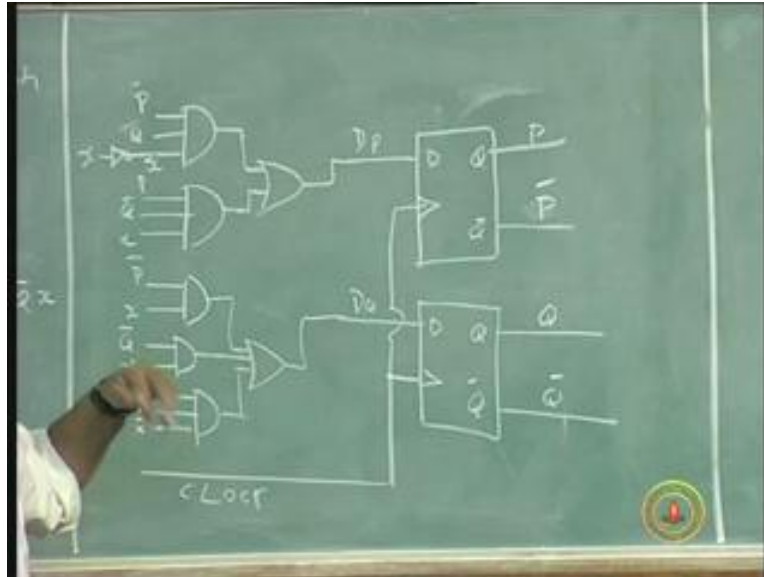


Then for the second map for Q power plus which is same as D Q the values would be 0 1 1 0 and 0 1 0 1 so this will have three terms one term is P bar x, this term is Q bar x and this term is PQ x bar.

The circuit corresponding to that is two flip-flops DP both are clocked, DP is P bar x Q bar and P bar is available Q is available x bar is of course is not available so x has to be inverted, make this as an external input P Q bar x, here this is Q (Refer Slide Time: 41:51 - 42:25) corrections made in slide.

The second expression is P bar x, P is available, x is available, P bar is available, Q bar x, PQ x bar, x bar is available because of this inverter. With D flip-flop even though we have only one input the logic is slightly more complex because of only one input. In J-K Flip-Flop because of lots of don't cares you get reduced logic. So this is a generalized system. I didn't have anything in mind when I wrote this state graph, this is a state graph (Refer Slide Time: 43:20), this is the transition table as well as the state table (Refer Slide Time: 43:26) this is simplification of the transition table, this is the logic diagram using flip-flops and gates.

(Refer Slide time: 43:33)



Of course I didn't have anything in mind, I just assumed a circuit with one input and four states, four states require two flip-flops use two flip-flops and those flip-flop outputs have to be fed back as input information so two state variables P and Q and one external input x so P Q x are the three inputs to the steering logic. So this could be anything, it could be a traffic light controller for example, there may be four states in which x may be an external input based on that the circuit will change state as an example, I didn't have anything in my mind I just wrote it arbitrarily. But this tells you how to formulate the problem to start with. First given a problem specification you have to come up with a state graph.

First of all you have to define external inputs and external outputs, what are the external inputs and what are the external outputs, then you draw a state graph. At that time you may not be clear about how many states are required. I may not be able to start with a number of states but as I develop this state graph I may know how many states are required to complete the problem. The circuit has to exist in many states and from one state to the next state it goes based on some requirements or some specifications. So draw the state graph, include the external inputs and external outputs, decide how many flip-flops are required based on the number of states you have, then draw the state table and depending on the type of flip-flops you go to the transition table and once you have the transition table you simplify the transition table using the Karnaugh Maps in order to get to the logic steering logic combinational part and then you draw the circuit diagram and of course you have to wire it and test it.

What I have not done in this case is of course one more modification is required but I will not talk about any outputs, I am assuming outputs are the state variables themselves so the value of P and Q at any given time is the output I have taken like in the earlier example also. In the earlier example I have taken six states A B C, no external inputs could be a counter as I said even though the counter doesn't count in a natural sequence

these are six states in a counter at any given time you look at the values of A B C you know the output. It's a binary counter, sequential counter those are the values required.

In this case for example there may be an external output we don't know. What is the external input? Say external input is x then I may have an external output. So I have to define an output also. How do I define an output in this graph? It is possible to define an output. Output in this state can be something 0, output in this state can be 1 so I may have an output called z and that z can be 0 in certain states and 1 in certain states so I can modify the circuit to have outputs, I did not make it in the beginning to keep the complexity level low so that gradually you can increase. So in other words we now have a tool of, we have studied combinational logic different implementations, we studied the basic sequence of building blocks, flip-flops, counters, shift registers and so forth.

Now we know a mechanism of representing a given system or a subsystem by means of a state graph. Of course we are assuming it's a sequential circuit. Once you have a state graph you know how to proceed and implement it in hardware using gates and flip-flops. Now the next step is to find is there anyway in which the logic becomes more and more complex. If the number of states becomes large and number of gates also becomes very large, number of states decides the number of inputs and outputs to these Karnaugh Maps so **there are** many terms. So can we make a simplification of the gate level into higher level of logic? We can think of combinational building blocks which are more compact to work with. I would like to have fewer parts and I have fewer parts, fewer components I will save space in my board, I will save power dissipation and I will save cost also.

We will talk about some of those things, how to reduce the combinational logic part from an extensive gate oriented, because these examples are very simple examples. When the examples become more complex, when the number of states becomes large, when the number of transition becomes large our steering logic becomes very large and extensive so is there anyway I can simplify the steering logic by using fewer components, fewer parts that's what we will see next.