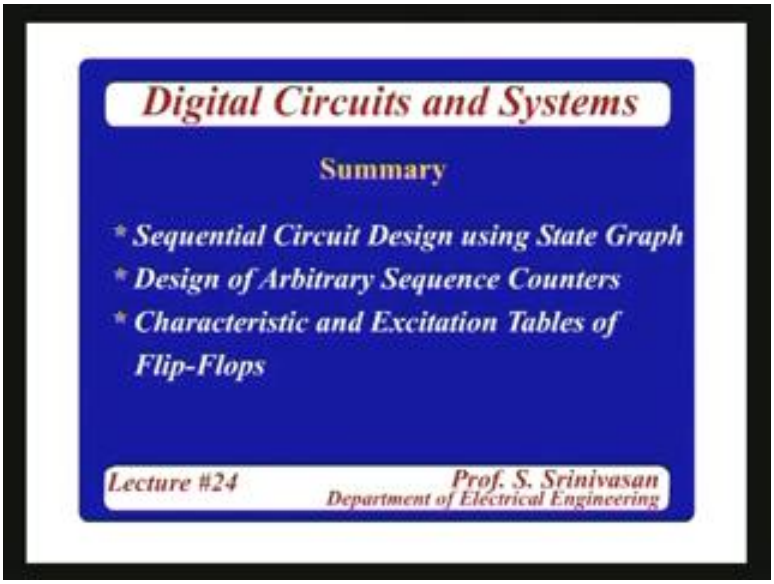
Digital Circuits and Systems
Prof. S. Srinivasan
Professor
Department of Electrical Engineering
Indian Institute of Technology, Madras
Lecture # 24
Design of Synchronous Sequential Circuits

(Refer Slide Time: 2:14)



I introduced the concept of state machine in the last lecture. Any digital system can be modeled as a state machine. Because unless it is a pure combinational logic the circuit of the system resides in several states and the behavior of the system is defined by a state graph or a state diagram which gives all the possible states in which the circuit can reside and the condition for transition from one state to another state. A state machine need not be very complex, it is a very simple counter. A counter is a state machine. So we want to really know how to model the given problem is a state machine then of course to see how to implement it how to design it and build it.

So we will see several examples as I said. In order to start the discussion we will start with a familiar circuit the counter. We will represent the counter as a state machine. So far we have not given that type of interpretation. To us a counter is a series of flip-flops connected in some particular way so that it can count up or count down or count from one particular count to another terminal count and all that. The same thing can be now got using a state machine model. So I will call this a counter design using state graph.

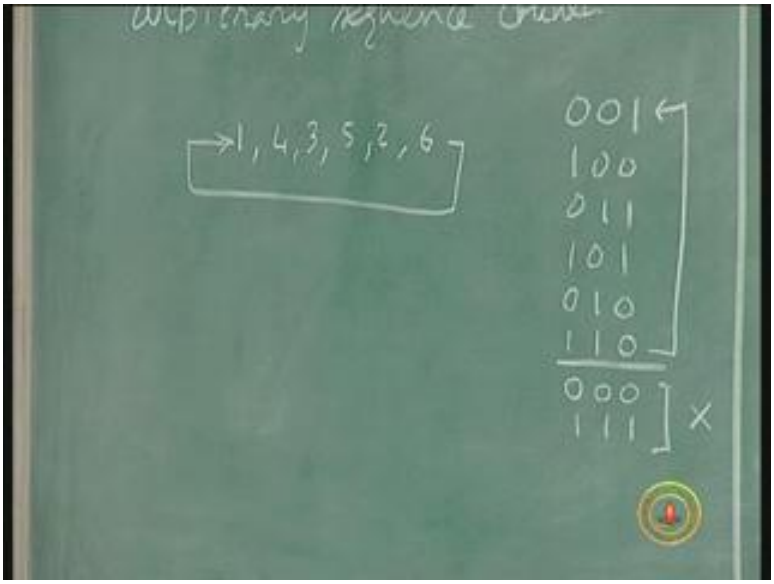
What I can do in this counter is to make a change deviation for what we have been seeing so far. I can have arbitrary sequence of count. A counter normally we assume that it will count up or count down stepwise. Up counter will start with 0 1 2 3 4 5 etc and down

counter will start with a particular number and go down. For example you can start with 7 6 5 4 3 2 1 0 and back to 7. What I mean by arbitrary sequence counter.

Let us have states which are not sequential in nature. For example, I can have a count of 1 followed by 4 followed by 3 followed by 5 2 and 6 and back to 1. It is an arbitrary sequence that's what I mean by arbitrary sequence. That means if you represent it in binary this is going to be 0 0 1 then 1 0 0 then 0 1 1, 1 0 1, 0 1 0, and 1 1 0 and back to this. So we are not having states 0 and 7. Thus it doesn't have to be sequential; all you have to know is the sequence in which the states have to appear.

We saw in last lecture that a state is defined as the condition of a circuit at a particular clock period. During a particular clock period the circuit remains in a particular state with flip-flops having different values and from there it goes to the next state in the next clock period and where it goes depends on the present state of the flip-flops as well as the external inputs. So, a circuit can remain in many states which are nothing but conditions of flip-flops because flip-flops are the ones which store the state information. The flip-flops are the ones which stores the condition of the circuit.

(Refer Slide Time: 6:26)



Therefore from a given set of conditions of a flip-flop to another set of conditions of flip-flop we have to go the transition will depend on the present state and the inputs to the flip-flops or the external inputs then it decides the next states. So we think of that as a state machine and this also is a state machine. Here this is only the binary I have written so how many flip-flops are required for this? Each state has three bits there are three variables which change from 0 to 1 all the time so let me call these flip-flops A B C or 3 this A B C are called the state variables.

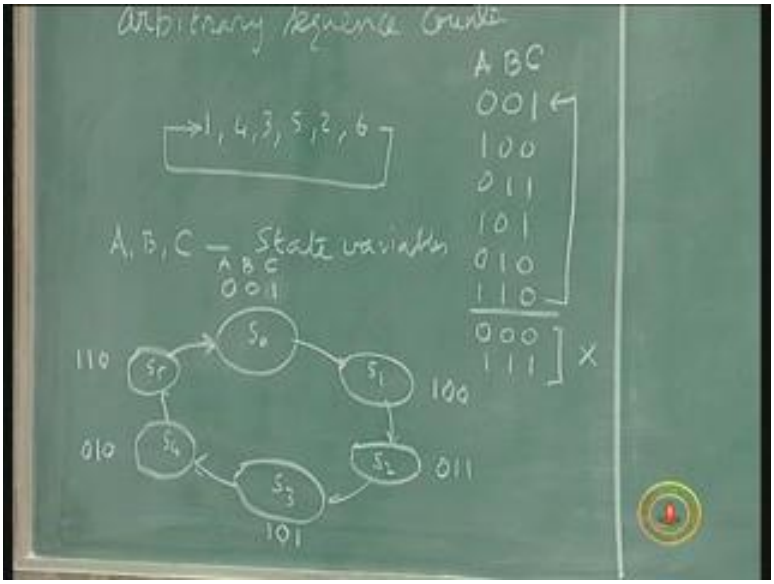
A is the output of a flip-flop, B is the output of another flip-flop and C is the output of the third flip-flop and these three flip-flops remain as the states 0 0 1 and then goes to 0 state

1 0 0 then they go to state 0 1 1 etc they go to state 1 1 0 and finally back to 0 0 1. So I can represent the state graph easily and it is a trivial case of no external input and only the clocks steers this. Every clock period we are assuming D flip-flops will go from one state to the next state. So I gave you different models of the state diagram or the system in which there can be external inputs, there need not be external inputs in which only the clock triggers. The clock transition triggers this state transition and the state transition will depend on the present state and the external inputs and if the external inputs are not there the state transition will depend on the previous states that's all nothing else.

Similarly we can have an output which is different from the state in which the circuit resides or the outputs or the state information itself can be used as an output as in the case of a counter. Counter you know what counter it is, that's what we want we count the output of counter what is the particular count what is the current count that is the output of a count so no external output. The counter is a case of a state machine in which there are no external inputs, no external outputs.

On the other hand I can have a count also with an external input like up or down. Suppose I want the count to go up what should it be like and if I my input is present it is up and if input is absent this is down. This also can be used as the external input. If you want in this case a simple counter is in which no external inputs are there so the clock is a triggering mechanism the counter flip-flops change from state to state to state based on the transition at the time of the transition of the clock.

(Refer Slide Time: 12:15)

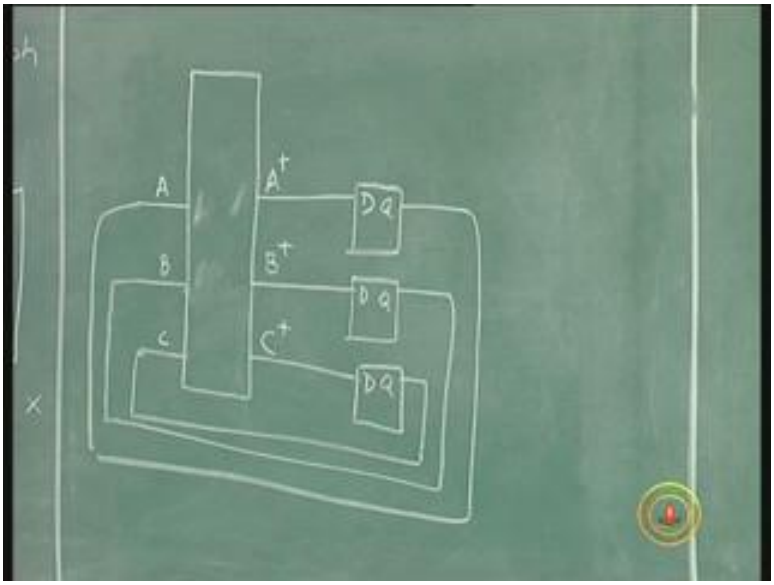


So we have a state machine model state graph, there will be six states so I will call this state S_0 S_1 S_2 S_3 S_4 S_5 they keep changing without any external input. That means every clock period or every clock transition there can be a state transition. For the first clock period it will be in S_0, in the next clock period the circuit will be in S_1, the next clock period the circuit will be in S_2 and so forth and each of the S are represented by the state

variables so I put the state variables outside the circle like 100, 011, 101, 010, 110, 001 these are the values of the variables A B and C the state variables.

So how this circuit going to look like? The circuit is going to look like this, the model of the circuit is going to look like this. We have three flip-flops called A B and C. The current state of the flip-flops A B and C are fed into the inputs of the circuit and next state is defined. For a given state A B C the next state is defined. So the present state will decide what the next state should be but in order to distinguish between these two variables I will just put A power plus sign, it does not mean anything, it is the same variable after the, the present state of these variables after the next state of these variables, the A plus B plus C plus represent the next value of A B C. So plus is given as sort of a notation. And that will be stored in the flip-flops let us use simple D flip-flops and outputs Q that will be fed back as this (Refer Slide Time: 13:55) and there is no external inputs no external outputs for this circuit.

(Refer Slide Time: 14:05)



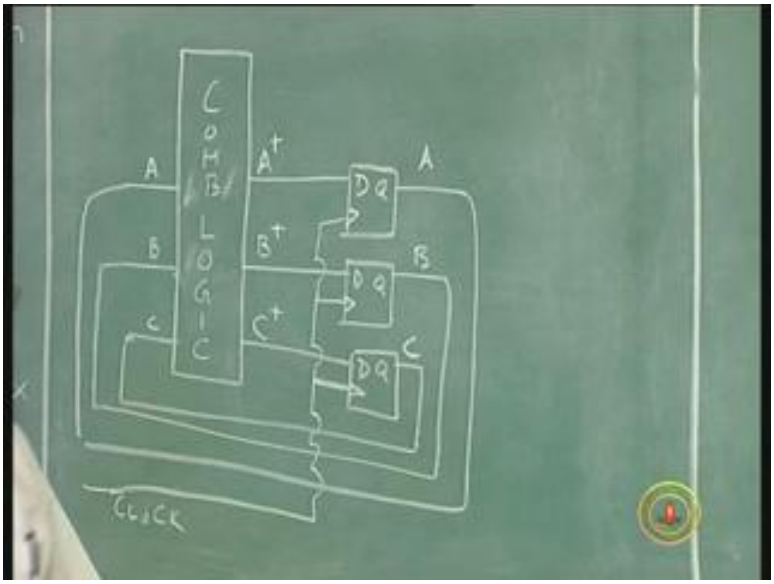
On these flip-flops the present state values are presented. Once the values of the present state variables are presented to the circuit it will come up with the next state value. The next state values will be available here which will become the next state not immediately but only after the next clock transition. At the next clock transition these values become the present state. Now these are the input values of the present state and the output values of the next state values.

The next state values will be taken in by these flip-flops stored in these flip-flops and presented back to this so that the states after that can be decided. So this is the flip-flops which are clocked and this is a pure combinational logic as I said. This combinational logic is designed such that once this input is given this becomes the output and if this input is given (Refer Slide Time: 15:50) this should be the output, if this input is given

this should be the output, this input is given this should be the output, this input is given this should be the output, and if this input is given this should be the output.

If you decide the combinational logic that way then as soon as the present state values are presented here the next state values will appear, the next state values will stay there when the clock transition occurs, the next state values are stored in the flip-flops and become the present state at that time, the next state becomes present state at the end of the next clock transition and that present state values will be presented here that will be used by the combinational logic to decide the value for the state after that and that will become the next state for that and so forth. So this arbitrary counter design is nothing but a simple combinational logic design where if I give this value this should be the output, if I give this value this should be the output and so forth. Therefore the combinational logic is a logic in which there are no flip-flops or anything, as I said the inputs decide the outputs and inputs are this A B C and outputs are A power plus B power plus C power plus even though the variable is called A I will call them A power plus B power plus C power plus to distinguish between these sets of variables otherwise we will get confused totally. So this Aplus Bplus Cplus becomes A B C at the next clock transition.

(Refer Slide Time: 17:19)



So what is the description of combinational logic?

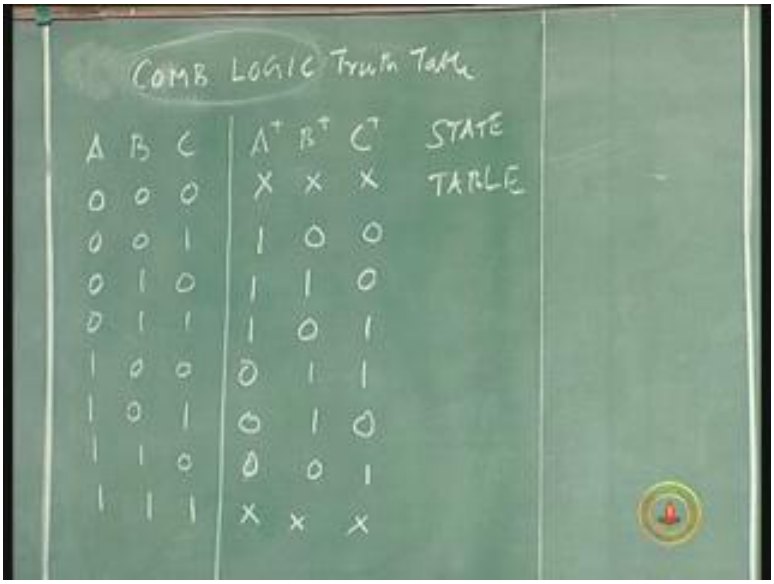
We will take A B C as inputs and output A power plus B power plus C power plus and I can always design the circuit using the map method. You know map method. but the only thing is we will not write in this order because if you want to use map method to simplify this combinational logic I would like to have it in a natural binary sequence so I will write all possible values of A B C and then write the next values corresponding to that. instead of saying 0 0 1 to start with and then 1 0 0 then it will get confused, when you try to draw the map again I will have to sort of fumble around, in order to avoid that I will put on in this sequence and write the next values from this given state graph.

Some of the possible present state values may not even exist for example 0 0 1 doesn't, 0 0 0 doesn't exist. If 0 0 0 occurs what is the next state? 0 0 0 is not one of the required states so 0 0 0 will not occur in this counter so the next state will be "don't care".

Similarly 1 1 1 occurs again it's not defined then it should be "don't care". For all other values of these present states the next states are defined. So, if the present state is 0 0 1 what is the next state? It is 1 0 0 and if present state is 0 1 0 the next state is 1 1 0, present state is 0 1 1 the next state is 1 0 1, present state is 1 0 0 the next state is 0 1 1, present state is 1 0 1 the next state is 0 1 0, present state is 1 1 0 the next state is 0 0 1. This is a state graph and this is a state table that's all (Refer Slide time: 20:20).

Have I written anything different from here? Whatever information I put in this form of a graph figure diagram has been converted into the form of a table so this is called a state table. So instead of combinational logic truth table this is a truth table of a combinational logic. Also, known as the state table here in this case because this table tells you the present states and corresponding next states, that's all.

(Refer Slide Time: 20:40)



This is what I said graph gives you. if there is an external input I will have to put an external input. If the present state is there and x is equal to 0 what will happen, if present state is there and x is equal to 1 what will happen. So if you have inputs also in addition to the present state variables here if there are external inputs that also can be accommodated in this by repeating one more column for each of the inputs.

If there has been an x then I will put an x or an extra column in this truth table then I will say the present state is this and x is equal to 0 the next state will be this. If present state is this and x is this then this will be the next state. And similarly there is an output in addition to the state outputs there is an external output I can also incorporate here on the

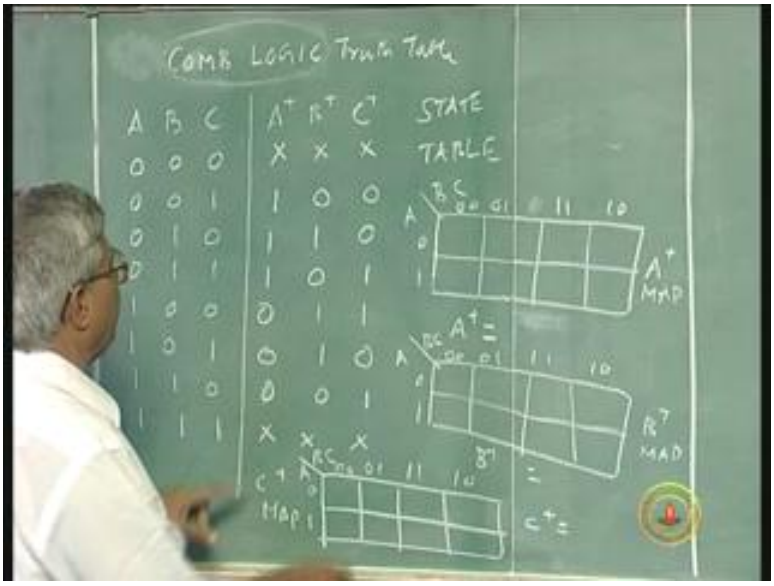
outside the output side. Suppose Z was an output in this state this should give an output of this, in this state this should give an output of this so that also I can accommodate.

A state table is a generalized truth table. The truth table has input output relationship and it involves the present state and the next state and if you have external inputs I can also include them and if you have external outputs I can also include them. So present state information and the input information is the left hand side of this table and the next state information and the output information they form the right hand side of truth table so such a generalized table is called a state table a state map a state map or a state graph is translated into a state table. And from here to combinational logic is easy we know that.

So I will have to draw for each of these A power plus B power plus C power plus a Karnaugh Map in which A B C will be inputs, I can simplify the map and find out what is the combinational logic it is. So I can draw the map here itself.

I give A B C this is A power plus map (Refer Slide Time: 23:27) then I will have the B power plus map so A power plus can be read from this map B power plus and you have C power plus map and you read C power plus from this map. So I can read A power plus I can read B power plus I can read C power plus and then I have what is it required inside this box then my counter design is completed. This is a generalized approach for a design. I can use this binary counter also sequential binary. Instead of this can I now do that? It does not matter, it is an arbitrary sequence I can have a natural sequence also, doesn't matter. So this is a generalized design concept I am giving you.

(Refer Slide Time: 25:00)



What is this now? The first column is this A power plus map so this is don't care 1 1 1, 0 0 0 don't care which is very easy, what is this A power plus is equal to A bar. And B power plus map 0 0 1, 1 1 don't care 0 then it is going to be this and this (Refer Slide Time: 25:48). You will have to include as many "don't cares" as possible to reduce the

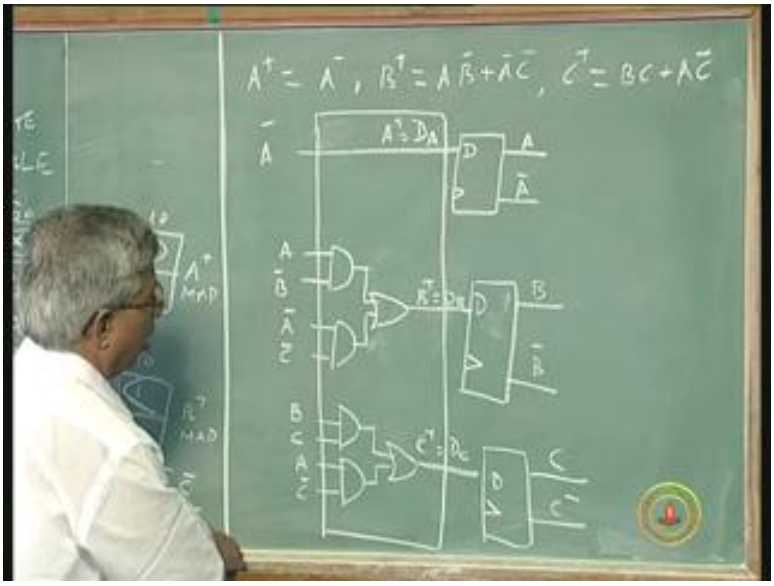
map. I find twist answers in exam questions. Some of you have not used the "don't cares" to the maximum possible extent. Use "don't cares" to the maximum possible extent to reduce the hardware complexity. So, for example this one and this "don't care" can be combined instead of reading this as a separate prime implicant. So what is this? This is A B bar A bar C bar, C power plus map is here 0 don't care don't care 1 1 1 don't care 0 0 1 1 0 1 don't care. Again these are the two prime implicants. That is A C bar and this is B C.

(Refer Slide Time: 27:35)

So I have my A power plus as A bar, B power plus as A B bar plus A bar C bar, C power plus as B C power plus A C bar and where are these A B Cs? These A B Cs are the same variables here in order any external inputs. These are the state variables which are given as input to the next state determination. Present state variables have to be given as inputs so that next state can be determined by the combinational logic. So the combinational logic will be A bar. The advantage is A bar B bar C bar are available because these are flip-flops so you have A bar also. So there is no need for invertors because flip-flops have Q and Q bar outputs. So that way I can directly connect my flip-flop D DA power plus or DA is the input of flip-flop A power plus which is nothing but this A bar.

And the second thing will be A B bar A bar C bar. Both are available, A A bar, B B bar and C C bar are available so these four variables are available, combine them in OR gate that will become my B power plus which is given to the second flip-flop B, A power plus or DB. Likewise I have BC AC bar C power plus is equal to DC same as DC which is the D of this flip-flop and this will be C C bar.

(Refer Slide Time: 31:50)



So we give A B C A bar B bar C bar as input to the combination logic so this is the combinational logic block we are discussing here and these are the three flip-flops we are discussing here, this is the clock (Refer Slide Time: 31:55) so this is my combinational logic block which is shown here. the block shown here (refer Slide Time: 32:30) is same as this so it contains say four AND gates and two OR gates all three inputs and then I get A A bar B B bar C C bar as inputs to this. Of course I have not shown that here but only shown A B C, here also you get A bar B bar C bar inside this.

So, A A bar B B bar C C bar which are available here are folded and taken as inputs and then you generate DA DB DC which will drive these D flip-flops to the next state at the clock transition. This combinational logic is also called the steering logic. Why is it called the steering logic is obvious. This logic steers this flip-flop from state to state. So, a counter is a generalized example of a state machine it can have a computer and have a washing machine, I can have a traffic light controller, I can have an elevator controller, I can have a liquid level indicator and whatever. What are the states and what the inputs are and what sequence and what inputs, what sequence the states should go to and with inputs. If you want this state table convert this into a state graph, convert into a state table use combinational logic that is familiar to us and drive those flip-flops steer those flip-flops using the combinational logic and go from state to state to state.

If the counter is a natural sequence instead of an arbitrary sequence I can do the same thing but we don't have to do that because we know that the clock can be fed in, it is a reduced solution we don't have to go through this but you can also get the same solution using this. Suppose I need to use for some reason, are there any questions as of now to this concept of state machine design?

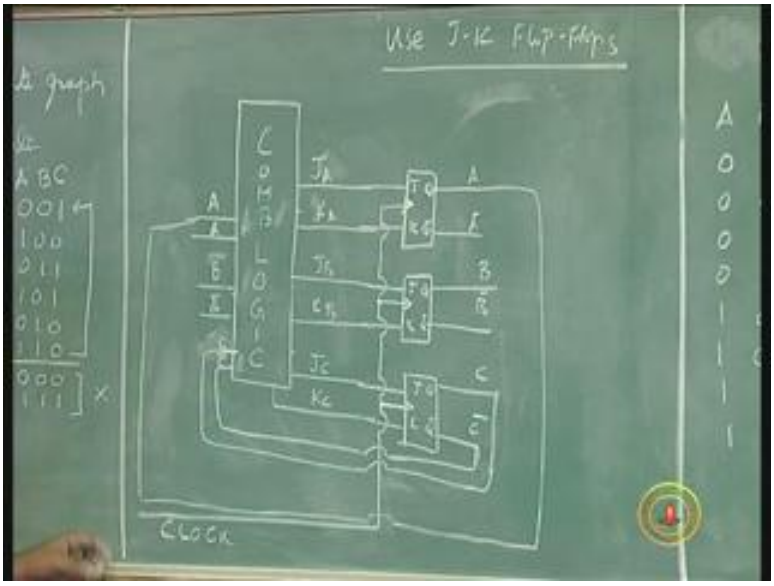
The whole thing is falling in place, the combinational logic, sequential logic, reading them independently each as a chapter to be forgotten when you go to the next chapter in the book so the whole thing is one grand thing, all used in a total of system design.

Any questions: …….(Refer Slide Time: 35:30) yes it can have, but that means for that count it has to remain in the same state. From here you can have one more count (Refer Slide Time: 35:56) 0 0 0 you can rather use it as an external input. Of course you can always do that let us say 1 0 0 you want count twice you can put 1 0 0 to 1 0 0 but then when you draw the Karnaugh Map you don't know which 1 0 0 you are talking about so instead of doing that you can have an external input defined when x is equal to 0 it remains there so you can put x is equal to 0 and for all of those x is equal to 1. So from S_0 to S_1 it will remain, both 0 and 1 it will take here, from here it will take here only if x is equal to 1 and if x is equal to 0 it will remain here so external input can be used more elegantly for that design.

Occasionally for some reason we may want to use flip-flops other than D flip-flops because we know several other flip-flops. We know JK flip-flops, we know D flip-flops, we know SR flip-flops, we know T flip-flops and so on. Suppose I say I want to use JK flip-flops instead of D same problem. The D flip-flop has an advantage Q plus is same as D that is the next state value of the output is same as the input give the input and clock it then it will become the next state value. Thus I could directly use this as D input A power plus next state information is same as the D input.

In JK flip-flop the next state depends on two inputs J and K so I may have to modify this circuit. if I want to use JK flip-flop instead of D flip-flop here I can have A and A bar, Q and Q bar J and K, J and K, J and k Q Q bar Q Q bar B B bar C C bar and all of them clocked, this is the clock source. Now the combinational logic should determine if I am in a present state A B C given by any of these values. The next state I have to determine based on this table so what input is to J and K are here in D flip-flop case, what input to D will make it the next state.

(Refer Slide Time: 42:00)



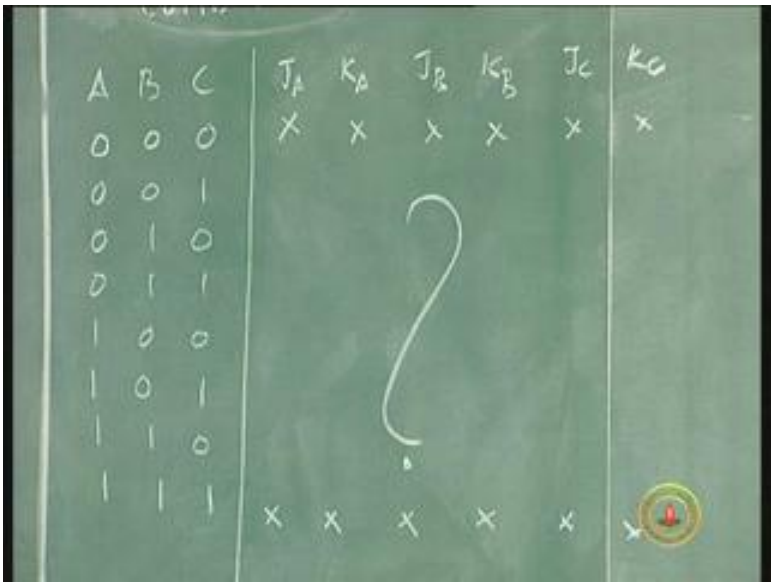
So for example we talked about A flip-flop, for the A flip-flop the present state value is 0 (Refer Slide Time: 39:45) and next state value is 1 so I need to make D is equal to 1 next state becomes. The next state of D flip-flop has to be 0 so I have to make d is equal to 0. So every clock period I have to change the value of D as per this table because next state value is same as the D input in D flip-flop and not so in a JK flip-flop. In JK flip-flop the output is decided by j and k values. If the present state is 0 and the next state has to be 1 we want J is equal to 1 K is equal to 0, if the next state has to be 0 I have to make J is equal to 0 K is equal to 1.

Since I need this type of things I need j input k input, j input k input in each of them so I need J_A J_B, J_A J_B J_C, K_A K_B K_C so the next state value will be J_A K_A, J_B K_B, J_C K_C. Of course I have to give A B C as inputs. Not only we need A B C but we have seen that A B C as well as A bar B bar C bar are also required. So, if you want to combine the drawing without making it too cluttered I can draw C C bar, C C power plus, C bar value, C value, I don't to show this so A A bar B B bar so I can draw it I am not showing the in between values assume that all of them are drawn. avoid overcrowding of this drawing, you can imagine B bar is connected here, B is connected here, A bar is connected here so normally we show only A B C even though A bar B bar C bar are also use in this generalized model of the system design we give only A B C as inputs even though you know that A bar B bar C bar will also be required.

Now the question is then if in the present state A is equal to 0 0 1 and the next state is 1 0 0 what should be the value of J and what should be the value of K. If the present state value is 0, the next state value of A has to be 1 so I need to give J is equal to 1 K is equal to 0 for this so this has to be expanded this table as to be expanded to include, the A power plus B power plus C power plus are not directly the inputs now. A power plus B power plus C power plus were the inputs in the case of D flip-flops because Q plus is same as D. Whatever is the present value of D that becomes the next state value of the

flip-flop output in the case of D flip-flop alone. In the case of JK flip-flop the next state value will be determined by J and K so what should be the value of J_A and K_A what should be the value of J_B and K_B and what should be the value of J_C and K_C? In order to the make change from 0 0 0 to 0 0 1 from 0 0 1 to 0 1 0, 0 1 0 to 0 1 1 and so forth of course 0 0 0 doesn't occur so they are "don't cares".

(Refer Slide Time: 44:20)



So I have to fill these values properly in order to go from state to state, steer the JK flip-flop from a set of values A B C, a set of values A power plus B power plus C power plus the next state I need to have a proper combinational logic. So the combinational logic will be having six outputs J_A K_A, J_B K_B, J_C K_C for present state inputs A B C and A bar B bar C bar so the logic will be going to be different and that logic will be determined by the truth table or the state table in which we will have to say what input of JA and what and KA will make a go from 0 to 0 from 0 to 0 0 to 0 and what input of B will make it going from this to this etc. So far we know only the behavior of the flip-flops it's called analysis or the characteristic table we call it.

So far we know the behavior of JK flip-flop for example take j and k and we know j is 0 k is 0 and q is memory we will call it 0 1 0 1 0 1 1 1 Q bar is it not. So you want to call this $Q_{n \text{ plus } 1}$ this QN or you want to call it Q plus this is Q bar plus Q plus memory can be written as Q if the next state value is same the present state value is Q this is the characteristic table, the behavioral table but you want a reverse table. If the present value is 1 and the next state value has to be 1 what should be the value of J and K.

So I need four transitions. The present state can be 0 or next state can be 0, the present state can be 0 and the next state can be 1, present state can be 1 and the next state can be 0, present state can be 1 and next state can be 1 so there are four possible transitions and for each transition what should be the value of J and what should be the value of K. Such a table is a complimentary table I won't call it universal table, I have to derive the table

from the characteristic table and such a table is called an excitation table, this is the characteristic table (Refer Slide time: 47:34) of JK flip-flop.

In order to make it more explicit what I am going to do is instead of making Q and Q bar etc I am going to say if the present state is 0 0 Q is 0 and next state is also 0.

(Refer Slide Time: 48:26)

This is an expansion of the characteristic table. Earlier I put memory, toggle and things like that has been removed and put explicitly. There are only two values possible for everything. Q can only take 0 or 1 so the present value of q is 0 next value of Q is 1 in the toggle mode, the present state value of Q is equal to 1 and the next state value is 0 for toggle mode, for memory the present state is 0 next state is also 0, present state is 1 the next state is also 1 so that is an expanded characteristic table. Instead of putting memory state and toggle state I have expanded it. If the inputs are 0 0 and the previous output was 0 next output is also 0, if input is 0 0 the previous output was 1 the next output was also 1, here this is 1 the memory (Refer Slide Time: 49:10).

Present state is 0 0 it's a memory. Now, if the present state is 0 1 the next state is 0 irrespective of what the Q was, if the inputs are 1 0 the next state is 1 irrespective of what the previous output was and J and K are 1 1 it is a toggle mode the circuit is in state 0 to go to state 1. From this table I have to derive what is meant by excitation table in which I will have to say if the value of Q is 0 and it as to go to 0 what should be J and what should be K? If it is 0 and it has to go to 1 what should be J and what should be K, if it is 1 it has to go to 0 what should be J and K 1 and 1. So what are the values of J and K for each possible transition of Q from, if Q can be 0 it has to remain 0, Q is 0 it has to become 1 if Q is 1 it has to go to 0 and if it is 1 it has to remain 1.

(Refer Slide Time: 50:45)

For each case what should be the value of J and K such a table is a sort of an inverse table of the transition. This can be derived from this. The excitation table you can look at this and do this and that table is required for me to fill this because here I have to say if the present state is 0 0 1 the next state has to be 1 0 0. That means flip-flop A has to go from 0 to 1 what should be J_A and K_A, if the flip-flop B is 0 the next state should also be 0 so what should be the value of J_B and K_B for the flip-flop to remain 0? Flip-flop C is 1 and it has to go to 0 for the next state what should be the value of J_C and K_C for this to become 1 to 0 transitions.

So first I will make the excitation table and then use that excitation table to fill those values of J_A K_A, J_B K_B, J_C K_C for each of the transitions given in this diagram then we have the combination logic for J_A J_B J_C, K_A K_B K_C by Karnaugh Maps and then make this diagram for the steering logic. We will proceed with this in the next lecture.