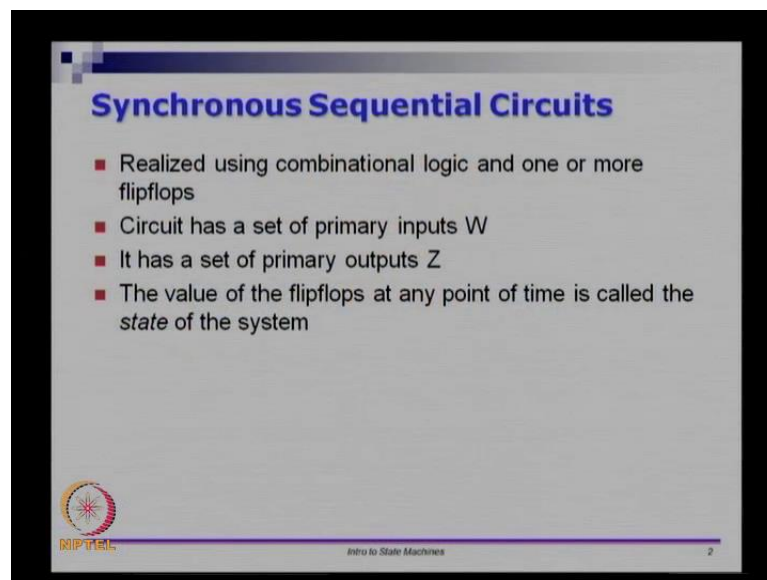


**Digital Circuits and Systems**  
**Prof. Shankar Balachandran**  
**Department of Electrical Engineering**  
**Indian Institute of Technology, Bombay**  
**And**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Madras**

**Module - 27**  
**Introduction to State Machines**

We are at module 27. In this module, I am going to talk about sequential circuits, and I will also introduce the notion of what is called a State Machine. So, we are going to look at synchronous sequential circuits.

(Refer Slide Time: 00:32)



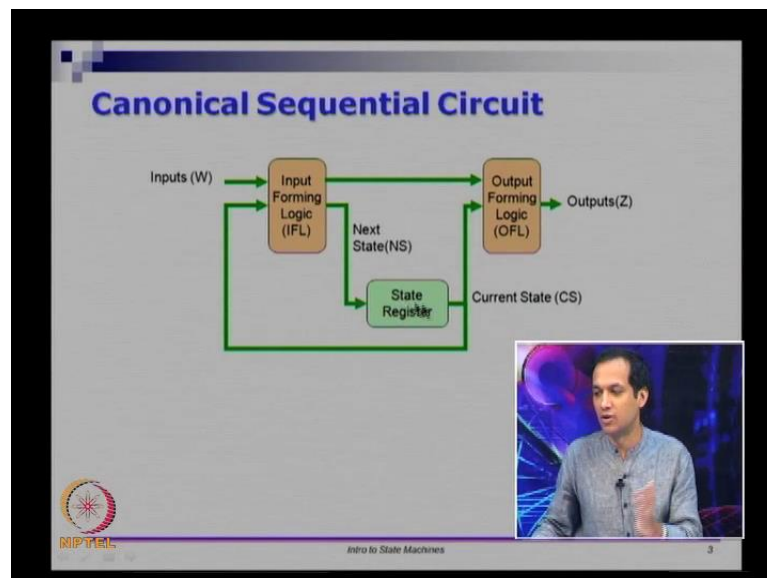
A synchronous sequential circuit is one in which, you have flip flops and you have a combination logics. So, you have a combination of both of these and the combinational logic has it is set of inputs, and you have a set of flip flops. So, as a circuit what you have is, you have a set of what are called primary inputs. So, the primary inputs are... So, if you think of a chip as a black box, those are the inputs that go into the chip, so we call them the primary inputs and there is a set of outputs that is coming out of the chip, we will call them the primary outputs.

So, any module or block or chip that we design has inputs and outputs and we have a set of flip flops and we are going to call something a state. So, a state of a system or a module or a block is defined as, a set of values that the flip flops have at a particular

point of time. So, of course, when we have many flip flops, we also have to name the flip flops. So, we will assume that there is some implicit naming of the flip flops or numbering of the flip flops.

And once we have that each of the flip flops can take 0 or 1 as a value. So, if I go and inspect the circuit at any point of time, I go and see what the values for various flip flops are and I can report that, this is at a state which is given by the vector of the values that the flip flops take. So, clearly if I have  $n$  flip flops then I will have  $2^n$  possible states that the flip flops can be in. So, I can say that the system can be in  $2^n$  states there.

(Refer Slide Time: 02:12)



So, let us look at a basic sequential circuit, so I am going to call this the canonical sequential circuit, because this is the most complete form of a sequential circuit. So, in the canonical sequential circuit what you have is, you have a set of flip flops which are called the state register. So, we have a state register, so even though it is shown as a single box here and in reality it could be any number of flip flops. Let us assume that there are some number of flip flops here and it is a register, which means it takes in a value and you can also tap it out in parallel.

So, you can give inputs in parallel to all the flip flops and you can also tap out the values in parallel from all the flip flops. So, you can go back and look at parallel in, parallel out register, just in case you want to refresh what this state registers. So, it is just a basic register, it is a collection of flip flops, each one can take one input and you can give one

output. The collection of these flip flops can all be accessed in one cycle.

So, you have a bunch of flip flops together and we have two forms of combinational circuits, one circuit called the input forming logic which is shown here and one circuit called the output forming logic which is shown here. So, what the input forming logic does is, it takes the primary inputs and it takes the values from the state registers. So, the output of the state register and the current set of inputs, you take them and then it is sent through this input forming logic or sometimes it is also called the next state logic, you have that.

And you also have the output forming logic, in which the state register along with some signals from the input forming logic form the output forming logic. So, let us see this circuit in it is, in all the glory that it has. So, you have a state register, the state register inputs we have going to call it the next state and the outputs we are going to call it the current state. So, at any point of time, so you have the current state and the current state goes through the input forming logic, this is a combinational circuit, they will come and appear as next state.

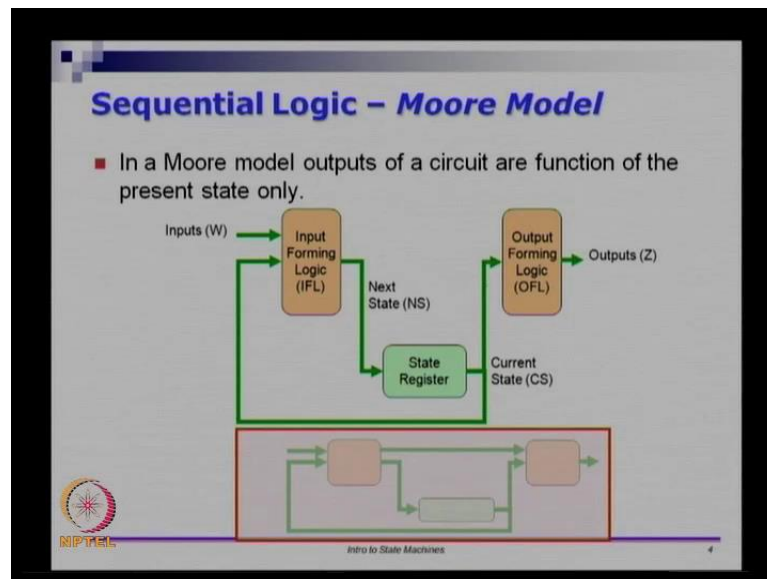
However, this next state will become, will get registered, will go into state register only when the clock comes in. So, I am going to start from the point where the clock is already come in and there are stable values in a register, we will call that the state of a system. And once I have state of the system, I take the current state and the set of inputs with the set of inputs in the current state, I can get what is called the next state and the next state will come in be at the inputs of the state register.

Until the next clock comes in, the values in the next state will not be copy to the current state. Remember, this one is a bus, it is not a single line it is a bus and similarly it is a collection of wires here, all these thick lines are actually multiple wires and each flip flop is supposed to take its corresponding input and place it at the output. And you also have this circuit, where you have the current state and the output forming logic which gives the outputs.

So, the meaning of this line is that the inputs of also pass on to the output forming logic. So, this arrow and this line here means, inputs are passed both to the input forming logic as well as to the output forming logic and from there you get the outputs. So, this is the canonical sequential circuit, so we will talk about various variations of this canonical circuit later.

But, as of now just remember that there is something called a state register which has N flip flops. There are two combinational blocks, the outputs come both from the input and the current state and the next state is derived from the current state and the inputs and such a machine model is called a Mealy machine model. So, it is named after a person it is called a Mealy model.

(Refer Slide Time: 06:35)



There is a small variation of this model, which is called the Moore model. In the Moore model what happens is, you have input forming logic, state register and output forming logic. However, the connection from the inputs to the output forming logic is missing. So, in the Moore model outputs of a circuit are a function of its current state only, it is not a function of the inputs.

Unlike the Mealy model which I have shown below, so I am showing it as a general module. So, this is a Mealy model in the bottom. The Mealy model if you notice, the inputs are going to the output, which means if I look at it from the primary outputs perspective, both the current inputs and the current state can control the outputs. However, in the Moore model the primary outputs are controlled only by the current state.

So, the current state dictates what the outputs are, it is not dependent on the current inputs. So, current inputs can only influence the state and state in turn has to influence the output, there is no direct influence of inputs on the primary outputs. So, such a model it is called the Moore model. So, these are two different models, we have already seen a

few circuits and some of these circuits actually fall under one or the other model, we can come back and look at that later.

(Refer Slide Time: 07:58)

**Analysis of Sequential Logic Circuits**

- Given a sequential circuit, what does it do? What do the state transitions look like?
- Analysis of sequential networks is done in three steps:
  1. **Logic Equations:** Determine the *flip-flop excitation* and sequential circuit *output logic equations*.
  2. **State Table:** State table is a tabular representation of the behavior of a sequential logic circuit. For a given input and present state of a circuit, it gives the output and the next state of the circuit. It is created using the characteristic table for the appropriate flip-flop.
  3. **State Diagram:** State diagram is a graphical depiction of a sequential logic circuit. Circles in this diagram depict the states and directed arcs depict the transitions.

The slide also contains two state transition diagrams. The **Mealy Model** diagram shows two states,  $S_j$  and  $S_k$ , as green circles. An input  $I_j$  leads to state  $S_j$  with output  $O_j$ . A transition from  $S_j$  to  $S_k$  is triggered by input  $I_k$  and produces output  $O_k$ . The **Moore Model** diagram shows two states,  $S_j/O_j$  and  $S_k/O_k$ , as green circles. An input  $I_j$  leads to state  $S_j/O_j$ . A transition from  $S_j/O_j$  to  $S_k/O_k$  is triggered by input  $I_k$ . The output  $O_k$  is associated with state  $S_k/O_k$ .

Analysis and Design of Sequential Logic Circuits 5

So, the first aspect of this video is going to be about analysis of sequential circuits. So, given a circuit, it is usually useful to find out, what the circuit is doing and this is the fundamental requirement from digital designers and it is also a good skill to have. Given a circuit what it does, rather than asking you to do design a circuit to solve a particular problem, this is the other way around. We are analyzing what a circuit is doing.

So, usually this analysis goes in three steps, the first step is you write a bunch of logic equations which determine the flip flop excitation and the sequential circuit. Basically, what it means is, if you given a circuit with flip flops in it, you have to first of all find out what the flip flops are, you should note the excitation table of the flip flops. If it is a D flip flop, you should know what it does, if it is a D flip flop you should know what it does and so on and you should go and write the output logic equations using the flip flop excitation.

So, when we, we will see a detail example in the next slide itself, but you need to know the table of equations for the flip flop. You should also be able to find out combinationally, what the next set of values are going to be, we will come to that in more detail later, then there is something called a state stable. So, state stable is just a representation of the behavior of the circuit. So, remember the canonical model, you are given a input and the current state of the system is stored in the registers.

So, based on the current state and the inputs that come in, you will have the next state which is affected. So, a state table is essentially a representation of given current state and given the current inputs, what is the output going to be and this current state could be any of the possible states. So, there are several valid states that a state register can be in. For each of these state combinations and for each of the input values, we have to specify what the next state is going to be.

So, the combination of what are the current inputs and current state to, what are the outputs and the next state, this is called a state table. So, again we will see an example in the next slide and finally, there is something called a state diagram. So, state diagram is a pictorial representation of what this circuit is doing. So, this is useful for analysis, because instead of looking at what the flip flops are or what the combinational logic is and so on, we can go and look at it as pure input output combinations.

We can just look at it as, these are the inputs and these are the outputs, if you have a state diagram. So, the state diagram has a few notations. Usually what you have is, you have all these states are represented by circles and you put a name inside them. So, typically we name them  $S$  underscore and number, so you have a circle which represents a state. So, if I take  $N$  flip flops, it can be in a state, so all the flip flops could have 0 or all the flip flops could have 1 or anything in between.

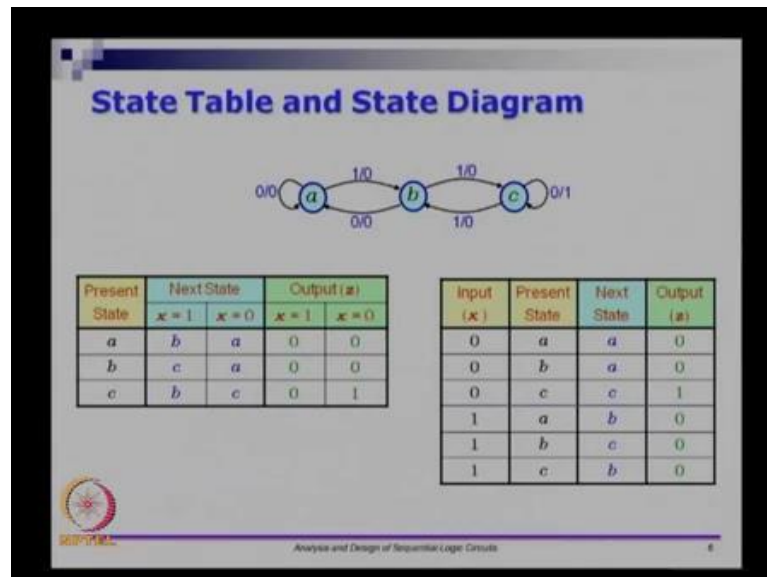
So, it is possible that your set of flip flops has  $2^n$  states, so the states could be in number  $s_0, s_1, s_2$  and so on have to  $s_{2^n - 1}$ . So, let us assume that there is some state  $j$ ,  $S_j$  and we are going to have this notation, that this state  $S_j$  you have a transition that is coming in to it. What you see here in the bottom is how you give notation for a state machine.

So, what you have is all these circles are represented by states. So, each state is marked with a number  $S_j, S_k$  and so on and the meaning of this part of the circuit is, if the current state is  $S_j$  and on getting some input  $I_k$ , it will go the state  $S_k$  and produce an output  $O_k$ . So, the current state comma input combination drives the circuit to state  $S_k$  and it should produce  $O_k$  as the output. So, such a model is called a Mealy model and equivalently there is a Moore model.

A Moore model is one in which, you have a state  $S_j$  and remember the outputs are completely controlled by the state itself. In a Moore model, the output depends only on the state. So, what you can do is, on arriving in the particular state  $S_j$ , you can

immediately tell what the different outputs are going to be. So, usually you mark it as  $S_j$  slash  $O_j$ . So,  $O_j$  is the set of all the outputs at state  $S_j$  and on getting an input  $I_k$ , you go to the state  $S_k$  again, because it is a Moore machine, the state uniquely determines what the output is going to be. So, from  $S_k$  you can uniquely say what all the outputs are going to be. So, this is a graphical representation, this will get a clear in a little while once we see the example.

(Refer Slide Time: 13:19)



So, let us take a particular state table and a state diagram, I will give a picture of all these things first and then we look at an example which takes a circuit and gets you these details. So, if you are already given a state machine for example, here it is a state machine, which has three states a, b and c and I want you to get comfortable with the notion of what the state diagram is and what a state table is.

So, let us look at this will call this a state diagram, because it is depicting states in this examples that are three states, the states are named a, b and c and let us look at each of these arcs. So, let us look at this arc first, so on this arc you see 0 slash 0, what that means is, on getting input 0, you produce an output 0, but you remain in the same state, the state a itself that is the meaning. So, the current state is a and if the input is 0, the output is 0 and you remain in the same state that is the meaning of this arc.

The meaning of this arc is, if the current state is a and if the input is 1, you produce a 0 and you go to state b, so this is the meaning of this arc. So, let me pick another arc here, the meaning of this arc is, if the current state is c and on 1, so anything before the slash is

input. If the current state is c and if the input is 1, you produce the output 0 and you go to state b. So, from the notation I can immediately say that, this is first of all a Mealy machine.

Because, the state is not dictating what the outputs are, the state transitions are dictating what the outputs are. So, when you go from a to b, on input 1 you produce 0 and when you are in a on 0 you produce 0. So, it looks like both the transitions from a are resulting in an output 0. However, you go and look at c, c on a 0 gives 1, but c on 1 gives 0. So, being in state c you cannot uniquely say, what the output is going to be, until you know the input also.

You should know whether the input is 0 or 1, if the input is 0 the output seems to be 1, if the input is 1, the output seems to be 0. So, given this I can say that this is actually a Mealy machine. So, this is a state diagram, let us not worry about what this is doing. So, if you are from Computer Science, you are probably seen what are called finite automata, either deterministic or non deterministic finite automata, you have drawn similar pictures.

In fact, there is very deep and equivalent connection between state diagrams that we see here in digital circuits and automata that you might have read in your discrete mathematics course. But, if you are not from the CS background, you do not have the panic. So, this is just a picture, this picture captures the notion of what are the values of flip flops and what will be the values of flip flops in the next cycle, what will be the outputs based on the current inputs, that is all it is capturing.

If you want to draw it as a state table, then what you usually do is, you go and write the present states. So, there are three states that are possible a, b and c, you write them as current states and you write two sets of columns. So, one set of columns called next state and one set of columns for outputs. In this case, there is only one output bit, there is only one input bit. So, let us assume that the input is called x and the output is called z, then the input can take 1 or 0, so the output is z.

So, let us look at the next state first, so let us ignore these columns for now, let us look at these two columns for now, let us see how to fill up these entries. So, if the present state is a and if x equals 1. So, if the present state is a and if x equals 1, you are going to state b here according to the picture, so you put b here. If current state is a and if 0 is the input, you come back to a itself, we mark that here. So, let us go and look at c for instance, c on



a 0 remains at c itself.

So, c on a 0 remains at c itself, c on 1 goes to b, so that is these two columns. So, will a leave for the output, the current state and the input will determine the output. So, a on a 1 gives 0, so a current state a on input 1 gives us 0, current state 1 on input 0 also produce as 0, that is because of this combination. So, if you are in a and if the input is 0, you are producing 0 as the output, so that is here. Let us look at c for instants, so c on a 0 gives 1 as output, c on a 1 gives 0 as output.

So, what these two are capturing are the next state comma output combination. The rows are listing the present state and these two columns are representing the other combinations which are the inputs. So, this is one way of drawing this state stable, another way of drawing the state table is as follows. So, you go and put all the input combinations, so the inputs could be 0 or 1, if the input is 0 the present states could be a, b or c.

Similarly, when input is 1 the present states could be a, b or c, this is listing all the combinations. If you look at it, it is looking at all the possible values of x into all the possible values of present state. So, there are two possible values of x, three possible values of present state, there are six different rows. For these rows, you then go and mark the next state comma output, so this is another way of drawing the state stable.

So, if you are given a state diagram, you should be able to draw a state table. If there are two inputs here, then this input combination instead of 0 and 1, you may have 0 0 0 1 1 0 and 1 1. So, in general if you have n inputs and if you have n states, then you will have  $2^n$  into m rows and the number of columns. See in this case, again there is only 1 output, if we have more outputs you just keep adding columns. So, if you have 2 outputs, you put z 1 and z naught may be. So, that is the general way in which you will draw the state stable given a state diagram. So, we will look at only single input, single output circuits for some time, before we go to the other circuits.

(Refer Slide Time: 20:15)

**Problem: What does the circuit do?**

1. Determine the **flip-flop excitation** and **sequential circuit output logic equations**.

$$D_0 = \overline{Q_0}$$
$$D_1 = Q_1 \overline{Q_0} + \overline{Q_1} Q_0$$

(there are no explicit outputs in the above circuit)

**MPTEL** Analysis and Design of Sequential Logic Circuits 7

So, now let us go and do analysis, I am giving you a circuit now. So, this circuit has two flip flops and that is some connection here and I also tell you that these are D flip flops, I am asking you what the circuit is doing, if you want to analyze a circuit like this, now we will go through three steps. So, I mentioned this in the slide earlier, we will go through three steps, the first step that we are going to do is, we are going to figure out what the flip flops are, what is the flip flop.

So, I have already mentioned that these are D flip flops, the second part of the first bullet is, we also have to find out what the equations are. So, now, let us go and look at the D inputs for each one of them, there are two flip flops  $Q_0$  and  $Q_1$ . Let us go and look at the D inputs, if I look at  $D_0$ ,  $D_0$  seems to be coming from  $\overline{Q_0}$  of this flip flop. So, I can write  $D_0$  is  $\overline{Q_0}$ . So, if you think about it these are two flip flops, if this is my state register, then this input is coming in here directly, so  $\overline{Q_0}$  is coming in here directly, which means  $D_0$  is  $\overline{Q_0}$ .

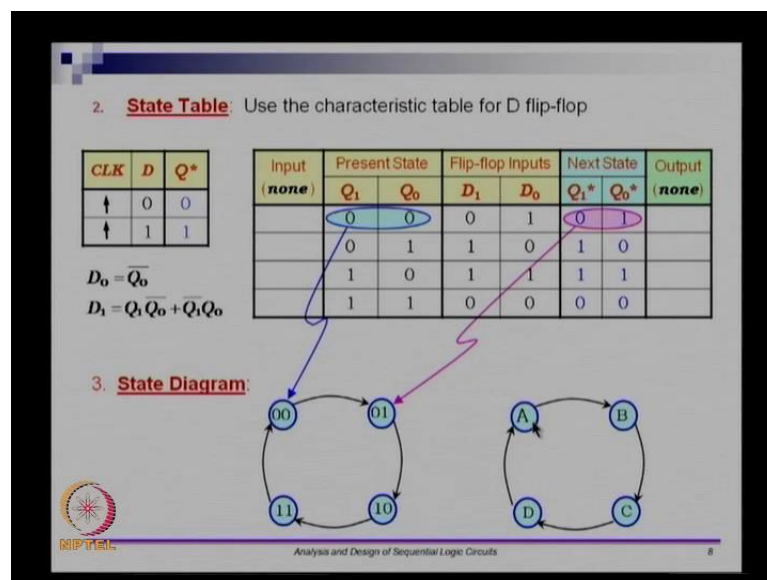
So, it looks like this is part of the next state forming logic, if you take  $Q_0$  complemented give it to  $D_0$  that will be a next state logic. So, from the current state and the inputs, in this circuit that does not have any input. So, based on the current state you are able to say what the next state is, so  $\overline{Q_0}$  is  $D_0$ . Let us go and look at the functionality of  $D_1$ , so it takes  $Q_1$  and  $\overline{Q_0}$  and it then takes  $Q_1$  and  $\overline{Q_1} Q_0$  and it adds them together.

So, it is actually doing  $Q_1, \overline{Q_0} + \overline{Q_1} Q_0$ .

So, this one there is actually the function of XOR between  $Q_1$  and  $Q$  naught, so  $D_1$  is  $Q_1 \text{ XOR } Q_0$  and these two are the values for  $D_0$  and  $D_1$ . So, there are no explicit outputs from the circuit, there are no explicit inputs to the circuit that the since to be a clock and only the flip flops.

So, in this case what we have is, we have some next state logic and there is no output logic. So, this is why I gave you the canonical picture, so in the canonical picture you have flip flops, you have combinational logic for the next state logic and you have combinational logic for the output forming logic, in this case the output forming logic is not there. So, it degenerates the canonical case to something else, but it still a valid sequential circuit, because we still have flip flops in them. Now, let us do the second step from here if I want to write the state diagram in state stable.

(Refer Slide Time: 23:18)



So, let us first write down the characteristic table of the D flip flop, we have already return that equations, the characteristic table is on rising edge of clock, when D is 0 Q will be 0 and on an rising edge of clock if D is 1 Q is 1 we already have that, now let us go and write the state table. So, we know that there are no inputs and we know that there are no outputs, so these two columns can be ignored, but I have given here and I am marked it empty to say that there is nothing to do here.

Then, I have  $Q_1$  and  $Q$  naught which are the present states and I want to see what the flip flop inputs are going to be they are  $D_1$  and  $D$  naught. So, let us ignore this column for a while now, you have  $D_1$  and  $D$  naught, so based on  $Q_1$  and  $Q$  naught there are

four possible combinations of  $Q_1$ ,  $Q_{\text{naught}}$ . So,  $Q_1 Q_{\text{naught}}$  is could both be 0 0 1 1 0 and 1 1 let us plug in these values to get the  $D_{\text{naught}}$  and  $D_1$ , so  $D_{\text{naught}}$  must be  $Q_{\text{naught}}$  bar.

So, if you look at this column, it will be the complement of this column, so this is 0 1 0 1, so  $D_{\text{naught}}$  will be 1 0 1 0 and  $D_1$  is  $Q_1 Q_{\text{naught}}$  complement plus  $Q_1$  complement not  $Q_{\text{naught}}$  or XOR of these two. So, you take the XOR of these two, so that will give you the vector 0 1, 1 0 these two columns seem to be the inputs that are coming into the a flip flop. So, from  $Q$  they when back to the flip flop, so these are the inputs that are coming into the flip flop.

Now, if these are inputs coming into D flip flop, then the clock comes in what would happen to the flip flops, then if the inputs had 0 and 1 at  $D_1$  and  $D_{\text{naught}}$ , after the clock comes in  $Q_1$  will get 0 and  $Q_{\text{naught}}$  will get 1. So, remember the star notation, star notation means next state, so at  $t$  if  $D_1$  is 0 and  $D_{\text{naught}}$  is 1 at  $t$  plus 1 or just after the clock comes in  $Q_1$  will becomes 0 and  $Q_{\text{naught}}$  will become 1.

So, if you want to get the next state, then at least for the D flip flop case you copy the D inputs itself directly those will be the next states. So, 0 1 you copy directly, 1 0 you copying, 1 1 you copying and 0 0 you copying. Now, what you have is, you have present states, you have inputs, you have next states and you have outputs, you do not need the flip flop excitation table, we do not need this one in anymore.

Because, this flip flop we got the equations from there we regard the states, so we do not need these two columns any more. Now, let us go and look at in input comma present state to output comma next state combination, in this example there are no inputs and there are no outputs. So, there is just a present state to next state transition, so let us look at the state diagram, if the present state is 0 0. So,  $Q_1 Q_{\text{naught}}$  up to different flip flops and if the take value 0 0, I am going to mark that using this blue circle and I am going to call this state 0 0, I am giving a name to it state 0 0.

So, state 0 0 what happens is, if you are in state 0 0 and when you get a clock pulse you are suppose to go to the next state and what is the next state you are going to go to, the next state that you are going to go to is on this these two columns, you go to 0 1. So, because that is what we have derive, so for. So, this 0 1 is the next state, then let us go and look at 0 1, if 0 1 is the present state that is this row, we go to 1 0 which is the next state, so we mark that and if you are in 1 0 as the present state, we go to 1 1 as the next

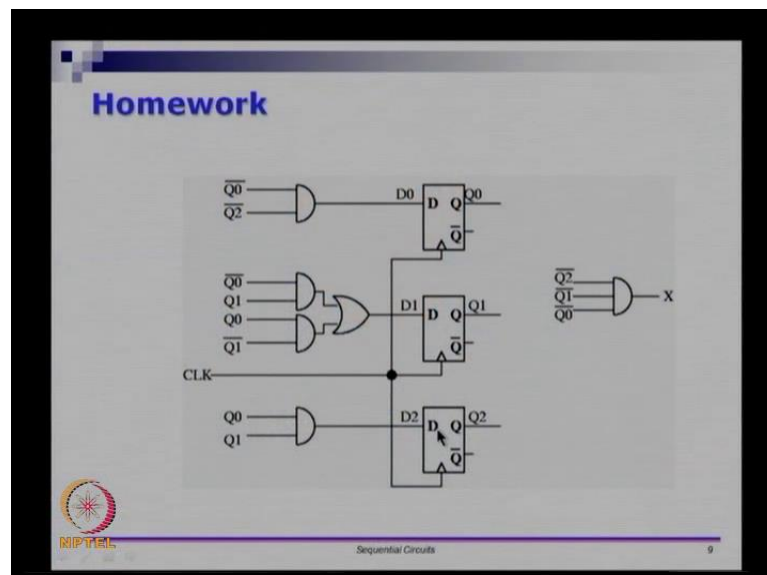
state and if you are in 1 1 as the present state, we go to 0 0 as the next state.

So, it looks like if we start with state 0 0, in one clock when the first clock pulse comes will go to state 0 1, next clock pulse will go to 1 0, next clock pulse will go to 1 1 and the next clock pulse will go to 0 0. So, I given the numbers here, we can also name them a, b, c, d if you want. So, this is marking what the values of the flip flops are directly, many times you just give the names to it, you give the names A, B, C, D and so on.

So, the way to read this picture is, if you are in state A no matter what the input does not, this case there is no input, if you are in state A, the next clock cycle will go to B, next clock cycle will go to CM next clock cycle will go to D and then you come back to A in the next clock cycle. And this keeps repeating forever as long as there are clock inputs given to the circuit, this will keep dangling from A to B to C to D that back to A and this will keep going inside them.

So, in fact this is just the state diagram of the up counter that we have, we just design that in the last class. So, this up counter if I go and read the flip flop values, so if I go and read the flip flops and directly give them as output, then this is actually the state diagram for an up counter. So, up counter should go from 0 0 to 0 1 to 1 0 to 1 1 back to 0 0, so this is a picture for a up counter. So, I want to you do this as homework.

(Refer Slide Time: 29:07)



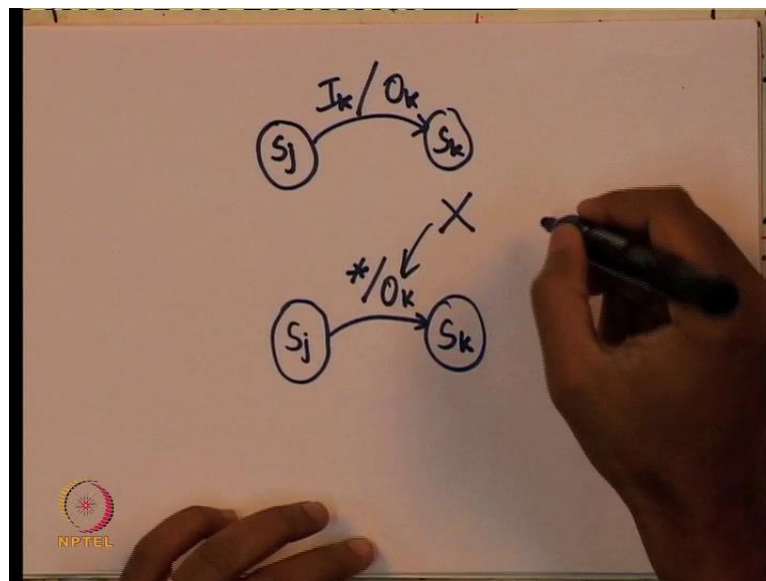
So, we are given 3 flip flops Q naught, Q 1 and Q 2 and you should always read this vectors Q 2, Q 1, Q naught. So, there are 3 flip flops, the Q 2 is the most significant bit and Q naught is the least significant bit and there is a certain connection given here, the

connections are not drawn as wires, but you can see where they are coming from. So, in this case  $Q$  naught bar is coming here,  $Q_2$  bar is coming here and so on.

The actual wires are not shown, if this is the case and if there is a common clock input, whatever would like to do is, based on the states, in this circuit again there is no explicit input to the circuit. However, there is an output called  $x$  which is derived from  $Q_2$ ,  $Q_1$  and  $Q$  naught and this is the logic description for  $x$ . What I want you to do is, go and write the state tables for it and draw the state diagram for this.

So, in this state diagram you will not have an input at all, you will only have a state transition which marks, if you are in a current state what will be the state that you go to in the next clock cycle and what will be the output that you will produce, you will not have a input combinations. So, in the state diagram that I showed earlier I said current state comma input combination will give output comma next state. So, in this case there is no input.

(Refer Slide Time: 30:38)



So, let us look at a picture here what we usually have in a state machine is, if we have state  $S_j$  on input  $I_k$ , it produces output  $O_k$  and goes to state  $k$ . So, this is the picture that I showed for a mealy machine or the canonical machine, in for this example what you would have is something like this, on state  $S_j$  you will go to  $S_k$  producing  $O_k$  and inputs do not matter, there is no input to the circuit explicitly, this picture also you can see that and this is  $O_k$  actually just a single bit  $X$ . So, what I want you to do is, broad table of this form. So, let see what you should be drawing, so I want you to draw table

like this.

(Refer Slide Time: 31:30)

CS			NS			
$Q_2$	$Q_1$	$Q_0$	$Q_2^*$	$Q_1^*$	$Q_0^*$	$X$
0	0	0				

If the current inputs are  $Q_2$ ,  $Q_1$ ,  $Q_0$  and you should figure out how many combinations are there. So, this is for the current state, you should tell me what the next state will be, which means you should give me  $Q_2^*$ ,  $Q_1^*$ ,  $Q_0^*$  should be that and should also tell me what  $x$  is going to be. So, you fill up for various combinations here to tell me what these entries are going to be, go and write this down. So, this is going to be part of your ((Refer Time: 32:06)) it is going to be a quiz question also, so let us move on to few different times of problems.

(Refer Slide Time: 32:13)

**Problem: Detect two consecutive 1's**

- A single input bit  $w$  is coming in one bit at a time.
- All changes must occur in the positive edge of the clock
- Detect two consecutive 1's. Output  $z$  must be 1 if for two immediately preceding cycles the input  $w$  was 1.

Clockcycle:	$t_0$	$t_1$	$t_2$	$t_3$	$t_4$	$t_5$	$t_6$	$t_7$	$t_8$	$t_9$	$t_{10}$
$w$ :	0	1	0	1	1	0	1	1	1	0	0
$z$ :	0	0	0	0	1	0	0	1	1	0	0

NPTEL

Intro to State Machines

10

So, one of the things that we do sequence of circuits is, we can do there are two different kinds of state machines. State machine is essentially you have a set of flip flops, you have the current state, you have the next state, you have inputs and outputs and such state machines can be use to do two different kinds of things, one it can actually do what is called detection. So, let us take a specific problem, I want to detect two consecutive 1s that is the problem description, let see the problem description in more detail.

So, you circuit a suppose to take a single input  $w$  and this input is coming in one bit at a time, there is only one bit that is give to you at a time. But, there is a serious of a inputs is coming in and this may be a never ending serious and all changes must occur in the positive edge of the clock, I am giving that you as constraint and what I want to do is, when the inputs are coming in I want to detect if there are two consecutive 1s in the circuit, so in the inputs.

So, the circuit should detect if there are two consecutive 1s, output  $z$  must be 1 if for two immediately preceding cycles, the input  $w$  was 1. So, you go and look at the current cycle, you go and look at the previous cycle, if both of them have 1 has the input, then the output should go to 1; otherwise, it should remain at 0. So, this is the problem description, I will give you a small example here.

So, let us assume that I have given you something for 11 clock cycles it could be never ending. So, there could be more clock cycles later and I am given you inputs for 11 clock cycles from  $t$  naught to  $t$  10. So, they are given as  $w$  and we are getting it as 0 1, 0 1, 1 0, 1 1, 1 0 0 and so on, the output  $z$  is defined as it should be 1, if the current input is 1 and then the preceding clock cycle the input was 1.

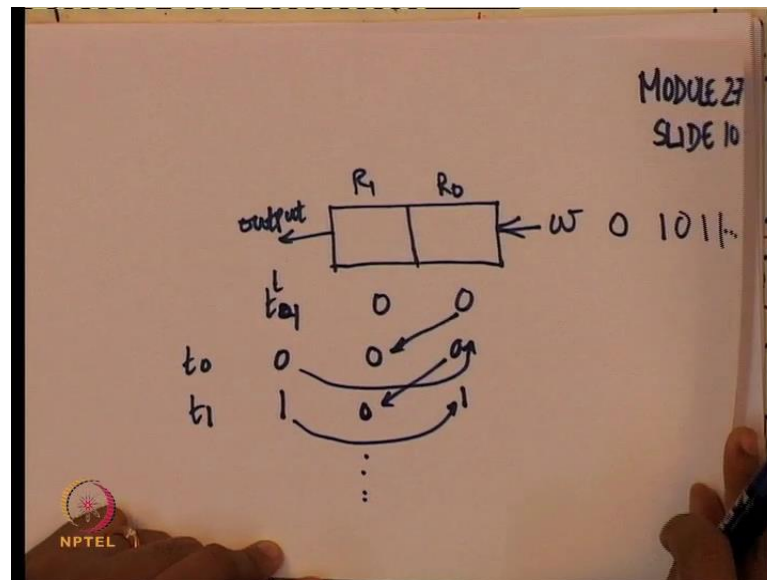
So, let us go and look at the series here, the current input is 1, but the preceding input is not 1. So, the output is 0 here, the current input is 1, the preceding input was not 1, so the output is 0 here, the current input is 1, so the preceding input is 1, so the output is 1 here. So, where ever you see a 1 1 combination you will get a 1, this output is also 1 because at  $t$  8 them input is 1 at  $t$  7 also the input is 1. So, preceding cycle and this cycle you should have 1 that is the description that we are asking.

So, this cycle and the preceding cycle we want them to be we want, so we want that here. So, in all the other places we have a 0, so let us say this is the problem that we have at a time and I want it to design a circuit for this. So, one way to do this, this as follows, so let us look at the picture here, so I am in module 27 and slide time. So, what I am going



to do is, so I say first of all the inputs are coming in 1 bit at a time, so whenever you see some description like that immediately what should come to your mind is, this is coming 1 bit at a time is there a case for a shift register. So, that is what come to you are mind.

(Refer Slide Time: 35:35)

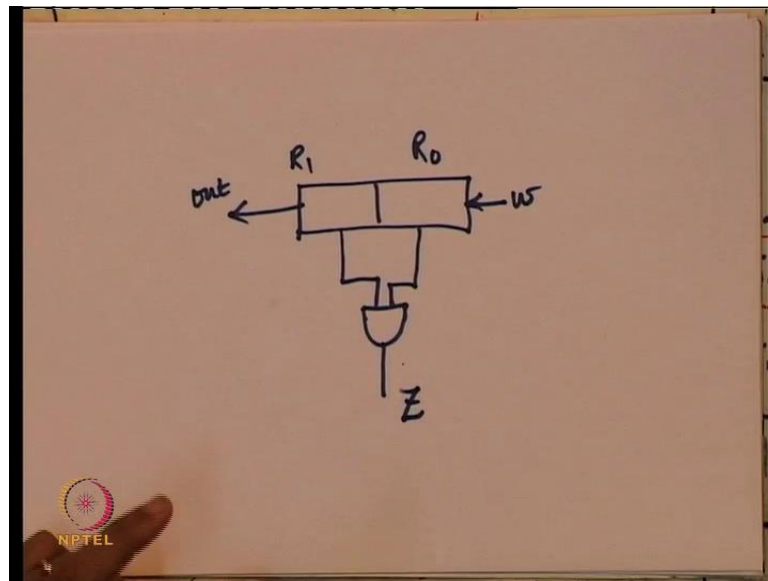


So, I am going to see if that is the case, so let say that there is a  $w$  that is coming in and if I use a shift register, I will have some output. So, there is some output going out, the some output, now first thing I want to see is, there are inputs coming in I want to find out if the current input and the preceding input or both once. So, if I want to do that then if I put it into a 2 bit shift register, so when the input comes in one clock cycle it comes here, the next clock cycle it comes here.

So, if I want to find out something in the in two consecutive clock cycles, then I can look at a 2 bit shift register. So,  $w$  is coming in as input, let me call this  $R_{naught}$  and this  $R_1$ , if I put it out 2 bit shift register in this case is a left shift register and this will go as  $R_{naught}$  and  $R_1$ . So, if I keep giving inputs to it, then so the sequence I gave was 0 1 0 1 1 and so on, I gave this as a sequence. So, initially 0 will get here, then 0 will move here and 1 will come here then...

So, let us write it down at... So, let us assume that at  $t_{naught}$  or let say  $t - 1$  the both of them are initialize to 0, then at  $t_{naught}$  I am getting 0 as input. So, what would happen is this 0 gets shifted here and this 0 gets shifted here. So, you will have 0 0 at  $t_1$  if I give 1 as an input, this 1 gets shifted to  $r_{naught}$  and this 0 get shifted here and so on, now since I used 2 bit shift register.

(Refer Slide Time: 37:38)



So, let me see how to solve the problem that we have a time, so we had  $w$  as the input and there was some output that is coming out, let us 1 bit output is coming out, if I take these two bits and I AND it together, what you get from this if both of these are 1, then the AND gate will give a output of 1. So, what this is capturing is, if there is any two consecutive cycles at which the inputs are both 1.


So, if there are two consecutive cycles were it was 1, if the input was 1 it will eventually appear in the shift register as two 1s. So, at some point of  $w$  is 1 1, then at some point  $R_1$  and  $R_0$  will both become 1, when  $R_1$  and  $R_0$  are both 1 the AND gate will have an output of 1, this what we asked in the problem. So, if I call this my output  $z$ , then this circuit is actually capturing the problem at hand. So, problem at hand was if there are two 1s in the inputs consecutively, then the output must be a 1 and this is detecting the case where there are two consecutive 1s. So, this is the fairly simple example let us make it slightly more complicated.

(Refer Slide Time: 38:58)

**Problem: Detect two consecutive 1's or 0's**

- A single input bit  $w$  is coming in one bit at a time.
- All changes must occur in the positive edge of the clock
- Detect two consecutive 1's. Output  $z$  must be 1 if for two preceding cycles the input  $w$  had 1 in both cycles or 0 in both cycles.

Clockcycle:	$t_0$	$t_1$	$t_2$	$t_3$	$t_4$	$t_5$	$t_6$	$t_7$	$t_8$	$t_9$	$t_{10}$
$w$ :	0	1	0	1	1	0	1	1	1	0	0
$z$ :	0	1	0	0	1	0	0	1	1	0	1

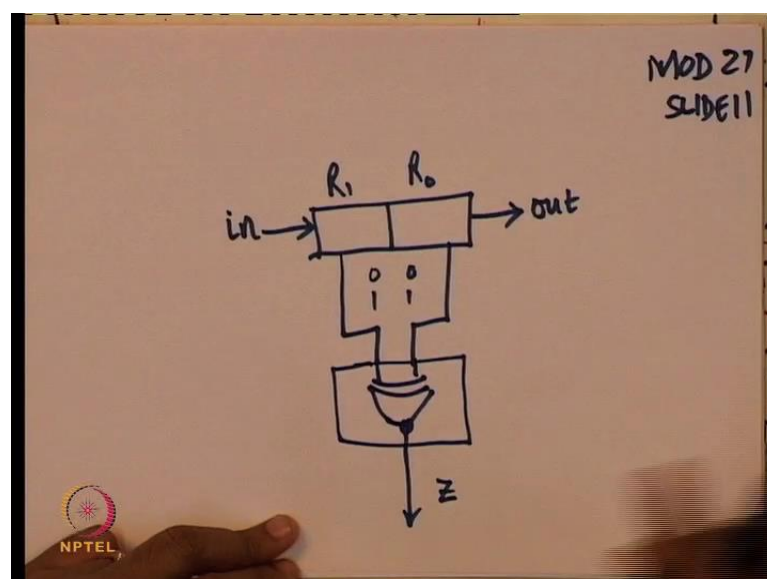
 NPTEL

Intro to State Machines

11

So, in this problem as before we have a single input bit  $w$ , we want all the changes to occur in the positive edges of the clock. So, we want the shift register essentially to be a positive edge triggered register. And now output must be 1 if for two consecutive cycles input had either 1 in both cycles or 0 in both cycles. So, let us look at this one, so here we have one two consecutive cycle. So, the output is 1 at this point you have two 1s in consecutive cycles the output is 1 at  $t_7$  and  $t_8$  are both 1 the output is 1, at  $t_{10}$  both the inputs are 0 the output is 1. Let say I want to take this problem and solve it, then again it is only a minor extension to what we did earlier. So, again I will draw the picture now.

(Refer Slide Time: 39:58)



So, we at slide 11 as before I will give it to a 2 bit register and I am now going to give it to a right shift register, just for the ((Refer Time: 40:06)) no particular reason. So, I will call this R 1 and R naught, the inputs are coming in from the left side and they are going to the right side. So, there is no reason why I should have used a left shift register in the previous one, there is no reason why I should use a right shift register here.

So, I am using a arbitrarily, now what want is if the combination is either 0 0 here or 1 1 here I want the output to be 1; otherwise, the output must be 0. So, what I could then do is, I can take these two bits, these two bits if I capture them and if I connect them with a circuit. So, I am going to put a circuit which is going to give my z, I want to put a circuit in such way that in two consecutive clock cycles, if the inputs are the same the output should be a 1.


So, if I want to two consecutive clock cycles I tap two consecutive positions in the shift register, their only two positions here R 1 and R naught and I want to give this combination for 0 0 it should be 1 for 1 1 it should be 1. So, to do that if I put an XNOR gate, if both the inputs are 0 or if both the inputs are 1, the output will be a 1. For other combinations, if I have a 0 1 here or a 1 0 here there are not two consecutive 1s there are not two consecutive 0s, in those cases the output will be a 0. So, for this problem for detecting to consecutive 1s or to consecutive 0s, this circuit will work.

(Refer Slide Time: 41:45)

**Problem: Detect 011**

- A single input bit  $w$  is coming in one bit at a time.
- All changes must occur in the positive edge of the clock
- Detect two consecutive 1's. Output  $z$  must be 1 if for three preceding cycles the input  $w$  had 011 as the input pattern.

Clockcycle:	$t_0$	$t_1$	$t_2$	$t_3$	$t_4$	$t_5$	$t_6$	$t_7$	$t_8$	$t_9$	$t_{10}$
$w$ :	0	1	0	1	1	0	1	1	1	0	0
$z$ :	0	0	0	0	1	0	0	1	0	0	0

 NPTEL

Intro to State Machines

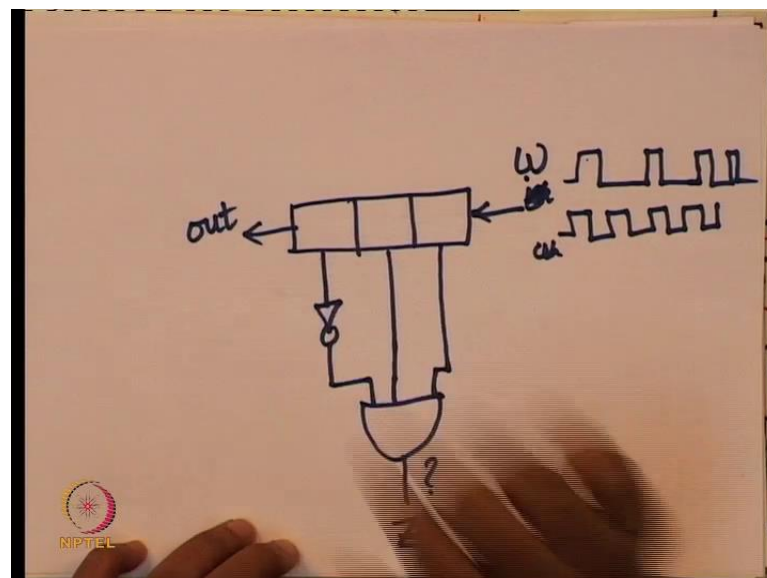
12

Let me make a slightly another modification to this problem, I want to detect a sequence 0 1 1. So, 0 1 1 is an input sequence, so I am having again a single bit  $w$  and it is getting

a 1 bit at a time, whenever there is a 0 1 1 I want to detecting. So, now, let look at this case, so there is this 0 1 1 here, so the output should be 1 here, again there is a 0 1 1 here the output should be 1 here, then all the other places it is 0, if you want to do something like this, first thing is you are asked for three consecutive cycles.

If you want to get three consecutive cycles, then the first thing you have to think about is how many flip flops do you want to remember three consecutive cycles. So, can you do the just one flip flop or two flip flops to the answer is no, you cannot remember three values your asking for three values in the consecutive cycles, how can you do with only one flip flops or two flip flops, you have to remember three of them.

(Refer Slide Time: 42:44)



So, as before I am going to use a 3 bit shift register, so earlier I used 2 bit shift register. Now, I am going to use 3 bit shift register I assuming that the inputs are coming in from the right side and the shift register is giving the output on the left side. So, at some point if I want 0 1 1 to be detected. So, the input is coming like this, let us say something like this and the clock pulse is like this, so clock pulse is something periodic.

So, you are looking at what the values of inputs are at difference points of clock. So, the inputs are coming in 1 bit at time, I want to detected there is an output of 0 1 1 if I want to do that, then I want this to be 0, I want this is to be a 1 and this is to be a 1. So, I take those combinations give it to an AND gate, when will this AND gate give a 1 this AND gate it will give a 1 only when this bit is 0, this bit is 1 and this bit is 1, which means at some point of time the input sequence you should I had 0 followed by 1 followed by 1,

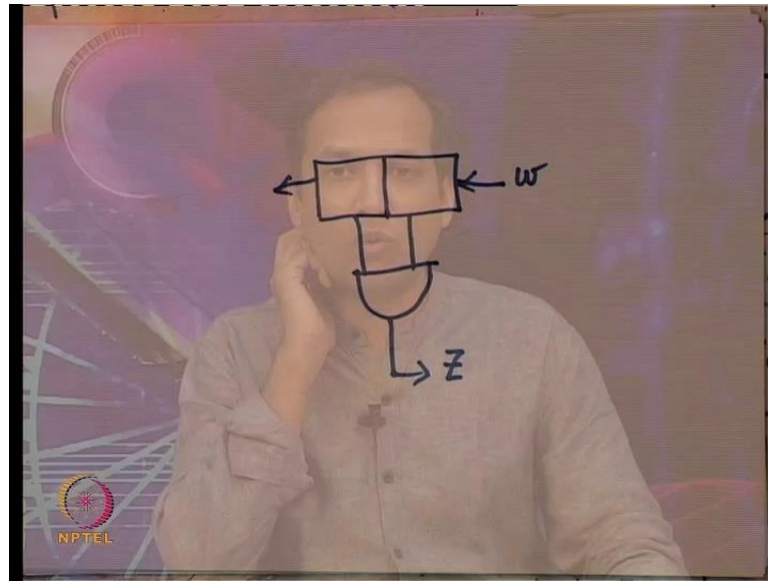
only in that case the output will become the AND gate if I will 1 and this I can write to my x.

So, if give w as input, then we get z as output and what we have is a fairly simple circuit in which you get this whole thing running. So, all these are sequential circuits, what we have is, this is like the state this sequential circuit that I showed earlier, except that there are few issues here, it is not like a canonical sequential circuit. Because, your taking in the inputs, the inputs are fed to the flip flops which is, there is some logic connecting one flip flop to the other and so on, the output is a function only of the currents states.

So, this is actually a Moore machine, the current input does not influence the output directly, the current input gets into the register and registered drive the output. So, this is a Moore machine, so it is a slightly degenerate case of the canonical model that I showed earlier. Now, we saw three different examples, we saw a sequence detection of 1 1, we saw sequence detection either 0 0 or 1 1 and now we are able to detect 0 1 1.

So, now, you can go and make up your own example detect 1 0 1, detect 1 0 1 1 or whatever and you trying a come up with the circuit own your own. So, the first thing will notice is for a problem like this it is seems to be trivial to come up with design like this. So, I want to you get a handle of such a model, so the key thing here is when we did this, there is a certain state of the system and this certain state of the system is actually changing the way things are working. So, there is a certain state and this state keeps changing, so what I would like you to do is, go and draw state machine for the detect 1 1 problems.

(Refer Slide Time: 46:00)



So, this is the circuit that I gave, so I said the input is coming in from one side and for detect 1 1 I said this is the circuit. What I would like you to do is, go and draw the state stable and the state diagram for this particular circuit. Once you get a handle of this go and try doing it for this 0 1 1 problem also, the circuit that I gave for the 0 1 1 problem, go and write the state stable and the state diagram for it. So, this will get into the habit of drawings state stable, state diagrams and so on.

And so once you do this slowly we will start connecting how this state machines were built, for these problems the state machine was easy to built. But, we have going to see more complicated examples in the next week, where coming up it is a such a circuit is not going to be easy, we need a more methodical and a structural very systematic way of approaching that problem, it cannot be something that we can think about and come up in a relatively quick manner.

So, we will see such problems where coming up with a circuit is not easy, we need a mechanism in where we actually start with the state diagram and then come up with the tables and then come up with the circuit, rather than the other way around. So, in the next week what we are going to see is, we are going to see what is call this synthesis problem, right now I ask for analysis and given a state diagram or given a circuit what give me the state diagram and so on.

So, given a circuit like this, give me the state diagram is the analysis problem. The synthesis problem is the other way around, given a state diagram or is give the state

stable and the circuit or given a state stable, draw the state diagram and give the circuit which realize is that and so on that is call the synthesis problem and we will see the synthesis problem in the next class.

So, what this does is it this brings me to the end of week 5 related to all the circuit a videos, there is one more video that I am going to record which is on Verilog. So, in this week also we will do one more module of Verilog and we will learn about what is called the always block. We have already seeing the assign statement, we have seen how to instantiate, we know how to use primitives, in the next video on Verilog your going to see something called the always block. I will give you the motivation required for that and that will be the final video for the week 5. So, this video actually brings me to the end of circuit related material for a week 5.

So, thank you and I will see you in the next video before we close this week, bye.