Switching Circuits and Logic Design

Prof. Indranil Sengupta
Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur

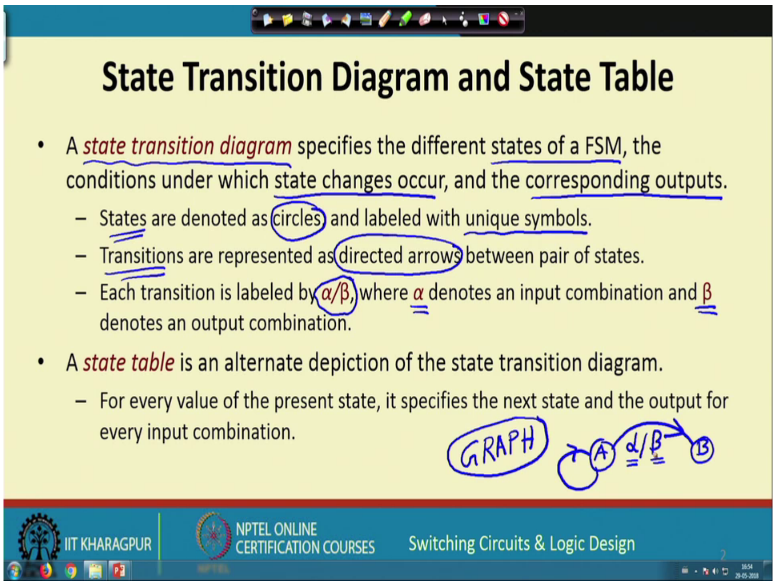
Lecture - 37
Synthesis of Synchronous Sequential Circuits (Part II)

So, we continue with our discussion on the Synthesis of Synchronous Sequential Circuits which we have started in the last lecture. So, we continue with the discussion in this second part of this lecture Part 2. Now, before we start you recall in the last lecture we mentioned the different steps that you need to go through to synthesize or design a synchronous sequential circuit.

The first step is most important is to capture the behavior from some specification, it can be a textual specification it can be any kind of specification, you may be having some idea in your head you are trying to capture it or there is some text which is written from there you can you are trying to capture the behavior.

So, you are trying to capture the behavior and you are try to formulize it by expressing it in the form of a state table or a state transition diagram. Now in this lecture we shall be spending some time specifically on this part of the entire flow that how to construct state transition diagrams and state tables starting from specifications.

(Refer Slide Time: 01:40)



State Transition Diagram and State Table

- A *state transition diagram* specifies the different states of a FSM, the conditions under which state changes occur, and the corresponding outputs.
  - States are denoted as circles and labeled with unique symbols.
  - Transitions are represented as directed arrows between pair of states.
  - Each transition is labeled by α/β, where α denotes an input combination and β denotes an output combination.
- A *state table* is an alternate depiction of the state transition diagram.
  - For every value of the present state, it specifies the next state and the output for every input combination.

GRAPH
A
α/β
B
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Design

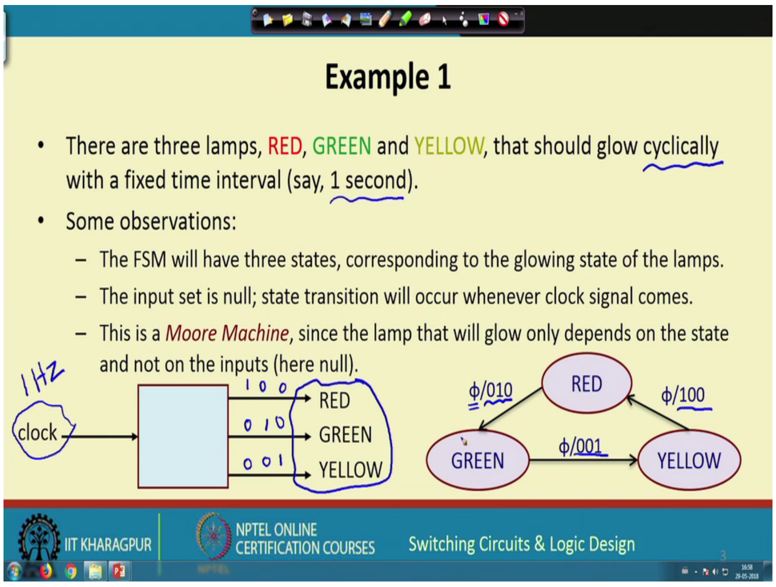
So, we concentrate on state transition diagrams and state tables in this lecture. So here we are going into a little more detail. So, here although we have mentioned this I am repeating; a state transition diagram when you see it is actually a graph, a graph contains some vertex some vertices and edges; it is a graph which specifies the different states of the Finite State Machine, the different conditions or the inputs under which the state changes will occur and also the output values that will be generated.

So, this state transition diagram captures the whole behavior of the FSM. Now notation wise this states they are denoted as circles and they are typically labeled with unique symbols, so that you can distinguish one state from the other. While in the earlier example you saw in the last lecture we labeled this states as A B C D fine.

Now there are states there are also transitions for example, this is one state this is one state, they are labeled with A and B, there can be a transition between A and B which is denoted by a directed edge an edge with an arrow, these are called transitions they are represented as directed arrows between a pair of states. Well these arrows can be self loops also, so a state an itself it can also be like this and each transition is labeled by 2 numbers alpha and beta where alpha denotes some input combinations and beta denotes an output combination.

So, if I write here alpha comma beta, it indicates that if you are in state A and input alpha has come then you go to state B and generate the output beta right, this is what the interpretation is. Now as we had said we shall again see some examples. State table is an alternate depiction of the state transition diagram, it is not something separate they represent the same information instead of showing it in a diagrammatic form it show it in a form of table that is the only difference ok. So, in this table just like in the diagram for every value of the present state, here the next state is specified and also the output values for each input combinations are specified right. So, these we will see, we will take a number of examples.

(Refer Slide Time: 04:44)



Example 1

- There are three lamps, RED, GREEN and YELLOW, that should glow cyclically with a fixed time interval (say, 1 second).
- Some observations:
  - The FSM will have three states, corresponding to the glowing state of the lamps.
  - The input set is null; state transition will occur whenever clock signal comes.
  - This is a *Moore Machine*, since the lamp that will glow only depends on the state and not on the inputs (here null).

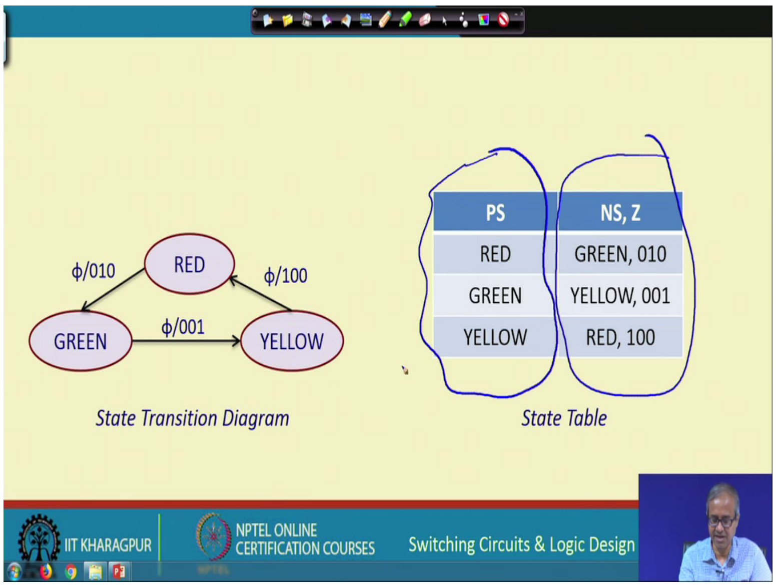
So, the first example let us take these this is a very simple example; a example goes like this. This is a very simplified form of a traffic light controller this is nowhere near to a traffic light controller but something similar, the only similarity is that there are 3 lamps; red, yellow and green; nothing else is similar form of a traffic light controller ok. Here the problem specification is like this; there are 3 lamps red, green and yellow that glows cyclically; that means, first red then green then yellow and again back to red and let us say with a fixed time interval let us say 1 second. Now this time interval we are not really bothered about because, will assume that our circuit is like this, there is a clock if I need a 1 second period then I will be applying a 1 hertz clock.

So, changes will happen every 1 second there are no other inputs there are no other inputs in the circuit and the outputs are red, green and yellow ok. Now first thing is that how to represent this problem in the form of a finite state machine. Frst thing is that let us define the states there will be 3 states corresponding to the 3 lamps that are glowing, red, yellow and green, the arrows will indicate state transition, red to green, green to yellow and yellow to red and the other thing we are assuming is that there are 3 outputs red, green, yellow let us assume in this order.

So, where when you are in the red state this line will be 1 this will be 0 this will be 0 which means red is glowing, if you want to glow green this will be 1 this will be 0 this will be 0 and if you want to glow yellow this will 1 this is 0 this is 0. So, now you can understand you see there are no separate inputs, that is why the input is shown as empty; 5 means empty.

So, whenever in the red state and a clock comes you go to green, the input is not the input is 5 and the output is 0 1 0; 0 1 0 means green is glowing ok. When you are in green and a clock comes you go to yellow 0 0 1 which means yellow is glowing and when you are in yellow you will go to red 1 0 0 means red and these goes cyclically right and you see because there are no inputs the states directly determines what the outputs will be. So, this is like a Moore Machine, the output depends only on this state it really do not depend on the applied input.

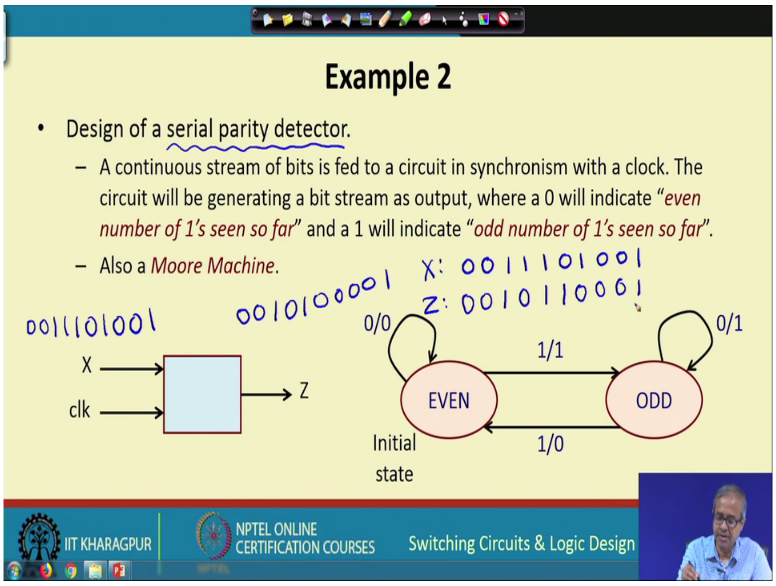
(Refer Slide Time: 08:01)



PS	NS, Z
RED	GREEN, 010
GREEN	YELLOW, 001
YELLOW	RED, 100

So, talking of the state table here this is a very simple problem. So state transition diagram is shown on the left, on the right side I am shown the equivalent state table. So, broadly there are 2 columns I had shown in the example earlier, one is for the present state, other is for the next state and outputs because there is no inputs there is a single column in the second part, the present state if it is red then the next state will be green and the output will be 0 1 0. If the present state is green next state will be yellow output 0 0 1, if it is yellow next state is red output is 1 0 0, so exactly what this state transition diagram specifies right. So, let us move on to slightly more complex problems to see how the state transition diagram and the state table can be constructed from there.

(Refer Slide Time: 09:04)



Example 2

- Design of a serial parity detector.
  - A continuous stream of bits is fed to a circuit in synchronism with a clock. The circuit will be generating a bit stream as output, where a 0 will indicate "*even number of 1's seen so far*" and a 1 will indicate "*odd number of 1's seen so far*".
  - Also a *Moore Machine*.

Let us see. This is a slightly more difficult problem. This problem is that of a serial parity detector. So, what is this problem? You first look at the circuit; the circuit is having an input X and an output Z.

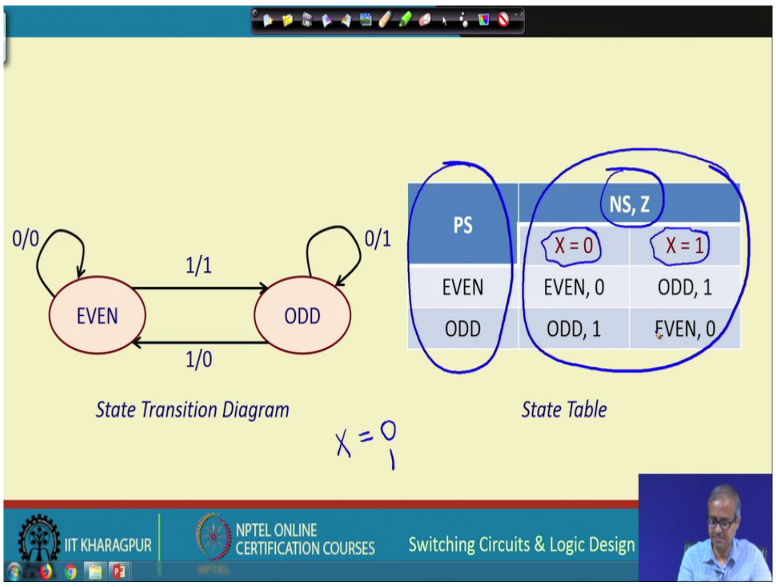
So, X is applied with a serial bit pattern let us say 0 0 1 1 1 0 1 0 0 1 something like this and output also will be generating a continuously bit pattern something like this. So, what will it indicate? This will indicate the parity of the input bit stream. So, if you just write down this bit stream one below that with (let us write it here let us say this is 0 0 1 1 1 0 1 0 0 1.

Now parity means with the number of 1's whether they are odd or even. If it is odd I will be outputting 1 if it is even I will be outputting 0, let us say that is our convention. So, we will be looking at the number of bits that you have seen so far and whatever is the parity the output is set accordingly. Let us start this is my input X. So, what will be my Z? Even 1 has come it is now odd parity.

So, again a 1 has come it is become even, well again a 1 has come it is become odd, 0 has come it remains odd, there is a 1 has come it is become even again, 0 again even again even 1 again odd, so this will be the output right. So, in this problem this X is the single input Z is the output of course there is a clock. If you think of this state what are the things you need to remember? You only need to remember one thing what was the parity till now. Was it odd or was it even? This you can represent by a single bit in 2 states let us name the 2 states as even and odd.

So, the corresponding state transition diagram will look like this even or let us say even is the initial state. So, if a 0 comes you remain in the even state, the output is 0. But now if a 1 comes you go to the odd state with an output of 1 and while in odd state if a 0 comes you remain odd output 1 and if a 1 comes you go back to even with the 0 right.

(Refer Slide Time: 12:12)


0/0
1/1
0/1
EVEN
ODD
1/0
State Transition Diagram
X = 0
1
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Design

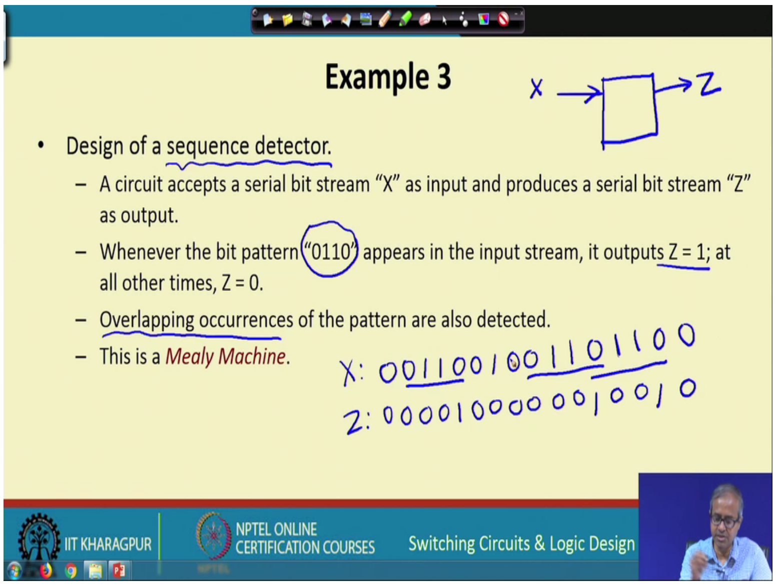
PS	NS, Z	
	X = 0	X = 1
EVEN	EVEN, 0	ODD, 1
ODD	ODD, 1	EVEN, 0

State Table

So, from here you can similarly construct the state table. Here you see this second part is little more complex, the first part as usual is the present state second part is the next state and the output; states there are 2 states even and odd.

Next state and output here there will be number of columns depending on all possible input combinations, because in this circuit there is a single input X and X can be either 0 or 1, there are 2 columns in the second part; one corresponding to X equal to 0 other corresponding to X equal to 1 and whatever is listed here is actually this NS and Z, next state and output. So, if you are in even let us say here if X is 0 then you remain in even and the output is 0; that means, this edge. If the input is 1 you go to odd with the output 1 it means this edge. So, like this for all the edges you can see the entries right, this actually quite easy to construct.

(Refer Slide Time: 13:28)



Example 3

- Design of a sequence detector.
  - A circuit accepts a serial bit stream "X" as input and produces a serial bit stream "Z" as output.
  - Whenever the bit pattern "0110" appears in the input stream, it outputs Z = 1; at all other times, Z = 0.
  - Overlapping occurrences of the pattern are also detected.
  - This is a *Mealy Machine*.

$X \rightarrow \boxed{} \rightarrow Z$

X: 0 0 1 1 0 0 1 0 0 1 1 0 1 1 0 0

Z: 0 0 0 0 1 0 0 0 0 0 0 1 0 0 1 0

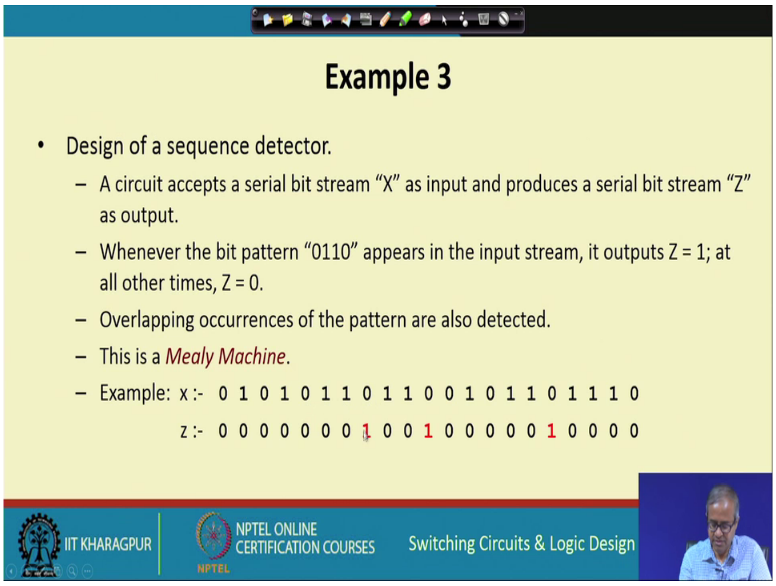
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Design

Now, let us look at a slightly more difficult problem. This problem is that of a sequence detector. So, what is a sequence detector? Well it is something similar to the previous problem in the sense that there is a single bit stream X as input and a bit stream Z as output, of course there is a clock I am not showing the clock. Now in X I am applying a bit stream. Now whenever this pattern 0 1 1 0 appears the output will be 1 otherwise it is 0.

So, let us assume take an example. Suppose my input is like this, so you see this 0 1 1 0 in what place it is appearing. See one place it is here one place it is here and there is a overlapping pattern here. So, I am assuming that overlapping patterns should also be considered this 0 1 1 0 before which finishes the second pattern starts.

So, in the output there will be a 1 out here there will be a 1 here there will be a 1 here, all the rest bits will be 0, this is this circuit that we want to design. So, this will be my X and this will be my Z right, this is a Mealy Machine because there is an input and the output will of course, also depend on the input and the state both right, so this is not a Moore Machine, this is an example of a Mealy Machine.

(Refer Slide Time: 15:31)



Example 3

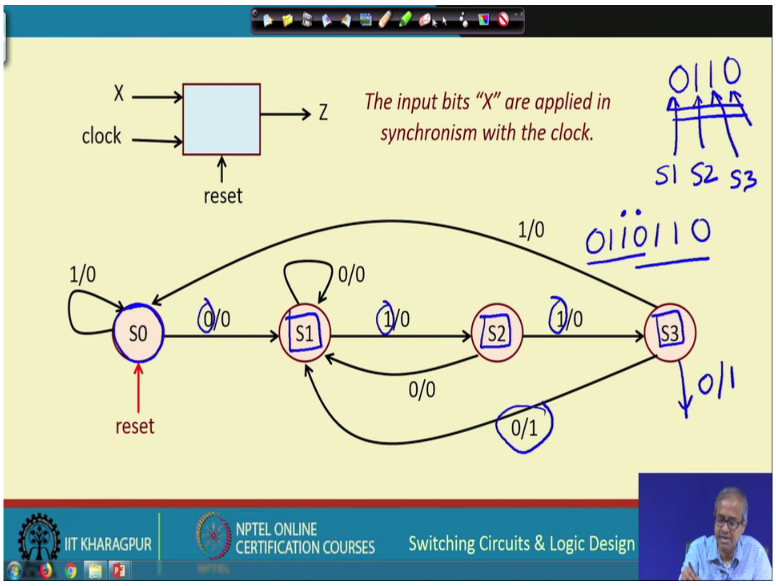
- Design of a sequence detector.
  - A circuit accepts a serial bit stream "X" as input and produces a serial bit stream "Z" as output.
  - Whenever the bit pattern "0110" appears in the input stream, it outputs Z = 1; at all other times, Z = 0.
  - Overlapping occurrences of the pattern are also detected.
  - This is a *Mealy Machine*.
  - Example: x :- 0 1 0 1 0 1 1 0 1 1 0 0 1 0 1 1 0 1 1 1 0

    z :- 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 1 0 0 0 0

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Design

So, here is another example you see 0 1 1 0, there is a 1 generated here 0 1 1 0, 1 generated here 0 1 1 0, 1 generated but here there is no overlap fine. Now let us see how we can construct the state transition diagram for such a circuit. Before I show it let us try to work out how it should be, let us see fine.

(Refer Slide Time: 16:13)



The input bits "X" are applied in synchronism with the clock.

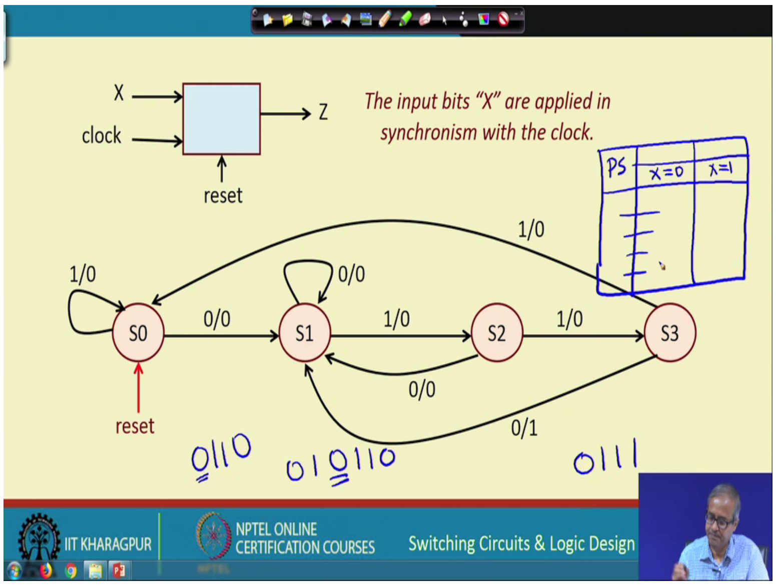
This is what the final solution is shown, let us try to understand why we have gone for this because, we want to detect the pattern 0 1 1 0 right. Well we do not know how many states will be required, say as we construct there more states are getting constructed. See the first thing is that we start with an initial state let us say S 0, 0 is my initial state. So, I look for the first 0, so whenever I get the first 0 I go to state S 1.

So, S 1 indicates that I have seen the first 0 this is my state S 1. So, while in S 1 if I see the next 1 here I go to state S 2, you see if I see a 1 I go to state S 2. Similarly if I see another 1, I go to state S 3 I see another 1 I go to state S 3 ok. Now if I see a final 0; that means, I have detected the pattern. So, from S 3 if I get a 0, so naturally I will have to output a 1 because I have got the pattern. Now the question is where should I move it back? Should I move it back to S 0 or somewhere else?

Now the point is that because we are allowing overlapping patterns, so let us consider an overlapping pattern like this, so this was 0 1 1 0 there is an overlapping 0 1 1 0. Now when you are in S 3 you are in S 3 means what, you are here; you are at this point right your S 3, now a 0 has come, now this 0 is also S 1 for the next pattern, so instead of moving it back to S 0 we are actually moving it back to S 1 because, you need also to check the next overlapping pattern and it may start here also.

So, you are actually in S 1 for the next pattern and for the other ones I think it is quite simple because, in S 0 if you get a 1 that means you are trying to detect 0 1 1 0, if you get a 1 in the beginning you remain in S 0.

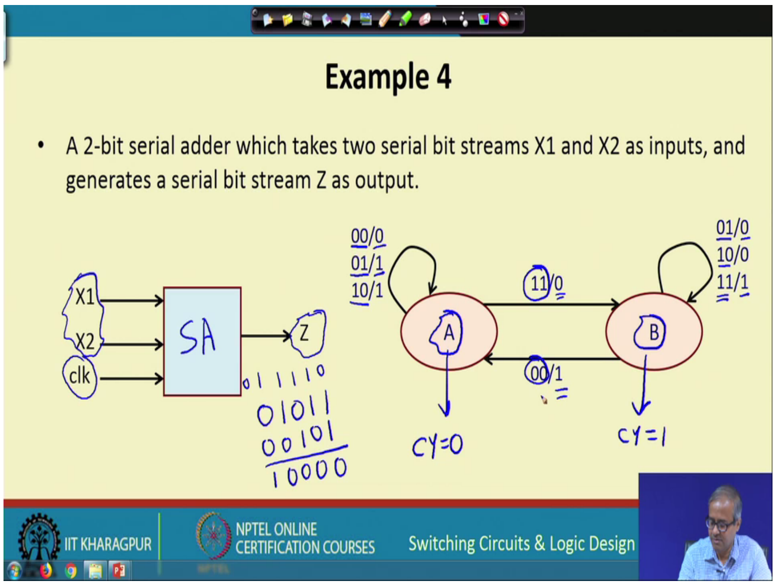
(Refer Slide Time: 18:53)



X
clock
Z
reset
The input bits "X" are applied in synchronism with the clock.
PS
X=0
X=1
1/0
1/0
0/0
0/0
1/0
1/0
0/0
0/1
S0
S1
S2
S3
reset
0110
010110
0111
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Design

Now in S 1 you have detected a 0, but if you get another 0 no problem you are still in the first 0 you remain in S 1, but in S 2 that means, you have got a 0 1 and again a 0 this can be the start of a 0 1 1 0, so you have to go back to S 1. Similarly, from S 3 to S 1 and in S 3 if you get a 1 which means it is a 0 1 1 1, so it is not the pattern you have to go back to S 0 and start from the beginning. So, you see drawing this diagram is not trivial you may have to think a little bit consider all possibilities and then come up with the diagram the edges fine. Now once you have drawn the edges the rest is fairly straight forward.

(Refer Slide Time: 19:46)



Example 4

- A 2-bit serial adder which takes two serial bit streams X1 and X2 as inputs, and generates a serial bit stream Z as output.

Other I am not shown in the state diagram there. Let us take another example state table I mean. Here in this example we consider see if you look at the previous example just a second. This was the example of a sequence detector just one thing let me tell you before you move on, so if you want to construct the state table, which I have not shown there will be 2 parts to it, the first part will be the present state there will be 4 states S 0 S 1 S 2 S 3 and in the next state because there is only 1 input so there will be one corresponding to X equal to 0 one corresponding to X equal to 1 ok. Just like that serial parity detector same kind of table will be there will be 4 rows here right.

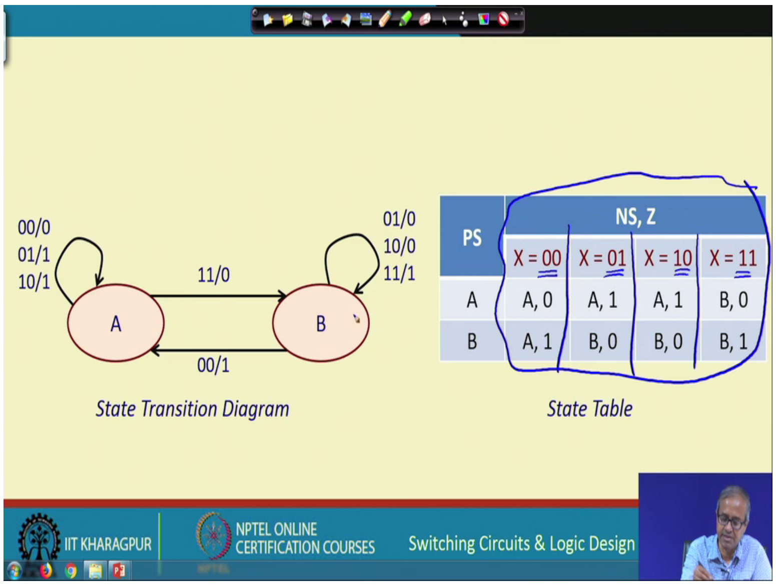
Now we consider an example which is a serial adder. Now I took this example earlier when we are discussing about the flip flops and the other simple applications of the flip flops. Now let us look into the design of the serial adder in a little more formal way. Now I mentioned earlier if you recall that a serial adder this is a serial adder is a circuit, which will be having 2 serial inputs X 1 and X 2 where the bits will be coming serially and this sum will be generated Z serially in synchronism with a clock.

So, when you add 2 numbers like 0 1 0 1 1 and let us say 0 0 1 0 1 1 1 0 with a carry of 1 0 with a carry of 1 0 with a carry of 1 0 with a carry of 1 1 with a carry of 0. Now initially carry is 0. So you see for a serial adder if you think what should be my states? For addition the only thing that I have to remember is the carry from the previous stage. So, that will be my state that will be my state the carry there will be one state variable, so there will be 2 values 0 and 1.

So, in this diagram I am representing them as 2 states A and B where a represents carry 0 and B represents carry 1. Now let us see how this state diagram works. So, when you are in state A and you have got an input 0 0, 0 0 and no carry so, what will happen? The sum will be 0 sum is Z sum is 0 and you remain in A means the carry remain 0. So, if you are applied 0 1 sum is 1 no carry remain in A, 1 0 same sum is 1 no carry, but when you get the inputs 1 1 then this sum will be 0 and there will be a carry of 1. So, now this state will change from A to B, B indicates carry 1 and while you are in state B the things are similar.

So, if you have input 0 1 with a carry 1 then the sum will be 0 and carry will remain 1, 1 0 also sum will be 0 carry 1 and if it is 1 1 sum will also be 1 carry will also be 1. But only when the inputs are 0 0 there will be no carry, so you move back to state A but this sum will be 1 ok. So, just you see and think how the process of addition goes on and you can get the you can get behavior of this serial adder in terms of the state transition diagram fine.

(Refer Slide Time: 24:25)

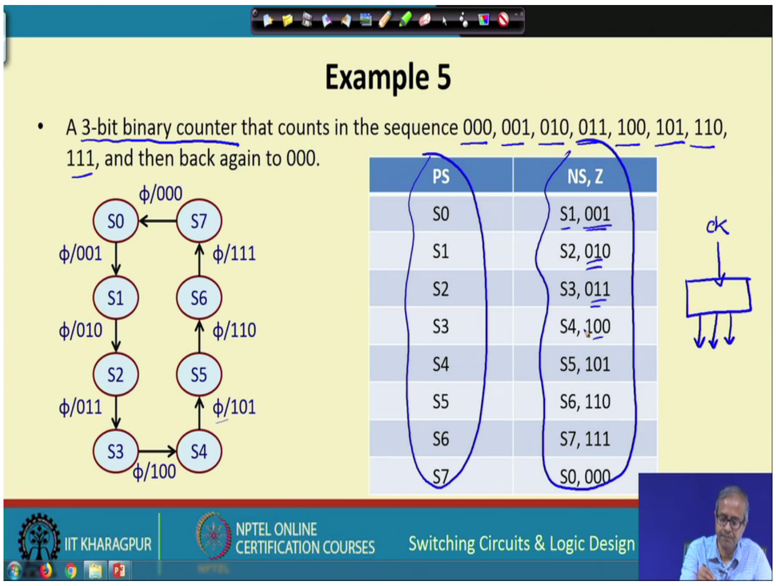


PS	NS, Z			
	X = 00	X = 01	X = 10	X = 11
A	A, 0	A, 1	A, 1	B, 0
B	A, 1	B, 0	B, 0	B, 1

State Table

Now, from this state transition diagram here you can construct the state table same thing. So here you see there are 2 inputs right X 1 X 2. So, in this case in the second part there will be 4 columns, 1 corresponding to the input combination 0 0 0 1 1 0 and 1 1 the rest is same. Exactly what this diagram shows? So, you are in state A, the input has come 0 0 you remain in state A and the output is 0 ok, like that. The same thing you can you capture it like this.

(Refer Slide Time: 25:12)



Example 5

- A 3-bit binary counter that counts in the sequence 000, 001, 010, 011, 100, 101, 110, 111, and then back again to 000.

PS	NS, Z
S0	S1, 001
S1	S2, 010
S2	S3, 011
S3	S4, 100
S4	S5, 101
S5	S6, 110
S6	S7, 111
S7	S0, 000

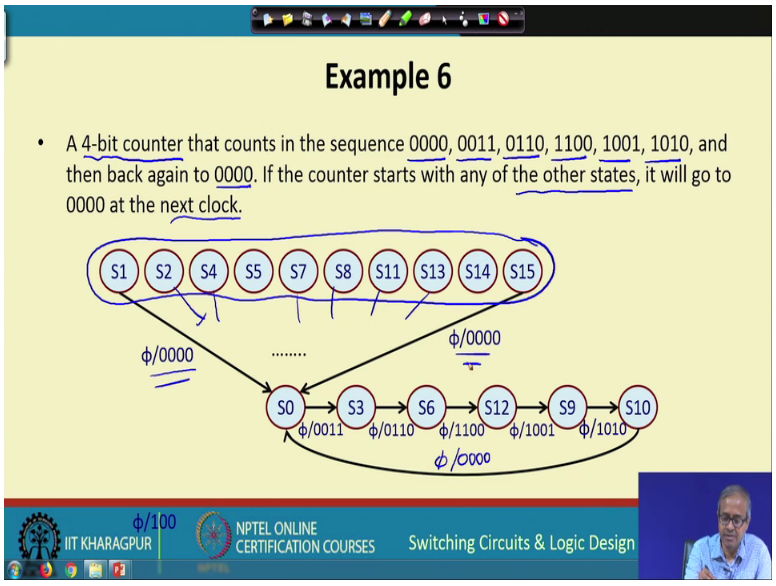
Let us take another example; this is a example of a counter. Well we shall be looking into the design of counter in more detail later, lot of different kinds of counters we shall see. But let us look into an example right here. This is a 3-bit binary counter. So what does this counter do? This counter is supposed to count in the sequence 0 0 0 0 0 1 0 1 0 3 4 5 6 7 and then back to 0. Now, the counter is a circuit like this, counter does not have any input externally there is only a clock and the outputs are generated in synchronism with the clock, the count value 0 0 0 0 0 1 sequentially they will be generated.

So, there is no separate input, so this state diagram state transition diagram is very simple, there will be 8 states I am calling them S 0 to S 7; I am just for convenience I am considering the binary equivalent, the decimal equivalent of the binary combinations and I am encoding states like this, for example S 6 means this 1 1 0 6 ok.

So, when you are in S 0 and a clock comes no input phi, you move to state 1 your output is 0 0 1; that means now you are in 1. From S 1 you move to S 2 0 1 0 2, S 2 to S 3 0 1 1, S 3 to S 4, similarly S 4 to S 5, S 5 to S 6, S 6 to S 7 and S 7 back to 0 0 0 0.

So, this state table logic is also quite similar. These are the present states there are 8 states and for each of these states there is no input, so there is a single column here. So, you show that what will the next state be, S 0 will go to S 1 S 1 will go to S 2 and what will be the corresponding outputs right, same information you are depicting fine.

(Refer Slide Time: 27:25)



Example 6

- A 4-bit counter that counts in the sequence 0000, 0011, 0110, 1100, 1001, 1010, and then back again to 0000. If the counter starts with any of the other states, it will go to 0000 at the next clock.

So, let us take one last example, this is a slightly more complex kind of a counter this is also a binary counter, but it does not count in the binary sequence. So, what it says is that this is a 4-bit counter that counts in the sequence first 0 0 0 0 then 0 0 1 1 then this then this then this, this. So there are 6 states and then back again to 0. So, you see I have just again used the decimal equivalence of the state S 0, S 3, S 6, S 12, S 9, S 12, S 10 and then back to S 0; this is my regular sequence of counting and this state transition is exactly labeled in this same way as in the previous example.

So, S 0 to S 3 it is labeled by 3 0 0 1 1 there are no inputs, S 3 to S 6 labeled by 6 0 1 1 0 and so on and S 10 back to S 0 this of course is missing, it should be 5 0 0 0 0 and it also mentions if the counter starts whenever you switch on the power it if it starts with any of the other states at the next clock it should go to 0 0 0 0.

So, if it is in any of the other 10 states out here, so all of them I have shown in a 2 that is dots, so there arrows from all of them. So, it will be going to S 0 right, so if you have a specification like this from there you can also construct the state table same way and from this state table you will be moving into the next step.

So, here we have looked at a number of examples for constructing the state table and the state diagrams. Now with this we come to the end of this lecture. Now from the next lecture what we will see? We shall look at some of the examples and try to go through the entire flow of synthesizing a synchronous sequential circuits. So, far we have only seen how starting from the behavior we can construct the state transition diagram and the state table. With the remaining steps we shall be learning in our next few lectures.

Thank you.