

Digital Circuits
Prof. Santanu Chattopadhyay
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 35
Finite State Machine

So, in our course, we will be next looking into another topic, which is Finite State Machine. So, to some extent, we have already seen some finite state machine as the name suggests. So, this finite state machine, it has got a number of states and the system, it transits through those states as time passes and the number of states that you have in the system is finite, it has got a fixed finite number of states and that is known a priori. So, you have to design a system that has got a number a known number of states and the transition between the states are also known and accordingly we have to design a circuit.

For example, if we have the talking about a vending machine ok; so, in the vending machine operation initially; so, there are some slots in to which the coins are to be put, after the coins have been put the buyer can select the item that he or she wants to buy and then if the money is sufficient, then the good has to be disposed. So, that goes in a sequence of events.

So whenever we have got this type of application they come under the broad heading of finite state machine so, will be looking into some of this finite state machine designs, in this part of the lecture.

(Refer Slide Time: 01:34)

Finite State Machine

- An electronic machine which has
 - external inputs
 - externally visible outputs
 - internal state
- Output and next state depend on
 - inputs
 - current state

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, a finite state machine is an electronic machine which has some external input externally visible outputs and some internal states. So, output and next state it depends on inputs and current state; so, output and next state. So, both will depend on the input value and currently whatever state the system is in.

(Refer Slide Time: 01:55)

Abstract Model of FSM

Machine is $M = (S, I, O, \delta)$ $S = \{A_1, A_2, A_3, A_4\}$

S: Finite set of states
I: Finite set of inputs
O: Finite set of outputs
 δ : State transition function

• Next state depends on present input *and* present state

Output $z^n(\lambda)$
 $\lambda(A_i, A_j) = O_k$

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, based on that it will depend; so, it is the abstract model of a finite state machine is like this. So, it is A 4 tuple; consider a machine M is A 4 tuple consisting of 4 states 4 quantities S I O and delta where S is a finite set of states ok, then the system at any point

of time, the machine at any point of time can be in any of those states. For example, if I say that my state set S is consisting of the state S_1 , S_2 , S_3 and S_4 ; what it means is that at any point of time, machine will be in one of these 4 states; S_1 , S_2 , S_3 and S_4 . So, it cannot be in any other state ok. So, they are all excluded.

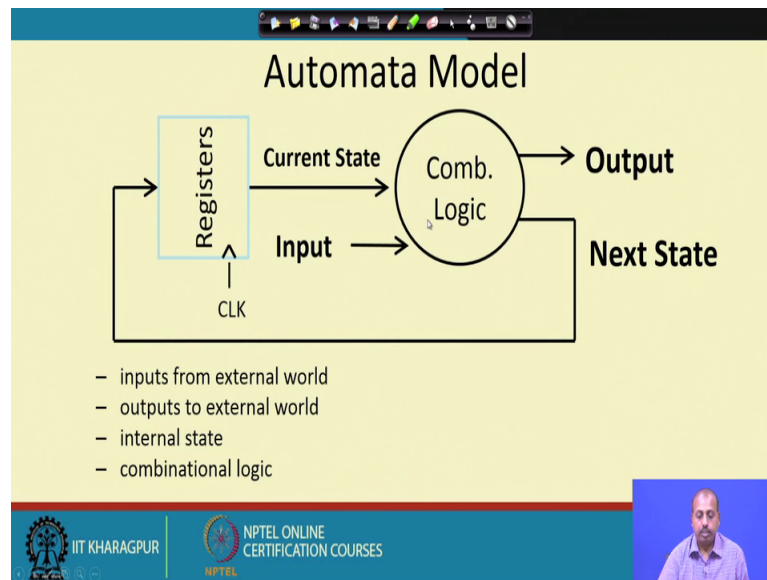
For example, if we if we if your machine is a counter and if you are thinking about a mod 16 counter, then this set S is the set of states 0 to 15. On the other hand, if it is a decayed counter, then the set of states S is a are the values. So, are the states 0 to 9. So, that way depending upon the system the machine that we are having this set of state will be determining the states in which the system or the machine can be at any point of time.

Then I is the finite set of inputs. So, it interacts the machine interacts with environment accordingly, it get some input from the outside and that is there is a finite set of input, then O is a finite set of output with respect to that vending machine example. So, when the coins are being put into the system or the user is giving a choice regarding the good that he or she wants to buy. So, they are coming as input to the system and when it is disposing the good by means of activating some switch. So, that is basically the output for that machine.

So, you have got this I and O and this δ . So, this is the state transition function. So, that defines how the state is transiting from one state to one state to the next state. So, next state depends on the present input and the present state. So, this state transition; so, this will be this determine by the present state and the next state of course, there is another function which is not written explicitly, which is the output function. So, output function; so, this is basically it is often represented by λ . So, this λ actually tells. So, λ over any state say S_i and some input combination say I_j . So, this will be telling us what is the output value say for all the outputs. So, it will tell you, it will tell me; what is the value of the output.

So, we will come it come to this slowly as you proceed ok.

(Refer Slide Time: 04:56)

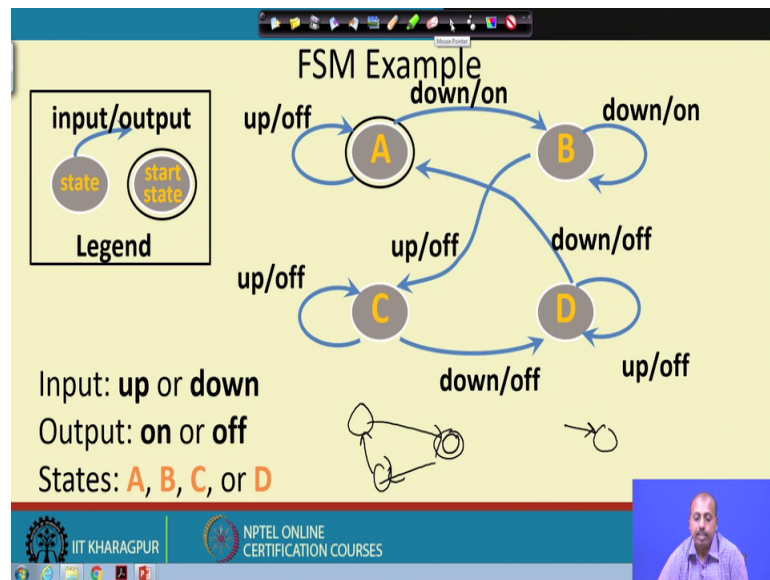


So, so, from the automata theory; so, you can visualize this structure, this machine like this as if we have got some register and this register is clocked and the this register holds the current state of the system. So, this current state comes as input to a combinational logic and the inputs that are there. So, that also come as a combinational logic and this combinational logic. So, it determines what should be the output and also what is the next state.

So, this next state values will be stored again in this register as the current state at the next clock pulse this next state values will be stored into this register as current state and they will be available as current state. So, this way the system will be go passing through different states and depending upon the input combination. So, it will produce some output also, it will make transitions.

Now, inputs are coming from the external world outputs are also given to the external world, the states are internal and the combinational logic is also there, which is determining these 2 functions.

(Refer Slide Time: 06:04)



So, let us take an example, suppose, we have got a system that has got a switch which may be either up or down and an output which may be either on or off. So, you can think about the electrical switch and a lamp. So, if the switch is in up position switch may be in either the up or down position similarly the lamp may be in on or off situation and the system has got 4 states A, B, C and D. So, when the system is in a state. So, the when the system is in a state the if you press the switch in the upward direction, then the the light should be off. So, it is I as long as is the switch is pressed in the is kept in the up position the light is kept off and it if the system remains in state A.

If you take this switch down ok, then it the light is turned on and if the system makes a transition to state B and as long as the switch is in down position ah. So, it will be remaining in state B and the light will be turned on, then if the switch is now put into the off up position, then it comes to state C and puts the light off and as long as in light as long as it is in state C, as long as you are putting this thing switch in the up position. So, it will remain in the off state and from if you put the switch in the down position from C, then the light remains off and similarly, at the D state also, if you put it in the up position the light remains off, if you put it in the down position, then also the light remains off.

So, once you are in state D ok, you see that in all the so, light is always kept off. So, whether you put the switch up or down, then the switch will light will remain off only when you are in going to state a then if you put the switch down, then it will be turned on

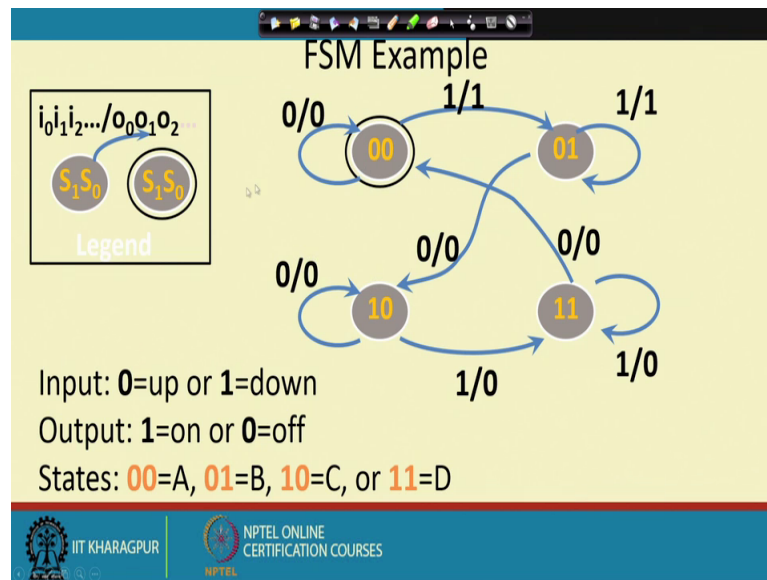
and then after that if your state B then as long as it is down it will be on. So, this is the hypothetical behavior of a system that has got is some input switch and some output which can be turned on or off.

And then this input. So, so this is the an example of finite state machine. So, it has got 4 states A, B, C, D. Now in these notations, there are some symbols ok. So, some we write it like this. So, this state; so, it is normally a state is given like this and if it is put within a it 2 concentric circles like here there is another circle which is surrounding this state name. So, that is the start state ok. So, any start state any arbitrary state transition diagram if you draw then you indicate the start state, if suppose, I have got some arbitrary state transition diagram like this to mean that this is the start states. So, I have to make it like this. So, if there are 2 circles, then that is a start state.

Or you it may be the other symbol is like other symbol sometimes used is like this ok. So, that is the start without any symbol coming without any labeling like that and whenever a transition is occurring. So, we label it in term like input slash output like. So, everywhere all this transition that you have; so, you see that the labeling is input with I first write down the input condition that will trigger that transition and then after the slash we write down what is the value of the output signals.

So, that you; so, in this case we have got a single output signal so that is that that is why we have got only the one output. So, if there are multiple output signals. So, they can be put here, similarly, if there is a complex input combination condition. So, that will be forming a Boolean expression that can be written as the left part of this before this slash. So, this way, we can represent a finite state machine by means of some diagram ok. So, mathematically, we can write it in terms of those delta and lambda functions in graphical notation, we can draw that diagram and then that also represents the finite state machine.

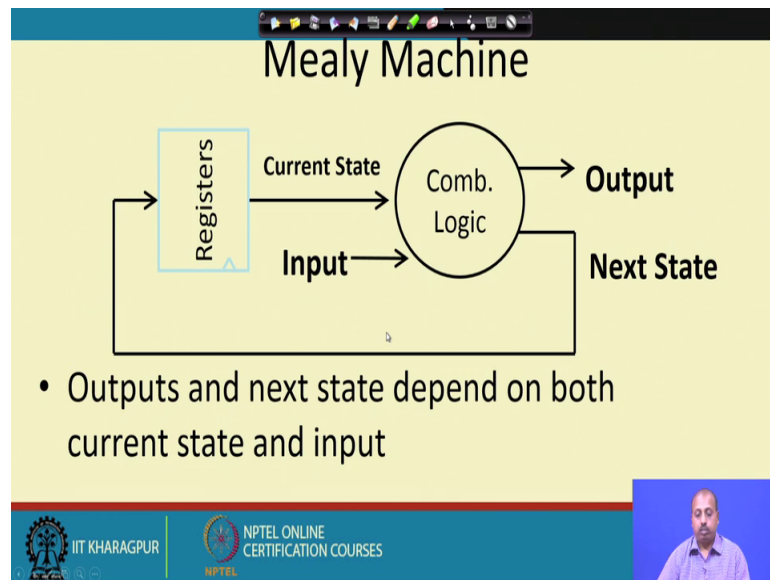
(Refer Slide Time: 10:27)



So, this is a slightly complex example where I have got this i_0, i_1, i_2 , there are 3 input more number of inputs, ok, i_0, i_1 in 2. So, we can have similarly as I was telling. So, there may be more number of outputs also. So, if this is the input combination. So, this should be the output combination that is the meaning of this labeling similarly for the states. So, instead of showing it by some name; so, you can also show them by means of the state the register content like here, I am writing here 0 0; that means, the state register contains 0 0 and when the state register contains 0 0 if the input is 0 output will be 0 and it will remain in the state 0 0.

Similarly, if the input is 1, then it will make a transition to the such that the state register contain becomes 0 1 and the output is 1. So, it will remain there. So, it goes on like this. So, here this as if this states A, B, C and D, they have being given a 2 bit code state, A is coded as 0 0, B as 0 1, C as 1 0 and D as 1 1. So, this is will come back to this coding again later so, as if these states have been given some name or some code.

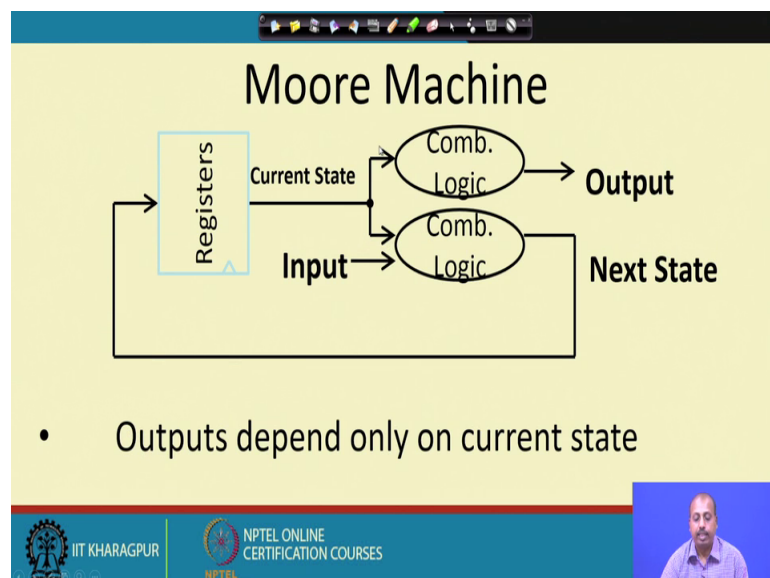
(Refer Slide Time: 11:45)



Now, the type of machine that we were looking so far so, they are known as mealy machine. So, in case of mealy machine what happens is that outputs and next state function they depend both of them depend on they depend on the current state as well as the input value. So, this output function and the next state function, they depend on the current state value and the input value. So, this they depend on both the value.

So, this is the standard mealy machine standard finite state machine that we have seen so far. So, they form the mealy machine category.

(Refer Slide Time: 12:25)

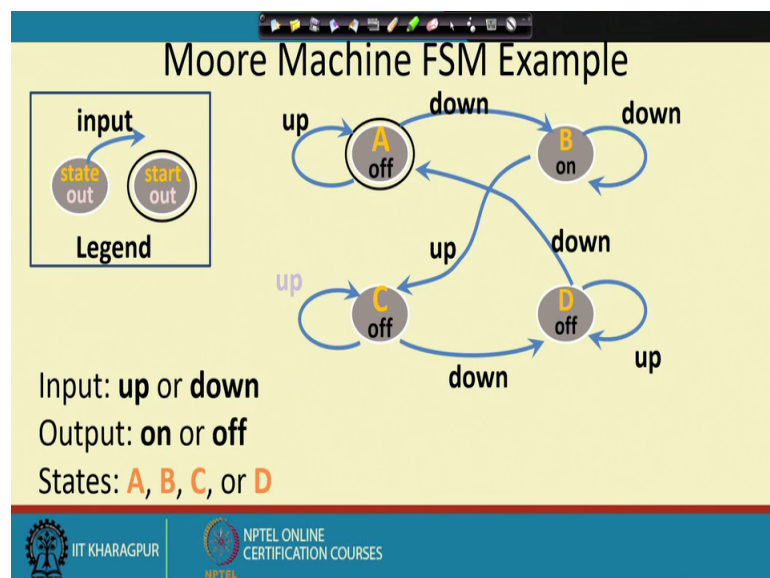


So, another type of machine that we can think about so, it is Moore machine. So, where the current the, this output depend outputs depend only on the current state. So, the, this input is not determining the current state input is determining only the next state part. So, the current state component and input; so, they are they are determining the next state part whereas, this output is determined only by the current state. So, input is not determining the output sorry the input is not determining the output; output is dependent only on the currents state for this type of configuration. So, it is said to be Moore machine.

So, apparently it seems that the mealy machine and Moore machine, they are 2 different types of machine. So, may be one of them is more powerful than the other, but that is not the case both the machines are equally powerful. So, whatever computation you want to represent using mealy machine. So, you can do it using Moore machine and vice versa, but the circuit wise of course, this since this Moore machine the output is dependent only on the current state, it may so happen that it has got more number of states because in case of mealy machine what will happen is from a particular state for different input combination.

So, you can have different outputs, but for Moore machine that is not possible. So, for all those different output combination; so, maybe we have to have separate states.

(Refer Slide Time: 13:56)



So, that makes Moore machine to have more number of states. So, here an example of this finite Moore machine based FSM. So, here you see that individual states ah. So, from the age level; so, we have dropped the output because output is not dependent on the input, it is dependent on the current state only. So, in state A, the output is always off at state B, it is always on, similarly, at state C and state D also they are always off. So, this an example of Moore machine.

Whereas for mealy machine. So, this is the standard example that we have previously seen. So, output is dependent on the current state as well as the input combination for example, if the current state is a input is up then only the output is off, whereas, the current state A, if the input is down then the output is on. So, that way output is dependent on the current state as well as input which is not the case for the Moore machine.

(Refer Slide Time: 14:58)

Create a Logic Circuit for a Serial Adder

- Add two infinite input bit streams
 - streams are sent with least-significant-bit (lsb) first

$$\begin{array}{r} \dots 101110 \\ \dots 01111 \\ \hline \end{array} \quad \xrightarrow{\quad + \quad} \quad \dots 00101$$

$$\begin{array}{c} \text{00/0} \\ \text{01/0/1} \end{array}$$

The slide includes the IIT Kharagpur and NPTEL Online Certification Courses logos at the bottom, and a small video inset of the presenter in the bottom right corner.

So, we can try to take some examples to see and understand how this finite state machines can be designed. So, it very the simple finite state machine that we can think about suppose we are adding 2 infinite input bits streams. So, this is some adder. So, what it does is that it there are 2 infinite bit stream, accordingly, it does this addition. So, it at first these 2 bits; 0 and 1 comes here as a result. So, these 2 bits are added the result is 1. So, it is outputting this one, then this one and one comes. So, sum is 0. So, it is outputting this 0 here and there is 1 carry. So, when the next 2 bits are coming. So, this 1

carry is also taken into consideration as well as a result. So, this bit becomes 1 and then this generates 1 carry. So, this carry is taken to the next bit addition and that is done by 0 that is result is 0.

But the point is that at any point of time it just adds 2 bits. So, it remembers what was the result of this previous addition whether it had generated a carry or not and based on that it is producing the next result. So, if we try to do this do realize this finite state machine. So, it is like this. So, if this is the initial state, then we can say that if the next input combination is say 0 0, then the output is 0 and it goes to a state where the carry was equal to 0, whereas, if the input was say 0 1 or 1 0 in both the cases, output is equal to 1 and it comes to a new state or we can do it like this we can make it slightly better actually we need to distinguish between the 2 cases in one case the carry is generated.

(Refer Slide Time: 17:01)

The slide is titled "Create a Logic Circuit for a Serial Adder". It contains the following text and diagrams:

- Add two infinite input bit streams
 - streams are sent with least-significant-bit (lsb) first

Below the text, there is a block diagram showing two input streams, "...10110" and "...01111", entering a rectangular block from the left. An arrow points from the block to the output "...00101".

Below the block diagram is a state transition diagram with two states represented by circles. The transitions are labeled with input pairs and output bits:

- From the top state to the top state: 00/0
- From the top state to the bottom state: 01, 10/1
- From the bottom state to the top state: 11/0
- From the bottom state to the bottom state: 11/1

The slide footer includes the IIT KHARAGPUR logo and the text "NPTEL ONLINE CERTIFICATION COURSES". A small video inset of a person is visible in the bottom right corner.

And in the other case, the carry is not generated. So, so, from this, I can say if it is 0 0, the output should be 0 and it goes to this state if it is say 0 1 or 1 0 if it is 0 1 or 1 0, in those cases, the output should be equal to 1 and it goes to that state ok.

So, in this state, you see that what we have got is the output there, there was no carry generated. Now if in from this state if the output if the input pattern was 1; 1 2 bits was 1 1, then the result should be 0 and it comes to a state where it remembers that a carry was generated, there was a carry now from this state if the next 2 bits are 0 0, then it has to

output a 1 because the carry was generated and we do not have any carry. So, we can say that if the next bit is 0 1 or 1 0 in that case, it has to output a 0 and it goes back to this state on the other hand, if this state is from this state if it is if the next 2 bits are again 1 1, then it has to output a 1 if the 2 bits are 1, then it has to output a 1 and it should go to a state where it has to generate a carry again.

So, in this way, we can complete this diagram and it will turn out to be a finite state machine by looking into all the combination. So, essentially from a particular state, you have to see the next 2 inputs and accordingly go to a go to the corresponding B. So, this design is slightly complex. So, lets us look into another simpler example for our understanding.

(Refer Slide Time: 19:04)

Create a Logic Circuit for a Serial Adder

- Add two infinite input bit streams
 - streams are sent with least-significant-bit (lsb) first

...10110 → [] → ...00101
...01111

- How many states are needed to represent FSM?
- Draw and Fill in FSM diagram

Strategy:

- (1) Draw a state diagram (e.g., Mealy Machine)
- (2) Write output and next-state tables
- (3) Encode states, inputs, and outputs as bits
- (4) Determine logic equations for next state and outputs

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, this is the exercise. So, I am leaving it for you. So, you just do it.

(Refer Slide Time: 19:12)

Another look at D latch/flip-flop

```
graph LR; 0((0)) -- D=0 --> 0; 0 -- D=1 --> 1((1)); 1 -- D=0 --> 0; 1 -- D=1 --> 1;
```

q_{old}	D	q_{new}
0	0	0
0	1	1
1	0	0
1	1	1

This is an example of a **state diagram**
more specifically a **Moore** machine

$q_{new} = D$

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, let us look into the D flip flop; how does it behave or D latch or D flip flop. So, in terms of finite state machine; so, you see that this is a Moore. So, if I say that at present the q output is 0, then if the D is equal to 0, then it will remain in these state and output will remain as 0 if D equal to 1, it will come to a state where the output is 1 and as long as D is equal to 1, it will remain in this state when D becomes equal to 0, it comes to this. So, this is the finite state machine that represents the behavior of D latch or D flip flop and this is the Moore machine because output is determined by the state ok.

So, this in state 0 output is 0 and in state one output is 1 and the, if you do a minimization you find that q new equal to D ok.

(Refer Slide Time: 20:07)

Another example - 2-bit counter

Counter starts at 0 (green) and increments each time the clock cycles, until it gets to 3 and then overflows back to 0.

Only input is the clock, we don't show that.

H_{old}	L_{old}	H_{new}	L_{new}
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, another example a 2 bit counter. So, the counter starts at 0 and at every clock pulse it increments by one. So, first it outputs 0 0, then at the next clock pulse it comes to 0 1, next clock pulse it comes to 1 0, then 1 1, then 0 0, again, this is a Moore machine because the output is determined by the input by the current state. So, this is 0 0 or 0 1, etcetera.

So, if you want to design the corresponding circuit, then what we have to do is we have to find the expression for H_{new} and L_{new} and accordingly you can design the circuit. So, this H_{new} and L_{new} .

(Refer Slide Time: 20:44)

2-bit counter

H_{old}	L_{old}	H_{new}	L_{new}
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

$$L_{new} = H_{old}'L_{old}' + H_{old}L_{old}' = L_{old}'$$

$$H_{new} = H_{old}'L_{old} + H_{old}L_{old}'$$

So, if you derive then H_{new} L_{new} is given by this expression which is L_{old} dash L_{old} bar and H_{new} is given by this expression. So, essentially what you have to do is that you have to take this H and L 2 flip flops and then we can do this minimization that is L_{new} is given by this L_{old} dash. So, this q bar line is connected here and it is done like this and this one your H_{new} is given by this expression H_{old} bar and L_{old} . So, this is and or gate based realization this is NAND gate based realization otherwise it is same.

So, this way, our finite state machine is this one and from there, we could come to a circuit which realizes the finite state machine.

(Refer Slide Time: 21:41)

2-bit counter with reset

R	H _{old}	L _{old}	H _{new}	L _{new}
0	0	0	0	1
0	0	1	1	0
0	1	0	1	1
0	1	1	0	0
1	x	x	0	0

$$L_{new} = R'H_{old}'L_{old}' + R'H_{old}L_{old}'$$

$$= R'L_{old}' = (R + L_{old})'$$

$$H_{new} = R'H_{old}'L_{old} + R'H_{old}L_{old}'$$

$$= R'(H_{old}'L_{old} + H_{old}L_{old}')$$

IIT KHARAGPUR
NPTEL ONLINE
CERTIFICATION COURSES

Now, if that counter, that we are talk talking about it has got a reset input like here this counter has got a reset input. So, here this if reset equals to 0, then it behaves in the normal fashion, but whenever this reset input is equal to one whatever be the state in which the counter is. So, it will come to this state 0 0 ok. So, it will be outputting 0 0. So, that is captured here. So, you see whenever R is equal to 1 whatever be the values of this H old and L old H new and L new will be 0.

Similarly, when R equal to 0 it behaves as a normal counter and it goes like this. So, L new is given by this expression and H new is given by this expression.

So, will output a 1 for each segment to be lit for a given state so, for 0; so, this is for 0. So, this will be output it will turn on this segment for one it will turn on these 2 segment for 2, it will turn on these segment. So, this way will go in a fashion like 0, 1, 2, 3, 0, 1, 2, 3 like that.

(Refer Slide Time: 23:23)

Counter with output functions

R	H ₀	L ₀	H _n	L _n	A	B	C	D	E	F	G
0	0	0	0	1	1	1	1	1	1	1	0
0	0	1	1	0	0	1	1	0	0	0	0
0	1	0	1	1	1	1	0	1	1	0	1
0	1	1	0	0	1	1	1	1	0	0	1
1	x	x	0	0	0	0	0	0	0	0	0

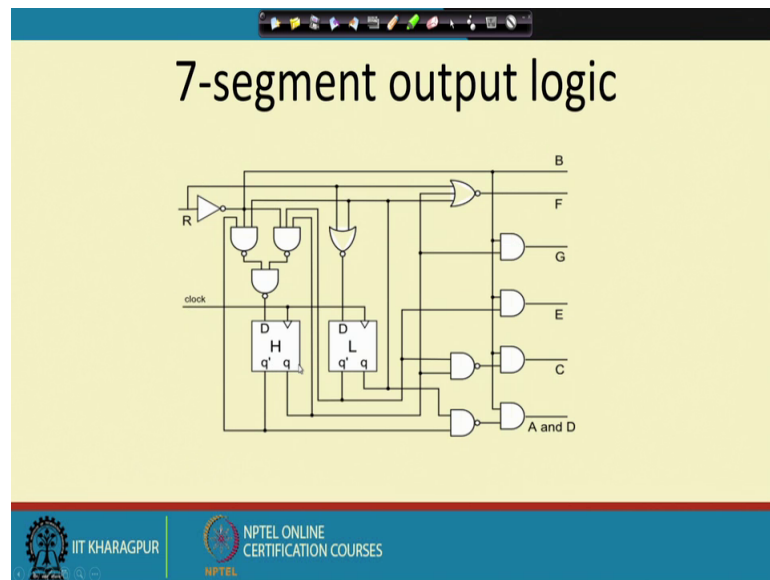
$A = D = R'H_0'L_0' + R'H_0L_0' + R'H_0L_0 = R'(H_0'L_0)'$ $B = R'$ $C = R'(H_0L_0)'$
 $E = R'L_0'$ $F = R'H_0'L_0' = (R+H_0+L_0)'$ $G = R'H_0$

So, from 0 0 from 0 0 so, it has to go to 0 1 from 0 1 it has to go. So, this is the state transition from 0 to 1, 1 to 2, 2 to 3, 3 to 0 and if this reset is 1, then it does not matter whatever be the state. So, it will come to the state 0 0.

Now, we have to see what is the pattern for this 7 segment display A, B, C, D, E, F, G. Now for displaying one, we know that for displaying this G segments for displaying this 0. So, only this g segment should be off and all other segment should be 1. So, that is done here. So, at the current state where 0 0 , it is turning on A, B, C, D, E, F and G is equal to 0, next at the state 0 1. So, this we have to display 1, accordingly, segments B and C, they should be turned on and the rest of the segments they should be turned off or put 0.

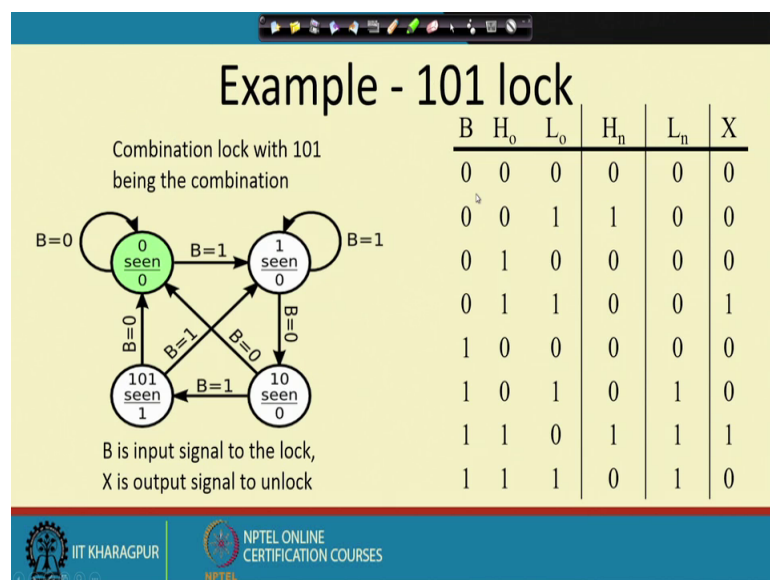
So, this way we can write down the truth table for this A, B, C, D, E, F, G and then do a minimization. So, A becomes A and D are their similar and this is the expression for a and D B equal to R dash C equal to this 1 and E equal to this. So, this way, you can have this all the expression for this seven segments ok.

(Refer Slide Time: 24:48)



And they can be fed to the individual logic. So, we can have this 7 segment display and this A, B, C, D, E, F, G, those lines are connected to the seven segments and this counters. So, once it is reset. So, this counter values will become 0 or from that point onwards. So, it can go on getting the individual values, it can go on getting the individual values.

(Refer Slide Time: 25:13)



Another example that we look into is 1 0 1 lock. So, this 1 0 1 lock is like this. So, we have got this whenever this pattern 1 0 1 is observed. So, the lock is set to be open. So,

the it starts in state where we have seen only A 0. Now B equal to 0. So, this is the start state. So, if I if I as long as we see B equal to 0. So, it is we have seen is 0, B equal to 1. So, it comes to a state, which is which we call one seen and the output is 0, then if B equal to 0, it comes to the state 1 0 seen and the output is 0 and B equal to 1, it is 1 0 0 1 seen and this output becomes equal to 1 again it has to look for this. So, 1 0 1; so, again it is one seen. So, it is comes to output is 0, it comes to 1 seen, then 1 0. So, like that this finite state machine is made.

So, accordingly, you can draw the corresponding truth table and get this signals realized.

(Refer Slide Time: 26:18)

101 combination lock



$$X = H_0 L_0$$

$$H_n = B' H_0' L_0 + B H_0 L_0'$$

$$L_n = B H_0' L_0 + B H_0 L_0' + B H_0 L_0$$

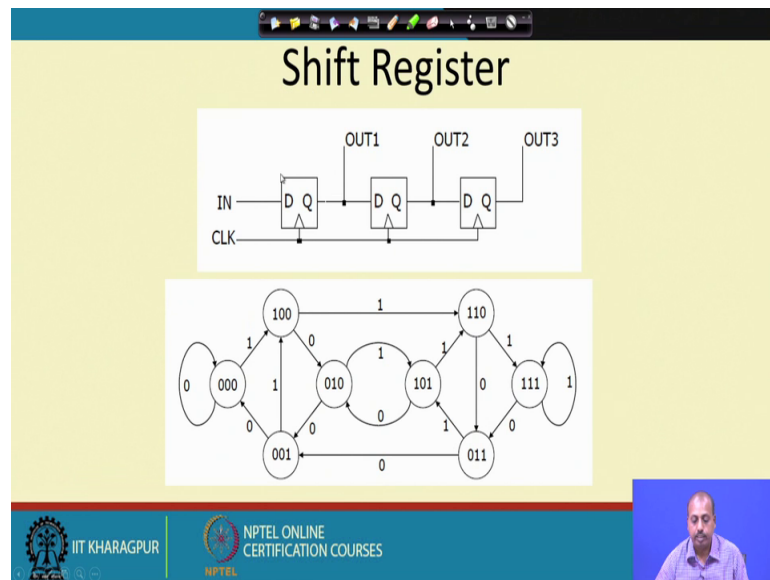
$$= B H_0' L_0 + B H_0 L_0 + B H_0 L_0' + B H_0 L_0$$

$$= B L_0 + B H_0$$

 IIT KHARAGPUR
  NPTEL ONLINE CERTIFICATION COURSES

So, this 1 0 1 combinational lock can be realized by this functions.

(Refer Slide Time: 26:26)



And then it can be converted into a, into some circuit for getting the thing.