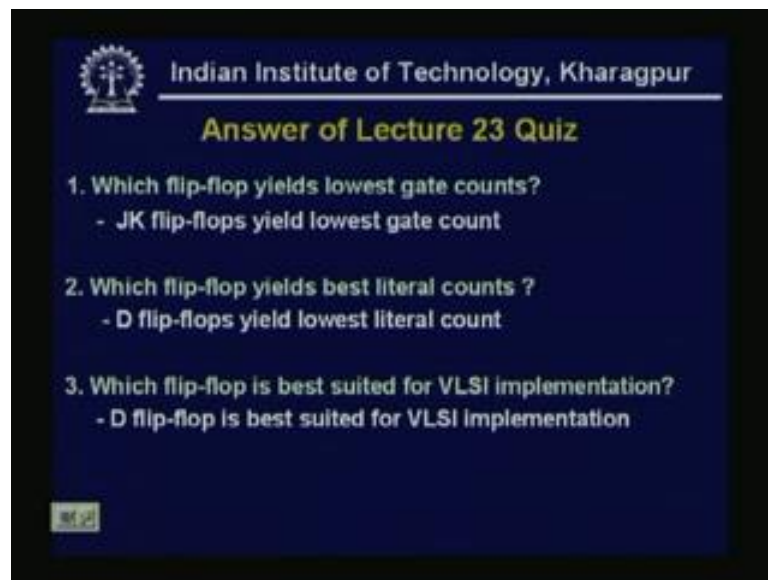


Digital System Design
Prof. D. Roychoudhury
Department of Computer Science & Engineering
Indian Institute of Technology, Kharagpur

Lecture - 24
Finite State Machine Design

In last two classes, we have read, how to design the synchronous sequential design. Now today we will read, the Finite State Machine Design. Again, that is a, sequential machine and synchronous sequential machine.

(Refer Slide Time: 01:12)



Before that, we quickly, see the answers of the lecture 23 quiz, now the questions are, which flip flop yields a lowest gate counts, and as already, we have read JK flip flops, D flip flops, RS flip flops, and we have seen, that JK flip flops, yield lowest gate count. Now, which flip flop yields best literal counts, the most popular D flip flops, yield lowest literal count, and, which flip flop is best suited for VLSI implementation. Again D flip flop is best suited for VLSI implementation, as already we have seen, that this is, very impact, and the simple to design.

(Refer Slide Time: 02:02)

Indian Institute of Technology, Kharagpur

Example: Odd Parity Checker

- Indicate whenever input bit stream has odd number of 1's

Present State	Input	Next State	Output
Even	0	Even	0
Even	1	Odd	0
Odd	0	Odd	1
Odd	1	Even	1

Symbolic State Transition Table

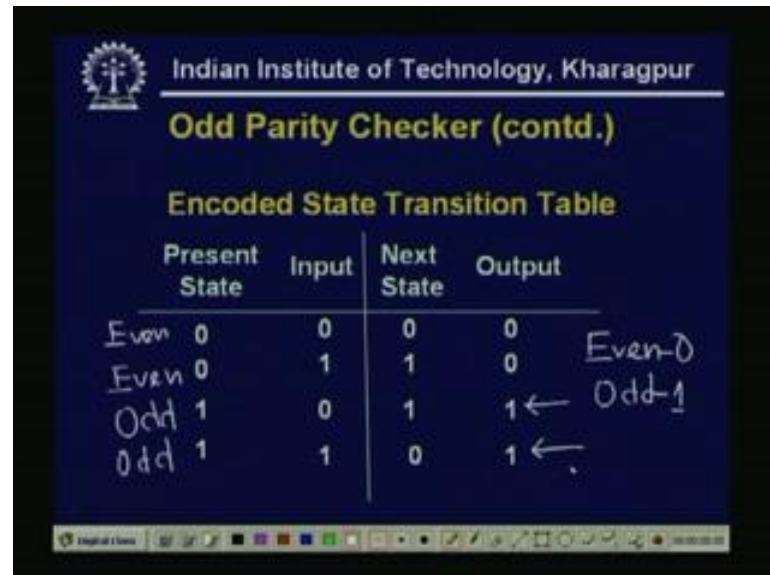
So, today we will start, the finite state machine design, we start with an example. Say example is, odd parity checker, all we know, parity means, say in a streams of 0's and 1's, if the total number of 1's is even, that means, number of x rows of all 1's become 0, then it is a even parity, and if the number of 1's are odd, then it is odd parity. Now, the parity checker, here we are seeing, that is a odd parity checker. So, indicate, whenever input bit stream, as odd number of points that means, it will generate a 1 output, whenever, odd number of 1's exist, in the input.

So, if we represent, the state transition table or state diagram, last day we have learned, then, if the present state is even, and the input as come as a 0, then the next state is also even, and the output is a 0. Now, if the present state is even, and 1 input 1's arrive, that means, next state will be odd. Because the earlier the present state was even, and now input has reached one. So, next state is odd, and the output is, if we, consider the present state, just the present output, this will be a 1.

And, if we 1, these 1 stored, and in the next clock cycle, we want to, generate this as a output, then actually in next clock cycle, this 1 will be give as a output. Similarly, that, if present state is odd, and 1 input 0 as come, obviously, next state is odd, because input 0 arrives. Then also 1 output is there, so this is again 1 output, and in the next clock cycle, it will be, taken as the output, so actually even, present state input 1, will generate, output, and odd present state, odd present state, input 0 that will be generating 1 output.

And, in the next clock cycle, that will be, shown as a output, that is why the outputs are shown here instead of in these places.

(Refer Slide Time: 05:38)



Indian Institute of Technology, Kharagpur

Odd Parity Checker (contd.)

Encoded State Transition Table

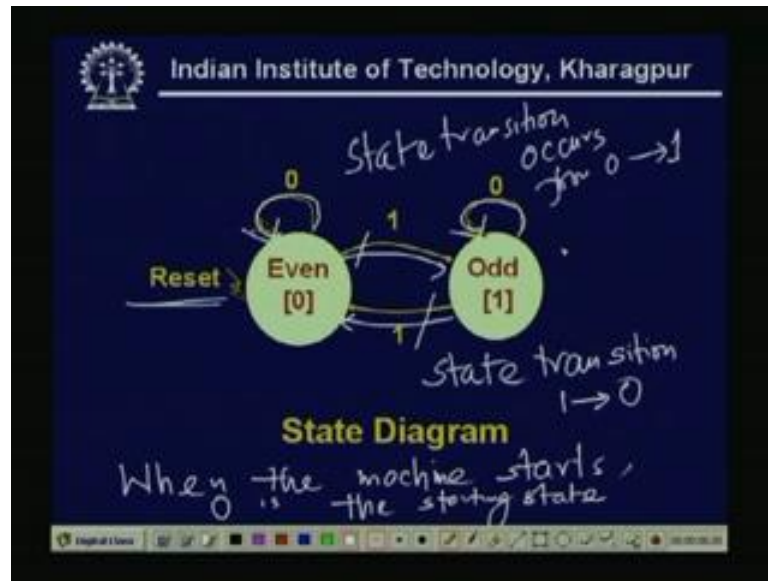
Present State	Input	Next State	Output
Even 0	0	0	0
Even 0	1	1	0
Odd 1	0	1	1 ←
Odd 1	1	0	1 ←

Handwritten notes on the right side of the table: "Even-0" next to the first two rows, and "Odd-1" next to the last two rows.

So, the state transition table, means that with respect to the 0 1, 0 and 1, now if we denote the even state, as 0, that mean, here there can be two states, even and odd. So, I need only 1 bit, to represent the state, and that even means, I telling 0, and odd means I giving 1, so this is called the state encoding. So, already we have seen, that actually this is, even. So, if the present state is even, and input is 0, next it is 0, output is 0, if the present state is even, that means 0, input is 1, it give next state is 1.

Because, next state is a odd state, odd is 1, and output is 1, that is show in the next clock cycle, this is a next clock cycle. Similarly, the odd state is encoded as 1, so present state is 1 odd input 0 then the next state is 1, and again 1 output, next clock cycle it will be, shown in the output line. Then, again odd present state input 1 next state is 0, output is 0, so this is the state and transition table, with the state encoded as 0 and 1.

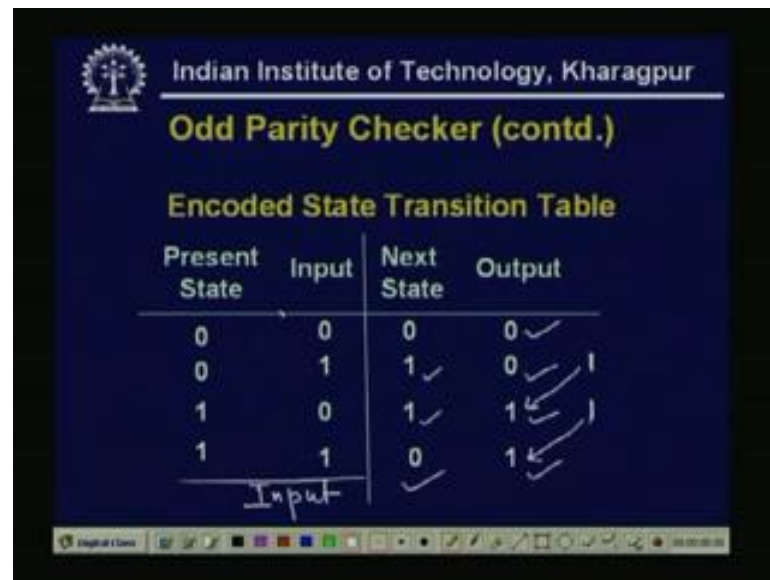
(Refer Slide Time: 07:25)



So, if we represent, that as a state diagram, again this, there are two states, so this is my even state mean 0, this is my odd state, that is encoded as 1. Now, that means, reset means, always when it start, when the machine starts, the starting state will be in 0, 0 is the starting state or initial state. Now, if a, if a, 0 input arrives, so this is a 0, and the in the next state will be even, so it lies, in the same state. Now, if a, 1 input arrives, then even state input is 1, it will be shifted to the odd state, so there will be a transition.

So, with one input, having even state, there will be a state transition, from even to odd, so this is a state transition, occurs from 0 to 1 state. Now, similarly, if the present state is odd, means the 1, encoded as 1, and 1, 0 input as come, then again it will, because it will be in a same state, but output will 1. And, it is a present state is odd, and again one input arrives, then it will come back to the even state. And that means, there will be a state transition, and that is a, state transition from 1 to 0, 1 to 0. See this is the state diagram of the FSM, the final state machine.

(Refer Slide Time: 10:45)



Indian Institute of Technology, Kharagpur

Odd Parity Checker (contd.)

Encoded State Transition Table

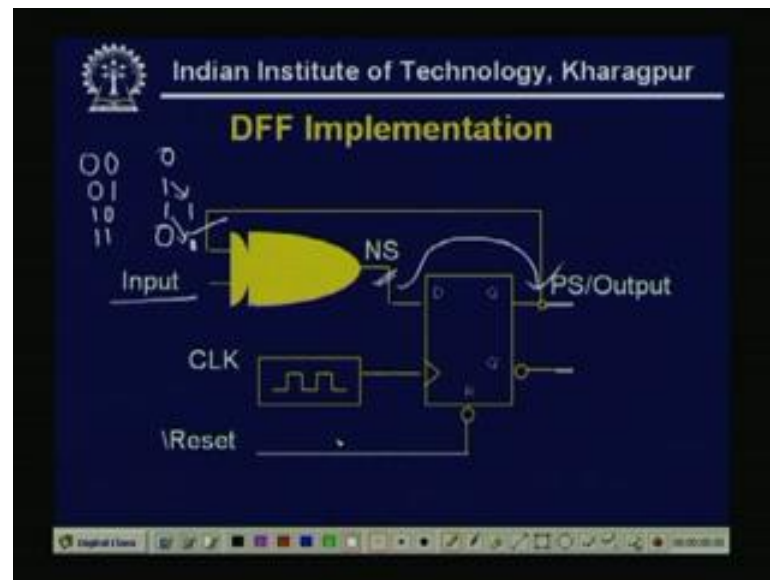
Present State	Input	Next State	Output
0	0	0	0 ✓
0	1	1 ✓	0 ✓
1	0	1 ✓	1 ✓
1	1	0 ✓	1 ✓

Input

Now, if the machine, just now we have design, if it is, implemented by a flip flop. So, what we have seen, that if we, are see the state transition table, we see that, if the present state, and say, these are my, if these are my inputs. Say, if these are my present state and input, are taken as the input of the machines, then, these 0 0 is output is 0, then 0 1. Again that next state is 1, and output is 0, 1 0 the next state is 1, and output is 1, 1 1 is next state is 0, and output is 1.

As, I mentioned earlier, that actually if we consider the present output, here it will be 1, and here it will be 1, but we are taking the output, after 1 clock cycle, that, that is, here after 1 clock cycle, this is shifted like this. And, that is why, after 1 clock cycle, the output, is being shown, at the output table. Now if we see the next state, see 0 0 0, 0 1 1, 1 0 1, 1 1 1 0, so this is nothing but, the truth table of 2 input XOR.

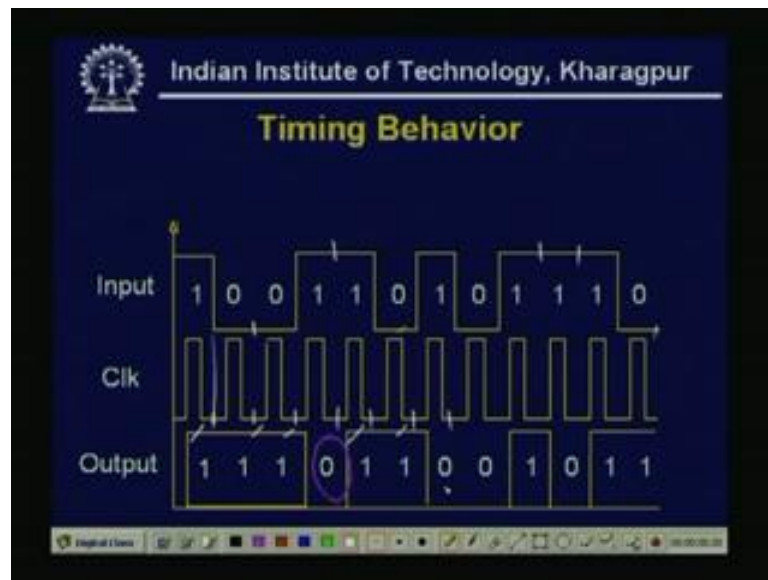
(Refer Slide Time: 12:10)



So, now this is my input, and this my present state, so this is my, this input is 1, and this present state, that has come here, so these are the 2 input of the XOR, and this is my next state. So, it will be, the truth table will be, 0 0 0, 0 1 1, 1 0 1 and 1 1 will generate a 0 again. Now another thing, say here the what is the function of this D flip flop, see, that this next state, that next state is 0 1 1 and 1 0 1, as it is a D flip flop. We know, that when the clock is high, the D input, will pass to the Q output.

So, after one clock pulse, these will be my output, so these will be the, output will be shown here. And this is the function of the... that means, after 1 clock pulse, the outputs will be passed, from here to here, and this is the design of the odd parity checker.

(Refer Slide Time: 13:42)



Now, if we see the timing behavior, so input is, say I am taking that high, clock pulse means 1, low is 0, so if it is a, bits stream 1 say 0 0 again 1 1 0 1 0 1 1 1 0 like that. So, that means, the clock is 1, 1 full clock pulse it is 1, so this is, up to this, it is 1. Similarly, here it is 0 0, 1 1, 0 here it is 1 like that. Now, when the clock is 1, see here input is 1, that means, odd number of points, so output will be, a high 1. Similarly, now next 0 as come, again 1 0, means then also it is a odd parity means, odd number of, once in the input bit stream, so output is 1.

Again, next bit arrives or next input bit is 0, so 1 0 0 again it is odd, so again the parity is, parity bit is 1, now 1, 1 arrives in the input bit, so there are even number of 1's, now there are two 1's. So, it becomes the odd parity checker, will generate a 0 output. So here it will be a, 0 output. Now similarly, again, if another 1 comes, then three 1's, so it will be a, odd parity generator will give 1 output, next 0 again 1 output, next 1 means, it becomes 1.

So, what we are seeing the timing behavior that with the clock pulses, that if odd number of 1's are counted, then the output generated, will be 1. And, if when, it become even number of 1's, in the input bit stream, then 1 0 output, will be generated.

(Refer Slide Time: 16:11)

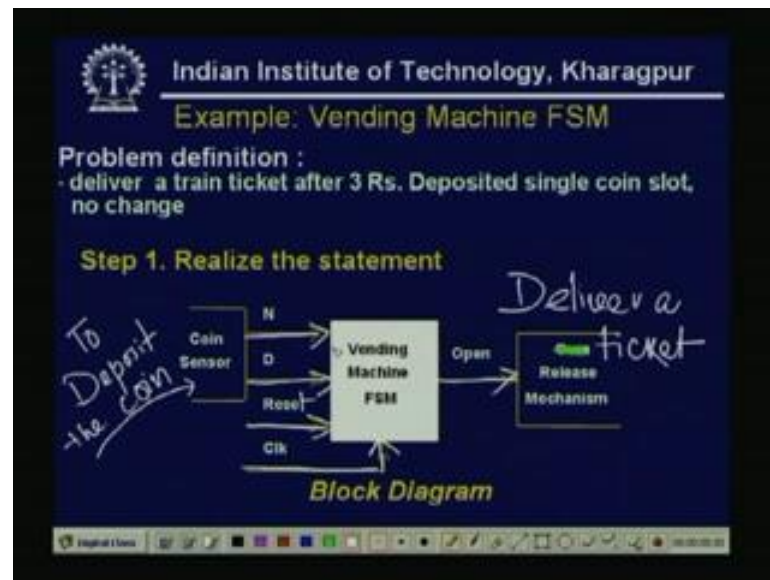


So, what will be the basic design approach, first we, we should realize the statement of specification, that means, problem statement should be specified clearly, that what machine or, for what input, what output will be generated. Then develop an abstract specification of the finite state machine. Here abstract specification means, that if we know, that machine last day we have defined, that the output or the next step depends, not only on the present input, but also in the previous state, or the previous state and the time.

So, we will be, specifying depending on the problem statement, that what will be the present state, work what will be the next step, and what is the status of the output then, then minimize the number of states. Because, always our aim is to design a compact machine, or compact circuit, or minimized circuit. So, if the number of states, n number of states are needed, then at least we need, $\log n$ number of flip flops. For this particular example we have seen, that for this odd parity checker, but we have two states, even and odd.

Even and odd states, so there are two states, so I need only 1 bit, to encode the states, 1 bit 0 or 1. Now, perform state assignment, and then chose, flip flop types to implement FSM state register. That means, we have read different type of flip flops, D T, RS flip flop, JK flip flop. Then which flip flop, we have to chose to design that particular FSM, and then implement the FSM. So, these are the basic design flow, of a FSM design.

(Refer Slide Time: 18:38)

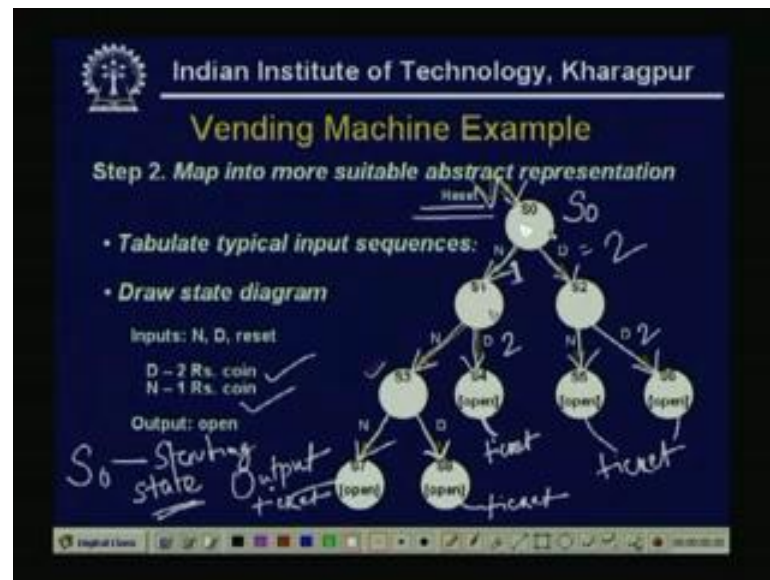


Now, we take, one very popular, and very common, example of the machine design, and that is called a vending machine FSM. Now, the problem first, the first step is the problem specification. So, problem is, that we have to construct, or we have to design a circuit, so that, that deliver a train ticket, the circuit will do the this function, that delivers a train ticket, after rupees 3 is deposited, as single coin slot, and no change. That means, if we deposit, 3 rupees by some coin, it will take the machine take only coins, then it will deliver it will produce a ticket, otherwise not, and there will be no change.

So, step 1 was that realize the statement, then, this is a block diagram level of the circuit. So see I have the, this is my FSM, that to be designed, and see this is a coin sensor, means here I am, putting or depositing the coin, to deposit the coin. See, there are two, type of coins, two inputs I have taken, that N and D, and this is a reset line, the machine reset line. Machine has another input clock, as already we have seen, when we have, learned the sequential machine design.

That, all the flip flops have to two lines one input is the clock, another the reset line, that means, where to start, for on which signal, it should start. Then open means, it generates a output, that means, it will deliver a ticket. So, it will, release or say deliver a ticket, and this is the overall block diagram that means, with the input output, and the machine itself.

(Refer Slide Time: 21:57)



Now, we try to, draw the state diagram, or the abstractive presentation. So, what will be the, typical input sequences. See, I will start a, initial state or the starting state S_0 , this is my S_0 . Now, say I have two type of input I have considered, one is called N, one is D. Now for these particular example, say D means, 2 rupees coin, and N means 1 rupee coin, we have not considered any 50 paisa coin, or 20 paisa coin, only two input we have taken the 2 rupees.

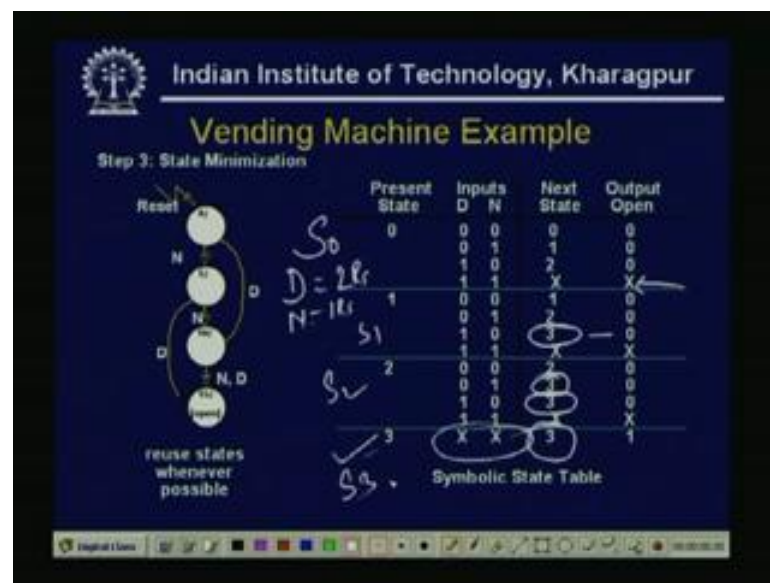
So, if we now, I deposit a 1 rupees coin, say it will process, because still, or, or it, it forwards the step, because there are chances, that another 2 rupees coin, may be deposited, or another 2 number of 1 rupees coin, can be deposited. So, it proceeds further, that means, a state transition occurs, from S 0 to S 1, now as already 1, N means 1 rupees coin, 1 has come. Now, there can be a 2 rupee coin, then 1 plus 2, 3 rupee coin. So, it will generate a output, that means, ticket will be delivered.

So, this opens means, that ticket is delivered. Now, from S 1, say 1 rupee coin is deposited, now if another 1 rupee is deposited, then it is 1 plus 1, and it goes to, state S 3. Now, if it is again 1, then actually 1 plus 1 plus 1 means 3, so again it will generate a output means, a ticket will be delivered. Now see, state S 3, the S 3 state means, already 2 rupees are deposited. Now, some one can put, another 2 rupee, then it is 2 plus 2 means four rupees, which is greater than 3, so again 1 ticket will be delivered means 1 output.

Similarly, if, in from the initial state, that mean, first someone puts, 2 rupees instead of 1, then it can be, 2, two other state, either someone can put another 2 rupees, then it is 2 plus 2, 4 which is greater than 3. And, someone can put 1 rupee, then it is 2 plus 1 means 3, so both the cases, both the cases a ticket will be delivered, in this case, so, in this case there are 5 situations, or 5 cases can occur, where this ticket can be delivered. These are the different way of depositing the coins, so that, I can get a train ticket, by 3 rupees. So, these are the abstract representation, of my problem.

Now, see. I start with these state S 0, so this is my, again this is after resetting, every time the machine, should start from this S 0 state, this is called the initial state or the starting state. So, every machine, as some starting state or initial state, where resetting, after resetting the machine will reach. So S 0, S 1, S 2, S 3, S 4, S 5, S 6, S 7, S 8, so there can be 9 states, here.

(Refer Slide Time: 26:52)



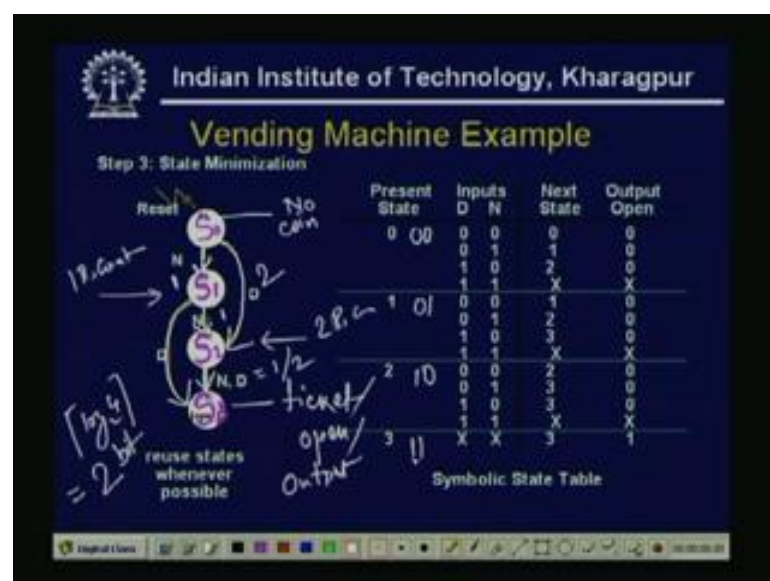
Now, first we see, the state table, so first is the what are the present state. So present states can be, that 0, 1, 2, 3 that means, S 0, S 1, S 2, S 0, S 1, S 2, S3 see, S 0, S 1, S 2, and S 3, only these are the intermediate states, from where the input, with 1 input, that 1 ticket can be delivered, and all are output state. So, if it is S 0 state, already we have seen, I can, I can give you 0 0 means, nothing as, been deposited, obviously, next state is 0, output is no output 0.

Similarly, if it is 0 mean, S 0 state, so you pick this is my, actually S 0 state, and this is that, if D is no mean, D means 1, D means 2 rupees coin, and N means 1 rupees coin. So, 1 1 rupee can be deposited, then the next state will be 1, because it, it state transitions, occurs from a 0 to S 1, output is 1, because I did not get, 3 rupees. Now, 2 rupees can be deposited, then the next state will be 2, and the output will be, again 0. Now, 1 1 means, in the S 0, 1 2 rupees, and 1 1 rupee can be deposited, but I have taken, not parallely, I have not taken the 2 inputs can be feed.

So, that why, it is, next state is taken as anyone, and any output state and the output is actually, output should be 1, and that should be, taken as S cross. Similarly, if it is now S 1, say again, already it is 1. So 0 0 means, next state remains in 1, output 0, if it is 0 1, that means, I have given 1 1, 1 rupees coin next state becomes 2, output is 0, then 1, 1 0 coin, now I got already 1 was there, and 2 rupees coin is there. So, 3, so this means, I got, I got a next state 3, so this is generate output 1.

Now, when a similarly, when it is S 2 means, 1 2 rupee coin is there, so 1 1 and 1 2 rupee both will give a output, and when, it is in 3 rupee, obviously, nothing any anything can be, already I have got 3 rupees coin. So, this is S 2, S 3 means, this is 3 rupees coin. So, always the output will be 1, now again by, simple state minimization, we can do that thing.

(Refer Slide Time: 31:17)



See earlier, earlier ((refer Time: 31: 02)) the initial we have, kept nine steps, S 0, S 1, S 2, S 3, and these are actually output states, so now it is minimized. So, if it is, if it is a, S 0 state means, there are no coin deposited, this is, that no coin deposited. Now say 1, 1 rupee has come, 1 1 rupee means, it will go to S 1. Now, if it is, 1 2 rupee has come, then also, so this is 1 rupee as come, this is 2 rupee as come, then also, it will be, it proceeds and it go to S 2.

Now, see, if the S 1 state, that mean S 1 means, S 1 means, 1 rupee coin is there deposited, 1 rupees coin. Now if someone puts 2 rupees coin, that means it is 3 rupee and it should give a, ticket. So, it will produce a ticket, that is the open, or that is the output, whatever we can tell. Now if here S 1, 1, another 1 has come, it will process it, if here it is a, it is a 2 rupees coin, 2 rupees coin, then anyone will also reach, S 3 or any 2, that mean 1 or 2, any input will make the 3 rupees deposited.

If it is one 3 rupees deposited, if it is 2, then already it was 2, now 2, so 4 rupees deposited, so in each of the cases, 1 ticket will be delivered. Now see, that earlier, when we have, initially from the problem statement, that state transition we have done, we used 9 states. Now, these 9 states, have been reduced to 4 state only, and these 4 state, now if we do the state encoding, we need 4, that means, log. So, ceiling of log 4 means, 2 bits are needed to encode, so first state encoding we need 2 bits. So, this 0, 1, 2, 3, that I can write, 0 0, 0 1, 1 0, 1 1 this is called the state encoding.

(Refer Slide Time: 35:14)

Indian Institute of Technology, Kharagpur

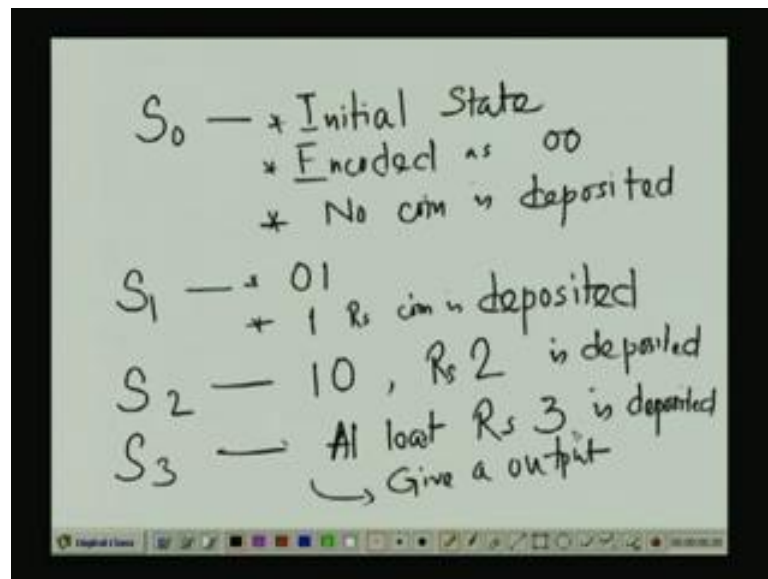
Present State	Inputs		Next State		Output
	D_1	D_2	D_1	D_2	
S_0	0	0	0	0	0
	0	1	0	1	0
	1	0	1	0	0
	1	1	X	X	X
S_1	0	0	0	1	0
	0	1	1	0	0
	1	0	1	1	0
	1	1	X	X	X
S_2	0	0	1	0	0
	0	1	1	1	0
	1	0	1	1	0
	1	1	X	X	X
S_3	0	0	1	1	1
	0	1	1	1	1
	1	0	1	1	1
	1	1	X	X	X

Step 4: State Encoding

$D = 2$
 $N = 16$


So, now we do the state encoding table, see, just now what I mention, that the present states are 0 0, then the inputs, that 1 rupee coin or 2 rupee coin, D means 2 rupees coin, and N means 1 rupee coin. So, if it is present state is 0 0, then if, no coin as been deposited, again it is, it remains in the same, output is 0, if it is 0 1 means, only 1 rupee coin is there. Next state is 0 1 means, so this is my S 0, 0 1 mean this is S 1, this is S 2 and this is S 3, so it goes to S 1, S 1 mean, only 1 rupees coin state.

(Refer Slide Time: 36:30)



So, if we write the physical interpretation of the states, what we can tell, that S 0 is first is, it is the initial state or starting state. So, each time after resetting, it will start from S 0, it is encoded as, encoded as 0 0, the physical meaning, what is the meaning that, no coin is deposited. Similarly, if it is S 1 means, encoded as 0 1, and here, 1 rupees coin is deposited. Now, if it is S 2, then 1 0, rupees 2 is deposited as coin, and S 3 means, either 3 or 4 means, at least rupees 3 is deposited. So, it will give a output, means 1, ticket is output means, 1 ticket is deliver.

(Refer Slide Time: 38:38)

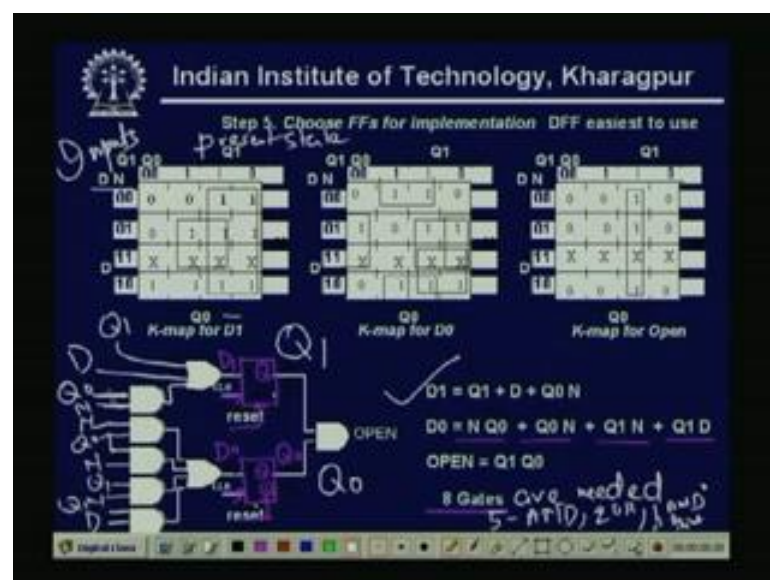
 Indian Institute of Technology, Kharagpur

	Present State		Inputs		Next State		Output
	Q_1	Q_0	D	N	Q_1	Q_0	
S_0 00	0	0	0	0	0	0	0
	0	1	0	1	0	1	0
	1	0	1	0	1	0	0
	1	1	1	1	X	X	X
S_1 01	0	1	0	0	0	1	0
	0	1	0	1	1	0	0
	1	0	1	0	1	1	0
	1	1	1	1	X	X	X
S_2 10	1	0	0	0	1	0	0
	1	0	0	1	1	1	0
	1	1	1	0	1	1	0
	1	1	1	1	X	X	X
S_3 11	1	1	0	0	1	1	1
	1	1	0	1	1	1	1
	1	1	1	0	1	1	1
	1	1	1	1	X	X	X

Step 4: State Encoding

So, if we see, this is my 0 0, that is my S_0 , 0 1, that is my S_1 , 1 0, that is my S_2 , and 1 1, that is my S_3 . Now, these 4 states are there, and then the all possible inputs we have considered, with every initial state. Or if the machine lies in that particular state, then if all possible inputs, or in any of the inputs come, then what will be the next state, and what will be the output, that we have considered, in the stating coding table.

(Refer Slide Time: 39:38)



See, now again it becomes, as a truth table, and which has, some 3 inputs as if, means here the output depends on the present state, as well as the present input. So, see, that

here D and N, that are the input means, that what coins are deposited, and here Q 1 is the state, of the present state, again Q 1, Q 0, that is also present state. So, if we draw the Karnaugh map, that means, D N and Q 0, Q 1 are the, as if they are treated as the inputs, and it will generate a single output, then we will be getting, this type of Karnaugh map realization, from the previous table.

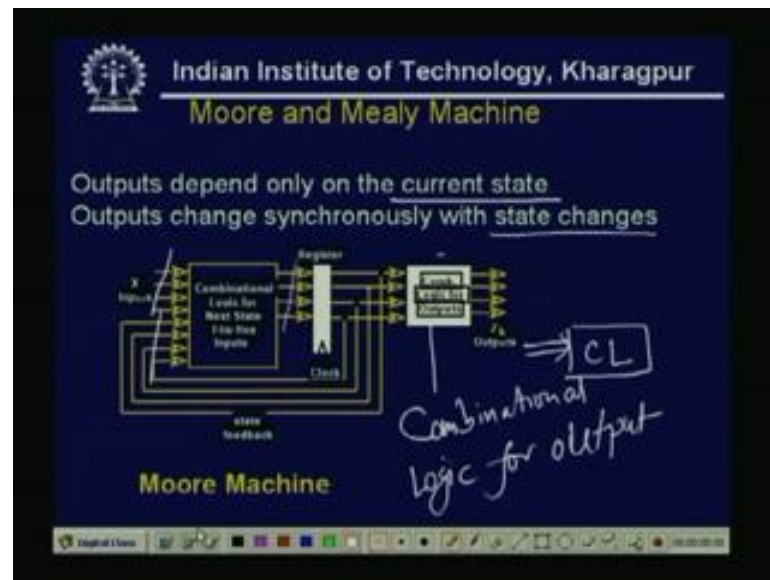
See, here that D N 0 0, Q 0, Q 1 is 0 0, then the output is 0, similarly, if it is 0 1, and still the D N inputs are 0 0, then it will be 0, in that way, you can do the thing. So, now after minimization, Karnaugh Map minimization, if we complete the circuit, then it will be like that, say for each output of the AND gate, there are Q and N. Now, after minimization we get that, D 1 is Q 1 plus D plus Q 0 N. So, if we realize that thing, then first is, this is a Q 1 and Q 0, so this is my Q 1, this is my say Q 0.

Now, again this Q 1, actually that should be D 1 is Q 1 plus D plus Q 0 N. So, here there will be, these all gate, has this one input Q 1, second input is D, and it will be Q 0 N. So, this is Q 0, this is N. So, it will, generate this D 1 input, so if I now put, one D flip flop here. So, this is my as if D input, and this is my Q input, and this is my clock, and this is reset. Now, what will be D 0, D 0 is N Q 0 plus Q 0 N, plus Q 1 N plus Q 1 D, so there are 4, 4 AND gate.

So, first we realize that, that N Q 0, so this will be N, this will be Q 0, similarly this will be Q 0, this will be N, this is Q 1, this is N, and this is Q 1 D. So, this will be the Q 0, and similarly, if I put, a D flip flop, these input then, these will be the D output, and these will be the Q 0, try if this is D 0, this is D 1. And, this is Q, Q bar this is reset, this is clock, so see 4 AND gates, or 5 AND gates, and 2 OR gate, and another 1 AND gate at the output, so total 8 gates are needed for the design.

Among which, 5 are 5 AND, 2 OR, and actually you can tell 6 AND, 1 AND at the output, so this 8 gates are needed for this particular design. So, now we have got a, this is one vending machine design, just now what we consider.

(Refer Slide Time: 46:50)

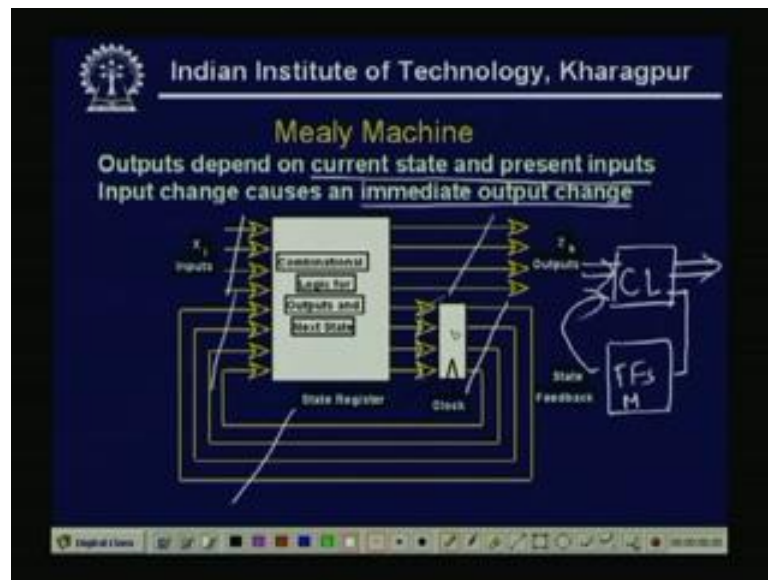


Now, we define two general machines or conventional machines, normally the overall machines, or the, of the any type of machines, that is classified as, Moore machine and the Mealy machine. Now, what do mean by, Moore and Mealy, that we defined now. Now again, the general machines as we have defined, that outputs depend, only on for the Moore machine, the output depend only on the current state, say only on the current state.

And, the output change, synchronously with state changes. That means, whenever a state changes, because the output depends only on the current state. So, the output will change, so if we see, the already we have seen, that this is a combination circuit, and here it was for the machine, it was some memory. But, here it does not depend on the previous state, so it is current state, so the input comes here, these are my inputs, and this is a register. That means, the present output of the combination and logics are stored.

And, again these outputs are feedback, to the combination block. As the remaining inputs and the output, depends on the current state, so this is the combination logic for outputs. So that, the machine depends only on the, current state.

(Refer Slide Time: 49:28)



Now, if it is Mealy machine, the output depends on current state, as well as the present inputs. So, this is a difference, that for Mealy machine, that current state and present input, and the input change causes an immediate output change. That means the output change, occurs only when the input changes. So, what will be the general circuitry, that these are the present inputs, present outputs, and as these depends on the current states and present inputs.

So, some of the output bits, it goes to a register, this clock or a we have seen this is actually, a set of flip flops, and then the output is feedback, as the remaining inputs of the combination logic circuits. So, this is the general circuitry that, one combinational logic, one combinational logic and the memory flip flops, or the memory elements flip flops, or I can tell memory elements are there. Some of the inputs, some of the outputs, and some of the outputs are feedback, and this has this is the circuitry, so this is my Mealy machine.

(Refer Slide Time: 51:09)

Indian Institute of Technology, Kharagpur

FSM Word Problem

- Problem defined in English is mapped to Formal Specifications

Example : Pattern recognizer

A pattern recognizer is a sequential system whose binary output at time t corresponds to the particular pattern recognized by the system

So, now we see, the FSM word problem, so a problem defines in English is mapped, to formal specification, say it is a pattern recognizer. Now, pattern recognizer means, that means, one machine you have to construct, where a particular pattern is, if it is feeded, then only one, output will be generated. Otherwise, always the machine will generate a 0 output. So, we define a pattern recognizer is a sequential system, whose binary output at time t corresponds to the particular pattern, predefined recognized by the system.

(Refer Slide Time: 51:53)

Indian Institute of Technology, Kharagpur

Pattern Recognizer

- A finite string recognizer has one input and one output. The output is asserted whenever the input sequence ...010... has been observed, as long as the sequence 100 has never been seen.

Step 1. Understanding the problem statement

Input: 00101010010...
Output: 00010101000...

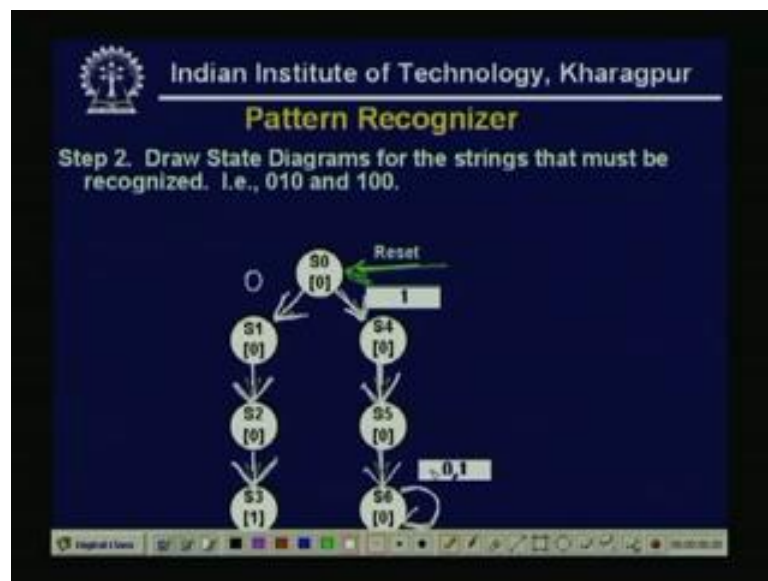
Input: 11011010010...

Handwritten notes: "terminate" with an arrow pointing to the end of the first input sequence, and arrows pointing to the '010' sequence in the first input and the '100' sequence in the second input.

Now, a finite string recognizer has 1 input and 1 output, the output is asserted, whenever the input sequence is say 0 1 0, as been observed. That means, if 1 0 1 0 arrives, then only it, it generates the output, and as long as the sequence 1 0 0 has never been seen. That means, if 1 1 0 0 is reached, then this logic is not valid, so far. Or this logic is not valid anymore, that means, that it will stop, generating that output for this condition, that is 0 1 0 arrives.

So, step 1 is that understanding the problem statement, say input. Now we see example, 1 0 has come output 0, 0 as come output 0, 1 has come output 0, now 1 0 has come so see, this is a 0 1 0, and the condition is satisfied. So, 1, 1 output is generated, this is 1 1 output, similarly 1 0 1, so it is now 1 0 1 0 is arrived. So this is again 1 1, that means, this is 0 1 0, this is a 0 1 0, this is a 0 1 0. Now, see 1 0 0 arrived, so this is a 1 0 0, that means, terminating, or, or the condition, which is not valid. So, it will be, now, see now 1 0 1 0 as come, but output is 0, but this output is 0. So, this is the problem statement written.

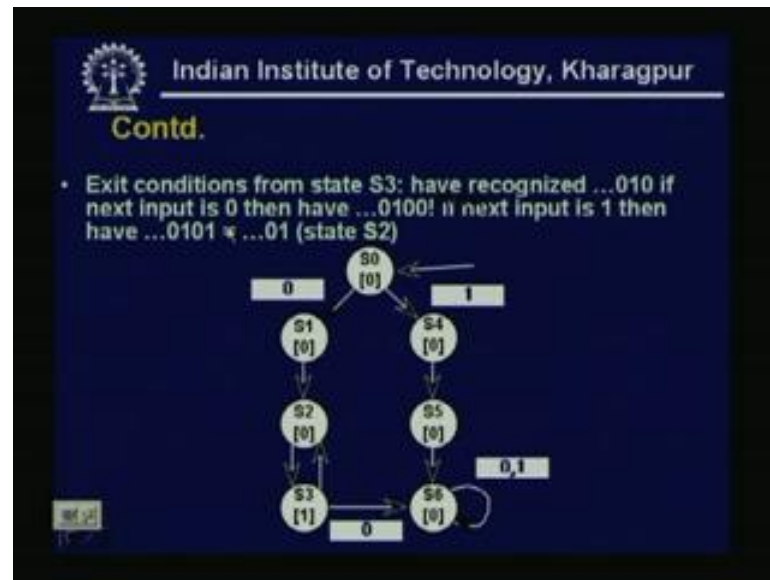
(Refer Slide Time: 54:20)



Now, if we draw the state diagrams for the string, that must be recognized. So, just now what example we have seen, that, this is a, reset initial state, this initial state, that reset. And now, that see, if it is 0 comes, then it is this 1, now again another 0 has come. So, again, it is a S 2, if it is another 0, then again it will be in the S 3 state. Now, if 1 1 1 comes, then obviously, it will be a different state. Because from the initial state if 0

comes, it will go to 1 state, it will in 1 state. Then again, some 1 has come, now again, so this is, if it is a 1 0, now it can be 1 0 0, or 1 0 1. So, this type of, different type state transitions will occur.

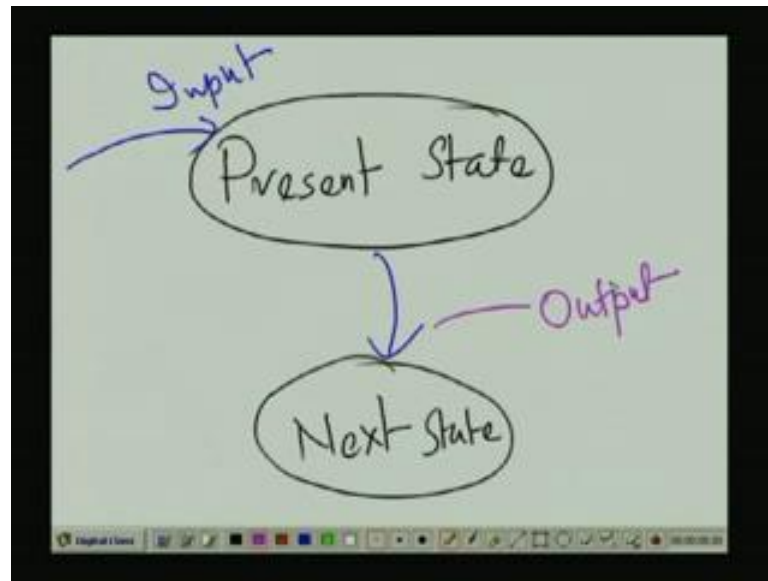
(Refer Slide Time: 55:59)



Now, exist condition from state S 3, so the exist condition from state S 3, have recognized that 0 1 0, if next input is 0, then have 0 1 0 occurs. Now, in next, input is 1, then have 0 1 0 1, so again that, what we have done, that we mainly we want to construct a pattern recognizer ((Refer Time: 56:42)) and the design principle, what the basic design approach ((Refer Time: 56:48)) what we have seen. That, first we have to realize the statement of specification.

So, that part we have done, that if that 0 1 0 comes, then only it will generate an output otherwise not. But if it is valid, until 1 0 0 pattern arrives, if 1 0 0 arrives, then this condition is not, it is not valid. That means, even after 1 0 0, even that 0 1 0 comes, then the output will not be 1, now that part we have read. Now, develop an abstract specification of the FSM. So, what we are doing the actually, the second step, so this step we have done, from the pattern recognizer. Now, the second steps means, now how many states are possible, because now we want to know if this is the present state.

(Refer Slide Time: 57:57)



If this is the present state, what we want? That if a input comes, then what will be the, what will be the next state, what will be the next state. So, for all the conditions, for all the cases of this pattern recognizer, that we have to design, and then, that time what will be, what will be the output, whether the output will be 0 or output will be 1. So, next day, we will be, starting from the step 2. So we have defined the problem, and next day, we will be doing the abstract specification, that is the state able, then the state encoding, and then how the machine will be design. So we will continue there we will be finishing here.

Thank you.