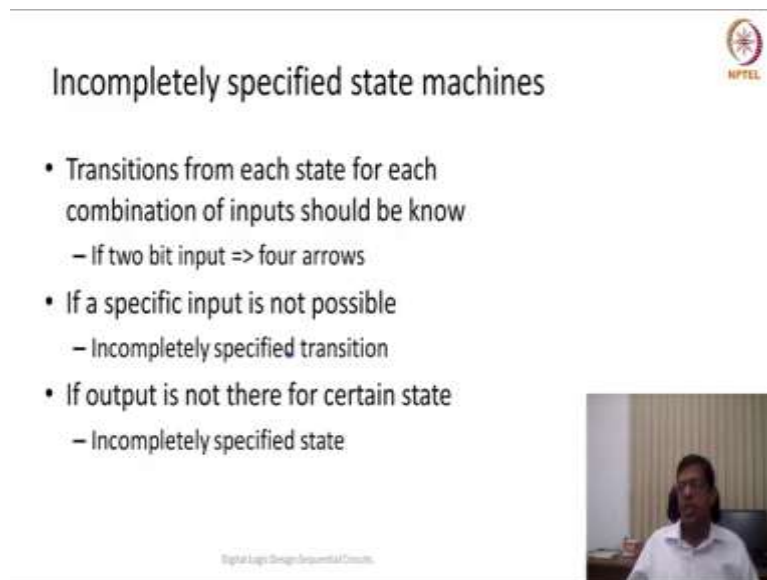



**Digital System Design**  
**Professor. Neeraj Goel**  
**Department of Computer Science Engineering**  
**Indian Institute of Technology, Ropar**  
**Lecture No. 55**  
**State Encoding**

Now, one more small topic, which I would like to cover is some time the state machines could be incompletely specified. So, you, this is an independent topic, it is not related to reduction, but it is like, whenever we try to design a state machine. So, for example, in the previous lecture we were trying to design state machines. So, whenever they were any state, you were saying let us complete the state machine. So, when we were saying let us complete the state machine, it meant that for all possible inputs, let us see what will be the transitions, next state transitions will all possible inputs.

(Refer Slide Time: 01:04)






**Incompletely specified state machines**

- Transitions from each state for each combination of inputs should be known
  - If two bit input => four arrows
- If a specific input is not possible
  - Incompletely specified transition
- If output is not there for certain state
  - Incompletely specified state

Digital Logic Design Department of Computer Science



So, now, you, so this also means that let us say, right now, in all our examples number of inputs are only 1. So, and this particular input all only had two values, 0 and 1. So, if instead of one value there, there will be two values, then we have to find all the transitions for all four possibilities of inputs. So, let us say two inputs are A and B, if AB is 00, then what would be the transition if AB is 01, then what would be the transition? If AB is 10, then what would be the transition so all four possibilities, we have to have all the transitions known for all possibilities. That is how a state machine would be complete.

Now, when we were studying combinational circuit, then also there were some incompletely specified functions. And when a function was incompletely specified, we were saying a function is incompletely specified when certain input combinations are not possible. Because

those input combinations are not possible, so we do not know what is going to be the output corresponding to that. In a very similar case, in a very similar way, a state machine would be incompletely specified when a particular possible possibility of inputs is not there.

So, let us consider the example. First example, we are taken in our case today. So, let us say, it is known that input is always going to be a BCD number. If input is a BCD number, then the possibilities of input four disjoint bits could be only from 0000 to 1001. Any number which is more than 10 is not possible. So, because those numbers are not possible, that will make those transitions as incompletely specified.

So, this incompletely specified state machines is a very interesting case, because in the state machine, it looks like that those arrows are missing. And this becomes even more interesting that when we are going to reduce these state machines. So, in case of reduction of state machines if the if it is incompletely specified then sometime we can optimize the way we were optimizing in K-maps.

In K-maps, if something was incompletely specified or they were do not care conditions, then we were at our will, we were trying to say that whether if they are helping us in reduction of a state machine, then we will optimize then we can combine with any other state. So, similarly, those kind of things are possible here also, but we are not going in detail those examples and but yes, that is the overall idea.

There is also one more possibility that the state machine could be incompletely specified when output is not there for a certain state. So, let us say it is a it could be Mealy Machine as well as Moore Machine. So, in case of a Mealy Machine, let us say for particular state transition from A to B, the output cannot be specified. So, there is no output. So, that means output is do not care. We do not know what the output is. Because nobody is going to read that output so output is not specified, and then also your state machine will become incompletely specified.



So, ,these incompletely specified state machines could have again, the opportunity for optimizations and the state reductions, and optimization of Boolean logic also, so, that always presents us like an opportunity.

(Refer Slide Time: 05:11)

**Procedure of state machine implementation**

- Design of state diagram or state table
- State encoding
- State table -> Transition table
  - Transition table is state is replaced with the encoded value
- Find boolean expression for next state of each register
- Find boolean expression for each output

Digital Logic Design Sequential Circuits




So, let us overall summarize our finite state machine design problem. So, when we have a problem in our hand we would like to let us say do a pattern detection or any other state machine we would like to design. So, the step for all of those things would be the first step is that we will design a state diagram. Sometime we can directly write a state table. So, after finishing the state diagram, then we have to design, we have to write the state table.

After finishing the state table, now towards the implementation we have to, the second step would be to encode the state. So, because so far all the states were given alphanumeric numbers S1, S2, S3 or ABCD, now, those states has to be encoded into binary numbers. When the states has been encoded to binary numbers, then the state table will also become a transition table, because they are in the condition table essentially, all the states are replaced by their encoded values.

So, once we have this transition table with us, then for each flip flop for each memory element we would find what is the next state equation. So, this we have done in our first and as well as the second lecture over our finite state machine model. So, then, after finding the next state for each of the flip flop, then so, you see then this next state and this state is different because now we are finding the next state for each flip flop. So, and after finding the next state for each of the flip flop then we would find the Boolean expression for each of the output. So, let us quickly see with an example.

(Refer Slide Time: 07:18)



### Example

State Table


Present state	Next state x=0	Next state x=1	Output X=0	Out X=1
A	B	C	0	0
B	D	E	0	0
C	E	D	0	0
D	H	H	0	0
E	L	H	0	0
H	A	A	0	0
L	A	A	0	1

Transition Table

F1F2F3	F1'F2'F3' x=0	F1'F2'F3' x=1	Z X=0	Z X=1
000	001	010	0	0
001	011	100	0	0
010	100	011	0	0
011	101	101	0	0
100	110	101	0	0
101	000	000	0	0
110	000	000	0	1

A = 000	E = 100
B = 001	H = 101
C = 010	L = 110
D = 011	

Digital Logic Design Sequential Circuits



So, in this lecture itself, we have seen that we have taken an example where we were trying to optimize, trying to find out the state table finite state machine for matching the pattern 1001 and 1010, and we came up with this this optimized or reduced state table. So, in this reduced state table, we have to first of all uniquely identify how many states are there. We see A, B, C, D, E and H and L. So, these are the 7 states which are uniquely there. So, for 7 states, we would require at least 3 bits represent. Now, we have to assign that, which 3 bits to be given to each of the state.

So, let us say, we simply follow the number that this A is given 0 then 1 then 2 then 3, 4 4 and 6. So, these billion numbers are given. Now, these numbers need to be replaced in the state table, so that we have a transition table with us. So, in transition table, because we require three bits to store these states, so we would require three flip flops. Let us say flop 1, flop 2 and flop 3. So, this flow 1 flop 2, flop 3 represents the present state value and F1 plus F2 plus and F3 plus a represent the next state value. So, and then we replace A with 000 B with 001 and C with 010, so on so forth for all other, other values.

Once this transition table is there, then we have to draw a K-map or we can write the Boolean expression in other way also for F1 plus in terms of F1, F2, F3 and X. Similarly, for F2 plus we have to find out the expression and for F3 plus also we have to find the expression, this would be the next state expressions for all the three flip flops. And even though it is a D flip flop then the method would be straight forward. But if it is JK flip flop or RS flip flop or T flip flop, then we have to map those equations according to the flip flop which is present with us.

Now, after finding the next state equations of all the flip flops, then we also find out the Boolean expression for my output Z. So, which for example, in this case, Z is 1 only in this case, so that means when F1, F2 and F3, F1 so I can write it like F1, F2, F3 bar and X that would be the expression of output.

So, now, one question here that which value should be mapped? So, when we are saying that ABCD are given some value, so A could have been given 000 as well as 001. So, similarly, all these 7 combinations, there is a good amount of possibility, good possibility that the total possibility of what could be unique combinations where these three bits could be assigned to A, B, C, D, E, H and L would be of the order of let us say, here 8 factorial, so the number is huge. So, and the problem is also important.

(Refer Slide Time: 10:52)

The slide is titled "State encoding problem" and features the NPTEL logo in the top right corner. It contains two main bullet points: "Number of states are N" with a sub-point "Number of minimum bits required  $\log_2 N$ ", and "Number of possible encoding possibilities - exponentially increase with N". A small video inset in the bottom right shows a man speaking. At the bottom of the slide, the text "Digital Logic Design Sequential Circuits" is visible.

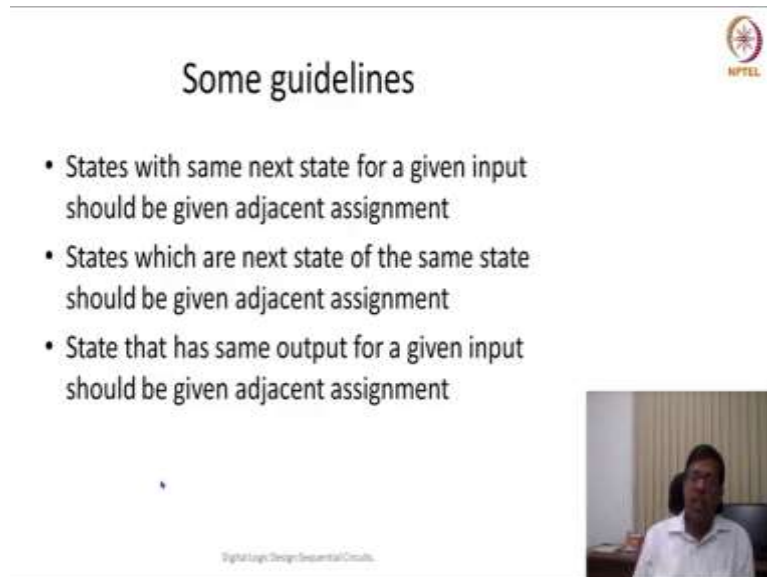
Then your question could be that why this column is important. So, we are summarizing here that if we have n number of states the minimum number of bits required would be  $\log_2 N$ . But the encoding possibility that which encoding should be given to which particular state is exponentially large. So, we can see in this case, if there are 7 states and there are 3, 3 bits which are present the total number of possibilities that what should be given to A and B and C, so, it would be the order of 7 factorial. Actually, it is more than that 8 factorial.

So, this this this number is huge. Which are the two questions now? Which state encoding should be given to each particular state, the second question is also important, but that why this encoding would be important. So, you see, one of the previous lecture, we have taken an example by showing that if we take a different encoding, then the number of gates which we

choose or where we find the next state equations as well as the output functions it would vary. So, what could be a good approach?

So, ideally it is difficult to find out what is the best encoding. It is going to take a huge amount of computation because number of possibilities are huge.

(Refer Slide Time: 12:29)



The slide is titled "Some guidelines" and features the NPTEL logo in the top right corner. It contains three bullet points:

- States with same next state for a given input should be given adjacent assignment
- States which are next state of the same state should be given adjacent assignment
- State that has same output for a given input should be given adjacent assignment

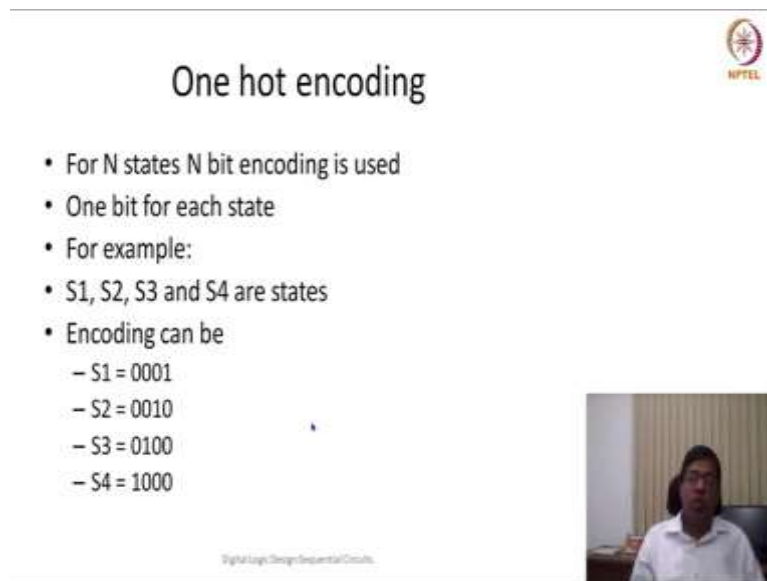
In the bottom right corner, there is a small video inset showing a man in a white shirt speaking. At the bottom center of the slide, the text "Digital Logic Design: Sequential Circuits" is visible.

So, what we do is, we try to find out try to follow certain guidelines, which can help. These guidelines suggest that if, so, if the transition is same. So, for example, if we are in the next state for a given input a same. So, let us say two states A and B have the same next state C, if the input is 1, then A and B both could be given adjacent assignment. When we say adjacent assignment, it is in terms of gray code that the number of bit difference is supposed to be 1.

Similarly, if the states which are the next state for the same state. So, for example, A's next state is B and C's next state is also B, then also A and B should be given adjacent assignment. Now, this this would help us to reduce the equations or reduce the binary Boolean expression for next state calculation. Similarly, we would like to reduce the output expression. Then we have to say that if the states has same output then they should be given adjacent assignments, so that would help us in reducing the expression of the output.

Yes, one thing which is also important to say here that it is, we would like to reduce, but it is a known it is a hard problem to solve, computationally hard problem to solve. So, we can always approximate. There is no optimal solution, optimal solution would require much higher computation. So, we will go with some guidelines and some approximation and we will find out the solution. One of the other solution, which could give a trade also.

(Refer Slide Time: 14:29)



The slide is titled "One hot encoding" and features the NPTEL logo in the top right corner. It contains a list of bullet points: "For N states N bit encoding is used", "One bit for each state", "For example:", "S1, S2, S3 and S4 are states", and "Encoding can be" followed by four sub-points: "S1 = 0001", "S2 = 0010", "S3 = 0100", and "S4 = 1000". A small video inset in the bottom right shows a person speaking. At the bottom of the slide, the text "Digital Logic Design Sequential Circuits" is visible.

Let us say, if we try to increase the number of bits which can be given, then the solution is always going to be simple. So, for example, if we use N bits for N states, we call it one hot encoding. So, then one bit would be given to each state. You can, if you want to try experimenting it, you can do it yourself that if you do this one hot encoding, the expression for next state as well as the expression for the output is always going to be very, very simple.

So, further in case of FPGAs or CPLDs, where each state is stored in a different flip flop and we have number of flip flops which are available. So, one hot encoding is quite popular because flip flops are cheap there so each LUT has at least one flip flop. So, that is why one hot encoding is quite prominent when we are implementing our circuits using FPGA.

So, let us say S1, S2, S3, S4 are 4 states then encoding would be will use 4 bits and one other bit will be 1 for each of the state. Rest of the bits would be 0. So, this is how we can use one hot encoding to reduce the total number of expression and reduce the size. So, we have no two X teams. One is reducing the number of flip flop. The other this one hot encoding would reduce the area, as well as delay. Thank you very much.

So, with this, I would like to close today's lecture. And in summary, what we have done in today's lecture is we have seen that how a state machine could be reduced. How the states could be reduced. And if we can, we have seen two methods of reduction of state machines. One method was completely a visual approach, where we were saying if two rows are equivalent, then we were two rows are same, then two rows of a state table is same then the two states are same.

The other approach was little convoluted where we have implied that two states are same because they are not different. If they are different, then they cannot be same. So, using this Converse formula, we have seen that if we can find out if two states are equivalent or not. So, with this, I would like to close. Thank you very much