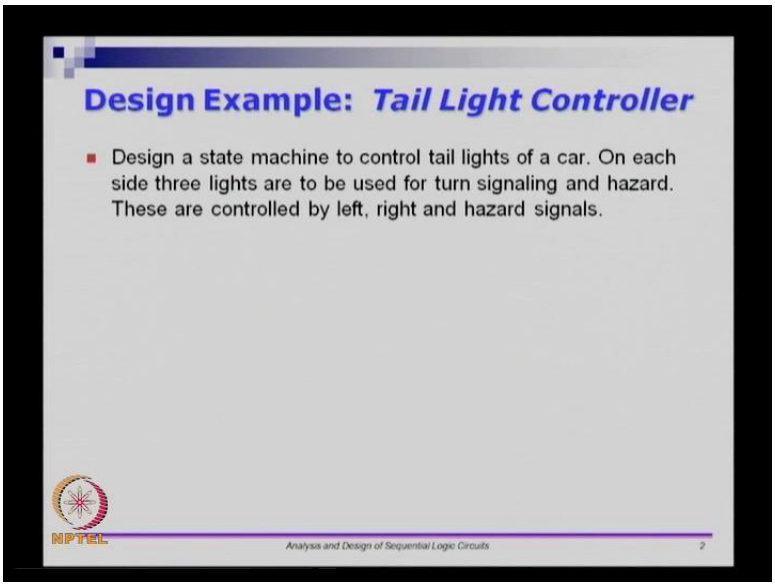
**Digital Circuits and Systems**
**Prof. Shankar Balachandran**
**Department of Electrical Engineering**
**Indian Institute of Technology, Bombay**
**And**
**Department of Computer Science and Engineering**
**Indian Institute of Technology, Madras**

**Module - 31**
**State machines 3: State Minimization**

Welcome to module 31 of week 6. In this module, we are still going to look at State Machines and we are going to learn something very fundamental state machines, which is called State Minimization. So, before we look at state minimization, I will show you a design example first, which is another very interesting example and a very simple example to understand. I will show that, and then we will talk about state minimization. So, let us look at this design example.

(Refer Slide Time: 00:43)

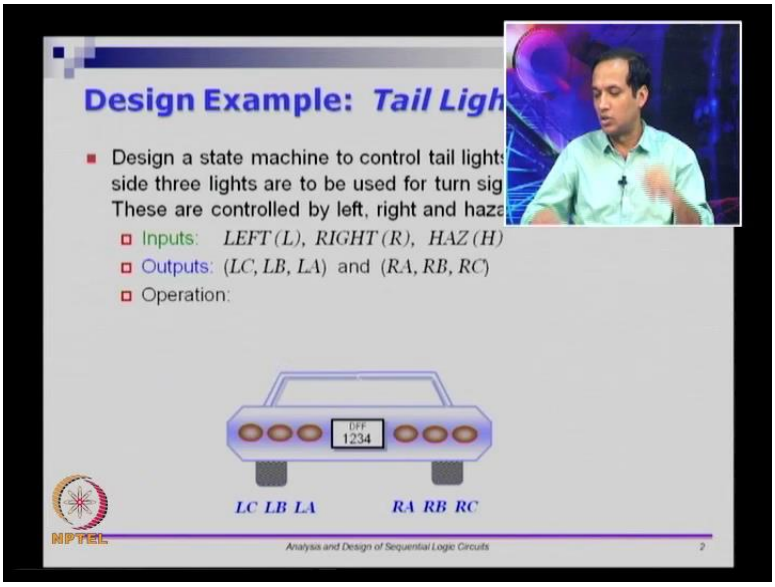



So, this is called a tail light controller. So, if you drive a car and we have going to design a state machine, which are going to control the tail lights of a car. So, we are going to have a car, which has three lights on left side and three lights on the right side. We are going to control it to turn signals, which means, if I want to a left turn, I want to show that I am turning left. If I want to do a right turn, I want to show that, it is turning right or I may have a hazard.

So, hazard means, there is some problem in my car, so I want to press a button inside

that, there is some panic and anyone, who is behind me should understand that, there is this lights blinking, which means there is a problem in the car. So, these are controlled by let us say left right in hazard signals. So, from a car control point of view, when I drive, so if I press down the indicator, I want to indicate left, if I press up, I want to indicate a right, and then if I press the button, I want it to press this hazard. So, let us assumes that they are 3 inputs; left, right, and hazard, which are given to the circuit and I want to control the tail lights accordingly. So, let us see what the tail lights are, the tail lights are going to have some interesting thing.

(Refer Slide Time: 01:56)



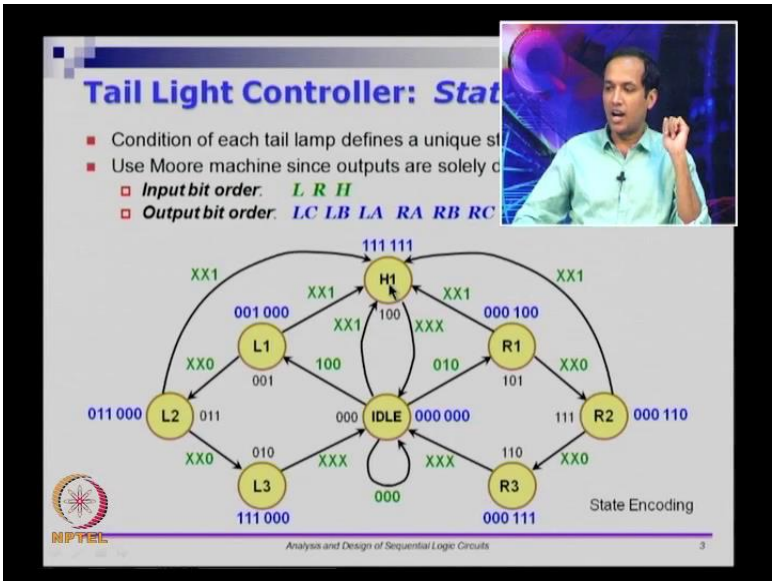

So, there are 3 inputs left, right and hazard and I am going to have 6 outputs, L C, L B, L A are 3 left side indicator lights and R A, R B, R C are right side indicator lights and the car I have is something like this. So, this is my car here, I have three lights on the left side and three lights on the right side and if I press left, then what I want is… So, you see the thing that is happening here, I want to start from the left most bulb, then turn on this one, then turn on this one and come back in turn off all of them, this is what I want for left.

So, for the right similarly, I want to start from here, then turn on this one, turn on this one and then comeback and then keep doing this over and over, let us say that is the thing that I want. So, this is not a typical signal that you will see in cars. So, in most cars, you see just one light on the left side, which keeps blinking and one light on the right side, which keeps blinking. But, this is a very simple example to give you, how to design a controller.

And if it is a hazard signals, what I want is, all the lights should blink and go off, it should keep blinking as long as the hazard is on. So, as long as I keep, if I have left indicator on, then I want this bulb, this bulb and this bulb to glow, then all of them goes off. Then, again this bulb, this bulb, this bulb glows and all of them goes off and so on, this keeps happening for a ever, until I remove the left indicator.

Similarly, for right indicator and if I want to indicate hazard, I press hazard equals 1, if I put hazard equals 1, all these lights should all blink. So, which means it should go to 1, go to 0, go to 1, go to 0, all together. So, let us say, this is the circuit that I want to design.

(Refer Slide Time: 03:55)




So, let us design a state machine to that. So, the ideal state, we will mark it as all 0's. The ideal state is one, you are driving, there are no conditions, you are going straight, you do not have a problem in the car, so all the outputs are 0. So, L C, L B, L A are the three left indicator bulbs and R A, R B, R C are the right indicator bulbs, all of them indicate 0. And if all the inputs are 0, I do not have any conditions, so I will stay in the ideal state of the lights itself, the lights are not glowing.

The moment I turn on the left indicator, the left indicator is turned on, names L equals 1, R equals 0 and H equals 0. The moment I do that, the first thing I want to do is, turn on the bulb called L A. Then, turn on the bulb called L B, turn on the bulb called L C and go back to the idle state. So, let us see this one here, I press the left indicator, the moment I press a left indicator, then the cycle that I want is, L 1 should glow then in the next cycle

L 1 and L 2 should both glow, then in the next cycle both L A, L B and L C, all 3 should glow.

So, I said L 1, L 2, L 3 are the state names. So, at L 1, I want L A to glow, at L 2, I want both L A and L B to glow and at L 3, I want L A, L B, L C all of them to glow. And I want this to happen as long as this hazard light is not turned on, maybe I put the left indicator on, but suddenly I see a problem, immediately I press the hazard, I want to be able to take care of that.

As of now we are explicitly checking, whether the hazard is 0. So, if the hazard is 0, then I go to the next state from L 1 to L 2, if the hazard is still 0, I go to L 3, if it is 1, we have to do something about it. So, this is for the left indication. Similarly, I will put something for the right indicator. So, as long as R is 1 and the other two are 0, I will go to the right indicator mode, the right indicator mode I start with arrays as one first.

Then, in the next cycle, I will make R B as 1, the next cycle I will also make R C has 1 and I will do this as long as the hazard signals is not turned on. And the moment I finished one cycle, no matter what happens I come to the ideal state. So, I indicate, so R A turns on, R B turns on, R C turns on and all of them go to 0. So, that finishes this and if my right indicator is still on, then I will keep doing the same cycle over and over, till the right indicator is removed. So, this is for the right indicator.

Finally, if I want the hazard, so if the hazard light is turned on, if I press the hazard button on my car, I want all the lights to glow and they should all go to 0. So, this gives you the feeling that it is blinking. So, all of them turn on, all of them turn off, all of them turn on, all of them turn off and this will keep happening as long as the hazard is 1, no matter whether I turned on the left and right indicator.

Even, if I turn on left and right indicator my, as long as the hazard switch is pressed, I want this blinking action. So, this blinking action will come using the state H 1. Now, I have to take care of several other transitions. So, if I from L 1, I have a X X 0 transitions, it means, it is take care of all combinations of L and R, but H to be 0. What about the case, where H is 1? So, if H is 1, even from L 1, I press the left indicator, but I see a problem, immediately I want to go to hazard. So, that means, from L 1 you directly go to H 1.

Similarly, from R 1, you directly go to H 1, the moment you see that there is a hazard and you will need that form L 2 also. So, the left light L A came on, then L B came on, I

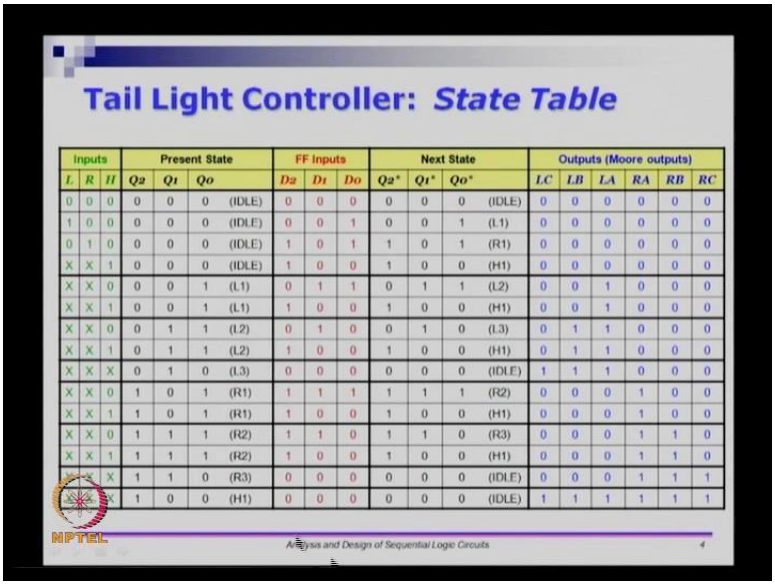
suddenly press the hazard, maybe I was turning and then I have to break, because there is I see somebody in the front, I press the hazard light immediately. So, in that case, you should immediately go to hazard and this is the overall circuit. So, the overall state diagram is, if from L 1 and L 2, if you see that hazard gets turned on, then you want to go back to the hazard itself.

But, from L 3, if you see a hazard, you press 1, what would happen, let us say hazard gets turned on now. Then, any way in the next cycle, you come to idle and after that, you will any way go to the hazard state. So, instead of having a transaction from L 3 to H 1 directly, we have one cycle delay to go from L 3 to H 1. So, let say that is the state machine and we can do, what is called state encoding.

In state encoding, what we do is, we for each of the state, instead of symbols H 1, R 1, idle and so on, we give bits. So, let us see the number of states here, there are 3 states for the left, 3 states for the right and 2 states for this, that is overall 8 states, 8 states can be represented using 3 states registers or 3 flip flops in the state register. So, there are 3 flip flops which means, they can take eight different combinations.

So, to the idle state, I am going to denote it as 0 0 0. So, let some Q 2, Q 1, Q naught, if it is all 0's, it means we are in the idle state, then we are given different names to different things. So, 0 0 1 here, 0 1 1 here, 0 1 0 here, 1 0 1 here, 1 1 0 here and 1 1 1 here and for H 1, we have showed 1 0 0. So, for whatever reason, we have done this.

(Refer Slide Time: 09:51)




And this big table tells you the state register, the state table. So, the present state could be

Q 2, Q 1, Q naught, there are 3 bits. So, there could be eight different combinations and for each of the combinations, you go and write, what are the conditions under which it should go to different states. So, this is the much more precise or concise thing, not precise. So, this is precise as well as concise.

So, what we have is, for Q 2, Q 1, Q naught which is all 0's, if L or H are all 0's. Then, you want the next state to be idle, which is all 0's and we can get that by making D 2, D 1, D naught as 0. So, this would mean all the lights will be 0. If left indicator is turned on and if right is off and hazard is off, then you want to turn on L A, to turn on L A, you want D naught to be 1. And you will see that at this point, the outputs are all 0's even now, so because it is dictated on the state.

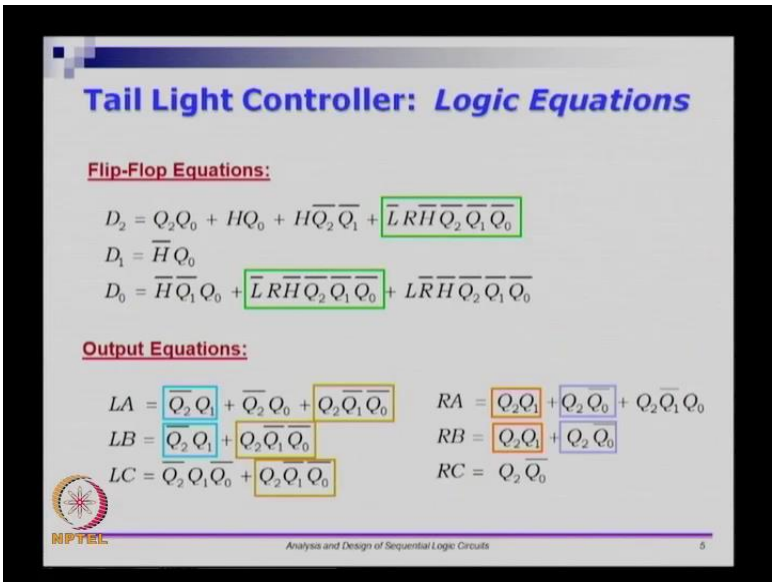
So, based on what state you are in, if you are in an idle state, you always have all the lights turned off. If you are in state L 1, you have L A turned on, if you are in state L 2, you have both L A and L B turned on, if you are in state L 3, all three lights are turned on. Similarly, R 1, R 2 and R 3 you have the output describe. So, there is the much more concise representation.

Instead of looking at L R H, which is 3 bits and Q 2, Q 1, Q naught which is 3 bits. Ideally, it should be writing 2 power 6 or 64 lines of the truth table, the state table. Instead, we are using the do not care conditions and concisely saying, I do not care about L and R, as long as H is 1 and if I mean the idle state, I should go to H 1 state which is the hazard state. But, I will still be producing all 0's as output, because when I go to the state hazard 1, it will take care of blinking.

So, at idle, I am supposed to be blank, at H 1, all lights are glow. So, this is the tail light control. So, it is very simple example that you can very, very quickly relate to, this table may look big, but the description is fairly simple. So, you go back to the state diagram and compare it with this, you will see that, the state diagram is captured in the state table.

So, this is obviously a lot of states, so sometimes you wonder, whether you really need so many states or not. So, you can go and derive the logic equations for it and design a circuit for it.

(Refer Slide Time: 12:36)

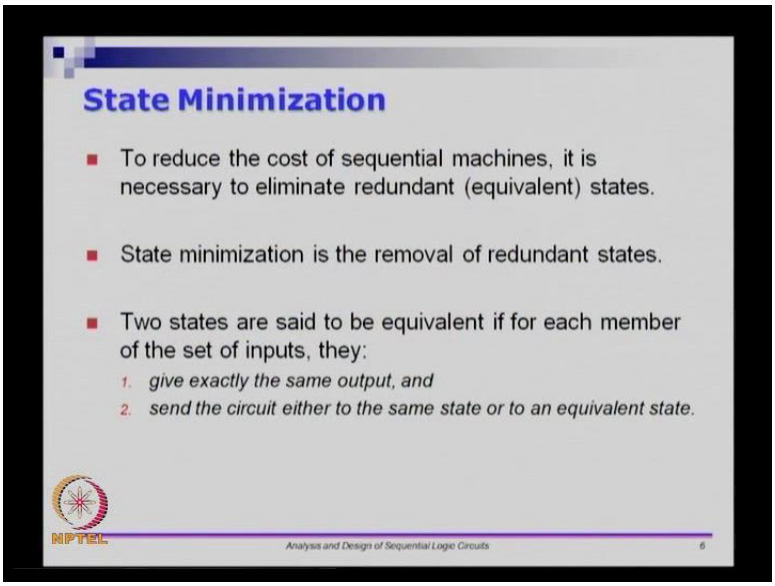



So, in this case, let us look at the logic equations D 2, D 1, D naught, there are lots of common things. So, of course, these are very complicated equations. So, D 2 has 6 variables in this term, D naught has 6 variables in this term as well as in this term and so on. So, let us assume that these are all correct has of now, let us not worry about the correctness.

One thing that is happening is, this term here is shared with this term here, these two are common terms, these two are common terms or these three are common terms across three different equations. These blue ones are common; these orange ones are common and so on. So, in some sense, it is not that you have to spend so many gates, there are terms that you can derive and use them to derive D 2, D 1 and D naught. You have a single circuit which does this term and share it to derive D 2, D 1, D naught.

Similarly, we have one circuit which will do Q 2, Q 1 bar, Q naught bar, use that for both L A and L C and L B and so on. So, this is something you can do. So, you can share the gate outputs. But, sometimes you wonder, whether all these states are necessary or not. So, to do that, we are going to look at what is called state minimization.

(Refer Slide Time: 13:45)

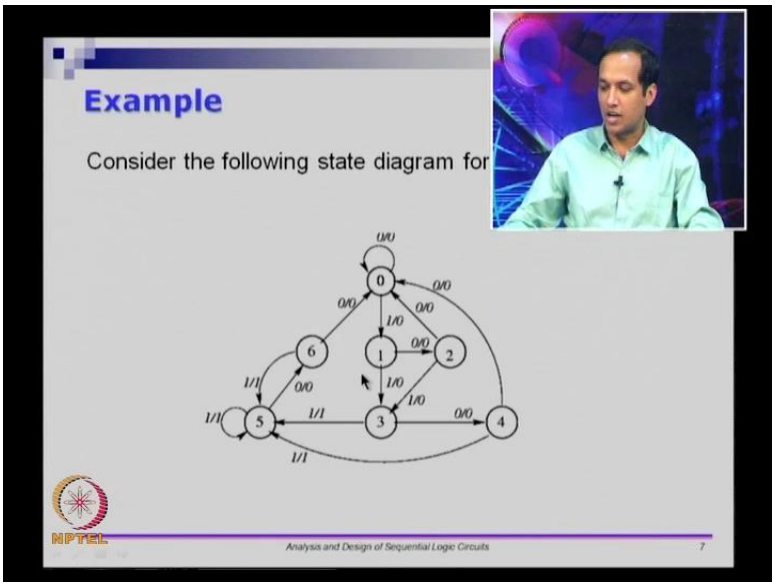



So, if you given a state machine, you may wonder, whether you have states which are not necessary or there any redundant states. So, to reduce the cost of the machine, because you can see this is in this example, all these comes with the cost, may be all the states are not necessary, I do not know. So, I need to have a systematic procedure by which, I can take a state machine and change it to another state machine which has much simpler gates, probably if you have flip flop and so on.

So, by doing that I will reduce the cost of the state machine and to do that, I need to eliminate, what are called redundant states? So, if two states do the same job, I can get rid of one of these states. So, state minimization is the removal of redundant states. So, to reduce the complexity of the circuit, we need to define something called equivalence between states.

Two states are set to be equivalent, in a simple state machine two states are said to be equivalent, if for each member of the set of inputs; they give exactly the same output and send the circuit, either to the same state or to an equivalent state. So, this is a loaded definition, but we will see an example in the next slide, this will make it clear.

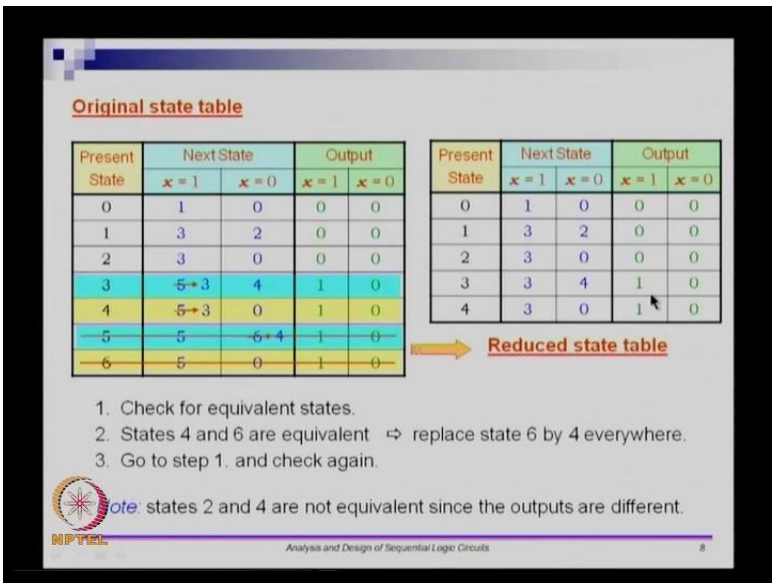
(Refer Slide Time: 15:09)




So, let us consider this state machine here. So, this state machine has 1 plus 3 plus 3, 7 states and there are some transitions given. Maybe, this is the state machine that you gave to me and I want to see, if I can minimize the state machine and I can get something equivalent, which means all the actions that you doing here, I should be able to do that also, I should not miss out anything.

So, the correctness of the state machine, whatever state machine you gave, I want to do the same set of actions, except that I want to see I can cut down the number of states. If I cut down the number of states, the number of flip flops will reduce and the complexity of the circuit is will also reduce, I want to see if I can do that.

(Refer Slide Time: 15:50)



So, let see how to do this. So, let say this is the original state table. So, the present state is marked as 0, 1, 2, 3, 4 up to 6. On X equal to 1, this vector here tells you to which state we are going to, on X equal to 0, this vector here tells which state, we are going to and these are the corresponding outputs. If you are given this original state table, I want to see if there are equivalent states.

So, what is the definition of the equivalent state, 2 state set to be equivalent, if on the same input, they give you the same output and the also move to a state, which are either the same or their equivalent that. So, this is the loaded definition, let see the definition in action now. So, let us look at state 4 and state 6, I am going to say that, they are equivalent, why are the equivalent, I go and look at X equal to 1 and X equal to 0. So, from 4, I will go to state 5 and state 0 respectively, I am also doing the same thing from state 6.

State 6, on input 1, I will go to 5, on input 0, I go to 0, not only that, the output that you are producing for X equal to 1 and X equal to 0 at state 4 are also the same that you produce at state 6. So, this makes states 4 and 6 equivalents. So, what I can then do is, these two states are doing the exact same action, I can get rid of one of them, I will get rid of state 6.

And wherever I see a transactions to 6, let see the table here, from state 5 on input 0 and transactions into 6. But, I said 4 and 6 are equivalent, wherever I see 6, I will put a 4. So, I will go and make that 4. So, wherever I see is 6, I will put a 4. Now, once more I go and check, if there are things which are equivalent. So, now, let us look at this, it is looks like 3 and 5 are going to state 5, on input 1, state 4, on input 0, because earlier it was going to 4 and 6.

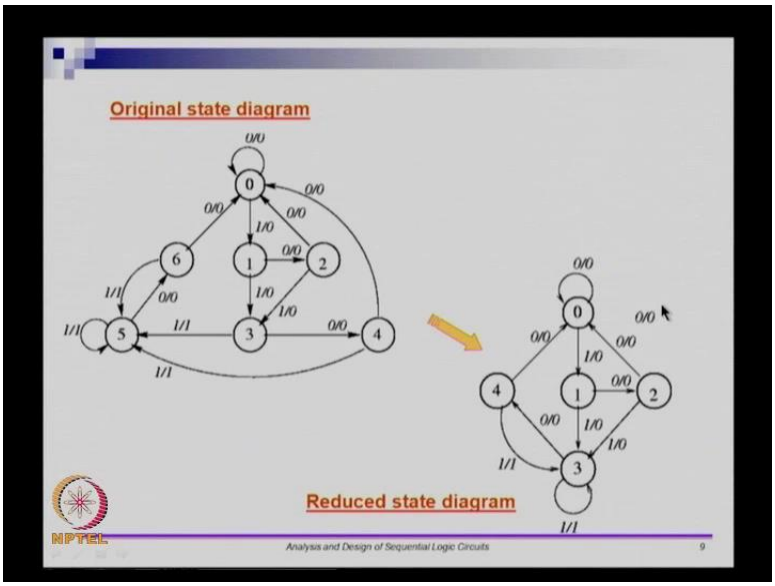
But, now 4 and 6, we have establish to their equivalent, which means, they are going to the same state on X equal to 0, they both seem to produce 1 0 as the output. So, this makes state 3 and state 5 also equivalent. So, we find out the first equivalence based on that now 3 and 5 becomes equivalent. Otherwise, you cannot say that 3 and 5 are equivalent, but we now resolve that 4 and 6 are equivalent, because of that 3 and 5 becomes equivalent and now, I can get rid of state 5 also.

So, you can get rid of 3 or 5; in this case, I get rude of 5 and if you do that, I want to replace all the 5 with 3's. So, now, I have this state's, I can go and check, if there are any more equivalences. So, I see a 3, 2 here or 3, 0 here, 3, 4 here, 3, 0 here and a 1, 0 here.

So, maybe states 4 and 2 could potentially the equivalent, but I go and look at the output, one is producing 0 0 as the output for 1 0 combinations and another is producing 1 0. So, that makes state 2 and 4 not equivalent to each other.

So, this leaves me with 5 states, so I can now draw simpler state table this called the reduce state table. So, wherever I see a 5, I put a 3, wherever I see a 6, I will put a 4 and get rid of the rows 5 and 6. So, that gives me the state, this example I have 5 states. So, the 5 states means, you still use 3 flip flops, but the logic probably reduces. So, you have fewer states then what you had and this will to exactly the same as what this state machine was doing. Whatever this state machine is doing, this state machine will do exactly the same, we have not change the functionality of the state machine at all.

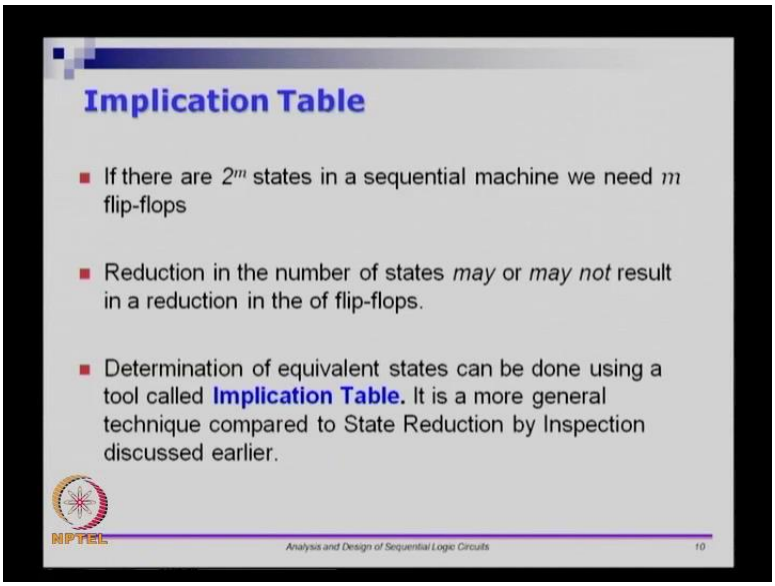
(Refer Slide Time: 19:52)




So, the original state diagram was this and the current state diagram is this. So, you can check, whether these two are equivalent By giving in some inputs. Assume a sequence 1 0 1, 0 1 1 and what not, you run it here and you run it here, it should, so if I give you 10 bits, sequence of 10 bits. The 10 bits will eventually take you to some state producing some output have all along the way.

The same 10 bits if I give here, it should take you through the states which are all equivalent, it should also I have produce the same set of outputs. You go and check that yourself and convince yourself that, this technical is correct.

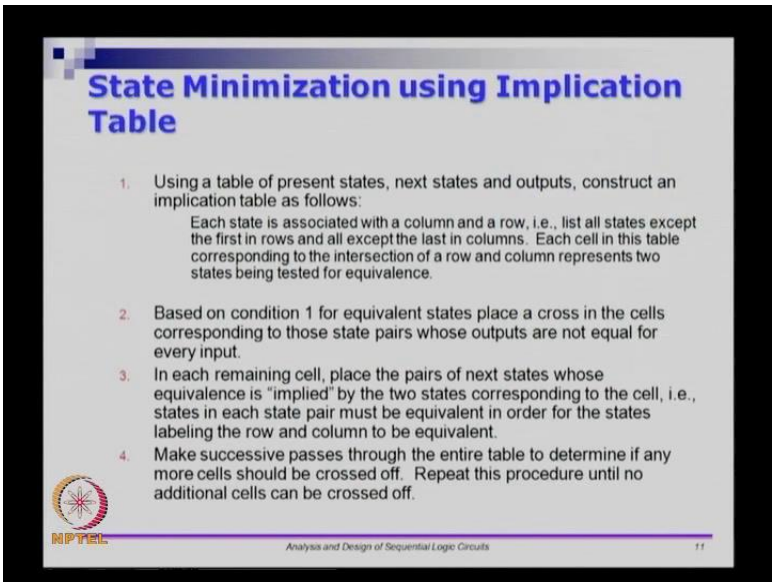
(Refer Slide Time: 20:30)




So, in general what happens is, if there are two power m states in a sequential machine, we need m flip flops. So, if have 6 states is equivalent to 2 power 3 or 8, it is closes to 8, we need 3 flip flops. Reduction in number of states; may or may not reduce the number of flip flops. In the previous example from 6 states we when to, so we had state from 0 to 6, then we went to states from 0 to 4. So, it is still has 5 states, which means, we need 3 flip flops.

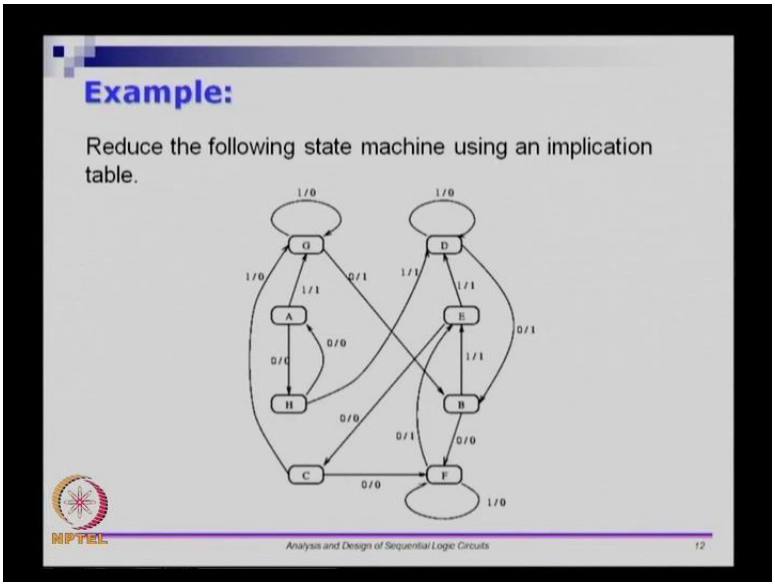
However, from 7 states, we had gone to 3 states or 4 states; it would have cut down 1 flip flop. So, there is more mechanical way of finding out the equivalents between states and I am going to do that using what is called a implication table. So, I will show that in a little while.

(Refer Slide Time: 21:22)



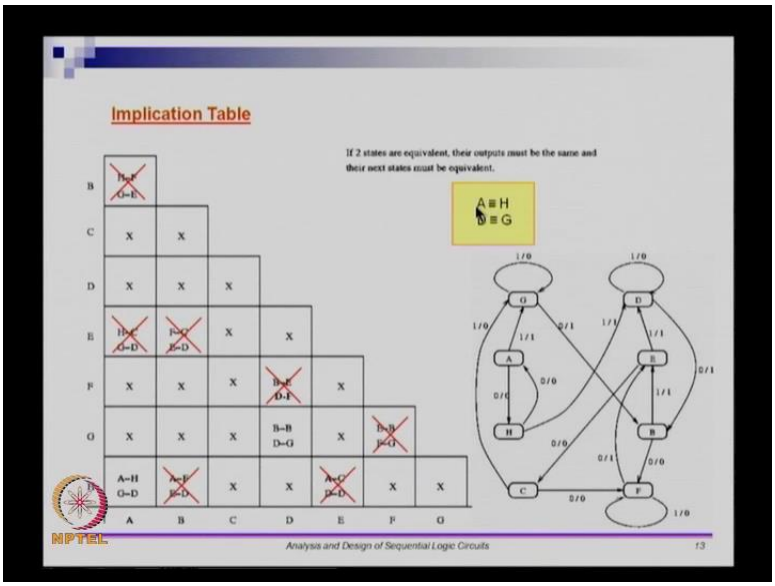

So, there is an algorithm with which you can do this, but I will show you the picture, I suggest that you go back and read the algorithm, if you get some were with the example, you go and use this algorithm and do this yourself. I will describe the algorithm directly.

(Refer Slide Time: 21:35)




Let say, this is the state machine that is given to you, so there are 4 plus 4, 8 states that are there, the states are number from A to H and there are transactions that are given. So, I want to go and write in implication table for it.

(Refer Slide Time: 21:53)




So, the way the implication table works is as follows. So, what you do in the X axis, you put the states. So, if have 8 states, I am going to put 7 of this states a, b, c, d, e, f, g. So, I can pick 7 states, I will put those 7 in some order here. And what I will do in the columns is, if I have A to G here, I will put B to H here. So, we know that a state is always equivalent to itself, we do not need that at all, we do not need check, whether is state equivalent it is at all.

A state is always equivalent itself, because it will produce us same outputs, it is going to the same state on it is transactions and so on, you do not need that column. So, instead what we have is, we have several states here, except the last state and similarly except the first state, you put the all the states here. So, that is in the set of rows and columns. Let us now go and inspect these cells that are inside. So, let us go and inspect each one of the cells.

So, I look at A and B. So, on the rows I will look at A, on the column, I look at B and I have some entry here. If I go to the state machine on A, on a 0 input go to H, B on a 0 input, you go to F. If you want A and B to be equivalent, then on 0, it should go to either the same state or it is go to the equivalent state. So, if I want A and B to be equivalent, then H and F have to be equivalent; that is what you have here.

Let us look at one transactions from A, if on one from A goes to G, one from B goes to E. If I claim that A and B are equivalent, then by the definition of equivalence G and E should either be the same state or they should be equivalent state. So, what we are

putting here is that, we if I want A and B to be equivalent, these two conditions must be satisfied; that is what it means.

If A has to be equivalent B, then these two conditions should be satisfied, H should be equivalent to be F and G should be equivalent to E. So, let us look at this cell for instance, if D has to be equivalent to F, then B should be equivalent to be E and D should be equivalent to. This statement for D to be equivalent to F, D to be F has to be equivalent to redundant. So, B should be equivalent to E, only then D can be equivalent to F. That is the meaning of this cell.

Let us look at this cell, if A has to be equivalent to H, A has to be equivalent to H on input 0, that is trivial, if this is equivalent that is also equivalent directly and you want G to be equivalent to D. So, this is a new condition. So, you can for each of the cells, wherever you want to equivalences is put that. Now, go and look at the X's, A on a 0, you produce 0, A on a 1, produces 1, let us go and look at C, C on a 1, produces 0 and C on 1 produces 0.

So, it is looks like, for A and C in the 0 transactions, if you have input 0, the output is 0, but when input is 1, A will produce an output of 1, C will produce an output of 0, this means A and C can never be equivalent. So, you put at X here. Similarly, you go and look at A and D. So, on 0, it produces 0, on 1, it produces 1, D on 0, it produces 1 and on a 1, it produces 0, which means, A and D can on be equivalent, you mark x here.

So, you keep mark in all these X's, wherever you know for sure that, they cannot be equivalent by the definition of equivalence. The equivalence definition says, there outputs must be the same and there next states must be equivalent. So, the output are not the same for A and C, A and D, A and F, A and H and so on. Now, what you have is what is called the implication table.

Given the implication table, now we want to see, where are we can minimize the number of states. From these 8 states, can I come to a different state machine, which has lesser number of states. So, now, let us go and look at each one of these in turn. So, I go and look at A, I go and look at A comma B, then for A to be equivalent to B, H should be equivalent to F.

So, let me go and check H verses F equivalence, from this table, I have already know that, H is not equivalent to F. So, H is not going to be equivalent to F, when A is never going to be equivalent to B, so I cross that. So, this is also becomes an F, X, A is not

going to be equivalent to B. Then, let us check this, for A to B equivalent to E, H should be equivalent to C.

So, let us go and check an H verse C, H was C is X. So, this is also not going to be possible. Let us go and look at the last one, for A to B equivalent to H, A should be equivalent to H itself. So, that is redundant statement, but G should be equivalent to D. I go and look at the column G verses D. So, we have not resolved with the G and D are equivalent yet. So, I will leave this now, I will go to the next column.

The next column I look at, I am trying to see, whether E and B could be equivalent. For E to B equivalent to B, F should be, so let me go one step back. For E to B equivalent to B, F should be equivalent to C. So, let us me look at F comma C. So, F comma C is a cross. So, that is not true, then if I want H to B equivalent to B, then A should be equivalent to F, which we already know is not true. So, this is not possible also, this column says C is not equivalent to anything.

So, this is not equivalent to anything, you know that from this columns C is not equivalent to anything and from this row also C is not equivalent to A and B there. So, C is not equivalent to any state at all. Let us look at this column, we want to check whether F is equivalent to D, if F should be equivalent to D, B should be equivalent to E. So, I go and look at B comma E, I have already know that B is not equivalent to E.

So, I cross that, then if I want D and G to be equivalent, B should be equivalent B; that is trivially true and B should be equivalent to G and that is also, if D and G is equivalent then D and G is equivalent. So, this is actually seems to be correct. So, we will keep that. So, at this point we have established that, D and G are actually equivalent to each other, there is no precondition required anymore.

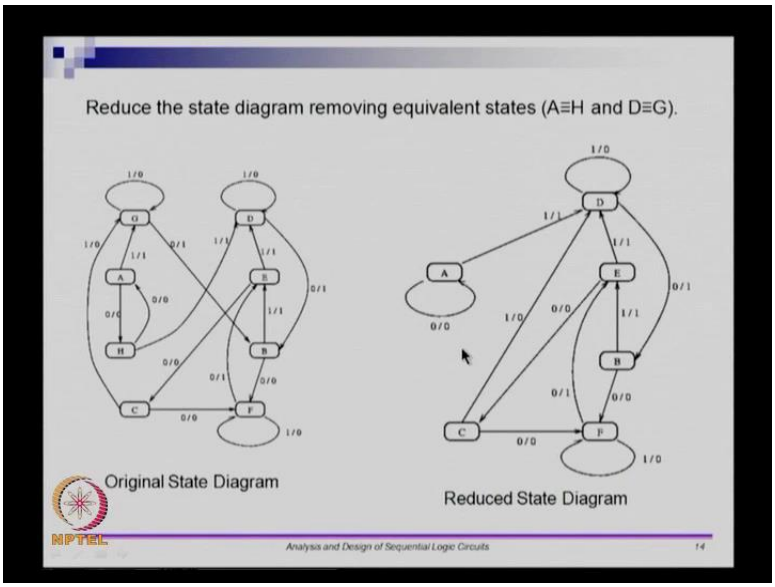
Let us look at this one, H verses E, can H P equivalent to E, for H to be equivalent to E, A should be equivalent to C, which I know is not possible, so I will cross that. Finally, if I want to look at G to B equivalent F, I want E to be equivalent to B. So, I will go and look at B column and E row, I know that that is not true. So, this is also not true. So, what we have done is, we need at several preconditions to be true.

So, that the row and column thus for every cell, the corresponding row and column, if they have to be equivalent to each other. Then the C A preconditions specified in the cell has to be satisfied, but it is not getting satisfied. Now, let us go and look at the table once more, B is equivalent to B and D is equivalent to G, which means, there is no

precondition any more. B is equivalent to B is trivially true, for D and G to be equivalent, I want D and G to be equivalent. So, that is that is a tautology, it is always true.

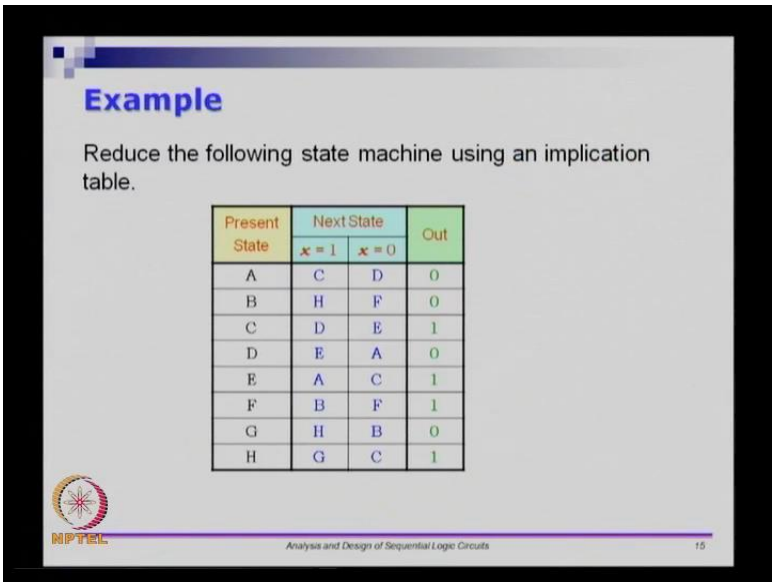
Now, let us comeback and look at this, for A and H to be equivalent A and H to be should be equivalent, it is trivial. So, by default you get that, but you also want D to be equivalent to G. We just establish that D is actually equivalent to G, so overall what we get is, A is equivalent to H and D is equivalent to G..

(Refer Slide Time: 30:35)




So, now instead of now eight different states here, I can replace that with the diagram of 6 states. So, A is equivalent to H and D is equivalent to G. So, from this state diagram, you can go and reduce the state diagram, this potentially reduces the number of state. So, in this case, it went from 8 states to 6 states, which means, it is going to remain at 3 flip flops. However, the combination logic for next state and the output logic, may actually reduce.

(Refer Slide Time: 30:56)




I want you to do this is an example. So, I give you this state machine, this is again having 8 states and this on X equal to 1, you go to these on X equal to 0, you go to these. In this case, the output is not on X equal to 1 or X equal to 0. The output is defined based on the state, which means, this is Moore machine. So, on a Moore machine, what happens is, so the previous example was a mealy machine, this is a Moore machine. If I want to find out equivalence for a Moore machine, then state which produces us 0 cannot be equivalent state which produces 1.

So, by looking at this, I can say that, A cannot be equivalent to C, E, F or H, B cannot be equivalent to C, E, F or H and so on, I can do this. So, what I want it to do is, drawn impeccant table, on the X axis should start from state A go to G. On the Y axis, you start from B go to H and you put all the crosses, were you know that the output of the states are actually different.

So, A and C have different output. So, A cannot be equivalent to C, A and G are different outputs, A and E or different outputs, A cannot be equivalent to E and so on. And whenever the outputs are equivalent, then you need preconditions. If A as to be equivalent to B, you already know that the outputs are 0. So, output is not going to pose problem.

However, for A to be equivalent to B, the preconditions must be, C should be equivalent to H and D should be equivalent to F, you put those preconditions and I want to go and solve this. So, you have 8 states, go and find out, how many states you want for this one.

I will leave this as a home work problem and this brings me to the end of this lecture modules.

So, what we did in this lecture module is, we started with the notion of a state machine and we want it is see we can derive another state machine, which has fewer states. The examples I should you still uses the same number of flip flops, but it is quite likely that, if you are not careful about the state machine, we end up with the very complicated machine with the lot of states. But, many of which are redundant, you want to go back in change it to state which has much simpler things.

In fact I would like you to go and try this on the tail light state machine also and see, if you can reduce the states are not. So, many states, it had 3 state for the left side, 3 states for the right side, one for idle and one for the hazard, wherever 8 states, can you go and see whether the number of states can be reduced even by one. Can you go and check, whether it is possible.

So, if you start with the state machine and if you cannot reduce the number of states, even by one state, then that is state machine is called irredundant, there is no redundancy this state machine. Every single state is important, every single transaction is important, you cannot get rid of the any of the states. So, that is the meaning for an irredundant state machine, but if there is redundancy, you want to remove it, so that the hardware becomes simpler.

Lesser flip flop means lesser logic and what not. So, you want to reduce the number of flip flops, equivalently the number of gates in the input logic and in the number of gates in the output logic, may also reduce. So, I have given several pieces of home work, I suggest that you go back them look at this and solve these problems for yourself. So, thank you and I will see you in the next module 32. In module 32, we will look at what is called state assignment. So, state assignment is also important. So, for we have been using symbols A, B, C, and so on, I want to talk about state assignment in the next module.

So, thank you and I will see you in a little while.