**Switching Circuits and Logic Design**

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

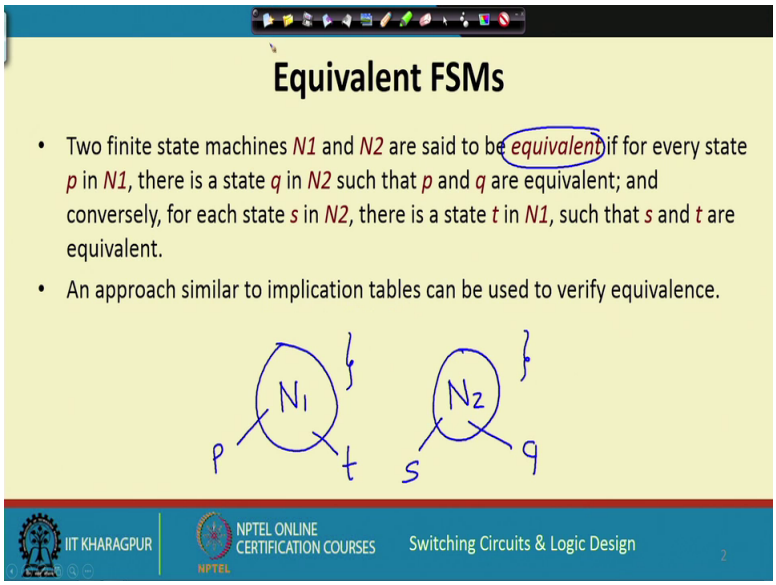
**Lecture - 41**
**Minimization of Finite State Machines (Part- II)**

So, here we continue with the discussion on Minimization on the number of states in a Finite State Machine. So, if you recall in our last lecture we discussed that given a finite state machine, how we can reduce the number of states. We talked about the notion of an implication chart how we can fill up entries in the implication chart, and using that how we can decide whether 2 states are equivalent or not.

So, in this lecture we continue with the discussion this is the second part of lecture on the minimization of finite state machines; now here we basically discussed first the concept of equivalence of 2 finite state machines. Earlier we talked about a single finite state machine and we explored whether 2 states S I and S J are equivalent or not. Now we are saying that we have 2 different finite state machines let say N1 and N2 and we do not know whether the 2 machines are equivalent or not meaning that they represent the same specification or not.

So, will have to find out in some way that some state x of the first machine and some state y of this second machine whether they are equivalent or not. And if we can find out such a one to one correspondence then we can only say that the 2 machines are equivalent. So, the concept that you follow is very similar to that of implication chart and we shall we explaining with the help of examples.

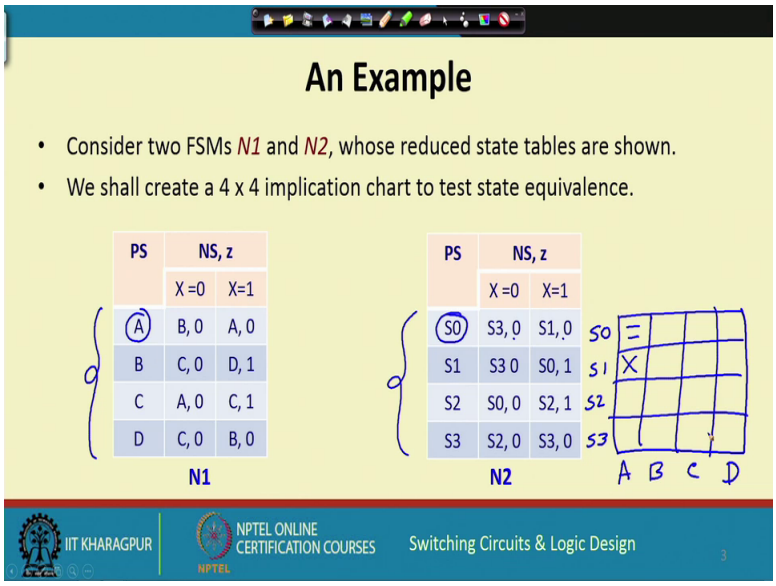
(Refer Slide Time: 02:12)




Now, first let us look at the notion of equivalent FSMs. Two finite state machines N1 and N2 that I have said that I have 2 FSMs I am calling them as N1 other one I am calling them N2. They are defined to be equivalent if for every state P that is there in N1 there will exists a state Q in N2 such that these 2 states are equivalent, as I said there will be a one to one correspondence. The first thing will be that the number of states in N1 and the number of states in N2 must be the same and secondly, there as 2 have has to be a one to one correspondence between the states of the 2 machines.

So, for every state P in N 1 there must be an equivalent state in Q and also conversely for every state S and N2 there must be a corresponding state t in N1. So, it has to be both ways. So, if we can show this then only we can see that the 2 machines are FSMs are equivalent and we shall be using a process which is very similar to implication table for verifying the equivalence.

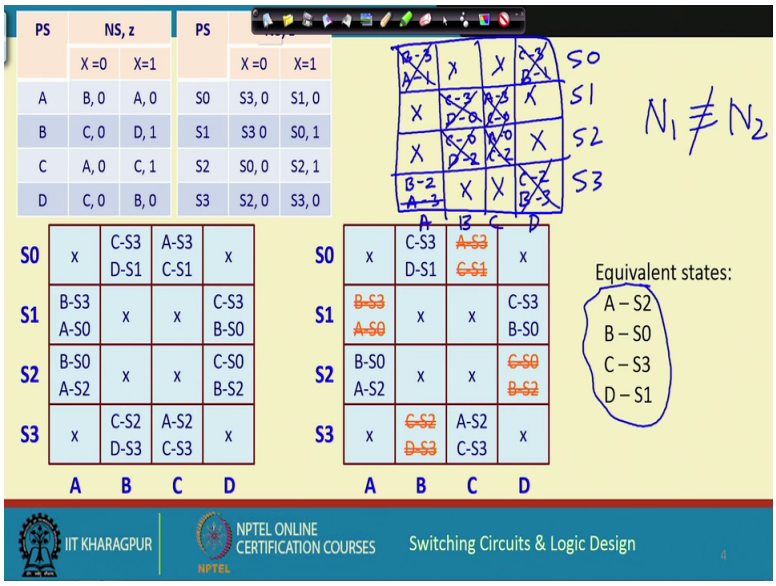
(Refer Slide Time: 03:44)




Let us take an example to illustrate the process because it will be easiest. So, we considered 2 FSMs N1 and N2 as shown, both of them contain four states here this states are A, B, C, D here the states are S0, S1, S2 and S3. Now we want to have a pair wise checking between these states of here and states of here therefore, now our implication chart will look like a square. So, it will be something like this, will be listing this states of the first machine along one side and this states of the other machine along the other.

So, our implication chart will look like this and we will be pair wise considering means every cell of this chart like for example, between A and S0 what will see? That we have a state A here we have a state S0 here. You see that if the input is 0, output is 0, input is 1 output is 0. So, outputs are 0 0 here also the outputs are 0 0. So, will have to write some entries here we cannot put a cross, but you see between A and S1, A is 0 0 this S1 is 0 and 1. So, we can definitely put a cross here. In this way will be filling up this table in a manner very similar to an implication chart let us see.

(Refer Slide Time: 05:38)




So, here I have shown I am just explaining. So, on the top I am showing the 2 state tables. This is the first state table and this is the second state table and let us see how we have filled this up this is the first iteration. So, we have shown A B C D along the columns and S0 to S3 along the rows. Between A and S0 let us see between A and S0 what is happening first let us see the ones where the that is entries between A and S1 A and S1, B S3 a S0. Between A and S1 you see between A and S1 B S3 A S0 I think there is a 20 just a second A and S2 I think this should be 1 there is a typographic error here this should be 1 that will be corrected.

Between A and S0 there is the cross because 0 1, 0 0 there is a cross A and S3 also you see A and S3 0 0, 0 1 there is a cross.

First look at the crosses B and S1 B and S1 B and S2 0 1, C and S 1. So, like this you just pair wise you compare and you fill up this table in a very similar manner ok. So, I have not actually verified this table you can check it yourself, let me do one thing. Let me just take this original table and forget this table, let me just work out this chart here separately.

Let me work out this chart separately. So, on this side we have A B C D on this side we have S0 S1 S2 S3, let us see from this table between A and S 0 0 0 and 0 0. So, we can put B S3 A S1 I am only writing 3 S I am not writing B S3 and A S1 A S1 between A and S1 0 0 0 1. So, there will be a cross between A and S2 0 0, 0 1 there also a cross, between

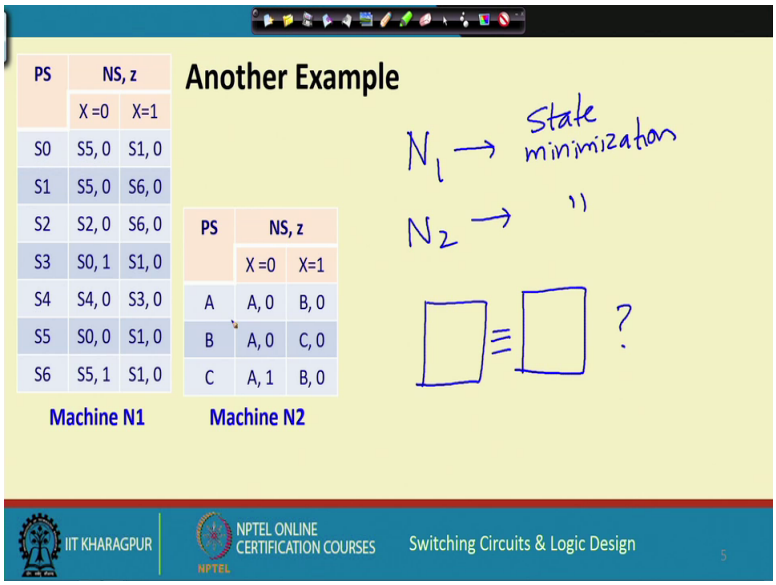
A and S3 0 0 0. So, B S2 B S2 and A S3 then B S0 B and S0 0 1, 0 0 across B S1 0 1, 0 1. So, C S3 D S0, C S3 D S0, B S2 0 1 0 1. So, C S0 D S2, C S0 D S2, then B and S3 B and S3 0 1 0 0 that will be a cross C and S0 0 1 0 0 cross C and S1 A S3 C S0 A S3 C S0 C and S2 C and S2 A S0 C S2, A S0 C S2, C and S3 that different cross D S 0 similarly D and S0 C S3, B S1 C S3, B S1, D S1 D S1 different D S2 also different D S3 C S2 B S3, C S2 B S3.

So, this is the correct implication chart corresponding with this table because these tables which are shown there are some type typographic errors somewhere. So, this you can try to figure out, but for these table the chart will be like this. Now in the next step what you do you explore this further like A 1 B 3 you look A 1, A and 1 it is already cross right. So, in the next step this will also get crossed B 2 A 3; B 2 is fine A 3 is also fine. So, A 3 is actually self. So, this you can cut it out only B 2 remains, then this one C 3 D 0, C 3 is a cross. So, this will also get crossed C 0 D 2, C 0 is a cross this will also get crossed.

Look at this A 3 C 0 A 3 is something there C 0 is a cross. So, this will also get crossed A 0 C 2 A 0 is a cross. So, this will also get crossed and lastly here C 3 B 1, C 3 is a cross this will get crossed. C 2 B 3 C 2 is cross this will also cross. So, you finally a left to the single entry B 2 right. So, this way you can find out which pair of states are actually equivalent. So, if you say it only between say this means that only you can find an equivalence between state B and S2, the others are not equivalent. So, we can say that machine N 1 and this machine N 2 they are not equivalent ok.

But the example that I showed here this was for some other state table. So, I leave you exercise for you to find out that were this mistake was in the state table and for which state table this is coming and the finally, equivalence states value like this right, but the process is like this process is very similar, just using the implication chart you work out and finally, whatever is left in this table that will tell you which state pairs are equivalent and if you confine like in this table shows, that four such correspondences for the four states, then you can say that the states are equivalent right.

(Refer Slide Time: 13:15)

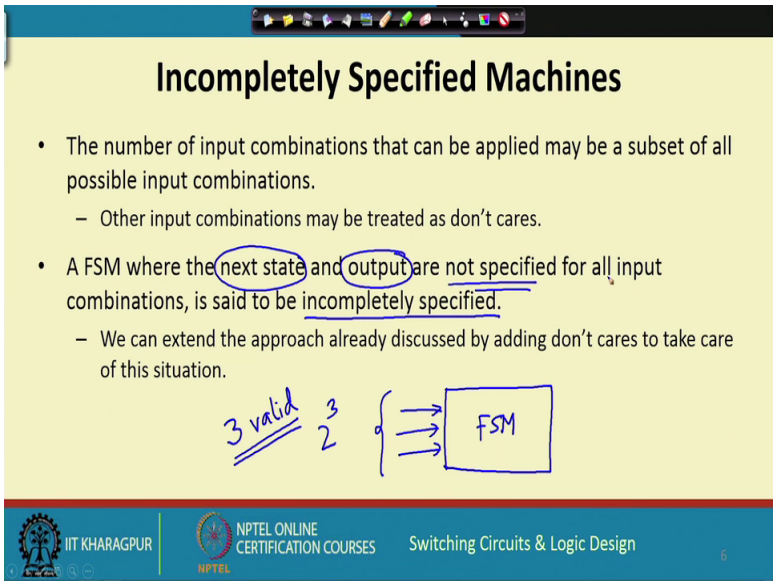



Let us take another example I am not working it out, but I am actually telling you how to do it. Suppose I have a problem like this, there are 2 machines where I can see in 1 machine there are 7 states other machine there are 3 states and someone asks you to check whether the machines are equivalent or not. Well at first look you may tell that well because the number of states are not the same so they cannot be equivalent.

But the next question is the 2 machines that are given to you there may be redundant states in those machines itself. So, you have not done any minimization. So, always your first step will be to reduce the individual machine as much as possible and then you check whether the number of states are same or not. If they are not same no need to check if the number of states are same then only you follow this process and test whether there is an equivalence corresponding to state pairs in the 2 machines.

So, here the process will be like this for the machine N 1, you will have to first carry out a state minimization; and the same thing will have to follow for state for the machine I mean N 2. So, after you have minimized it then whatever machines you get is 2 state tables. So, if you find that the number of rows are the same, then you follow the process and test whether they are equivalent or not right. So, I leave this as an exercise for you, you can basically try this out and check whether you are getting this equivalence or not ok.

(Refer Slide Time: 15:22)




So, whatever you have seen so far, concerns the equivalence of states in a FSM or in general equivalence of 2 FSMs totally. Now what we have not mentioned so far there can be some instances some problems, where there can be the notion of some don't care inputs some inputs or some output values can be don't cares. So, while we are doing state minimization how do you handle don't cares. Let us take an example to illustrate this and such specifications where there can be don't cares are in general referred to as incompletely specified machines.

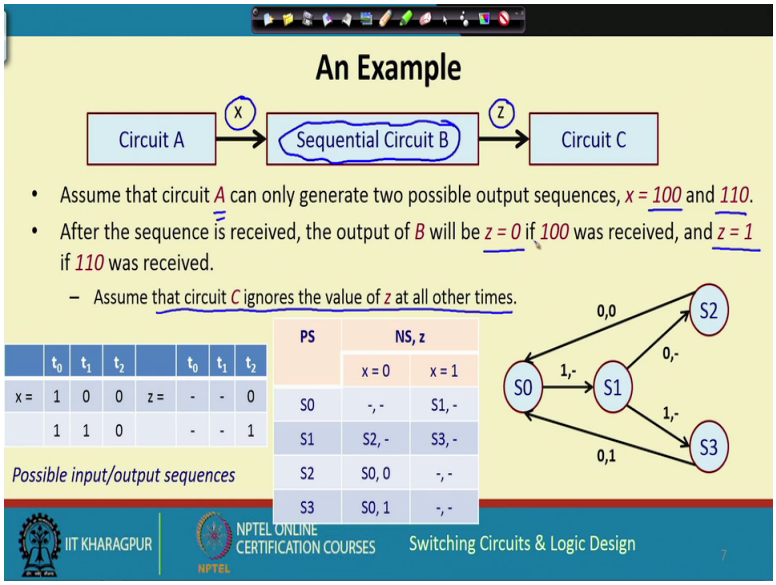
So, this specification is not complete, some of the information like inputs and outputs are not mentioned they are called incompletely specified machines. Now what do mean by incompletely specified machines? There are two things first is that; the number of input combinations that can be applied can be a subset of all possible input combinations. Like for example, I have a circuit let us take an example let say there are 3 inputs. Because there 3 inputs there is an FSM that we are trying to design this an FSM. Because there are 3 inputs there can be 8 possible input combinations 2 to the power 3, but let us assume out of these 2 to the power 3 only 3 input combinations are valid which means during normal operation only one of these 3 input combinations can come.

So, when you are designing the FSM the remaining five inputs will be regarded as don't cares. So, I can suitably fix them depending on my convenience when I am doing state minimization. So, they can be set to 0 or 1 as per my convenience ok. So, other input

combination make it treated as don't cares. So, in general we say that an FSM is incompletely specified where the next state and also the output values are not specified for all input combinations, for some of them they are not specified. But we have seen earlier during state minimization, we are looking at the outputs and also the next states. You see more states you can club together that is better for us because our FSM will get smaller we can find out equivalent states.

So, the next state and the output you can suitably set during the process of minimization. So, that more states can be merged together and the FSM can become smaller, this is a basic idea illustrating with an help of example ok, but the idea is basically this let us see let us take an example.

(Refer Slide Time: 18:49)




This a very simple example I am just illustrating it. Suppose we are trying to design this sequential circuit B ok. Now the input of the circuit is coming from some other circuit let say A and this input we are calling as X and the output is going to some other circuit C this output you are calling as z right. Now how this circuit is working? Let us say that this circuit A is generating a serial bit stream. So, X is not a parallel number serially some bits are coming one by one, 1 bit at a time in synchronism with the clock the bits are coming.

So, I am assuming that this circuit A can generate only 2 possible output sequences bit patterns 1 0 0 and 1 1 0. And this sequential circuit is designed in such a way that when 1

0 0 is received z will be set to 0 and when 1 1 0 is received z will be set to 1 now you understand this situation. I am trying to design a circuit which is being fed with 3 bits in sequence and out of the 8 possible combinations only 2 of them are valid they are 1 0 0 and 1 1 0. Now in one case the output should be 0 if 1 0 0 is received and the other case the output should be 1.

So, for the all remaining 6 cases because such inputs will never come so I can as per my convenience I can set the output to 0 or 1 whatever, because it will not matter because those inputs will never come. And the other thing is that the output will become 0 or 1 once this entire sequence is received, but before that it does not matter the output can be either 0 or 1, that also I can set according to my convenience. Only at the end of the third cycle the output will denote that whether 1 of those 2 strings have been received or not 0 or 1 ok.

So, what I am saying is that assume that the circuit C ignores the value of z at all other times, only when this 3 ones have been received then only the output will be either 0 or 1 depending on which of the 2 sequences have been received. So, the possible input output sequence I can depict like this, the first circuit is generating x which can be either 1 0 0 or 1 1 0 the others can never come. And the corresponding z output only at the end it will be either is 0 or 1 this t 0, t 1, t 2 indicates the times the clocks, but before that these 2 states are don't cares I do not I do not care what is coming here, but only I care what is the final value right.

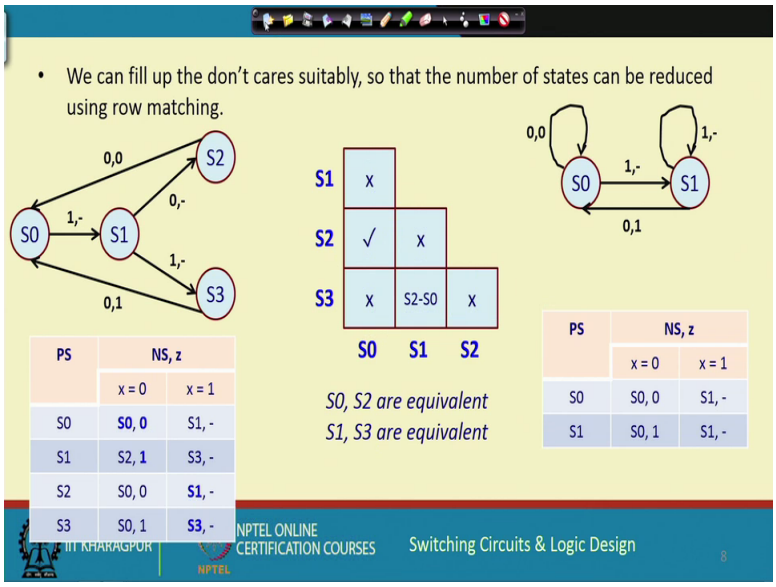
Now, depending on this I can create this state table or first I think need to be easier to show the state transition diagram. So, here I require four states because I need to identify 3 bit sequence, I start with an initial state suppose I get a 1 I move to state S1, but when I get the first one the output can be don't care that is why the output is dash, dash means don't care.

Now, I am in state S1 when the first one is received. Now if the next bit is 0 then possibly 1 0 0 is coming I move to state S2. So, again the output is don't care, but if the next bit is 1 I move to S3 because possibly 1 1 0 is coming 1 1 again the output is don't care, but if 1 1 0 has actually come then only my output is coming to 1, and if 1 0 0 has come then also output will be 0 right.

So, the output is being generated only in these 2 cases. So, all other you can see they are don't care combinations just like for example, from S2, I am only showing an output transition when there is a 0, but what will happen if the input is 1 this is don't care. So, I can put any next state as per my convenience right these are the places where I can play with and for this state transition diagram the corresponding state table is this. So, directly this is a one to one mapping for S0 if x is 0 nothing is mentioned that is why dash comma dash it is not shown.

Next it is also undefined or don't care output is also don't care. But only for x equal to 1 1 comma dash; that means, S1 comma dash for x equal to 1 S1 with dash similarly from S1 there are 2 states from 0 or 1. So, that 2 entries from S2 there is single for 0 for 1 not specified similarly for S3 ok.

(Refer Slide Time: 24:58)




Now, from this state transition diagram or state table now we can fill up the don't care suitably so, that we can carry out minimizations effectively. Now all these dash which were there. So, here in this blue that I have shown bold, these are the ones we have filled them up suitably like this dash we have filled up with S0 this dash with S1 this dash with S3 so, that some states can be made equivalent.

Similarly, this output has been made 0 this output has been made 1. So, this you can arbitrarily fill up and your objective will be to make as many states equivalent as possible. So, accordingly the next states and outputs you can set to either 0 or 1. So, I am

just showing an example here now in an ad hoc way I have filled it up ad per my convenience.

So, with this assignment let us see how the implication chart will look like. There are four states S0 S1 S2 along here S1 S2 S3 along here let see S0 and S1. S0 and S1 you say that they are differing here 0 1 in their output. So, we have put a cross S0 and S2. See since S0 and S2 I have put a tick just to make them equivalent you see there was this was S 0 0, I had made it S 0 0 just to make them equivalent similarly S 1 dash I have made it a S 1 just to make them identical, because they are now identical you can mark a definite tick here and S0 S3 0 1 that different so this is also x.

S1 S2 S1 and S2 1 0 they are also incompatible so, a cross S1 S3. So, S1 S3 1 dash 1 dash outputs are same, but S2 S0 S3 same S3 S3 S2 S0. So, I have marked it S2 S0 and S2 and S3 is also different 0 1 x. So, we have to obtain the implication chart like this. Now after we have obtained like this let us look at S2 S0, S2 S0 are also equivalent S0 S2. So, this 1 also you will be changing it to a tick. So, finally, S0 and S2 will be equivalent and S2 and S3 also will be equivalent. So, S0 and S2 you can merge together into a single state, S1 S3 you can merge together into a single state.

So, first let us look at the state table. So, if you merge S0 and S2 together you see S0 remains same S0 0 and S1 dash S2 is merged with it. So, S2 has gone so S1 S3. So, S1 is S2 1 now S2 has become S0. So, it will become S0 1 S3 dash S3 is now gone S3 has become S1. So, it is S1 dash and S3 is also same. So, this is the reduced table and the reduced state transition diagram will look like this, there will be only 2 states from S0 of the input is 0 you remain in S0 with an output 0 0 0, if the input is 1 you go to S1 with output don't care when you are in S1 if the input is 0 you remain in S1 you go to S1 this edge with the output 1, but if the input is 1 output is don't care you remain in S1.

So, for this specification of the problem this FSM can be reduced drastically in size as you can see, and the final design will become much smaller. So, essentially what we have seen is that, in the process of state minimization given FSM state table you can reduce size; we can use this same principle to identify with 2 FSM equivalent or not. And lastly we have mentioned if the FSM is incompletely specified there are many examples where we do have such incompletely specification, then we can suitably set the don't care entries in the state table such that as many of this states can be marked as equivalent

in the example that you cited only we could made 2 or 2 such states were made equivalent by suitably filling them up. And the net result is that our final FSM specification becomes simpler and our circuit becomes much smaller right.

So, with this we come to the end of this lecture and we have completed our discussion on minimization of finite state machines. Now in our next sequence of lectures we shall be looking at the design of some very specific finite state machines or sequential circuits namely resistors and counters, which are very useful building blocks in designing complex systems. So, this discuss we shall staring in our next lecture.

Thank you.