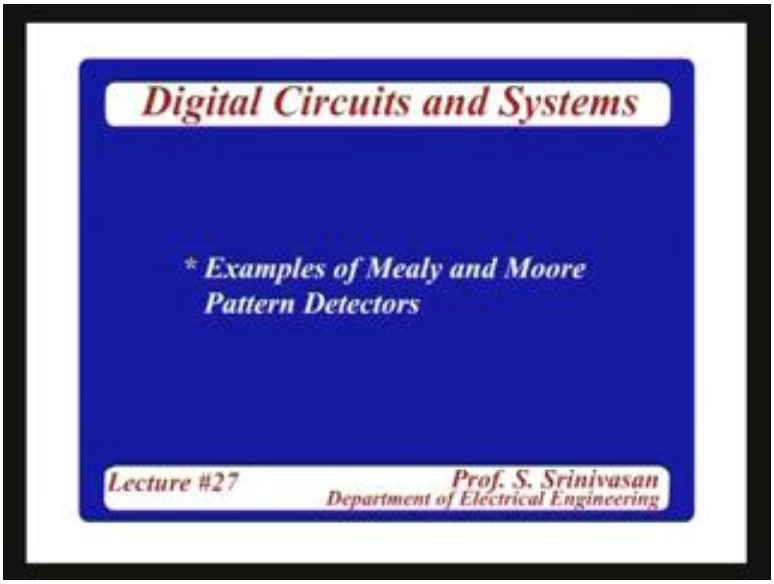
**Digital Circuits and Systems**
**Prof. S. Srinivasan**
**Department of Electrical Engineering**
**Indian Institute of Technology, Madras**
**Lecture - 27**
**Pattern Detector**

(Refer Slide Time: 1:43)




So, we talked about Moore machines and Mealy machines based on the outputs. In a Mealy machine a Mealy state graph the output at any state is defined by the state at the input. So in a given state there can be more than one output because there are more than one input possibility. On the other hand in a Moore machine the output is tied to a state. A particular state will have only a particular output and the output will change only when the machine changes from that state to another state. For example, if your machine is in state $S_0$ and there is an input variable called x so in state $S_0$ if x is 0 the output can be 0. If x is one the output can be one. On the other hand in a Mealy machine a Moore machine we will say at state $S_0$ the output is 0 the output has to change so it has go to another state. So we will do some more machines the drawing of the state graph.

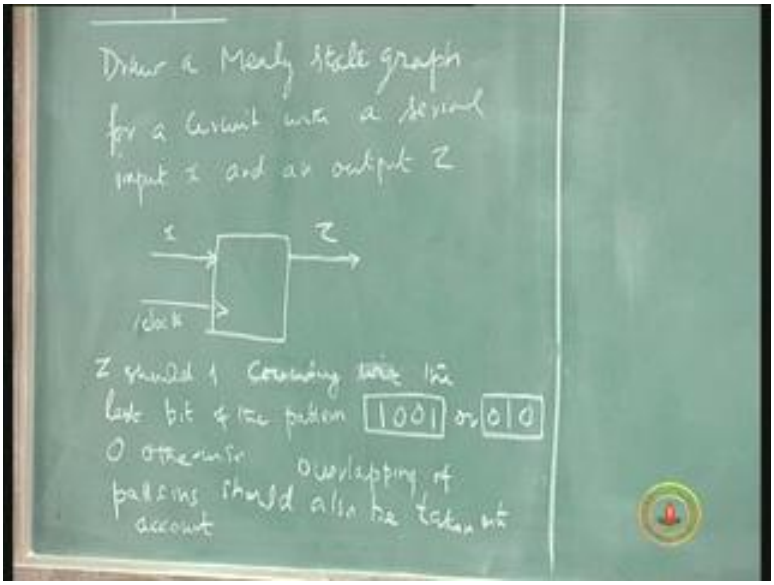
As I said that is the most important step, the reason I want to spend a little extra time in this is this is a most important design step. If you have understood the problem very clearly if the problem has been stated clearly to you by the user the client and you are the designer first of course you have to understand the problem clearly of course that person has to state the problem clearly to you and ask all questions to understand that there is no ambiguity, that is one certainty. Having said that the first step is to draw the state graph.

Once you have the state graph the rest will fall in place. That is why I thought it's important and the designers job is to get a best state graph possible for a given set of

specifications. From then on to the rest of the design procedure is simple because this is a standard procedure. Anything which is standard can be simplified and also even automated. Today you have the age of computers and automatic design Computer Aided Design they call it CAD or they call it e d a electronic design automation these are all the industry terminology for automatic design but all that will come only after you have got your state graph.

Of course the state graph also the computer can generate but then it may be sub-optimum the designers creativity designers innovation the designers specialization can be shown only at this stage of drawing a good state graph, that's why I thought we will spend some extra time and give you one or two more examples. Of course we will also do systems later on and there also we will have to start with state graphs but here at this point in time in our course since we are talking about state graphs just taking a state graph's example without any sort of overall system view I will just define a system with an input and output and the conditions for which these outputs has to be there and then we will have to draw the state graph. So let us take an example of a Mealy state graph.

(Refer Slide Time: 9:04)




Draw a Mealy state graph for a circuit with one input x with a serial input x and an output Z that means at every clock pulse at the edge of the clock a new input bit is sent into the hardware that means there is an input which comes, it can be same also, it is sampled during every clock transition. the input is sampled at every clock transition, and if the value is changed it will take a new value and if the value is not changed it will still take the value as the old value so it will be a series of 1s and 0s. and then you want an output Z is also of series of 1s and 0s and then the relationship between x and Z will be the following.

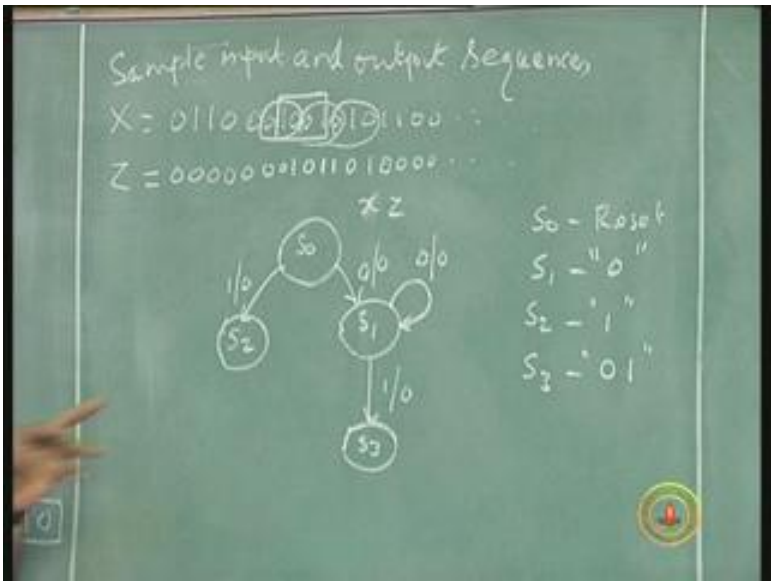
Whenever there is a pattern, we did the pattern generator, Z should be 1 coinciding with the last bit of the pattern 1 0 0 1 or 0 1 0 either this or that happens so if the input bit has

a pattern with 1 0 0 1 along with the fourth one output would be 0 0 0 1 here. Similarly the input is 0 1 0 the pattern 0 1 0 is detected by the circuit and output Z should be 1 corresponding to this 0 during that clock period and for the rest of the time the output is 0. So we will also assume that overlapping is allowed, overlapping of the two patterns or the same pattern repeated also has to be considered as completion of the pattern, overlapping of the patterns should also be considered as (Refer Slide Time: 8:43)
that means there is a series of bits 0 1 1 0 0 0 1 0 0 1 0 1 0 1 1 0 0 and that goes on, so this is the sample input and output sequence.

We are looking for this pattern 1 0 0 1 here or 0 1 0, there is 0 1 0 here, there is 0 1 0 here, 0 1 0 here (Refer Slide Time: 10:05) so Z should be 0 0 0 0 0 0 0 1 0 1 1 0 1 0 0 0 0, this one is for completion of this 0 1 0, this one is for completion of 1 0 0 1, this one is for completion of 0 1 0, this one is for completion of 0 1 0. It is a similar example to what we did in the last lecture but slightly more involved in the sense there are two patterns one is a larger pattern and then the overlapping is also there. So, we wanted a Mealy machine that means from each state you will have to look at the possibility of x being 1 and x being 0 and you have to decide the next state and the output for each of those.

So let's start with always 0 state reset set you call it $S_0$, the input is 0. If an input is 0 output is 0 so it is x and Z, output is 0 go to state $S_1$ so $S_1$ state is a state where pattern 0 as been identified. It's 1 from another state, x is 1 Z is 0 output is 0 and in this state 1 would have been registered or memorized or recognized whatever you want to call it. Then $S_1$ again can be 0 or 1 and if it's 0 it's not going to give you any extra information so it can be the start of a new sequence 0. And on the other hand if it is 1 it will go to S3 because we have now identified a 0 followed by a 1 which could be the first two bits of 0 1 0 pattern.

(Refer Slide Time :12:50)



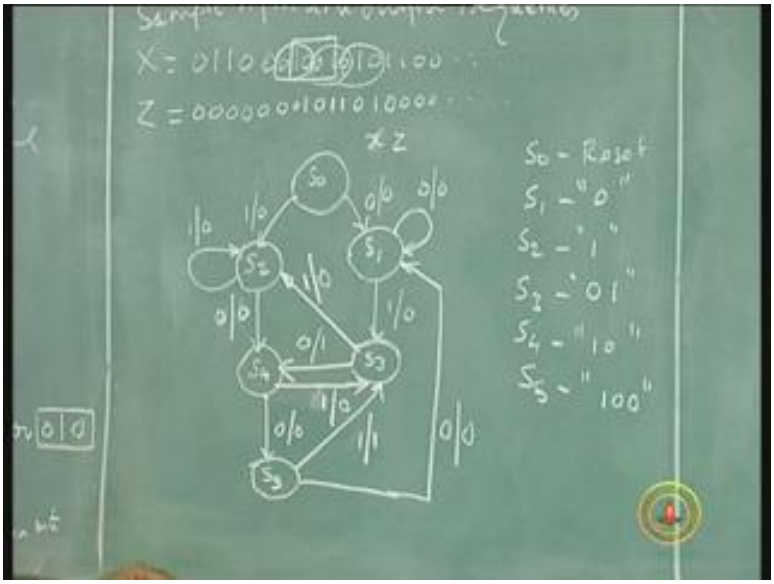
Likewise in $S_2$ if 1 comes it could be the beginning of a new pattern so no extra output it continues to be in the same state. And if a 0 occurs on the other hand it will be 1 0 it could be the first two bits of 1 0 1 pattern because when you want to detect both of these 1 0 0 1 and 0 1 0. So if 0 occurs let us create a new state called $S_4$ in which 1 0 will be remembered. That means up to this point of pattern 1 0 as occurred and that's what it means.

Hence you have to proceed like this for each state until you have your patterns and no more possibility is there. So in $S_3$ 0 1 if 0 occurs 0 1 0 pattern is completed and that could be also part of 1 0 0, 0 1 0 could be the first two bits of 1 0 0 1 pattern which is already here in $S_4$ so if 0 occurs here you go to $S_4$ which is 1 0 but you produce an output of 1 in this case because we had completed it as pattern 0 1 0. The x is 0 leading it to 1 0 state and a new 1, 1 is the output because 0 1 0 is now completed with this.

On the other hand if 1 occurs here it's 0 1 1 but 0 1 1 has no meaning and only the last bit 1 is of any value so that is in this state $S_2$. So 1 occurs here (Refer Slide Time: 14:55). So we exhausted $S_3$ and also finished the pattern 0 1 0. Now let us continue with S4 and see how to complete the pattern 1 0 0 1.

$S_4$ is 1 0 so if 0 occurs here that will be part of 1 0 0 1 so I will have to create a new state for that $S_4$ with 0. We can put a new state called $S_5$ which will remember 1 0 0. There is no output again because 1 0 0 1 only when that is completed we will have an output of 1. When 1 occurs in $S_4$, $S_4$ is what 1 0 so if 1 occurs there is 1 0 1 and the first two bits will be 0 1 0. So if 1 occurs in $S_4$ it can go to 0 1 which is $S_3$. So $S_3$ and this is 1 0. Now $S_5$ and in $S_5$ if 0 occurs then it is 1 0 0 0. So 1 0 0 0 has no value except that the last 1 0 is a value which is $S_1$ so if 0 occurs then it is (Refer Slide Time: 17:00). On the other hand if 1 0 0 I if 1 occurs here the pattern is completed and the last 1 0 1 of that 1 0 0 1 could be the part of the pattern 0 1 0 so $S_3$ Z 1 0 0 1 $S_3$ so I will have to go here. It is 0 with an output of one because it has completed just now the sequence of 1 0 0 1 1 is the input and 1 is the output.

(Refer Slide Time: 17:39)




Of course from here we will have to the state graph state table and all that, we will talk a little bit about it. but one thing I want to say here is the problem is simple, two patterns and you can mentally sort of visualize the whole thing but the problem becomes more complex. It's not a pattern detector but some other problem or even a longer pattern or more than two patterns. You will lose track of some of the combinations that's already been there. Of course you take notes here so do not worry about having creating an extra state. thumb rule of any design is don't worry about extra hardware to start with. To start with you make sure that all requirements are taken care of, all specifications are covered completely. You can have hardware which is redundant.

Of course it is not efficient in terms of speed cost power but that is better than having a hardware which will not work properly which is inaccurate, which will give an erroneous behavior so when in doubt create a new state. Supposing you are in a particular state like $S_4$ and then 0 occurs whether it should go to one of the existing states or it should create a new state of course if there is a simple thing like that you can always think for a minute and visualize it and go there.

But in case if the diagram becomes longer, bigger and you are not able to really check all the possibilities into account you can as well create a new state and proceed so you may end up with more than the minimum number of states that's not a serious difficulty even though more states means more flip-flops and more flip-flops means more driving logic and more driving logic means more and more hardware that takes more space, cost, more money and consists more power. All that is fine but your system will still work because you are taking care of all specifications. so do not be over worried about simplifying. of course it doesn't mean you should design inefficient circuits all the time but when in doubt create a new state.

Later on of course there are techniques to make sure whether this is indeed the simplest possible state graph, any extra states can be knocked off by a process similar to this Karnaugh Map.

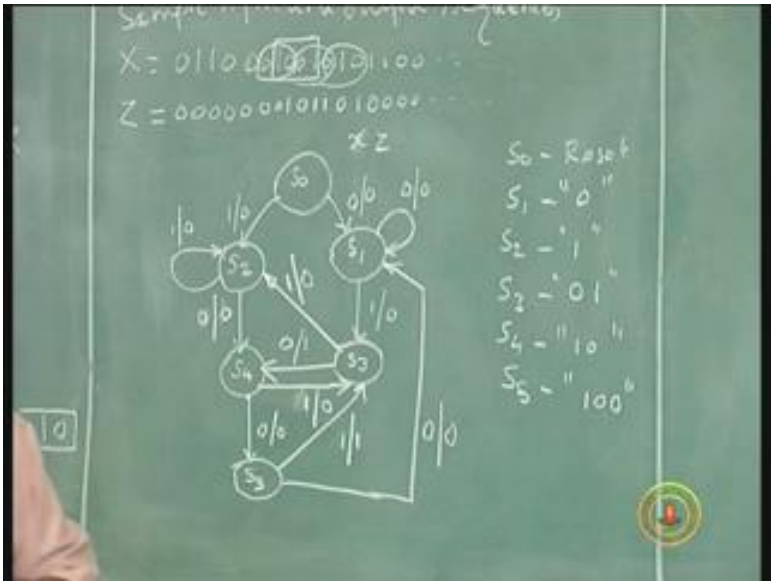
We went through the Boolean algebra, reduction of the terms, reducing the term to the minimum using Karnaugh Maps, similar concept is also possible in state reduction. so once you have a state graph you can reduce the state graph to the minimum possible number. A state graph is a minimum possible number of states. Later on I can show you how a state graph with more states than required can be reduced to a state graph with a minimum number of states there is a technique for that.

Even if the technique is not available I would rather go with a conservative state graph which will take care of all my possibilities rather than going and taking I will draw a simpler graph and then ignore some of the possibilities. Between these two options I will any day prefer the option in which I will have more states but I can solve the problem to the entire specifications, that is one point I wanted to make here.

State reduction to minimum number of states that's a technique of doing it. Many text books talk about it now-a-days. If possible we will cover it even though I don't want to do it at this stage.

The next thing is once you have number of states you must know how you proceed. You have to assign the state now. there are six states 0 to 5 there are six states, for six states a minimum of three flip-flops are required so you can sort and call these variables A B C or P Q R whatever and then start giving 0 0 0 0 0 1 that is an arbitrary assignment. This is called an assignment. Once you have a state graph which states symbols other than binary values converting the states from symbols and letters into binary variables is called a state assignment.

(Refer Slide Time :22:29)



So one is reduction of states, reduction of state graph, these are steps you need to do which I said I am skipping at this point in time, second is the state assignment. State assignment is a process of mapping or assigning binary values to the states. I cannot start with 0 0 0 0 0 1 because I never know when I start a problem how many states are going to be in total so how many binary variables I should get started with I don't know otherwise I would have started with 0 0 0 here then I will go to 0 0 0 state will go 0 0 1 state 0 0 1 state will go 0 1 0 state so I can make a graph like that but I don't know because I don't know how many variables I have to start using so I would probably start with symbols and then later on after finding out how many states are required I will assign.

Now I know how many state variables are required. Now I know that three state variables are required so I am going to call them P Q R whatever. So I would say $S_0$ is 0 0 0 and S1 is 0 0 1 making this assignment etc. This is called state assignment. this step of converting the symbols state symbols into binary values is called state assignment. Here again I can do it a little more efficient, this is an arbitrary assignment, natural binary sequence I have taken and assigned it.

Finally the state values become the present state values and then the next state values depending on the state graph and from there you go to the flip-flop inputs and outputs so the Karnaugh Map is drawn based on the transition from the present state to the next state. It is the simplification of the hardware finally. The hardware that we are going to finally get, the steering logic we are going to get depends on the present state and the corresponding next states. So from $S_0$ the present state and $S_1$ the next state what is the best possible assignment of $S_0$ and $S_1$ so that a minimum hardware is required for transition, this is also a problem of interest in reduction hardware. So there is a possibility of optimally assigning the binary values to the states rather then randomly or arbitrarily assigning as we did in this case, can I make a more efficient assignment, hardware efficient assignment. by this what I mean is instead of blindly saying $S_0$ is 0 0 0, $S_1$ is 0 0 1, $S_2$ is 0 1 0 instead of saying that can I use some sort of a rule and figure out which state should be next to which state.

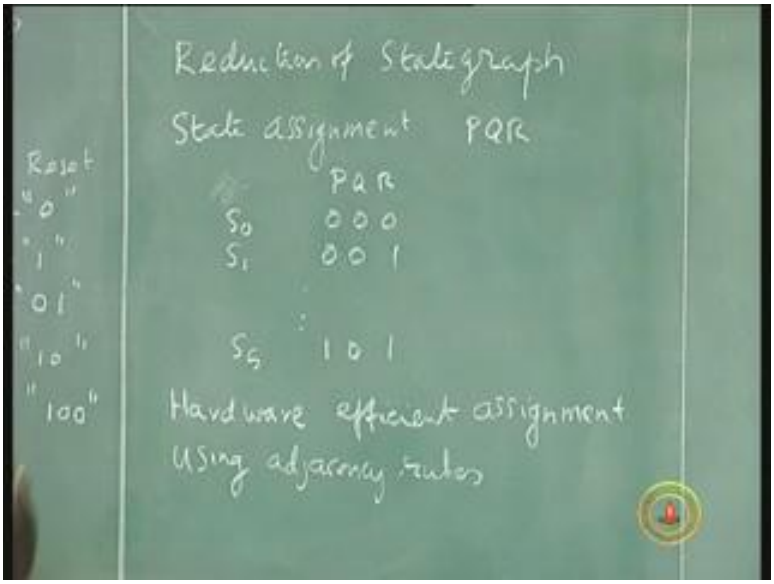
Whenever the state assignment of one state differs from another state in only one variable these are called adjacent states. A 0 0 0 and 0 0 1 are adjacent states like the Karnaugh Map. In Karnaugh Map adjacent cells differ only in one variable.

Like that in binary assignment also adjacent states are the states which differ in only one value of the state variable and other state variables are the same so use some adjacency rules. this is also possible. These are steps to reduce the hardware. This is very important when it becomes for example from nine states I am able to reduce to eight states it’s a significant reduction because instead of four flip-flops I can use three flip-flops. State reduction becomes more meaningful if I can reduce the number of flip-flops. Even otherwise it will be meaningful because if I have less states instead of six states I may have four states even though in both case have used three flip-flops I will have more don't care values when the number of states is less and when I have more don't care values my

hardware is simpler you know that. when I have more don't care entries in my state table my hardware is simpler. So to that extent it will also impact the hardware but more significant reduction is possible if you can even reduce the number of flip-flops.

The same argument I can hold here, if you have assignment states such that the transition always happens from one state to the next state which is differing in only one variable the switching activity or the transition is minimum that is also efficient. Again I am going to skip this part of the thing this point. I want to tell you these are the things which are possible which can be done and which should be done for efficient design but we will have to first go through the design procedure and then later on if I have time we will do it.

(Refer Slide Time: 27:50)




Why I am not giving lot of importance to these at this point in time it is because in today's technology nobody bothers about hardware because gone are the days when you have to count the number of gates. If you can do something with seven gates instead of nine gates, seven gats instead of eight gates you are very happy I will reduce one NAND gate. today as I said these ICs Integrated Circuits which have a large number of gates inside then on one single chip I can have a gate equivalent function even though we call it gates it does not mean that thousand gates will be sitting there.

There are functional parts functional blocks within that IC which are capable of delivering the performance of above thousand gates. So what is the point in having a gate with a performance like that and thousand gate and then reducing it from 767 to 734. You can only say I have made it less. Of course little bit increase is there in terms of switching activities less so to that extent there is a power saving but the size is the same and IC is sitting in a board so there is some marginal advantage. Anyway there is some marginal advantage. but the amount of advantage in terms of cost and the size and even the power consumption is not as significant as the earlier days where each one is a separate IC. I

have to buy and put an IC, one AND gate means one extra thing I have to put, for one multiplexer I need to put one extra thing, one flip-flop I need to put an extra thing so for each one I have to go and buy an IC and put it there it takes more space, more power, more money, more everything, speed is reduced so those days hardware reduction was very significant.

Today it is we say thanks to the technology advancement called VLSI technology, advances in VLSI technology. It is so common that everybody talks of VLSI even without understanding what it is, they simply talk of VLSI. We have newspapers which talk about VLSI courses. because of that this hardware is available in plenty, it is not very important unless it's a critical issue.

For example, if I have thousand gates solution nearly thousand gates solution which is slightly exceeding thousand gates I will try to accommodate in a thousand gates solution other than the next possibility which is two thousand gates. I would rather do some exercise to reduce it to less than 1000 so that I can use a smaller IC than a 2000 gate IC. So these are certain critical areas or certain speed issues when it comes to high speed applications. Today technology both speed-wise and hardware availability-wise there is plenty we don't have to worry about the speeds of normal applications except in very high speed applications, high speed computing. Likewise hardware availability is enormous, cost saving is minimal if at all there is any. because of these things only we are skipping these steps now there is no point in doing it.

But at the same time as a designer if you want a digital design you should know that it is available here, when you need you should be able to apply it. You may be an IC designer, later on you may doing a very sophisticated IC in which every component is critical in the sense of area and space and everything so you have to go and start optimizing block by block you should know what are the techniques available to you for that so I don't want to sort of not let you know that there are things like this which could be done but it is not important especially in small systems like this. Talking about a pattern generator with three flip-flops and a bunch of gates the whole thing is over which can be accommodated in one small integrate circuit a Medium Scale IC M S I the whole design is one M S I. I buy a M S I and put it there and it works so why bother about trying to do some of these things. that is the reason I am skipping some of these steps now at this point.

We will do an arbitrary assignment most of the time until we have time to revisit these areas, let's try to do some of these things if possible. So we will do a normal assignment. from here you know how to do it that means I have do a state table which can be also considered as a transition table depending on the type of flip-flops. So I have to say present state input next state output, and the present state values are let us say PQR and next state values are P power plus, Q power plus, R power plus and let us use D flip-flops so that PQR are same as DP DQ DR and for other types of flip-flops I should be able to use a separate thing. Hence this state table is also a transition table. The output is Z. So we will have 0 0 0, 0 0 1, 0 0 0 0 and the last state is 1 0 1 and if x is 0 what happens, and

if x is one what happens and the other two states are not being used. And this row fully is don't cares (Refer Slide Time: 34:33).

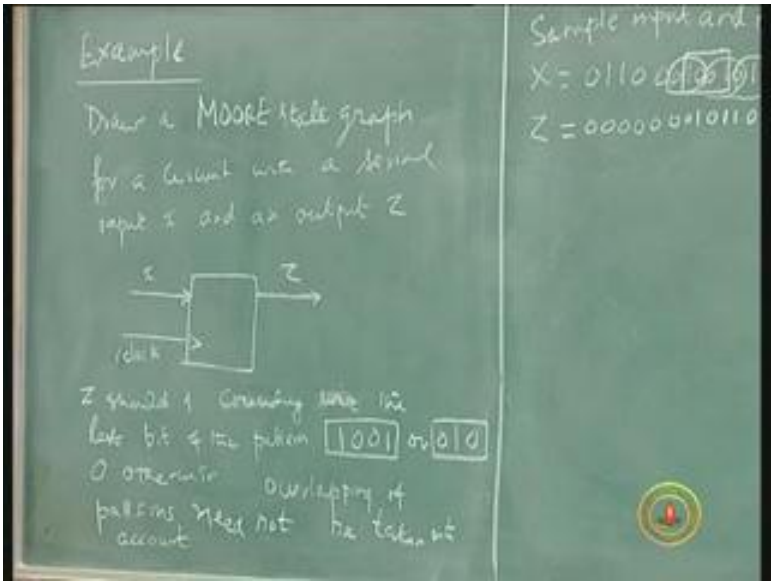
Remember that in the don't care also the output should be 0s. so you can complete this table. you know how to do that from here, this is 0 0 0, this is 0 0 1 so 0 0 0s I will write one entry, in 0 0 0 x is 0 it goes to 0 0 1 output is 0, in 0 0 0 input is one it goes to 0 1 0 and so on. Please fill this and design the circuit using D flip-flops and gates. That is an exercise you can do it yourself, you have done enough of those by now.

This is how you start from a basic problem definition with very clearly stated specs. Take all possibilities into account draw the state graph, from the state graph you do the state assignment, reduce the state if necessary and if possible reduce the number of states, give a state assignment either randomly, arbitrarily or using some thumb rules or design rules to make it efficient and from there you go to the state table and design the type of flip-flops, if the flip-flops are of the ordinary type then you draw a transition table which is other than the state table and from there you draw the Karnaugh Maps and simplify and then draw the gates, circuits the steering logic using gates and the flip-flops which will be driven by this steering logic and then your circuit is over and then you start giving the patterns and then test it, so build it and test it and then it works. This is only a sample sequence you don't have to give the sample sequence, we should give any sequence. whenever any sequence you give this pattern is identified and it should immediately give an output of 1.

This is total procedure with a slightly involved I wont say it is a very complicated problem slightly involved compared to what we did in last lecture. We will do one more example of the state graph generation this time using Moore machine. We have not used that so far, we have done only Mealy. So let us do a Moore machine state graph and the rest of the things is easy. Once you have a state graph you know the state assignment and I am not going to talk about all these things one more time. We will have a state graph and from there you know the reduction is necessary then you go to the state assignment these are arbitrary or efficient assignment and then this logic design using hardware. Maybe we will use the same problem and instead only change this to MOORE here (Refer Slide Time: 37:39).

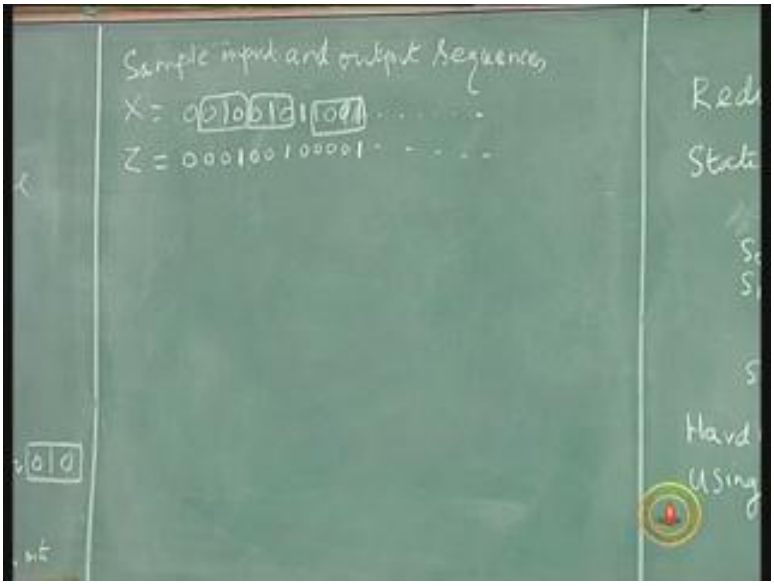
This time we will not identify overlapping patterns, I will give you two variations. That means I should clearly have a 1 0 0 1 output will be 1 that 0 1 will not considered as a part of another sequence and again it as to start with 0 0 1 0 only then it can be a sequence pattern so non-overlapping so I am going to reduce this, overlapping of the patterns need not be done.

(Refer Slide Time: 38:29)




Design a Moore machine again for the same patterns and do not have to worry about overlapping. We will always assume that when a pattern is completed it starts all over again. That means after every output 1 you have to go the reset state and start all over again. That's what is non-overlapping. You start with a reset state, the pattern is completed and since no overlap is allowed you have to start with the reset state and again wait for the next pattern to come.

(Refer Slide Time: 41:05)



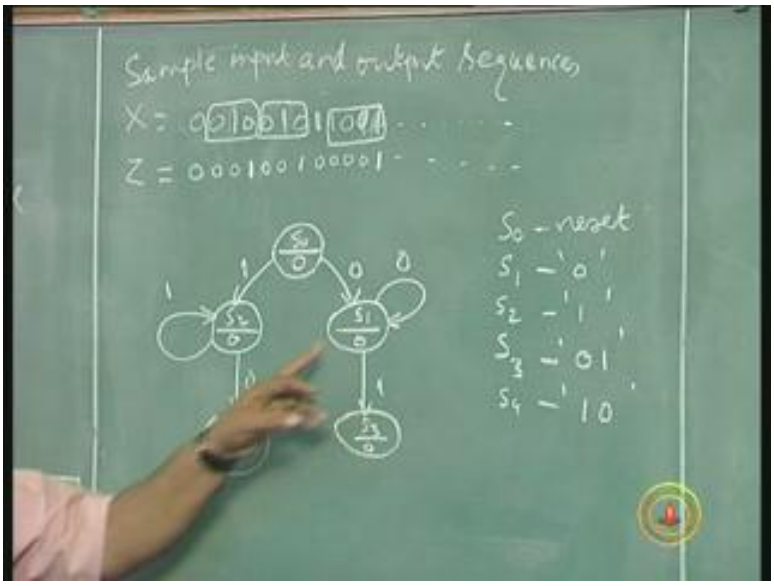

Now if you want to use the same sequence as an example, now 1 0 0 1 is identified here but this 0 1 0 I will not recognize as a part of a pattern because it's non-overlapping so

this 1 0 0 1 will produce an output of 1 until then the output will be 0. ((Refer Slide Time: 39:55) correction in slide) z is 0 0 0 1. So we have an output here because of this pattern. Now this 1 0 0 1 gets knocked off then this 0 1 0 will also be identified since it's a separate pattern non-overlapping.

It's only a sample sequence it can be anything and then 1 0 0 1. This type of input output if you want. We have two variations in the problem that I have made. One is that we have a Moore machine not a Mealy machine that means the output will be decided in the state and not along the arrows the second thing is the reset of the pattern is detected and then will start all over again. We will have to quickly draw the state graph and the rest of things you will do it yourself as homework. Complete this also as a homework.

So let us start with $S_0$ the reset state 0. now I don't have to draw the arrow, this and this but i have to draw the output within this state. So usually you put $S_0$ that means in this $S_0$ state the output is 0 that's what it means. Along the arrow we mark the inputs and inside the state we mark the output. So we go to $S_1$ and $S_1$ is a 0 state remember, 1 $S_2$ one state members and now here it can be 0 or 1 and if it is 0 it continues to be in this state as long as any number of 0s is there we have a $S_1$ assuming that the latest 0 is the only 0 so 1 will go to this (Refer Slide Time: 42:50), $S_3$ is 0 1 state. In $S_2$ if 1 occurs continue to be here and if 0 occurs it will be 1 0 which is again new. Then you are in $S_3$ it is 0 1 so 0 occurs and the pattern is completed.

(Refer Slide Time: 43:42)



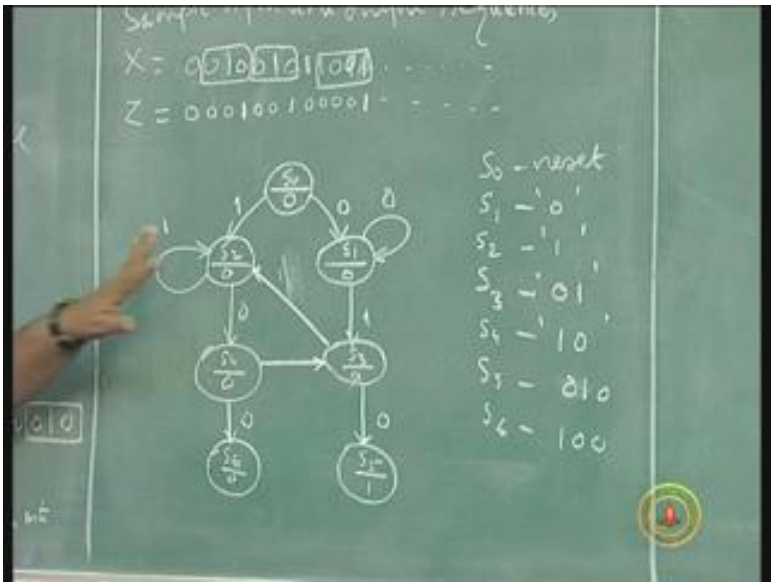

$S_2$ to 0 will be 1 0 why 0 1 0?

We have 1 already here and here there is 1. Since already there is 1 here, we are here only if 1 has occurred. if it is a fresh 0 it would have been here. This could be a new start, this could be old 1 0 0 1 or starting of a new pattern here, it can either go here or here you are saying. I think we will stick to this because we have two patterns to detect first so we will

consider that little later on separately. From here 0 occurs I would rather like to use that as a pattern 1 0 0. If it is not going to result in a pattern then you are losing that extra 0 that's what you are saying. $S_4$ to $S_3$ we are going to do it now. let us complete this.

Now in $S_3$ if 0 occurs what will happen? 0 1 0 completed it will go to $S_5$ I cannot go into any other thing because output has to be 1 here, a new state in which output has to be 1. I cannot go to one of the states that is already there because all those states are only 0s. So this is 0 state, that is $S_5$ 0 1 0 if 1 is occurring here, this is 0 1 state, this has no meaning this has to go here, the 1 occurring here will go here. Let us look at $S_4$ now. In $S_4$ if 0 occurs it will go to $S_6$ so if 1 occurs it will go to $S_3$ this is 1 0 1 so 1 0 1 would be 0 1. So that is the 0 you are worried about and that 0 has now been taken care of, this 0 would not have been lost now. You don't have to return it there because it would anyway go to the $S_3$.

(Refer Slide Time: 47:10)

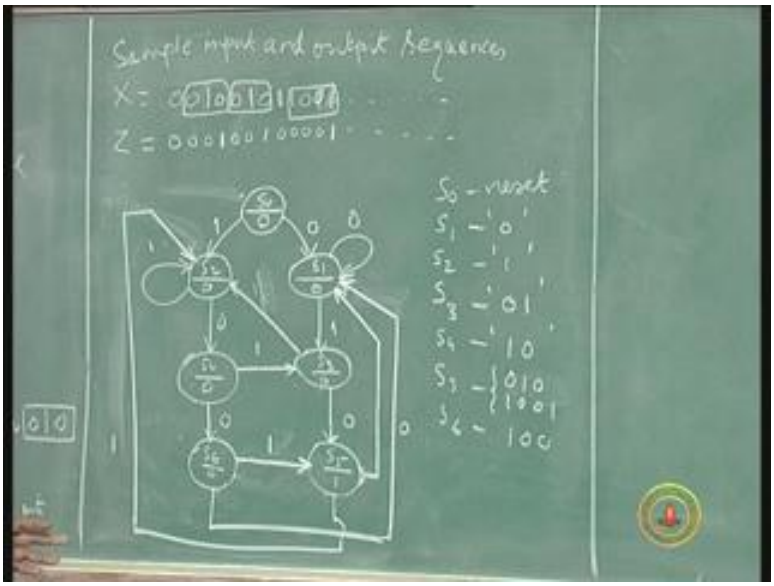



So this is 0 now $S_3$ and this is a1. So 1 we consider both arrows here, now $S_6$ is 1 0 0. in s six if 0 occurs there is no use because 1 0 0 0 has no meaning except the last 0. if on the other hand in $S_6$ a 1 occurs the pattern will be completed the output will be 1. So in $S_6$ if 1 occurs we will go to $S_5$ make an output of 1 and since there is already a state with output of 1 we will put this output here. That means $S_5$ is a state where we will have output 0 1 0 as well as 1 0 0 1 with both having an output of 1. And if on the other hand as I said if 0 occurs this 0 has no meaning and it could only be a start of a new sequence after resetting, a new sequence 0 1 0 and that is where $S_1$ comes into picture because $S_1$ remembers $S_0$.

So if 0 occurs it goes here. $S_5$ is a state where an output is 1. that means we have completed a sequence either 0 1 0 or 1 0 0 1. Whether 0 or 1 occurs it should only be a start of a new sequence. If 0 occurs it will go to this $S_1$ because that is the start of a new sequence with 0 for 0 1 0 and if 1 occurs it will go to $S_2$ because that is starting point of a

new sequence 1 0 0 1. So 1 will take you to the $S_2$ state which starts a new pattern with one start for 1 0 0 1, if 0 occurs it takes us to state $S_1$ the start of a new sequence 0 1 0 here.

(Refer Slide Time: 49:50)




Now the diagram is completed. It has six states plus one so seven states totally. Again we use three flip-flops. You can draw the state graph with present state values arbitrarily assigned as 0 0 0, 0 0 1, 0 1 0, 0 1 1, 1 0 0, 1 0 1, 1 1 0 and then input x is equal to 0, what is the next state, what is the output, of course output is different from the state now so with input x is equal to 0 what is the next state then for each of the present state you write the output that becomes the state table and depending on the type of flip-flop you can do the transition table and finish the hardware.

So I leave the rest of the problem as an exercise to you. Draw the state graph using these flip-flops, get the Karnaugh Maps for the next state variables as well as the output and then draw the logic diagram of this. With this we will complete the state graph drawing as an exercise. but of course you will visit this area again when you do a system design naturally again state graph will become a part of the system design. Also, if time permits as I mentioned earlier we will talk about some simplification procedure for reducing the number of states and also for hardware efficient assignment. we will continue with more concepts in system design from next lecture.