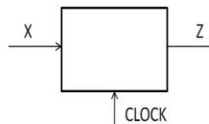



Digital System Design
Professor Neeraj Goel
Department of Computer Science Engineering
Indian Institute of Technology Ropar
Lecture 51
Derivation State Graph

Hello, friends. In our previous lecture, we have seen that how given a sequential circuit, how do we analyze it and how do we create state graph and state tables. Today, we will do a design exercise and we will try to see how we are going to derive a state diagram, state graph and state tables given a problem statements. So, to understand how these state tables and state diagrams are derived, it is important to take a, we will take couple of problem statements. We will start one by one.

(Refer Slide Time 00:56)


Sequence detector





X	0	0	1	1	0	1	0	0	1	0	1	0	1
Z	0	0	0	0	0	1	0	0	0	0	1	0	1
Time	0	1	2	3	4	5	6	7	8	9	10	11	12

Digital Logic Design: Sequential Circuits.



So, the first problem, we are going to take is a Sequence detector. Sequence detector is a simple circuit where we are assuming that there is input, single bit input which is coming at a one input in one cycle and there is a single bit output jet. So, within this sequence detector, it is trying to detect a particular sequence.

So, let us say, this is the input. For the time being, let us also assume that we even do not know what sequence we are detecting. So, this is the input X and this is the output Z and, at different times. So, when we are saying times, so we are assuming that this X would be stable for one whole clock period and it would change only when clock will change. So, once a clock is changed, let us say clock is changed at the positive edge. So, after positive edge, this

X will change and X will remain the constant value. If it is 0 it will remain 0, if it is 1 it will remain 1.


Now, if we see, the sequence we are detecting is actually 1 0 1, so, you see that initially we are assuming it a 0, at time equal to 0 we are assuming Z equal to 0. And then X is 0, then next input come at time 1 is again 0, so the output is 0. Now, next input came is 1, output will again be 0. Remember we are detecting 1 0 1. So, when again, another 1 came, because the sequence is now 0 1 1, so the output is going to be 0.

So, then 1 0 came, 0 came, so 0, we see the last three pattern, the answer is 1 1 0, so that means, again the pattern has not matched and the output is 0. When this 1 came, then we see the last couple of inputs, we see 1 0 1, the, because the pattern has matched, the output is 1. Then, let us say another input is 0. So, because the previous three bits are 0 1 0, so the output is 0. Now, with the same, the next input is 0 so 1 0 0, these are the three bits, it is not matching, so the output is 0. Now, if 1 came, then 0 0 1, these are the previous three bits, so the output is 0.

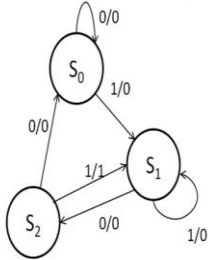
Here also, the output is 0 because the previous three bits are 0 1 0. Now, here, the previous three bits are 1 0 1, so the output is 1. Here, the previous three bits are 0 1 0, so the output is 0. Now, here the output is 1 because the previous three bits are 1 0 1. Now, one thing to notice here that the output will be 1 when my pattern has matched. So, in the same cycle, as soon as the pattern will match, the output will remain 1, during that clock period. So, how do we start? So, we do not know. First of all, that what all we need to store.

So, one thing is clear that we need to store last three bits. Whatever input is there currently, along with that we have to store something of the previous bits which has been received. So, let us see how we can draw the state diagram, how we can implement the circuit given that we even do not know what we need to store, in what format we need to store. So, the state diagram as well as state table synchronous, sequential system will solve these questions that how we can store this information that previous three bits are this. So, let us start drawing the state diagram.

(Refer Slide Time 05:21)




State graph



Sequence 101

S0: Initial state
 S1: 1 bit matched ("1")
 S2: Two bits matched ("10")

Present state	Next state		Present output	
	X=0	X=1	X=0	X=1
S ₀	S ₀	S ₁	0	0
S ₁	S ₂	S ₁	0	0
S ₂	S ₀	S ₁	0	1



So, just keep in mind that we have 1 0 1 is the sequence and my output would be 1, when I have received this whole sequence. So, whenever the last 1 is received, then we can give the output as 1. So, whenever we are drawing the state diagrams, state graphs, so what we do is, we first of all initialize, we assume that there is an initial state S0. This initial state S0 means we do not know what was the previous thing, so what was the pervious input. For sake of convenience we just assume that the previous input or the initial input is, is 0.

So, because initial input is considered as 0, now, if another 0 comes, then again, we are still in the same state because nothing has been detected and the sequence has not started. However, if in the initial state, if we receive 1, so that means at least one of the bit has matched so, we should move to some new state. So, still my output is 0. You see this is a Mealy machine where we are representing the output at the transition. So, the transition is, the input is 1 and the output is going to be 0. Now, in S1 state, one bit has matched. This 1 has matched so, basically 1 bit has matched.

Now, if we receive 0, after that then we should move to a new state S2. But the output is till going to be 0. So, and further if we receive 1, then the output is certainly going to be 1, because we have matched this sequence 1 0 1, but the next state is, is S1 because S1 signifies that at least one bit has matched. So, let us say after 1 0 1, again, after 1 0 1, then we have received this, because this now, 1 would be there, it would signify that at least 1 bit has matched so we can draw this state diagram by looking at the example stream and the example output that could be one of the methods of doing it.

The other method is that, we keep in our mind that what is the initial state, so basically meaning or semantic meaning of all the states. S0 here means clearly that initial state or not bit has matched. S1 means that at least one bit, 1 has matched and S2 means that two bit 1 0 has matched. So, then, we are in the S2 state.

So far, this state diagram or state graph is incomplete because for S0 we know where to go if input is 0, we know where to go when input is 1 but at S1 state, if input is 0, we know we have to go to S2 but we do not know where to go if my input is 1. So, let us think about it. I am currently in S1 state. In S1 state, the next input is also 1 so, there is 1 which has already matched, then another input is 1, so where should we, should we go to S0, S1 or S2?

So, the pattern, if it is 1, another 1 is there, so, then the sequence will become 1 1, so which does not match S0 which is not equal to this S2 also, so what, at least one bit still remain matched so, we can say that if the input is 1, the next state is again going to be S1. Now, so, with this S1 state is complete, because we know, where to go when input is 0, we know where to go when input is 1. What about S2? For S2 we know, where to go when input is 1, but where to go when input is 0.


So, when input is 0, that means the sequence becomes something like this 1 0 0; 1 0 0. So, 0 0 means that it is not matching anything in this sequence so possibly we have to go to initial state where nothing has matched. So, this is how we can draw a state diagram which is also, complete, basically, we have seen all the scenarios, all the cases that what would happen in each and individual state if the input is 0, input is 1.

So, given this complete state diagram or state graph, then we can draw the state table. So, I am directly writing the state table in terms of S0 and S1 so, if it is S0, then this is, the next state is S0 and if X is 0, then, X equal to 0, then the next state is S0, if X is 1, then next state is S1. So, it is essentially the same representation but written in a tabular form.

Now, if the present state is S1, the next state is going to be S2, if input is 0, next state is going to be S1 if input is 1. Now, for S2, if the present state is S2, the next state is going to be S1, if input is 1 and S0 if input is 0. So, S0 is here, S1 is here. Now, output is 1 only in one case when S2, it is going from S2 to S1, the present state is S2 and next state is S1. And the input is 1.

So, it is only in this case when the output is going to be 1. So far, we have created this state table also, now state table is representing in terms of S_0 and S_1 . We also, know that these states would be stored in flip-flops. So, then the next step would be that we have to do state encoding.

(Refer Slide Time 12:11)



State table

Present state	Next state		Present output	
	X=0	X=1	X=0	X=1
S_0	S_0	S_1	0	0
S_1	S_2	S_1	0	0
S_2	S_0	S_1	0	1

AB	A'B+		Z	
	X=0	X=1	X=0	X=1
00	00	01	0	0
01	10	01	0	0
10	00	01	0	1




Digital Logic Design: Sequential Circuits.

So, in the state encoding, we have to assign some bits to S_0 , some bits to S_1 . So, the total number of states are 3, S_0 , S_1 , S_2 . We would require at least two bits to represent that which particular to represent S_0 or S_1 or S_2 . So, let us say, keep it simple, we say S_0 would mean 0 0, S_1 would mean 0 1 and S_2 means 1 0. So, it is simple, decimal representation for those states.

Then we can write this whole state table as a, we will say that these two bits would be stored in two flip-flops A and B where the output is going to be AB and then we can write these next states as A plus and B plus means that the input to my flip-flop, assuming that the flip-flop is a D flip-flop. So, if X equal to 0, then we can say that the next state is S_0 so, 0 0 is written. If the next state is S_1 then we have written 0 1.

Similarly, for the present state is 0 1, the next state is going to be S_2 , that means 1 0 and it is going to be S_1 if X equal to 1 so, that means it is going to be 0 1. Now, if present state is S_2 that means 1 0, the next state is going to be S_0 if X is equal to 0. That means 0 0 here. And if X equal to 1, the next state is going to be S_1 , so that means 0 1 here. So, this way, we have written the state table. Now, the next thing is we have to simplify it, we have to find out the values for A plus and B plus.

(Refer Slide Time 13:59)



State table

AB/X	0	1
00	0	0
01	1	0
11	x	X
10	0	0

$A^+ = X'B$


AB/X	0	1
00	0	1
01	0	1
11	x	X
10	0	1

$B^+ = X$

AB/X	0	1
00	0	0
01	0	0
11	x	X
10	0	1

$Z = AX$

AB	A'B'		Z	
	X=0	X=1	X=0	X=1
00	00	01	0	0
01	10	01	0	0
10	00	01	0	1



Digital Logic Design: Sequential Circuits.

So, for that we can draw a K map, we can see what would be the value for A dash. Let us first do for A dash. A plus, so for A plus, this is essentially, now, we can write in this axis value of AB and in columns we write the value of X. So, you see, AB cannot be 1 1. So, we can take, we can assume that AB, when it is going to be 1 1, then it is a don't care condition.

So, given this don't care condition, then we can fulfill, fill this K map entry is that it is 0 1 X 0 here. So, if we try to solve for A plus, then we can form this particular cube and because of this cube, it would be X dash and B. So, we can write X dash and B would be the next state value for A. Similarly, we can compute the next state value of B dash, B plus. So, for B plus, it is 0 0 0 here. So, 0 0 0 here, this is X, this is X and this is 1 1 1, so, we can write 1 1 1 here.

So, here, because of this don't care condition, this can become a cube. So, I can say B plus is equal to X. Similarly, we can also, draw the K map for Z. It is again straightforward here 0 0 0, 0 0 1. So, when we will combine, we can combine, we can form this cube so, that means my Z value is going to be X and A.

So, this is how we can, then we can, if we would like to further, we can draw two flip-flops, both of them being D type where input of A flip-flop is going to be X dash and B and input to B flip-flop is going to be directly X and my output would be A and with X. So, this is how we can implement that using two flip-flops and using this whole process. So, this was a Mealy machine.

(Refer Slide Time 16:33)

Moore Machine for the same problem

Sequence 101

S0: Initial state
 S1: 1 bit matched ("1")
 S2: Two bits matched ("10")
 S3: Three bits matched ("101")

Digital Logic Design: Sequential Circuits.

State table

AB/X	0	1
00	0	0
01	1	0
11	x	X
10	0	0

AB/X	0	1
00	0	1
01	0	1
11	x	X
10	0	1

AB/X	0	1
00	0	0
01	0	0
11	x	X
10	0	1

$A^* = X'B$ $B^* = X$ $Z = AX$

AB	A*B*		Z	
	X=0	X=1	X=0	X=1
00	00	01	0	0
01	10	01	0	0
10	00	01	0	1

Digital Logic Design: Sequential Circuits.

Now, there is one challenge with the Mealy machine that here we are taking an assumption. The assumption was that my X will not vary during the clock period. If X will vary during the clock period, so now, because you see here, if X would vary along with the clock period, then my Z will also, change with the clock period, within that clock period. So, assume that there is one clock period, within that clock period my X was varying from 1 0 0 1 and it was not stabilizing. So, that means my output will also not stabilize.

So, therefore, sometime, it is preferable to have Moore machine where my, my output does not depend on the current input but it depends only on the past inputs. So, here, let us assume that when this sequence is 1 0 1, after this sequence is 1 0 1, then we will say that my output

is 1. So, if my previous three inputs are 1 0 1, then only output is going to be 1. So, output does not depend on the current input but only on the past inputs.

So, if I would like to draw the state diagram for this sequence, for a Moore machine, then let us try to start. Again, we will start with S0, which is my initial state, assuming that nothing has matched. So, this assumption that nothing has matched, also, clearly says that the output is going to be 0, again because it is Moore machine so, output depends on my state, in which state I am. It does not depend on the current input.

So, because my S0 state means that nothing has matched, so the output is 0. So, given that the, we are in the initial state and we further have a 0. So, initial state also here assumes that my, I am assuming that that before this initial state, it was 0 before this initial state, we still have to assume that before the initial state, there was a 0 in the, it was initialized to 0 essentially, in other words we are saying that we have to, we have to assume something as a minus 1.

So, what sequence we have to assume, what input we have to assume as a minus 1. Minus 1 means that when time has not started, then what should we assume. So, here the assumption is that it was 0. So, if it was 0, then next 0 came, we are still in the same state that nothing has been detected, not even a single bit.

However, if 1 is there, then we have to move to the next state, just like a Moore machine, Mealy machine that 1 would signify that at least 1 bit has matched. So, we will go to the next state. Still, the output is going to be 0 because, output has not matched, the whole sequence has not matched.

So, then we are there in the S1 state with the output 0. Now, if the next input is 0, then we move to the next state which means at least two bit had matched. So, two bits has matched, we have moved from S1 to S2 state. Now, further, if 1 came, then we move to the third state which means that all the three bits has matched. Here, in this state because we are saying all the three bits has matched, 1 0 1, all the three bits had matched, then the output is going to be 1.

So far, so good. It is very sweet so far, so now, we can write the meaning of all the four states. S0 means initial state, nothing has matched. S1 means at least one bit has matched, 1. The S2 means two bits has matched, 1 0 and S3 means all the three bits has matched, 1 0 1

has matched. Now, next steps are very crucial to understand and it requires some bit of judgment that whenever we are completing rest of the state machine, that what should be the next state.

So, for example, here for S0 state, the state machine is complete because we know where it is going when input is 0 and we know where it is going, what would be the transition when input is 1. However, when we are in state S1, then we know what would be the transition, what would be the next state when input is 0. What would be the state when the input is 1?

Now, if input is 1, what is going to be the sequence? Sequence is going to be, there must be a 0 here, 1 1. 0 1 1. So, that means, what has matched? 0 1 1 means only one bit has matched because it is not helping in any other match. So, we will say that if the input is 1, then S1 state will still remain S1 state so, the next state for S1 is going to be 1, if the input is 1. Now, for S2, I know that if the transition is 1, if the input is 1, the transition will be towards S3 state but what would happen if the input is 0?

So, let us see, if the input is 0, the previous three bits is going to be 1 0 0. So, from 1, then 0 then 0. 1 0 0 means that we are probably, we are probably not matching any of them, so we have to fall back to the initial state, so that means nothing has matched. 1 0 0 is not helping in any way in this matching of the sequence, so we will fall back to S0. So, this way we will be able to complete this S2 state also and then we will go back to S3.

In S3, we, what would happen if the input is 0? Here, we do not know what would happen if input is 0 or what would be the transition when input is 1. So, let us say, input is 0. So, if input is 0, then it will become 0 1 0. 0 1 and 0. So, 0 1 0, it, it is not matching initial state. It may match initial state but we will say 0 1 0, it is not matching S3 but 0 1 0, so if we remove that 0 part then 1 0 is matching the S1 state. These two bits are still matching.

The previous two bits. 1, this is 1 and the current input is 0. So, that means these two bits are matching, so, we can say that we can go back to our S2 state. So, if input is 0, then we can go back to S2 state. That is done. Now, what would happen if the input is 1?

If the input is 1, then the sequence is going to be 1 and then another 1. Because this is 1, another 1 so, that means it would probably would be the S1 state which means that at least 1 bit is matching. So, we can put an arrow towards, from S3 to S1 and we can say that if input is 1, then we go to this state. So, this is how, we can create a state diagram for a Moore

machine, Moore machine. Now, after the state diagram, then we have to decide that what should be the encodings or what would be the state table.

(Refer Slide Time 25:20)

State table



Present state	Next state		Output
	X=0	X=1	
S ₀	S ₀	S ₁	0
S ₁	S ₂	S ₁	0
S ₂	S ₀	S ₃	0
S ₃	S ₂	S ₁	1

AB	A ⁺ B ⁺		Z
	X=0	X=1	
00	00	01	0
01	10	01	0
10	00	11	0
11	10	01	1

A⁺ = BX' + XAB'

B⁺ = X

Z = AB

Digital Logic Design: Sequential Circuits.


So, looking at that, I have directly written my state table like this, where we say this is my present state and what would be the next state, when input is X equal to 0 and what would be the next state when X equal to 1. So, the, from the diagram to state table, we have to see nicely, we have to see row by row for each present state, we have to see that what would be the next state when input is 0, we have to see what is the next state when input is 1 and then we write this state table neatly. So, once the state table is written, the next step is that we have to do encoding.

So, let us give, again decimal numbers to this, these states. We can say S₀ means 0 0, S₁ means 0 1, S₂ means 1 0 and S₃ means 1 1. So, which also, means that we would require at least two registers, two flip-flops. Two flip-flops, let us call them A and B and, so, we can write this table, this AB means the present state and A plus B plus is the input to my D flip-flop that means that is going to be the next state and whenever active edge would be there, this next state would become the present state.

Because we are saying that the S₀ is 0 0, S₁ is 0 1 and S₂ is 1 0 and S₃ is 1 1, so we can correspondingly write the value of S₀, S₁, S₂, S₁, S₀, S₃ and S₂ and S₁ here. And now, we can, we can find the value of A, A plus and B plus and Z also. So, if we see, A plus, so, A plus, I am not drawing the K map directly looking at the table, so you can see, A plus would be there when this is 1, then this is 1 and when this is 1.

So, because this and this has only single bit of difference so, we can say, A plus is going to be X dash and B, X dash and B and for this one, we, there is no combination forming, so there is no cube forming, so we can write it as X A B dash. For B equal to 1, so, B is 1 only for all of these four cases, so we can say B equal to X B plus is going to be B equal to X and Z is only true when A and B both are 1. So, we can show it systematically using K map. Right now, I have done directly by looking at the state table.

(Refer Slide Time 28:22)



State table (alternative encoding)

Present state	Next state		Output
	X=0	X=1	
S ₀	S ₀	S ₁	0
S ₁	S ₂	S ₁	0
S ₂	S ₀	S ₃	0
S ₃	S ₂	S ₁	1


AB	A' B'		Z
	X=0	X=1	
00	00	01	0
01	11	01	0
11	00	10	0
10	11	01	1

$$A+ = X'(AB' + A'B) + XAB$$

$$B+ = XA' + AB' + A'B$$

$$Z = AB$$

Digital Logic Design: Sequential Circuits.



So, there could be other encoding also. So, there is no enforcement, there is no rule that we have to always follow decimal coding so, right now, here, I am trying to do it using Gray Code. So, S0 means 0 0, S1 means 0 1 and S2 means 1 1 and S3 is 1 0.

So, using this Gray encoding to my states, now, let us see what would be the value of A plus and B plus. So, if I look at A plus. A plus would depend on this 1, this 1 and this 1, so if you see, that there is no cubical, there is no cube formed, because this is 0 1, this is 1 0 so, you cannot combine any of the terms. So, you have to write X dash AB dash plus A dash B.

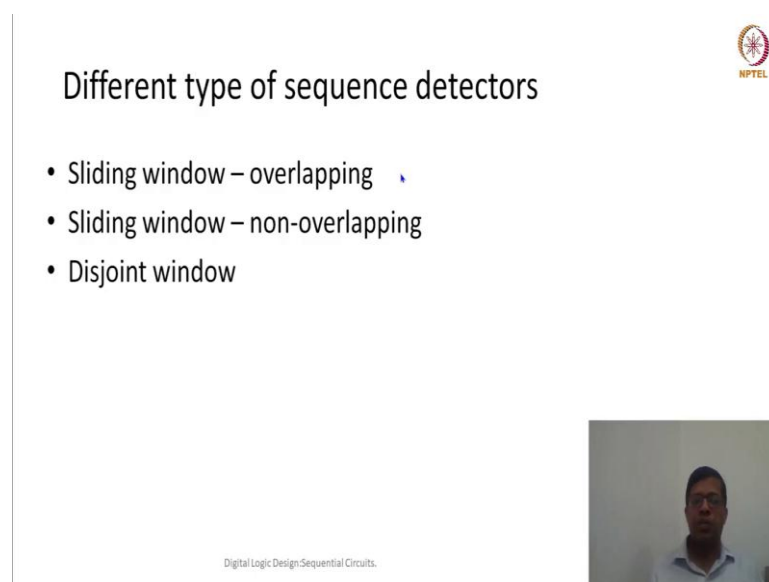
Similarly, this one also, cannot be combined with either of these ones, so we have to write X A B. Now, for B dash or B plus, so for B plus also we see that this is 1 here, this is 1 here, 1 here and 1 here. So, yeah, I think I have not written it correctly, but, we can combine this, we can these two, so it is going to be, so for B plus, it is, this is 1, this is 1, so this will become A dash B and because this is 1, this is 1, so this become AB dash and because of this particular cube, this will become X and A dash. So, and Z equal to AB, Z will not be affected.

So, what you have seen that based on my encoding, because I have changed my encoding, the size of my A plus or the logic which is involved in computing the next state has become more complex or it would require more number of gates.

So, which, which also, gives us a hint that the state encoding, if you would like to optimize the total number of gates or total number of representations, total number of gates to implement a particular state diagram or a state table, then what encoding do we choose may also be important in determining the number of gates.

We are not right now focused on reducing this particular number of gates but this can be thought of as a good implementation or good optimization problem that what should be my state encoding such that the, the number of gates in this next state computation is minimum.

(Refer Slide Time 31:28)



Different type of sequence detectors

- Sliding window – overlapping
- Sliding window – non-overlapping
- Disjoint window

Digital Logic Design-Sequential Circuits.

So, here, for this particular example, we have taken couple of assumptions. The first assumption was that my initial state, in the initial state, we were assuming that previously to the initial state, there was a 0. So, assuming the previous, in the initial state, first time when this circuit has started entering, we were assuming that initially there was a or basically T minus 1 at T equal to minus 1 the input was 0.

So, this is first assumption, the other assumption was that we were continuously checking for the previous three bits. So, when we are checking for continuously and previous three bits, so, previous N number of bits. We call this particular type of sequence detector as sliding

window. Because, my window is always sliding, I am always checking at previous N inputs if the sequence detector is saying to detect, detect N inputs.

So, further, the other assumption or the other characteristic of this particular output what we have discussed today was that although one pattern has matched, if one pattern has matched then we have also, thought of, like the two patterns can be overlapping. So, if there is a 1 0 1, then again 0 1. So, these 1 0 1 and 0 1 could be the overlapping patterns. So, there is a 1 which is appearing in the first pattern as well as in the second pattern.

So, this particular type of sequence detector, what we have discussed in today's lecture is sliding window and overlapping type. So, where, one pattern can overlap with the other pattern and the window is always sliding. On the other hand, there could be other detectors which we will say non-overlapping, so means that if I have detected 1 particular pattern, the next pattern cannot be part of my, cannot be part of my first pattern. So, in case of non-overlapping, my state diagram is going to be different.

But we are still saying that, we are still sliding window, sliding window means that we are always looking at last three bits. So, this sliding window and non-overlapping could also be an interesting in different problems, interesting and different problems because we are looking at different kind of solutions here and different state diagrams would be created, different implementation would be there.

The third interesting case would be disjoint window. Disjoint window means that, my inputs, whenever we are going to detect, we are detecting only in the, for example here, we were detecting three bits, so we are always detecting in the set of three bits 0 1 2, then 3 4 5, then, 3 4 5, then 6 7 8, so basically, there is no sliding window.

We are always saying that first three bits, then we will match, if it will match then we are good, otherwise we will again go back to the reset state. Again, next three bits, here in case of a disjoint window, what would be my initial T equal to minus 1 value, it does not matter. Why it does not matter, because we are always saying, we have three bits, 0 1 2, 3 4 5, similarly, 6 7 8, then 9 10 11. So, basically because we are always in the group of three, we are using a disjoint window, it does not matter what is T equal to minus 1. So, it does not depend on what is the previous window.

So, now, there could be other cases where we are assuming that at T equal to minus 1, if the input, if T equal to minus 1, we assume that my input was 1. So, then in those cases, when T equal to minus 1, the input is 1, the state diagram would be slightly different. It all depends on what is the specification, what problem we are solving and then it would depend on, then it would say that how we are going to create those state diagrams.

So, finally, like, once, like in next couple of lectures, we will try to design couple of different sequence detectors so, that we will come to know, we will do some practice that how to design these state diagrams and so that we can know that how to implement these state machines and then using flip-flops.

So, this exercise we will try to do multiple times, then you will understand it more. We will take some more complex examples also and with those examples we will try to understand couple of different concepts in designing sequence, synchronous sequential machines. Thank you very much.