Digital System Design

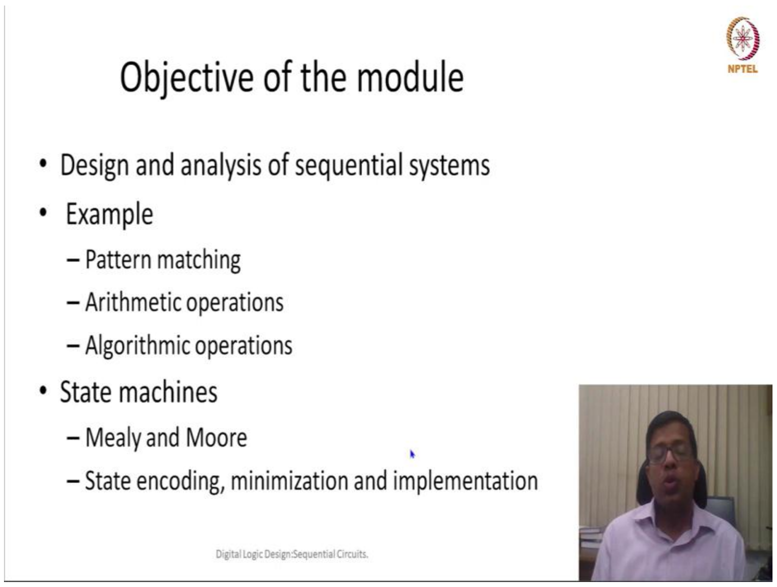
Professor Neeraj Goel
Department of Computer Science Engineering
Indian Institute of Technology Ropar

Lecture 50

Sequential Circuit Analysis

Hello, everyone. Today we are going to start a new module, Sequential Circuit Design. In our previous module we have seen the fundamentals of sequential circuit design where we have seen how fundamental elements of sequential circuits can be designed for example flip-flops, registers and then some simple sequential circuits also, we designed like counters and shift registers.

(Refer Slide Time 00:45)



Objective of the module

- Design and analysis of sequential systems
- Example
  - Pattern matching
  - Arithmetic operations
  - Algorithmic operations
- State machines
  - Mealy and Moore
  - State encoding, minimization and implementation

Digital Logic Design:Sequential Circuits.

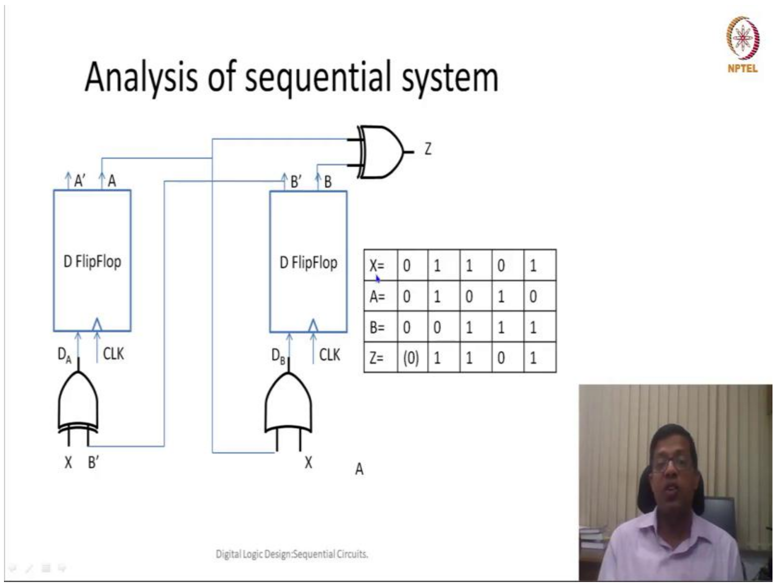
In this module we are going to extend the whole information, whole knowledge of sequential circuits and we are going to apply to design bigger applications. We can call them sequential systems also. So, we will look at different applications, for example pattern matching or some kind of arithmetic operations which can be performed more efficiently if we do sequential, design by using sequential circuits. And also, if given an algorithm, how can we implement that using sequential circuits. So, this also, we will see using examples.

On top of that, we will also, try to analyze a given sequential circuit. And the overall methodology would be that these sequential circuits are designed using state machines and all of them, as we discussed in our previous module also, they are going to be synchronous sequential circuits, means that they will be always synchronized with one clock signal, one global clock signal.

So, there are two kinds of sequential circuits. They are driven by state machines, either Mealy state machine or Moore state machine. And while understanding how to design these sequential systems, we will see that these state machines need to be encoded, then these states need to be minimized and then we implement these state machines using gates and flip-flops.

So, with this objective, let us start with first understanding how a sequential circuit looks like. So, to meet this particular smaller objective for this lecture, to meet this objective, we will take an example of a sequential circuit and then try to analyze it.

(Refer Slide Time 02:38)



Analysis of sequential system

X=	0	1	1	0	1
A=	0	1	0	1	0
B=	0	0	1	1	1
Z=	(0)	1	1	0	1

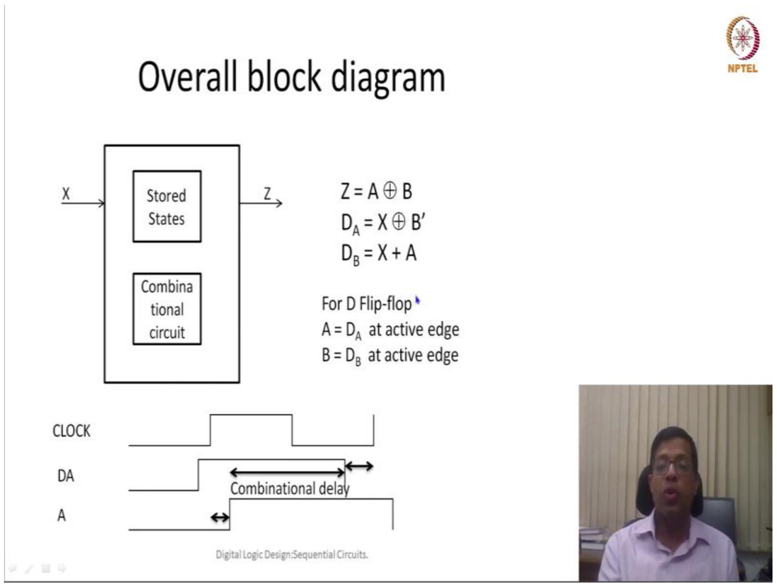
So, let us take this sequential circuit as an example which has two flip-flops, both of them are D flip-flop and it takes DA and DB as an input and this DA takes X as input as well as the output of the other flip-flop B dash. And similarly this flip-flop takes an input, A as an input and X, another X as an input. Z output is actually the XOR of A and B.

So, you see, this is a, this overall sequential circuit can also, be, can, if we try to analyze it, if we try to understand how does it work, so, we have to first of all think of that what would be the initial state. So, let us consider the initial state or initial, the flip-flops initially were 0 0. So, this is how we try to analyze a sequential circuit. We try to assume that the input is 0 0.

So, if the state, initial state of the flip-flop is 0 0 and then let us consider input is also, 0. So, because A and B both are 0, my X is 0 so, what would happen the next state? Because this X is 0 and B dash is going to be 1. Because B dash is going to be 1, DA is going to be 1 because this is 1 in the next clock cycle, next edge of the positive edge of the clock cycle, A will become 1. So, this A will become 1. And now, since X is 0, and A is also, 0, initial state is 0 so, OR of them is going to be 0. DB is going to be 0, so, because DB is 0, so, B is also, going to be 0. The output is going to be 1 0.

Now, let us consider for the example that the next input is 1. If the next input is 1, the, so, based on the first previous state my output is going to be 0 because A is 0, XOR of both of them is 0. So, what we are going to show with this example, you can work out this further on with this input sequence, you will see that my next A and B depends on what is value of X, what was the previous A, what is the previous B. And then we can find out what would be the next A, next B. The overall system depends on the input value of A, but, input value of X but this X is essentially modifying the value of A and B in every state.

(Refer Slide Time 05:40)



Overall block diagram

$Z = A \oplus B$

$D_A = X \oplus B'$

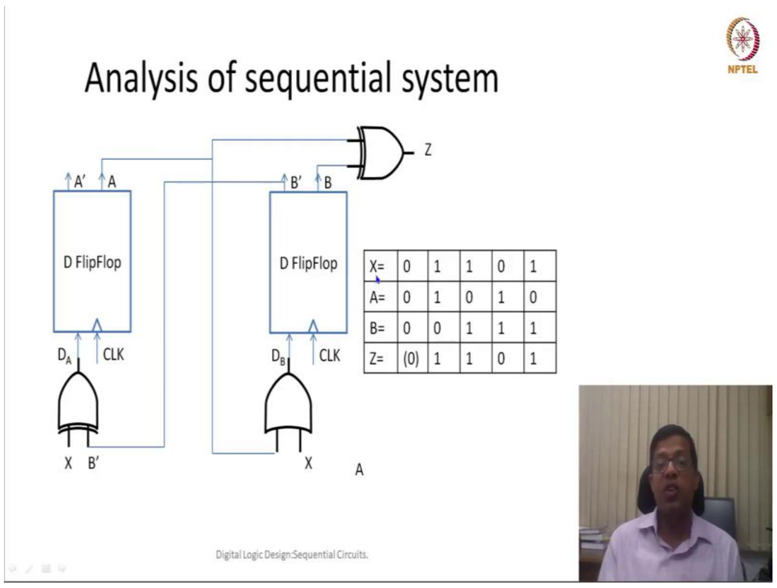
$D_B = X + A$

For D Flip-flop

$A = D_A$ at active edge

$B = D_B$ at active edge

Digital Logic Design:Sequential Circuits.



Analysis of sequential system

X=	0	1	1	0	1
A=	0	1	0	1	0
B=	0	0	1	1	1
Z=	(0)	1	1	0	1

So, that way we can write the overall block diagram as like, the input is X but the output is Z. Now, Z does not depend on X directly because this X is influencing two states which are stored inside or two flip-flops which are stored, which is A and B. Now, these A and B are further not affecting the Z directly but these A and B are also, further being given in a, it depends on the previous state A and B and then the combinational circuit which drives the next state. And the next state depends on A, B as well as X. So, because of this whole of these combinations, now my Z depends on the stored states, it could depend on X also.

So, that is why we said initially, when we were defining a sequential circuit that means my output does not depend on the current value of X, because it depends on, this current value of X would modify the, the stored state values and these stored state values would also, modify, will depend on like, the next state would depend on the previous state as well as the value of X.

So, overall my sequential circuit depends on the previous state values and the current X value. So, in the previous case, based on this diagram we have written the value of Z, DA and DB. So, Z is equal to A XOR B and DA is equal to X XOR B dash, DB is equal to X plus X OR A. Now, we know D flip-flop, A, DA will become A, whenever there is an active edge. Similarly, DB will also, become B when there is an active edge. So, the translation from DA to A and DB to B is kind of a transparent or straightforward in case of a D flip-flop.

So, let us try to understand that if given such a circuit, how should we go ahead and analyze. So, we can break this circuit into a combinational circuit to understand, to analyze. What we can do is, you also, see that the value of A and B, because they are the output of my, they are the output of my flip-flops, they are not going to change for one clock period.

So, I can assume that as an input to my, calculation of my next state. So, basically, calculation of my DA or DB. Or, and this DA and DB will become the next state so, I can say this, this is, DA and DB is going to be A plus or basically the next state. DA is the A plus value and DB is the B plus value, that means the next state of the flip-flop B.

So, if we see that then we can, if we are trying to break this whole circuit into a combinational circuit, then what we can do is, yeah, so, then, then we can create basically, we can analyze and we can say which all states are possible and what would be their next states. So, before going into that explanation, let us also, see one more important point which we have although seen in our previous module but let us reiterate that.

So, you see that if this is my regular clock and let us say DA value, because of this combinational logic will get settled to 0 and sometimes it will get settled to 1. Now, at the active edge of the clock, at the active edge of the clock, this DA value will be transferred to A. But there would be certain delay that is because of, because of the flip-flop delay.

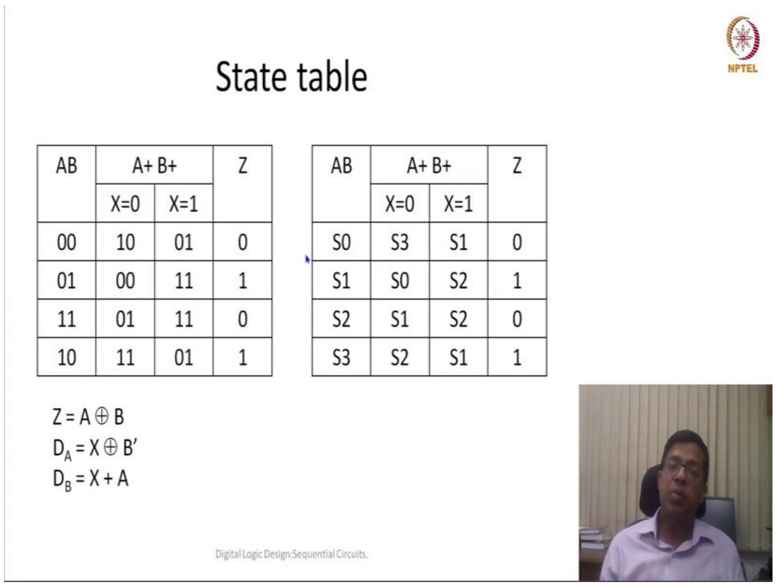
Now, once this A is available, then DA value will further change. You see that once A is available, then it will change the value of DA. So, not in this case but whenever, so, here we can assume that this is DB. So, basically, when A is available, X is available, then value of DB will further change.

Now, this whole is a combinational circuit. Now, this combinational circuit is taking two inputs. One is original input X, the other is the output of the previous, output of the flip-flops. So, the hold timing diagram could be understood like this that this combinational delay, this combinational circuit, the worst case combinational delay could be this. So, that means my DA value or the output of this combinational has to get stabilized within certain combinational delay.

And that certain combinational delay should be lesser than the setup time plus, so, this should be lesser than the clock period, further this should settle down before the hold time of this clock. So, hold time of my register. So, before hold time, this gets stabled down and we also, have to understand that this combinational delay will start only after the propagation delay of the flip-flop. So, these, all these three things, combinational delay, worst case combinational delay plus the setup time plus the hold time, this will determine our clock period.

So, this point was there to reiterate so, that we can understand that whatever combinational circuit we have, how that would affect the clock period and how things would internally manage and would be, there would be delays so, that they would get finally settled. Now, let us look at, to, get back to the analysis that given these state, these behavior equations, how do we come up with the, with the understanding of this sequential circuit.

(Refer Slide Time 12:16)



State table

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

AB	A+ B+		Z
	X=0	X=1	
S0	S3	S1	0
S1	S0	S2	1
S2	S1	S2	0
S3	S2	S1	1

$Z = A \oplus B$
$D_A = X \oplus B'$
$D_B = X + A$

Digital Logic Design:Sequential Circuits.

So, I have rewritten the equations here. Z equals to A XOR B, DA equal to X XOR B dash and DA equal to X plus A. Now, to analyze that, what I can do is, I can initialize, I can assume that the A and B values are 0 0, 0 1, 1 1 and 1 0 and then I can create the truth table. So, you see that whatever is the A dash is actually the DA and whatever is the B dash is actually DB. So, given the X equal to 0, I can calculate that if previous, because X is 0, and my A and B, both were 0 so, DA is going to be 1 and DB is going to be 0. So, 1 0 is going to be the output.

See, if X is one then DB would be 1 and DA is, will become 0. So, 0 1 would be the next state. So, similarly, we can compute the next values of truth table from these equations and we can also, compute the value of Z from this A XOR B equation. So, this, this whole table is called a State table because now my Z depends on what is the value of, what is the value that is stored inside the flip-flop.

Inside the flip-flop, A and B is stored which would be available for one clock period. So, now, because A and B is stored within the flip-flop, my Z depends on here because here equation says A XOR B, it does not depend on X. But X is modifying the value of A and B, by saying that this is the next state. Because of DA and DB, my, my X is changing the value of DA and DB so, that means it is changing the value of next state for these registers.

So, that is why, A and B, the values which are stored inside the flip-flop are called state registers, so, in other words, what we can say is that my final value of output depends on in which state, I am. It does not matter like, whether I call this state as S0 or S1 but the important point is, if the state is 0 0, then output is going to be 0.

So, this way, it is, it is more systematic to name all these states a state numbers like S0, S1, S2 and S3, so, if I try to map 0 0 as S0 and 0 1 as S1, 1 1 as S2 and 1 0 as S3, I can rewrite this table as this table, the new table. This table is actually called a state table. Here, we are not defining, what would be the meaning, what would be the value of S0. So, whether S0 is going to be 0 0 or 0 1, but we are here defining that S0 is a state where if X is 0, then next value, next state is going to be S3 and if X I 1, the next state is going to be S1 and output is going to be 0.

So, this State table essentially describe the behavior of the circuit. So, further I would like to emphasize that it does not matter that what would be the encoding but if these are the state transitions, like, if X is 0, then how, which state I will go to the next state, which state would be the next state.

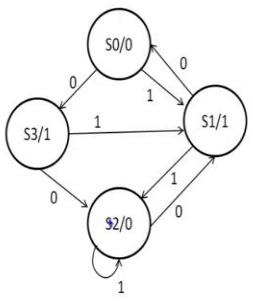
So, this way, this state table actually characterized our, our sequential circuit. So, this is one of the important representations of my sequential circuit. You remember that when we were doing a combinational circuit, the combinational circuits were represented by either the expressions, Boolean expressions or using Truth Table.

Here, sequential circuits are more appropriately described using State table. These State tables are very much like a Truth Tables but here, the encoding of the state is not defined. So, if it is defined then it is good but even if it is not defined then also, it makes complete sense. Now, this is one of the representation. The other representation of my sequential circuit is called State diagrams.

(Refer Slide Time 17:19)

State diagram

AB	A+ B+		Z
	X=0	X=1	
S0	S3	S1	0
S1	S0	S2	1
S2	S1	S2	0
S3	S2	S1	1



Moore State machine

Digital Logic Design:Sequential Circuits.

So, in the state diagram, we represent our, our sequential circuits in form of a state diagram. It is a graphical representation. In this graphical representation, each of the state represents one circle. Now, here, in this particular example, because my output does not depend on input, but it depends on the (out), depends on the state. So, the state diagram explicitly says that if I am in State 0, then the output is going to be 0. And if I am in State 1, then output is going to be 1. And if I am in State 2, S2 then the output is 0.

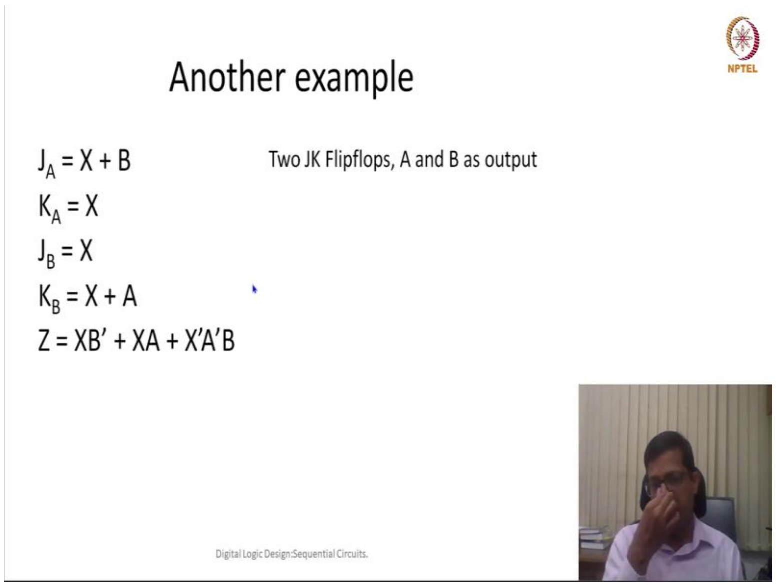
Similarly, if I am in state S3, then output is going to be 1. So, circles represent states, which state I am in. And if output is associated with that state then output is also, giving by writing a dash and then writing the output, corresponding output. Now, the circle represents states and now, there would be edges, that also, represent transitions.

Now, if X equal to 1, then S0, the next state is going to be S1. For S0, the next state is going to be S1, and similarly, if input is 0, then the next state for S0 is going to be S3. So, arrows, represent the transition of the state from S0 to S3, when the input is XOR, these arrows are annotated with the value of input.

If input is 0, then S0 would be, S0 will move to S3 set and if input is 1, then S0 will transition to S1 state. Similarly, for S1 also, we can find out that if input is 0, then it will transition to S0 state and if input is 1, then S1 will, will transition to S2 state. Now, for S2 state also, we can check before the condition. If input is 0, then it will transition to S1 state and if input is 1, then it will transition back to itself. So, there is a feedback loop from S2, it will still remain S2.

Now, for S3, if the input is 1, S3 will transition to S1 state and if input is 0, then S3 would transition to S2 state. So, this particular type state machines, where output does not depend on the input but it depends on the states, it is called Moore Machines or Moore Machines. So, this is one type of the Moore machine, one type of the state diagram. Let us take another example.

(Refer Slide Time 20:31)



Another example

$J_A = X + B$

$K_A = X$

$J_B = X$

$K_B = X + A$

$Z = XB' + XA + X'A'B$

Two JK Flipflops, A and B as output

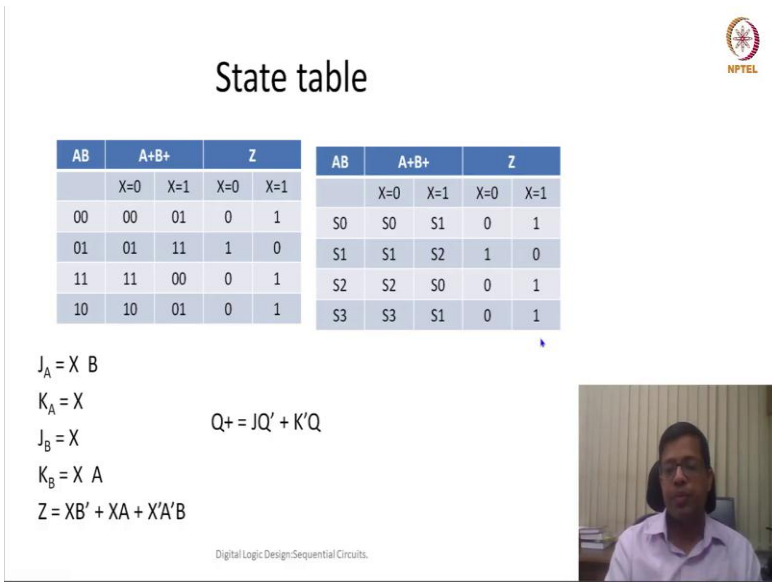
Digital Logic Design:Sequential Circuits.

So, in the other example also, we have two flip-flops but to make things more diverse, we have taken JK flip-flops. So, in JK flip-flops there are A and B are two outputs and for A flip-flop, JA and JK, JA and KA is the input and for B flip-flop, JB and KB is the input. And output depends on both, A, B as well as value of X.

So, you can re-imagine this also. There are two flip-flops A and B, whose output is A and B. Now, the input of both the flip-flops J and K depends on X as well as the output of the flip-flop. And similarly the output, final output of the circuit depends on both X as well as A as well as B.

So, if we need to analyze this, again we can first, what we can do is, we can create a state table. So, in the state table, we can enumerate all the possibilities of A and B and then we can calculate what are the next states. And from the next states, the next states depends on what is the value of X, what is the value of, what is the value, what would be the next state when X is 0, what is the next value, next state when X is 1.

(Refer Slide Time 21:59)



State table

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

AB	A+B+		Z	
	X=0	X=1	X=0	X=1
S0	S0	S1	0	1
S1	S1	S2	1	0
S2	S2	S0	0	1
S3	S3	S1	0	1

$J_A = X\ B$

$K_A = X$

$J_B = X$

$K_B = X\ A$

$Z = XB' + XA + X'A'B$

$Q+ = JQ' + K'Q$

Digital Logic Design:Sequential Circuits.

So, because now, we need to find out what is the value of next state, I, we can find out the value of next state by using the characteristic equation of JK flip-flop. For JK flip-flop, the next state is defined by J Q dash plus K dash Q. So, whenever we need to find out the next state, so, basically A dash and B dash, then we need to use this particular, this particular, this particular equation and using this particular equation, we have to find out the value of next state.

So, here, I am computing directly for you, you can do it as your homework. And maybe we can use our, like, I can upload a tutorial where we can work out all these problems at least one of these problems so, that it can, it can be helpful but yes, for now, let us take this State table as the worked out example. Now, we again, we enumerate all the possibilities of A and B, 0 0, 0 1, 1 1 and 1 0 and then we calculate the value of A plus and B plus.

So, let us solve one of the, one of the row. So, if JA, it depends on X and B and K depends on 0. So, let us say initially X is 0, so, if X is 0, JA and KA both are going to be 0, similarly, JB and KB both are going to be 0 because JA and KA is 0 so, A dash is going to be 0 because it is the previous state. Similarly, because JB and KB, both are 0 so, next state of B is also going to be remain same.

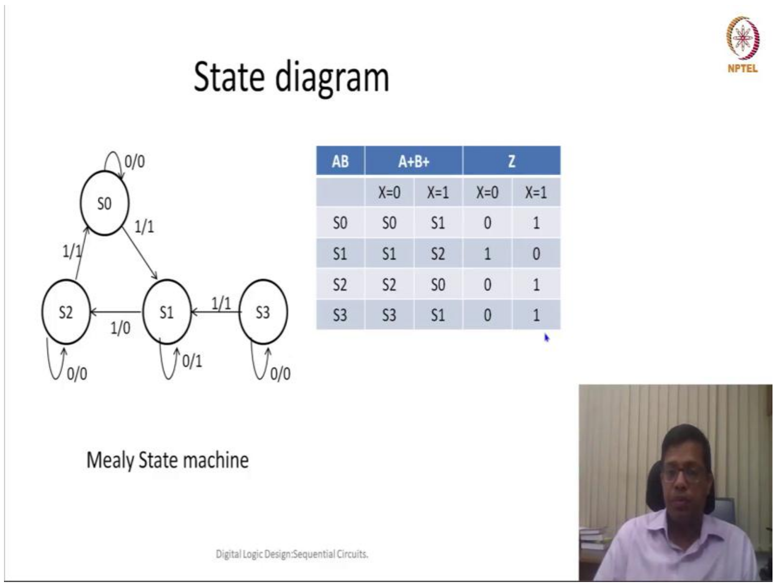
While, when X is 1, then JA is 0, KA is 1. Because JA and KA is 1, the next output is going to be 0 for A plus. A plus value is going to be 0. On the other hand, JB is going to be 1 and KB is going to be 0. Because JB is 1 and KB is 0 so, B plus is going to be 1. Similarly, Z is going to be 0 because Z depends on, Z depends on, because all of these values, this is 0, X is 0 here and here B is 0 because of that if X is 0, my Z is 0.

However, if X is 1, then B dash is going to be 1 and X is 1 so, output is going to be 1, rest we can ignore. So, this is the way, we have to calculate the value of A plus and B plus in all these scenarios. If we calculate the values of A plus and B plus in all the scenarios, then we will get this state table. Now, like the previous example, we can map 0 0 to S0, 01 to S1, 1 1 to S2 and 1 0 to S3, then we can finally create this as a state table.

So, you see the difference here in this state table, one of the columns represents what are the present states, the next state column depends on the input combinations. If input combination is X equal to 0, then the next state is going to be different. If input combination equal to 1, then the next state is going to be different.

Similarly, here Z depends on the value of X so, there would be, if X is equal to 0, then there would be these values of Z and if X equal to 1, then these would be the values of Z. So, this is how, we write the state diagram. Now, using the state diagram, we can again further create the state, sorry, this is the state table. Using the state table, we can create the state diagram.

(Refer Slide time 26:16)



State diagram

$$\begin{tikzcd}
 & S0 \arrow[loop above, "0/0"] \arrow[dr, "1/1"] & & \\
S2 \arrow[loop below, "0/0"] \arrow[ur, "1/1"] & & S1 \arrow[loop below, "0/1"] \arrow[ll, "1/0"] & S3 \arrow[loop below, "0/0"] \arrow[l, "1/1"']
\end{tikzcd}$$

AB	A+B+		Z	
	X=0	X=1	X=0	X=1
S0	S0	S1	0	1
S1	S1	S2	1	0
S2	S2	S0	0	1
S3	S3	S1	0	1

Mealy State machine

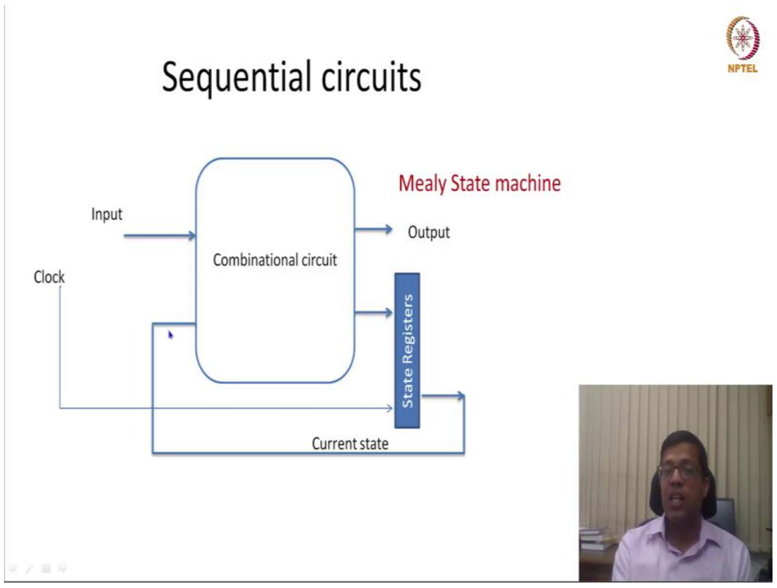
Digital Logic Design:Sequential Circuits.

So, here, we have again 4 states. So, now, output does not depend on the state so, in the circle, in the circle of the state each state, we are going to write only the state number, S0, S1, S2 and S3. For each transition this time, now, S0, if the input is 0, then output is 0. So, here, we have to represent the output along with the edges. So, each transition would mean that what is the input what would be the output. So, numerator says the input and denominator says the output.

So, if the input is 0, output is going to be 0, in case of S0. Now, if input is 1, then S0 will get transitioned to S1 and output is also going to be 1. Similarly, for S1, if the input is 0, then output is going to be 1, and if input is 1 then output is going to be 0 and state will transition from S1 to S2. So, similarly, from S2, if input is 0, then it will still remain S2. If input is 1, then S2 will be transitioned to S0.

And the output is going to be 1. So, this denominator represents the output. Similarly, for S3, if S3, input X is 0, then it will still remain S3 and the output is 0. If input is 1, then S3 would be transitioned to S1 and the output is going to be 1. So, since here my output depends on not only the current state but it also, depends on what is the current input, so, this particular type of machine is called Mealy State machine. In the Mealy State machine, my output Z depends on what is the internal state, what is the current input.

(Refer Slide Time 28:53)



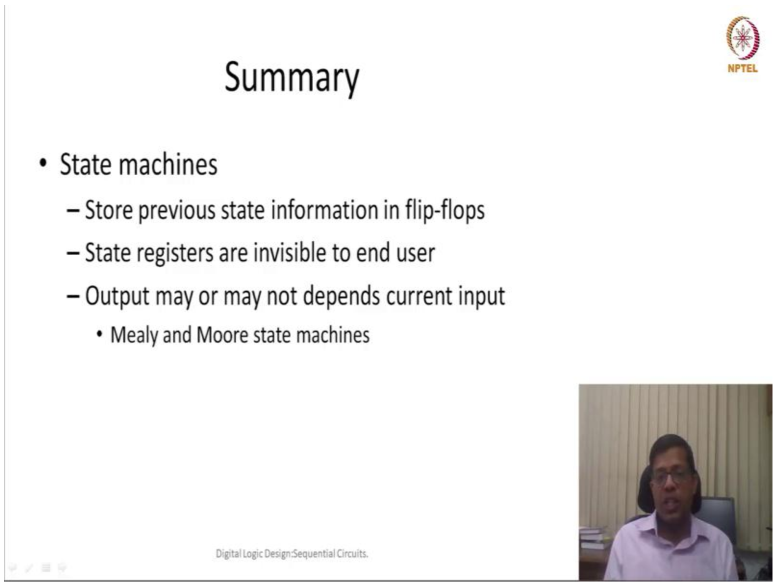
Sequential circuits

So, now, we can summarize this whole sequential circuits into this diagram, where we are saying that actually there is going to be the input and some combinational circuit. This combinational circuit would define what would be the next state. So, these state registers would be the array of flip-flops and whenever the clock period is there, whenever the active edge of the clock would be there, so, this calculation, would affect the next state and then next state is also, given as the input.

So, essentially, to the combinational circuit, there are two inputs, one is the net, one is the current input, plus one is the present state. So, this present state is also, going to compute the next state values and the output, if my output depends on the, only on the state registers then we call is Moore state machine or Moore state machine.

If my output depends on input and the present state, you see, here this combinational circuit also, have one of the input is the original input and also the state, the state registers output is also considered as input. So, when output depends on input, present input plus the present state, then this is called Mealy State Machine.

(Refer Slide Time 30:24)



Summary

- State machines
  - Store previous state information in flip-flops
  - State registers are invisible to end user
  - Output may or may not depends current input
    - Mealy and Moore state machines

Digital Logic Design:Sequential Circuits.

So, with this, I would like to summarize this lecture and will close. So, what we have understood that in the state machines or in a sequential circuit, the previous state information is captured inside the flip-flop. And many a times, this state information would be invisible to the end user or to the final output. Because this state registers as well as the state information is invisible, so, output, it appears that it depends on not only the current input but all the past inputs.

And fortunately, all the previous inputs, it could be infinite but we can summarize all the previous infinite inputs into number of finite states. And because all the infinite inputs can be summarized in form of finite state, there is, that is why it is also, called finite state machines. So, these finite states or these states would be usually invisible to the user, that is why because they are, they are not visible that is why many a times this encoding is also, not visible whether S0 would have, would be, would be encoded as 0 0 or 0 1 although it depends on the, it impacts the impacts the implementation, it impacts the combinational circuit which we were designing inside the state machine implementation.

But to the end user, many a times, to the end user it does not matter whether you are defining S0 or S1, different states as what encoding. So, the other thing we have learned is the output may not depend on the current input, of course it would depend on the past input but it may not depend on the current input. If it depends on the current input, we call it a Mealy state machine. If it does not depend on the current input but only on the past input then we are calling Moore State machine. So, with this, I would close. And we will you see in the next lecture. Thank you.