Switching Circuits and Logic Design

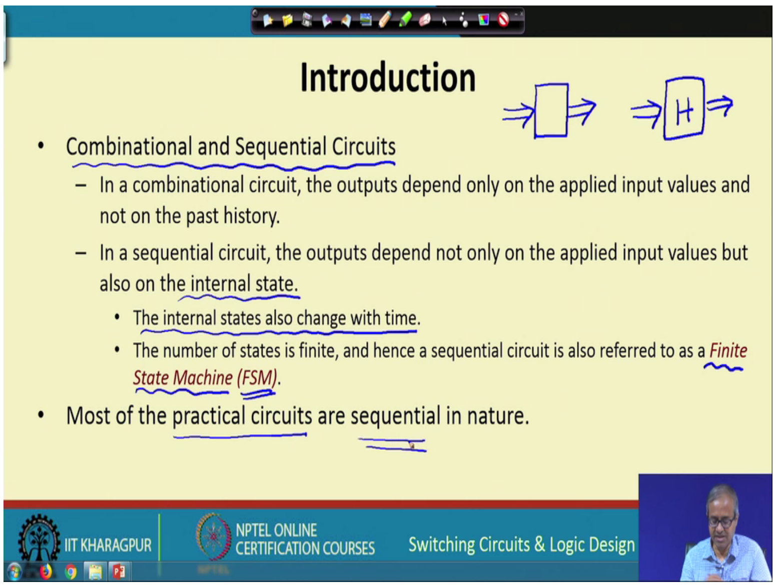
Prof. Indranil Sengupta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 36
Synthesis of Synchronous Sequential Circuits (Part I)

So, if you recall our last few lectures we have been discussing about the various sequential circuit components, which helps us in designing synchronous sequential circuits. In particular we had discussed the design of the various kinds of flip flops, how they can be excited, how the flip flops states can be changed from the various input states to the output states? And of course, we have seen the different ways in which we can convert one kind of flip flop to another.

So, now with that background let us see how we can design or synthesize general synchronous sequential circuits starting from the specification; our target is to obtain the final gate level and the flip flop level circuit from the specification. So, the title of talk today's Synthesis of Synchronous Sequential Circuits, we are starting with part 1.

(Refer Slide Time: 01:23)



Introduction

- Combinational and Sequential Circuits
  - In a combinational circuit, the outputs depend only on the applied input values and not on the past history.
  - In a sequential circuit, the outputs depend not only on the applied input values but also on the internal state.
    - The internal states also change with time.
    - The number of states is finite, and hence a sequential circuit is also referred to as a *Finite State Machine (FSM)*.
- Most of the practical circuits are sequential in nature.

H
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Design

So we have already mentioned earlier that circuits are of 2 types combinational and sequential.

So, let me repeat it once more because this difference is extremely important now what I have said a combinational circuit is one, where whatever inputs you are applying; the outputs will be depending just on those applied inputs and nothing else ok. But it contrast when you have a sequential circuit, here you are applying some inputs all right, but the outputs that are generated that are dependent not only on the inputs, but also some internal history of the system. There is a concept of a history which is maintained by the system which is called the state the state of the system. So, whatever the output you will be getting that will be depending not only on the inputs are applying, but also on which state the circuit is presently in.

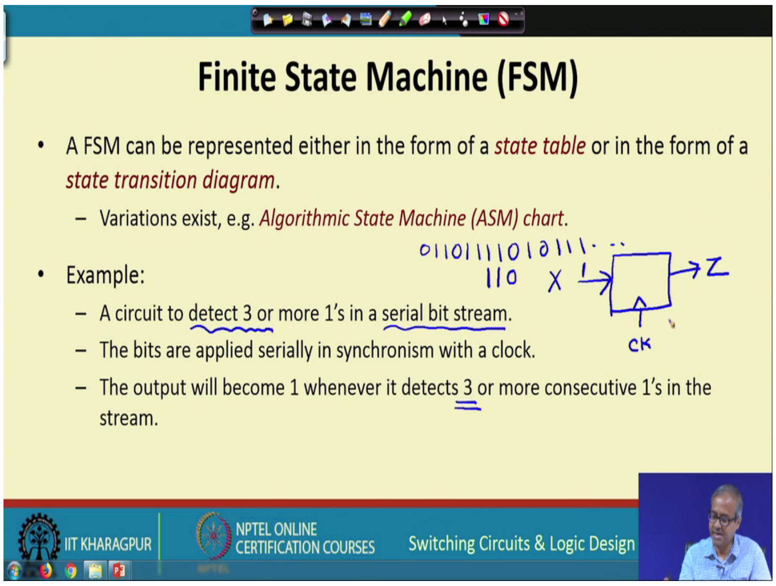
So, we mentioned very briefly earlier that the flip flops that you have studied they help in maintaining or remembering the state of such a sequential circuit or a sequential machine we sometimes call. So, whatever you upset is mentioned here in a combinational circuit; the outputs depend only on the applied inputs, but in a sequential circuit the outputs depends on the applied inputs.

And also on the internal state this is what I just mentioned; and another point to note is that as we go on applying inputs in some sequence the circuit also goes on changing its state. So, the state of the circuit or the state of the system changes with time in response to the applied inputs. So, this is an important point; the internal states of the machine or the circuit changes with time and just one thing you remember; I am saying that the system or the circuit is remembering some history that is the state of the system.

Now from a practical point of view; you just think if the number of possible states in the system is infinite can you really built such a system? You see this state or the information about the state you maintain in flip flops or a set of flip flops. If you have n number of flip flops they can be in 2 to the power n possible combinations. So, a maximum of 2 to the power n distinct states can be represented, but if we talk about infinite states; this indirectly means that we require an infinite number of flip flops which practically you really cannot built right.

So, from practical point of view we talk about something called Finite State Machines or in short FSM. Finite state machine is nothing, but a sequential circuit where the number of states or the internal history information is finite in number that is why it is called finite state machine ok. And of course, I mention this repeatedly that most of the practical circuits we see around us are sequential in nature; very rarely you will find the circuit for the output depends only on the inputs and not on the history alright.

(Refer Slide Time: 05:20)



Finite State Machine (FSM)

- A FSM can be represented either in the form of a *state table* or in the form of a *state transition diagram*.
  - Variations exist, e.g. *Algorithmic State Machine (ASM) chart*.
- Example:
  - A circuit to detect 3 or more 1's in a serial bit stream.
  - The bits are applied serially in synchronism with a clock.
  - The output will become 1 whenever it detects 3 or more consecutive 1's in the stream.

0110111101011....
110
X
1
Z
CK
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Design

So, let us look into these finite state machines in a slightly more detail; the first thing is that whenever we talk about a finite state machine or a sequential circuit; well we have to start from somewhere right. Somehow we have to specify the behavior of the circuit for example, in a combinational circuit or a combinational function we specify the behavior either by using a switching expression a function or by using a truth table and so on. So, in exactly the same way for a sequential circuit there must be some very systematic way to specify the behavior of the circuit or the system.

So, the two ways you can do that is either in the form of something called a state table or a state transition diagram. You note that state table and state transition diagonal equivalent, they represent the same information, but in two different ways. State table is a tabular representation; state transition diagram is a diagrammatic representation. So, it is fairly straight forward to convert from one form to the other.

So, whenever actually designing a sequential circuit it is good enough to start with any one of the two you do not need both, but in the examples that we shall see in some subsequent lectures; we shall be showing you both the representations just for your convenience right. There are some other ways also to represent finite state machines particularly more complex ones; like one of them is algorithmic state machine or ASM;

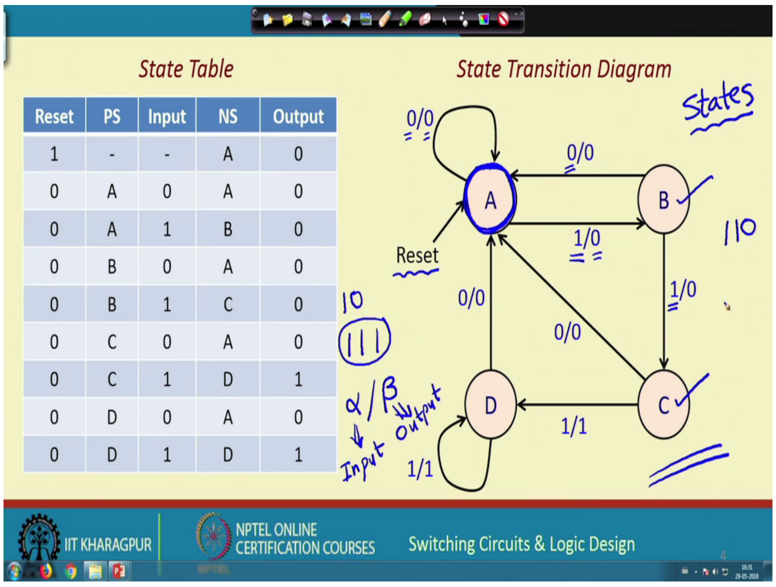
this we shall be discussing briefly later ok, but right now we shall not be considering this ASM chart in this discussion, we shall be talking about this state tables and state transition diagrams.

Let us take a very simple example to illustrate; consider that we want to build, a circuit this circuit would be like this there is a circuit which will be having a single input let us say X it will be having an output Z and of course, there will be a clock. So, what you are saying is that in this input X; we are applying a sequence of bits; that means, a serial bit stream. For example, we are applying a bit stream like let us say 0 1 1 0 1 1 1 1 0 1 0 1 1 1 and so on. Now the circuit should be such that you will be detecting 3 or more consecutive 1s in this stream. And these bits are applied in synchronism with the clock, we apply a bit, apply a clock, apply the next bit, apply a clock and so on.

So, the output will become 1 whenever it will detect 3 consecutive 1s; 3 or more in fact. So, in this case the output will be also 0; it is out here it will become 1 because there are 3 1s, there are 4 1; here also it will remain 1, well again it will become 0 and again it will become 1 out here because there is again 3 1s. So, we want to design a circuit like this right.

So, the first thing that you realize is that well this is not a very trivial or very simple kind of a specification. Somehow we have to represent or capture this information in the form of a table or as I said in the form of a diagram say transition diagram. Let us see how from this specification that we want to detect see consecutive sequence of 3 or more 1s how we can formally capture the behavior and represent it in a systematic way ok.

(Refer Slide Time: 09:56)



State Table

Reset	PS	Input	NS	Output
1	-	-	A	0
0	A	0	A	0
0	A	1	B	0
0	B	0	A	0
0	B	1	C	0
0	C	0	A	0
0	C	1	D	1
0	D	0	A	0
0	D	1	D	1

Let us look into the diagram on the right this diagram first. You see here I have shown a diagram this is called a graph where I have some nodes or vertices which are represented as circles. Here you see there are 4 such vertices A B C D; this vertices represents states of the system.

So, there are 4 states; now how many states will be there for a given problem that you will have to figure out. The designer has to figure out how many states are required; in this problem as you will see 4 states are sufficient ok, these are A B C D. And I am saying that state A is the starting state; so when you start this system it begins with state A, there can be a reset input. So, whenever you reset the system it will initialize itself into state A.

Now what I am saying is that whenever there are consecutive 3 1s the output has to be 1. So, how do you captured this in this diagram? Well we shall be discussing this in more detailed later, but let us start try explain with this example how we have constructed this diagram. Well while in state A see there are some arrows you can see these are edges this arrows connect one state to another.

And this arrows are labeled with 2 numbers let us say alpha and beta; this alpha indicates inputs that that mean what input is coming and beta indicates that what output will be generated. You see when you are in state A you are looking for 1 1 1 right, but if you see there is a 0 well you will have to remain in state A because you have not yet started the sequence; it is a 0. So, you have no information about 1 1 1, you remain in A and your output is 0 because you have not yet got 1 1 1 1. But as soon as you get the first 1; let us see this edge you get the first 1, you go to state B. State B indicates that I have found a single 1 1 1 has been found that is the meaning of state B and still the output is 0 because we have found only a single 1.

Now, while in state B; that means, you have got a 1 if you find there is again a 0; that means, again you have to start from the beginning. So, if you in state B if you see that the next bit is 0, you again go back to state A right, but if you find that there is another 1 which means now I have seen 2 1s; 1 1 I go to state C; so state C indicates that I have seen 2 1s.

Similarly, while in state C there is another 0 coming; you go back to state A, you have to search from the beginning again. But while in state C if you get another 1 which means let me clear this which means 1 1 and 1; then you go to state D with the output of 1. Because now we have found 3 consecutive 1s B C and D; D means at least 3 consecutive 1s have been found and here you see the output is also becoming 1.

Now, while you are in state D which means already 3 1s are there; if there are more 1s coming you remain in state D; this edge you remain in state D and the output continues to be 1. But whenever you find that there is 0 coming; that means, you again have to start from the beginning; you go back to A right. This is how you capture the specification of the problem in the form of a state transition diagram, where the nodes indicate the states, the edges represent the transitions.

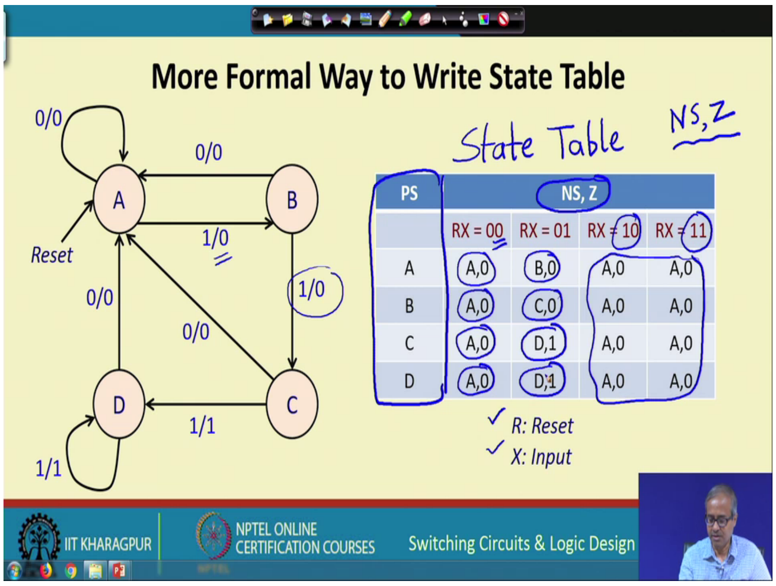
Now the same information you can capture in the form of a truth table kind of a structure; this is called a state table. Well there is another more conscious way of representing this, but I am first showing the simpler one. Let us assume that there is a reset input; so, when the reset input is 1. See here we are talking about present state, next state the applied input and the output.

So, whenever reset is activated irrespective of what the present state is the next state will be A; that means, you will be going to A and the output is 0. So, when the reset is 0 which is the normal mode of operation; that the inputs are coming one by one. Now you see this table; so when you are in A and the input is 0; according to this diagram you remain in A; you remain in A with output 0 0 slash 0.

Well if you are in A, but the input is 1; that means, you are taking this path not sorry not this path; So, the input is 1 and the output is 0 and you go to state B. So, state B with output 0; so where in state B; 0 you come back this one. If B is 1 you go to C this one. In C similarly if it is 1 you go to D, in D if it is 0 you go back to A, if D it is 1; you remain in D right.

So, these 2 are equivalent actually ok; so, you can either show it in a diagram form or in a table form. Now what I have said is that this table can be expressed more concisely or in a more compact way in this form I am showing here.

(Refer Slide Time: 16:33)



More Formal Way to Write State Table

State Table NS, Z

PS	NS, Z			
	RX = 00	RX = 01	RX = 10	RX = 11
A	A,0	B,0	A,0	A,0
B	A,0	C,0	A,0	A,0
C	A,0	D,1	A,0	A,0
D	A,0	D,1	A,0	A,0

✓ R: Reset
✓ X: Input

0/0
0/0
A
B
1/0
Reset
0/0
1/0
0/0
D
C
1/1
1/1
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Design

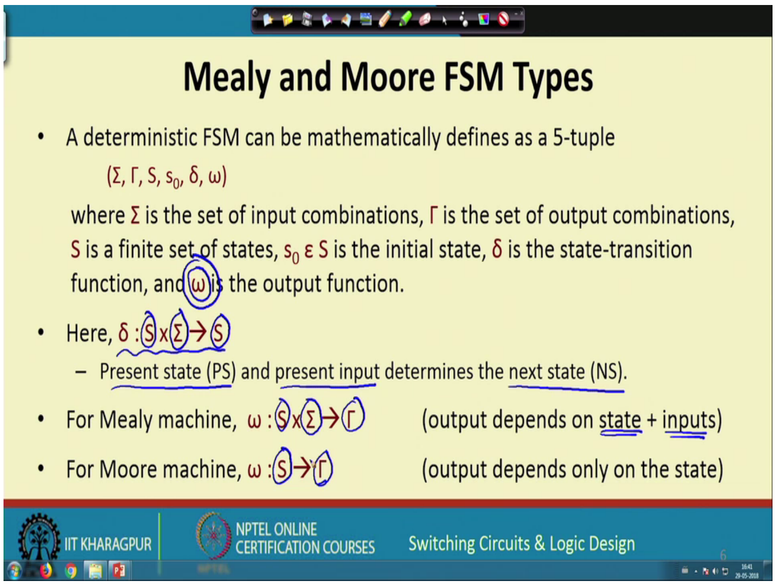
So, this state transition diagram will be showing side by side for convenience and on the right has side whatever I am showing; this is the concise state table more concise form of state table. So, let us see how we have done it; you see there are broadly 2 columns. The first column specifies the present state of the system PS; it can be A B C or D. While here while and the next part of it specifies 2 things, next state comma Z; Z is the output next state is where you are going comma Z is the output value that is being generated.

Now as you can see there are 4 columns in the second part; there is something called R X; what is R X? R represents the reset input; X represents the actual bit that is coming the actual input. So, whenever R is 1; that means, you are re setting; so, you can see the third and fourth columns where R is 1.

So, your next state is becoming A because it is reset to A and the output is also 0. So, you are resetting the system here, but when R is 0 depending on the value of X; you are taking a decision here X is 0 ok. So, when is X is 0 you see from A if the input is 0; you remain in A with output 0, you write A comma 0 this is the next state and output. So, if it is 1 here you go to B; if it is 1 you go to B with output 0.

From B if it is 0 you go to A with output 0 from B if it is 1, you come to C with output 0 in this way you go on. From C you go either to A or to D with output 1, from D again you either go to A or go to D with output 1 ok. So, this is a way to express the state table in a concise way; in all our example subsequently; we shall be using this concise notation of this state table to represent a functional behavior.

(Refer Slide Time: 19:06)



Mealy and Moore FSM Types

- A deterministic FSM can be mathematically defines as a 5-tuple

  $(\Sigma, \Gamma, S, s_0, \delta, \omega)$

  where Σ is the set of input combinations, Γ is the set of output combinations, S is a finite set of states, $s_0 \varepsilon S$ is the initial state, δ is the state-transition function, and ω is the output function.
- Here, $\delta : S \times \Sigma \rightarrow S$
  - Present state (PS) and present input determines the next state (NS).
- For Mealy machine, $\omega : S \times \Sigma \rightarrow \Gamma$ (output depends on state + inputs)
- For Moore machine, $\omega : S \rightarrow \Gamma$ (output depends only on the state)

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Design

Now, let us look into some formal definitions of finite state machines; there are broadly 2 kinds of finite state machines called Mealy machine and Moore machine. But before coming to the distinction between these 2 kind of machine, let us try to formally define, mathematically define what a finite state machine is.

Well from a designers point of view; it is a circuit, it has some memory some state. Inputs are coming, outputs are generated, states are changing this is intuitively specifying, but mathematically when you capture this information; how should you specify it is specified in this way. A deterministic finite state machine this is deterministic because whenever you are applying some inputs; the output is deterministic legitimate, there is no ambiguity.

It is mathematically determined as a a 5 tuple sigma, gamma, S, S 0, delta and omega. What that these 6 things means? This is actually 6 tuple not a 5 tuple actually 6 tuple; this sigma means this set off input combinations. Suppose this circuit has 3 input lines, there will be 8 input combinations; so, sigma will denote this set of all possible 8 input combinations.

So, sometimes we call it as the input alphabet what are the possible input combinations that are possible ok. Similarly gamma is the set of output combinations in the same way let us say if there are 4 output lines; there can be sixteen possible output combinations 2 to the power 4 that is gamma set of all possible output combinations. Capital S denotes this set of states; well it depends whenever you have drawn the finite state machine, this state transition diagram you have some idea how many states are.

So, S denotes set of the states like in the example that I have given there are 4 states A B C D ok. And small s 0 which is member of S is the initial state because you have to mention from which state you have to start alright. And of course, this is not enough, you also have to specify something called delta which is the state transition function. That means, it will tell you that if you are in a state S i and if some particular input I comes which state you will be going to S j that is defined by this function delta.

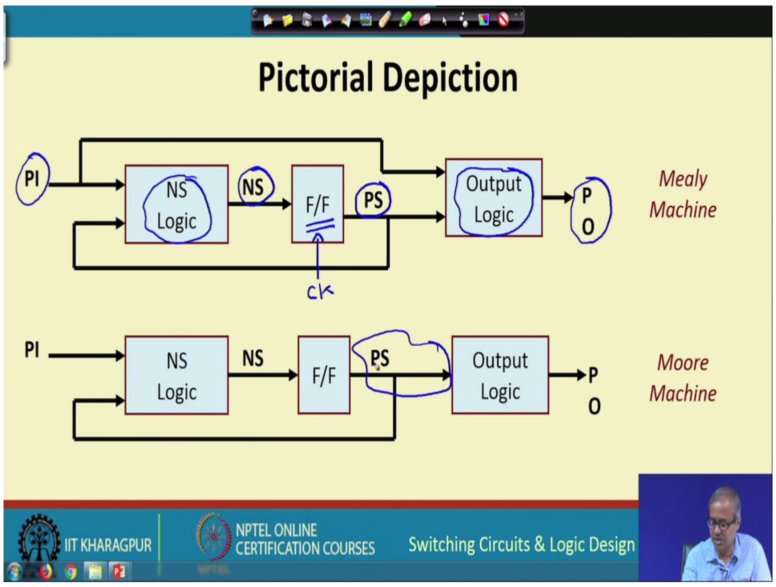
And there is another function called omega this is called the output function; it says that when you are in certain state and some input comes what should be my output? So, if we specify all this 6 things your specification of the FSM is complete right. So, delta can be mathematically expressed like this; given a state, given an input from the input alphabet it determines the next state.

Mathematically in the function notation you specified like this Present State in short PS and the present input, this alphabet determines the next state NS ok. Now, let us distinguish between Mealy and Moore; you see Mealy and Moore are identical have to delta, it is omega that distinguishes these 2 types. For a Mealy machine, the output depends on the present state and also on the present inputs; that means, the output gamma depends on S as well as input alphabet.

But in a Moore machine; Moore machine you can say is a subset of a Mealy machine. In a Moore machine, the output depends only on the present state; it does not depend on the input. So, the output is determined only by the present state; there are many systems, where the output that is being generated is not determined by what input you are applying; rather than which state you are in. Well a classy example is a traffic light controller where there are some states which we are going through and depending on which state you are in; so, either the red or the yellow or the green lights are glow.

So, it is not that you are applying some input depending on that you decide which light to glow not exactly like that ok. So, Mealy and Moore machine are two different types of machines these are the formal definitions.

(Refer Slide Time: 24:40)



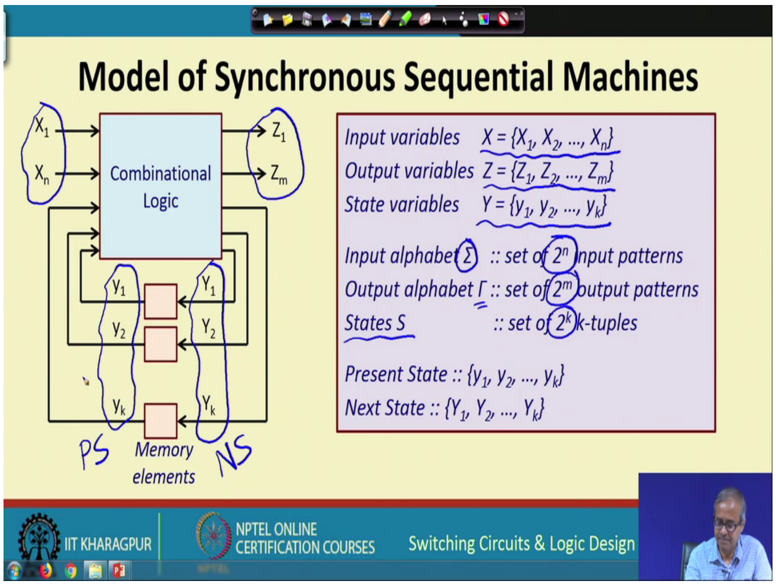
Pictorial Depiction

So, pictorially speaking the Moore machine and Mealy machine can be depicted like this. For a Mealy machine the primary inputs are coming like this PI is the Primary Inputs, PS is the Present State, NS is the Next State. Now you see in any circuit realization there will be a set of flip flops, which will be storing the present state the outputs of the flip flops that will define the PS ok.

And there will be some combinational circuit here which will take the primary inputs and also this present state and it will determine the next state. Well one thing I am not shown this there it is the clock the clock is also there. So, whenever the clock comes the next state will get stored in the flip flop and it will now become the present state ok.

And there is another combinational circuit call the output logic which again takes the PI and the present state as input and generates the primary outputs PO. For a Moore machine; the first part is the same, but for output logic it takes only one input the PS that it does not take PI as the as the other input. The output depends only on the present state this is how you distinguish between these 2 types of machines.

(Refer Slide Time: 26:12)



Model of Synchronous Sequential Machines

Input variables $X = \{X_1, X_2, \ldots, X_n\}$
Output variables $Z = \{Z_1, Z_2, \ldots, Z_m\}$
State variables $Y = \{y_1, y_2, \ldots, y_k\}$

Input alphabet Σ :: set of 2^n input patterns
Output alphabet Γ :: set of 2^m output patterns
States S :: set of 2^k k-tuples

Present State :: $\{y_1, y_2, \ldots, y_k\}$
Next State :: $\{Y_1, Y_2, \ldots, Y_k\}$

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES Switching Circuits & Logic Design

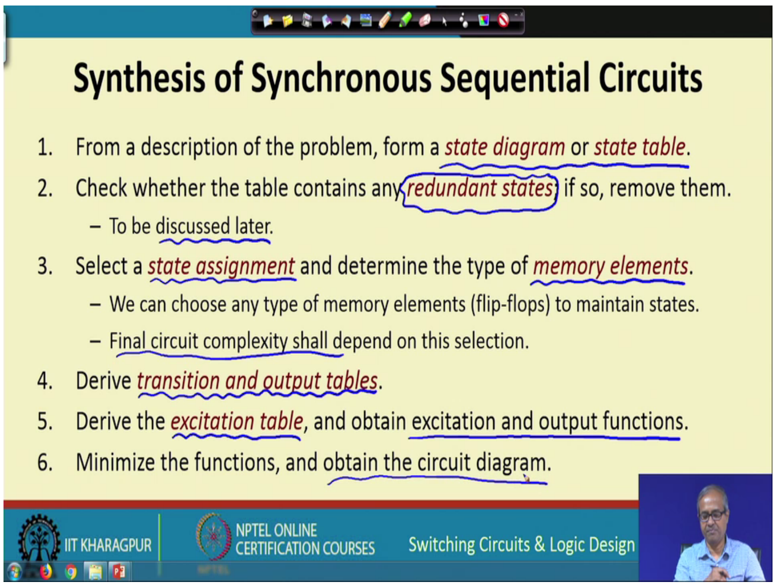
Now, when you talk about a synchronous sequential machine; a synchronous sequential machine is a sequential machine which is synchronized by a clock. So, whenever the clock signal comes the active clock edges comes; the states keep on changing. The flip flops are triggered by some leading edge or falling edge of the clock only then some changes will take place.

So, pictorially speaking; so, a synchronous sequential machine can be model like this; there are n input lines. Let us say the input variables X 1 to X n there are m output lines; this Z 1 to Z m, there are state variable let us say there are memory elements or flip flops these are flip flops.

There are k number of flip flops; so, I am saying that there are k number of state variables which are y 1 to y k small y 1 to small y k; these are called state variables. Now according to our definition of FSM the input alphabet consisting of all possible input combinations; because there are any inputs there will be 2 to the power n such combinations. Similarly output alphabet there are m outputs all 2 to the power m combinations and this states there are k number of state variable; so, there will be 2 to the power k states.

So, as this diagram shows this small y 1 to y k; this is my present state and the capital Y 1 to Y K which is fed to the input of the memory elements; this will be my next state. So, the clock is not shown when the clock comes the next state will become my present state right copy to the present state ok.

(Refer Slide Time: 28:17)



Synthesis of Synchronous Sequential Circuits

1. From a description of the problem, form a *state diagram* or *state table*.
2. Check whether the table contains any *redundant states*; if so, remove them.
   - To be discussed later.
3. Select a *state assignment* and determine the type of *memory elements*.
   - We can choose any type of memory elements (flip-flops) to maintain states.
   - Final circuit complexity shall depend on this selection.
4. Derive *transition and output tables*.
5. Derive the *excitation table*, and obtain excitation and output functions.
6. Minimize the functions, and obtain the circuit diagram.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Design

So, now, let me just tell you about the steps that are required when you synthesize a synchronous sequential circuit. These steps are as follows; first you start with the description of the problem for any problem you have to start with the description. You will be forming the state diagram or the state table this is the first step this is 1 way to formally capture the specification of the system.

Then this second step we shall not be considering now; this we will discuss later because this table or the diagram may contain some redundant states; that means, you can remove or reduce the number of states in some way. So, we shall learn later how to reduce the given state table or a state transition diagram or in other ways to minimize it we shall see it later.

So, this is an optional step we skip this for now for the time being; then we do something called stat assignment. What is state assignment? In the example that have seen so far we name the states are A B C and D, but ultimately this states will be represented by flip flops. So, the 4 states can be represented in binary; let us say using 0 0 0 1 1 0 1 1 that is called state assignment. Each state is assigned a binary code and also you decide what kind of memory elements will be choosing J K, D, T or SR ok.

And of course, the final circuit complexity will depend on the selection and we shall see how to do it. From this state assignment and selection of memory elements you first determine something called transition and output tables and then the excitation table. And from the excitation table you use some minimization technique; you start with something called excitation and output functions; minimize them and finally, obtain the circuit diagram.

Now, we shall be looking at a number of examples to illustrate this process. So, that it becomes quite clear to you right, but this is the overall steps for any kind of synthesis of synchronous sequential circuits; that you are trying to attempt. So, with this we come to the end of this lecture where we have talked about something about finite state machine FSMs, the different types of FSMs.

And finally, we talked about the steps that are required to synthesize a circuit starting from a FSM specification. In the next few lectures, we shall be looking at some examples and see how this synthesis is actually carried out.

Thank you.