INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

NPTEL

NPTEL ONLINE CERTIFICATION COURSE
An Initiative of MHRD

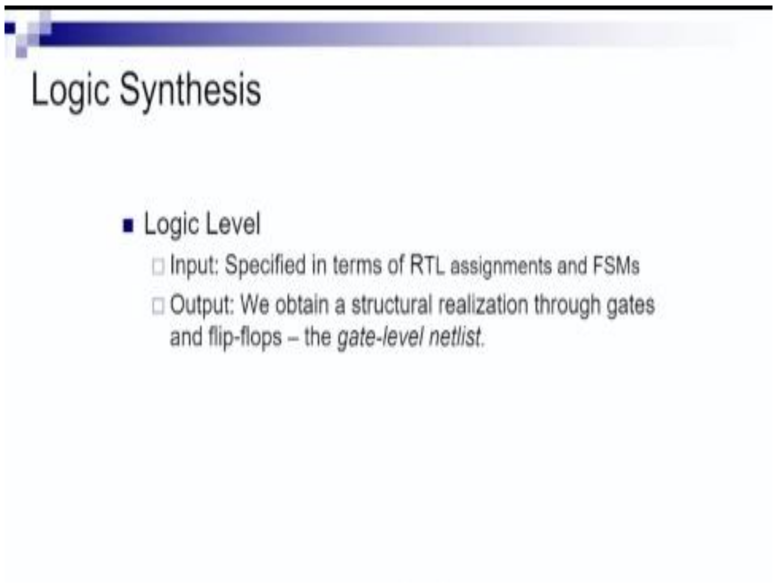
VVLI Design, Verification & Test

Dr. Arnab Sarkar
Department of CSE
IIT Guwahati

Logic Synthesis

You welcome in this module we will discuss logic synthesis as mentioned in the introductory lectures logic synthesis comes after the high level synthesis step the objective of high level synthesis was to update a macro level architecture of the circuit in terms of functional units number of registers numbers and types of Max's de Max's buses etc… The objective of the logic design or logic synthesis on the other hand is to obtain the micro level architecture of the circuit in terms of gates and flip-flops.

(Refer Slide Time: 01:05)

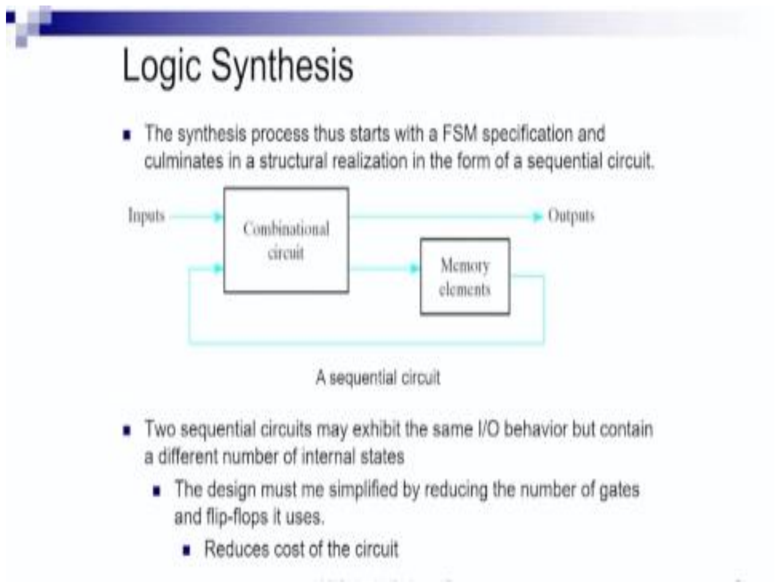


Logic Synthesis

- Logic Level
  - Input: Specified in terms of RTL assignments and FSMs
  - Output: We obtain a structural realization through gates and flip-flops – the *gate-level netlist*.

So the input to the logic synthesis step are in terms of the RTL assignments and F essence so at the output of the high level synthesis step we obtained a set of RTL assignments register transfer level assignments and we said that the controller FSM will control at which time step which of these are tail assignments are to be excited or activated now these comes this if this controller FSL and the RTLF assignments come as input to the logic synthesis step and the output to the logic synthesis step is a structural realization of the circuit through gates and flip-flops and the data structure that is used to represent such as structural realization is called a gate level net list.

(Refer Slide Time: 01:58)



Logic Synthesis

- The synthesis process thus starts with a FSM specification and culminates in a structural realization in the form of a sequential circuit.

Inputs
Combinational circuit
Outputs
Memory elements

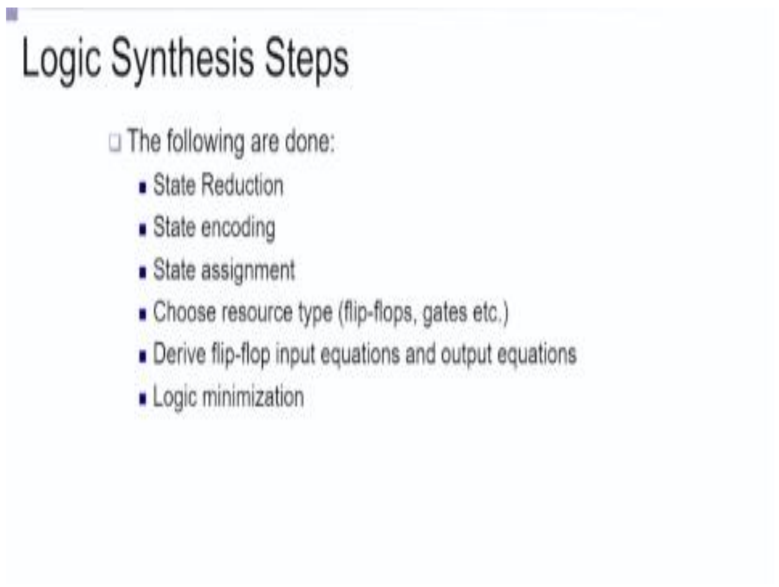
A sequential circuit

- Two sequential circuits may exhibit the same I/O behavior but contain a different number of internal states
  - The design must me simplified by reducing the number of gates and flip-flops it uses.
    - Reduces cost of the circuit

So logic synthesis process starts with an FSM specification and culminates in a structural realization in the form of a sequential circuit as we know a sequential circuit consists of memory elements along with optional combinational elements as well this figure shows the overall structure of any sequential circuit. It has an optional combinational part it has memory elements and outputs. The outputs are also optional the inputs to the memory elements and the outputs are a function of the outputs of the memory elements which are fed back to the combinational part and also the external inputs.

The memory elements are realized through flip flops however realizing a sequential circuit design in terms of gates and flip-flops also requires many optimization steps and why are essentially these optimizations required because the great level design has also many many choices in terms of the delay performance and power requirements of the circuit fundamentally two sequential circuits may exhibit the same your behavior but contain a different number of internal states so therefore the design must be simplified by reducing the number of gates and flip-flops now basically reducing the number of states also reduces the cost of the circuit.

(Refer Slide Time: 02:40)

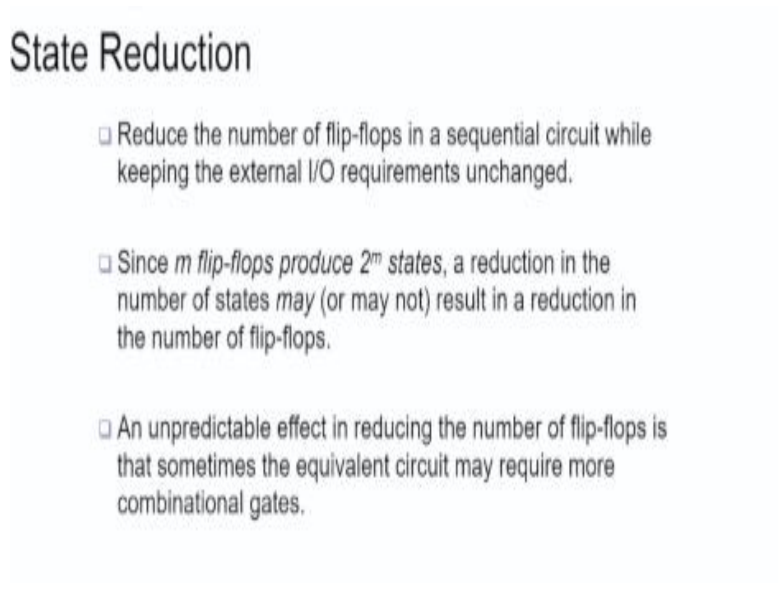


Logic Synthesis Steps

- The following are done:
  - State Reduction
  - State encoding
  - State assignment
  - Choose resource type (flip-flops, gates etc.)
  - Derive flip-flop input equations and output equations
  - Logic minimization

So what are the basic steps in logic synthesis the first state reduction then state encoding and assignment then we have to choose resources the types of flip-flops and gates to use then we have to derive flip clock input equations and output equations for the type of flip-flop that we have chosen and we have to do logic minimization after all these steps we will obtain a structural realization of the circuit

(Refer Slide Time: 04:08)



State Reduction

- Reduce the number of flip-flops in a sequential circuit while keeping the external I/O requirements unchanged.
- Since *m flip-flops produce 2^m states*, a reduction in the number of states *may* (or may not) result in a reduction in the number of flip-flops.
- An unpredictable effect in reducing the number of flip-flops is that sometimes the equivalent circuit may require more combinational gates.

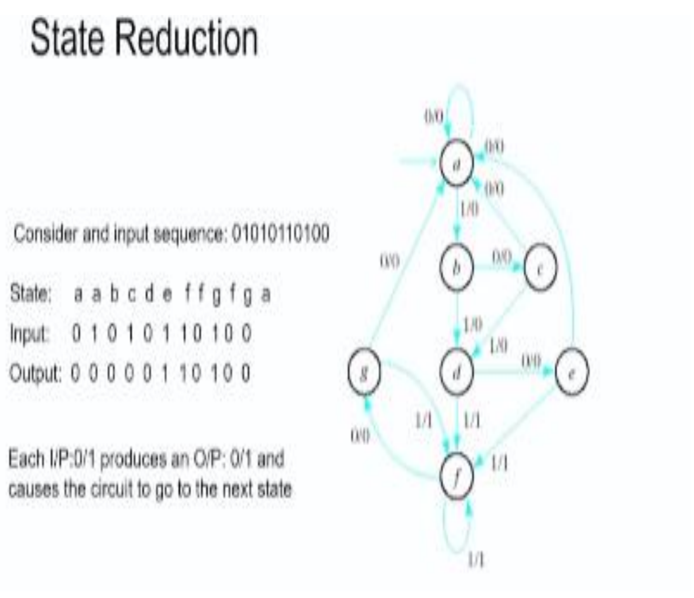
So the first step state reduction so what is the objective reduce the number of flip-flops in a sequential circuit while keeping the external are your requirements unchanged so we want to reduce the number of states we said that there can be two different realizations with two different number of states for the same input output behavior since M flip-flops produce 2 to the power M States a reduction in the number of states may result in a reduction in the number of flip-flops so we know that each flip-flop can hold one bit of information through one flip-flop.

I can define two states and hence to obtain a structural realization of an FSM which has two to the pod am states we require at least em flip flops however reduction of states may or may not result in the reduction in the number of flip-flops for example let us say if the number of states in an FSM is between 8 and 16 we require at least four flip-flops so if we reduce the number of states from say 15 29 we will still require for flip-flops to obtain a structural realization of this FSM.

And hence we won't obtain a reduction in the number of lip flops however if we if you are still using that same for flip-flop design and then we have done a state reduction and this production has resulted in the number of states reducing from 12 to seven then previously if you were using for flip-flops now we will be able to realize the same design using three flip-flops and unpredictable effect in reducing the number of flip-flops is that sometimes the equivalent circuit may require more combinational gates and why is that.

So because I have reduced the number of states we require more combinational logic to obtain the inputs and hence the resulting circuit may have a bigger area may require a bigger area than the circuit which used more flip-flops now we will understand how state production is realized we will take this sample FSM for understanding how state reduction is done.

(Refer Slide Time: 06:30)



State Reduction

Consider and input sequence: 01010110100

State: a a b c d e f f g f g a

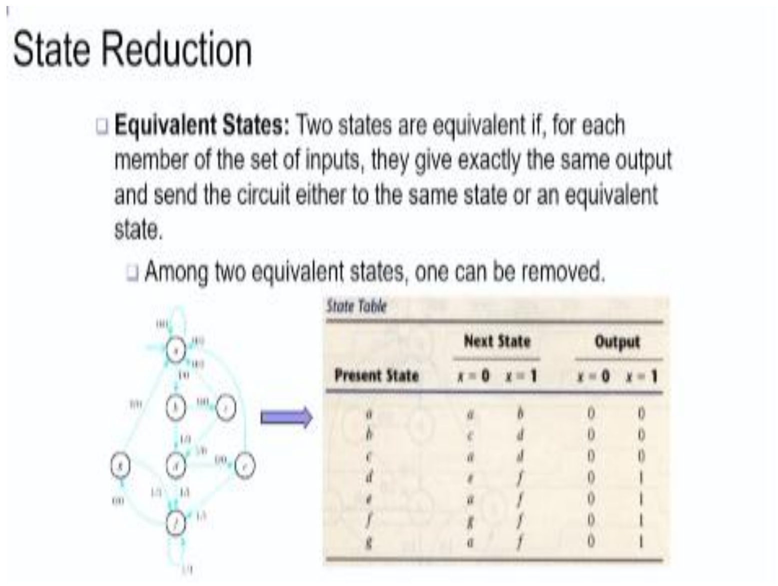
Input: 0 1 0 1 0 1 1 0 1 0 0

Output: 0 0 0 0 0 1 1 0 1 0 0

Each I/P:0/1 produces an O/P: 0/1 and causes the circuit to go to the next state

So in this FSM we see that is the initial state this FSM consists of seven states A, B, C, D, E, F, G on this FSM we will now take a sample sequence and see what is the input output sequence and what are the next stage so the initial state is a will we get the input you know who produce the output zero and go to the next state if the current state is a and the input is 1 we produce the output 0 and go to the next end B.

(Refer Slide Time: 07:09)



State Reduction

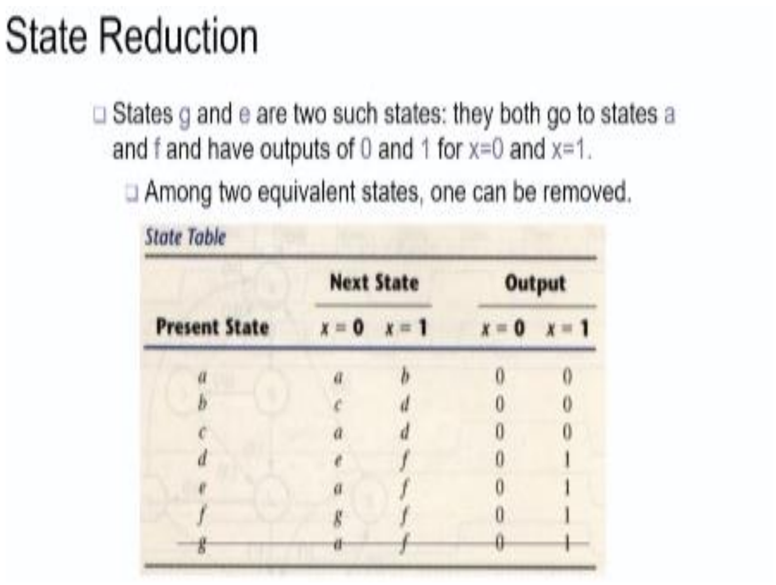
- **Equivalent States:** Two states are equivalent if, for each member of the set of inputs, they give exactly the same output and send the circuit either to the same state or an equivalent state.
  - Among two equivalent states, one can be removed.

State Table

	Next State		Output	
Present State	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	*a*	*b*	0	0
b	*c*	*d*	0	0
c	*a*	*d*	0	0
d	*e*	*f*	0	1
e	*a*	*f*	0	1
f	*g*	*f*	0	1
g	*a*	*f*	0	1

Now how do we do state reduction we do state production by finding out equivalent states to states are equivalent if for each member of the set of inputs they give exactly the same output and send the circuit either to the same state or an equivalent state so let us say we have two arbitrary states x and y what do we have to do we have to find out that for all possible inputs here we have two inputs 0 and 1 do they go on each of these inputs to the same mix tape and do the produce the same output if that is so then we can say that these two states are given and if two states are equivalent over all sets of inputs they produce the same next state and same output then we can remove one of these states this is how we do state reduction.

(Refer Slide Time: 08:15)



State Reduction

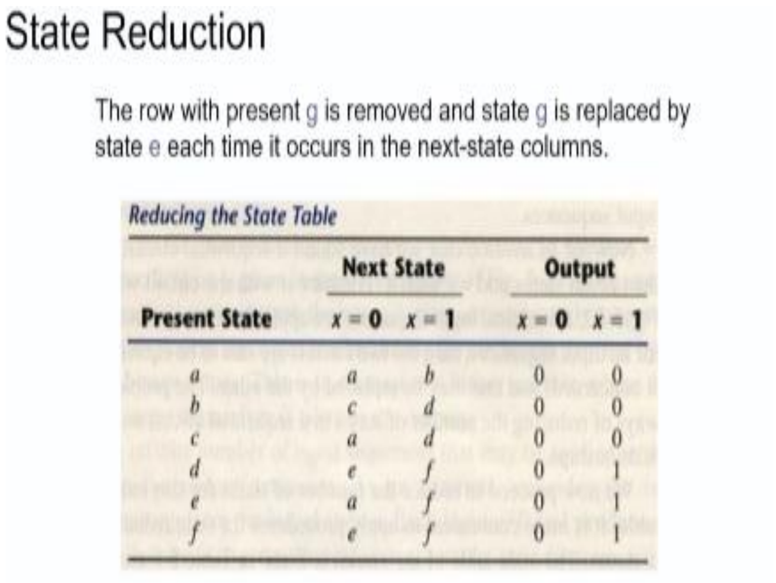
- States g and e are two such states: they both go to states a and f and have outputs of 0 and 1 for x=0 and x=1.
  - Among two equivalent states, one can be removed.

State Table

	Next State		Output	
Present State	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	*a*	*b*	0	0
b	*c*	*d*	0	0
c	*a*	*d*	0	0
d	*e*	*f*	0	1
e	*a*	*f*	0	1
f	*g*	*f*	0	1
~~*g*~~	~~*a*~~	~~*f*~~	~~0~~	~~1~~

In this FSN that we have we obtain the corresponding state table first here there are two states G and E which are equivalent and why are the equivalent because for both these states when the input is x equals x equals to 0 they go to the same next state a when the input is 1 on the other hand they go to the same next state F when the input is X the same both of them produce the same output and when the input is x equals to 1 they again produce the same output 1 so in terms of both next states and outputs they are same and hence one of these states here G we have decided to remove.

(Refer Slide Time: 09:03)



State Reduction

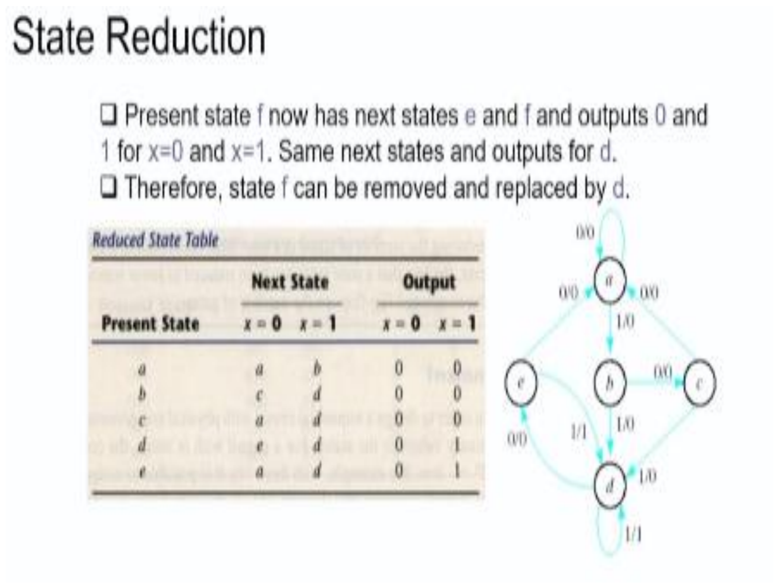
The row with present g is removed and state g is replaced by state e each time it occurs in the next-state columns.

Reducing the State Table

	Next State		Output	
Present State	**x = 0**	**x = 1**	**x = 0**	**x = 1**
a	*a*	*b*	0	0
b	*c*	*d*	0	0
c	*a*	*d*	0	0
d	*e*	*f*	0	1
e	*a*	*f*	0	1
f	*e*	*f*	0	1

So the row with present state G is removed and state G is replaced by state E each time it occurs in the next state column so when G is removed g cannot appear in the next state next eight columns anymore and so wherever g was previously there if you see the previous state here at F we had G so we have to replace this g with the e because for each place in the next eight columns when g occurred we have to now replace it with E and then we have to continue the same thing.

(Refer Slide Time: 09:45)



State Reduction

- ❑ Present state f now has next states e and f and outputs 0 and 1 for x=0 and x=1. Same next states and outputs for d.
- ❑ Therefore, state f can be removed and replaced by d.

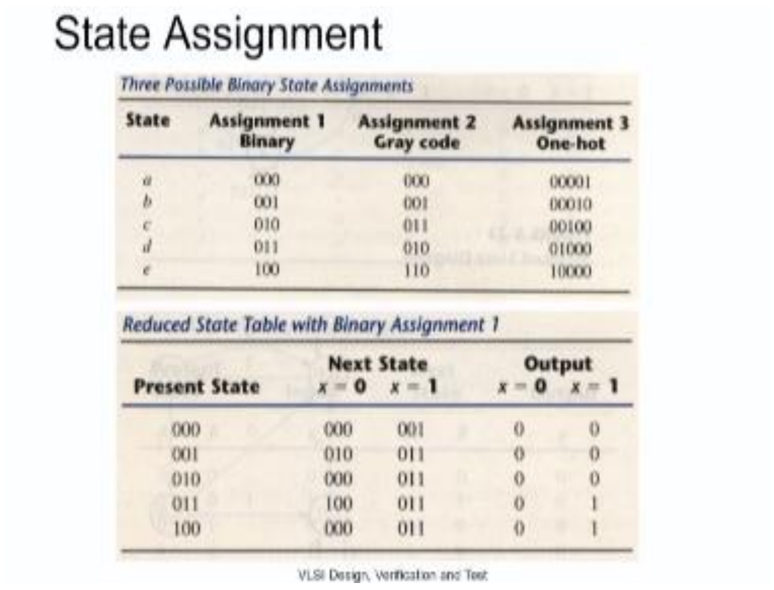
Reduced State Table

	Next State		Output	
Present State	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	*a*	*b*	0	0
b	*c*	*d*	0	0
c	*a*	*d*	0	0
d	*e*	*d*	0	1
e	*a*	*d*	0	1

After G is removed and G is has been replaced by E in the next eight columns we see that the present state F now has next States E and F and output 0 and 1 for x equal to 0 and x equals 21 same next states and outputs for D as well so what we are trying to point is that D and F are equivalent things we can remove one of them here we have decided to remove f so if we remove F then all the places in the next eight columns where F is currently there has to be changed to with D.

So likewise we see that here we have deep and here again we have D so therefore state F can be removed and replaced by D we finally obtain this realization of the FSM it is a reduced FSM the original FSM contains seven states the changed FSM car now contains five states. However we will see that for all sequences of inputs it is going to produce the same sequences of outputs and hence both the input-output behavior of both these officiants are identical.

(Refer Slide Time: 11:09)



State Assignment

Three Possible Binary State Assignments

State	Assignment 1 Binary	Assignment 2 Gray code	Assignment 3 One-hot
a	000	000	00001
b	001	001	00010
c	010	011	00100
d	011	010	01000
e	100	110	10000

Reduced State Table with Binary Assignment 1

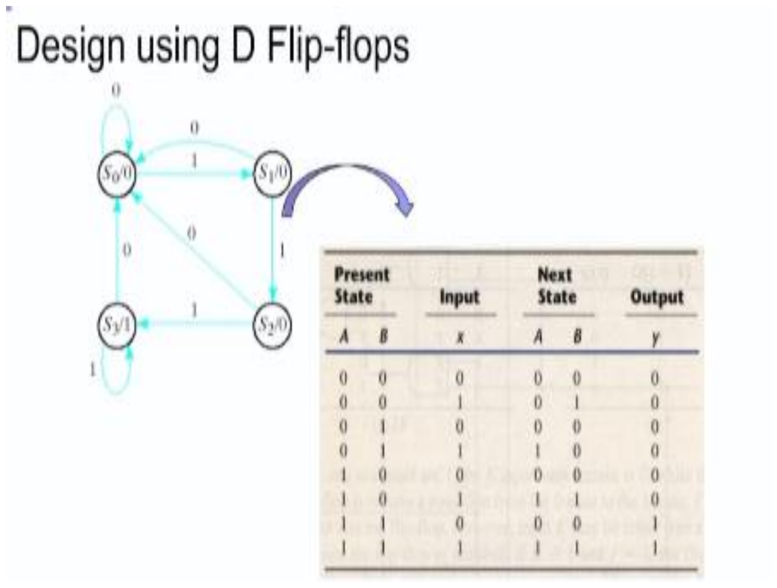
	Next State		Output	
Present State	$x = 0$	$x = 1$	$x = 0$	$x = 1$
000	000	001	0	0
001	010	011	0	0
010	000	011	0	0
011	100	011	0	1
100	000	011	0	1

VLSI Design, Verification and Test

After obtaining such a state reduction we will now have to do state as Hymen an now a structurally realization cannot have use the letter names for the state's ABCDE we have to use binary coded names and therefore we have to use an encoding scheme and encode the states with that particular encoding scheme. Here we have shown three distinct encoding schemes so this one is a simple binary encoding this one is gray code and this one is one hot encoding and in this diagram.

We have shown that with simple binary encoding this is what we get when the input is A we have represented a with 0000 and the and the present state is X equals to 0 we remain at the same state A and hence the output state is also 0 0 0 right the present state is 0 0 which is A and the next state will also be A and hence this is also 0 0. When the input x equals to 1 it goes to state B and produces output 0 so therefore B has been encoded with 0 01 and the output produced is 0 the encoding scheme is this A is 0 0 0, B 0 01 , C is 0 1 0 , D is 0 1 1 and E is 100 this is the simple encoding scheme that we have used ok and after we have obtained

(Refer Slide Time: 12:59)



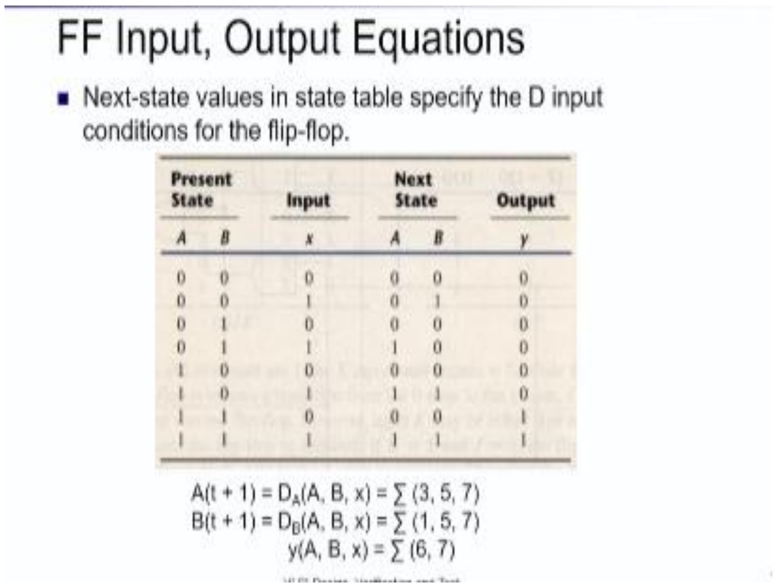
Design using D Flip-flops

Present State		Input	Next State		Output
A	*B*	*x*	*A*	*B*	*y*
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	1	1	1

The state encoding now we have a reduced FSM with binary coded encoding scheme, and now we are ready to obtain a design. The next work is to choose flip-flops and here we have chosen D flip-flops to design ok we have taken another sample FSM this is a mood type FSM the output is produced at the states here is an assignment using the simple binary scheme because this is a four-state FSM it can be realized using to flip flops and the state encoding that we have chosen is simple binary and a zero here is 0 0 S 1 is 0 1, S 2 is 10 and S 3 is 11 right.

And here is the state table corresponding to the state diagram here and when the present state A B is zeros you know that is we are at n 0 and the input is 0 we remain at 0 0 that is in s 0 and the output produced again is zero when the present state is 0 0 and the input is 1 we produce the next 8 0 1 that is we go to s1 and the output is again 0 at s 1 against a at 0 when the present state is 01 at s 1 when the input is 0 we go back to state s 0 and the output is 0 and likewise we proceed.

(Refer Slide Time: 14:40)



FF Input, Output Equations

- Next-state values in state table specify the D input conditions for the flip-flop.

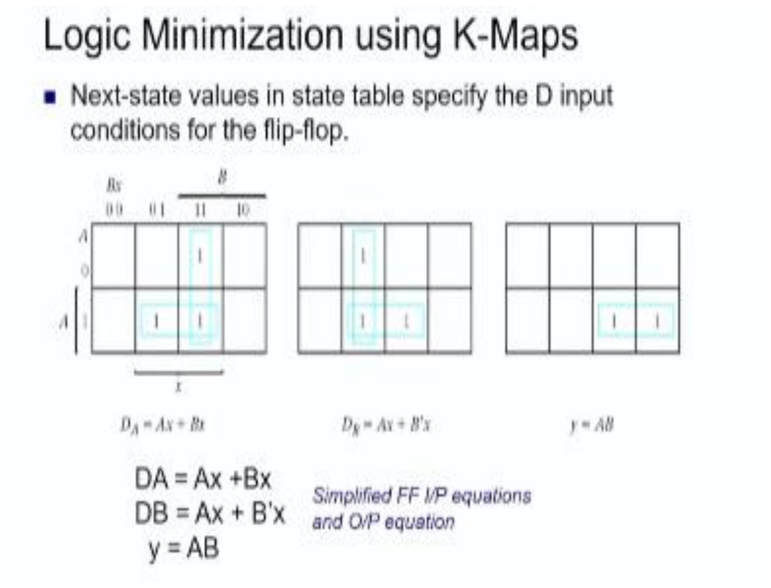
Present State		Input	Next State		Output
A	*B*	*x*	*A*	*B*	*y*
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	1	1	1

$$A(t+1) = D_A(A, B, x) = \sum(3, 5, 7)$$
$$B(t+1) = D_B(A, B, x) = \sum(1, 5, 7)$$
$$y(A, B, x) = \sum(6, 7)$$

Now from this state table we can obtain the flip-flop input equations and the output equations now because this is a D flip-flop it is simple to obtain because the next state value is in the state table specify the D input conditions for the flip-flop and hence the next state values tell me the D input conditions right so we want to obtain the input conditions for the D flip-flops and the output right so 80 plus 1 equals two D A B X equals 2 3 5 7 min term number three min term number five and min term number seven produce a 1 here right we see that this one this one and this one producer one so min term number 3 5 and 7 producer 1 for B in the we see that a 1 is produced here and here so at min term number 1 min term number 5 and min term number 7.

So the output is produced for 1 5 7 and the output Y A B X equals to 6 7 why because the output is 1 in these two in these two rows so this is min term number 6 and min term number 7. So after we have obtained the conditions for which the flip-flop, inputs and the outputs should be 1.

(Refer Slide Time: 16:22)



Logic Minimization using K-Maps

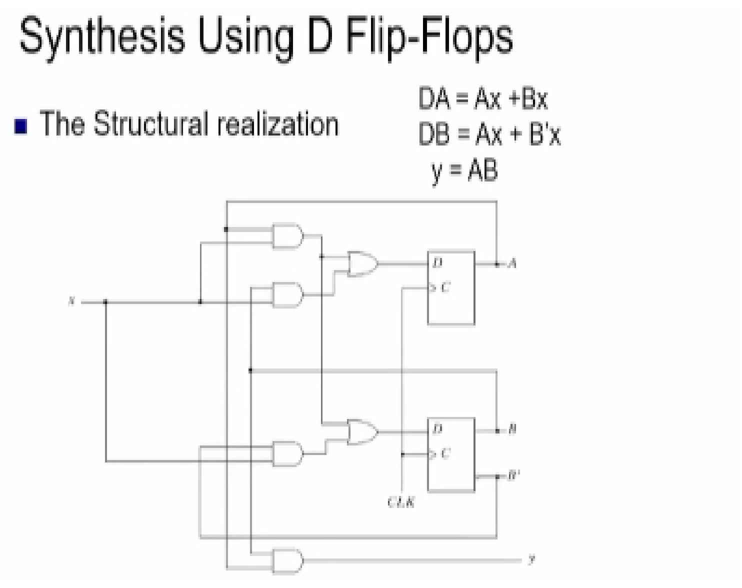
- Next-state values in state table specify the D input conditions for the flip-flop.

DA = Ax +Bx
DB = Ax + B'x
y = AB

Simplified FF I/P equations and O/P equation

We will try to do a logic minimization here we have shown a very simple logic minimization using Karnaugh maps so next state values in the state table specify the D input conditions for the flip-flop and we know that if because it is 3 5 7 this one is 0 1 1 3 this one is 1 0 1 which is 5 and this one is 1 1 1 which is 7 and after simplification of this we find that D A equals to X plus B X likewise we obtain for D B DB equals to x plus b bar x and y equals to A B. So this is how we obtain the input equations and output equation.

(Refer Slide Time: 17:24)



Synthesis Using D Flip-Flops

- The Structural realization

$DA = Ax + Bx$
$DB = Ax + B'x$
$y = AB$

After obtaining the input equation equations and output equation. We can obtain a structural realization of the circuit here we see that this one produces A X this one produces be B X this one produces B and this one is X and hence this gate produces B X and this D equals to AX plus BX. Similarly this year for this input we see that it is equals to AX plus B bar x and y equals to A B we come to the end of this module.

Queen Barman
Rekha Hazarika

CET Administrative Team
Susanta Sarma
Swapan Debnath