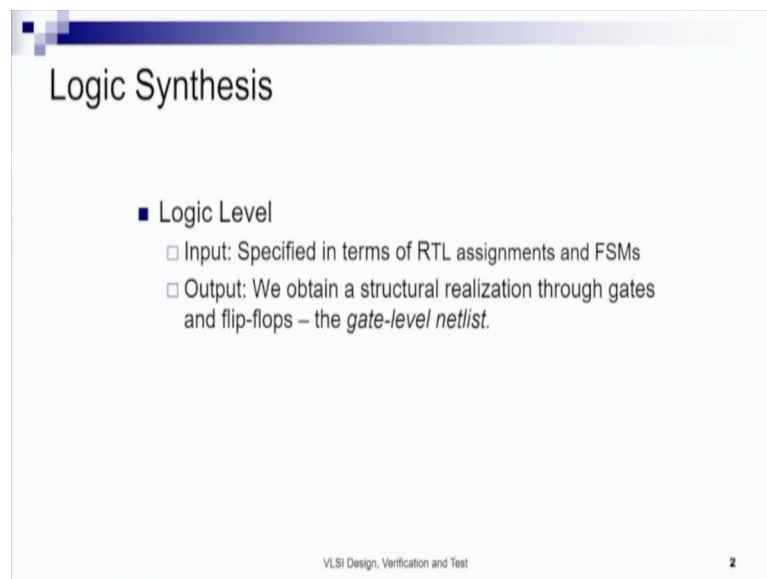


Embedded Systems – Design Verification and Test
Dr. Santosh Biswas
Prof. Jatindra Kumar Deka
Dr. Arnab Sarkar
Department of Computer Science and Engineering
Indian Institute of Technology, Guwahati

Lecture – 13
Logic Synthesis

Welcome in this module we will discuss Logic Synthesis, as mentioned in the introductory lectures logic synthesis comes after the high levels in the system. The objective of high level synthesis was to obtain a macro level architecture of the circuit in terms of functional units, number of registers, numbers and types of muxs, demuxs, buses etcetera. 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: 01:08)



Logic Synthesis

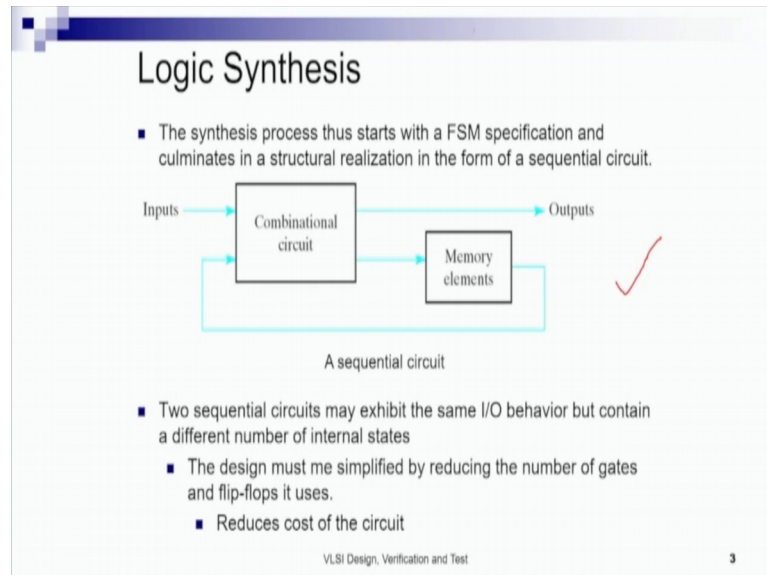
- Logic Level
 - Input: Specified in terms of RTL assignments and FSMs
 - Output: We obtain a structural realization through gates and flip-flops – the *gate-level netlist*.

VLSI Design, Verification and Test 2

So, the input to the logic synthesis step are in terms of the RTL assignments and FSMs. 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 RTL assignments are to be excited or activated. Now, these comes these if this controller FSM and the RTL F 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 a structural realization is called a gate level netlist.

(Refer Slide Time: 02:02)



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 gate level design has also many choices in terms of the delay performance and power requirements of the circuit. Fundamentally two sequential circuits may exhibit the same IO 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.

(Refer Slide Time: 03:42)

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

Now, basically reducing the number of states also reduces the cost of the circuit. 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 flop 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: 04:12)

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 IO 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^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 1 bit of information, through 1 flip flop I can define 2 states. And hence, to obtain a structural realization of an FSM which has 2^m states we require at least m 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 4 flip flops.

So, if we reduce the number of states from say 15 to 9, we will still require 4 flip flops to obtain a structural realization of this FSM. And hence we would not obtain a reduction in the number of flip flops; however, if we if you are still using that same 4 flip flop design and, then we have done a state reduction and this reduction has resulted in the number of states reducing from 12 to 7, then previously if you were using 4 flip flops now we will be able to realize the same design using 3 flip flops.

An 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.

(Refer Slide Time: 06:35)

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 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

VLSI Design, Verification and Test 6

We will take this sample FSM for understanding the how state reduction is done. So, in this FSM we see that a is the initial state this FSM consists of 7 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 states. So, the initial state is a we will be get the input 0 we produce the output 0 and go to the next state a. If the current state is a and the input is one, we produce the output 0 and go to the next state b.

(Refer Slide Time: 07:13)

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.

Present State	Next State		Output	
	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

VLSI Design, Verification and Test 7

Now, how do we do state reduction? We do state reduction by finding out 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. 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 2 inputs 0 and 1 do they go on each of these inputs to the same next state, and do they 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: 08:14)

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

Present State	Next State		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
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

VLSI Design, Verification and Test 8

In this FSM that we have we obtain the corresponding state table first here, there are two states g and e which are equivalent and why are they 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: 09:07)

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

Present State	Next State		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
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

VLSI Design, Verification and Test 9

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 columns anymore. So, wherever g was previously there, if you see the previous state g here at f we had g . So, we have to replace this g with the e , because for each place in the next state columns where g occurred we have to now replace it with e and then we have to continue the same thing.

(Refer Slide Time: 09:49)

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

Present State	Next State		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
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

VLSI Design, Verification and Test 10

After g is removed and g is has been replaced by e in the next state columns, we see that the present state f now has next states e and f and output 0 and 1 for x equals to 0 and x equals to 1. Same next states and outputs for d as well so, what we are trying to point is that d and f are equivalent hence, 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 state columns where f is currently there has to be changed with d.

So, likewise we see that here we have d 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 7 states, the changed FSM can now contains 5 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 FSMs are identical.

(Refer Slide Time: 11:13)

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

Present State	Next State		Output	
	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

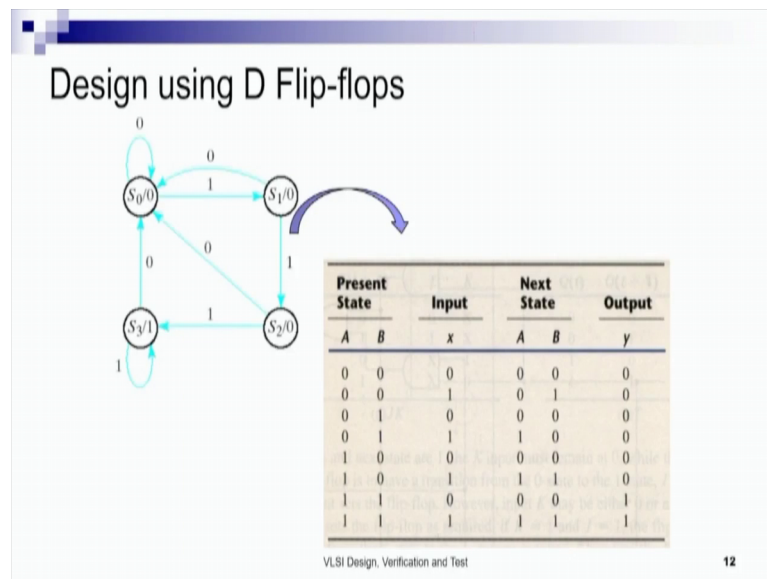
VLSI Design, Verification and Test 11

After obtaining such a state reduction we will now have to do state assignment. Now, a structural realization cannot use the letter names for the states a b c d e 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 grey 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 0 0 0 0 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 0 1 and the output produced is 0, the encoding scheme is this, a is 0 0 0 b is 0 0 1 c is 0 1 0 d is 0 1 1 and e is 1 0 0, this is the simple encoding scheme that we have used ok.

(Refer Slide Time: 13:03)



And, after we have obtained 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 two flip flops. And the state encoding that we have chosen is simple binary and S 0 here is 0 0 S 1 is 0 1 S 2 is 1 0 and S 3 is 1 1 right. And, here is the state table corresponding to the state diagram here, and when the present state a b is 0 0 that is we are at S 0 and the input is 0, we remain at 0 0 that is in s 0 and the output produced again is 0, when the present state is 0 0 the and the input is 1, we produce the next state 0 1 that is we go to S 1 and the output is again 0, at S 1 again say

at 0 when the present state is 0 1 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: 14:45)

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		A	B	
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

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

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

$$y(A, B, x) = \sum (6, 7)$$

VLSI Design, Verification and Test 13

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 values 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, you want to obtain the input conditions for the D flip flops and the output right. So, $A(t+1) = D_A(A, B, x) = \sum (3, 5, 7)$ min term number 3, min term number 5 and min term number 7 produce a 1 here right. We see that this one this one and this one produce a 1 so, min min term number 3 5 and 7 produce a 1 would be when 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 $y(A, B, x) = \sum (6, 7)$, 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: 16:26)

Logic Minimization using K-Maps

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

$D_A = Ax + Bx$ $D_B = Ax + B'x$ $y = AB$

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

Simplified FF I/P equations and O/P equation

VLSI Design, Verification and Test 14

We will try to do 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 and 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 Ax plus Bx, likewise we obtain for DB DB equals to Ax 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: 17:27)

Synthesis Using D Flip-Flops

- The Structural realization

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

VLSI Design, Verification and Test 15

After obtaining the input equations and output equation, we can obtain a structural realization of the circuit here, we see that this one produces Ax , this one produces Bx , this one produces Bx , this one is B and this one is x and hence this gate produces Bx it and this D equals to Ax plus Bx . Similarly this a 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.