

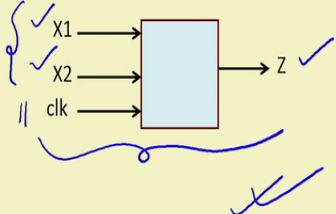
**Switching Circuits and Logic Design**  
**Prof. Indranil Sengupta**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology Kharagpur**

**Lecture - 38**  
**Synthesis of Synchronous Sequential Circuits (Part III)**

So, in the last lecture we were talking about the method of Synthesizing a Synchronous Sequential Circuits and you have seen the first step of it, starting from the specification how to construct the state transition diagram and state tables. Today we shall be looking at a complete worked out example starting from the state table or state transition diagram how we can go through the other steps and arrive at our final circuit diagram. So, this is the third part of our lecture on Synthesis of Synchronous Sequential Circuits.

(Refer Slide Time: 00:56)

**Example 1: Serial Adder**



**State Transition Diagram**

```
graph LR
    A((A)) -- "00/0  
01/1  
10/1" --> A
    A -- "11/0" --> B((B))
    B -- "00/1" --> A
    B -- "01/0  
10/0  
11/1" --> B
```

**State Table**

PS	NS, Z			
	X = 00	X = 01	X = 10	X = 11
A	A, 0	A, 1	A, 1	B, 0
B	A, 1	B, 0	B, 0	B, 1

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

## FSM Synthesis

- We have seen how the state transition diagram / state table can be constructed from the problem description.
- To construct the circuit, the following further steps are required:
  - a) Assign unique binary code to each state: state assignment.
  - b) Construct the transition/output table.
  - c) Select type of memory elements (SR or JK or D or T) and construct excitation table.
  - d) Obtain the excitation and output functions, and minimize them.
  - e) Realize the functions using gates or any other combinational circuit modules.
- We shall illustrate the process with examples.

A	00	001
B	01	010
C	10	100

IIT KHARAGPUR

NPTEL ONLINE  
CERTIFICATION COURSES

Switching Circuits & Logic Design

So, let us recapitulate what we are said about the synthesis of FSMs. In the last lecture we have seen through a number of examples how we can construct the state transition diagram and also the state table starting from the problem description.

But now the question arises from the state table what next. So once you have constructed the state table there are a few steps that are left to be done; these I shall be illustrating with the help of examples from this lecture onward. The first step that we shall go through now is something called state assignment. State assignment means that you are assigning some unique binary code to the states assign unary unique binary code to the states.

Now the way you do state assignment can be different for example, if you have 3 states let us say ABC, then you can just assign the binary code 0 0 0 0 1 or 1 0 let us say or you can use something called one hot encoding, that means one of the bits in the state representation is 1. But of course, here you require 3 state variables, there can be many other alternatives ok, but in the examples that we shall be showing we shall be looking at the most compact representation, say for 3 state variables 3 states, we need 2 bits or 2 state variables to represent.

Now after state assignment we will be constructing something called the transition and output table. Now this can be constructed directly from the state table as we shall see from this state table after state assignment we shall be going to the transition and output table. Now once transition output table is done we will be selecting the type of memory

elements that what kind of flip flop we are choosing and accordingly we will be constructing something called the excitation table.

Now, you recall when we had discussed the various kinds of flip flops, we talked about the excitation requirement of the flip flop; for example, for a T flip flop if you want to go from state 0 to state 1 you have to apply T equal to 1. So, for every pair of present state and next state you know what exactly you have to apply to the inputs of the flip flop.

So, accordingly you create or construct something called the excitation table and from that table you can obtain some functions one is called excitation function other is called output functions and once you get them you can minimize them and once you have minimized you can realize them using gates or using any other modules as you feel like. Now let us illustrate this steps that I have mentioned through some examples.

So, the example that we take in this lecture is that of a serial adder which we have already discussed in the last lecture. So, in the last lecture whatever we have discussed is shown in the slide. So we have said that this is our functional depiction of the serial adder, there are 2 serial inputs  $X_1$  and  $X_2$  this is the serial output  $Z$  and of course there is clock. State transition diagram state table of course, for synthesis the steps we shall be using the state table representation.

So, state diagram we are not looking at right now we shall be looking at state table. So, just let us recapitulate once in the state table. So, in the state table we have the present state specified in the first column and the next state and the output specified in the second column for all possible values of the inputs because, here we have 2 inputs  $X_1$  and  $X_2$  there can be 4 different input combinations. So, capital X is the combination of  $X_1 X_2$ , it can be 0 0 0 1 1 0 1 1. So, state table depicts the whole behavior. Now, let us see how we can proceed with the next steps.

(Refer Slide Time: 06:12)

• State assignment:

- Two states, and so one bit is sufficient.
- Suppose we assign 0 for state A, and 1 for state B.

PS	NS, Z			
	X = 00	X = 01	X = 10	X = 11
0	0, 0	0, 1	0, 1	1, 0
1	0, 1	1, 0	1, 0	1, 1



*After state assignment*

PS	NS				Output Z			
	00	01	10	11	00	01	10	11
0	0	0	0	1	0	1	1	0
1	0	1	1	1	1	0	0	1

*Transition/output Table*

$A \rightarrow 0$   
 $B \rightarrow 1$

*X<sub>1</sub>, X<sub>2</sub>*



 Switching Circuits & Logic Design

So, as I said first step will be the state assignment, because in this example there are 2 states, so we can use a single bit. Let us say state A may be represented by 0 and state B can be represented by 1, well you remember this is not unique this is just one of the state assignment we can try out. So, you can have other state assignments also you can have the reverse 1 for A and 0 for B also fine.

Now, with this state assignment your state table becomes like this, you see this is exactly the same as the state table. What we have done? We have replaced A by 0 and wherever B was there we have replaced it by 1. The here everything have remains same you see here A and B was there we replace it by 0 and 1 here also A was there here also A and B was there here also A and B was there here also B and B was there, the rest you have not touched.

(Refer Slide Time: 07:37)

- State assignment:
  - Two states, and so one bit is sufficient.
  - Suppose we assign 0 for state A, and 1 for state B.

PS	NS, Z			
	X=00	X=01	X=10	X=11
0	0,0	0,1	0,1	1,0
1	0,1	1,0	1,0	1,1

$X_1, X_2$

PS	NS				Output Z			
	00	01	10	11	00	01	10	11
0	0	0	0	1	0	1	1	0
1	0	1	1	1	1	0	0	1

After state assignment
Transition/output Table

NS, Z

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Design

Now just one thing you just see that in this state table every entry of the state let us say here here every entry, so what are we specifying we are specifying 2 things, we are specifying the next state as well as we are specifying the output separated by commas, but for convenience let us separate these out in 2 different tables; let the NS be there in one table and Z with there in another table. It is again not anything different, but instead of writing them in a single table separated by commas for convenience let us separate them out and this is called transition and output table.

You see the first part of the table is for NS second part of the table is for the output Z exactly the same table you see the inputs the NS 0 0 0 1 0 1 1 1 1 you see 0 0 0 1 0 1 1 1 and the outputs 0 1 1 0 1 0 0 1 0 1 1 0 1 0 0 1. So, it is exactly the same thing right and these this X value instead of X, I am also just writing here 0 0 0 1 1 0 1 1.

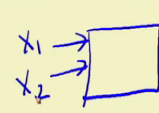
(Refer Slide Time: 08:56)

- State assignment:
  - Two states, and so one bit is sufficient.
  - Suppose we assign 0 for state A, and 1 for state B.

PS	NS, Z			
	X=00	X=01	X=10	X=11
0	0,0	0,1	0,1	1,0
1	0,1	1,0	1,0	1,1

PS	NS				Output Z			
	00	01	10	11	00	01	10	11
0	0	0	0	1	0	1	1	0
1	0	1	1	1	1	0	0	1

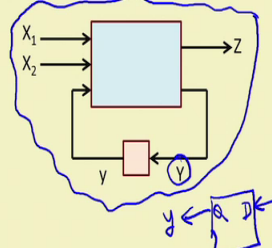
After state assignment
Transition/output Table



These are actually the value of X which means X 1 X 2 because, you recall the 2 inputs of a circuits are X 1 and X 2, the 2 serial inputs that are coming those are X 1 and X 2 right.

(Refer Slide Time: 09:16)

- Select memory element, and construct excitation/output function.
  - Suppose we use D flip-flop.



PS	NS				Output Z			
	00	01	10	11	00	01	10	11
0	0	0	0	1	0	1	1	0
1	0	1	1	1	1	0	0	1

Excitation/Output Table

Now, the next step is to select memory elements. Now, it is a choice we do not know which one will be best. I can try out D flip flop T flip flop JK SR. let us start with D, because it is a simplest kind of flip flop. So, let us select D flip flop. Now our model of

the FSM was like this, there are 2 inputs  $X_1$   $X_2$  single output  $Z$  this was the next state capital  $Y$  present state small  $y$ .

Now, when we are using D flip flop what we are actually trying to do is that here instead of this pink box we will be using a D flip flop D and Q where in D we will be applying some value and this Q will be generating this small  $y$  right. So, in the original table we had the capital  $Y$ , but when we convert it into something called excitation and output table, see this was the table which we had seen earlier transition and output table just in last slide we had seen this.

But from this after we have selected D flip flop, now we are coming to something which is called excitation and output table. Now excitation output table will look exactly same as this because, in case of a D flip flop; so whatever you are applying same thing will be coming out just like your model here so there will be no change.

So, here actually whatever you are showing here they indicate the value of D. So, instead of  $Y$  see here  $Y$  so  $Y$  you can call as if you are applying the D value here. So, whatever is D that you are applying here, there will be the same thing, because this capital  $Y$  becomes small  $y$  whenever clock comes for a D flip flop same thing happens. So, whatever you are applying to D that goes inside when the clock comes ok. So, you have the excitation and output table like this.

(Refer Slide Time: 11:58)

The slide displays two truth tables for a 2-input, 1-output logic function. The first table shows the next state  $Y$  as a function of inputs  $X_1, X_2$  and present state  $y$ . The second table shows the output  $Z$  as a function of inputs  $X_1, X_2$  and present state  $y$ . A logic diagram shows a combinational logic block with inputs  $X_1, X_2$  and output  $Z$ , and a D flip-flop with input  $D$  and output  $Y$ . The flip-flop output  $Y$  is connected to the D input of the flip-flop.

$X_1 X_2$	00	01	11	10
$y=0$			1	
$y=1$	1	1	1	

$X_1 X_2$	00	01	11	10
$y=0$		1		1
$y=1$	1		1	

$Y = X_1 X_2 + X_1 y + X_2 y$ 
 $Z = X_1 \oplus X_2 \oplus y$

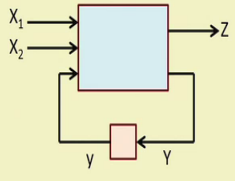
$Z = x_1' x_2' y + x_1' x_2 y' + x_1 x_2' y' + x_1 x_2 y$

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

Now from the excitation and output table if you just look at it just let me go back ones, if you just look at this excitation and output table you see that what are the inputs here.

(Refer Slide Time: 12:18)

• Select memory element, and construct excitation/output function.  
 – Suppose we use D flip-flop.



PS	NS				Output Z			
	00	01	10	11	00	01	10	11
0	0	0	0	1	0	1	1	0
1	0	1	1	1	1	0	0	1

y	NS (Y)				Output Z			
	00	01	10	11	00	01	10	11
0	0	0	0	1	0	1	1	0
1	0	1	1	1	1	0	0	1

*Handwritten notes: X1, X2 above the second table; y, Y, Z with arrows pointing to the respective tables.*

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

Here my inputs are on one side I have my input small y and on the other side here I have my inputs X 1 and X 2. So, 0 0 0 1 1 0 1 1 so if you just interchange these 2 columns it will become like a corner map right, in corner map you have 0 0 0 1 1 1 and 1 0. So, if you just reverse it, it will be like a corner, similarly for the output part it will be like a corner map ok. because you already have these you already have this, if you just interchange the last 2 columns it will become just like a corner map and for the first part you are actually generating the function for this small y whatever this small this actually capital not small y capital Y and here you are actually generating Z. So, if you just work this out the same thing I have mentioned, the corner maps will look like this, for the first one the corner map will look like this.

So, if you try to minimize it there will be 3 cubes one like this one like this and one like this. So, if you see there will be 3 terms, here I have shown X 1 X 2 on this side and small y on this side and for the second part this for the output function the 1's are like this, you see there are no cubes possible you cannot minimize it, so this is actually the exclusive OR functions. So, I have shown in an compact form in an exclusive OR form, but actually but if you just want to write it in the expanded form it will be like this; this 1 will correspond to X 1 bar X 2 bar y this 1 will correspond to X 1 bar X 2 y bar OR this



1 will correspond to  $X_1 X_2 \bar{y}$  OR and this 1 will correspond to  $X_1 X_2$  and  $y$ ; this is nothing but the XOR of  $X_1 X_2$  and  $y$ .

So, you see once you have generated these functions  $Y$  and  $Z$ , this is actually a D flip flop you have chosen here. So, you have as good as designed the circuit because, from this function you can directly generate the value of  $X$  because here this will be nothing but an exclusive OR gate XOR gate  $X_1 X_2$  and  $y$ , this will be  $Z$  and for generating  $Y$  you will be using a circuit like this there will be 3 AND gates and an OR gate this will be generating capital  $Y$ .

(Refer Slide Time: 15:44)

The slide displays two truth tables and two logic diagrams. The first truth table is for function  $Y$  with inputs  $X_1, X_2$  and  $y$ . The second truth table is for function  $Z$  with the same inputs. The logic diagram for  $Z$  shows an XOR gate with inputs  $X_1, X_2$  and  $y$ . The logic diagram for  $Y$  shows three AND gates with inputs  $(X_1, X_2)$ ,  $(X_1, y)$ , and  $(X_2, y)$  respectively, followed by an OR gate.

$X_1, X_2$	00	01	11	10
0			1	
1		1	1	1

$$Y = X_1 X_2 + X_1 y + X_2 y$$

$X_1, X_2$	00	01	11	10
0		1		1
1	1		1	

$$Z = X_1 \oplus X_2 \oplus y$$

So, the inputs will be  $X_1, X_2, X_1 y$  and  $X_2 y$ . So, for generating the output  $Z$  you need an XOR function as you can either use an XOR gate or you can break it up into AND or NOT gates and for generating the next state capital  $Y$  you need 3 AND gates and an OR gate right. This is actually how you do the synthesis the basic idea is this.

(Refer Slide Time: 16:27)

• Alternate design.  
 – Suppose we use SR flip-flop

PS	NS				Output Z			
	00	01	10	11	00	01	10	11
0	0	0	0	1	0	1	1	0
1	0	1	1	1	1	0	0	1

Y	SR				Output Z			
	00	01	10	11	00	01	10	11
0	0X	0X	0X	10	0	1	1	0
1	00	00	00	00	1	0	0	1

Now, the same design let us try out an alternative, suppose instead of D flip flop we try with SR flip flop right. Now if we use SR flip flop actually what we are trying to do? Now, see earlier we had this D flip flop there are single input single output Y, but now you see if you consider SR flip flop here, now there will be not 1 but 2 inputs SR on one side and here the output will be Y. Now this circuit actually now will be have to generate both S and R because this capital Y is the FSM model but when you are mapping this memory element to an SR flip flop, this capital Y will get split into 2 inputs S and R right.

So, from the transition and output table when you map it to this, just see how this table is getting constructed. Second part is identical, second part there is no change this is just the outputs. But for the next state here the values that are shown are the values of S and R just see it carefully. Look at this your present state was 0 next state is 0, for an SR flip flop you recall what are the excitation function for SR flip flop, just recall for SR flip flop what do you have to apply.

Suppose I want to go from 0 to 0, for SR flip flop I can either apply 0 0 or I can apply 0 1 which means 0 do not care, but if I want to go from 0 to 1 there is only one way 1 0. If I want to go from 1 to 0, I have to apply 0 1, but if I want to go from 1 to 1 0 0. So, now you see here you are going from 0 to 0, 0 to 0, 0 to 0 all 3 see you see 0 X, 0 X, 0 X, 0 to 0 is 0 X 0 to 1 0 to 1 is 1 1 0 then 1 to 0 1 to 0 ok.

Actually 1 to 0 this will be 0 1 not 1 0 1 to 0 and rest are 1 to 1, 1 to 1, 1 to 1. So, it will be 0 0 0 0 0 0 so you see here all again you have constructed this specification for the map from where you can minimize, see you can again minimize using corner map from here.

(Refer Slide Time: 19:39)

The slide displays three Karnaugh maps for variables  $X_1, X_2$  and  $y$ . The first map shows a '1' at  $(y=0, X_1X_2=11)$ . The second map shows '1's at  $(y=0, X_1X_2=01)$ ,  $(y=0, X_1X_2=11)$ ,  $(y=1, X_1X_2=00)$ , and  $(y=1, X_1X_2=10)$ . The third map shows 'X's at  $(y=0, X_1X_2=00)$ ,  $(y=0, X_1X_2=01)$ , and  $(y=0, X_1X_2=10)$ . A handwritten 'S' is next to the first map with the expression  $X_1X_2y' + X_1'X_2'y$ . A handwritten 'R' is next to the third map with  $R=0$ . A logic circuit shows  $X_1$  and  $X_2$  entering a block that outputs  $Z$ , with feedback loops  $S$  and  $R$ . A handwritten box contains six circles arranged in two rows of three.

Just I am showing you just 1 thing just you can rectify this error because, this 1 0 would actually be 0 1 so we can make it correct. So, what we do? The first part of the table which contains the pairs 2 those we split it up into S and R the first one we used for S second one we used for R right.

So, you make that correction without this correction I am just showing you just to illustrate what you be done. So, you just see where the ones are you put them in the k map minimize them you will get S, you will get R and in this case you will get Z or the output and this will be the function you can implemented. But if you want to just try to rectify that confusion which was there just a second let us go back, so from this let us try to construct the map from here.

(Refer Slide Time: 20:53)

• Alternate design.  
 – Suppose we use SR flip-flop

PS	NS				Output Z			
	00	01	10	11	00	01	10	11
0	0	0	0	1	0	1	1	0
1	0	1	1	1	1	0	0	1

Y	SR				Output Z			
	00	01	10	11	00	01	10	11
0	0X	0X	0X	10	0	1	1	0
1	10	00	00	00	1	0	0	1

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES Switching Circuits & Logic Design

Let us construct the map Y here and on this side it will be X 1, X 2. So, it will be 0 0 0 1 1 1 1 0 and 0 1, this is for S ok, let us say this for S, S is the first one first 0 1. So, 1 you just only note down ones and 1 these 2 ones are there for S there is a there is a 1 here and 1 in there. And for R what will happen for R? For R the corner map will be the right side there are no ones there is no 1, so R will be 0.

So, R you do not apply anything, and S there will be 2 terms there will or of these 2 and output similarly output in the same way it will be a corner map you can just do it. I am sorry this will be 1 will be here because it is 1 1 it will be here. So, in this way you can just generate the functions and after you generate this functions, you can actually minimize them and after minimization, you can realize this circuit like this, this is the basic idea how you do it.

So, here I have just worked out just one example that of a 2 bit serial adder and at some more examples I shall be explaining in our next lecture also. So, we come to the end of this lecture. If you recall we have worked out the complete synthesis flow with the help of a simple example that of a serial adder.

We had seen earlier in our last lecture how we can construct the state transition diagram and the state table, and from the state table we looked at the various steps that you need to do started from state assignment, transition output table, excitation table, then

minimizing using the corner maps and so on and get the final circuit. So, we shall be working out some more examples in the next couple of lectures.

Thank you.