

**INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI**

**NPTEL  
NPTEL ONLINE CERTIFICATION COURSE  
An Initiative of MHRD**

**VLSI Design, Verification & Test**

**Dr. Santosh Biswas  
Department of CSE  
IIT Guwahati**

**Module XI: Built in Self test (BIST)**

**Lecture II: Built in Self Test-2**

Okay, so welcome to lecture 2 of module 11 in testing. So build in self test so in the last lecture we are looking at the this case that with the assumption that I mean whatever test pattern or whatever testing we are discussing at length went for offline testing, so what you mean by off line testing that is cycle test fabricated, it is put in the ATA and we are applying test patterns to test it.

But then we also saw that nowadays due to the deep of micro design what happens that even if the circuit is fine when we have sold it off or when it has been shift to the vendor I mean shift perform the vendor through the customer even after putting into this is system it may have failures, because of the deep design so all these type of soft failure and what you can called insito failures and it is seen in the picture.

So to have them what the idea was that to test that so what we did is we found that we can develop technique in which case what we can do we put a miniature version of the test pattern generator and a miniature version or what you can called subset version of the test pattern analyser or test pattern compactor. So now everyday when you take starts your operation what you do you activate your test pattern generator and that is the miniature version which is kept on chip and you use your random test pattern comparator to find out if anything is gone wrong.

So if you can do that at every start of your circuit actually we called as it as a built in self test. So in the last lecture we have seen that what we basically require in this BIST architecture, so

we require test controller, we require a test pattern generator and its size should be or area should be very less because it is on chip circuit.

So that area constant is very high. So we have seen that we can use the linear feedback shift registers and we have seen as standard LFSR and how using a series of simple EXOR two input EXOR gates we can easily develop near exhausted kind of test pattern, they are just like excluding the case of all 0s we can generate pattern from 1 to  $2^{n-1}$  kind of stuff and a subset of them are required for testing.

And the area requirement is nothing but single XOR gate as we are seen in the example in the end of last lecture or it can be attenuate our few EXOR gates. So and that is much, much less than the gate required by deterministic test pattern generator that will like you count like 1, 3, 5, 7, 9, 11 and 32 something like that.

So if you go for deterministic test pattern generator like this so on chip circuit will be very high because you have to go for final state machine design or gate optimization and all and finally you can see that you will require much more than few XOR gate which is the case of standard feedback LFSR or the standard LFSR or any other type we will see.

The basic idea is that sugar random test pattern generation using linear feedback shift registers are much more area compare to standard deterministic shift test pattern generator by using the final state machine approach of digital electronics if you do. So because of this low area requirement of LFSRs we find them are very good candidate for this pattern generator in case of build in self test.

(Refer Slide Time: 03:16)

**Standard LFSR: Example**

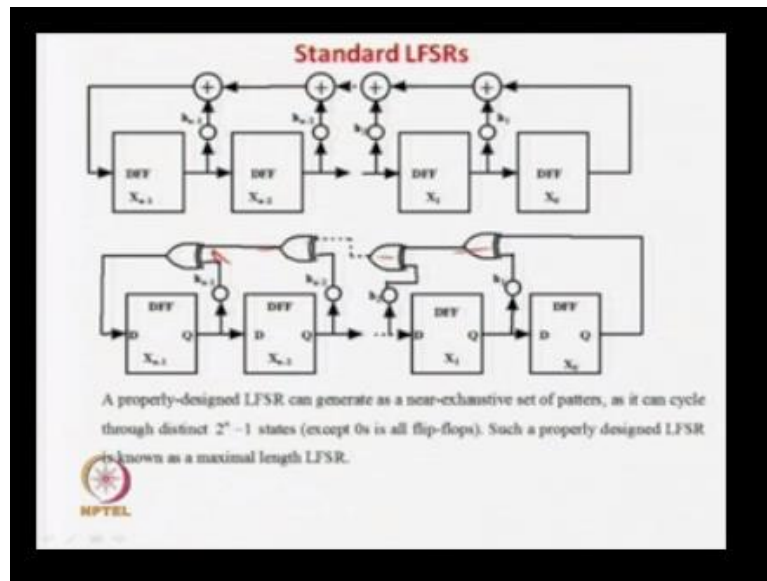
$$\begin{bmatrix} X_0(t+1) \\ X_1(t+1) \\ X_2(t+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_0(t) \\ X_1(t) \\ X_2(t) \end{bmatrix}$$
$$\begin{bmatrix} X_0 \\ X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ \vdots & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

So the LFSR generates 7 patterns (excluding all 0s) after which a pattern is repeated. It may be noted that this LFSR generates all patterns (except all 0s) which are generated by a 3 bit counter, however, the area of the LFSR is much lower compared to a counter. In a real life scenario, the number of inputs of a CUT is of the order of hundreds. So LFSR has minimal area compared to counters (of order of hundreds).

So in the last lecture if you remember so we have seen the something kind of LFSR of this nature which we called as standard LFSR. Today we will see another type of LFSR, but why do we require another type of LFSR we are just quickly get them modification so what was the idea is that you can see that in case of standard LFSR there is chain of XOR gates like this if there is also another feedback from the XOR gate so you require another XOR gate over here.

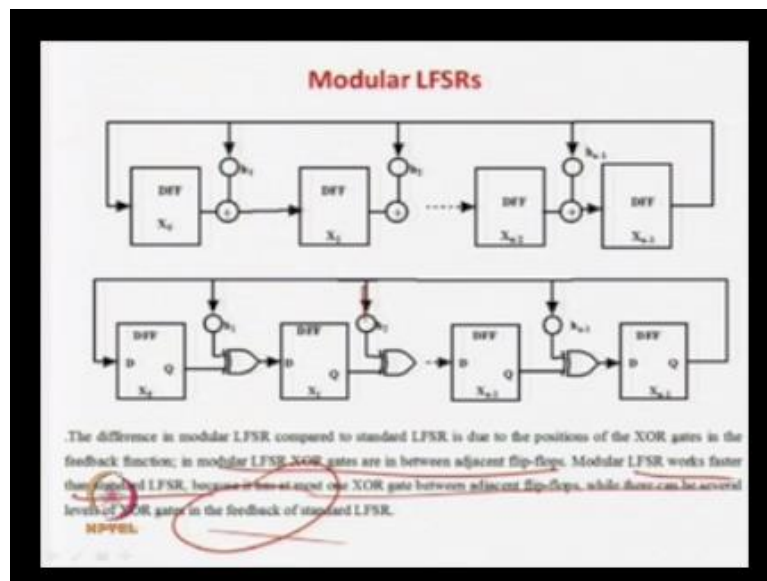
This input will be there and this will be feeding so forth. So the DMA increase in case of LFSRs because.

(Refer Slide Time: 03:45)



If you have a high number of XOR gates as you can see if all these taps are 1 then 1, 2, 3, 4.. N taps or n-1 XOR 2 input XOR gates will be there which is repeating this one. So there will be a lot of delay involved.

(Refer Slide Time: 03:58)



So to avoid this delay so we will study a new type of LFSR today that is actually called LFSR and then also we will see for the response compotator and all those things. So need to avoid those delays peoples have developed another kind of LFSR which is called the modular LFSR it job is almost the same it will also have characteristic polynomial it is also generator near exhausted random patterns and same thing.

The only advantage you will have here is that there will be delay will be less because in this case the architecture you can see the other way round. So in this case there is no sequence of XOR gates, so as in the case I mean on chain of XOR gates are not there, so the delay will be minimize. So what the idea here so you put the XOR gates just like that, I am sorry the deep flip-flop having in the case of standard of LFSR.

But now instead of having chain of I mean, if you remember the standard LFSR we had direct connection between these two and this feedback to this here actually we involved a lot of XOR gates. So that actually cause it delay, but here we are not doing it here the architecture will be different, here there is a direct feedback from here to here and each of the inputs of the XOR gates I mean for the D registers you can either use of feedback from this or you may not use feedback.

Again in this case if  $H_1=0$  then the direct connection this feedback will not be used. And if this  $H_2$  is 1 say then this feedback will be used by an XOR gate from this one and then it will give the input to the next flop. So this is how the basic architecture of module LFSR. So in this case we have seen the difference in modular LFSR compared to standard LFSR is due to the position of the XOR gates.


So here the XOR gates are position in such way so that there is only one XOR gate which is in the delay between this one and this one. But in the previous case there is a lot of XOR gates was here, so there is a lot of delay compared to the output of these last flip of the compared from the first flip-flop kind of start, to avoid this we have now gone to this new architecture in this case we have actually only one what you can call this one XOR gate which is the input of all the flip-flop so the delays are in less.

So there is only one LFSR in modular LFSR between adjacent there of flip-flops there is only one XOR gate, so this LFSRs work faster than standard LFSR because at most one XOR gate is between the adjacent flip-flop, while in the case of the other one you can be lot of gates in the feedback of standard LFSR. So that is why are moving to what you called modular LFSRs.

(Refer Slide Time: 06:14)

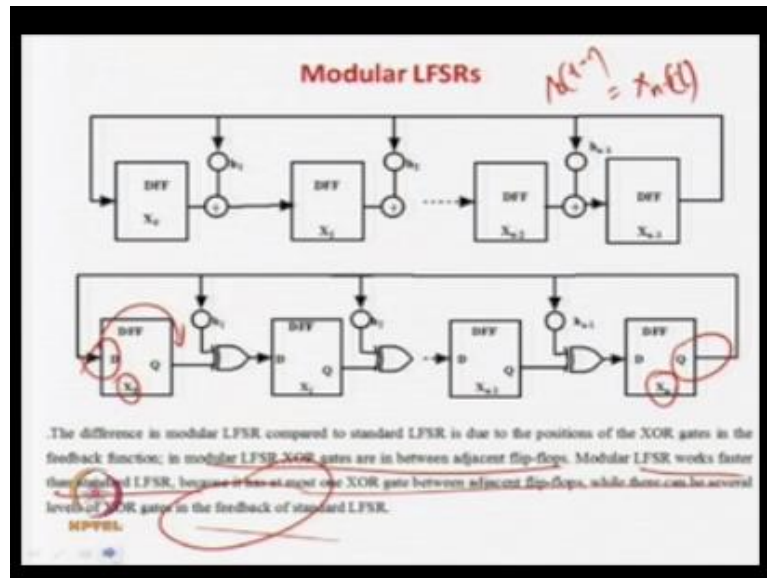
**Modular LFSRs**

In modular LFSR the output of any flip-flop may or may not participate in the XOR function; if output of any flip-flop  $X_i$  say, provides input to the XOR gate which feeds input of flip-flop  $X_{i+1}$  then corresponding tap point  $h_i$  is 1. In the circuit representation  $h_i=1$ , then there is an XOR gate from output of flip-flop  $X_i$  to input of flip-flop  $X_{i+1}$ ; if  $h_i=0$ , then output of flip-flop  $X_i$  is directly fed to input of flip-flop  $X_{i+1}$ .

$$\begin{bmatrix} X_0(t+1) \\ X_1(t+1) \\ X_2(t+1) \\ \dots \\ X_{n-2}(t+1) \\ X_{n-1}(t+1) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & 0 & \dots & 0 & h_1 \\ 0 & 1 & 0 & \dots & 0 & h_2 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 0 & h_{n-2} \\ 0 & 0 & 0 & \dots & 1 & h_{n-1} \end{bmatrix} \begin{bmatrix} X_0(t) \\ X_1(t) \\ X_2(t) \\ \dots \\ X_{n-2}(t) \\ X_{n-1}(t) \end{bmatrix}$$


Now we see just like in the previous case we will have matrix definition for those and we will also have what you can called this characterize polynomial and all.

(Refer Slide Time: 06:24)




So let us see what is the case so in this also if you want a feedback, then so here one thing is there so we have to remember that  $X_0$  input of  $X_0$  is equal to output of  $X_{0+1}$  that means if you say that  $X_0 T + 1$  is nothing but equal to  $X_0$ ,  $X_{n-1T}$  so that has be understood. And whatever is the value here at the time  $T$  will be value at this one at value here that means here the value  $T + 1$ . So that is what we have establish.

(Refer Slide Time: 06:49)

**Modular LFSRs**

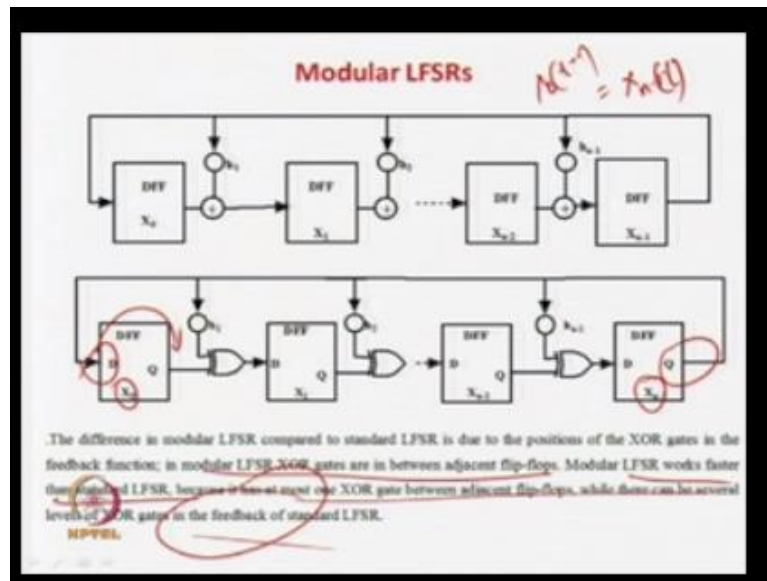
In modular LFSR the output of any flip-flop may or may not participate in the  $X$  function; if output of any flip-flop  $X_i$  say, provides input to the XOR gate which feeds input of flip-flop  $X_{i+1}$  then corresponding tap point  $h_i$  is 1. In the circuit representation  $h_i = 1$ , then there is an XOR gate from output of flip-flop  $X_i$  to input of flip-flop  $X_{i+1}$ ; output of flip-flop  $X_i$  is directly fed to input of flip-flop  $X_{i+1}$ .

$$\begin{bmatrix} X_0(t+1) \\ X_1(t+1) \\ X_2(t+1) \\ \dots \\ X_{n-2}(t+1) \\ X_{n-1}(t+1) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & 0 & \dots & 0 & h_1 \\ 0 & 1 & 0 & \dots & 0 & h_2 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 0 & h_{n-2} \\ 0 & 0 & 0 & \dots & 1 & h_{n-1} \end{bmatrix} \begin{bmatrix} X_0(t) \\ X_1(t) \\ X_2(t) \\ \dots \\ X_{n-2}(t) \\ X_{n-1}(t) \end{bmatrix}$$


So in this case also you can see here that if you consider this one is your diagonal matrix. So this is your  $X_{i+1}(t+1)$  this is your  $X_i(t)$  okay. So first one we already know that what is the value of  $X_0(t+1)$  that is  $X_{n-1}(t)$  will be nothing but  $X_{n-1}(t)$  will be direct connection.



(Refer Slide Time: 07:10)




Because there is direct connection from here to here okay.

(Refer Slide Time: 07:13)

**Modular LFSRs**

In modular LFSR the output of any flip-flop may or may not participate in the XOR function; if output of any flip-flop  $X_i$  say, provides input to the XOR gate which feeds input of flip-flop  $X_{i+1}$  then corresponding tap point  $h_i$  is 1. In the circuit representation  $h_i = 1$ , then there is an XOR gate from output of flip-flop  $X_i$  to input of flip-flop  $X_{i+1}$ ; if output of flip-flop  $X_i$  is directly fed to input of flip-flop  $X_{i+1}$ .


$$\begin{bmatrix} X_0(t+1) \\ X_1(t+1) \\ X_2(t+1) \\ \dots \\ X_{n-2}(t+1) \\ X_{n-1}(t+1) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & 0 & \dots & 0 & h_1 \\ 0 & 1 & 0 & \dots & 0 & h_2 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 0 & h_{n-2} \\ 0 & 0 & 0 & \dots & 1 & h_{n-1} \end{bmatrix} \begin{bmatrix} X_0(t) \\ X_1(t) \\ X_2(t) \\ \dots \\ X_{n-2}(t) \\ X_{n-1}(t) \end{bmatrix}$$


So that is what is the case  $X_0 T + 1$  this one is equal to all this will be multiplied by 00000 only this one will be there because this one will be productive this one. So we are going to get an  $X_0 T+1$  is equal nothing but  $X_{n-1} T$  that is why this last column will have one here okay.

(Refer Slide Time: 07:34)

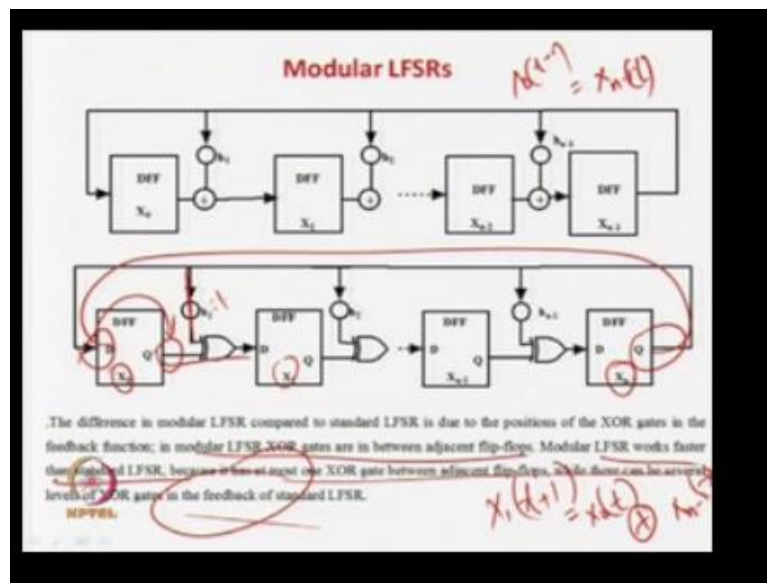
**Modular LFSRs**

In modular LFSR the output of any flip-flop may or may not participate in the XOR function; if output of any flip-flop  $X_i$  say, provides input to the XOR gate which feeds input of flip-flop  $X_{i+1}$  then corresponding tap point  $h_i$  is 1. In the circuit representation  $h_i = 1$ , then there is an XOR gate from output of flip-flop  $X_i$  to input of flip-flop  $X_{i+1}$ ; if  $h_i = 0$ , then output of flip-flop  $X_i$  is directly fed to input of flip-flop  $X_{i+1}$ .

$$\begin{bmatrix} X_0(t+1) \\ X_1(t+1) \\ X_2(t+1) \\ \dots \\ X_{n-2}(t+1) \\ X_{n-1}(t+1) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & 0 & \dots & 0 & h_1 \\ 0 & 1 & 0 & \dots & 0 & h_2 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 0 & h_{n-2} \\ 0 & 0 & 0 & \dots & 1 & h_{n-1} \end{bmatrix} \begin{bmatrix} X_0(t) \\ X_1(t) \\ X_2(t) \\ \dots \\ X_{n-2}(t) \\ X_{n-1}(t) \end{bmatrix}$$


Now next will next we can see is how you can develop the expression for  $X_{i+1}(t+1)$ .

(Refer Slide Time: 07:39)



So if you see  $X_{n-1}(t+1)$  this one is what  $X_{n-1}(t)$  will depend on what it will depend on this one okay, so it will obviously be depending on this one so it will be what it will be  $X_0(t)$  okay XOR there can be XOR if  $h_1$  is 0 then there is directly dependent on there when there is direct connect then it will be nothing but  $X_{n-1}(t+1)$  is  $X_0(t)$ , but in this case  $h_1 = 1$  then this feedback will also come into picture then we have put XOR what is that  $X_{n-1} \times T_0$  so the previous this is not  $T_0$  this is  $T$ . So then the feedback will also come into picture.


(Refer Slide Time: 08:22)

**Modular LFSRs**

In modular LFSR the output of any flip-flop may or may not participate in the XOR function; if output of any flip-flop  $X_i$  say, provides input to the XOR gate which feeds input of flip-flop  $X_{i+1}$  then corresponding tap point  $h_i$  is 1. In the circuit representation  $h_i = 1$ , then there is an XOR gate from output of flip-flop  $X_i$  to input of flip-flop  $X_{i+1}$ ; output of flip-flop  $X_i$  is directly fed to input of flip-flop  $X_{i+1}$ .

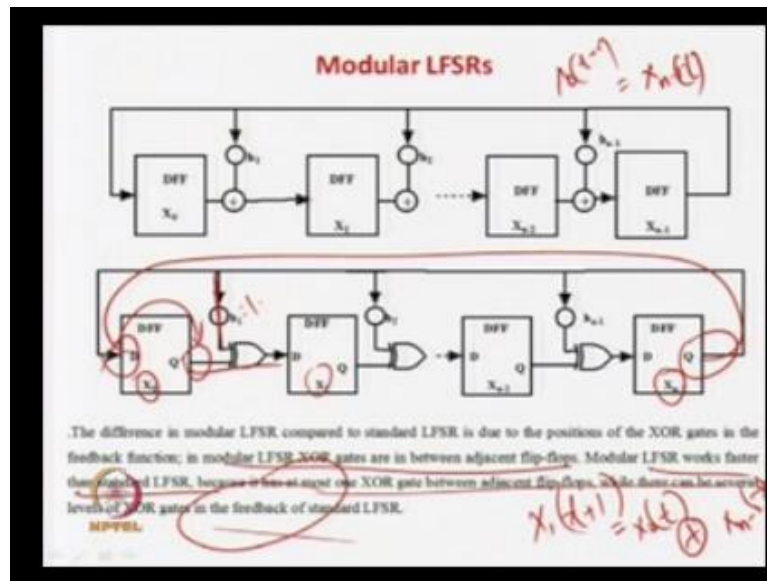
$$\begin{bmatrix} X_0(t+1) \\ X_1(t+1) \\ X_2(t+1) \\ \dots \\ X_{n-2}(t+1) \\ X_{n-1}(t+1) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & 0 & \dots & 0 & h_1 \\ 0 & 1 & 0 & \dots & 0 & h_2 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 0 & h_{n-2} \\ 0 & 0 & 0 & \dots & 1 & h_{n-1} \end{bmatrix} \begin{bmatrix} X_0(t) \\ X_1(t) \\ X_2(t) \\ \dots \\ X_{n-2}(t) \\ X_{n-1}(t) \end{bmatrix}$$

*Handwritten notes:  $X_0(t)$ ,  $h_1$ ,  $h_2$*



So that is the thing you will get so  $X_0 X_1 T+1$  is what it is nothing but this guy will be there so it will be this one product will be 1 so it will be nothing but  $X_0 T$  XOR okay. So because of this one will be gone because of this, this will be product in to 0 everything will off because of all the 0 excepting this  $H_1$  will there it will be  $H_1 \cdot X_{n-1}$  this guy into this one okay.

(Refer Slide Time: 08:51)



So what it says that so  $x_{n-1}$  this feedback will be considered if  $H_1=1$  it will get direct feedback from this one.

(Refer Slide Time: 08:57)

**Modular LFSRs**

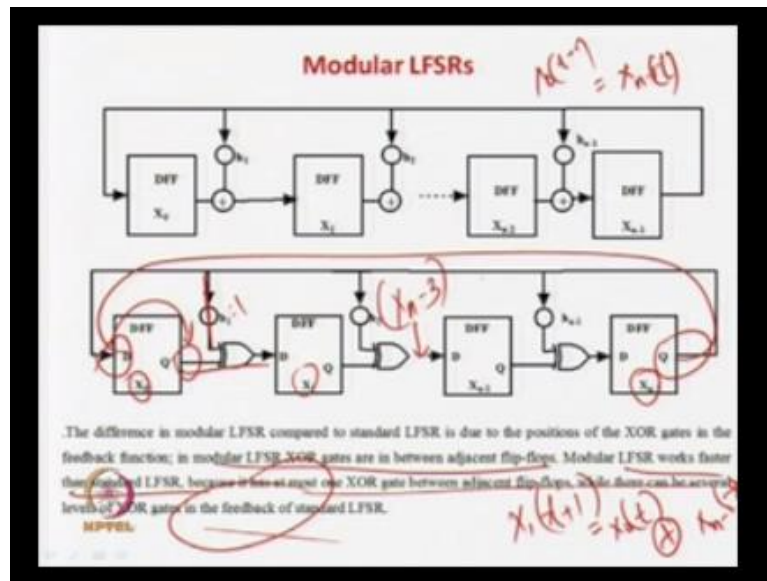
In modular LFSR the output of any flip-flop may or may not participate in the XOR function; if output of any flip-flop  $X_i$  say, provides input to the XOR gate which feeds input of flip-flop  $X_{i+1}$  then corresponding tap point  $h_i$  is 1. In the circuit representation  $h_i = 1$ , then there is an XOR gate from output of flip-flop  $X_i$  to input of flip-flop  $X_{i+1}$ ; if output of flip-flop  $X_i$  is directly fed to input of flip-flop  $X_{i+1}$ .

$$\begin{bmatrix} X_0(t+1) \\ X_1(t+1) \\ X_2(t+1) \\ \dots \\ X_{n-2}(t+1) \\ X_{n-1}(t+1) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 & h_2 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 0 & h_{n-2} \\ 0 & 0 & 0 & \dots & 1 & h_{n-1} \end{bmatrix} \begin{bmatrix} X_0(t) \\ X_1(t) \\ X_2(t) \\ \dots \\ X_{n-2}(t) \\ X_{n-1}(t) \end{bmatrix}$$

*Handwritten notes:  $X_0(t)$  circled,  $h_1$  circled,  $X_{n-1}(t)$  crossed out,  $X_{n-2}(t)$  crossed out.*

So if  $h_1 = 0$  if you consider that this will also be gone this will be gone so  $X_0(t+1)$  will be nothing but only this first product  $X_0(t)$  + sorry  $X_1(t+1)$  will be nothing but  $X_0(t)$  will be there okay. So this is how you can develop for everything so just like let us also look at another flip-flop say in between.

(Refer Slide Time: 09:20)




So in between you can consider this is  $X_{0n-3}$  you can consider this flip-flop over here. So it will depend on  $X_0$   $X_{n-4}$  and also the feedback. So just you can look at it so if you can consider  $X_0$  we consider sorry this one we can consider. So it will depend on  $X_{n-3}$  as well as the feedback is required okay.



(Refer Slide Time: 09:41)

**Modular LFSRs**

In modular LFSR the output of any flip-flop may or may not participate in the XOR function; if output of any flip-flop  $X_i$ , say, provides input to the XOR gate which feeds input of flip-flop  $X_{i+1}$ , then corresponding tap point  $h_i$  is 1. In the circuit representation  $h_i = 1$ , then there is an XOR gate from output of flip-flop  $X_i$  to input of flip-flop  $X_{i+1}$ ; output of flip-flop  $X_i$  is directly fed to input of flip-flop  $X_{i+1}$ .

$$\begin{bmatrix} X_0(t+1) \\ X_1(t+1) \\ X_2(t+1) \\ \dots \\ X_{n-2}(t+1) \\ X_{n-1}(t+1) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & 0 & \dots & 0 & h_1 \\ 0 & 1 & 0 & \dots & 0 & h_2 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 0 & h_{n-2} \\ 0 & 0 & 0 & \dots & 1 & h_{n-1} \end{bmatrix} \begin{bmatrix} X_0(t) \\ X_1(t) \\ X_2(t) \\ \dots \\ X_{n-2}(t) \\ X_{n-1}(t) \end{bmatrix}$$


So it will be dependent on so here so this is the thing, so for this we will have 1 over here, so it will be depending on  $X_{n-3}$  will be there so this will be if you consider this row 001, 10H2 will be there use the diagonal element, so if will be you will have something  $X_{n-3}(T)$  will be there. So  $X_{n-2}(T+1)$  will be nothing but equal to this will be there and this will be productivity this guy this all others will be zero.

So it will be equal to  $X_{n-3}(T)$  then you have to use XOR then this will go off because of this product with the zero it will go off and  $H_{n-2}$  will be there in  $X_0 H_{n-2}$  product of this one. Okay. So this will be  $X_{n-1}(T)$ , so depending on whether this is one or zero so this factor that this means these two factor that is  $H_{n-2}$  product of  $X_{n-1} T$  will be there or will not be there will depend on the this value.


Otherwise if there then you will have to XOR this one with this factor and if  $H_{n-2}$  is 0 then this will not be there and  $X_{n-2} T + 1$  will be  $X_{n-3}$  into XOR into T. So that is what is being obtained that is each of this flip-flop is dependent on the other flip-flop XOR with the feedback of  $X_{n-1}$  if the corresponding H is one else LTs are 0. So there is very simple this thing very simple stuff so this can be represented by a matrix format as we have already seen. So this the matrix format here is you have diagonal matrix over here like this row is all 0, here it is 1 and here is corresponding H.

(Refer Slide Time: 11:31)

**Standard LFSRs**

This LFSR in terms of the matrix can be written as  $X(t+1) = T_x X(t)$ .

$$\begin{bmatrix} X_0(t+1) \\ X_1(t+1) \\ \dots \\ X_{n-3}(t+1) \\ X_{n-2}(t+1) \\ X_{n-1}(t+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & 0 & \dots & 0 & 1 \\ 1 & h_1 & h_2 & \dots & h_{n-2} & h_n \end{bmatrix} \begin{bmatrix} X_0(t) \\ X_1(t) \\ \dots \\ X_{n-3}(t) \\ X_{n-2}(t) \\ X_{n-1}(t) \end{bmatrix}$$




So similarly this if you at this our standard LFSR if you look so it is also having a similar stuff, but everything is being protected. So this here this is your standard diagonal matrix okay and this is all 0 sorry just a minute, so here this is for the standard LFSR already we discussed in the last class. So this is your diagonal matrix so this is the column with all 0 and one bit will be one and here all HS.

(Refer Slide Time: 11:55)

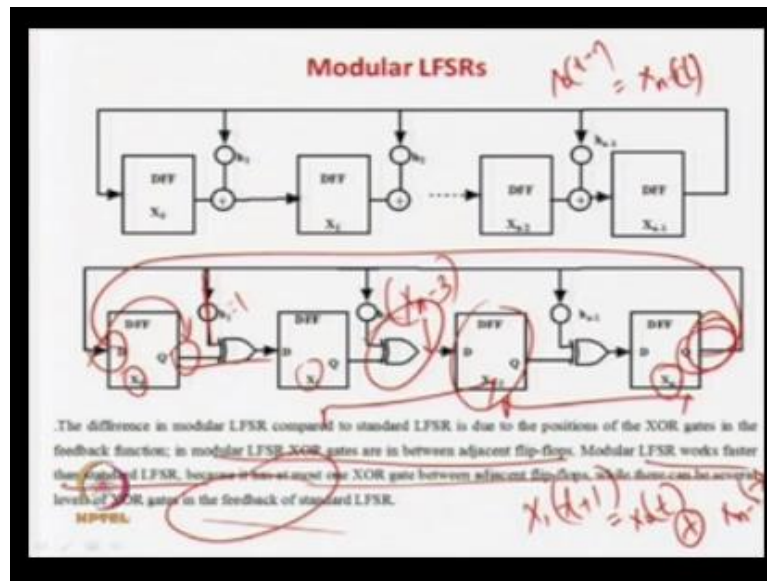
**Modular LFSRs**

In modular LFSR the output of any flip-flop may or may not participate in the  $X$  function; if output of any flip-flop  $X_i$ , say, provides input to the XOR gate which feeds input of flip-flop  $X_{i+1}$ , then corresponding tap point  $h_i$  is 1. In the circuit representation  $h_i = 1$ , then there is an XOR gate from output of flip-flop  $X_i$  to input of flip-flop  $X_{i+1}$ ; output of flip-flop  $X_i$  is directly fed to input of flip-flop  $X_{i+1}$ .

$$\begin{bmatrix} X_0(t+1) \\ X_1(t+1) \\ X_2(t+1) \\ \dots \\ X_{n-2}(t+1) \\ X_{n-1}(t+1) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & 0 & \dots & 0 & h_1 \\ 0 & 1 & 0 & \dots & 0 & h_2 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 0 & h_{n-2} \\ 0 & 0 & 0 & \dots & 1 & h_{n-1} \end{bmatrix} \begin{bmatrix} X_0(t) \\ X_1(t) \\ X_2(t) \\ \dots \\ X_{n-2}(t) \\ X_{n-1}(t) \end{bmatrix}$$


So this is just and if you consider for modular LFSR there is nothing but it is bit around about it, something has been rotator a kind of thing, but just doing them, but they are actually equivalent in nature.

(Refer Slide Time: 12:03)




But by using this type of rotated kind of an architecture then delay between this flop to this flop, this flop because very low because you have only one XOR gate in between.

(Refer Slide Time: 12:14)

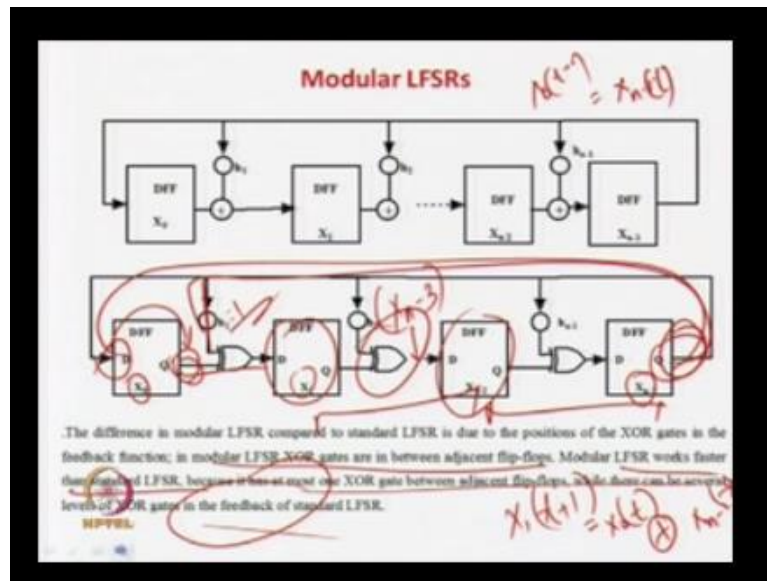
**Modular LFSRs**

In modular LFSR the output of any flip-flop may or may not participate in the XOR function; if output of any flip-flop  $X_i$  say, provides input to the XOR gate which feeds input of flip-flop  $X_{i+1}$  then corresponding tap point  $h_i$  is 1. In the circuit representation  $h_i = 1$ , then there is an XOR gate from output of flip-flop  $X_i$  to input of flip-flop  $X_{i+1}$ ; output of flip-flop  $X_i$  is directly fed to input of flip-flop  $X_{i+1}$ .

$$\begin{bmatrix} X_0(t+1) \\ X_1(t+1) \\ X_2(t+1) \\ \dots \\ X_{n-2}(t+1) \\ X_{n-1}(t+1) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & 0 & \dots & 0 & h_1 \\ 0 & 1 & 0 & \dots & 0 & h_2 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 0 & h_{n-2} \\ 0 & 0 & 0 & \dots & 1 & h_{n-1} \end{bmatrix} \begin{bmatrix} X_0(t) \\ X_1(t) \\ X_2(t) \\ \dots \\ X_{n-2}(t) \\ X_{n-1}(t) \end{bmatrix}$$


So that is why we have move to move something called modular LFSR because of the delay constraints which we are avoiding in case of modular LFSR compare to standard LFSR. So whatever I told you there are in text so in modular LFSR the output of any flip-flop near may or may not participate in the XOR function, so if the output of any flip-flop is provide input to the XOR gate with the feedback this one then the corresponding H is one as it zero.

(Refer Slide Time: 12:45)




So this is whatever we have discussed that the output of one flip-flop will control the output of one flip-flop will control the output of another flip-flop, but if you also want to depend on the feedback of the  $X_{n-1}$  flip-flop.

(Refer Slide Time: 12:52)

**Modular LFSRs**

In modular LFSR the output of any flip-flop may or may not participate in the XOR function; if output of any flip-flop  $X_i$  say, provides input to the XOR gate which feeds input of flip-flop  $X_{i+1}$  then corresponding tap point  $h_i$  is 1. In the circuit representation  $h_i=1$ , then there is an XOR gate from output of flip-flop  $X_i$  to input of flip-flop  $X_{i+1}$ . output of flip-flop  $X_i$  is directly fed to input of flip-flop  $X_{i+1}$ .

$$\begin{bmatrix} X_0(t+1) \\ X_1(t+1) \\ X_2(t+1) \\ \dots \\ X_{n-2}(t+1) \\ X_{n-1}(t+1) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & 0 & \dots & 0 & h_1 \\ 0 & 1 & 0 & \dots & 0 & h_2 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 0 & h_{n-2} \\ 0 & 0 & 0 & \dots & 1 & h_{n-1} \end{bmatrix} \begin{bmatrix} X_0(t) \\ X_1(t) \\ X_2(t) \\ \dots \\ X_{n-2}(t) \\ X_{n-1}(t) \end{bmatrix}$$



Then the corresponding it should be one, so that is what actually is written over here.

(Refer Slide Time: 12:58)

**Modular LFSRs**

This LFSR given the matrix can be written as  $X(t+1) = T X(t)$ . In this case  $X_2(t+1) = X_{n-1}(t)$ , which implies that  $X_{n-1}$  directly feedbacks  $X_2$ .  $X_1(t+1) = X_2(t) + h_3 X_{n-1}(t)$ , which implies that depending on  $h_3 = 0$  (or 1), input to  $X_1$  is  $X_2$  (or  $X_2$  XORed with output of  $X_{n-1}$ ). Similar logic holds for inputs to all flip-flops from  $X_1$  to  $X_{n-1}$ .

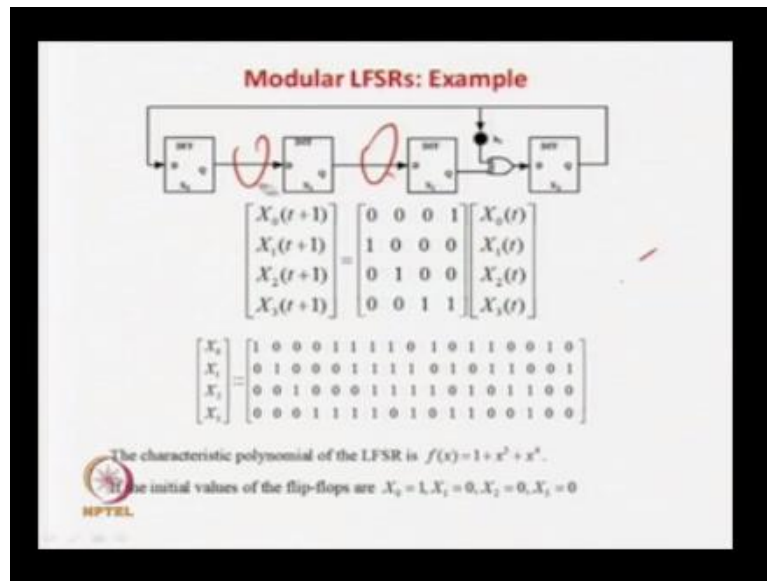
This LFSR can also be described by the characteristic polynomial:

$$f(x) = 1 + h_1 x + h_2 x^2 + \dots + h_{n-2} x^{n-2} + h_{n-1} x^{n-1} + x^n$$


So I mean so again this equation will be  $X$  this is the equation so  $X(t+1) = T X(t)$  this matrix this  $T(S)$  and it is the next value of flip-flop and this is the present value of the flip-flop okay. So this is what the case so already we have discussed about how do we get this equations and then again this is your characteristic polynomial, so effects will have one here and  $X^n$  okay this depending on the last two terms and in between you will have this values depending the values of  $H_1$  if  $H_2$  if you consider them to be 0 or if not consider them to be 1. So based on that you can get the characteristic polynomial is same in the case of standard LFSR.



(Refer Slide Time: 13:41)



So only difference is that so again lot of theory is exist as we have discussing in the last class the lot of theory is exist to find out whether this character polynomial will generator a near complete set of random patterns whether and actually and whether it will on also the case whether it may not be able to generate all those things.

So there are lot of theories that exist that given a characteristic polynomial like this they gave me lot of analysis which can predict or which can say that or which determine that if any modular LFSR has this characteristic polynomial then if you generator near exhaustive random set of patterns stood in the all zeros and sometimes there also exist some characteristic polynomial which will not generate all possible exhaustive set of patterns.

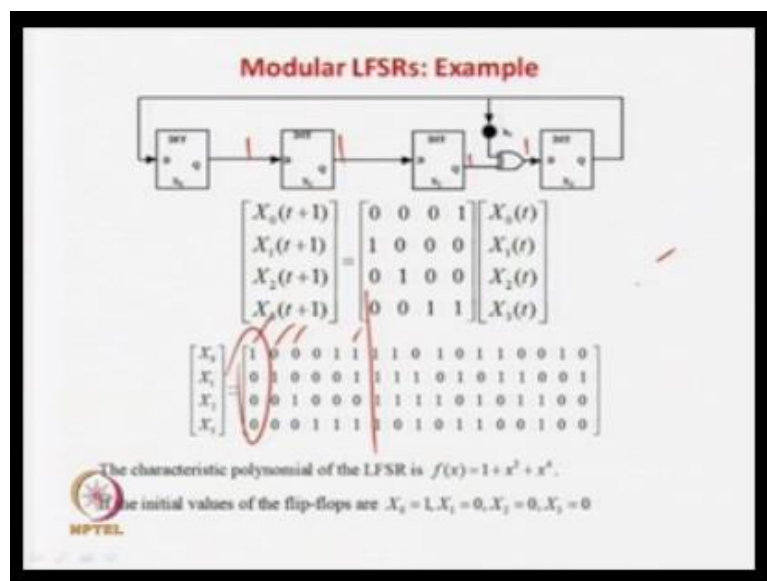
So when depending on your requirement you can select your characteristic polynomial and how is the characteristic polynomial selected based on this whether you require a feedback from this XOR gate or whether you require of like this so it always possibility. So in this example the person has decided not to use this so he has required only for the X3 flop you require a feedback from this and for other case there is required.

So here the characterized polynomial will be 1, 2, 3 4 so polynomial will be  $1/x^4$  plus only  $S_3$  is zero so it will be  $H3X^3$  plus this one, so this is one you can eliminate this so it will be  $1+x^3+x^4$  so this X3 and X4 are already there and  $X^3=X$  the factor X3 comes here that is X3

is 1 and there is no product like  $X_1 X^2$  because in this case this guys are 0 this Xs are 0 so they are removed this characteristic square.

Now there exist the literature so you can find out whether this characteristic polynomial will generator rare exhaustive set of pattern. So in this case if you look at it we will find out that this characteristic polynomial fall in the class where you can generator the near exhaustive set of patterns.

(Refer Slide Time: 15:32)



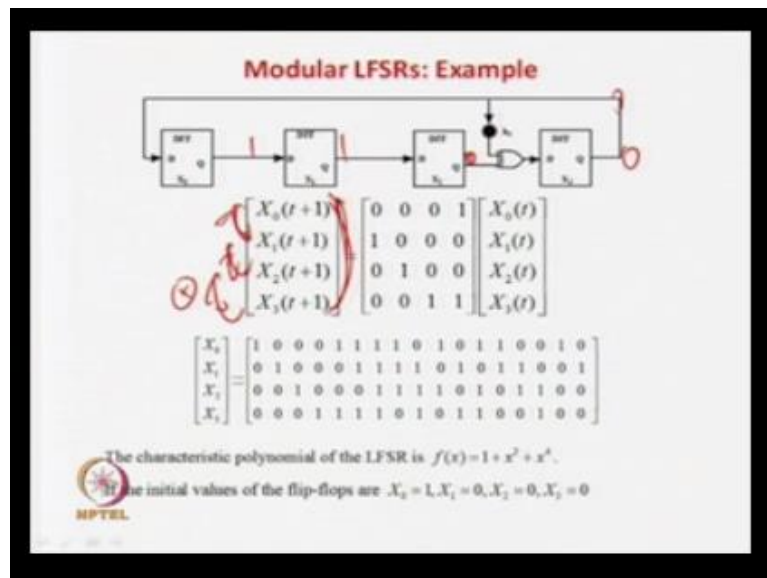
Now what we have found out that another thing which also depends as in this case of standard LFSR there is one more thing that is important in actually called the seed vector. Seed mean what is the initial reset values, so depending on that the next series of, I mean if you can see series like this also there is no one can prevent you from that so you can also have this at the seed vector then you will start from this one.

And if you have any other seed vector then you can start from that the vector and is very important to find out that how do you select the seed vectors, then actually for example if you say that these are the four vectors which are actually used from my testability case so better I start from this one then very quickly I will from here up to here I will get all the back for testing and then my job will does then I reset back here.

Then obviously if this has the seed vector, but if our important test patterns have this one, this one and this one then obviously you will start from this vector and you will go forward from here and we will start from here. So what is the ideal seed vector we will tell you way to start from and then what is the sequence of patterns. So obviously we will try to give such seed vectors so that you can get the required test patterns for testing which you want to keep for testing in the early sequence right.

So that was the idea okay, so what we are saying that so that is already that same case existed in case of for this standard LFSR where we have to see say the seed vectors and use it in such a way, so that the very quickly we can get the required test patterns. So in this case let us assume that if the person assume that maybe this may be important as vectors so you have give the seed vector.

(Refer Slide Time: 17:03)



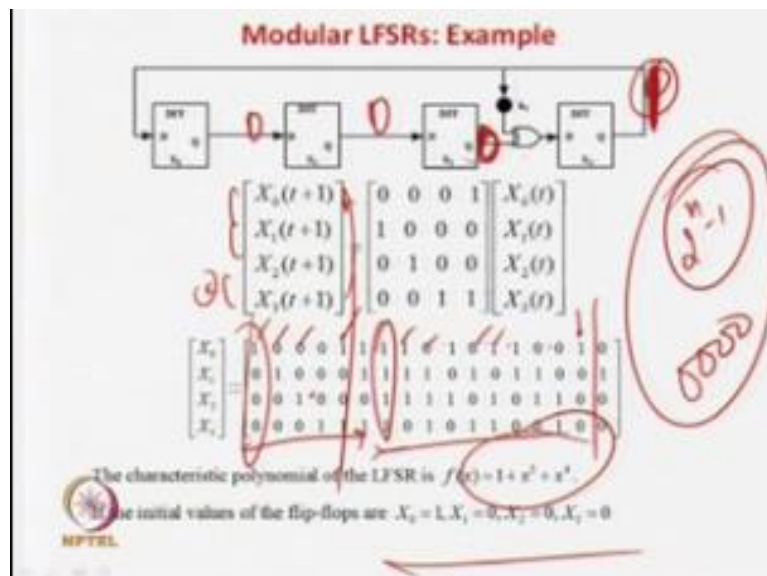
Now let us see how it happens how we get this patterns so same example as in the previous case also. So he has used X0 as one and other as 0 so this is seed vector. Now what next, so this one how do you write about it so we already know that this is our diagonal vector, so diagonal matrix so it is there, and then this one is all 0 so first one you have keep it as one that is already there and in this case H3 is 1 H2 and H1 as 0 so you will get 00 and here you put a 1, because H3 is 1, this is how you develop the matrix correct.

So now as we know that here X0 will determine X1, X1 will determine X2, X3 will determine sorry X2 will determine X3 right, X0 will remain X1, X1 will remain X2, X2 determines X3 and X3 sorry, sorry. So in this case X1 will determine X0 will remain X1 if here is a direct control because there is no feedback is required, but if the feedback had been taken into picture.

So X1 will control X1 with the feedback so in this case there is no feedback for this line. So we can directly say that X0 is controlling X1, similarly there is no feedback over here so we can say that X1 is controlling X2 and in this case there is feedback, so X2 is controlling X3 with the feedback. And then directly again you can say that this X3 will be directly controlling X1.

So in this case there is no feedback, so I mean no feedback involve so there is direct control. And here actually from here to here, from here to here, and from here to here this control or control involved in XOR from this one but in this case there is no feedback, so there is a control and in this case there is a feedback so there is dependent control from X2 to X3.

(Refer Slide Time: 18:49)



So now if you can see what happens so in this case this is very easy so direct seed vector so it is one there is 0 and 10 sorry 1 then 0, then 0, and then 0 this is the case. So after the next iteration what will be the case we know that so this was our control here is an XOR control

and there is a this control is there. So obviously so this value will come here, this value will come here, the third value will come here obviously we can solve this one.

So this value is coming here, this value is coming here and this value will come here obviously with an XOR condition. So in this case what happens so to determine the value of X3 we have to XOR 0 with this XOR. So 0 XOR 0 is already a 0 so we get a 0 over here, and this should also extremely directly control X0 so this value will go over here. So very easily we can get the vector of 0100 so this is the case.

This is what direct control here in this case no dependency and in this case there is X2 is controlling X3, but with the XOR with this one. So in this is case is 0 XOR 0 so we get a 0 and this is the case, and this is one will be directly feedback over here so we get 01000, so what is the next vector here, so the next vector here will be nothing but it is 0 over here then it is a 1 over here it is going to 0 over here and it is going to 0 over here so this is how you get it. So, now you can again find out the next vector how you can determine, so in this case we know that this one is the direct control, this one is the direct control, so you get this directly.

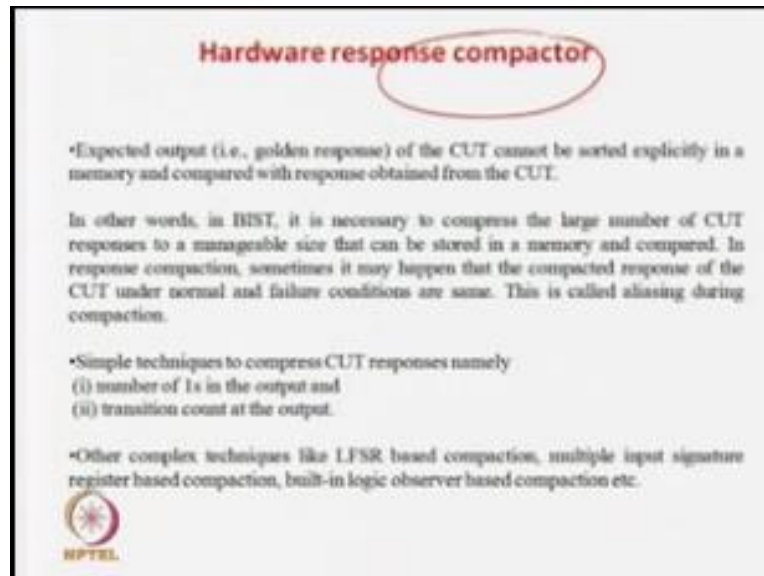
But, this third value will depend on the XOR value of this one and this was again 0 XOR 0 is 0 so you get this value over here and this is the feedback from here, so you get the vector 0010, so it is a 00 and now it is a 1 over here and it is a 0 over here. So, now you can see the fourth vector, so fourth vector as we know again this is used to a direct controls over here now this controls from here to here will depend on XOR of this value with XOR of this value 0 XOR 1 is a 1, so you get a 1 over here and this one is actually of your feedback over here.

So now you get the value of 000 and this is going to be a 0 now and this one will be a 1 now. So, this is what your next vector is over here. Similarly there is a same way you can calculate them this one. So, as a same way you can find out this whole sequence of patterns so we will find out that we started from 1000. So, after so many patterns you will find out that there will be reputation if you count 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15. 15 patterns, so  $2^{n-1}$  if you enclosing the all four 0 is a exhaustive test pattern have been generated by using this electricity polynomial this was this sheet.

So, in this case if you have application when you have to generate four bit vectors for your testing and assume that this 3 are your more important test patterns and then this one, so what

you can is a you can take these are the sheet value and then you will get patterns up to this and then you can reset any vector value back. And again if you are assuming that these are your important test patterns then you can have these are the sheet value and then you can go about this. So this is a very similar case.

(Refer Slide Time: 21:58)




**Hardware response compactor**

- Expected output (i.e., golden response) of the CUT cannot be stored explicitly in a memory and compared with response obtained from the CUT.

In other words, in BIST, it is necessary to compress the large number of CUT responses to a manageable size that can be stored in a memory and compared. In response compaction, sometimes it may happen that the compacted response of the CUT under normal and failure conditions are same. This is called aliasing during compaction.

- Simple techniques to compress CUT responses namely
  - (i) number of 1s in the output and
  - (ii) transition count at the output.
- Other complex techniques like LFSR based compaction, multiple input signature register based compaction, built-in logic observer based compaction etc.

 NPTEL

So as a very similar case like modular LFSR and standard LFSR whatever you take the basic idea is that you can generate a near exhaustive state random patterns and area over this very, very less, so in this case as you can also say there is only one XOR gate is involved. But only in case of standard LFSR in the feedback first flip flop to the last flip flop kind of a stuff. The whole feedback mechanism in the first feedback to the last feedback or someone you can say that flip flop to the first flip flop.

So whatever the longest feedback in standard LFSR in sequence of XOR gate. So, that was reading to some kind of a delay. In modular LFSR it is also the same case we can have a characteristic polynomial and we can see that the characteristic polynomial in a very lies way by using a simple sequence of XOR gates and you can similarly generate a near exhaustive set of test patterns, but here the delay will be very, very less, because in between appear of flip flops there can be at max only one XOR gate. So, modular LFSR has the all the features that a standard LFSR has, but only with the added advantage that there is more extra delay or more extra huge delay this is of the extent of XOR gates.

So, again I was discussing many times can whether any LFSR will generate all the  $2^{n-1}$  actually answer is no there is only a few characteristic polynomial for which is the maximum length there actually called primitive polynomial, so at least they are the people are found out a list that like this person in this paper. Here found out a at least of such polynomial so that you can have a refers to that then if you want to design a circuit which requires an exhaustive set of patterns then you can go about safety characteristic polynomial that is call the primitive polynomial and you will be able to generate what you can call a near exhaustive set of patterns.

Or sometimes you may not be require a huge set of  $2^{n-1}$  patterns if you have any limited set of patterns will be enough, So in that case you can go for selecting another different kind of characteristic polynomial which may not be a primitive polynomial so you did not generate the exhaustive set of patterns you can generate a very particular set of patterns then a particular sequence.

But, depending on the characteristic polynomial and whatever you call the sheet vector you can get your desire random test pattern generator accordingly at a very, very low area fault. So, that is the actually the beauty of what you call this linear feedback sheet registers in case best. So, major improvement just you recall once more. So major requirement in case of best we require a pattern generator however taken the pattern generator cannot be a very huge area of pattern generator because in case of ATE the area over rate of the pattern generator in case of a automatic test equipment is not of our concern.

Because a huge equipment then you can do that, but whenever you are going for institute testing, so all the chips will have on chip in a test pattern generator so this area is very, very restricted so how you can do it, so option is that you can generate a deterministic test pattern generator as you have done using a finessed machine based implementation as usual digital design fundamentals, but here the area overall very, very high, because we have to optimize your number of gates because the standard procedure you know you have to generate a final state machine with all the states and the sequence.

And then you have to find out optimize the number of Kats to flip flops and this is the standard procedure is follow. But you can generally find that the number of gates will be

much, much larger than a single XOR gate or sequence of single 2 input XOR gate as is the number gates will be must rather than if you are using a deterministic federation machine to generate it.

But, the linear feedback shift registers or the modular LFSR or the standard LFSR they are actually very much great wound in this case because using a single sequence of 2 input XOR gates or a few number XOR gates you can generate the desired characteristic polynomial and a sheet that will give you the sequence of patterns will require for doing the testing and the area already extremely smaller compact to what you call a deterministic final state machine plus random generator.

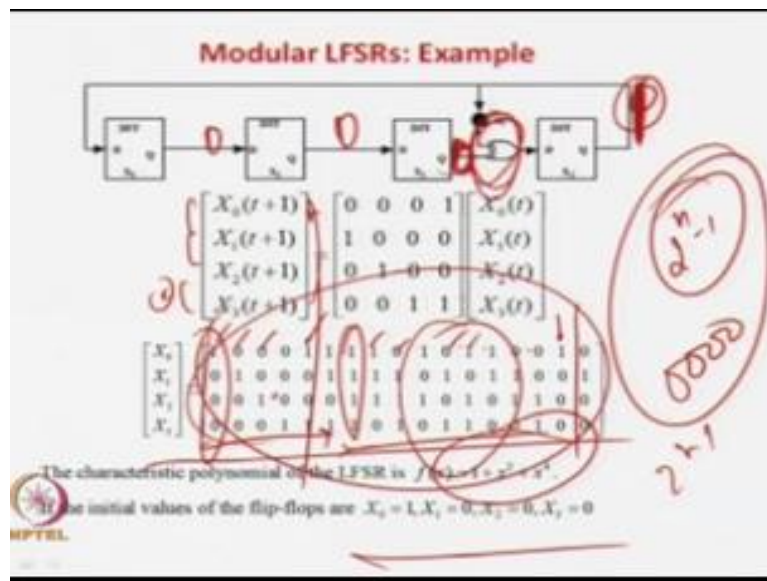
So, that is why we all go for LFSR with a design the LFSR with the desire characteristic polynomial and if you have a exhaustive set then you have to go for a primitive polynomial set least can be found in this paper and if you do not require that you can design another kind of a characteristic polynomial, so that you may not have the exhaustive set of patterns, but you will have the required set of patterns, again if you want to select the sequence of patterns is such a way so that your require test patterns are arrive before so you can select the sheet accordingly and your job is done. So, that is how we all design what you call the pattern generator in case of best using LFSR.

Now the second part of your BIST design as I told you. BIST has three component main component there is a test controller that controls the this one so this is the standard digital fundamentals how can you control your circuit there is the control for the design then was your standard test pattern generator so which is already saw using the LFSR that is the modular LFSR and standard LFSR now they remain in the another part is actually call the response compactor or response comparator that is the circuit will design generate some responses based on the input patterns.

So this input patterns already we have know that given this LFSR but are the pattern generated by the circuit and then also you can easily compute that what will be normal response of the circuit corresponding to this pattern set.



(Refer Slide Time: 27:07)



Because given a characteristic polynomial and their design because this design is premade so you already in this circuit so you know that this is the sheet so we know that this is the sheet and this is the sequence pattern that will be generated and also we know that if the circuit is operating normally then what will be the output response corresponding to this pattern.

That is can be already calculated and that can be stored in a round and it can be compared, but now you see because the round is also very expensive component when you are thinking on chip design then you cannot go for a flat design like use that the patterns from the test pattern generator that is a LFSR then for all this patterns will be explicitly store the full output response of the circuit in a round and then you go for a compress. This is call a flat design.

This is the pattern this is the output is store is a round you apply the input of the circuit and compare the responses using a flat design. But, now at the round is very, very expensive so here also what we have to do we have to go for some amount of completion or compression so that your ROM area is very, very small and you say one something. But, obviously when you have compressing of this the last compression so you can find out that there will be some lose it fault coverage, but still the major requirement of our case was here in this case was then that you have to go for a low area ROM inside so that can be achieve by compressions so that would be now studying in this lecture.

(Refer Slide Time: 28:26)

**Hardware response compactor**

- Expected output (i.e., golden response) of the CUT cannot be stored explicitly in a ROM and compared with response obtained from the CUT.
- In BIST, it is necessary to compress the large number of CUT responses to a manageable size that can be stored in a memory and compared. In compression, sometimes it may happen that the compacted response of the normal and failure conditions are same. This is called aliasing during compression.
- Simple techniques to compress CUT responses namely
  - (i) number of 1s in the output and
  - (ii) transition count at the output.
- Other complex techniques like LFSR based compaction, multiple input signature register based compaction, built-in logic observer based compaction etc.

NPTEL

So, that is what I told you the expected output response that cannot be stored explicitly because of the size in the ROM so in other words what you say in be is it is necessary to compress the large number of card response in the circuit under test response to manageable size.

So, it can be stored in the memory and compressive also your target should be there but there should not be any aliasing so what you mean by aliasing kind of a thing so that is do not compress so the fault can be detected and if you compress it the fault cannot be detected so we will see with example so it happens that due to compression the fault will not been able to detected.

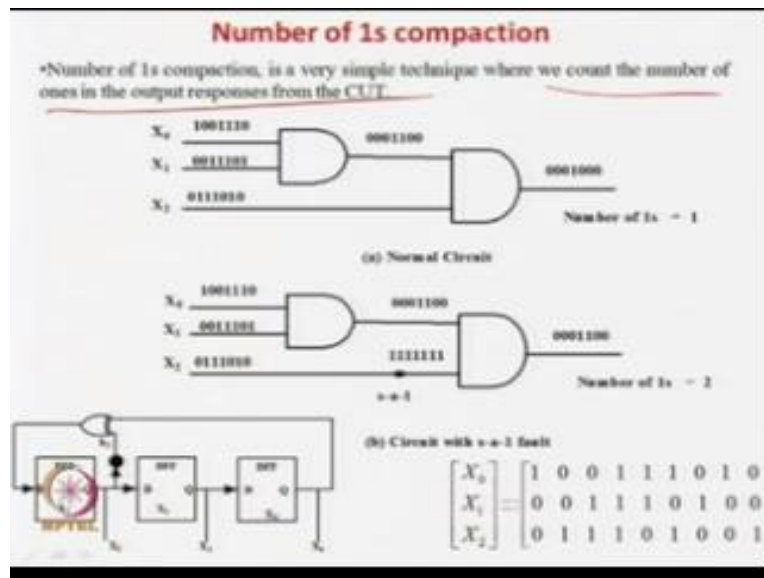
Then it call an aliasing so your target should be first set operation so that aliasing is minimize so that will be expanding to examples so this is actually called that will be in more clearly so some simple technique for compression there lot of techniques you can understand but there some logic compression are and non logic compression are there.

This is a huge area in case of image processing, image compression in communication and lot of things so lot of theory exists in case of compression and again recovery and all those things so that will not anything scope of this course so what we will see. We see two very

simple what you call compression techniques what is the number of ones and the count of the output.

So there many other lot more kind of compression techniques like LFSR based compression, multiple input signature based compression and so far. So there all part of the advance testing kind of a stuff so as I already discuss in this course that is course covers overview of testing design and verification in a one umbrella. So, we are not going details of all this. But they are the two very basic compression techniques and will be dealing in detail with this okay.

(Refer Slide Time: 30:24)



So we will first see about number of wants compression so what is the idea so the idea is very simple it counts the number of ones in the output like for example let us start with an area as on the name the technique whether you count the number of ones in the outputs of the circuit so let us see how it's a compression so what the idea is that if you look at the old I mean first LFSR we are studying in the circuit so this if you can consider about the first LFSR that is the standard LFSR so this the sequence of patterns to be generated.

(Refer Slide Time: 30:38)

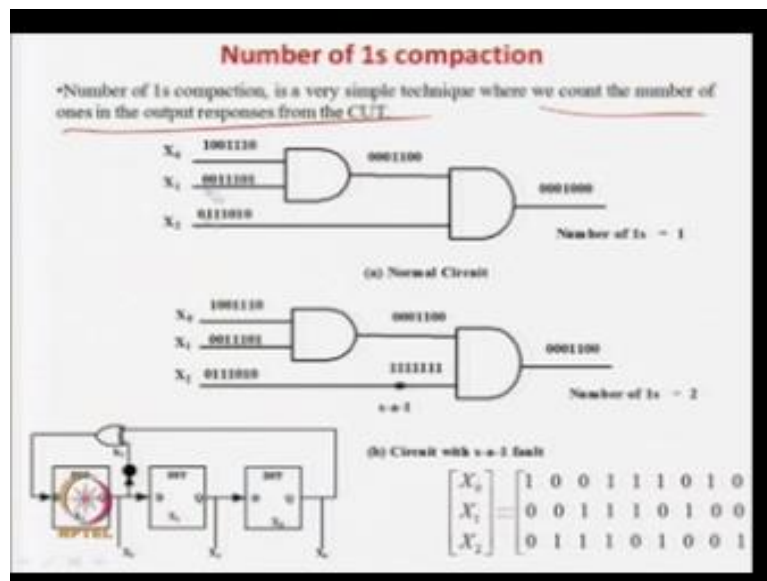
**Standard LFSR: Example**

$$\begin{bmatrix} X_0(t+1) \\ X_1(t+1) \\ X_2(t+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_0(t) \\ X_1(t) \\ X_2(t) \end{bmatrix}$$
$$\begin{bmatrix} X_0 \\ X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

So the LFSR generates 7 patterns (excluding all 0s) after which a pattern is repeated. It may be noted that this LFSR generates all patterns (except all 0s) which are generated by a 3 bit counter, however, the area of the LFSR is much lower compared to a counter. In a real life scenario, the number of inputs of a CUT is of the order of hundreds. So LFSR has minimal area compared to counters (of order of hundreds).

So if you look at this will be the all set of pattern which is require for testing so you have used this kind of LFSR so you just that is your assumption so that assumption is generating that those patterns so you can see these are your patterns so that is the generated by that standard LFSR okay.

(Refer Slide Time: 30:50)



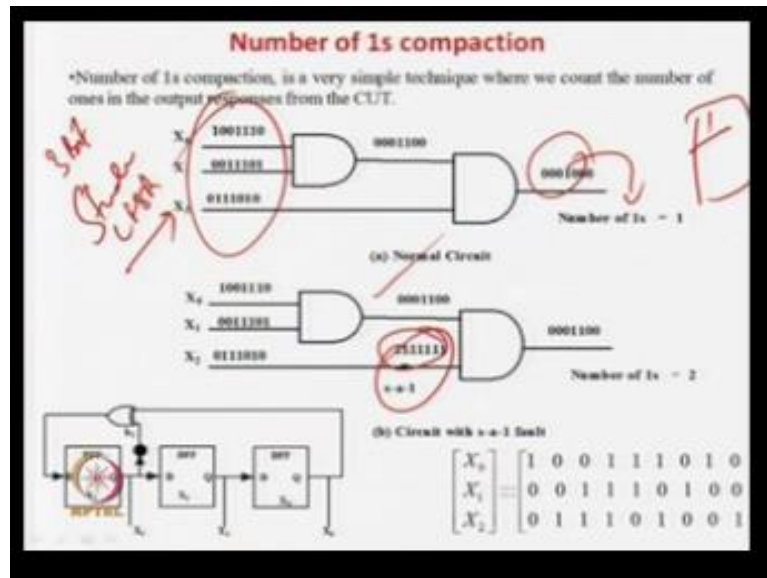
So now we are giving this as input so assume that all these were very important for testing in all case now if you have assume the circuit is in the normal compression so it is the 100 so the output over is 0 the output is 0. So only in this case were all the inputs are 1 like this 1 so only one case all the inputs are 1 so the answer is 1 all other cases the answer is 0. So if you apply the patterns from the standard LFSR discussed in the last class as input should be circuit your output will be this one all 0s and 1 and then again 0 so number of 1s in this case is 1 so what your ROM will record wrong will say that if the standard LFSR.

If you are giving as this is the input from that one that this 3 bit LFSR last example were giving their your input wrong will say that circuit under a normal case will have only a single 1 so the ROM will store only 1 for this whole set of input but if you are in storing what you can call if you are using the slash storage then what will be the case you have to store this whole 1001.

But not assume that there will lot of output from the circuit then you can assume that the number of response and then have to be stored for this output which should be quite large in number inducing size of your ROM so even if in case of a single bit output your ROM should have stored this whole value for this series of inputs. So again this will take a large size in the ROM so we have done a compression. So what is the compression light it is number of faults

in the circuit so in the output of the come so there is only one so you store a one so only 1 bit is required in a ROM where let us consider a socket 1 fault over here.

(Refer Slide Time: 32:28)



So socket 1 fault will here will make all this case all 1s like this is 1 vector and suppose 7 or 8 vectors were applied 7 vectors are applied because all 0s cannot be applied so this our standard LFSR which were using this was the characteristic polynomial already we have seen that okay so now your this is the socket 1.

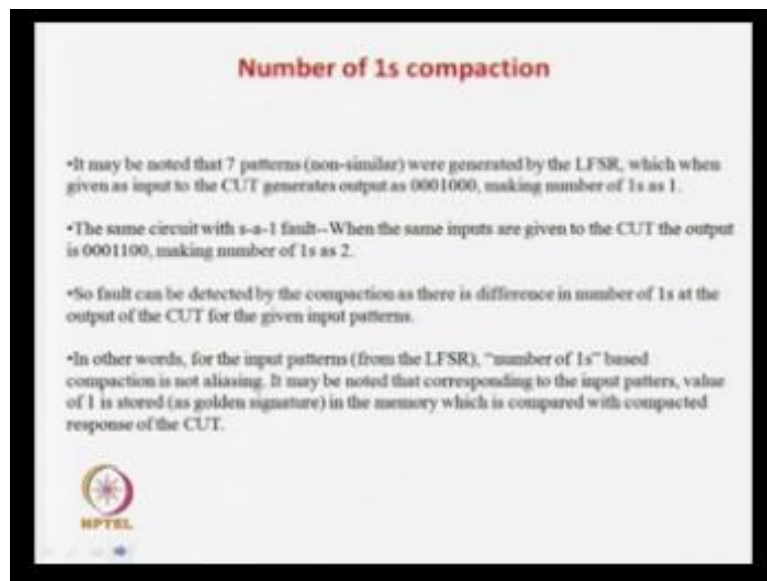
So for all cases so if you apply here 100 in normal case 0 is applied so the normal case it should be have been 0 over here and you can see over here but now is the socket 1 fault every time it will be a 1 so you will have seven 1 because in the series of the seven test pattern are applied in this case every instance should be 1 with the socket 1 fault now this again it is normal so you get this value same over here.

But now in the previous case so I mean we got only one over here but in this case is also get one over here because in this case we will get one over here because of this case we have another instance of all 1s so you get the number of 1s as 2 in this case so if you just apply all the patterns with the socket 1 fault over here so we will find out there is one more instance of this one were both are one and you get answer as 1.

So here number of 1s is at now what happens if you apply the set of patterns and these set of patterns and ROM you have stored 1 but output response getting showing 11 so in this case you detect that is the socket 1 fault over here. So in this case we can say that what is the requirement the requirement is ROM with the single bit ROM over there and single bit 3 bit LFSR are with this one so it can test the socket 1 fault here.


Similarly this is test of pattern series of other faults over here okay that we can easily find out but by using 1 bit before so emphasis here is that so you have done a compression 7 bit to 1 bit that is by counting in number of 1s and there was no idea seen for this point so no idea seen for this fault means what that is we will with this compression. If the fault is there you can detected and if fault is not that also can be detected.

(Refer Slide Time: 34:31)



**Number of 1s compaction**

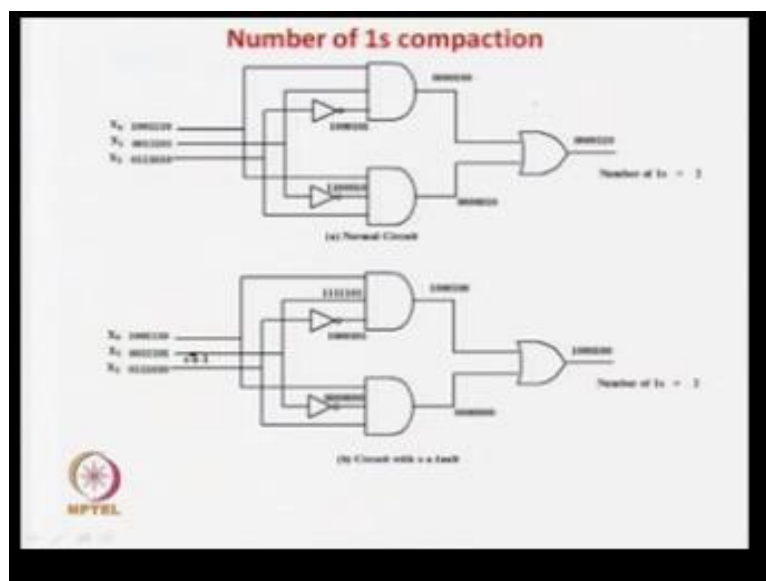
- It may be noted that 7 patterns (non-similar) were generated by the LFSR, which when given as input to the CUT generates output as 0001000, making number of 1s as 1.
- The same circuit with s-a-1 fault--When the same inputs are given to the CUT the output is 0001100, making number of 1s as 2.
- So fault can be detected by the compaction as there is difference in number of 1s at the output of the CUT for the given input patterns.
- In other words, for the input patterns (from the LFSR), "number of 1s" based compaction is not aliasing. It may be noted that corresponding to the input patterns, value of 1 is stored (as golden signature) in the memory which is compared with compacted response of the CUT.

 NPTEL

Now to explain more explicitly will turn example where there will be aliasing so now just let us I am just repeat what I have was seeing about wants compaction so in the last case we see that there was 7 non similar patterns and I mean there is no repetition patterns were reflected to LFSR the circuit generated this all so the number of 1s is a 1 so this is the compression the same circuit when the socket fault there are two 1s so number of 1s is 2 so as it is difference in the number of 1s so the fault is detected.

So we can say that number of 1s bit compaction is non aliasing because in this case the fault could be detected in non aliasing for the fault so it may be noted the corresponding input values of 1 is stored as a golden signature in the memory which is compared with the response of the cut. So that is the single bit response compaction at the more important stock here was that is the fault could be detected so and there was a no compaction in this one so again but now we will see that this not life is always as green and they always tell so now in this case will see another example where there will be aliasing.

(Refer Slide Time: 35:21)



So this they was to gather circuit in this case and same sequence of inputs are given as this one because and you are using this circuit because is the generating a near exhausted sets of patterns excepting your 0s so you apply this so you can easily find out what will be the values over here and here if you find out this value so it will be so like it.

I mean there will be two 1s in this case because if you find out this OR gate so this is the one instance and another instance so the two instance there will be output is the 1 the number of 1s in this case is going to 2 now let us take a case where there is a socket 1 fault over here so the socket 1 fault here means all the case every time it will be all 1s will be the case so if you consider the one over here.



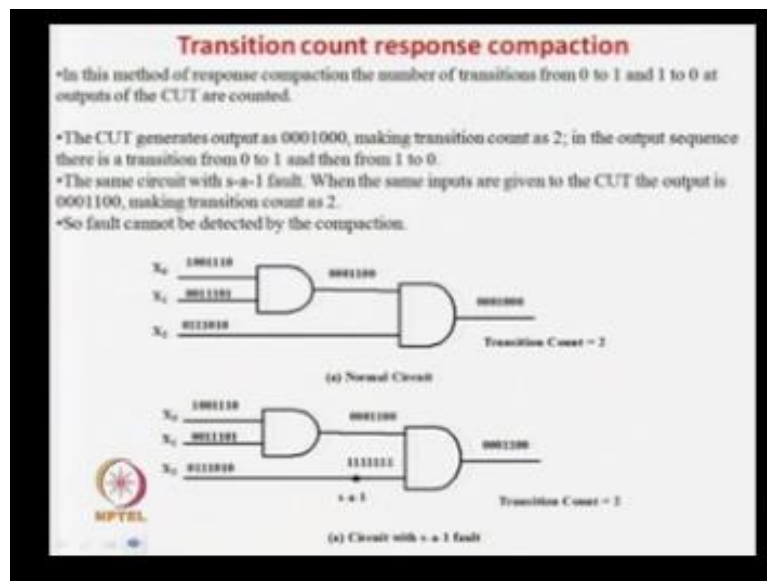
So welcome to lecture- 11 of module 11 in testing. So build in self test so in the last lecture we are looking at the this case that with assumption that I mean whatever test pattern are or what are testing we are discussing at length we are off line testing so what you mean by off line testing that is the fabricated, it is put in the ATA and we are applying test patterns to test, but then we also saw that now is due to the deep of micro design what happens that even if the socket is fine when we have sold it off or when it has been shift to the vendor I mean shift perform the debtor through the customer even after putting into this is system it may have failure because of the deep design.

So all these type of soft failure and what you can called failures and it is seen in the picture so to have them what the idea was that to test that so what we did is we found that we can develop technique in which case what we can do we put version of the test, but on generator and version or what you can called subset version of the test pattern analyser or test pattern compactor.

So now everyday when you take starts your operation what you do you active your test pattern generator and that is the version which is kept on zip and you use your random test pattern comparator to find out if anything is gone wrong. So if you can do that at every start of your socket actually we called as it build in self test.

So in the last lecture we have seen that what we basically require in this architecture so we require test controller we test pattern generator and area should be very less because it is on zip socket so that area consist is very high. So we have seen that we can use the linear feedback ship registers and we have seen as standard LFSR and how using serious of simple sockets we can easily develop near exhausted kind of test pattern, they are just like excluding the case of all 0000 we can generator pattern from 1 to 2 power in -1 kind of staff and subset of them are required for testing and the area requirement is nothing but single socket.

(Refer Slide Time: 41:19)



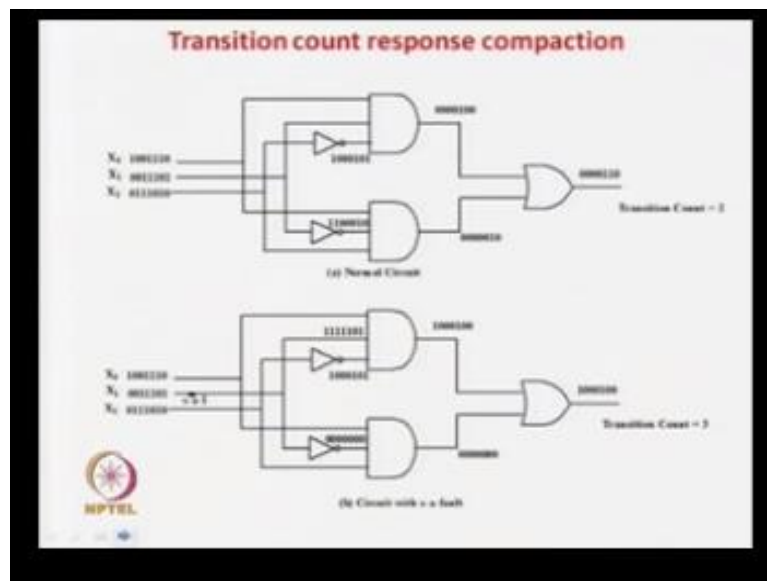
As we are seen in the example in the end of last lecture or it can few excoriate and that is much, much less than gait required by determination test pattern generator that will like you count like 1, 3, 5, 7, 9, 11 and 32 something like that so if you go for determination test pattern generator like this so on zip socket will be very high because you have to go for final statement mission design get optimization and all and finally you can see that you will require much more than few socket which is the case of standard feedback LFSR so I mean standard LFSR or any other type will see.

The basic idea is that sugar random test pattern generation using linear feedback ship registers are much more area compare to standard determine ship test pattern generator by using the final statement machine approach of digital electronics if you do so because of this low area requirement of LFSR we find them are very good candidate for this pattern generator in case of build in self test. So in the last lecture if you remember so we have seen the something kind of LFSR of this nature which we called as standard LFSR.

Today we will see another type of LFSR, but why do we require another type of LFRS we are just quickly get them modification so what was the idea is that you can see that in case of standard LFSR there is chain of socket like this if there is also another feedback from the XR socket so you require another XR socket over here.

So if you are using number of one counts as I was discussing if you are using number of one count so for here so in this case if you are good you are lucky because of that fault is not having a aliasing effect. But now if you have selecting your scheme at the number of transitions then what you are going to have a number of one transitions if you are using so in both the cases with and without the faults you are going to have a other transitions count and then they will be an aliasing.

(Refer Slide Time: 42:59)



Now just as second circuit has transistor so this is circuit has stuck at one fault okay. Because of this circuit one fall so this inversion also this same circuit we have to consider now this will leads you a output and default this one is a output. Now if you consider here is that the number of transitions is up to one transitions but now if you look at this output is fault so the number of transitions count is one transitions count and another transitions count.

But if you do the number of one is two and this gives 2 for this circuit and this fault counting once is having an aliasing when the number of transitions has are the transitions count response compaction is not an aliasing effect. So that is what I told you is that depending on circuit depending on faults an depending on compaction technique they can be different aliasing effect.

So they are for what you have to do you know what are the patterns that will be generated so you can decide that by your random pattern generate by that your aliasing effect. Now you have to find out a compaction scheme then you have to find out which is the most important fault and which are getting deducted and which are not getting deducted because of the aliasing effect.

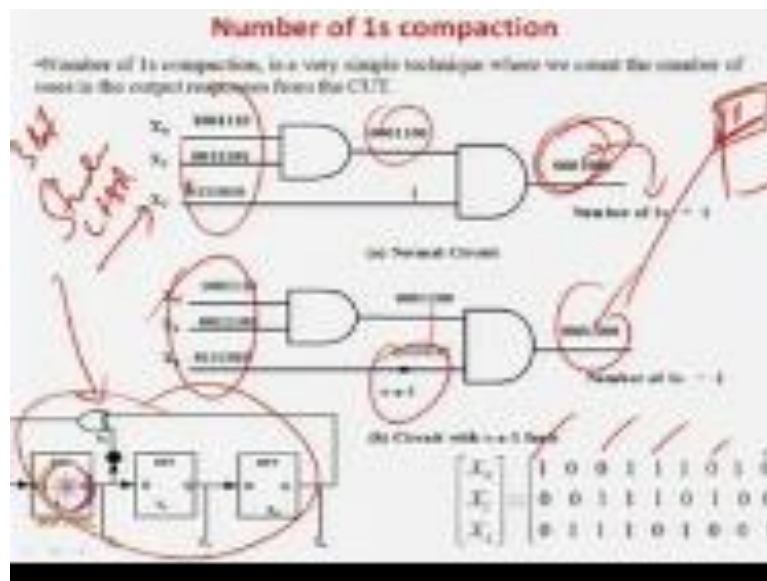
Then accordingly you find out the coverage and you do your design basically it is a multi criteria optimisation problem as you can see so all are interdependent variable. Like say there are 10 very important faults which you improve deducted and this ROM area you know so you have to find out which is the best compaction gate.

So depending on the best compaction technique are depending on the compression technique some faults are deducted or some faults are not deducted .again which faults will be deducted dependent upon which input patterns you have applied. So if you have an exhausting technique pattern generation like if you are using characteristic primitive polynomial is it fine.

But sometimes we may not be using a primitive polynomial using may wonders much smaller area of the LFSR or you may not be wanting the full par generation that may better much more area then what is the case you can use a another LFSR without a primitive polynomial should not go for a exhausted pattern generation.

But you can have a whatever is equal to that number of patterns but again for a compaction technique you have used and these subsets of test patterns you are using for this pattern generation may have a very high areas that means what that is you have taken a compaction scheme now that compaction scheme is determining which are the test patterns you have to apply.

(Refer Slide Time: 45:30)



So for an example you have a selection a compaction technique is very good area is less and you are getting very less aliasing effect. But the test patterns to be used are this one this one or all alternate. So if you are using all alternate then you have an problem then you have to RAM through the whole LFSR.

But if you are using an compaction technique same compaction technique may be you are using this sequence of this technique then you have an lot of aliasing. So then you had a problem then you have to use this one this one and alternate test and its take a long time to generate.

And also have an case may be here in which case if you are using only this four then you are using compaction techniques then your coverage is very very high. Then you add a gate design then your life is good because only by using this four consecutive patterns you can apply very quickly form your LFSR and all these faults are been tested by a compaction technique.

So that is why it becomes a important design challenge that given a compaction technique or sometimes you may have to required to change a compaction technique. Because for an example if you know that if i applied these alternative patterns it takes a long time to generate

so what i can say that can i have these patterns or can I have at least approve this number of patterns someone of them so I can generate this patterns very quickly from this LFSR.

But still the compaction technique is less amount of aliasing. So all those things to be studied by the sectors and there are lot of CAD tools over here which will tell you the results that you given as circuit then you are saying that if i want to said these are the test patterns and these are compaction technique what is the fault coverage.

Then it may if you are getting low coverage then if you have to hither change your compaction technique or change the patterns from there and so for. So again that is why i told you is a interior solution so by looking at all these things have to decide which test patterns have to applied which is the compaction technique you to use and depending on which test pattern you will apply.

You can decide it on your characteristic polynomial or you can decide on your C. So that is why it did design of the best of your challenging problem so that you can appropriately covered most of these faults at the same time you can do it very quickly using the initial set of patterns.

So one more important thing before you stop for the discussion that is two components technique you told gate so if it n bits output then what is the gain in storage. so if there are all ones so if all ones then what will be your the case then you require how many need to storage so there will be end number of ones.

So what is the number of this sequence is to be storage then it will be lot to end similarly transition count can be 1 so 10101010 so 1, 2, 3, 4, 5, 6 so you can get the order of n. So if you are storing in this fault so you require n bits to store but if i count the number of 1 if everything is in once then the whole squares then the value will be n.

And how many bits are require to store in the RAM it will belong to the end. Similarly in case of transitions so transitions count so in the most case there will be all the followers may flips and the value can be at most in the order of n. so number of bits require to store n in a RAM are in binary is again long of too n.

So if a RAM size if you store this output flats so you require  $n$  bits to store okay if output is  $n$  bits. But if you are using on count or transition count so you require long too  $n$  so that is the amount of factor we have safe in this case of these two compaction scheme. But again aliasing effect and all those things.

Now also to understand the factor for one such output  $n$  it is a large number of inputs so you will be saving in every output you will be saving the values of  $n$  to log off okay. So that was about our discussion on BIST architectures so we have discussed very important components of this one in the random pattern generation using LFSR and also seen other most important component of this response compaction and compression.

So compaction error is very simple it is compression between two compression with one so it just a bit wise compression can be used. But very important part is how you compact your answer response. So compaction should be said as an aliasing is minimized. So in this case we have given an example that given the circuit given the fault given the input pattern and the compaction is nature the aliasing has to be changed.

So it is our design problem and you have to find out which one is the best fault okay. So now we will go the question and answers session. So in a BIST how do you know which LFSR is to be used for test pattern generation that is a very important problem that we were discussing last now that given a circuit how to decide which is an LFSR how do you decide which is your pattern generator and how do you decide which is your compaction technique how do you decide to go to C so all will depend on how quickly you can do your tested and within minimum number of patterns and how good coverage you can taken.

That the two many things you need to achieve so first is that LFSR is as the number of inputs that is very obvious any input circuit then there will be  $n$  bits required in the LFSR so it has to be  $n$  bits so it is an obvious. Secondly the characteristics polynomial should be primitive polynomial and it generate all possible so this is an general statement if you have most of the cases you may be required where all patterns to do the testing. So you can go for a primitive polynomial so in other cases all 0's cannot be generated as we will see it very soon.

So other way you can find out the which is the most important patterns to be generated so according you can say the primitive polynomial you can use the it may not be characteristic

polynomial if you are targeting on few of the vectors that you can do and also but for exhausting pattern you can give primitive polynomial all patterns to be generated.

So in next class or in next part of the course will have an another very interesting part that is memory till now we have seen the circuits how to do the circuit in offline mode also seen in inceptive mode so this is the third part of the circuit memory is not all combination sequential circuit it is a special design circuit that stores a lot of value or lot of binary values in circuit without taking that much amount of area.

So simple test scheme do have seen like D algorithm ATPG do not generally hold for our memory. Memory architecture is quite different from normal sequential and compaction so in next two lectures will be discussing about how you can do testing of memory

Thank you.

**Centre For Educational Technology**

**IIT Guwahati**

**Production**

**Head CET**

**Prof. Sunil Khijwania**

**CET Production Team**

**Bikash Jyoti Nath**

**CS Bhaskar Bora**

**Dibyajoti Lahkar**

**Kallal Barua**

**Kaushik Kr. Sarma**

**Queen Barman**

**Rekha Hazarika**

**CET Administrative Team**

**Susanta Sarma**

**Swapan Debnathi**