

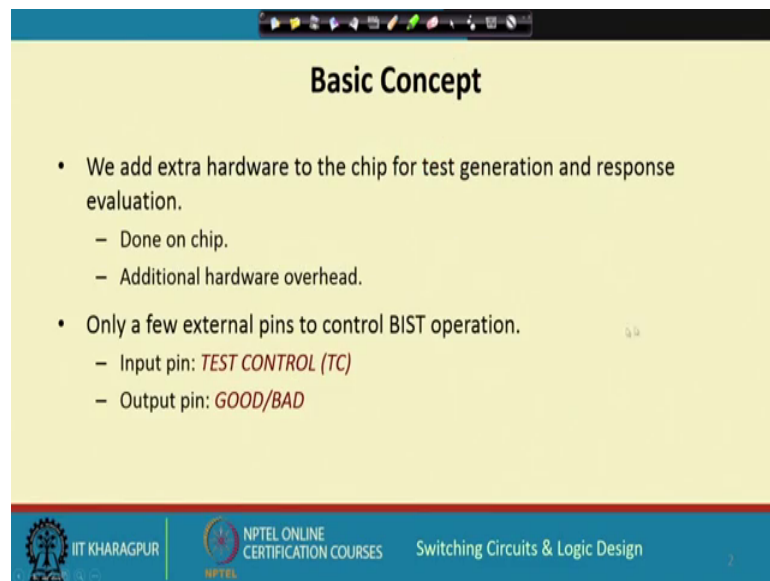
Switching Circuits and Logic Design
Prof. Indranil Sengupta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 59
Built-in Self-Test (Part I)

Continuing with our discussion on testing, in this lecture we shall be talking about something called Built in Self test. Well, in some sense you can say that this is one kind of extreme in the area of testing because what we are saying that the circuit that you want to test will have the ability to test itself.

So, a circuit can do something called Self Testing, this is the basic idea and let us see how it works. So, this is a first part of this topic on built in self test.

(Refer Slide Time: 00:57)



Basic Concept

- We add extra hardware to the chip for test generation and response evaluation.
 - Done on chip.
 - Additional hardware overhead.
- Only a few external pins to control BIST operation.
 - Input pin: *TEST CONTROL (TC)*
 - Output pin: *GOOD/BAD*

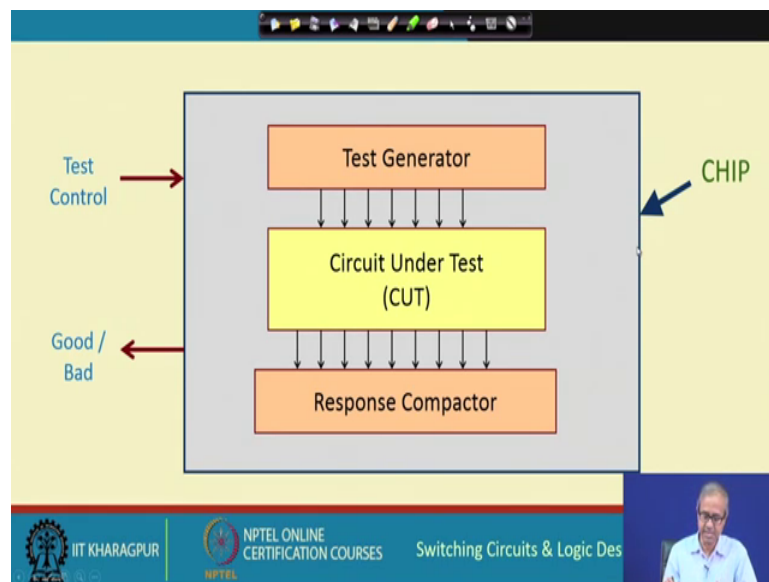
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Switching Circuits & Logic Design

Let us first try to understand the basic concept what the idea is all about. Well, the first thing is that we want to test a chip, some circuit which is there in the chip. First important thing to note is that we put in some extra hardware in the chip. And this extra hardware we will have 2 responsibilities. First is that it will automatically be generating some test vectors and it will be evaluating the responses of the circuit. Both these things will be carried out by the extra hardware.

So, from outside we need not have to apply any test factors and on the outside we do not have to evaluate the circuit responses. Everything will happen inside within the chip or within the circuit.

So, this as I said this will be done on the chip and because of the extra hardware there will be some additional overhead involved. And to support this kind of built in self test in short BIST operation, we need two additional pins; one is an input pin which will be something like an activation signal. I can tell the chip that well now you can test yourself, test control and one output pin which the chip tell us that whether the chip is good or bad. This is how it works.

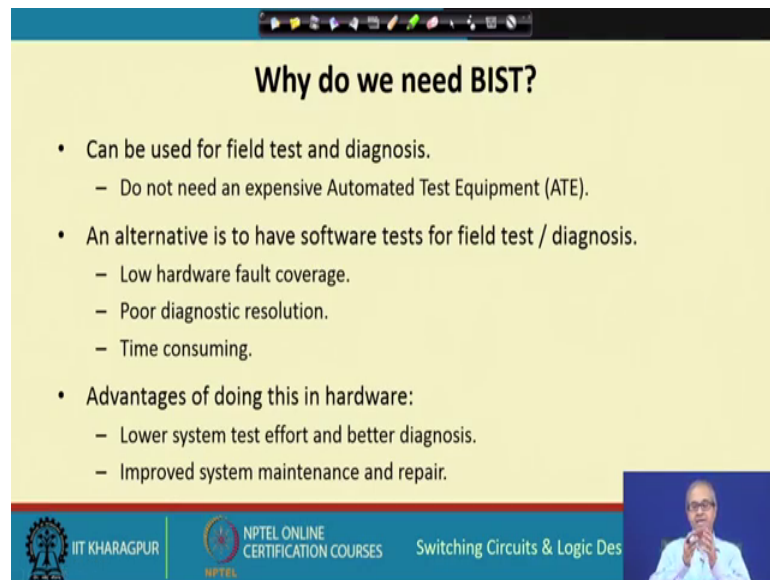
(Refer Slide Time: 02:35)



Now, this is the very high level schematic. This is the schematic of a chip where I have said we have two additional pins; one is something called test control and other is good or bad. The chip will report to the user to the outside world whether the chip is good or bad. Now, if you see inside the chip in addition to the circuit you want to test you will be having some test generator circuit and also some response evaluated call it response compactor circuit inside it.

So, we will be looking into the detail of this test generator and response compactor in the next few slides.

(Refer Slide Time: 03:27)



Why do we need BIST?

- Can be used for field test and diagnosis.
 - Do not need an expensive Automated Test Equipment (ATE).
- An alternative is to have software tests for field test / diagnosis.
 - Low hardware fault coverage.
 - Poor diagnostic resolution.
 - Time consuming.
- Advantages of doing this in hardware:
 - Lower system test effort and better diagnosis.
 - Improved system maintenance and repair.

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

But before that let us try to understand the basic question why do we need BIST, what are the additional advantages we get if we have a built in self test facility in a chip. Let us try to understand these advantages.

First and most important thing is that we can do field test and diagnosis ok. What is meant by field test? Field refers to the area where the chip is actually being used. While it can be used inside our mobile phones, it can be used inside a computer system, in a laptop or in any specialised circuitry which is deployed anywhere in an industrial control plant anywhere. So, wherever it is used that is called the field. And what you are saying conventionally the chips will be tested in the laboratory before hand and then they will be put in the circuit and then they will be used.

When I say that they would be tested in field it means, let us say a laptop, while the chip is inside the laptop the chip test itself. We do not have to take the chip to the lab or the laptop to the lab for the purpose of testing right.

So, we do not required something called Automated Test Equipment which is normally used to test circuits or chips which is typically a very expensive equipment.

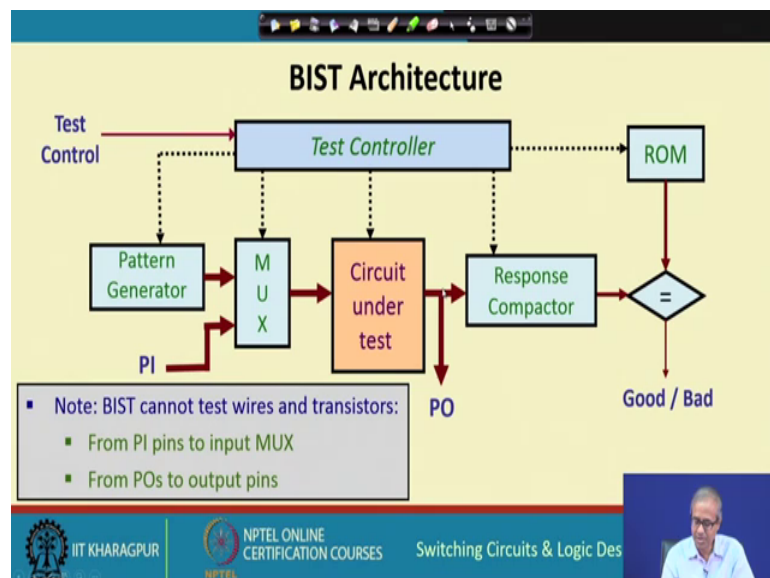
Now, you can also want to compare this BIST approach to something called software test for field test and diagnosis which normally we see in our PC's or laptops. Whenever you turn on the PC you might have seen that there is some self testing going on, there are

some messages which are coming. And in case there is some problem either in the motherboard or in the hard disk interface somewhere there will be some error message either in the form of a audio beep or in the form of a textual message that will come that come a failure in the memory system also ok. So, these are called software tests for testing and diagnosis because there is a software which is running which is trying to test the various subsystems.

But the problem is that the fault coverage is typically not that high. And you cannot have good diagnosis. You can say just the motherboard is bad, but exactly which chip of the motherboard is bad you cannot tell that you cannot pinpoint that right. And of course, it is time consuming; it takes many seconds to complete the test. But if you do it in BIST in hardware the advantages is that you can have much better diagnosis. Why? Diagnosis means to locate the source of fault.

So, if every chip contests itself then the chip which is faulty can tell you that well I am bad. So, you can replace only that chip. And another advantages that because of that you can have much improved system maintenance and repair capabilities. These are some of the additional advantages you have.

(Refer Slide Time: 07:03)



This is a little more detailed schematic for the BIST schemes or the BIST architecture. Let us see in the centre you have the circuit that you want to test circuit under test. Normally, the circuit will take it is input from some primary input lines that is call it PI,

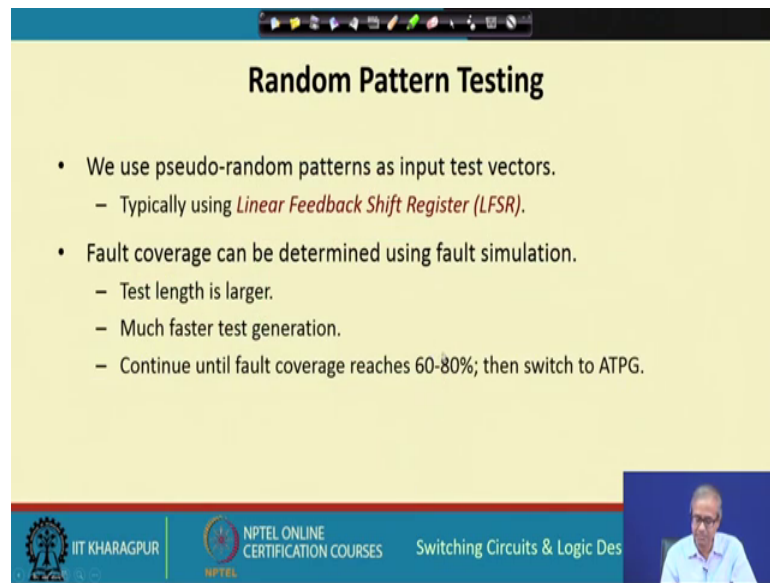
this can be coming from outside. But while in the test mode there will be some kind of a pattern generator inside the chip. The input to the circuit will be coming from the pattern generator and there will be a multiplexer here, so which will be selecting either PI or the pattern generator outputs to be fed to the circuit.

Similarly, the output side normally the output will go to some primary output lines, but now here some kind of response compactor circuit is there which will be compressing the output or compacting the output to a small some kind of signature we will talk about this.

And the good signature will be stored in a small memory, ROM which you can compare and at the end if it matches you say that which chip is good. If it does not match you say your chip is bad. And there will be a finite state machine you can call it a test controller that will be control the operations of the multiplexer, pattern generator, response compactor everything. And it will be activated whenever the test control pin goes high, I mean is activated. This is how the whole thing works.

But the point to notice that, well there are some drawbacks. See some paths in the circuit you cannot test like from the primary input pin to the input multiplexer this part you are not able to test. You are testing with respect to patterns from the pattern generator, but from the primary input if there some fault in this path you are not able to test this. And similarly the primary output the circuit output your compact in the response compactor, but if there is any fault in this part of the circuit sorry then this part will not be able to test. These are some of the drawbacks here.

(Refer Slide Time: 09:34)



The slide is titled "Random Pattern Testing" and contains the following content:

- We use pseudo-random patterns as input test vectors.
 - Typically using *Linear Feedback Shift Register (LFSR)*.
- Fault coverage can be determined using fault simulation.
 - Test length is larger.
 - Much faster test generation.
 - Continue until fault coverage reaches 60-80%; then switch to ATPG.

The slide footer includes the IIT Kharagpur logo, NPTEL Online Certification Courses logo, and the text "Switching Circuits & Logic Des". A small video inset in the bottom right corner shows a man speaking.

Now, with respect to test pattern generation inside the chip the standard technique that we follow is some kind of random pattern generation because we talked about test pattern generation earlier. Well, we can spend a lot of time and effort to generate an optimum set of test patterns, but when you say I want to generate these test patterns automatically by hardware how will you do it? If the test patterns do not have any relationship between them then the only way you made it is that you will have to store it in some memory and from the memory you cannot put the patterns one by one, but the trouble is that the overhead of the memory can become high.

Let us say if we need 1000 test patterns, you need a large enough memory to store 1000 patterns and also on the output side you will be requiring 1000 circuit responses ok. So, your hardware requirement will be larger. So, the standard way is to use some kind of random pattern we call it pseudo random patterns because these random patterns can be repetitively generated. And how do we generate? We already discussed this kind of a shift register structure using linear feedback shift register which we mentioned. It can generate very good random patterns. So, using LFSR you can generate these random patterns.

And well you know what is meant by fault coverage. In BIST, I may want that while I require 95 percent fault coverage, but how many such random patterns are required to achieve 95 percent coverage I cannot predict beforehand. This has to be done a priori

through fault simulation. You can generate the patterns from this LFSR, you can carry out fault simulation and find out how many faults are getting detected.

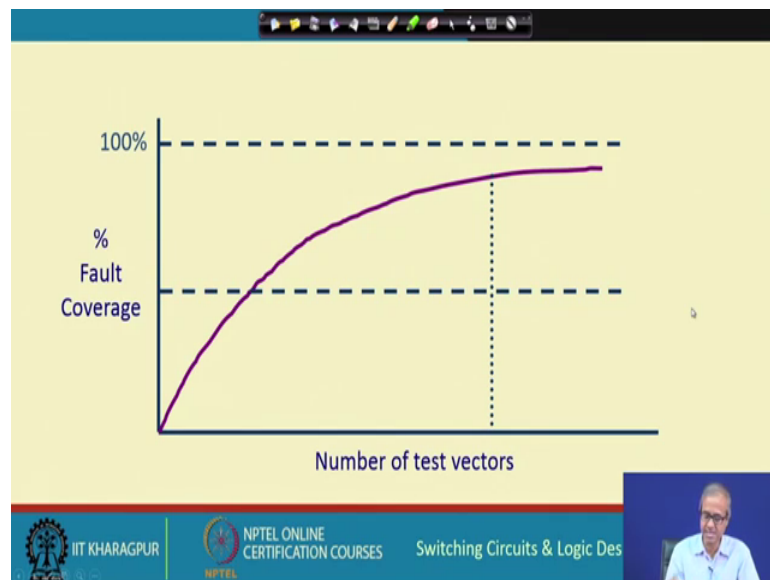
So, as soon as it reaches 95 percent you know that how many patterns need to be generated and that way you can configure your test generation inside the chip.

So, there are two things because of LFSR based test generation a test length may be much larger. Like you may required only 10 test patterns to be generated, but LFSR the patterns are generated in some random order.

So, there you may find to achieve 95 percent you may need to generate let us say 200 test patterns. So, the number of test patterns required maybe larger, but you really do not care because a generating tests at a very high frequency. So, it will take hardly a fraction of second.

Much fastest test generation; so, this is normally how it. There is sometimes you can combine random pattern with automated test pattern generation based testing also, but here we are not discussing this.

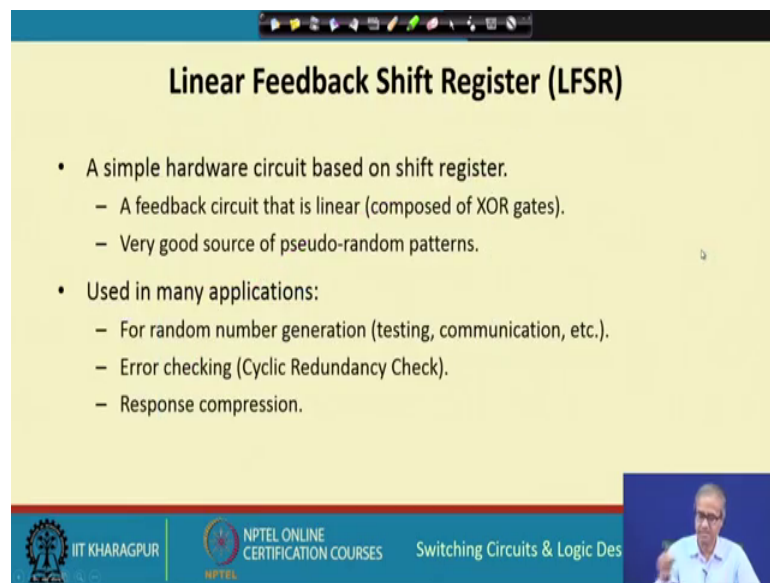
(Refer Slide Time: 12:53)



So, this is the typical behaviour of a circuit. So, as you increase the number of random test vectors and through fault simulation, if you calculate the fault coverage, you will find that it will increase like this, initially very rapidly, but it will slowly level off. Well, for very rare cases it will reach 100 percent, but normally it will level off in the range of

let us say 80 percent, 90 percent, 95 percent like that. So, you can fix up some acceptable level, you reach up to that you see that how many test patterns you will be requiring to reach this acceptable level, this is your acceptable level ok. So, you run the LFSR for these many clock cycles, fine. This is how you proceed.

(Refer Slide Time: 13:46)



The slide is titled "Linear Feedback Shift Register (LFSR)". It contains the following text:

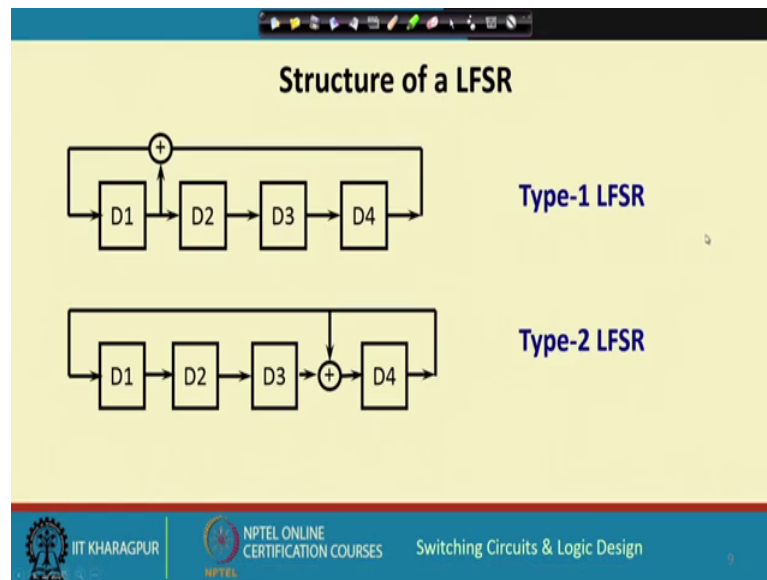
- A simple hardware circuit based on shift register.
 - A feedback circuit that is linear (composed of XOR gates).
 - Very good source of pseudo-random patterns.
- Used in many applications:
 - For random number generation (testing, communication, etc.).
 - Error checking (Cyclic Redundancy Check).
 - Response compression.

The slide footer includes the IIT KHARAGPUR logo, NPTEL ONLINE CERTIFICATION COURSES logo, and the text "Switching Circuits & Logic Des". A small video inset shows a man speaking.

Now, coming back to linear feedback shift register, we have earlier very briefly talked about it. Let us have a relook into it. Linear feedback shift register is a simple hardware circuit which is based on shift register. You recall there is a feedback circuit which is consisting of Exclusive OR gates and earlier we told that exclusive OR is a linear function. That is why we call it a linear feedback shift register. And it has been found that LFSR can generate very good pseudo random patterns.

And talking about applications, not only for testing there are many other application where this LFSR is used. For error checking, using cyclic redundancy check, later on for response compression or compaction we shall see it later, there again LFSR will be using, for data communication application where there can be errors you can again use LFSR for error detection and stuff like that ok. So, there are many applications, but here we are interested in test pattern generation for the time being.

(Refer Slide Time: 15:06)



So, LFSR looks like this. There can be two types of configuration. Earlier, we looked at only the first kind of configuration. This plus is actually an Exclusive OR gate. So, we have just shown it like this. This is actually a two input XOR gate. So, one input is coming from here. And other input is coming from the output of D4 from here and the output of the XOR gate is feeding to the input of the first flip flop.

So, there are 4 D type flip flops which are connected as a shift register. And we use a linear feedback circuit using Exclusive OR and from some tapping point we take the feedback connections. This is called type 1 LFSR.

Well, you can have another kind of LFSR design also where the Exclusive OR gates are not outside in the feedback, but they can be just inside in the forward shift register path that means, this Exclusive OR for example, can be connected directly like this. One input coming from here output of D 3 and one input coming from out of D4 from here right and output will go to D 4. So, like this you can have here here here, it depends where you want the tapping points. This is type 2.

Now, we shall see later this type 2 LFSR is more suitable for response compaction, but for test generation purposes normally we use type 1 LFSR.

(Refer Slide Time: 16:52)

Pattern Generation Example

$f(x) = x^4 + x^1 + 1$

Characteristic Polynomial

D4	D3	D2	D1
1	0	0	0
0	0	0	1
0	0	1	1
0	1	1	1
1	1	1	1
1	1	1	0
1	1	0	1
1	0	1	0
0	1	0	1
1	0	1	1
0	1	1	0
1	1	0	0
1	0	0	1
0	0	1	0
0	1	0	0
1	0	0	0

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

Let us take an example of pattern generation using this kind of type 1 LFSR. Now, this is an example of a type 1 LFSR. And I mentioned earlier very briefly, but let me again tell you here that the behaviour of an LFSR depends from the points from where you are taking this feedback connection ok. Now, let us say if every flip flop output you regard as the coefficient of a polynomial x to the power 4, x to the power 3, x to the power 2 and x then you see from where you are taking the feedback. And of course, this is x to the power 0 or 1. You are taking feedback from x to the power 4 and x .

So, you define something called the characteristic polynomial of the LFSR. Here, you show the corresponding terms; x^4 and x because the output of the XOR you are feeding to x to the power 0, this x to the power 0 term is always there ok. This will be the characteristic polynomial of the LFSR. This is called characteristic polynomial.

Now, let us assume that this LFSR we initialize with 1 0 0 and 0. You see yes 1 point, if you initialise it with all 0, it will remain in the all 0 state because Exclusive OR of 0 and 0 is 0, 0 will feed back. So, it will never come out of 0, but let us say we feed it 1 0 0 0.

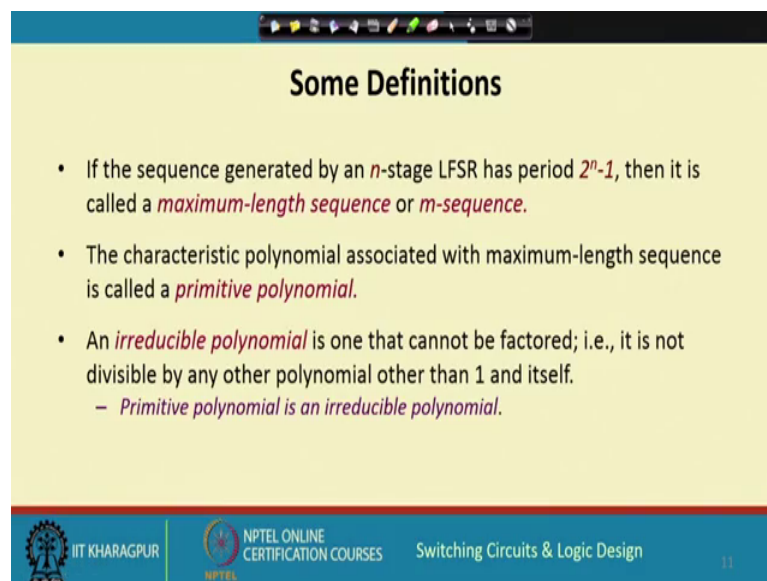
So, I show it like this. The first it is 1 0 0 0 and clocks are applied one by one. So, what will happen in the next state? You see 1 and 0 are fed to the input. So, XOR output will be 1. So, in the next clock cycle, 1 will be fed back and everything else will be shifted right. So, it will be 0 0 0 1 you see, next is 0 0 0 1. So, like that if you just check you will see that it will be generating various patterns 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15. And

after the 15th pattern again 1 0 0 0 comes back. So, you see what we have told that for LFSR the all 0 pattern if we apply it will never come out of it.

So, in 4 bits there are 16 possible patterns. So, there are 15 remaining non zero patterns. Now, for this example for instance if you start with any non zero state like 1 0 0 0, the LFSR will go through all possible 15 states and in a random order. You see if you look at the decimal equivalence you will have an idea; 8 1 3 7 15 14 12 10 5 then 11 6 12, this 13 this 12 9 2 and then again 4. So, these are apparently random.

So, you see this LFSR is able to generate all non zero patterns in some random order. This is one very good property of a LFSR that it can generate all the patterns for a given size if you choose a characteristic polynomial in a proper way. So, we will have some discussion on this.

(Refer Slide Time: 21:02)



Some Definitions

- If the sequence generated by an n -stage LFSR has period 2^n-1 , then it is called a *maximum-length sequence* or *m-sequence*.
- The characteristic polynomial associated with maximum-length sequence is called a *primitive polynomial*.
- An *irreducible polynomial* is one that cannot be factored; i.e., it is not divisible by any other polynomial other than 1 and itself.
 - *Primitive polynomial is an irreducible polynomial.*

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Switching Circuits & Logic Design | 11

So, the LFSR let us say an n stage LFSR which generates all $2^n - 1$ patterns like in the example I took earlier. We call it a maximum length sequence that the LFSR generates the maximum length sequence in short we call it m sequence and the characteristic polynomial of the LFSR that generates an m sequence is called a primitive polynomial.

So, from the point of view of test generation we are more interested in primitive polynomials because they can generate all the patterns and we need as many patterns as we want.

So, here there is a characterization of a primitive polynomial. It is a polynomial it is called irreducible which cannot be factored. It does not have any factors and this primitive polynomial is a type of irreducible polynomial. I am not going with the detail of this. Because if you want such primitive polynomial you can refer two books you will find that a big list is given.

(Refer Slide Time: 22:26)

n	Coefficients	Polynomial
3:	1 0	$x^3 + x + 1$
4:	1 0	$x^4 + x + 1$
5:	2 0	$x^5 + x^2 + 1$
6:	1 0	$x^6 + x + 1$
7:	1 0	$x^7 + x + 1$
8:	6 5 1 0	$x^8 + x^6 + x^5 + x + 1$
16:	5 3 2 0	$x^{16} + x^5 + x^3 + x^2 + 1$
32:	28 27 1 0	$x^{32} + x^{28} + x^{27} + x + 1$
64:	4 3 1 0	$x^{64} + x^4 + x^3 + x + 1$

A comprehensive list is available.

Here, I am showing list of primitive polynomials up to n equal to 64, but in the books you will see up to several thousands this kind of things given. The ideas that if we use a polynomial like this which means I need an XOR with forget 1, this is the output with 4 inputs. I need a 4 input XOR in the feedback, but it is guaranteed that I will be generating so many unique random patterns, non zero random patterns. These are all primitive polynomials.

So, I did not have to calculate primitive polynomials. A list is already prepared by someone; we can simply take from that list ok.

So, you see if I have a 64 input circuit I can take a 64 bit LFSR. And through fault simulation I can find out how many patterns I need to generate to reach a certain fault

coverage because 2 to the power 64 is a really a huge number, we never would required to apply or possible patterns right.

(Refer Slide Time: 23:48)

Some Interesting Properties of m-sequence

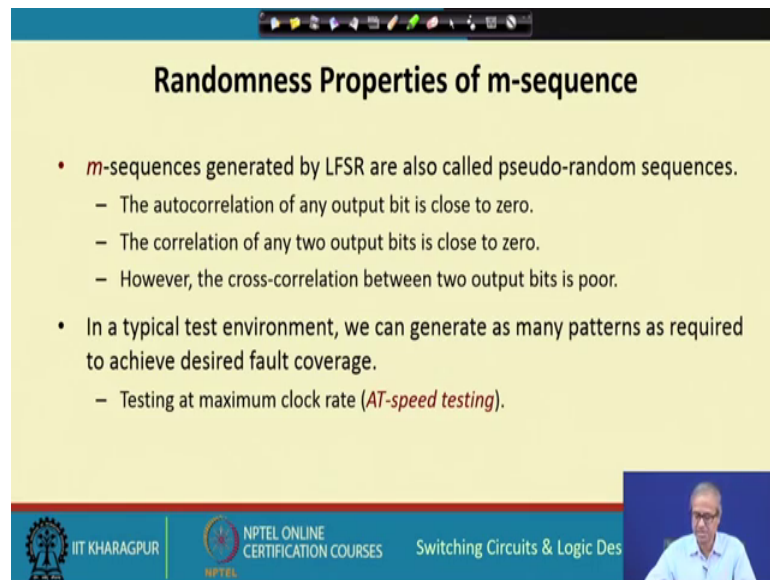
1. The period of $\{a_n\}$ is $p = 2^n - 1$, that is, $a_{p+i} = a_i$, for all $i \geq 0$.
2. Starting from any nonzero state, the LFSR that generates $\{a_n\}$ goes through all $2^n - 1$ states before repeating.
3. The number of 1's differs from the number of 0's by one.
4. If a window of width n is slid along an m -sequence, then each of the $2^n - 1$ nonzero binary n -tuples is seen exactly once in a period.
5. In every period of an m -sequence, one-half the runs have length 1, one-fourth have length 2, one-eighth have length 3, and so on.

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

So, there are a few interesting properties, I am not going into all of them because all of them may not be interesting in this context that the period of the m sequence as I said is 2 to the power n minus 1 because after 2 to the power n minus 1, the patterns repeat itself.

So, starting from any non zero state as the truth table showed the LFSR will go through all the non zero 2 to the power n minus 1 states before repeating. And in every column of the truth table if you check that the number of 1's will be different from the number of 0's by 1, number of 1's should be more by 1 just because the all 0 pattern is not there that is why number of 0's will be 1 less and the next two points are not so, important in the present context. So, let us not discuss this right now.

(Refer Slide Time: 24:50)



The slide is titled "Randomness Properties of m-sequence". It contains the following text:

- *m*-sequences generated by LFSR are also called pseudo-random sequences.
 - The autocorrelation of any output bit is close to zero.
 - The correlation of any two output bits is close to zero.
 - However, the cross-correlation between two output bits is poor.
- In a typical test environment, we can generate as many patterns as required to achieve desired fault coverage.
 - Testing at maximum clock rate (*AT-speed testing*).

The slide footer includes the IIT Kharagpur logo, NPTEL ONLINE CERTIFICATION COURSES, and the course title "Switching Circuits & Logic Des". A small video inset shows a man in a light blue shirt speaking.

So, randomness property is something which is important in our context. The maximum length sequences that are generated they are called pseudo random sequences. You see there are some standard tests for randomness. It is found that most of the random this kind of randomness tests are very well satisfied by the patterns which is generated by LFSR. Like, the autocorrelation of any of the output bit whatever bit patterns is generated is close to 0. The correlation of any two output which is close to 0, but the cross correlation is poor because it is a shift register, whatever pattern is generated by 1 bit in the next bit the same pattern will be generated by respect to 1 bit time shift. This is the only drawback. Just other than that all other properties are very well satisfied ok.

So, as it said in a typical test environment we can generate as many patterns as required. And this another advantage, you think of this scan base testing scan path which we were discussed in the last lecture. For applying every test pattern we have to serially shift some pattern in a shift register then apply the pattern which means, we cannot operate the circuit at the maximum possible rate with which circuit is to supposed to operate, but here we can. The test patterns can be generated in the maximum related frequency of the circuit here, let us say 1 gigahertz 2 gigahertz whatever it is. And this is called AT speed testing that we are testing the circuit at the maximum clock rate. This is another advantage. Many of the timing errors also get detected in this process.

So, with this we come to the end of this lecture where we have discussed how test patterns can be generated in a built in self test environment. In the next lecture, we shall be looking at the other part how this circuit responses can be compact it to take a decision whether the circuit is good or bad.

Thank you.