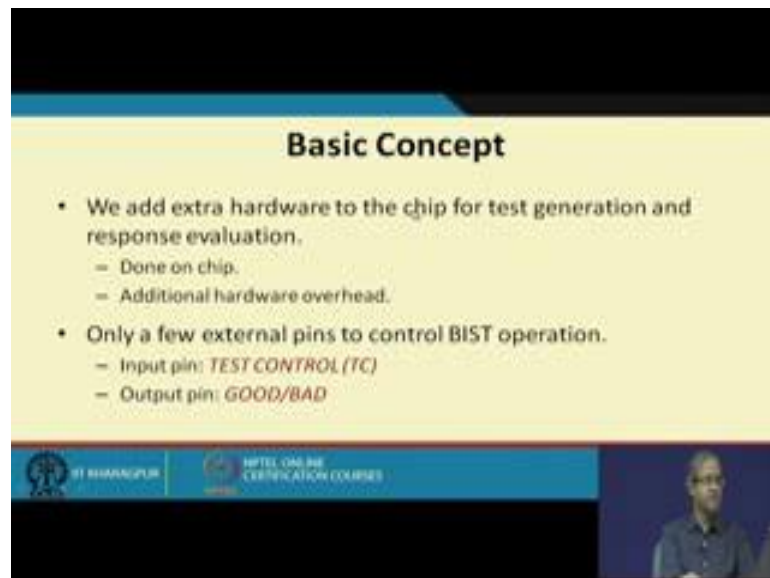


**VLSI Physical Design**  
**Prof. Indranil Sengupta**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Kharagpur**

**Lecture – 56**  
**Built-in Self-Test (Part 1)**

Now, we talk about built in self test. So, technique fair by chip can test itself, let see this salient features what are the requirements and then you shall see how we can achieve this to some extent, we cannot do as good as external testing, but what best we can do let see that.

(Refer Slide Time: 00:40)

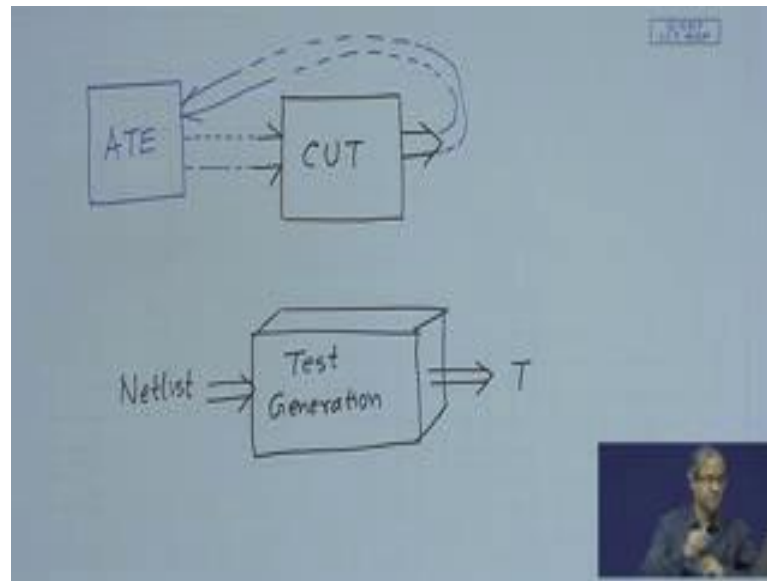


**Basic Concept**

- 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*

So, the basic concept behind built in self test is to add some additional hardware inside the chip, such that test generation and response evaluation can be done inside.

(Refer Slide Time: 01:09)



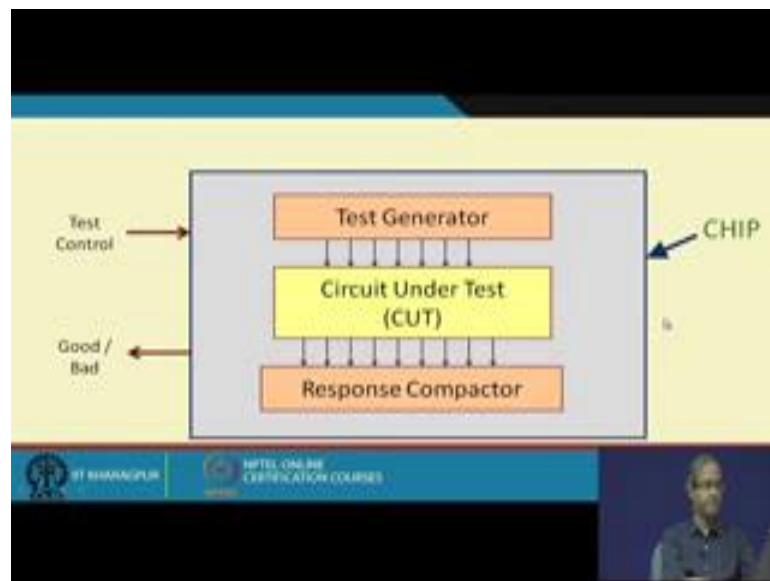
Let us see what is the conventional approach, what we do? The conventional approach is that I am just trying to show it diagrammatically here. So, I have a circuit under test CUT, I have a functional specification there are inputs there are outputs.

So, what I do? I use tool first, I use a test generation tool where just as the input I shall be providing with this circuit net list let say or this some kind of specification, as I test generation tool will give me a set of test directors T. Now this is done once, now once the circuits are manufactured what we do is, we use something called an automated test equipment or ATE, we load the test patterns in the ATE memory and ATE will be generating this inputs to this chip which will be setting on the ATE, and similarly the outputs will be fed here.

So, ATE will be in control of applying the test patterns and evaluating response, and the point you notice that this ATE is an extremely expensive equipment well although we use it, but the total cost of manufacturing also has a component which is the cost of the ATE. So, this is the more conventional approach, but here our test generation and response evaluation both we are doing inside the chip itself. So, we are not relying on an external test generator to generate the test may be you are something or some external ATE to apply the test. Naturally to do this on chip we need additional hardware. So, suppose we have implemented this kind of extra facility inside the chips so that it can test itself BIST.

So, to control the BIST operation we need minimum of 2 pins. So, there will be 1 pin which you will a test control, so here will be telling the chip that well now you test yourself, that pin if activated will cause the chip to start the testing and that will happen internally only with no external intervention, and once the testing is over there will be a output pin saying good or bad it will tell that well, I am good or I am bad. So, BIST normally will not tell anything more than that it will just tell either chip is working good, or there is something wrong that has been detected.

(Refer Slide Time: 04:21)



So, this is how now the insider of my chip will look like, earlier we had only our circuits under test, but now we have a test generator we have a response compactor, we shall come to this a little later why we need this; and externally as I said we need one control signal to start or activate the BIST operation, and we need one control signal to evaluate the result of the test, all right.

(Refer Slide Time: 04:57)



The slide is titled "Why do we need BIST?". It features a yellow background with a blue header and footer. The header contains the title. The main content is a list of bullet points. The footer contains logos for "IT MANAGER" and "INTEL ONLINE CERTIFICATION COURSES". A small video inset of a person is visible in the bottom right corner of the slide.

### 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.

Lets now look at the motivation, why do we need BIST and what are the advantages? We first one I already mentioned that we do not need an expensive automated test equipment. You see this ATE is very sophisticated equipment; it is bulky equipment it has to be installed in a very controlled environment. So, when you want to test the chip we have to go near the ATE, put it on the ATE and ATE will be just applying the test vectors and test it. But now if you have BIST feature in a chip, it is not necessarily to go to the ATE to test the chip, you can even test it in your lap that is your field.

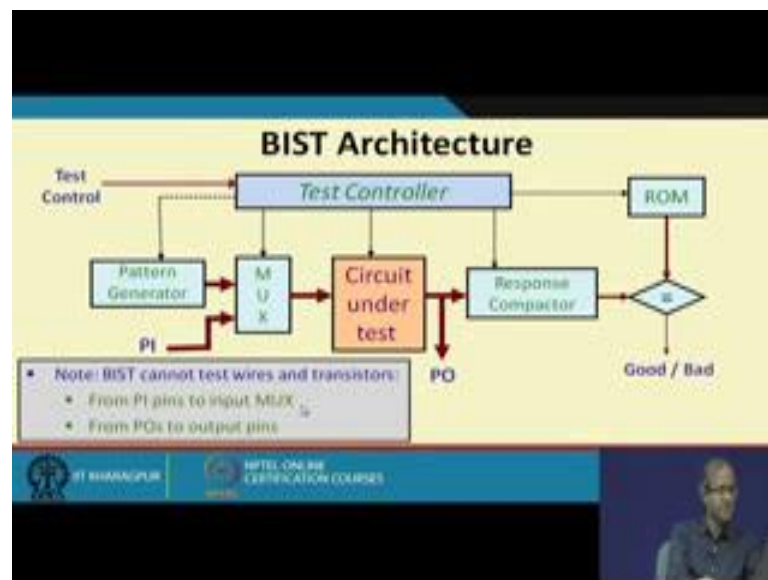
So, can be used for field test, diagnosis means you are trying to identify which chip is bad diagnosis is the identification of the source of failure. So, normally in a system we use something called software tests; like you might of seen in a desktop or a laptop when you turn on the machine for the first time, there is a software diagnostic is starts running that is a software program. So, it does some rooting checks it checks memory etcetera, and if it finds that something is wrong with reports with a code. So, we will have to decode what that code means to identify what or which sub system is not working correctly right.

Now, BIST can be an alternative to this kind of software tests, but the software tests with the problem is typically software tests they carry out testing in a very loose and abstract way, the fault coverage in terms of the hardware is pretty low the diagnostic resolution is also pretty low. Sometimes you shall only say that there is a fault on the mother board of

your PC, but where is the mother board that it does not say and also because it is a software program which is running, it will be more time consuming. But in BIST we are trying to do this in hardware, and if you do this the advantages are of course, be test effort will be lower because the time taken will be less, diagnosis will be better because chips at the chip level you can do the testing and since you can do it on field, your system maintenance and repairing costs will be also much less.

So, instead of every time there is a field will have to go to one place to find out the source of the fault, you can do it in your own small lab also fine.

(Refer Slide Time: 07:54)



Now, the BIST architecture over all it looks like this. So, I have my circuit under test which I want to test. So, what are the components I need? So, this whole picture to something that goes inside the chip of course, I need a pattern generator. So, which will be generating my test patterns, now I can argue that well I have very good test generator it can generate some very good small set of test patterns, let me keep it in a ROM and let the ROM be inside the chip. But it is possible, but the over rate of that ROM will be much higher, from circuits you may find that you need 1000 test vectors. So, you need a ROM with 1000 words just to stored that right and of course, they access time of the ROM will be slower as compare to hardware.

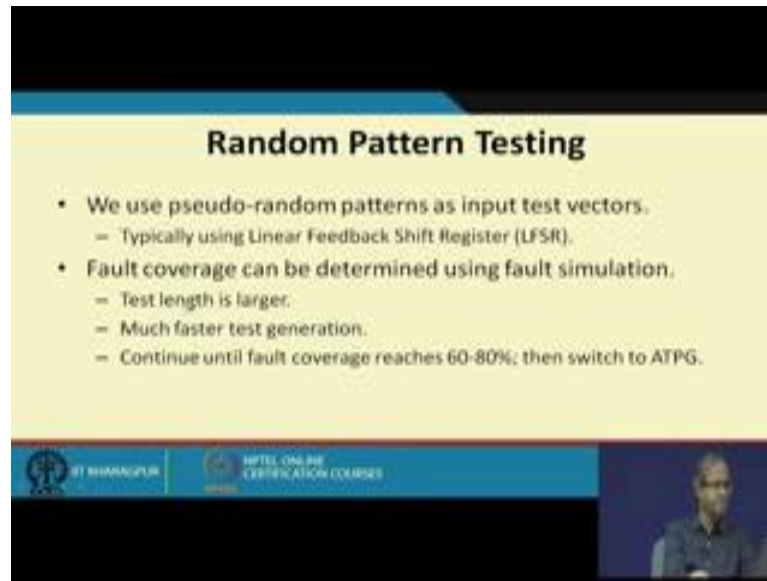
So, typically what is done this pattern generator is some kind of a hardware structure which can run very fast and generate some patterns which will be use for testing. So, this

multiplexor will be selecting whether I mean you are in the normal mode of operation where this primary input should go in or you are in the test mode where this pattern should go in. Similarly here in the normal mode the output of the circuit will go to the PO the primary inputs, but during the test mode you have something called a response compactor inside the chip. Let me just explain this, well I means again let us argue that I am applying 1000 test patterns to a circuit there are 10 output lines.

So, I will be generating 1000 bits of information and each of this 10 output lines corresponding to the 1000 test vectors. So, I will be having how many 1000 each there are 10 such line. So, it will be 10 kilobytes of data then you have 100 kilobytes of data sorry sorry 10, 10 lines in to 1000 yes 10 kilobytes of data. So, what you can argue is that well I will store these 10 kilobytes of expected response again in a ROM, and when the test is going on I will compare them bit by bit, but the here again I will say that was you can do it definitely, but your average will be become too much. If you are saying that you will be using a ROM to store your input vector to store your output response, you need ROM of a very large capacity large size; because for a modern this circuits, circuits let say you made a requiring millions of test vectors and in the circuit there can be 100 of outputs. So, the volume of test data that you need to store can be pretty huge ok.

So, you need something called a response compactor we shall come to this, that will you reduce the volume of the test data to a relatively small volume and you can use a small ROM to store the correct response for that small value after compaction. So, after compaction you compare with the compacted response with the corresponding golden value that is stored in the rom. So, if they match you say that your test has passed it is good, if it fails you say it is bad. But there are few things you see in this circuit there are few things which the BIST is not checking like connection from P I to the multiplex, it is only testing the pattern generation to circuit and circuit to response compactor, but this P I connection PO connection if there is any fault here this paths are not tested. So, this has to be remembered.

(Refer Slide Time: 12:14)



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.

At the bottom of the slide, there are logos for "IT MANAGER" and "INTEL ONLINE CERTIFICATION COURSE". A small video inset in the bottom right corner shows a man speaking.

Now, random pattern testing is a very important and interesting concept. So, far what we have said and seen we said that we have a circuit, we have a set of false let us try to find out a nice small set of false that can detect. We also said the test generation is complex and it takes time particularly for larger circuits. Now there is an alternative, here we say that we just forget test pattern generation, we do not use a test pattern generator at all let us use some kind of random patterns well, random patterns means they will be purely random patterns, but what you generate are actually pseudo random patterns.

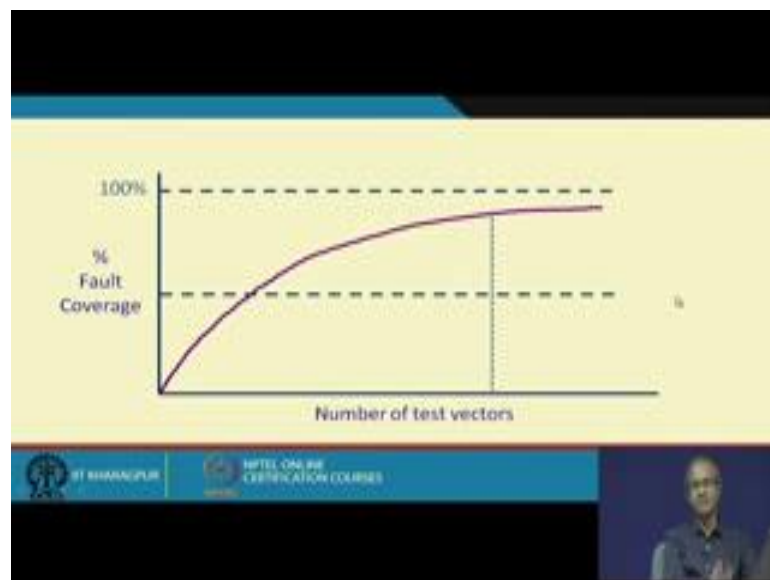
Pseudo random pattern means patterns which can be repeated in time, and this pseudo random patterns are typically generated using a hardware structure, which is called linear feedback shift register or LFSR. So, what we can do we can use an LFSR it is a very simple hardware structure or say we can generate 1000 random patterns and using fault simulation, we can find out that using those 1000 test patterns random patterns what is the fault coverage, how may percentage of faults getting detected. So, if you see that our fault coverage has reached an acceptable value, we can say that well let us use this many patterns I am giving example suppose we find that your desirable level is reached only after applying 1 million test patterns.

You see for a conventional test generation this 1 million was a very large number, because 1 million patterns have to be generated stores somewhere, in ATE we will have to load this pattern in ATE memory, but here I have a LFSR which will be simply

running for 1 million clock cycles, there is nothing to store anywhere right. And say means if I use a clock frequency of 1 gigahertz, then this 1 million pattern will take only 1 millisecond of time to have to operate right. So, here time is also not that much of an issue. So, you need to you need to seems or (Refer Time: 14:49) this also. So, the test length can be larger, but the test generation is very trivial and we can continue until either we reach a desired level of fault coverage.

But sometimes it is not applicable to BIST, we can see that beyond a point we are not able to detect false anymore then for the remaining falls you can switch to an ATPG tool, but for BIST application we do not do this we do pure random pattern testing usually.

(Refer Slide Time: 15:22)

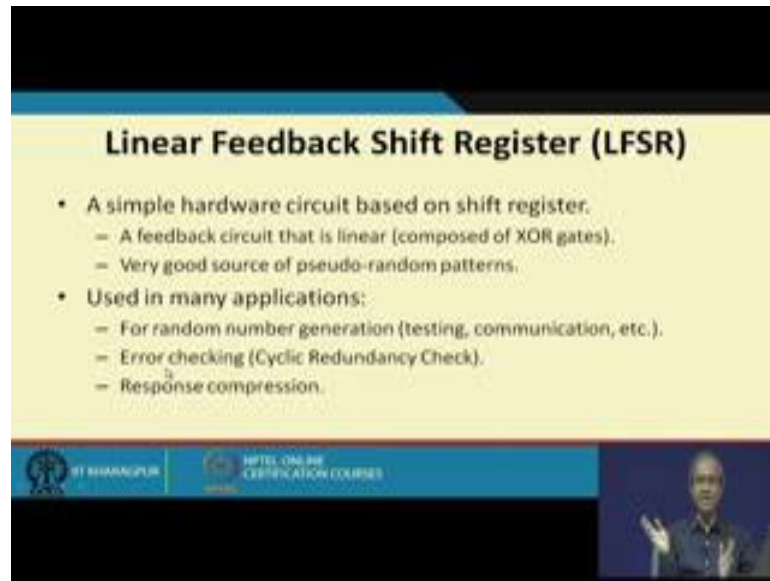


So, the behavior of typical circuits is like this. So, as I go and applying number of pseudo random test patterns, these are the number of test vectors and applying pseudo random, so through fall simulation I am continuously monitoring the fault coverage with typical curve is like this. So, it will go on rapidly increasing, but beyond the point it will now a less level of. So, if you can identify this point, then you can say that well this is my optimum number of test vectors I will apply so many.

Now, this number will depend vary from circuit to circuit. So for a given circuit, so this using for simulation we will have to find out the value, right.



(Refer Slide Time: 16:07)



**Linear Feedback Shift Register (LFSR)**

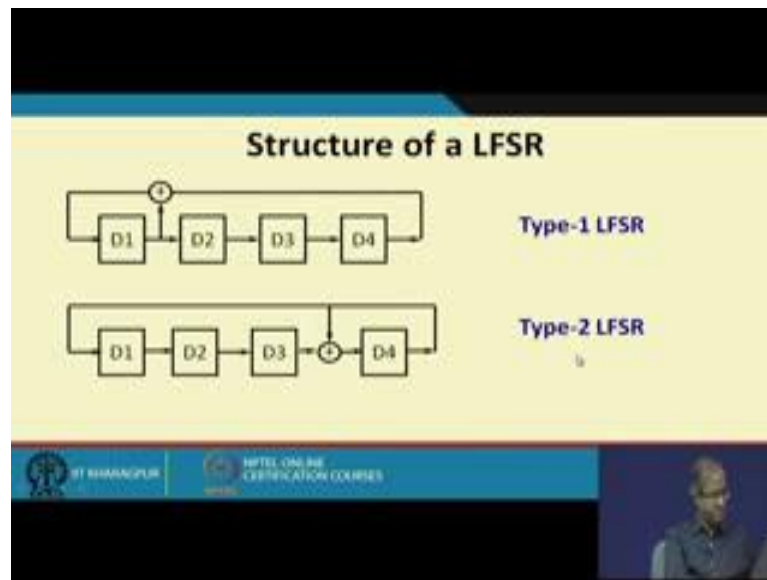
- A simple hardware circuit based on shift register.
  - A feedback circuit that is linear (composed of XOR gates).
  - Very good source of pseudo-random patterns.
- Used in many applications:
  - For random number generation (testing, communication, etc.).
  - Error checking (Cyclic Redundancy Check).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

IT MANAGER | INTEL ONLINE CERTIFICATION COURSES

Now, let us see, what is the linear feedback shift register; now LFSR is as then an implied it is some kind of a shift register, it is hardware circuit based on a shift register there is a feedback circuit also and the feedback circuit is linear feedback. Here we are not going in to the mathematical detail, this feed circuit consist of exclusive OR gate and XOR function can be proved to be a linear function that is why it is called linear feedback. And LFSR such been found to be very good source of pseudo random patterns, and has been used in many applications. Random number generation is of course, one application, but many of you may have heard of cyclic redundancy check which is used for error checking, for communication when you store some data on hard disks everywhere there can be errors.

CRC is a very common and popular way to detect errors and here again we use an LFSR to generate some kind of a CRC check some or signature. And we will see later that where we talk about the compaction of the responses, there also again we can use a LFSR this we will see later.

(Refer Slide Time: 17:37)



So, how does an LFSR look like? So, there are 2 alternate structures of an LFSR, type one or type 2. Type one is the classical structure where you can identify the feedback this is an XOR gate, you see D1, D2, D3, D4 or these are flip flops connected as a shift register these are D flip flops. So, the output of the last flip flop is fed back in to the input of this XOR, and some other output like here D1 is also fed as the input of XOR. So, here I have a choice I can use this I can take from D2, D2 D3 are all of them together and output of the shift register is fed as the input to the first flip flop.

So, you see this LFSR does not have an external input. So, once you load the initial value and apply the clock, it will go and running in an autonomous fashion it will be generating some patterns continuously. So, there is an alternate structure of LFSR which is used more commonly for data compaction, where instead of using XOR in the feedback circuit, we do like this we use the feedback directly, but we can use this XOR gates somewhere in between and whenever you use XOR gate you also take one of the inputs from this line. So, there can be an XOR gate here, so, one input will come from here, one input from here like this. So, this is called type-1 LFSR, type-2 LFSR.

(Refer Slide Time: 19:20)

The slide titled "Pattern Generation Example" illustrates a 4-bit Linear Feedback Shift Register (LFSR). The diagram shows four flip-flops labeled D1, D2, D3, and D4. The output of D4 is XORed with the output of D1, and the result is fed back into D1. The characteristic polynomial is given as  $f(x) = x^4 + x^1 + 1$ . To the right of the diagram is a truth table showing the sequence of bits generated by the LFSR over 16 clock cycles. The table has columns for D4, D3, D2, and D1, and rows for each clock cycle from 0 to 15.

	D4	D3	D2	D1
0	1	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	1	1	1
4	1	1	1	1
5	1	1	1	0
6	1	1	0	1
7	1	0	1	0
8	0	1	0	1
9	1	0	1	1
10	0	1	1	0
11	1	1	0	0
12	1	0	0	1
13	0	0	1	0
14	0	1	0	0
15	1	0	0	0

So, let us take the example of a generating a pattern using an LFSR. So, let us take type 1 LFSR like this, where I have 4 flip flops 4 bit LFSR the feedback connection is like this from D4 and from D1 the output of the XOR is fed to D 1. Now the behavior of an LFSR will depend quite a lot on the points from where you are taking the feedback connection. Now this feedback connection are the structure of this LFSR can be defined by something called the characteristic polynomial, like this particular for LFSR is represented or characterized by this polynomial, where does this mean? You see x to the power 4 x is a parameter, x to the power 4 represents this, x to the power 4 x to the power 3, x to the power 2 x to the power 1 and x to the power 0.

So, here we have a connection from x to the power 4 and x to the power 1 that is why these 2 terms are there, and 1 will also be there because you are connecting to x to the power 0. So, if instead of this there is a connection from here, then instead of x 2 a 1 it should be x to the power 2, x square if there is a connection form here it will have been x cube. So, depending on your typing points from where your connecting to the input of the exclusive or your characteristic polynomial will vary this is called characteristic polynomial.

Now, let us see the patterns which are generated this D1 should be here; D4, D3, D2, D 1. Suppose we are initializing the LFSR with 1 0 0 0, 1 0 0 0. So, as we apply clock there will be a shifting and a new bit will be getting in which will be the XOR of D4 and D 1.

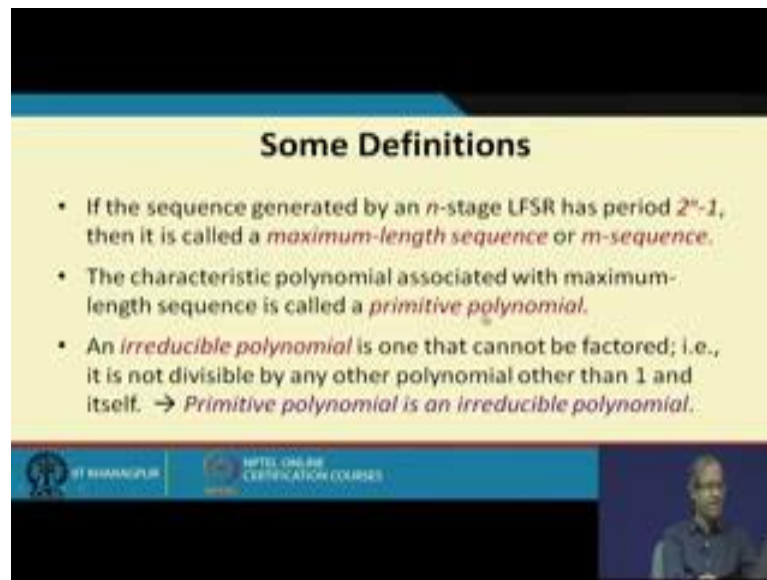
So, this is D1 and this D4 right. So, you go and shifting like this D4 and D 1, 0 and only 1. So, next time we shift it this 1 will get shifted in D 1, next 1 and 0 you shift again 1 will be shifted. So, 1 1 1 and 0 again 1, 1 will be shifted 1 1 1 again 1 0, again 1 will be shifted, but now 1 and 1 0 XOR is 0. So, now, 0 will be shifted 1 and 0 again 1 will be shifted like this.

You see as you go and applying clocks, you will see some patterns have been generated and after certain number this 1 0 0 0 is repeating. So, again after 1 0 0 this same pattern will come again. So, let us count the pattern 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 patterns have been generated. 15 patterns is generated before repeating. So, can you guess which pattern is missing here? The all 0 pattern is missing, you see for LFSR the all 0 pattern has very you can say dangerous implication, like once you load this LFSR with all 0 pattern it will remain in the all 0 state, because XOR of 0 and 0 will be 0, 0 will be shifted. So, right shift and 0 comes in. So, it will remain or 0.

So, whenever we are one thing to LFSR for a data generation application, we should never load it with all 0 and this is very good kind of LFSR we have chosen because we are able to generate all the other 15 patterns accepting 0. And if you look at the decimal equivalent of this numbers 8 1 0 0 0 is 8, 1, 3, 7, 15, 14, 12, 10, 5 then 11, 6 or so on these numbers look random. Now there are some standard tests for randomness. So, develop evaluated the patterns which I generated by LFSR with respect to randomness, and it is in found that the randomness quality is pretty good other than one test which fails, the other tests pass quite satisfactorily.

So, even if you want serial bit pattern you take the output of any of the flip flop, this is also random 0 1 1 1, and 0 1, 0 1 1, and 0 0 1, 1 0 1 this is also random pattern you can either take a parallel pattern as random or serially you can take one the output of any one of the flip flop right, but only one test as I said which is not satisfied with respect to random this is that, because the basic structure is a shift register you will see there is a pattern diagonally 0 0 0 0 these are getting shifted 1 1 1 1. So, these are getting shifted right. So, this is called cross correlation among the different bits, this is one property where it fails.

(Refer Slide Time: 25:08)



The slide is titled "Some Definitions" and contains three bullet points. The first bullet point states that if an  $n$ -stage LFSR has a period of  $2^n - 1$ , it is called a maximum-length sequence or  $m$ -sequence. The second bullet point states that the characteristic polynomial associated with a maximum-length sequence is called a primitive polynomial. The third bullet point states that an irreducible polynomial is one that cannot be factored; i.e., it is not divisible by any other polynomial other than 1 and itself. It concludes that a primitive polynomial is an irreducible polynomial. The slide also features logos for IIT BHARANGPUR and NPTEL ONLINE CERTIFICATION COURSES, and a small video inset of a speaker in the bottom right corner.

### 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*.

Now, some definitions of LFSR; now we have taken in this example 4 bit LFSR and we had seen that we have generated 15 patterns. So, in general for an  $n$  bit or  $n$ -stage LFSR, so we can have at most  $2^n - 1$  patterns generated. Such a sequence generated by an LFSR, where all the patterns other than the all 0 pattern are getting generated that is called maximum-length sequence or  $m$ -sequence. Now the characteristic polynomial you like in this LFSR the tapping points they define the characteristic polynomial. So, the characteristic polynomial which generates an  $m$  sequence is called a primitive polynomial.

Now, for this example this was primitive polynomial that is why 15 patterns were generated. So, this, another definition that means, irreducible polynomial is a polynomial which cannot be factored, now primitive polynomial is a type of irreducible polynomial. So, there are ways to identify this kind of primitive polynomials.

(Refer Slide Time: 26:33)

**Some Example Primitive polynomials**

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.*

NPTEL ONLINE CERTIFICATION COURSES

There are comprehensive lists of these primitive polynomials available in various places. so I am showing you a few. These number indicates the number of flip flops; that means, number of stages in the LFSR, and these is kind of a coding from where you are taking the tap, 3 1 0 means you are taking the tap of course, from the last stage 3 will be there and from 1 and 0. So, the polynomial will be  $x^3 + x + 1$ , for 4;  $x^4 + x + 1$  similarly for 16 this 32 this 64. So, you see. So, if you take 32 bit LFSR with a primitive polynomial like this; that means, there will be these many tapping points, then you can have a circuit which can generate  $2^{32} - 1$  pattern; that means, 4 billion patterns. We have a very convenient circuit which can generate 4 billion random patterns ok so it is a big advantage.

(Refer Slide Time: 27:41)

**Some Interesting Properties of m-sequence**

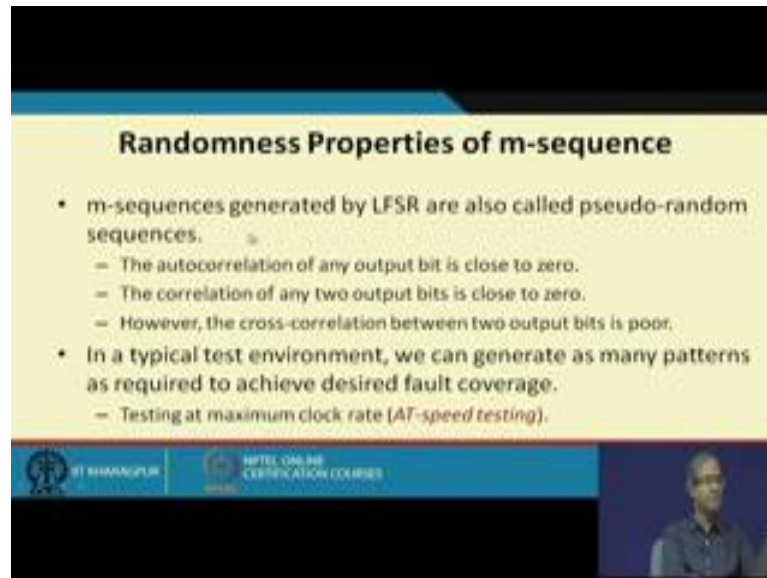
1. The period of  $\{a_n\}$  is  $p = 2^n - 1$ , that is,  $a_{p+i} = a_i$  for all  $i \geq 0$ .
2. Starting from any nonzero state, the LFSR that generates  $\{a_n\}$  goes through all  $2^n - 1$  states before repeating.
3. The number of 1's differs from the number of 0's by one.
4. If a window of 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 BHARANGPUR NPTEL ONLINE CERTIFICATION COURSES

So, this m sequences also have some interesting properties like as I said firstly, the period before it repeats is 2 to the power n minus 1; that means, the bit  $a_{p+i}$  will be equal to  $a_i$  for  $p$  equal to  $n$  minus 1, it repeats after  $p$ . Starting from any non 0 state, the LFSR will go through all the other 2 to the power n minus 1 states before repeating this we have already seen. Number of ones differ from number of zeros by 1. So, be if you look at the output column for any of the bits, see for a normal truth table where you have all 2 to the power n, number of zeros are once are equal. Now here we do not have the all 0 pattern, so now, zeros will be 1 less right? So, number of ones will exceed number of zeros by 1, and here there is some other properties also which are not very important in the present the context.

Like if you slide a window of width  $n$  across sequence of bits generated, then each of the possible binary patterns will be seen exactly once in a period, like what I mean is that like you look at this if you slide window of width 4 like a 0 1 1 1, then 1 1 1 1, then 1 1 1 0, 1 1 0 1, 1 0 1 0. So, these will be appearing exactly once and uniquely.

(Refer Slide Time: 29:28)



**Randomness Properties of m-sequence**

- 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*).

IT MANAGER | INTEL ONLINE CERTIFICATION COURSES

Now, of course, there is some other property where it runs, so here we are not wanting it. Now randomness properties of m-sequence is something which you are interested about here, because m sequences have very good pseudo - randomness properties; the auto correlation cross co relation is 0, but cross co relation between 2 output bits 2 consecutive output bits is poor as you have said because of the shift register connection.

So, in a typical test environment like BIST, we can generate as many patterns we required. If I say that I need 1 billion patterns fine I can do that. So, I means with a 1 gigahertz clock I need only 1 7 to apply the test, and the good thing is that we are able to test that the maximum clock speed this is called at speed testing. Where some faults which also arise because of delays in the circuit the critical timings they will also get detected. In scan designs we often we are not detecting such kind of falls.

So, with this we come to the end of this lecture in our next lecture, we shall be looking at the other part of BIST. So, we have seen the test generation part, but now after we have obtain the responses how do I compact and compare that part will be seeing in the next lecture.

Thank you.