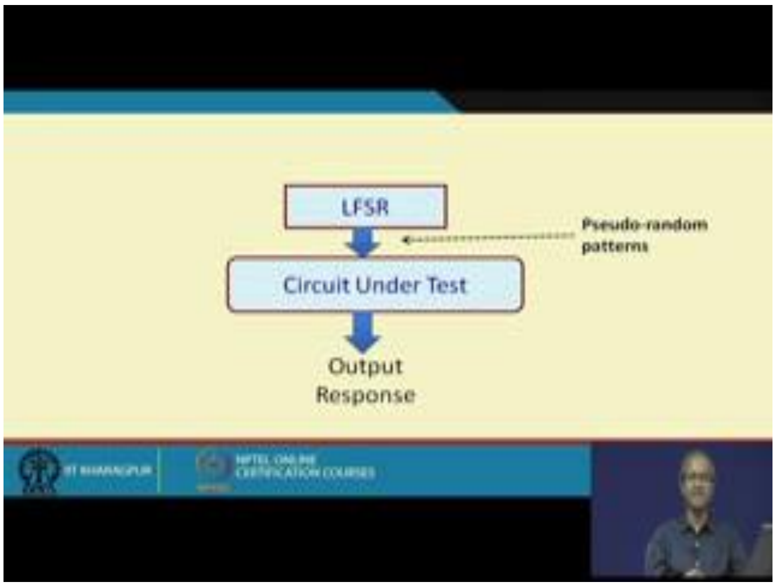
VLSI Physical Design
Prof. Indranil Sengupta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 57
Built-in Self-Test (Part 2)

So, now we will be continuing with our discussion on BIST. So, in our last lecture if you recall we were talking about pseudo random pattern generation, and how we can use LFSR to generate the patterns for building such type of a circuits. Now we have also said that LFSR test patterns are very cheap and inexpensive; in the sense that the hardware required for the test generation if very simple they can run at the maximum rated frequency of the chip, so that you can apply the clock frequency at the rated you can apply the patterns at the rated clock frequency, which can also help in detecting many delay related falls.

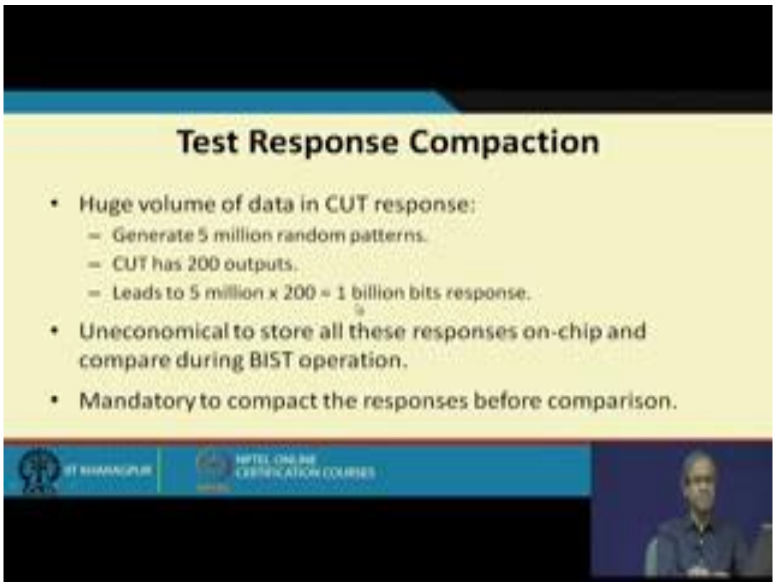
(Refer Slide Time: 01:06)



So, today we continue our discussion and look at the remaining part of the BIST architecture. We have so far seen that how a circuit under test can be excited with pseudo random patterns from a linear feedback shift register. But suppose I have a circuit with 100 inputs, and I am applying let us say 1 million or 1 billion patterns. So, I will be having 1 billion sets of output responses here what do I do with those or 1 million such set. So, we really cannot afford to store all the responses in a ROM some kind of a memory, and compare them vector by vector it will be too expensive, right.

So, what is done is that, using some method you have to reduce the size of these test responses to a small and manageable volume before we can actually compare with the corresponding fault free value right. So, that is the objective here.

(Refer Slide Time: 02:12)



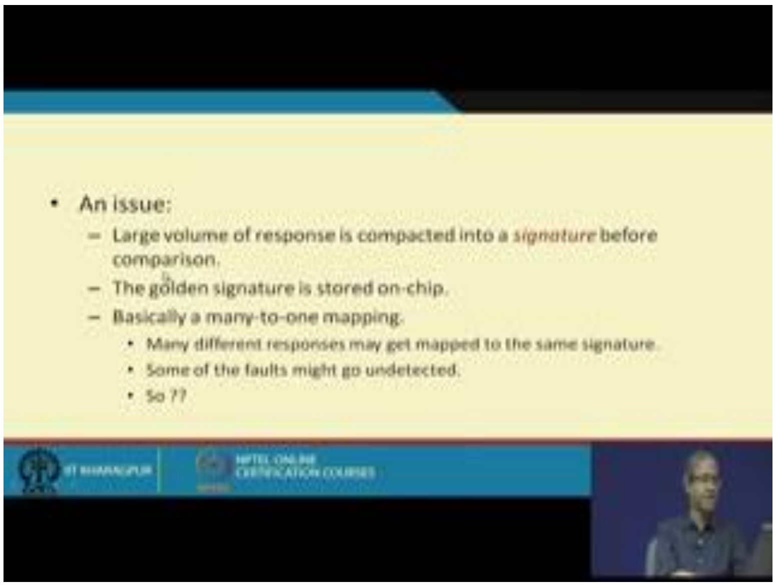
Test Response Compaction

- Huge volume of data in CUT response:
  - Generate 5 million random patterns.
  - CUT has 200 outputs.
  - Leads to 5 million x 200 = 1 billion bits response.
- Uneconomical to store all these responses on-chip and compare during BIST operation.
- Mandatory to compact the responses before comparison.

So, we talk about something called test response compaction. Now we shall see a little later how compaction is different from compression. Compaction and compression are not the same thing. So, I have just now said that the circuit under test is when you are applying the test data, the responses can generate huge volumes of data like one example 5 million patterns are applied, there are 200 outputs. So, you are lending into 1 billion bits of response. So, you really cannot store so many bits (Refer Time: 02:57) in ROM on chip, and compare during testing right. So, it is absolutely mandatory to do some kind of a compaction to reduce these billion bits well.

Now you can you may ask well I have to start with this 1 billion bits, well I have to compact I agree, but compact to how much should it be 1 billion to 1 million, 10000, 1000 what will be a good number? We shall see using some theoretical justification, that practically compaction is done in such a way that the final value that is generated is just a 32 bit or 64 bit number.

(Refer Slide Time: 03:53)

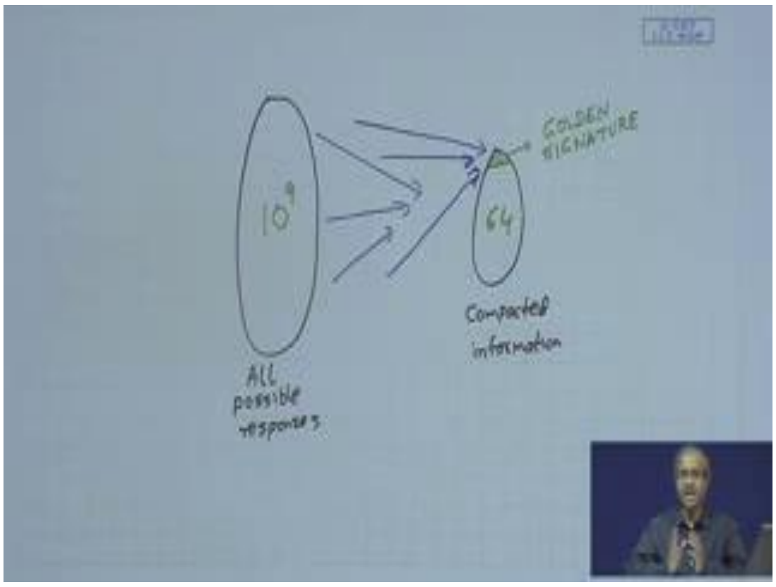


- An issue:
  - Large volume of response is compacted into a *signature* before comparison.
  - The golden signature is stored on-chip.
  - Basically a many-to-one mapping.
    - Many different responses may get mapped to the same signature.
    - Some of the faults might go undetected.
    - So ??

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

So, from 1 billion we come down to just 32 bits or 64 bits. So, it is a huge compaction this will see. So, there is one issue that arises here. So, we are doing compaction.

(Refer Slide Time: 04:08)

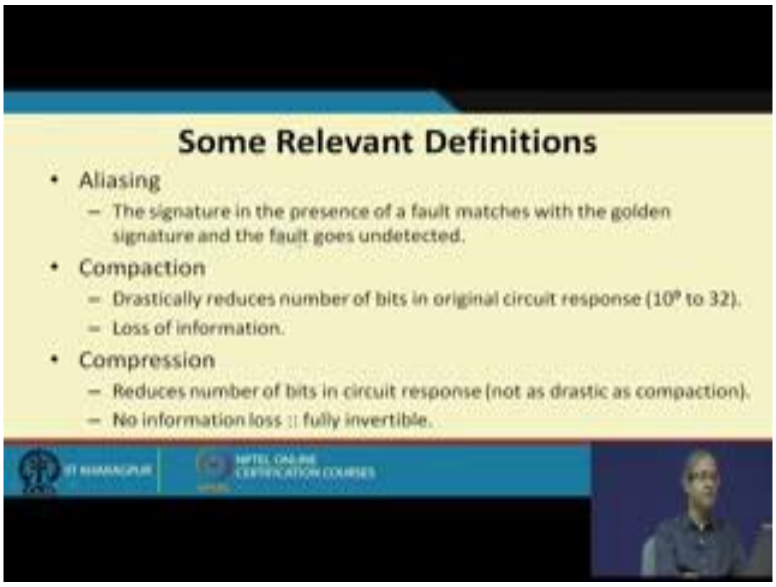


So, compaction means what, we are doing some kind of many to one mapping, like we are mapping a large set into a small set. Let us say this is the set of all possible responses, and this is the set of compacted information. So, the example that I have just now mentioned I said that we are possibly compacting 1 billion bits of data into let us say 32 or 64 bits of data, it is a huge compaction.

Now what I do I find out that for my circuit without any fall, what should be my good compacted information that I say this is my golden signature. This is my golden signature. So, after my test experiment is done, I compare my compacted information against my golden signature. But the problem here is because I am doing a very kind of heavy kind of many to one kind of a mapping, 10 to the power 9 things are getting mapped into 64 things. So, even many possible responses might get mapped into the same golden signature. So, some errors might get mapped into the golden signature and those faults will (Refer Time: 05:53) they will go undetected, yes we except that because we are doing compaction some faults will go undetected but what is the probability of that we shall evaluate that later on.

So, this is what we are saying the golden signature is stored on chip many to one mapping, some of the falls might go undetected. So, we will have to look at it that what is the probability for that.

(Refer Slide Time: 06:22)



Some Relevant Definitions

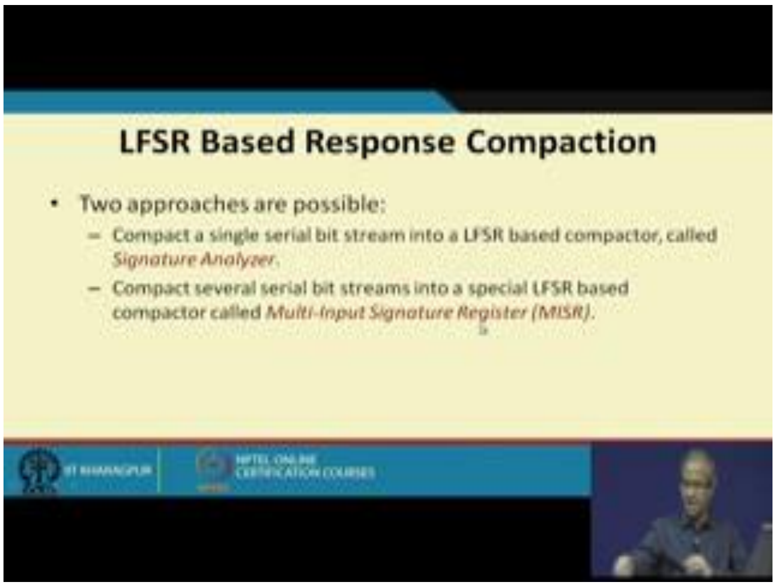
- Aliasing
  - The signature in the presence of a fault matches with the golden signature and the fault goes undetected.
- Compaction
  - Drastically reduces number of bits in original circuit response (10^9 to 32).
  - Loss of information.
- Compression
  - Reduces number of bits in circuit response (not as drastic as compaction).
  - No information loss :: fully invertible.

So, we first look at some relevant definitions; this aliasing is something we was just now talking about. This aliasing is the phenomena, that the signature in the presence of a fault matches with the golden signature and therefore, the fault goes undetected.

So, I have a circuit, I apply those 1 billion patterns, I get the 64 bits signature, I compare it with my good signature I say that there is a match. But it may so happen that there was some fault in the circuit, but somehow the final signature remained the same, because it is a many to one mapping. Now let us differentiate between compaction and compression; compaction is a mechanism which drastically reduces the number of bits like we mentioned 10 to the power 9 to 32 or 64, this results in loss of information quite obviously, because from this 32 bits we cannot go back and recover this 10 to the power 9 original responses, this is a one way mapping.

But in compression which we normally use in our computers we do zip, we do rare to compress our files that is a reversible process. We do compression; we can again do a un compression to get back to our original. So, compression also reduces number of bits, but not this drastically, but there is no information loss we can reverse the process; there is a process of compression, there is a process of decompression or un-compression.

(Refer Slide Time: 08:09)

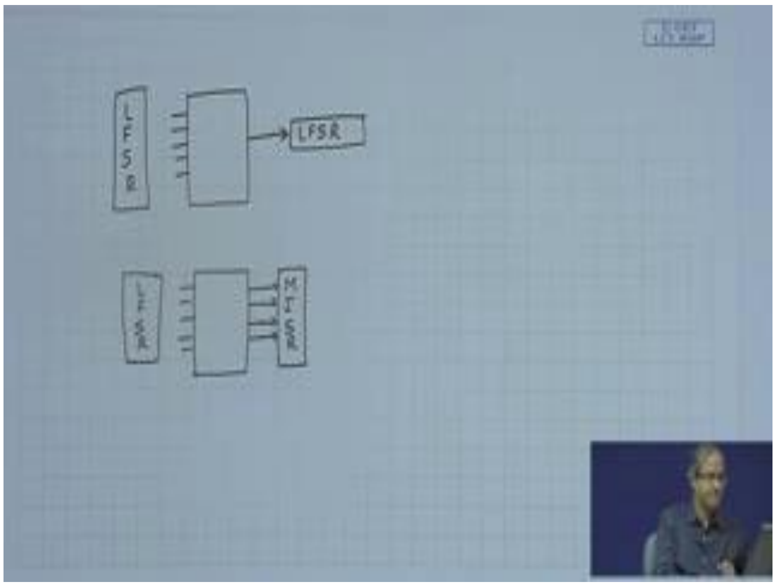


LFSR Based Response Compaction

- Two approaches are possible:
  - Compact a single serial bit stream into a LFSR based compactor, called *Signature Analyzer*.
  - Compact several serial bit streams into a special LFSR based compactor called *Multi-Input Signature Register (MISR)*.

Now, when I talk about response compaction and here again we are using LFSR. So, 2 broad approaches are possible, one where we compact a single serial bit stream into a compactor this is called signature analysis, and the hardware is called signature analyzer, and as an alternative several serial bit streams can be compacted like it is called multi input signature register like I am just showing you schematically.

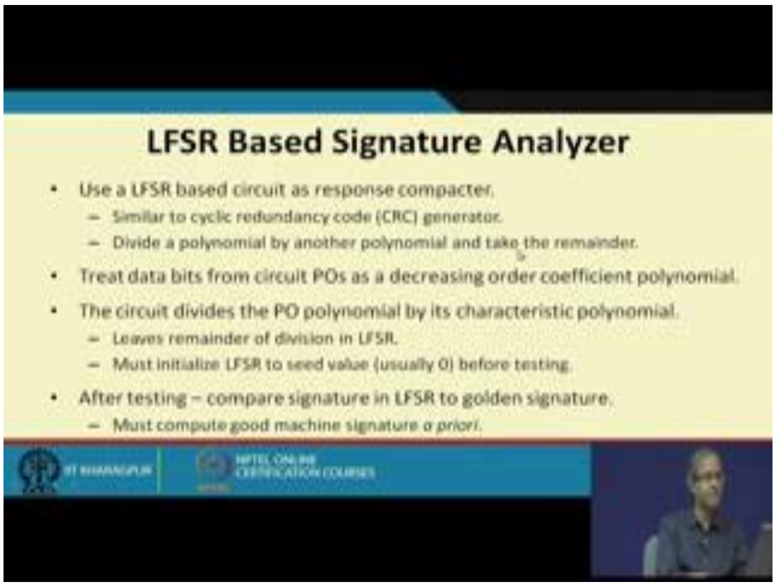
(Refer Slide Time: 08:47)



Like suppose you have a circuit there are multiple inputs, but a single output let us say. So, what I can do at the input side I can use an LFSR to generate some random patterns and feed it here, and on the output side here again I use a special kind of LFSR, which will accept serial data from here and it will do the compaction here.

But if I have a circuit like this, well let us say there are multiple inputs and also multiple outputs. So, here I can use an LFSR, and here I can use one such LFSR with too with every output, but that will be an expensive solution, but rather what I use I use a special kind of an LFSR which can take multiple inputs, multiple input signature resistors. So, I compact multiple output data simultaneously, this is what the main difference here.

(Refer Slide Time: 09:55)

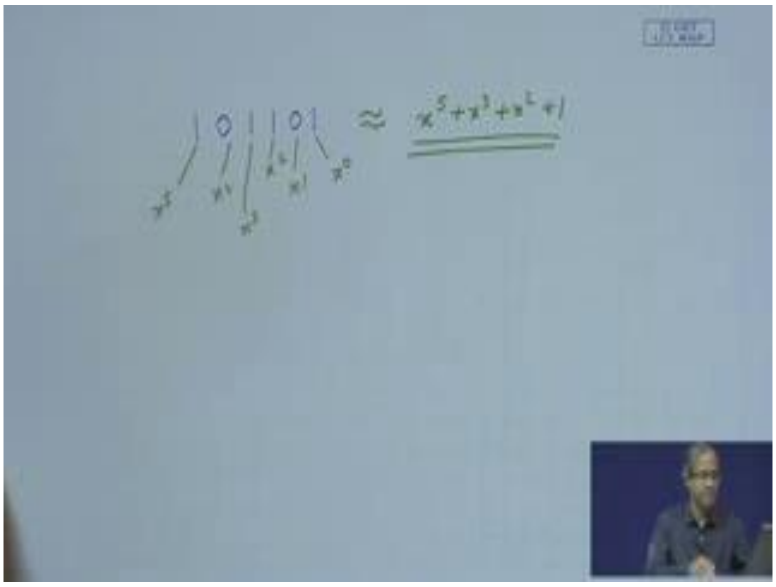


LFSR Based Signature Analyzer

- Use a LFSR based circuit as response compacter.
  - Similar to cyclic redundancy code (CRC) generator.
  - Divide a polynomial by another polynomial and take the remainder.
- Treat data bits from circuit POs as a decreasing order coefficient polynomial.
- The circuit divides the PO polynomial by its characteristic polynomial.
  - Leaves remainder of division in LFSR.
  - Must initialize LFSR to seed value (usually 0) before testing.
- After testing – compare signature in LFSR to golden signature.
  - Must compute good machine signature *a priori*.

So, let us first look at the LFSR based signature analyzer which can compact a single bit stream. So, the process is exactly similar to cyclic redundancy code generator, which is used for error detection in many applications. Mathematically what we do here is basically we are dividing a polynomial by another polynomial and take the reminder.

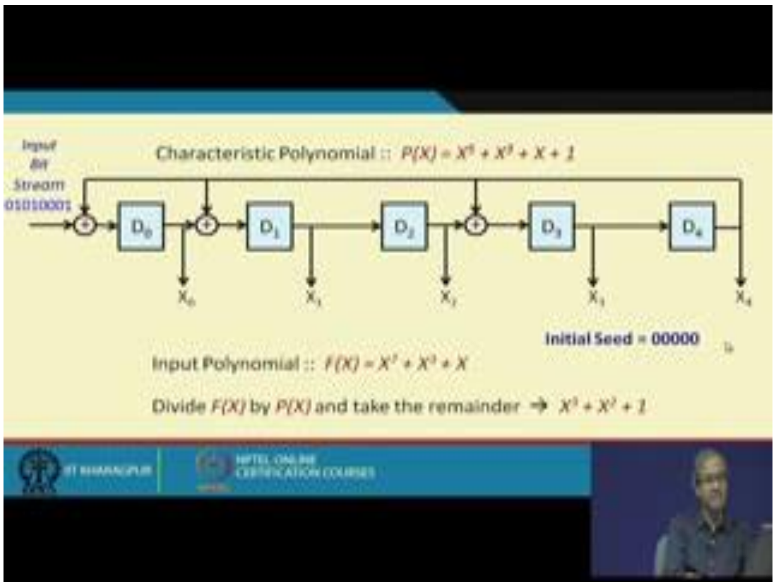
(Refer Slide Time: 10:30)



Now, you see any bit pattern; let us say I write a bit pattern 1 0 1 1 0 1. Now bit pattern can be also be represented as a polynomial like this, that every bit position can be regarded as the coefficient of a polynomial x to the power 0, x to the power 1, x to the power 2, x to the power 3, x to the power 4, and x to the power 5. So, this bit pattern we can express this as a polynomial just take the ones, x to the power 5, plus x to the power 3, plus x square plus 1.

So, any binary bit stream can be represented as a polynomial. So, here you are saying is that the process is essentially one of dividing a polynomial by another polynomial and take the reminder that will be the cyclic redundancy code and in this case our signature. So, we treat the data bits that are coming out of the outputs of the circuit as a decreasing order coefficient polynomial just as I had mentioned; that any bit stream can be regarded as a polynomial. So, essentially what this method does. So, I have an input polynomial which is the output bit stream which the circuit generates, we divide it by the characteristic polynomial of the LFSR and after all the bit patterns are fed whatever remains in the LFSR that is the signature that is the reminder after division, and that reminder we are calling as this signature.

(Refer Slide Time: 12:36)



Characteristic Polynomial :: $P(X) = X^5 + X^3 + X + 1$

Input Polynomial :: $F(X) = X^7 + X^3 + X$

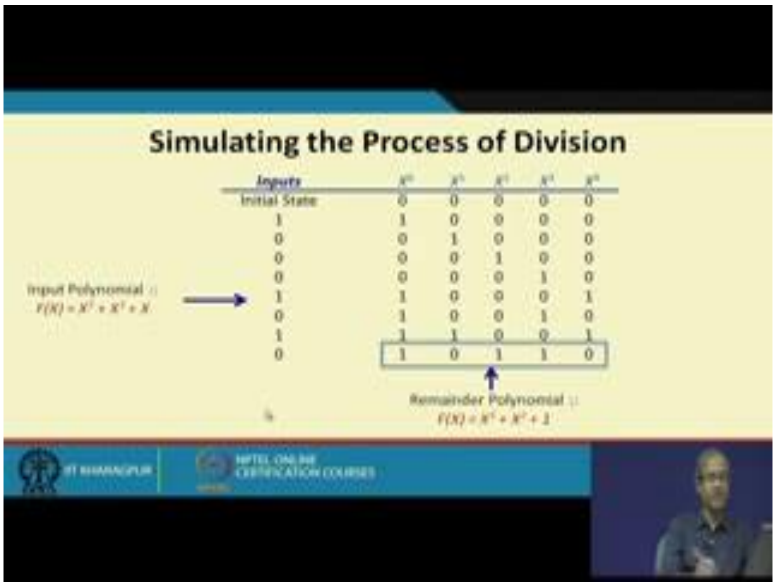
Divide $F(X)$ by $P(X)$ and take the remainder $\rightarrow$ $X^3 + X^2 + 1$

So, we are finally, comparing that reminder with the good reminder, in presence of means in the absence of faults for the good circuit. So, the good machine signature must be computed a priori and stored in a ROM, so that after the experiment we can compare right. So, let us take an example here. So, we have a 5 bit LFSR as shown here. So, characteristic polynomial if you look at the coefficience here. So, it will be like this 1, x, x cube, and x 5 and the input bit stream let us suppose bits are coming like this.

So, input polynomial if you look at it 0 1 2 3 4 5 6 it will be x to the power 7, plus x to the power 3 plus x to the power 1. So, input polynomial can be regarded as x 7, x cube, plus x. So, we have to divide this polynomial by this polynomial, and these are the outputs of the LFSR let us see, but here we do not need the outputs really, because here we are doing some kind of compression compaction; so we are dividing F X by P X and take the remainder. So, if you do it by hand modulo 2 division, you can check that the remainder is coming as X cube, plus X square, plus 1.

Let us see, but whether this hardware also generates the same thing. So, we load it with the initial seed like this.

(Refer Slide Time: 14:00)



Simulating the Process of Division

Inputs	X^0	X^1	X^2	X^3	X^4
Initial State	0	0	0	0	0
1	1	0	0	0	0
0	0	1	0	0	0
0	0	0	1	0	0
0	0	0	0	1	0
1	1	0	0	0	1
0	1	0	0	1	0
1	1	1	0	0	1
0	1	0	1	1	0

Input Polynomial :: $F(X) = X^7 + X^3 + X$

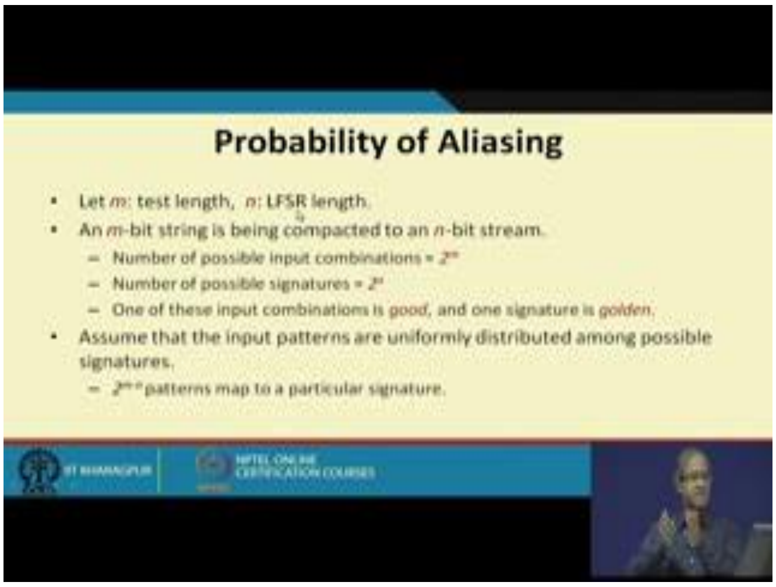
Remainder Polynomial :: $F(X) = X^3 + X^2 + 1$

So, we simulate the process of division step by step will load, the point to notice is that initially we load it with the all 0 pattern all 0, then we apply this sequence of bits 1 0 0 0 1 to the LFSR, and depending on the feedback some of the bits will get modified accordingly. So, I suggest that you can work this out, and see whether these patterns are matching. So, after all the bit patterns have been applied, the data that remains in the chip register is 1 0 1 1 0, this bits correspond to the input polynomial X 7 plus X 3 plus X 1 and the remainder that is coming that is X 3 plus X 2 plus X 1 which is supposed to be the remainder.

So, what we have shown with the simple example is that this special kind of LFSR structure can really divide a polynomial by another polynomial, and compute the remainder.

This concept is used in CRC code generation, and in the present context we are use it we are using it for signature generation. So, for detecting whether the faults are taking place or not fine.

(Refer Slide Time: 15:22)

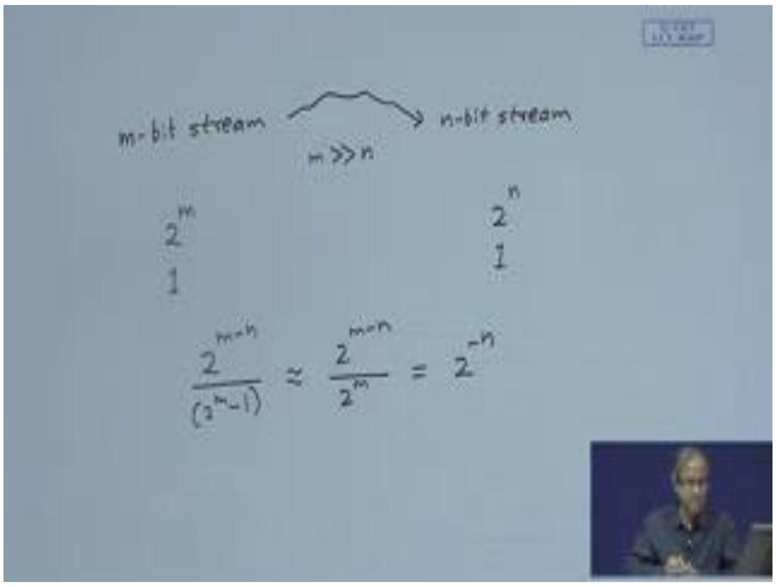


Probability of Aliasing

- Let m: test length, n: LFSR length.
- An m-bit string is being compacted to an n-bit stream.
  - Number of possible input combinations = 2^m
  - Number of possible signatures = 2^n
  - One of these input combinations is *good*, and one signature is *golden*.
- Assume that the input patterns are uniformly distributed among possible signatures.
  - 2^{m-n} patterns map to a particular signature.

Now, comes the important part. So, what is the probability that a fault has occurred, but we are not able to detect it, because it has matched to the correct signature. So, let us make a simple calculation based on some assumptions. So, we assume that m is the number of bits that you are compressing or compacting and n is the LFSR length so; obviously, m will be large as compared to n, the m is much larger.

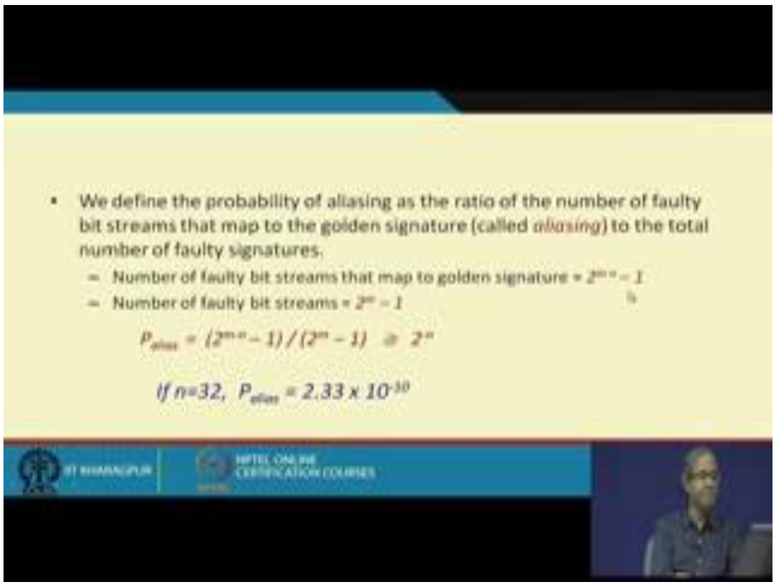
(Refer Slide Time: 16:06)


m-bit stream
n-bit stream
$m >> n$
2^m
1
2^n
1
$\frac{2^{m-n}}{(2^m-1)} \approx \frac{2^{m-n}}{2^m} = 2^{-n}$

So, we are means actually we are doing something like this, we have an m bit stream this is coming from the output of your circuit, you are mapping it to an n bit stream in the LFSR. So, as I said m is much greater as compared to n. Now what means if I talk about m bit stream, we revert it like this because of some error or fault in the circuit this m bit stream will become something different; so what are the possible m bit streams? There are 2 to the power m possible m bit streams, similarly there are 2 to the power n possible n bit streams, and out of them one is the good stream, similarly here one is the good signature this is what we are telling in this slide.

So, we are saying number of possible input combinations 2 to the power m, possible signature 2 to the power n, out of these one is good and one of these signature is the good or the golden signature. So, we make an assumption that this input 2 to the power m patterns are uniformly distributed among the signatures. So, there are 2 to the power m patterns which are matching into 2 to the power n. So, for any particular signature how many patterns will be mapping? 2 to the power m divided by 2 to the power n, or 2 to the power n minus n. So, these many patterns will also be mapping into the golden signature ok.

(Refer Slide Time: 17:46)



- We define the probability of aliasing as the ratio of the number of faulty bit streams that map to the golden signature (called *aliasing*) to the total number of faulty signatures.
  - Number of faulty bit streams that map to golden signature = $2^{m-n} - 1$
  - Number of faulty bit streams = $2^{m} - 1$

$$P_{alias} = (2^{m-n} - 1) / (2^{m} - 1) \cong 2^{-n}$$

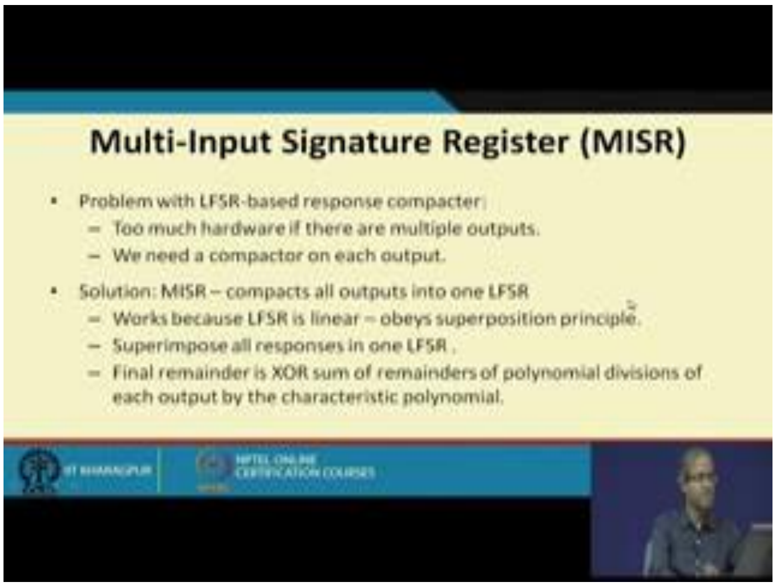
$$If\ n=32,\ P_{alias} = 2.33 \times 10^{-10}$$

So, we define the probability of analyzing as the ratio of the number of faulty bit streams that map to the golden signature; that means, the faults that are going undetected to the total number of faulty signatures.

Now, I said there are 2 to the power m minus n patterns which are mapping into a signature also the golden signature out of them one is good. So, number of faulty signature will be this minus 1 that map into the golden signature. Similarly 2 to the power m is the total input bit streams out of them one is good. So, 2 to the power m minus 1 will be faulty. So, probability of aliasing will be this divided by this. Now because m is much greater than n, this approximates to 2 to the power minus n which is independent of m. Just you can check 2 to the power m minus n, divide by 2 to the power m minus 1. So, we are saying m is much greater than n. So, you can approximate this as 2 to the power m minus n, you can ignore this 1 divide by 2 to the power m, this is 2 to the power minus n.

So, let us take an example of 32 bits LFSR, if n is 32. So, here is. So, how much is 2 to the power minus n? 2.3 into 10 to the power minus 10. So, this is the probability with which some fault may occur, but it will go undetected. But 10 to the power minus 10 is really a very small number, if we use a 64 bit signature 2 to the power minus 64 will be even much smaller. So, these numbers give us a confidence that even if there is a chance of some fault getting undetected, the probability is very very low and this method is pretty good and accepted right this is the basic idea.

(Refer Slide Time: 19:56)

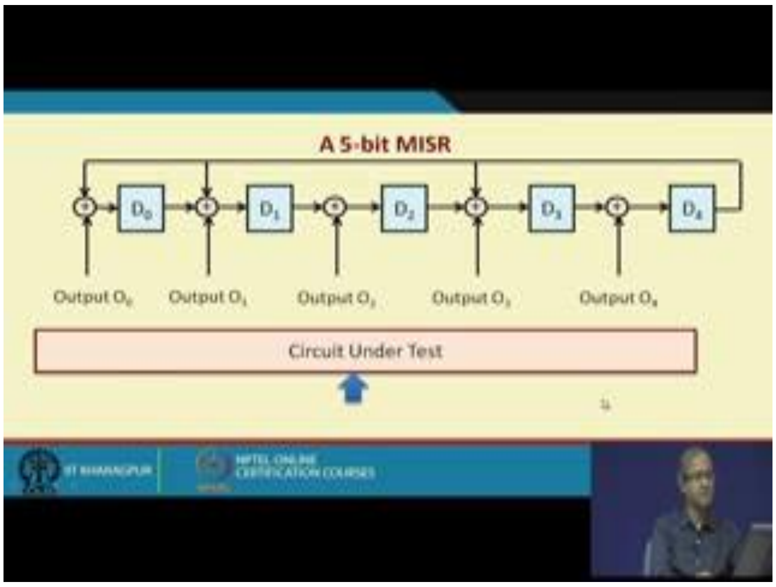


Multi-Input Signature Register (MISR)

- Problem with LFSR-based response compacter:
  - Too much hardware if there are multiple outputs.
  - We need a compactor on each output.
- Solution: MISR – compacts all outputs into one LFSR
  - Works because LFSR is linear – obeys superposition principle.
  - Superimpose all responses in one LFSR .
  - Final remainder is XOR sum of remainders of polynomial divisions of each output by the characteristic polynomial.

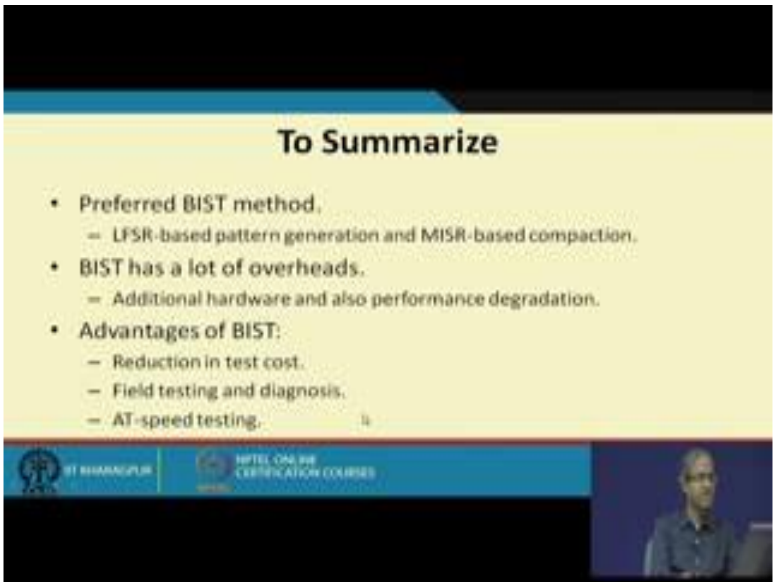
So, multi input signature register is just an extension of this concept, because the method that we have discussed is able to compress only a single bit stream. So, if there are multiple bit stream, multiple outputs, you need one such LFSR for every output. So, you go for something called multiple input signature register which is called MISR in short MISR, here we compact all the outputs into one LFSR together. There are some properties. So, means if you compact each of the individual inputs separately, compute the signature and take the XOR of the final, you will be getting the final result at the end so this can be proved. Because this LFSR feedback is linear, it obeys superposition principle; and superposition principle says that if we apply one input at a time, and see the response and if we apply all of them together if you take the sum it will be the same response. So, this happens.

(Refer Slide Time: 21:08)



An MISR looks like this. So, MISR is a modified version of a type 2 LFSR you can see, where we have put an XOR gate between every pair of flip flops. But depending on the characteristic polynomial, I can use this feedback point to some of the XOR gates only maybe I put it here, here and here, but these 2 there is no connection. But on the other side we have added one additional input of this XOR gates, when I have a circuit under test I am applying some patterns, there can be multiple outputs which are generating simultaneously I feed them simultaneously on these lines and this is how compaction takes place right.

(Refer Slide Time: 21:59)



To Summarize

- Preferred BIST method.
  - LFSR-based pattern generation and MISR-based compaction.
- BIST has a lot of overheads.
  - Additional hardware and also performance degradation.
- Advantages of BIST:
  - Reduction in test cost.
  - Field testing and diagnosis.
  - AT-speed testing.

So, to summarize we looked at the various technologies, some of the technologies actually there are there are some others also the preferred BIST method we have seen; that LFSR based pattern generation and MISR based compaction. There are overheads of course; you need the pattern generator, response compaction, control circuits compotator etcetera. But the advantages of BIST are also quite high so that in many applications, many chip designs you may prefer to go for BIST, because it gives you much higher convenience. Reduction in test cost obviously, you can do testing and diagnosis on field and most importantly at speed testing, you can test at the clock frequency of the circuit.

So, with this we come to the end of this series of lectures on testing.

Thank you.