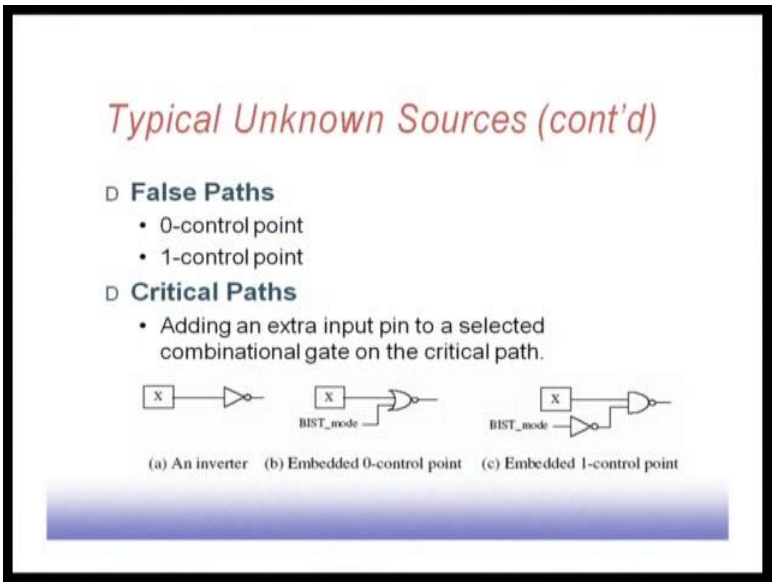
Digital VLSI Testing
Prof. Santanu Chattopadhyay
Department of Electronics and EC Engineering
Indian Institute of Technology, Kharagpur

Lecture – 25
Logic BIST (Contd.)

Some more unknown sources are false paths. So, false paths they cannot be excited because of the logic of the circuit is such that so they cannot be excited. So, what we can do if there is a false path then we can put a 0-control points so that this path is force to 0 or we can have 1-control point.

(Refer Slide Time: 00:21)

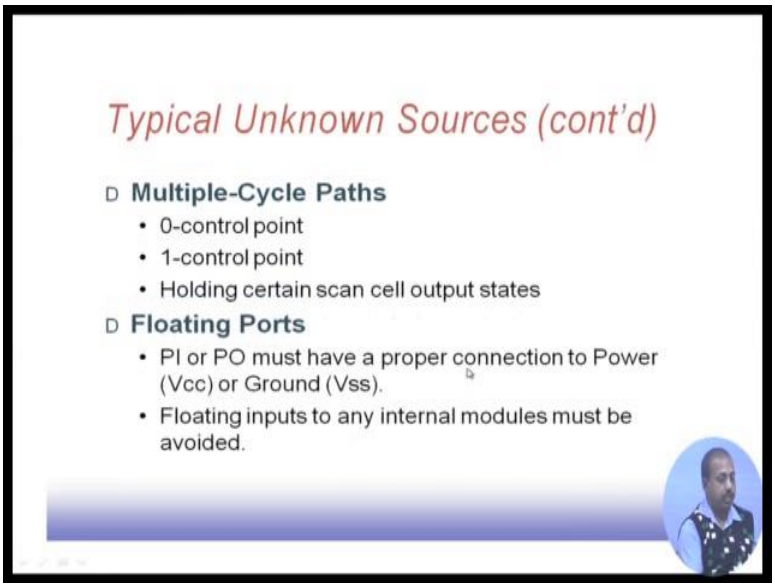


So, whatever we do whatever is the requirement based on our analysis, so we will be putting it one of putting it to one of those control points. So, that the false path gets a excited. So, it gets a value of one or 0 as desired then there are critical paths. So, critical path the problem is that adding an extra input pin to a selected combination gate on the critical path. So, this is a one inverter, this inverter gets input from X this is on a critical path so this is the; in this case this is the inverter is there now if this inverter is there then one possibility is we do not want to increase the delay that occurs in the operation of the circuit.

So, what we do we put a NOR gate and the BIST mode control can see here. So, this way we can introduce an embedded 0 control points. So, if you want to make this X as 0 at

this point always due to the creation or due to the removal of unknown then we have to do it like this. So, you have to embed a 0-control point in this fashion. On the other hand if we required that in this point should be 1 after this, in the BIST session this point should be 1, so for that purpose we have to embed 1-control point. So, in that case, in the BIST mode, it as to be inverted and it goes to a NAND gate. You may notice that we are not increasing the delay, like delay of this path was one gate. So, here also it is one gate here also it is one gate. So, for critical paths, we had one extra input pin to a selected combination and logic on the critical path.

(Refer Slide Time: 02:20)

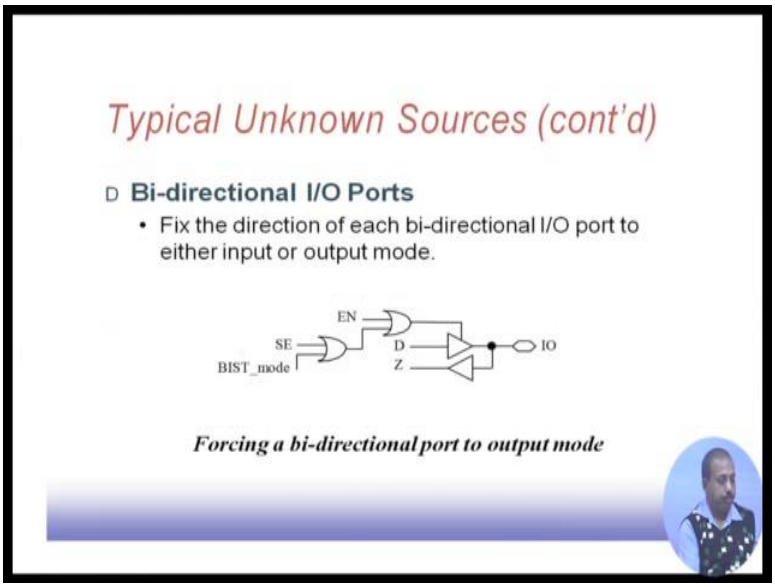


Now, there may be multiple cycle paths. So, they are that is the data is available after multiple cycles. So, may be that delay of the system is such that the combinational logic does not evaluate in one short, so that way if we have got multiple cycle paths. So, for the circuit requires multiple cycles to get stabilized, we have got, we can introduce on 0-control point 1-control point or we can hold certain scan cell output states. So, we can hold the scan cell output. So, that they ah do not change their value if the during this multi cycle operation. So, that is another way of stopping this, so unknown sources.

Then floating ports, so primary input on primary output may have a proper connection to power Vcc of the ground, this primary inputs or floating port some port which is floating, we can we can put these input primary input or primary output to some proper values floating inputs to any internal modules must be avoided. So, we should not have any

floating input for internal gate, internal portion, but for the primary input and primary output, we can connect them either to power supply or Vcc or the ground in the BIST mode.

(Refer Slide Time: 03:44)



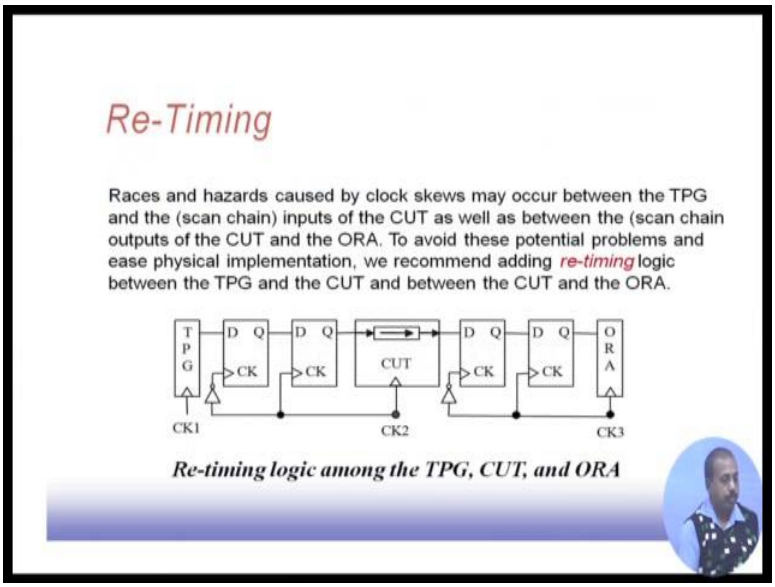
Bidirectional IO ports, this is another problem. So, we because in, this is a normal operation in the when the scan enable is 0 then and in the BIST mode is also 0 then this one comes this comes.

Now if the enable line is enabling this D by, if this buffer then these D values is available at the output pin. Otherwise this is this pin is acting as input pin, so value comes through this. Now if we do not want that, if you want that they will this bidirectional pin. So, what we may want is that we want to configure it as output mode in the during BIST operations. So, during BIST operates this BIST mode signal is one as a result this is one this is one and this one they whatever be the value of this enable line whatever will be the value of this scan enable line. So, this line will get a one as a result the spot will be configured as an output port.

So, we can fix. So, in this case we have fix the direction of this IO port to output mode. Similarly you can have some circuitry on the lower side for this z. So, we can control this line to be always input in the BIST mode. So, whatever is required whether we want to the sense a line as input or output, we can put a control over that. Sometimes we need to do a re-timing, re-timing basically needs a needed when there are races and hazards by

caused by clocks skews because the text pattern generated. So, the out basic module if like this there is test pattern generated there is a circuit under test and there is a response analyzer.

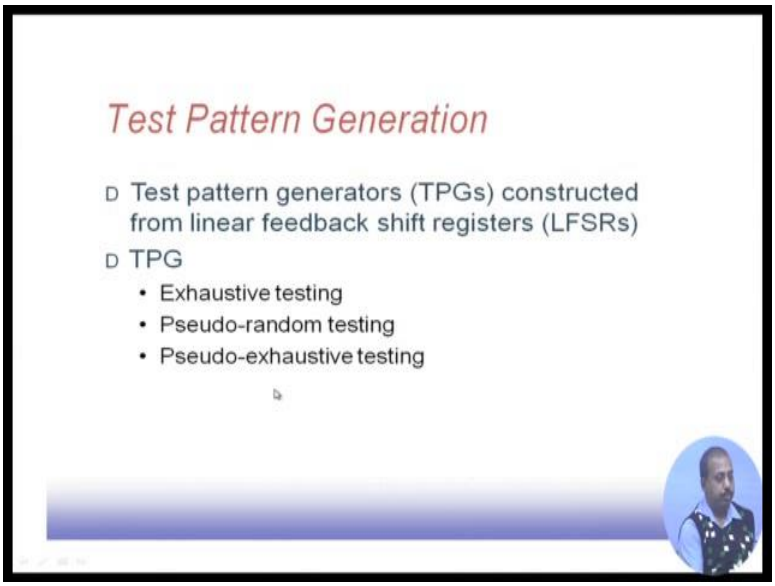
(Refer Slide Time: 05:12)



Now, each of them may operate on a different clock. So, these test generated may work on clock 1, test circuit may be operating on some clock 2 and this ORA may act as a clock in a separate clock CK 3. Now they are may be problem with this clocks because when we are, if they are the same clock then distribution of the clock into all these module. So, that will create that may create some clock skew.

So, what is done is that re-timing logic is added between this test pattern generator and the circuit under test and between the circuit under test and the output response analyzer. So, even if these clocks are different. So, they are out of phase there is a clock skew means you can think that as if they are three independent clocks - clock 1 clock 2 clock 3. So, these two D flip flops are added to operate on the clock 2 and two D flip flops are added to operate on clock 3. So, that they are the problem of skews will not occurred. So, this is the re-timing logic that is added which will avoid the clock skew. So, this is required to be done this modification has to be done to the circuit.

(Refer Slide Time: 06:49)



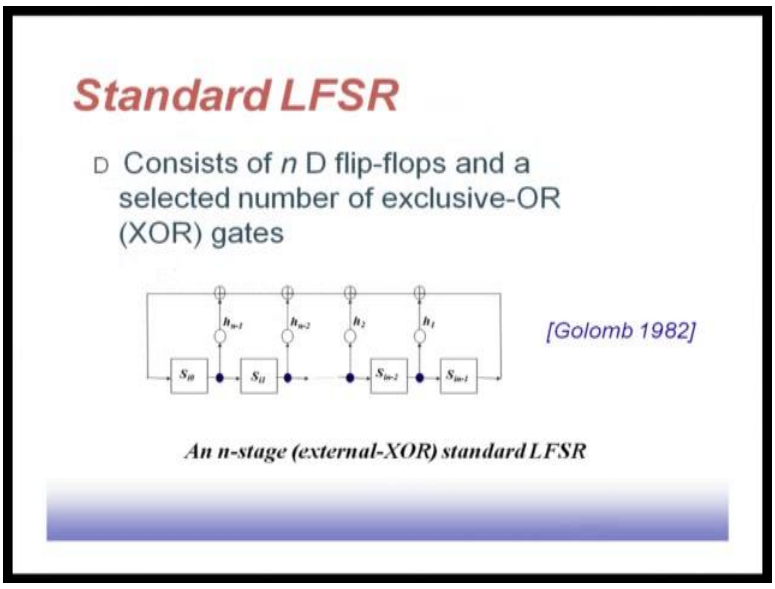
Now how do we generate the test pattern? So, these are the rules that we have seen while modifying a circuit or making a circuit BIST ready some modifications are to be done. So, that is unknowns do not come to into the BIST operation and these clock skew problem do not occur in the BIST operation. Similarly this tri state buffers they are also taken care of, bi directional IO they are taken care of. So, all to ensure that these unknowns do not come into the operation of the system because then this response analysis will become long, so while doing the test planning parts, we had assume some values for those points. But in reality when it operates it comes up with some other values. So, if the values are unknowns then the values may be different. So, as a result the final response of the golden signature may not match with the response that we get from the circuit under operation and even if the circuit is correct. So, it will be identified as a faulty circuit. So, that is why all these excess needs to be need to be blocked and this clock skews problem has to be solved.

Next we look into the test pattern generation process now how this test pattern can be generated now you see that typically this deterministic test patterns generated. So, it has given us a set of test patterns. So, as a test pattern generated what we can say is that we can have a memory where all these test patterns are stored and we can have a counter sort of thing that can produce addresses for the memory and each count value it outputs one particular pattern and that is going to be applied to the circuit.

So, that is that, that is the standard ATE type of operation, but when we are talking about this BIST operation we do not have such a sophisticated mechanism. So, we need to make this test pattern generator very simple and also this response analyzer have to be very simple. So, there are many such many such configuration may such circuits that can act as test pattern generates. So, in most of the cases they are constructed from linear feedback shift register are called LFSRs. So, apart from this linear feedback shift register. So, you can use some counters as test pattern generator, you can use cellular automata based structures for this test pattern generation. So, they are also there in the literature, but for our course. So, we will be the restricting ourselves to LFSR.

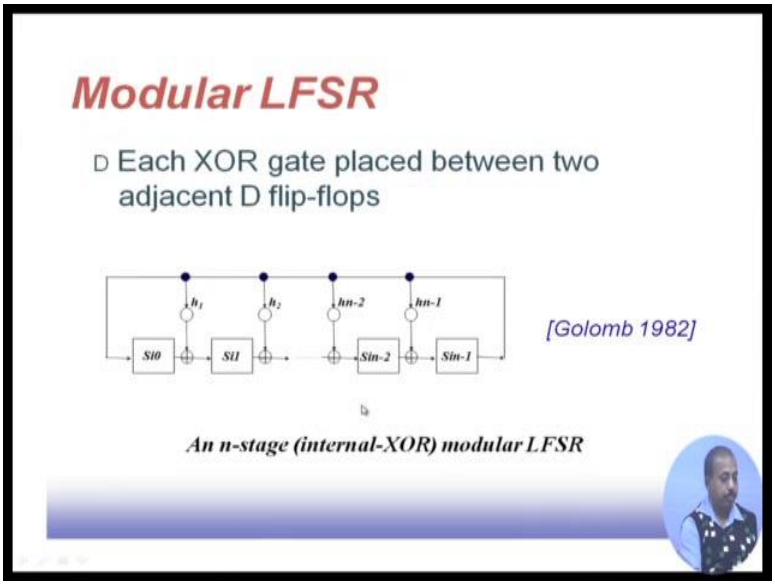
Now, this test pattern generation philosophy, it can be for different types of situations like we can we can try for exhaustive testing. So, exhaustive testing required that all patterns are to be applied to the circuit for testing then in pseudo random testing. So, this is basically a random testing, but it is pseudo; that means, after generating some pattern. So, it will repeat itself that is a pseudo random testing and pseudo exhaustive testing, it is it is not exhaustive at all the points. Like if it is an n input circuit maybe it is a for k inputs it is exhausting, other places it is not exhausted. So, there are subset of output for the test pattern generator where the generated pattern is exhaustive in nature. So, that can be there.

(Refer Slide Time: 10:20)



So, one standard LFSR it was proposed by Golomb in 1982. So, it consists of a n number of D flip flop and a selected number of exclusive OR gates. So, we have got this thing. So, this is n stage this is called external-XOR LFSR because the XOR gates are at the hour they are external to the sales. So, these are S i0, i1 and these are the flip flops that we have and then depend the, so h value. So, h they are actually switch whether this output goes to the XOR gate or not. So, that is controlled by this h. So, if h n minus 1 is one then this output will be XORed then this n minus 2 is 1 then this will be XOR. So, you see that the value that is coming back to S i0 is nothing, but a weighted sum of all these previous value is h 1, h 2, this is the current this is the output of S in minus 1 and then it is h 1 times this one h 2 times this. So, that way it is a weighted sum of the previous stages.

(Refer Slide Time: 11:30)

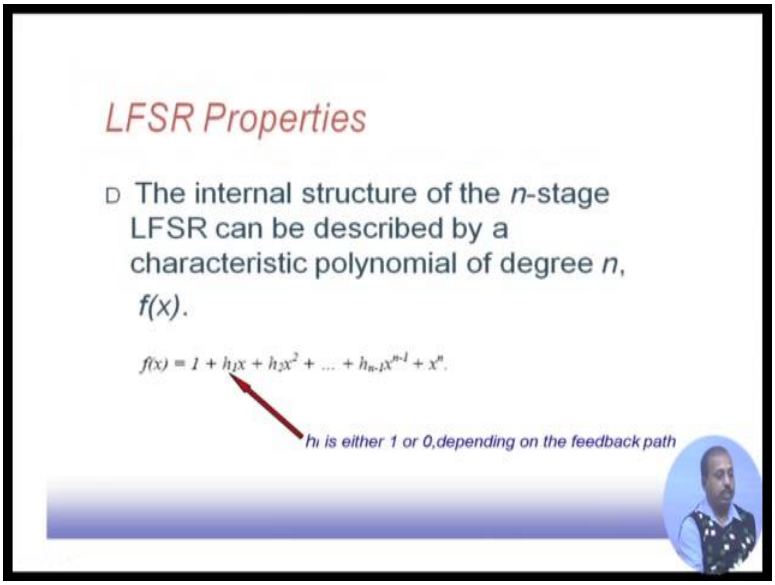


Another one is known as modular LFSR. So, in modular LFSR this XOR gate is placed between two adjacent D flip flops. So, the structure is like be different. So, between two adjacent flip flops. So, you have got this XOR gate input and this whether this value will be XOR here or not. So, that is controlled by this which h 1, this h 1 will be controlling that. So, the advantage of modular LFSR is that you can very easily add further modules further more flip flops to these chains.

So, extending this thing becomes easy, though functionally this module LFSR and this one this external LFSR external-XOR LFSR. So, they are going to be same standard

LFSR and this modular LFSR, both of them can realize similar set of functionalities, but both the structures are there.

(Refer Slide Time: 12:23)

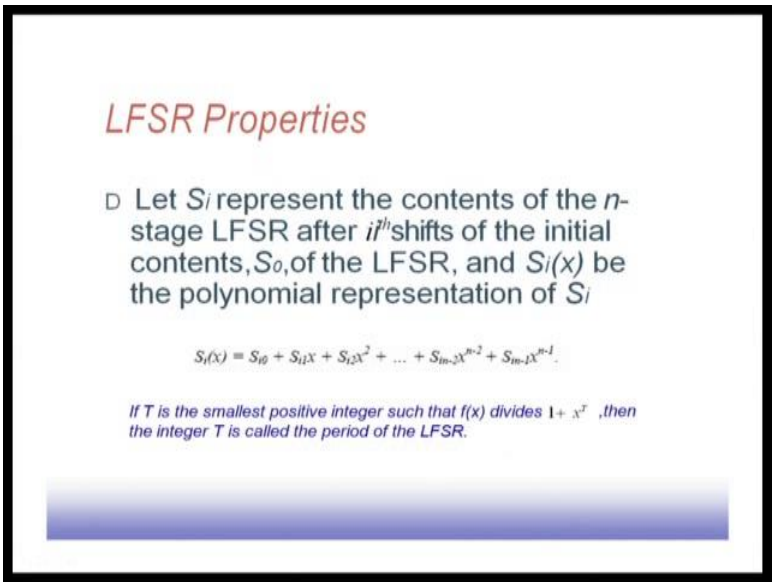


So, there is very strong theory based on polynomial that can describe the properties of this LFSRs. First of all this internal structure of n stage LFSR it can be described by a characteristic polynomial of degree n f x. So, f x is 1 plus h 1 x plus h 2 x square plus h n minus 1 x power n minus 1 plus x to the power n where h i is either one or 0 depending upon the feedback path.

So, you see 1 plus h 1 x. So, this is actually if we say this as 1, so this is a one step before that. So, this is h 1. So, h 1 x 1 step back before that h 2 x square that way it is coming. So, h n minus 1 x n minus 1 x power n minus 1, so that is and x to the power n is the value that we are computing.

So, that way this polynomial is representing the connection pattern. So, where ever this h i are 0 or 1. So, if it is 1; that means, on the feedback path this particular term will be there. So, that XOR gate will be there in the circuit and if it is not there; that means, the XOR gate is not there.

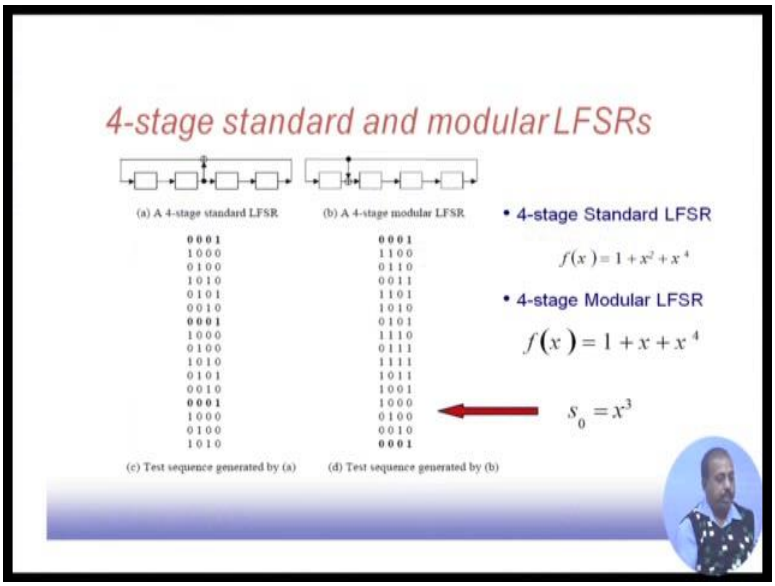
(Refer Slide Time: 13:41)



So, content of the LFSR at any point of time can be represented by S i. So, S i x is a polynomial representation of S i. So, this is again given by S i 0 plus S i 1, S i 2 x square. So, these are basically the contents of the values at various points and if the value is as to be considered then it is this term will be 1, if this is 0 naturally this is. So, S i 0, S i 1, S i 2, these are the values of different flip flops and if it is one that term x power of x is coming into picture if that term is 0 then it is not coming.

So, content of the LFSR is represented again by a polynomial. So, there is these LFSR they show some periodic properties. So, if t is the smallest positive integers such that f x divides 1 plus x to the power 10 x to the power t then the integer t is called the period of the LFSR. So, the thing is that if you have this LFSR then after time period after t search shifts like if you initialize the LFSR with some pattern and then just go on giving clock to the LFSR then it will go on involving itself and in the evolution process after t clock cycle it is going to repeat the sequence that it is generating. Sequence in the sense the sequence of the content of different cells that are there so that is going to be repeated. So, this t is called the period of the LFSR.

(Refer Slide Time: 15:22)



So, it gives rise to many interesting properties like see say this one. So, here we have got, if we count it as say. So, this standard LFSR, if you count it as 1 x, x square, x cube, x to the power 4 then you see that while doing this addition. So, we are taking the terms like say this 1 is definitely there plus this x, x square term is coming and x to the power 4 term is coming.

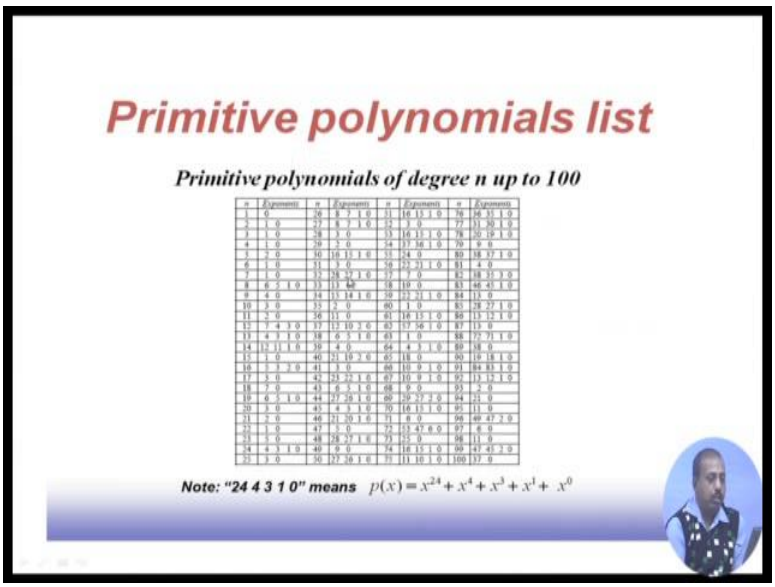
So, they are summed up and it is fed to the first flip flop and in say of modular LFSR also it is 1 x, x square if you x to the power 4. So, x to the power 4 is definitely coming in all the cases you see from the last cell we are always taking the feedback. So, this x to the power n terms is always present. So, there is no switch to control it. So, this x to the power n term is always there, but otherwise it is coming from this other delay terms. So, here it is 1 x square 1 x, x square x cube x to the power 5, here it is one x square x cube x to the power 4. So, these appropriate terms are getting.

Now you see that this is a four stage standard LFSR. So, if you start with the pattern 0 0 0 1 if you feed it the patterns 0 0 0 1 then after. So, after one cycle what will happen? So, this is basically the sum of the second cell and the last cell. So, 0 and, this 0 and this 1. So, 0 and 1 they are summed up, so value is 1. So, you see that value is shifted on to this flip flop. So, this value become 1 and rest of the BIST. So, they are just shifted. So, these three bits, they are getting shifted through this shift register they are becoming 0 0 0.

In the next this, this second cell and again the second cell and forth cell they are XORed. So, second cell forth cell XOR the content is 0. So, that is coming back to the first cell so this is becoming 0, but this one shifts to the next cell these 0 shifts to the next cell these 0 shifts to the next cell. So, we get 0 1 0 0. So, in this way you see that after this many patterns say 1 2 3 4 5 6 after the generating six patterns, it is repeating itself. So, this has a got a period of six. So, on the other hand if you look into say this particular pattern, this particular LFSR. So, you start with 0 0 0 1 and if you just follow the logic that is captured by this, if we are going to, if we are giving a clock signal a clocks to this flips flops continually.

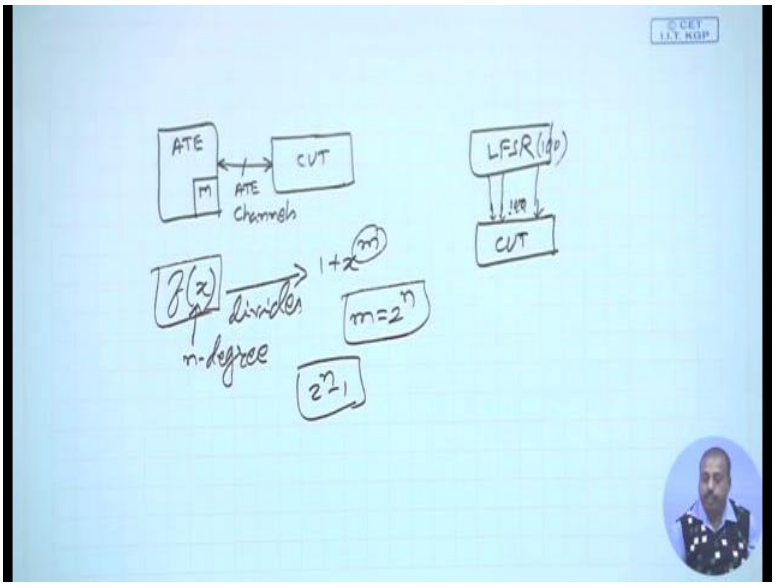
So, after initializing with 0 0 0 1, the value that will be computed the first cell. So, these one is coming back to the first cell. So, one comes here the second cell gets the sum of the last cell and the first cell. So, last cell is 0 and 1, they are actually XORed, you get 1 at this point. So, this way this shifting is going on and interestingly. So, this gives a much longer sequence you see and it has repeat itself like after generating 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 pattern is repeated. So, it has got a periodicity of 16. So, 16 pattern is going to be repeated and content of any, at any point of time if you say you see the configuration, so this is 0 the initial configuration is x cube because this bit is set to 1. So, this bit initial configuration S 0 is x cube. So, this is 1 x x square x cube. So, that way if this content is x cube this cell is x cube. So, that way in the initial value is all scans also be represented by a polynomial.

(Refer Slide Time: 19:24)



Now there is a concept of primitive polynomial. So, primitive polynomial says that the polynomial, if the characteristic polynomial if it is the, if I have the got some characteristic polynomial f x and if we are trying to figure out this type of polynomial 1 plus x to the power m such that f x divides 1 plus x to the power m.

(Refer Slide Time: 19:38)



So, we are looking for the value of m so that this f x will divide x to the power m. Now if this x to the power m this m happens to be 1 to the power n where this f x is an n degree polynomial. So, these happens to be 2 the power n then we say that this f x is a primitive polynomial.

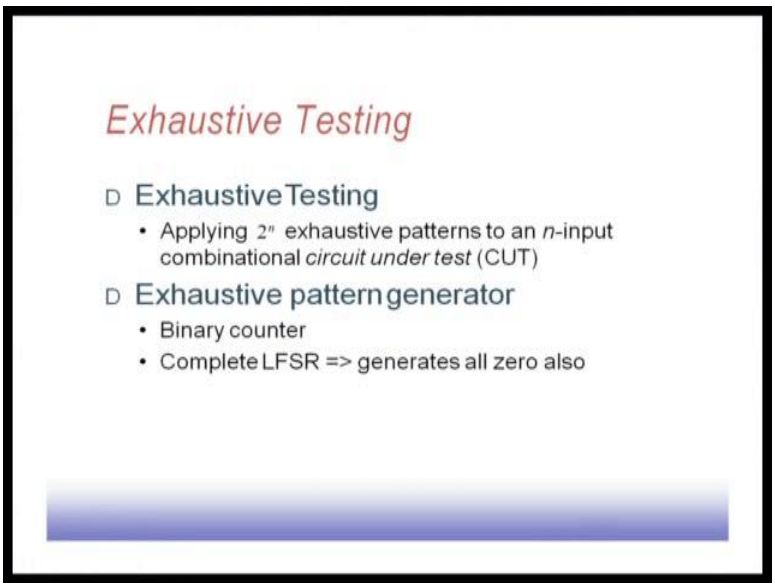
So, this is that, so primitive polynomial if it is there. So, it has got the property that it will repeating the patterns generated after generating all 2 to the power m patterns or basically not all 2 to the power n, 2 to the power minus 1 pattern, although it will not generate, but all accepting all although it will generate all the patterns. So, you start that LFSR with any pattern, any non 0 pattern. So, it will generate all the 2 to the power n minus one pattern then only it is going to repeat itself

So, that way, this type of polynomials are known as primitive polynomials. So, if an LFSR has got this property; that means, we can use it for test pattern generation because it will generate a large variety of patterns. So, it is not going to repeat in a shorter cycle. So, there is, many experimentation has been done to find the exponents the polynomials this type of primitive polynomials, this list are available. Like if you are trying to

formulate a polynomial of say degree 24, degree 24. So, in this 24 you see that it is written 4 3 1 and 0. So, x to the power 24 term will always be present because that is x to be power n term, but the remaining typing points will be at 4 3 1 and 0. So, this becomes the primitive polynomial for n equal to 24.

So, this list is available and in fact, this list is available of up to I think up to 512 number of 512 n equal to 512. So, this list is very big that is available. So, for our purpose whatever polynomial we need, whatever if I have got say - if I have got a these LFSR who act as the test pattern generator and we see this that we have to feed to some circuit and this circuit has got say 100 inputs, 100 inputs. So, one straight forward way maybe that we use an LFSR of length 100 which are the 100 bit LFSR and feed this directly to this. So, this patterns that are generated by the LFSR will be feed to the circuit as test patterns. So, if it a primitive polynomial it will ensure that it will not repeat its sequence before generating 2 power n minus 1 are new patterns.

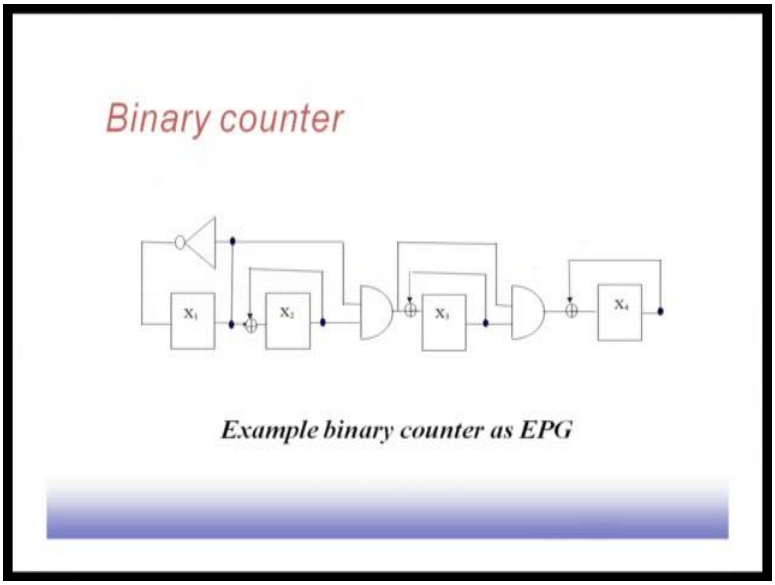
(Refer Slide Time: 22:53)



So, we will look into many testing techniques for this BIST environment the first one is known as exhaustive testing. So, exhaustive testing means we apply 2 power n exhaustive patterns to an n input combinational circuit under test, and exhaustive pattern generation. So, there can be a binary counter. So, binary counter, since we want to generate we want to apply all 2 power n patterns. So, if I have a binary counter that goes from 0 to 2 to the power n minus 1. So, it generate all those patterns then of course, we

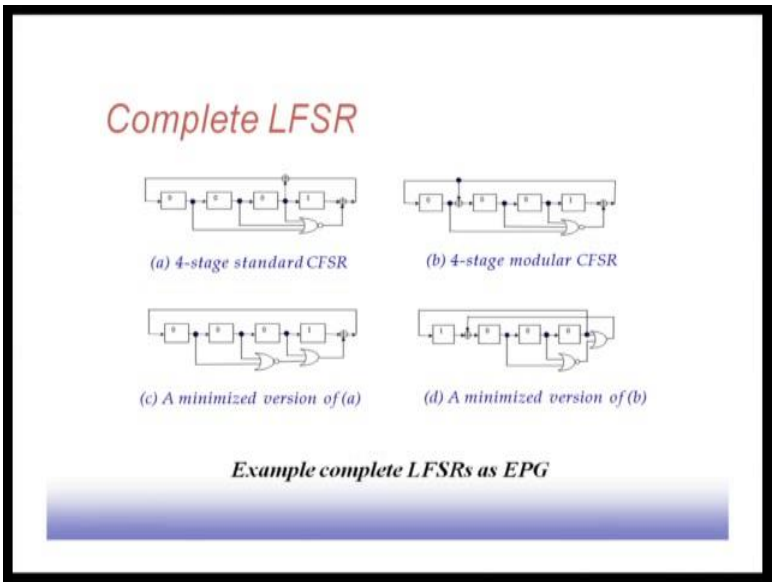
can use it as a exhaustive pattern generator. Other possibility is that we can also use LFSR, but LFSR the problem is it does not generate all 0 pattern, at any point of time the LFSR has become its content has become all 0 then it is just going to remain at that stage it is not going to be updated. So, that has to be LFSR has to be modified. So, that it generates all 0 state as well.

(Refer Slide Time: 23:58)



So, this is one example for a binary counter can be used as an exhaustive pattern generator. So, it is not detailed here, but you can check like if you start with any pattern I say 0 0 0 0 and go on giving clock to this flip flop. So, these are actually flip flops x 1, x 2, x 3, x 4 these are flip flops. So, if you go on giving clock to this flip flops then in successive cycle. So, it will generate new patterns and it will repeat after generating all the 16 patterns, so 0 to 15. So, it may not be in that sequence, but it will generate all those patterns and then only it will come back to 0. So, that way it can be done.

(Refer Slide Time: 24:39)



Other possibility is to have complete LFSR. So, complete LFSR is something the normal LFSR standard LFSR. So, this will this generates this generates the 2 to the power n minus 2 patterns. So, it does not generate the all 0 pattern. Now here there is a modification what is done? When the cell contents are 0 0 0 1, use a NOR gate here. So, if this NOR gate getting all the inputs as 0. So, it will make the output as 1 and this 1 and 1. So, that will this XOR gate will output will make 0 where the. So, as a result this cell will get a 0 and in the next cycle this 0 will shift here, so content of the cell will become all 0.

So, that way this circuit this cell will all this LFSR will generate all 0 as well and when the content is all 0 then this gate output is 1 and this also being 0, this will become 1 and in that case this 1 will be fed back here and then it will acts as normal circuits. So, it will it will not remain in that all 0 stage continually. So, that why this complete LFSR can be constructed from standard LFSR by adding this extra NOR gate.

The same is true for modular LFSR like we can add this extra NOR gate and we can ensure that this is not going to remain in the all 0 stage. So, this is some minimization that is done. So, this basically some logic minimization, by doing this logic minimization, we can convert this circuit into this. So, that this gate count gets reduce. So, here we are requiring 2 XOR gate 1 NOR gate and in this modified version we require 1 NOR gate, 1 OR gate and 1 XOR gate since the XOR gates are costly. So, this

will reduce the circuit over here. Similarly here also we are reduced we are requiring 2 XOR gate and 1 NOR gate the modified circuit will become 1 XOR gate, 1 OR gate and 1 NOR gate, so that can be done.

We will continue in the next class.