

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

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

VLSI Design, Verification & Test

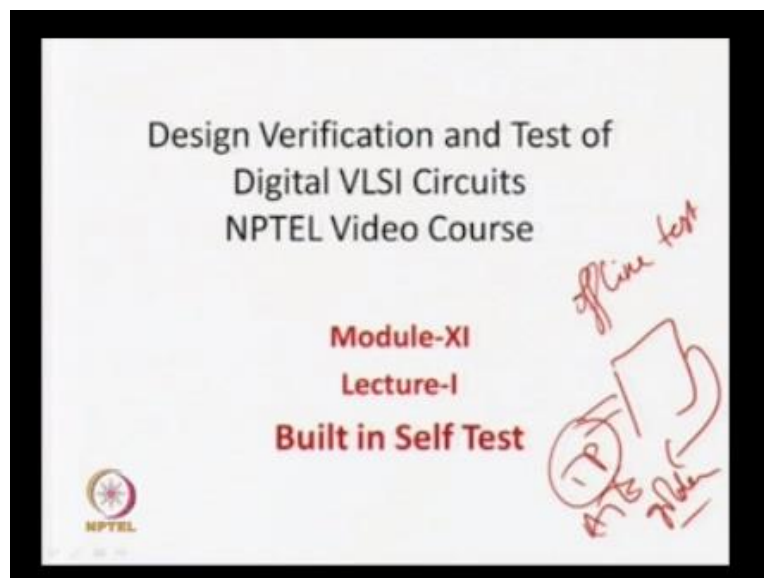
**Dr. Santosh Biswas
Department of CSE
IIT Guwahati**

Module XI: Built in Self test (BIST)

Lecture I: Built in Self Test-1

Okay, so welcome to the next lecture on the next module on VLSI testing part of this NPTEL course. So by till now what we are discussing were talking mainly about different type of fault models applied to sequential circuits then combination of circuit sequential circuit how could you test efficient test patterns and all those things. So but actually if you want to club all these things we can coin one term that is called actually offline testing.

(Refer Slide Time: 00:54)



So we can call it as offline testing, so in this what happened so you had a circuit so you applied some test pattern through an ATE and this golden response was compared okay, that is like all the effort so whatever test patterns you want to apply so that has to be determined so for that we have found out simple algorithm like the algorithms then we have found for sequential circuit we use what you called these time from expansion method.

And sometimes we are used the algorithm along with the time for testability scheme called the scan chain and also so we have also our random test pattern generation can be help to find out this test patterns. But as a whole what was the idea, the basic idea was that the cheaper view fabricated you will put some all these test patterns match the golden response and your job is done and you sell off in the market.

But this assumption you can take I mean if you are working say around say 90s or pre 90s kind of a thing, but as our technology or as our running integration technology is becoming sophisticated and more sophisticated we are going to dip some micron designs so they may other problem that arise is that once you manufacture your chip you sell it to the market you cannot guarantee that the chip will be okay throughout its life time or the expected life time that means what some of the faults may even develop when your circuit is an operation or it means the develop after say one year after some six months of operation.

Now your whatever the circuit is placed extremely complex board. So now if something does not work then you have to be part that which part of the circuit is not working in a complex system like a mother board or you can say that the complex mother board of your, what you called the laptop or mother board of your mobile.

So one of the cheapest not working, but as a whole your getting some of the email functions. So now it is very difficult for a person or a engineer to debug by looking at by taping the external points and debugging and finding out exactly which it is not working, because such type of failures are very common in dips some micron designs that individually each chips are okay they were in the many faults when they were in the fabricated then we sold it in the market.

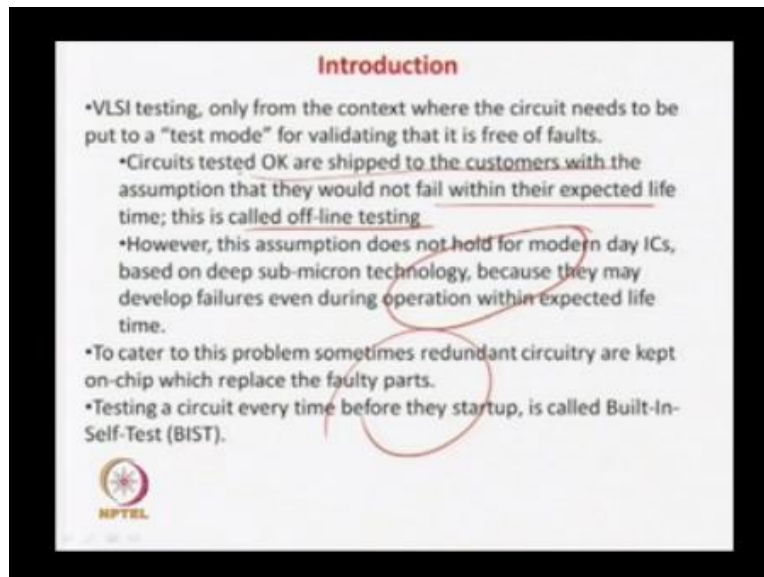
But after operating for some say 6 months so that is within the lifetime very well in the lifetime there is some problems. And then you have to debug so the debug takes a lot of time because you have to find out exactly which chip is wrong then you have to replace that chip, and so then I mean now the chip is already fabricated in the board, so you cannot apply the test patterns and all those things.

So the lot of problems in easy to testing for the next level of testing from offline actually this is called offline already we discussed. All lectures still now are offline and they something called insito. Isito that means your chip is already placed in the board and now you want to know whether still it is operating fine, because that way is only the way you can debug your circuit, if something is there problem in the system. So now it is very difficult as it is already placed in your like this is your mother board and your chip is already fabricated and solded over here.

So it is very difficult to apply test patterns and all those because it is very difficult to access individual chip used will be connected to some other parts of the chip. So that will actually make your life very difficult. So now what is the solution, so what you can do so that is the requirement of insito testing is very much important the important arise, because of the some micron is false which can occur even after fabrication and totally being tested to fault free they can occur when the system is in operation or you will deployed it.

So on now on chip that is insito testing is the requirement for all circuits nowadays. So that means what that is actually called built in self test, that is what we are going to see in today's lecture.

(Refer Slide Time: 04:13)



So what it says so that means what the idea is here that we have to have some arrangement in the chip or in the board that can apply some test patterns at can analyze the response and tell that the circuit is fine that circuit is fine and so forth. In other words so once your circuit is to be at the offline tested to be okay and file using an AT you select to the market. Now I put it on my mother board or I put in my system.

Now every day before my chips starts its operation that at this before your boot your computer or before you start your advance mobile phone every chip will test itself. So obviously that testing part what you calls that testing cannot be as exhausted as your doing an AT, because AT is a sophisticated never you the huge memory sophisticated machine is with huge memory where you can apply your patterns and where you can compare your response very efficiently and come fast.

But obviously if you want to make a miniature version of the AT and you put it on chip, so what happens after your chips start its operation you can test it partially or you can say that I can apply some very important test patterns and I cannot estimate with that circuit is working fine or not.

So the debug problems very ease in this case like for example so the 10 chips in your board and your circuits is not working fine.

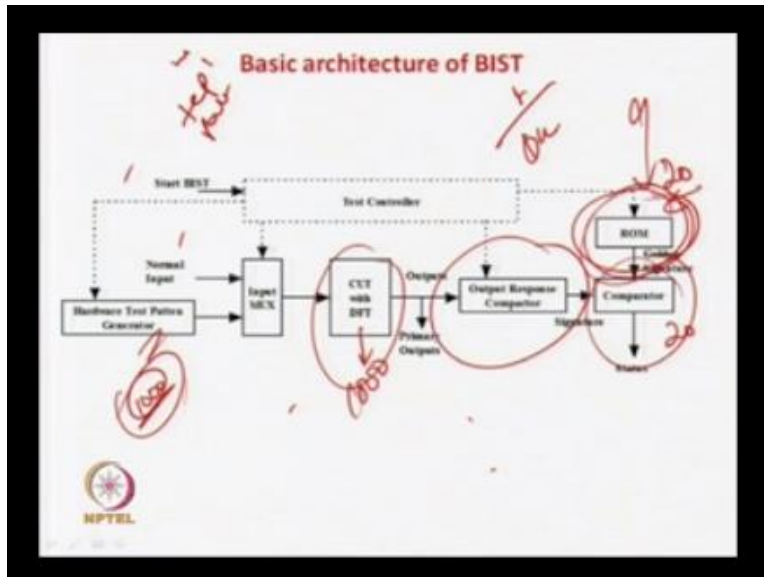
Then what you can do you can built in self that is actually insito testing that is you can say that a part of the test pattern already you have used for offline testing can now we applied for another circuit which is on chip that is a separate circuit which is used only for testing your circuit as testing pattern application and response analysis. So obviously their sub part of your test patterns will be applied in all the chips before they start their operation and any one of the chip you will find that the chip is not working fine.

So you can very easily pin point that this chip or these are the chips where there is the problem, so that can be very easily debug and you can pin point because you do not have to because all the chips are soldered already put in the board. So you cannot apply any external elements external tester, but that is not required, but because your test pattern application circuit and test pattern analysis circuit are already in the chip.

Now you can the chips were this day on chip testers which is actually called built in self test that is called built in that is already built in tester is built in the circuit. So that will respond with saying that this circuit is not working fine or this chip is not working fine, so you can easily pin point that chip is not working fine you can eliminate that chip and put another chip. So your test time because very, very simple and easy for a test engineer when he is doing testing or a board level or a system level.

So that is what the introduction do best, so circuits when tested or shipped to the customers with the assumption they will not fail with the expected lifetime these are offline testing already discussed. But in modernizes this will may not hold, because even during operation the circuits may fail to get out of those problem we test every circuits before they start up an putting some circuit on chip. So that is actually called built in self test.

(Refer Slide Time: 06:53)



So now will come to the basic architecture of best how it works you see? So this is actually your circuit with DFT some may be you can this DFT can be scan chain then etc.. for some DFT is there. Now so some normal inputs which you apply so the primary inputs of your circuit so this inputs will go by out the marks in test more of this been selected that is more equal to zero normal mode.

So that will go the operations and you will get primary output, so this is you can think that this is the only small part of your circuit which is actually your normal circuit already there are general circuit which is been used. Now what happens so actually you can say that in other blocks you can see many other blocks, but that area is not very high so when we will discuss all that while the area is not high.

So this will actually consumes 99% of the area the main circuit and all other parts are very small in size as we will see. So now what happens now your circuit is now this part is 99% of area that is anyway will see later. So you see that this part of the chip is tested this part of the chip, this part of the circuit is tested well and verified fantastic fine and tested okay and it is sold in the market after offline testing.

Now it can happen that every time before you start your operation that every time you start so it may have some faults or it may have some fault which is develop when circuit is operated. Now as all circuit are fabricated in the chips in the board so it is very difficult to debug where there is the problem. So you have to do insito testing that even after deployment of the chips as the circuit in the boards you want to test them.

So how do about it so some extra circuits will be required to do that so what is actually called hardware test pattern generation. So what is that so this will actually continue test patterns which you have determined using D algorithm or your D algorithm scan chain or random test pattern generation or time for expansion whatever will be the case. So these are the test patterns which you have already generated.

Now you have to observe that this test patterns are generally applied from an AT, AT is an huge expensive machine big machine is a lot of memory and it is very powerful one in that sense. So but now this is your main circuit and obviously say its area is say 1000 unit you cannot have 10000 unit circuit to test that is infeasible and people will not appreciate that. So what you have to do you can only have a test pattern whose outer of area may be 10 or 100 max if it is 1000 it cannot be even order of 100 if area should be 10.

So what it accuracy do will see that is actually sub gather were random test pattern generation and obviously it is very difficult to generate all the test patterns which an AT you can generate because of that memory constant and other constant so you can generate very important test patterns or in other way you can say that we are going to test this circuit for a very few most important factors of only test pattern all generated by AT algorithm or in other words another example can be say test pattern 1 detects 10 fault yet balance to detect 20 faults test pattern detects 1 faults and so on.

So you need on the use of those test pattern detects more number of parts because you are limited by the memory and hardware requirement of the test pattern generator. So you will say, you use some percentage of the test patterns which were using in the offline testing. So now when your

test mode is 1 then this pattern to the update circuit these are actually subset of offline test patterns.

Then these patterns will be applied to the circuit, but these are actually a subset of offline test pattern so this test patterns actually a subset of offline test patterns and this is a subset okay so this is what you do it. Now what happens so a subset of they will be applied now there is I will tell you they are wrong so in the wrong module store you store actually whatever is the golden response or the expected response for this test patterns.

Now you can see that again this is on chip circuits so this area is 1000 unit we are send a mean circuit so we said that we give it 10, we give it 20 kind of thing competitively give it 20 this can be 10. So we can say that at max we can have 10% area over it of the general circuit. So including all other block this is a 1000 area over it some unit then all other test control are that for including everything you can apply only say 100 units of area.

So you can think that if I use I mean what you can say for 100 test pattern maybe there are 10,000 test patterns in the for offline testing now you have got on the 1000 you selected very good test patterns for official test pattern we detects more number of faults and you generate with the random pattern, then in the RAM is ROM if you want to store the golden response corresponding to all this 1000 test pattern the ROM size will be very, very high so that you cannot do.

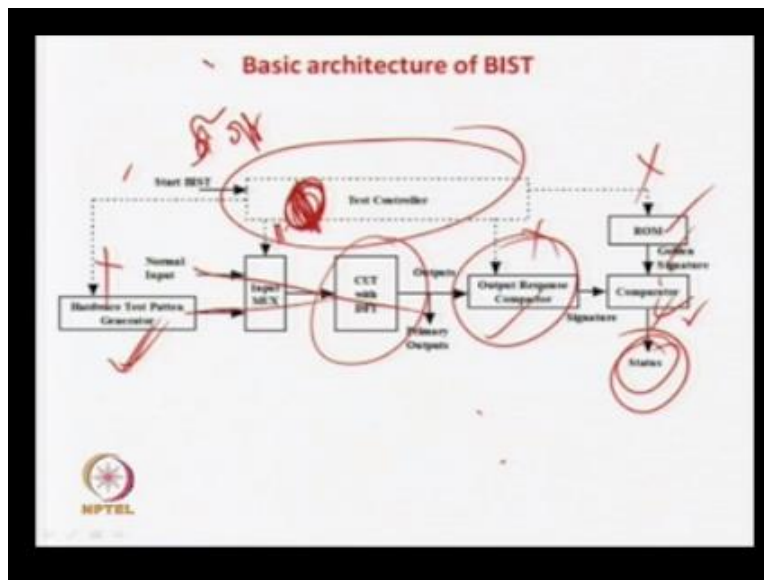
So exact response for this 1000 test patterns or whatever cannot be stored in the ROM. So whatever you do that we are something for a response compactor that is whatever response you get we do a directly using the comparator we compress them. So by compressing there is something for aliasing which will see in this lecture. So aliasing means what so by compression if the logic compression that sometime, because of the compression you may lose the efficiency that is mean if the compression was not there you could have detected the fault.

But because the compression the detection may not be there that is for losing or aliasing is there present in that, but you are to compress it, because if you have do not compress then the ROM

size will be very, very high and it is not possible, because then if you have corresponding to all the test patterns which you are directed by the test pattern generator you store the answer what you can call it as golden response in the ROM, the ROM size will be very, very high and you cannot do that.

So you have to compact it and then for the compact response for the compact you can have a compact test pattern in the output compact patterns you can have the response in the ROM, then you can compare with the alternative. Then in this comparator and if it matches then it is there with see all these three examples and along with all this there is something called a test controller.

(Refer Slide Time: 12:22)



So now what is the test controller this whenever the circuit will start this operation so you will say that start bits so that some kind of a signal and what it will do. It will make $m=1$ because it will select this pattern it will ask it to generate the random patterns it will ask this to compact the responses as well as then it will also allow the compressor will start. And whenever you say that this is done so you get the status and all the circuit is fine.

So you can go for the normal operation of the circuit so in that case you say that bits is off then controller will make $m=0$, it will stop its operation deactivate it and it will detected with similar to deactivated all. Because one more thing you understand that whenever your circuit is doing the operation so only this part of the circuits will be running unnecessarily this one, this one, this one should not do this operation or should not being execution and then there will be lot of for consumption.

So the test control takes care of all this, so whenever you want to do a best you make $m=1$ this is connected you start operating all this, you start and get the results status was the test is done then the test control will make $m=0$ it will make so the normal input will go through and all other parts of the circuit will be this part will be made off okay so that is saving in this one.

(Refer Slide Time: 13:38)

Basic architecture of BIST

Hardware Test Pattern Generator:

- This module generates the test patterns required to sensitize the faults and propagate the effect to the outputs
- As the test pattern generator is a circuit (not equipment) its area is limited.
 - So storing and then generating test patterns obtained by ATPG algorithms on the CUT (discussed in Module Xi) using the hardware test pattern generator is not feasible.
 - Instead, the test pattern generator is basically a type of register which generates random patterns which act as test patterns. The main emphasis of the register design is to have low area yet generate as many different patterns (from 0 to 2^n-1 , if there are n flip-flops in the register) as possible.

The slide includes a logo for NPTEL in the bottom left corner. Handwritten red annotations include a circle around the text 'its area is limited', a circle around 'test pattern generator', a diagram of a register with a clock input and data input/output lines, and the binary strings '00001' and '11110' written in the bottom right.

But that is basically the best architecture so but here one point have to observe that somehow we have to make this ROM very small size and also the test pattern generation being small size, because if you compact when AT, AT also is the hardware pattern generator and a ROM which actually analysis the response, but AT is a very big machine with lot of memory power so you

may not limit the number of test patterns only the test is limited by the test time, because if you are using too many test patterns then your test time will be higher.

And so your AT cost will increase, but from the memory requirement size and all this maybe not much of a concern, but here it is very much concerns because it is a circuit which is good on chip and your main circuit is with some area restriction some 1000 unit of area is there nobody will allow you to use a best which is again have 1000 area or equivalent area of the normal circuit.

So your best area over it is very small say 10% or 8% or 5% is one of the better area over it compared to the circuit. So that is why it is very critical to design the which test patterns you have to apply and how you compact them, so that is very important so that we will see slowly in our as we proceed though our lecture okay. So now let us see whatever we discuss so there is actually a this was the hardware test pattern generator.

So what happens so this test pattern generate the patterns equate to sensitize the fault and propagate the effect of the output that is the same test patterns which we develop in D-algorithm or its time form the sponsor method or whatever you can call the random pattern generation, but in this case the test pattern generator is not an equipment it is a circuit so this area is limited that is why you can use only a part of them.

So storing and generating test pattern of the ATPT algorithm this one is not visible like whatever we have done by AT algorithm what times expansion method cannot be directly applicable. Instead our test pattern is basically generator is what that we will see today is basically time for register which generates random patterns. So it not a specific pattern generator like you can say that I request 0001, then 1110, then 0011, then 1111, these are the four patterns are required okay.

So for that you will build a circuit it is very difficult to do that because area over it very high. So what we do in this case we use something called a random test pattern generation register it will generate all patterns randomly. So you can use and so it is a very small area over there we will

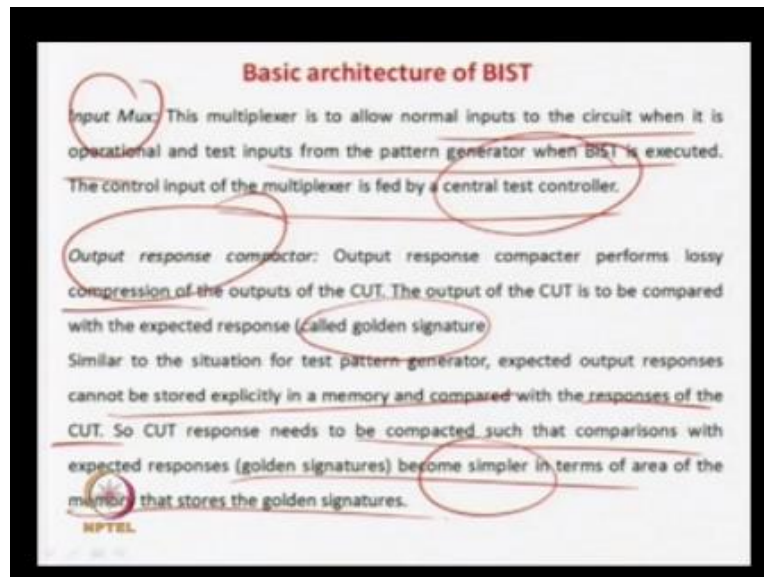
see it will start generating lots of random patterns and now what is our duty is that you can select that I mean you can put a, as we will see you can control the sequence of the random patterns.

So you can say that it will generate all random patterns saved from 0 to 2^{n-1} generally 0 is not generated we will see wise the case. So like say 1111 to 2^{n-1} all the patterns it will kind of thing sorry, 0001 to 2 all patterns will be generated, but whatever you require so you can make the registers of this random pattern generation in such a way, so that whatever you start generating there is something call the seed as we will see it will determine what is the sequence of random patterns.

But it can generate all pattern in between all 0001 to 2^{n-1} that is 1 to 2^{n-1} so and say you 1 pattern 00001, then 1110, then 11 these are the four patterns you want. So you can arrange in such a way so that in initial few iteration you will get all the required three patterns and then the other random patterns will be generated as they not required for you. So after this 3 or 4 steps when you get all the results all the patterns you can reset the circuit and then again you can restart it whenever you require.

So but you will see that the random pattern generator will not generate on the specific 3 or 4 patterns you requires it will generate all random patterns from 1 to 2^{n-1} . Now we have power duty is there or you can generate the random pattern such a way so that whatever patterns you require occur initially before the undesired which is do not require for the random pattern. So once all the required random patterns are generate you can reset the random pattern generator then stop the best the best kind of this that is the basic philosophy. We will see with the examples.

(Refer Slide Time: 17:42)



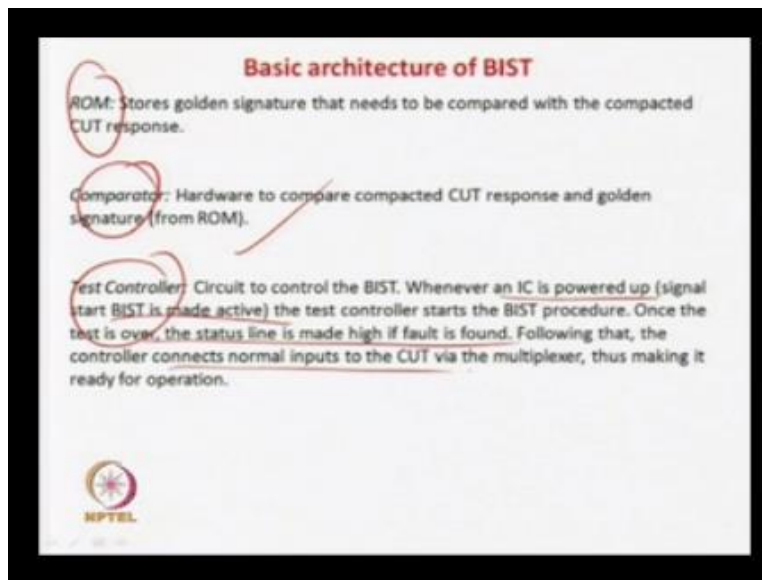
We have also seen an input multiplexer so input multiplexer is nothing but it actually normal input to the circuit when the circuit is operational and test input when disc is executed. The control of the multiplexer is the what you can call the central test control as we are already seen so it besides on the controller. So output response compactor so the output response was the logic compression as we already told you while you call the logic compression, because if you are using non compress structure then for all the random patterns which we are using as test patterns the output response will have to be stored in a ROM.

So the output response I mean for all it will be very, very high the large size is very much restricted what you can call the golden which one will be very large in that case if you are not compacting it, but similar to the test pattern generation the output response is cannot be explicitly stored in a memory and responses cannot be compared, because it will make the drum size very, very high.

So what you have to do, so we have to compact it so that golden signature is not become simpler in terms of area of the memory that showed the golden signatures. So that using compressive so

that whatever you have to compare will be smaller in area size or in size so your ROM area will be less, but is a logic compression.

(Refer Slide Time: 18:40)



So sometimes we will find that we are actually using some of the test coverage because of the compression. But still we have to go about it because you cannot have a legacy or you cannot have an existence of using a huge area awarded in this case okay. The next element was a ROM so which stores the golden response to be compared again the golden signature here in this case is the compacted signature.

So this is your comparator so the comparator is the comparator circuit is the digital bit wise comparator so it will compare the output response of this compressed output response of the circuit maybe golden signature in the ROM. So this is equal it is say yes if this is a failure it will that it is a fault and the test controller, test controller is the controller of the circuit, so whenever the IC is powered on if the B signal is made high.

So this B procedure is started once the B status is high or low whatever you know that it there is no fault in the circuit then you can collect the circuit to the normal input to the must then your

circuits will start this operation. But if you find out that there is a problem in your circuit then the B circuit will not allow your circuit to operate and it will give message this part of the circuit is not working in the motherboard that is chip is not working.

So you replace the chip and then all your circuit will start operating otherwise you cannot operate your system with the 1G fault. So that is another advantage of this with one or two fault a chips your system will not start operating so that in middle of some operation you get a malfunction then you will lose your data and support. So it will not along with your circuit or your system to boot up say these are the problem here you better recover it. And replace the chip and then start the operation.

(Refer Slide Time: 20:08)

Hardware pattern generator

There are two main targets for the hardware pattern generator—

- (i) low area and
- (ii) pseudo-exhaustive pattern generation (i.e., generate as many different patterns from 0 to $2^n - 1$ as possible, if there are n flip-flops in the register).

Linear feedback shift register (LFSR) pattern generator is most commonly used for test pattern generation in BIST because it satisfies the above two conditions.

There are basically two types of LFSRs,

- (i) standard LFSR
- (ii) modular LFSR

Handwritten annotations: "of 0001", "n", "2^n - 1", and a diagram of a shift register with feedback.

So that was about the basic best architecture okay that and about the required made of all this. So now what we are going to see there are two major parts of your circuit here, one is your BIST test pattern generator and another part is what, another part is about your this response compactors.

This is compared with the digital circuit you have already learnt in the digital course BIST controller is nothing but the controller circuit it depends on based on requirements you have to generate some signal so it can be the design using digital circuit fundamentals and kind of or we map the optimization compared with the digital circuit so only two important parts which you do not know, we have to learn in the testing module is the test pattern generator this random test pattern generation and the response compactor and then the access.

So these are the only two modules which we will see other can be recollected or you can go back and look for your digital circuit fundamentals and you can do that. So what is the random of this pattern generator. So what you see say for example, if you have a circuit like this say it has an input and some output. So and you know that there is expected to 2^{n-1} kind of test patterns will be possible over here.

So we have just found saying that 0 is not possible we will see why 0 is not possible to be applied. So let us assume that there are in this test pattern input they are order of 2^{n-1} patterns can be possible, so along them say you want K test patterns to be applied okay. And this K test patterns which you want to apply which you have determined the D algorithm or timeframe expansion or whatever so this K test patterns has to be applied to analyze the circuit for failure in the test.

Now what we have to do if I generate a circuit which generates all this K patterns one by one the area will be very, very rare that we will see with an example. But now what we can do the other way is that there is something very much important area is low area over there, so they some sort give textual which takes very, very less area over it which actually cost pseudo random pseudo exhausted pattern generator.

So it generate at most almost all you can say all patterns from actually 0 is not there I mean 0 will generate from 1 2^1 and there are N flip-flop in the register they should have a random pattern generator register is there, so we can start from 0001 and 2^{n-1} for all patterns it will generate among them. And they will be in a random fashion, so there is actually cause pseudo exhausted

and it is also called random pattern, because these patterns are generated in a random way depending on a initial seed.

So initial seed means you have note the register with some value other than all 0s and after that you start generating the patterns in a random manner or it is actually called pseudo random, because it will depend on the pure there is nothing called pure random because you depend on this C and take an actions. So you will generate a pseudo random patterns and among them pseudo random patterns at some of these, pseudo random patterns they are 1, 2, 3, 4 something this will generate.

And among them your K patterns will obviously be there which you need to apply to this one. And if you want to apply this in a specific order like K1, K2, K3, K4 which are among those K this must be applied we require a very sophisticated circuit like initial pattern is a 00001, then 1111, then 00010 something like this.

So if you want to generate this three patterns which are your greatest patterns for this so you know that you can implement it is and you can optimize using a sequential circuit using your 1, 2, 3, 4 for using the four bit register and come combinational circuits you can implement it, you have to this is your first output, then this is your next output, then this is your next output. So you can easily go for digital circuit design in the state machine and these are the outputs you can do that.

And you know that you have to go for optimization of the gates and all. So there will be some gates will be involved and the area will be very high if you ever put two bit N is actually 100, 200 you know. So if you have a high number of bits okay, so your area over it of this counter or what you call test pattern generator if you want to be deterministic and you will be very, very high.

So what we do with that we used the pseudo exhaustive random disc controller generator so it will generate all random patterns between this, but whatever we do you want to select this K importance one from there. But now one important thing here is that so somehow we will see

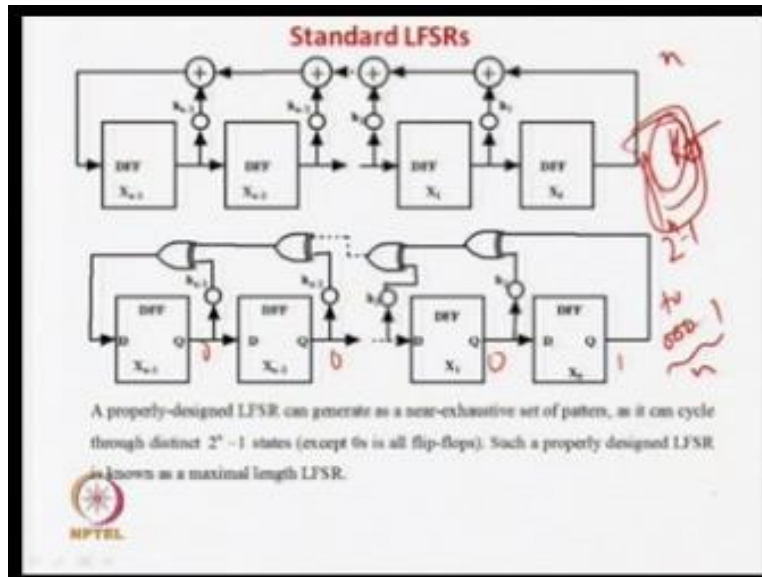
intelligently you can use the starting one so that your K inputs which are as meaningful for you come as fast as possible.

So they are basically such type of registers are called linear feedback shift registers is almost commonly used for all development base because we satisfy two conditions. So LFSR you should know the turn that is called linear feedback shift register this is the feedback over here, so they are used in case of BIST because they generate all possible, all random super random patterns all possible mean between 1s and 2^{n-1} if the pseudo random.

And D depends on the seed so by the controlling the seed you can, I mean make the register in such a way so that the patterns of importance after first and then your patterns unnecessary to our sockets okay. And there are two type of elements that we will study one-by-one, one is the standard LFSR and is the modular LFSR. So we will see that but the basic idea is that they are very small in area and secondly what, and secondly what is the another advantage.

The another advantage here is that it is a, the area over it will be very, very small that is one thing and it will generate all possible patterns in a pseudo random fashion, so you can take whatever is required and that can be changed by using the seed okay.

(Refer Slide Time: 25:02)



So we will see one-by-one. So there is two types and standard LFSR and modular LFSR, linear feedback shift register. So this is the standard LFSR, so why we call it a linear feedback shift register you can see the lot of feedback, so these are feedbacks over here and we studied that in a bit detail so that we can see what is basically happening. So you see these are the register so there is having a deep flip-flop.

We have already told that deep flip-flops are generally used as in the circuits so this is the D and Q so this is again D and Q and so forth. So you can always done the correction, now what you can see is that why you call it linear feedback shift register, because this is again D and Q, so the output of the last pattern you can call 0 whatever take 0, this is connected to this one okay, so all these process are nothing but a XOR gates.

So properly designed LFSR can generate a near exhausted state of patterns it can cycle to this one states except all 0s a properly LFSR is known the maximal length and it is available will come to that, nothing what we are doing. So what is the idea here, so the output of the first flip-flop the last it is not flip-flop it is connected to this, so there is the feedback. And you can and

this is H1 this is a switch actually and the switch or connection is there or the connection are there we will see about it.

So either this Q is connected to one input of the two input XOR gate. And one before flip-flop or the last flip-flop before the XOR if the other input is in this way, then the output of this XOR gate will be the here, the output of the second last flip-flop this one will be connected to this one. And so forth and the first flip-flop will have this one, so first flip-flop output, input will be having a feedback from itself as well as this can be the feedback of this one.

Now there is a very important thing over here, so this will generic architecture of a linear feedback shift register why you call the feedback, because you see this output is depending on this feedback, this I mean for this case or the input of the first flip-flop in that way $X-1$ depends on the feedback of all other flip-flops. But for the other cases you see they are directly connected, this output will be here, then this output will be here, this output will come to the input of the next one.

So this output will control this flip-flop will directly come to this ... and so forth. But the input of the first flip-flop will be written by the feedback from other flip-flop. And you can say that the linear feedback so if in the linear feedback shift register. Now it is not mandatory that you can see there are some Hs are there, $H-1-2..H1$ so they can be either 0 or 1. So what do you say that, if I say that $H=1$ that means what this flip-flop this multiply I mean sorry this XOR gate will be there and this thing will be there.

But if it can also be the case that they who do not require a feedback from this register or this register or this register. So if we say that we do not require a feedback from this KX - I mean what you call this $X1$ flip-flop then what we can do is that these things this will be not be there $H1$ will be equal to 0 so this is not there at all this feedback is directly connected to this one if you say that $X2=1$.

So your $H1, H2, H3, H4$ etc can be 0s and is. So if it is a 1 then this feedback will be there from that flip-flop and if it is a 0, then that feedback will not be used the other it will be the bypass of

that one. So by changing this H1, H2, H3 to Hn-1 so making them 0 and 1 we will get different types of LFSRs okay, so by and they will have some polynomial as we will see there is something called a characteristics polynomial for this one.

So whatever flip-flop or whatever other way so by changing this H1, H2, H and this one I having different values of 0s and 1s, if it is 0 that means this XOR gate feedback is involved of the H1 is 1 then this flip-flop is used in the feedback, if it is 0 then this flip-flop is not used in the feedback. So this will be a direct connection over here and this thing is not there. So by using this H values to be 0 and 1 you can get different configurations of this LFSRs.

So that is what we are saying a properly designed LFSR, properly designed means by using a proper values of X H0s and H this H values 0s and 1s, you can generate our LFSR which will generate a near exhaustive or I can also say exhausted test patterns as it can cycle from 2^{n-1} states 0 is gone there, such property in LFSR is maximum length so that is what we require. So what we require, so we require some say K test patterns to be generated out so we will see how it generates okay.

So now if you choose properly the values so let us say there are any inputs if you choose properly the values of this one, then you can generate 2^{n-1} to n bit all those patterns 1 to 2^{n-1} all these patterns can be generated thus in the exhaustive manner we call near exhaustive because 0 is not included. And it will be in a random fashion, and depending on the first seed like you initialize your registers in some values.

If you have a different I mean like you can start with a 0001 so this can be one starting point. So this is called the seed, so using different seeds you can get the different random and sequence of the patterns will change, but it will cycle to 2^{n-1} to all 0 and so forth or you can see it will start from your C and it can circle and the circling length 2^{n-1} including all zeros.

Now this is called the maximal length LFRS because it is full cycle, but sometimes we may not require full cycle staff because you may only required k important test pattern to be generator so using different configuration of this h1 and h0 you can also go for I mean what you it cannot be

maximal length LFSR so its cycle will be less than 2^{n-1} it does not generate all patterns in between 1 to 2^{n-1} .

But generally what we do is that we use maximal length LFSR because we want all patterns to be generated because we do not know which are test patterns that may be required so we do is that we require we generally use our maximal length LFSR and then we use C such way so that the important test patterns you require they occur first and then the unnecessary test pattern will occur after those in the cycle.


So one all your important test patterns are generated you can reset your plan you called this linear feedback register. So we will see this why an examples so things will be more clear and sometimes we want to if you can set this H1 and h0, h1, 2, 3, 4 and all this we can do it you very carefully and you can have a non-exhaustive, non-maximal length LFSR which you can design such way so that only the test pattern you require or regenerate by this.

So that why it gives you a lot of I mean flexibility and area over you can see nothing but only some sockets so at all these H will not be one actually some of them will be one and some of them will be zero so what is the area over it some flip flop and only some of the XOR gates nothing more than that.

(Refer Slide Time: 31:35)

Standard LFSRs

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

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


Why nothing more there is required because even only if you want feedback from the particular flip flop XOR gate is there otherwise the area is not there because the XOR gate is not present. Now this whole thing can be represented by a matrix so it always generated by matrix so what is there so generator in matrix we called x_{t+1} what is the value of the register in time $t+1$ is dependent on the time present value of matrix is matrix and present input value so x_{t+1} is its into h_t so this is your $x(t)$ that is present value and this your next value for all the flip flops and this is the characteristics matrix for t_s .

So we will see how it can be generated so you see what is the idea so idea here is that if you observe it very carefully so what is the idea the input of x_0 sorry, the output of x_1 . Similarly the output of x_{n-1} so you can say that $x_0 x_1$ you can see x_2 the output of x_2 you can say is controlling the input of x_1 and so forth, so you see what if represent if say that what is x_0 plus $t+1$ what is the x_0 t_1 will be actually equal to what is output of x_0 $t+1$ will be the input D so the output of x $t+1$ will be equal to actually x_t this one so whatever the value here will be in the time period $t+1$ will be value of D at time T .

Now what is the value D at time T is the output of x_1 at time t so you can say that x_0 of time $t+1$ is nothing but equal to x_1 value of time t , because of time t the output of x_1 is equal to the input of x_0 and at time $t+1$ is output will come here so this is the stuff, similarly it can be interrupted for all other except this first flip from the first flip are dependence on a lot of feedbacks, so that is why if I use the matrix so what will find out so you have to observe that this is nothing but a diagonal matrix is there.

So this the first column only one beat is one all others are zero this is diagonal only one and here actually have get the value of H_1 and $H_2 \dots H_{n-1}$ whether you want to have feedback or you do not want to have feedback. So now what is the value of x_0 $t+1$ you can see that equal to zero and zero this is equal to $0x_0$ this one that is $1.x_1(t) + \text{some } 0. x_2(n) \dots$ all will be zero. All this things will be eliminated and you will have only x_0 $t+1$ is equal to one that is one you can eliminate this x_1 t that is what is required these things are eliminate because of this 0 .

Now similarly you can easily see that if I want to find out what is value of this x_1 of $t+1$ that is this one so it is actually what this will be 0 so it is zero factor this one will be multiplied by this one so zero factor so this one will be have to be multiplied by this one so it will be $1.x_2(t) + \text{all } 0000$ will be there so x_0 $t+1$ equal x_2 t that is what so what is the output of this value at $t+1$ so x_1 $t+1$ it will be equal to what will be x_2 t so this value is coming here in the next flock.

So similarly you can find that for all this equalization holds now only you have worry about the flip flock because it comprise the feedback. So that we have to see once so this is the last flip flock you can see over here that you can see this is last feed flip flock here so what is the depend on it is depend on x_0 okay, and then plus if H_1 is there then only it will be H_1 if it whatever with the case right so that x_0 will be there this one x or with then x_1 of t kind of thing right.

So x_1 sorry it will be what sorry, the equation is something like this input is dependent on this output XOR with this one if H_1 is 1 similarly this output or with this output if H_2 is 0 similarly this one with exceed with this one if this is one and so forth if something is zero that will not be consider. So let us see we will come back so x_{n-1} that is sorry, the last flip flock $n-1$ is $t+1$ this one is equal to what it is x_0 t this one is there correct, this one multiplied by this multiple H . as

said nothing but XOR so this one is $H_1 \cdot x_1 t$ plus $H_2 \cdot x_1 t \dots H_{n-1}$ then your $x_{n-1} t$ that means what you see if somewhere x is 0 this H is 0 this is not used in the feedback then this part will be gone it will be only other way.

And you just looking at the circuit you can find that is the case so if even say we consider all these things are zero for this. This one is zero, this one is zero, this one we assumed that so only this thing is there so what will be x_{n-1} , $x_{n-1} t_{n+1}$ it will be nothing but x_0 of t this we have consider okay, and it will be only this one is 1 so it will be plus it is XOR H_1 that is actually one now 1 so eliminate this one, so now it is one so it is $1 \cdot x_1$ of t this is the case and all others are zero.

So it will directly go and feedback over here, but if you want to use another over here so you have to again put another XOR here so it will be again x_2 of t so wherever H value is equal 1 so you are using 1111 one more XOR gate, so actually there is big waves of very big XOR gate over connected over here and there are all inputs coming through this one. So, which ever flip flock wants to give a feedback to the first flip flock so that H has to be made one and it will be included in the equation.

But if you see if nobody wants to give the feedback then what happens then only if fast this feedback will only be there all others will not be there. So the output of the first flip flock will be actually exploit so this will be remain and all this will go that what has been reflected by this line so it is $x_0 x_{n-1} t_{n+1}$ is equal first will be so this why all as kept at one so whatever may be the keys if nobody wants to give feedback to the first flip flock actually last one the x_0 will have to give it so that will be there so x_0 will not allow will be there and this part will depend whether H_1 or H_2 was 0s or 1s.

So if first wants to give a feedback it will be one so it XOR or nothing but XOR so h_1 is 1 so it will be 1 so it will be there so this will give feedback. So $x_0 t + x_1$ XOR $x_1 t$ then all others may not be liking to give a feedback so it will be all 0 and maybe this H_{n-1} may want to give a feedback so it will be XOR then H_{1n-1} is 1 so you will have $X_{n-1} t$ so this is also going to a feedback.


So these are this matrix form also generators this or also represent this circuit. There are two representations of what you can call this LFSR standard LFSR.

(Refer Slide Time: 39:18)

Standard LFSRs

Leaving behind the first column and the last row T_n is an identity matrix; this indicates that X_n gets input from X_1 , X_1 gets input from X_2 and so on. Finally, the first element in the last row is 1 to indicate that X_{n+1} gets input from X_n . Other elements of the last row are the tap points $h_1, h_2, \dots, h_{n-1}, h_n$. The value of $h_i = 1$, ($1 \leq i \leq n-1$), indicates that output of flip-flop X_i provides feedback to the linear XOR function. Similarly, the value of $h_i = 0$, ($1 \leq i \leq n-1$), indicates that output of flip-flop X_i does not provide feedback to the linear XOR function.

This LFSR can also be described by the characteristic polynomial:

$$P(x) = 1 + h_1x + h_2x^2 + \dots + h_{n-1}x^{n-1} + h_nx^n + x^n$$


So now so we are not going into this very depth theory we are not going to be there is very in depth theory which is available so we says that then we can say just a minute so one more thing will say that also it can be represented by a function if you have this we can have function representation of this one so how do you write it.

So we say that FX is equal to 1 is always there and Xn is always there so if you have this fixed we can easily see how it happens so LFSR there is a characteristic polynomial we say that it is 1 and this xn- and the last bit is x1 so if it is link of the flip flop is n so there is another factor involve that is XL and H1x+, H2x²+... Hn-1 and Hx-1 and so on and so forth.

So wherever you are putting H=1 so this x factor will be there and our x² will be there and wafer it is not there H is 0 that factor will be not there, so the same matrix we can also represented by a what you can call a characteristics polynomial then there are lot of theories so we will give you

the reference so that you can see there is some reference to find over if what is the characteristic polynomial which gets this is the characteristic polynomial which is also define which can be derived on this matrix.

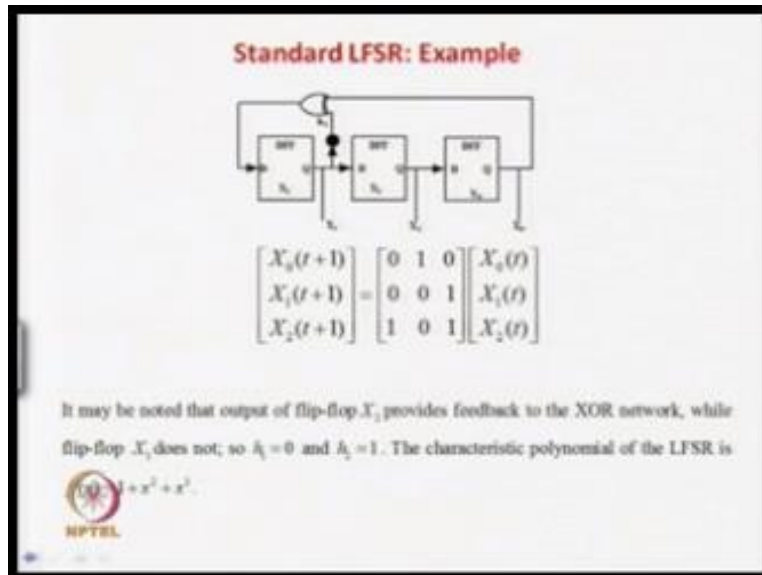
So if then there is lot of theories which you can check that even this characteristics polynomial this say something call prime characteristics per some I mean coat and quote of this things are there so we are not going into depth on all the theory because it will take you to a very depth analysis of this polynomial functions and all those things or characteristic functions and all so I mean so this is a minimal prime terms are there and so many other exotic theories there which are not going into but given this type of I mean characteristics polynomial this theory which will tell you whether this flip flop or whether this register is going to generate the exotic state of test patterns or not that is whether it will cycle through 1 to 2^{n-1} or not.

So I mean if you have I you choose the values of H in such a way then you can find out that characteristic polynomial is such that it will generate all possible patterns from 2^{n-1} to 1 and so forth but there are some if you choose the H in different way so the characteristic polynomial will change and then you can determine that it is not going to generate the extortive set of patterns you not cycle through extortive patterns so all those theories are there which can be easily determine by to get characteristics polynomial.

So depending on your require you said the H accordingly and in using the theory you can easily check whether it will rotate through this what you can call the extortive set of test patterns or not. So I mean this is a generally the English whatever we discuss about the matrix so it is written that I mean depending on H1 and so forth the flip flop may provide a feedback or may not provide a feedback so that what is by explained about this matrix written in text over there you can go through and but this is the representation of the characteristics polynomial.

So I mean so we are not going into the theory but by using the theory and the characteristic polynomial you can easily find out whether I mean what you can say whether these are it will go through in your exotic case or it will go for the near extortive generation and so forth. So we will start with an example.

(Refer Slide Time: 42:16)



So because there is a lot of theories involved in this LFSR so better we explain with an example, so let us see this is a LFSR right, so this is x_2 , x_1 and x_0 so this is there and you can see that this guy is not giving a feedback only this guy is x_1 is in this case is 0 because x_1 is not giving a feedback. But x_0 is must always give a feedback so you cannot do anything so this is the feedback is there so only x_2 is one this guy is 0 so no feedback is there.

So some persons has designed an LFSR in this way. Now what will be the matrix so you know that this is our case this is the diagonal matrix so this is the case this for this column the last bit will be 1 why this last bit is 1 because we always know that x_0 must give a feedback to the first flip flop, so this has to be kept as a 1 because this we have to derive x_2 this is multiplied by this factor this factor is multiplied by this factor so always we are going to get what you call the feedback, okay.

So this one is multiplied by this one you always keep this as a 1 so implies that $x_2(t+1)$ will be determined by x_1 this is the minimal thing that is required okay, now let us see it represents so this thing I mean the diagonal matrix this is the 1 last bit is 1 and this is H_1 and H_2 , H_1 is 0 you put a 0 H_2 is 1 so you put a 1, so you can easily see that what is the case so x_0 is $x_0(t+1)$ is

nothing but x_1 product of this one so x_0^{t+1} is nothing but x_1^t then you can say that x_1^t is nothing but this one multiplied by this one so it is only nothing but x_2 so this is nothing but x_2 and in the final case that is this last one this one is nothing but is equal to x_0^t plus this zero so these should not be there XOR with x_1 .

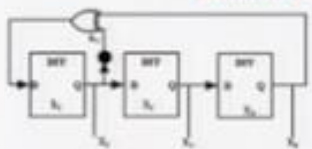
That is what you are getting x_2^t XOR with x_0^t so this only determines this one this is the case. Now this is the matrix representation also we can go for the what is then called this one this characteristic what do we say you also have to go for the characteristic polynomial based LFSR representation so this is the case so you know that this thing is there so $3t$ plus so x_3^3 will be there x_1 will be there so this one is $H_1 x_1$ plus $H_2 x^2$ is there so H_1 is 0 so this term is not there only this one is there and so this is the characteristic polynomial of this LFSR standard LFSR.

Now an important thing I have to tell that the theory is there to study this you find out what is this type a characteristic polynomial so it will tell you that this is the characteristic polynomial which can generate all exhausted test pattern, so that theory will not going to do that because and there is also a list I mean there if you look at the references some standard references already uploaded I mean in this course you can see the sight, so there are some standard text books so the standard text books is given the appendix or a list of characteristic polynomial with a exhaust in nature.

So this type of characteristic polynomial least are available with a proof to be in theory that will generate the exhausting test patterns except the all 0 of course. So you can select appropriately as you want, right.

(Refer Slide Time: 45:24)


Standard LFSR: Example



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

$$\begin{bmatrix} X_0 \\ X_1 \\ X_2 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ \dots & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

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

 HPTEL

So now we will go to the example, so in this case already that we take this one that is $1+x^2+x^3$ is already known to be generating our exhausting test pattern is so that you can easily go and find out that is in the least of the exhaustive accepting zeros of all patterns of the delay of this characteristics in that least of exhausted test pattern polynomial set though there is the least of a characteristic polynomial in these the character is that it will generate all possible patterns ,so this one falls in that can be followed in the least.

So you will accept the all possible patterns theory you can look over there, so this is your matrix already we have seen so is requirement where want to see that it works less so what is this I already was mentioning that you have to give a C. Why the C cannot be all zeros, so you have to give a seed, seed means it is the starting point, so starting point is you are giving a 1 over here you can see x_0 is 1, x_1 is say that it was 0 and it is a 1.

So you can said that reset your flip flop accordingly, because with by any many meaning means you can do that so it is that. So you can also start with a 111 that is not a problem, so any seed you can apply so, this seeds say for example I have told you so for example for testing I said details so forth very important test patterns are say this one, say 100 is important then you can

say 011 is sorry, so important test patterns are say 001 then you can say that 100, okay so important test patterns are 000 this one and say 111, okay now so what you will and what is my duty so what I will see is that if I start with this so this is start point so this is seed so this is start point so this is seed then first clock cycle this pattern is to 1 pattern is testing my faults.

Next deviation I will get this very good, so second pattern is done. Now this third pattern is not of requirement not my requirement, because I mean 111 so this will not tested anything so this will go blank okay, now what happens now this 111 is require and it is your three patterns are done so you can stop over your and whenever you require again, you can reset and come back.

So another two ways of doing it so one you can say the characteristic polynomial in such a way so that we generates all this pattern it is possible it may not always possible that it will generate only this patterns in this sequence. Then you have to soluble in a deterministic manner as you do it for a combinational circuit. So you can try out that you want to began a sequential circuit which will generate this, this sorry this, this and this that is only this, this and this.

So you will find out that it will take much more gates then a single XOR gate it will be much, much higher so you have to wait using a current optimal optimization say or queen matrix optimization these are the current state then if you apply a 1 then you go here and the next set it will here and so forth then you do go for this minimization of a disease 1d this is 1d, this is 2d this is 3d so three d3+ are they have to optimize the input for all this.

So the standard digital circuit design digital state machine design you can see you always know that the digital design that is the state machine designs are there so you have to optimize this. So this will be much more higher than a single XOR gate. So it will go by that way your circuit can design or implement to bring out this pattern this pattern and this pattern, but it will take much more much more number of gates then require.

So that is our constringe that we cannot have such a huge number of gates while we were generating a test pattern generator because the BIST area over here is required restrictions, so what you have doing instant you are using a this random pattern generator so it is very simple as

we will see only one XOR gate is there and we generate the near exhausting test patterns excepting the case of 000 that cannot we generated here you can see that you will hand you will have 000 only. All others are degenerated and for we require only 1 XOR gate our requirement is this one this one and this one.

So only which so we start with this seat okay, and only you have we get one in the sequence based in and all of us are very good for us. But if we start with this seat then is a problem your this is an important requirement you can say this is an important pattern, but then again you have to get all the junk patterns yet you will be go back to this.

So seats the lecture and characteristic polynomial is very, very good very, very important. So if your sometimes can have just of requirement of test patterns you can easily find out another characteristic polynomial which is generate only this patterns and your job is done. That is very good, but sometimes it may not be possible that so that case what we do we go for the exhausted case and then we actually put the CD in such a way so that the important patterns come fast other than the non important patterns.

So in this case there put as 00 so you see what is the next of the next situation what is going to do. So this X0 input will control this and so we will control this in the next time. So you know that what will going to happen so x1 will go to the value of this one sorry, so X1 will control this X0 and X2 will control X1 in the next deviation and X0 with feedback, this feedback will come to next this is cycle.

So X1 will control X0, X2 will control X1 and this X0 with all feedbacks are all control X2 so these all your matrix so this one will go this way and this last beat will depend on part it will depend on this one XOR with this one because this was not involved in this one so one XOR is 0 nothing but a 1, so this 1 you are going to get by x sorry this x0 with x2 so yo u XOR this guy and this guy you are going to get a 1, so this what the next pattern is generated.

So what is the next pattern now, so the next pattern when is 0 next pattern is now 000 and it is a 1 over here now. So now what is the next pattern over here you know that this one will control

this one and X_2 will control this one. This is the case and now how do you determine X_2 it will depend on the XOR of 0 and XOR of 1 which is again nothing but a 1, so this will all you get 000011 so now we are going to getting 0 and 1 correct.

So now again this will be the case and now this last bit is again depending on one XOR 0 so this is one, so now we are going to get a 100 so it is one then next one x_1 is 1 and this 0, so similarly you can find out the next pattern will be something like this so 0 XOR 0 is your something like this. This how this pattern will go on and what even find if you can study that just if go by this whatever I did so if you can go through we will find out that after 001 that is what X_0 is zero X_1 is one and this one so you will get repetition.

Obviously the repetition has to be there this there this one is going to there, this one is going to there and 1 XOR 0 is a 0 1 XOR 1 is 1 sorry, so you will get 1 001 and sorry, 001 next we are seeing going to just see what is after that so it is X_0 is 0, X_0 is 1 after this pattern what will be the case and this is one, so you know that it will be 0 over here it will be so what is the next pattern we are just trying to compute what is the next pattern after 001 so X_0 0 is 0, X_1 is zero and X_2 is one over here.

So what is the next pattern of this one. So you know that X_1 will determine this one zero so it will be zero over here then what is the next pattern over here so this is the case and this next pattern will be one over here and this you can determine 0 and 1 XOR with this one is again 1 so it is and we will get again 1 over here. So you can see that this pattern is repeated. So that is what I am saying so then again this pattern will be there and it will be lot of repetitions over there so what is the next emphasis was that if you start with this one so what is going to happen is that you are going to get all this pattern after that there is another pattern which is actually repeated.

So that means what in this is generator 7 patterns this is an exhaustive case so this polynomial has you already know that this pattern was exhaustive pattern, so exhaustive what you can say this polynomial was a I mean what you can called exhaustive kind was such it will generator an exhaustive set of patterns so it has generator all the 7 patterns starting from chip and after that there is repetition 001 you can see that there is repetition over this.

So that means what based on this it was generated this pattern and then another pattern will start over here, so that is after generating exhausting pattern one more pattern is repeating so in this expression was what you can called is exhausted polynomial so it generators all the patterns possible. So you can according you can decide on this seat and then you finally all your important patterns with up to here and after that you can reset.

So basically what we have achieved we have achieved or you can also think that many important patterns are here so you can then start with one as a seat and go above this way, okay in the very bad situation or you can say that that in the worst situation you can have something like so this is an important pattern, this is an important pattern, this is an important and so forth.

Okay, then you have start with another seat and see how you can achieve this pattern in better way so that more on that we will see in the next class, but today what was the main idea was that to show that just using one XOR gate and changing the values of I mean H1, H2 and H3 and so whatever so you can generator exhausted set of pattern so area requirement is only one extra XOR gate and by depending on this seat so if you start with this seat we this will be pattern that we can see, so you can try with different starting seats and your patterns will be different.

But again if you cycle over the exhaustive seat that is very important that it will go around this all the exhaustive patterns, but see that if you this set of patterns are important for you there are including this one and so forth so you start with this seat so very quickly you will get this coverage important set of test patterns and you can stop and go around with this. Similarly if you have some other important I mean your test patterns important in this one. Okay, so this one here is also repetition you can see there is a repetition over here so upto this only there is a thing after that it repeats, okay.

So after this 100 this one is a was a repetition so after that again if you apply we get another repetition was 011 so you start repeating from this end so 1, 2, 3, 4, 5, 6, 7 so after 7 these are repetition so this guy shows 100 this is 100 these are the repetition so after 7 it will all there will

be always are repetition that you can easily see. So this is the only exhaustive part this part is your 7 element 1, 2, 3, 4, 6, 7 after that it gets repeats then you can say it will be repeating.

So after this one we saw what was the pattern it was 011 these was the pattern we already saw and it will be repeating. So this whole guy will be again repeating that already we have checked. This and this one we have seen from here, so again this will be repeating nature. So this is the block of this one so depending on whatever is were important said whether you are important set lies over here or important set lies over here when choose the sheet accordingly and you can do it.

So another advantage is by choosing this sheet properly and properly there is characteristic polynomial you can use the small area over it pattern generator to generate the pattern of importance, but if you want to do it in a deterministic way that is I need a circuit which will generate only this 3 and not any other then you will be in a big problem. Because, you have to go for a final stage mission design of the digital electronic fundamental.

So, you will have to optimize the number of gates and all so, it will be much larger than a single XOR gate. So, we do not have that much area flexibility or that much area over it capability in this so you have to go by this approach. We will go for a linear feedback with register and then by choosing the seat properly wherever is our important test patterns are there which choose the seat accounting early and it your job.

But, the not the golden in life with not as been as this sometimes we may have patterns like this 11 gap then what is the case then we are at a lose then what is the idea then I mean at every alternative test patterns are not of use to us. So, we have to go around type test it will be larger. But, you could design a circuit which would all the generate this test pattern the area would have been larger. But, you could have done your testing this faster.

But, anyway we do not have the area flexibility. But, there are what is the option sometimes you can find out some characteristic polynomial and some seat so, that it will generate only this patterns or maybe one or two more than that but we always cannot guarantee. But, there is also

it is also maybe possible which you are not looking at in this course in them, but that also you can do offline study. But, here you have to but what is the emphasizing that you have understand that we are generating all the exhaustive patterns and based on the sheet we can generate the important patterns first and using a very, very less area over it that is only one exhausting.

So, this was about the LFSR and what is the importance in pattern generation that is random pattern an area this one, okay so only all zeros cannot be generated that is very obvious you can find out if I put all zeros then what is the case, so all zeros then one is zero exhaust zero is again zero, so this thing will keep on generating zero it will cannot generate anything else. So, all zeros is not allowed, so this is not allow generate. Oaky, now you has think that if these are 3 input patterns now if we think that the inputs are 1000 result. So, if you are to generate an exhaustive pattern like this, so this your BIST you require 1000 flip flops and some in the worst case 1000 XOR gates or even less than that.

But, if you want to generate a counter, but the pattern generator that the deterministic pattern generator like a digital circuit finessed machine design this is only 1000 bits then its area over it will be extremely high, because for each of the inputs of this flops it will generate a different combination of circuit which will generate the virtual patterns there will be optimization and there will be at least few gates for each of the input of the flip flops which will be very high in number.

But, in this case we require only a few number of XOR gates to do this so with this we stop our lecture for this day and in the next class on this module we are going to see another type that is standard LFSR was there so we will see the next type of LFSR and its impact which will call the modular LFSR so, what is the advantage and what is the disadvantage and then what you will also see about the response compaction that why will you require a response compaction what is the advantage and all so that will be covering in the next lecture which will complete our discussion on BIST, thank you.

**IIT Guwahati
Production**

**Head CET
Prof. Sunil Khijwania**

CET Production Team

Bikash Jyoti Nath

CS Bhaskar Bora

Dibyajoti Lahkar

Kallal Barua

Kaushik Kr. Sarma

Queen Barman

Rekha Hazarika

CET Administrative Team

Susanta Sarma

Swapan Debnathi