VLSI Design Flow: RTL to GDS

Dr. Sneh Saurabh

Department of Electronics and Communication Engineering
IIIT-Delhi

Lecture 43
Built-in Self Test (BIST)

Hello everybody, welcome to the course VLSI Design Flow RTL to GDS. This is the 34th lecture. In this lecture we will be discussing built in self-test or BIST. Specifically in this lecture we will be looking at distinguishing features of BIST based testing and then its architecture and then we will be looking at random test pattern generation and how test response is analyzed in BIST based testing. In the earlier lectures we had looked into ATE based testing where ATE stands for Automatic Test Equipment. Now ATE based testing is very efficient, efficient in terms of that because it allows a high degree of automation and also the quality of testing in ATE based testing is very high because the fault coverage that we get in ATE based testing is pretty high.

So ATE based testing is very popular. However ATE based testing has some disadvantages and what are those disadvantages? The first is that the cost of testing is high. Why is the cost so high? Because it requires expensive automatic test equipment. So the fixed cost is high for ATE based testing.

Additionally the volume of data that is required in ATE based testing is pretty high and therefore the test time is also high and therefore the cost of testing becomes higher in ATE based testing. Another disadvantage is that the ATE based testing can be done only in a production environment. So after we have done the fabrication of a die we carry out ATE based testing. But it lacks the capability of doing field testing meaning that once we have fabricated our die we have found that die to be good and then we integrated it into a system and that system is being used and at the system level if we suspect that whether our integrated circuit is working properly or not so testing of the integrated circuit in the field or when the integration of a chip has been done at the system level that is not possible for ATE based testing. And another disadvantage of ATE based testing is that it requires probes, probes to apply the test patterns and also to read the response out of the chip.

But these probes have inductance and capacitances and therefore these we cannot carry out at speed testing for use in ATE based testing. But some types of faults, for example delay faults, require that we do testing at speed or at a higher clock frequency and those kinds of testing are difficult to perform using ATE based methods. So as an alternative another type of testing which is known as built in self test or BIST is also commonly used in integrated circuits. Now in BIST what we do in BIST we have test oriented additional hardware and software features inside our integrated circuit. So inside our chip we have some special hardware and software which will do the testing of an IC internal.

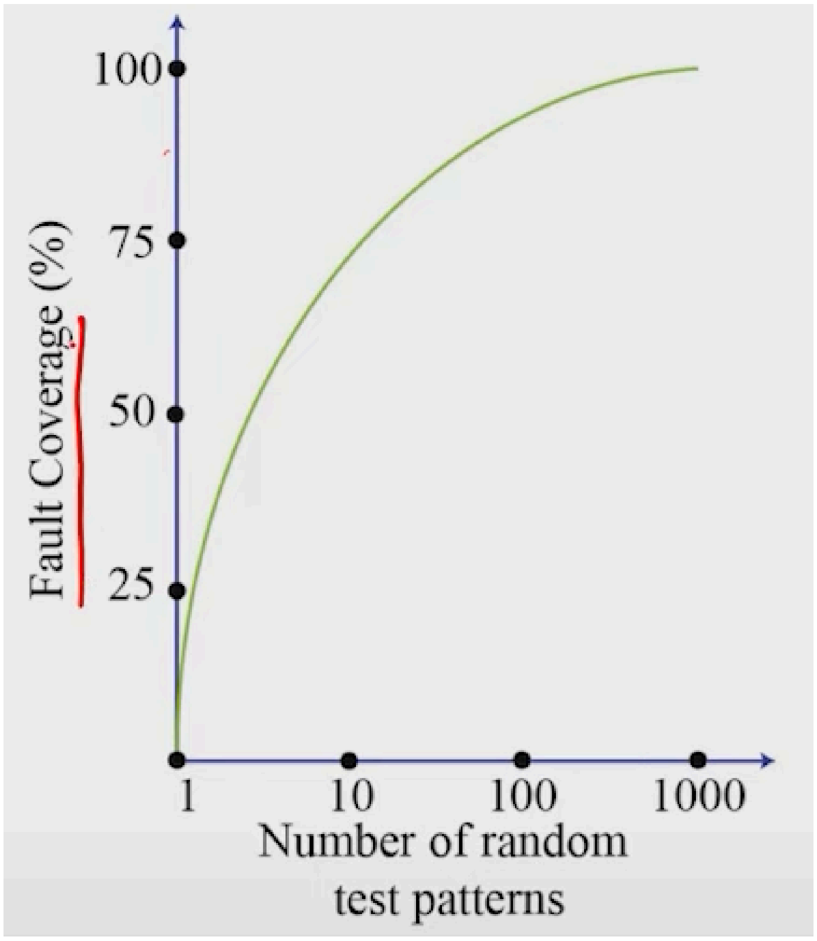
So given an IC the IC will do its own testing or test their own operation using the infrastructure of BIST or built in self test. Now what is the advantage of it? It reduces the dependence on the external ATE for testing. So all the test infrastructure that is required for built in self test is contained within our design or integrated circuit and therefore external is external patterns need not be applied and not be and the response need not be read out of the chip. But those kinds of the generation of the test patterns and the analysis of the response are done internally within our circuit. And these kind of testing or BIST kind the testing using BIST method is possible in field meaning that once we have integrated our IC into our system and we suspect that whether our integrated circuit is working correctly or not whether it has it is it has become faulty or not we can run the BIST test test infrastructure and the and the BIST method will tell us whether our circuit is working fault free or it has having some fault.

So the testing can be done in the field also if our circuit has the best infrastructure. Now to build the best infrastructure within our design it will require some cost in terms of hardware meaning that it will occupy some area and so on. So we want the best infrastructure that we embed within our circuit that is not taking a heavy cost in terms of area and in the infrastructure of the chip itself. Then if the best infrastructure itself takes a large amount of area say on our die then probably it is not cost efficient and therefore the best method employs a special kind of method so that the area overhead of embedding the best infrastructure is limited and what are these extra features of best infrastructures. So the first thing is that it uses or it contains a pseudo random pattern generator inside the chip.

So within our chip there are random numbers and the random pattern generator and these patterns are actually used to test our chip chip or so these test vectors are dynamically generated and used while testing. So what do we save in this case because of using pseudo because of using pseudo random pattern generator. We save on the area or we need not need not store a voluminous set of packed test vectors for example suppose we want to apply one million test vectors . Now if we want to apply one million test vectors one method could have been we save all those test vectors in our chip in our in our in our in our chip in some ROM or so on. But then that ROM itself will be too big and therefore that the testing infrastructure will take so much area that it will not be efficient .

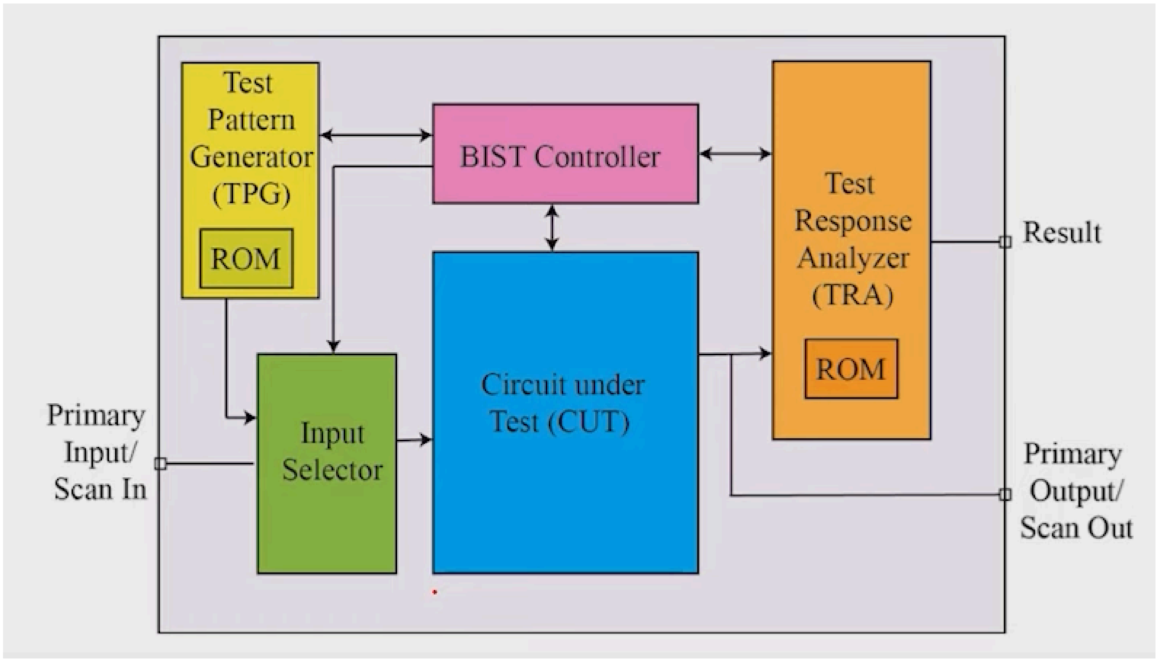
So by using a pseudo random pattern generator we are avoiding storing test vectors within our chip but on the fly or dynamically it can generate those test patterns and be and those test patterns can be used in the best method. And the second thing is that in the best method we do what is known as signature analysis. So we do not compare the exact response that we get from our circuit with some golden response. Rather what we do is that we derive a signature out of all the responses and compare only the signature obtained from our chip compared to the golden signature. And therefore as a result what do we save? We save storing or we avoid storing voluminous expected response inside the chip.

So the test infrastructure is fully contained within our chip but we are not storing test vectors, we are not storing the test response. What we do instead is that we have some infrastructure to generate pseudo random pattern patterns and also we do a kind of signature analysis. How these are done we will see in the subsequent slides. Now in this best testing as we have said that it uses a random pattern generator or the test patterns are random. Now an obvious question is how good are random test patterns in detecting our faults, say stuck at faults.



Now to assess that we can plot the fault coverage fault coverage of stuck at faults and versus the number of random test patterns. So note that on the x axis the number of the x axis is logarithmic. How many kinds of circuit will see this kind of behavior or this kind of curve? Now what can we infer from this curve? We can infer that the stuck at fault coverage rises to 100 percent so this is 100 percent logarithmically and reaching 90 percent of the coverage or say 85-90 percent of the coverage is pretty easy. Say it will require hundreds of patterns or in that range.

So what this curve or this observation tells us is that even if we use a random test pattern then 85 to 90 percent of the mistakes can be covered and therefore the best best method or best best testing relies on random test patterns. So it will not be able to cover 100 percent of the faults but we can still get some fair idea of whether the faults are there in our circuit or not using just a few numbers of random test patterns. Now let us look into the architecture of BIST like how a chip in which the best method has been implemented looks like. Now this circuit under test is basically the main circuit part of our circuit which is delivering the functionality and all other parts are basically showing the infrastructure which we need to add in our circuit to implement the best best method. Now what are these extra infrastructures that we need to add.



So the first one is the test pattern generator. So the test pattern generator generates the test pattern for testing and it can also contain a small size ROM to target some specific faults in our circuit. So most of this test pattern generator will be a kind of random test pattern generator but to target specific faults we can also use ROM to store some test patterns which we can also apply to our circuit and achieve a better coverage of the fault in the best best method. Now depending on whether we are in the best mode, whether we are testing our circuit or whether the test circuit is performing normal operation there is an input selector. So whenever we want our circuit to be tested in that case the input will be selected from the test pattern generator otherwise input will be coming from the primary input or say scan.

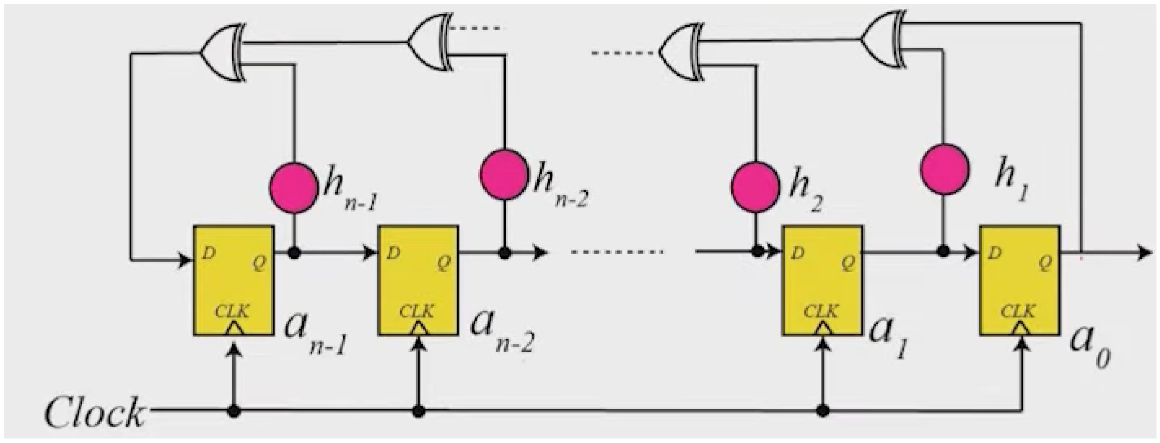
So the input selector is a kind of multiplexer and it will be controlled by the best control. So when on receiving the instruction from the best controller the input selector will take the input from the test pattern generator and it will apply to the cut or circuit under test. Now given the input the circuit under test will produce an output. So the cut may have some fault in that case that the best infrastructure must be able to detect that fault. So the output of this circuit under test goes to test response analysis.

So a test response analyzer generates signatures for the test response and it compares. So given the sequence of responses that will be received from the circuit under test, the test response analyzer will generate a signature and then compare it with the golden signature which is stored in its ROM. So ROM contains only the golden signature; it does not contain the exact response that we expect from the cut . So it only contains the ROM containing only the golden signature and therefore the size of the ROM can be much smaller. So given a large sequence of outputs that will be produced by the cut the test response analyzer will somehow process those in that information and convert it into a signature that is the test signature .

That test signature and then it compares with the golden signature which is stored in the ROM. If the test signature and the golden signature match we say that the circuit the cut is not having a fault and that result will be produced here . And in case the circuit the cut was having some fault then the signature that will be derived for the sequences produced by the cut that will not match or that is not expected not to match the ROM and the or that is not to that is expected not to match the golden signature stored in the ROM and in that case the cut has the fault in cut has been detected and that output will be produced at the result . So all these operations of the the best based testing is controlled by this best controller   this is a best controller it basically gives instruction to the test pattern generator that generate the test pattern then it gives instruction to the input selector that select input from the from the test pattern generator rather than from the external world and then it tells tells the test response analyzer to now now process the information coming from cut and produce a golden signature as produce a test signature and compare with the golden signature and produce the result at the output whether that the cut is having a fault or not. So all these operations are controlled by this best controller and the strategy for overall test control is the most difficult part of the best .

So it is the it is the responsibility of the designer to design this best controller there are tools available that helps us in designing this best controller but this is the most difficult part that we might need to design an FSM which controls this operations in the proper sequence and then produces the correct results or the expected result at the output . So the designer needs to design this best controller carefully. Now let us look into how we can generate test patterns or random test patterns in based based testing . So we typically use linear feedback shift registers to generate random test patterns. Now why do we use this kind of circuit or why do we use a linear feedback shift register or LFSR? The reason is that it creates or it is able to produce a random test pattern that is repeatable meaning that if we say that the LFSR is given a starting value or seed then from that seed it will always produce the same sequence sequence in the next clock cycle and so on .

So the the test pattern that will be produced by LFSR will be repeatable and the and LFSR is a kind of circuit or it is a kind of circuit which can produce the random test patterns very efficiently and the area penalty that is required to implement LFSR that is very small and that is why we use LFSR in in based so that we can we can get a a a large number of random test patterns with the minimal area penalty. Now what does a LFSR look like? So in this figure I have shown a special type of LFSR which is known as standard LFSR . Now in this LFSR if you look carefully then this is the same as a shift register . So whatever is the value at Q it is going to the next flip flop whatever is going to Q is again going to the next flip flop and so on . And so this is a kind of a shift register. The only difference is that it has the first flip flop or the flip flop whose output is labeled as a n minus 1.



So this flip flop is getting feedback from other state elements . So there is feedback and the feedback consists of a logic gate or XOR gates . So we can tap the output of any flip flop to provide the feedback . So these pink circles which are shown here indicate whether we are tapping the output of the nth state elements in the feedback path or not. If we are tapping it then we are saying that H n minus H i will take a value of 1.

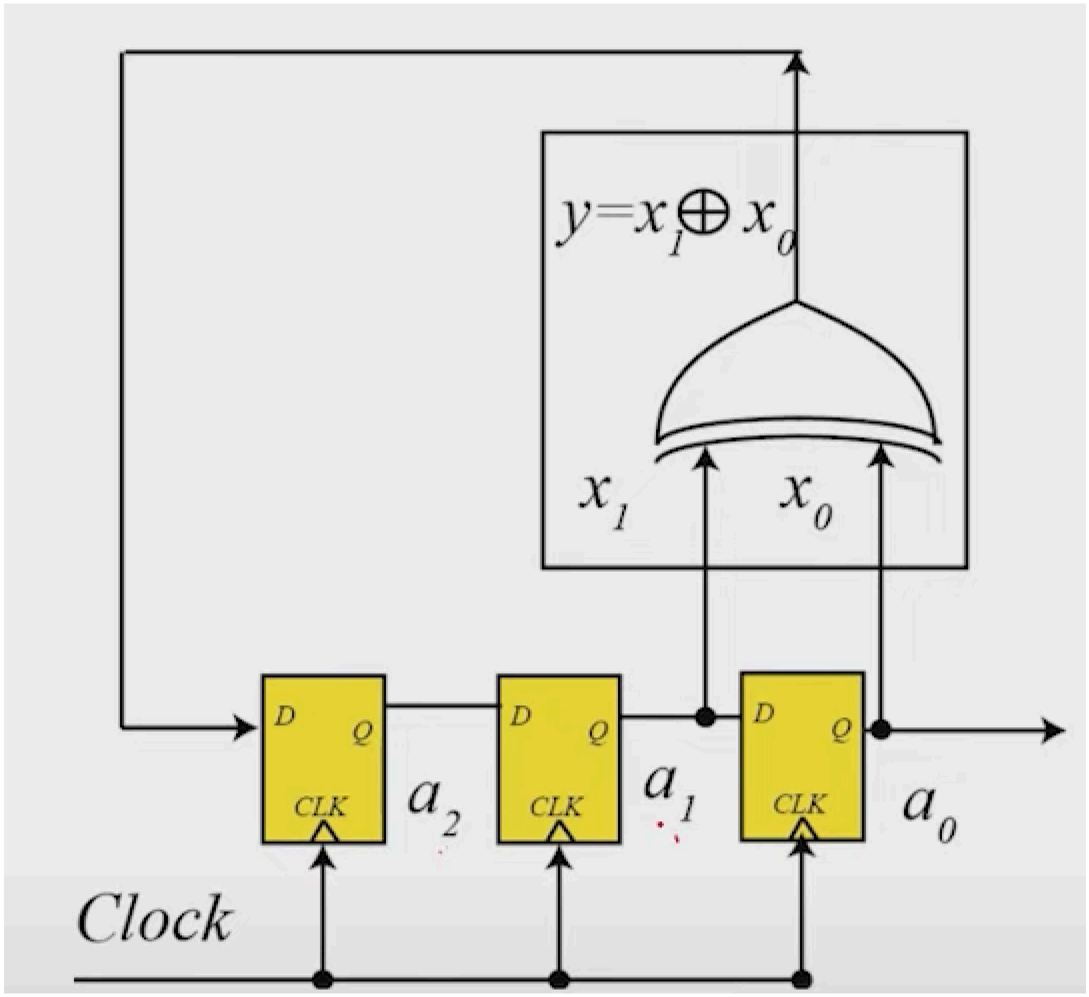
$$a_{n-1}^{k+1} = a_0^k \oplus h_1 a_1^k \oplus h_2 a_2^k \oplus \ldots \oplus h_{n-1}\, a_{n-1}^k$$

If we are not tapping then we are saying that H n minus 1 is taking a value of 0 . Now how can we understand the functionality of this LFSR? So let us assume that in this LFSR that is shown here there are n state elements; they are marked as a0 a1 an minus 2 to a n minus 1 these are n state elements or flip flops . So let us indicate their output or the value at the q-bit at the kth clock cycle as a_0^k a_1^k a_n^k minus 2k and n minus 1 k . And then we can predict what will be its value in k plus the 1th (k+1)clock cycle . So it is easy to see because it is a kind of shift register kind of structure.

So whatever will be the value of a_1^k it will go to the next flip flop in the next clock cycle. So a_0^k plus 1 that is the value of a 0 in k plus 1th clock cycle will be the same as the value of a 1 in kth clock cycle . So this will be true for all the flip flops a 0 to a n minus 2. But what about a n minus 1 or the last flip flop . This flip flop the D pin is connected to the logic of XOR gates .

And therefore whatever is the D at the D input that will become the value in the next clock cycle whenever the clock comes . So the value of the nth flip flop or a n minus 1 which is indicated here for this flip flop in the k plus 1 clock cycle will be the XORs of a0 k the first one a0 k and then a1 k assuming that h 1 is there then if it is being tapped then we have h 1 element h 1 this term will exist . And similarly if s2 is being connected or this XOR gate exists then s2 will be there and will be taken so this term will contribute the term h2 a2 k and so on . So the last flip flop is taking the last flip flop and a n minus 1 flip flop is receiving the feedback from all other state elements . And this feedback or what will be the value this will be the value that will be produced by this XOR this logic consisting of XOR.

Now these things will become more clear if we take an example. So let us take an example of an LFSR. In this LFSR there are only three state elements: 0 a1 a2. So first let us write its next state equation . So for whatever the value was a of a 1 in the current clock cycle that will go to a0 in the next clock cycle .



So a_0^1 plus 1 is a_1^k. Similarly what was the value in a 2 in kth clock cycle that will go to a 1 in the next clock cycle. So we will have a 1 k plus 1 is equal to a2. Now what about a2? Now a2 is receiving at the D input the XOR of a_1 and a_0. So we have a_2^{k+1} meaning that the value of a2 in the next clock cycle is whatever we have value of a0 in the kth clock cycle XOR with whatever the value we have at a1 in the kth clock cycle . So this is the next state equation which we can write for this LFSR.

$$a_2^{k+1} = a_0^k \oplus a_1^k$$

Now what will be the state diagram or how the states will be traversed for this LFSR. So we can draw the state diagram using these equations . First let us assume that the initial state of the LFSR is what is shown here 1 0 0 . So here we have 1, we have 0 here and we have 0 here . Now in the next clock cycle what will happen? Now in the next clock cycle this 1 for a2 will go to a 1.

So we will have 1 here and this a 1 that will go to a 0 and will have 0 here . And what will be the value of a2 in the next clock cycle? It will take a value which is XOR of these two . So XOR of 0 and 0 0 XOR 0 is equal to 0. So in the next clock cycle what will be the value of a2 it will be 0. So if we start with 1 0 0 the next state will be 0 1 0 .

Now if we assume that the next state is 0 1 0 similarly we can compute. So if we assume that we have value 0 1 0 then in the next clock cycle what will happen? This 0 will come to this, this 1 will go to this sorry 0 is going to this so here 0 this will go to 1 and what will the value of a2 it will XOR of 1 and 0. 1 XOR 0 is equal to 1. So we will have 1 here.

So in the next state it will be 1 0 1 . So this way we can see how the states will be traversed for an LFSR . Now an interesting case happens if all the states are 0 initial states suppose all of them were 0 then what will be the next state? Now this 0 will come here will have 0 this 0 will come here will have 0 and what will a2? a2 will be 0 XOR 0 is again 0. So the next state of a2 will also be 0. So if it is in a 0 0 0 state it will always get stuck and that is what it is shown . So a 0 0 0 state is all 0 state is a kind of a forbidden state for an LFSR because if it gets into that it will get stuck into it .

So however if we choose the feedback logic properly then we can get a long sequence of states in an LFSR . So if there are say n elements n elements in the or n state elements in an LFSR then we can have maximum 2 to the power n states. Out of that 0 0 0 is forbidden . So one state goes away. So if we design our LFSR properly then we can get these many numbers of states in a random manner.

So note the states that we are getting here then the 1 1 1 0 0 0 1 0 then 1 0 1 1 1 0 and so on. It is looking something like a random pattern . It is not a kind of a counter or something . It is generated in a random path in a random way . So for an LFSR the test patterns generated by an LFSR have most of the properties of random numbers .

So this is one of the biggest reasons why we use LFSR for generating random test patterns in for in the in based based based testing . And if it is desirable that we have a long sequence of test patterns before it is repeated . So it is desirable to have a long sequence of test patterns before it begins being repeated. So that good fault coverage is achieved and how can we do it? We can get a long sequence of random test patterns if we choose number of n say maybe more than say 20 or 30 in that order then the if we have a sufficient number of state elements in the in the in the LFSR and also if we have if we choose the feedback function properly then we can have long sequences of of long sequences of random test patterns generated by an LFSR . However some faults may not be covered by random pattern generators .

Now for example just to give a sense of it suppose there is an or gate and it is consisting of say 5 inputs . Now if we want if we want to test that whether the output is stuck at stuck at 0 or not we or rather stuck at 1 or not then to detect this stuck at 1 fault we must have a 0 at this output and to have a 0 all the inputs must have a value of 0 all the 5 inputs must have a value of 0. Now if we try to get these patterns of all 0s in a using a random using a random pattern generator then what will be its probability? Its probability will be 1 divided by 2, raised to the power 5 or 1 divided by 32 . So if by random test random test patterns we may not be able to detect some kinds of fault . So it is very difficult to test some types of faults or test element components in some types of circuits and therefore we can augment the test pattern generator with some deterministic test vector also .

So we do not rely totally on random test pattern generators, but we have a few test patterns which are derived using the ATPG tool and those test patterns are stored in the ROM . So it will be so and those test patterns should be applied also along with random test patterns in this base testing. Now what will be the advantage of it? It will help us achieve high fault coverage with a small number of test vectors 1. So now how do we analyze the response that we obtained from our circuit under test. So if we are applying 1 million test patterns suppose there are 1 million test patterns and that there are say 100 outputs 100 outputs.

Now if we want to store all of the responses in our circuit in our circuit itself then how much memory do we require? So we will require 10 to the 100 million or 10 to the power 6 into 10 to power 2 these many numbers of bits and if divided by say 8 will get the number of bytes that we need to store. Now if we think it will be or if we consider this number it will be around 12.5 megabyte of memory. Now this is too much memory if we want to put it in our circuit and it will require a large amount of area . And therefore storing the response of our circuit within our circuit within our circuit which we are designing is impractical .

So what we do best is that we use some method of compaction. So what is compaction? Compaction is a method of reducing the number of bits in the circuit response and how do we do compaction? We do it by using a statistical property of a circuit and using that statistical property we derive a signature, usually a number computed for a circuit from its responses . For example suppose there were 100 million bits now out of this 100 million bits that is in the output response we just count the number of ones in those bits for the golden circuit for the circuit which is not having any fault and treat it as a golden signature . And then we carry when we do the best based testing we do a signature analysis. We say we compute the signature of the good circuit and store it in the RAM and sorry in the ROM in our best and then compare it with the signature of a potentially faulty circuit.

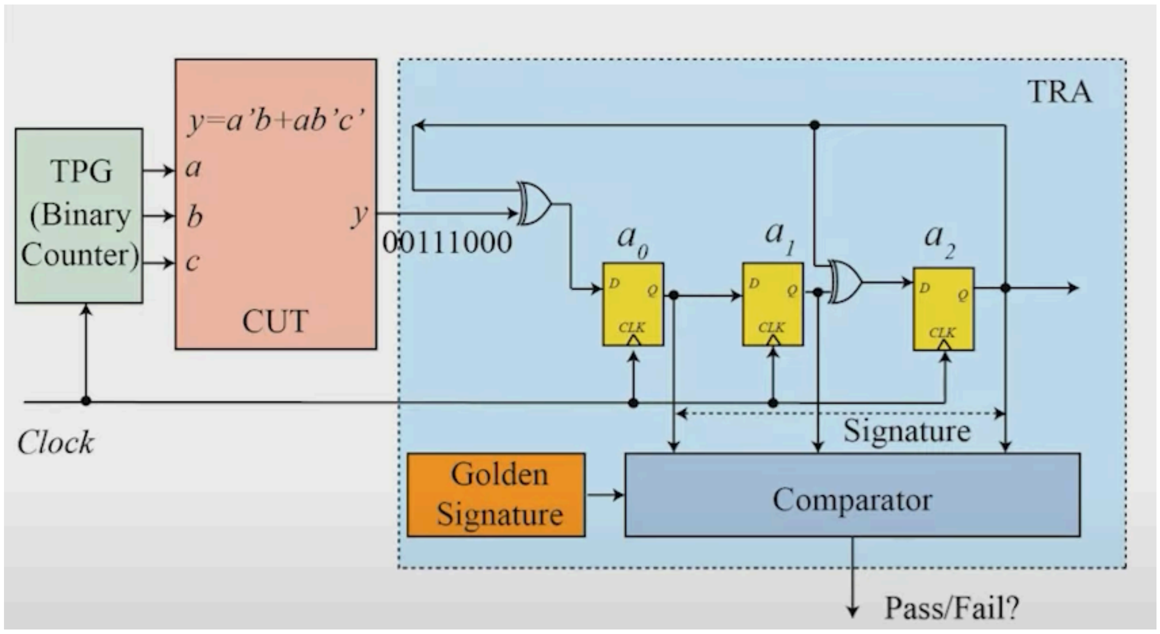
So while doing testing we compute the test signature. So we compute, for example, the number of ones that we are getting in 100 million bits that is the test signature and we compare the number of bits that are expected to be won in the golden circuit or a fault free circuit. If those numbers match we say that the circuit is good else it is faulty. If the golden signature matches the test signature then the circuit is assumed to be good it is assumed to be good else it is assumed to be or it is else it is faulty . Now the desirable thing is that the signature of a good circuit and faulty circuits must be different but that is difficult to attain . So and if they are not the signature of good circuits and faulty circuits are not different then what we say that aliasing has occurred.

So during testing it is possible that the signature of a good circuit and faulty circuit match . If there is a good circuit and say number of ones is matching in the good circuit and number of ones are matching in a faulty circuit . If that matches and then we say that aliasing is occurring and in that case even though the circuit is faulty we will not be able to capture that fault or detect that the circuit is faulty . So why does this aliasing occur? So the aliasing occurs because we are doing compaction and this is a lossy compaction this is not a lossless compression of the data it is a lossy compression meaning that out of say 100 million bits we have found a signature a compact signature but we have lost a lot of information . We have lost all the information about what are the correct sequences of zeros and ones that will appear and therefore there is a loss in information and that loss in information leads to aliasing. That is not a good situation and we should avoid it .

So now how to compute the signature? So one method can be ones counting as I described just count the number of ones across all circuit responses . So but what will be the problem with this one counting of course there will be a lot of aliasing . So this can be aliasing because different permutations can yield the same signature. Suppose there were say 4 test vectors t1, t2, t3, t4 the golden circuit was producing the output say 0 0 1 0. So the number of ones in the ones counting method in this case is 1 .

Now suppose our circuit was faulty and it was producing a response which is a permutation of it. For example it was producing 1 0 0 0 . Now in this case the number of ones is still 1 . So the responses are not matching but the number of ones in the response is matching the fault free circuit and therefore we will consider these two circuits as fault free which is wrong and this occurs because of aliasing .

So counting is not a good method of computing the signature. So how do we compute the signature? So one of the techniques that we use in based testing for computing the signature is known as use with the help of modular LFSR. We again take help of LFSR for example there is another type of LFSR which is known as modular LFSR in which the XOR gates are between two flip flops . So let me show you an example. So this is an example of a modular LFSR and in this case what we are in this is the modular LFSR including the clock signal .



So this is the modular LFSR. In this case what we are doing is that one of the XOR gates input from our cut is the circuit under test . So in this case what happens is that the circuit will produce a sequence of 0s 1s and that will go to the modular LFSR it is called modular because this XOR gate is between two flip flops in the in the between two flip flops rather than it in the earlier example that we saw that was standard flip flop sorry standard LFSR in which XOR gate was external to the to the to the shift register . But here it is internal to the shift register and it is known as modular LFSR. So in this case the input will come the sorry the output produced by the cut will be going as input to the modular LFSR and it will go on computing computing the signature and after n clock cycle it will get a signature or the values at this these state elements and that will be considered as the test signature . Now this test signature can be computed for a fault free circuit and that for that is considered as a golden signature and that is stored in the ROM of the test response analyzer and after a clock cycle when all the test all the test patterns.

So TPG is the test pattern generator so test pattern generator is generating random test pattern which is going as input to this cut and after all the test pattern had been have been have been given to the cut and the cut has produced the output and it has gone through the LFSR the modular LFSR a test signature will be generated and that test signature is compared with the golden signature if those those signatures match we say that it is passing if they are mismatched and mismatched then we say that the the the cut is having some fault . So this is a structure which is known as a single input signature register . So why do we use this kind of infrastructure or why do we use modular LFSR in test response analyzers? The reason is that in this case the aliasing probability can be kept lower . So the chances of aliasing is lower in this case compared to say once counting and therefore this kind of infrastructure is widely used . So in this case another point to note is that we have to start with some seed value .

We load these LFSRs or or the state elements with some initial state and then we start analyzing the response . Because the behavior of an LFSR will be deterministic if we start with a seed value a known deterministic seed value and therefore we must load the LFSR with a known seed value before starting the operation of this or this test response analyzer. Now let us look into what are the advantages and disadvantages of best testing. The advantage is that it reduces the cost because we do not need expensive automatic test equipment and we also need not store test patterns because we are generating it using test LFSR . And then we can do a speed test for you in the best best method because we need not interface external signals for testing.

And the other advantage of best is that we can perform testing in the production environment. We do not need expensive ATE and after our chip has been integrated at the system level we can still carry out the best best best test . Now what are disadvantages of best best best testing. So the first disadvantage is that it requires additional silicon area for LFSR for test response analyzer and best controller and so on. And the area of our circuit will increase and as we have seen earlier the yield can come down and also reliability can decrease in a circuit in which best is implemented. There can also be performance loss if the best infrastructures or added infrastructure comes in the critical path of our design .

And the best infrastructure will need some pins to indicate whether now we are going into the best testing and also some pin or external port which will indicate whether the best testing has passed or failed and so on. So the best testing can require additional pins or ports in our design and therefore it may require bigger packages . So these are some of the disadvantages. However we still use best in our industrial designs and the biggest advantage of best is that it allows us to do testing in the field .

And therefore best is very popular. Now if you want to know more about the best I will suggest that you can go to this or you can refer to this textbook. Now to summarize in this lecture what we have discussed is the best architecture and what are the special features that we need to implement or we need to incorporate in our design to implement the best testing. Now with this lecture we have covered all the topics that we wanted to cover related to design for the test. From the next lecture onward we will be looking into physical design. Thank you very much.