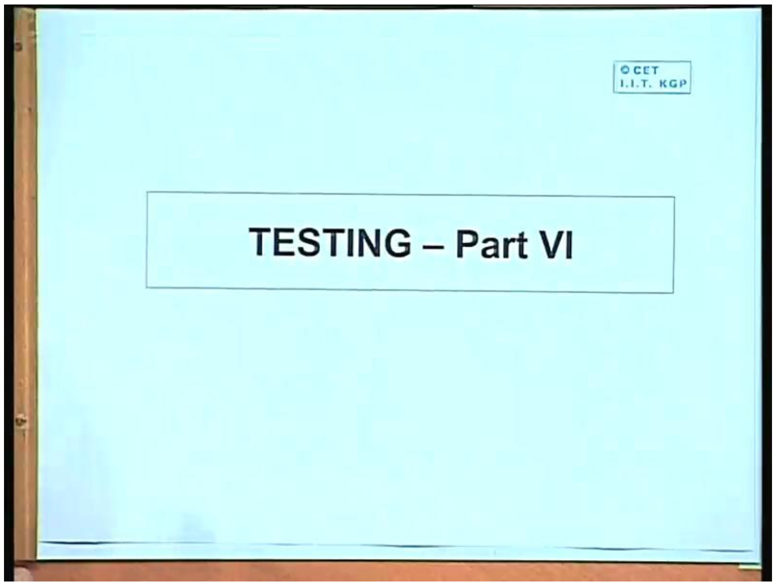
Electronic Design Automation

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

Lecture No #35
Testing - VI

We have talked about LFSR based zero random pattern generator using um using weight of "0.5" and any arbitrary weight power of 2 in our last lecture.

(Refer Slide Time: 01:20)

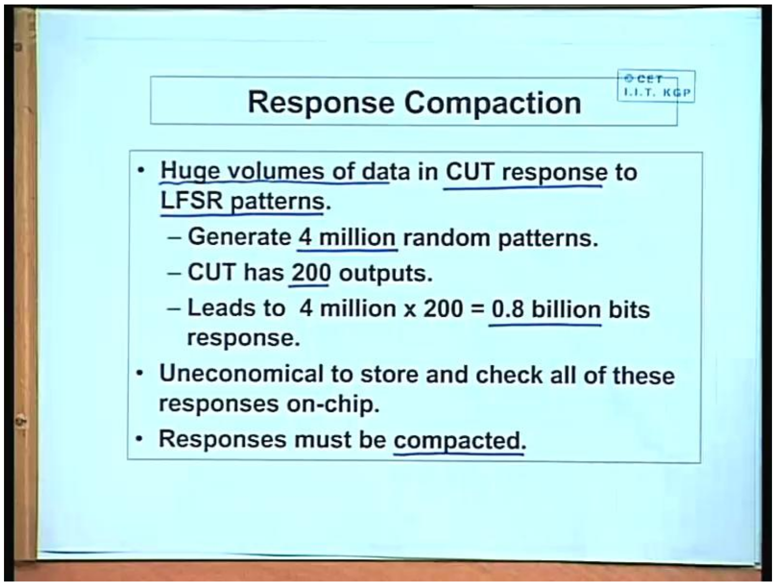


©CET
I.I.T. KGP

TESTING – Part VI

Now in this lecture we would be talking about the next thing we need to do in build in self-test namely, how to compress the response, response compaction. So before actually telling you how you do it using a LFSR first introduce some terminology and some basic concepts.

(Refer Slide Time: 03:02)



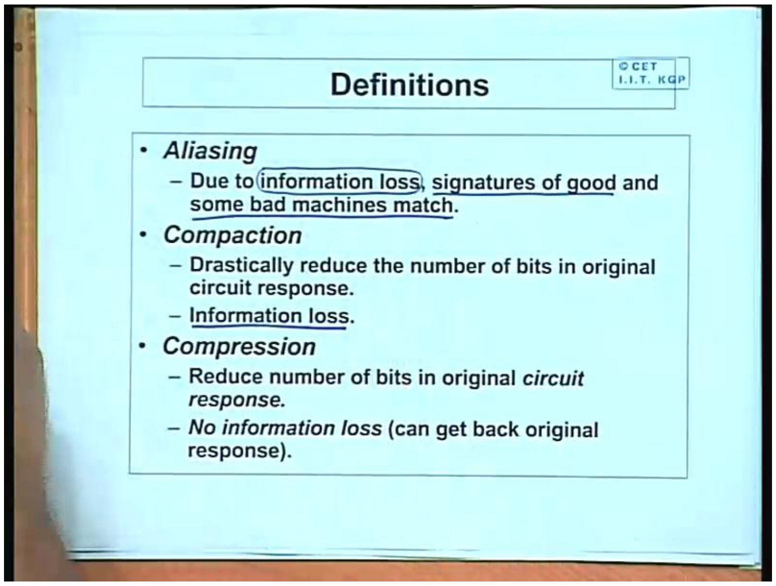
Response Compaction

- Huge volumes of data in CUT response to LFSR patterns.
  - Generate 4 million random patterns.
  - CUT has 200 outputs.
  - Leads to 4 million x 200 = 0.8 billion bits response.
- Uneconomical to store and check all of these responses on-chip.
- Responses must be compacted.

But first let us see what this response compaction actually involves okay. Now you recall that in a built in shift environment we are feeding the CUT circuit under test using a large number of LFSR patterns; typically in the order of 1000s. So the CUT output response generates huge volumes of data. Just an example, suppose using LFSR we are generating 4 million random pattern which is very typical. And the circuit has 200 outputs. So the amount of output data which gets generated is about “0.8” billion bits. Now this is clearly uneconomical to store this pointed billion bits and compare them individually with the corresponding fault free response.

Of course this would have been the best thing if you could have done that. But since it involves such a huge volume of data we cannot afford to do this. So it is mandatory for compacting the response in some way before comparison. Of course compaction should be search that our confidence in the quality of testing should also remain high. You should not be that we are making a cross compromise and something by compaction we are trying to remove the problem of storage but the quality of testing is going down. That should not be irricase also. So let us see what is done in practice. But before that we try to give some definitions.

(Refer Slide Time: 03:28)



Definitions

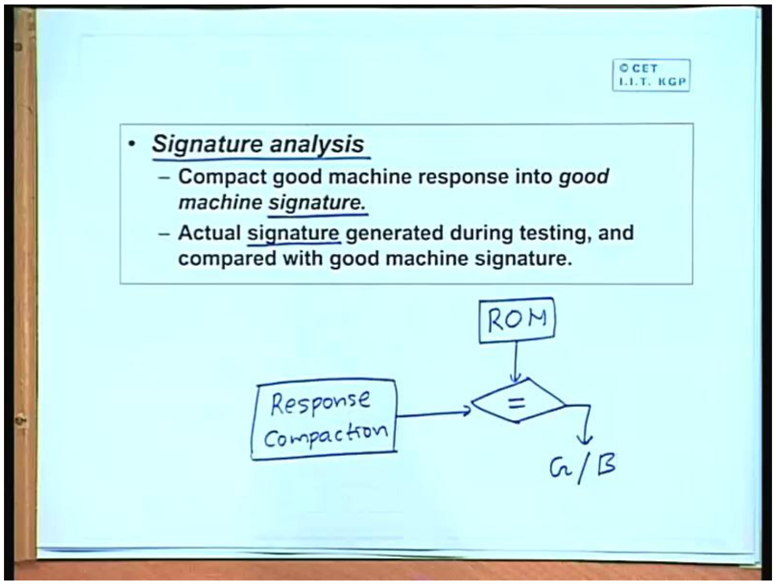
- *Aliasing*
  - Due to information loss, signatures of good and some bad machines match.
- *Compaction*
  - Drastically reduce the number of bits in original circuit response.
  - Information loss.
- *Compression*
  - Reduce number of bits in original *circuit response.*
  - *No information loss* (can get back original response).

See we are doing some sort of a compaction from you can say bigger or larger response we are trying to compact it into a smaller one. So whatever algorithm you choose there is always a chance that two different responses may get compacted into the same so called signature. This is something which is called aliasing. Signatures of good and some bad machines may match because we are doing compaction. Compaction means we are leading to some information loss. Like a 1000 bit data you are compacting into a 16 bit data for example. Naturally we are incurring information loss. Say a 1000 bit data 2 to the power 1000 possible combinations.

Now a 16 bit signature has only 2 to the power 16. So naturally two patterns can always match to the same signature they can match. So there will be information loss and hence aliasing. The process of compaction involves a drastic reduction in the number of bits. Since we are reducing the number of bits naturally this involves information loss. Compaction is used in built in self-test. But we have another related term compression but compression is used in a different context. Compression also tries to reduce the number of bits in the response but using some well defined algorithm so that the original response could be restored by using a reverse algorithm.

So there is no information loss. Like in a computer when we use compression program like compress or win zip or something. So you know that we can again come back to the original one by using a decompression algorithm. So compression decompression is something different. We are trying to compress or reduce as for as possible. But we are keeping sufficient amount of information so that we can also perform the reverse mapping in the in some latest stage. So we do not lose any information later. So actually it is not compression that we are interested in we are interested in compaction.

(Refer Slide Time: 06:02)



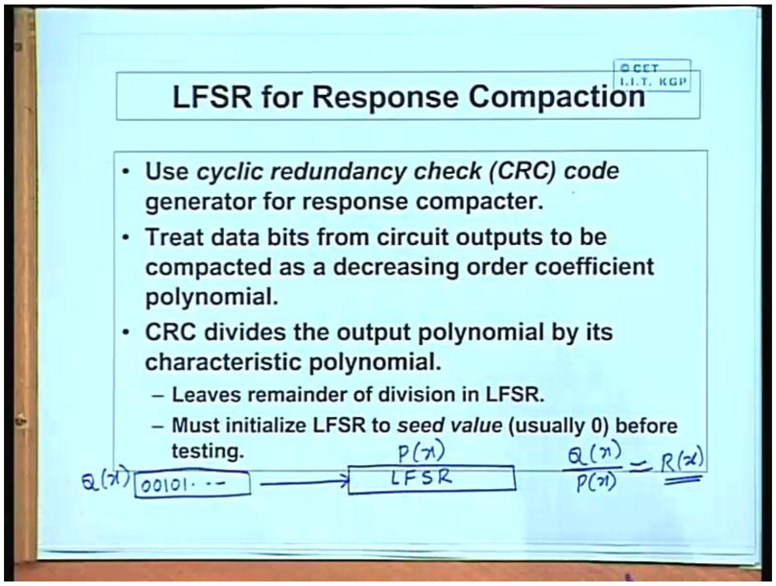
© CET I.I.T. KGP

- *Signature analysis*
  - Compact good machine response into *good machine signature.*
  - Actual signature generated during testing, and compared with good machine signature.

ROM
Response Compaction
=
G / B

Now the method which is used in BIST is traditionally called signature analysis. This says that you compact the good machine response into a small value which is called the signature. Now during testing you again compute the signature and compare the signature with the good machines signature which is stored in a ROM. Say in a ROM you stored the good machine signature and using a response compaction block. You generate the signature during the actual testing. These two values you can compare and you can report if it is good or bad. This is the basic idea. Now LFSR in the signature analysis and response compaction these has been traditionally used much before this testing came into pin.

(Refer Slide Time: 07:14)



LFSR for Response Compaction

- Use *cyclic redundancy check (CRC) code* generator for response compacter.
- Treat data bits from circuit outputs to be compacted as a decreasing order coefficient polynomial.
- CRC divides the output polynomial by its characteristic polynomial.
  - Leaves remainder of division in LFSR.
  - Must initialize LFSR to *seed value* (usually 0) before testing.

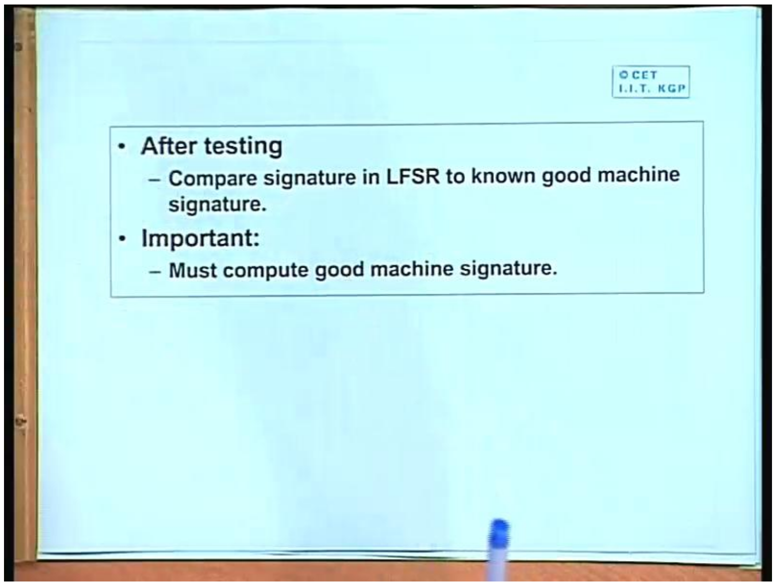
$$Q(x)\ \boxed{00101\cdots} \longrightarrow \overset{P(x)}{\boxed{\text{LFSR}}} \qquad \frac{Q(x)}{P(x)} = \underline{\underline{R(x)}}$$

In fact if we look at this linear feedback shift register for response compaction, well this compaction uses the same technique which has been used since long time. Well you must be familiar with the cycle redundancy check code which is used in a wide variety of application like your diss subsistence input output networking. So wherever you need to check for some errors bit errors you use CRC checks some at the end of the message. So that you can recomputed CRC after receiving and compare if the match okay. So LFSR for response compaction what they do that is exactly similar to this CRC check code generation. Now the why CRC operates I am not going in the detail. See you have an LFSR. Now this LFSR realizes a characteristic polynomial okay. Now you feed the LFSR with a stream of bits as 0 1 pattern okay. Now this input bits stream you can treat that the bits or forming the coefficient of a polynomial Q x.

If P x is the characteristic polynomial of the LFSR, then what you do is that, you are actually dividing the polynomial Q x by P x and generating the remainder polynomial R x. R x is the remainder okay after division. So whatever is a remainder polynomial that is treated as the CRC code or the signature in the present context. So after all the patterns have been apply the way the LFSR response compactor is designed automatically whatever data or remains here at the end, this is the remainder R x. But in order to do this, initially the LFSR must be initialized to all zero.

But here, there is no harm in putting 1 0 because well some data is also coming from outside. So there is no fear that the LFSR will get stuck into the all zero pattern like it was for the pattern generation case okay. So what you do that after the testing is complete?

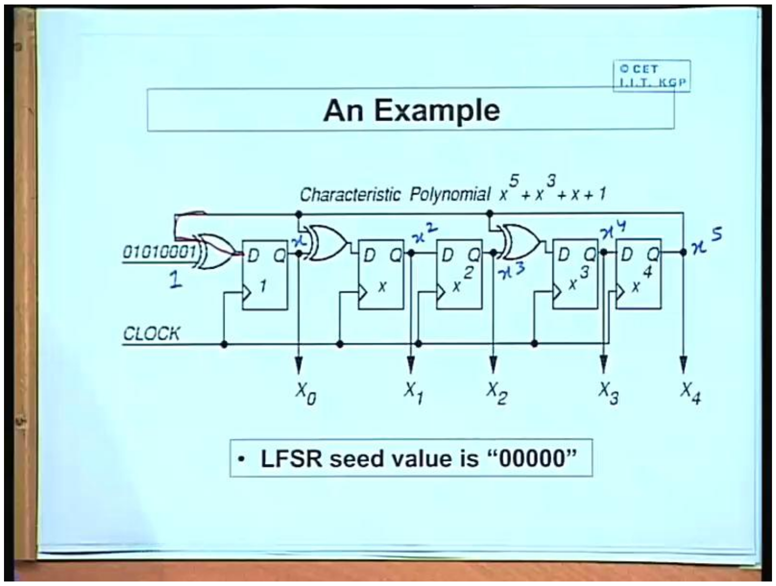
(Refer Slide Time: 09:50)


© CET I.I.T. KGP

- After testing
  - Compare signature in LFSR to known good machine signature.
- Important:
  - Must compute good machine signature.

You complete signature whatever is they in LFSR to the known good machine signature. Of course you must compute the good machine signature through simulation before and stored in a ROM okay. This is the scheme. So now let us see how using LFSR you can compute the CRC check word or the signature. It is fairly simple of course LFSR can be designed in a number of different ways.

(Refer Slide Time: 10:18)



An Example

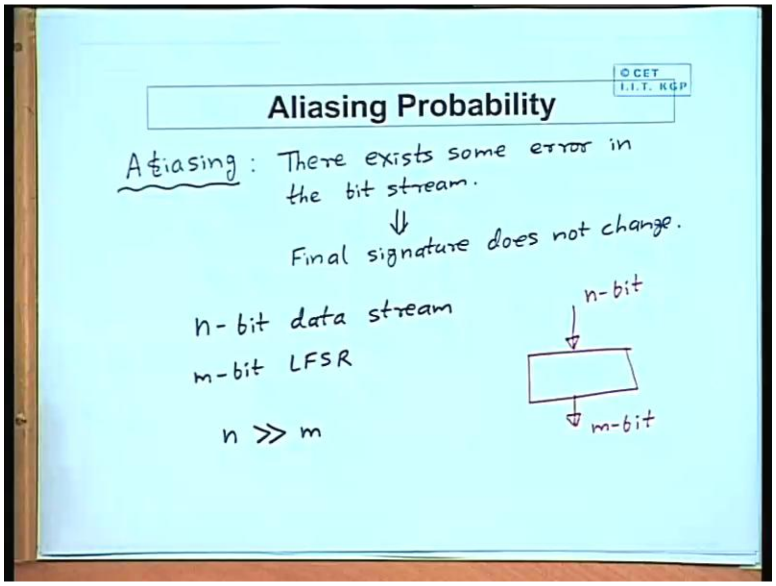
- LFSR seed value is "00000"

This is an LFSR with internal XOR okay. So here I am showing a LFSR which realize this characteristic polynomial x to the power 5 x cube x and one these are the coefficients. Now we look that from which points you have taken the outputs. This corresponds to x to the power five here x 3 power 4, this 1 x to the power 3 this one x to the power two and this 1 x. Of course the input this is treated as one. You see we have XOR of course the output will always to be fed back. So x to the power 5 will always be there. In addition you have connections from x cube x and one. So these are the terms which are there. So given any characteristic polynomial you can construct the corresponding LFSR like this.

Now the only difference from pattern generator is that in a pattern generator you did not have this XOR rather this line was directly connected to the input. But now you have an additional XOR in the input stage where the input bit stream is being fed of course a clock is there. And the LFSR is initially loaded with the all zero pattern. So now the circuit responses are coming you are applying the clock and whatever response remains finally here that is the remainder polynomial and that is the signature. So the concept of signature compression or response compaction using LFSR is very simple in concept. You can use a very simple structure like this LFSR to do both pattern generator and response compaction.

Now since the structure of the pattern generator and response compaction are so similar, now you can understand why you can easily built a structure like that BILBO, built in logic block observer. That was a general purpose structure which depending on a requirement you can configure it as a test pattern generator or the response compactor okay. But first now let us try to analyze some issues related to the aliasing probability. That we are compressing a big stream into an LFSR signature where number of bits is equal to the number of flip flops in the LFSR typically 16 or 32. So what is the price you are paying or what is the chance that fault may occur. But we are unable to detected the faulty signature is the same as the good signature. So let us try to make a simplified analysis of the so called aliasing probability okay.

(Refer Slide Time: 13:25)


Aliasing Probability

Aliasing : There exists some error in the bit stream.

$\Downarrow$

Final signature does not change.

n-bit data stream

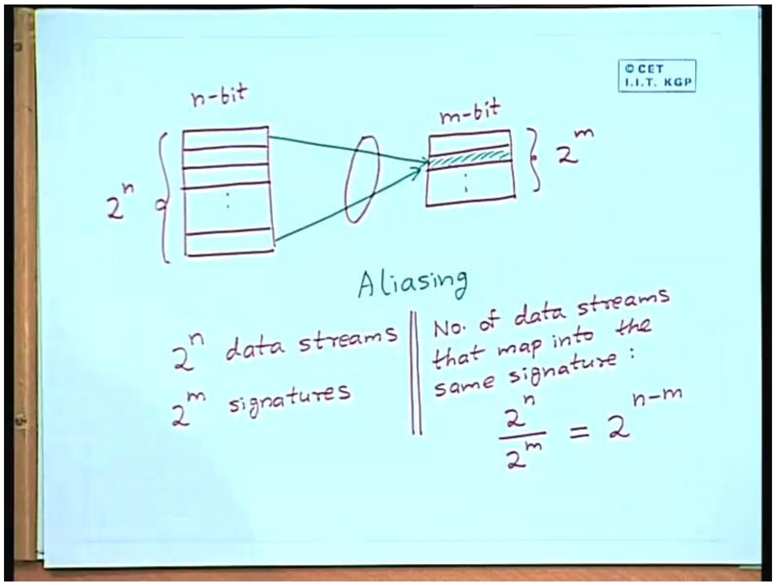
m-bit LFSR

$n \gg m$

©CET
I.I.T. KGP
n-bit
m-bit

So aliasing probability as I said it is a probability that a LFSR taken place but we are unable to detect that error. So aliasing, actually means that there exists some error in the bit stream. But even in the presence of this error the final signature does not change which means that we are unable to detect this fault. Now let us try to do a simplified analysis of this problem lets say that we have a n bit data stream and we have a m bit LFSR typically of a most practical purposes the val of n is much larger compare to m. Because n can be of the order of milli arms and m can be of the order of 16, 32 or 64 maximum okay. Now here let us try to see what does this mean, we are trying to compress an n bit pattern into an n bit LFSR. So we have response compactor block were at the input we are feeding and n bit data as the output you are getting a$_m$ bit compress response. So what does this mean this actually means is that.

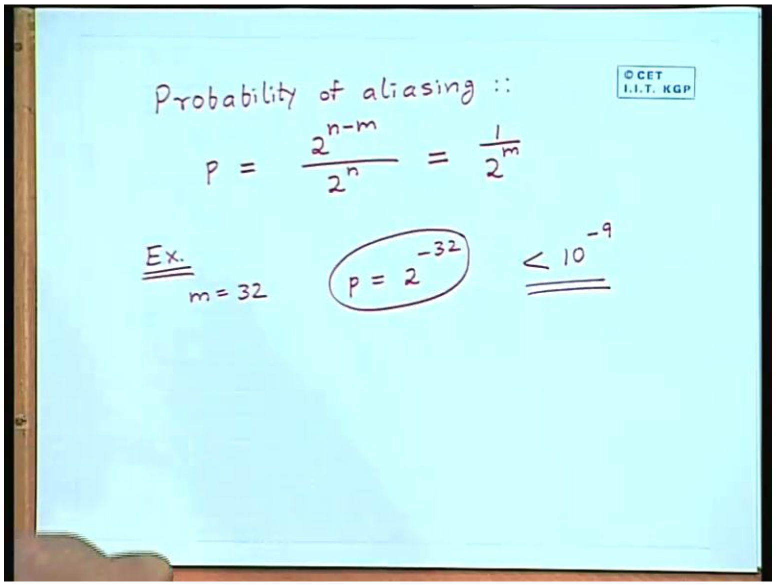
(Refer Slide Time: 15:27)


©CET I.I.T. KGP
n-bit
m-bit
2^n
2^m
Aliasing
2^n data streams
2^m signatures
No. of data streams that map into the same signature :
$\frac{2^n}{2^m} = 2^{n-m}$

Well the input was n bit okay. Now in n bit we can potentially have 2 to the power n possible combinations. And similarly for the LFSR for m bit we can have 2 to the power m possible bit combinations. So it is always the there is a chance that several of these 2 to the power n patterns might get mapped into the same n bit pattern out here. This is what we term as aliasing. See if this is the fault free signature for example, then if two patterns map into the fault free signature, then means some of this patterns will also be the faulty pattern one of them will be the good one, the other will be faulty. So even if this kind of faulty takes place we will not be able to detect it okay. So let us see that what is the chance or probability of this thing happening. So what we have just said is that we have two to the power n possible data streams and we have two to the power m different signatures okay.

Now from this we are trying to n find out number of data streams that map into the same signature. This will be 2 to the power n by 2 to the power m assuming well here we are made a cross assumption. We are assuming that the patterns are uniformly distributed to respect to where they map okay. It is not that more number of patterns are mapping to this signature and few numbers of patterns and mapping to some other signature no. We are assuming that there is a uniform distribution. So number of data streams which can map into the same that means I am talking about the cardinality of this which can map into the same signature will be approximately equal to 2 to the power n minus m okay. So this is factor and from this we can try we can we can find out the probability of aliasing.

(Refer Slide Time: 18:39)



Probability of aliasing ::

$$p = \frac{2^{n-m}}{2^n} = \frac{1}{2^m}$$

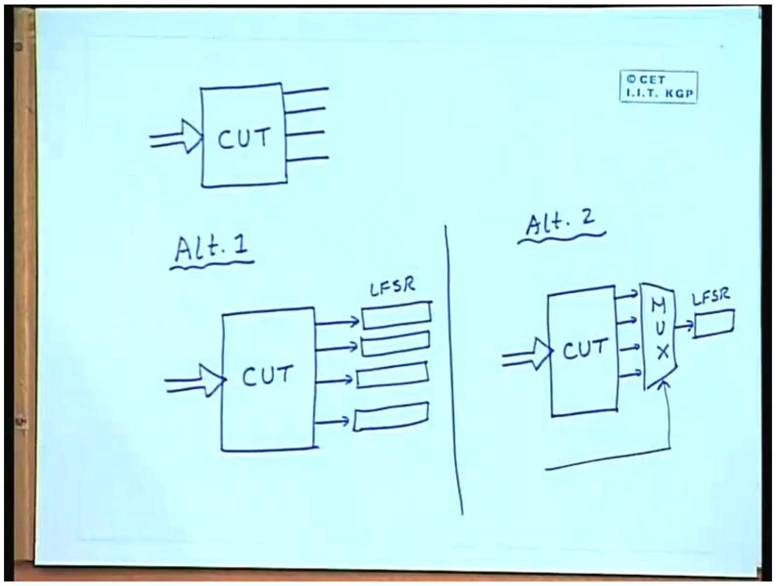
Ex.

$m = 32$ $\quad \boxed{p = 2^{-32}} \quad \underline{\underline{< 10^{-9}}}$

Now we can find out the probability of aliasing as follows. P equal to number of input patterns which map into the same signature just what you have just calculated number of input data stream that map into the same signature 2 to the power n minus m divide by total number of patterns. This is the probability of aliasing okay. Now this if you see it becomes equal to 1 by 2 to the power m. So you observe one interesting thing that well of course we are made some assumption like uniform distribution of the patterns. But the result we have got is that we have got a probability which is independent of the number of bits in the input stream. This depends only on the size of the LFSR.

For example if m equal to 32 then your probability becomes 2 to the power minus 32. 2 to the power minus 32 means, what this is less than 10 to the power minus 9. So you can see that using a 32 but LFSR we can get an aliasing probability which is sufficiently small. And because it is sufficiently small, we can get a fair good amount of confidence in the you can say in the chance that we are detecting an error when it occurs with respect to this scheme. Because of this small probability that we get out here of aliasing, people use this with fair amount of confidence okay fine. Now there is an issue. Suppose we have talked about how we can compact bit stream using an LFSR. But suppose you have scenario like this.

(Refer Slide Time: 21:02)

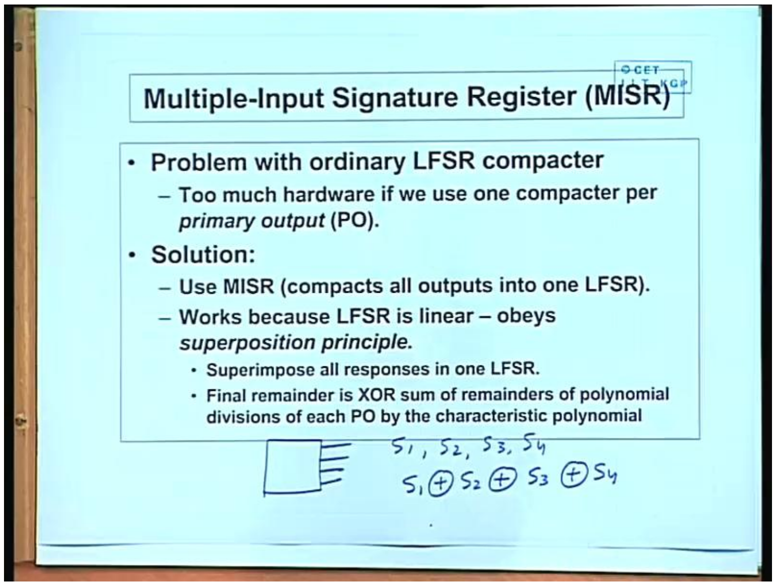


You have a circuit under test these are the inputs. But instead of one output say there are four outputs. So here what we do if we have one output you can use straight away an LFSR out there and do it. So here you can have two alternatives alternative one. So alternative one is something like this. Here your CUT its circuit under test you have the four outputs and to all the four outputs you connect a signature compressor compactor. These are all LFSR's. So if there are if the k number of outputs you will be using you will be using k number of LFSR for response compaction. Now here the drawback is that you are requiring to use so many LFSR's.

But the advantage is that you can you can compress all of the outputs in one go. So you will have to apply the pattern only once. The other alternative is that you can somehow save on the LFSR. This alternative two is something like this. This says that you again have your circuit under test here. You feed them with inputs and the four outputs you first fit them to a multiplexer. You fit them to a multiplexer and the output of the multiplexer you feed to an LFSR this is the LFSR. Of course the MUX will be controlled by the test controller. Now here you can see that by selecting the MUX inputs one at a time you can compress one of the circuit outputs at a time in the LFSR.

So the advantage is that you are saving on the LFSR one LFSR will do. But the prize you are paying is that the testing time is going up. Well for testing the circuit you will have to apply this patterns all these patterns four times. First time for output umber 1, again you repeat it for output number 2, again for output number 3 and output number 4. So the first alternatives request more hardware but less time this alternative request less hardware but more time. So we have to do something where we can well we can get an acceptable solution with less amount of hardware and also less amount of testing time.

(Refer Slide Time: 23:55)



Multiple-Input Signature Register (MISR)

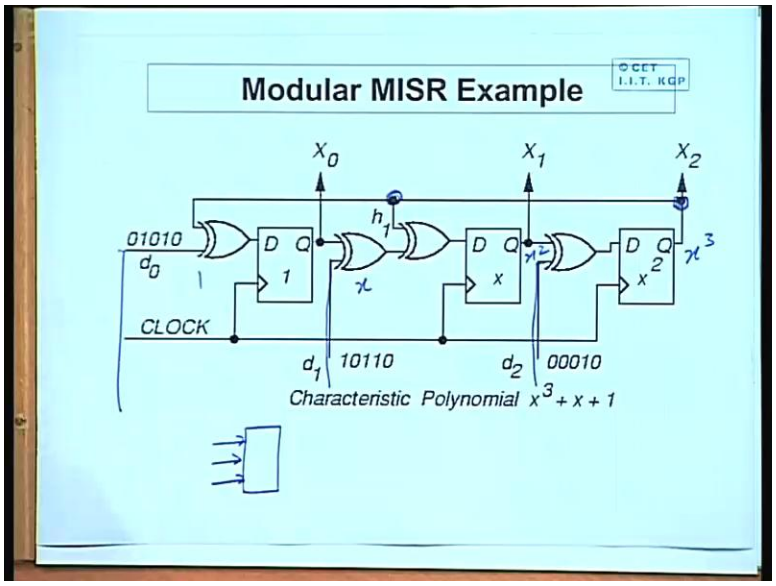
- Problem with ordinary LFSR compacter
  - Too much hardware if we use one compacter per *primary output* (PO).
- Solution:
  - Use MISR (compacts all outputs into one LFSR).
  - Works because LFSR is linear – obeys *superposition principle.*
    - Superimpose all responses in one LFSR.
    - Final remainder is XOR sum of remainders of polynomial divisions of each PO by the characteristic polynomial

S_1, S_2, S_3, S_4

$S_1 \oplus S_2 \oplus S_3 \oplus S_4$

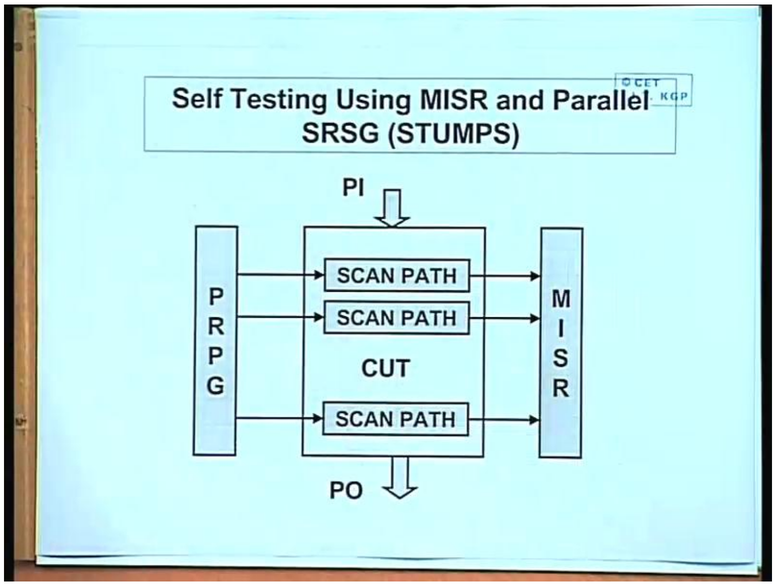
So some people proposed and extension to the linear feedback shift register this is called multiple input signature register or MISR. So this is what we are just mention with the diagram that if you ordinary LFSR compactor one with each output then you need too much hardware. So solution is to use a signature register were there are multiple number of inputs you can compact all the outputs simultaneously into one LFSR. This is the cracks. Now this MISR the white works is that because LFSR has a linear property well I am not going to details of this. It obeys superposition principle. So what will happen is that suppose you have a circuit with four outputs, so if you compress this four outputs individually you will get four different signatures S1, S2, S3, S4. Now if you compress all of them together using an MISR the signature will be getting is actually the XOR some of this S1, S2, S3 and S4. So final remainder will be the XOR sum of the individual signatures corresponding to each of the outputs. So this is also quite acceptable in terms of testing. So MISR is also a structure which is quite popularly used and MISR looks like this.

(Refer Slide Time: 25:34)



This realize the polynomial x cube, x plus 1, this is x cube, x square, x and 1. You look at the feedback path feedback path as a connection here it is a connection here. And of course in the input so it is x cube x and one. But in addition we have now one XOR gate with each stage an additional XOR gate. These XOR gates are used to feed the input data of course this is another input, this is one input and this is one input. So externally you can feed three inputs together this d1, d2, and d2. So as if now you have a structure signature register which can accept three bits streams together. Now if we have a 32 bit register you can compress 32 bit data stream at the same time or simultaneously. So this is how and MISR works. Now using MISR a number of you can say built in self test schemes or architecture have been proposed which can be used for self testing or also for DFT in some cases. Well one such popular structure I am showing now.

(Refer Slide Time: 27:05)



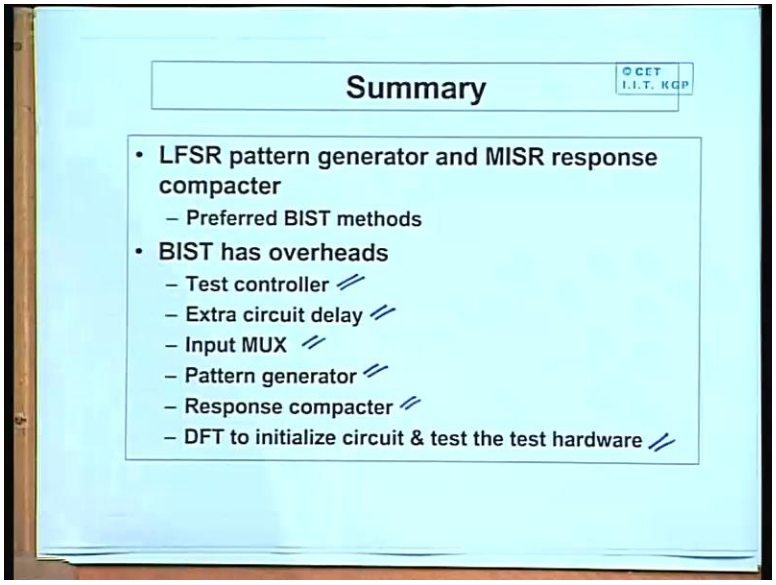
Self Testing Using MISR and Parallel SRSG (STUMPS)

This is a structure called STUMPS. STUMPS stands for self testing using multiple input signature register and parallel well SRSG is serial random sequence generator STUMPS. So here the architecture looks like this. Suppose you have a circuit out here this is a circuit in a test this is a sequential circuit. So the sequential circuit can have many flip flops. And the flip flops you have configured them in possibly more than one scan chains. You can use a zero random pattern generator. The each of the individual bits you can used to feed the individual scan chains and the outputs which you get you can compress using MSIR.

This architecture says that DFT scheme using a scan path is not necessarily used always in conjunction or connection with a deterministic test pattern. That means you need not always have to apply the test pattern deterministically from outside and you need not have to just observe the responses by shifting them out always. But rather you can use a scheme like this also for you have this scan chains in place you apply some primary inputs, you can apply again some pseudo random patterns if you want. And to the scan chains also you apply pseudo random patterns. So you can apply pseudo random patterns here, you can apply pseudo random pattern here.

In the primary outputs also you can do a compression. So you compress values and the final compressed values only you compare. This method also has been found to work pretty well. Because before just adapting any such method in actual practice we will have to find out or evaluate through simulation that how good or how bad this method is and well means what are the implications of using it. You will have to evaluate the fault coverage through simulation applied of course before that will implement a BIST scheme okay. So to summarize this BIST schemes which you have mentioned this linear feedback shift register pattern generator.

(Refer Slide Time: 29:50)



Summary

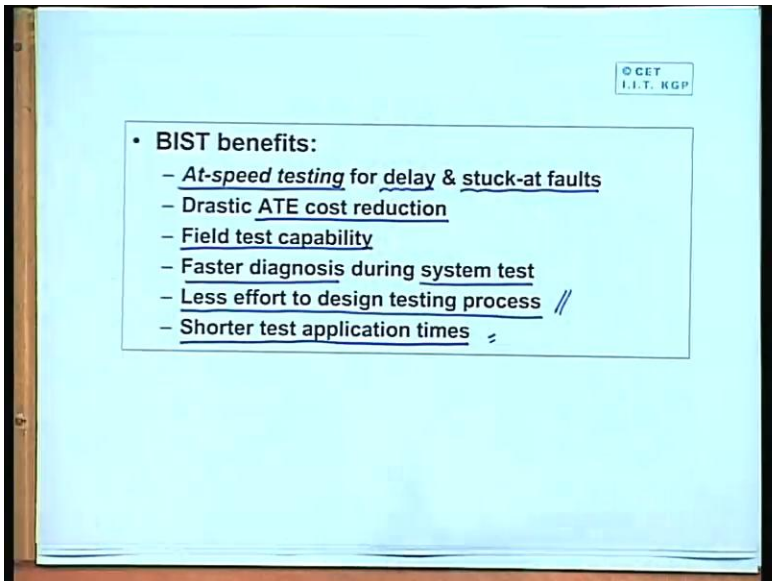
©CET
I.I.T. KGP

- LFSR pattern generator and MISR response compacter
  - Preferred BIST methods
- BIST has overheads
  - Test controller
  - Extra circuit delay
  - Input MUX
  - Pattern generator
  - Response compacter
  - DFT to initialize circuit & test the test hardware

And multiple input signature register response compacter. Structures based on these are preferred in BIST architectures. The preferred BIST methods used these but you should also keep in mind that BIST has its own prices that you have some overheads that we should incurve. Well the more complex your architecture is the more complex will be your test controller. Because it is a responsibility of the test controller to control the different BIST subsystems response compactor pattern generator who will be activated when, and when will be comparing the response everything has to be controlled by that controller with actually an FSM its actually final state machine. And since you are introducing some extra gates some multiplexors in the critical path were circuit delay is will increase of course input MUX are there. You need pattern generators and response compactors.

Not always you will find some, you can say some register which was there already in your circuit which you can expand or extend as a BILBO register to act as them. Sometimes you need to add an extra response generator or pattern generator. So an addition you need some DFT schemes for the purpose of initialize in the circuit because BIST is fine. But you will also have to initialize the circuit to a known state particularly if it is a sequential circuit. If it is a sequential circuit and if you cannot reset the flip flops to a known state your signature can also vary okay. So you will also have to have some kind of scan path kind of a thing. So that you can set the circuits in to some known state and from there you can apply the stimulus and you can find out the signature of their okay. But overheads are there of course but BIST also offers number of advantages which are pretty attractive.

(Refer Slide Time: 32:13)



© CET
I.I.T. KGP

- **BIST benefits:**
  - *At-speed testing* for delay & stuck-at faults
  - Drastic ATE cost reduction
  - Field test capability
  - Faster diagnosis during system test
  - Less effort to design testing process
  - Shorter test application times

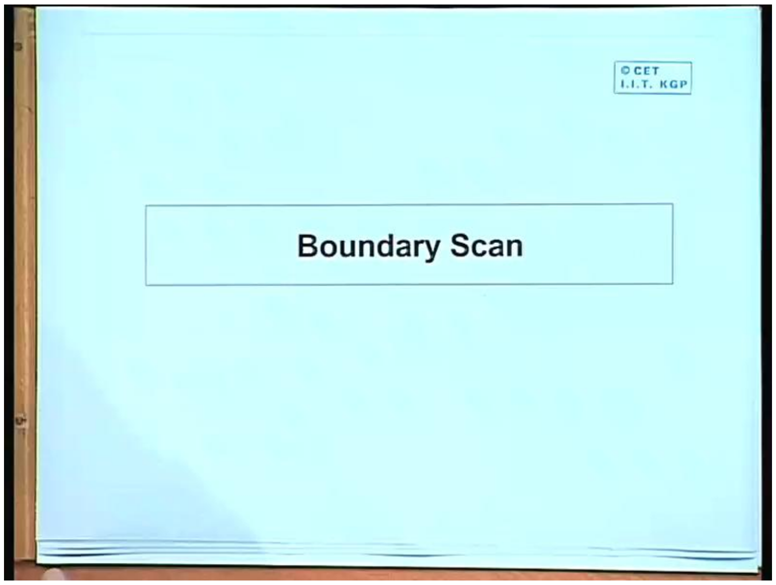
Now the most important advantage gain is that you can test the circuits at the full operational speed. If the circuit is design to work at a sequence of 2 gigahertz, your test pattern generator and compressor can also work at two gigahertz okay. This is called at speed testing which I had mentioned before and this at speed this at speed testing you can use for detecting stuck at faults its okay. Now addition you can also test some additional kinds of faults called delay faults. Delay faults occur or show because of some you can say mismatch of tolerances in the delays of the circuit components. Due to this delay faults there can be some signal clichés and hazards that may occur in the circuit. Now if you are doing at speed testing there is higher chance of this delay faults to show up as compare to testing at a slower frequency okay.

And since you essentially do not need an expensive ATE to do this drastic reduction in the cost of automated test equipment and you can test with in your own lab in your field of work. So you can have it field test capability. So you can build a system the system will have a self test capability it is not that the chip has to be tested by the manufacturer only at their own site and they will be they will be delivering the chips to finally. And since we have of these interesting features you can have faster diagnosis during system test. Of course less effort to design testing process is very important because you are eliminating essentially the process of test generation. Because you are not using any conventional test generation algorithm of course. Of course here there are both plus and minus points. Since you are using pseudo random patterns as your input stimulus it is likely that you may not get for all circuits sufficiently high fault coverage okay.

But the advantage of gain is that you are eliminating the expensive process of deterministic test pattern generation. Okay but these are the trade off you will have to look at an accept and of course shorter test application time. Because everything is done inside the chip or at a very high frequency the total test application time also gets reduced okay. So we have looked at some schemes for of course earlier we talked about methods like test generation and fault simulation. Subsequently we talked about design for testability techniques which are very essential to crack hot circuits like sequential circuits whether lot of flip flops inside. And then we talked about built in self test techniques where you can use in addition. Some more hardware where this stimulus generation and response compression can also be done on chip.

But manufactures and industry people they faced some problem earlier because well these schemes were well understood and well accepted. But there was some lack of standard. Like if there is two company say S1 and S2. So S1 brings up a chip S2 also brings up another chip with all this testability features inserted. But if I want to design a system employing these 2 chips from two different companies, so how do I make or means how do I design a scheme for system test. So is there any feature so that S1 and S2 can work hand in hand towards the testing? So industry people realized that since there was lack of standard in these of testability features well schemes were proportional schemes are accepted fine but there are no standard as such. So that filled that there has to be some kind of a standardization and when the first effort of the first step in the standardization process.

(Refer Slide Time: 36:59)

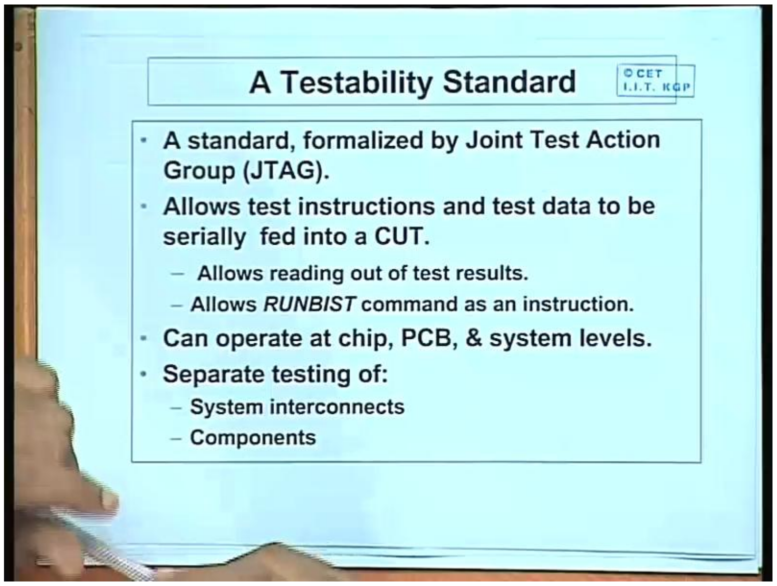


©CET
I.I.T. KGP

Boundary Scan

This was taken up by a joint committee and the came up with the scheme called boundary scan. Now the committee which came up with the standard.

(Refer Slide Time: 37:11)



A Testability Standard

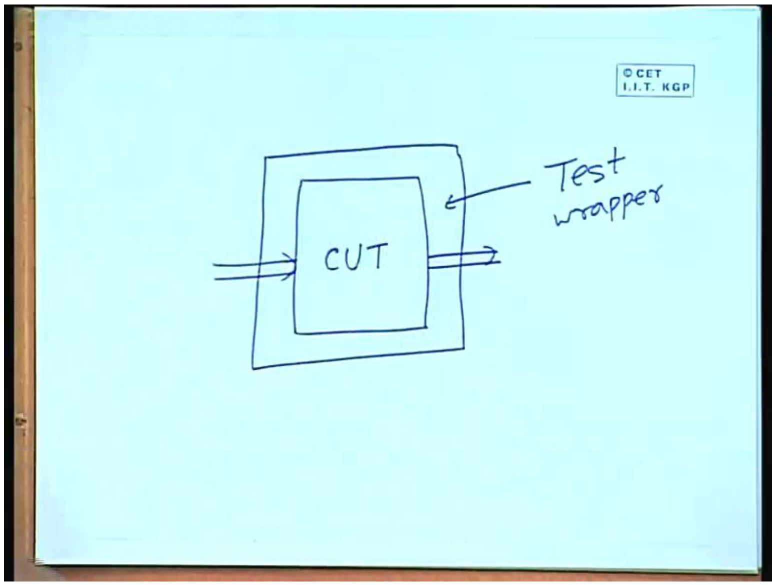
- A standard, formalized by Joint Test Action Group (JTAG).
- Allows test instructions and test data to be serially fed into a CUT.
  - Allows reading out of test results.
  - Allows *RUNBIST* command as an instruction.
- Can operate at chip, PCB, & system levels.
- Separate testing of:
  - System interconnects
  - Components

They included they include almost all the leading industry people and the consortium was called joint test action group. And this JTAG name has become so popular that this boundary scan is sometimes also called synonymously as the JTAG standard okay. JTAG and boundary scan are used or today used together synonymously. Because you can today look at some chip say for example some APJs some APJ chips buy from the market you will see there in the pin just some of the pins they have called these are these are for JTAG complaints. JTAG complaints means they are they have boundary scan feature in build or embedded in the chip okay they call it JTAG.

Now using this testability standard what you achieve is that, this standard allows test instructions and test data to be serially fed into a circuit under test. See test instructions are something which is new which means you are encountering here. Because you see all the chips may not be of the same type some chip maybe using conventional scan path based DFT some chip maybe using BIST. So externally when you are instructing the instructing the chip to do something. It is something similar to giving a small test instruction. That means what kind of test you want to do okay. So this JTAG standard this includes a set of such test instructions in the standard architecture itself.

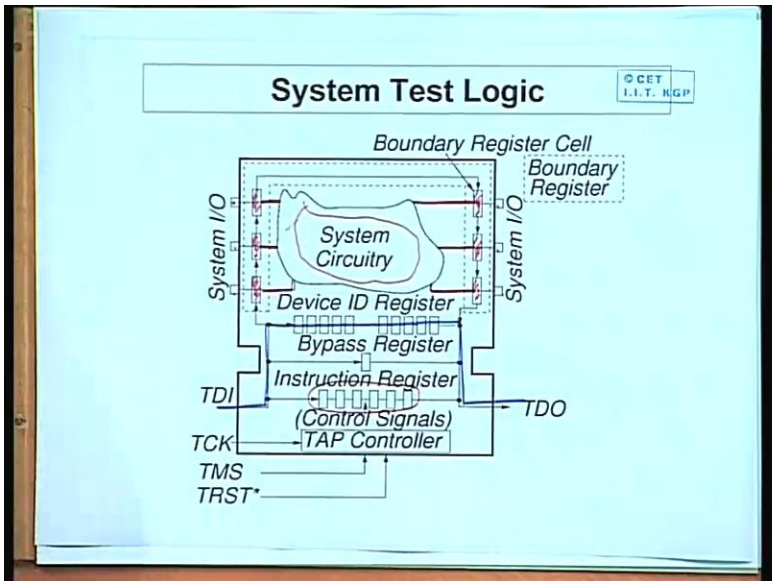
And using the test instruction you can feed the test data and you can also read out the test results. There are also some special RUNBIST commands so that if you have a built in self test features in embedded inside a chip you can activate that. Normally we will be having DFT scan path thing so that you can activate scan path you can feed in data you can scan out the response. But if you have a complete BIST thing inside a chip you can also activate it from the outside. JTAG is totally hierarchically in nature it can operate at the chip level at the board level, which the circuit board level and even at the system level. JTAG allows the separate testing of components and also the system interconnects. Now the essential idea behind JTAG is that.

(Refer Slide Time: 40:12)



Say you have circuit under test or a chip okay. I am calling CUT general this a chip. Now you want to embed a boundary scan standard on top of this. So what do you do is that you have some kind of a so called test wrapper which you built around this. Now test wrapper is a fairly complex piece of hardware we will see shortly. There are a number of features in the test wrapper. Now through this test wrapper you can activate the different testing mechanism in the CUT you can well in case of normal operation you can say that well I 1 2, I 1 2 just apply the inputs to the CUT in the normal operation mode. Now the testing modes they will be features to activate scan paths to activate BIST and so on. But I am showing you a very rough high level diagram of this wrapper what are the essential things which inside.

(Refer Slide Time: 41:21)

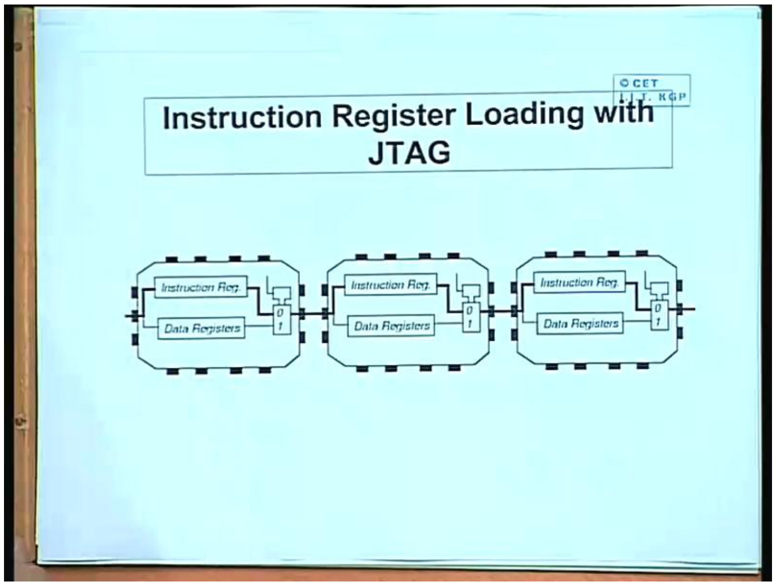


And you can appreciate that this will be fairly complex. You see this is the total chip and inside this was the circuit under test which calling this is the system circuitry. And these are the, or these where the input output pins okay. So the input output pins which we were there. Now what you do you put a flip flop just along with each input output pin. This is the boundary of the chip okay well whatever you do inside the circuitry is separate. But in addition in the boundary of the chip you provide a scan path. This is why this method is called boundary scan. There are flip flops just with each IO pin. They are change into a shift register okay. These IO pins also feed the system circuitry as you can see. So using these when they need arises you can use them to activate BIST inside or DFT inside. If there are scan path inside the circuitry you can scan them in scan them out. Now inside in addition to this there are lots of other things.

For example out here you have an instruction register. Now in the instruction register is actually a register compressing of a few number of flip flops. Now you can shift in some bit pattern here and now depending on the bit pattern the kind of instruction you are trying to execute that will be interpreted an executed. This instruction register can be fed through this test data input there is something called a bypass register. Bypass register is sometimes used if you are using several such chips in cascade I will just show um show diagram shortly. Several chips in cascade and through the test data input you want to program all of this shift, one after the other. So the first one you can do this in this path the second one you can follow a bypass path through this. This is a bypass path. So you can use the bypass path to access the next such chip which is the n sequence.

And in the bypass part in addition there are some device ID register where the device ID of all this individual chips can be stored there. And means when you are sending the TDI you can also specify the device ID it can check automatically and do it. And all this things are controlled by a test access point controller TAP controller and there are some signals like test mode select reset clock and so on. So as you can see that embedded boundary scan in a chip is a non-trivial task. Here you need to add lot of additional circuitry in addition to your circuit and as compare to schemes like boundary scanner scan path you need to add lot more. Now the advantage you gain is the flexibility of the chip you get finally. You will see that if you have boundary scan implemented, you have a number of advantages using which you can diagnose and test chips boards and systems using the same kind of schema technique. Now as I said that you can have several such chips connected in cascade.

(Refer Slide Time: 45:27)

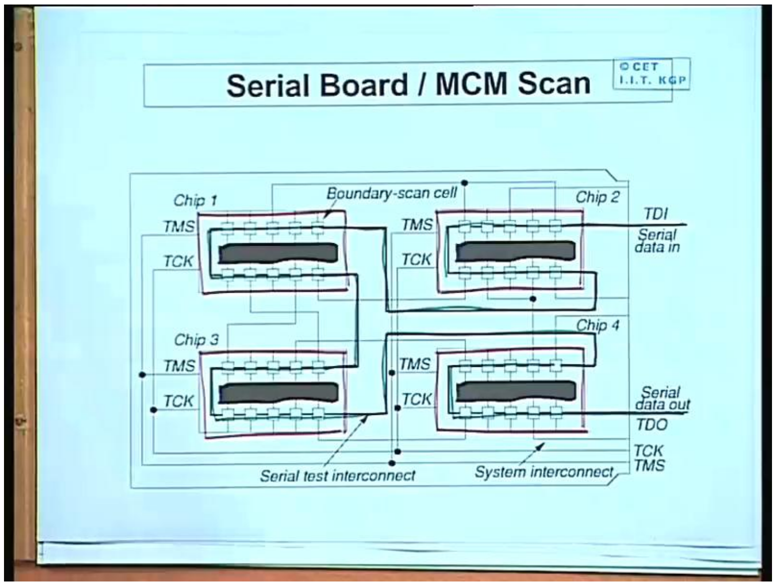


Instruction Register Loading with JTAG

Say this diagram shows three such chips connected in cascade. And I am showing not all some of the registers only instruction register and data register. So when you are feeding the instructions it will be following this path, so actually for this you have the bypass registers. But in this diagram I am not showing the bypass register I am showing as if they are following this paths and they are rippling through and you can load the instruction registers of all this chips through this paths. Now addition you have the actual data register data registers are actually the boundary scan cells. These are through the same TDI input you can access these data registers okay this data registers you can access.

These boundary scan cells, they actually compress a data register so when you are actually testing scan in and scan out you will have to use the data registers. So this diagram illustrate instruction register loading so you can load the instruct load the instruction registers. There will be some multiplexers in the output will have to select this to the next stage and so on. Now the advantage of boundary scan primarily is not for testing individual chips. But for testing a number of chips for in the line together in a board or in a system okay. So first let us see that when there are several such chips with boundary scan or JTAG capability embedded in it in a board what are the advantages we gain okay.

(Refer Slide Time: 47:18)

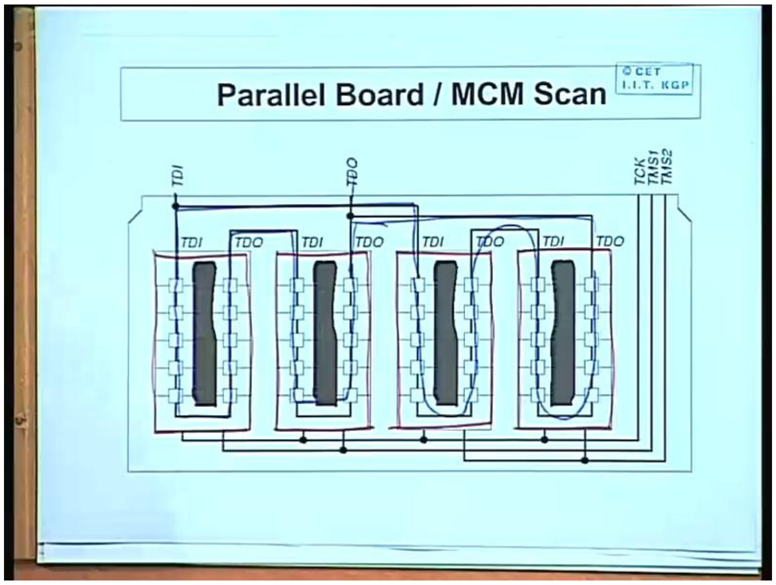

Serial Board / MCM Scan

This is an example I have given this can be a printed circuit board which consists of four chips these are the four chips I am defining. This is one, this is one, this is the third one and this is the fourth one. So it can be a printed circuit board with four chips or it can be a multichip module also multichip module is actually there are four different chips which are put inside the same IC package. So although its looks like single IC, but inside the IC there are 4 chips put inside. So it can work similarly for both. So what is says that if each of these 4 has boundary scan capability then the boundary scan register can be changed together.

Like the test data in which is entering the printed circuit board this can go into the boundary scan register of the chip it can come out of here. It can feed the TDI of the next chip it can again come out of here. It can feed the TDI of the third chip it comes out of here and it can feed the TDI of the fourth chip and can finally come out of here. Now the advantage we gain out of this is that well at the level of the PCB we have single TDI and a single TDO. Well if you had sufficient number of pins we could have 4 TDI and 4 TDO. So that we could have just tested them together. But the main advantage of boundary scan is that is you have this you can change them all together and you can shift the test data serially.

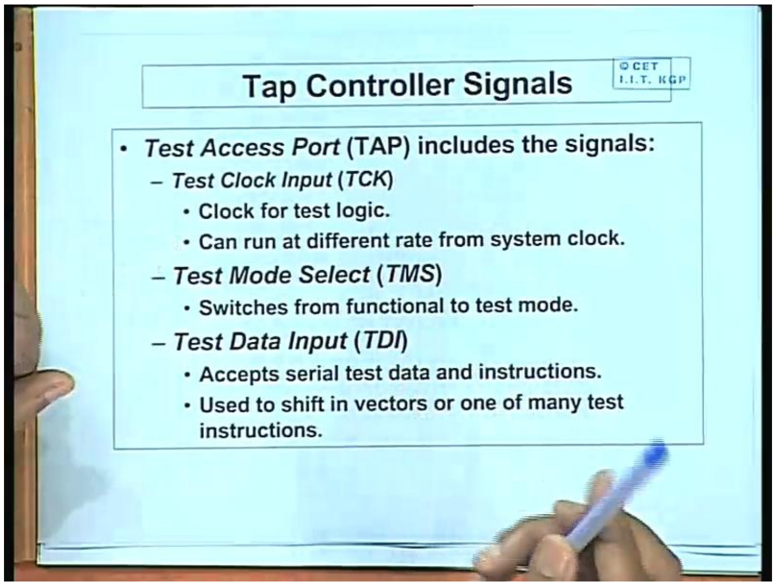
So that we can access all the boundary cells of all the chip inside and we can do scan in and scan out of all of them together. Now this is the advantage which boundary scan offers. Now this chips can now be coming from different manufactures also it is not necessary that they are manufacture by the same company. So if they comply with JTAG standard they can put the chip and the same board and yet have the capability of boundary scan. Now it is not necessary true that always you have to change all the chips together in the same scan chain. So I have another example.

(Refer Slide Time: 50:15)


Parallel Board / MCM Scan

Here also there are four such chips. But we have some kind of a parallel operation. So it is up to the user it is up to the designer that means what kind of testing mechanism is desired. So you can have parallelism okay. Here you see TDI the test data input is feeding 2 chips simultaneously. Similarly there is another test data okay this TDI is feeding this 2 together and you look at this one. So after falling through this change this is entering the second chip. Similarly this one is flowing through like this. TDO is one they are all type together. So it says that you can parallel test these two. But since you have one TDO you can do it one at a time. But if you have two different TDO’s, then you could have tested them together parallel okay. So depends what kind of resource you have you can have some change together you can have two blocks which you can do in parallel and so on. So this boundary scan is a very flexible method using which you can carry out or you can plan your testing activity exactly how do you want to do the testing. Now just to give you an idea regarding the kind of signals.

(Refer Slide Time: 52:06)



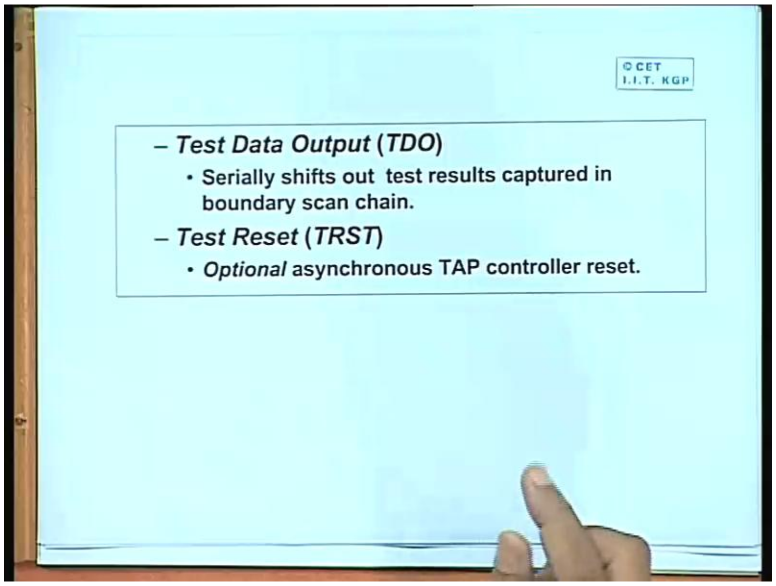
Tap Controller Signals

© CET I.I.T. KGP

- *Test Access Port* (TAP) includes the signals:
  - *Test Clock Input* (*TCK*)
    - Clock for test logic.
    - Can run at different rate from system clock.
  - *Test Mode Select* (*TMS*)
    - Switches from functional to test mode.
  - *Test Data Input* (*TDI*)
    - Accepts serial test data and instructions.
    - Used to shift in vectors or one of many test instructions.

That we have in the TAP controller or test access port controller. So the TAP controller this was shown in the diagram previously, so just I am showing the diagram once more this diagram has so many signals okay. These are all going into the TAP controller. So I am showing always I am briefly telling you what the signals here. Of course you have a clock, test clock input. Now during testing you can run this clock at a different rate from system clock it is not necessary that it should run at the same clock frequency. There is a test mode select pin. So here you can switch from normal mode to test mode. This is a test data input through which you can scan in the test data you can also scan in instructions. And you can do it in parallel you can used to shift in vectors of one or many test instructions. So you can basically shift in data which can go to one or more than one such boundary models together.

(Refer Slide Time: 53:23)

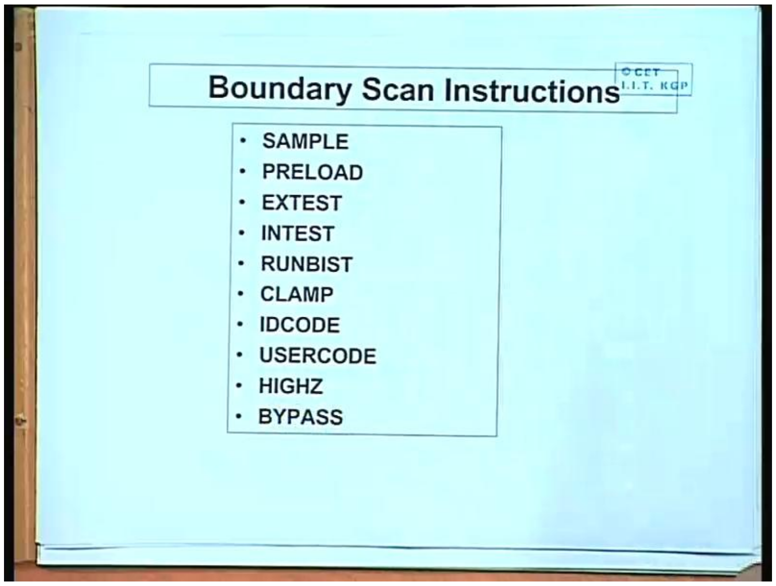


© CET I.I.T. KGP

- *Test Data Output* (*TDO*)
  - Serially shifts out test results captured in boundary scan chain.
- *Test Reset* (*TRST*)
  - *Optional* asynchronous TAP controller reset.

There is a test data output which can serially shifts out the results it is a test reset. Well this is optional where just using which you can you can reset the TAP controller. Because TAP controller maybe in some state. You wont to reset it and you can start a new testing cycle. So this test reset is a signal is which you can reset they TAP controller FSM.

(Refer Slide Time: 53:55)



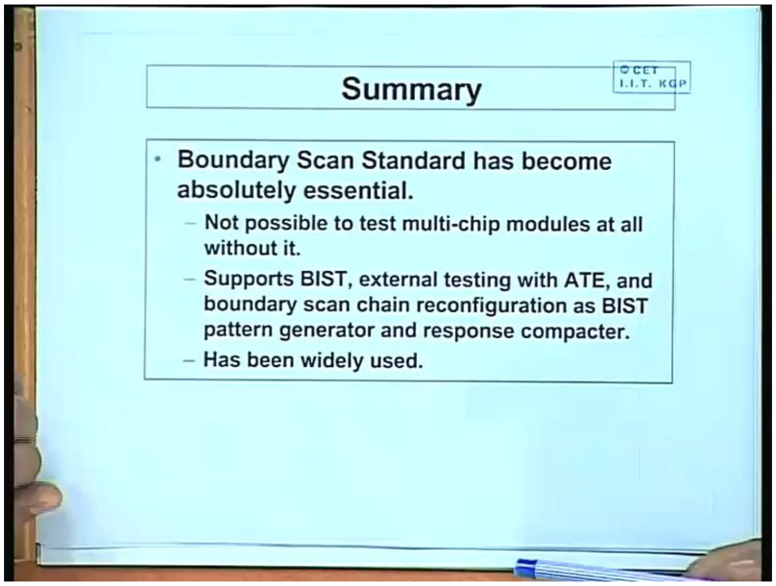
© CET I.I.T. KGP

Boundary Scan Instructions

- SAMPLE
- PRELOAD
- EXTEST
- INTEST
- RUNBIST
- CLAMP
- IDCODE
- USERCODE
- HIGHZ
- BYPASS

And I am not going into the detail these are some of the typical instructions which are there in the boundary scan sample preload execution test intest RUNBIST that there are many. So using this you can test the functional models you can test the interactions you can just run BIST inside a chip and there are number of other you can put one of the models into the bypass mode and so on. So it is quite flexible in that sense.

(Refer Slide Time: 54:28)



Summary

- Boundary Scan Standard has become absolutely essential.
  - Not possible to test multi-chip modules at all without it.
  - Supports BIST, external testing with ATE, and boundary scan chain reconfiguration as BIST pattern generator and response compacter.
  - Has been widely used.

Just to summarize boundary scan has emerged as a standard. And there are many chips and products which are now in the market which are based on this standard. If you have multichip modules for example in a chip, so unless you have boundaries can standard it is very difficult to test such. And boundary scan standard is totally flexible its supports built in self test its supports external testing with ATE. And you can have scan chain reconfiguration also. And it has been widely used many environment in the industry. So here we try to give you very rough overview about the about the essential idea of the concepts in the different aspects of testing process. We talked about test generation faults simulation design for testability built in self-test and boundary scan. But we only try to give you can say any introduction to the problem but there are a number of other problems involved related problems involved which need to be taker need to be address need to be solved. If we want to have if you want to arrive at a chip or a design or a testability scheme which is practical and cost effective and of course to be attractive to a large community of users.