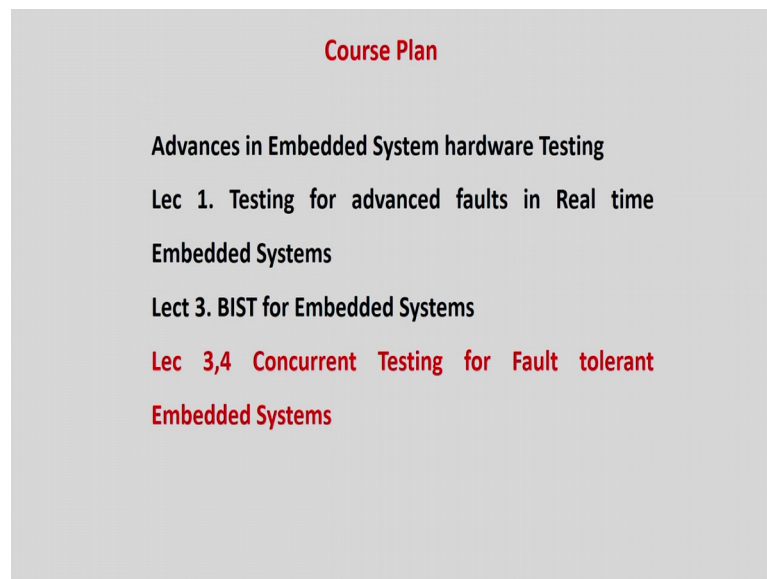


Embedded Systems – Design Verification and Test
Dr. Santosh Biswas
Prof. Jatindra Kumar Deka
Dr. Arnab Sarkar
Department of Computer Science and Engineering
Indian Institute of Technology, Guwahati

Lecture - 34
Concurrent Testing for Fault tolerant Embedded Systems – 1
Part-3: Embedded System Testing

Hello everybody, as we know that we are in the 3rd part of the course which is on testing. And within this 3rd part on testing now we are in the advances of embedded system hardware testing in which case we have already the discussed about two main aspects.

(Refer Slide Time: 00:35)



Course Plan

Advances in Embedded System hardware Testing

Lec 1. Testing for advanced faults in Real time Embedded Systems

Lect 3. BIST for Embedded Systems

Lec 3,4 Concurrent Testing for Fault tolerant Embedded Systems

One is about some kind of very realistic fault models which are required for advanced embedded system hardware like real time fault models or we can say delay fault models which meet the real time requirements.

Then we have seen that basically what is called built in self test that as these fabrication technology of the embedded hardware are becoming more and more sophisticated, so we have to test each of the devices before this data. And we have discussed at length in the previous lecture on built in self test. Now, with the next two lectures we are going to see

a very very important concept and a very new concept of testing which is called online testing or concurrent testing that is very very important.

(Refer Slide Time: 01:18)

Different types of VLSI Testing

- ATE (Automatic Test Equipments) based testing
 - Test patterns are applied to a circuit and outputs are compared with
 - the expected fault free response using ATE.
 - It requires a non-operational mode for testing.
 - Difficulty in at-speed testing because ATE speed is in MHz range while operational VLSI core speed may be several GHz

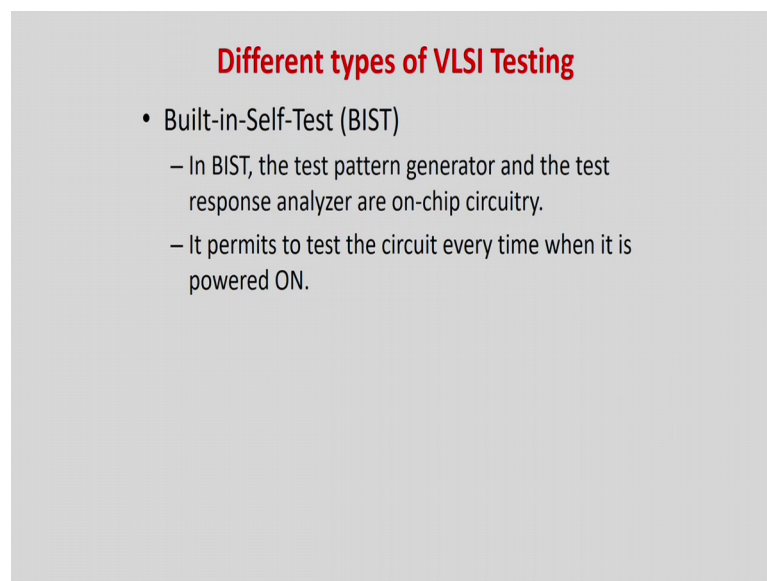
So, that is basically see or this first two test cases that is lecture 1 and some sort is a lecture 2 basically or all the basic concepts of test we have seen till now are something called offline testing. So, what do you mean by an offline testing? So, basically what do you mean by an offline testing and base we can categorize both of them in this part of a offline test because, offline test is something like when the device is not doing is regular activities.

Now, the first basic concept of test which was we call as manufacturing test or which also call it as the ATE based testing. So, in ATE based testing what happen? Test patterns are generated beforehand by or ATPG algorithms and then the device is test into the automatic test equipments, patterns are applied to the circuit and that is device and the expected fault free response or analyze by the ATE.

So, basically it requires something called a non-operational mode of circuit and of course, this is not at all deployed in the system at all. So, basically already discussed so many times that ATE based testing is devices fabricated pushed in a tester and you test by physical signals.

So, of course, there is no question of operational mode of the circuit because the circuit is not yet deployed in the system. So, the main issue here is that you can test it only once after it is manufacturing. So, if it develops any faults during its lifetime this takes place. These test technologies cannot cater to that. So, for that we have something which is called built-in self test. So, a difficulty of ATE based testing is basically ATE speed testing is a bit difficult or with expensive I should say because more number of test patterns means more high speed ATEs are required.

(Refer Slide Time: 02:54)



Different types of VLSI Testing

- Built-in-Self-Test (BIST)
 - In BIST, the test pattern generator and the test response analyzer are on-chip circuitry.
 - It permits to test the circuit every time when it is powered ON.

And also secondly, basically we should also add that it is an offline test. Offline test basically in the sense that if the system develops a fault after it is deployed in the system ATE based testing cannot handle that. Secondly, to do this what we have seen to address such cases we have seen there is something called a built-in self test. So, built-in self test basically the test pattern generator and the test response analyzer are on-chip circuitry.

And what it allows it permits to test the circuit under test every time when it is powered on. That is first is called offline or simply I should call it ATE based offline, after manufacture the chip is tested sold to you. Then after that is some manufacturing if some defects come up after it is deployed in your system you have to use something called built-in self testing.

(Refer Slide Time: 03:41)

Different types of VLSI Testing

- On-line Testing (OLT)
 - Off-line test strategies (like ATE based testing, BIST, etc.) need to withdraw the circuit from normal operation for testing which may not be always permissible.
 - OLT (also called concurrent testing) can be defined as a technique to monitor a circuit and detect the occurrence of a fault within a finite time of its occurrence during normal operation.
 - There is no test pattern generator to generate test patterns.
 - But it requires DFT circuitry to test itself for all input patterns that it encounters during normal operation

But, again always in both the cases like ATE based testing or built in built in self test based testing we call it as offline test, because we need to withdraw the circuit from normal operation for testing which always may not be permissible. So, I mean I am not going to the second clause right now, but this DFT base test, this ATE based testing or BIST based testing what you have to do? In ATE based testing forget about the normal operation of the circuit even circuit is not deployed here.

But even in case of BIST what happens you have to halt the normal operation of the circuit because already we have seen the architecture yesterday that is a multiplexer. So, either you give the test patterns from the hardware pattern generator or the normal inputs come through. So, in that way you have to stop the operation of the circuit and then you apply some patterns or in fact, you should not call it stop basically even before the circuit starts it is operation you have to apply the BIST.

So, while the BIST is going on normal inputs cannot come. So, at many points of time sometimes not only BIST is applied at the startup, in many many cases nowadays people apply BIST in intermediate points of time also there is if you have some ideal time of your circuit you can apply the BIST and test. But in all these cases you have to be very very important to remember then normal circuit inputs cannot go. So, the circuit has to withdraw itself from the normal functional mode or if it applied at the start time, so of course, the circuit has not started it is operation yet.

So, all these test technologies which you have discussed at length in the last 2-3 weeks are basically all offline. Why offline? Because the circuit is not doing normal operation or basically or it is not yet deployed even. Even if you are talking about BIST still the circuit has to stop it is word or it is applied at the start up. So then the circuit cannot or the in the all these procedures the circuit cannot be tested why it is doing it is operation so, either at manufacturing after manufacturing or just before you starts it is job.

Now, a very important question comes. See I have a very very important till mission critical application like a anti locking break system or it is used in automatic avionics control. So, this system is through it is operation and mean time of fault develops. So, what you will do? So, the direct answer should be I should stop the operation immediately and as soon as possible I should switch to a redundant module. So that is ok, that is the only way you do, that is actually fault tolerance.

But, there is someone to detect that the fault has occurred online that the and in that case you cannot stop the circuit from doing it towards to test it. In other words, I cannot say that I hey I want to test this circuit, so stop it is operation I want to test it if only find then you can start it is operation again. That is known as is not possible in mission critical must be, in any of the system is basically because if you want to stop somebody's job then basically either it will be delayed we will lead to deadline misses and so many other things can happen in picture.

So, in most of the real time systems or real time; real time embedded systems there is no liberty to stop the operation of a circuit and test it on the run and then again say, the hey it is fine you will start the operation again. So that actually philosophy will not hold in any of the real time embedded systems because we are very hard placed on timing requirements. So, therefore, the another class of testing which has to be a very new concept of test has come up which is called online testing OLT or also called concurrent testing.

Can be define as a technique to monitor a circuit and detects the of occurrence of a fault within a finite time of it is or occurrence during it is normal operation, very important to observe the clauses. It is called monitoring; circuit is doing it is job in the monitoring what is happening because the all offline testing you stop the circuit and you can apply a test patterns at your will. Because, ATPG means as you have seen you are going to

generate very very good test patterns, that this test patterns are very good, they can detect all the faults and their minimal set of test patterns so you can apply that. But for that you have to stop the operation of the circuit and apply your desired patterns.

But when the circuit is doing its normal job then basically what? Then you cannot apply your required patterns, patterns are already coming which is its normal flow of operation. You can just observe or monitor this circuit's operation for those patterns and if any fault is detected you have to do some action. So, therefore, there is no test pattern generated to generate test patterns in online test. Online test basically you cannot stop the circuit from its operation, so there is no test pattern generation required because you cannot apply any test patterns at your will.

Normal inputs are coming and you have to observe whether there is some deviation from the normal behavior or some fault is detected and then you have to decide the occurrence of a fault within a certain finite delay. Why I should say finite delay? Because a fault has occurred and you are not able to detect it for years, will actually lead to catastrophic situations.

Because in case of online test you do not have a you cannot apply your desired test patterns so that you can immediately detect the fault that you cannot do. You have to only rely on whatever patterns are coming or input patterns are coming and you have to observe the circuit based on some kind of a some kind of techniques as we will discuss today and then you have to declare whether the circuit is having any faults or not. And that also we should our techniques should be such that after the occurrence of the fault you should do it within a very short amount of delay.

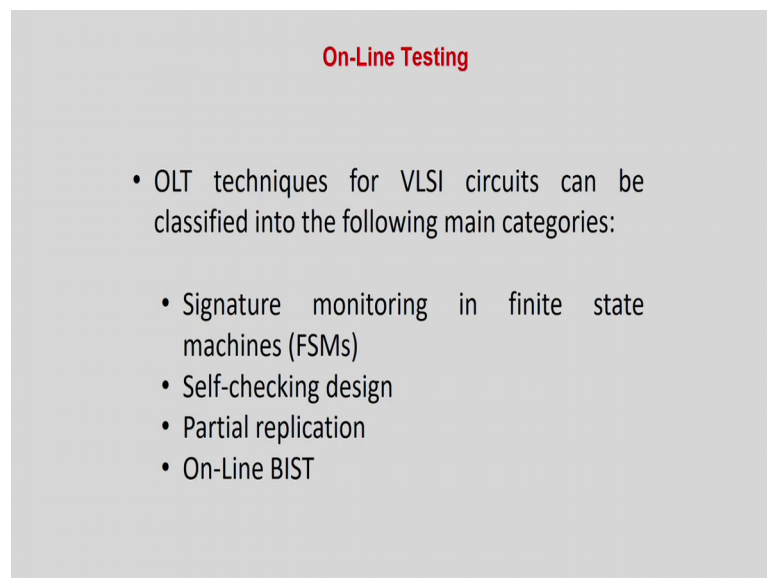
So, they are saying there is no test pattern to generate test set because you cannot apply any test set, but of course, there are a lot of DFT circuitries which will basically use these patterns that are normally coming to the circuit and use it to find out whether there is any problem or not in the circuit or not. So, the philosophy I can tell every nice nutshell that in test mode you can apply your patterns and get the circuit testing.

But in normal online testing you do not have any patterns because you cannot apply any patterns. So, whatever normal patterns are coming in it is normal course of operation you have to use it and you have to monitor, basically you have to monitor the circuit and

find out whether any problem is there or not and that has to be done without stopping the circuit.

So, if you cannot stop the circuit means you have to totally rely on whatever inputs are coming in normal codes and based on that you have to detect whether a fault is happening or not. So, it is a very very difficult way to do it and very expensive also because you have to put lot of extra circuitries to do all this. Because, in case of the offline test we have seen the extra circuitry in sequential circuits are nothing but your scan chain and you have to find out basically your patterns and you can analyze that, but in case of a online testing things are becoming are extremely difficult.

(Refer Slide Time: 09:52)



On-Line Testing

- OLT techniques for VLSI circuits can be classified into the following main categories:
 - Signature monitoring in finite state machines (FSMs)
 - Self-checking design
 - Partial replication
 - On-Line BIST

With this I can just take you 2 minutes for a very interesting technical story on this I think we all know about our very famous Cardiac Surgeon Devi Shetty. So, once he went with the car to a person for repairing then the car mechanics said that we also repair the car heart you also repair human heart, but your and our pay difference are so high. So, why is it? The answer was very interesting that you do a offline repair on the car and I do a online repair on the human heart. That is you stop the cars operation, you remove the engines and repair it, but I cannot do that on the human being.

Human heart is running and I have to do the repair or test on that so that is the philosophy of online and offline test. Offline test is the pretty simple job you stop it and test it, but online testing is the circuit has to do it is normal job among the in that period

of time you have to do it. And nowadays it is a very very important concept coming in all embedded system because all mission critical systems are having basically embedded electronics, car avionic, car crews, obvious automatic breaking system, even the whole aircraft control is based on embedded controllers. So, it is a very very important stuff which is coming into embedded system testing.

So, we have dedicated two lectures on that. So, today we will see some of the techniques and in the next class some other kind of techniques were online testing or sometimes also called concurrent testing of hardware in embedded systems. Basically, the if you look at your online testing of VLSI or embedded systems can be mainly categorized into signature monitoring in final state machine self checking design partial replication and on line BIST. So, today basically we will try to have a look at the first two because the other two will be looking at the next class. They are slightly simpler, in theory sense compared to the first two.

(Refer Slide Time: 11:28)

Signature monitoring techniques for OLT

- Modeling the circuit in terms of a FSM
- Signatures are state sequences of the FSM traversed during execution of the circuit.
- The basic idea is, runtime signature of a faulty circuit is different from the normal circuit, called signature invariant property, which is checked by the monitor at the check point nodes

So, what do you mean by for signature monitoring base test techniques for online testing? So, whenever we call a online testing in terms of finite state machine modeling, so we all know that all sequential circuits can be model in terms of finite state machine. Then we find out some kind of signatures in the finite state machine model of the circuit. And then basically the signature is such that the runtime signature of a faulty circuit is different from the normal circuit. And of course, you cannot I mean if you just arbitrarily

take any input patterns or sorry any input sequence or any state encoding sequence that will not be a signature.

So, signature has to be very well quantifiably found out from the FSM model. That is circuit is there, you make an FSM model and therefore, and then very carefully set select some state and state bits or output bits so that the normal flow of the circuit and the faulty flow of the circuit the signature will be different. Now of course, if I take all the output bits and all the state bits then of course, if there is any deviation from the normal behavior you can catch it, but in that case your monitor itself will be the circuit again.

I will give you an example and I will tell you. That again one thing you have to know that whatever extra circuits you are using for monitoring should not be as large as the circuit itself because then nobody will take your technique. That one circuit is given to you another circuit you are reply applying to monitor it which is as large as itself because, then who is going to have a fault probability your circuit under test or your working circuit may have a fault and if you have a very large circuit also to monitor it that may also have a fault. So, nobody can keep a faulty or a similar kind of a fault nature circuit for it is monitoring.

Something like this a circuit, a x is there which is doing your job you are putting another similar circuit to monitor it both are equal probability of having failures online so, you will not relay. So, basically what happens, so in this case what we do we take a circuit under test which is your working circuit which will be quite large and then you take a miniature version or you take a miniature modeling or miniature sized online tester to monitor it.

And it is expected that and statistically also we have found if a circuit is more the larger a circuit is more number of gates and more number of faults it will have. So, higher probability of faults they were or high probability of development of faults in a bigger circuit is there compared to a smaller circuit.

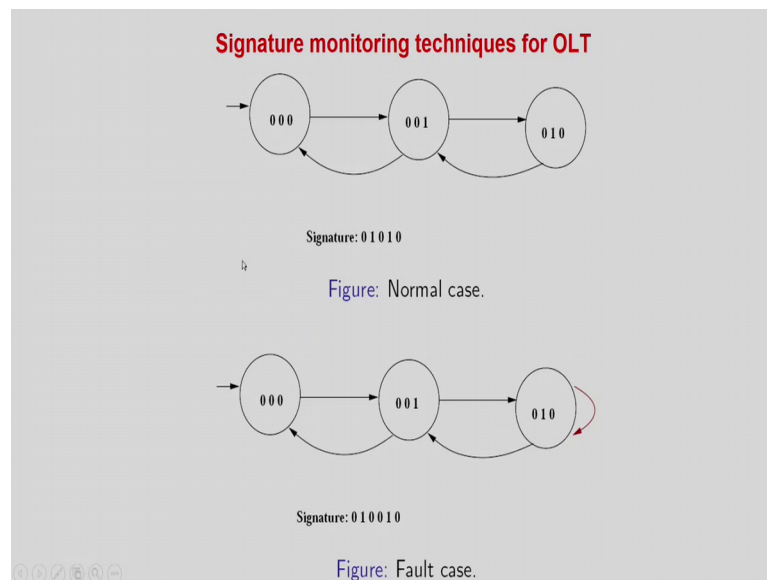
So, our philosophy is that for online monitoring the circuit under test should be large and the monitoring circuit should be small. So, therefore, we cannot explicitly I mean have such kind of signatures so that the monitor itself becomes as large as the circuit. So, we have to take certain bits of the circuit state encoding, certain bits of the output so that you can make a signature so that if the normal circuit is there and the faulty circuit is there

the signature should be different in the faulty circuit. But again this monitor size should be small.

And there is something called a signature invariant property that is this the BIST we will be used to may make a signature should be such that the normal and the faulty behavior should be different. But always it may not be possible just like we have seen in the last class or in the few classes already we have seen then that there is something called signature compaction in BIST you have seen that there is something called output compaction. We do a output compaction to reduce the law so, sometimes there may be aliasing.

So, similarly in this case if you are making a signature that is nothing, but lousy compaction so, sometimes some faults may be alias. But still, there are lot of techniques to improve such signature invariant properties so that even and there should not be very less or no aliasing. But anyway those details coding theory will not be going into rather this codes we are going to give you a broad idea than what is a signature monitoring based techniques and you can easily appreciate that there will be aliasing.

(Refer Slide Time: 14:56)



So, for example, this is some circuit and this is the final state model. So, you can see the state bit represents 0 0 0, 0 0 1 and 1 0 1 and if there is a fault there is a back loop. There is a self loop which is in the fault and it is in normal case there is no fault. So, this is (Refer Time: 15:11).

Now, you can tell me that if I want to make a monitor basically you can what you can do is that if you say that I can just observe that there is that there is a basically the it should go like this and there should not be any kind of a loop like this I want to make a circuit to monitor. So in fact, you will be landing have been modeling a circuit like this. What I mean to say is that if you want to observe that there should not be any kind of a loop in this position and if the loop comes I will declare, it is an error.

So, I want to make a monitor like this that means, basically you are making a circuit like this to monitor this. So, that is an explicit monitoring on the circuit which actually do not like because of the area over it; rather we would like to take a signature. So, if you assume that this is the only fault that can happen or the fault effect basically. So, what we can do? We can you can make a signature which would be comprising of this or the last bit of the circuit. So, what is the signature pattern if you see? 0, then 1, then again 0, then again come back 1 and again it is a 0. So, it is a alternating 0s and 1s; 0 1 0 1 0 1 that port so, this is a normal this is a signature on the normal case.

But in this case what happens if there is a loop over here. So, signature will be 0 1 0 and there will be a repeat 0 so, if the repeat 0 you know that is a fault. So that now you really observe that to monitor the same circuit the circuit size will be much much lesser than the circuit corresponding to this finite state machine.

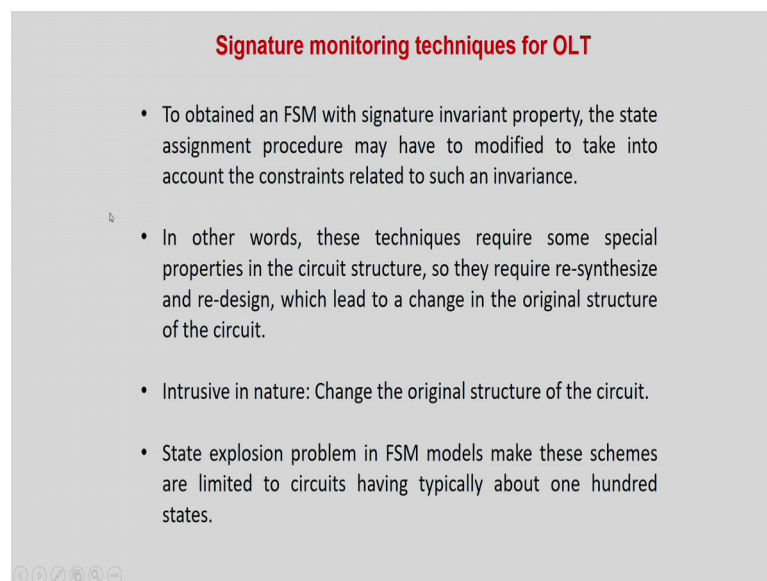
But of course, as I told you there can be lot of aliases. In this example there may not be any aliasing because, but I think from the because this is a lousy compaction based scheme. So, from the BIST examples I have given you can easily appreciate that there can be some kind of aliases. And there are lot of other techniques to this remove this alias fault aliasing, sometimes they modify the circuit itself so that the invariant properties preserved.

So, what is the invariant property? Invariant property is that whatever signature you select like or this case signature our case is the last bit of the circuit LSB of the finite state machine states. So, the signature invariant property is that with this failure or whatever the failures or circuit faults you have consider the normal signature will always be different from the fault signature. So, always it may not be possible as I told you for a given signature and a given circuit. So, there are lot of techniques means which is you can read through which are called anti; anti means techniques for I mean which are

actually basic idea is called to maintain the signature invariant property or maintaining signature invariance. There are a lot of techniques to do that sometimes we change the signature, sometimes we modify the circuit itself so that this invariant properties always get maintained and there are some other techniques to do that.

But, in this course or in this slide I have try to give you an idea what is a signature and what is signature invariance and how to maintain a signature invariance is a long story which are not going into. But of course, you should appreciate that without such ensuring such signature invariant properties not all faults will be detectable there can be aliasness, aliasing. But why do we take a signature? To reduce the size of the monitor circuit. So, in this case you I have shown you that you have to this is this is a signature of a alternating 0s and 1s this is a signature, any error you have we will get double 0. This is only assuming that there are only one fault of this nature in the circuit.

(Refer Slide Time: 18:09)



Signature monitoring techniques for OLT

- To obtain an FSM with signature invariant property, the state assignment procedure may have to be modified to take into account the constraints related to such an invariance.
- In other words, these techniques require some special properties in the circuit structure, so they require re-synthesis and re-design, which lead to a change in the original structure of the circuit.
- Intrusive in nature: Change the original structure of the circuit.
- State explosion problem in FSM models make these schemes limited to circuits having typically about one hundred states.

So, whatever I told you to obtain a signature with signature invariance property this state assignment procedure has to be modified to take into account the constraints related to such an invariance. There are a lot of theory in picture so, in which case if you say that the alternating 0s and 1s is my signature and that is that should be the invariant one.

So, there are a lot of techniques the state bit encoding, state bit modification and say a several other ways which will require to change the circuit structure or redesign the circuit structure so that such invariance is required.

And many people may not like this because you can do a lot of things with the DFT because this the monitor is the test design a prerogative, but I am the circuit designer and this is my finite state machine design. Without changing the basic functionality if I do certain kind of modifications I maybe very unhappy about it but of course, you cannot change any functionality of the circuit.

Like for example, I will give you this is a 3 input and gate whatever job you do with this, this is also the same job will be done by this structure also, but you may say that there can be a delayed in there may be slight delay increase because of the two levels of two levels of the circuit.

So, in that way people sometimes modify the circuit design or state encoding so that the invariance is maintained. And within the invariance maintained no there will be no fault aliasing. But we call it actually intrusive nature of a circuit that is you change the circuit itself for online testability. So, sometimes people have not people do and appreciate much about the circuit invariance or circuit sorry circuit intrusiveness. So that is one challenge in finite state machine based online testing that it changes the circuit at many many points of time.

Finally, also the second problem is that finite state machine is not a scalable model. So, for very large circuit you cannot at all make a, I mean cannot make a finite state machine itself. So, no question of how do you test it using the signature monitoring property of FSMs.

So, it was a very good starting point for online testing circuits. For this was one of the first propose methodology which are done online testing for small circuit. But when things started becoming larger this technique could not be applied, mainly for scalabilities issues and also even in the beginning phase also some people are not happy with it because, they were asking you to change the circuit structure to maintain signature invariance.

(Refer Slide Time: 20:16)

Self-checking design

- Self-checking can be defined as the ability to verify automatically whether there is any fault in the circuit without the need for externally applied test patterns.

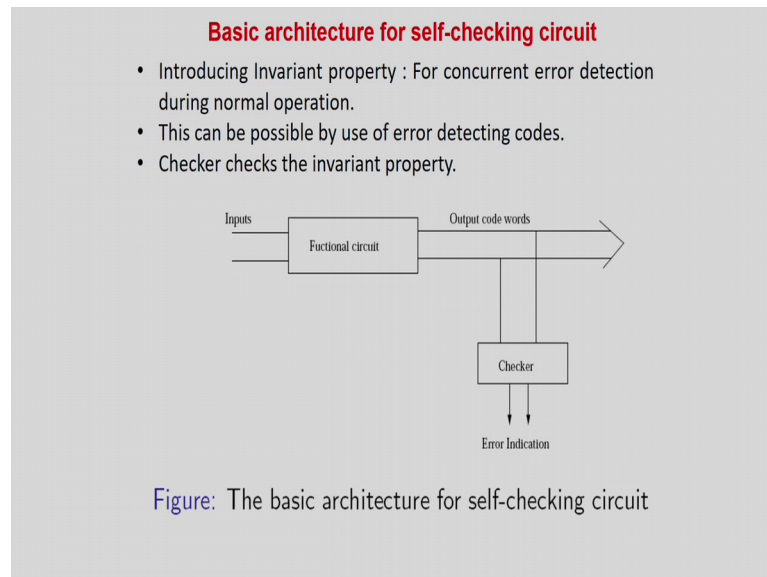
Then there is some another concept came up which is called self checking design. A self checking circuit can be defined as the ability to verify automatically if there is any fault in the circuit without the need for externally applied test patterns. So obviously, here also it is same case there is no externally applied test patterns only you have to look the look at the monitors. But, they have given a very specific name for such class of circuits in which case without applying any external test patterns you can obtain the circuit, you can derive whether it is a fault in the circuit. And they use they encode the circuit outputs using some coding theory some codes like parity codes, Berger codes, m out of a n codes.

Even you can tell me that signature monitoring is also some kind of a self checking design because, there is a signature invariance, you have to check just the signature invariance property. But, I mean self checking design was a new term which was coin after this finite state machine signature monitoring, in which case they say that a circuit is self checking if you are able to automatically determine whether there is any fault in the circuit without any applied pattern.

In how; all the outputs of the circuits will be always encoded by a particular mechanism. It will be either parity, I think we all know about parity codes. That is the output if normal we will follow a odd parity or a even parity. Sometimes there is a hot one encoding so, the output of the circuit will always be a hot one encoding or m-out-of-n,

like if there is a n bit output always n number of ones will be there for all cases so that way the output of the circuit will be always encoded by a particular encoding I mean encoded theory by coding theory and encoding. And any violation, any fault in the circuit will lead to a non-code word so that is the basic idea of a self checking design.

(Refer Slide Time: 21:52)



So, we will take an example so, it basically looks like that. So, the functional circuit inputs a normal flowing inputs whatever is coming you have no control on that, but the outputs will always be a code word. Either is a parity code m-out-of-n code, Berger codes many coding techniques are there and there will be there. And this is a checker; the checker will just check whether the output is a proper code word or not even parity odd parity m-out-of-n, hot one. So, if the code is matched it will say that there is no error. The idea of such a circuit is that if there is a fault and the circuit is effected by a fault it will go to a non-code word.

Like for example, if I assume that is a odd parity circuit so, this is a odd parity circuit. So, always the output will be the odd parity, if there is a fault in the circuit it will just become a even parity. But it can never happen that under fault there will be a another code word can come therefore, example I say that for input one this is the output for input two under normal condition this should be output not by a fault input two cannot lead to this one. If by input two and failure if you go to this pattern as the output then

your circuit will say it is normal, checker will say it is a normal because this is a odd parity, but that will never happen in a self checking design and the faults consider.

Later what will happen, if there is a fault I two can be mapped to something like this which is a non-code word, so that is the basic idea of self checking circuit is the code words. That is the if there is a failure the output will be a non-code word, but on the low circumstances the output will be mapped to a wrong key code word so that is how you can easily detect the faults. But of course, again some invariant properties that is the invariant property that this should happen, if there is a fault it should be a non-code word if there is a if there is no fault always should be a code word which is correct.

But there cannot be a mapping wrong mapping to a code word. With the fault you cannot map one odd parity code to another odd parity code it will not happen it will go to a even parity. To get all this basically the circuit will have some kind of a invariant property sometimes again the circuit must be modified and some stuffs are there as we will see. So, but the means we are not going to as much depth of the theory, but finite state machine modeling requires a very huge amount of circuit change into obtain the signature invariant property.

But in case of means self checking code based output theory this change is very very minimal as we will see. So, therefore, this is a more popular architecture which is applied for online testing, examples we will making things very very clear. So, I take a simple half adder circuit so, we know that this table is basically for your half adder circuit, I think which you all know this is your half adder circuit.

(Refer Slide Time: 24:21)

Error detecting codes used in self-checking design:

Parity Code

- Separable code, adds only one check bit depending on even or odd parity.
- To illustrate self-checking using parity code, let us consider the circuit of the half adder.

x	y	s	c	p (even parity)
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	0	1	1

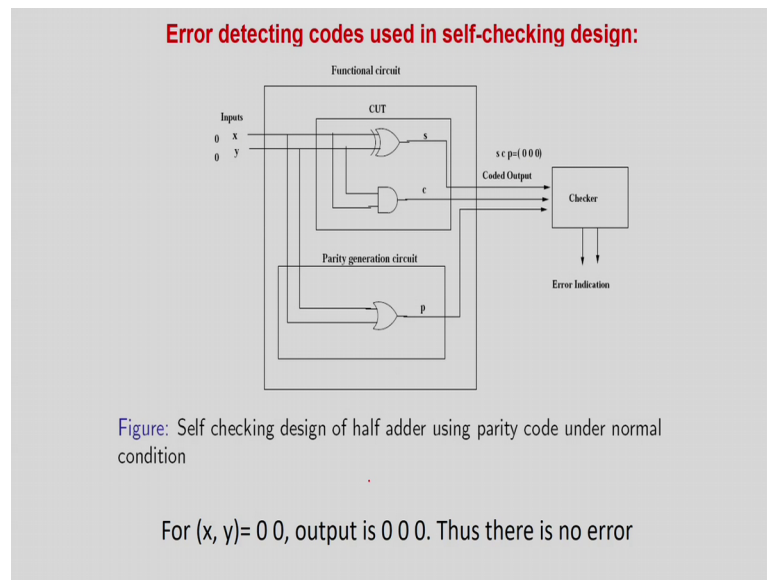
Table: Truth table of half adder where output encoded with parity code.

$S = x' y + x y'$, $c = x y$ and $p = x + y$

So, 0 0 sum carry 0 0 0 1 sum will be one carry will be 0 and for 1 1 the sum is 0 carry is 1 very simple. Now, we know that in this case of a circuit any of this inputs can come sorry, any of this inputs may come and we have no control on the inputs, but still we have to detect whether there is a fault or not. So, basically what I will do there are say that I will use a parity code. So, what the parity is? They have taken an even parity.

Now, the property of this circuit has to be made such that if there is a fault the circuit will be odd parity and if there is normal there will be even parity. So, I will add an even parity, so 0 0 0 all other cases will be 1, right. So, this is this function for sum, this is the function for carry and this is the function for parity that is x or y . Now, this is your circuit, let me zoom it for you, if you look at it this is your circuit.

(Refer Slide Time: 25:08)



Now, very interesting this is your cut this is the device under test inputs are there and basically if you look at it this is your parity generating circuit this is your parity circuit and we have already seen the parity circuit here is nothing, but x or y. So, I have put an x or y in it and this is your parity output, correct.

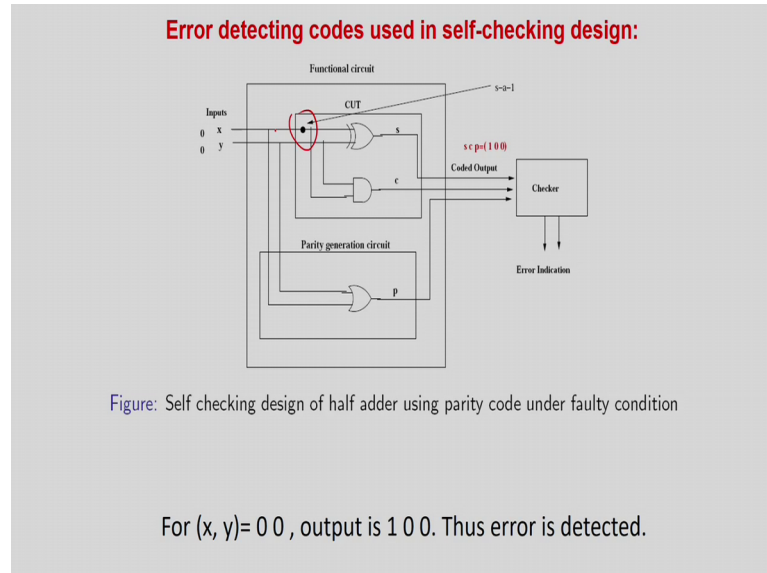
And this is a checker this checker job is nothing but it just checks even parity that is it. So, now, you can see of course, the size of the cut is much larger than the circuit of a parity generator and you can assume that if the circuit is quite large your cut will be much much larger, cut in the sense is circuit under test which is your basic circuit functionality which is doing your main function of the circuit.

And the parity generator which is the DFT circuit will be much much smaller than this, so you assume that the failure probability here or the failure probability here is much much lower than the circuit which is large in nature. Because, statistically it is found larger a circuit is more number of gates larger silicon area so the fault probability is are higher. So, if you apply the pattern $0 0$ so, you will find out that basically sum and carry are 0 and the last pattern parity is again also a 0. So, it will find out that the parity is all 0 even parity knowing.

Now, again let me just take a stuck at fault in this line. So, it is the stuck at fault in this line so, basically here x and y will go. So, this is the fault over here inside the cut and one more assumption I should tell you we are not going to the exact proof, but here again

we are assuming a single stuck at fault principle either the fault will be in the cut or the fault in the circuit or the fault in the checker.

(Refer Slide Time: 26:35)



And it has been shown mathematically that this checker based scheme if there is a fault in any of the either 3 circuits this faults can be detected by the scheme, not only the fault can happen in the cut it can also be there in the parity generator also in the checker. So, anyway such a stuck at single stuck at fault is there this cycle will give you an error. But I am not going into the theory of this because it involves lot of coding theory principles. Means, whenever you talk about VLSI testing the areas after BIST or starting from BIST is the entirely involved in a very very complex coding theory.

So, why it happens, what is the proofs of all this are very very involved and we have separate courses of coding theory on that. So, for the for this course I would request you just you take the word from me and just assume that this is the case, because most of the testing errors relay on this theory and if very interested ones can go for the proofs. But we all know that this property will holds we can we can apply in the test. So, for the time being the fault can be anywhere only one fault at a time s, basically the same property will hold.

So, in our case we are assuming a fault in the cut because this is main of the consideration. So, if you look at it your output will be basically 1 0 1 because this is XOR, so 0 XOR, 1 is basically 1 so, now, this will be 1 0 0. And basically what happens?

So, the output is 1 0 1 means a 1 0 0 so, is a odd parity through the fault it is going to be detected, right. So, this is a very simple way of handling parity.

So, in parity what happens? This parity checker is such is augmented with the circuit so that if the, so that with this expected working output basically it will add a bit here so that always in the normal circumstances of the cut or it generate a circuit the output will be a even parity. Any stuck at faults basically anywhere as in the cut or parity generator will lead to so called odd parity it will be detected.

Now, so it seems that parity is a very good way of handling it just add one bit parity and your job is basically done. So, I even we are applying any input patterns if I observe any problem in the output your job is done. But, before going to this we all know the parity based coding technique has some kind of issues what they are. Say for example, there is a circuit something like this there are lot of circuits over here and I mean I have a parity stuck at fault over here and I am using a parity bit encoding and may be this is a gate which is a fault over here. So, it is output of two circuit cones, cone means I mean this circuit you representing these are gate whose over the output primary output of this part of the circuit depends on this another primary output also depends on the same gate.

So, let us say that due to this fault it changes the output from 1 to 0, right and the same time as this gate is depends also depends also controls another output primary output of the circuit. So, also it changes the values from 1 to 0 and we are assuming that there is a also a one more line which is always giving you the one which gives a answer is basically correct. So, you should have got the answer as 1 1 and 1 right, but due to the modification it will be 1 0 0. And we are assuming that we are having a or as in the case we are using a even parity. So, here also you were using even parity so, also we are having a even parity over here.

So, even parity means which will be generating a 1 over here and also it will be generating a 1 over here, this is the error case and this is the normal case. So, this is your expected output say that it is a all ones are there. So, this will be the parity bit will be 1 to make it as a 1. Again due to error what happens basically this 1 is anyway generated and this 1 is again this one should be 1 1 1, but actually the output you are getting as 1 0 0. But, still again it is an odd parity even parity sorry the even parity 1 1 1 is a even parity so, I have to add a 1.

Here also one is added because the output was expected to be 1 1 and 1, but due to fault there are two bit flips. So, two bit flips from 1 to 0 and 1 to 0. So, even that is what and again you are not able to detect this why? Because parity is a single bit error detecting code I think is a very well known theory. So, what happens if there two bit flips? So, one bit flips which is in the case you can very easily detect because it is changing from even parity to odd parity, but here are two bit flips in this hypothetical example 1 to 0 and 1 to 0.

(Refer Slide Time: 30:37)

Error detecting codes used in self-checking design:

Unordered codes

- Multibit error of type either $1 \rightarrow 0$ or $0 \rightarrow 1$ but not both is called Unidirectional errors.
- **Unordered codes can detect Unidirectional errors.**
 - *m-out-of-n* code
 - All valid code words have exactly m 1s and $(m-n)$ 0s
 - Berger code
 - The check part is the binary representation of the number of 0s in the information part. If $I=1001001$, then $C=100$.

So, if there are two bit flips basically you cannot do anything because the parity is maintained. So, if there 3 bit flips you can easily detect, 2 bit flips you cannot detect because still the even parity will be maintained 4 bit flips we cannot do that. So, we all know that parity is basically a single bit or even number or odd number of errors which is can detected.

So, but it can always happen that there can be a gate whose who is controlling two outputs of the circuit and basically it is affecting both of them. So, in that case parity circuit parity error detecting codes will be a huge aliasing in fact, that is true. Parity technique has lot of a aliasing properties, so in many of case the fault coverage will be very very less even less than 50 percent or even lower because wherever there is a fault which is having multiple gates and there is and we also sometime call this unidirectional

error that is from 1 to 0 and 1 to 0 same direction in the error happens. So, the parity is balanced and you cannot be able to detect it.

So, people also we will first tried with parity bits parity way of detecting circuit faults online, but then we find out lot of aliasing property as a expected and then they started moving to more advanced kind of error detecting codes. So, we will see there, this as I told you the literature is huge, but today we will see another one which is slightly better than with parity bit parity technique. Before that we will just give you a small definition.

So, basically one definition is called multi bit error. So, what is actually a multi bit error means the there is a change from 0 to 1 or 1 to 0 and it should happen in more than 2, 2 or more number of bits very obvious. And there is something called unidirectional error; unidirectional error means basically either the error will be in all the bits will be from 0 to 1 or 1 to 0, but there cannot be two lines where one is the error here and one is the error here. So, we will see later today that multidirectional errors are very very difficult to be detected, unidirectional error means both way 0 to 1 or 1 to 0, but there are multiple bits can have it.

But again I have seen you have seen in parity that in parity if there are two such unidirectional errors also it cannot be detected. Unidirectional means all 0 to 1 or 1 to 0 same direction in the error changes. And I should tell you that why it basically why such means why such type of failures are slightly less because as means will have predicted that unidirectional error are more possible end circuits there statistical finding.

So, bidirectional errors or multidirectional errors means some of the bits will go from 0 to 1, someone going to 1 to 0 then extremely difficult way of such type of faults are very very difficult to detect and you require a very sophisticated coding theory will do this. And more sophisticated test codes you are going to use more sophisticated will be your code generator circuit will be and your checker circuit will be. I can do that, I can use a very very advanced codes and I can find out that it will detect multidirectional errors multi bit errors, but again this size will be larger and your exercises will be few type.

So, basically, so today we are going to look at a simplified case in which case we are saying that it is basically nothing, but a multi bit error is possible, but again as you are again repeating multi the bit error is not possible to be detected in parity 2 bits not possible, 4 bits not possible even if it is the unidirectional error. And multidirectional

errors of course, if one bit goes from 0 to 1 and one bit goes from 1 to 0, so parity cannot do anything because the number of 1s are balanced, correct. So, because there is will be no parity change but, so anyway. So, we have already discuss the disadvantages of parity. Parity is mainly basically for odd number of bit changes that also unidirectional, right.

So, now, we have been focusing on slightly modified slightly making the problem bit tougher where there are bit flips, but the bit flips are unidirectional all from 0 to 1 or 1 to 0, but it can be in the more number of bits. So, multi bit unidirectional error is what you are going to consider. And further the parity bits of course, will not hold because already we have given the example because, if two bit flips are there then it will actually change the for it will still maintain the odd parity or even parity, correct.

So, now, basically we will see that there is another type of codes which can handle multiple bit, but unidirectional codes. They are actually we called unordered codes or unidirectional to detect unidirectional errors the codes are called unordered codes and some example of unordered codes are m-out-of-n code. So, m-out-of-n code means if this number of bits are say 1, 2, 3, 4, 5 so, out of that we can say 4 out of 5 code. So, 4 out of 5 code means basically only 4 will be 1.

So, this is one way out of n codes of 1 1 0 1 and so forth and finally, you will have 1 1 1. So, any of the 4 bits will be one among the 5 codes this is actually called 4 out of 5 code. That means, if the circuit does not have a fault the output will always be following this code of 4 out of 5 and if there is an error it will be having some other different codes which will not be 4 out of 5.

Another is called Berger codes; in Berger code basically is the binary representation of the number of 0s in the information part of the main output of the circuit. So, in this case number of input ones are 1, 2 and 3 so, the Berger code will be sorry 1, 2, 3 ok. So, it be the output should be 0 1 one not 4. So, as many number of ones in the output sorry in this case is number of 0s, so anyway you can take a either 0s or 1s. So, number of 0s are 1, 2, 3 4 so, the value of C is 4 or you can also take the number of 0s also.

So, this will be the information part and instead of the parity part or some other part or that is the code part we have to put the binary component, this is also a unordered code. We will tell you what is the meaning of exact meaning of unordered codes right now? But, for the time being we have to just learn the fact or till now we have carried out the

fact therefore, parity bit the problem is it cannot have multi bit multiple bit errors may not be detected and of course, multi directional faults cannot be detected at all.

And also we will see later that multidirectional faults are very difficult to be detected, but still can we try to do something about multi bit unidirectional errors. So, in that case we are trying to focus on a new type of code coding theory which is called unordered codes. We will see the definition, but before going to your definition of an unordered codes basically, we are basically telling you about two code coding styles one is Berger and what is m-out-of-n code. So, I think we just remember this then we will go to the next slide.

(Refer Slide Time: 36:47)

Error detecting codes used in self-checking design:

Unordered codes

- No two different code words X and Y such that X covers Y .
- X covers Y means that X has a 1 in each bit position Y has a 1.
- For $X=11000$ and $Y=10001$, then neither X covers Y nor Y covers X and $(X, Y) \in$ **Unordered code**.
- If $X=11001$ and $Y=10001$, then X covers Y and $(X, Y) \notin$ **Unordered code**.

So, what is a unordered codes? Unordered codes are nothing but there are two codes which cannot one cannot cover the other. So, what do you mean by a code covering? Say for example, I have the numbers like this one, 1 1 0 1 and the other number is basically 1 0 0 1. So, in this case you can see here 1 1, here 1 1 and basically this is a 0 and this is a 1 over here that means, this guy which is x has one in all the positions in which the value or the vector Y has a 1. So, X has a 1 here, X has a 1 over here, here 1, 1 is extra and it is a 0 over here so, you can say that X will cover Y.

But, of course, means the idea is that X covers Y means that X has a Y 1 in each bit position where Y is 1, so X will cover Y. But, unidirectional codes are something which nobody covers the other. For example if you see X is 1 1 0 0 0 and y is say 1 0 0 0 1. So,

in this case you see this is X and this is Y. So, this guy has a 1 over here and it is a 0 over here this are matched and this is having a 1 over here. So, you can see here Y is having an extra 1, here X is having a extra 1. So, X and Y cannot cover each other so, we will say that X and Y in this case are unordered codes.

Now, what is the beauty of unordered codes? You can very easily appreciate the fact over here that multiple single direction bit flips cannot can be detected by this code. So, again I want to reemphasize that in coding principle based online testing the output of the circuit will always be one code word. So, if I take these two as the code words say which are covered. So, 1 1 0 0 1 and 1 is 1 0 0 0 10 so, of course, as you have seen this guy is actually covering.

So, now, what happen? Due to some kind of error you again I am telling unidirectional error. So, what can happen? So, unidirectional is a multiple bit flips so, but single direction. So, let us assume that this guy is changing to 1 and again we are saying that multiple bit will 0 to 1 and so again we can say that this guy is changing to 1; let me just take and take the 1. So, 1 1 0 0 1 and this one is 1 0 0 0 1, right so, X is basically covering Y.

And now let us see that what is the case? So, we will see that basically if the multiple bit flips then what will be the issue. We will show basically that unordered codes will can withstand the effect of multiple bits unidirectional error that is what we are going to see in elaboration. But if there is the as I told you if, so this one will not be a able to tolerate a unidirectional bit flips multiple bit flips.

But, if the codes are something like this in which nobody covers each other it will be easily able to resist the multiple even, if the multiple number of bit flips unidirectional bit flips that is what we are going to show with examples. So, I think till now we have I mean if you are with me. So, we are looking at some kind of a advanced codes compared to parity which can withstand multiple bit flips, but unidirectional and for that we are going to use the unordered codes.

And of course, I as you have seen that this is actually a unordered code you can easily appreciate. Now, of course, you have to appreciate that this m-out-of-n code I am taking an example of 4 out of 5 codes you will be appreciating that this is a unidirectional code. Why? Sorry it is a unordered code. Why? Because you see all the lines all the bits all the

vectors will have at least four 1s. So, you can see that this one is this one is having a 1, this having a 0, here this is having a 1 and this is having a 0. So, what it means? It means that basically none of them can actually cover each other.

Similarly, if you look at it last two bits, so these two is having a this one. So, the last two bits basically if you look at, so this one is having a 0 and this having a 1 and this is having a 0 and this is having a 1. So, none of the vectors can cover each other. So, hot 1 encoding, hot 1 means only one bit is a one m-out-of-n code I have given the example with 4.5, none of the code words can cover each other.

So, basically what we have said, we have we have basically what we are discussing over here is something like. So, the that means, the m-out-of-n code basically is a unordered code because none of the bits are covering each another. That we have explained using 4 out of 5 coding technique, that is the m-out-of-n. And by this definition we have looked at that this is a Berger m-out-of-n code is a unidirectional is a unordered code.

Now, what remains we will try to develop the philosophy that such unordered codes can detect unidirectional multiple bit errors. So, indirectly we will show that if you are having if you are saying unordered codes like m-out-of-n code, Berger's codes and apply to circuits. So, if it happens that a fault will lead to multiple bit unidirectional errors this can be detected.

So, it will be a much better efficient compared to a parity code. Why? Because the parity codes can only withstand mainly single bit flips and if there even number of bit flips are there even if there are unidirectional it will not be able to do it because the parity count is maintained ok. Same example we are going to do and we will try to get the philosophy.

(Refer Slide Time: 42:07)

Error detecting codes used in self-checking design:

m-out-of-n code

- Example of generating self-checking design of Half adder using 2-out-of-4 code.

x	y	s	c	u_1	u_2
0	0	0	0	1	1
0	1	1	0	1	0
1	0	1	0	0	1
1	1	0	1	1	0

Table: Truth table of half adder where output encoded with 2-out-of-4 code.

- $s = x' y + x y'$, $c = x y$, $u_1 = x' + y$ and $u_2 = y'$

So, in this case same half adder you are using, but instead of basically using the parity bit here we are using a 2 out of 4 code so, only 2 bits will be a 1. So, 0 0 means your option is nothing, but you have a only the option of having 1 1. 1 0 so you are using the option of 1 0 over here. 1 0 again you are using this one, this two we have we have bit careful because the two numbers are same as the output 1 0 and 1 0, but you are using the code of 1 0 over here and 0 1 over here as the code. But, again if you count there only two ones in each position; again it is a 0 1 they have put the value of 1 0 over here.

So, basically this is basically your information part and this is your code part. Now, you see so, in case of parity the extra bit was 1, here the extra bit is 2. So, the I keep on always telling that hardware cannot create any match. Hardware is always better performance you want you have to pay a cost. So, parity you are one can detect only single bit errors. Here you are detecting multiple bits errors single unidirectional errors. So, you are using some kind of a code techniques which are unordered codes, still use the number of bits are entries output bits for the parity was 1. Now, it is basically two of course, some benefit you are going to get, but again of course, the cost etcetera will be higher.

So, again this is the function for sum, this is the function for carry and the two bits in this case. So, this is the function for U 1 and this is the function for U 2. Now, you look at it so, if you look at the parity circuit there was just a single gate required for this. Now, for

this one you see you require a quite larger circuit to check for to check bit generation. In this case basically is checked bits, right so, more facilities I want I will require a larger circuit. So, here you may argue that this check bit generation circuit is much larger than the cut.

In fact, this is a small example so it looks like that but if there are in case of real practical circuits the technique of error detection codes of the Berger or m-out-of-n codes are much much lower than the circuit itself. So, if you take a large circuit which is automatically happens after a synthesis of the circuit. So, I apply a 0 0 sequency I have a applied 0 0 over here. So, basically the output is normal case will be 0 0 and the parity bit generated will be sorry the I mean or the means self checking codes generated will be 1 1. So, in this case the output will be 1 1 as you will see over here will be generated by the circuit. So, it is 0 0 1 1 so, the checker finds that there are two ones out of it so, it is detected.

Now, so again apply a stuck at fault in this case. So, I apply a stuck at fault if you apply a stuck at fault you will find out that this parity generator circuit will always put 1 1 is the output because the expected output was 0 0, but because of the faults this bit has been basically changed. So, this bit has been a change and in this case you are going to see that the three 1s are coming as the output which is violating 2 out of 4 codes and the checker will basically detect an error.

So, this is the so, in this example if you cannot exactly find out what is the benefit because your both detected using stuck at using parity as well as you are detecting using a m-out-of-n codes the 2 out of 4 codes then you can ask me what is the benefit.

Just as a homework you can try doing it. You take a as I told you take a circuit where is a gate correct and basically there are two flips out of it. So, the fault over here is basically making a flip of 1 to 0 over here and 1 to 0 over here and assuming that there was a 3rd output as I told you which is a 1 and you take even parity. So, in this case the parity will be generating as a 1 and in this case also now this one is a 1 and in this case also parity is a 1, but as the two bit flips parity will not be able to detect.

Now, you go for a m-out-of-n code, is you take m-out-of-n code. So, basically in this case you will not be a parity. So, in this code this take a let us take a m out of code. Then assuming that it is a basically 3 out of 4 that you can ensure so 1, 2, 3. So, 3 out of 4 code

means basically or you can take 3 out of 5 code so, this is the 1st output 2nd output 3rd output. So, all are 1 and this is the code words are all 0 0 because we are assuming that basically it is a 3 out of 4 code.

Now, basically in the normal mode the output is going to be a 1, as simple as that and basically as expected it will give the value of 0 0. Now, you see the error will be incurred. Why? Because the number of 1s is 10 0 0 0, so only 1 is the output. So, parity will not be able to detect it, but the m-out-of-n code in a in this case the m-out-of-n code is 3 out of 5 it will be able to detect it. Why? Because normal case the answers will be number of ones will be 3 and the fault gets the answer will be actually 0.

Again I mean just again appreciating the fact even if this bit also becomes a 1 sorry, even if this is the 4th bit we are considering. So, if you are taking the 5th 4th 1st bit basically or the 4 output even this also gets flipped from 1 to 0 that can also be detected because 1, 2, 3 all 0s basically output should be basically 3 ones out of 5. Again this is very important so, again I am just repeating for the sake of understanding in a very different manner.

So, in this circuit as I told you the difference is not actually being captured. The idea is why because in this case even if it is stuck at one the this is fit into the AND gate so, the effect of the fault is not observed. Like in this case also this is a stuck at 1 and if you look at it this wire is also going to the AND gate. So, AND gate stuck at one does not get into a picture for this input case, right. But as I told you what and what you can try out basically something like you take some other inputs maybe this in circuit or some other circuit you try to take a picture where there is a single gate, but it will affect two outputs error time, also you should have a third normalizing output.

In this case it is only two outputs are there both will be affected, but in your example you can take a another output which is not be affected. And assuming that the output should be normal case should be 1 1 1 and parity I have already told you and you are using a 3 out of 5 code so, you put a 0 0 as the code. Fault as I told you it may have it flip from 1 to 0 it can be have the flip from 1 to 0. So, and this is your normal one normal bit which is not changed at all. So, as the bit added were the coding will be 0 0, because the expected answer is 1 1 1. So, I had been basically detecting the error because the number of one counts changes.

Similarly, we can also appreciate the fact that basically for parity it cannot be called. Same thing also can be tried out in the example, but you just check for a different test data. So, our now one idea is there saying you say in a and you can ask me that I am telling you that for the same circuit for different input pattern the fault will be alias. Then I should worry or I should not worry. In this case I should say I would not should not worry because in this case I require at least some of the test patterns should detect it.

It may not be always possible that all for all patterns the fault will be immediately detected. If you want to do that the cost will be exorbitantly high; high means what for as I told you for this circuit for this scheme or even for the or let me take the not the good scheme basically that is the parity scheme if I see I can tell you that for this input pattern for this fault this even by the single parity bit it is detected.

But, you can easily find out that in this circuit it may happen therefore, some other patterns basically the fault will not be detected by the parity scheme. Then you should ask whether I should worry basically it is a matter of fact that you should worry or not it depends on the 7 circumstances because I know, therefore, at least this pattern 0 0 the fault will be detected by the parity scheme.

But, it may not be detected by for some other patterns say 1 1, it may happen then basically what should I do. Of course, it by the pattern 1 1 it will not be detected because this stuck at 1 over here and the fault will not be activated. But should I worried because here you want to understand that here I do not have a any scope to apply the pattern of 0 0. So, if the pattern 1 1 comes the faults will obviously, not be detected by any scheme even forget about parity because the fault is not sensitized at all.

But, again I have no scope of directly applying the pattern 0 0 I have to wait till the pattern 0 0 comes. So, I should not bit I should not be very worried about it because I am expecting that sometime 0 0 will come and the fault will be detected. Only my main concern should be if there is no pattern which can detect a fault that means, there is for any of the patterns which could have detected the fault the fault gets aliased because of this the error detecting scheme then I should be very very worried.

(Refer Slide Time: 50:31)

Error detecting codes used in self-checking design:

Single stuck-at faults in the circuit can be detected if such a fault results in either a single bit error or unidirectional multi-bit errors at the outputs, and the outputs are encoded using a single and unidirectional error detecting code.

Again repeating this is the very important concept so, I am repeating it again. That given a circuit maybe for some of the input patterns and the what is scheme selected the fault is aliased or the fault is not sensitive even not sensitized at all I do not worry at all or I should not worry that much. Because, as long as there exists some patterns for which the fault will be sensitized and it will be detected by the parity or by the coding technique then I should not worry that much because I expect that the pattern will come.

But, if it happens because I always think that all patterns basically come at regular interval. But, if you are you have if you have made a mistake or there is a problem in which case what happens, basically if you try to think in that direction that you are you have you are landed into a situation where for the circuit none of the patterns will be able to detect the fault. Because of the aliasing properties of the circuit fault or fault getting masked or some other reason as you have shown the parity code or some other code may not be able to take the fault and for any of the pattern the property holds there is a question of be worry.

So, we should always try to think in that manner that there should be some patterns for which the fault will be sensitized and detected by the coding thing then I am happy with it. But, on the other hand we should not we should be very worried if no patterns can do that for the coding technique used that is what is the idea. So, therefore, in a nutshell we have seen that basically we will take a circuit have some coding technique applied and

ensure that at least for some of the patterns the code will be detected then I will be very happy. So, this is one way of handling the online testing methodology.

But again, so in the conclusion what they have said that single stuck at faults in the circuits can be detected if such a fault result in either a single bit error or unidirectional multiple bit errors at the outputs and the outputs are encoded using single or unidirectional error detecting code. That means, it has been shown theoretically if the circuit has been encoded using single error detecting code that is parity or unidirectional multi bit code like m-out-of-n code or unordered codes.

Then stuck at faults will be detected if they lead to a single bit output fault or multiple bit multiple bit unidirectional fault. But again if I change this statement then if a fault results in multiple bit non-unidirectional fault that is one will be going from 1 to 0 another will be going from 0 to 1 such situations also happen, but that is very very difficult to get detected by simple error detecting codes very complex techniques exist, but that will make the circuit tester avoider very very large. How will you handle such kind of non-unidirectional errors? That we will try to focus in the next lecture.

Thank you.