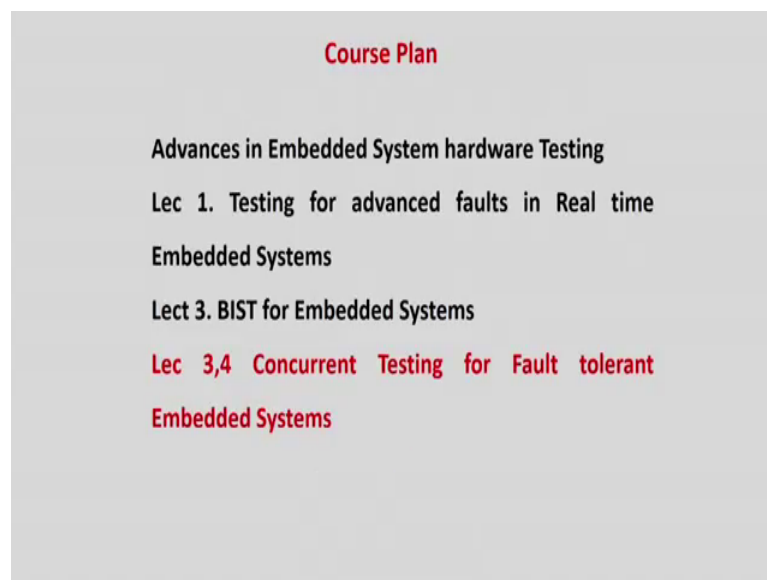


Embedded Systems - Design Verification and Test
Dr. Santosh Biswas
Prof. Jatindra Kumar Deka
Dr. Arnab Sarkar
Department of Computer Science and Engineering
Indian Institute of Technology, Guwahati

Lecture – 35
Concurrent Testing for Fault tolerant Embedded Systems - 2

So, students, hello and welcome to the third part on the Embedded System Design Verification and Test course and we are dealing with Embedded System Testing.

(Refer Slide Time: 00:35)



So, as we are dealing with the advanced part of testing as of now. So, in the last lecture we have covered a part of concurrent testing for fault tolerance and embedded system; that is after looking at the advanced fault models and built in surface then we started looking at why online testing are or concurring testing is very much required in modern embedded systems. Because modern embedded systems as you know are very much requiring all mission critical applications like avionic crux control etcetera.

So, if any fault happens before they start up of the device then we can detect by best and if it is in a manufacturing defect then we can do a test at before the by an ATE before the chip is shipped to the market. But, if anything problem happens why it is in operation

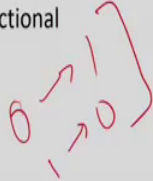
then this is only one way to detecting which is called online testing or concurrent testing and that is very much required for fault tolerance.

And, we talk we discussed in last lecture that we will be giving two lectures on this and already third one we have covered in the last class.

(Refer Slide Time: 01:33)

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.



Today we are going to look at the second part of that online testing business. So, what we had basically seen that mainly when you are going to get online tests there are several schemes like signature monitoring in final state machines and then mainly we looked at upon major emphasized on basically something called self-checking design in which case the inputs and the outputs are all coded using some error detecting quotes.

And, then in the output we have to find out whether if there is some error or fault in the circuit then the output will not be belonging to that particular codeword like if you have a even parity. So, if there is a fault the output will be an odd parity and so forth. But, then we have found out that parity is not a very good way of handling online test mechanism because parity is basically checks one bit error or if you have odd number of errors basically in the output. But, main problem is that we have seen that if there is even number of errors or bidirectional errors. Like bidirectional means, 0 to 1 line will be from 1 to 0 to 1 and 1 will be from 1 to 0 or if the even number of bits get change and parity cannot do anything.

Then we looked at something called unidirectional code that is the amount of encodes or Berger codes then they have slightly more power than you are basically your parity bit error detecting code because they can detect multiple bits even odd number of bit errors are possible to be detected by this unidirectional codes, but of course, then the number the error should be unidirectional that is either all bits become 0 to 1 or 1 to 0.

So, basically what we have seen that single stuck at faults in the circuit can be detected if such a fault results in either a single bit error; signal bit error is very easy to detect basically by any parity mechanism will be detected it will be detected or unidirectional multi bit errors at the output. So, unidirectional means again I am emphasizing that if the errors are all from 0 to 1 or 1 to 0 then it is easy to detect. Extremely difficult cases happen if they are bidirectional errors that is one goes from 1 to 0 and the other line goes from 0 to 1 that actually balances the errors.

So, in fact, very is very difficult will take such errors and extremely complicated code coding may be used to do it. But, then the tested area will also become very high here one thing is very important to understand that I can do online testing and I can go for even directly multidirectional errors bidirectional errors everything I can do, but if for a circuit of size x I am putting another circuit of size $10x$ or something to detect the errors online that is not a very good attitude. It is like something like killing a mosquito with a cannon. You have a circuit size is x , you have to use slightly smaller circuit or at least with a much lower area than x which can detect online errors in that.

Basically the idea is that if your one circuit under test of x area and you have the tester whose circuit area is small then I already told you in the last class then probability of happen probability of errors occurring in that smaller circuit is lower than the larger circuit. Because in case of online tests or built in self test both the circuit as well as this tester on chip.

So, same chip basically both of them are there is not like an offline test with an ATE is there and the tester actually test the chip, but in this case both the circuit as well as the tester are on the same chip. So, faults can happen either in the circuit or in the tester itself. So, tester has will very carefully built. So, it should be have a smaller area compared to the cut. So, then you can assume that the probability of faults occurring in the testers are less. So, we have to always try to keep the tester area small.

So, basically that is why if you have thinking about multidirectional errors if we extremely difficult codes are available or lot of modification in the circuits are required to detect such errors and it may require the circuit tester where to become very high. So, we when we talk about online testing we are very much scared about something what is called as a bidirectional errors.

So, what we have learned in the last class that single stuck at faults if the result in a single bit output error or multiple bit unidirectional errors, then there lot of codes available to detect it and the circuit area requirement for designing the testers are also very less. So, one important one is parity, but as again discuss parity can detect only single bit or odd number of bit errors.

(Refer Slide Time: 05:30)

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**.

But, slightly advanced are basically you already seen unordered code basically again I am just giving a repetition of yesterdays lecture because we will start continuing from this point. So, the recap we have seen there is something on a unordered code. Unordered codes are something like one should not cover the another that is basically if you look at it. So, in this point this is a 1 and this is a 0, but another in this point 1 minute. So, basically the idea of a unordered code is something like this one, that is, this one will have a 0 and this is a 1 and this point. So, this one is going to cover this and in this case this one is going to cover this. So, nobody is going to cover each other. So, therefore, in

this case basically they are actually call unordered codes neither X covers Y nor Y covers X.

(Refer Slide Time: 06:14)

Error detecting codes used in self-checking design:

Unordered codes

- Multibit error of type either 1→0 or 0→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, we have seen that basically the formal although we have not discussed basically, but unordered codes can be used to detect multiple bits unidirectional errors the proofs exist. One thing at this point I would like to tell you this coding theory was basically used or you or massively used in engineering for some other purpose that is a communication error.

So, whenever we are communicating some bars of bits over a medium like wired medium or more important in the wireless medium. So, maybe you are transmitting a packet where this a header let this is a network packet there is a footer and lot of 0 1 1 0 1 some bits we are transmitting. And, your transmitting over the medium may be a air is a wireless communication due to lightening etcetera what happens basically some of the bits get flipped. So, say the bits are flipping through the air and then this a lot of lightening or some disturbances happen then what happens the lot of bit get flip to one direction.

So, what happens basically let us assume that the numbers or something like this is going is a lightening. So, everybody is pulled to 1. So, what happens, this one will also become 1 this one will also become 1 and all others bits will be 1, because of this closed coherence nature. So, if some data is flowing from one part to another over the wave and

these are lightning or noise which comes then try they try to flip the bits in one direction because it is it is not because of a error as in the circuit. In the circuit what happens the error is a stuck at fault or some fault in a gate which gets reflected in the output in some case of bits getting flip from 0 to 1 or 1 to 0.

But, coding theory was mainly used for detecting errors in the communication channel communication channel the errors does not happen because of faults in the wires or something. It basically happens because of interference or some lightening or external noise. In such cases what happens basically all the bits tried to get biased by the neighbors or by the noise and they go to try to go in one direction. In other words to keep the long story short errors in communication channels are generally unidirectional that is all become 0 to 1 or all becomes 1 to 0 based on the influence of the neighbors or based on the lightening effect or some others for called interference.

But, that phenomena is not true for circuits because circuit you have one point these a stuck at fault and it can reflect obviously, in multi directional errors at the output that is why this coding theory basically are widely use for online testing, but still it does not is not a very good match basically for online test because it results in multidirectional errors in circuits. But from where the branch of engineering from where the motivation has been borrowed is communication channels. There the probability of occurring of multidirectional errors are less as I have explained, detailed theory you can read any communication textbook and it will be clear, right.

And, also in that case the error detecting hardware is present at one part of the channel. So, for example, this is the transmitter and this is the receiver in the receiver end of the means of the channel you will have a error detecting hardware. So, in this case there is no limitation in the area of the hardware. You can use an extremely complicated coding theory and you have a hardware box to test it test that whether transmission was proper or not because the communication channel which are considering as the device under test and the tester are two separate entities and they are not in a single point, but in case of circuit the challenge is much higher because the circuit as well as the tester are on chip. So, you cannot use a very complicated coding theory to do that.

So, therefore, again I am repeating in case of online testing bidirectional errors are basically nightmares. So, we cannot use a very complicated code to do it. So, people try

to handle it in a different manner as we are look in this codes. But, what people do many people assume that I will only detect unidirectional errors bidirectional errors I will forget these are because basically that will make my tester very difficult and as already I told you mostly your testing is done after design and manufacturing, then bits is slightly lower, online testing is further lower. Because in online testing you cannot test as vigorously as the manufacturing test in a ATE then if you do that your tester will become much larger in your circuits itself. So, this is a pyramidal structure basically.

So, therefore, I can I have to compromise many things in online testing. So, many people tell that we will design a tester, but it will only detect unidirectional errors. So, you have simple coding theory to do that. So, basically what was the motivation? So, basically one error will be converting from 1 to 0 and the other will be from 0 to 1, but they there will be there cannot be in a mix. This either all bits will be from 1 to 0 or all bits will be changing from 0 to 1. So, that is actually a unordered code.

Now, I am not going into the exact proof because as I told you this coding theory has come from communication error and the lot of theory proofs are there why it will work there already proof guarantees are there. The proof guarantee says that if there unidirectional errors unordered codes can detective the formal proof you can read it, but with are not going to bring it in the testing VLSI testing or embedded system perspective because I as I again I am repeating we just take the proofs and we just use this concept for testing.

But, slightly some idea I will give you very important coding unidirectional codes are m out of n codes already we have used and basically Berger code. So, Berger codes basically m out of n codes is exactly m number of 1's and m minus s number of 0's. The Berger code basically you have a information part the number of zeroes are counted and you put a binary count for that. In this case there are four 0's 1 2 3 4 and three 1's. So, C is the coding part that is you will be representing the number 4 because there are four number of 0's.

So, now you can be very easily appreciate the fact as I will show you that in there unidirectional errors they can be very easily detected, right. For example, I am taking a hot one coding; one out of encode. So, let me say that is the number sorry it will be only

1 0 0 1 dot dot dot. So, these are the only course possible I am and here I am taking m equal to 1 that is only one 1's in the entire binary vector.

Now, I can tell you there can be very easily you can appreciate that if there unidirectional errors, what is going to happen. Unidirectional happen means 2 1's 2 0's will become 1 or one 0 can become a 1, but you cannot have this 0, becoming 1 and this one sorry you cannot have two different ways like this 1, this 0 becoming 1 and this 1 becoming 0 that cannot happen basically, but be careful that if it happens if the error cannot be detected, right one is a coding like this sorry. So, this is one code like 1 0 0 0 0.

So, if is a bidirectional error this is going to become a 1 and this is going to become a 0. So, in this case this is another the error vector, but still this follows the a hot one encoding, but unidirectional error it is the assumption of non unidirectional errors prohibits this that we are assuming that such errors are not going to happen which will change one bit from 0 to 1 and one bit from 1 to 0, that we are not allowing.

So, if it is not allowed then you can very easily appreciate the fact that either this one will become a 1 or this or both of them or this one. So, it will obviously, change the number of ones. It will no longer remain a hot while a encoding. So, if the any unidirectional error which moves from this one to this one then it will change the number of 1's which will no longer remain a single one, so, error will be easily deleted. Other way even if the other way unidirectional error happen like for example, this one will becomes 0, then the number of ones will be 0 which is also not possible in the hot one encoding.

(Refer Slide Time: 13:29)

Error detecting codes used in self-checking design:

Unordered codes

- Multibit error of type either 1→0 or 0→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, in fact, you can easily try with two ones the idea is very simple if you are using two one encoding. So, in this case if you see that 1 1 then 0 0 0 then it will be again 0 0 maybe 1 1 0 these are the codes. Any you can easily appreciate the fact any type of unidirectional error which is the number of ones and as the number of ones are fixed in the encoding very easily you can detect, but if this a bidirectional error then actually there can be error balancing the main problem of bidirectional error is error balancing bit number balancing like, sorry.

So, this one will change to 1 and this one will change to 0; so, again the number of 1's remain 2. So, errors are easily balanced. So, bidirectional errors are basically very nightmare for online testing. Now, we will see what people have tried to avoid some people as I told you assume that if there is a bidirectional error we are not going to detect it. Simply fault coverage will come down or still whatever is possible to do with unidirectional unordered codes detect initial all fault leading to unidirectional errors will only be detected if some fault results in a bidirectional error it will not be detected, bring down the fault coverage we cannot do anything because we want to keep the tester area low. That is one group of people actually work by this mechanism, but there were lot of scientist who try to avoid bidirectional errors which we will see in a very short while from now.

Similarly, as a homework you can also try to find out that in the Berger code also if there is a bidirectional error. So, is there unidirectional error you can easily detect it, if these a bidirectional error you can easily I mean you can it really get marks you can easily try this out, right. For example, in this case these 1 0 0 1 0 0 1 there four number of ones. So, the count 4 is there if you there is a unidirectional error maybe this one is going to become 0 and this 1 is going to become 0. So, the number of zeros in this case should be increasing. So, here the number will be 4, error can be easily detected.

But, if these a bidirectional error this one become from 1 to 0 and this one become from 0 to 1, there can be error balancing. So, fault will not be detected because still the number of zeros will be 4.

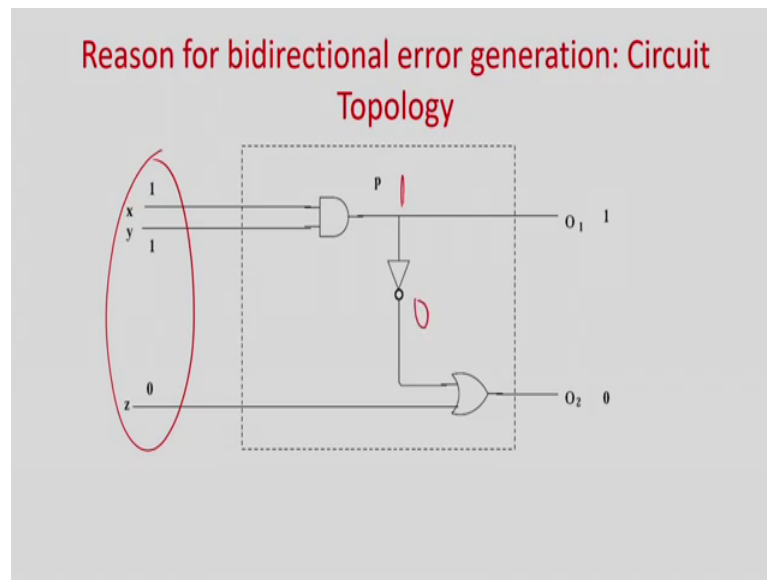
(Refer Slide Time: 15:14)

Solution to bidirectional errors

There is no efficient error detecting code to detect the bidirectional error. Thus all the designers are interested how to eliminate bidirectional errors.

So, basically so, what is the idea there is no efficient error detecting code to detect bidirectional error. Thus all designers are interested how to eliminate bidirectional this is the nightmare. So, we have to eliminate it. So, there are lot of techniques which people have tried to do that I modified codes modified testers and so many other things, but again as I told you the modified coding based techniques to detect bidirectional errors did not get much popularity because they increase the size of the circuit and they put lot of constraints like this type of inputs can be possible this type of inputs cannot be given and so forth. So, if they are not got much popularity and also we are not discussing in details.

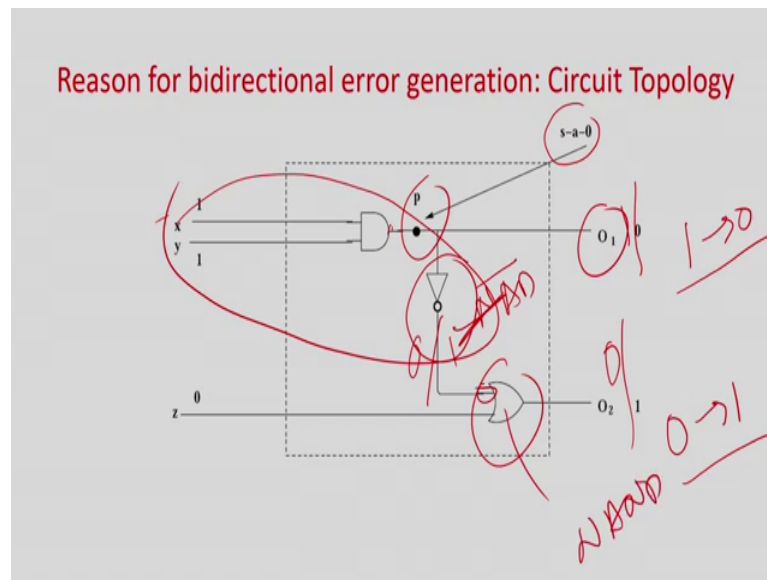
(Refer Slide Time: 15:50)



But, very interestingly some people have tried to do something a very interesting way of avoiding bidirectional errors, by what? They first tried to find out why bidirectional errors are happening. Because in circuits in communication as I told you bidirectional errors are rare because the in the air the packets are traveling due to lightening etcetera all zeros becomes 1 or all 1 becomes zeros. Bidirectional errors do what happen in communication, but in circuit is very easily possible as I will show you and then people try to find out how we can avoided and then this another direction of research we try to design circuits which will not have any kind of bidirectional errors, right.

So, first people try to find out why this bidirectional errors occur. We are taking a very simple circuit like a AND gate, NOR gate etcetera. So, these are the inputs and these are the outputs very easily you can find out basically. 1 1 the answer is the 1. So, for inverter 0, 0 and 0.

(Refer Slide Time: 16:38)



Now, let us take a simple stuck at fault over here. Now, at the simple stuck at fault over here what happens basically you can see it actually affects this gain as well as affect this gain. Whenever a fault is affecting multiple number of output lines there is a big possibility of bidirectional errors. So, one thing you can do you can make all circuits very much independent there that is one individual circuit will give a output, another circuit will be giving another output that is more such case will be there that is basically if something like this if I can replicate like, for example, this is one output right. So, I mean we can say that basically this part of the circuit is giving the output for this O 1.

Now, for O for this one circuit is there I basically this is one circuit, I eliminate and I or eliminate the other part. So, one circuit is just like this right it delivers the output of O 1. Next what I can do? What I am trying to tell you is that I am not going to allow sharing of output gates that is for example, this output is dependent on this gate. This output O 2 is basically dependent on this gate as well as it is depending on this. So, there is some part which is common to both the outputs. So, if there is some part of the circuit which is common to both the output then any fault here will affect both the output. If we can avoid it that is very simple, then a fault can affect only one output and then there cannot be any chances of any bidirectional. Even parity code can detect the errors in this circuit because any single stuck at fault can affect only one output, right.

So, what I will do that that is I will try to replicate this circuit so much that there is no common like for given one output there is no common circuitry which is shared by 2 outputs like. For example, I take this one it have one circuit like this part is not there this is for O 1. So, O 2 what I have to do for O 2 what I have to do is that basically again I replicate everything this one I am not keeping it I am I will be replicating this one is that. So, another part of the circuit so, they will be another some circuit will be like this. This is for O 1, correct and this is x and y and another circuit is this one. So, this x, y and z this is for O 2, so, either this. So, basically you can see. So, this is for O 2 and this circuit is for O 1.

So, if a for stuck fault and we are all assuming single stuck at fault. Single stuck at fault occurs here only the O 2 output will be affected O 1 output will not be affected. So, if I keep on replicating these circuits like this or because not allowing any kind of logic sharing between the outputs then you will not have any kind of multi bit error, forget about unidirectional bidirectional any single stuck at fault only one output line will be affected because the circuit is design in this manner.

Some group of people try to do this, but in fact, people found out that if you try to do this there will be lot of replication of circuits because this one for O 1 this is what is required. For O 2 also this part basically of the circuit is common to both. So, why I want to repeat it twice? So, if you are not generally will not repeat if you start repeating the shared logics then the circuit size will be extraordinarily high. But, people have found out that if you can still use have liberty because people are trying to find out different solutions for online testing.

So, if somebody said that you have liberty, so that you for any given output you have no common sharing of gates between two outputs then you can simply use parity codes and you be having online fault tolerance. But, then the circuit size become extremely high and the no designer accepted it because I have made a very small circuit very optimized circuit in area and basically for testing you are externally going it up. So, basically the term is called intrusive design.

I am again putting quote unquote intrusive I give you a design like this then you tell me hey there is lot of resource sharing and this will lead to multi bit errors. So, what I will do is that I will take this part separately as O 1. So, I will put it something like this and

basically this is the output O 1 and for O 2 basically you just have to sorry you just basically have to remove this and this is another part of the circuit which will before O 2, but these replication of these gates. Some of the designers will not like it at all because he is saying that unnecessarily the test engineer is making my design very cumbersome. So, therefore, there is one way of doing it that is no logic sharing between the outputs. So, in that is one way.

Another way is that people try to find out that is one way people from that why bidirectional error circuit and then they try to avoid it by one minute just do not have any logic sharing between the outputs. And, therefore, but people have dragged lot of papers on this direction that you make parity groups and so forth, you can read on this. That is a future research lot of work is still being going on that instead of fully entirely separating the outputs having no logic commonalities between these, can be make groups then we can we do solving by graph coloring problem and n number of papers along there, but the idea is that, minimum logic sharing between the outputs.

So, if there is no logic sharing no output can lead to multiple bit errors and it will be very easy, but that will make the circuit very high. As a tradeoff people tried to basically see that if some groups can be made those literature you can read on. But, then another direction people found out that say this a stuck at 0 fault over here. So, stuck at 0 fault means normally it should be 1 with fault it has become a 0 and in this case so, it is from 1 to 0. This is the flip direction and in this case if you see one stuck at 0. So, in the answer is one normally it should have been 0, it is 1 and here the output is normally it should have been 0, but the failure it is a 1. So, it is from 0 to so, it leads to a bidirectional error.

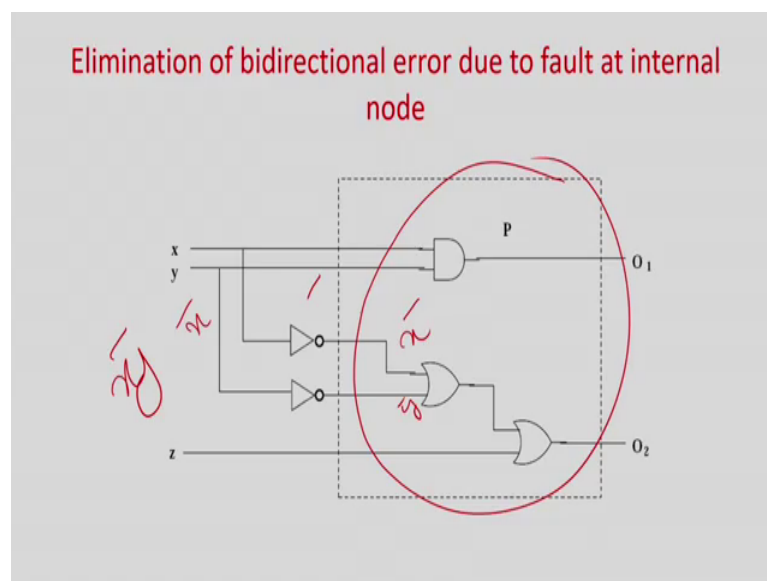
Then people very easily realized maybe bidirectional error is happening because there is a inversion because in this case error direction here basically. So, if you look at it, so, in this case the direction is changing from 1 to 0. So, in this case basically if you see this is the stuck at 0 fault and in this case the direction is from 1 to 0 from 0 to 1. All of this is happening because of an inversion. So, people have found out that main reason for bidirectional errors are inversions.

So, if there are no inversions generally the flip direction is a unidirectional process or people have discovered for most of the topologies. So, what people will try to do is that

they started designing something call a inverter free logic. So, if you just remember this circuit. So, this is what is the circuit is. Now, and people you can easily you will be also appreciate at this actually is creating the problem this inverter is mainly creating the problem and that is why there is a bit flip.

Now, what people have tried to do is that they will not put any inverters in the circuit there is call a inverters free design because from one direction 0 to 1 other line 1 to 0, obviously, because of inversions in this circuit.

(Refer Slide Time: 22:49)



Some people try to use De Morgan's law and put all the inverters out of the circuit and this is verified this is a logic this does not O 1 does not have any inversion logic, so, you can directly put it. But, in this case basically if you look at it this part of the circuit is a inverter logic. In fact, is nothing but you can tell that these a here is a NAND logic. So, this is a NAND logic basically. So, this part is a NAND logic. So, this is the input this is the OR gate 2 is the OR gate number 2 and. In fact, the input is inversion and an inversion which is the second input sorry the first input to this or gate is a NAND logic this is a NAND logic.

So, NAND logic can be very easily implemented by De Morgan's law you invert the inputs and use a OR gate. So, in fact, obviously, you cannot design a circuit without inverters, not at all possible. Of course, because for NAND is a inversion it is required for universal gate design. But, what they will do they try to bring out all the inverter

logic out of this circuit and here we will get x and here we will get \bar{x} . So, I am put some inverters here.

But, in reality the circuit output or basically you assume that there is a machine which will generate x and x complement, that you are assuming that that is error free. Of course, I always keep on saying in all my lectures that you cannot create magic. So, in this case a circuit can only be designed with inverters because inverters are universal logic that involved in universal logic. You cannot design any circuit without inverters.

So, what we have they have tried to do they are not putting any kind of inverters inside a circuit. All the inverters are pushed out by using De Morgan's law. And, but still inverters will be required for giving inputs. So, a basically there is some circuit like this is x and this is \bar{x} there is some circuit which will generate x and x complement and give to the inputs. I will not put any inverters. For just for showing you showing it I put some inverters over here, but if you know that these some device which will keep x and y and also \bar{x} and \bar{y} . So, they can be feed over here that is \bar{x} and this is y .

So, if you can you have you can do as a homework as this no inversion over here there will be no flip flops sorry there will be no inverter. So, no stuck at fault will lead to bidirectional errors because bidirectional happen means one is going from 1 to 0 another is going from 0 to 1. So, there should be inverter logic. So, all inverters if we push out on this circuit then there will be no kind of a bidirectional errors.

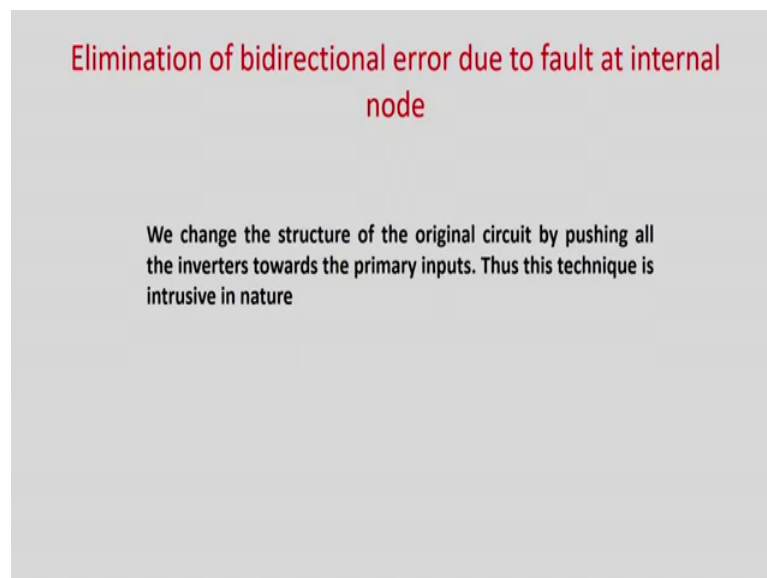
So, if I keep on telling the story there are hundreds and thousands of papers which has come out how to handle bidirectional errors. One as I told you circuit replication that is do not share any logic between the outputs. Another way is that you push out all the inverters through the output side and basically you apply signals x \bar{x} y \bar{y} whatever both versions you have to keep as the input to the circuit. n number of different techniques has been found out which basically use eliminate bidirectional errors because detecting bidirectional errors are very difficult and you require very complicated coding theory to that.

So, group of researchers or most of the researcher in online testing of embedded system case try to push avoid bidirectional errors. Like one time when I am telling you by avoiding sharing of logic between the outputs pushing all the inverters there some of the major techniques. Lot of other techniques are there which you can read references will be

given in the course. So, you can do it, but this I think by this two example you have going to idea that what is a coding theory, how it is applied in circuit to test online by error detecting codes and how what is the problem of bidirectional errors and how they can be avoided some basic techniques or the first line or the premium techniques which we are discovered with at the base line of those online test by error detecting codes I have told you basically.

So, more or less this is for your syllabus that if you want to know more, there are number of research papers on handling bidirectional errors, different type of coding theory, etcetera.

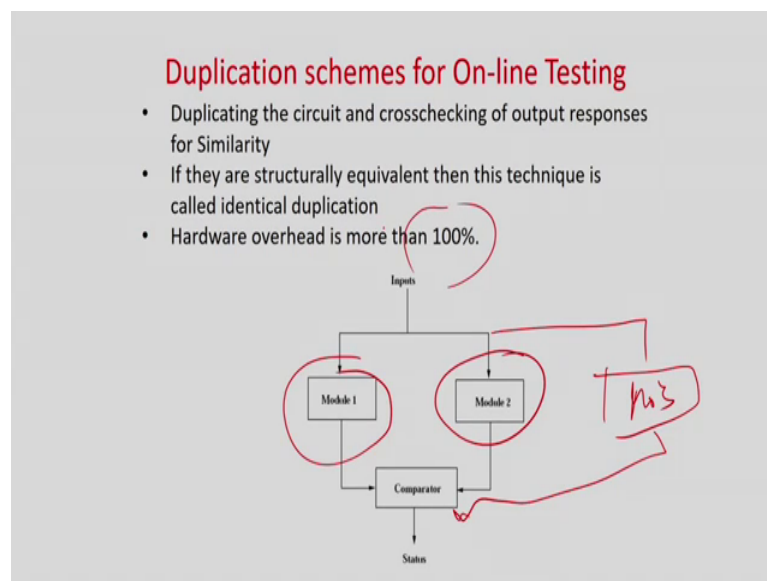
(Refer Slide Time: 26:17)



If you look at it for bidirectional circuit or handling bidirectional errors people avoid it, to avoid it some way you have to change the original circuit. Like one is basically as I told you have to as I told you have to separate the outputs. So that no output any log there is no logic sharing between outputs. Here you have put the inverters throughout the outside and by that you have to apply lot of De Morgan's law in the. In all these mechanisms basically the original circuit will get change and in modern days no designer will allow so much changes to your circuit for getting it online testing. Of course, some changes are allowed, but basic changes like putting all the buffers outside, no logic sharing, nobody is going to take it actually they are called intrusive designs.

So, there is lot of tussle between the test engineers and how much intrusivity will allow. I am a designer I have given a circuit which is optimizing area timing and power. The test engineer is changing lot of change designs in my circuit I may not be like to do that and in fact the changes allowed are very minimal by the test engineer. So, therefore, this in modern community of online testing, this error detecting quotes are not that much accepted. This was some around 10 to 15 years back this was a good techniques to be applied, but nowadays people are not very happy with it because it involves lot of changes in the circuit which we are called intrusive designs.

(Refer Slide Time: 27:34)



So, now with which we stop the basically discussion on error detecting codes we will see how different another group of researchers, how they are handling online testing and their main emphasis is to keep this circuit interact. Then I am not going to change your original circuit, may be slight changes I will do, but very minimal. But, of course, there cannot be any magic. So, these peoples are involving more number of area overheads compared to the bidirectional error detecting code theory.

The best advantage of error detecting codes for online testing is the low area overhead that is very important and test cover is very high. So, if you assume the I will not have any bidirectional errors or I design a circuit. So that bidirectional errors are avoided in that case the fault coverage is very high. But, as I told you intrusivity is not liked by most of the design engineers.

So, some people thought of handling it in a very different manner. So, they started from very fundamental design then how I can do online testing without changing the circuit itself. Very interesting is replication; normal circuit is there you replicate the module and make a comparator. So, if both of them are giving the same output, there is no error, if there is somebody is giving around different output than the other there is some problem where the error is. Of course, you can detective to make a more robust you can put module 3 same thing same input will be going over here and comparator is instead of a comparator you take the best of 2 volts.

So, all no problem all three are giving the same answer no problem. If there is a problem if you take the majority volt the assumption is most likely only one of them will develop the error. But, in this case the area overhead is becoming very high.

(Refer Slide Time: 28:58)

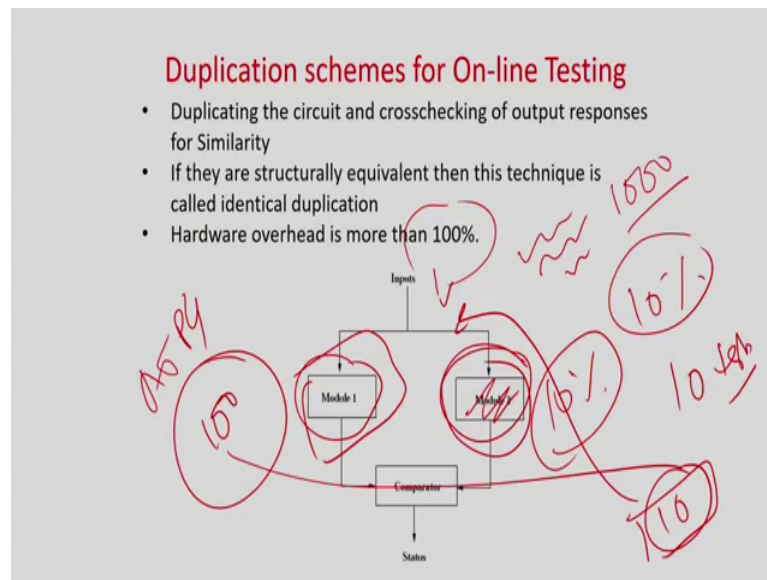
Duplication schemes for On-line Testing

When replicating module is a minimized version of the circuit-under test then this is called partial duplication. The idea of design partial duplication is:

- Generation of test vectors using standard ATPG algorithm.
- Then using a subset of these vectors to design the partial replication circuit.
- Efficiency strongly depends on the prior knowing regarding the input patterns that the circuit under test likely to receive as well as those that occur rarely.
- The advantage of this technique is due to its non-intrusiveness.

Then people started thinking about partial duplication that why we unnecessarily I want to replicate everything because anyway in case of error detecting codes also here making very strong assumptions right only unidirectional errors will be there; that means, I am restricting the stuck at faults which will lead to only unidirectional errors. So, that means, still again in the coding theory also I am compromising on the fault coverage and detection latency I will cover more on detection latency later, but still there is a compromise.

(Refer Slide Time: 29:25)



So, why not I make this module instead of a 100 percent replica I make 10 percent replication then only for the very important 10 very important parts of the circuit which are required to be tested I will make a very small replica, error detecting code also compromising, I am also compromising. So, therefore, they call it basically a partial replication. So, in this case what people do they take a circuit like this, go for ATPG, find out all the test patterns then they use a subset of these vectors to design the public replication circuit.

Like for example, there are 100 test one thousand test 100 test vector say for module 1 to test all the faults and say that somebody says that only 10 percent area overhead I will allow for testing you take the most important 10 percent that is 10 test vectors which are most important that they are have the maximum coverage like in which already I have told you how to select good test vectors.

So, use only those test vectors because in online the what happens all the inputs are going to come. So, at present say that maybe actually 1000 inputs are possible. Among them I have found out that these 100s are enough to test all vectors or in let me tell in other direction. Say 1000 different type of inputs can appear over the circuit. So, if I want to go for a 100 percent guaranteed 0 detection latency everything I want to do it if it just replicate the module then everything will be very high.

But, for example, as somebody has told me I have only a restriction of the only 10 percent area overhead and then if you go for an ATPG then you find out the hundred test vectors are enough to test this circuit. But, among the hundred test vectors maybe only 10, I can keep because the area overhead has to be very small. So, I will take say only 10 percent of 100 that is 10 vectors and I will design module 2.

So, module 2 will be a replica of this circuit, but it be a miniature version because only if this 10 important test vectors are appearing in the circuit I will to the test. Because you cannot apply any test vector online testing vectors are coming. So, whenever this important test vectors come then you have to go for the comparison that is the idea of basically partial replication, right. Take only the important test vectors and basically whenever those vectors are applying coming in the input because you cannot control the inputs then you go for testing.

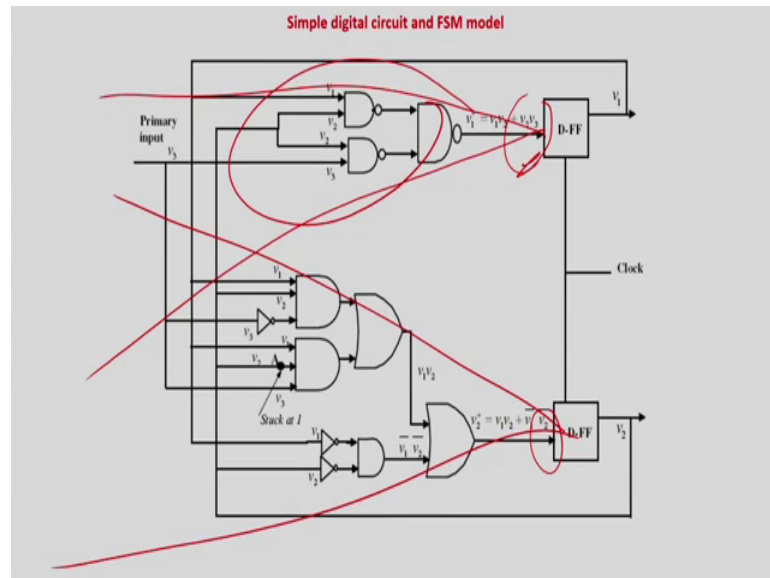
But, if they, but the efficiency strongly depends on the prior knowledge regarding the input test patterns that is circuit is likely to receive as well as that occur rarely because these 10 may be very important test vectors. You design a circuit with this 10 test vectors, but this 10 test vectors never come in the circuit because if I want the test engineer and I have an ATE or a BIST I will applied this 10 test pattern. But, in online testing the inputs come and among there if something is good vector you have to test it or I should not call it good if there are in this module then only the testing is possible because in this module you are keeping a very small subset of the test vectors on which the test comparison can be done.

So, this 10 vectors are very good no doubt there very high fault coverage very good points they tester all these things, but in the normal scenario this 10 test vectors never come. So, in this case you have very good test tester like module 2, but inputs may not arrive that manner which will be testing your circuit. But, if I use a offline test or a BIST then I will apply this test one among the test patterns I will apply, but online testing you cannot apply.

So, therefore, efficiency will drawn strongly depend on the prior knowledge that whether this important test vectors will come. So, the problem is to first we have to find out very good test patterns which is very efficient and secondly, we have to find out whether they will occur at regular intervals or not if it is possible then only you can make for a

efficient partial replication based tester. But, what is the advantage in advantage is non intrusiveness that is very important that I am not going to do any changes in the original circuit. Circuit is there a part of the circuit is there your take making our comparison only for the partial part which is at replicated the testing can be done the advantages non intrusiveness.

(Refer Slide Time: 33:08)

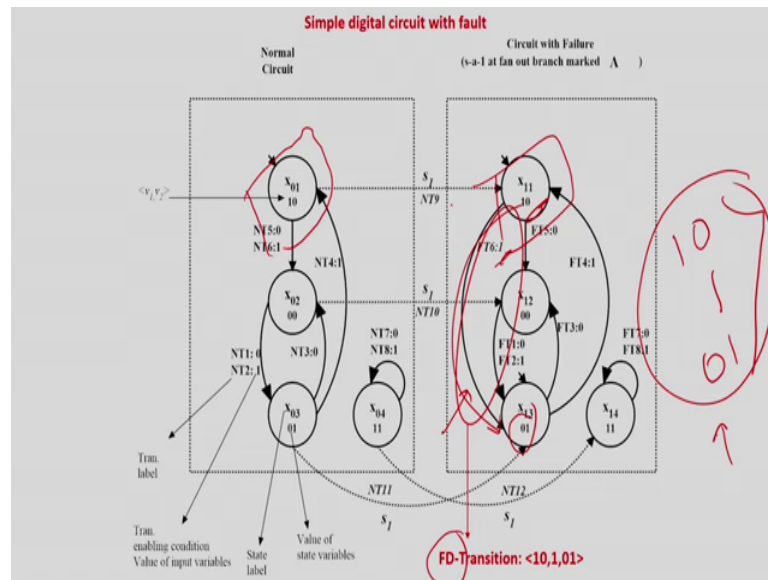


So, again I am telling you nowadays online testing is mainly done by this partial replication based method because no test engineer or design engineer will allow you to do massive changes in the circuit, nobody will to allow you to do that. So, therefore, partial replication is now becoming a defective standard of online testing. Let me without going into more theory let me give you an example.

This is a circuit, you can go over to that and basically there is a fault over the stuck at fault over here, right. So, this output is basically v_1' is v_1, v_2, v_3 and this is the output for the second. So, I in this case again I mean you can just you always try you can see this is one part of the circuit and this is another part of the circuit then something called cone of influence. Cone of influence means what is the sub part of this circuit which actually influences the output similarly if you look at it this is the one output of a circuit and these are the gates which actually impact this output or a control the output. So, this is actually gates are in the cone of influence of this of this output and for this output v_2' this is what is the cone of influence.

So, cone of influence is very simple you take a output of the circuit and what are the gates and innates which actually control the output that sub part of the circuit is called cone of influence for that output since this definition will be using. So, I am just telling you with in brief.

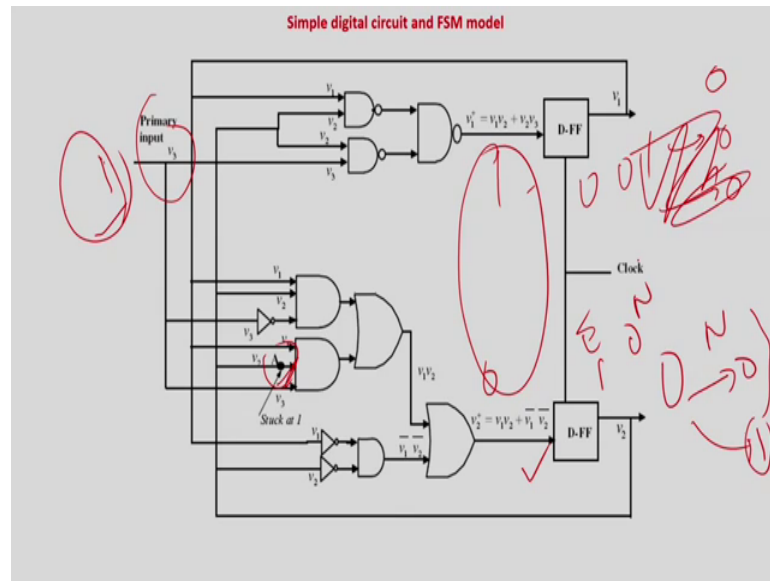
(Refer Slide Time: 34:22)



Now, they make a now I will make a final state machine later I will tell you how to handle this final state machining complexity.

So, they have if you look at it this is the normal final state machine and this part corresponds to the fault final state machine because, now our fault has occurred.

(Refer Slide Time: 34:37)



So, without the fault basically the machine will behave like this. So, they are 1 2 3 4 states basically two flip flops. So, there are 4 states some state may be unreachable we are assuming that the state starts from one 0 and basically with the input 0 which will come over here because these a input is only one that is v_3 and v_1 and v_2 are primary secondary primary input then their feedback from the flops.

So, basically if the v_1 and v_2 are the state register value so, the present state is 1 0 that is if the present state is 1 0 and basically if the input is 0 or 1 we are going to get the next state as 0 0. So, even if is the present state is 1 0 the input is 1 or 0 anything in next state is going to be 0 0 that is what is showing by this like anyway this is very self explanatory a simple digital design fundamentals. But, with a fault now I am going to draw the final state machine with the fault. You will appreciate that you have to clearly do it by analyzing you can easily find out that basically the there is only one transition which is different from the normal compared to the failure.

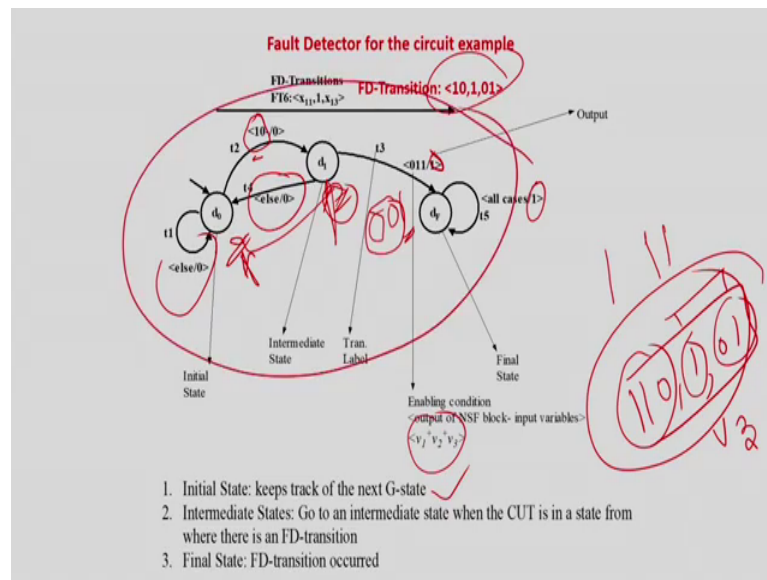
What is that one? If this test if the present state is 1 0 that is present state is 1 0 and the input is basically a 1 the input is basically a 1 then in the normal case the next state is 0 0, but the fault case it is a 0 1. So, normal state the out sorry so, 1 0. So, the normal case it will be going to 0 0, but for the fault case it is become it is going to the next state that is 0 1, this is again fault case will be 0, but this case is going to be a 1. So, this one as I as you when easily see this fault is only affecting this cone output. So, there is no effect

over here. So, in this case the from the state from the present state is basically if you look at it 1 0. So, present state is 1 0 the next state is always going to be a 0 if the output input is a 1 irrespective of the fault, but in this case basically if there is a fault the output is going to become a one and in the normal case the output is going to be a 0.

So, normal case one 0 the input is one is 0 0 faulty case same state same input the output is going to be a 0 1. So, basically we call this as the fault detecting vector or a test vector because that is can differentiate between the normal and the faulty behavior. It is nothing, but your simple normal ATPG you can do and find out the vector with differentiates. This just as the procedure I am going to show you that how they a go for a online tester design by this method partial replication method they are taking this procedure. You can also find this vector the vector is basically nothing, but initial state that is 1 0 is the 1 0 is the initial state present state is input is 1 this input is a 1 and basically the next state is basically 0 1. In this case of error that is 0 1 in case of error normal case it is 0 0 this is error.

So, this one you can easily find out by your normal ATPG mechanism also, but in this the theorem or this theory I am discussing is basically on the online testing design in final state machine. So, they handle it in a different way and then they will remove the complexity, but you can take it from me. But, the same test vector can also be found out by ATPG or by n number of different techniques are available; like normally we have done by scan chain by scan chain method you can do it also by basically you can that is what we call time frame expansion method any method you can use to do it. But, for the time being we can find out the present state is 1 0, input is 1, next state 0 is the fault detecting line. So, that is what actually is going to detect the error.

(Refer Slide Time: 38:00)



Now, they are going to design a online tester to detect it. So, that for the time being let us assume that is only this because in this case there is only one edge which is actually taking the error. So, very easily we have to detect this is the only one which detects the error. So, you have to design a online tester using this mechanism or by using this FD-transition.

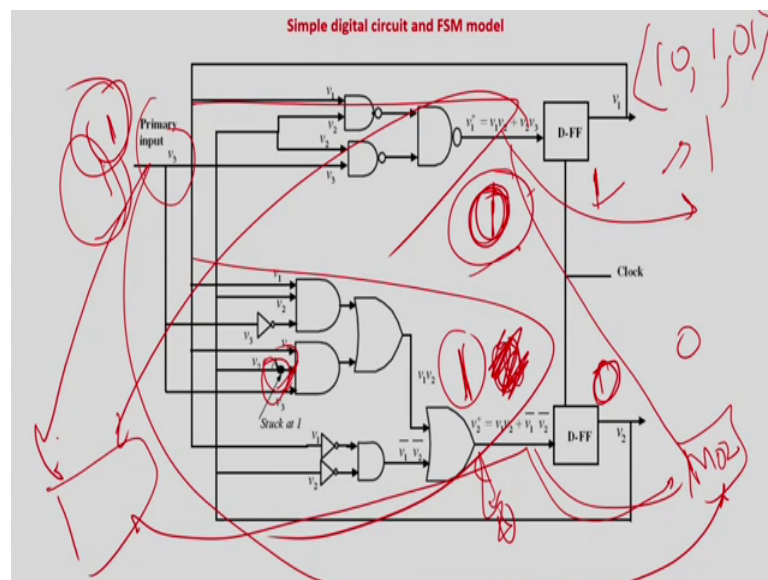
So, the job is make a normal version of the final state machine make a faulty version of the final state machine collect all such FD-transition. FD-transition means fault detecting transition collect all first transitions which are making a difference. Using those transitions you will make a module 2 that is what is our module 2, this one you are going to make module 2 which is going to detect the errors online. So, how to I do that this is the final state machine design of the test vector for this.

The algorithm is simple take a normal model final state machine model all fault final state machine model, collect such FD-transitions. In this case there is only one FD-transition and using those FD-transition you have to design module 2 this line actually shows the design of that tester. So, what do you do keep this initial state this is online tester module 2 design I am teaching you. So, basically this is a sorry this is the final state machine level implementation of the tester. So, what it is doing basically. So, what it looks is d 0, d 0 is the this is the machine model of the circuit under test and module 2 design basically is this FN detector, right.

So, now this is a tester. So, what the tester is going to do? The at the initial state of the tester here always going to track whether I am in this state called 0 1 sorry 1 0 state encoding 1 0 because only from this state basically such a vector is there when a fault detection can be done otherwise you need not do anything as the detector can slip for this for low power.

So, what they do this design if you look at it this test and basically if you look at it this tester will basically probe the values of all the state variables and the inputs of course, the tester has to probe the values from the circuit under test. So, that it can detect it you obviously, and the tester and the circuitry both are on chip you can do that.

(Refer Slide Time: 40:06)



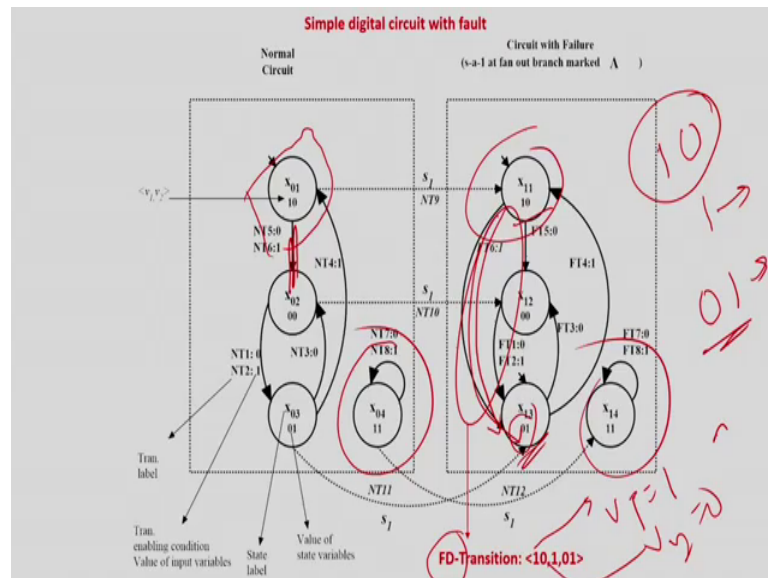
So, the if I say that this is the tester that is module 2. So, it will it is going to probe this value, is going to probe this input as well as it is going to probe this input. So, with this it is it can do testing.

So, what did is going to study is going to study if the input bit is 0 1 sorry 1 0 from where the fault can occur. So, so it is saying the initial state is keeping the track of the next is g state. So, it is always keeping a track whether in the next state it is going to be the value called 0 1 sorry 1 0. 1 0, 1 0 is this state from when the error can be detected. So, always one 0 here means the next value it will be 1 and 0.

So, these lines I am trapping. These two lines I am trapping. So, the module to a detector always try to find out if there is a 1 0 in v 1 and v 2 respected is such the next step is going to be 1 0 from with the from detecting transition is there. So, always trying to see this case that if it is 1 0 the last the third bit is basically for the input line with which is in material in this case. So, always I am going to check whether it is a 1 0. It is a 1 0, very good I am going to the next state of the tester is the next state of the tester not the machine of the circuit.

So, I am going to get d 1; d 1 is a very interesting because in d 1 what did the initial state is always keeping a track if the next state is one it is a 1 0 then only the tester will wake up because this fault detecting transition is only occurring from the state 1 0 otherwise else it is going to sleep 1 0 has come ha very good; that means, the machine in the next step is going to be either in this state or this state that is same state called 1 0. Then it goes to intermediate state. And, in the intermediate state what is going to try to study? In the intermediate case is a you have to study that if the input is 1 and the next state is basically 1 1, that is what is the error. If you look at this is the error let me clean it for you next state I have to this is the error.

(Refer Slide Time: 41:55)

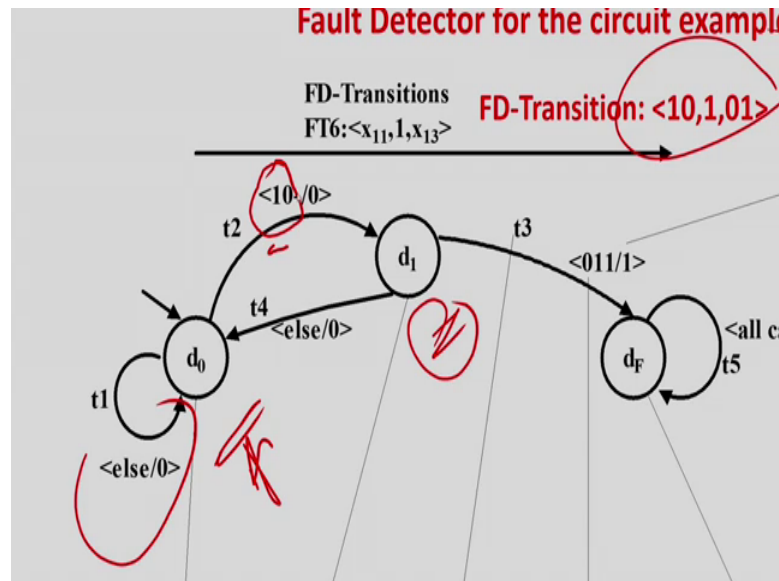


That initial state 1 0 input is 1 the next state is 0. So, first is a track that the circuit is now going to be in this state 1 0. Next what is to find out the tester that the input is 1 and the

next state is 0 1 instead of 0 0 normal case it should be 0 0, but for the fault if just come to a 0 this is what is has to gone to tracked.

So, you see first it is tracking 1 0. Next you have to track that it has gone let me zoom it for you.

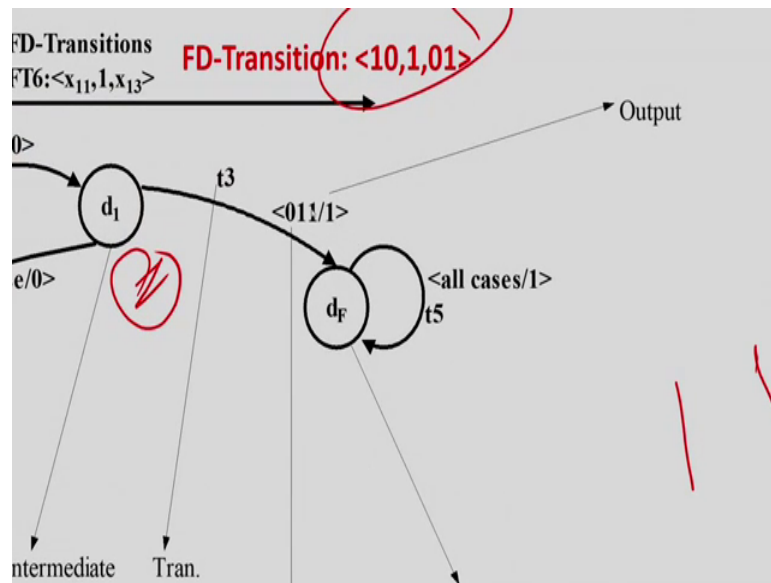
(Refer Slide Time: 42:22)



The three bits of the test are first I have to tell you the three bits of the tester is something like this. It is the first two bits of the tester are the test state registers v_1 v_2 probing and the third bit is the input that is v_3 . So, it is in this case 1 1, 1 1 this is the director called 1 0 input is 1 and the next state is 0 1. So, this is the case the error is being detected by the fault detecting transition. So, what I am doing initially is probing it is probing all the three lines v_1 , v_2 and v_3 .

So, v_1 v_2 is 1 0; that means, the circuit is next I am going to be in the state called error detecting state that is 1 0. So, in this case you will go to a intermediate state of module 2, that is the tester, is very happy that it can happen that from this next state the fault can be detected, otherwise it cannot do anything it will be looping ok, fine.

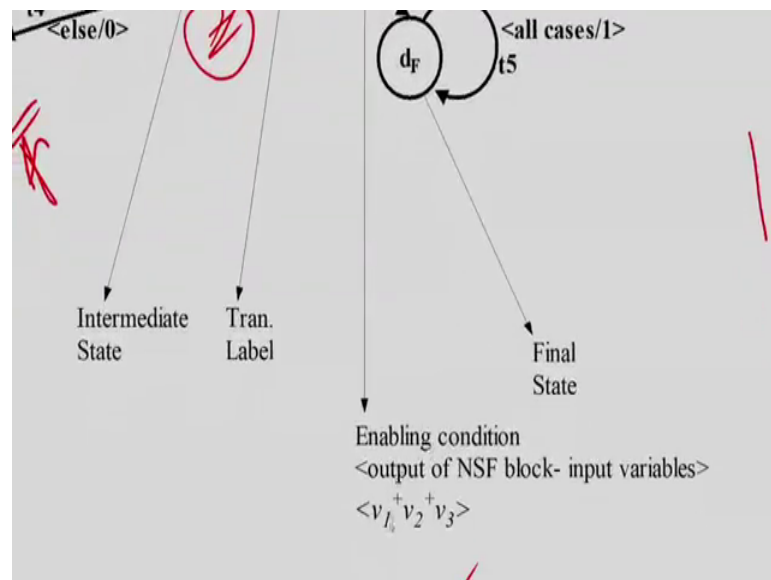
(Refer Slide Time: 43:07)



The next state is going to be 1 0 it is coming to an immediate.

Now, what is going to track is going to track that the input that is v 3 basically is going to be as shown in the error.

(Refer Slide Time: 43:15)



It is seeing v 1, v 2 and v 3. So, it is going to see that the input is one very good this is the input and the next state is going to be 0 1 then it is an actually at fault because the normal case it should be 0 0 1 input 1 the output state registers are 0 0. In such case it

will again go back to the initial state of the module 2 of the tester it will again go to sleep.

But, if the fault is there then some state 1 0 the next state is going to be a one 0 1 and the input will be 1, that is this part is matching. This one actually corresponds to this and this input v 3 and the output 0 1 is corresponding to 0 1 and 1 and the fault and the fault is detected any you will land to the final state where the output is a 1. This is the output of the basic enough for tester which says that it is a fault right see all other cases the output is basically a 0. These also a 0, these also a 0, these also a 0, but here from here it is going to the last state of the test I get stuck over here because the fault has been detected.

So, basically if you look at it so, in the initial state that is d naught you can see keeps a track of the next G-state. So, that is it is going to find out whether a state has come in the circuit from where the error detection is possible in this case is 1 0 where the intermediate state go to an intermediate state when the CUT is the in a is the in a state from where there is an FD-transition. So, it is tracking that 1 0 is possible or not in the next case; that means, now the circuit has come cross state from where error detection is possible. So, the tester is going to go to the intermediate state and final state basically if an error is detected that is 0 1 is the next state and 1 was the input.

So, in this case the fault is detected and then we get stuck in the tester saying that the error has been detected. The output is actually telling you that error has occurred ok. And, if it happens that basically this if the circuit is normal we are going to get value of 0 0 1 this is the input and this is going to be the next step. So, it will again go back to d naught that it is corresponding to 1 0 and the fault is not flagged that is what is the design of an FN detector.

And, this actually corresponds to only this single transition we have to repeat this for all the transition. So, how many in this case there is only one transition which is detecting the error which is are calling FD-transition. In the general case next example I will show you there can be n number of such transitions, for all of them you have to build a tester like this and this tester is a final state machine you will be synthesized a circuit and it will correspond to your module number module number 2. So, that is the tester module.

Now, you can ask me a question basically that a final state because we require a final state machine module to do this and final state machine model for the circuit we are

making a final state machine model, but actually if these two test expressions how can you handle it.

(Refer Slide Time: 45:50)

Identification of FD-transitions directly from the circuit description using OBDD

- Partition the circuit into cones of influences.
- Generate OBDD corresponding to the cone circuitry under normal condition.
- Generate OBDD corresponding to the cone circuitry under faulty condition.
- XOR the two OBDDs.
- The variable values corresponding to the paths to leaf node "1", are input combinations for the FD transitions.

So, this group of people working on this partial replication based online testing has given a formulation that the same thing can be done symbolically and I am not going to go in very details of this because already this has been taught by Professor Deka in Symbolic Model Checking. In symbolic model checking what we do would not go for explicitly final state machine model of the circuit rather we represented by a what do I call as the binary decision diagram.

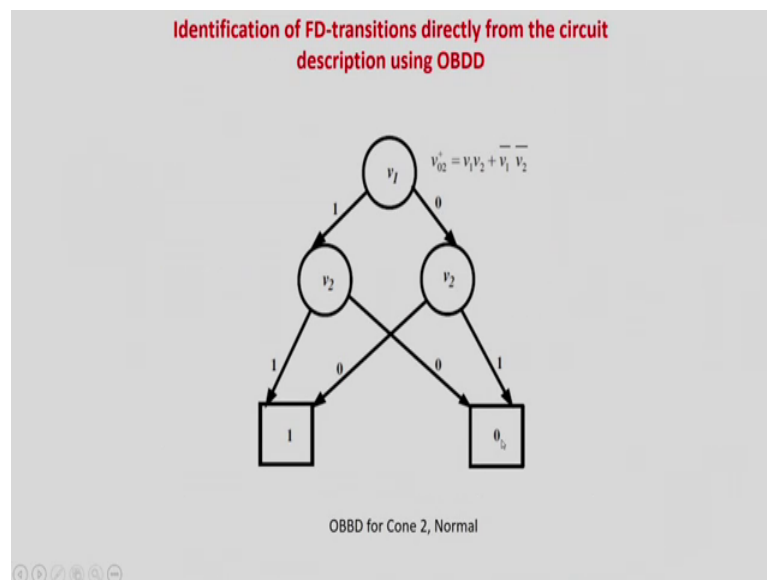
So, what they will do in this case I need to just find out this fault detecting transitions. So, what they will do what is the step partition the circuits into cones of influence. Generate OBDD correspond to the cone circuitry under normal condition that is what this is the circuit. So, you take a cone like you take a cone like this. You generate you represent this part as a BDD, normal mode and also all the cones will represent by BDDs that is a symbolic representation of the final state machine. Then you make a faulty representation of all the BDDs and then you XOR it. And, then basically if we XOR the normal and the faulty BDD all paths leading to the one of this of this BDD is going to be the test vectors we differentiate the fault, example I will take.

So, a partition the circuit into cones generate OBDD correspond to the cone under normal condition. Generate OBDD of the cone under faulty condition, XOR the two

OBDDs. The value of variables corresponding to the paths to leaf node one are the input combinations for FD-transition; that means, those input combinations basically differentiate the normal and the faulty condition, very similar to symbolic model checking with Professor Deka has taught in details.

So, I will just going to give you an example over here in this circuit if you look at basically if I remove this. So, this circuit if you look at so, this fault has no impact on this cone. So, if that I can not the I may not use it for this example basically I am interested in only cone number 2. So, in cone number 2 basically. So, I will represent v_2 in a BDD in the normal mode.

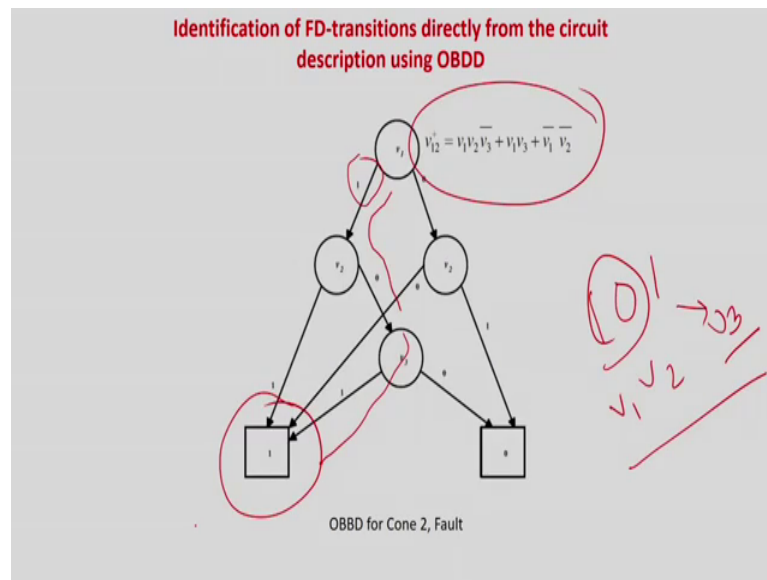
(Refer Slide Time: 47:36)



So, this is what is the BDD under normal mode. I am calling it $v_1 v_2$ basically just to represent a normal mode if for a failure I will represent by $v_1' v_2$. So, 1 corresponds to the 0 correspond to the normal number 1 for the first fault second fault third fault I can just represent a simple nomenclature.

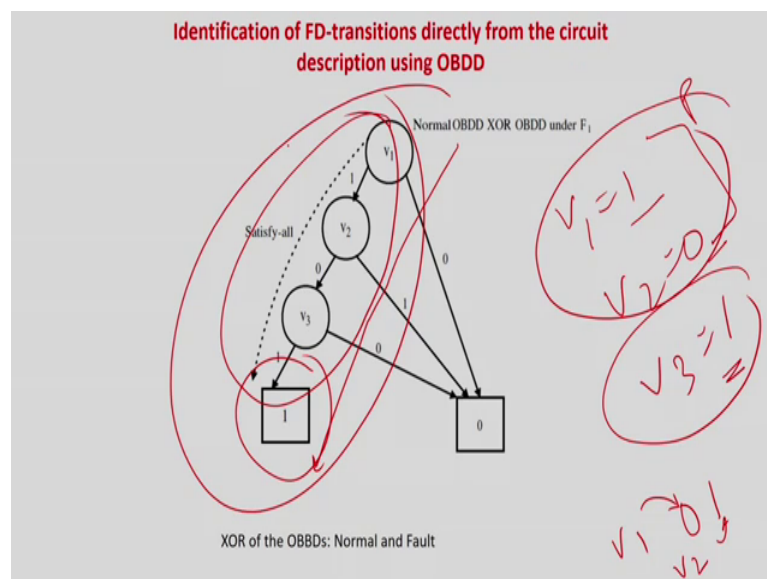
So, the if you look at is the BDD representation of this cone simple Boolean algebra, right. Next what I will do next I take this stuck at fault over here I take this stuck at fault over the Boolean representation gets changed and I generate another BDD.

(Refer Slide Time: 48:07)



Just is the homework you can take this circuit final the Boolean binary representation and representing BDD. Already in details we have taught you in the verification part of the course. So, this is the cone representation for the fault and this is for the normal.

(Refer Slide Time: 48:22)



Now, you do an XOR. So, this will be the XOR presentation now it is very import. So, XOR means this is the BDD representation of the normal fault 1 normal BDD fault BDD and XORing. So, any path it leads to 1 means this is the vector which leads to different values in the normal model versus fault the model BDD version of course, BDD version

is the in the representation of the circuit version. So, very simple idea normal BDD, faulty BDD XOR any part leading to 1 means basically the XOR value is 1. So, when the XOR value has 2 million representations are one if there is a difference in the output.

So, therefore, look at v equal v_1 equal to 1, v_2 equal to 0 and v_3 equal to 1 is basically the vector which represents a difference in the normal and the failure model, so, that you can find out with this. Now, how we are using representing FD-transition like this. So, already we know what was are FD-transition. Our FD-transition is nothing, but 1 0 1 and 0 1 these are FD-transition. So, if you look at it if you look at your vector what we have found out we have found out v_3 equal to 1 and v_1 and v_2 is 1 0; so, 1 0 1. So, it is say. So, this is v_3 and this is basically the values from here. So, in this case v_1 is equal to 1 and v_2 equal to 0 and v_3 equal to 1. So, that creates a difference. Sorry, let me go to the FD-transition. So, oh here also you can find out also FD-transition if you look at we can get the value.

So, this one is v_1 equal to 1, v_2 equal to 0 and v_3 equal to 1. So, that is the FD-transition; that is from this present state if the input one is apply the error can be detected. The same thing I can also find out by this one v_1 equal to 1, v_2 equal to 0 and v_3 equal to 1. This corresponds to the present state and this corresponds to the input. So, instead of using a final state machine representation basically here I am using a this I have found out by basically a efficient representation that 1 0 1 is creating the difference. But, the same thing I am also finding out by basically a symbolic representation this tells you that this is the present state and this the input from with the basically error can be detected, but what is the error effect how do I find out what is the error effect.

So, error effect this tells you that this vector v_1 1 0 1 state and this input is going to lead to an error detection that is basically nothing, but this point that this is the state and this is the input where the error difference is there. But, what is the final output that the how the error is reflected that is 0 1, how do I find out that cannot be told by this basically in this BDD. This BDD is just telling you that these are the input condition in which the error can be detected or the difference is there that is this one.

Now, how do I find out the error representation error representation is 0 1 that is v_1 and v_2 is basically this one how do I find it? It is very simple basically you take this vector and apply it here because this is representing the cone the vector is 1 0 1 is the present

state this is v_1 , v_2 and this is v_3 . So, this is nothing, but the BDD representation of the cone under fault. So, if this is the input what is going to be the output reflection very simple you put here as a 1 v_2 is equal to a 1 and v_3 is not required over here does the answer is a 1.

So, what it is saying that 1 0 1 basically can detect the error that is found, but what is the error output? For that we have to just take this bit and apply in this circuit or in the BDD you can apply in the circuit also will tell you the value. So, we applying the value 1 0 1 under fault you can easily find out from the vector it is a I mean BDD itself. So, this is going to give the value of a 1. So, this is going to be the value of a 1. So, this is 1.

So, if the value is 0. So, slightly I made a mistake. So, the value is a 0. So, what is the value v_1 equal to 1 v_2 equal to 0. So, I have to occupies this path and v_3 basically is 1. So, v_3 is basically 1. So, this is the path I have to basically take. So, this going to tell me, so, v_1 equal to 1, v_2 equal to 0 and v_3 is equal to 1. So, basically tells that the output is a 1. So, the next state output is going to be a 1.

Similarly, if I want to find out what is the going under error what is going to be the value of. So, again I am just telling you that basically this one actually tells you this part actually tells you that this is the input condition in which case the error can be detected that is v_1 equal to 1, v_2 equal to 0 and v_3 is equal to 1. But, what is the impact or how the output is represented is not told by this BDD that is if I apply 1 0 1 that is the present state and the input the error can be a it is can be detected or differentiate it, but what is the output I have to find out. For that you have to test this vector take this vectors and apply in the circuit very simple output is 1 0 1. So, I will take this input here and apply see the saying that is the one and it is a 1 is the input and the a number is basically 1 0. So, this is 1 and this is 0, this is 1 and this is 0.

So, this is the BDD has told me that this input and this state condition can detect the error, but what is the error output you have to just apply in this circuit and you can find out. If I apply it you will be very easily able to find out that it is nothing, but the answer will be a 0 and a 1. This is the basically the error output this. Again I repeating the BDD has told you that 1 0 and 1, this is the error detecting code or FD-transition input condition. But what is the effect then the BDD will not tell you the just apply this value to the circuit and you can find out for the error output.

So, in this case it is a 0 over here, if we apply this you can easily test it and the answer is a 1 over here. Had it been normal we are going to get the value of a 0 over here. So, very simply you do not have to go to the final state machine at all. So, what I am going to do I am going to take basically the normal condition because this is the cone only I am interested over here, but in the general case you have to repeat it for all the cones because the fault is over here. So, I am just interesting in this cone. So, I am going to generate the normal cone, BDD representation of the normal cone, faulty cone and XOR it. So, if you are XORing it you are going to find out the input condition in which case the fault can be detected, in this case is 1 0 and 1.

Now, I will take this vector and I will basically apply it in the circuit with a simple simulation. So, what is the value 1 0 where the error can be detected and the input is a one I just simulate this circuit and it will tell you simulate circular of course, is a fault it will tell you that the answer is a 0 and a 1 and if the error was not there the answer would have been a 0. In fact, I am not interested in this part. So, this will actually give you the FD-transition.

So, 1 0 1 this is the FD-transition initial part error detecting part that will be told you by the BDD, but what is the output in the in case of an error will be that is actually 0 1 will be found out by simulating a circuit. Circuit simulation is very simple complexity you need not at all bring the FSM into picture, but the very easily you can find out the equivalence you can see 0 1 1 sorry 1 0 that is 1 0 apply a 1 the fault case is 0 1. So, this the same thing I can find out without applying the BDD sorry without applying the final state machine.

So, this partial replication what it does again? Just I am repeating. So, you take a circuit you take the normal model final state machine model find out the vector or the all transitions which are relate to a difference in case of normal and faulty representation using this general the FN-detector that is which tries to detect or which tracks the FD-transition that is initial state it will try to find out whether the circuit is in this state, next it is going to find out whether the input is this and the next it is going to be this. So, the error can be detected that is what is the job of the FD-transition and FN-detector which uses an FD-transition done synthesis your tester is ready.

But, the problem is how to generate this test vectors or this FD-transitions very difficult because we are doing is efficient. People have avoided it by using a BDD based approach in this case what you do you divide the circuit into cones take the normal BDD, faulty BDD, XOR the BDD and it will tell you what are the input states and input combination which can lead to the error detection, but the in this case is 1 0 and 1. But, it will not tell you that what is basically the outputs of the circuit corresponding to that input FD transition condition for that simply you have to apply this state values like in this case 1 0 and the primary input it will get you get the error value and 0 and a 1.

(Refer Slide Time: 56:55)

Partial Replication based OLT

Design of FN-detector does not change the original structure of the circuit. Thus it is a non-intrusive design.

The FN-detector is designed from FD-transitions. If some FD-transitions are dropped

- Then area and power of the detector are reduced at the cost of higher detection latency.
- Then fault coverage drops however, it leads to reduction in area and power overhead.

And, use this you can avoid states complex states complexity and still you can design basically you are FN-detector circuit which will be online texture to the system. In this example I have shown you a case of a fault if this is only one FD-transition and you designing, but it will not always be the case or a simple example I will tell you, but what about the advantages.

The design of the FN-detector does not change the original structure of the circuit. So, it is an non intrusive design that is the key challenge then it does not change the circuit at all you just take this and basically the FN-detector will be something like if you synthesize this FN-detector it will just probe this line, will just probe this line and it will just probe this line and it will concurrently run with the circuit. So, need not change anything if you look at the structure it just probes the three lines that is v 1, v 2 and v 3

just observes these two straight lines and the input lines and tries to track where the circuit is going.

So, there is no change in the circuit itself. So, is the very nice design, but you can see one thing of the appreciate that this FSM is also not very small. For each such every transition will require a small FSM structure like this. So, of course, I will do lot of FSM optimization can be done, but still the FSM for the FN-detector or the tester is not very small it size will be much larger compare to the error detecting code or finance or signature monitoring etcetera. So, this is a partial replication base method. So, always you have the appreciate the fact that partial replication based methods are always higher in size compare the other counterparts.

So, basically if we look at so, what I have done I have taken a circuit either represented by FSM model or by BDD I found out the FN-detector and I represent a fault detecting detector which is going to be synthesize circuit through the online testing, but this circuit is not very small compared to the original final state machine. So, the FN-detector or tester would be very comparable with the cut. How do I avoid it, but the key role I have to appreciate here is a is a very online very in non intrusive design non intrusive design. It just draw from lines from the circuit under test and takes the decision FN circuit is going on the FN-detector is basically or tapping some lines from the CUT that is the state lines and the input lines and trying to find out whether this circuit behavior is proper or not. So, that is the key role of partial replication.

So, any circuit with the partial replication based architecture, are non-intrusive, but the circuit size is much higher. So, what I can do? So, because anyway there is no matching, so, either you have we have to take a very large circuit to go for non intrusivity and good fault coverage the tester circuits will become larger. But, if we want to drop bring the tester circuit down and go for the coding theory based approach then lot of assumptions come like either this circuit has to be modified so that there is no bidirectional errors and so, many constraints coming making the circuit is an intrusive. So, that we will do outline.

So, this partial replication base circuit is very good circuit does not required to be change which is one of the most for most important requirement. I design a circuit for testing if you want to change it and may not be very happy with it. So, therefore, partial replication

base testing and gain a lot of emphasis in the real world. So, in this case, but how do I trade off because you have a very big you have a circuit, but a very big FN-detector not good.

So, you drop some FD-transition. So, in this now in this example I had only one fault and one FD-transition, but in a real case they for a given fault lot of FD-transitions will be there. If you want to and the circuits tester size will become very high what you can do you drop it some you blindly drop. So, that the in area of the circuit comes, but which one you will drop another research area that which are the important FD-transitions. FD-transitions are very much correlated with faults; this fault; so many FD-transition then you like to find out which are the most important faults to keep only them. Other non important faults you not at all bringing into the picture also for the given faults they say 10 FD-transitions try to find out with FD-transitions occur frequently because I told you.

So, for example, in this case there is a state in which case is the unreachable state, right if the unreachable state. So, we should not take any kind of FD-transition from that state because under normal conditions the circuit will not reach that state. So, even if there are some FD-transition starting from the unreachable states you may drop. So, look at the quality of the FD-transitions good quality FD-transitions are the ones which occur very frequently. So, more occurring more because in online testing you cannot control the inputs. So, whatever inputs are coming based on that you have to take decisions.

So, FD-transition occur and frequently you keep in the tester; that means, then you will be a very high probability of getting such transitions coming in the circuit under test and you can detect the faults occurring rarely FD-transition you can throughout. Like the example I have given you unreachable state throughout all the FD-transition starting from those states because anyway the CUT will never come there. So, lot of trade offs are possible by controlling the number of FD-transitions. In case on the contrary, in case of error detecting codes you do not have much control that is you assume that this you take a coding theory it will detect odd number of unidirectional error I take another coding theory it will do this.

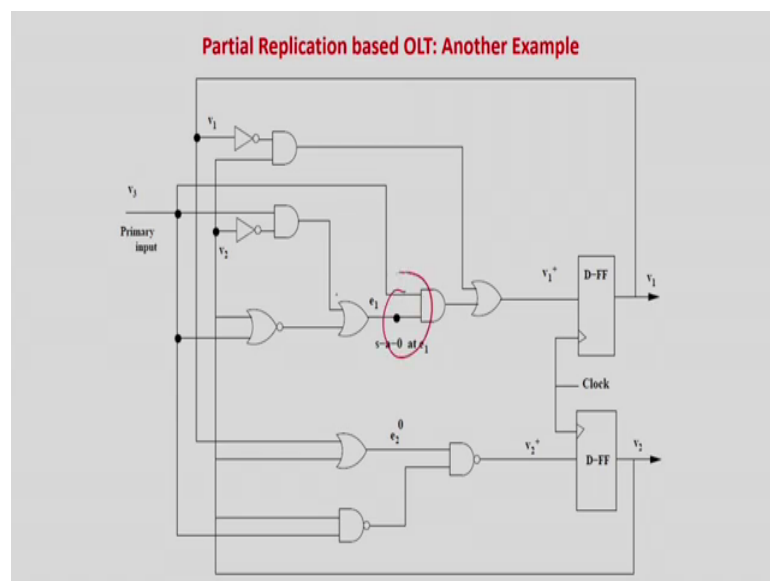
So, based only error detecting code you have to basically take the flexibility, but in case of partial replication the flexibility is very dependent on the test perspective because in the error detecting code base technique the tradeoffs like uni-directional, multi

directional, odd detection, even detections, so, some of the techniques I have discussed. All of them will depend basically on the coding technique used that is going to give you the tradeoff, but in case of partial replication the tradeoff comes directly from the testing prospective. 100 FD-transitions are there, which one I want to keep because more FD-transition the tester design will be larger.

So, you drop all the FD-transitions which are from this state the circuit never reaches some FD-transitions remain keep it those FD-transitions which occur more frequently. So, you can understand the tradeoffs are both from online, both in the partial replication as well as in the coding theory or whatever technique is possible. But, in case of partial replication the control is more just by controlling in by just by taking the number of FD-transitions or dropping some of them is the control phenomena of the area of the FD-transition. But, these controlling the FD-transitions or by chain by taking some of them and dropping some of them is directly related to the faults and test. So, it gives you a more better control over testability compared to your coding theory.

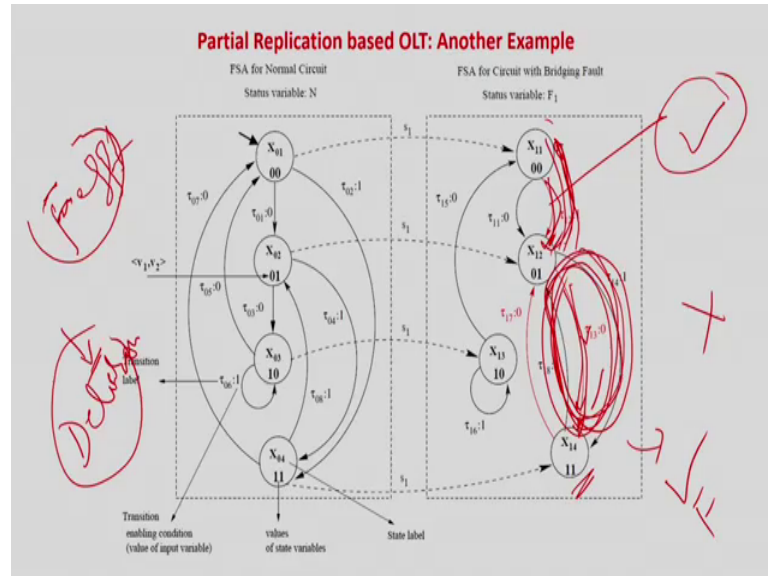
So, therefore, every partial replication based online testing is very good or much better compared to coding theory. So, the area a what I done the power of the detector are reduced at the cost of higher detection latency. Why this what you mean by latency? Here I have to come to the fact that what is the latency.

(Refer Slide Time: 63:12)



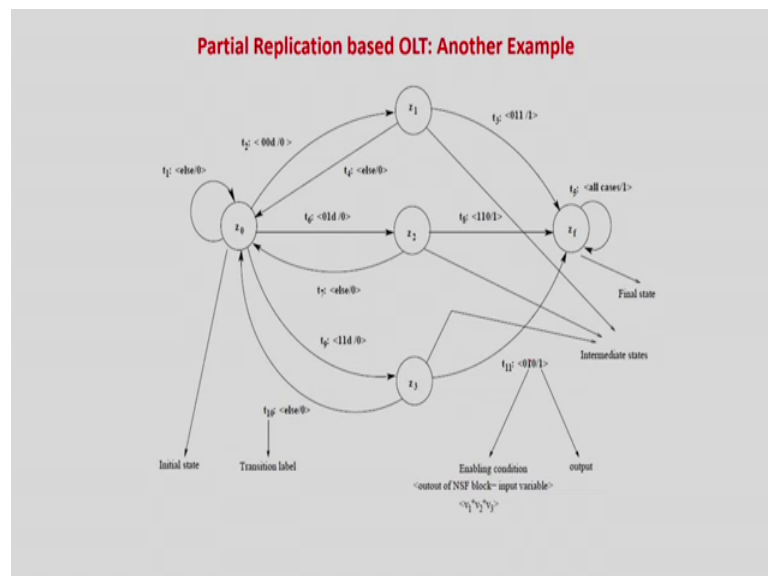
Say and let me just go to an example it is better another example I have taken with the stuck at default one.

(Refer Slide Time: 63:16)



Look at this one is very important same thing, but in this case basically there are three FD-transitions; one from 0 0 input is 1, this way one is another and this one is 3. You can I am not going to the elaboration you can easily look at this circuit and you can find out that these three are the FD-transitions or these cases which differentiate the normal and the faulty behavior.

(Refer Slide Time: 63:36)



So, if you make the FD-transition so, there will be three parts this is for the first transition, this for the second and this is for the third. So, for the same behavior same pattern say everywhere you have to track that whether I am this is going to be my state if it is so I have to wake up and if you just find out whether in the next state the input is 1 and you are going to this state then the error is detected same idea of FN-detector design, but more number of FD-transitions more number of states in the FN-detector. So, higher with the circuit of the tester.

Now, how do I compromise? You drop one FD-transition let us assume that I am going to drop maybe this two FD-transitions. So, basically maybe this will be there and these two portions will be drop. So, your FD FN-detector state machine will be small because you are drop some transitions. Now, what will be the impact? The impact will be because I have no control on the operation of this final state machine. It can go over here it can go over and no control because the inputs are coming from the environment and the circuit it traversing these FSMs. So, it can happen that what it can happen is that it may come here it may go here it is going totally traversing over this I have no control.

But, sometimes among this may be so, after so many days once it takes and what the idea of trading of this area of the tester you have dropped this FD-transition. You have not consider is a FD-transition in the detection director design. So, what can it can happen. So, basically it will revolve over here, revolve over then this one is happening error has occurred instead of basically this is occurred because this is what is showing that is the difference on the normal behavior error has occurred. But, you will not be able to detect it because the detector does not take account of this transition.

So, now, the error detection latency, then again you have to wait for a long amount of time maybe after this will be going over here, here, here and here maybe after some time it may go over here because in this just assume that if there is a I mean there is a basically transition from here to here just assume then you can go from here then assume that this I have kept and this I have not kept. So, still it is going around here going around it comes over here fault is flagged off that is error has occurred.

But, again after that what happens error has occurred, but now it is revolving in this manner revolving in this manner I am not able to detect any fault because there is no behavior difference. But, the fault has already fault error has started it is place because it

is has taken place I am not going to detect it. After than assuming that these are transition over here just assume then after sometime this transition has occurred and your FN detectors has detected because it is captured in the FN-detector then you can flag of an error.

So, there is now lot of latency because whenever this transition has taken place the error effective studied, but again it was revolving in this manner. So, error was not captured even in between this one has might have a error again occurred because nothing can be done because you have not taken into this taken into fact of this transition in the FN-detector, but after some time after lot of time say it has gone to this state and from where this error transition again occurs and basically now you can detect the error because that is here in the FN-detector. So, some time has passed. So, that is actually fault latency.

What is fault latency that is fault effect the effect to detection this is called fault effect to fault detection. So, this difference in time is called the latency the simple very simple. So, I am not defining it formal in the course, but latency means error has occurred and the effect is observed and when you detect it. So, you can a very easily appreciate the fact that if the if I take all these three FD-transitions like in this case detection latency is 0, because it has come over here gone over here whenever the fault effect happens you can easily capture it. These all the FN FD-transitions you are keeping in the FN detector.

But, if I remove some of them then there are very good chances of basically your elimination of error detection latency because for example, as I told if you have not taken this and you have taken only this and I have not taken this only this then you have to wait till this occurs and your fault is getting detected in between the FN-detector FD-transitions will occur circuit fault will be show in the circuit output, but will not be able to detect it because I am not kept into picture.

Now, if you ask me a question which; obviously, if you should to take all the three, but the size of the FN-detector becoming higher CUT size will become compared to the tester size. So, reduce it I have to drop some FD-transition, even if you ask me with FD-transition I should reduce. So, if you look at the fact here in fact, if you look if once I reach over here after that I cannot reach back this state. So, this transition occurrence probability is very low, but if I am in this loop, so, once I will be here I will come over here and after that if you see I will be must do just I will be looping in X 12 and X 14.

So, I will be either looping by this manner and sometimes this and sometimes this transitions will occur.

So, a probability of occurring of these two transitions basically this and this probability is much higher by looking at this test structure I can tell you. So, if you give me the choice that you have to take only two FD-transitions and of course, take only this two. If you give me the choice of taking only one I will basically take any one of them two because once the fault happens then the probability of moving in these states is much higher than moving in the state 1 1. So, if I am going in state 1 1 basically then only this FD-transition when occur. So, probability of occurring τ_{12} is much lower than the probability of descends I will take this two. If you ask me to take only one then I can I can take either this or this, but if you allow me to take all three, then only I am going to take. So, these are choice.

So, you have to look at the structure of the FSM and then find out which FD-transitions are more viable to occur you try to keep it best to keep all, but we will be make the area very high. So, you have to tradeoff that which one I have to consider. So, now, this is actually with respect to the probability of occurrence of FD-transition and this is corresponding to this fault. Now, for all faults you have to repeat this procedure. So, if you actually which faults to keep and which faults to not keep that is very simple which are faults very important and somebody told you there most frequently occurring in your online operation of the circuit just take these faults all others you draw.

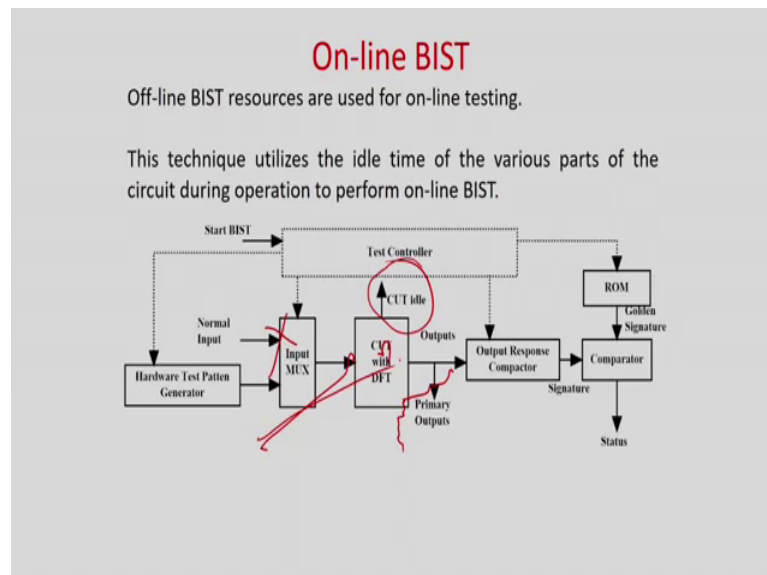
So, therefore, what it tells basically this partial replication base circuits idea is very simple. You take circuits, take the important faults, find out the FD-transition design, the FN-detector drop all FD-transitions whose probability is lower. So, I can I have a very good control basically of designing a FD FN-detector that is the tester which much lower area overhead there is partial replication than the circuit itself, but I have a bit control that irrelevant faults I will drop highly occurring FD-transitions I will keep and so forth.

So, therefore, partial replication based online testing methodologies now very widely applied for online testing or fault tolerance of embedded systems. Just before I just close down one thing I should tell you that this I am not going to elaborate that without drawing this circuit or final state machine how I can find the probability of occurrence or good quality of FD-transition. Similarly, there are symbolic models to compute this

because what Professor DeKa has told you in the verification part is very important. Whenever there is an FSM based operations which are happening in circuit testing verification there is always a symbolic base counterpart to handle the state base expression.

So, therefore, you should not go bias by the fact that partial replication based online testing always requires a FSM model. No, I have shown this in FSM model for illustration to you. So, you understand the concept, but in reality always will be use the symbolic methods.

(Refer Slide Time: 71:02)



Last, but not the least somebody had tried to used BIST for online testing. What is the idea this is simple BIST architecture I have explained. So, many times in the last few lectures I am not going to go over again, but a very important line here is called the CUT is idle that what is BIST. BIST is always applied when the circuit is starting off. So, you know that there is a normal input and there is a test pattern generator and there is a MUX.

So, when you want to apply the BIST, so, this input will be CUT and the target tester input will be going from the LFSR or some pattern generator and you are going to evaporator and details when we are discussing BIST, but now it is in a online testing mode. So, in online testing mode you cannot stop this you cannot stop, no. You cannot

make it stop because the inputs are already coming and the circuit is operating. So, if you stop the inputs is no longer remains an online testing.

But, some scientists have taught that it is not always that all circuit modules are operating. There is sometimes called idle modules of circuit, but with parallelism and pipelining coming into picture the idle cycle sub circuits are very less, but in generally you know that all parts of the circuit may not operate at all possible in time. Sometimes some circuit may be operating and some may be at sleep mode. So, there is a something some important line they have brought it out as the output called circuit is idle.

If the circuit is idle for some amount of time because the other parts of the circuits are working in that case you can stop the normal input and apply the BIST. So, this is call actually online BIST online BIST and offline BIST is very similar in online offline BIST the circuit is not start it is operation. So, you just give the input from the test pattern generator and check whether the circuit is normal or standard BIST. Online BIST is same like offline BIST, but just you have to find out whether this circuit is idle for some amount of time whenever it is idle you just top the normal input and do the BIST that is basically online BIST.

(Refer Slide Time: 72:46)

On-line BIST

Off-line BIST resources are used for on-line testing.

This technique utilizes the idle time of the various parts of the circuit during operation to perform on-line BIST.

Three issues that are related to the performance of on-line BIST.

- Availability of idle time.
- Minimizing test length, so as to fit within the idle time available.
- Test scheduling.

- Efficiency of this technique depends on the amount of idle time available.
- This is not an efficient technique for OLT because present day circuits have parallelism and pipelining techniques for high utilization of its modules

But, this is not very popular this was initially it was very popular because you can basically use the normal BIST to do online both and both offline. So, it was very good very well appreciated. But, nowadays basically this popularity is going down because

basically totally depends on the idle time. In the circuit time idle time is very less you cannot even test the circuit properly only when the circuit is idle you can apply the BIST because the online test you cannot stop the inputs.

Input can be stop only when if the circuit is idle by it is own nature that is given an application, application is running, but for certain time the circuit is not doing some work I am I am not shopping the normal inputs to come, but because of the application which is running it is idle for some amount of quickly you do the best during that time. So, it is totally depending on the idle time and therefore, if the idle time is very less the BIST will not be very successful. And, why is popularity is going down very much nowadays because now nowadays parallelism and pipelining and in picture they always try to bring down the idle time of the circuits for high efficiency.

So, therefore, online BIST is not a very popular technique nowadays for all applications mainly. Generally, if you have designed a very peculiar circuit where will be lot of idle times then they can apply this.

(Refer Slide Time: 73:52)

Summary of OLT techniques

- Signature monitoring in FSMs.
 - Intrusive OLT methodology.
 - State-explosion problem in FSM.
- Self checking design using error detecting codes.
 - Hardware redundancy.
 - Intrusiveness.
- Duplication schemes for On-line Testing.
 - Hardware overhead is more than 100%
 - Requires prior knowing regarding the input patterns that the circuit likely to receive.
- On-line BIST
 - Depends on the idle time of the modules.
 - High utilization of their modules (using parallelism and pipelining)

So, as of now basically this is summary of the online testing schemes. So, we have seen first this signature monitoring in FSMs is a intrusive methodology and also the state explosion problem. Then we have seen the basically error detecting codes. The good advantage of error detecting codes is the area is very low, but against the intrusive design you have to change the circuit entirely.

So, therefore, people started looking at duplication based online testing scheme, but the problem is if you go for pure duplication the area overhead is more than hundred percent. So, if you go for a partial replication then you should have a prior knowledge of the input patterns etcetera. Online BIST is a very good solution, but problem is that idle times are required, but in modern days due to parallelism and pipelining idle times are not there.

By comparing it you can find out the duplication scheme is one of the best and most widely accepted scheme because again the area of overhead is very high. Instead going for a full duplication or people are not going for full duplication they basically reduce the area overhead by dropping lot of error detecting transitions, but what they will drop how they will drop basically totally in the control of the test engineer. So, he makes a good look and survey of the circuit and that gives the more power to control the test patterns for the testing and it tradeoffs basically that is lowering the FD-transitions in the tester design is more tester test oriented.

So, therefore, among all the methods we have discussed as of now duplication is scheme is most widely accepted in a general sense, because it is not hundred percent duplication it is partial duplication and what is partial that is which faults he will take which test patterns will take which totally in the control of the test engineer. So, therefore, he makes a good survey of the circuit topology good study on the input applications and he makes a very nice duplication circuit for you which is very efficient in testing yourself.

So, with this basically we come to the end of this lecture on online testing of embedded system circuits and we have in this two lecture series I have tried to give when you have a good spectrum of different techniques and what are the advantages and disadvantages. So, with this we come to the end of this module and in the next module we will be trying to look at of the basically because as of now we are mainly talking about hardware which is basically ASIC that is basically application specific ICs. But, nowadays lot of embedded systems will be are developing of FPGs. So, the next module will be trying to look more at from more from the perspective of software hardware combined testing and mainly this is applicable for programmable devices.

Thank you.