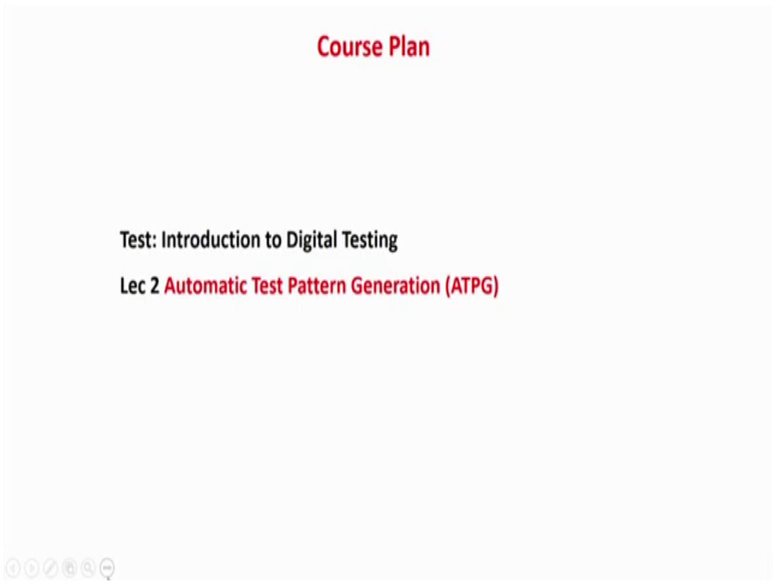
Embedded Systems – Design Verification and Test
Dr. Santosh Biswas
Prof. Jatindra Kumar Deka
Dr. Arnab Sarkar
Department of Computer Science and Engineering
Indian Institute of Technology, Guwahati

Lecture – 27
Part-3: Embedded System Testing
Automatic Test Pattern Generation (ATPG)

Welcome to the next lecture on the testing part of Embedded System Design Verification and Test course. So, as we were seen in the last lecture, that we are talking on the third part which is on embedded system testing.

(Refer Slide Time: 00:43)



Course Plan

Test: Introduction to Digital Testing

Lec 2 Automatic Test Pattern Generation (ATPG)

So, in the last lecture we have seen the basic introduction to digital testing and; what is the importance of digital testing in the context of embedded system, design verification and test flow. In the today's lecture basically we will slightly more into details of the prerequisite module that is on the introduction to testing. We will see today the concept of automatic test pattern generation.

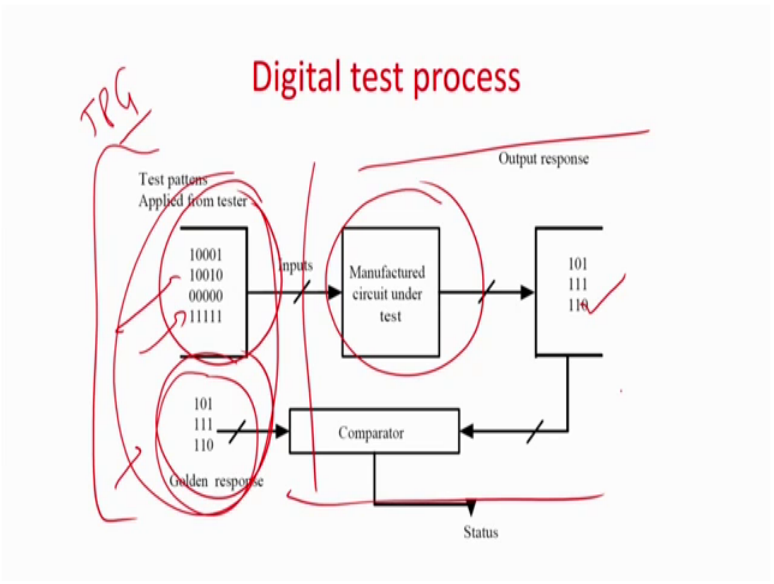
So, basically if you remember last lecture what you had discussed is that as the all embedded systems basically use a digital platform for their computation which is a major component of the embedded system design. So, there can be lot of faults which come due to the improper, or do some lacunas which has which are there in the fabrication process.

To find out such faults basically we have to go for digital we have to go for testing. And we have seen the testing basically comprises of two parts; one is you have to plan basically what you are going to do the test that is we have to decide what patterns you have to apply? What you will do? What you want to measure? What is the golden response etcetera and the second part is the physical testing.

That is you have to put the chip you have to connect the I means equipments and measure. So, that is actually a more or less are clerical kind of a job or in the other sense is more of a physical kind of a job. So, mainly we will be looking in this course on the test plan that is given a circuit design, or given a hardware component of the embedded systems how do you plan to testing that is what are the test patterns you will give, what you will measure etcetera.

So, in this lecture mainly you will be concentrating on given of a circuit or a hardware component. Then how do you automatically generate the test patterns, that is how do you generate the test patterns means how do you plan for the test patterns which will be applied to perform physical testing.

(Refer Slide Time: 02:18)



So, basically this is your you can say that a digital test process in which case in fact, I should say that this is the physical testing process. So, this is the manufacture circuit and the test there is the physical circuit we just be manufactured you will be put into a machine which is called an automatic test equipment and then you apply inputs you come you have the outputs.

So, this is something called golden response. Golden response is basically for such given inputs these are this should be the outputs. If there is no problem in the circuits or there is no defects in the circuits then for this inputs this should be the outputs. Now, you put the manufacture circuit in the tester you apply this inputs physically you measure the output and then you make a comparison.

So, if this output matches with the golden response then there is no problem in the circuit and it can be shift to the customer. Now this is the physical testing process, but today we are going to look at this and we are going to mainly look at this that is basically how do you decide what test pattern to apply? And how do you find out what are the golden responses. So, that you have a test plan in picture before the devices is fabricated. So, what is the device goes for fabrications you have some amount of time.

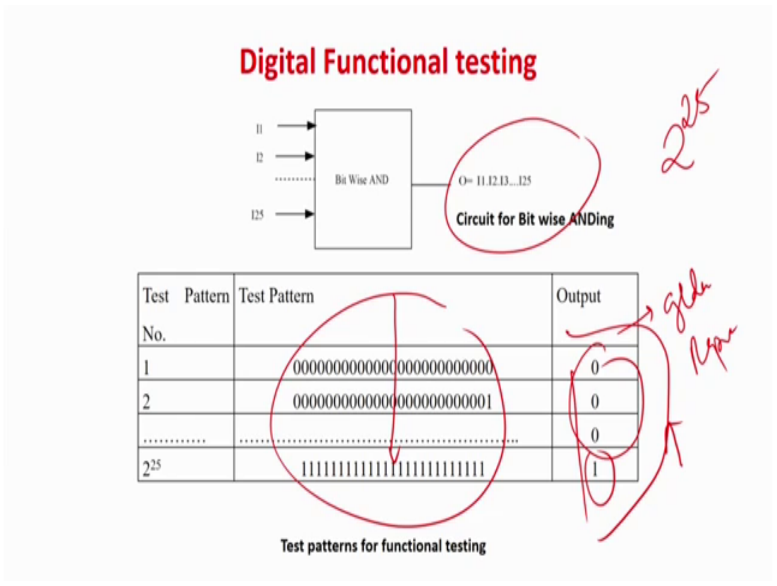
So, in that amount of time basically or even concurrently with the design process you have to find out such sets. So, this is actually called test pattern generation and if you are doing it automatically we called as an automatic test pattern generation. So, mainly this part of this course will be concentrating on how you can automatically generate test patterns or how can you do automatic test planning that is the goal of the course. And this part actually as I told you is basically something physical that you apply the test patterns measuring and comparing. So, there is no much we can study over here. So, as I said this is basically something called a automatic test equipment.

(Refer Slide Time: 03:52)

Automatic Test Equipment

So, we basically put these devices over here and there is this is lot of internal computers and test pattern generation test pattern analyses inside. So, we have to just program that we are the test patterns to be applied this is the golden response expected. you will do the automatic testing for you and it will just give a beep or non beep saying there is circuit is normal, or circuit is having a fall. So, after that you give the test plans to these automatic test equipment everything is very much automated.

(Refer Slide Time: 04:23)



Digital Functional testing

I1
I2
I25
Bit Wise AND
O= I1.I2.I3....I25
2^{25}
Gldn
Rspn

Circuit for Bit wise ANDing

Test Pattern No.	Test Pattern	Output
1	0000000000000000000000000	0
2	0000000000000000000000001	0
…………	………………………………………………	0
2^{25}	1111111111111111111111111	1

Test patterns for functional testing

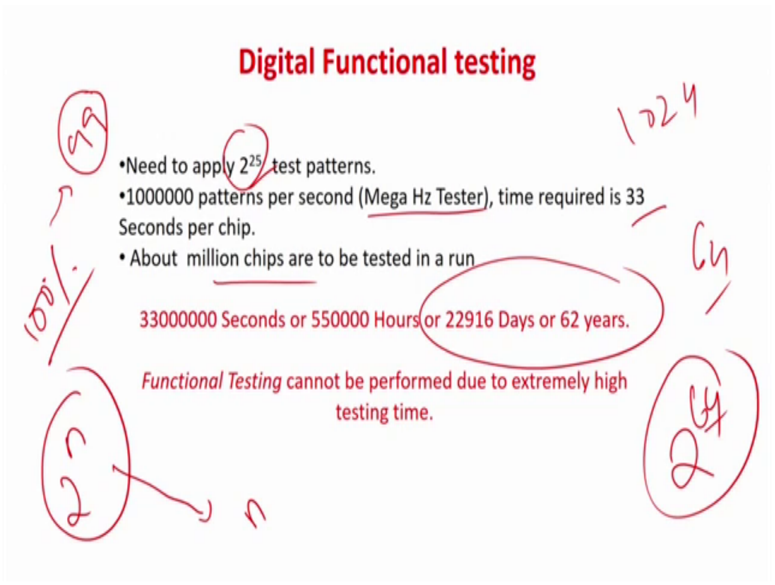
So, now we come to the concept of testing, so basically are the test plan. So, let us assume that it is a 25 input AND gate circuits. So, there is a 25 it will be the input; let me zoom it. So, basically this is a simple circuit in which case we have 25 input lines, and 1 output lines and the output is bit wise and for all the inputs. Now since I want to test it. So, basically how many inputs will be possible as there are 25 input lines, so it will be 2 to the power 25, so many different input patterns can be possible and I have enumerated them.

So, you can see all 0's to all 1's and for all cases the output will be 0 and accepting in the case of all ones the answer will be a 1, so this is actually your golden response. Now what will you do? Now you will give all these test patterns to your test program and these are the expected golden response. Now when this device will be fabricated you will put in the tester, the tester will apply all this patterns, it will measure the output and actually to compare with this.

If all the bits are compared 5 our circuit is normal as there is a fall. Now the question arise this is actually called functional testing. Because there are device you want to apply all the patterns to verify that it is properly operating or not. In fact, as discussed earlier testing can be very while because start with functional testing then we can go for speed testing, then we can go for power testing, then we can go for delay testing, then we can go for transistor level testing, it will be very wide. But for the practical sense basic as I told you we test for power, we test for delay, and we test for functionality. So, two type will be mainly restricting to basically the functional testing part of it, automatic test pattern generation for the functional test.

So, in this case all this patterns has to be applied and you will be able to do the test. So, it is a very simple procedure if there is a circuit of 100 inputs you generate all possible input combinations that will be 2 to the power n, get the basically the golden response and you have to just report that these are my all my test patterns which is nothing, but all possible combinations. The golden response we have to obtain by simulation of some other method and then you have to give it to the test. This is the basically the idea of functional testing since very easy. But now is a very simple circuit with 25 inputs, all possible case is 2 to the power 25.

(Refer Slide Time: 06:27)



Digital Functional testing

- Need to apply 2^{25} test patterns.
- 1000000 patterns per second (Mega Hz Tester), time required is 33 Seconds per chip.
- About million chips are to be tested in a run

33000000 Seconds or 550000 Hours or 22916 Days or 62 years.

Functional Testing cannot be performed due to extremely high testing time.

1024
22
100%
64
2
n
2^{64}

Now, if you have a mega hertz tester because giga hertz testers also exist. Now what is mega hertz testers people are not using this is slightly absolute data, but it is a I should tell you that there are very very few companies in India, I think very few may be I know one of them which is called Tessolve which has such fascinate is in India. Mainly such fascinates are generally in the Taiwan or such places where there are basically Fab industries. But I should tell you that we have to purchase the time from the tester, because there are very expensive equipments and we have to pay the duty for using it may users that I want to use the test for 1 hour; you have to pays huge amount of money to do this.

Because the testers are so expensive that to make profit the test testing vendors actually rent their machines and the renting is in hours. So, more time you spend on the tester more money we have to pay. So, our goal is mainly to do as much chip testing as possible within the given limit of time. Because if you are taking more amount of tester time and you are able to test few number few number of chips at that time the cost of testing per unit chip will be higher. So, our goal mainly is to use give less time of the tester, and to test more number of chips in that time. So, that the test cost per unit comes down.

So, if I have to apply all 2 to the power 25 patterns and assuming that the mega hertz tester then time required is 33 seconds per chip, this is I have to apply half second half a minute for each chip. And generally then allowed 1 million chips which are fabricated in a run and you have to test all of them. So, basically it will require some out around these many days or years to checking.

So, these are extremely difficult procedure and infeasible. So, this is a simple that 25 bits input. So, practical circuit will have 1024 or it can even as 64 bit inputs. So, if you have to apply all 2 to the power 64 combinations forget about 1 million samples in years and millions of years also you will not be able to test a single chip. So, functional testing is very simple apply all 2 to the power 25 patterns or 2 to the power n patterns; where n is the number of inputs all possible combinations you apply match with the golden response and you are super confidence that your chip will function accurately as per the functionality.

But again 2 to the power n is a exponential complexity and even for a simple AND gate you can see their exorbitant or in feasible numbers. So, if the circuit has 2 to the power 64 inputs I can we can do the mathematics you will find that not even a single chip can be tested in a 100s of years. So, functional testing is very practical, but it is not is very theoretically fine and you can guarantee that everything is proper only by full functional testing. But it is I am very sorry to tell that you cannot apply it practically then what to do because, our job is to give guarantee to the as such because 2 to the power n, this is full functional testing.

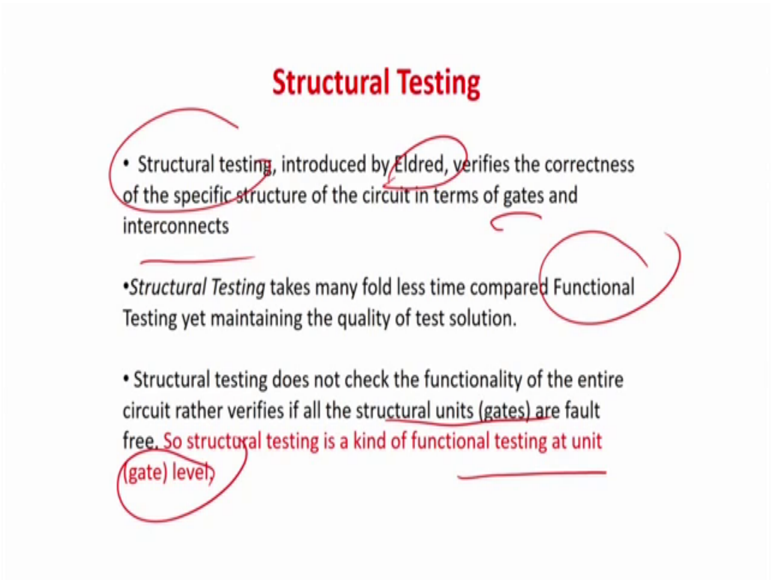
This can give 100 percent guarantee that your chip will be operating 5 if you are testing is passed, but we do not have so many time amount of time. So, what you want to do? We want to bring it down bring the number of test patterns down, but still we are trying to give very good guarantee; may be this may be a 100 percent guarantee, may be instead of 2 to the power n. We will use only n that is linear complexity or even lower number of test patterns and may need the assurance they will come to 99 percent, or even slightly less than 90 because slightly less than 100 percent.

But I cannot assure give assurance of ninety percent and sell the chips that that are possible. So, what do you mean by assurance here? As I discussed yesterday say you manufacture a 100 chips, you have to test their you have to find you find out their 30 chips are abnormal, we throw them out and 70 chips we sell into the market saying that all are proper. So, here is the assurance when I say this 70 percent chips which I am selling to the market I have no defects, there I should be absolutely proper. So, there if I go for 90 percent error rate, then you are gone.

That means basically out of the 70 chips which I am ascertaining to be having no faults out of them again only 90 percent will be correct nature, again 10 percent that is out of 70 may be around 7 chips will have defects which I am selling to be saying them to be normal this is not actually allowed. So, assurance will be more than 99 percent plus that is when I declare say around 70 chips are normal from the 100 chips manufactured. Among them in a very large sample may be 1 or 2 may be wrongly bid.

So, what do you mean by wrongly bid means I then need to be proper the tested tells is the chip is normal you sell it to the marker, but still it has a defect. So, that should be very very low. So, that are that are means by assuring the quality then as I it is a very simple story I should not drag it longer, that functional testing is not practically possible.

(Refer Slide Time: 11:01)



Structural Testing

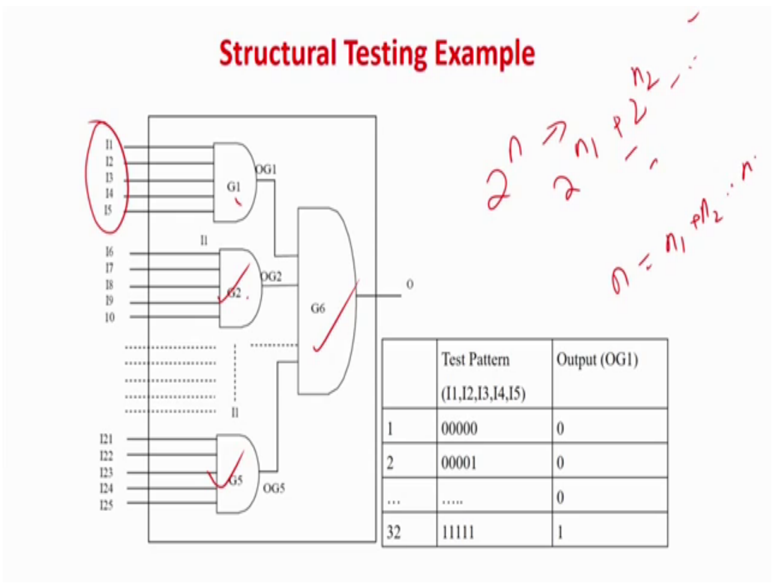
- Structural testing, introduced by Eldred, verifies the correctness of the specific structure of the circuit in terms of gates and interconnects
- *Structural Testing* takes many fold less time compared Functional Testing yet maintaining the quality of test solution.
- Structural testing does not check the functionality of the entire circuit rather verifies if all the structural units (gates) are fault free. So structural testing is a kind of functional testing at unit (gate) level

There is something came into picture which is very interesting which is called structural testing and he was the gentle man Eldred. So, who invent who actually I gave the idea of structural testing. So, what I said is that basically functional testing is very very complicated used on timing consumption. So, he told that it basically test this circuit in terms of gates and interconnects, do not test the functionality, test the functionality only at the level of gates and interconnects.

And here shown, that if you do this of course, you are not going to give me as high guarantee as a functional test. But he has shown it by statistics applying into a millions and millions of test runs that if I do such a philosophy that test this circuit at the gate level functional level. And, basically you apply we will not think about the full functional level testing he has shown the number of test patterns will come down drastically, but still this test quality or the test guarantee is more than 99.9 percent plus.

So, the structural testing does not test check the functionality of the entire circuit rather it rather the structural part that is the gates are fault free or not. That means structural testing is a kind of functional testing at the gate level. That means, he will not test the circuit entirety itself that means each of the individual gates you will test in a functional level and you will be forgetting that what the whole circuit does example is better.

(Refer Slide Time: 12:21)



Structural Testing Example

I1
I2
I3
I4
I5
G1
OG1
I6
I7
I8
I9
I10
G2
OG2
G6
O
I21
I22
I23
I24
I25
G5
OG5
$2^n > 2^{n_1} + 2^{n_2} + \ldots$
$n = n_1 + n_2 + \ldots n$

	Test Pattern (I1,I2,I3,I4,I5)	Output (OG1)
1	00000	0
2	00001	0
...	.....	0
32	11111	1

This is the same circuit I have shown you 25 inputs and 1 output and gate, maybe it is the made up of 5 and individual AND gates of 5 inputs each. So, 1, 2, 3, 4, 5 gates and that the final AND gate and this is the output, now say that I want to test each of the and gates only at that level and norm the integration. So, basically with G 1 how many you will require? So, there are 5 inputs, so you require this table all combinations you require to do this.

So, what is the time? What is the number of patterns two to the power 5 that is equal to 32. Now I will test these I will test this means then I will test this means and my job is done. So, I will do it at the individual level I will test this gate, then this gate, this gate and so forth, so it will be 32 into 6 what is nothing but 12 192. So, from 2 to the power 25 which is very very large I have come to 2 to the power 5 into 6 that is 192.

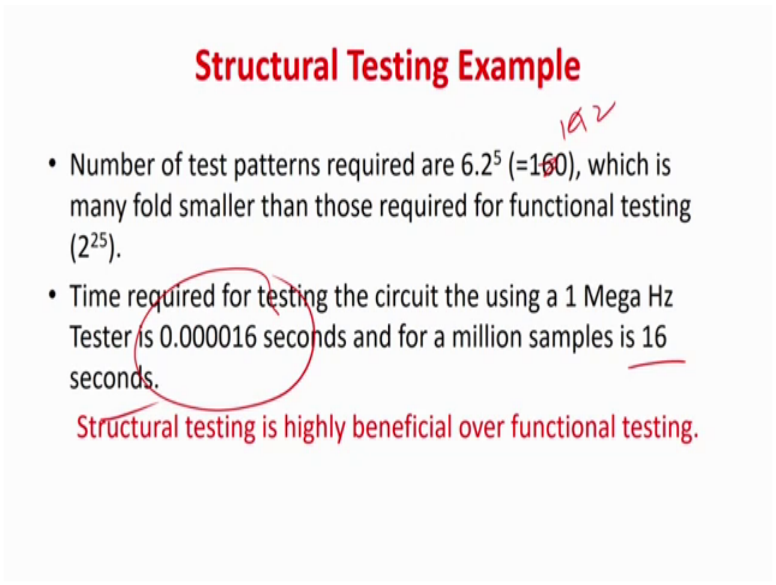
So, huge junk basically so that it is something like for divide and concur or I should tell you that it is from 2 to the power n we are from there we are going to 2 to the power n 1 plus 2 to the power n 2 dot dot dot where n is equal to n 1 plus n 2 dot dot n. So that means, we are breaking down this n into smaller smaller numbers. So, we are if you look at your a simple algorithm and complexity theory this brings downs the complexity drastically.

So, here on the example I am testing all the gates individually and I have to apply only 196 (Refer Time: 13:49) of 100 range patterns. Whereas, if I want to test the entire functionality is 2 to the power 25 it is extremely difficult, huge jump in the number of test patterns. If you have huge jump in the reduction of this test patterns then the test time will be very very less.

Because now I have not applied 2 to the power 25 test patterns I will just apply 196 test patterns and my job will be done. So, this is actually a structural testing what is the pro cons basically you are not testing the entire circuit functionality. But statistically people have observed that is whenever this gentle man actually proposed this idea.

So, yes not there any kind of mathematical analysis basically he has tested this idea about 100s and 1000 of test runs and found out that if you still go for structural testing you are you are assurance quality is more than 90 percent, 99 percent plus. So, huge reduction in number of test patterns, but still slight compromise in the test quality. So, people started working for structural testing and it became a defect to standard.

(Refer Slide Time: 14:47)



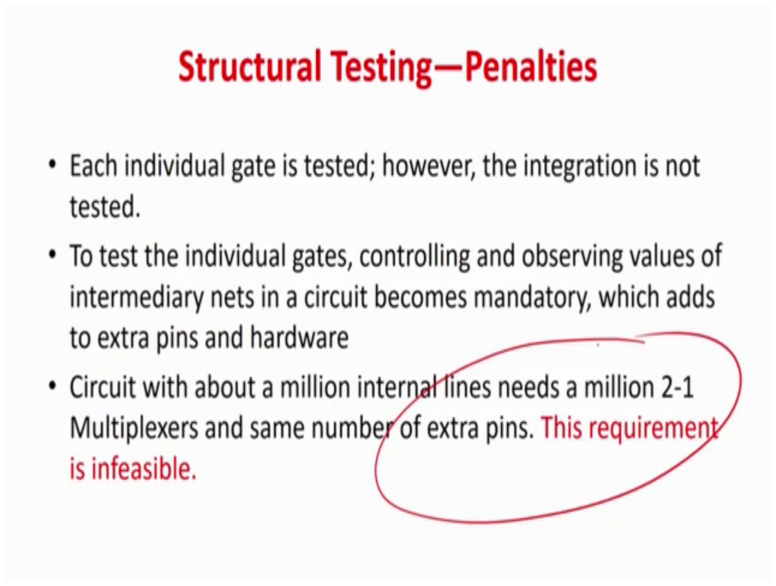
Structural Testing Example

- Number of test patterns required are 6.2^5 (=1~~6~~0)192, which is many fold smaller than those required for functional testing (2^{25}).
- Time required for testing the circuit the using a 1 Mega Hz Tester is 0.000016 seconds and for a million samples is 16 seconds.

Structural testing is highly beneficial over functional testing.

Now, certain issue start coming in. So, basically sorry this is not. So, the number of test pattern requires is 6 into 2 to the power 5. And basically 192 which is many fold smaller than node required for 2 to the power 5 and basically the time required for testing the circuit will be 1 mega hertz tester is so low. And for 1 million samples is just 16 seconds, so it is very very feasible. So, 1 million samples can be tested in just 16 seconds, so natural testing became highly beneficial over functional testing.

(Refer Slide Time: 15:19)



Structural Testing—Penalties

- Each individual gate is tested; however, the integration is not tested.
- To test the individual gates, controlling and observing values of intermediary nets in a circuit becomes mandatory, which adds to extra pins and hardware
- Circuit with about a million internal lines needs a million 2-1 Multiplexers and same number of extra pins. This requirement is infeasible.

But there are some penalties because engineering is never a matching you do some optimization, you have to pay cost at another end. So, what is the cost now let us look at this circuit. So, I think we might have all scene your dual in line package chips that the black chips with some lakes outside that is a normal IC's whenever you open a mother board of the computer you have seen there is some IC's is black box black colored chips are there.

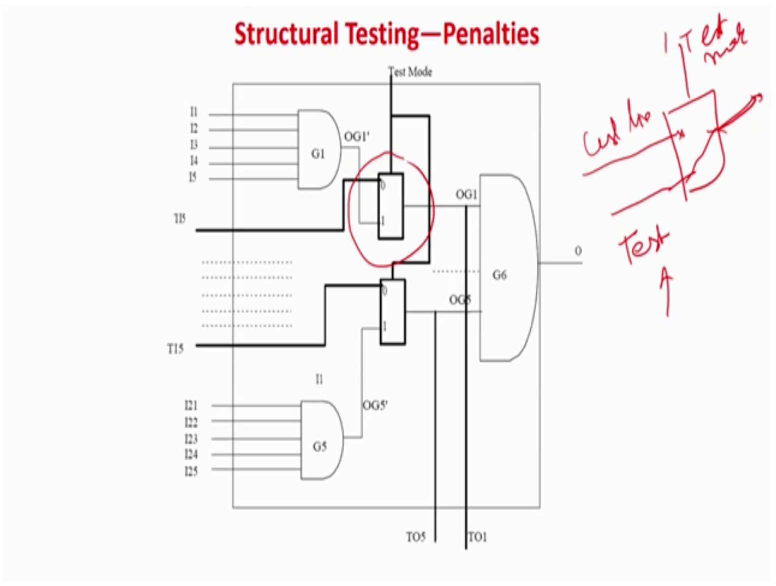
Basically that what I am trying to say that these part is basically sealed, this is sealed in a wrapper. So, this is the black structure we always seen the at dual in line package you know some chips and you see some pin out. So, basically this part is black box and there highly sealed. So, if you have to test this gate say so you will apply patterns over here there is no problem. But how do you measure this output? Similarly how do you measure this output? How do you measure this output? Now this is a problem. Secondly, when I have to test this gate number 6 this observation is 5, but how do I apply pattern so over here directly.

How do I do it? Because these are combined lines and this is the structure is closed. So, how do I have access to it? So, it is called structural testing as the problem of observability and controllability, difficult to observe this line sorry difficult to observe this line as well as difficult to control this line. How do I do this because they are packed. So, I have to some of make pin out extra; obviously, the reason is there is no other solution we have to bring out extra pins, or use other extra arrangement to control such internal lines then only you will be able to do structural testing because I am doing all the gate testing individually.

So, therefore, I should have full access to these gates, but as this is a closed box. So, I rotated access to some of the pins. So, that is actually called observeability and controllability problem. So, one main issue is that now let us look at the penalties of structural testing each individual gate is tested; however, the integration is not tested, but that is the minor issue. Because already by handling large number of cases we will be a found out that the compromise is not the much. But to test individual is controlling and observing values of intermediary needs becomes mediatory which have to extra pin or hardware that I will tell you.

This is the major problem that how do you access the internal lines? And to add intern test internal lines as I will show you will require extra piece and some multiplexing arrangements. So, for each individual internal line which has to be controlled and observe you have to put some extra cycles. So, circuit with 1 million internal lines require some million amount of multiplexer and large number of extra pins so, that is actually a killing point with an example I will show you.

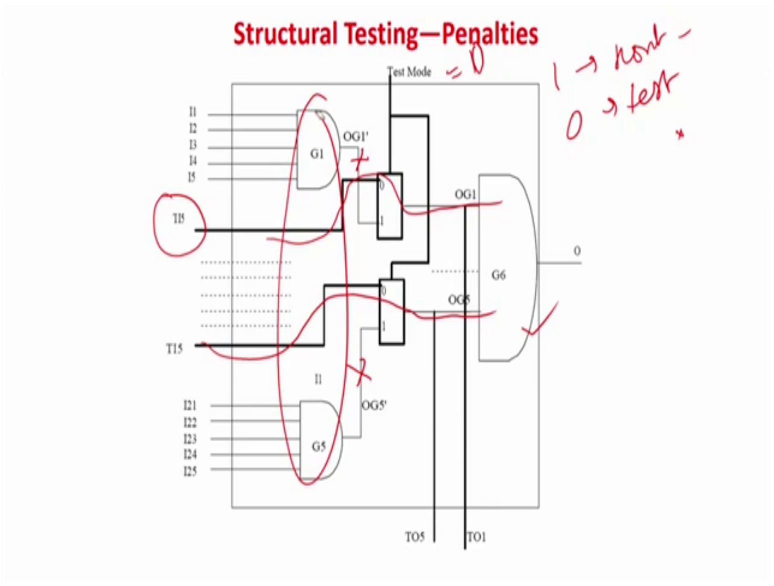
(Refer Slide Time: 17:50)


Structural Testing—Penalties

So, let me zoom this part. So, as we know this is the output of this gate and you have to observe this line and also you have to for the testing of gate G 6 you have to control this line that is we have to give input directly right. So, for example, there is then for to do this we actually put something called a 2 is to 1 multiplexer. So, assume that this is the normal output of a circuit and this is some extra out test pin is your test pin, and this is a normal cut circuit output circuit line.

We actually do is to we put a 2 is to 1 multiplexer and there is something called a test more and this is the marks output. So, when the circuit is operating normally I am not interested do test I will put 0 in the test volt. So, this line will be connected to is and your circuit will be operating properly. And whenever I want to do test what I will do is? I will remain this line as 1, and this part of the line will be connected to the output. So, circuit output will be blocked and something test bit I can give any input which will be going to the gate to be tested. So, such an arrangement I have to make and this arrangement is already displayed over here.

(Refer Slide Time: 19:01)



So, for example, I want to test the circuit I will make test mode equal to 1, I am assuming that I am testing gate number 6, so, I am in a test mode. So, this multiplexer this is this is line sorry I have just I have made a mistake. So, basically test mode 0 means it is in a test and during testing and 1 means I am doing a normal mode, 1 means normal and 0 means test. As I just if you flip the bits of the multi I mean multiplexer the same thing you get doing the other way around.

In this circuit 1 means normal, 0 means test. So, I want to do a test, so I will make the basically test mode multiplexer value equal to 0, the select line. So, of course, if you see so this line will not be in position to continue, so this line will be cut because it is 0. So, the output of AND gate 1 cannot go to the input of gate 6, but this is the test pin which is now exited because it is going over here and it is going through this one. So, you can easily apply any pattern to the gate number 6 directly by passing G 1. Similarly for the other being also basically if you look at it this will be cut, and this line will be going at the input.

So, by this multiplexing mode you can directly give all the test patterns to the give gate number 6 without having to be buffered by G 1 and G 6. So, you are bypassing this entire part and you are directly giving bit as a output, you are giving the test patterns directly to the G 6. Now say that I want to operate the circuit in a normal mode and I am doing testing at this point you make this gate equal to test mode equal to 1. So, this one will be connected over here, this one will be connected over here. So, automatically circuit in the is in your normal operating system mode right.

So, this is the way we are doing a control. So, this is way this is how all the internal lines has to be controlled. Because otherwise your normal circuit gates will start probing your vapor because here I want to apply 1 0 1 my test patterns, but if I have if I am not able to block this gate then I will not be able to directly apply the test patterns to the gate under test this is your gate under test. So, this is how actually I can make the lights controllable by putting 2 is to 1 multiplexers.

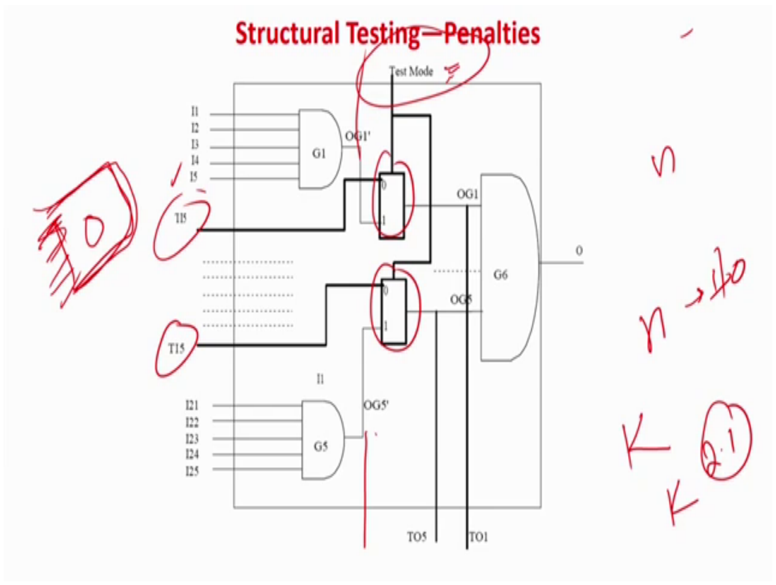
Now, again, but here in for the gate 6, I apply all the patterns and observation is no problem, but if you just seeing about the testing of this gate this is of this gate then basically the problem is other way around that is I will be able to control this lines very easily. But how do I observe this lines has the output.

So, this is are not a very difficult problem you just put out some extra pin outs which are the test pin outs. So, in this case whatever the test mode may be may be to be 0 and 1 I do not bother in this case, but some extra pin outs I will make for such gates. Because for observation controlling is more difficult, because for controlling we have to stop the input from the normal circuit and you have to, but test pins which can gives the values of the gate under test.

But for observation this is going to this output I have no problem we just put some extra pin outs from this gates. So, we get directly observe them as the output. So now what are the cost I am paying to go for structural testing? I have to have whatever intermediate lines you want to test because I am testing this gate I am testing this gate at individual level.

So, there is some intermediate lines which I have to test, so extra pin outs I have to bring. And if there is a gate which is internal which I have to test it, so I have to put some extra multiplexer so that I can put I can control this lines. So, there are lot of extra output lines which will be coming out of the intermediate lines. So, if there are n intermediate lines which has to be observed.

(Refer Slide Time: 22:32)



So, x n extra IO lines will be brought out for observation and if there are K internal lines which has to be controlled. So, therefore, there will be K; 2 is to 1 multiplexers this way plus K extra input lines which will be your test input lines and there is 1 test mode pin. So, now we can easily understand if there is 100s and 1000s of such internal lines which has to be controlled and observed how many extra pin outs? And how many extra multiplexers has to be put?

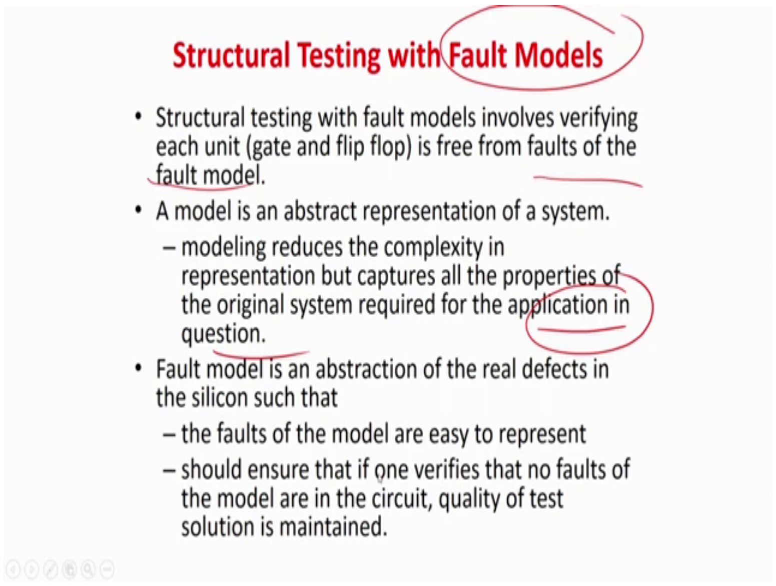
Extremely difficult because your circuit cannot have the huge number of IO pins, because that is blocked by the area of your chip covering material this is a in line package. So, your chip is packaged before selling into to the market your chip is packaged, in fact there will be the die. So, of course, this is not very long it is may be 5 centimeters by 5 centimeter, 4 centimeter by 4 centimeters, or something like that, but we cannot have n number of lines.

These lines are generally restricted to 48, 64 this is the ball grid array package it can be in the level of 500s and so forth. But you cannot have more than million such order of millions of IO lines, but if you see structural testing and if I have to do internal you have to probe the pins and you have to control the lines millions of extra IO pins will be there.

So, structural testing also heat a water link, but what happened is that it has given a very good inside that if you go for full functional test is the huge problem you cannot go do that. Structural testing opened up an avenue that you need not test this circuit in entirety.

Basically you test the individual gates itself still the test coverage is or test confidence is very very high, but the problem is that very much amount of extra pin out is required how do you handle this? So, in open them avenue and also it resist a bottle neck.

(Refer Slide Time: 24:16)



Structural Testing with Fault Models

- Structural testing with fault models involves verifying each unit (gate and flip flop) is free from faults of the fault model.
- A model is an abstract representation of a system.
  - modeling reduces the complexity in representation but captures all the properties of the original system required for the application in question.
- Fault model is an abstraction of the real defects in the silicon such that
  - the faults of the model are easy to represent
  - should ensure that if one verifies that no faults of the model are in the circuit, quality of test solution is maintained.

Then something extremely interesting came up, but again I have to emphasize one part here. In all this cases basically people have not them any kind of mathematical analysis to prove that: what is the confidence I will get with compromised amount of testing. Basically people have found out functional testing 2 to the power n structural testing n. But the confidence I get is very very near to in both of them the one may be 100 percent, or there may be 99.9 percent plus.

This they have done by statistical finding they are that 100s of millions of different test runs and they are found out then structural testing is almost equivalent to functional testing in terms of confidence. But the number of test patterns to be applied is very very less. So, that way they have found out that they are very correlated not pure mathematical way of determining they are done basically by testing and statistical results and 100s of hands all experiments.

But the structural testing opened up an avenue where may be started thinking then we can still go for testing without thinking of functionality directly at a structural avenue. But it heat a bottle neck or heated a heat the wall when basically we found out there are so many extra pin outs will be required. Then some people improvised on it and then landed into something called fault model, but this now even I will not test the functionality of the gates.

So, initially full functionality then I came to functionality of the gates then people certain think in why I should test functionality of the gates. Let me try by something right I will not even test the functionality of the gates. They said that can I find out certain cases which are happening more prompt that is such circuits are there having some kind of defects. Can I such some of pin out or find out which are the most provable defects and I assume that to be the my standard fault models that is maybe there are lot of devices which we have in our daily life.

Most of the things have some kind of failures, but if you start a in depth we will find out that for this class of a systems this is one of the most important failure it has. This is another system at the class of system and it suffer from this is the peculiar fault it is, but it is more proven to such type of faults. So, can we have kinds of what is called as fault models that is they are some of the standard defects which the chips are most likely to suffer from.

So, they are started thinking of some directions like fault models. So, structural testing with fault models involves verifying each unit is free from the faults of the fault model. That is basically now they will not test the functionality of the gates whether they know from statistics that these gates is most likely is suffer from this faults. So, our job will be just to find out that the gates are free of those faults are not even going for functional test.

We will give more examples to elaborate on the concept model is an abstract representation of a system, modeling reduces complexity in representation, but captures all properties of the original system required for the application in question this is very important. What is the model? That means, as I told you in the last class faults can be of fault mod fault can be of a huge nature functional structural deny, the silicon level manufacturing level so many different type of faults can be possible. But can we restrict about the models that is for a digital functionality of a circuit what are the most possible fault models or most likely fault models your circuit is going to suffer.

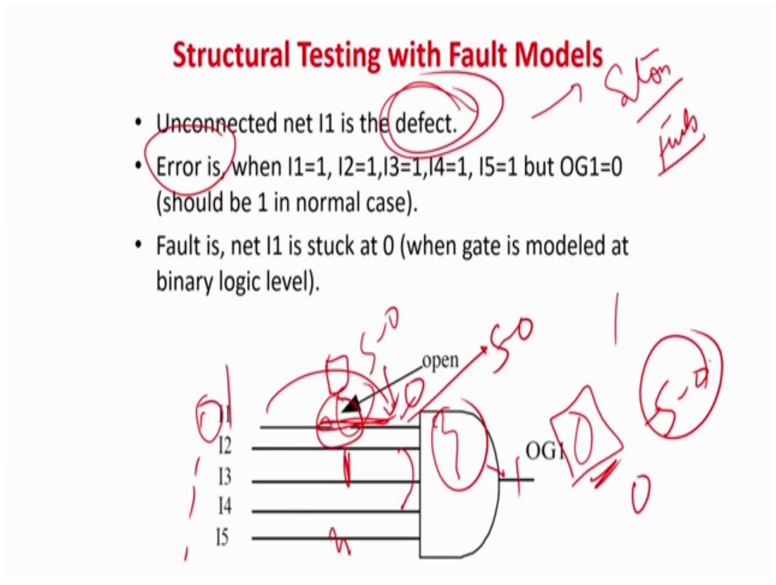
So, our application in these case is embedded systems and operating in a digital mode of operation. So, what are the most typical type of faults my circuit can suffer from? And I will only verify that my circuit does not have such faults. Then I can be more or less confident that the circuit does not have any problem, but again people have found out statistically that from functional to structural to structural with fault models you are having some is in testing, but actually there will be slightly compromise.

Like for example, 100 percent to 99.99 percent here it may be 99.98 percent slight reduction in fault quality will be there. But of course, that quality draw cannot be very huge like from 100 to 90 and then 90 to 80 such kind of drops with different faults testing strategies is not actually allowed. The compromise will be very very less that the ease of testing with time to taken for testing should be coming down drastically.

So, fault modeling was the key buzz word and actually it has solved most of the problems in testing. People have strategically found out that you need not test the functionality of the gate civil whether you trying to find out good quality fault model which mostly the gate suffered from and try to verify that such faults do not existing the gates at all. And if this is if we will establish that point by testing then there is 99.98, or 99.98 percent assurance that there is no I means you are not selling bad chip to a customer telling that it has it is having more defects that is your billing is proper.

So, fault model is an abstraction of the real defects in the silicon such that the fault models are easy to represent because we can I cannot give high complexity fault model. So, that you take years to generate the test patterns and also huge number of test patterns are generated there is also not there out. And it should ensure that one verifies that no faults of the fault models are there in circuit quality of test solution is maintained that if I some of verify that no such faults of the model are present then quality is more than 99.9 percent plus.

(Refer Slide Time: 29:35)



Structural Testing with Fault Models

- Unconnected net I1 is the defect.
- Error is, when I1=1, I2=1,I3=1,I4=1, I5=1 but OG1=0 (should be 1 in normal case).
- Fault is, net I1 is stuck at 0 (when gate is modeled at binary logic level).

open
I1
I2
I3
I4
I5
OG1

So, let us take a AND gate and basically say that some of you are not able to dope this certain problem is that these are open, basically this line is open. So, this is the unconnected net is the defect, so this is basically your defect is somewhere you are not able to manufacturing there is a problem. Error is now what is an error when we apply all this as 11111, but output gate is actually 0 whereas, it should be a 1, because this is open.

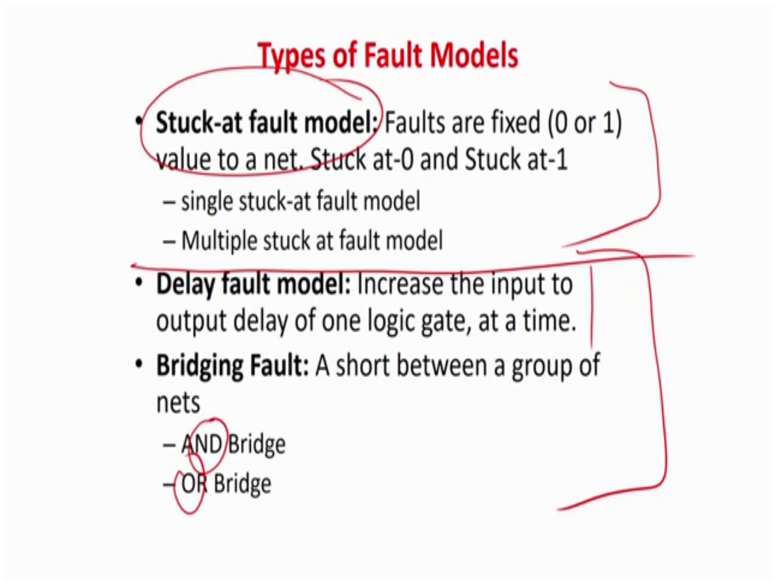
So, basically this is one type of a defect there can be n number of defects like this may be getting shorted all right. Then basically what may be in these two lines get shorted I mean then this line, it can also be an open, this line can also an open. So, there are n number of different type of defects which is possible.

So, this is your basically defect what is an error? Defect is basically your physical defect that happens you fabrication, error is the logical manifestation and what is the fault which is our model. So, basically our mode this is what is real defect at silicon level, this is your functional Boolean function what happens this is just talking about digital circuits. But fault or the fault model is that you will find out that this line is stuck at 0 that means whenever you are applying all ones or whatever this line basically is if you applying a 1 still 0 will be there and also the output line will be 0 stuck. So, that is we call that this line is stuck at 0, and this line is also stuck at 0, so this is actually a fault model. So, real defect is here not been able to properly dope this silicon dope with copper

It may also happen that the copper is very thin voltage is not being nothing it can also happen the happen that there is some problem with the if we ask n number of different type of defects are there. But the result and this you apply a 1 over here this one is actually not passing it over here, and all ones you are getting the value 0 over here which is the error that is your Boolean manifestation. And what is the fault model? The fault model is this line is stuck at 0, and this line is also stuck at 0, stuck at 0 means if when we have apply a 1 the value will be remaining at 0.

So, this defect can be enormous it can be also at some point it can also be due to some kind of manufacturing problem in the transistor which is insider your AND gate that can also lead to that this line the means if you are giving a 1 this is not actually passing to the output. But we will abstract everything out we will just assume that such a representation is stuck at 0, for this line as well as for this line.

(Refer Slide Time: 31:54)



Types of Fault Models

- **Stuck-at fault model:** Faults are fixed (0 or 1) value to a net. Stuck at-0 and Stuck at-1
  - single stuck-at fault model
  - Multiple stuck at fault model
- **Delay fault model:** Increase the input to output delay of one logic gate, at a time.
- **Bridging Fault:** A short between a group of nets
  - AND Bridge
  - OR Bridge

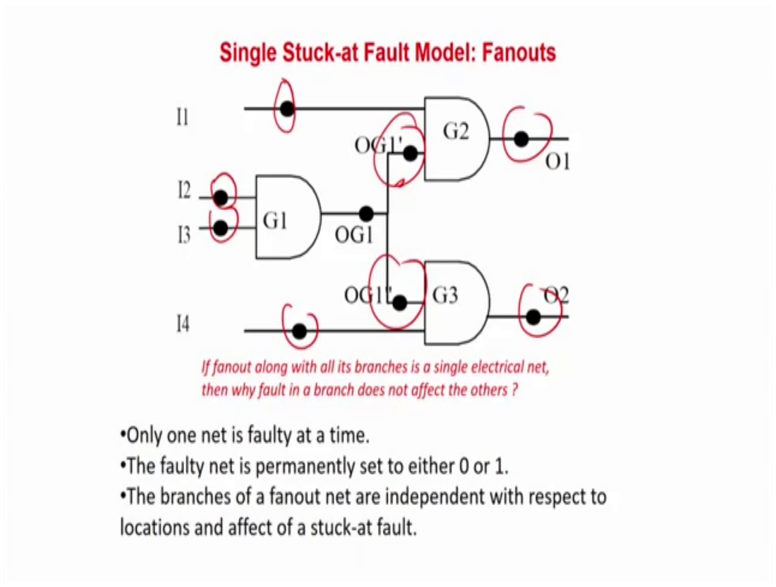
So, this is actually called fault model. So, people have found out different type of fault models which are very much closely related to the defects. But if you really think of defects the ponderosa box will open up. So, people have found out that defects can be many, it can be wide, it can be at the net level, it can be at the transistor level, it can be at so many different places. But how do you abstracting as a model? So, they are easy to represent and it is easy to test still the quality of assurance is high. So, by studying all these defects at the silicon level and device physics level people have found out many popular and widely accepted fault models. How do make it widely accepted statistical experiments.

So, what is one of the very important one is called the stuck at fault model that is the assume that each net will be either stuck at 0, or stuck at 1, there is something called delay fault model. That is this line the time taken to go from 0 to 1 is higher than direct, and this direct to fault from 1 to 0 the time taken for this fault is basically higher than the specify timely. Bridging fault between two gates there can be short the short can be and short or an or short this is after all the short basically this line will be a being like this. This two independent lines, but due to some short it will be a being either and short or it will be a being like an or short. Then different type of fault models, but these are the most accepted ones.

Now your job as we will see is just to verify then none of the gates I mean in current discussion basically we will be studying only on this. Whenever we will be talking about advance fault models in this lecture in this course we will go for such mould kind of advance faults, but today our discussion is mainly limited to stuck at faults.

Because this is the most widely accepted and till about last 15 years back this was the only fault model practiced. With more sophistication in fabrication technology slightly more is improved or advance fault models are also have to be considered. For stuck at fault model is very simple you assume there is a line the line is either stuck at 0, or stuck at 1. So, it captures a huge number of decades.

(Refer Slide Time: 34:00)



Single Stuck-at Fault Model: Fanouts

If fanout along with all its branches is a single electrical net, then why fault in a branch does not affect the others ?

- •Only one net is faulty at a time.
- •The faulty net is permanently set to either 0 or 1.
- •The branches of a fanout net are independent with respect to locations and affect of a stuck-at fault.

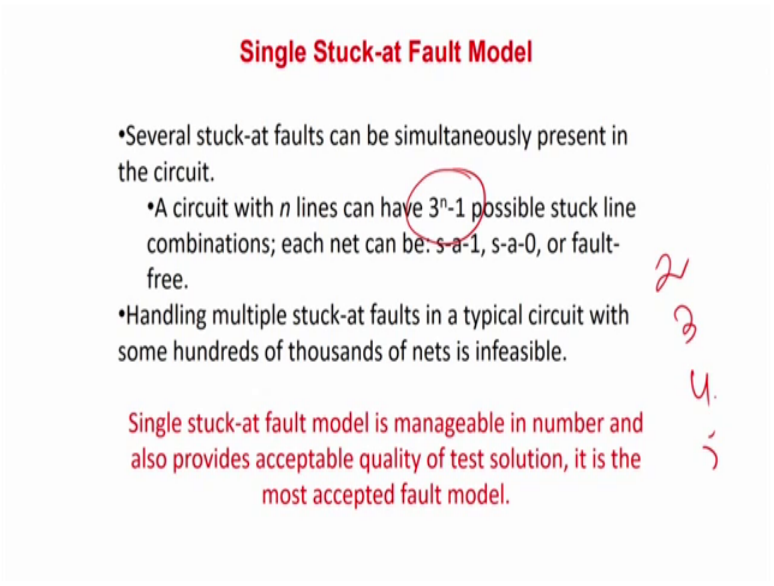
Now, what you have to do? You have to just verify that none of the line circuit lines is having a stuck at 0, or stuck at 1 problem. So, this is basically your I am giving you the circuit with example there is sometime small interesting example here. So, this is the look at this fan out is your fan out. So, as expected you will just say that either this line will have stuck at 0, or stuck at 1, there is only two faults possible.

But if you see these are continuous line still basically you are assuming this fan out to be independent and this fan out to be independent. What do I mean by this? That is I have to consider this fan out stuck at 0, stuck at 1 possibility and also I have to also consider this fan out branch independently and also this fan out branch independently. Now why? This is because people have found out that if you are assuming that such faults are not possible such faults are not possible; only this line can be stuck at 0 and stuck at 1, the there is certain quality in the correlation; because, as I told you this fault models are just abstraction.

So, you assume that there is stuck at 0, stuck at 1, stuck at 0, stuck at 1. So, all this lines will be having stuck at 0, or stuck at 1 faults. I will try to ensure that none of the lines is having a stuck at 0 and stuck at 1, I can I will also do it for OG 1, but logically looking at as this is the continue was net I can drop this fan outs. But if you do this we will have statistically found out that the quality of test comes down these are statistic, again I am telling you all this fault models have been coming out from the statistical improvisation, it is not by sorry statistical findings it is not by any theory.

So, therefore, along with this you have to also think about these two lines as independent all fan out lines will be independent and you have to ensure that none of this fan out lines or the normal lines or this ordinary lines should have a stuck at 0 or a stuck at 1. So, you have to apply test patterns and ascertain that none of the lines as either stuck at 0, and stuck at 1. If you can do this then it can be 99.999 percent sure at least this circuit is having no defects. So, this is something called a for stuck at fault model, which is very much in popular and with basic model we can stucked or teaching with. So, let us see what is the assumption you can look at it?

(Refer Slide Time: 36:04)



Single Stuck-at Fault Model

- Several stuck-at faults can be simultaneously present in the circuit.
  - A circuit with *n* lines can have 3^n-1 possible stuck line combinations; each net can be: s-a-1, s-a-0, or fault-free.
- Handling multiple stuck-at faults in a typical circuit with some hundreds of thousands of nets is infeasible.

Single stuck-at fault model is manageable in number and also provides acceptable quality of test solution, it is the most accepted fault model.

2
3
4.
5

Now, there are very interesting cases. So, how many at a time how many lines we will take. So, people always called as a single stuck at fault model; that means, I will assume that only this line can be stuck at 0, or stuck at 1 fault and I will find out test patterns to verify it. But I will not taken assumption that these two lines can be stuck at 1 or stuck at 0 at a time I can do that. But in this case what is the number of possibilities if there are n lines then there can be 3 to the power n minus 1 possible stuck at line combinations that is two lines can have stuck, any two lines, any 3 lines, any 4 lines, any 5 lines can be having a stuck at 0s and stuck at 1.

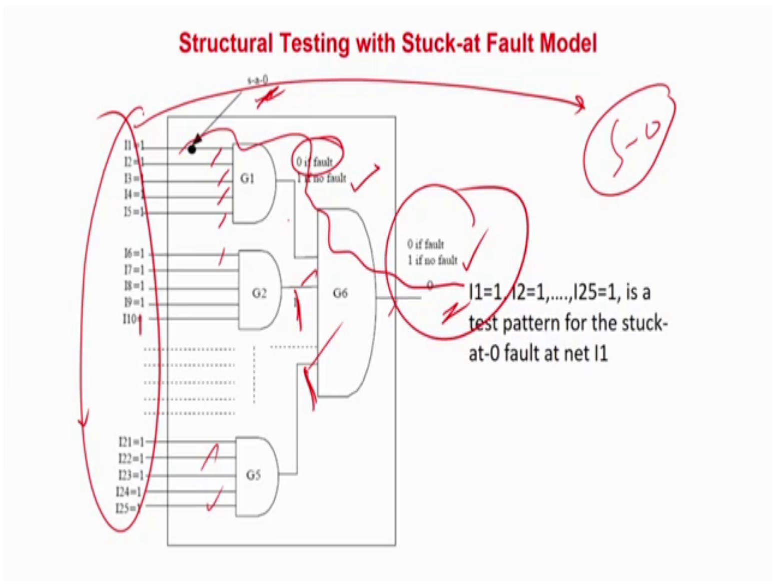
Like these two line can have if I assume that any two lines can be stuck at 0, or stuck at 1 together. So, the number of combination is these two will be a combination, these two will be a combination, these two will be a combination and so forth, huge number of such combinations will come up. Then also you can think few lines can also be stuck at 1, or stuck at 0 at a time, then such number of fault should be very very large and it is 2 to the power n minus 1. So, this is actually called multiple stuck at fault model.

So, due to the increase in this complexity people have dropped this multiple stuck at fault model concept. Their single stuck at fault model is manageable in number and also provide acceptable test quality solution it is the most accepted fault model. There is no problem that if you assume multiple combinations, but the number of faults will be so high, so you will be end up in the generic more number of test patterns and the main goal of reducing the number of test patterns or number of test pattern generation time will be gone. So, therefore, to side compromising quality we will assume that at a point only one of the lines can be stuck at a fault.

But all lines has to be tested first you assume that this line is stuck at fault, test it whether the stuck at fault is there or not, you repeat for this, repeat for this, repeat for this, all these lines you repeat one by one at a time. But at no point of time you will be you will assume that there can be combinations. So but in reality there can be combinations because if you assume that this line can be stuck why not this line. But people have found out that doing this means slightly increase the test quality, but it will not I have too much impact on the, but the amount of impact on test generation time when the number of fault is so high that the benefit will be nullify.

So, therefore, people go for this model only one net is fault at a time the net is permanently either stuck at 0 or all turns out the finite or independent with respect to the locations and after stuck fault that is you have to consider these are these two lines independently. So, handling multiple stuck at fault in a typical circuit with 100s of line is infeasible all though it may give you a better accuracy, but this is not possible.

(Refer Slide Time: 38:40)



Structural Testing with Stuck-at Fault Model

I1=1, I2=1,....,I25=1, is a test pattern for the stuck-at-0 fault at net I1

So, now examples we will see the benefits. So, what are the structural fault benefit? Structural test benefit was you need not apply 2 to the power n combination it is much lower, but the problem was you have to bring you have to probe lot of internal lines. Structural testing with fault models actually elevated this probing this we will see how? So, if you can that is our goal structural testing is very good, but only problem is that you have to probe out lines. if somehow that can be elevated or then our job if that can be nullify then our we are in the winning position.

So, assuming that I have to do testing of this and gates, so you have to check that none of this lines are having stuck at 0, and or stuck at one at a time. So, individually you have to stuck one line at a time, assuming that this is the first line we are taking with this any assuming to be a stuck at 0. So, if the stuck at 0 of course, what you have to do you have to apply a 1 over here. So, is the 1 means if this line is normal, so 1 will pass over now the normal line is 0 it will be 0 effect.

Now, of course all other lines has to be 1 of course, because if any of the lines is 0. So, this line will be nullify before if I make alpha is equal to 0 whatever be the case with this line fault or non fault the output of the AND gate 0 1, AND gate of G 1 output will be 0 and I cannot do anything. So, I have to make all this line as 1, so this is actually sensitized and I can look at the fault value. Similarly this gate output has to be coming out over here because importantly I cannot probe any of these lines I cannot probe, I can only probe this lines and I can only put control values over this lines

So, I have to observe these fault impact at this point of time. So, what I have to do? So, if any of this line input lines is a 0 to the AND gate the output will be a 0 irrespective of the fault effect, so I cannot have I can I cannot effort to do that. So, what I will have to do? I have to make all these lines 1, so if I may able to do this and in fact, all this lines I have made 1.

So, this line is actually sensitized to the output, so that I can observe the directly. the implication is I make all this lines has ones, that is all this lines has 1. Will allow me to say this value of this line at the output without having to put any kind of probes in between that is the advantage of fault model base testing that I do not need any probe over here, but still I can see the value at the output.

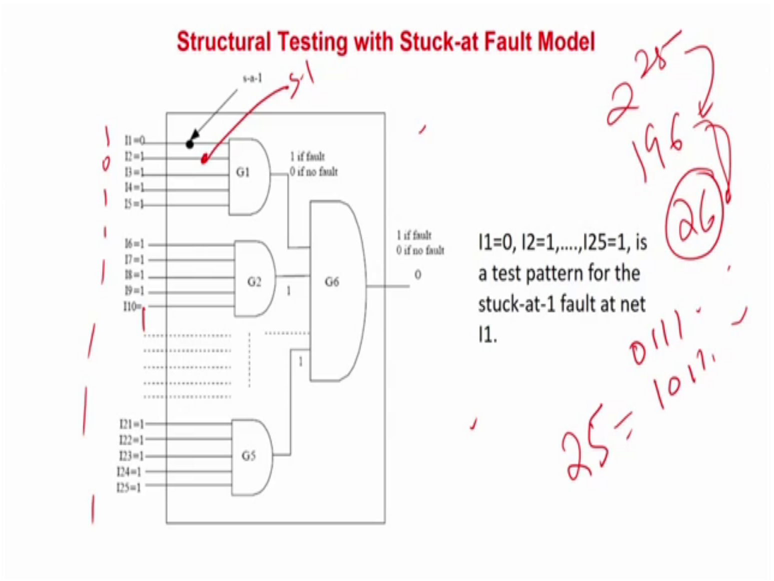
Now I put a 1 over here, and all ones over here. So, if the circuit is normal you will get the output as the 1 over here. If there is circuit is having a stuck at 0 fault over here you will get a 0. So, the output will be something like this 0 if there is a fault, and 1 if there is a not a fault. So, basically this is your test pattern all ones, all ones is your test pattern and if I am observing 1 at the output you can be very much sure that this note as packet 0 at 1 fault this is done.

Now, another interesting advantage of structural testing is fault model comes. You can easily observe that this pattern not only test this struck at 0 fault, but it will test this struck at 0 fault here herein all the lines will be ascertain not to have a stuck at fault. That means, the single test pattern actually test multiple stuck at 0 faults in this circuit. So, that is one very interesting phenomena which people have found out that if you are able to test the circuits with stuck at with fault models basically one test pattern will test multiple fault models.

So, here one test pattern is testing all this stuck at 0 faults due to repetition with the now with the different fault. Because this line has to be tested with the stuck at 0, or stuck at 1, but with the all the one pattern I have tested stuck at 0 at all the lines. Now my testing for stuck at 0 is over now I am taking a stuck at 1 fault over here. Now what I have to do? Again I have to put all the lines as a 1 because all the inputs of this AND gate should be a 1, otherwise the fault effect will not propagate.

Now, this is a stuck at 1, So, I will have to off course put a 0 over here. So, one if there is a fault because is to normally is which is 0, so 0 should pass over here. But if you stuck at 1, the answer is a 1 and basically if 1 there is a stuck at 1 fault over here, and a 0 if not fault.

(Refer Slide Time: 42:39)



Structural Testing with Stuck-at Fault Model

I1=0, I2=1,....,I25=1, is a test pattern for the stuck-at-1 fault at net I1.

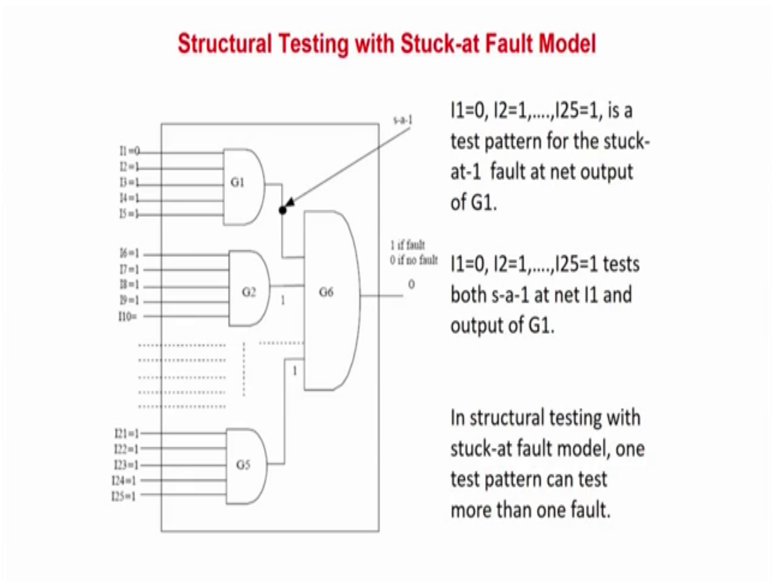
So, in this case this line basically test a stuck at 1 fault over here, this 1 also will takes test a stuck at 1 fault over here as well as it will also stuck at 1 fault here. Interestingly when I am testing this stuck at 1 fault, I am assuming that this is the only stuck at fault which is the existing these are non existed.

Now the next will be I will be eliminate this I will assume that there is a stuck at 1 fault at this point of at this point I want to testing it. It will be it will be interested to observe that the same test pattern can be applied basically. Similarly next what I will do is a next I will can eliminate this and assume that this is the only line which is at the stuck at 1 fault over here, of course the same pattern will be to the test.

Indirectly what it means? This test pattern we will test stuck at 1 here, stuck at 1 here, and stuck at 1 here. So, one test pattern here test for 3 stuck at 1 faults. Now if I want to repeat this for this line stuck at 1, so your answer will be 10111111; repeating in this manner with find out that there is 25 lines. So, 25 running combinations like 01111 then 10; 25 such patterns with test the whole circuit for all point stuck at 1. So, 25 plus 1 26 patterns will prove that there is no stuck at faults.

So, we started from 2 to the power 25 then we came to 196 and now we have come to 26 number of test patterns. So, this junk is very well established because was this first one we said that we can do structural testing still we can have very high polarization with functional test. But the problem was that we have to put lot of multiplexing arrangements, then we came near with stuck at fault models or fault models. Then we have find out that this number is also coming down and also more importantly would not need extra pin outs.

(Refer Slide Time: 44:22)



Structural Testing with Stuck-at Fault Model

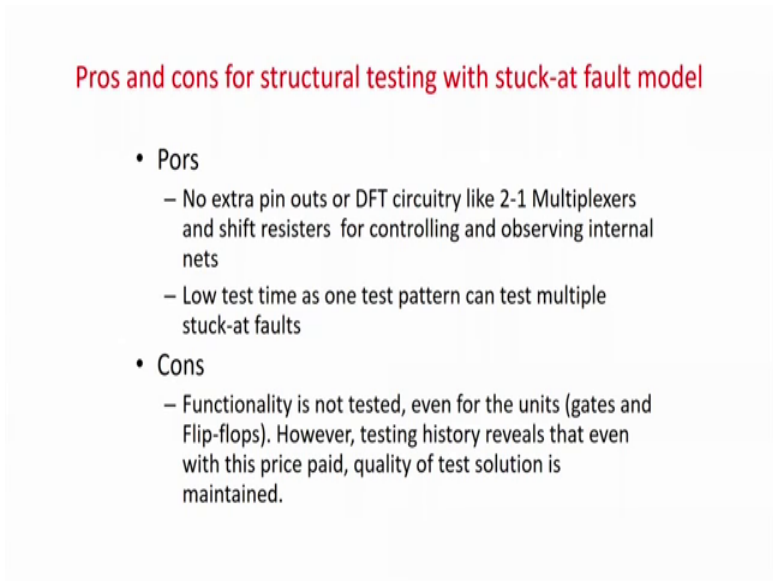
I1=0, I2=1,....,I25=1, is a test pattern for the stuck-at-1 fault at net output of G1.

I1=0, I2=1,....,I25=1 tests both s-a-1 at net I1 and output of G1.

In structural testing with stuck-at fault model, one test pattern can test more than one fault.

So, therefore, structural testing with fault models became de facto standard and all testing. Now in case my embedded system hardware, or pure VLSI hardware are with structural fault models, and structural testing with the concept of fault models. So, this example you can look at it. So, in structural testing, one stuck at fault model which stuck at fault model, one test pattern can test more than one fault, so that is another benefit of structural testing with fault models.

(Refer Slide Time: 44:45)

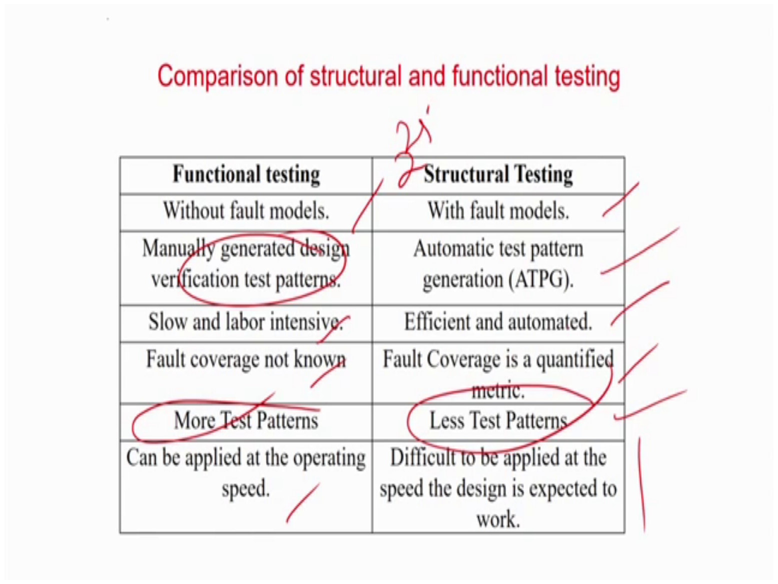


Pros and cons for structural testing with stuck-at fault model

- Pors
  - No extra pin outs or DFT circuitry like 2-1 Multiplexers and shift resisters for controlling and observing internal nets
  - Low test time as one test pattern can test multiple stuck-at faults
- Cons
  - Functionality is not tested, even for the units (gates and Flip-flops). However, testing history reveals that even with this price paid, quality of test solution is maintained.

So, again comparing pros is no extra pin outs or DFT circuits, DFT circuit means extra circuits which we have to put it to is testing, multiplexer, shift resister nothing is required. Low test time as one test pattern can test multiple fault. Problems are functionality is not tested not even at the gate levels. So, in structural testing we do not test the functionality even the gates at the structural testing was testing at the gate level functionality, but here we have totally it shown out the functionalities of the gates. However, testing history or statistics reveal that even with this 5 space the quality of solution is maintained that is slightly compromised in test quality.

(Refer Slide Time: 45:29)



Comparison of structural and functional testing

Functional testing	Structural Testing
Without fault models.	With fault models.
Manually generated design verification test patterns.	Automatic test pattern generation (ATPG).
Slow and labor intensive.	Efficient and automated.
Fault coverage not known	Fault Coverage is a quantified metric.
More Test Patterns	Less Test Patterns
Can be applied at the operating speed.	Difficult to be applied at the speed the design is expected to work.

So, therefore, this was the huge revolution in testing that I can test at structural level with fault models, but still the compromise in quality is much much less. So, this is the comparison functional test is without fault models, structural test is this fault models, functional test, manually generate test patterns basically we never applied 2 to the power 25.

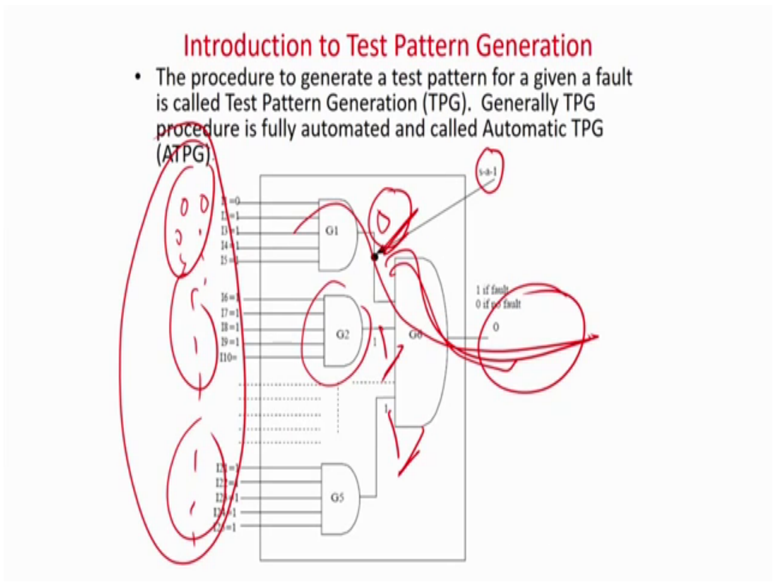
So, or 2 to the power n basically people restrict it to a less number by manually intervening. That is we know the functionality of the circuit you know what test patterns to test it. There is it can be taken from the verification fees. So, functional testing can still be done without 2 to the power n, but lot of manual intervention is required to tell which test patterns to apply. Slow and labor intensive, fault coverage is not known, more number of test patterns and cannot be applied at the operating speed. So, those things we will see later.

But the idea is that huge number, and in fact, if you want to reduce it you have to manually find out which are the most important test patterns. Structural testing is with fault models you can always apply a algorithms to generate the test patterns. Because you know that we did not at all looked at the functionality of the circuit, just the structure of the circuit is there, stuck at fault model is there. We have to find out which test patterns can verify that null of the faults are there.

Today we will also see some of the ATPG kind of algorithms; efficient and automated, fault coverage is the quantified metric; that means here if I say that I am functionality tested your circuit. What do I mean by that? Have you applied the 2 to the power n not possible and you have to tell the engineer knows the functionality of the circuit and he has applied the most important test patterns to check it. So, there is no quality, not quantity verification quantitative guarantee, but here I can tell there I have tested the circuit with 100 percent coverage for stuck at faults.

And we know that if somebody has then such that testing with 100 percentage coverage for stuck at faults the accuracy is 99.9 plus. So, you know that if I am setting 1 million chips very few of them will be wrongly build. So, this assurance is given to the customer, less test patterns and basically these are the points will seen later basically, so this is what is the advantage of structural testing with fault models.

(Refer Slide Time: 47:26)



Introduction to Test Pattern Generation

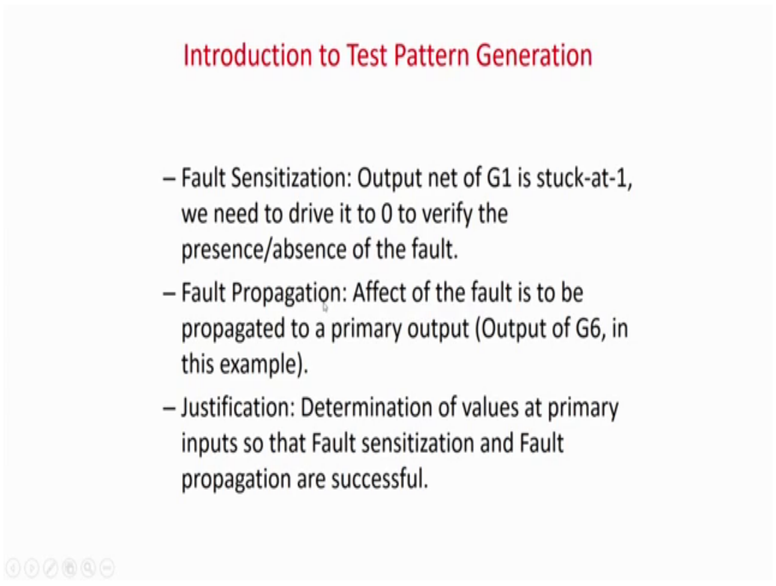
- The procedure to generate a test pattern for a given a fault is called Test Pattern Generation (TPG). Generally TPG procedure is fully automated and called Automatic TPG (ATPG).

Now, the second concept that given fault model in a given circuit, will I manually generate the test patterns is it possible? No. In functional testing there is no solution because the engineer has to low the functionality of the circuit and you just generate the test patterns manually, but in case of structural testing with fault models we can automate it. So, this is actually called automatic test pattern generation algorithms the algorithm will generate the test patterns for you. So, what it is something like stuck at 1. So, if the circuit is stuck at 1 you have to apply always apply 0 to it, then you have to propagate the value to the output. So, we have to propagate the value to the output means this is the AND gate so all other inputs to be 1.

So, this is we are calling as propagation simple graph traversal. So, 0 means sensitization it will come to the output all other lines has to be 1. So, then you have to find out to make this line 1 what was the in output this should be the case and so forth and this line is 0. So, how can we make this line is 0? So, it can be either 0 111, or it can be all zeros or any of the pattern. So, whatever I was doing like this is a simple graph traversal with Boolean function manipulation we will give you the test patterns to test the stuck at faults.

So, this is actually called automatic test pattern generation and is a three phase algorithm. Sensitize whatever we stuck you apply a given value propagate the value to the output. So, that you can observe and now justify this requirements. There is all ones or some values will to it. So, this is actually called a 3 part three phase this pattern generation algorithm and I can easily write a C code to implemented. In fact, there lot of cad tools like (Refer Time: 48:53) max lot of mobile stores tools, also we if you give the circuits and the fault model automatically the test patterns will be told to you.

(Refer Slide Time: 49:05)

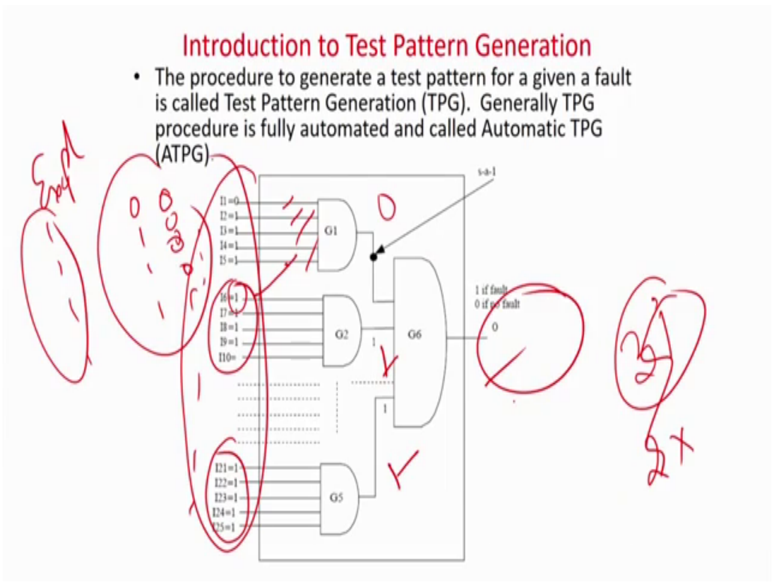


Introduction to Test Pattern Generation

- Fault Sensitization: Output net of G1 is stuck-at-1, we need to drive it to 0 to verify the presence/absence of the fault.
- Fault Propagation: Affect of the fault is to be propagated to a primary output (Output of G6, in this example).
- Justification: Determination of values at primary inputs so that Fault sensitization and Fault propagation are successful.

So, what I was telling is a three phase algorithm fault sensitization, fault propagation and justification. Sensitized means stuck at 1 you apply 0 and vice versa, propagation means the fault effect has to be propagated to the primary output. And justification means you have to go back so that this whatever price I want I want to I want this value, so what are the primary input values I have to keep. So, this is a, but it is not as drain as there can be lot of conflicts. So, if there are conflicts we have to apply different test patterns to solve it I will take examples. Like in this case basically if you see for justification I recall all 1's over here.

(Refer Slide Time: 49:38)



Introduction to Test Pattern Generation

- The procedure to generate a test pattern for a given a fault is called Test Pattern Generation (TPG). Generally TPG procedure is fully automated and called Automatic TPG (ATPG)

So, there is only 1 solution, all 1's over here, but I require a 0 over here. How many different patterns can be possible for this gate? It can be all 0s, or it can be 0111, so any pattern accept all 1's. So, accepts this any pattern can be applied to get a 0 over here. So, there is some kind of conflict; conflict means you require a 1 for this you require a line this line to be 1, but to, but for some other purpose this line cannot be 1. So, there can be conflicts.

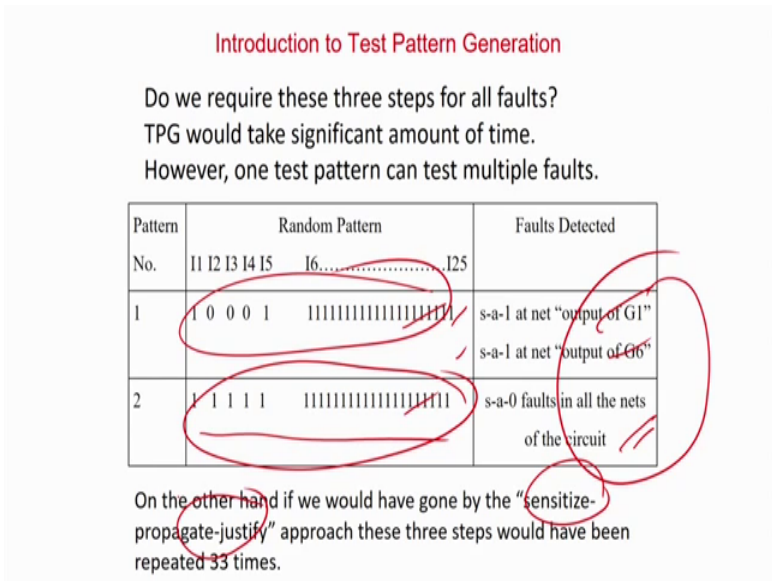
So, to and then you have to reiterate, you have to find out another way to generate the test pattern. So, it is not always green some there is there will be conflicts then we have to again reiterate and find out which test patterns can be able to tested, but again one important thing is that for some faults there cannot be any test patterns. So, in that case you will keep on iterating and finally, we will declare that there is no test patterns to there is no test pattern to test that fault and there is fault is basically a redundant fault.

But anyway there is some stray cases I am not much emphasizing in this course because this things are well elaborate is any kind of standard digital testing course, but the idea is that and trying to say either sensitized propagate and justify is the approach to generate a test pattern. But sometimes there can be conflicts in which case you have to reiterate and find out the new test patterns.

But for some faults even if to lot of free fully iterations we will find out that this is no patterns which can tested and such faults are called as not directable faults. So, this is the three phase algorithm ok. So, how do you do this? Now this is the very interesting concept. So, in this case if you see how many faults are there can be 25 lines, 25, 26 and 20, some 30 they may be say 30th, how many 25, 30, 31 lines in this case.

So, how many stuck at faults can be possible? 2. 2 into 31 say around 60 stuck at faults will be there, stuck at 0, and stuck at 1. Now what I have to do? I have to do it for all the lines, all the lines I have to find out take up for testing, take up for generate the test pattern by this (Refer Time: 51:29) propagate and justify algorithm.

(Refer Slide Time: 51:30)



Introduction to Test Pattern Generation

Do we require these three steps for all faults?
TPG would take significant amount of time.
However, one test pattern can test multiple faults.

Pattern No.	Random Pattern I1 I2 I3 I4 I5	I6..........I25	Faults Detected
1	1 0 0 0 1	11111111111111111111	s-a-1 at net "output of G1" s-a-1 at net "output of G6"
2	1 1 1 1 1	11111111111111111111	s-a-0 faults in all the nets of the circuit

On the other hand if we would have gone by the "sensitize-propagate-justify" approach these three steps would have been repeated 33 times.

So, if their circuit is very big. So, this integration has to be done for all the lines into 2, because it is length can have a stuck at 0 and stuck at 1. But people have found out not require to do like that even there is a some approach called random test pattern generation that is you just randomly apply some patterns and see how many faults are getting tested, or given some random patterns you find out how many faults will be tested if I apply those random patterns. And then some faults will still remain which is called difficult to test faults.

For those of course, you have to go for this deterministic test pattern generation by sensitize propagation and justify. There are very simple example interesting and funny example see if you go to a fare and there this is a mela, that is a fare if you go there is a lot of balloons in this screen and you have to take a gun and you have to shoot down the balloons. So, since that there are around 100 balloons which are in the screen, so initially and you have to shoot down all. So, what will be your approach? You will randomly start firing. So, most of the balloons will be blasted and only few of them will remain for them you have to go for a man shoot. The same thing happens for testing that if there are all faults will be tested instead of just going for sensitized propagated and justify algorithm that is m shooting do not go by there. Apply some random patterns and see how many faults get tested? After certain amount of time you will find out that even up the applying random patterns some faults are not being tested; that means, they are difficult to test faults and un testable fault. So, you have put a one by one such faults and go for sensitized propagate and justify base faults.

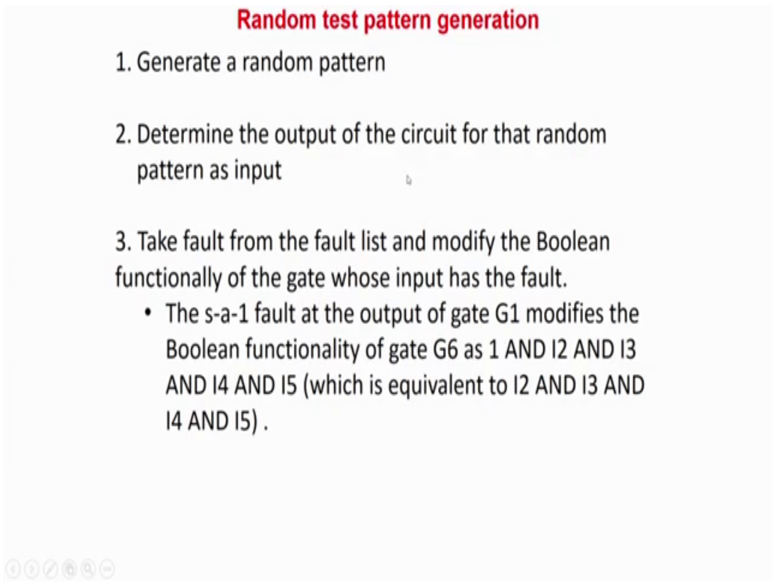
So, but if I base testing all faults will be covered by that skip accept few will remain or few may remain which are non testable fault, so we go by in this. So, for example, I assume that I have to test the circuit this circuit and I apply a random pattern like 10011 and 1 you will find out that stuck at 1, fault at this 1 is tested, stuck at 1 fault of this 1 is tested, you take another in another random input like this.

All stuck at 0 faults in the nets of the circuit. So, how many some 30 faults are tested and come faults are tested go by this. So, randomly I apply this pattern and find out which faults get tested this is the 2 fault getting tested, I apply all ones is a random pattern. Because random is some arbitrary random pattern you generate and apply and see how many faults get covered. If this is generate a randomly we will find out that stuck at 0 faults in all nets are come generally in random test patterns also be sometimes apply some deterministic ideas that we always use all 0, and all 1 patterns to go for testing.

Because they have slightly high coverage's but these are some off shoots from the off shoots from the algorithm, but for a very normal situation you just see I apply random patterns may be there are the two random patterns generally I apply I found out the these are the faults already covered. I will keep on doing it with random patterns and finally, I will see find that now there is no improvement that is from one random pattern to the next random pattern the number of extra faults being detected is becoming less. So, at the time I will stop and start applying the sensitized propagate and justify approach.

So, with this deterministic algorithm is not required to be applied for all cases. We will be required only for difficult to test fault and un testable faults.

(Refer Slide Time: 54:33)



Random test pattern generation

1. Generate a random pattern

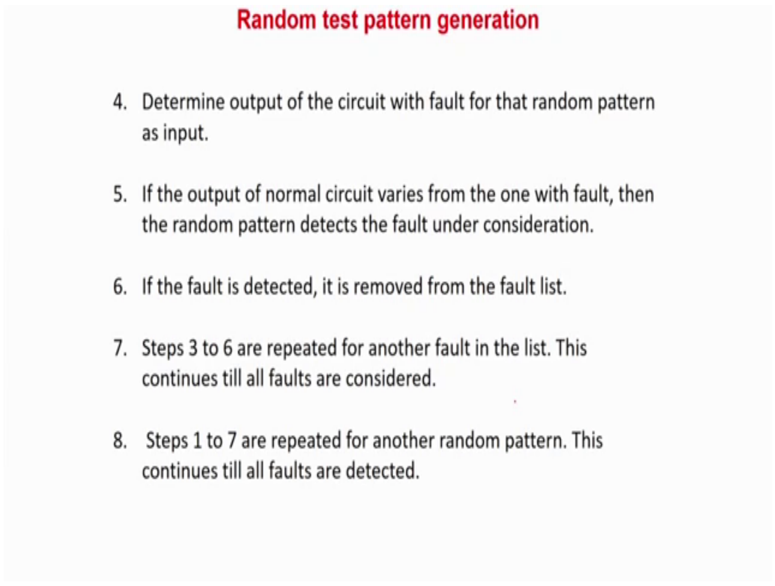
2. Determine the output of the circuit for that random pattern as input

3. Take fault from the fault list and modify the Boolean functionally of the gate whose input has the fault.
   - The s-a-1 fault at the output of gate G1 modifies the Boolean functionality of gate G6 as 1 AND I2 AND I3 AND I4 AND I5 (which is equivalent to I2 AND I3 AND I4 AND I5) .

So, basically what is the algorithm? Generate a random pattern determine the output of the circuit for that random pattern as the inputs. Take faults from the fault list and modify the Boolean functionality of the gates would inputs has the faults that is one I am doing. I am taking a random pattern I am applying this to this one I find out the normal values. Then I take one fault at a time and I change the Boolean values and I find out which faults are getting detecting by the random patterns.

Those faults I will keep on deleting from my fault list. Like in this case this is example we have found out that these two random patterns is directing this faults and easily I can find out by modifying the Boolean functions over here.

(Refer Slide Time: 55:08)



Random test pattern generation

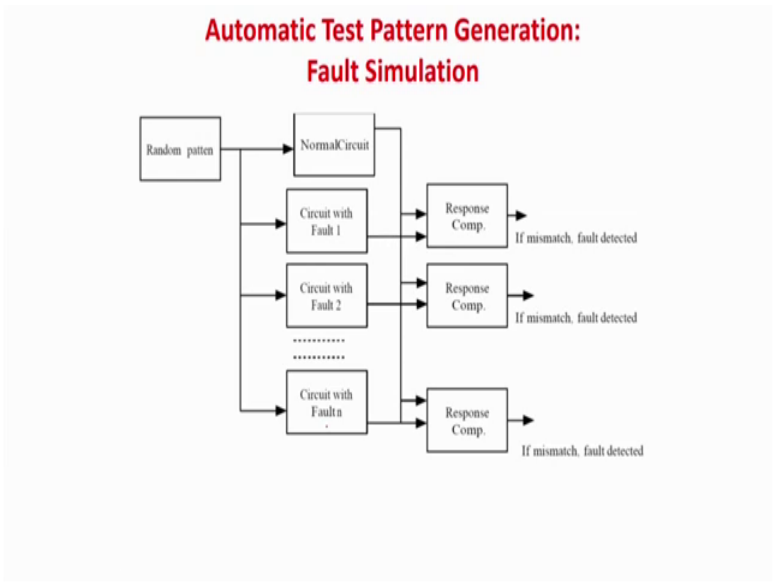
4. Determine output of the circuit with fault for that random pattern as input.
5. If the output of normal circuit varies from the one with fault, then the random pattern detects the fault under consideration.
6. If the fault is detected, it is removed from the fault list.
7. Steps 3 to 6 are repeated for another fault in the list. This continues till all faults are considered.
8. Steps 1 to 7 are repeated for another random pattern. This continues till all faults are detected.

That is even if you forget about the idea of manipulating Boolean functions you can just see that I random I randomly generate a test patterns, and find out which faults are getting detected. This I go by comparing with the normal output of the circuit for that given random pattern.

And whichever faults are getting detected by difference in the output values in the fault case versus normal case those fault you can remove. And finally, it keep on doing it till you find out that basically the number of extra test patterns like extra faults getting per new test pattern is not very high. Determine the output of the circuit with fault for that random pattern if the output of the normal circuit varies from the one with fault then that random patterns detected fault under consideration.

If the fault is detected remove from the list. These are repeated for another fault in the list this continuous till all faults are considered. You take a random pattern find out the normal value. Find out which faults are getting detected by the random pattern simple you take one fault at a time apply the random pattern if the output of the circuit is different from the normal one that fault is detected as this fault is not detected by this random pattern. You keep on doing it for all the faults which faults are detected you throw them out.

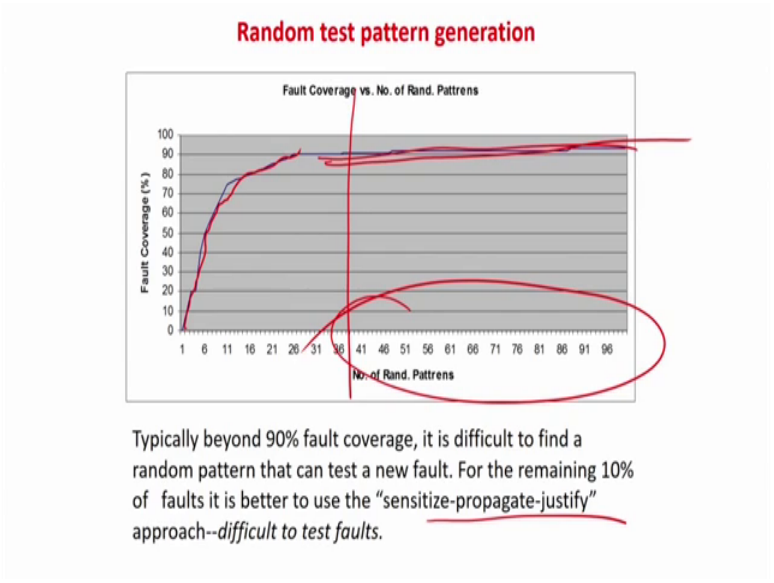
(Refer Slide Time: 56:21)



Automatic Test Pattern Generation: Fault Simulation

And next you take a another random pattern and repeat the same procedure with the number of faults remained. You keep on doing it then we will find out basically this is the diagram which shows this a random pattern normal circuit; circuit with fault 1, 2 and 3 you simulate with the random pattern. If there is a miss match that fault is detected and then this faults will be thrown out from the list and you can if to repeat for a different random pattern very simple logic.

(Refer Slide Time: 56:33)



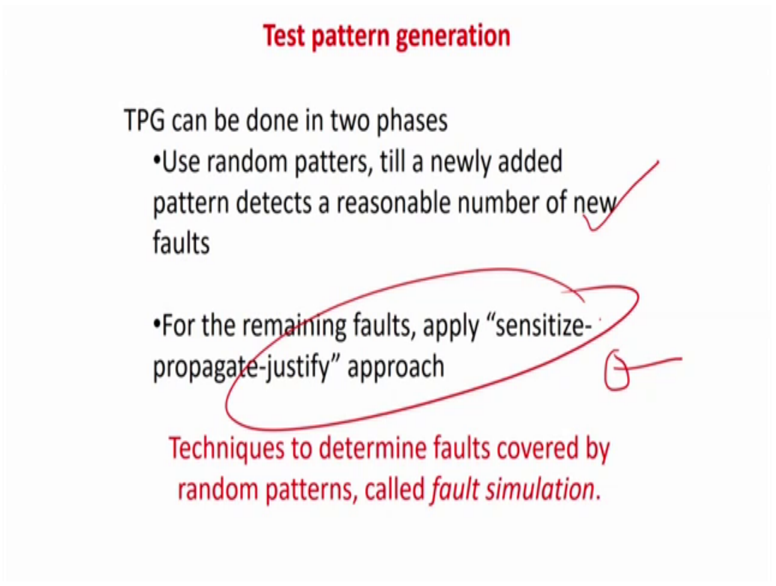
Random test pattern generation

Typically beyond 90% fault coverage, it is difficult to find a random pattern that can test a new fault. For the remaining 10% of faults it is better to use the "sensitize-propagate-justify" approach--*difficult to test faults.*

After sometime you will find out that with new test patterns the number of fault being detected will become very high. But after a certain amount of time you will find out that even with new number of random patterns this is almost flat or slight increase. That is first random pattern you take 100 faults get detected, second random pattern you take another 100 fault detected, then 80, then 70, new faults will get detected by the one more new test random pattern you are taking.

But after a certain amount of time you will find out that when there will be very less amount of faults remaining with the new test pattern the number of random test pattern. The new faults getting detected will be very less. So, at that point we will stop and means start applying sensitized propagate and justify approach to test those faults. So, basically that is what is the most critical part of testing, because random test pattern generation is very very simple just generate a pattern find out which fault is there.

(Refer Slide Time: 57:25)



Test pattern generation

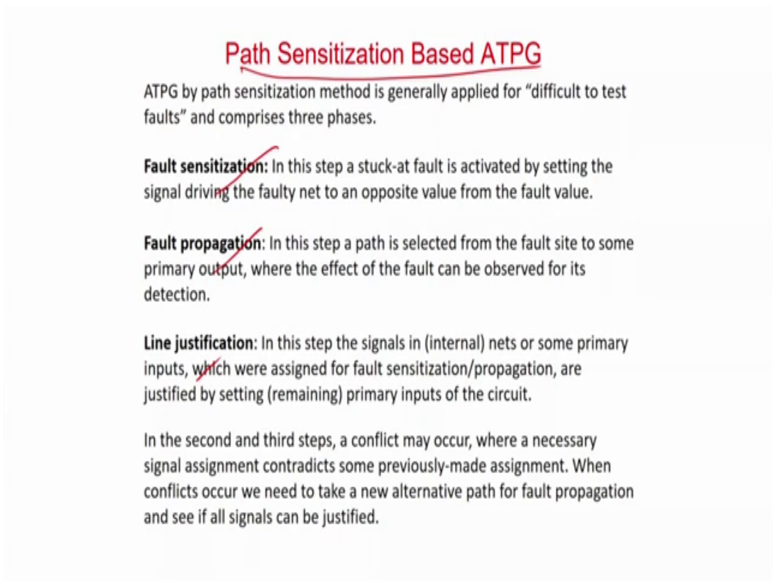
TPG can be done in two phases

- Use random patters, till a newly added pattern detects a reasonable number of new faults
- For the remaining faults, apply "sensitize-propagate-justify" approach

Techniques to determine faults covered by random patterns, called *fault simulation*.

So, that is why we say that test pattern generation is done in two phases; one is the random manner in whichever faults are remaining we have to apply sensitized propagate and justify approach. And also more than 99.999 percent faults will be detected by these two phases, but few faults may remain undetected because they are non testable faults. And that can you will be only able to find out by this approach.

(Refer Slide Time: 57:44)



Path Sensitization Based ATPG

ATPG by path sensitization method is generally applied for "difficult to test faults" and comprises three phases.

Fault sensitization: In this step a stuck-at fault is activated by setting the signal driving the faulty net to an opposite value from the fault value.

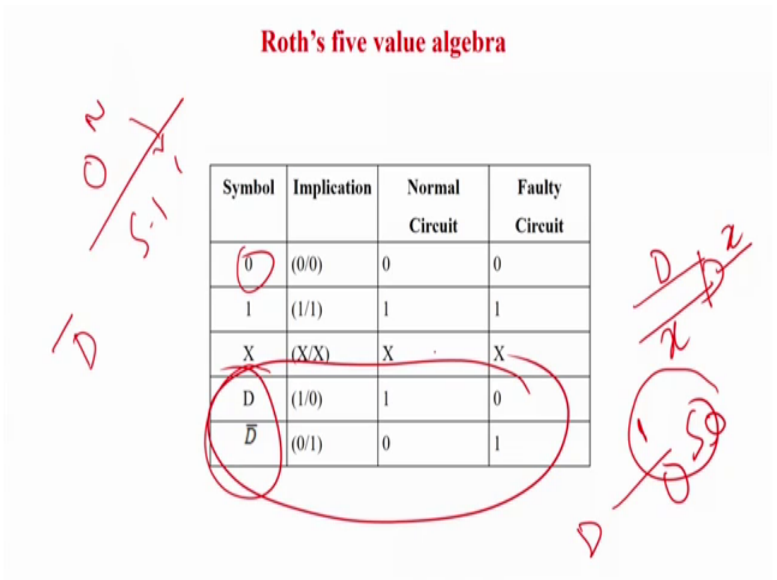
Fault propagation: In this step a path is selected from the fault site to some primary output, where the effect of the fault can be observed for its detection.

Line justification: In this step the signals in (internal) nets or some primary inputs, which were assigned for fault sensitization/propagation, are justified by setting (remaining) primary inputs of the circuit.

In the second and third steps, a conflict may occur, where a necessary signal assignment contradicts some previously-made assignment. When conflicts occur we need to take a new alternative path for fault propagation and see if all signals can be justified.

So, this is what is the basically this is the this is called path sensitization based ATPG that is Sensitization Propagation and Justification. So, with these three phase algorithm, we can find out the faults how the faults can be tested which are difficult in nature.

(Refer Slide Time: 58:00)



Roth's five value algebra

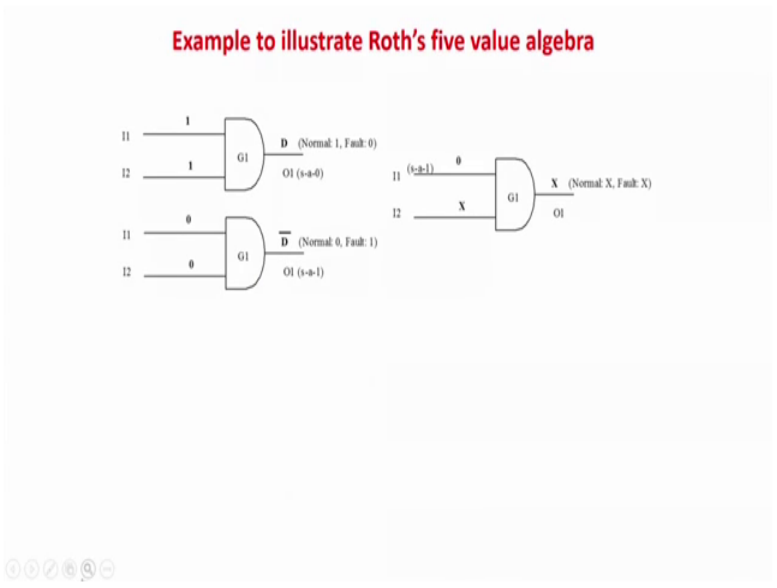
Symbol	Implication	Normal Circuit	Faulty Circuit
0	(0/0)	0	0
1	(1/1)	1	1
X	(X/X)	X	X
D	(1/0)	1	0
$\bar{D}$	(0/1)	0	1

0
s-a-1
/D
D
x
s-a-0
D

Now, I will go I will give you an example which will clear basically how test patterns can be generated for when the three phase algorithm for the which is called the generalistic test pattern. Before that there is some Roth's five value algebra you have to know 0 means 0 0; 1 means 1 1; X means X X; X means say for example, this is and gate. The input is 0, and the this input I do not know the output is X so this is a unknown input.

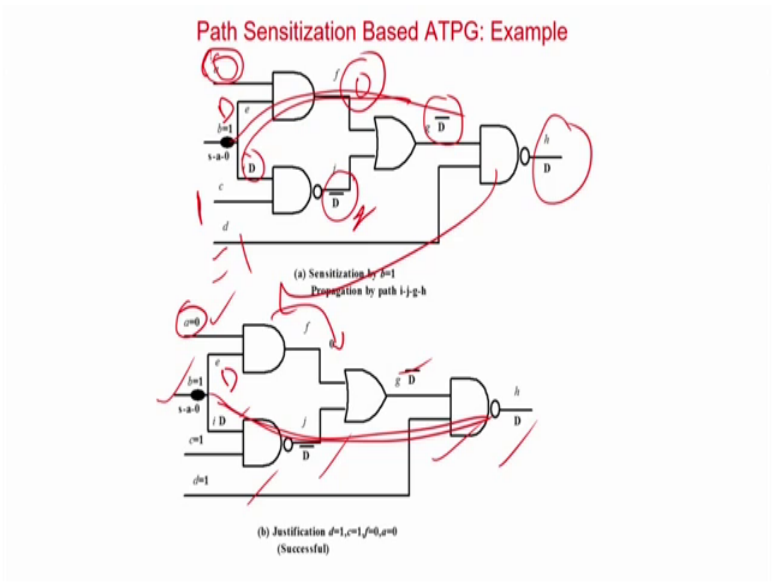
There is something these are the two things we generally introduce apart from normal Boolean algebra D and D bar. So, D means what? D means normally the line is 1, but due to stuck at fault this line is becoming a 0. So, this is D and D bar is something like this line normally is 0, and because of a stuck at 1 fault the line is becoming a 1. So, there are the two basic lines we introduce.

(Refer Slide Time: 58:50)



So, this is what is the example you can just have a look at it, so 11 the answer is a D; that means may be this is if the stuck at 0. So, normally answer will be 1 because the stuck at 0 will be 0. So, this line I have to put a D, stuck at 0, stuck at 1; 0 X the output of the AND gate will be a X. Because I do not know if this line is a 0, the output will be a 0. If this line is a 1, then the answer will be a D. So, basically; that means X means unknown. So, in this case it is 0 0; the answer should be a 0, but due to stuck at 1 fault the answer is a 1. So, I will represent as a D bar, it is just Roth's five value algebra.

(Refer Slide Time: 59:25)


Path Sensitization Based ATPG: Example
b=1
s-a-0
D
$\overline{D}$
h
(a) Sensitization by b=1
Propagation by path i-j-g-h
a=0
c=1
d=1
(b) Justification d=1,c=1,f=0,a=0
(Successful)

Now, I will tell you the example of how automatic test pattern generation is done and also show the example of a conflict. So, this is the circuit assume and we have to test for stuck at 0 at all the lines. I am just taking a single stuck at 0 faults over here. Then what is my sensitization? So, I have to make b equal to 1, now interestingly there will be one path through propagate by this and one path through propagate by this.

So, while lot of iterations will be required because we will have multiple options that your hand, so you have to decide at what you have to do. Now what is the lot of advances in algorithms have come up for such pattern generation which will tell you an estimate which is the better path. But I am not going into such advances in advance algorithm, because they are part of advance test architect advance test, advance VLSI test courses. Here I my job is to show you the basic idea of testing and how it is applicable in embedded systems.

So, all this heuristics I am not going into. So, basically which part you want to choose. So, there are some heuristics which will tell you which is the best one, but anyway at present I do not have such heuristics in hand. So, I will take the path e, f, g, h. So, what is 1 normal 1 fault is 0; that means, what it is D, that is normal 1 fault 0 is D by Roth's algebra. So, I have to propagate this values at the output, now you since the AND gate, so this should be propagated D and there is the inversion is an AND gate. So, it will be in inverted to D simple Boolean logic. So, this is sensitization.

Now I have to propagate I have to justify this is sensitization this D prime is propagation and now I have to sense justify it. Now justification means what this is D prime, this is a NAND gate. So, always the answer of the NAND gate should be other input should be a 1, if this is a 0 directly the NAND gate will be 1 and this fault propagation value will be lost this is the OR gate, so this D has been propagated to here. So, I should have a 0 over here as the justification because why because this if I make a 1 over here the output of the AND gate will be 1, and the fault propagation will be loss. So, I put a 0 over here; AND gate of course, this has to be a one otherwise this will not propagate. So, this is a 0.

So, how do I get a 0 in the NAND gate? So, basically there is only one choice that is you have to put a 1 over here as this is the, we have to put a 1 over here. So,, but by this b equal to 1; this is D, as well as these this is D, but I require a 0 at the NAND gate, 0 at the NAND gate means basically I require a 1 and a 1 over here this should be 1, this should be 1; this is 111 is getting. Only if the possible this can be a 0, but now you see I require a 1 over here, but this is stuck at 0 apply 1 so D. So, this is D is coming over here, so this is basically a conflict at j. So, as I told you life is not we are agree you can have lot of conflicts.

So, justification tells the j equal to 0. So, j equal to this 1 is equal to 0, if an only if j equal to 1, so sorry. So, this is 1 I have told you, so justification we will require j equal to 0, but basically this is already D by this, so this sorry. So, this is already at D over here you can easily observe over here. So, this D is over there and basically it will be a D prime over here something like this. So, there is a conflict or more better way to observe is something like this. As I told you, so this is the 0, so NAND gate becomes a 0 only if both of them are 11, because 11 is 1 for a AND gate and inversion will be a 0.

So, this is the only combination you have, but here is already D. So, this is the conflict basically this line we cannot get a 0 over here. So, there is no question having a 0 over here. So, with that there is a conflict in this one this case. So, this choice of path has been a problem for us. So what I have to do? There is another path possible to do this have this path not been available though you could have say this fault is a non testable fault, but now I have another path available, so I will use the city region. So, I have given this example basically to show that: what is an automatic test pattern generation algorithm.

Sensitize propagate and justify also it is giving your picture that always you may not get, always you may not get basically what I have told is that always you may not get this is so called your solution in one iteration there can be multiple iteration. So, now, I have they are taken this path, so 1 0. So, this is D of course then basically these inversions it will be D D inverse; D will be the inversion because of the NAND gate this is the propagation path. So, D bar again in inversion is D, now you have justified this.

So, justify of course, this gate will be a 1 correct, then this OR gate the other end should be a 0 this I can very easily get to be a 0 because I can make it as a 0, but this line is also a D. But basically by making this line is a 0 I can get a 0 over here, so this is justified. Now what? Now, I can just simple put a 1 over as c, if I put a 1 at the c, so this D will be propagated here and there is a justify and D prime will be gone here. So, the conflict at G is now no longer there, so it is basically successful test pattern generation.

So, you see I have a 0 over here, only interesting is that this 0 is making this line directly as 0 even if it is a D over here this is actually plot. So, there is no conflict happening and similarly is what is the idea this is you can see this is what is been done. So, this shows you basically what is an automatic test pattern generation for difficult to test fault. Sensitize, propagate, and justify. So, I have to repeat it for all such single faults because I have done for this, I have done for this, I have to done I would do it for all the lines and basically your job is done. If I by only the line I means whichever lines could not be tested by the random test pattern generation algorithm.

So, all these lines which were tested by the random test patterns need not be covered by this method. So, it brings down the complexity hugely because more than 90 percent of the faults will be detected by the random test pattern generation. A few of the difficult faults will remain which I have to go by this algorithm, then these are there are lot of the basic algorithm is called D algorithm.

There are lot of variations I put n fan etcetera and with lot of heuristics coming into picture with which tells you then I should take this path before and not this path before. Because if I could have taken a correct path in the beginning there should not have been any iterations; there are lot of modifications which brings more efficiency to the test pattern generation algorithm. So, there can be plant in any kind of a standard hardware testing course, but here I have try to give you the gist that, what is an ATPG based testing. What is the basic algorithm and why there can be iterations and so forth.

And you have to keep it for a and you have to keep on repeating it for all the fault still you find out a test pattern which detects it the difficult to test faults. And finally, you also find you may find out some faults for which no test pattern could be generated because you have would have exhausted this part this for all the paths you would have exhausted, still you could not have got a sensitized propagate and justify successful. So, that fault is a un testable fault, so those faults cannot be tested at all.

So, with this basically test patterns will be generated. So, when the circuit has to be tested this faults at these patterns has to be applied physically and the golden response in this case is D. So, in the previous case whatever be the golden response means what are they may expected response that the test pattern apply to the circuit we will tell you and that has to be match to the physical testing and if it is proper this circuit is to correct as there is a fault. So, with this we come to the end of automatic test pattern generation for combinational circuit. In next class we will see how thing start becoming complex if there is sequential elements in this.

Thank you.