Advanced VLSI Design
Prof. Virendra K. Singh
Department of Electrical Engineering
Indian Institute of Technology – Bombay

Lecture - 35
VLSI Test Basics – II

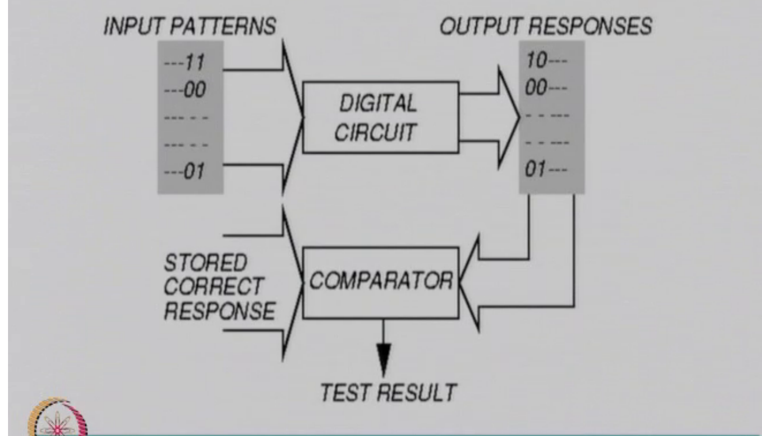
Welcome to the lecture series on Advanced VLSI design course, today I will discuss about basics of VLSI test during last lecture we will discuss about the test economics why test is so important? How much it will cost? If you apply a sequence of test vector onto your chip and we discussed that it has two parts one part is to generate a compact test set and then you have to apply that compact test set to each and every chip.

And now when you apply a test to each and every chip it cost you something about 2 to 5 cents per second that means if you test your chip say for a minute it may cost you about 5 dollars sorry about 3 dollars and hence it is too expensive to apply a test for minutes, therefore we have to test our chip somewhere between few seconds to minutes that is the reasonable time, but we have also seen that if you want to apply functional test.

So that means if you want to apply all sequence of test vectors that may be applicable to a chip say a chip as n input then there are 2 raise to the power n inputs you can apply right and today's chip have say hundreds of inputs then it may take billions centuries. Now you see the difference between the exhaustive test and reasonable time in which you can apply the test.

So there is a huge difference now it is the responsibility of test engineers to bring down this exhaustive test to a test set which is which can be applied in reasonable time and it gives you almost similar confidence that you got from or you would have got from the exhaustive test, okay.

(Refer Slide Time: 03:01)



So now here let us look at what are the various methodologies how I can generate that compact test set that can be applied with in few seconds, as I discussed earlier as well that how good your test set should be that depends on how much defect level that you can afford to, like for example for Automatic applications the kind of defective parts per million manufactured parts they are looking for a very, very low ideally they want 0.

But here that can be from somewhere from 1 to 10 defective parts per million manufactured parts are affordable, so your test quality depends on what kind of PPM figures you are looking at, now let us look at the test application process so you develop a compact test set that compact test set is stored on an automatic test equipment which is pretty expensive device you apply that to your device under test collect the test response.

And then bring back to the automatic test equipment, automatic test equipment also stores the correct response of the chip and then it compares with the collected response if there is a match the device is good if there is a mismatch device is bad and then you have to throw that device out, so this is the basic test methodology now as I said that we will discuss in this lecture how we can generate this compact test set that can be applied in reasonable time.

So now look at what we want say I have circuit under test, this circuit under test has say n inputs x1 to xn, so now here if I apply this input x1 to xn the output say it has assume it has single

output then it will evaluate as this function as f of x1, x2, xn when there is no fault, so if say there is a fault or say there is a defect due to that defect the function may not evaluate same as this f so say this evaluates under the defective condition when I apply same input set x1, x2, xn the function will evaluate to f alpha.

So now the question is if this function which is defect free and the evaluation of function which is defective if both are same in that case here this input set cannot detect the defect but if both of these are different in that case you can detect that device has defect, so now our aim is to find out a set of or input sequence that can detect all the defects which may possible in the circuit under test right.

So now and here the size of the sequence should be as small as possible and keep it in mind we want to detect all the defect here I am using a word fault I will explain little bit later why we use word fault? So for a while you can think of that fault and defects are synonym of each other but keep in mind that we have interested only knowing whether a device under test is faulty or defective or its defect free.

We are not interested in diagnosing that what kind of defect or what kind of fault it has, so it is like here go no go test so it will tell us whether device is defective or device is defect free, so as I said that I we want to develop a test sequence that can detect all the defect and one of the prime requirement is input x1 to xn can be the test for a given defect or given fault if it produces different output when there is a fault and without fault so that means here the function should evaluate in different way that is the mandatory requirement.

(Refer Slide Time: 07:51)

Test Basics

For an n input circuit, there are 2^n input combinations.

Ideally we must test for all possible faulty functions.

This will require an exhaustive test with 2^n inputs

x1	x2	x3	f
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Since we cannot apply the exhaustive test set our best bet is to target likely faults!

Now if you look at if I have a circuit with three inputs in order to test that exhaustively what I need I exhaustively apply all the possible inputs and check whether I am getting the correct output which I am supposed to get, if but here as we discussed earlier that this exhaustive test set is not applicable. Hence here so since we cannot apply exhaustive test set our best bet is to target the likely faults or likely defects which may occur during your manufacturing process right.

So this brings us to at point, so now here if I say that here I have to target few likely faults or defects.

(Refer Slide Time: 08:45)

Test Basics

Defects Faults and Errors

A **Defect** is a physical flaw in the device, i.e. a shorted transistor or an open interconnect

A **Fault** is the logic level manifestation of the Defect, i.e. a line permanently stuck at a low logic level

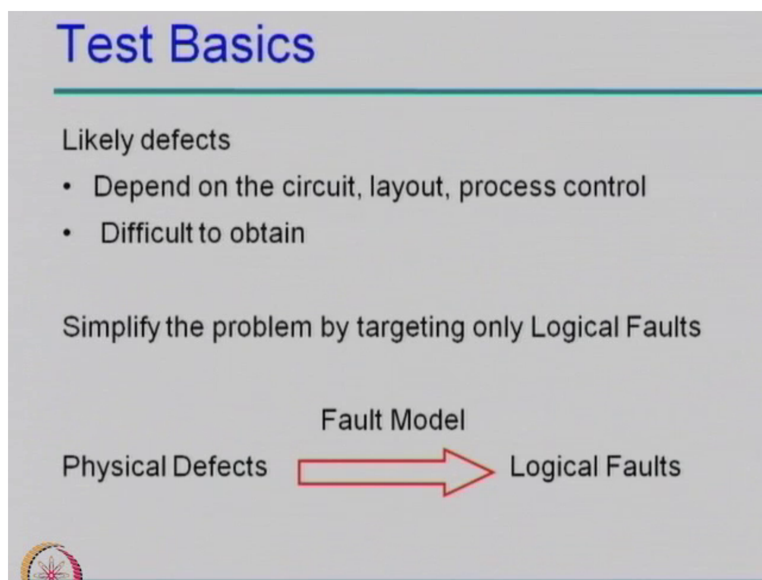
An **Error** occurs when a fault causes an incorrect logic value at a functional output

Let us define the basic terminology we use three different terms defects, faults and errors. Defects are the physical flaw in the device which we discussed earlier that what are the various reasons to get defect while you are manufacturing a circuit. Faults are the logical manifestation of these defects so that means here due to defect line may look like permanently stuck to logic 0 or permanently stuck to logic 1.

So that means here the effect is such that you see the effect like one line is connected to either power line or ground line and now here when there is a fault it causes incorrect logical operation and then you will get the different functional output, so now in the presence of fault you will get an erroneous output, so now the so here defect is physical phenomena, fault is logical manifestation.

So now this is at the device level if we look at little bit higher level of abstraction then we will see the fault, so now here so that means if you want to generate or develop a very compactive test set we have to go one abstraction higher one level of abstraction higher and so that means here rather than targeting defect I should target faults.

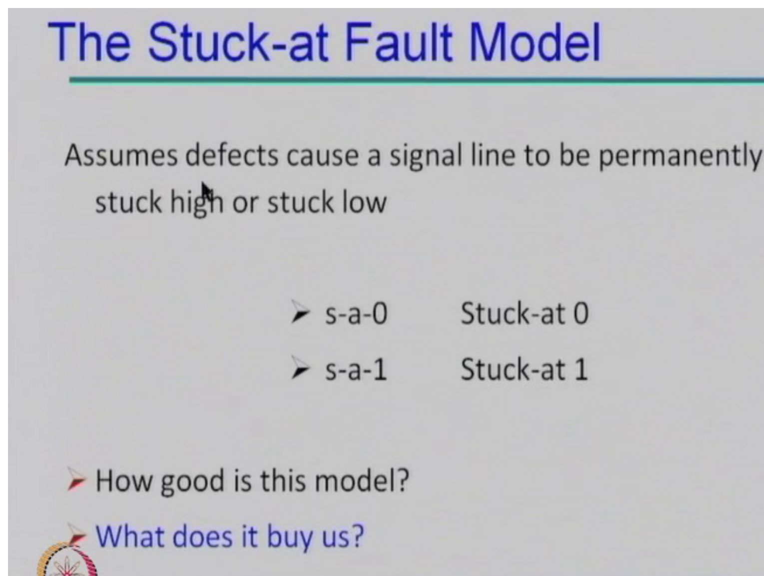
(Refer Slide Time: 10:29)



So at that is so that means here we are modelling this defects in terms of their manifestation as logical faults, so now here likely defects depend on the circuit layout or process control and all

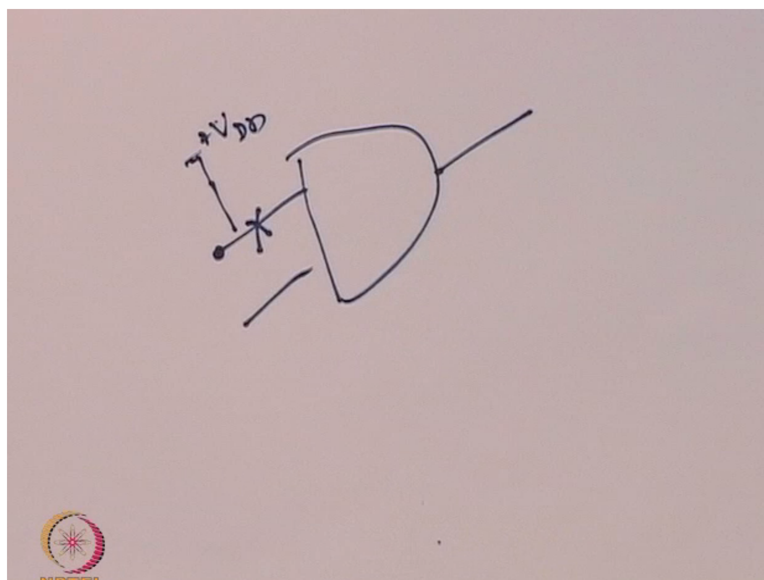
these likely defects are very difficult to obtain, so as I discussed then here we go one abstraction higher and model these physical defects as logical faults and that is known as fault mode.

(Refer Slide Time: 11:04)



So now here the what kind of fault model we can have let us say the very simple fault model that was conceived way back about 5 decades ago what is this assume is that here defects cause a single line to be either permanently stuck to logic high or logic low and that we call as stuck-at 0 fault or stuck-at 1 fault that means here we assume that any line in a circuit like for example if I have this AND gate.

(Refer Slide Time: 11:37)

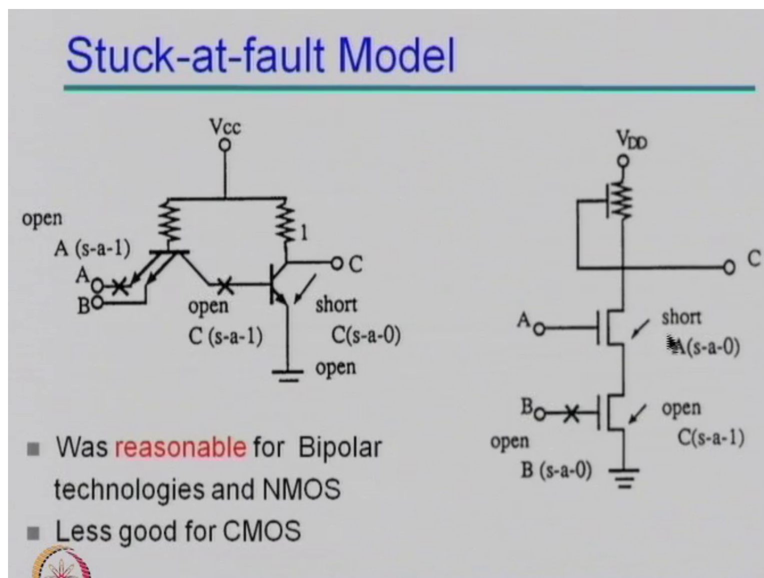


So it has three lines now here any line this line can treated as or due to defect this can permanently connected to or logic 0 that means ground or permanently connect to your power supply right, so that is the very simple model we can use, now here there are two questions first thing is how good this model is and second is what does this buy us. So first let us look at how good this model is.

So let us say you have a TTL logic, so you have a transistors now here the possible defect maybe this transistor shorted if this transistor shorts then now here what can be the logical manifestation of this defect that this transistor shorted, then of course if this shorts in that case here this node will always see the ground potential right, so that ground potential means here irrespective of what you apply at the input signal lines always here you will see 0 or ground potential at lines C.

Hence here I can say that this physical defect transistor is shorted we can model as node C or this output line can stuck to logic 0, in the same way here if I say that this input is open if this input is open in that case here this transistor will not conduct and then here the fall-out effect of that is you will see nodes C is always at V CC.

(Refer Slide Time: 13:54)



And hence here that you can say that that node C is stuck to logic 1 this is the case when in the TTL Technology. Look at the most technology say in NMOS say first look at if this transistor is shorted first look at if this transistor is open, if this transistor is open in that case here always you

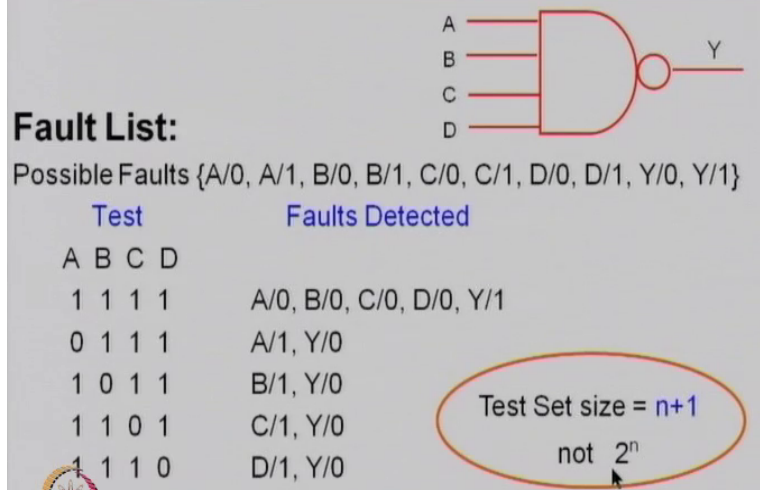
will see this point or node C is always at V DD hence you this defect you can model as node C stuck to logic 1.

And if this transistor is shorted in that case here the effect of that would be like here because this is shorted so that means this will look like the node A is always grounded right, if it is always grounded then you will see the transistor is always conducting and hence that is same as your transistor short, so now here this because the actual defect is this transistor is short but now here I can model that as stuck-at 0 fault here at logic at node A.

So now here if you see that there are couple of defects which can be model as same fault okay, so now here so now this gives you a big picture that here various defects can be modelled as logical faults, now good thing about the logical fault is that we are assuming that every line can stuck to logic 0 or logic 1 by and large most of the defects in Bipolar technology and NMOS Technology we can modelled as stuck-at faults.

This so this works reasonably well for by Bipolar and NMOS technology it is less good for CMOS because couple of defects cannot modelled as stuck-at 0 or stuck-at 1 fault, then here we have to supplement that model with another fault model that we have to use while we are testing, so this gives you a sense that how good this model is whether we can model various defects as fault or not, now the question is if I model these defects as logical faults then what it buys us.

(Refer Slide Time: 16:32)

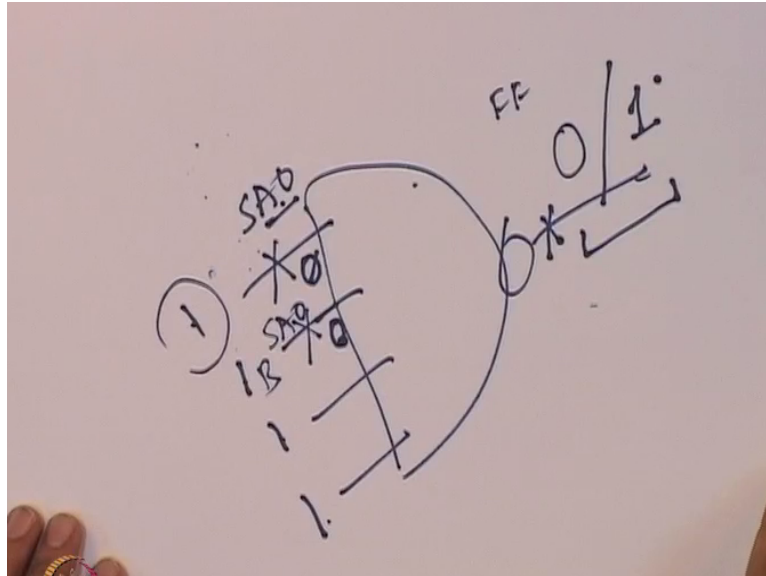


So now here let us look at a 4 input NAND gate it has A, B, C, D as input and Y as output, now as we say that here we can model the defect as a logical fault and logical fault assuming that any line can either stuck to logic 0 or stuck to logic 1, so now here and keep in mind that here we are assuming single stuck-at fault that means here there can be only one fault in the circuit we will discuss whether this assumption is reasonable or not in a minute.

But I assume that here there can be only one fault at a time in the circuit now how many faults can be there, so now we have 5 lines so there can be 10 faults because every line can stuck to logic 0 or stuck to logic 1, so now here like line A can stuck to logic 0, line A can stuck to logic 1, line B can stuck to logic 0, line B can stuck-at to logic 1, line Y can stuck to logic 0, line Y can stuck to logic 1 right.

Now the question is how I test these 10 different faults now here let us say I apply a test as some input, let us say I apply 1 1 1 1 as input A, B, C, D what it can test if I apply 1 1 1 1 then.

(Refer Slide Time: 18:09)



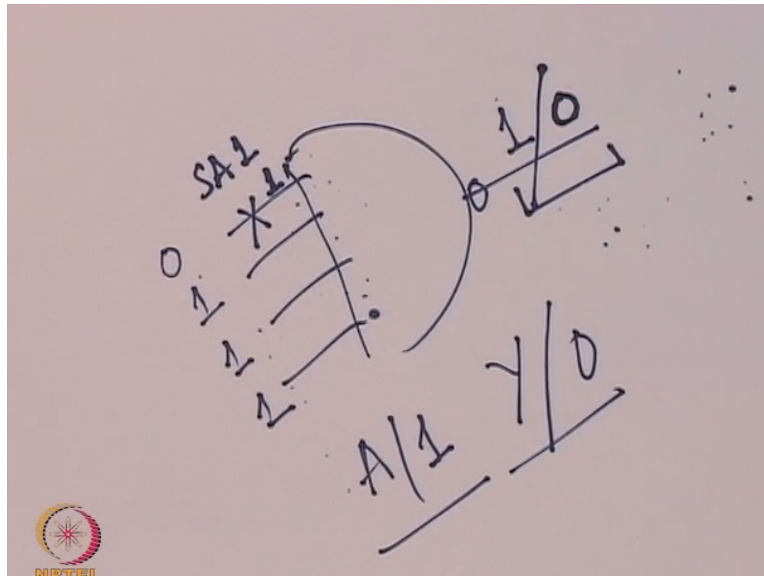
If in this circuit if I apply 1 1 1 1 here and say I have stuck-at 0 fault here, so if there is no fault in that case the output would be 1 right, if this fault is there what will happen irrespective of what I apply here this point is always at logic 0 and by application of this 1 1 1 1 I will get output as 1, right so now if you look at the output of the circuit for the fault free circuit I will get output 0 for faulty circuit I will get output 1.

And these are distinguishable outputs hence I can say that this input can detect this fault stuck-at 0 right, so now this way here we can say that this will detect stuck-at 0 fault to, stuck-at 0 fault at line A, now what about the stuck-at 0 fault at line B this is B right, if I again here if there is no fault in the circuit my fault free behaviour is 0, now if there is a fault here and we are assuming that there can be single stuck-at fault.

So now this node would be this point would be permanently connected to ground hence here the fault output of faulty circuit would give you output 1, hence here again this is also distinguishable so that means here we can also detect B is stuck 20 this way if you work out we can also detect C is stuck to logic 0, D is stuck to logic 0 and Y is stuck to logic 1, because here your fault free behaviour is 0.

And then if this point is stuck to logic 1 then always it will give you output 0 right, so it is a distinguishable fault free and faulty behaviour, so this vector can detect these 5 different faults, now if I have 10 different faults.

(Refer Slide Time: 20:43)



If I apply a vector 0 1 1 1, now what will happen if I apply 0 1 1 1 what it can detect, so if I apply 0 1 1 1 my fault free behaviour is 1 right, what it can detect if there is a stuck-at 1 fault here so under faulty condition this input will permanently tie to V DD or logic 1 and hence the behaviour of faulty circuit will give you output 0, hence again this is distinguishable faulty and fault free output.

So this can detect A is stuck to logic 1 this can also detect Y is stuck to logic 0 because here your fault free behaviour is 1 so if Y is stuck to logic 0 in that case output would always be 0 hence you can distinguish the fault free and faulty behaviour so you can also detect Y to logic 0, so this can this vector can detect two more faults, then in this way if you apply 1 0 1 1 this can detect B is stuck to logic 1, Y is stuck to logic 0.

If you apply 1 1 0 1 then this can detect C is stuck to logic 1, Y is stuck to logic 0, if you apply 1 1 1 0 you can detect D is suck to logic 1 and Y is stuck to logic 0 right, so now here if you look at this how and this cover your entire fault list these are the modelled fault so that means here if I

can detect all these 10 different faults I can say that my circuit is fault free because I covered all modelled faults.

Now how many test vectors we need, we need only 5 vectors if you apply exhaustive test set in that case you need 2 raise to the power 4 that means 16 test vectors, now there is a use reduction you have reduced all most one third, now so and this is sufficient test set to test all the outputs in this circuit, so now here if you look at more closely what it brings you on your plate see here your test size will reduce to n + 1 for 2 raise to the power n.

Because here you have 1, 2, 3, 4, 4 lines for input lines and so now here and you need 5 test right, now here that number is 4 + 1 is 5 you do not need 2 raise to the power n right, so that this is big thing you are converting a problem from exponential complexity to linear complexity and we discussed that here this exhaustive set we cannot apply but here anything which is linear we can apply.

So that means here if you have a circuit with say n inputs or that maybe 100 in that case if I can convert that problem in to n + 1 it is sufficient for me so that 101 test I need to apply, this so in this case this fits well if you have a circuit which is fan out free and single output then again you can say that here you can you need a test size of n + 1 test. If there are fan-out's in that here number will slightly increase by the number of fan-out's you have in the circuit, but for fan out free circuit again this relationship faults good.

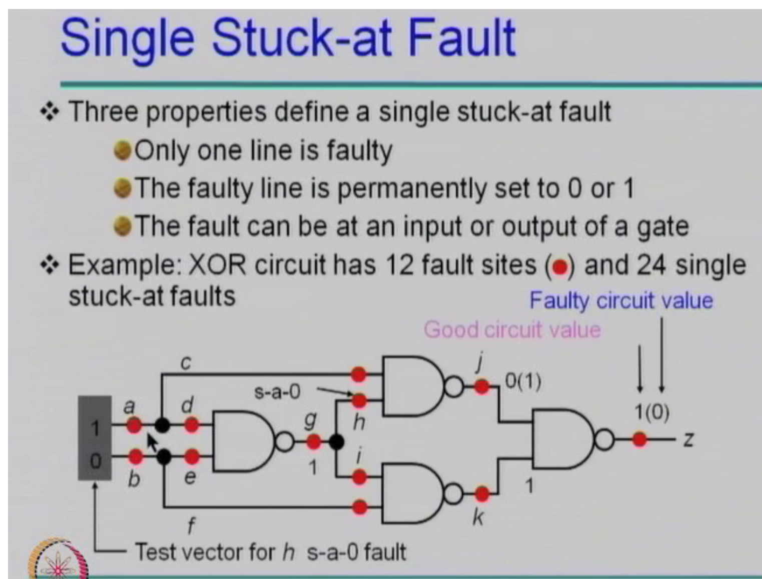
Hence, here we can say and if there are fan-out's in that case here again this will increase by the number of fan-out's. And so means the big thing this brings into our plate is you are converting exponential complexity problem into a linear complexity problem and that is big thing and that is why here we do not target defects we target faults which are logical manifestation of the defects. As I mentioned in last lecture that defect oriented test is still open problem.

And some of you may provide some good solution which can advanced state of the art, okay. So now here we are converting this exponential problem into linear problem, one thing again to notice is that this if you look at closely one test can detect multiple faults or one fault can be

detected by multiple test, so there is no uniqueness relationship between test vector and faults, so this gives us opportunity to reduce the number of test.

Now here our objective is to find out smallest test set that can cover all the faults. So now here if say this vector can detect Y0 in that case I can choose that then I do not need to target Y0 again, if this vector can detect 5 faults in that case here these 5 faults again I do not need to test, so now here there is non-uniqueness relationship between the faults and test vector.

(Refer Slide Time: 27:02)

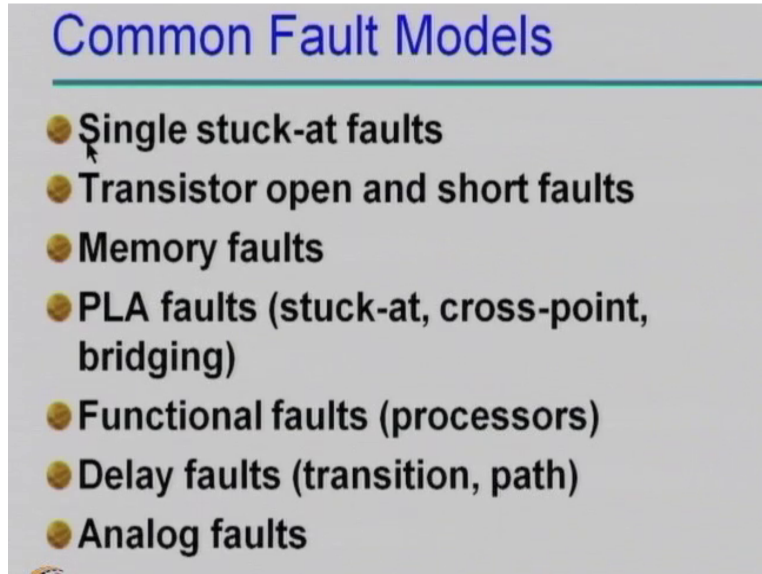


So now here say look at this circuit this is very popular circuit all of you know this NAND-NAND implementation of XNOR gate, so this fault this circuit has 12 different sites now here another very important think I would like to bring to your notice is that this fault at stem of fan-out point is different from fault at the fan-out branches, because here they may have different manifestation this output though here this output are re-converging.

But here this may possible that they will not re-converge this output may go to some primary output and this fan-out branch can take you to some different primary output, so now they have different manifestation so in terms of logical error, hence here we treat the fault at the input of a fan-out point and output of fan-out point differently, so now here we have total 12 different sites hence there can be 24 different faults.

Now means working in the same direction earlier we discussed that here if there is a stuck-at 0 fault at line h and if I apply input as say 1 0 in that case will it detect the this stuck-at 0 fault it will because your fault good circuit value would be 1 and faulty circuit value would be 0 and hence this can detect this particular fault.

(Refer Slide Time: 29:03)



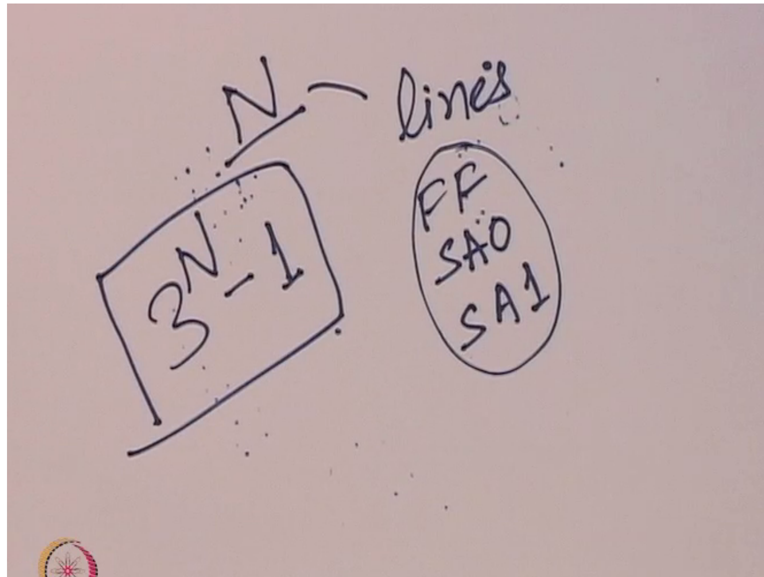
So now means here we by modelling these faults as a single stuck-at fault we can generate a compact test set and we have seen that we can reduce the problem from exponential problem to the linear complexity problem, so the first fault model that was given way back about 5 decades ago is this single stuck-at fault model that assume that only one net can stuck to logic 0 or logic 1 at a time.

And so now here the number of faults that you can have is twice the number of nets or number of lines that you have in your circuit that is linear and that is why we are converting the problem in linear complexity problem, now we can argue that how reasonable this assumption is do you think that always we can have only one fault in the circuit or one defect in the circuit, though multiple defect can manifest at the same for as well.

So that may not be reasonable so now what is reasonable, reasonable is that there can be a multiple defects, multiple defect may result into multiple faults now you need to target multiple

stuck-at fault rather than single stack-at fault, now if you look at the number of multiple stuck-at fault.

(Refer Slide Time: 30:39)



If assume that there are N number of single stuck-at faults or in the circuit or N number of line or N number of lines or nets you have in the circuit out of this N line now here any line can give you may be the fault free so that maybe fault free or that may stuck to logic 0, that may up to logic 1, so any line can be in 3 states fault free, stuck to logic 0, stuck to logic 1.

Now if there are N lines how many total permutations can be there in the circuit that would be 3 raise to the power N, the output of this 3 raise to the power N instances only 1 instance would be fault free when all the lines are all free all other instances are faulty instances so now here the number of faulty instances would be 3 raise to the power N – 1, so that means that there can be 3 raise to the power N - 1 multiple stuck-at fault in this circuit.

Again here this problems, this problem converts to the exponential problem and all these 3 raise to the power N - 1 fault may not be targeted but people observed that if you because of non-unique relationship between your fault and test vector you are likely to detect large number of multiple stuck-at faults, so in other word if I say test all or 100% single stuck-at fault I am likely to detect 80 85% multiple stuck-at faults using the same test pattern that gives us big relief.

And so now this is the standard practice in the standard in the industry that they are using single stuck-at fault model and they that can cover large number of multiple stuck-at fault, now the so this is one of the very popular and common fault model industry uses, as I said that this single stuck-at fault model can model most of the defects in TTL, NMOS and large number of defects in CMOS.

There are some faults which are like transistor open and transistor short so transistors stuck open, stuck short fault which are really not modelled by single-at fault in CMOS logic so that means here we have to test for those faults as well and then here another fault model was given for those kind of faults which are not covered by single stuck-at faults or the transistor stuck open and stuck short faults.

Then other kind of circuit that we have in practice are memory now if you look at memory, memory does a very specific operation or job what it does is it stores value what you want so logic 0 or logic 1 it retains that value until you would like to rewrite it right, so what you want is that if you would like to write 0 it should write allow you to write 0, if you write 1 it should allow you to write 1 and it should retain that value until you rewrite that.

So this because of this does or this performs a limited functionality so rather than going for stuck-at kind of fault which is very appropriate for random logic we go to functional fault model and then here we model those faults as like whether memory is able to write value 0 or not whether memory is able to write value 0 to a cell or not and whether it is able to retain that value or not so these are some of the functional fault model that we use.

And this reduces the complexity tremendously and hence this functional fault models are being used for memory, other kind of circuit that we may have in practice are the Programmable Logic Arrays in that means other than stuck-at fault there are some other faults which may appear which are unique to clearly those are like here there can be a cross connection between the two lines so cross point or bridging faults.

The circuits like processors perform many operations but still it is the set is limited like it a processor is supposed to execute a couple of instructions those are finite in number and the behaviour of each and every instructions is well defined so now here what we can do is rather than going for stuck-at fault model we may target the functional fault model that means here how instructions are being executed whether it execute that correctly or not.

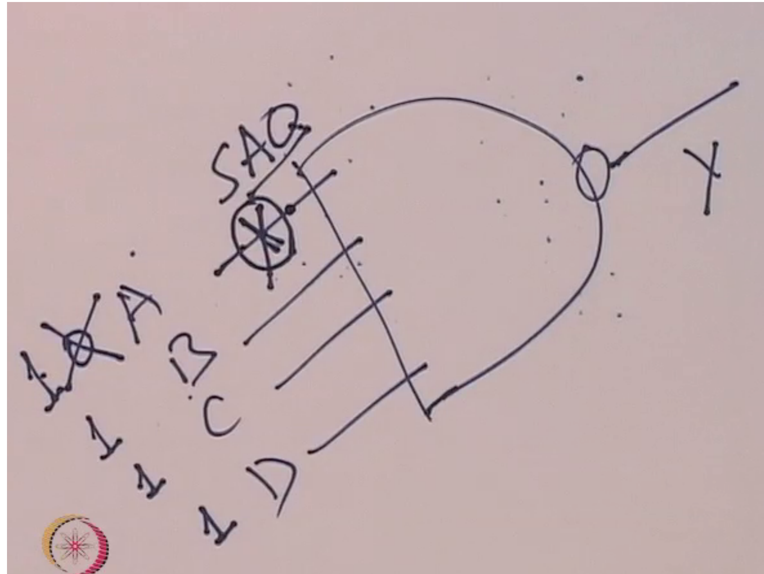
There are some faults which are like which does not lead to the malfunctioning of the circuit if you do not care about the frequency of operation but they result into malfunctioning of the circuit if you test at certain frequency that means here if you operate your circuit at slower frequency your circuit may work perfectly fine but if you operate at the designed frequency your circuit may not be operate at that frequency.

And there are couple of fault model which were developed to test for such kind of performance oriented fault and those are popular model or transition delay fault model and path delay fault model, other class of circuit that we have in practice is analog circuit unfortunately till today we do not have any fault model like single stuck-at fault model which was given for digital circuits.

By and large the analog circuits are tested against specifications though there were attempts and then people develop some alternate test methods that in that by observing some of the derived parameters of the circuit and which reduces complexity significantly, so these are various common fault model we use, single stuck-at fault model is pretty common so most of the industry do test their circuit for all single stuck-at fault and delay faults.

And of course analog circuits are tested for the analog faults and memory is always tested for the using functional fault model, okay. Now here let us concentrate on the single stuck-at fault model which is pretty common and all these circuits are tested for single stuck-at fault, now the question is how I should generate test for each and every fault, each and every single stuck-at fault in the circuit.

(Refer Slide Time: 39:28)



So if you have any line in the circuit or n net in the circuit you may have 2 n number of faults now here the question is how I should generate that test, so say you have this circuit which we discussed earlier a 4 input NAND gate A, B, C, D are the input, if I say I have stuck-at 0 fault here now what you means how you can detect that if as we have discussed that if I apply 1 1 1 1 as input here always I will get faulty and distinguishable faulty and fault free behaviour at the output.

And hence you can detect that fault but now I have just reverse problem of the earlier one, now I have fault at my hand and I and I want to find out a test vector that can detect the particular fault what I need that my basic requirement is that whatever input test input I apply that should produce distinguishable faulty and fault free output.

So now here this net is connect is stuck to logic 0 will it ever produce distinguishable output if I apply 0 here it will never produce irrespective of what I apply at B, C and D hence this is ruled out, now what I should apply here if I apply 1 here in that case there is a possibility that it may produce distinguishable fault free and faulty output depending on what I apply at B, C and D.

If I apply B as 0 in that case it will never produce distinguishable fault free and faulty behaviour right, in both of the cases it will produce output as what would be that output that output would be so I apply 1 0 1 1 and output will be 0 right, in both of the cases it will be 0 faulty and fault

free behaviour, so now what I want that the line which is stuck to some logic value 0 or 1 I should have at least inverse or opposite value to that.

So now if it is stuck to logic 0 I have to have 1, if it is stuck to logic 1 then I have to have 0 right, and as I said that this main potentially propagate or may produce faulty and distinguishable faulty and fault free behaviour at the output provided that I assign the another input or side input to this gate as non-controlling value, so what is the controlling value of a gate the value which always determine the output.

So if I apply 0 value to any of the input output would be determined by that, output would always be 1 right irrespective of other values, so now I have to apply non-controlling value that what does that mean that if I apply non-controlling value to other inputs that means here output would be determined by this input only right, so if I apply opposite value then here output would be would be distinguishable.

So that means here I have to have 1 1 1 so what it can tell you that if I want to detect a fault here what should be my test vector, first I should have different value or opposite value to the faulty value at that faulty site and other input must be at the non-controlling value right, this is the basic criteria based on that we have to generate test for each and every fault. Now there are two mechanisms one is the Algebraic way.

Algebraic way is based on the Boolean Algebra so if you look at Boolean algebraic manipulation say if you have circuit with n input x1 to xn then output of that circuit would be f would be given as f of x1 to xn right, this may be x1 x2 + x2 x3 + x1 x2 x3 + x3 xn something like that, now always I can decompose this output or this function in sub functions with respect to an input say if I guess all of you know the Shannon’s expansion theorem.

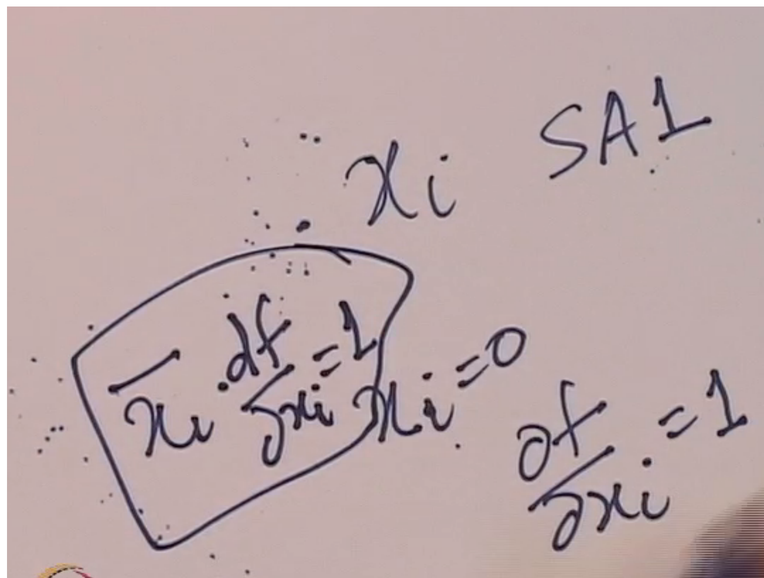
What it says that if I want to decompose this function with respect to x1 that I can write that x1 dot f of x1 x2 x3 xn so that means here this function is the output of the circuit when x1 = 1 + x1 can be 0 rights, so x1 bar dot f of 0 x2 to xn right. So that means here this function would be

output of the circuit when x1 is 0 right, so that this way I can decompose a circuit, what I want is that if there is a I want to have distinguishable faulty and fault free behaviour.

Now depending on your fault location and nature of fault in order to test as I said that we have to apply the opposite value to the faulty value at the fault site right, so now if fault is stuck to 0 in that case here the faulty behaviour would be given by this one right, and in order to test that here I have to apply the opposite value that means I have to apply 1 right, then your fault free behaviour should be we given by this one right.

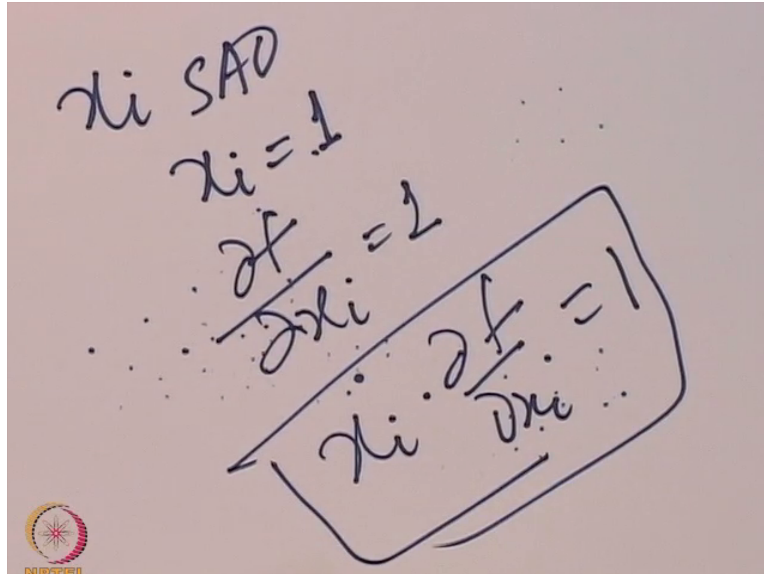
So now here what I want this should be distinguishable that means here the XOR of this function and this function should evaluate to 1 and that XOR operation is defined as Boolean difference, so that means what I want is that here this Boolean difference must evaluate to 1 and I have to apply opposite value of the faulty value at the fault site.

(Refer Slide Time: 46:20)



So now here if I say that line say xi is stuck to logic 1 in that case here so if xi stuck to logic say 1 in that case here I have to have opposite value right, that means here xi should be 0 and del f upon del xi this would always also evaluate to 1 what I can say is that here your xi bar dot del f upon del xi should evaluate to 1 this is the condition to detect xi stuck-at 1.

(Refer Slide Time: 46:56)



If xi stuck-at 0 in that case what I want that xi should put to value 1 and you should produce the distinguishable faulty and fault free output so that means here del f upon del xi must be 1 so what I expect is xi dot del f upon del xi this should evaluate to 1 this gives you the condition to generate test vector algebraically.

(Refer Slide Time: 47:26)

Fault Detection

- $x_i\, df(x)/dx_i = 1$ for s-a-0 at x_i
- $x_i'\, df(x)/dx_i = 1$ for s-a-1 at x_i
- As an example, let us consider the function

x1, x2, g, x3, f

- $f(x) = x_1x_2 + x_3$
- Thus $df(x)/dx_2 = x_3 \oplus (x_1 + x_3) = x_3'x_1 = 1$. Then
- $x_1 = 1$ and $x_3 = 0$.
- For the SA1 and SA0 faults on x_2, the patterns are then $x_1x_2x_3$ = (100) and (110), respectively.

So let us look at example so these are the two conditions that I already discussed, now let us look at this circuit so the output f can be given as x1 x2 + x3 now you have to determine or detect stuck-at 0 fault at x2 so now what should be the condition first thing is x2 should be assigned with value you 1 if it is stuck to 0 and it should be assigned as value 0 if this net is stuck to logic 1.

Now you so and then here that fault effect should be propagated to the primary output what are the conditions that Boolean difference with respect to x2 should evaluate to 1 right, so that means here this x1 what is the Boolean difference with respect to x2 that is if you put x2 = 1 in this will evaluate as x1 + x3 right and if you place x2 = 0 in that case this will evaluate to x3 right.

So this would be x3 XOR with x1 + x3 that will give you x3 bar x1 this should evaluate to 1 when this can evaluate to 1 this can evaluate to 1 only when x1 must be 1 and x3 must be 1 this gives you the propagation criteria and so now if you want to detect generate a test for stuck-at x2 stuck-at at 1 in that case here x2 must be 0 right, so now here x1 must be 1, x2 must be 0 and x3 must be 0. Right?.

This gives you the detection condition for stuck-at 1 fault at line x2, now for stuck-at 0 fault x1 must be 1, x2 should be opposite value so that means here that should be 1 and output must be so x3 must be 0 this is the detection criteria, what if this XOR or this Boolean difference is 0 that means that your so circuit is not able to transfer the fault effect to the primary output what does that mean that this input or this signal is not contributing in making a decision right.

So that means this input is redundant input hence those faults are known as redundant faults, so now this methodology will either give you the test vector or it will say that this fault is redundant fault and now here you do not need to test for that because here anyway that fault may not have any implication in your functioning of your circuit hence even if that fault is there you are okay with that.

So this will give you the faulty so test vector or identified fault as redundant fault, hence I can say that this method is a complete method this can identify fault as a testable fault and give you the test vector and identify fault as untestable fault or redundant fault, so now we compute two different parameters 1 is known as Fault Coverage and other is known as Fault Efficiency.

(Refer Slide Time: 51:24)

Fault coverage is defined as the number of detectable faults divided by the total number of faults right, so say you have total 1000 faults and 994 are detectable but here 10 faults are not detectable in that case here your fault coverage would be something 990 / 1000 x100 that becomes your 99% why the because 10 fault are redundant they are coming from the redundancy in the circuit.

Sometimes you have redundancy because of ease of manufacturing or because ease of design like for example if you design a 4 bit adder you do not design that 4 bit adder with 3 full adder and 1 half adder generally we design with 4 full adders, so that means here one of the half adder is redundant in that design but because VLSI favors the irregularity and hence we prefer to have four full adders, and then so that redundancy due to that redundancy we get lower fault coverage.

(Refer Slide Time: 53:02)

We define another term that our metric to evaluate how good our test is that is referred as fault efficiency, fault efficiency is defined as total number of detectable faults divided by total number of faults minus total number of redundant faults those you identify as redundant fault, so like here now in this example we say that 990 faults are deductible and 10 faults are identified as redundant fault.

So that means here this is 1000 - 10 x100 this gives you the 100% fault coverage so that means here we are able to detect all the faults which are potentially which may pose malfunctioning of the circuit and this Boolean difference method or algebraic method can give you always guaranteed 100% fault efficiency, now why I defined this two metric why not only 1 metric what additional information my another metric gives you fault.

The difference between the fault coverage and fault efficiency gives you how much redundancy you have in the circuit, so now we identify the kind of redundancy, so now in the circuit and then if it exceeds beyond the limits certain kind of redundancy is admissible to us say maybe 10% redundancy in the circuit may be admissible, so now if redundancy is below 10% it is, okay.

Otherwise test engineer has to report this information to the design engineer say that look your redundancy in the circuit is 30% are you okay with that if he is not okay that he has to resynthesize his circuit and bring down the redundancy in the circuit, so again this is the another

task of test engineer to identify redundancy in the circuit and report it back to the design engineer if it exceeds beyond the certain limit or beyond the tolerance limit.

As I said that here due to certain reason like here means in order to minimize the propagation time or delay of the circuit or if it results into regular structure here we deliberately introduce redundancy sometimes in the circuit, so this is the algebraic way to generate test set, now the problem here is always we need to do Boolean manipulation and Boolean manipulation is not feasible for fairly large design which has say several hundred million gates.

So this approach is not scalable in order to have scalability here we have to go for algorithmic methods and some of the algorithmic methods were proposed in the literature which are based on the path sensitization algorithm those where D-Algorithm which was first complete algorithm given by Roth from IBM in 1966, then there were improvement on that method and the another algorithm which is known as PODEM and that was given by Goel from IBM again in the 1981.

Then and this algorithm PODEM was almost hundred times faster than the D and then the another faster variant was developed by Professor Fujiwara from Osaka University at that time that was known as FAN in 1984, so these are the three basic algorithms and most of the today's CAD tools are largely based on either PODEM or FAN algorithm.

Then there were several improvements in which they tried to support these algorithms by learning from the circuit and those are two of the popular algorithms are SOCRATES and SPIRIT all these algorithms are based on path sensitization, so this path sensitization based algorithmic method to generate test we will cover in the next lecture, I with this I complete this lecture here. Thank you very much for patience full listening.