INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

NPTEL
NPTEL ONLINE CERTIFICATION COURSE
An Initiative of MHRD

VLSI Design, Verification & Test

Dr. Santosh Biswas
Department of CSE
IIT Guwahati

Module VIII: Fault Simulation and Testability Measures

Lecture III: Fault Simulation – 3

So today will be covering the third lecture on fault stimulation.

(Refer Slide Time: 00:30)

Design Verification and Test of
Digital VLSI Circuits
NPTEL Video Course

Module-VIII
Lecture-I,II and III
Fault Simulation

So what we have discussed in the last two lecture that for finding out to find out the test patterns for second fault there can be two basic approaches in one approach we sensitize the fault propagate the affect and then justify the lies which is called the sensitization propagation and justification then we are find out that these actually the algorithm is not a very simple algorithm because you can see that first you have to sensitize then you have this propagate and then you to justify I need to repeat for all the second fault and sometimes they may be requirement of utilizations because sometimes a propagation are your what you may call the justification may not be successful.

So also this is a fact that one pattern can take more than our number of faults. So if you are I mean doing sensitize propagate and justify for different fault so what may happen is that this you may be landing up designing the same pattern for detecting multiple faults. She is not a problem but the thing is that the same algorithm or the same pattern you are generating by running the sensitize propagate and justify approach multiple number of time which is actually hampering your efficiency, then we have seen that other approach which is called the random fault simulation based test pattern there is not approach in which the idea is you take the random pattern and then you find out how many faults get detected with their approach.

So define that there some K number of faults are detected by that, then you drop those K faults and then you take other random pattern and keep on doing it till you find out that I mean a new pattern is not able to detect and sufficiently less number of faults. So the idea is better than the previous approach because you can see that one pattern detecting K number of faults so that can be found out by random faults stimulation can followed by fault stimulation in one go so you did not repeat sensitize propagate and justify for the K faults.

So that we can do these only for the easy to test faults approach each the process saturates and then we have to obviously go back to sensitize propagate and justify approach and the what we have seen that we seen different kinds of fault stimulation examples so what is the basic idea of fault stimulation was that we have a normal circuit and they have a fault circuit and then we find out whether a pattern creates behavioral difference at the output with the fault. So we have seen very simple serial fault stimulation.

(Refer Slide Time: 02:35)

Design Verification and Test of Digital VLSI Circuits NPTEL Video Course

Module-VIII

Lecture-I,II and III

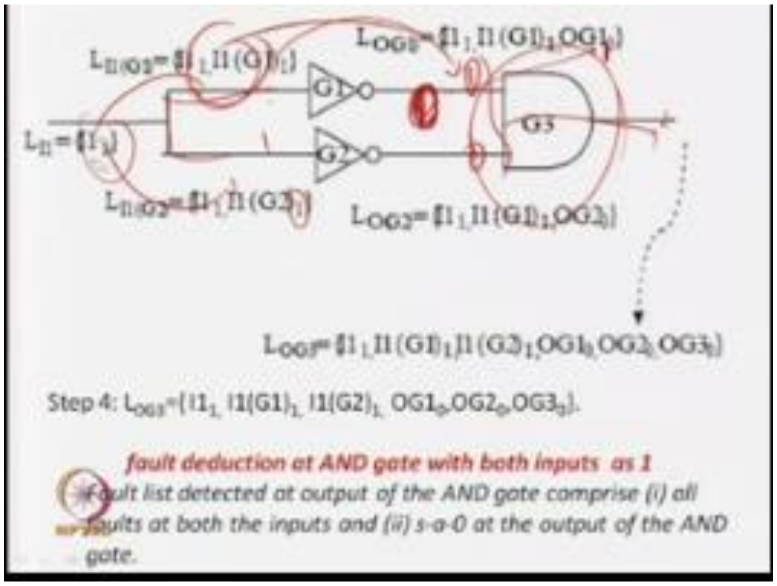
Fault Simulation

Which was the very simple one in each case what we do we take up fault which they take a pattern and then find out whether one fault is detectable or not and then if it detectable it is what you call your written in job the fault and the try with another fault and keep on doing it then you apply another random pattern and so for but actually the idea here was that it was very slow because we are going for serial based approach one pattern one fault then next fault and next fault and so on till you cover all the faults and the process reviews for the next day pattern.

So you have seen that this is actually bit slow then when we have gone for parallel fault stimulation in which case for each line we have a array which represents some end bits or which is depending on the parallelism of your computer you can have end bit array or a two end bit array or something like that one bit represents the normal circuit and all other bits of the array represents one faults faulty case each and then we can apply the random pattern and then we can find out how many faults can be detected by the random pattern and if your window size or your array size is an then we can reduce the listening about n-1 fault.

Because one bit is for the normal circuit all that we have discussed so that parallelism I mean analyses your algorithm by a factor of n-1 if n is the your window size or your array size then we have seen that in this case what is the idea that if your computer bit processing or what you call the parallelism of the bouncer is low or maybe that is 32 bit and also if there is some 1000 faults so at least you have to repeat it 1000/32 my ceiling factor that number of times if you repeat the what you call this a parallel fault stimulation.

(Refer Slide Time: 04:10)


L_I1OG1={I1_1 I1(G1)_1}
L_OG1={I1_1 I1(G1)_1 OG1_0}
L_I1={I1_1}
G1
G2
G3
L_I1OG2={I1_1 I1(G2)_1}
L_OG2={I1_1 I1(G2)_1 OG2_0}
L_OG3={I1_1 I1(G1)_1 I1(G2)_1 OG1_0 OG2_0 OG3_0}

Step 4: L_{OG3}={I1$_1$, I1(G1)$_1$, I1(G2)$_1$, OG1$_0$,OG2$_0$,OG3$_0$}.

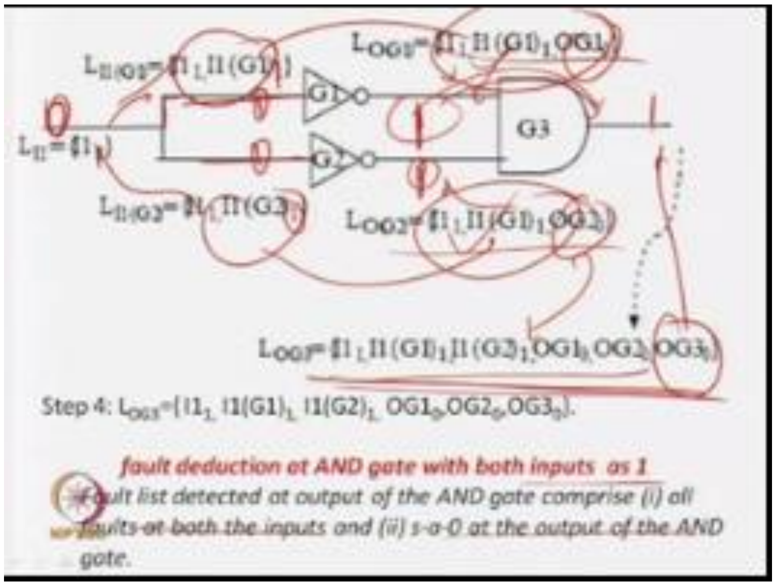
fault deduction at AND gate with both inputs as 1

fault list detected at output of the AND gate comprise (i) all faults at both the inputs and (ii) s-a-0 at the output of the AND gate.

So then we found out then we started discussing about another algorithm in which case can we do the whole thing in one go that is we apply one random pattern and in one scan of the circuit can we deduce the results about all the faults that can be detected by this then we saw the detecting fault stimulation was a very good approach for that in which case we apply one random pattern and then at each net of your circuit this one this one this one this one we find out what are the fault or we find out the faulty that is detectable by that random pattern.

And then actually you can in one go find out which are the patterns detectable by that fault then we started finding out some rules like for example if it is a naught gate then what ever is the fault list here and propagate the value here and then if the value here is the zero then we have to also add the value here is the one then have to add a stock at one at this net and if the here the input signal value is zero then you have to apply a socket one fault here and the whole list on your propagates here then it is a find out array then whatever is here get propagated here and here in addition to if it is a one signal here there socket zero and vice versa and then similarly we have seen the rules for an AND gate like if the case is 11 then whatever fault list at both the nets will be added here plus some socket zero socket one in the output then if it is 00 then we are found out a only the common set of faults in between this can propagate to the output.

(Refer Slide Time: 05:34)



$L_{I1(G1)}=\{I1_1, I1(G1)_1\}$
$L_{OG1}=\{I1_1, I1(G1)_1, OG1_0\}$
0
$L_{I1}=\{I1_1\}$
G1
G2
G3
1
$L_{I1(G2)}=\{I1_1, I1(G2)_1\}$
$L_{OG2}=\{I1_1, I1(G1)_1, OG2_0\}$
$L_{OG3}=\{I1_1, I1(G1)_1, I1(G2)_1, OG1_0, OG2_0, OG3_0\}$

Step 4: $L_{OG3}=\{I1_1, I1(G1)_1, I1(G2)_1, OG1_0, OG2_0, OG3_0\}$.

fault deduction at AND gate with both inputs as 1

Fault list detected at output of the AND gate comprise (i) all faults at both the inputs and (ii) s-a-0 at the output of the AND gate.

So we are starting out some kind of rules in detecting fault stimulation which can basically to scan the circuit from input to the output you can find out or the faults can be detected but for that we require some rules so in the last lecture so we covered some rules like for example if it is find out net then whatever fault list here propagates here so you see this list here and this list here then your applying a random pattern zero over here.

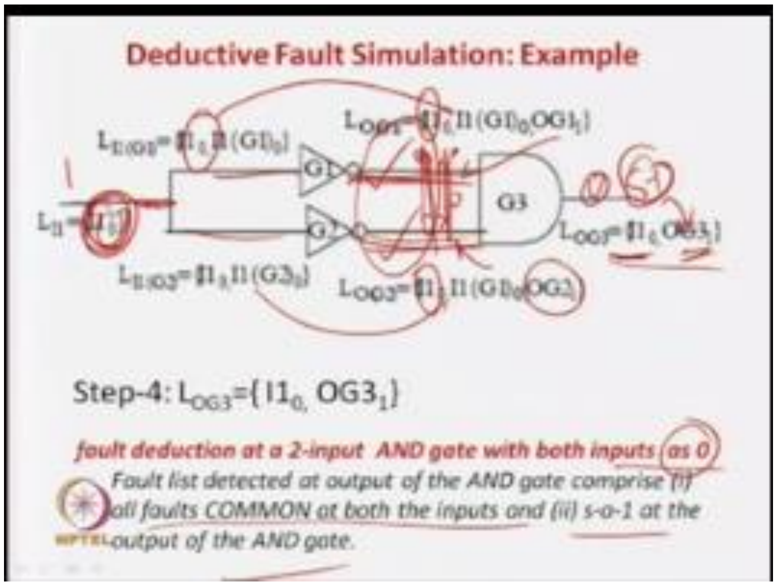
So the detects socket one fault here similarly it also and the value is 00 over here signal value we was applying a 0 so socket one is detected at this net and for the inverter what is the rule we saw at whatever here and whatever here both the list will be here now this the value is 00 here and it is 11 here so some socket 0 that is detected here and this the socket detected here.

Now you know that for the AND gate what you have seen the rule for the AND gate there will be the four rules okay for 0001 then you have for 10 then for 11 for all the four cases of your AND gate you have to have the rules so in this case 11 so you know that in the AND gate then both the inputs are 1 then if this is input is 1 then whatever fault affect will propagate similarly this is one here so whatever is the fault ever can be propagated because 1 is the wrong controlling input of an AND gate but if it is a 0 you know that nothing propagates the other end so in this case both the values are 11 signal value because the random in is 0.

So whatever the list over here and whatever the list over a whole thing you know that it will come over here so you can if you study this list and this list for the whole list will be actually taking care of this one and this one because they are applying 11 and 11 the signal value output is 1 so you can also detect a socket 0 for at this net so this becomes your what you called the pattern and in these are the fault list for this random pattern 0 you can detect this n number of fault.

So we find out that the rule for AND gate then both the inputs are 1 then all the faults are both the inputs of the input and socket 0 at the output of the AND gate this the rule for AND gate when everything is 1.

(Refer Slide Time: 07:44)



Deductive Fault Simulation: Example

$L_{I1}=\{I1_1\}$
$L_{I1(G1)}=\{I1_1, I1(G1)_0\}$
$L_{I1(G2)}=\{I1_1, I1(G2)_0\}$
$L_{OG1}=\{I1_0, I1(G1)_0, OG1_1\}$
$L_{OG2}=\{I1_0, I1(G1)_0, OG2_1\}$
G1
G2
G3
$L_{OG3}=\{I1_0, OG3_1\}$

Step-4: $L_{OG3}=\{I1_0, OG3_1\}$

fault deduction at a 2-input AND gate with both inputs as 0

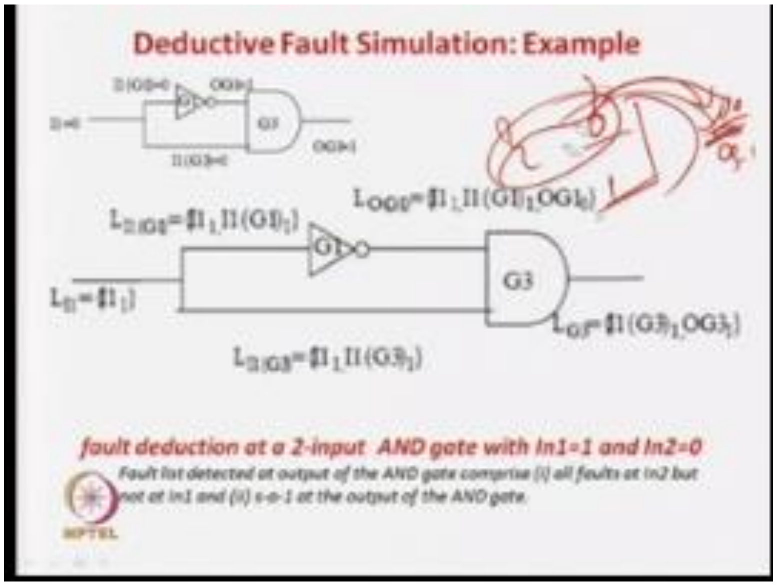
Fault list detected at output of the AND gate comprise (i) all faults COMMON at both the inputs and (ii) s-a-1 at the output of the AND gate.

Then we have taken the case when the output here was a 1 so you can detect socket 0 fault over here socket 0 fault at this net and then this will be 1 over here so you get 0 over here so the whole list comes here because of this inverter and then you can add a socket 1 fault at this net. Now also we are also discuss in the last class at both the AND gate inputs are 0 then if the AND gate input is 0 one input is 0 in AND gate we know that the fault cannot be propagated by the other end so what will be found out that only one fault which is common to both of them so this is landing up a socket 0 over here and socket 0 over here.

So that is the only fault which can be detected at the output that we have seen that only fault which is common to both of them you can be detected over her e and of course you get a 0 and 0 over here, so socket 1 fault at this net can be detected so both this the fault common that is socket 0 for at this line is affecting both this input and this input so this is common so that is the only thing will propagate here and other things will not be propagated. So the rule is when both the inputs are 0 then we thought that into the nothing could be propagated because we need only one gate of the input is 0 they nothing propagates to the AND gate.

But we were surprised to find out that in this case if some input is their affecting both the inputs or your AND gate then that is the only fault that is propagated. So fault list of the AND gate comprise then both of the AND gate inputs are 0 all faults which are common to both the inputs so only this fault was common to both the inputs and that propagates plus a socket 1 fault at the output is obviously detected because 0 and 0 here leads to 0 over here, these rules we have seen in the last class.

(Refer Slide Time: 09:27)



Deductive Fault Simulation: Example

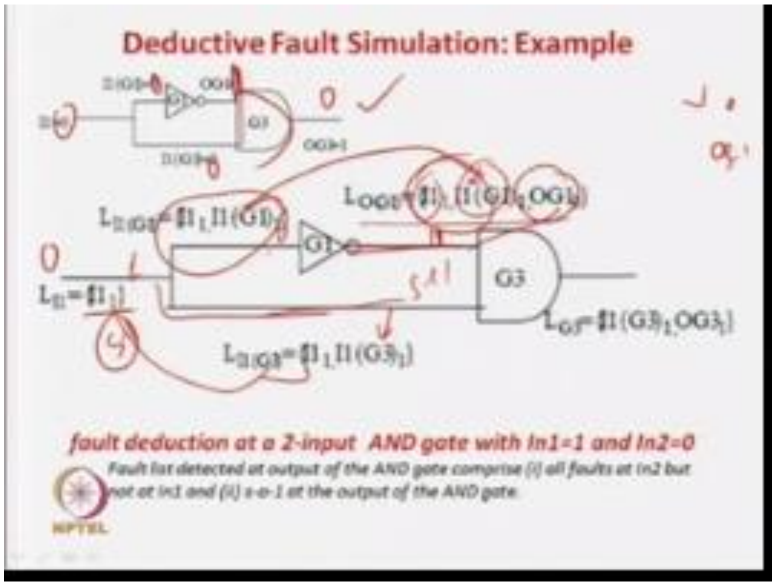
$L_{I1:G1}$={I1_1, I1(G1)_1}
L_{OG1}={I1_1, I1(G1)_1, OG1_1}
G1
G3
L_{I1}={I1_1}
$L_{I1:G3}$={I1_1, I1(G3)_1}
L_{G3}={I1(G3)_1, OG3_1}

fault deduction at a 2-input AND gate with In1=1 and In2=0

Fault list detected at output of the AND gate comprise (i) all faults at In2 but not at In1 and (ii) s-a-1 at the output of the AND gate.

Now there are two more rules which they need that is for an AND gate that is 10 AND gate 01 so we will see what are the rules for this thing okay. So I mean this would be this pretty obvious I mean in your class it should be obvious because you see whenever it is the 0 and an 1 so whatever in tuition says that will follow so obviously if this is one this whatever is here will propagate over here so whatever is the fault list here can be propagated here as well as the output is 0 so obviously 1 socket 1 at this net is also detectable so idea is that so whatever is the fault is at design can be propagated from here but detective fault stimulation rule and also this is 01 so here you will get the value 0. So socket 1 fault at the output also detectable.

(Refer Slide Time: 10:13)



Deductive Fault Simulation: Example

$L_{I1} = \{I1_1\}$
$L_{I1(G1)} = \{I1_1, I1(G1)_1\}$
$L_{OG1} = \{I1_1, I1(G1)_1, OG1_1\}$
$L_{I1(G3)} = \{I1_1, I1(G3)_1\}$
$L_{G3} = \{I1(G3)_1, OG3_1\}$
G1
G3
0
s-a-1

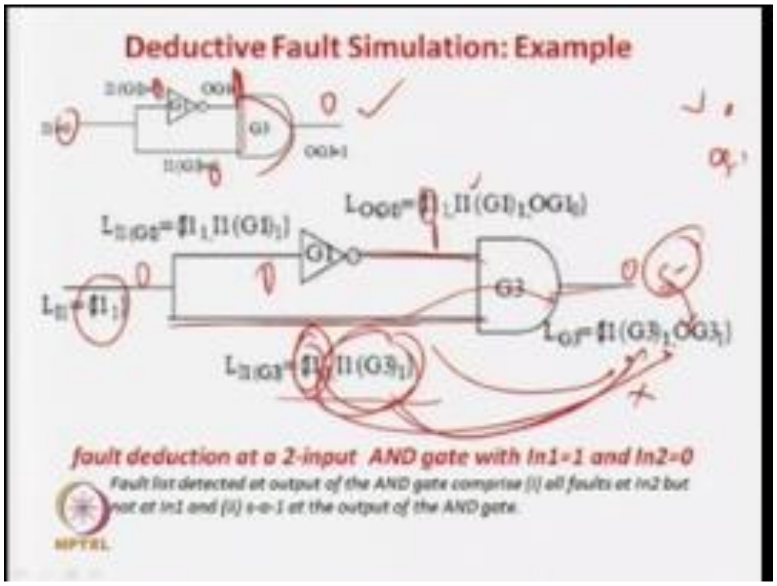
fault deduction at a 2-input AND gate with In1=1 and In2=0

Fault list detected at output of the AND gate comprise (i) all faults at In2 but not at In1 and (ii) s-a-1 at the output of the AND gate

So that inaudibly is the case and that is what is going to happen so let us see so we have taken another example here so this is the inverter and all so we put a 0 in the input so if it is the 0 so you note that this input is 0 so this going to be 1 and this is 0 thee answer is 0. So now in the input to the AND gate this 1 is 10 so let us study the rule so we apply a 0 over here so socket 1 fault is detected at this net so is this 1 now and then find out rule so this 0 and this a 0 so obviously this 1 gets propagated here and this 1 also gets propagated here and then socket 1 fault at this net is here and a socket 1 fault at this net is reflected over here.

So this is the case nobody inversion in with the inverter so we know that whatever fault we used to hear will get here so we have this one and this one which is propagating from this end so these two will get propagated over here because of the inverter and here we had a 0 over here so the answer is the 1 over here so socket 001 so socket 0 fault at this net is also detectable. So this is the new thing that is added and these two are propagated from the input of the inverter.

(Refer Slide Time: 11:19)



Deductive Fault Simulation: Example

0
G1
G3
OG3=1
0
$L_{I1(G2)}$ = {I1, I1(G1)}
L_{OG1} = {I1, I1(G1), OG1}
G1
G3
0
L_{I1} = {I1}
L_{OG3} = {I1(G3), OG3}
$L_{I1(G3)}$ = {I1, I1(G3)}

fault deduction at a 2-input AND gate with In1=1 and In2=0

Fault list detected at output of the AND gate comprise (i) all faults at In2 but not at In1 and (ii) s-a-1 at the output of the AND gate.

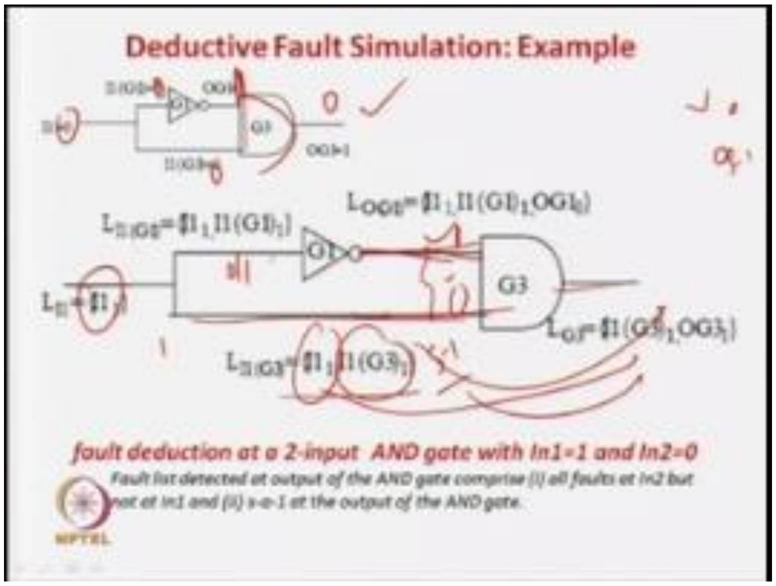
So now what is the status so the status here is let us see what to the status so the status here was the signal here is the 0 s o this is the 0 this is the 0 and 1. So now you see what are the fault lists here and what are the fault list here then we can discuss so now you see so in this case so this net is 1 and this net is 0 so whatever is available here will propagate here so this is the very well known fact okay.

So I mean what happens is the let us see what happens so in this case so whatever fault that will actually impact this net can be propagated over here because is the 1 over here that is what you are expecting and of course this is the 1 and 0 to the answer is 0. So here the socket 1 will be detected at this net so this is here so we expect that I1 this one will be propagated here so this is here. But you see this fault should also have been propagated here but which is not the gate so why this fault but is I11 why this fault should be propagated this fault but the socket 1 at this need why it should be propagated because we failed that in an AND gate what happens so this net value is 1 okay.

So as this is the control units does not have any may this one then everything from here passes from here to here so the whole fault list would propagate but there is the problem so whole fault list would not propagate as our expectation was that in this case if it is 00 or expectation was that nothing would propagate here but we saw that only those faults with their common to both of them propagate similarly in this case there is slight deviation from the intrusion our intrusion says that at this net is 1.

So whatever fault is here will propagate so that is why this is propagating that is true but you kept observe one thing that if something this one is common to here and here so that common part we cannot propagate so let us see why it is not propagate so you can just see why it is not detectable so let us study the case.

(Refer Slide Time: 13:12)



Deductive Fault Simulation: Example

I1(G1)=1
OG1
G1
G3
0
OG3=1
I1(G3)=0
I1
L_{OG1}={I1_1, I1(G1)_1, OG1_0}
L_{I1(G1)}={I1_1, I1(G1)_1}
G1
G3
L_{I1}={I1_1}
L_{G3}={I1(G3)_1, OG3_1}
L_{I1(G3)}={I1_1, I1(G3)_1}

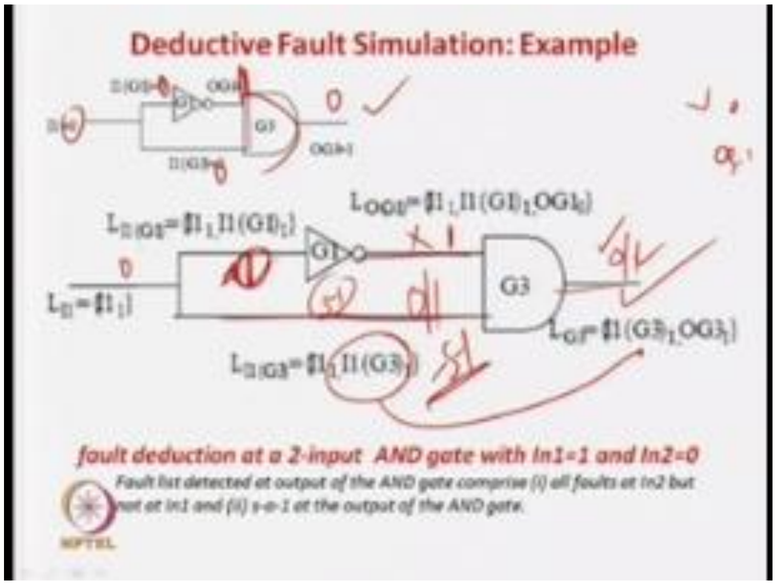
fault deduction at a 2-input AND gate with In1=1 and In2=0

Fault list detected at output of the AND gate comprise (i) all faults at In2 but not at In1 and (ii) s-a-1 at the output of the AND gate.

So in this case it is socket 0 so this socket 1 is detectable over here so this is a 0 normal case fault 1 case so 0101 so it is 10 because of the inversion and your output will be 0/0 so this part is not detective this socket 0 results to a 00 yet do not detectable. So why this is not detectable because this socket 1 fault at this net impact this net by making a 0 to 1 and also these net were making from a 1 to 0 so what is happens the output is 00 that is normal gate is also 0 failure gate is also 0.

So it is not detectable because this socket 1 fault actually is affective both this value is 1 and this value is 1 both this net as well this net so even if this is 1 this fault whatever in this case should propagate over here but as this fault actually affecting both is nets so even if this is the 1 this things not propagate but this fault that is I1 G1 31 that is this net. This net socket 1 usually propagate over here because this is 1 the signal value is 1 so whatever here is propagates over here and this fault does not have this impact so just they get recently very easily verified so I mean in case you apply as 0 over here.

(Refer Slide Time: 14:26)



Deductive Fault Simulation: Example

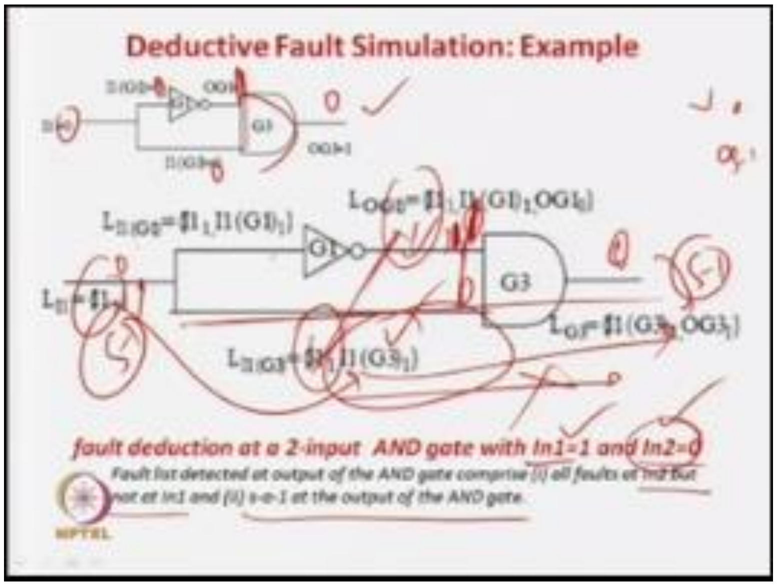
0
OG3=1
$L_{OG1}=\{I1_1, I1(G1)_1, OG1_1\}$
$L_{I1(G1)}=\{I1_1, I1(G1)_1\}$
G1
G3
0
$L_{I1}=\{I1_1\}$
$L_{I1(G3)}=\{I1_1, I1(G3)_1\}$
$L_{G3}=\{I1(G3)_1, OG3_1\}$

fault deduction at a 2-input AND gate with In1=1 and In2=0

Fault list detected at output of the AND gate comprise (i) all faults at In2 but not at In1 and (ii) s-a-1 at the output of the AND gate.

So its signal is 1 it is the 0 so if you just this net is socket 1 this net so the output is 1 in the normal case and here the signal is 0 over here, so this the 1 over here is of 0 or so the output is 01 so for this socket 1 fault at this net what happens normal case it is 0 fault case it is 1 so if the fault case detected at this network. So that means this net this also propagated at this list so why this possible because this socket 1 fault at this net has no affect in this line. This propagates to the output and it can be detectable.

(Refer Slide Time: 15:00)



Deductive Fault Simulation: Example

G1
G3
OG3=1
$L_{I1} = \{I1_1\}$
$L_{I1(G1)} = \{I1_1, I1(G1)_1\}$
$L_{OG1} = \{I1_1, I1(G1)_1, OG1_0\}$
$L_{I1(G3)} = \{I1_1, I1(G3)_1\}$
$L_{OG3} = \{I1(G3)_1, OG3_1\}$

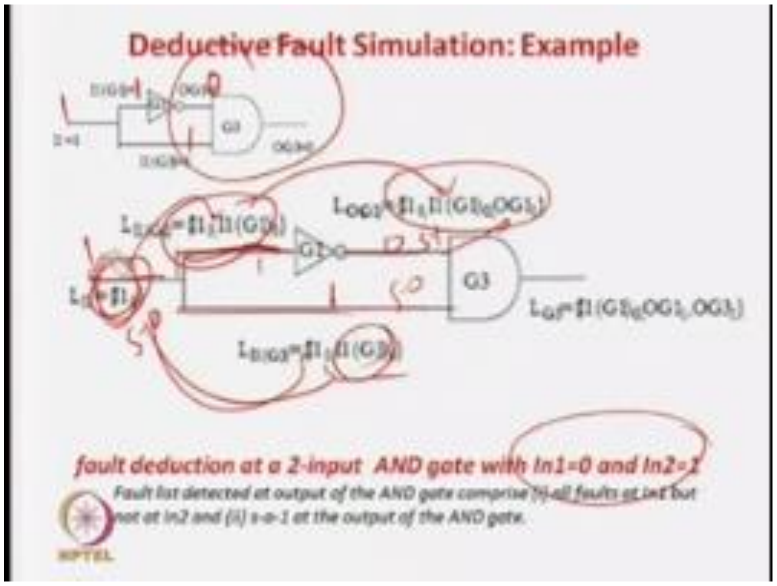
fault deduction at a 2-input AND gate with In1=1 and In2=0

Fault list detected at output of the AND gate comprise (i) all faults at In2 but not at In1 and (ii) s-a-1 at the output of the AND gate.

But this fault that is the socket 1 fault over here at this point affect this net as well as this net both of the net is affected so even if this net is one this fault list is available over a it cannot propagate through the output so the rule is fault detection at two input add again with I1 = 1 and I2 = 1 is all faults in I2 obviously because this is a 1 so whatever fault in I2 will propagate but not in I1 and if there is a common fault between this net and this net.

So that will not propagate even if this is a 1 but any fault which is common to both of them will not propagate other than whatever fault list you have seen not common to between these two will obviously propagate because this net is a 1 and whenever one input of AND gate is 1 the other gate propagate and obviously as target 1 at the output of this AND gate is lift able because this is one and this is zero so the output is a zero. So any target 1 found at the output will be detected so the rule for AND gate with 1 input as 1 and other input is 0.

(Refer Slide Time: 15:59)



Deductive Fault Simulation: Example

G1
G3
G2
$L_{I1(G1)} = \{I1_1, I1(G1)_1\}$
$L_{OG1} = \{I1_1, I1(G1)_0, OG1_1\}$
$L_{I1} = \{I1_1\}$
$L_{I1(G3)} = \{I1_1, I1(G3)_1\}$
$L_{G3} = \{I1(G3)_0, OG1_1, OG3_1\}$

fault deduction at a 2-input AND gate with In1=0 and In2=1

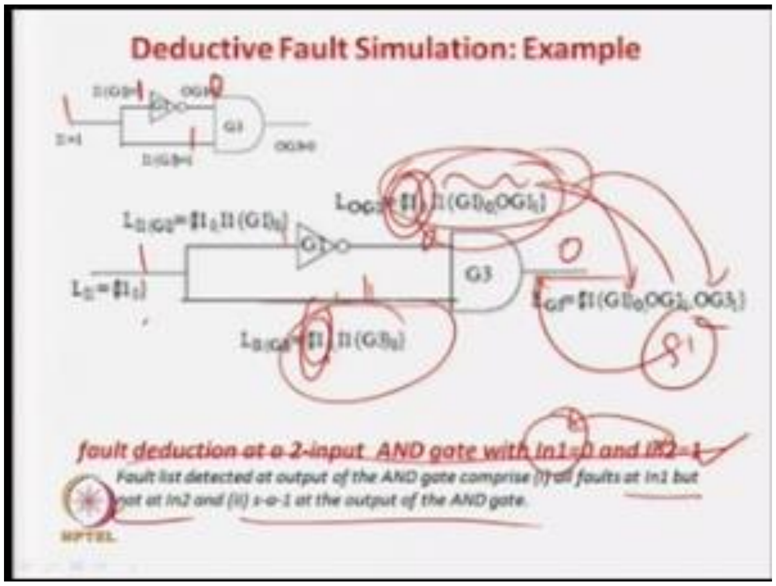
Fault list detected at output of the AND gate comprise (i) all faults at In1 but not at In2 and (ii) s-a-1 at the output of the AND gate.

So what is the rule the rule is whatever at the 1 gate is 1 one 1 input of the AND gate is 1 so whatever you have at the other fault list other input fault list obviously directly propagated because the other input is a 1 but if it is a common fault in between both the inputs of the AND gate then that fault will not be propagated because it is affecting both the lines so that is not ok so if there is same that is the rule so this is another configuration for the AND gate was left that is this is 0 and this net is 1.

So here the rule is obviously these we have been seen just have a look so if this is the AND gate we are taking and the normal circuit we are taking in this case now the run up pattern is a 1 so the run up pattern is a 1 means it is a 1 over a it is 0 over a and it is a 1 over a so you can study the rule about 0 1 for the AND gate so now as your applying a 1 over a so socket 0 at this net is detected so this you are going to have and in this net where the fen out rule so this is a 1 1 this is 0 in the fen out rule this will be propagated over here as well as a stock at 0 fall at this net will be detected because this is 1.

Similarly this one is propagated here this affect is propagated here with the fen out rule and this is 1 so our stock at 0 fall at this net is detectable over here so this is what is the case now because of the inverter the whole stock will be propagated over here and signal here is a 0 so our stock at 1 at this fault is detected by this one so these three list you are having so finally what you are having over here.

(Refer Slide Time: 17:24)



Deductive Fault Simulation: Example

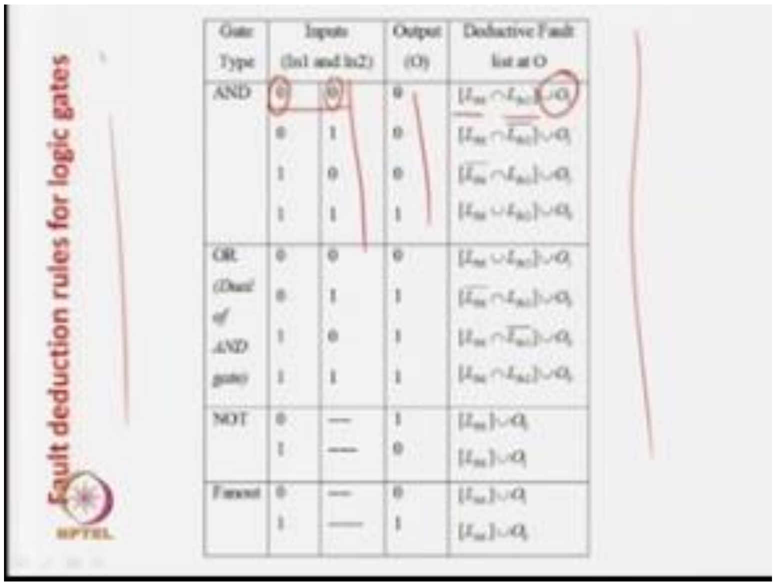
fault deduction at a 2-input AND gate with In1=0 and IN2=1

Fault list detected at output of the AND gate comprise (i) all faults at In1 but not at In2 and (ii) s-a-1 at the output of the AND gate.

So finally we are having you can see that it is a 1 over here it is a 1 over here 1 over here 0 over here so this is the list over here and this is the list over here now in to 2 you can say that as this line is a 1. So whatever whole thing here will propagate here but actually this is not will be the case because this is a stock at 0 and this stock at 0 is common to both these points so this common fault will not be propagated other than that these two will be propagated here and 0 1 is the answer is a 0 over here.

So what stock at 1 fault and this will be detected so this is here, so we have these two faults propagated this is the common fault is not detectable is not propagated and once target 1 found at the output so the rule here is again the same if the fault detection at the AND gate in 1 is 0 and in 2 1 is 1 all fault at in 1 obviously whatever fault here this will be propagated because other input is 1 but not in input but it is a common fault over between them they will not be propagated and a stock at 1 at the output of the AND gate is also detectable.

(Refer Slide Time: 18:12)



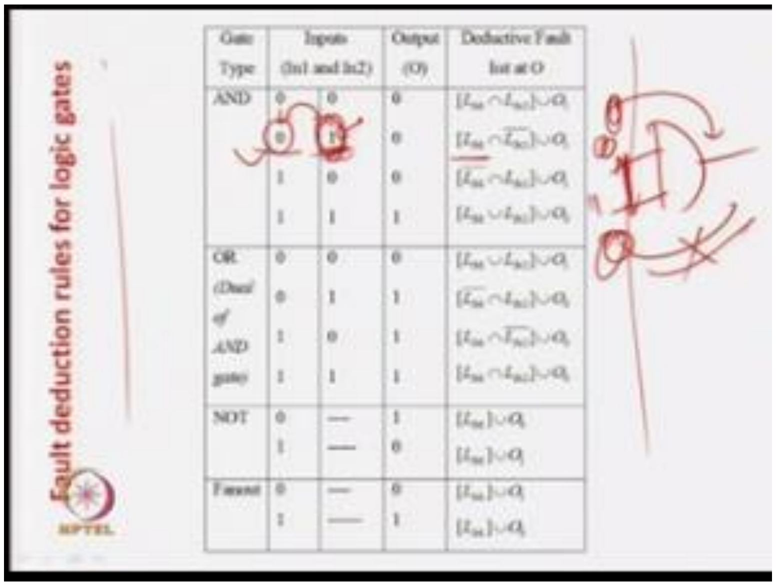
Fault deduction rules for logic gates

Gate Type	Inputs (In1 and In2)		Output (O)	Deductive Fault list at O
AND	0	0	0	$\{L_{In1} \cap L_{In2}\} \cup O_1$
	0	1	0	$\{L_{In1} \cap \overline{L_{In2}}\} \cup O_1$
	1	0	0	$\{\overline{L_{In1}} \cap L_{In2}\} \cup O_1$
	1	1	1	$\{L_{In1} \cup L_{In2}\} \cup O_0$
OR (Dual of AND gate)	0	0	0	$\{L_{In1} \cup L_{In2}\} \cup O_1$
	0	1	1	$\{\overline{L_{In1}} \cap L_{In2}\} \cup O_0$
	1	0	1	$\{L_{In1} \cap \overline{L_{In2}}\} \cup O_0$
	1	1	1	$\{L_{In1} \cap L_{In2}\} \cup O_0$
NOT	0	—	1	$\{L_{In1}\} \cup O_0$
	1	—	0	$\{L_{In1}\} \cup O_1$
Fanout	0	—	0	$\{L_{In}\} \cup O_1$
	1	—	1	$\{L_{In}\} \cup O_0$

So we have seen that how he rules can be determined for all these cases so now just let us tabulate this one fault detection rule for the logic gates so that means what will happen in case of detective fault simulation is that we apply our random then you find out the rules at the input these are the faults that are detectable then what we will do then we find out that if this is a fen out or if this is a AND gate or if it is a OR gate then we have to apply the rules and then we have to find out that if these are the set of fault list at the input then what are the set of part list for the output that you have to determine.

So you need rules so let us see what are the rules so you can see here that AND gate the four cases 0001 1011 so these are the outputs inputs four combination these are the outputs now you see so what it says that if both the inputs are 0 so you have seen what is the if both the inputs are 0 only those faults which are common to both of them will be propagated at the stock at one pint of the out0put will be detected so we write list of n1 intersection list of n2 that means whatever is common in input 1 and input 2 that will be detected in 0 1 that is stock at one point of the output of the AND gate is detectable now you see this case 01.

(Refer Slide Time: 19:25)



Fault deduction rules for logic gates

Gate Type	Inputs (In1 and In2)		Output (O)	Deductive Fault list at O
AND	0	0	0	$[L_{In1} \cap L_{In2}] \cup O$
	0	1	0	$[L_{In1} \cap \overline{L_{In2}}] \cup O$
	1	0	0	$[\overline{L_{In1}} \cap L_{In2}] \cup O$
	1	1	1	$[L_{In1} \cup L_{In2}] \cup O$
OR (Dual of AND gate)	0	0	0	$[L_{In1} \cup L_{In2}] \cup O$
	0	1	1	$[\overline{L_{In1}} \cap L_{In2}] \cup O$
	1	0	1	$[L_{In1} \cap \overline{L_{In2}}] \cup O$
	1	1	1	$[L_{In1} \cap L_{In2}] \cup O$
NOT	0	—	1	$[L_{In}] \cup O$
	1	—	0	$[L_{In}] \cup O$
Fanout	0	—	0	$[L_{In}] \cup O$
	1	—	1	$[L_{In}] \cup O$

So what we have seen that is 01 means what we have seen so some faults actually in this case whatever faults at this list to be propagated that is in list of valve only propagated but at the same time if they assure that any fault at this net will not be propagated that is if there is some common fault over here and here so that is not propagated so what we can say is that all the fault list here will be propagated and all the faults list which are here at this point will not be propagated.

Then it automatically takes care of the fact that if there is some common theme between these two that will not be propagated why because this input is also whatever here will be propagated but if there is something common that will not be propagated and whatever fault is at the valve will not be propagated because this input is zero basically this is the idea that this is zero and this is one so fault list here would be propagated and as this is a zero and this is a one the fault list here will not be propagated so automatically it says that if there is something common in between these two then that will not be propagated because whatever here is not allowed would be propagated.

So how we write this we write in a very simple way we say that l1 all the fault intersection l2 bar or whatever here will be propagated whatever here will not be propagated and obviously the output is zero so I started one fault will be also propagated or the output will also be detectable this is the propagation list this is the output which is fall with the output of the gate which is detectable because the output is zero for the input placement now for one zero you can also see that whatever here will be propagated fault list whatever here will not be propagated.

So that is l1 bar and intersection l2 that is whatever here will be propagated whatever here will not be propagated so any fault common in between these two will also not be propagated plus the stock at 1 at the output is also deductible because the output answer is 0. Now we have seen the best case was 1 and 1 both the inputs of the AND gates are 1 then what is the idea, least over here, least over here all are detectable and this is the one.

So these stacked 0 in the output will be detectable and whatever least here and whatever least here so it is linear in 2, so both of there, both the fault will be propagated. So this is for the AND gate, for the OR gate actually for testing purpose or input or whatever purpose this is just the dual and it is just the dual if you give a very dual route you can find out. So if it is 00 output is 0 so this is an OR gate, so if both of them are 0 the output is 0.

So 0, so though in case if it is OR gate if you apply a 0 over here, so whatever the intact over here gets propagated, because it is the dual of an AND gate. In case of AND gate if one input is 1 then whatever happens at the other output gets propagated and in case of OR gate it is just a dual so if the answer is 0 over both of them are 0, so whatever here propagates over here. But, you know that if in a OR gate if one input is 1, then the answer is 1 and nothing gets propagated.

So just by applying the dual routes we will find out that both of them are 0, then both these will be propagated thus the union, and of course the output is the 0, so stacked 1 will be detected 01. So here in this case we know that this is very helpful, this will allow you to propagate, but this 1 stops. So whatever here will be propagated and whatever here will not be propagated, this is an OR gate, so in this sorry it is an OR gate.

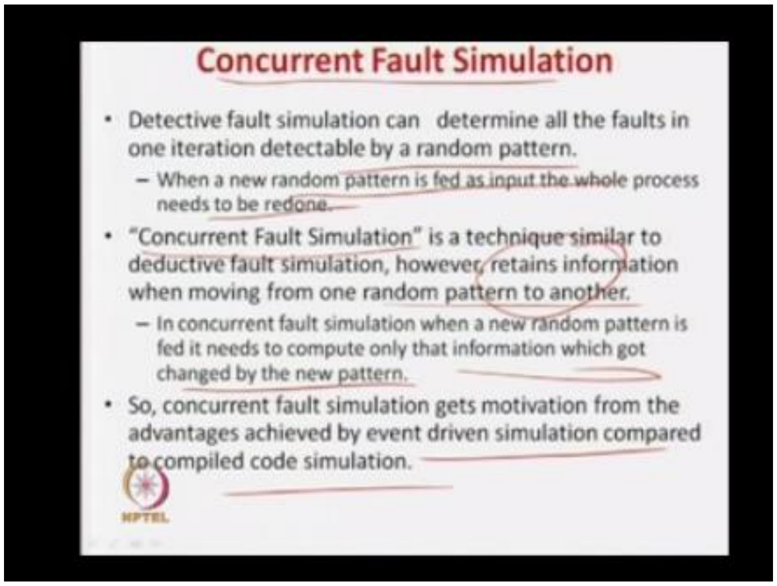
So whatever is the question was we have a 0 and a 1, so we know that this in whatever here nothing, this will not be propagated over here, because this is 1 and if this is a 0 so whatever least over here will be propagated. So you write L1/ that nothing of this area will be propagated and whatever here will be propagated that is intersection L1x and as 01 the answer is 1 over the output so stacked 0 for that output will be detected.

So similarly you can find out the rule for 10 which is this one and 11 in this case it is just like 00 for the AND gate nothing gets propagated that is if there is something common fault is for both of them and 11 the answer is a 1, so in the output case it will be stacked 0 fault will be detected. So that is what is taken over here in the table. Well in NOT gate we know that whatever is the input of the not gate will be propagated whatever in the input of the NOT gate is propagated.

So it is the inverter, so whatever here is automatically propagated here but if the input is a 0 over here we get the answer 1 over here so stacked 0 fault is detected if is the 1 over here the answer 0 over here and the stacked 1 fault is 0 that is what is in store. And in case of fan out it is the same thing like an inverter so you have seen the fault here, so whatever the list here will be propagated here and here.

So whatever is the list they are propagated and you will get the value of 0 now stacked 1 is detected and if you get the value 1 as stacked 0, it is just the reverse. So if you get the signal value 0 now stacked 1 fault is detected in the signal value is 1 as stacked 0 fault. So this is actually the fault detection table for the faults.

(Refer Slide Time: 24:01)



Concurrent Fault Simulation

- Detective fault simulation can determine all the faults in one iteration detectable by a random pattern.
  - When a new random pattern is fed as input the whole process needs to be redone.
- "Concurrent Fault Simulation" is a technique similar to deductive fault simulation, however, retains information when moving from one random pattern to another.
  - In concurrent fault simulation when a new random pattern is fed it needs to compute only that information which got changed by the new pattern.
- So, concurrent fault simulation gets motivation from the advantages achieved by event driven simulation compared to compiled code simulation.

Now what happens so once you have that, so given a circuit you apply a random pattern then in your goal here is there, you have to travel the circuit once and in the one traversing of the circuit what you have to do, you have to find out that where the fault that is detectable. So here are some rules we are applying it is the inverter fault of the rules AND gate fault of the rules with OR gate part of the rules by this way applying the rule from level 0 to level 1 to level 2, level 3 and so on for this circuit you go to the output.

And easily you can find out the list that all the faults which are in the list are detectable by that input. And then that is in one scan of the circuit this is done. Now you find out also there we will say 30 faults in circuit then in one scan of your circuit you find out that there are some 20 faults have been detected. Now if some 20 faults have been detected so it is a good news so another 10 faults are remaining.

So if again you have to apply these things and you keep on, there is a very good and advantages approach compared to severe and balance for simulation, because in one row you can reduce the information about all these. But here you have to think once that, once I have applied one pattern then I have gone for a detective fault simulation at the output then I find out these are the faults are simulated are detectable then I drop them.

And then I can use random and again I redo everything. So what was the discussion on fault simulation by event driven simulation and compile core simulation. So in compile core simulation what was the idea that one pattern you apply one fault you do find out and then forget everything and then again do another simulation. But if I know there is circuit structure is very simple, the circuit structure is not changing from one fault to another fault as such no gate is added or no gate is detected kind of a thing.

So why we are unnecessarily computing the circuit values again and again when you are going for one pattern to another pattern. Then we saw that event reversibility is a very good approach in this case what do we do, so we find out the output or the input and the fault on normal case, then we retain some values or all the value of the circuit we will take and then what we do when we apply another pattern or another fault maybe then which was able to recomputed only both version of the circuit with got change as an impact or something.

In the new pattern or new fault or something, so that was the idea that this is missing in our detective fault simulation. So in one fault we apply the scan to the circuit by applying the rules in the table which you have seen then if I know this is the release of the faults which are detectable. But we generally cannot or generally do not save anything in that algorithm and again we have to redo everything.

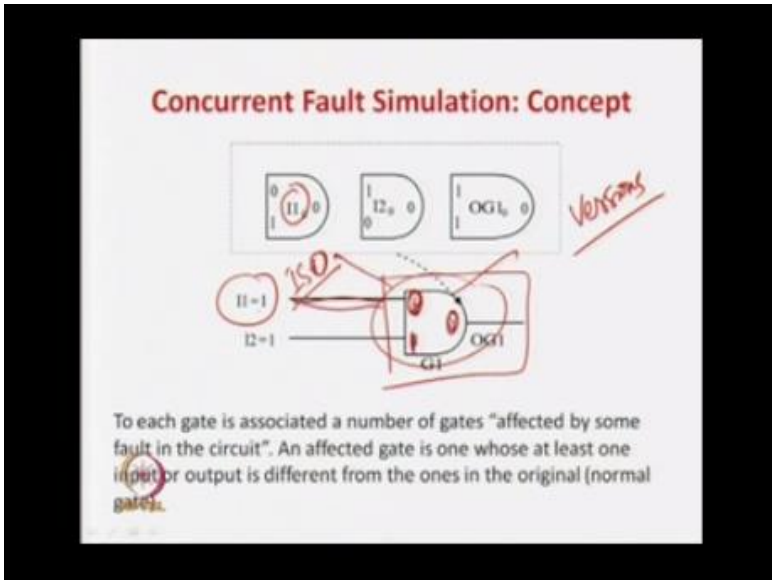
So the last fault simulation technique we will start is the concurrent fault simulation. So that is actually same thing as detective fault simulation guide that in one scan of the circuit you are going to find out what are the faults that we detectable, but at the same time we are not going to throw away the results. So if by applying one pattern we find out that these faults are detectable, then we do not throw eliminate all the results.

We apply another pattern and then we will try to recomputed only those things which are got to getting changed that is our event driven simulation. So you can think that concurrent fault simulation is a event driven simulation plus detective fault simulation. So that idea you have to see, that is what we are saying a detective fault simulation which we saw last can be driven all the faults detectable by a random pattern.

But when a new random pattern is applied the whole process has to be redone that is just like a compile code simulation. So you have to do something better, so the last is concurrent fault simulation technique similar to deductive fault simulation, but you have to retain the information that what I have done, so you have to retain when you are going for another random pattern. So in the concurrent fault simulation the random pattern is said it needs to conclude only that information which got changed.

So that is the idea for a detective in our default event driven simulation. So concurrent fault simulation gets motivated and one touch the event driven simulation over the compile code simulation. So that is the basic idea.

(Refer Slide Time: 27:39)



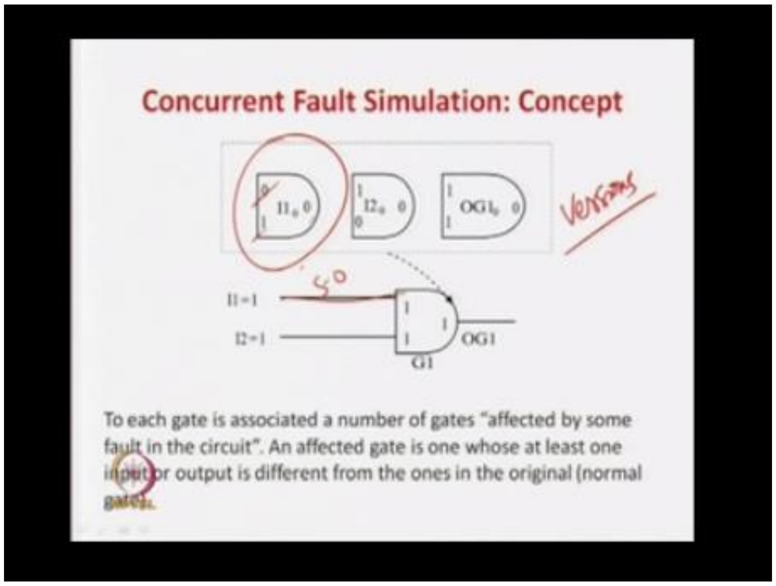
Concurrent Fault Simulation: Concept

To each gate is associated a number of gates "affected by some fault in the circuit". An affected gate is one whose at least one input or output is different from the ones in the original (normal gate).

So we will see the basic concept of a concurrent fault simulation. So let us take an AND gate, so this is your AND gate say and then you apply 11 and the answer is 1. So you have to first draw this normal condition then you have to find out that these versions of the gates we have to draw some versions of these gate which are affected by the different type of gates like for example, you see if I0, I1 are gate 1 that means if this net is a stacked one.

Then what happens this input is 0 it because stacked 0 sorry this is stacked 0 you apply 11 you are applying. So this gate is stacked 0 so input will be 0, this is 1 and obviously the output will be 0. So this is one fault which is affecting this gate, so this information you have to store somewhere.

(Refer Slide Time: 28:25)

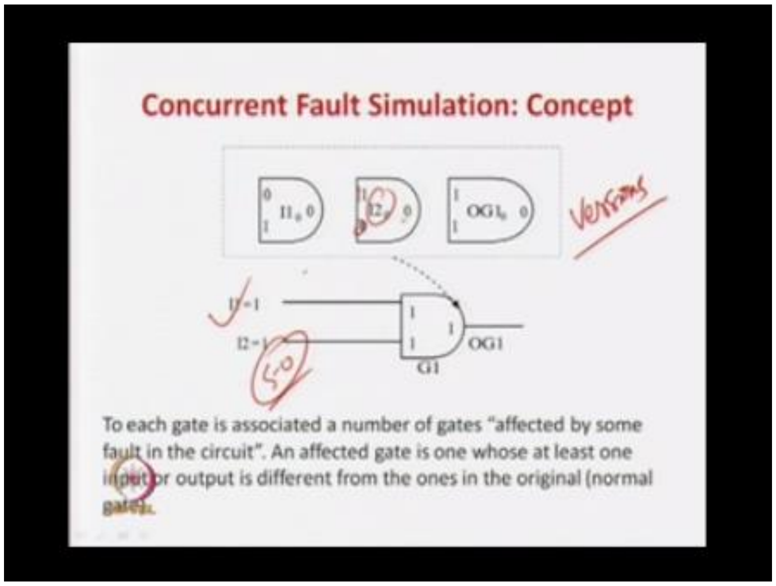


Concurrent Fault Simulation: Concept

To each gate is associated a number of gates "affected by some fault in the circuit". An affected gate is one whose at least one input or output is different from the ones in the original (normal gates).

So this is the gate which is saying that this gate is a stacked 0, so this answer is 0 this is 0 and the output is 1.

(Refer Slide Time: 28:35)



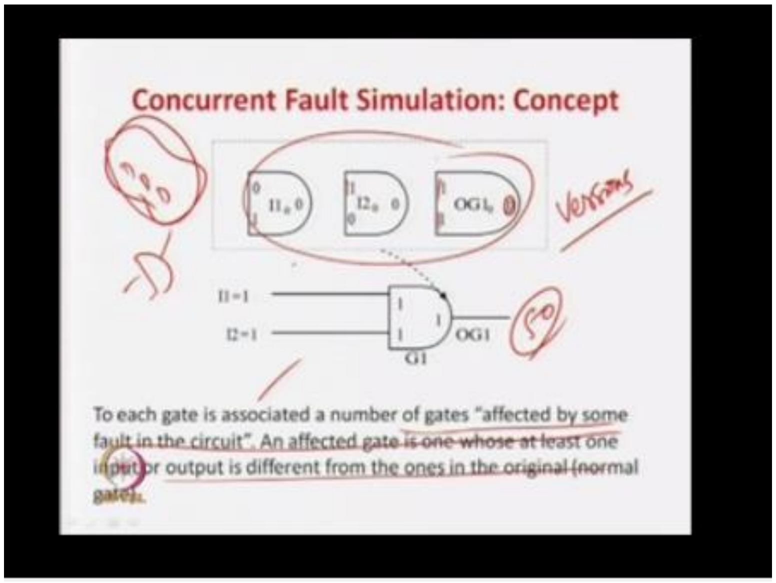
Concurrent Fault Simulation: Concept

I1₀
I2₀
OG1₀
Versions
I1=1
I2=1
s-0
G1
OG1

To each gate is associated a number of gates "affected by some fault in the circuit". An affected gate is one whose at least one input or output is different from the ones in the original (normal gates).

So this is one version of the circuit and this stacked 0 fault here. Now and the stacked 0 fault can be here so this is case it is 10 because this maybe stacked 0 the output will be 0.

(Refer Slide Time: 28:45)


Concurrent Fault Simulation: Concept

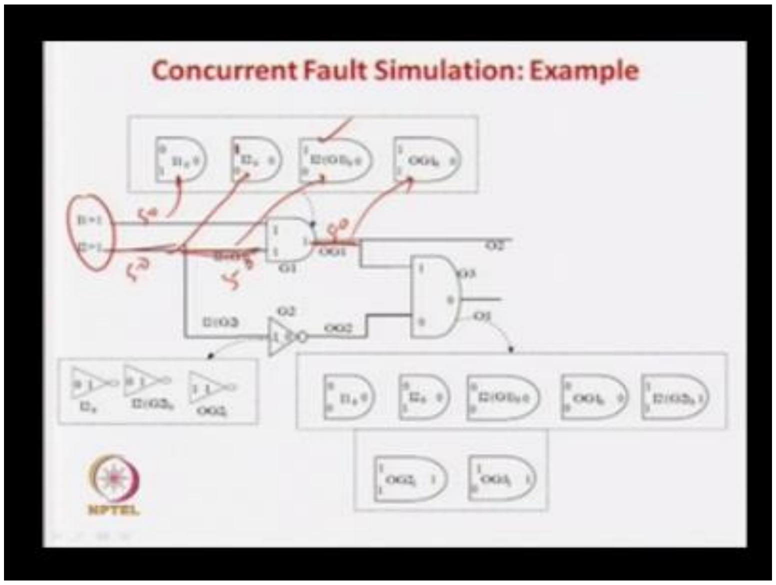
To each gate is associated a number of gates "affected by some fault in the circuit". An affected gate is one whose at least one input or output is different from the ones in the original (normal gate).

So this is the another version of this gate when there is a stacked 0 for length is there. Similarly there can also be a stacked 0 at this point so this version is a 11 then the output is 0. So for each gate you affect, you attack some gates in this circuit which is affected by at least some fault see each gate is associated with the number of gates affected by some faults in the circuit. So for each gate you have to attach a bubble.

So this is the one gate say and you attach a bubble, so this bubble will comprises all other versions of the same gate with different, different faults in the circuit. So if there are some 30 faults in the circuit, so your bubble I have some 30 faults which are affecting the circuit, even if you have some 30 faults which affect the 0 obviously so there are only, I mean you have this fault, I mean gate list in that bubble.

So in other words in concurrent fault simulation what are the first step we are going to approach to each gate we will attach a bubble, in the bubble there will be some other gates. And these are the gates which are affected by some fault in this circuit, only those versions of the gates will be available. So if there are 30 faults in the circuit and 20 of them affect to one gate, so the bubble for that gate will have 20 more gates of the version so you store that one. So that is what is the first we have to do.

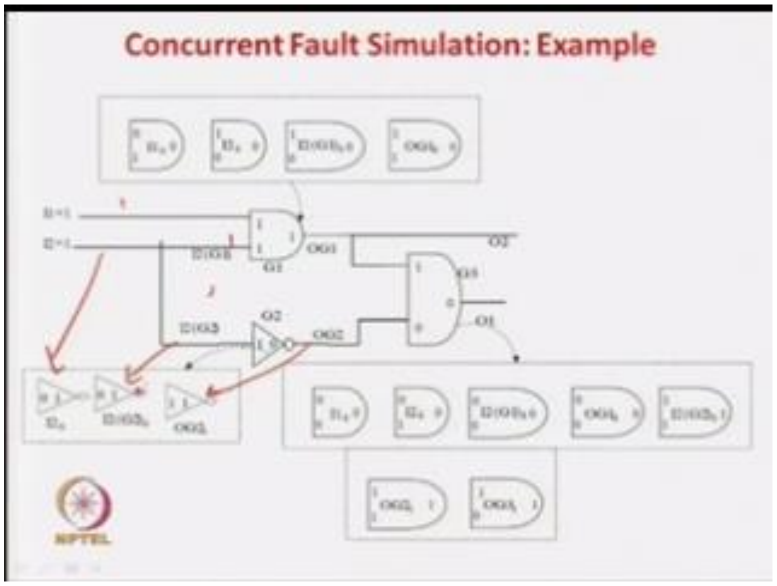
(Refer Slide Time: 29:53)



Okay, so let us take the same example and let us see how we can make this becomes more clear, so we are applying the random pattern 11. So we know that, so we are now considering this gate. So what are the faults, so stacked 0 fault at this gate, so this is 0, so this is 0 knot 10 and the answer is 0. So this one is reflected over here, similarly there can also be a stacked 0 at this net so is the I20 so in this case this will be the value 1 this will be a 0 over here and this 01 the answer is 0.

So and the fault list you are having for this gate. Now you already told that this is 1, this maybe the fan outlet so this is different so I2G1 stacked G2 is also possible, so in this case we are going to have this gate. So this is the same, these two I mean stacked 0 you have stacked 0 here this stacked 0 fault and stacked 0 fault over here the same affect. So these are the two gates with same affect 10 and the output is 0, but you have to keep the version, because they represent two different faults. Now this 11 so obviously a stacked 0 at this let the output over and stacked 0is also affecting 11 the answer is 0 so for this version you have this.

(Refer Slide Time: 31:01)


Concurrent Fault Simulation: Example

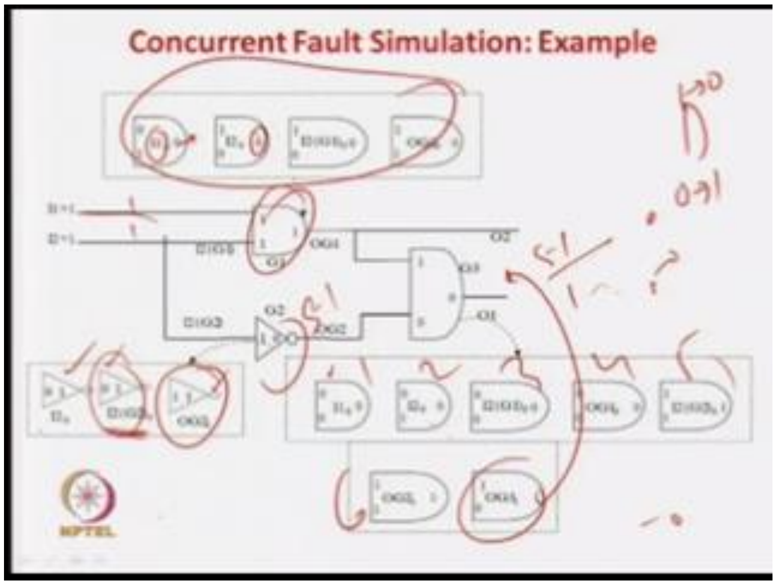
So this AND gate has four gates in the bubble corresponding to a stacked 0 fault here so for this force stacked 0 faults you have this thing in the bubble correct so this is the 4 things of the bubble okay so you can also thing that this is a kind of a parallel of simulation that you maintain an array but here the array size is unlimited we are not going for any kind of a computer architecture oriented algorithm so whatever it may be mean if some 100 fault sacked I get so you will have some 100 gates in the bubble so here we are not using architecture kind of a thing.

So now just let us look at it is 1 1 so 1 1 and in this is case is a 1 it is 0 now let us study for us inventor what will be the cases so for the inverter you can easily find out that normal case it is 1 and it is a0 so if this gate is stacked 0 over here obviously so I20 so this is I20 so I20 what happens so this one will be 0 this is stacked 0 and this 1 is 1 so this is reflected by a stacked 0 fault here that is the change for this gate similarly a stacked 0 fault here will also affect.

So now will be 0 and the answer here will be a 1 normal case it was a 10 but in the fault case it is 0 1 so this one I20 stacked 0 affect is also reflected over and of course so 11 normal case it is 1 and the answer is 0 so as stacked 1 fault at this 1 wheel will also be affecting so in this case you see the input is 1 the answer should be but a stacked 1 fault here will make it a 1 so 11 is affect is also shown over here so these are the 3 faults which is going to give you the bubble list of 3.

So this is the first case this is the first case sorry just a minute so this is your first case this is your second case and this is your third case so this as some c faults in the fault layer so this I how we are making so now the thing will all the benefits will start coming when we apply another l down pattern.

(Refer Slide Time: 33:02)



Now go back let us firsts see what is the case no we have so what we have in the normal case 11 1110 so this so also 11 and this is 0 and the answer here is a 0 now for this gate also with there will lot of gates in the bubble which is actually affecting this 1 so now let us see how can we find these are the faults thing very number faults affecting this 1 so you can see that this 1 depends on 3 value the value here the value her.

So these are the two I mean this gate 10 is the normal case and then you have to find out for all you have to the find out the bubble list for this gate that is so we have to find out the bubble list that is what are the faults which are actually affecting this agate so the normal case is a 10 the answer is 0 so you have to find out for all cases of faults all versions of this AND gate where at least these are the difference so here instead of 1 if it is a 0 you have to put if that means if it is a 0 for here you have to put that version.

If this is a 1 you have to but the version and if this is 1 in this we have to the version so all such cases which changes the input signal values the output signal value you have to put it now you can study this list so you can see here the answer is a 1 correct but now if this stacked 1 if this sacked 1 fault 1 is 0 so this gate output will be a 0 so here whether you are going to get a 0 and the answer is 0 so this 1 you have to keep so how your driving this we are deriving this from here that is with any signal that this signal change if it is from 1 to 0 then you have to put it.

Now how you know that this will be changing that this fault this is the fault I1 stacked 0 this stacked 0 this stacked 0 will result as 0 at this gate output this we can find out from this bubble list okay and then this 0 with the signal will change so this 1 we have to put it we can put it now you see that is another one which is also the same thing so this 1 is also changes from 1 to 0 so that value also you have to put it over here so that is a 1 this is not affected the answer is a 0.

So this is the case which is affected now you can also see that this gate is also affecting the signal so this 1 is also you will be available over here and this stacked 0 that is this net stacked 0 means that is this gate fault is also change this value so you are going to have this here so what we are having so 1, 2, 3, 4. 1, 2, 3, 4 this 4 gates is actually change the signal from 1 to 0 so they are actually include in this bubble now you see obviously now we will your go to check for this input okay so this input means is a 0 to 1.

So now you can check that if something change from 0 to 1 thus will be included so you can see that these are the 3 gates this one this one and this one I think these are the 3 gates which are actually changing the value of this one correct this one is changing from 0 to 1 so now you can see OG1 okay sorry so this gate consider this is already over so now you can think that something else which are changing the value from 0 to 1 we have to include and I2 G2.

So you can find out that if this is the gate so I2 G2 this is I2 G20 so in this case here the answer will be 1 so this one you can include it over here and I20 is also the case were actually I20 already we are including over here that is if this gate I20 fault is there so then this actually this 1 is included over her so already this is taking into picture because this fault I2 stacked0 affect this gate as well as this gate so in this case the signal changes are from 1 to 0 and it is from 0 to 1 so both signal changes are there so this gate actually can be brought down from this list as well as from this list.

Any way so this gate is already added so this gate this gate is changing the value 1 here so 0 to 1 so this gate impact your are bring out over here and O2G1 that is this 1 this net this net you can think of as O2 output to this net this net staked 1 so this is the picture so it also changes the value of this 1 that is 0 to 1 so that 1 also you have to bring it over here and in this case you can see that it is already 1 and this fault that is 02 that is I2G2 stacked 0 this I2G2 stacked 0 so this one stacked 0 that this net stacked 0 converts this to 1.

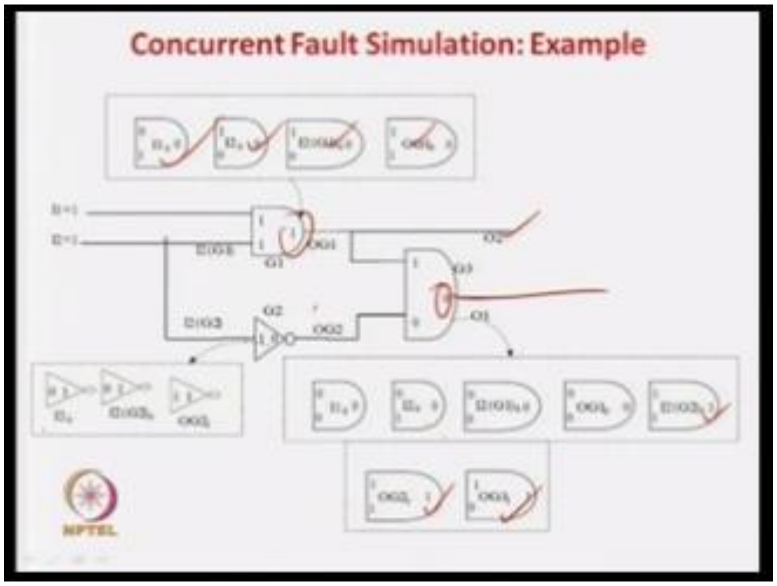
So here the signal changes from 0 to 1 11 the answer will be 0 so you have to write that this gate is saying at 11 the output is a 1 that is we have to write in that way okay so and this 1 this stacked 1 fault sorry this is a this list already we have discussed now this net stacked 1 this net stacked 1 this net stacked 1 actually converts this from 0 to 1 and 11 the output is 1 so this net O2 OG2 stacked 1 is also over here and the so these are the gates which actually get propagated this 1 , 2, 3, 4, 5, 6 this 6 gates are propagated from the previous gate input to this 1.

So this you can also say concurrent fault simulation also this is just like some fault these are prorogated from the previous level to the next level but here actually the gates the this gate versions are propagated from the previous level to next level same concept is there so not much change but and also one more for the patter is there which you one more thing will be here like we have seen the directive fault simulation that here if you answer is 0 and then the stacked 1 fault at this gates detected that as to be added to the list.

So in this case also the answer is we are saying that the pattern is 01 00 so any signal change will be affected so signal change at this net will be o1 stacked 1 then this signal change will be a 1 so this gate list is added that is 1 0 and the here answer is a 1 because this net is stacked 1 so the last signal at the signal change at the output so this 5 this 5 fault this gates 1, 2, 3, 4, 5, 6 this 6 are propagated from the previous values and this one corresponds to a stacked 1 fault over here so in this case.

So this how we get this fault list concurrently with this one so with still now you might see that the concurrent fault simulation is nothing but it is a detective simulation only.

(Refer Slide Time: 39:05)



Concurrent Fault Simulation: Example

In detective simulation what we are doing with this propagating the fault list from 1 part to the another part and we are actually propagating the bubble list if the gates otherwise there is no difference the difference is start coming into picture only when we will change this random pattern that let us see how many faults are detected so now this is your bubble list.

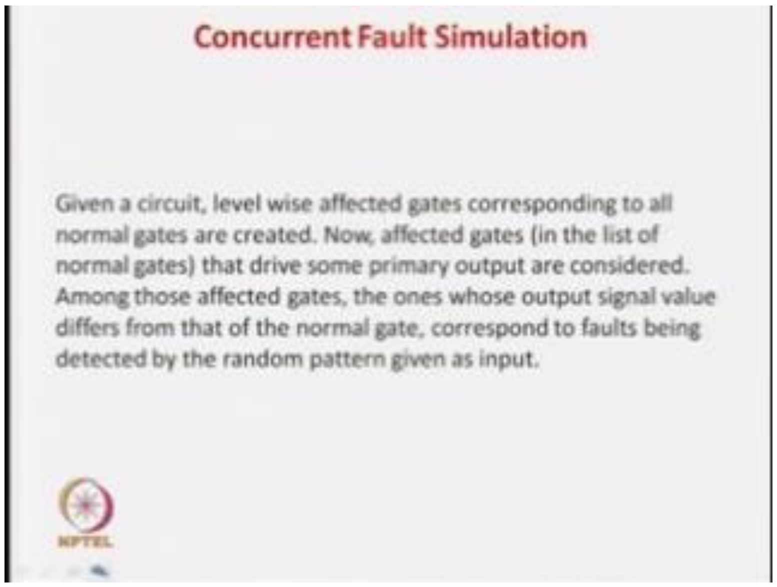
So you know that in this case the normal case the answer is a 0 so where ever the answer is a 1 at the output of the AND gate you can know that the fault is detected so in this case you can find out that there are two faults that is 02G2 stacked 0 that these net stacked 1 and this is the stacked 1 and this net stacked 1 so these two faults are detectable because the output of the AND gate is

1 and in normal case the answers are 0 but now similar also you can see that at 02 also some faults can be detected.

Now what are the faults that can be detected here the answer is a 1 so normal case 02 will be 1 but here this is 0 the output this is the output so all this 4 faults are detected at this net and this 1 sorry O2G1 that is fault and output stacked 1 fault are detected at this thing so this is the same thing like our what you call this detected fault simulation we can find out the fault list so here the fault list is not very directly computable when it is very generated that is in case of a detected fault simulation what happens the fault list will tell you that is faults are directly detectable.

That you get a list over here that 01 some list over there oaky and from list you can say that these are the fault which are detected but in this case we have to just so simile computation that which are the gates is output is different so in this case this is different this is different this is different so this is the fault are detectable at this net and in this case the answer is a 1 so 1, 2, 3, 4 this 4 faults are detectable over here.

(Refer Slide Time: 40:50)

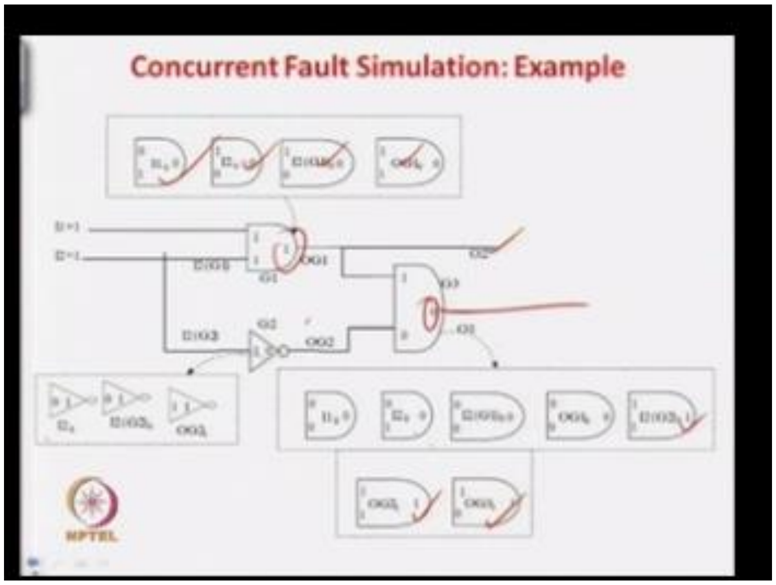


Concurrent Fault Simulation

Given a circuit, level wise affected gates corresponding to all normal gates are created. Now, affected gates (in the list of normal gates) that drive some primary output are considered. Among those affected gates, the ones whose output signal value differs from that of the normal gate, correspond to faults being detected by the random pattern given as input.

So this is just a very simple extension or you call the seam version of your detectable\ fault simulation in the concurrent fault simulation instead of propagating the four list you are just propagating the series of gates that is the one thing.

(Refer Slide Time: 41:02)



Concurrent Fault Simulation: Example

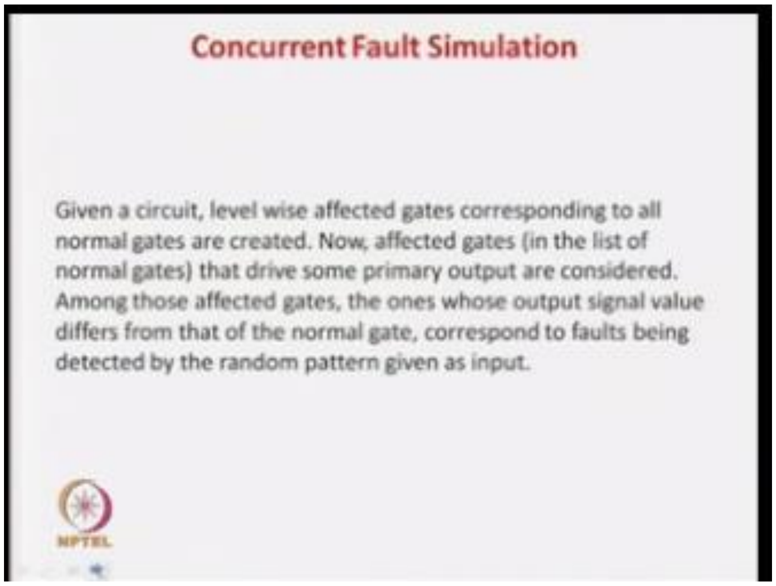
And one other thing you should observe that in case of deductive false simulation we have the faulty list which tells this is your fault list say so it says that only fault one fault 2 say fault 3 are detectable but it will not retain any information which is non deductive so here you see we have 1, 2, 3, 4 so 4 gates are unnecessarily which you are hanging over there but they are not deductive you so these things are hanging over there so there nothing to do unnecessarily they are occupying you memory.

But we have seen we will see in the next slides that because of these things some information is retained that is now if we change the value from 0 to 1 or 10 whatever then we need not recomputed the whole thing we5 just recomputed whatever changes have happened over here only those things will be recomputed and that is like a even driven simulation something will be same.

But in case of this detective false simulation even with the fact that we have only the set of one to one only this things formation will be available because they are been rejected so this unnecessary things are not there but wherever these are changing pattern you have to redo everything so that is the bit trade off, so we have to think that if you are doing a deductive false simulation.

Then at the output we will have information about only these three false because these three false are deductible by this one only and this information about this unnecessary four gates because they are output is 0 and in the normal case it is 0 so the false are not deductable they will not be there but the trade off is that whenever there be a change in pattern then again you have to redo everything.

(Refer Slide Time: 42:22)

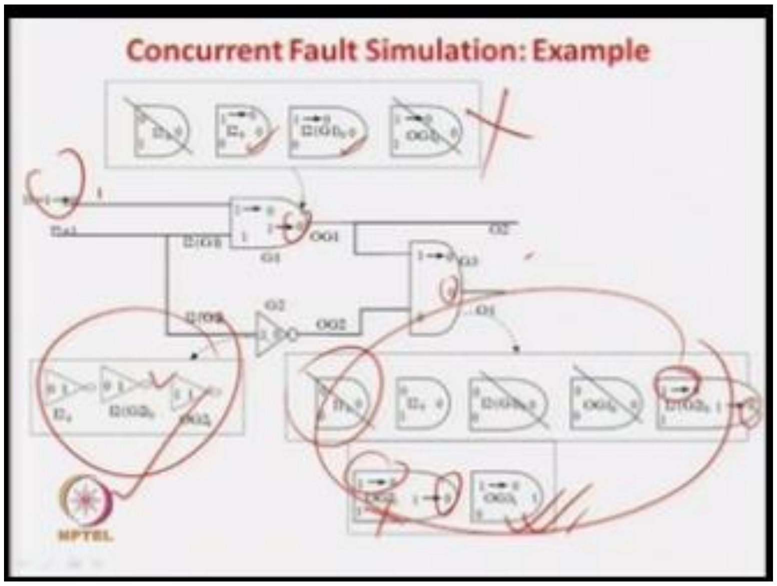


Concurrent Fault Simulation

Given a circuit, level wise affected gates corresponding to all normal gates are created. Now, affected gates (in the list of normal gates) that drive some primary output are considered. Among those affected gates, the ones whose output signal value differs from that of the normal gate, correspond to faults being detected by the random pattern given as input.

But here if the change in pattern they will be noting have to do much only whatever changes are required that has to be done like that is what thins thing which we have already discussed not to redo so the concurrent sorry.

(Refer Slide Time: 42:37)



Concurrent Fault Simulation: Example

So this is what is the next slide which we are going to see so what we have discussed that, now we have to see the where it fields that unnecessary we were carrying some additional gates so unnecessary we were carrying some additional gates so unnecessary what we are doing so unnecessary what we are doing, so unnecessary we were carrying out some gates which could not detect any false.

So those gates were carrying with us but in case of deductive false simulation those gates were not been carried so now what is the advantage that you have to illustrate. That I think if you just see so this gate this gate unnecessary will be carrying out because the output was 0 and this output is also 0 but now we will see that what is the advantage of carrying these gate okay because this is the trade off.

In case of deductive false simulation you will not do this but still if wherever new pattern is applied due to redo everything here we will let us see here in concurrent false simulation like carrying this date gate as you can see what is the advantage so now let us change the new pattern for 11 to 01 so these are change, now as you have already seen it what you call our deductive forward sorry event even simulation.

OR computation will only be for those cases where these are changed so let us see what happens to initially it was 11 and the answer was a 1, in normal case we have to apply 11 the answer was a 1 in this case now these are change in signal value this is changing from 1 to 0 so what we will do we recomputed only when things are changed so re-computation is 1 to 0 okay here it is a constant so they gate output is this one, correct.

Now this gate was corresponding to a start gate 0 forward over here so it was a 0 over here 1 over here and a 0 over here correct so this and this new hand so what we are going but this is the old thing that is retained so you can see now your normal gate will become 01 and a 0 so that is your normal gate this is your normal gate which makes 010 because this pattern you have to check.

So this gate you can delete okay this gate which we are going to delete, now there is so now so what is the case so this gates deleted now there is another gate was I2 start gate 0 so this corresponds to the second gate in this least equilibrium but this I2 start gate 0 was a start gate 0 over here so now a start gate 0 over here was so this input was a 0 because this gate is start gate 0 now there is an impact of change is from 1 to 0.

And the answer is off course a 0 but and this gate will be retained okay so why this gate will be retained because initially the gate it was a 10 and the answer was a 0 now sorry so whenever when the input signal was 11 at the input so what this gate worth a start gate 0 for this look like 10 and the answer is 0 and just start gate one for at this, it was a start gate 1 for it over here and the answer we are applying a one over here you are applying.

Now we apply a 0 over here so this one will be changing from 0 the output will there will be no change so you need not compute this and you need not compute this, this is as well retain in same as in the previous case when we are applying a 1 over here and only this small change is this one so these are signal change because now the gate normal gate looks as 010 this is the normal case this one and here with the fault of where the gate is 000.

So this is the signal difference over here so this gate is going to be retained so that is again we see that what we are doing so this gate is retained so what we are doing over here very simple thing we are going to event-event simulation so we are just changing whatever signal here required this is the change here and in this case after doing this change you can find out here is that this gate and this gate remains same in that so you can drop it.

But for the second case it is a I2 start gate 0 fault this is the signal change from here this done not change this remains same so this things and this things retained from the previous computation and we so say that these are still a signal difference between this one so this gate is retained, so some computation that the computation at this level and computation at this level are same, now let us see the third gate.

The third gate was I2 G1 start gate 0 so this line is start gate 0 over here so now if similarly you are applying here is a 1 is the 0 over here so in this case these are 0 so we know that this start gate 0 over here to signal this signal is 0 this is only a change over here and this thing is retained and this thing is retained so we can find out that what is the ordinary gate the signal was 1010 so in this case it is 0 and 0 and answer is 0.

So again by the same logic of I2 start gate 0 this gate is also retained but this computation and this computation is same now let us look at the output from the gate what is this change so whenever the false are not over here you are applying the change from 011 to 01 and here is the start gate 0 fault over here so you can say this is change from this signal these signal and this signal values retain same from the previous computation.

And now this gate is having 10 from 1 to 0 you are going here so the answer is the same and the answer output is 0 and this is 1 so these two gates become normal gate and this output start gate 0 gets become similar so this gate is also has to be drop so now what happens these two gates drooped and these two gates keep on retaining but we need not compute the value here and need not compute this value.

They may just carry propagated from the previous one now in this case you see if you are applying a 1 over here then this can this change this circuit change is over the signal change changes over here so the fault list at this gate all the things will be retained because the single changes over her and in this line there is not impact so there is no impact in this line so if there is no impact in this line then you did not thing about anything.

You can just copy paste what was your available over here so if it I2 start gate 0 okay so you are applying a 1 over here so it will become 0 over here that is 0 over here and the answer will be one over here now in the correct sense it should be a 1 and the answer should be a 0 so that is this fault list this gate is already retained from the previous list so previous list of when the input was a 11.

So what we have seen in this example that whenever the fault is affect or this input changes this one so this thing can directly be retimed over here because the change in the input from here this will not affect this one so this is the advantage of a concurrent fault simulation over that fault simulation that many of the portion of this circuit like this portion of this circuit is not affected by the change in this one.

Also you can directly add this bubble that is you need not even think of in this case of the angle these are change over here so some impact on this gate is available as ever happens so you did not go and deleting this gates and you keep on retaining and in this gates also you have same thing competition of this input and computation of this input but for the this portion of the circuit you need not think about the just you retain the value.

That is what is the importance of our detective sorry a concurrent false simulation over a detective false simulation now you need not keep non this thing now by the same rule so this was the now this is the change so in case this will be this change from 1 to 0 and now you can see that this correspond to this net start gate 0 so if this thing this whole list is from the old competition this from the old competition.

So this is the what we are having from the old competition so now you can see that because of this change so you will be normal condition will be 00 and the answer is the 0 so this is about this gate okay, so now you see that I1 this start gate 0 means this 00 and 0 so this gate exactly becomes similar to the normal case of this one so you can delete it, now I2 start gate 0 so this note start gate 0 if you think.

So I1 start gate 0 if you think so this one will become 0 this one will become 1 so this is a 1 okay and this two are same as in the previous competition so this gate is retained so in fact what we are doing here so just by this change from 1 to 0 then this is the old list which was available with us so we are just find finding out what were the changes by this change what were the changes in this gates we are finding out.

That is the bubbled list and if something becomes equal to 000 that fertility and if something is not become equivalent we just retain it so for I2 0 these are signal difference between this one and this one so this gate is retained now I2G1stack at 0. So i2g1 this one it will comes as stack at 0 so then what happens this gate this one will this stack at 0 so this one is stack at 0 so this one is the one so this is the one this is stack at 0 so here it will be a 0 output is 0, so now the thing is that for this case you apply a one u1 you get a 0.

So this i2 stack at i2g this part stack at 0 we will leave to this gate is 000 which becomes equivalent to the gate under in normal condition wherever you are applying 01, so this gate will keep on deleted. Similarly, you can easily find out that if this is the stack at 0 over here i2g10 then the input becomes 000 which is again similar to the normal condition of this gate under 01 so this gate is deleted so this way you find out that two of these gates deleted, okay so here two gates deleted this one is retained.

Now you just think of the case i2g2 stack at 0 so this one is stack at 0 if you think then what happens then this one will be 0 so this one will be a 1 and here there will be a change from 0 to 1, because initially the pattern applied for the one over here okay, so the pattern one over those applied 1 over here and if this is a stack at 0 fault here so there is no effect over here, okay so no affect was here so there it was 11 the output was a 1, so your gate was looking like 11 and the output was a 1.

So this was the whole competition now you are applying this change so this change is leading to this change this is change is leading to a this change, so this gate will be now having 010, but still this gate will be retained because they have single difference over here, so you can see that because of this change in this one this competition is not affected only this change and this change you have to get. So some cases this false I means it is a competition is same.

So this gate is retained that ith this gate i2g1, sorry i2ge stack at 0 sorry this one is i2g2 stack at 0 this case is retained over here because the single difference but only you have to compute this change in the input which is arise because of this change for this is retained. Now similarly, you can easily verify that what are the other gates which are retained as o2g1 stack at 0 that this one if you are getting going to get the stack at 1 then this one will be changed and in output will be change this one will be a 1 but the single difference with this normal case so this is retained and og3 the output g3 so if you just think of this case so this fault is also when it is going to retained because this is the stack at 1, okay and the single change only because of this single change.

So you write like this and this gate which corresponds to o1 stack at 1 is still retained because the single difference that is in the it was just because of this change application so it will become 000 in the normal case but if the stack 1 the answer here will be a 1 so by the previous competition only you can see that these gates are retained. So what we have done basically by this signal change what happen by this thing this is single change which is happen which is not affecting this gate in any ways so directly you retain all the gates without doing anything.

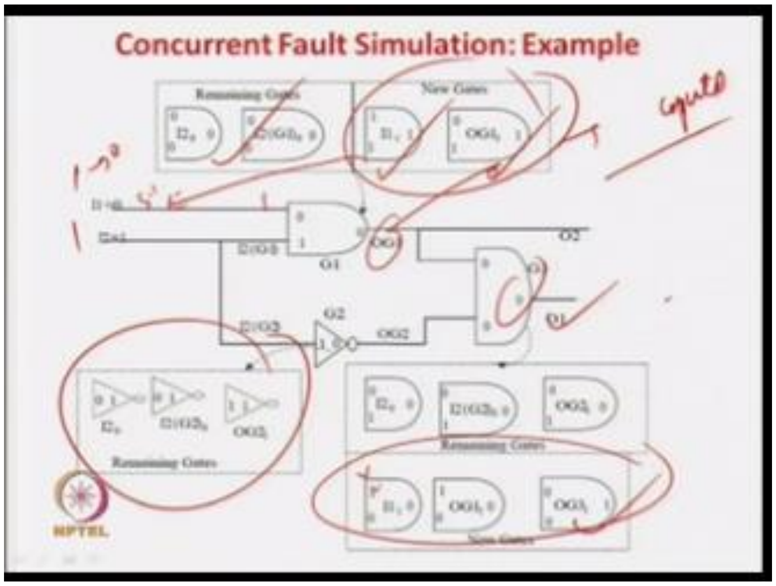
But for this gate and for this gate there is some change this is the signal change that is happened so based on that you find that this gate, this gate, this gate get deleted because here the signal after this change it will become 000 and this gates will also become 000 which becomes equivalent to this so these are dropped but for this things this one, this one, this one and this one they retained because there is a signal change.

So now you can easily derive that which are the false which can detected over this so you can find out that there is a signal this is the form which is detected because there is answer the 0 and here the answer is 1 so 1 for the detected but this one now the answer becomes 0 and this will be the answer becomes 0 so this fault does not get detected by this one and this gate you can see that none of the false get detected because the answer is a 0 over here and 0 over here.

So in this case also the answer is 0 over here so none of the false get detected by, no additional fall I mean what you can say by this divide only this false get detected. So but what essentially we have done so we have retained this gate directly without going for kind of additional computation and in this case also we have gone for some computation but the competition are minimal here actually did not do any competition but still now we can find out and we can deleted.

In this case only this sums small competitions which are pointed by we have to do and then we can find out the fault case.

(Refer Slide Time: 55:25)


Concurrent Fault Simulation: Example

So now after doing this what we have done we have just said that these are the remaining gates these are the remaining gates and these are the remaining gates. Now there is a small thing you have to do so one by one and one does you have got so we have just converted so what do you can call that by concurrent fault simulation as we are retaining the information about the gates so we can just for the new change in input what we can do is that we without doing for any re-computation we can find out this one we can find out this one we can find out this one and we can find out this one with this is without any competition and here is some small competitions are required so we are use, reusing the information whenever the pattern is changing.

But now some new false will also get added for that obviously we have to do the re-computation so what are the advantage of concurrent fault simulation so whenever a new pattern is applied so what we are doing whatever is possible those information's we are retaining like this information we are retaining this information we are retaining this information we are retaining, but there is some new false will also get added for that you have to complete approach. So in concurrent fault in detected fault simulation we are remaining many enough you.

But in concurrent fault simulation we are remembering whatever is possible. Write for example, I will tell you what are the new gates that are adding so now you are applying a 0 and a 1, so obviously what are the new fault that is going to be added is a stack at 1 fault over here that is i1 stack at 1, so it is i1 stack at 1 then what is the signal value it will be a 1 over here it is 0 and the answer is 0, so this is 1 for which has to be added this stack at 1. Previously you were applying 11 so obviously we the no question was stack at 1 was here.

Now you are changing the value for 11 to 01 so obviously one stack at 1 for and this one will be apply. Now you see so now this is a new false so now in this case previously you were applying previously if you remember you were applying11 and the answer was a 1, so in this case you were retaining a stack at 0 false now because you are changing the value from 11 to 01 so obviously the answer here the 0 so obviously 1 stack at 1 fault as og1 will be added over here.

So in this case it is 00, 01 but here the instead of 0 the answer is 1, so two more gates have been added and obviously these two false are detected here. and you have to the see that these are all re-computed for here did the concurrent fault simulation and detected fault simulation are more difference you have to again re-compute all this thing, but where we has saved we have saved here we have saved here and we have saved here so whatever is possible we are remembering. Similarly for this gate if you check some new gates will be added so what is the new gate that is added so now obviously you can see that this is stack 1 fault here this is actually 1 this s tack at 1 fault over here.

So with stack at 1 false then the answer is 1 over here so there will be a signal change here so this gate will be added by this one, so stack at 1 here means the answer is 1 over here so instead of 0 the answer will be 1, 110 so this stack at 1 for is a new fault which is added which is get retained over here then actually 0g1 so this one was a new gate this one that is stack at one so initially it was stack at 0 so this new gate is also added over here so it is 11 the answer is 0 and of course previously I think in previously when you are applying 00 sorry previously you were applying 11 so the answer was something whatever.

So now here it which was stack at 0 in the previous case, now you have to apply because here it is 00 the answer is 0 so a stack at 1 fault at this net will be applicable over here. So these are the new gates this one with because of this stack at 1 fault which is newly added over here because the change in pattern so these are the two gates which is that added and now in this case the answer is 0 over 00 the answer 0 over here so you can add a stack at 1 fault here the o3 stack at 1 input is 00 the answer is 1 because the net is stack at 1.

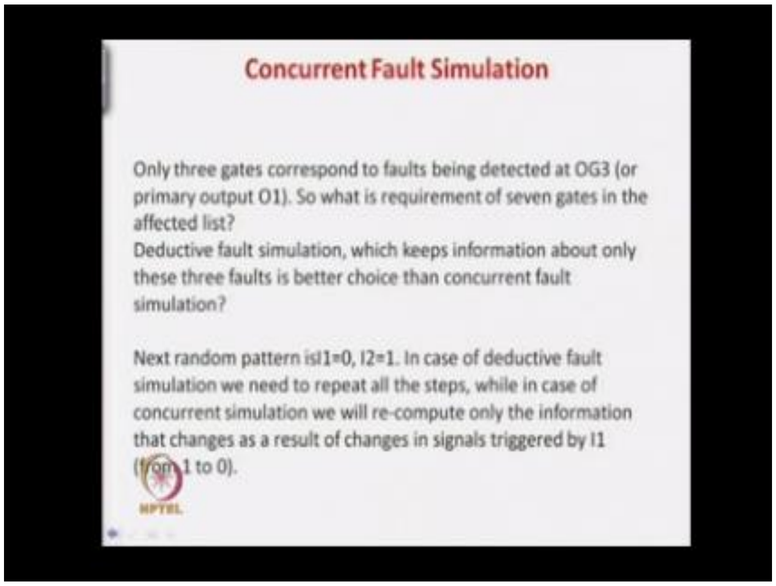
So three gates get added over here and so you can see that because of this pattern so the output here is a 0 so fault can be detected this stack at 1 fault can be detected over here and this and these two fault that is a of stack at 1 this fault is detected over here that is of stack at 1 and i1 stack at 1, so these three false are again detected by the pattern 01, so these new gates are again totally based on re-computation.

But still what is the advantage that what are the remaining gates this information you can retain from your previous pattern that is 11, and whenever you change the pattern to 10 so computation regarding the remaining gates are very minimal or n on computation is required you just get the values only for the new gate you have to redo competition. So concurrent fault simulation were deducted from the advantage is that you retain whatever is possible.

In deduction fault simulation you get nothing, but here you try to retain something as much as possible. But still then what is the penalty we have we are paying that some of the gates like for example the date gates you remember. So this, this gates have no affect like this gate, this gate, this gate, actually we are just keeping it, we are just carrying the data you can think.

There was not having any affect they could not deduct any fault in the previous case also so not in the present case. But still we are keeping this information because there may be reuse in some other cases. So that is because by carrying these dead animals you can think that some computation is safe. That is what is the idea so here also you may not know whether this gate could lead to any kind of fault deduction but still you are retains the because,

(Refer Slide Time: 01:00:51)



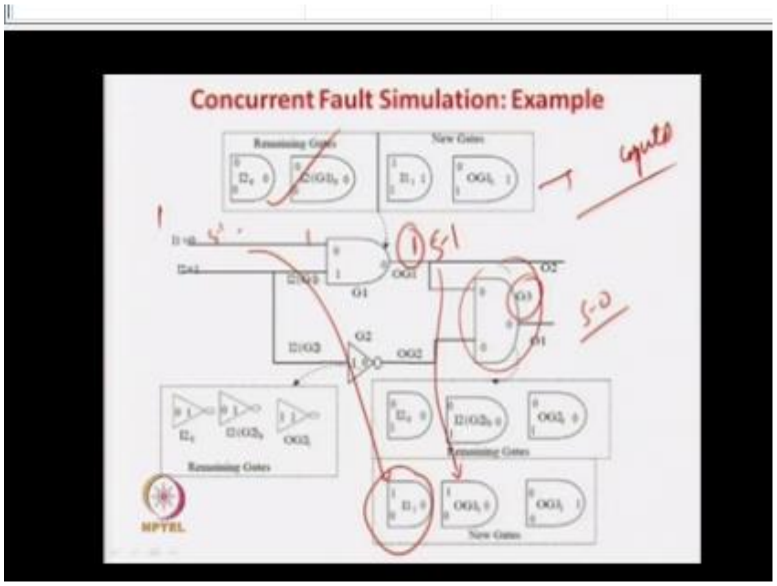
Concurrent Fault Simulation

Only three gates correspond to faults being detected at OG3 (or primary output O1). So what is requirement of seven gates in the affected list?
Deductive fault simulation, which keeps information about only these three faults is better choice than concurrent fault simulation?

Next random pattern isI1=0, I2=1. In case of deductive fault simulation we need to repeat all the steps, while in case of concurrent simulation we will re-compute only the information that changes as a result of changes in signals triggered by I1 (from 1 to 0).

We feel that sometimes it may be possible to do this one. So that is basically the idea of concurrent fault simulation okay. Right that is what is the case so now what was the decision summaries so only three gates correspond to fault been deducted at this one. So what is the reason requirement of seven gates in the affect case? That is the very important thing that is if you look at this example, so in this case that say,

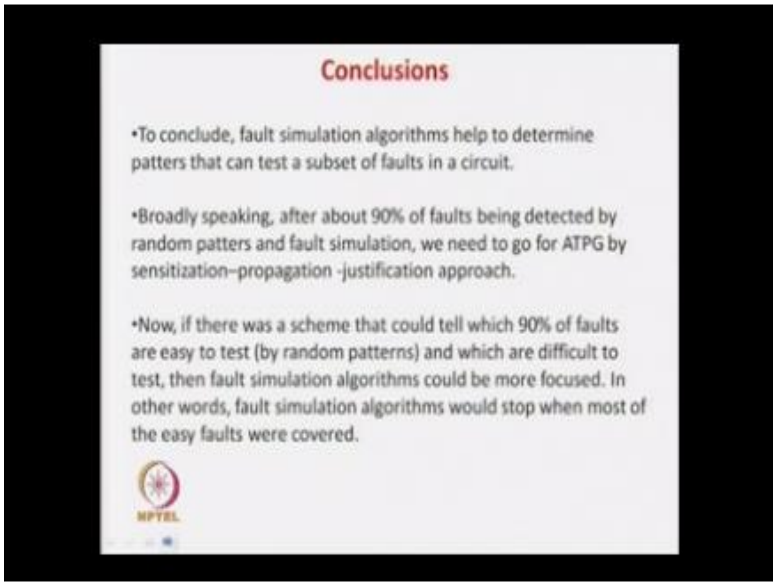
(Refer Slide Time: 01:01:18)



Concurrent Fault Simulation: Example

So only this gate, this gate, and this gate has a output difference with this one. Then why are you carrying these dead animals so that is question that is asked over here. That only three gates deducted this fault then why are you carry all this the idea over here was that whoever there is a change in the random pattern then those dead gates may help in creating some other faults which we have seen. And then this recompilation is seen that is what is the idea of a concurrent fault simulation over a deductive fault simulation. We get the fault simulation we do not carry anything what we do we do not carry anything.

(Refer Slide Time: 1:02:11)



Conclusions

•To conclude, fault simulation algorithms help to determine patters that can test a subset of faults in a circuit.

•Broadly speaking, after about 90% of faults being detected by random patters and fault simulation, we need to go for ATPG by sensitization–propagation -justification approach.

•Now, if there was a scheme that could tell which 90% of faults are easy to test (by random patterns) and which are difficult to test, then fault simulation algorithms could be more focused. In other words, fault simulation algorithms would stop when most of the easy faults were covered.

We just for a pattern we find out what are the faults of the deductible in one go that is why the fault list will have only those elements which are deductable and then you forget everything then we form new pattern. But in case of your concurrent fault simulation what we do we even if we compute some doubles for all the gates, the bubbles for all the gates are having some values. If there is a single difference in the gate we keep that fault gate in the bubble and we retain all the gates at all the level. Even if they deducted some faults or not deduct the faults.

Now whatever faults that are deducted we are very happy we can drop them, but this date gates that is there are having some signal difference with some gates. But they are not able to deduct they faults still we are retains them. When you are giving And or input, and random pattern then that new random pattern do that is dead gates may help to get some faults. Then those re computations are safe, so that is what some saving some re computation but still we have to carry the date load of some gates in the memory.

So that so that is the advantage and disadvantage, so that was about the fault simulation so to conclude we can say that concurrent fault simulation helps in diagram dis pattern. We apply some random patterns and then we find some what are the faults which can be deducted by them.

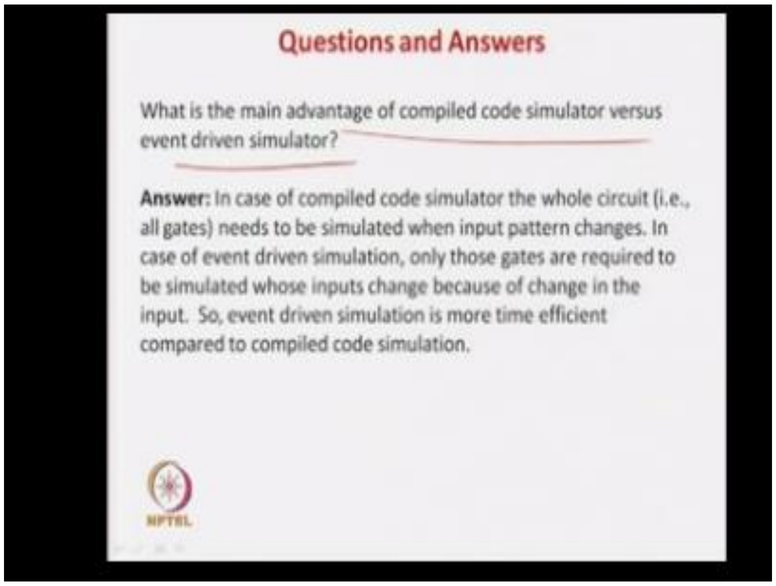
So you can use serial which is the 2:1 you can have parallels or computer support system lot of parallels then you go for parallel for simulation otherwise you can go for deductive fault simulation or concurrent fault simulation.

If we are using deducting fault simulation in one go we will find out what are the faults deducted by a pattern. But you do not remember anything next pattern you have the you have to redo everything. But in con current fault simulation you retain some information that can be reused but for that you have to sometimes carry to get node of gates. Which are not helpful in any kind of fault for the current pattern? But for the next pattern they may do some so you have to restrain them.

But now we are we see that experimental is seen that this procedure of random pattern is fault generation or whatever can help only for 90% of the fault, reaming 10% of fault are very hard to deduct faults and that you have to do by what you have to go by random the syndicate propagators and justify approval. So in the next lecture what we are going to see, we are going to see if someone or some algorithm can tell you that theses 90% of the faults are is easy to test faults. So apply random pattern faults theses 10% of the faults are defeated. So do not drive forward so that it will be very good so at least you can go for fault simulation we will not touch the 10 difficult faults.

We will only touch the 90 faults but her nobody tells you see which this easy fault which is difficult fault is then you are in a blind case. Even for this 10 difficult test faults you are applying the random pattern and you are failing ever time. For all the random pattern you are not able to look within a test pattern faults so next class we will see some kind of.

(Refer Slide Time: 01:04:29)



Questions and Answers

What is the main advantage of compiled code simulator versus event driven simulator?

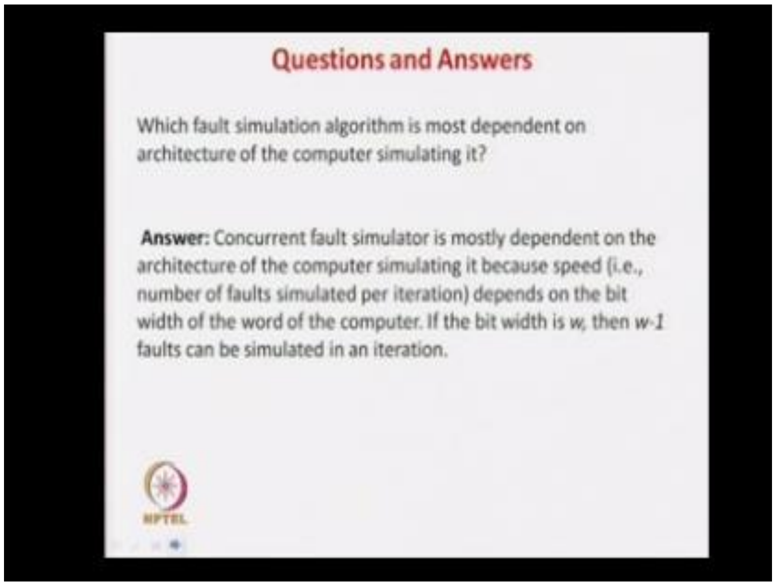
Answer: In case of compiled code simulator the whole circuit (i.e., all gates) needs to be simulated when input pattern changes. In case of event driven simulation, only those gates are required to be simulated whose inputs change because of change in the input. So, event driven simulation is more time efficient compared to compiled code simulation.

Examples of algorithm like type algorithm which we call it will tell you which easy faults are and which are difficult faults. And you only apply random pattern for the easy faults and you keep up a side that difficult faults for since there propagation and justify. Now before we conclude let us come to the question answer session so we are saying that what is the main advantage of compiled code simulation versus event driven simulator. So here what is the answer so we have already discussed main times like now we are having for concurrent fault simulation deducted for fault simulation?

And for normal fault simulation and normal circuit simulation mind so in com pile simulation we have one way circuit into our program anyway execute for our pattern. Then we forget everything you just change everything then again you have to redo everything but in what you call even in simulation we remember whatever we was done in the previous. And only those changes for our fault for our pattern whatever are the small changes in some reasons of the circuit only there we do the re computation.

So what are advantages and disadvantages obviously for event simulation it is faster because you recomputed anything which is redone there. Where the same time you rename the values from the previous intimation so you have to have some extra thing in memory, but gain is p because we are avoiding re computation.

(Refer Slide Time: 01:05:38)



Questions and Answers

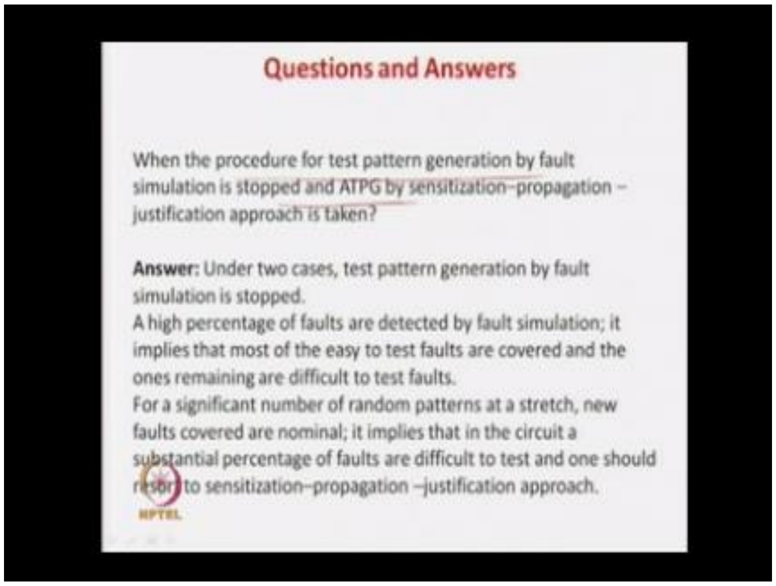
Which fault simulation algorithm is most dependent on architecture of the computer simulating it?

Answer: Concurrent fault simulator is mostly dependent on the architecture of the computer simulating it because speed (i.e., number of faults simulated per iteration) depends on the bit width of the word of the computer. If the bit width is *w*, then *w-1* faults can be simulated in an iteration.

Second question is among the entire four faults simulation algorithm which you seen like serial parallel deductive and concurrent which depends mainly on the computer architecture. So among them we have seen only the parallel fault simulation is depended on architecture because to net you assign a array and the length of the array is processor parallel. So you have to think about big parallel so you can computer to be parallels so you can maximum to be array in all the inputs.

And for all the other case like serial deductive and concurrent we do not at all think about the memory of the computer and we do not think about the parallels of the computer with the think that the faults sixe what are affected gate size and accordingly we attached to each gate or attached to the fault. So that is depends nothing on the parallel of the computer it is just the it is just a program. And we think that what the case is,

(Refer Slide Time: 1:06:25)



Questions and Answers

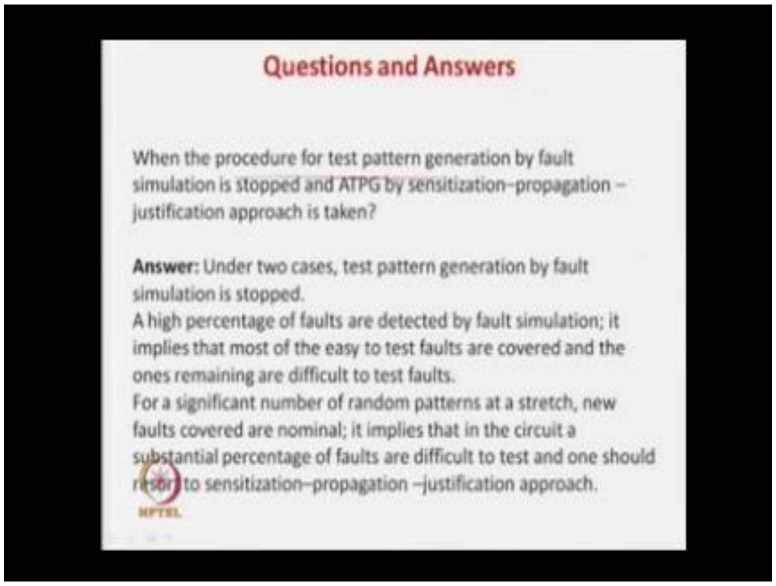
When the procedure for test pattern generation by fault simulation is stopped and ATPG by sensitization–propagation – justification approach is taken?

Answer: Under two cases, test pattern generation by fault simulation is stopped.
A high percentage of faults are detected by fault simulation; it implies that most of the easy to test faults are covered and the ones remaining are difficult to test faults.
For a significant number of random patterns at a stretch, new faults covered are nominal; it implies that in the circuit a substantial percentage of faults are difficult to test and one should resort to sensitization–propagation –justification approach.

We computer that for this fault gate the fault gate we have to add that is fault version you have to adds. For this gates this is the fault test and just we keep it we do not think what is that architecture what is the parallels of the computer and that many fault list we have to add nothing like that. So gate this gate as so many gate in the fault we just skip rhythm, but in case of parallel fault simulation only those number of patterns or these number of arts can be operation that much parallels is supported by your computer.

So that is among all the four faulty algorithm parallel fault simulation is mainly the architecture the last question is when.

(Refer Slide Time: 1:06:59)



Questions and Answers

When the procedure for test pattern generation by fault simulation is stopped and ATPG by sensitization–propagation – justification approach is taken?

Answer: Under two cases, test pattern generation by fault simulation is stopped.
A high percentage of faults are detected by fault simulation; it implies that most of the easy to test faults are covered and the ones remaining are difficult to test faults.
For a significant number of random patterns at a stretch, new faults covered are nominal; it implies that in the circuit a substantial percentage of faults are difficult to test and one should resort to sensitization–propagation –justification approach.

For test generation for random this one is stopped and ATPG by sensational algorithm like propagation sensor is taken. That is we start up apply your random apply test patterns and find the 10 faults and deducted and faults are keep on going for. And after some time you have to stop that what will you stop criteria taping criteria is when we find out the and applying some random patterns the number new faults which are deducted is very, very less or sometimes even 0. Then you apply your random pattern number of faults deducted in the initial will be set next random in 12 and random in 8 you keep on doing.

But after some time we find that the number random patterns we are applying if the number of new fault is 0 or very, very less so after that you can say that we have to stop because I am having random and it is not helping in anyway and I have to go for sensitized propagator and justifier approach. So thank you and we come to the end of the lecture on fault simulation in the 3 hours lecture was on that and the next class we will see the can somehow we find out that is difficult to test them and it is easy to test faults. So difficult to test faults we will go for sensation propagator and justifier approach and easy faults we will go for random pattern. So with this we come to the ends thank you.