

**Design Verification and Test of Digital VLSI Designs**  
**Prof. Dr. Santosh Biswas**  
**Prof. Dr. Jatindra Kumar Deka**  
**Indian Institute of Technology, Guwahati**

**Module - 8**  
**Fault Simulation and Testability Measures**  
**Lecture -3**  
**Fault Simulation - 3**

So, today will be covering the third lecture on fault simulation. So what we have discussed in a last two lectures that for the defining out define of test patterns for a stuck at faults there can be two basic approaches. In one approach we sensitize the fault propagate the affect and then justify the lines which is called the sensitization propagation and justification approach.

Then we have found out that these actually the algorithm is not a very simple algorithm because you can see that first you have to sensitize then eliminate of this propagate and then you have to justify and it will be for all the stuck at faults and sometimes they will be requiring requirement of iterations because sometimes propagation or your what you may call the justification not be successful.

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

So if we find that there are some k number of faults detected by that then you drop those k faults and then you take a another random pattern then keep on doing it till you find out that I mean a new random pattern is not able to detect some sufficient less number of faults. So the idea is better than the previous approach because you can see that that I one pattern detecting k number of faults so that can be found out by a random fault stimulation when followed by fault stimulation in one group.

(Refer Slide Time 02:15)



So you did not repeat sensitize propagate and justify for this  $k$  faults. So I mean what you can do this only for easy to test faults approve the process saturates and then you have to obviously go back to sensitize propagate and justify approach and then what we have seen that we have seen different kind of fault stimulation for example, so what is the basic idea of fault stimulation was that a we have normal circuit and we have a fault circuit and then we find out whether a pattern creates a behavioral difference at the output with the fault.

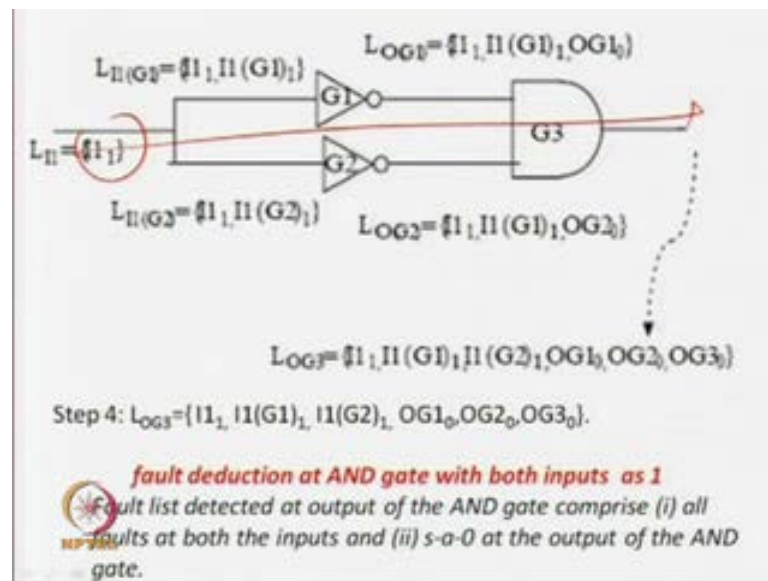
So we have seen very simple serial fault stimulation which is which was the very simple one in which case what we do we take up fault and then we they which we take a pattern and then find out whether one fault is detectable or not and then if we detectable it is a what you called you returning job the fault and we try with another fault and keep on doing it then you apply a another random pattern and so forth.

But actually the idea here was that it was very slow because we are going for serial based approach one fault one pattern one fault then next fault and next pattern and so on till you cover all the faults and then process repeats for the next test pattern. So we have seen that this actually bit slow then when you have gone for parallel fault stimulation in which case for each lying we have a array which represent some  $n$  bits or which is the depending on the parallel approach computer you can have a  $n$  bit array or a two  $n$  bit array something like that one bit represent the normal circuit and the other all other bits

of the array represent one faults for the cases and then we can applies is random pattern and then we can find out how many faults can be detected by the random pattern and include windows size or array size is n then we can introduce the reasoning about n minus 1 fault because 1 bit is for the normal circuit that we have discussed. So that parallelism I mean analysis your algorithm by a factor of n minus 1 if n is the your window size or your array size.

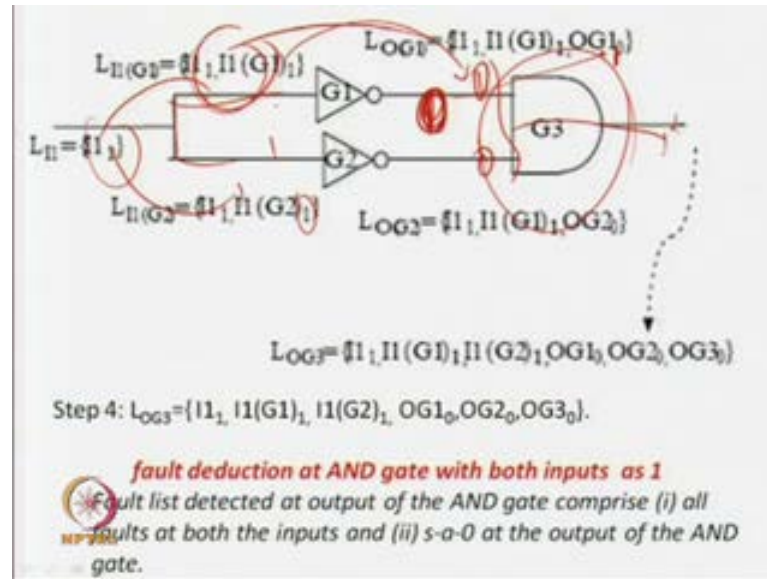
Then we have seen that in this case what is the idea that if you computer bit processing or what do you call the parallelism of the processor is low or a mini it is only 32 bit and also it there is some 1, 1000 faults so at least you have to repeat it 1, 1000 by 32 sealing factor that time number of times you repeat it what do you call this a parallel fault stimulation.

(Refer Slide Time 04:03)



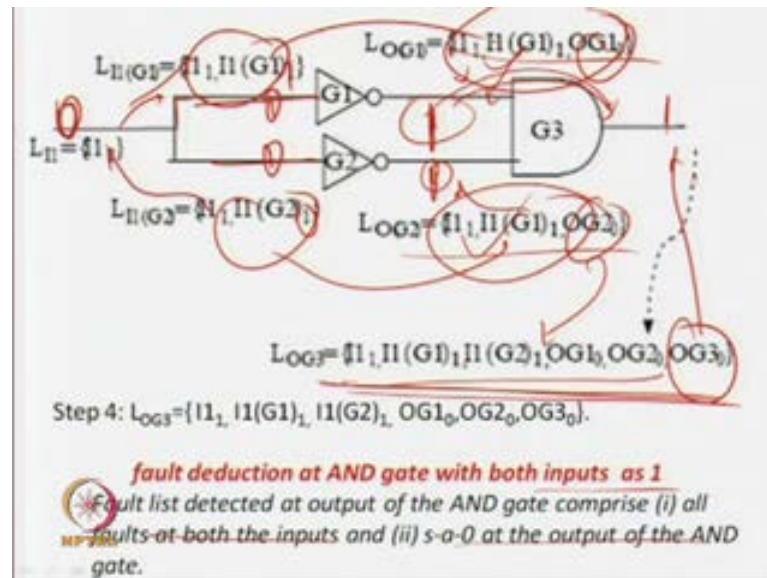
So then we found out then we started discussing about another algorithm in which case can we do the whole thing in one group, that is we apply the one random pattern and in one scan of the circuit can the reduce the results about all the faults can be detected by this then we saw that detective fault 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 and this one we find out what are the fault or we find out the fault least that is it is detectable by that random pattern and then actually you can in one group find out which are the pattern detectable by the fault.

(Refer Slide Time 04:41)



Then we started finding out some rules like for a example, if it is a not get then what happen is the fault list here you propagate the value here and then if the value here is a 0 then we have to also add as sorry if the value here is a 1 in you have to add a stuck at 1 and this net and if the here the inputs signal value is 0 then you have to apply a stuck at 1 fault here and the whole list on here propagates here then it is a a fan out array then whatever here get propagate here and here in addition to if it is a 1 signal here then it is a stuck at 0 and vice versa and there similarly, we have seen the rules for and AND gate like if the cases 1 1 then whatever fault least at both the nets will be add here last some stuck at 0, stuck at 1 and the output then if it is a 0 0 then we have found out a only the common set of faults in between this can propagate to the output. So we are starting out some kind or rules in detective fault stimulation which can basically you scan the circuit from input to the output you can find out or the false can be detected that for that we require some rules.

(Refer Slide Time 05:42)



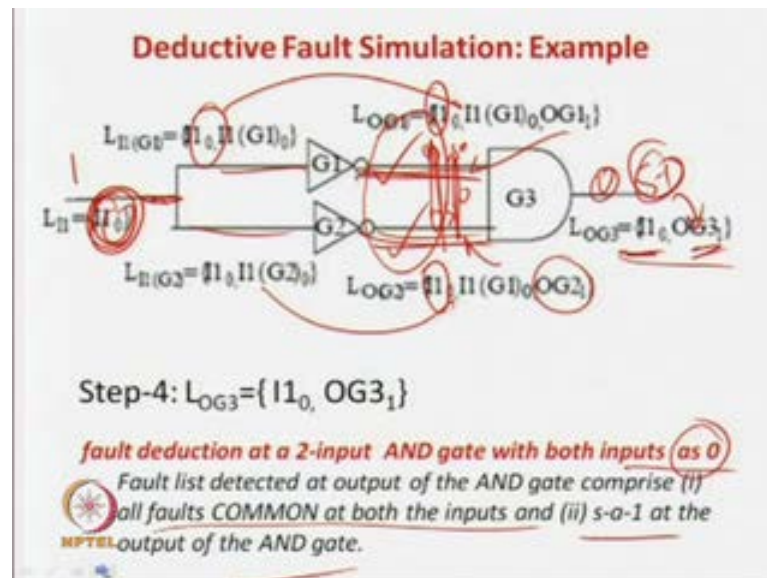
So in the last lecture so, we covered some rules like for example, if is fan out net then whatever fault is here propagates here. So you see this least here and this least here then you are applying a random pattern 1 over here and sorry you are here applying a random pattern of 0 over here. So the pattern apply is a 0 over here so the detects the stuck at 1 fault here similarly, is also here and the value is 0 0 over here signal value it is a applying a 0. So stuck at 1 is detected at this net another stuck at 1 is detected at this net and for the inverter what is the rule we saw that whatever here and whatever here both of the both the this will be here.

Now this is the value is 0 0 here so it is 1 1 here so, some stuck at 0 sorry so this is the stuck at 0 that we rejected here and this is stuck at 0 that is detected here. Now you know that if are the AND gate what we have seen the rule for the AND gate there will be four rules so for 0 0 1 0 1 then you have for 1 0 and then for 1 1 so four so for all the four cases of your AND gate you have to have the rules. So in this case is 1 1 so you know that in AND gate when both the inputs are 1 then if I means if this is input 1 then whatever fault effect will propagate similarly, this is 1 1 here so whatever is the fault effect can be propagated because 1 is the non-controlling input of AND gate but, if it is 0 you know that nothing propagates the other end.

So in this case both the values are 1 1 signal value because of the random pattern 0. So whatever the least over here and whatever the least over here the whole thing you know

that it will come over here. So you can if you study with this least and this least so the whole least will be actually taking care of this one and this one because we are applying 1 1 and 1 1 is signal value output this 1 so you can also detect a stuck at 0 fault at this net. So this becomes your what you called the pattern I mean this are the fault least for this stuck at for this random pattern 0 you can detect this n number of faults. So we found out that the rule for AND gate when both the inputs are 1 then all the faults at both the inputs of the input and stuck at 0 at the output of the AND gate this is the rule for AND gate when everything is 1 then we have taken the case when the output here was a 1.

(Refer Slide Time 07:41)



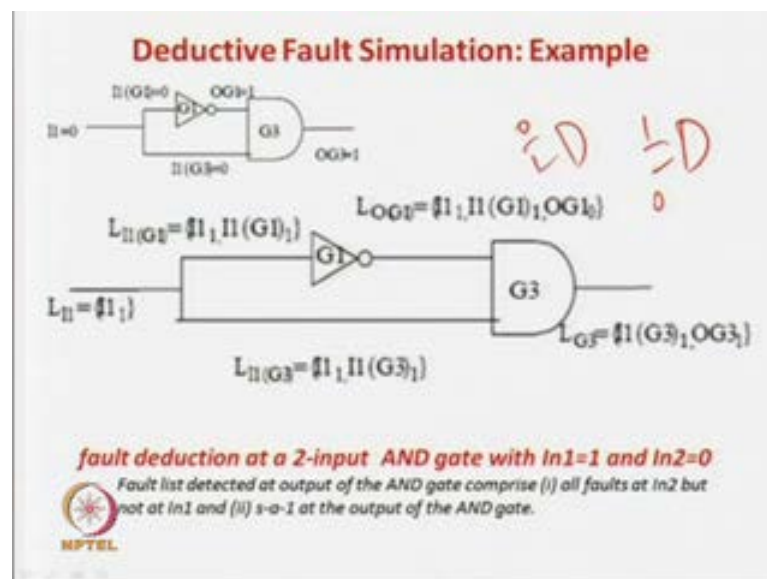
So you can a take stuck at 0 fault over here stuck at 0 fault at this net at this net and this will be 1 over here. So you get a 0 over here so the whole least comes here because of this inverter and then you can at the stuck at 1 for at this net you can at a stuck at 1 for this net. Now we have also discussed in the last class at the both the AND gate inputs are 0 and if the AND gate input is 0 1 input is 0 in AND gate we know that the fault cannot the propagated by the other end. So what will be found out that only one fault which is common to both of them.

So this fault is common to both of them so this is landing up an a stuck at 0 over here and stuck at 0 over here. So that is the only fault which can be detected at the output that we have seen that only for which is common to both of them it can be a detected over here

and of course, you get a 0 and 0 over here. So stuck at one fault at this can be detected so both this the this is the fault common that is the stuck at 0 fault at this line is affecting both this input and this input this is common.

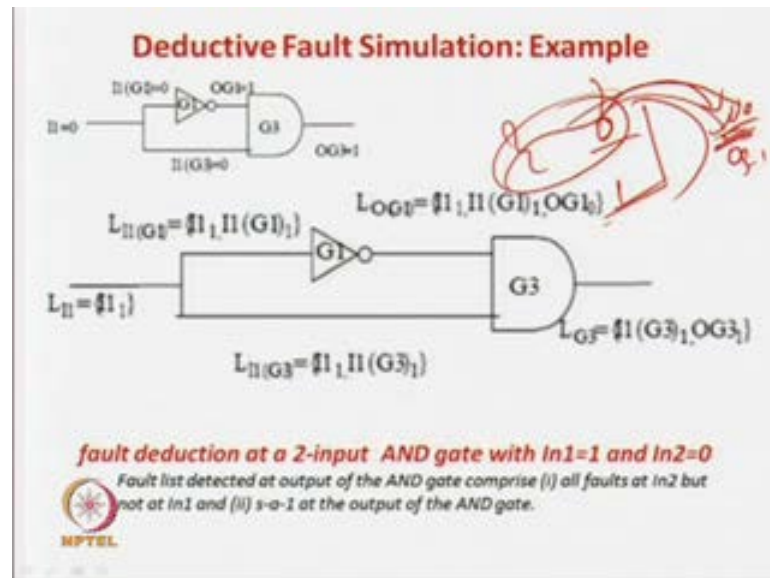
So that is the only thing will propagate here and other things not be propagated so the rule is when both the inputs are 0 then we thought that input will be nothing be propagated because a if you any 1 gate of the input is 0 nothing propagates to the AND gate but, we where the surprise to find out that in this case if some output will be there is affecting both the inputs of your AND gate then that is the only fault that is propagated. So fault least at the AND gate comprise and both of the AND gate inputs are 0 all faults which are common to both the inputs so this only this fault was common to both the inputs and that get propagates faults a stuck at 1 fault at the output is obviously detected because 0 and 0 here which was 0 over here.

(Refer Slide Time 09:24)





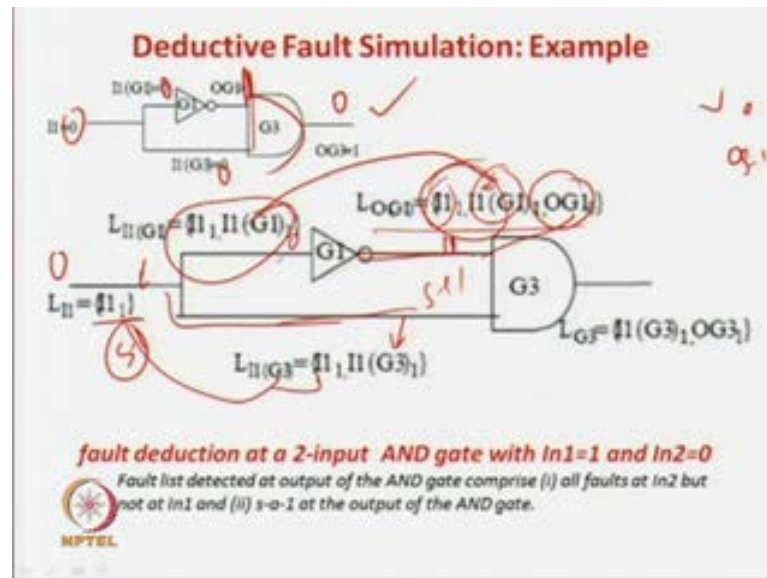
(Refer Slide Time 09:35)



These rules we have seen in the last class. Now there are two more rules which remain that is for an AND gate that is a 1 0 and AND gate 0 1. So will see what are the rules for this thing. So I mean this will be this pretty obvious I mean in your class which should be obvious because you see whenever this is 0 and a 1 so what our institution say that will follow. So obviously, if this is one's of this whatever is here will propagate over here so whatever is fault least here can be propagated here as well as the output is 0 so obviously, 1 stuck at 1 at this net is also detectable. So idea is that so, whatever is the fault is at the same can be propagated from here by detecting fault stimulation rule and also this is 0 1 so the here will get the value 0.



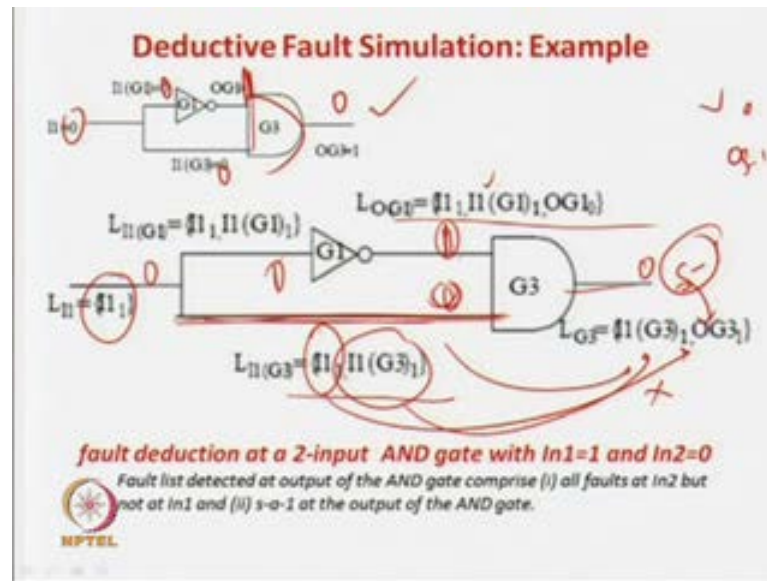
(Refer Slide Time 10:09)



So stuck at 1 fault the output is also detectable so that is intuitively 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 an inverter or not so we put a 0 in the input. So if it is 0 so you know that this input is 0 so this is going to be 1 this is 0 the answer is 0. So now in the input to the AND gate this 1 is 1 0 so let us study the rule so we apply a 0 over here so stuck at 1 fault is detected at this net so it is this one.

Now (( )) the fan out rule so, this is a 0 and this is 0 so obviously this one gets propagated here and this one also gets propagated here sorry and then a stuck at 1 fault at this net this net is here and a stuck at 1 fault at this net is reflected over here. So this is the case now by the inversion in inverter so we know that whatever fault list here will get here. So we have this one and this one which is propagating from the end. So these are these two will get propagated over here because of the inverter and here we had a 0 over here.

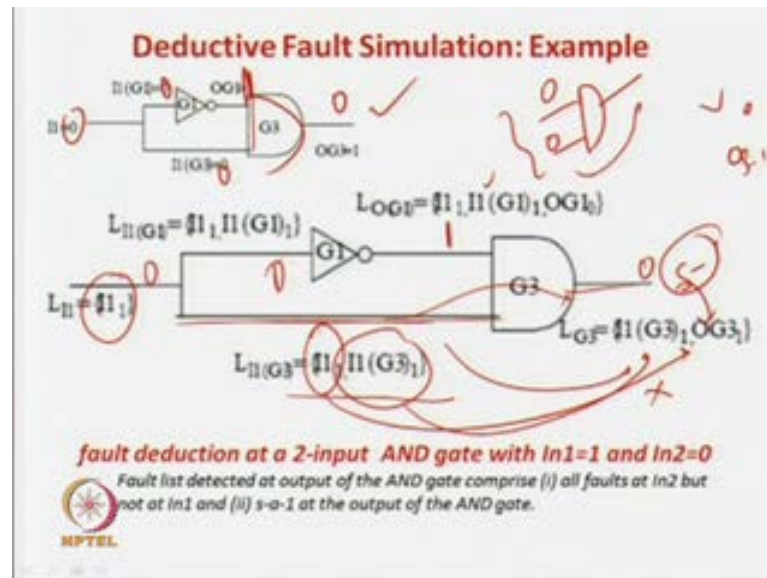
(Refer Slide Time 11:19)



So the answer is the 1 over here so, a stuck at 1 0 1 so stuck at 0 fault at this net is also detectable. So this a new thing that is added and these two are propagated from the input of the inverter. So now what is the status? So the status here is let us see what is the status, so the status here was the signal here is a 0 so this is a 1 this is a sorry this is a 0 this is a 0 and this is a 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 is a this net is 1 and this net is 0 so whatever I mean a whatever is available here will propagate here.

So this is a very well-known fact so I mean what happens is that let us see what happens so in this case so whatever fault that will actually impact this net can be propagated over here because there is 1 over here. So that is what we are expecting and of course, this is a 1 and a 0 so the answer is 0 so here is a stuck at 1 will be rejected at the net so this is here so we expect that i 1 this 1 will be propagated here. So this is here but, you see this fault should also had been propagated here at which is not the case.

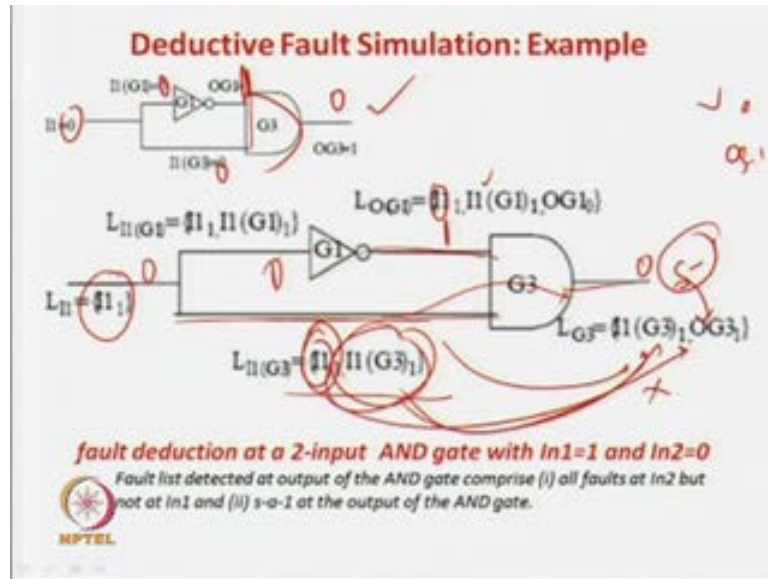
(Refer Slide Time 12:21)



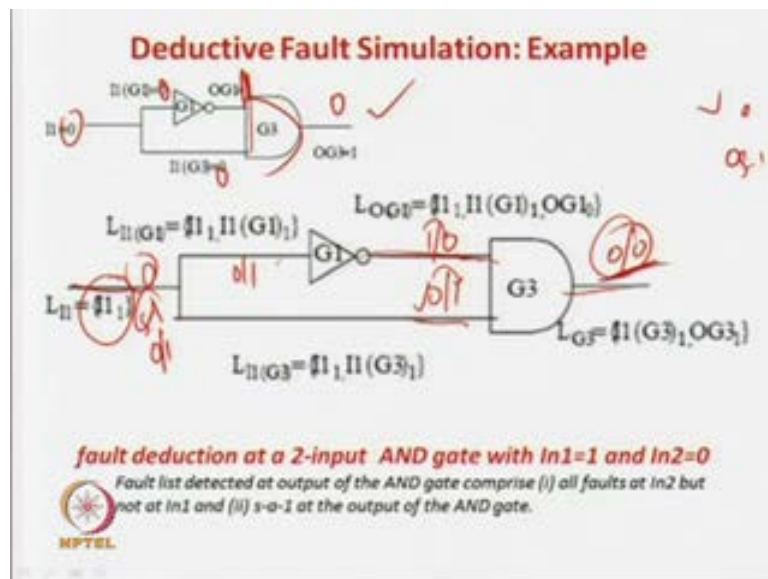
So why this fault that is 1 1 why this fault should be propagated? This fault that is stuck at 1 at this thing why it should be propagated? Because we failed that in a AND gate what happens so this net is this net value is a 1. So as this is the that's what have any mean this is 1 and then everything from here passes from here to here so the whole layer should propagate but, there is a problem. So whole for layer 1 propagate as our expectation was that in this case if it is 0 0 our expectation was that nothing would propagate here what we saw that only those faults which are common to both of them propagate similarly, in this case slight deviation from the intuition there our intuition says that this net is 1.

So whatever fault propagate so that is why this is propagate this is true but, you have to observe one thing that if something this one is common to here and here so that common part you cannot propagate. So the let us see why it is not propagatable so you can just see why it is not detectable. So let us study the case. So in this case it is stuck at 0 so the 0 so this stuck at 1 is detectable over here. So this is a 0 normal case fault 1 case so 0 1 0 1 so it is 1 0 because of the inversion and your output will be 0 slash 0. So this fault is not detected this stuck at 0 results to a 0 0 here this is not detectable.

(Refer Slide Time 12:50)

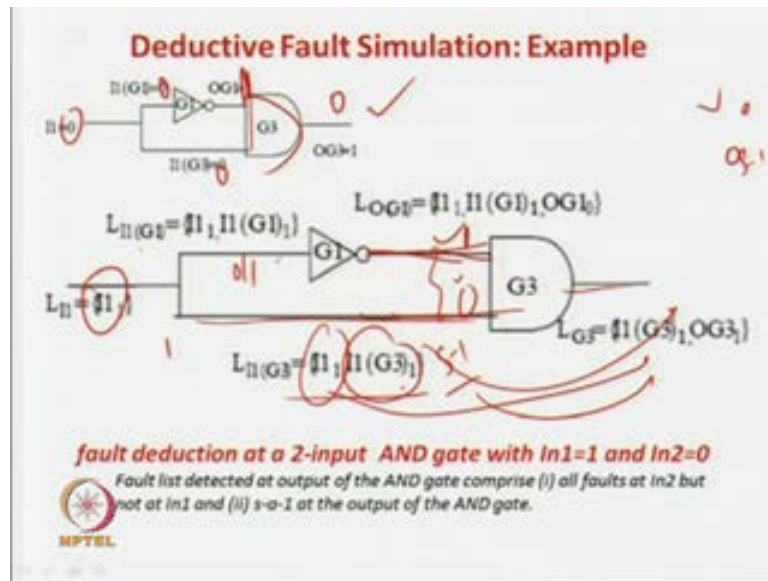


(Refer Slide Time 13:08)

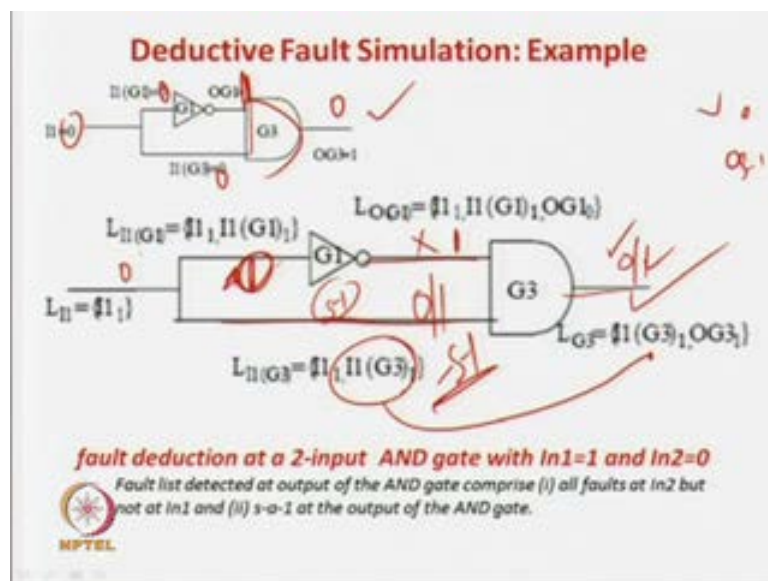


Now why this is not detectable because this stuck at 1 fault at this net impact this net by making a 0 to 1 and also this net by making from a 1 to 0. So what is happens output is 0 0 that is normal case also 0 failure case also 0 so it is not detectable because this stuck at 1 fault actually these affecting both this value is 1 and this value is 1 both this net as well as this net. So even if this is a 1 and the this fault this 1 should whatever is in this case should propagate over here but, as this fault actually is affecting both this nets.

(Refer Slide Time 13:46)



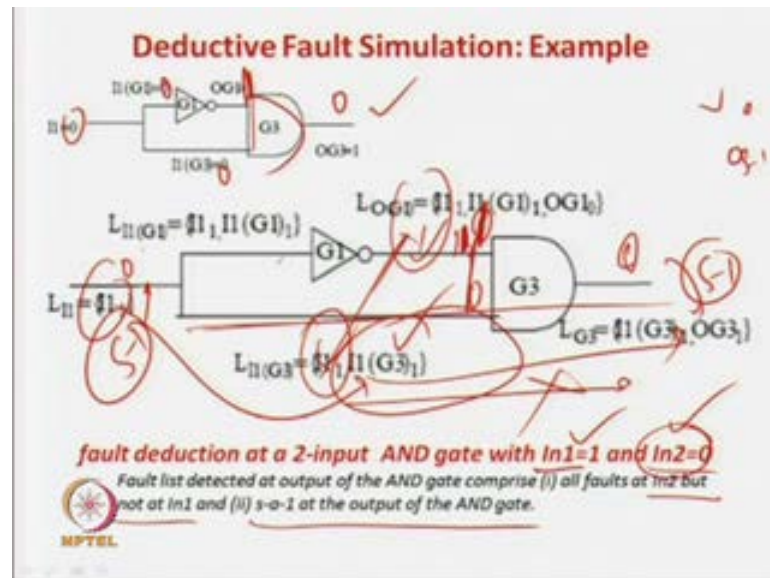
(Refer Slide Time 14:19)



So even if this is the 1 this thing not propagate but, this fault there is i 1 g 1 c 1 that is this net this net stuck at 1 easily propagatable over here because this is 1 this thing single value is 1. So whatever here is propagates over here and this fault does not have this impact so just you can this can be very easily verified. So I mean in case you apply a 0 over here so it is a signal is 1 it is a 0 so it is a this net is stuck at 1 this net so output is 1 in the normal case and here the signal is sorry it is a single is 0 over here so this is a 1 over here this is 0 1 so the output is 0 1.

So for this stuck at 1 fault at this net what happens normal case it is 0 fault case it is 1 so of the fault is detected at this net 1 so that means this net this is also propagated at this list now.

(Refer Slide Time 14:56)

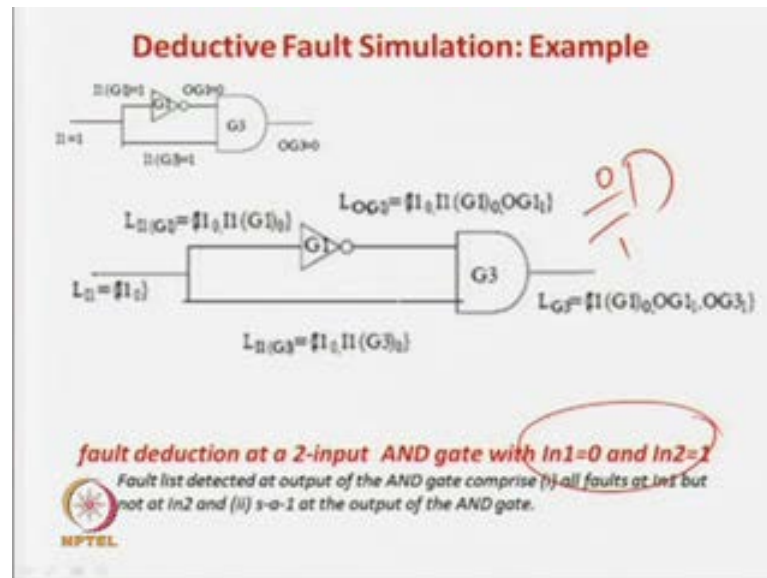


Why it is possible? Because this stuck at 0 fault sorry this stuck at 1 fault at this net has no effect in this line. So it is propagate this propagates to the input and output it can be detectable but, this fault that is a stuck at 1 fault over here at this point affects this net as well as this net both of the net is effected so even if this net is 1 and this fault this fault list is available over here it cannot propagate to the output.

So the rule is a fault deduction at 2-input AND gate with  $n1$  equal to  $i1$  equal to 1 and  $i2$  equal to 1 is all faults in  $i2$  obviously because this is a 1 so whatever fault in  $i2$  will propagate but, not in  $i1$  that is 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 then whatever fault list here which is not common to between these two will obviously propagate because these net is the 1 and whenever 1 input of AND gate 1 and other gate propagates and obviously stuck at 1 at the output of this AND gate is detectable because this is 1 and this is 0 so the output is the 0 so any stuck at 1 fault at the output will be detected.



(Refer Slide Time 16:17)

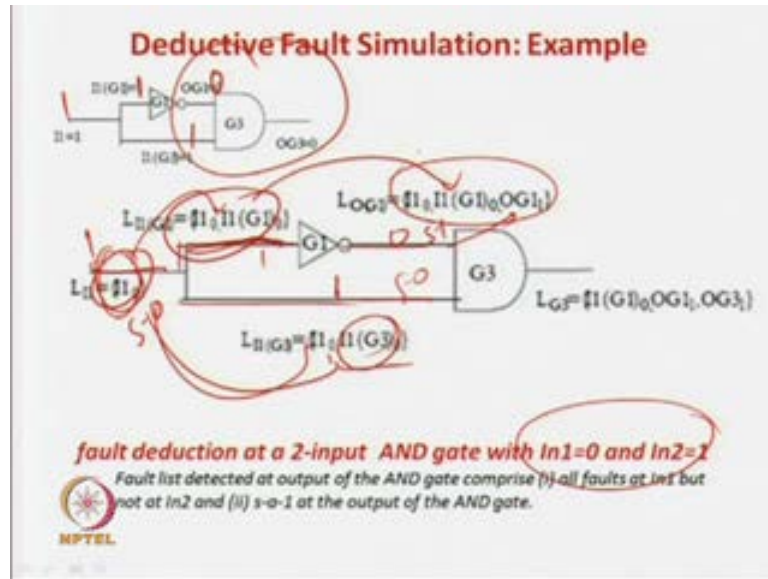


So the rule for a AND gate with one input stuck at one input as one and the other input is 0 so what is the rule? The rule is say whatever at the one gate is 1 one input of the AND gate is 1. So whatever you have the other fault list another input fault is always directly propagated because the other input is the 1 but, it is a common fault in between both the inputs of the And gate and then that fault is will not be propagated because it is affecting both the line so that is not the case. So with the same rule but, that is 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 the same just have a look. So this is the AND gate we are taking and by normal circuit we are taking in this case we have now the random pattern is a 1.

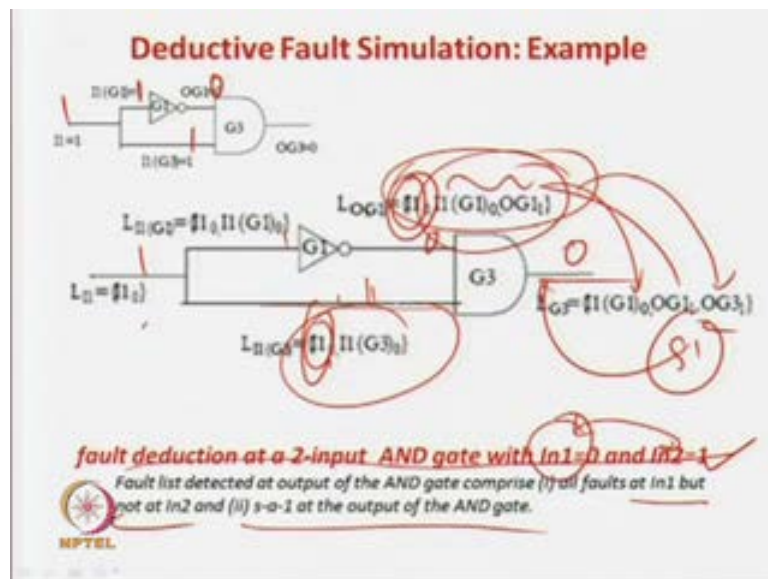
So the random pattern is a 1 means it is a 1 over here, it is a 0 over here, it is a 1 over here. So we can set the rule about 0 1 for the AND gate. So now as you are applying 1 over here so stuck at 0 at this net is detected so this you are going to have and in this net by the fan out rule. So this is a 1 1 this is 0 by the fan out rule this net this will propagated over here as well as a stuck at 0 fault at this net will be detected because this is 1 similarly, this one will propagated here this affect is propagated here by the fan out rule and this is one a stuck at 0 fault at this net is detectable over here.



(Refer Slide Time 16:31)



(Refer Slide Time 17:21)



Now this is what is the case. Now the because of the inverter the whole stuck will propagated over here at signal here is a 0 so a stuck at 1 at this fault is detectable by this one. So this three list we are having so finally, what we are going to having over here so finally, we are having you can see that it is a 1 over here, it is 1 over here, 1 over here, 0 over here. So this is the list over here and this the list over here. Now intuitively, you can say that at this line is a 1 whatever whole thing here will propagate here but, actually it is not be the case because this is stuck at 0 and stuck 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 means the answer is 0 over here so a stuck at 1 fault at this will be detected so this is added.

So we have these two faults propagated this is the common fault is not detectable is not propagated and 1 stuck at 1 fault at the output. So the rule here is again the same if this fault deduction at the AND gate in 1 0 and into 1 is 1 all fault at in 1 obviously whatever fault here list will be propagated because other input is 1 but, not in into but, if there is a common fault over between them they will not be propagated an a stuck at 1 and the output of the AND gate is also detectable.

(Refer Slide Time 18:14)

Fault deduction rules for logic gates

Gate Type	Inputs (In1 and In2)		Output (O)	Deductive Fault list at O
AND	0	0	0	$\{L_{In1} \wedge L_{In2}\} \cup O_0$
	0	1	0	$\{L_{In1} \wedge L_{In2}\} \cup O_0$
	1	0	0	$\{\bar{L}_{In1} \wedge L_{In2}\} \cup O_0$
	1	1	1	$\{L_{In1} \wedge L_{In2}\} \cup O_1$
OR (Dual of AND gate)	0	0	0	$\{L_{In1} \wedge L_{In2}\} \cup O_0$
	0	1	1	$\{\bar{L}_{In1} \wedge L_{In2}\} \cup O_1$
	1	0	1	$\{L_{In1} \wedge \bar{L}_{In2}\} \cup O_1$
	1	1	1	$\{L_{In1} \wedge L_{In2}\} \cup O_1$
NOT	0	---	1	$\{L_{In1}\} \cup O_1$
	1	---	0	$\{L_{In1}\} \cup O_0$
Fanout	0	---	0	$\{L_{In1}\} \cup O_0$
	1	---	1	$\{L_{In1}\} \cup O_1$

So we have seen that how the rules can be determined for all these cases. Now just let us tabulate this one the fault deduction rule for the and I mean logic gates. So that means what will happens in case of deductive fault simulation is that we apply a random pattern then we find out the rules that it at the input these are the faults which are detectable.

Then what we do? Then we find out there if there is a fan out or if the AND gate if there is 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 fault list at the output that we have to determine. So we need rules. So let us see what are the rules? So you can see here that AND gate the four cases 0 0 0 1 1 0 1 1 so these are the outputs inputs four forms is these are the outputs. Now you see so what is says that if both the inputs are 0 so you

have seen what is the rule both the inputs are 0 only those faults which are common to both of them will be propagated and at the stuck at 1 fault at the output will be detected.

So we write list of n 1 intersection list of n 2; that means, whatever is common in input 1 and input 2 that will be detected you e n o 1 that is stuck at 1 fault at the output of the AND gate is detectable. Now you see this case 0 1 so what we have seen? That is 0 1 means what you have seen.

(Refer Slide Time 19:30)

Fault deduction rules for logic gates

Gate Type	Inputs (In1 and In2)		Output (O)	Deductive Fault list at O
AND	0	0	0	$[I_{in1} \cap I_{in2}] \cup O_1$
	0	1	0	$[I_{in1} \cap \overline{I_{in2}}] \cup O_1$
	1	0	0	$[\overline{I_{in1}} \cap I_{in2}] \cup O_1$
	1	1	1	$[I_{in1} \cap I_{in2}] \cup O_1$
OR (Dual of AND gate)	0	0	0	$[I_{in1} \cup I_{in2}] \cup O_1$
	0	1	1	$[\overline{I_{in1}} \cap I_{in2}] \cup O_1$
	1	0	1	$[I_{in1} \cap \overline{I_{in2}}] \cup O_1$
	1	1	1	$[I_{in1} \cap I_{in2}] \cup O_1$
NOT	0	---	1	$[I_{in1}] \cup O_1$
	1	---	0	$[I_{in1}] \cup O_1$
Fanout	0	---	0	$[I_{in1}] \cup O_1$
	1	---	1	$[I_{in1}] \cup O_1$

So some faults actually in this case whatever fault at this list to be propagated that is in list of one will be propagated but, at the same time you have to 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 factor if there is some common thing in between these two that will not be propagated why? Because this input is 1.

So whatever here will be propagated but, if there is something common that will not be propagated and whatever fault is at the 1 will not be propagated because this input is 0 because this is the idea that this is 0 and this is 1. So fault list here will be propagated and as this is a 0 and this is a 1 the fault list here will not be propagated. So automatically it says that if there is a something common in between these two then that will not be propagated because whatever here is not allowed to be propagated.

(Refer Slide Time 20:29)

Fault deduction rules for logic gates

Gate Type	Inputs (In1 and In2)		Output (O)	Deductive Fault list at O
AND	0	0	0	$[I_{in1} \cap I_{in2}] \cup O$
	0	1	0	$[I_{in1} \cap \bar{I}_{in2}] \cup O$
	1	0	0	$[\bar{I}_{in1} \cap I_{in2}] \cup O$
	1	1	1	$[I_{in1} \cup I_{in2}] \cup O$
OR (Dual of AND gate)	0	0	0	$[I_{in1} \cup I_{in2}] \cup O$
	0	1	1	$[\bar{I}_{in1} \cap I_{in2}] \cup O$
	1	0	1	$[I_{in1} \cap \bar{I}_{in2}] \cup O$
	1	1	1	$[I_{in1} \cap I_{in2}] \cup O$
NOT	0	---	1	$[I_{in1}] \cup O$
	1	---	0	$[I_{in1}] \cup O$
Fanout	0	---	0	$[I_{in1}] \cup O$
	1	---	1	$[I_{in1}] \cup O$

So how we write this? We write in a very simple way we say that 1 1 all the fault is intersection 1 2 bar that whatever here will propagated, whatever here will not be propagated and obviously the output is 0 so a stuck at 1 fault will be also propagated or the output will also be detectable this is the propagation list in this the output which is fault at the output of the gate which is deductible because the output is 0 for the input this one.

(Refer Slide Time 20:47)

Fault deduction rules for logic gates

Gate Type	Inputs (In1 and In2)		Output (O)	Deductive Fault list at O
AND	0	0	0	$[I_{in1} \cap I_{in2}] \cup O$
	0	1	0	$[I_{in1} \cap \bar{I}_{in2}] \cup O$
	1	0	0	$[\bar{I}_{in1} \cap I_{in2}] \cup O$
	1	1	1	$[I_{in1} \cup I_{in2}] \cup O$
OR (Dual of AND gate)	0	0	0	$[I_{in1} \cup I_{in2}] \cup O$
	0	1	1	$[\bar{I}_{in1} \cap I_{in2}] \cup O$
	1	0	1	$[I_{in1} \cap \bar{I}_{in2}] \cup O$
	1	1	1	$[I_{in1} \cap I_{in2}] \cup O$
NOT	0	---	1	$[I_{in1}] \cup O$
	1	---	0	$[I_{in1}] \cup O$
Fanout	0	---	0	$[I_{in1}] \cup O$
	1	---	1	$[I_{in1}] \cup O$

(Refer Slide Time 21:11)

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	$[L_{in1} \cap \bar{L}_{in2}] \cup O_1$
	1	0	0	$[\bar{L}_{in1} \cap L_{in2}] \cup O_1$
	1	1	1	$[L_{in1} \cap L_{in2}] \cup O_1$
OR (Dual of AND gate)	0	0	0	$[L_{in1} \cup L_{in2}] \cup O_1$
	0	1	1	$[\bar{L}_{in1} \cap L_{in2}] \cup O_1$
	1	0	1	$[L_{in1} \cap \bar{L}_{in2}] \cup O_1$
	1	1	1	$[L_{in1} \cap L_{in2}] \cup O_1$
NOT	0	---	1	$[L_{in1}] \cup O_1$
	1	---	0	$[L_{in1}] \cup O_1$
Fanout	0	---	0	$[L_{in1}] \cup O_1$
	1	---	1	$[L_{in1}] \cup O_1$

(Refer Slide Time 21:25)

Fault deduction rules for logic gates

Gate Type	Inputs (In1 and In2)		Output (O)	Deductive Fault list at O
AND	0	0	0	$[L_{in1} \cap L_{in2}] \cup O_1$
	0	1	0	$[L_{in1} \cap \bar{L}_{in2}] \cup O_1$
	1	0	0	$[\bar{L}_{in1} \cap L_{in2}] \cup O_1$
	1	1	1	$[L_{in1} \cap L_{in2}] \cup O_1$
OR (Dual of AND gate)	0	0	0	$[L_{in1} \cup L_{in2}] \cup O_1$
	0	1	1	$[\bar{L}_{in1} \cap L_{in2}] \cup O_1$
	1	0	1	$[L_{in1} \cap \bar{L}_{in2}] \cup O_1$
	1	1	1	$[L_{in1} \cap L_{in2}] \cup O_1$
NOT	0	---	1	$[L_{in1}] \cup O_1$
	1	---	0	$[L_{in1}] \cup O_1$
Fanout	0	---	0	$[L_{in1}] \cup O_1$
	1	---	1	$[L_{in1}] \cup O_1$


Now for 1 0 you can also see that whatever here will be propagated fault list whatever here will not be propagated so that is 1 1 in bar and intersection 1 2 that is whatever here will propagated, whatever here will not be propagated. So any fault common in between in these two will also not be propagated plus a stuck at 1 at the output is also detectable because the output answer is 0. Now we have seen the best case was 1 and 1 if both the inputs of the AND gates are 1 then what is the idea o list over here, list over here all are detectable and this is a 1 so this stuck at 0 at the output will be detectable and whatever

list here and whatever list here. So it is 1 1 union 1 2. So both of them both the fault is will be propagated so this is for the AND gate.

(Refer Slide Time 21:33)

Fault deduction rules for logic gates

Gate Type	Inputs (In1 and In2)		Output (O)	Deductive Fault list at O
AND	0	0	0	$[I_{in1} \wedge I_{in2}] \cup O_1$
	0	1	0	$[I_{in1} \wedge \bar{I}_{in2}] \cup O_1$
	1	0	0	$[\bar{I}_{in1} \wedge I_{in2}] \cup O_1$
	1	1	1	$[I_{in1} \wedge I_{in2}] \cup O_1$
OR (Dual of AND gate)	0	0	0	$[I_{in1} \wedge I_{in2}] \cup O_1$
	0	1	1	$[\bar{I}_{in1} \wedge I_{in2}] \cup O_1$
	1	0	1	$[I_{in1} \wedge \bar{I}_{in2}] \cup O_1$
	1	1	1	$[I_{in1} \wedge I_{in2}] \cup O_1$
NOT	0	—	1	$[I_{in1}] \cup O_1$
	1	—	0	$[I_{in1}] \cup O_1$
Fanout	0	—	0	$[I_{in1}] \cup O_1$
	1	—	1	$[I_{in1}] \cup O_1$



For the OR gate actually for testing purpose or input purpose or the whatever purpose it is just the dual that is just the dual if you the dual rule you can find out. So if it is a 0 0 output is 0 so this is an OR gate. So if both of them are 0 the output is 0 so 0. So with those in case of an OR gate if we apply a 0 over here. So whatever the impact over here gets propagated because it is the dual of an AND gate in case of AND gate the 1 input is 1 then whatever happens is the other outputs get 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, in you know that if in an OR gate if one input is 1 then the answer is 1 and nothing is propagated.

So thus by the applying the dual rules will find out that if both of them are 0 then both the list will be propagated that is the union and of course, the output is a 0 so stuck at 1 fault will be detected 0 1 so here in this case we know that this this this is very helpful, this will allowed to you propagate but, this one stops. So whatever here will be propagated and whatever here will not be propagated this is an OR gate.



(Refer Slide Time 22:06)

Fault deduction rules for logic gates

Gate Type	Inputs (In1 and In2)		Output (O)	Deductive Fault list at O
AND	0	0	0	$[I_{in1} \cap I_{in2}] \cup O$
	0	1	0	$[I_{in1} \cap \bar{I}_{in2}] \cup O$
	1	0	0	$[\bar{I}_{in1} \cap I_{in2}] \cup O$
	1	1	1	$[I_{in1} \cup I_{in2}] \cup O$
OR (Dual of AND gate)	0	0	0	$[I_{in1} \cup I_{in2}] \cup O$
	0	1	1	$[\bar{I}_{in1} \cap I_{in2}] \cup O$
	1	0	1	$[I_{in1} \cap \bar{I}_{in2}] \cup O$
	1	1	1	$[I_{in1} \cap I_{in2}] \cup O$
NOT	0	---	1	$[I_{in1}] \cup O$
	1	---	0	$[I_{in1}] \cup O$
Fanout	0	---	0	$[I_{in1}] \cup O$
	1	---	1	$[I_{in1}] \cup O$

So in this sorry this is an OR gate. So whatever is the question was we have a 0 and a 1 so we know that this mean whatever here nothing this will not be propagated over here because this is the 1 and is this is a 0. So whatever list over here will be propagated so you write 1 1 bar that nothing of this area will be propagated and whatever here will be propagated that is intersection 1 1 into and a 0 1 the answer is the 1 over as the output so a stuck at 0 fault as the output will be detected.

(Refer Slide Time 22:53)

Fault deduction rules for logic gates

Gate Type	Inputs (In1 and In2)		Output (O)	Deductive Fault list at O
AND	0	0	0	$[I_{in1} \cap I_{in2}] \cup O$
	0	1	0	$[I_{in1} \cap \bar{I}_{in2}] \cup O$
	1	0	0	$[\bar{I}_{in1} \cap I_{in2}] \cup O$
	1	1	1	$[I_{in1} \cup I_{in2}] \cup O$
OR (Dual of AND gate)	0	0	0	$[I_{in1} \cup I_{in2}] \cup O$
	0	1	1	$[\bar{I}_{in1} \cap I_{in2}] \cup O$
	1	0	1	$[I_{in1} \cap \bar{I}_{in2}] \cup O$
	1	1	1	$[I_{in1} \cap I_{in2}] \cup O$
NOT	0	---	1	$[I_{in1}] \cup O$
	1	---	0	$[I_{in1}] \cup O$
Fanout	0	---	0	$[I_{in1}] \cup O$
	1	---	1	$[I_{in1}] \cup O$



(Refer Slide Time 23:10)

Fault deduction rules for logic gates

Gate Type	Inputs (In1 and In2)		Output (O)	Deductive Fault list at O
AND	0	0	0	$[I_{in1} \cap I_{in2}] \cup O_1$
	0	1	0	$[I_{in1} \cap \bar{I}_{in2}] \cup O_1$
	1	0	0	$[\bar{I}_{in1} \cap I_{in2}] \cup O_1$
	1	1	1	$[I_{in1} \cup I_{in2}] \cup O_1$
OR (Dual of AND gate)	0	0	0	$[I_{in1} \cup I_{in2}] \cup O_1$
	0	1	1	$[\bar{I}_{in1} \cap I_{in2}] \cup O_1$
	1	0	1	$[I_{in1} \cap \bar{I}_{in2}] \cup O_1$
	1	1	1	$[I_{in1} \cap I_{in2}] \cup O_1$
NOT	0	---	1	$[I_{in1}] \cup O_1$
	1	---	0	$[\bar{I}_{in1}] \cup O_1$
Fanout	0	---	0	$[I_{in1}] \cup O_1$
	1	---	1	$[\bar{I}_{in1}] \cup O_1$

So similarly, you can find out the rule for 1 0 which is this 1 and 1 1 in this case it is just like 0 0 for the AND gate nothing gets propagated excepting if there is something common fault is for both of them and 1 1 the answer is the 1. So in the output case it will be stuck at 0 fault will be detected so that is what is stated over here in the table for the NOT gate we know that whatever is the in input of the NOT gate will be propagated whatever in the input of the NOT gate it is propagated.

(Refer Slide Time 23:29)

Fault deduction rules for logic gates


Gate Type	Inputs (In1 and In2)		Output (O)	Deductive Fault list at O
AND	0	0	0	$[I_{in1} \cap I_{in2}] \cup O_1$
	0	1	0	$[I_{in1} \cap \bar{I}_{in2}] \cup O_1$
	1	0	0	$[\bar{I}_{in1} \cap I_{in2}] \cup O_1$
	1	1	1	$[I_{in1} \cup I_{in2}] \cup O_1$
OR (Dual of AND gate)	0	0	0	$[I_{in1} \cup I_{in2}] \cup O_1$
	0	1	1	$[\bar{I}_{in1} \cap I_{in2}] \cup O_1$
	1	0	1	$[I_{in1} \cap \bar{I}_{in2}] \cup O_1$
	1	1	1	$[I_{in1} \cap I_{in2}] \cup O_1$
NOT	0	---	1	$[I_{in1}] \cup O_1$
	1	---	0	$[\bar{I}_{in1}] \cup O_1$
Fanout	0	---	0	$[I_{in1}] \cup O_1$
	1	---	1	$[\bar{I}_{in1}] \cup O_1$

So it is an inverter. So whatever here is automatically propagated here or the if the input is a 0 over here we get the answer 1 over here so stuck at 0 fault is detected if is a 1 over here the answer is 0 over here and a stuck at 1 fault is 0 that is what is being stored and in case of fan out it is the same thing like an inverter so, that we have seen the fault gate so whatever the list here will be propagated here and here.

(Refer Slide Time 23:54)

**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.



So whatever is the list they are propagated and if you get the value of 0 a stuck at 1 is detected and if you get the value 1 a stuck at 0 is detected that is just the reverse if you get a single value 0 a stuck at 1 fault is detected if the single value is 1 a stuck at 0 fault. (( )) So this is actually the fault deduction table for the fault list so what happens? So once you have that so given a circuit we apply a random pattern then your goal here is that you have to traverse the circuit once and in the 1 traversal of this circuit what you have to do? You have to find out that what are the faults that are detectable.

So here are some rules we are applying that if there is an inverter what are the rules for the AND gate what are the rules, if it is an OR gate what are the rules by applying the rule from level 0 to level 1 to level 2 level 3 and so on for the circuit you go to the output and easily you can find out this list that all these faults which are in the list are detectable by that input and then in that is in one scan of this circuit this is done. Now you find out also there are we say 30 faults in the circuit then one in one scan of your circuit you find out that some 20 faults have been detected.

Now if some 20 faults have been detected so is the good news. So another 10 faults are remaining so you again you have to apply this things and you keep on going this is a very good an advantages approach compare to serial and parallel for simulation because in one goal you can reduce the information about all the faults.

But here we have to ap think once that once I have applied one pattern then I have gone for or deductive for simulation at the output then I find out the these are the faults are simulated a deductable then I drop them and then I take a new random pattern and again I read you everything so what was our discussion on fault simulation by event driven simulation and compile course simulation. So in compile course simulation what was the idea that one pattern you have apply one fault you do find out and then forget everything and then again do another simulation but, you find out that 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 detect kind of a thing.

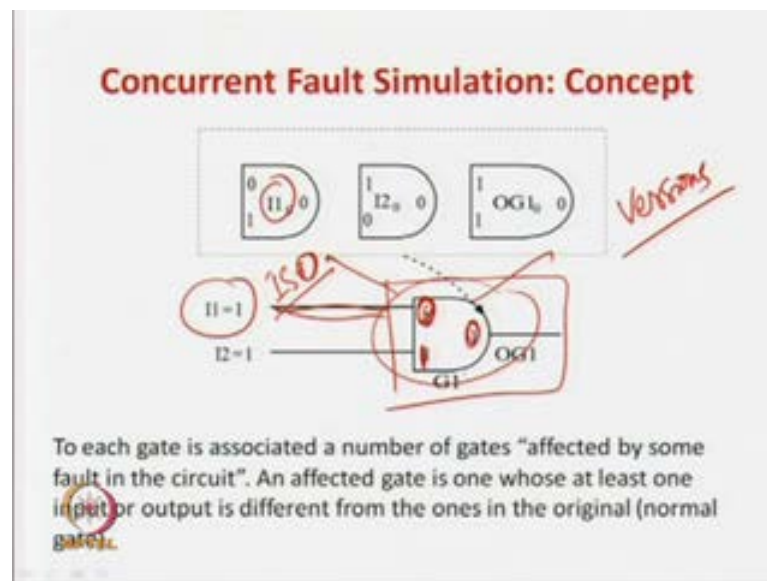
So why we are unnecessarily computing this circuit value again and again when you are going for one pattern to another pattern. Then we saw that event driven simulation is a very good approach in this case what we do? So we find out the output or the input on the fault of normal case then we retain some values or say all the values of circuit we retain and then what we do when we apply another pattern or another fault may be then we should able to re-compute only those portions of the circuit we got change that an impact of something, impact of a new pattern or a new fault or something so that is the idea that is this is missing in our deductive fault simulation. So in one fault we have apply then we scan to this circuit by applying the rules in the table which you have seen then we find out these are the list of the faults which are deductable.

But we generally cannot or generally do not save anything in that algorithm and again we have to read you everything. So the last fault simulation technique will study today is the concurrent fault simulation. So that is actually same thing as deductive fault simulation again that in one scan of this circuit you are going to find out what are the faults that will be detectable but, at the same time we are not going to through away the results. So if by applying one pattern we find out that these these these these faults are detectable then we do not through eliminate the all the results we apply another pattern and then we will try to re-compute only those things which are got getting changed that is a even driven

simulation. So you can think that concurrent fault simulation is a event driven simulation plus detective fault simulation. So that idea we have to (( )).

That is what is saying the detective fault simulation which will saw last can determine all the faults detectable by a random pattern fine? But, when a new random pattern is applied the whole process is to be redone. That is just like a compile fault simulation so you have to do something better. So in the last that is concurrent fault simulation technique similar to detective 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 when a random pattern is fed it needs to compute only that information which got changed. So that is the idea for a detect I mean what you call a event driven simulation. So concurrent fault simulation get motivated advantages achieved by event driven simulation over the compiled code simulation so that is the basic idea.

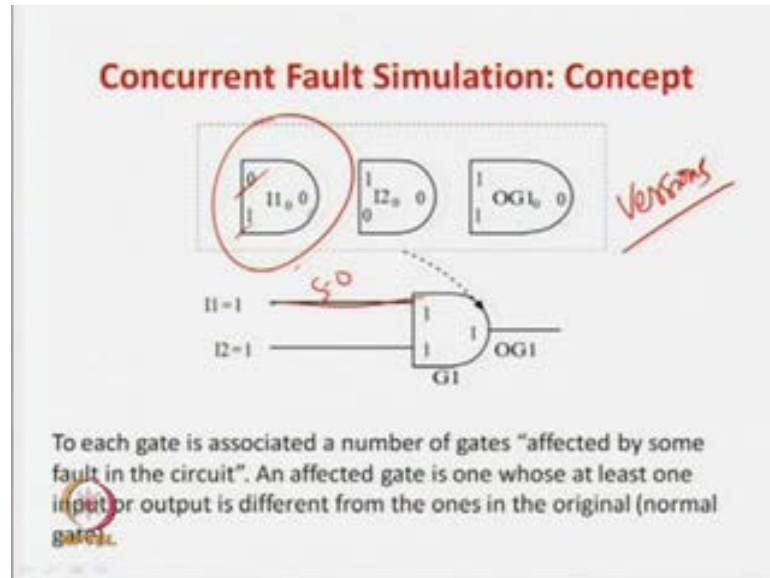
(Refer Slide Time 27:32)



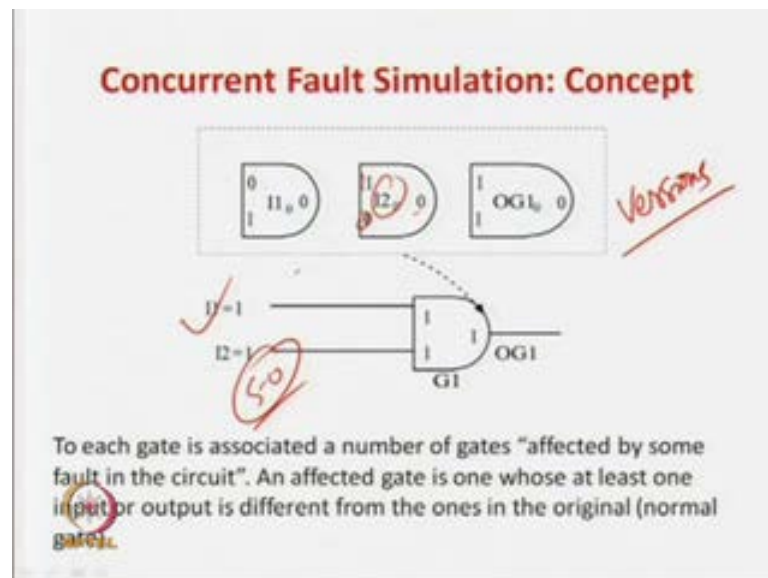
So you will see the basic concept of a concurrent fault simulation. So let us take an AND gate so this is your AND gate set and then you apply 1 1 and the answer is a 1. So this you have to first draw this, normal condition then you have to find out that these versions of the gates we have to draw some versions some versions of this gate which are affected by the different type of gates like for example, you see if i 0 i 1 stuck at 1 that if this net is the stuck at 1 then what happens? The your this input is 0 it because stuck at 0 sorry

this is stuck at 0 apply 1 1 you are applying. So this net is stuck at 0 so this input will be 0 this is 1 and obviously the output will be a 0.

(Refer Slide Time 28:20)



(Refer Slide Time 28:29)

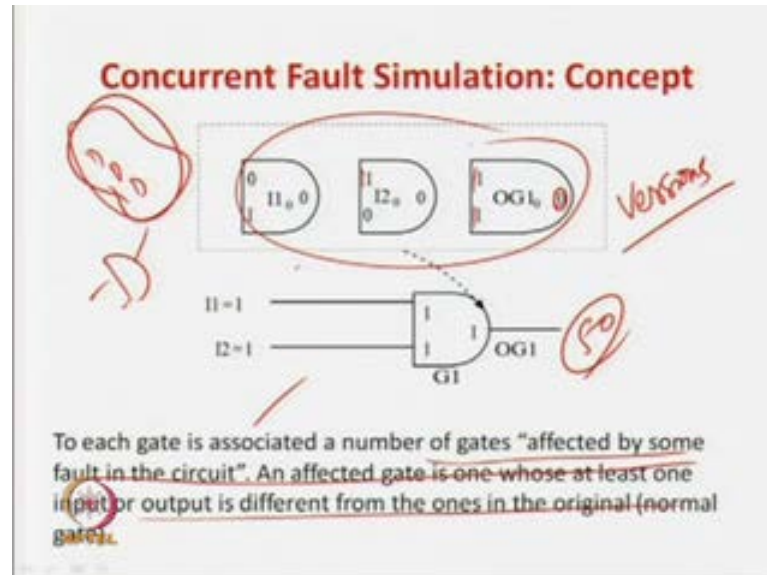


So this is 1 of 1 for which is affecting this gate so this information you are have to store somewhere. So this is the gate which is saying that this gate is the stuck at 0. So this answer is 0 this is 1 and the output is the 1 so this is the 1 version of the circuit and with a stuck at 0 fault here now another stuck at 0 fault can be here. So this is the case it is a 1 0 because this is net is stuck at 0 and the output is obviously 0. So this another version of

this gate where there is a stuck at 0 fault at this gate similarly, there can also be a stuck at 0 at this point so is the version is the 1 1 and the output is 0.

s 0.

(Refer Slide Time 28:40)

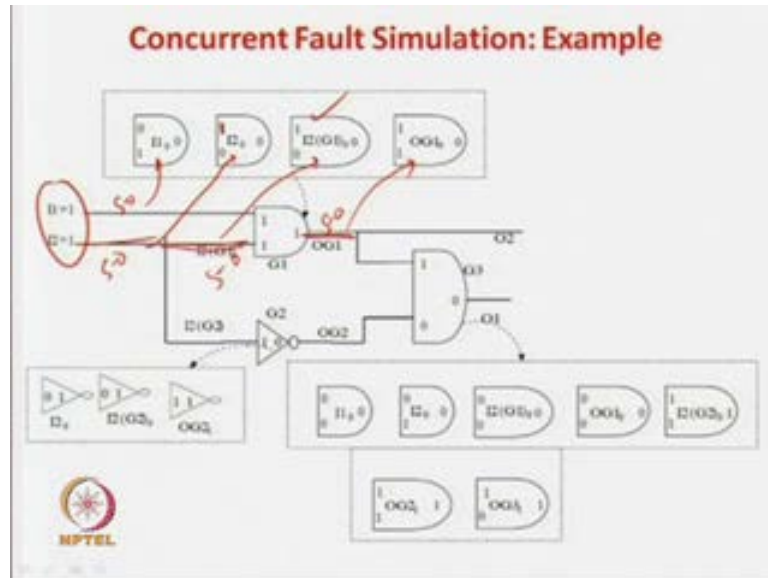


So for each gate you affect you attach some gates in this circuit which is affected by at least some fault, see each gate is associated with a number of gates affected by some faults in this circuit. So to each gate you have to add you have to attach a bubble so this is a one gate say and you have to attach a bubble. So this bubble will comprises all other versions of the same gate with different faults in the circuit. So if there are some 30 faults in the circuit so your bubble will have some 30 faults which are affecting this circuit when if you have some 30 faults which affect this gate obviously so, then only I mean you will have that this fault I mean gate list in that bubble.

So to in other words in concurrent fault simulation what is 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 the circuit only those versions of the gate will be available. So for if there are 30 faults in a circuit and 20 of them affect to a 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. So let us take this same example and let us see how we can make this (( )) more clear so we are applying the

random pattern 1 1 so you will know that we are now considering this gate so what are the faults?

(Refer Slide Time 29:53)



So stuck at 0 fault at this net so this is 0, so this is 0 not 1 0 and the answer is 0. So this one is reflected over here similarly, there can also be a stuck at 0 at this net this is the i 2 0 so in this case this is the will be the value 1 this will be a 0 over and this 0 1 the answer is 0. So another fault list you are having for this. Now we already told that this is one this net is a fan out net so it is different so i to g 1 stuck at 0 is also possible so in this case we are going to have this (( )) so this is same so this two come in a stuck at 0 here stuck at 0 here the stuck at 0 faults and stuck at 0 fault over here.

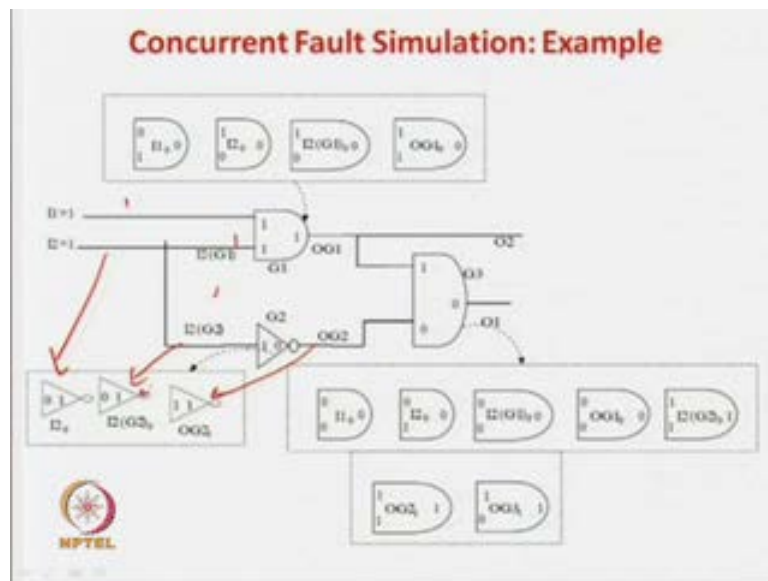
A same affect so these are the two gates which save effect 1 0 and the output is 0 but, you have to keep the version because this represent two different faults. Now this 1 1 so obviously a stuck at 0 at this net o output 1 stuck at 0 is also affecting 1 1 the answer is 0 so for this version you have this. So this AND gate has a four things a four gates in the bubble corresponding to a stuck at 0 fault here, stuck at 0 fault here, stuck at 0 fault here and stuck at 0 fault here. So for these four stuck at 0 faults you have this thing in the bubble correct?



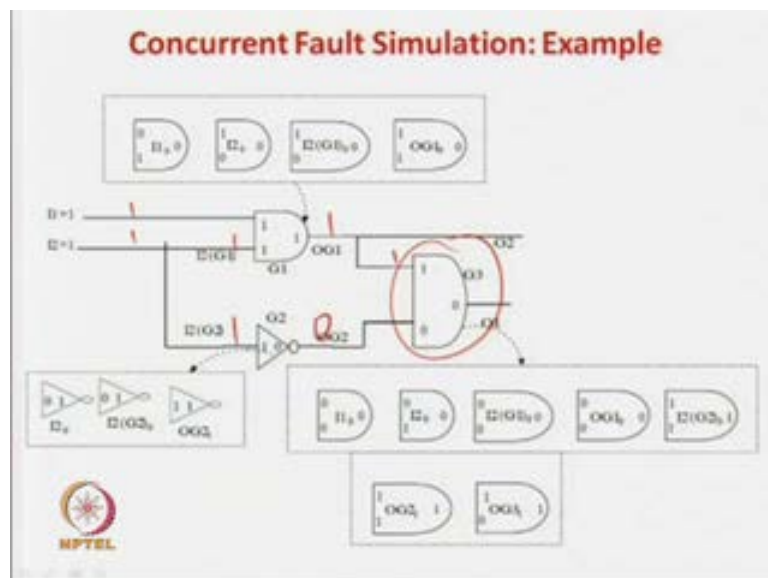


case it is 1 and it is a 0. So if it if this net is stuck at 0 over here obviously so i to 0 so this i to 0 so i to 0 what happens? So this one will be 0 so this is stuck at 0 and this one is 1. So this is reflected by a stuck at 0 fault here that is the change for this gate similarly, a stuck at 0 fault here will also affect. So now it will be 0 and the answer here will be a 1 normal case it was a 1 0 but, in the fault case it was a 0 1. So this one i to 0 stuck at 0 affect is also reflected over here and of course, is a 1 1 normal case.

(Refer Slide Time 32:40)

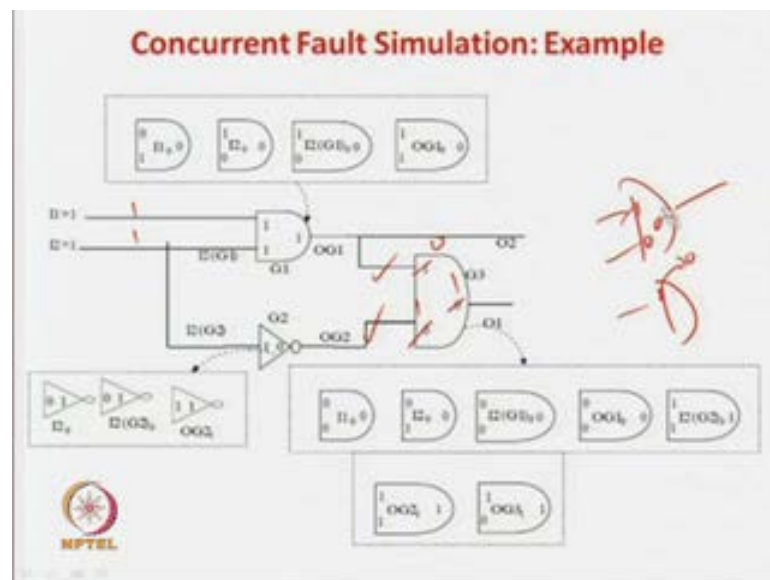


(Refer Slide Time 33:01)



It is a 1 and the answer that is a 0 so a stuck at 1 fault at this 1 will also be affecting. So in this case you see the input is 1 the answer should be 0 but, a stuck at 1 fault here will make it a 1 so 1 1 this affect is also shown over here. So these are the three faults which is going to give you the bubble list of three. So this is the first case, this is the second, 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 has some three faults in the fault now we are making so now the thing will all the benefits will start coming when you apply another well-known pattern. So let us first see what is the case know now we have so what we have any normal case 1 1 1 1 0 so this is also 1 1 and this is a 0 the answer here is a 0.

(Refer Slide Time 33:23)

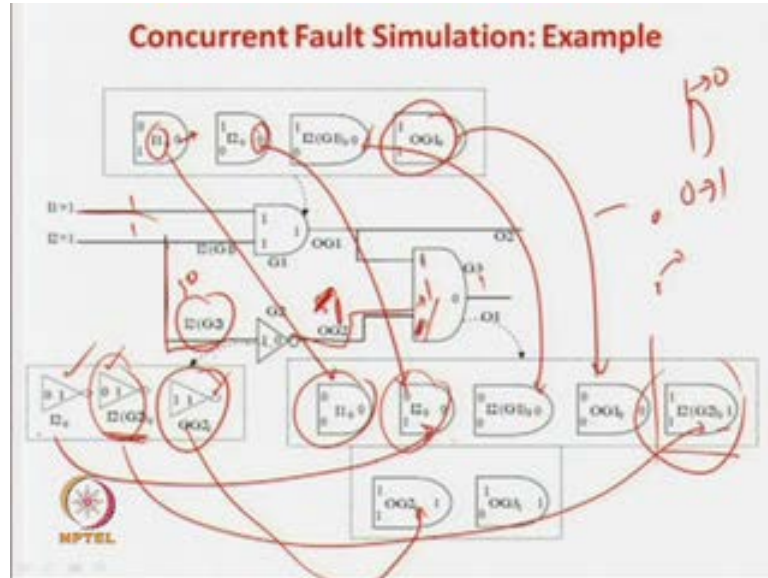


Now for this gate also there will lot of gates in the bubble which is actually affecting this one. Now let us see how can we find out these are the fault this thing within number of faults affecting this one. So you can see that this one depends on three values, the value here, value here. So these are the two value i mean two this this gate 1 0 is the normal case and then we have to find out for all we have to find out 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 gate. So the normal case this is the 1 0 the answer is 0.

So we have to find out for all cases of faults or all partials of this AND gate where at least this is a difference so here instead of 1 if it is a 0 you have to put if that means if it is a 0 over here we have to put that version if there is a 1 you have to put the version and

if there is a 1 in this case you have to put the version so all such cases we changes the inputs signal value the output signal value you have to put it.

(Refer Slide Time 34:02)



Now you can study this list. So you can see here the answer is a 1 correct? But, now if this stuck at 1 if this stuck at 1 fault is 0 so this gate output will be a 0 so here we are you are going to get a 0 and the answer is 0. So this 1 you have to keep so how we are deriving? This we are deriving this from here that is (( )) any signal says 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 i 1 stuck at 0 this stuck at 0 will result a 0 at this gate out but, this is we can find out from this bubble list and then this 0 means the signal is changed 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 one will also changes value from 1 to 0. So that value also you have to put it over here so that is a 1 this is this case not affected the answer is a 0. So this is the case which is affected now you can also see this gate is also affecting the signal so this one is also you will be available over here and this stuck at 0 that is this net stuck at 0 means that is this gate fault is also changing this value.

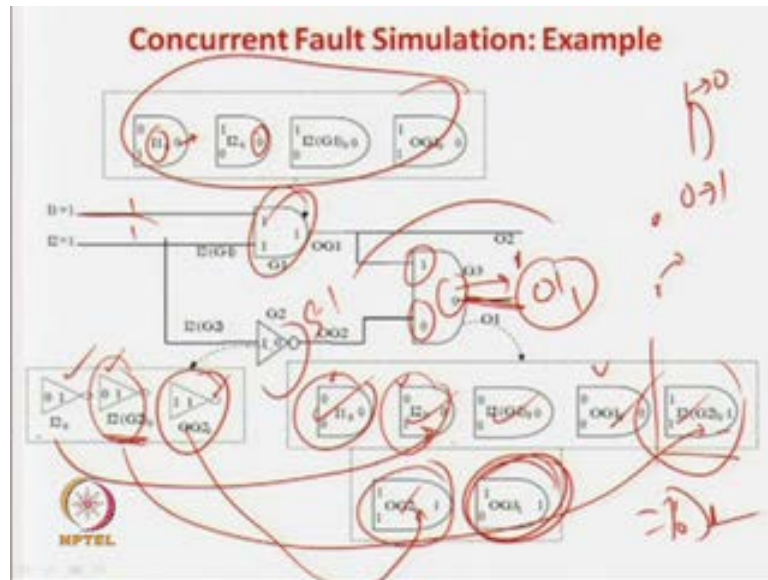
So you are going to have this here so what you are having so 1, 2, 3, 4 1, 2, 3, 4 this four gates check to the chain signal from 1 to 0 1 to 0 so they are actually input within this bubble. Now you see obviously now we are going to check for the simple input. So this

input means is the 0 to 1 now you can check that if something changes from 0 to 1 that is will be included so you can see that these are the three gates this one, this one and this one I think these are the three gates which are actually changing the value of this one correct? This one is changing from 0 to 1.

So now you can see o g 1 sorry this get we are concurrent this is a already over. So now you can think that slow signals which are changing the value from 0 to 1 we have to include an i 2 g 2. So you can find out that if this is the gate so i 2 g 2 this is the i 2 g 2 is 0. So in this case here the answer will be a 1 so this 1 you can include it over here and i 2 z 2 is also the case but, actually i 2 0 already we are including over here.

That is if this gate if i 2 0 fault is there so then this actually this one is included over here. So already this is taken into picture because this fault i 2 stuck at 0 i 2 stuck at 0 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 on from this list as well as on this list anyway so this gate is already added.

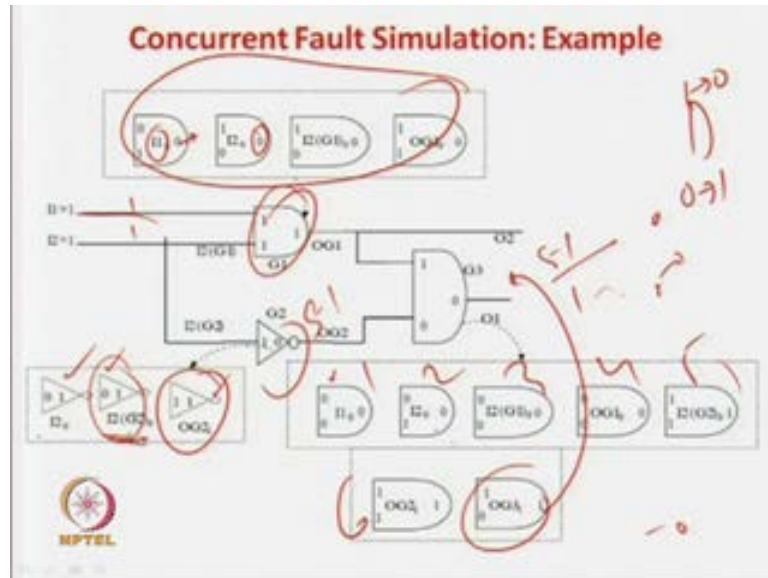
(Refer Slide Time 37:25)



So this gate this gate is changing the value 1 here so 0 to 1. So this gate impact we are bringing out over here and o 2 g 1 that is this one this net this net if you can think of thus o 2 output to this net this net stuck at 1. So this is the picture so it also changes the value of this one that is 0 to 1 so that one also you have to bring it over here and in this case

you can see that this is already one and this fault and this fault that is o 2 that is i 2 g 2 stuck at 0 is i 2 g 2 stuck at 0 so this 1 stuck at 0 that is 1 stuck at 0 converts this to 1.

(Refer Slide Time 38:38)



So here the single changes from 0 to 1 1 1 the answer will be 0. So you have to write that this gate is saying that 1 1 the output is a 1 that is you have to write in that way so and this one this stuck at 1 fault sorry this is the this list we have already discussed. Now this net stuck at 1 this net stuck at 1 actually convert this 1 0 to 1 1 1 the output is a 1 so this net o 2 o g 2 stuck at 1 is also (( )) over here and the and 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 inputs to this one so this the you can say that concurrent fault stimulation also.

This is just like a some fault these are propagated from the previous level to the next level but, here actually the gates this gates versions are propagated from the previous level to net pulse same concept is there is not much change but, I am also one more for the pattern is there which we all the one more thing will be there like we have seen in the fault simulation that here if your answer is 0 in this stuck at 1 fault have this gate detected that are should be added to the list. So in this case also the answer is we are saying that the pattern is 0 1 0 0 say any signal change will be affected.

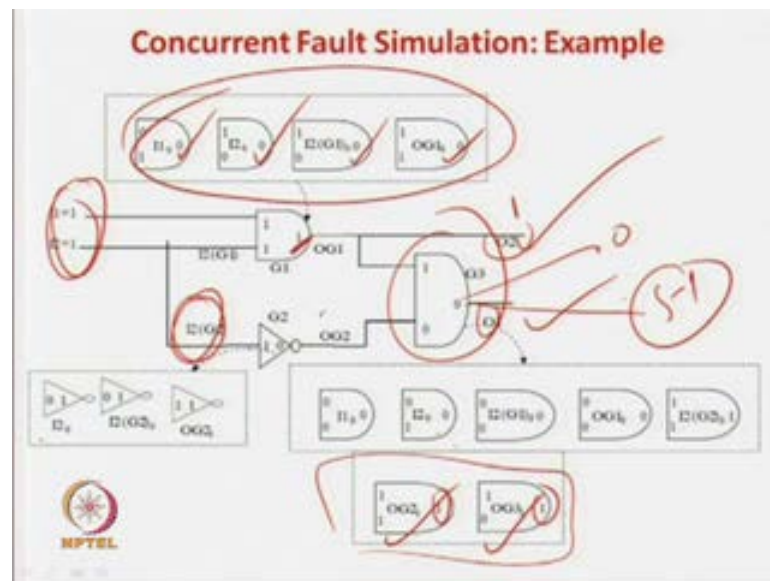
So signal change at this net will be o 1 stuck at 1 then the signal change will be a 1. So this gate list is added that is 1 0 and here answer is a 1 because this net is stuck at 1 this net stuck at 1 so the last signal and the signal change at the output so this five this five



fault this gates 1, 2, 3, 4, 5, 6 these six are propagated from the previous values and this one correspond to a stuck at 1 fault over here so in this case.

So this is how we get this fault list concurrently we generate this one. So this still now you might see that the concurrent fault simulation is nothing but, it is a detective fault simulation only in detective fault simulation what you had you just propagating the fault list from one part to the another part and he has been actually propagating the bubble list of the gates otherwise there is no deference.

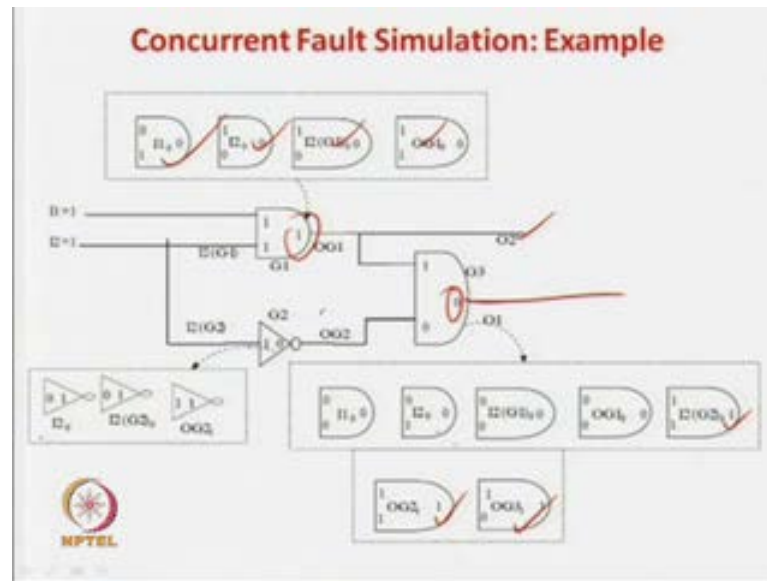
(Refer Slide Time 39:10)



The differences start coming into picture only when will change this random pattern but, 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 the 0. So wherever the answer is the 1 at the output of the AND gate you go know the fault is detected. So in this case you can find out that there are two faults that is o 2 g 1 stuck at 0 the these net stuck at 1 this the stuck at 1 and this net stuck at 1. So these two faults are detectable because the output of the AND gate is the 1 and in normal case is a answer is 0 but, now similarly, also you can see that o 2 also some faults can be detected.



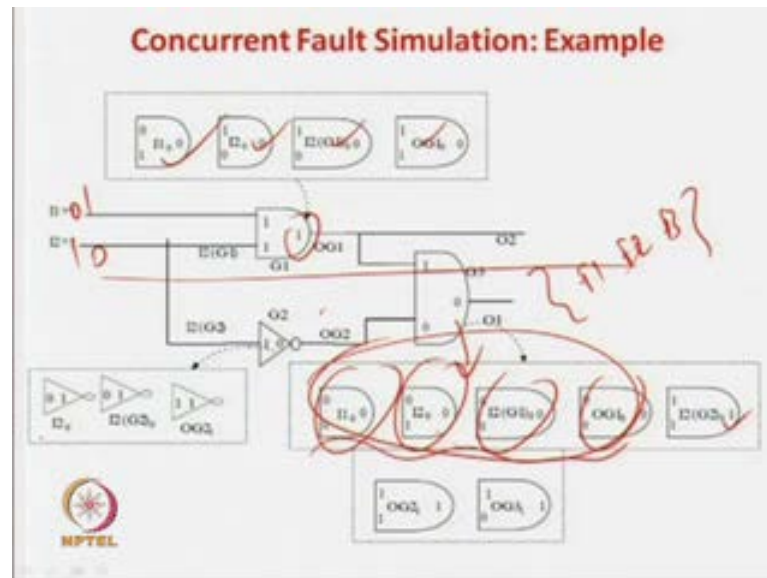
(Refer Slide Time 40:31)



Now what are the faults that can be detected here the answer is the 1 so normal case also two will be a 1 but, here this is 0 this is 0 the output this is the also 0. So all this four faults are detected at this net and this one sorry o 2 g 1 that this fault and output stuck at 1 fault are detected at this net. So the same thing like our what do you called this detective fault simulation we can find out the fault list.

So here the fault list is not very directly computable but, it can be very easily related that is in case of deductive fault simulation what happens the fault list will tell you that is faults are directly deductable that you get list over here that is o 1 some list over there and from the list you can say that these are the faults which is detected but, in this case we have to just do simple computation that what which are the gates this input is different. So in this case this is different, this is different and this is different.

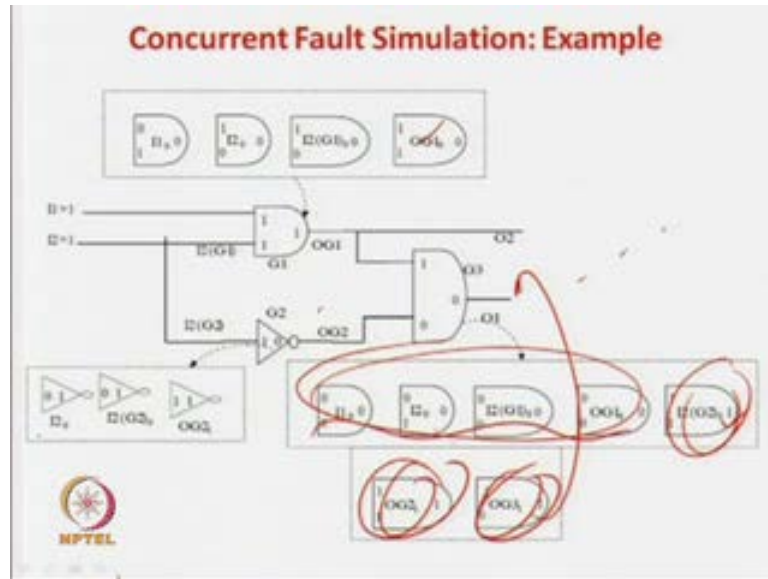
(Refer Slide Time 41:00)



So these three faults are detectable at this net and in this case the answer is a 1 so 1, 2, 3, 4 this four faults are detectable over here. So this is just a very simple extension or you can call the same version of your detective fault simulation in the concurrent fault simulation instead of propagating the fault list whereas, propagating the series of lists that is the one thing and one, another thing you should absorb that in case of deductive fault simulation we have the fault list which tells this is a fault list say so we test that only fault 1, fault 2 say fault 3 are deductable but, it will not retain any information which is non-detectable so here you see we have 1, 2, 3, 4. So four gates are unnecessarily which are hanging over there but, there are not detecting your fault so this things are hanging over here.

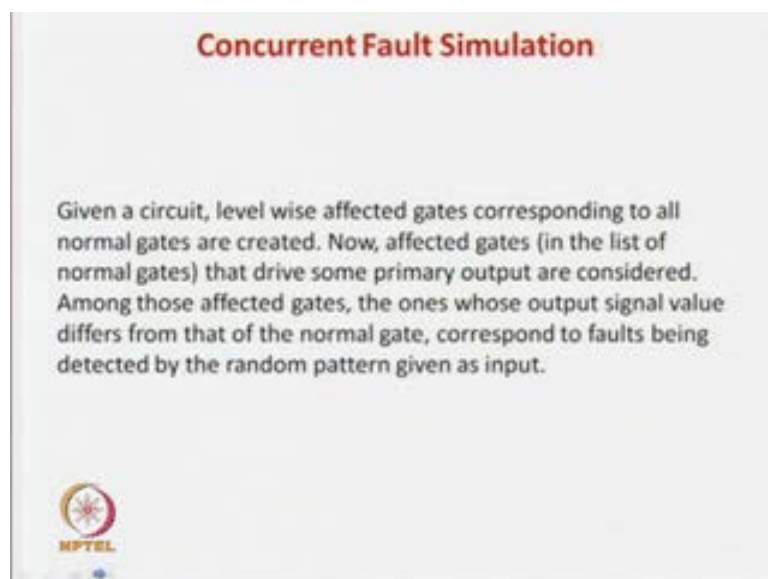
So there are nothing to do unnecessarily there are after going your memory but, we are see will see in the next slides that because of these things from information is retain that is now if we change the value from 0 to 1 or 1 0 whatever then we need not re-compute the whole thing we just re-compute whatever changes have happened over here only those things will be re-computed and that is like a even resolution something will be same.

(Refer Slide Time 41:46)



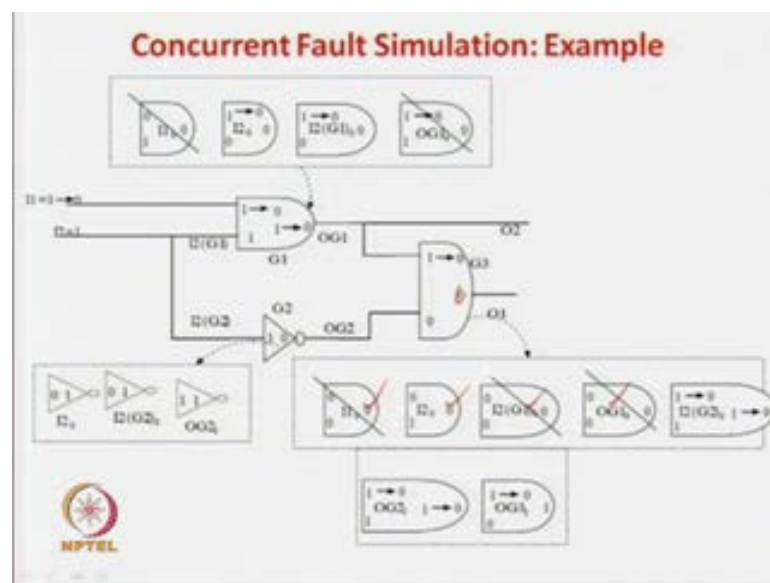
But in case of this jet fault simulation even with the fact that (( )) only the set of say 1, 2 only this three information will be available because that had been detected. So this unnecessary things are not there but, whenever there is a change in pattern you have to reduce everything. So that is a trade off so you have to think that if you are doing a deductive fault simulation where as the output you will have information about only these three faults because these three faults are deductable by this one and this information about this unnecessary four gates because they are output is 0 and the normal case it is 0 so the faults are not deductible they will not be there.

(Refer Slide Time 42:16)



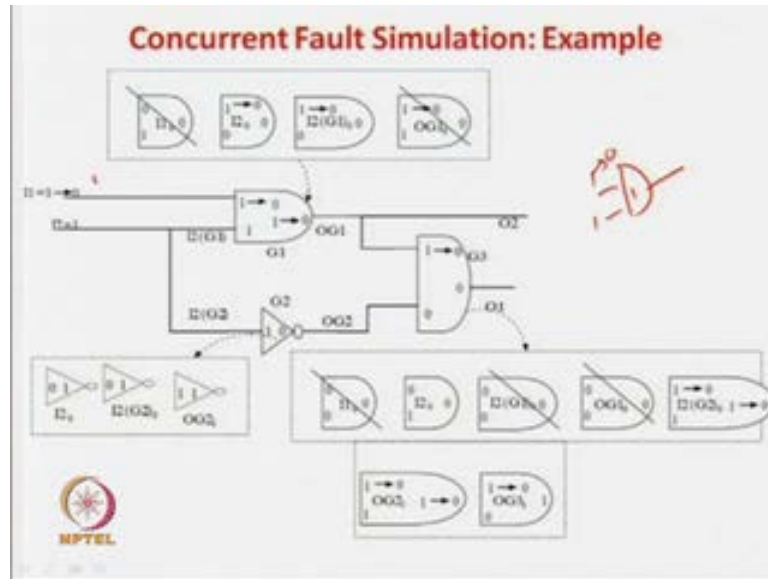
But the trade off is that whenever there will be a change in pattern then again you have to do everything but, here if there be the change in pattern there will be nothing we have to do much only whatever changes are required that has to be done like that is what these things which we have already discussed not to reduce. So the concurrent fault so, sorry 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 benefits that unnecessary we were carrying some additional gates. So unnecessary what we are doing so unnecessarily we will be carrying out some gates which could not detect any fault so those gates were carrying with us.

(Refer Slide Time 42:53)

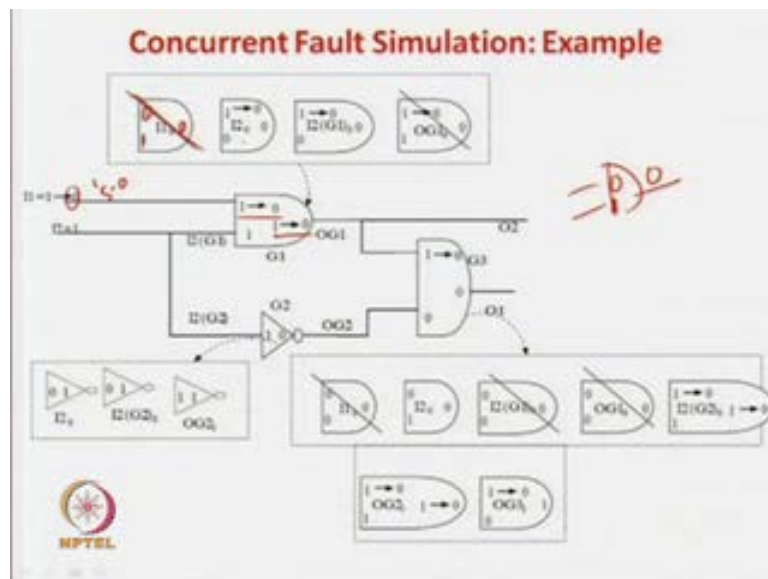


But in case of detective fault simulation those gates will not be carried. So now what is the advantage that you have to illustrate that I think if you just see so this gate, this gate, this gate this will unnecessarily be carried out because the output was 0 and this output is also 0 but, now we will see that what is the advantage of carrying these gates because this is a trade off in case of detective fault simulation we will not do this but, still if whenever new pattern is applied you have to reduce everything here will be let us see here in concurrent fault simulation by carrying this dead gate if you can see what the advantage. So now let us change the input pattern from 1 1 to 0 1 so there is a change.

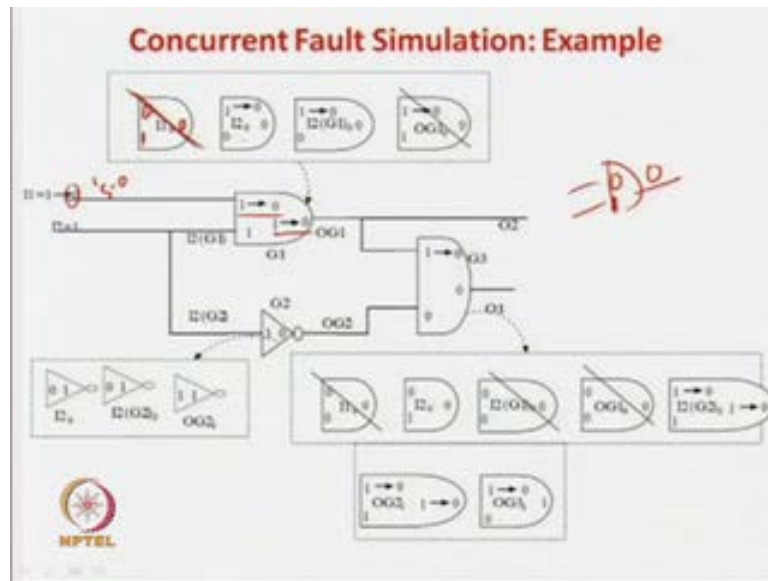
(Refer Slide Time 43:36)



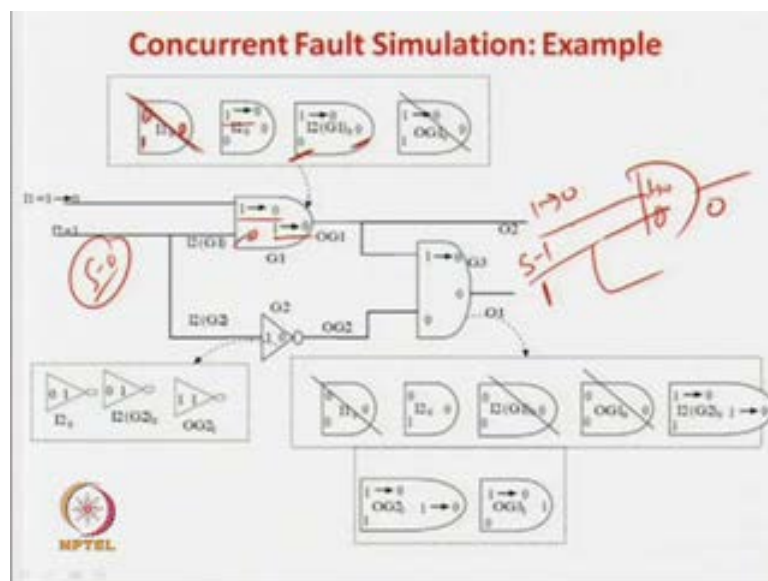
(Refer Slide Time 43:49)



(Refer Slide Time 43:50)



(Refer Slide Time 44:34)



So now as you already seen in what do you call our detective what do you sorry even driven simulation or computation mean only the fault those cases where there is a change. So let us see what happens so initially it was 1 1 and the answer was a 1 then in the normal case we have to apply 1 1 the answer was a 1 in this case. Now there is a change in signal value this is changing from 1 to 0. So what we will do we re-compute only when things are changed so re-computation is 1 to 0 here it is a constant so the gate output is this one correct? Now these gate was corresponding to a stuck at 0 fault over

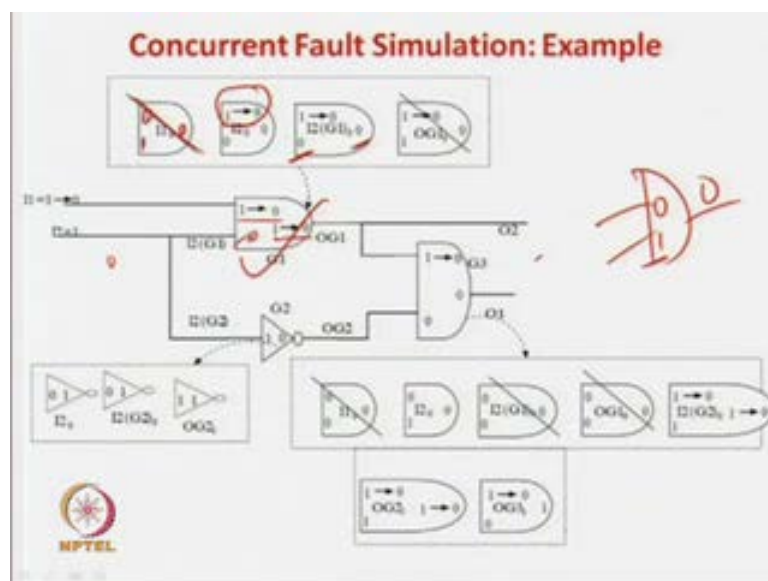


here. So it was a 0 over here 1 over here and 0 over here correct? So this and now this new and so what we are going but, this is the old thing that is retained.

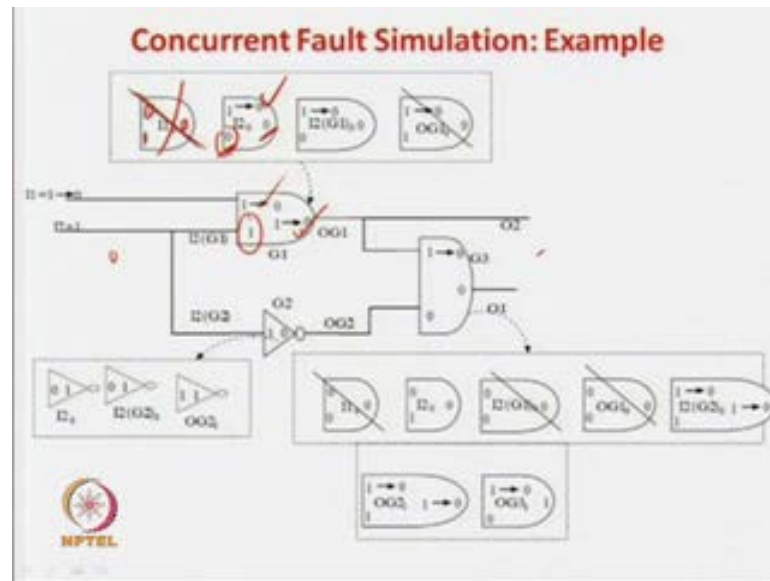
So you can see now you are normal gate will become 0 1 and a 0. So that is your normal gate this is your normal gate recognize 0 1 0 because this pattern that you have to change. So this gate you can delete this gate which we are going to delete. Now there is so now so what is the case of this gate is deleted? Now this another gate was i 2 stuck at 0 so this corresponds to the second gate in this list if you remember this i 2 stuck at 0 was a stuck at 0 over here. So now a stuck at 0 over here was so this input was a 0 because this gate is stuck at 0 now there is an impact of change is from 1 to 0 and the answer is of course, a 0 by and this gate will be retained. So why this gate will be retained? Because initially the gate will it was a 1 0 and the answer was a 0.

Now sorryso whenever when the input signal was 1 1 and the inputs so what this gate what the stuck gate with the stuck at 0 fault here look like 1 0 and the answer is the 0 at the stuck at 0 1 fault at this it was a stuck at 1 fault over here the answer you applying a 1 over here you applying now you apply a 0 over here. So this one will be changing from 0 the output will be no change. So you need not compute this and you need not compute this these values were retained same as in the previous case when we are applying a 1 over here 1 1 over here and only this small changes at this 1.

(Refer Slide Time 45:25)



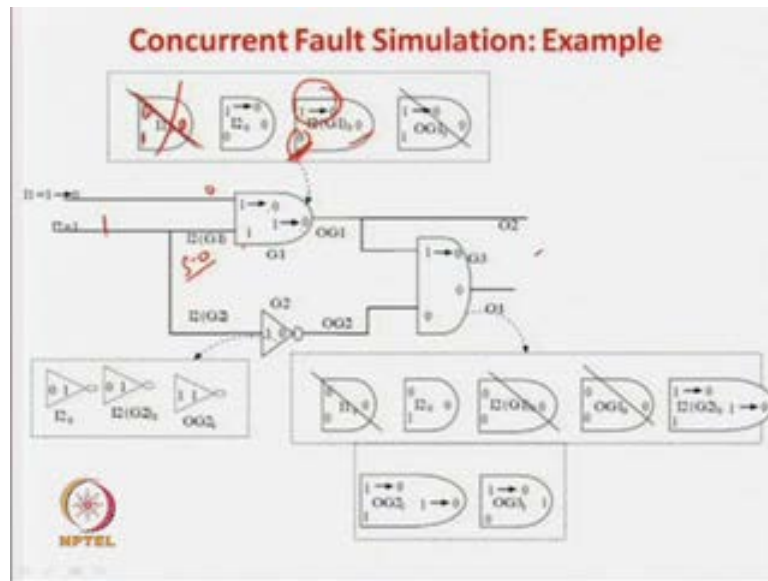
(Refer Slide Time 45:43)



So there is a signal change because now the normal gate looks as 0 1 0 this is the normal gate this one and here to the fault affect the gate is 0 0 0 so the signal difference over here. So these gate is going to be retained. So that is again we see can see that what we are doing so this gate is retained so what we are doing over here so what we are doing was the very simple thing we are going for even driven simulation.

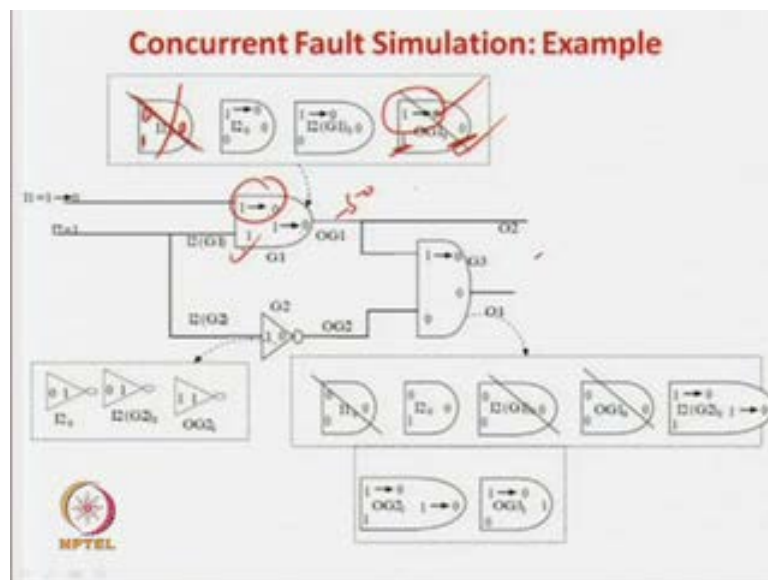
So we are just changing whatever signal here require this is a change here, this is a change here and in this case after doing this change after doing this change you can find out here is that this gate and this gate remain similar so you can drop it but, for this second case there is a i 2 stuck at 0 fault this is the signal from here this does not change this remain same so this things and this things retain from the previous computation and we so they say they there is the signal difference between the this one. So this gate is remaining so, some computation at the computation at this level and the computation of at the this level are same. Now let us see the third gate the third gate was i 2 g 1 stuck at 0 this line is stuck at 0 over here.

(Refer Slide Time 46:29)



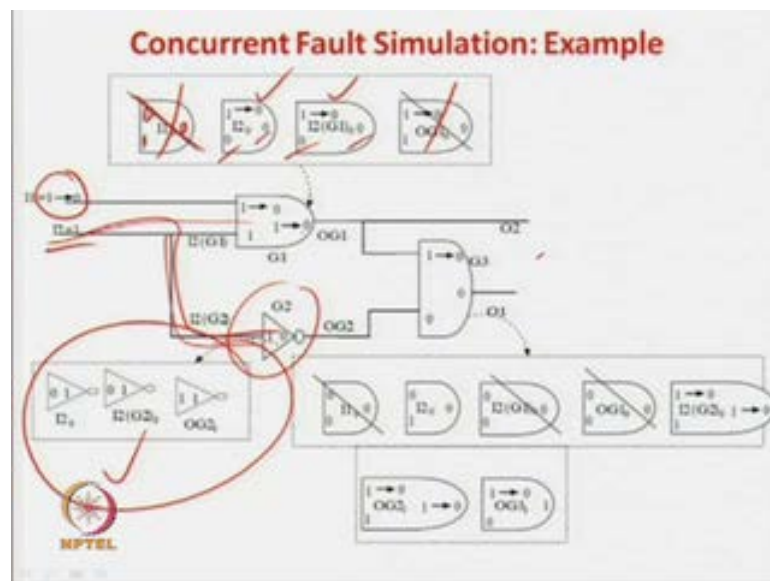
So now the signal your applying here is the 1 this is the 0 over here. So in this case is the 0 so we know that this stuck at 0 over here this signal is 0 this only a change over here and this thing is retained and this thing is retained. So we can find out the that the ordinary gate the signal was 1 0 1 0 so in this case it is 0 0 and answer is 0. So again by the same logic of i two stuck at 0 this get is also retain but, this computation and this computation is same.

(Refer Slide Time 47:10)



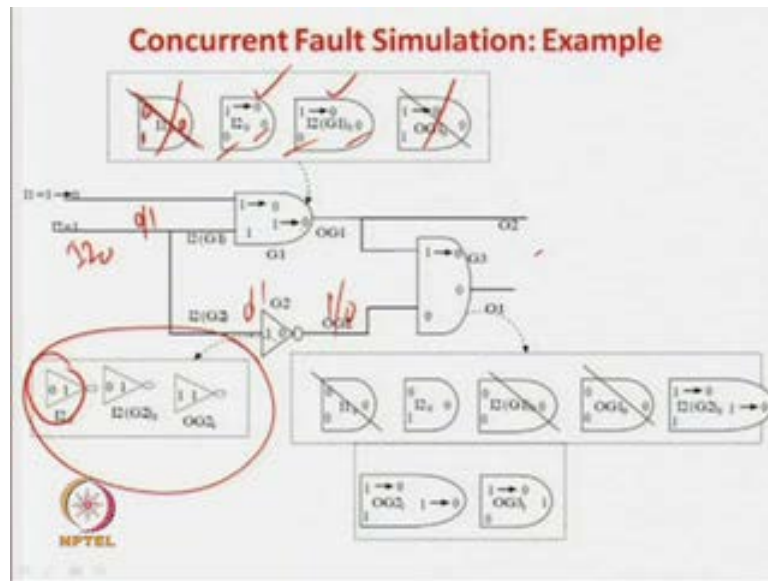
Now let us look at the output of the gate but, it is change so whenever those faults are not over here applying the change from 0 1 1 to 0 1 and there is a stuck at 0 fault over here. So you can say the this change this signal, this signal and this signal values retain same from the previous computation and now this gate is having 1 0 from 1 to 0 your going here so answer is the same and the answer output is 0 and this is 1. So this two gates become normal gate and this outputs stuck at 0 gates becomes similar so this gate also has to be dropped.

(Refer Slide Time 47:30)

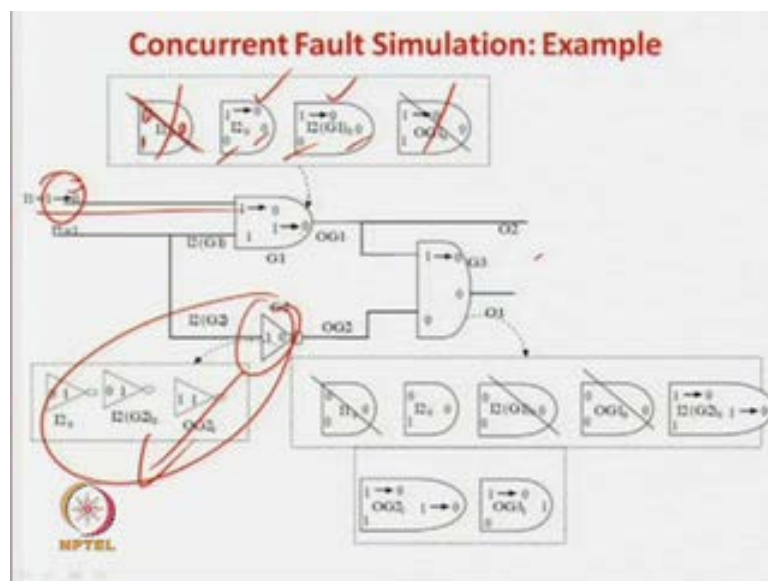


So now what happens this two gates dropped and this two gate gates keep on retaining but, we did not compute the value here and did not compute this value, did not compute the value, they can just carry propagated from the previous one. Now in this case you see if you are applying 1 over here than this n change the circuit change over signal change over here. So the fault list at this gate all the things will be retaining because this signal change over here and in this line there is no impact sorry there is no impact in this line. So if there is no impact in this line then we did not think about any think you can just copy paste what was where available over here so if it is i 2 stuck at 0.

(Refer Slide Time 48:13)

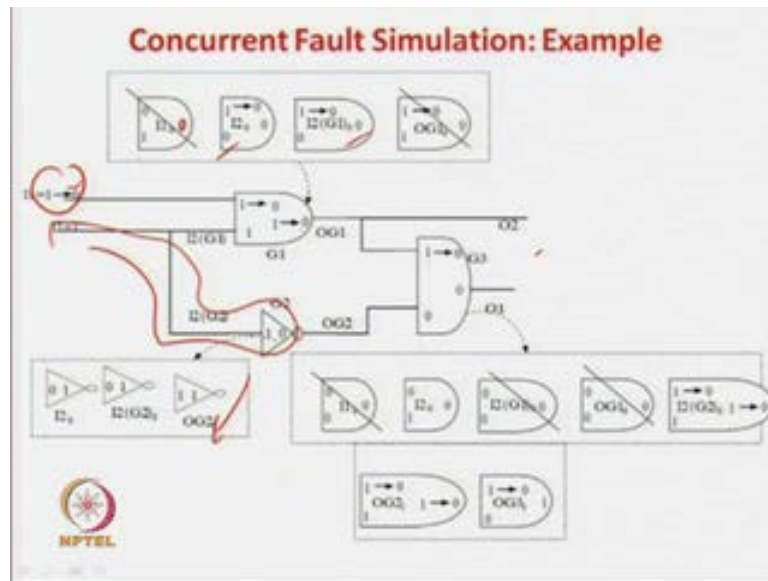


(Refer Slide Time 48:37)

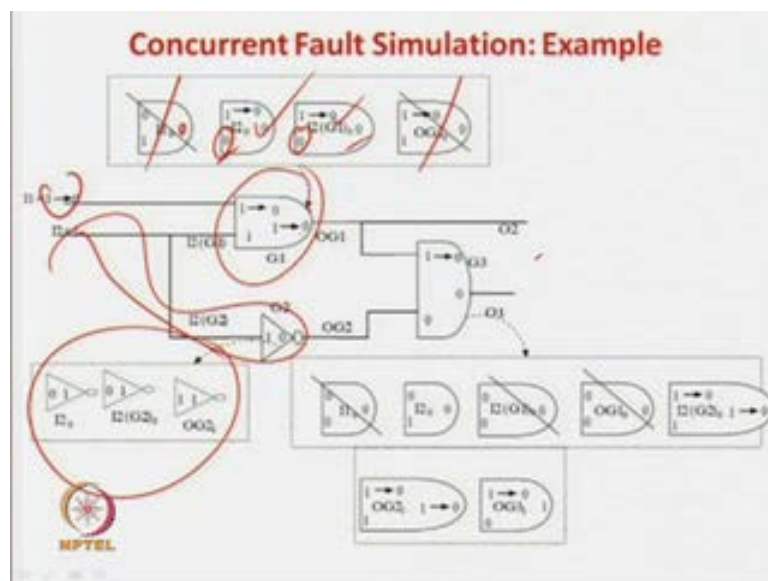


So you are applying 1 over here so it will become 0 over here, that is 0 over here and the answer will be a 1 over there by in the correct sense it should be a 1 it should be a 1 and answer should be a 0. So that is this fault list this gate is already retain from the previous list. So previous list of when the input was 1 1. So what we have seen in this example that whenever the fault is affect of this input change this one. So this thing can directly retained over here because the change in the input from here is does not affect over here.

(Refer Slide Time 48:56)



(Refer Slide Time 49:04)

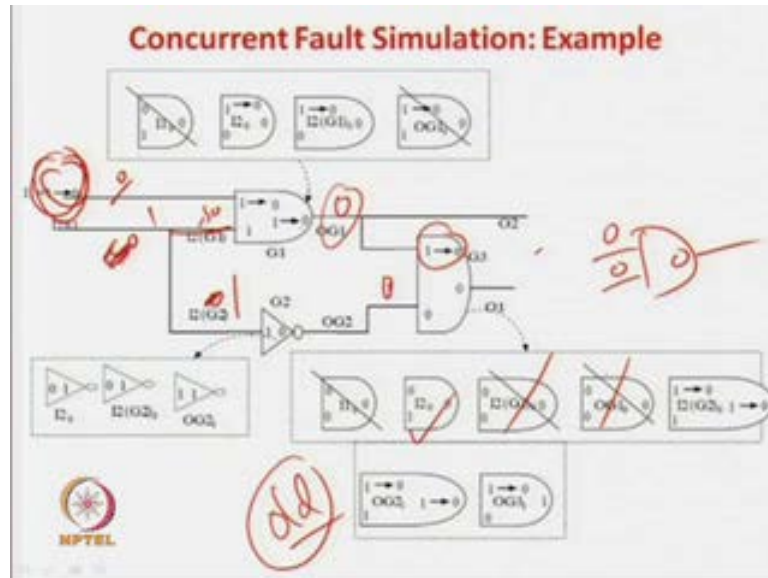


So this is the advantage of a concurrent faults simulation of 30 fault simulation, that many of the portion of the circuit like this portion of the circuit, this portion of the circuit is not affected by the change in this one also you can directly add this bubble list that is you did not even think of in this case in this case of the AND gate there is a change over here so, some impact on this gate is available as have happens you have (( )) go on deleting this gates and keep on retaining and in this gates also you have saved the computation of this input and computation of this inputs but, for the this portion of the circuit you need not think about anything just you retain the value that is what is the



importance of deductive sorry a concurrent fault simulation over a detective faults simulation than you need not keep on this thing now by the same rule.

(Refer Slide Time 49:33)



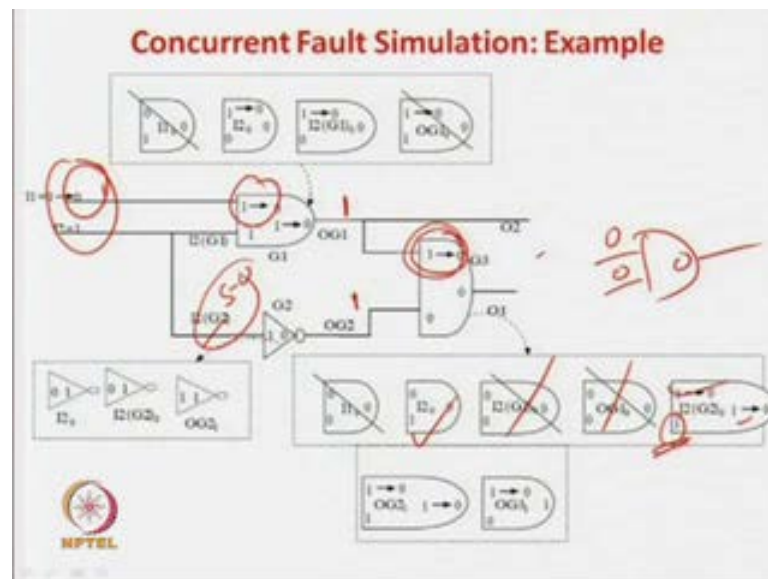
So this was now the this is change so in this case this will be the change from 1 to 0 and now you can see that this is this corresponds to this net stuck at 0. So this whole is from the old computation, this from the old computation. So this is the what we are having from the old computation. So now you can see that because of this change so you will be normal condition will be 0 0 and the answer is a 0. So this is about this gate now you have see that i 1 the this stuck at 0, this stuck at 0 means at the 0 0 and 0 so this gate exactly become 0 0 and 0 so this gate exactly become similarly, the normal case of this 1 so you can delete it.

Now i 2 stuck at 0 so this net is stuck at 0 if you think so i 2 stuck at 0 if you think so this one will become 0 this one will become 1. So this is a 1 and this two are same as in the previous combination so this gate is retained. So in fact what are we doing over here so just by this change this change from 1 to 0 then this is the old least which was available with us. So we are just finding out the what are the change this by change what are the changes in this gates we are finding out that is the bubble list and if something becomes equal to 0 0 rule that were deleting and if something is not become equivalent with just retaining.

So for  $i_2 = 0$  this is the single difference between the this one and this one so this gate is retained. Now  $i_2 = i_2 = 1$  stuck at 0 so  $i_2 = 1$  this 1 in to complex stuck at 0. So then what happens? This gate this 1 will a stuck at 0 so this 1 so this stuck at 0 so this 1 is a 1 so this is a 1 this is stuck at 0 so here it will be a 0 output is 0.

So now the thing is that for this case apply a 1 you 1 this is get a 0 so the the this  $i_2$  stuck at  $i_2 = 1$  this fault stuck at 0 we lead to these two gates is 0 0 0 which becomes equivalent to the gate under a normal condition whenever will applying 0 1 so this gate we cannot deleted.

(Refer Slide Time 51:53)

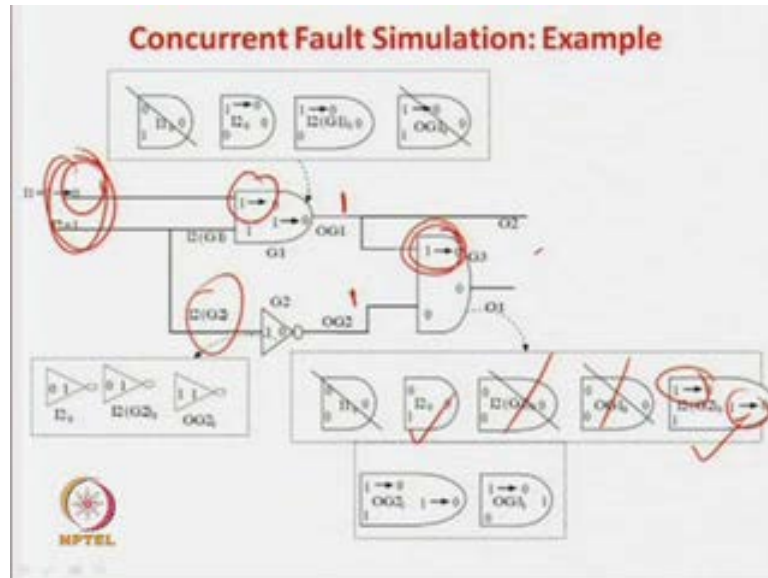


So similarly, you can easily find out that if this is a stuck at 0 over here  $i_2 = 1 = 0$  and the input become 0 0 0 which is again similar to the normal condition of this gate under 0 1. So this gate is deleted this one you find out that two of this gates deleted. So here two gates are deleted this one is retain now you just think of the case  $i_2 = 2$  stuck at 0 so this one is stuck at 0 if you think then what happens? Then this one will be 0 so this one be a 1 and here there will be a change from 0 to 1 because initially the pattern applied was the 1 over here.

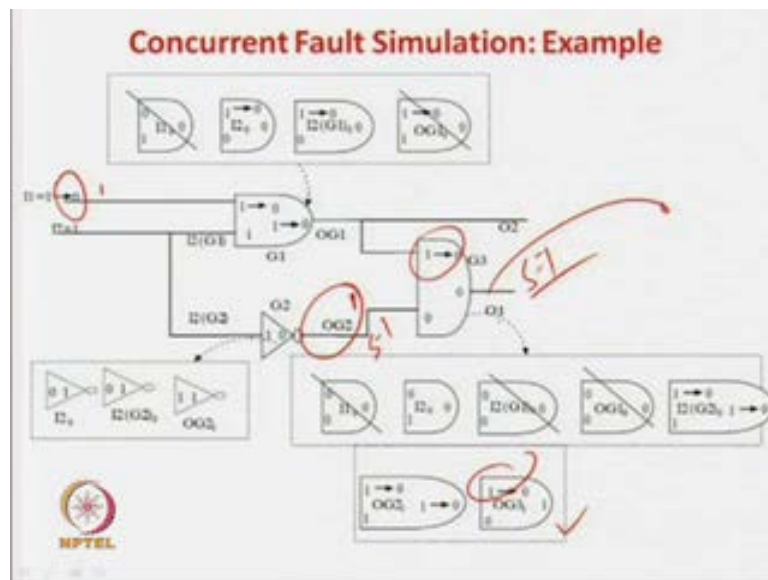
So the pattern one 1 was a one was a applied 1 over here and if is a stuck at 0 fault here. So there is no affect over here so no affect was over here so it was the 1 1 output was a 1. So your gate was looking like 1 1 and the output was a 1 so this was the key old computation. Now you are applying this change, so this change is getting to this change,

this change is getting to a this change. So this gate will be now having 0 1 0 but, still this gate will be retained because they are signal difference over here.

(Refer Slide Time 52:55)



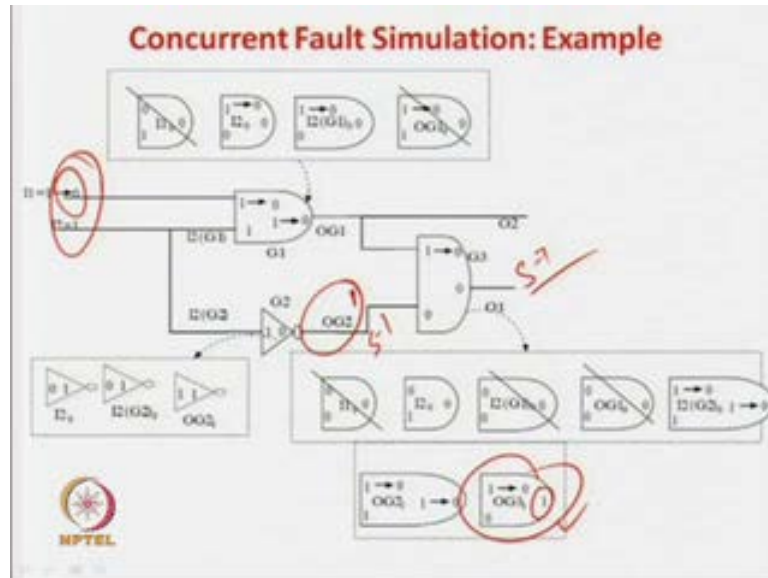
(Refer Slide Time 53:14)



So you can see that because of this change will this one this competition is not affected only this change and this change you have to get so, some cases this faults are means a competition is same so this gate is retained that i this gate i 2 g 1 i 2 sorry this i 2 g 2 stuck at 0 sorry this one this i two g two stuck at 0 this case is retained over here because

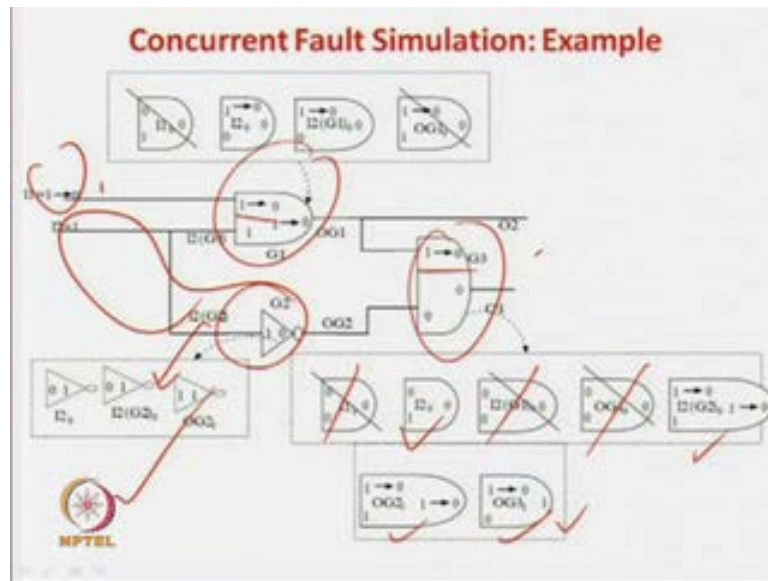
of signal difference but, only you have to compute this change in the input which is arise because of this change so this is retained.

(Refer Slide Time 53:43)

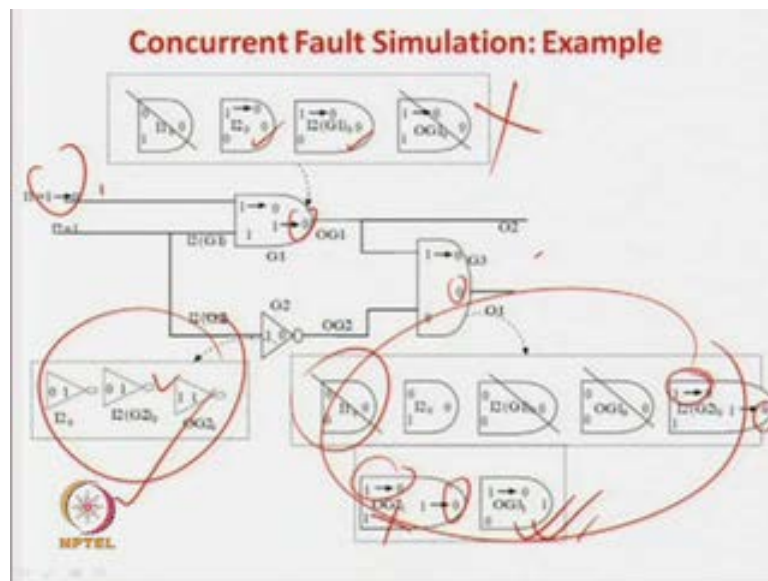


Now similarly, you can easily verified that what are the other gate which are retained is o 2 g 1 stuck at 0 that this one if you are getting going get a stuck at 1 then a this one will be changed in a output will be changed this 1 will be a 1 but, there are signal difference in this normal case so this is retain n o g 3 this is the output o g 3 so you just think of this case. So this fault is also will going to be retained because this is stuck at 1 and this is the signal change only because of this signal change so you write like this and a big gate which corresponds to 1 stuck at 1 is still retained because there are signal difference, that is just because of this change application so it will become 0 0 0 in a normal case but, it is stuck at 1 the answer here will be a 1 so by the previous competition only you can see that this gate side this gate is retained.

(Refer Slide Time 53:58)



(Refer Slide Time 54:32)

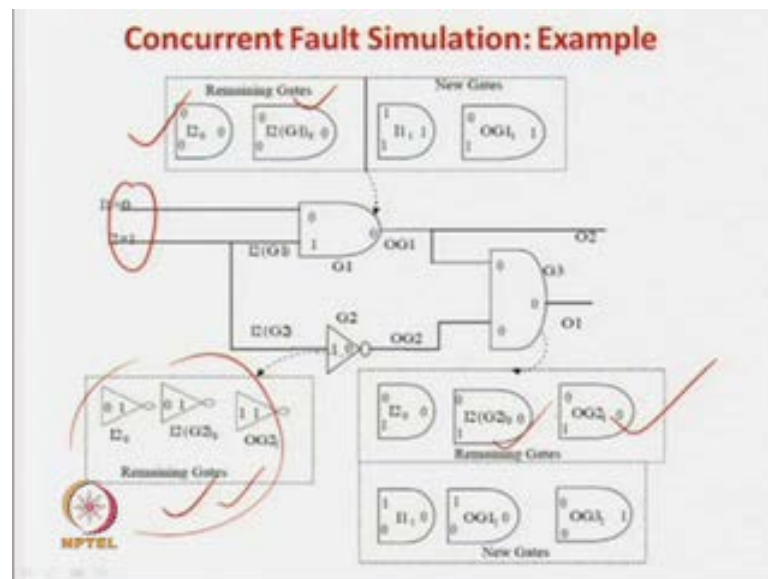


So what we have done basically by this signal change what happened? By this thing this is signal change which is happened which is not effecting this gate in any way so directly you retain all the gates without the anything but, for this gate and fault this gate there is some changes this is signal change that is happened. So based on that you find that this gate, this gate, this gate is deleted because here the signal after this change it will become 0 0 0 and this gates will also become 0 0 0 which becomes equivalent to this so this are dropped but, for this thing things this one, this one, this one and this one the retained because there is signal change.

So now you can easily derive that which are the fault which can be detected over this. So you can find out that this is a signal this signal fault which is detected because the here the answer is a 0 and here answer is the 1 so 1 fault is detected but, this one now the answer becomes 0 and this one the answer become 0. So this fault does not get detected by this 1 and in this gate you can see that none of the fault gate detected because the answer is the 0 over here 0 over here. So in this case also the answer is 0 over here so n1 of the fault gate detected by no additional fault I mean what you can say by this the all only this fault gate is detected

So but, what essentially we have done? So we are retain this gate directly without doing for any kind of additional computation and in this case also we have gone for some computation but, the computation are minimal here actually did not do any computation but, still directly you can find out that we can deleted in this case only this some small computations which are point out by we have to do and then we can find out the fault case.

(Refer Slide Time 55:23)

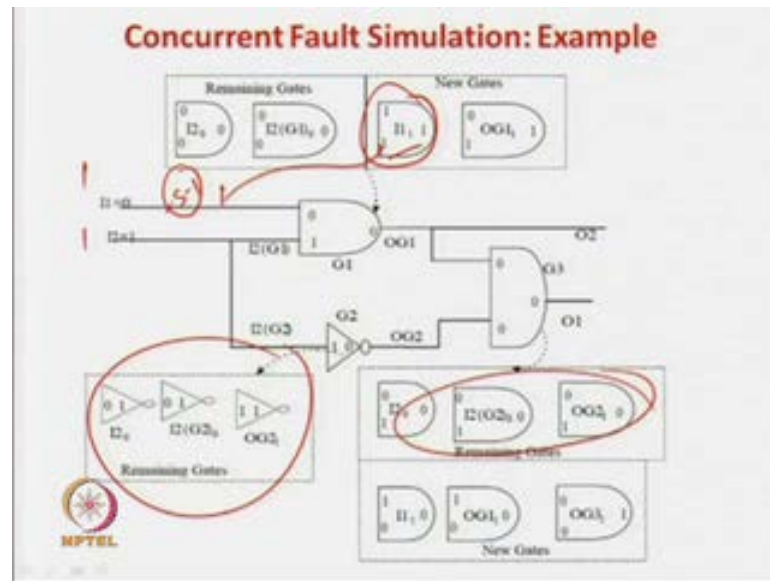


So now after doing this what we have done we have just say that these are the remaining gates, this are the remaining gates and this are the remaining gates. Now these are small thing you have to do what by what advantage you have got so we have just converted so what you can call that by concurrent fault stimulation as we are retaining the information about the gates. So we can just for the new change in the input what we can do is that



without doing for any computation we can find out this one, we can find out this one and we can find out this one with this is without any computation and we have some small computation are required. So we are using reusing the information whenever the pattern least but, now some new faults will also been added for that obviously you have to do the re-computation. So what is the advantages of concurrent fault stimulation?

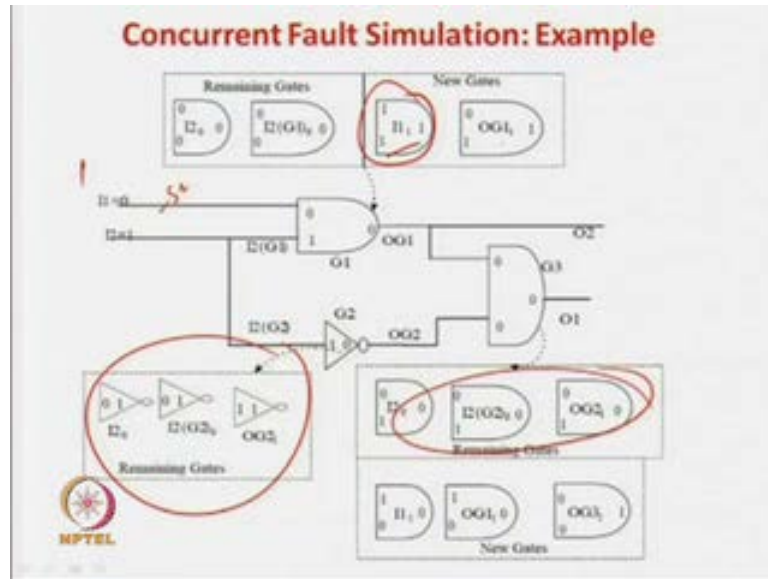
(Refer Slide Time 56:07)



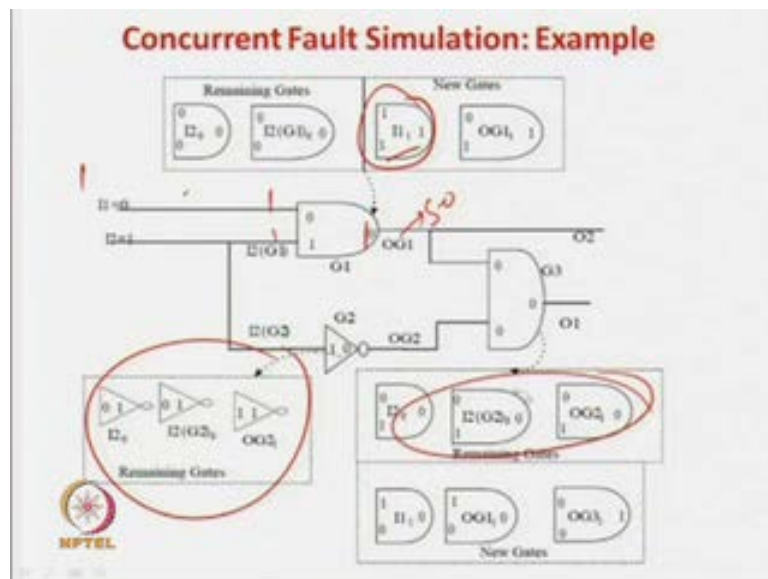
So whenever a new pattern is applied so what we are doing? Whatever is possible those informations we are retaining like this information we are retaining, this information we are retaining, this information we are retaining but, there will some new faults will also get added for that you have complete effects so in concurrent fault in deducting fault stimulation we are remaining in remembering nothing but, in concurrent for stimulation we are remembering whatever is possible.

Like for example, I tell you what are the new gates that are added? So now you are applying a 0 and a 1. So obviously what is the new fault that is going to be added the stuck at 1 fault over here this is i 1 stuck at 1 so it is i 1 stuck at 1 then what is the signal value it will be a 1 over here, it is the 0 and the output is 0. So this is 1 fault it has to be added this stuck at 1.

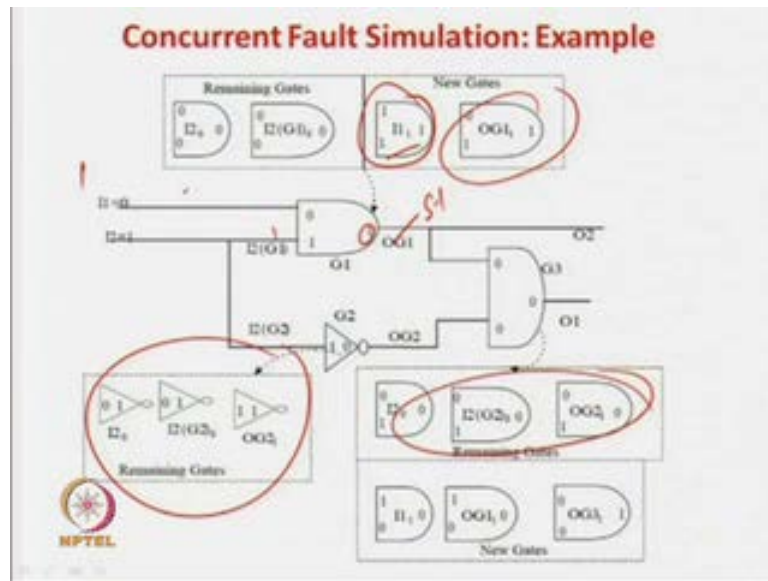
(Refer Slide Time 56:49)



(Refer Slide Time 57:02)

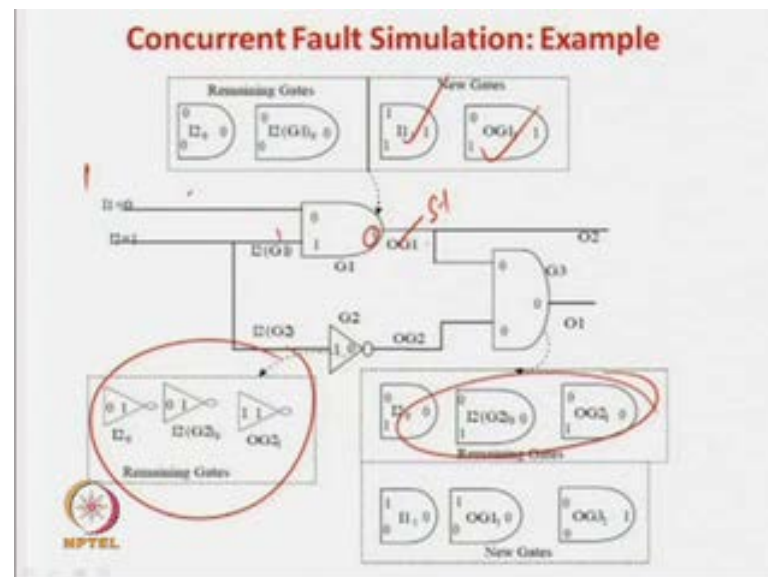


(Refer Slide Time 57:12)

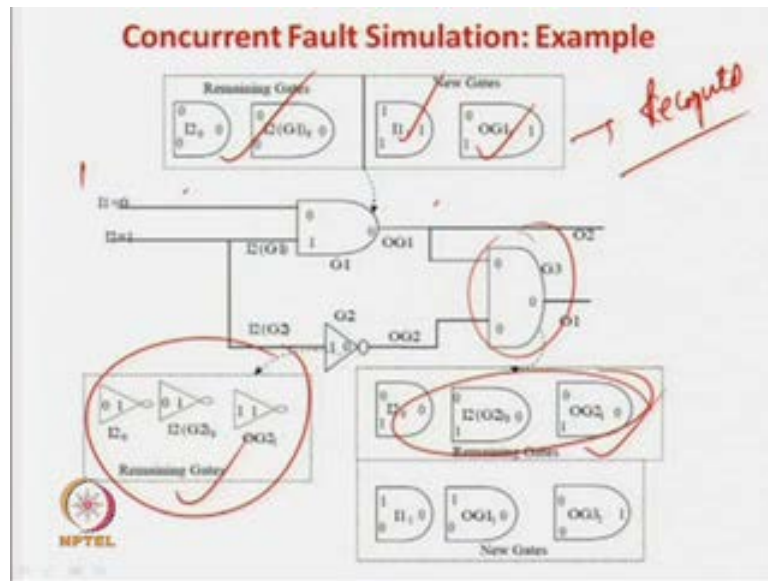


Previously you are applying 1 over here so obviously there is no question of stuck at 1 was there. Now you are changing value from 1 1 to 0 1. So obviously 1 stuck at 1 fault and this 1 will be applied now you see now this is a this is a new fault now in this case previously you are you were applying previously if you remember you are applying 1 1 and the answer was 1. So in this case we were deducting a stuck at 0 fault.

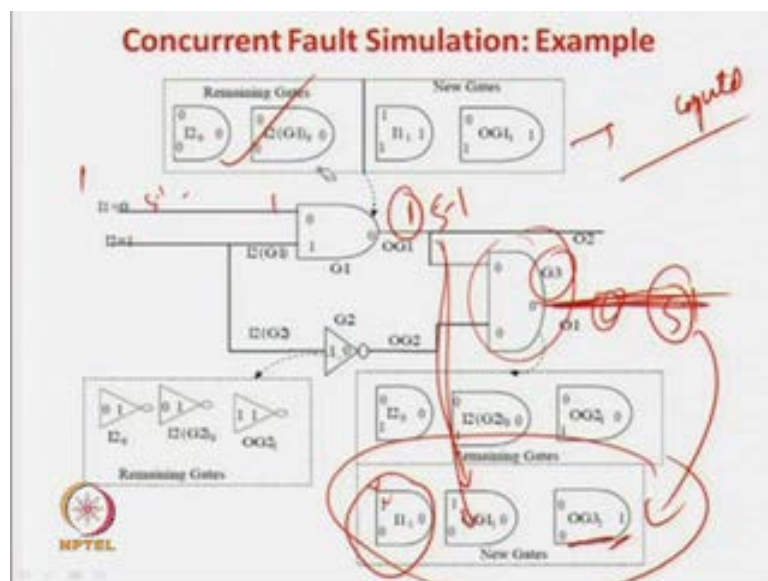
(Refer Slide Time 57:24)



(Refer Slide Time 57:29)



(Refer Slide Time 57:55)

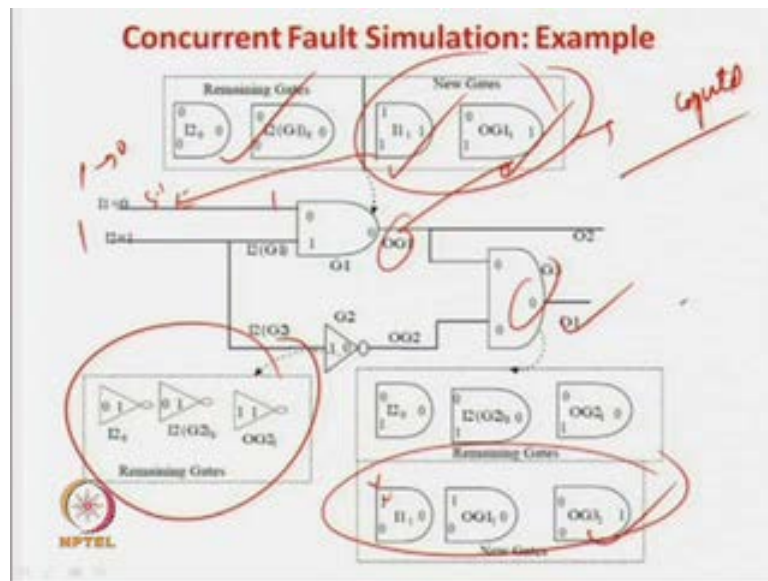


Now because you are changing value from 1 1 to 0 1. So obviously, the answer here as a 0 so obviously, 1 stuck at 1 fault a o g will be added over here. So in this case it is 0 0 0 1 but, we have the instead of 0 answer is 1 so two more gates have been added and obviously this two faults are detected here and you have to test see that these are all recomputed for here detect concurrent fault stimulation and deducting fault stimulation are no difference.

So you have to again re-compute all this things but, where we have saved we have saved here, we have saved here and we have saved here. So whatever is possible here remembering similarly, for this gate if you change some new gates will be added. So what is the new gate that is the added? So now obviously you can see that this is as tuck at 1 fault here this is this is actually 1 this stuck at 1 fault over here. So we stuck at 1 fault then the answer is the 1 over here so there will be a signal changing here.

So this gate will be added by this one so stuck at 1 here means the answer is a 1 over here so instead of 0 the answer will be 1 1 1 0. So this stuck at 1 fault is a new fault which is added, which is gate retained over here then actually o g 1 so this one was the new gate, this one there is stuck at 1 so initially it was stuck at 0 so this is new gate is also added over here. So this is the 1 1 is answer is 0 and of course, previously I think a previously when you are applying 0 0 sorry previous going we applying 1 1 so the answer was something whatever.

(Refer Slide Time 59:17)

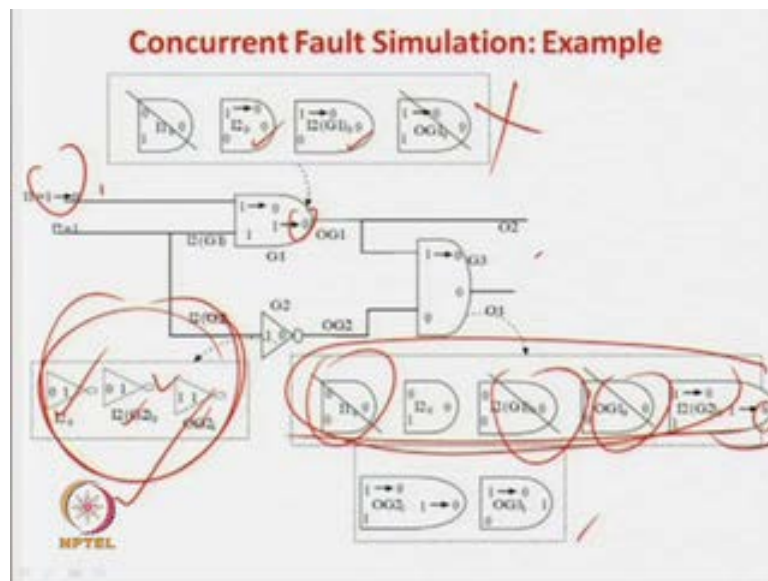


So now here it was it is a stuck at 0 in the previous case now you are have to apply a because here it is 0 0 the answer is 0. So a stuck at 1 fault at this net will be applicable over here. So these are the new gates this 1 because of the stuck at 1 fault which is newly added over here because of the change in pattern. So these are the two gates which is the added and now in this case the answer is 0 over here 0 0 the answer 0 over here.

So you can add a stuck at 1 fault here this is 0 3 stuck at 1 input is 0 0 the answer is 1 because this is the stuck at 1. So three gates get added over here so you can say that because of this pattern so the output here is a 0. So what fault can be detected this stuck at 1 fault can be detected over here and this and this two fault that is a 0 g stuck at 1 this fault is detected over here, this is 0 g stuck at 1 and i 1 stuck at 1.

So these three faults are again detected by the pattern 0 over here. So this 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 1 1 and whenever you change the pattern to 1 0 so computation regarding the remaining gates are very very minimal or no computation is required the just gate the value only for the new gate you have to reduce computation. So concurrent fault stimulation over detected fault advantage is there you retained whatever is possible in deductive fault nothing but, here you try to retain something as much as possible but, still then what is penalty we are paying that some of the gates like for example, the dead gates if you remember.

(Refer Slide Time 01:00:15)






(Refer Slide Time 01:00:43)

**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?

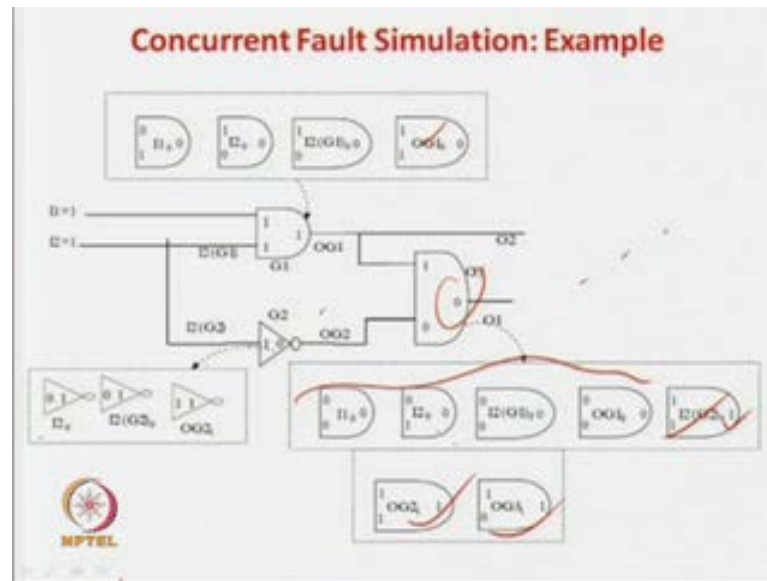
Next random pattern is  $i1=0, i2=1$ . In case of deductive fault simulation we need to repeat all the steps, while in case of concurrent simulation we will re-compute only the information that changes as a result of changes in signals triggered by  $i1$  (from 1 to 0).

 NPTEL

So this gates had no affect like this gate, this gate, this gate actually we are just giving it we are just carrying the dead animal you can thing whatever not having any affect, they could not deduct any affect fault in the previous case also not in the present case but, still we are giving this information because they maybe reuse in some other case. So that is by because of a carrying this dead animals you can think that some computation is same that is what is the idea. So here also you may not know that may the this, this, this, this gate would lead to any kind of fault detection but, still you have to retaining then because we feel that sometimes it may be possible to do this one. So that is basically the idea concurrent simulation.

So that is what is case now so what was the discuss and summarize so only three gates correspond to fault being detected at this one. So what is the requirement of seven gates in the affected list that is very important thing that is if you look at this example.

(Refer Slide Time 01:01:13)



(Refer Slide Time 01:01:21)

### 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 is  $I_1=0, I_2=1$ . 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  $I_1$  (from 1 to 0).

The slide contains text explaining the requirements of deductive fault simulation compared to concurrent fault simulation. It states that only three gates correspond to faults detected at OG3 (or primary output O1). It asks for the requirement of seven gates in the affected list. It then compares deductive fault simulation, which keeps information about only these three faults, to concurrent fault simulation. A specific example is given: the next random pattern is  $I_1=0, I_2=1$ . In deductive simulation, all steps need to be repeated, while in concurrent simulation, only the information that changes as a result of changes in signals triggered by  $I_1$  (from 1 to 0) needs to be re-computed. The NPTEL logo is visible in the bottom left corner.

So in this case that I say so only this gate, this gate and the this gate has a output difference by this 1 then why are you carrying this dead animals? So that is the question that is being asked over here, that only three gates detected this fault then why are you carrying this things the idea where was thing whenever change in the random pattern, then those dead gates may help in detected some other fault which you have seen and then this re-computation is same that is what is the idea of a concurrent fault simulation over a detective fault simulation.

(Refer Slide Time 01:01:50)

**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.

NPTEL

The detective fault simulation we do not carry anything, what we do? We do not carry anything we just for a pattern we find out what are the fault are rejectable in 1 1 go that is why in the fault list will have only those elements which are detectable and then we forget everything then we apply new pattern but, in case of your concurrent concurrent fault simulation of what we do? We even we compute some bubbles for all the gates bubbles are having some values. It the signals difference with the gate we keep that faulty gate in the bubble list and we retain all the gates and all the levels even if it (( )).

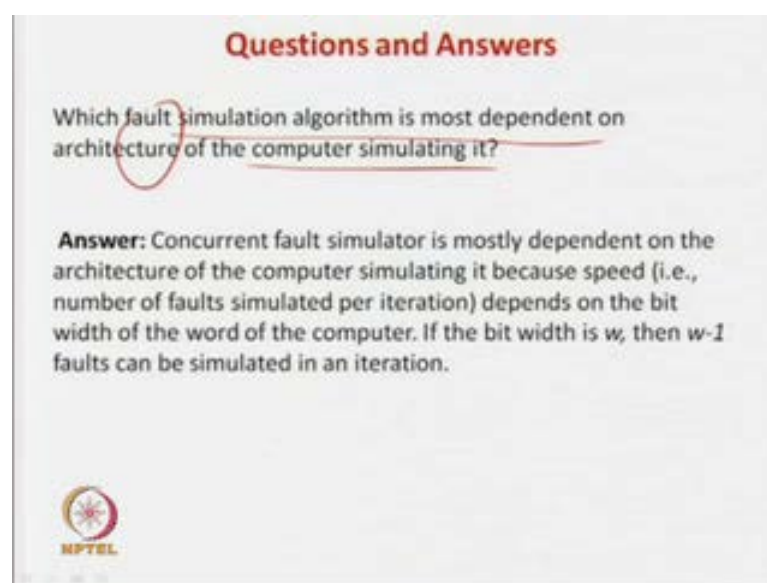
Now whatever faults are detected we are very happy because we can drop them but, the dead gates, that is they are having some signal difference with some gates but, they are not able detected a fault still we retaining them when we are giving another input, another random pattern. Then that new random pattern is that is dead gates may help to detect some faults. Then those we computations are failed. So that is what some by saving some computation but, still we have to carry the dead load of some gates in the memory. 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 random test pattern generation. We apply some random patterns and then we find out what are the faults will be detected. So you can serial which is the slowest one, you can have paralyze computers supports lot of parallelism then you can go for pattern fault simulation. Otherwise you can go for detective fault simulation or

concurrent fault simulation. If you using detective fault simulation in 1 go will find out what are the faults detected by a pattern. But, you do not remember anything next pattern we apply we have to read everything. But, in concurrent fault simulation you retain fault simulation that can refused but, for that you have to sometimes carry the dead load of gates which are not help in any kind of fault which and for the current pattern. But, the for the next pattern they made do some so we have to retain them but, now we see that experiment seen that this procedure of random pattern based fault generation whatever can help only for going about 90 percent of the fault. Remaining 10 percent of the faults are there will be hard to detective faults and you have to do by what you have to go by random sensitized propagate and justifier.

So in the next lecture what we are going to see? We are going to see if somebody or some algorithm can tell you that this 90 percent of the faults are easy to test fault. So apply random patterns for that. This 10 percent faults are difficult, so do not try for them. So that will very good so at least in the fault simulation we will not touch the ten difficult faults. We only touch the 90 easy faults. But, here if you nobody tells you that which is the easy fault, which is the difficult fault then you are in a blind case. Even for this ten difficult the test faults you are applying the random pattern generation based algorithm and you are failing every time. And all the random patterns you are not able to the test pattern generation.


(Refer Slide Time 01:04:22)



**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.

 NPTEL

So next class will see some kind of example of algorithms like scope algorithm call which will tell you which are easy fault and which difficult fault then we apply random pattern for the easy faults and you keep assign the difficult faults for sensitizing the propagate and justifier. Now before we conclude let us come to the question and answer session. So we are saying that what is the main advantage of compiled code simulation, event driven simulation? So here what is the answer, so we have already discussed many times like it know for the concurrent fault simulation, detective fault simulation and normal fault simulation and normal circuit simulation kind.

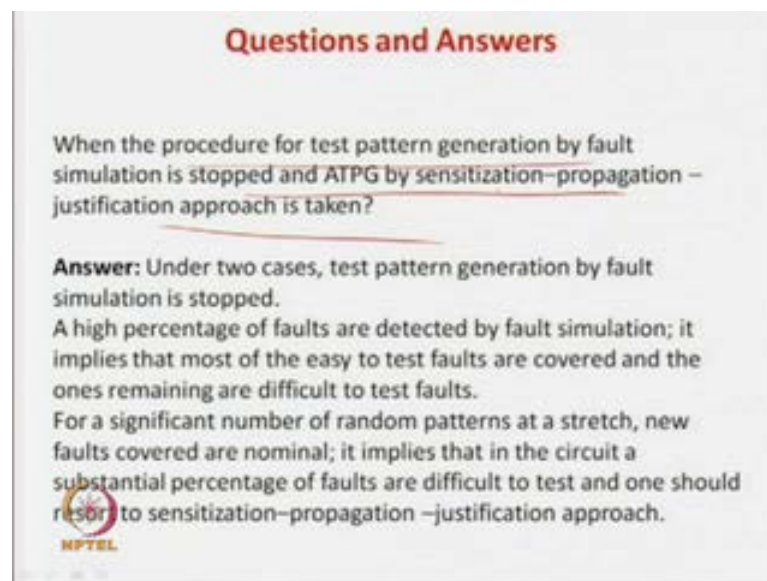
So in compiled code simulation, we have convert the circuit into a (( )) and you execute for a pattern. Then you forget everything you just change anything, then you have to redo everything. But, what you call event driven simulation we remember whatever was we have done in the previous iteration. And only those changes for a fault or for a pattern whatever are the small changes in the some regions or the circuit only there will do the re computation. So what is the advantage disadvantage obviously event driven simulation is faster. Because you do not re compute anything which is redundant. But, at the same time you do remember the values from the previous iteration so you have to have some extra thing in memory, but, gain is free because we are avoiding re computation.

Second question is among all the four faults simulation algorithm which you have seen like serial, parallel, detective and concurrent which depends mainly on the computer architecture. So among them we have seen only the parallel fault simulation is depend on architecture because to these net to assign an array. And the length of the array this processed parallelly. So you have to think about bit parallel, so if you compute a 32 parallel so you can have maximum 32 bit array in all the inputs.

And for all other case like serial, detective and concurrent we do not at all think about the memory of the computer we do not think about the parallelism of the computer. You will never think that what are the fault list size, what are the affected gate list size and accordingly we attach to each gate or attach to the fault list. So that depends nothing on the parallelism of the computer. It just a it just a program and we think that what is the case we compute that for this fault gate is these are the fault gate you have to add there is fault version you have to add for this gate this is the fault gate you have just we keep it.

We do not think what is the architecture what is the parallelism of the computer available and that many fault list you have to add nothing like that. So again this gate as so many gates in the fault just keep them. But, in case of parallel fault simulation only those number of patterns or those number of bits can be operated on that much parallelism is supported by your computer. So that is why among all the four faults simulation algorithms parallel fault simulation is mainly dependent on the architecture.

(Refer Slide Time 01:06:18)



**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.

**NPTTEL**

The last question is when the procedure for test pattern generation by random this 1 is stopped at a t p g algorithm by proposition sensitization is taken. That is we start applying a random test patterns and find out the ten faults are detected next random pattern line faults are keep on going for it. And after sometime you have to stop then what is the stopping criteria. Stopping criteria is when you find out that and applying some random patterns, the number of new faults with a detecting is very very less or sometimes it can be 0.

Then you apply a random pattern number of faults detected initially will be said 10, next random pattern 12 another random pattern 8. You will keep on doing it. But, other sometime you will find that the number of the random patterns you are applying with the number of new pulse rating is 0 or very very less. So at that time you can say I have to stop because adding of random pattern is not helping me in any way and I have to go for sensitize propagate and justify approach.



So thank you and we come to end of the lecture on fault simulation in the three hours lecture was on that. And in the next class we will see the scan somehow how we find out which are the difficult to test fault and which are the easy to test fault. So difficult to test fault we will go for sensitize propagate and justify approach and easy faults we will go for random pattern with . So with this we come to the end.

Thank you