**INIDIAN INSTITUTE OF TECHNOLOGY GUWAHATI**

**NPTEL**
**NPTEL O NLINE CERTIFICATION COURSE**
**An Initiative of MHRD**

**VLSI Design, Verification & Test**

**Dr. Santhosh Biswas**
**Department of CSE**
**IIT Guwahati**

**Design Verification and Test of**
**Digital VLSI Circuits**
**NPTEL Video Course**

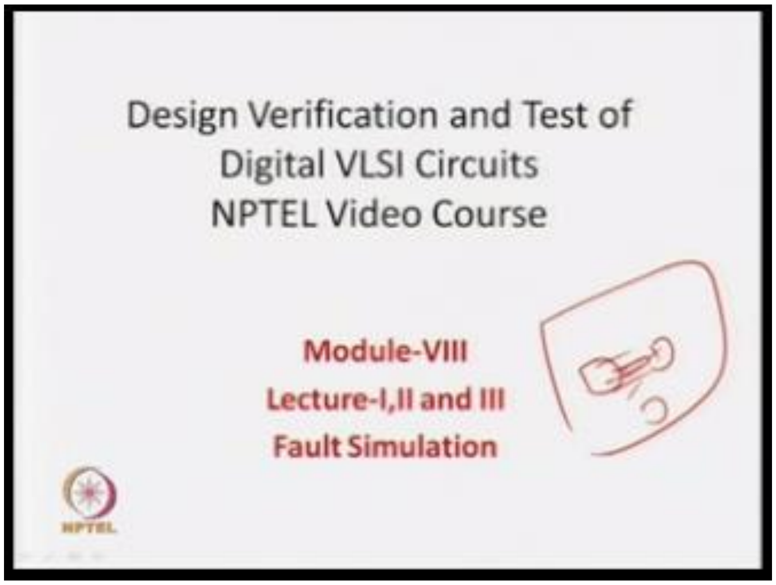
# Module-VIII
# Lecture-I, II and III
# Fault Simulation

Okay, welcome to the next module on testing this is module number 8 in the last one module what you have seen is that we have understand what is basic of VLSI testing why do we require testing so the main point of requirement of testing we saw was that as the yield of all VLSI circuits are lower because we are continuously moving into higher and higher some micro technologies are advanced technologies so our accuracy or design I mean correctness has not yet match you and then we move to another high technology so I mean there is always chance of failures in the chips or defects in the chips.

And our main goal is to eliminate out the faulty chips and then sell out or give to the customers if normal once. So now the yield is lower because of the consciously up liftmen in technology so what the main aim is that we have to do good testing or somewhat VLSI testing kind of staff is required so that we can eliminate out the good chips and discard the faulty chips and for that testing is important paradigm in case of circuits.

Next we have seen that testing can be functional and structural so in functional testing if there are any input we give $2^n$ all combinations of input and check here is a golden response so here our main emphasis is to find that the chief performs find or it performs okay, for all the combination of inputs, but as you know that if the number of inputs of a circuit is over 100 or 200 then $2^n$ is a invisible number and testing will take years and it will run into mean such huge time there is not visible. So we have gone for what you called as structural testing so in structural testing what we do we are not much considered about the functional it is of the circuit whether we break the circuit.

(Refer Slide Time: 02:07)



Design Verification and Test of
Digital VLSI Circuits
NPTEL Video Course

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

Into whole circuit we have seen we break them up into small, small modules and each module we test them functionally and internal wires you can say that we test structurally, so that in the case we have seen that if your circuit modules or which we are breaking up into sub modules, which your testing functionally then if your modules are quite large then what happens is the number of test input combination are also high at because of $2^n$ is also not a very small number, if you are some modules are higher.

But the advantage that you gain is that the number of intermediate lines between the sun modules are lower and so testing or controlling and observing this intermediate line is easier. On the other hand if you make small modules of your whole circuit that is if you have large circuit then you have small, small modules then number of inputs patterns per module is very less that is an advantage but then the number of intermediate lines in between the modules is very high and controlling them and observing them remains a challenge for which you read either pin outs or you will require multiplexer elements or you require a shift register and so forth.

And then we have seen to eliminate out all the problems we have seen one for structural testing using fault module.  So what is the fault module in the structural testing with fault module would not concern any, we are not concern at all about any kind of functionality of the circuit whether we want to find out or whether we want to verify that given circuit there is no fault from the fault list that you have seen that stuck fault module is one will accepted fault module.

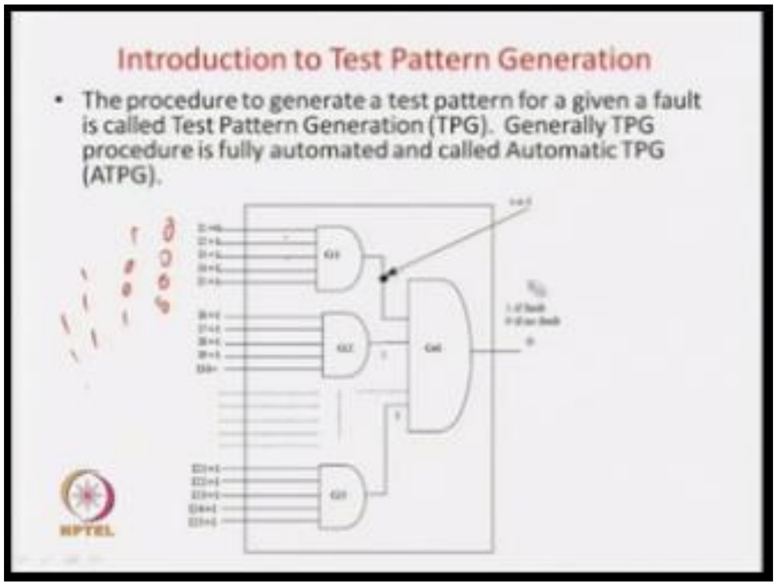
In case which we have seen that we have given circuit with AND gates and OR gates then you have to apply patterns so that you are able to verify there is no stuck at false at any of the line, then we have seen the advantages of stuck fault model is that we d o not require intermediate pin outs we do not require intermediate multiplexers or shift registers to control and observe the lines.  Further the number of one test pattern can be detect more than one faults, so the advantages that if there are two and falls in our circuit where is there number of lines still two n number of test patterns are not require you require much less number of test patterns.

Because one test pattern can detect a large number of faults.  So there are lot of advantages when are you going for structural test with the fault module and stuck fault model you will accepted one and also you have seen that structural testing with the stuck fault module is widely accepted because if you can verify that the circuit does not have any stuck at faults then you can be 99.9% plus sure that the circuit is functionally or you can say that it can be confident 99.9% accuracy that the circuit is free of defects.

So all those things make actually structural testing with circuit fault module, widely accepted practice or widely accepted module. So in this module what we are going to see is that how can you generator the test pattern that is given circuit with end number of false then after fault collapsing you can reduce the number of faults then for the remaining whatever the case like given circuit, and there are different faults which are the final list which has been direct after falls collapsing by dominance and equivalence then how you can generate test patterns which can detect  default that is called test pattern  and generation.

So in this module or in this module number 8 our main emphasis will be how to get test patterns which can determine which can or in other words how you can find out patterns which can detect the stuck at faults which are in your circuit. So in the first lecture from this one of fault stimulation and this is quite long lecture so you can see will be spending three days or you can called 3 modules and 3 some modules 1, 2 and 3 will be fault stimulation.

(Refer Slide Time: 05:29)



## Introduction to Test Pattern Generation

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

G1
G2
G4
G3
NPTEL

So let us explore this is our so what is our test pattern generation as I told you so given circuit fault if you can this is fault say then automatically you have to find out which pattern can test this fault.  So this called actually test pattern generation this is called test pattern generation and now if the circuit is quite large then you need to obvious automatic and therefore it is called automatic test pattern generation that is ATPG.

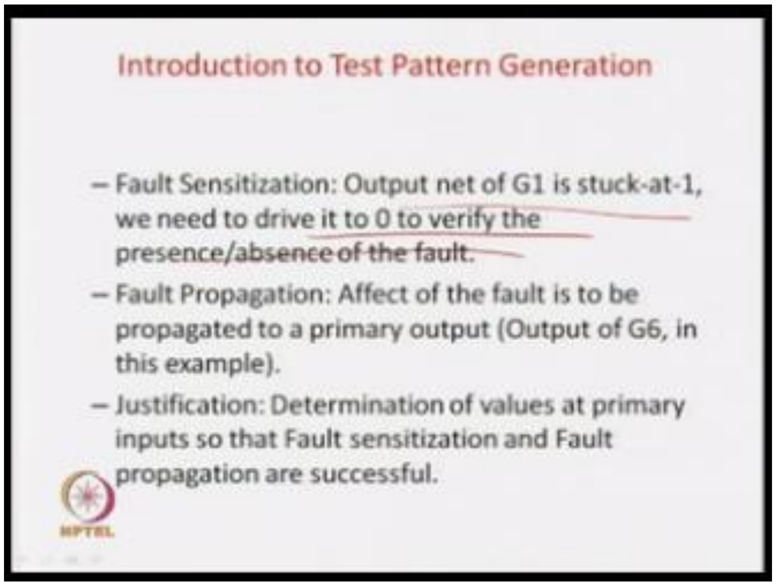
So in this module we are going to look into details on this one, say for example this is your circuit which is also see in your last lecture. Now so this is a stuck at 1  now say  I want to find out the test pattern which can detect this fault, okay so that is my question that is the main problem of ATPG, so now you have to also try for fault at this point and fault at this point and so forth.

Obviously the number of false will be limited by the fault collapsing algorithm.  Now actually this is the three process stages so you can also find it out as what I mean what you called adopt, in adopt manner also you can find out for example, you can say the stuck at 1 so obviously you have to apply a 0 and then this value should be propagated this affect fault which we propagated to the AND gate.

So this should be 1s and then actually the fault affect is you have apply 0 normal case and the circuit one to fault case it will be 1 and this fault affect has to be propagated at the output so all the other input of the AND gates said to be 1 then if all the AND gates output are to be 1 then all the inputs have to be one for all the gates and for this case we have get 0 over here so either the inputs can be all 0s or it can be  1001 accepting 1111  all 1s any other pattern can be applied so that will be a test pattern generation.

There is test pattern for this fault that is any other combination other than 1111 and all this have to be one accepting all 1s at this gate any other pattern like 0111 or any other patterns accepting all 1s will test this fault.  But there is proper algorithm we require to do this there is actually called fault sensation propagation and justification. Let us see what we does so first is called the again just like or you previous we have discussed the same thing so the first is called fault sanitation.

(Refer Slide Time: 07:28)

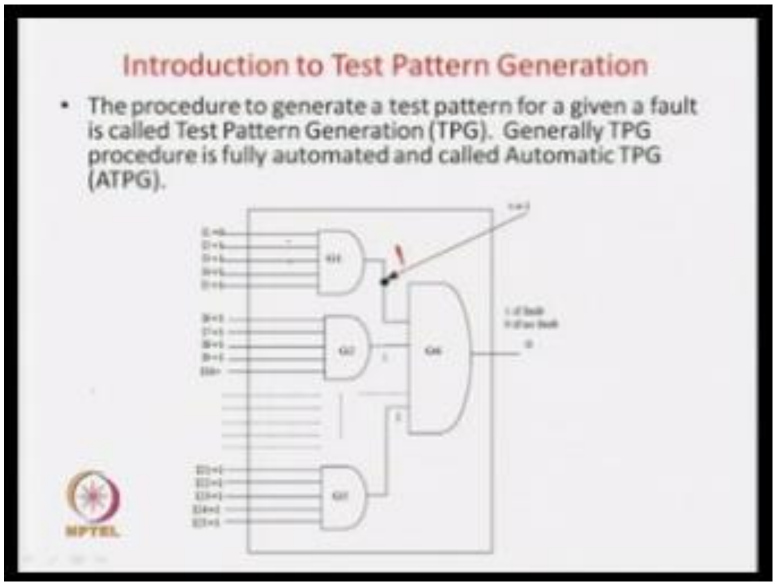


## Introduction to Test Pattern Generation

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

So what you mean by fault sensation we say that the output net of G1 is stuck at 1 we need to derive to 0 to verify the presents or absences of fault, so just you can take the analogy of electric bulk which is used, which is so now if you want to test it you have to make the switch on, so if you switch the bulb on then if it is fused then the bulb is not use.  That means what in that case your sent sting the fuse fault. The bulb is fused is so is the fault you sensation. So in this case they are saying that you are sensitizing it.

(Refer Slide Time: 07:57)



## Introduction to Test Pattern Generation

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

So sensitizing means if it is a stuck at 1 so you have to apply the reverse value, so I have apply a 1 over here there is actually called sensitization. Now next step is actually called if the fault probation and now as already discussed in the last chapter or I mean last lecture that for structural testing with fault models you did not need to bring out  any extra pins neither you require any kind of a multiplexer I mean neither you require what you call a shift register kind of a thing, that  means if you have a circuit and then you have a lot of gates inside and reinforce structural these are the primary outputs and these are the primary input.

Then we do not require any kind of extra pin outs or extra what you called this shift register to control this these things are not required, so in structural testing we structural fault model structural testing with fault model, so what is our idea  is that whatever you have  to do you have to do with from the primary inputs  and the primary outputs that is why what we have do this fault effect so what are saying this fault effect the fault is what in the normal case sorry, this is stuck at 1 so you have to apply a 0 and so the effect is this one which is actually 0 in case of fault sorry normal fault 0 and fault is 1 this effect has to be propagated to this output.

So because whatever you do only do at this output we cannot have this pin out is not allowed, no pin out is in fact extra pin out is allowed, so you have to bring this out. So now in the output also effect will be this one so you already retain one is fault and 0 is no fault. Now this is actually call the fault propagation now the fault propagation is propagate. Now you have to justify that, what do you mean by justification, does you mean by determination of values at primary inputs so that fault sensitization and fault propagation are successful.

Let us see what does it means, so again and I told you, you do not have any multiplexer arrangement here you do not have any multiplexer arrangement here you do not have any multiplexer arrangement here, so in directly what is happening you cannot control this lines okay, so I mean to have this fault value propagated from this point from this point to this point you cannot for any additional control here.

So whatever control you have to do at these lines and this lines you have to do only from the primary input so our structural testing with fault module is nothing but controlling through the primary input and observing through the primary outputs. So now you have to sensitization the propagation to justify so have to justify that this effect or this input and this input is by controlling the primary inputs you have to justify that these things are successful.

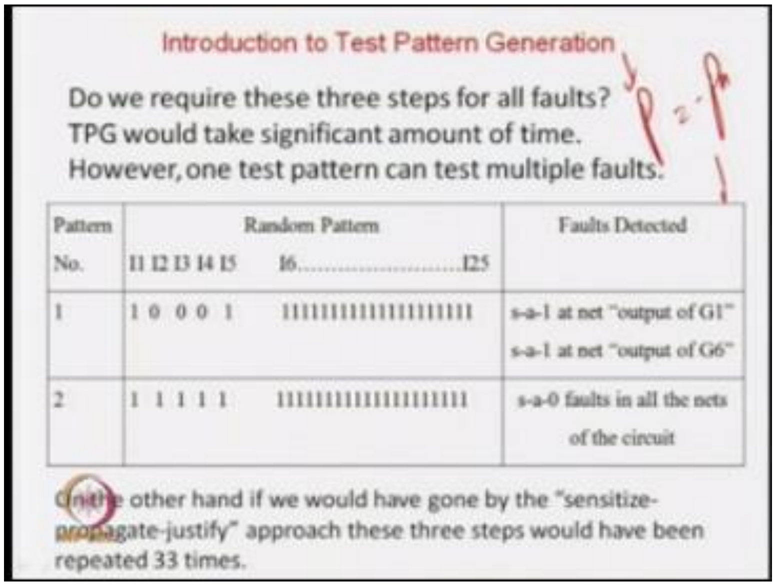
So it is an AND gate so you just go from one level so next level is that this effect has be propagated to this one, so if this effect has to be propagated all the other inputs of this AND gate are has to be 1. So this is the level 1 justification, now this is level one justification, right now again now what happen so again level 2 justification is saying that output of this gate is one and one put of this gate is one and output of this gate is 1, so how can you get this justification of the second level to be 1 all input of AND gate has to be 1.

Similarly for all this has to be one, now remains this one so in this case you require of value of 0 so this has to be justified at level 2, so again justification AND gate output to be 0 is all pattern or any pattern excepting all 1s, so your ATPG algorithm will generate 0111 it can generate 0000 any one of that is arbitrary and that depends on the some level of few circuit or some accuracy that we will see in the when you we be dealing in deals or ATPG we will it will determine whether you will generator 0111 or all 0 or 0001 it depends on major effective.

But for the time being the ATPG generation algorithm we will justify this 0 by any pattern which is 0111, 00 anything so this case we will assume that this is pattern that is 0111 has been generator.  And for justification of 11 in this case you get 1111 so test pattern generator is 0111 and all other inputs are 11, so it tests our stuck at 0  fault , sorry a stuck at 1 fault here. So this is basic steps of faults sensitization propagation and justification so this for all the false if you do this you are going to get what do you call or test pattern generation for all the fault.

Now when all the test patterns are generated they are stored in the memory so when this IC is fabricated and it comes from the what do you call fabrication unit so each IC you have to apply those inputs and you have to verify whether none of the false stuck at fault  will be present, so if you apply all the test patterns so you automatically we will find out that and if none of the test patterns is giving otherwise result that is if all the test patterns are verifying that there is no fault so you can say that your circuit is free of stuck at faults and it can be shift to the market.

(Refer Slide Time: 12:27)



## Introduction to Test Pattern Generation

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

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

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

So as I already told you so the test pattern generation procedure a root generate any pattern in the table 1, so it can generate all 0s with all 1s so this can be 1111 it can be 110 so $2^{25-1}$ test patterns can be possible for the stuck at fault, and it depends on the ATPG  algorithm whether we will generate 0000 for the first AND gate or 0001 for this one or 0111 for the first gate so it depends by anything you would do your job because we require 0 at the output of the first gate and all 1s and  1s a t the output of all the other AND gates, right.

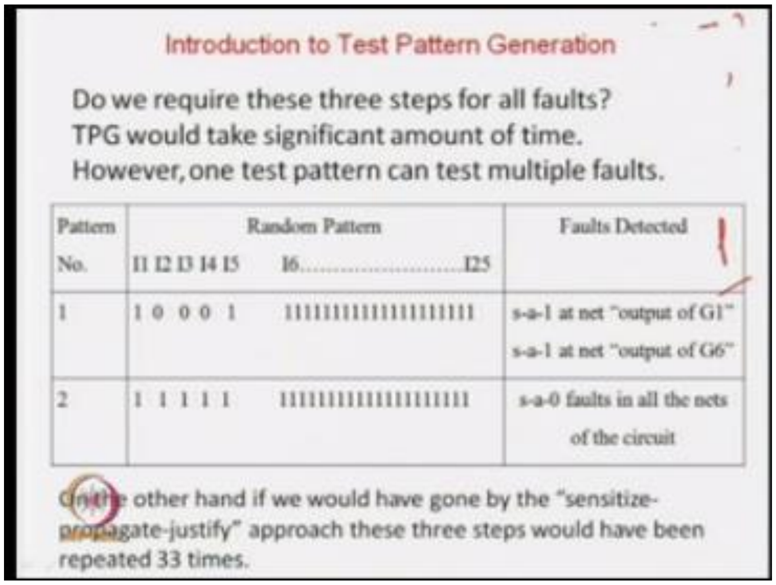
Now the question arise this that do you require to do this  individually for all faults, so now what do our goal in the  last module we have seen that our is to reduce the test generation type for the test application so we have to go down in time so from $2^n$  input patterns we came to k when k it he number of stack at faults that is 2 l so it is nl so there are 2 number of stack at 0 can be possible and  if there are 2k number of sorry if they are n number of lies and 2 n number stack at faults can be possible so k patters that may for each stack faults if we assume in the one pattern required so k pattern is 2 n patterns is required in the maximum case but again you can see that by structural fault coalescing by domains and equivalent if that is much less leave in k.

So it is k by a fact k is also k much reduce than k so even if there are n lies in the circuit number of test pattern I mean stack at faults are much less than 2 end to the collapsing and so test pattern are requires the much less than so now what again we are going we are going another level down so that even if we have something take p number of faults after stack after all the fault collapsing so we require at least p number of test patterns to detect it because 2n is the number of faults or for the 2n is the number of if n is the nets in the circuits.

So at last you have 2n stack at faults stack at 0 stack at 1 each line like collapsing we have reduced it p now can go even less than p can we have any number of can we have we have declare say that the number test pattern requires even less than p the answer is yes because sometimes you may find out that 1 test pattern can detect more than one number of that also can be possible so that is actually called mean what you call single pattern can detect multiple number of.

So there fault if you are repeating if you are taking the approach of sensory propagate and justify so what you are doing you first take one fault the you do sensor propagate and justify and generate one pattern say p1 you are generating for this now you take another pattern do a sensitization propagation and justification done you get pattern p2 similarly you keep on doing this keep on doing this and after that you say gate pl is the number of fault with $l^{th}$ number fault you do and you generate safety pk pl sorry pl is the pattern I am sorry what is what we were saying that you have fault p.

(Refer Slide Time: 15:15)



## Introduction to Test Pattern Generation

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

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

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

Then you fault p2 then you have fault pl so for that you are generating say pp1 this pattern 1 then for p fault number 2 you are generating pattern 2 and here you generate pattern for fault PLK but now you find out that TL 1 that is for that is for fault p1 if the pattern we are PPl1 is equal to PPLK  PPL or PL that is for the n number of fault and the first number of fault the same pattern is generated by automatic test in generation that is sensory propagate and justify so you are at the last because the same pattern that is at PPL1 after the pattern for the first fault you have generated that is what is happening that is the same pattern is detecting more of fault that is the first pattern is deducting fault number as an fault number so unnecessarily we are wasting time in automatic test for in generation sensory propagation and justify with same pattern can retain multiple number of.

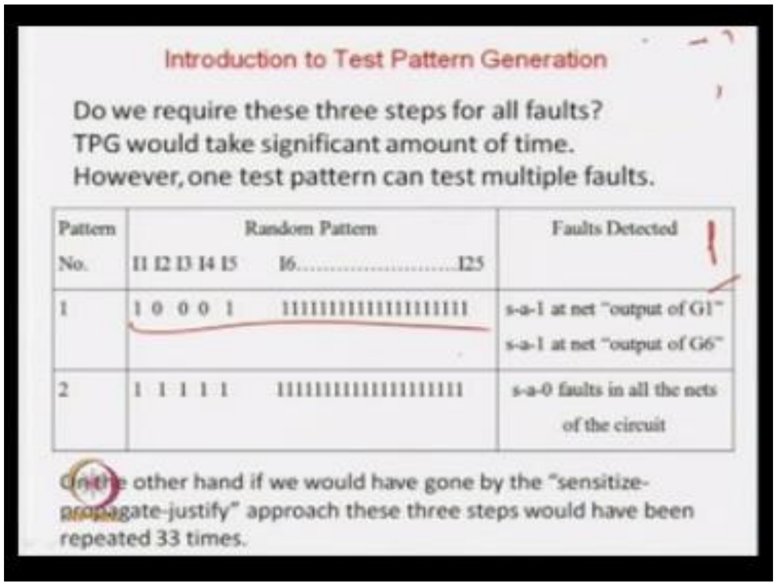
So let us think try to think it in some other way can I do something so that we apply one pattern and find out how many faults are detected and then remove those faults and then try to apply another pattern and say how many faults are detected and so forth so in this case what we will say we will save the unnecessarily load of the case where one fault can one pattern can deduced multiple number of faults like in this case pattern number one is deducting 1 as well as pattern what you call the fault number l.

So these are regency but if you have say that if I apply say pattern PP1 and then you find out which fault are deducted then automatically you will be able to tell that deceits the fault P1 and also gets fault T sorry PL so in this case you will be saved from executing the same algorithm twice fir P fault number 1 and fault number n so we will see that so that is actually called fault deduction by random what you called random pattern generation.

So in this case sensitive propagation and justify by this approach when you are going for test it is called a determine approach in this case you are sensitive fault propagating the affect and you are justifying these and when you are doing randomly you can apply a pattern and then see how many faults are deducted and again repeat this then I call this random pattern generation.

So we will see that so both them have their own advantages so we will see as on now it appears that which is more advantages the advantages you apply a pattern and see how many fault are deducted let u consider the same circuit if you consider that circuit.

(Refer Slide Time: 17:32)



## Introduction to Test Pattern Generation

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

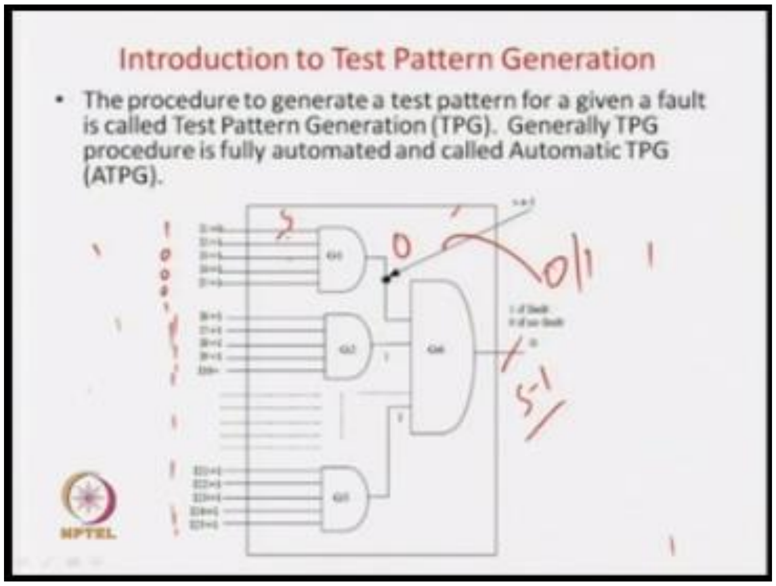
| Pattern No. | Random Pattern I1 I2 I3 I4 I5 I6.........................I25 | Faults Detected |
|---|---|---|
| 1 | 1 0 0 0 1 1111111111111111111 | s-a-1 at net "output of G1" s-a-1 at net "output of G6" |
| 2 | 1 1 1 1 1 1111111111111111111 | s-a-0 faults in all the nets of the circuit |

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

If we consider that we will take the same circuit here the same circuit we are going to consider and now say what can you do say or some faults or their some stack at faults and stack at faults 0 over there so now what you do so you say randomly you apply this pattern now lets us all of us same time you dream that this is a very good pattern you apply then you can find out that what are the faults that are deducted.

So you can find out that more than one faults will be related by this pattern so 00001 11111 so these pattern let us see what happen so this case you are applying all once.

(Refer Slide Time: 18:10)

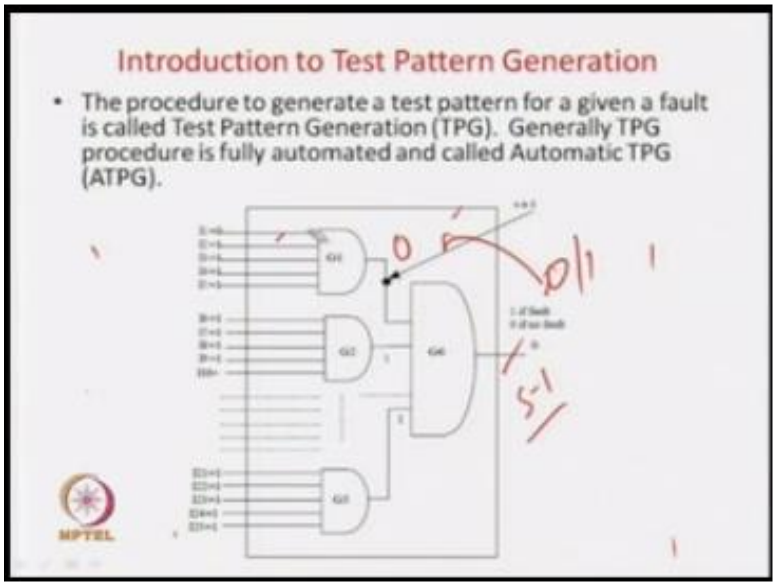


## Introduction to Test Pattern Generation

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

And here you are applying say 1000 01 and pattern apply so which faults are deducted obviously this is 11 so this is 0 over here so this 11 and this is something so okay so obviously a stack at 1 fault is deducted over here yeah so obviously a stack at 1 fault will be deducted by this because all lines are 1 and you apply the pattern 000001 so this is 0 and so with the stack at 1 fault is there similarly the same thing you can also say that it is a also deducting a stack at what you can say is also deducting it as 0 over here so obviously it is also deducting a stack at one fault over here because in this case the answer is 0 but in the stack at 1 fault so the answered will be this 1 so similar you can see that one random pattern is deducting this two fault okay.

(Refer Slide Time: 19:07)



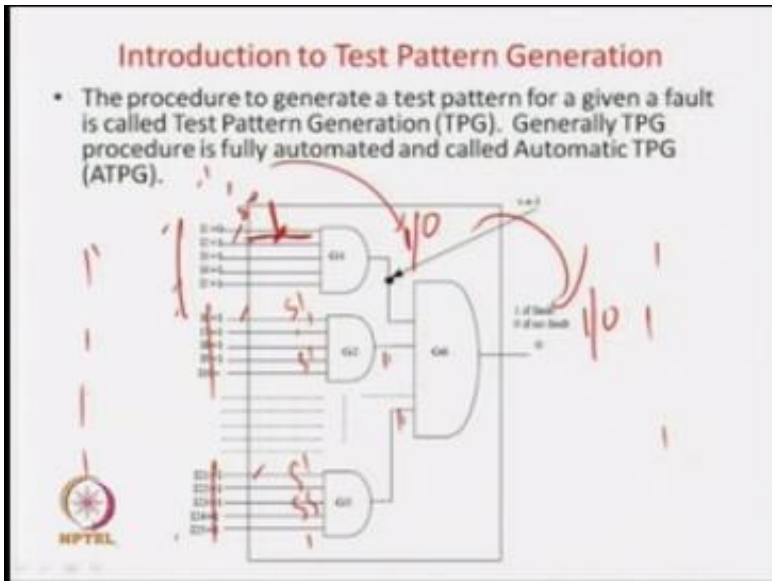
## Introduction to Test Pattern Generation

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

G1
G2
G3
G4
0
0|1
s-1
1 if fault
0 if no fault

So we can also have more example of similar nature so in this case now if you see that I apply this pattern so in this case it is so it is showing 2 faults stack 1 at the net of this one and stack 1 at the net of this okay two faults now so the next random pattern is this 1 all once so all once this one is going to happen so if your applying all once over here all once then obviously stack at 0 fault at any point of the circuit is deducted that is stack 0 of here will be deducted as stack at 0 will  be here deducted a stack at 0 here will be deducted and so forth but if you are going for the now if you take the example let me just elaborate on this part.

(Refer Slide Time: 19:45)



## Introduction to Test Pattern Generation

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

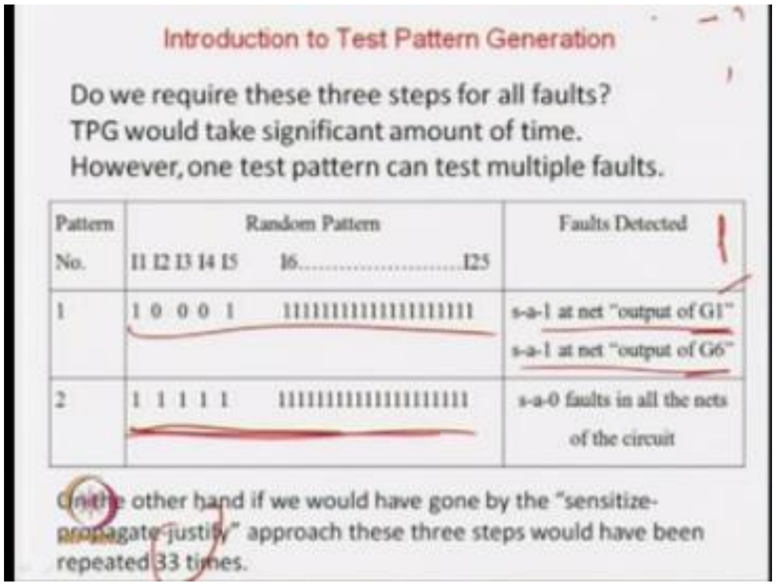
So by fault collapsing and all those things so we have seen that generally with are going to have a stack at 0 at one input if the AND gate and all stack at 1 will there all stack 1 will be there right so in this case also 1 stack 0 will be there for input and stack at 1 at all the input lines will be there all I am getting only one will have a stack at 0 fault by if you can now the Random pattern generate is all once so you can easily find out that if you are applying all one then what can happen is same pattern we deducted a stack 0 fault here it can deduct a stack a o fault here and also it see the stack at fault stack at o fault here.

So one random pattern can deduct stack at 0 faults at all lines in the circuit and if you consider fault collapsing we will not have stack at 0 faults that all the lines of the circuit but we will have one stack faults at each input each AND gate input so if there are 5 AND gates so you will be able to deduced 5 stack at 0 faults by the same pattern so random but if you are using the sensory propagate and justify approach then for then 5 stack at fault as this point say so what we have to do you have to first apply you one over here like for example if you take the control so it is a lot of simple one so if you stack 0 fault over here.

So you have to apply a one over then you have to propagate the value here so if you want to propagate the value here so all other will have to 1 then sorry you have to propagate the here so it will be in case of normal one fault in case 0 and then the affect has to be propagated from here because we are testing the socket 0 fault here so one we have to apply.  This is the normal one fault 0 normal one fault 0 now we have to these the propagation I have justify so these lines has to be one right and then if these has to be all others has to be one all others has to be one and here also everything has to be one if it is also.

So again you have to repeat the same thing for stack at zero fault here and stack at zero fault. There is long procedural.

(Refer Slide Time: 21:37)



## Introduction to Test Pattern Generation

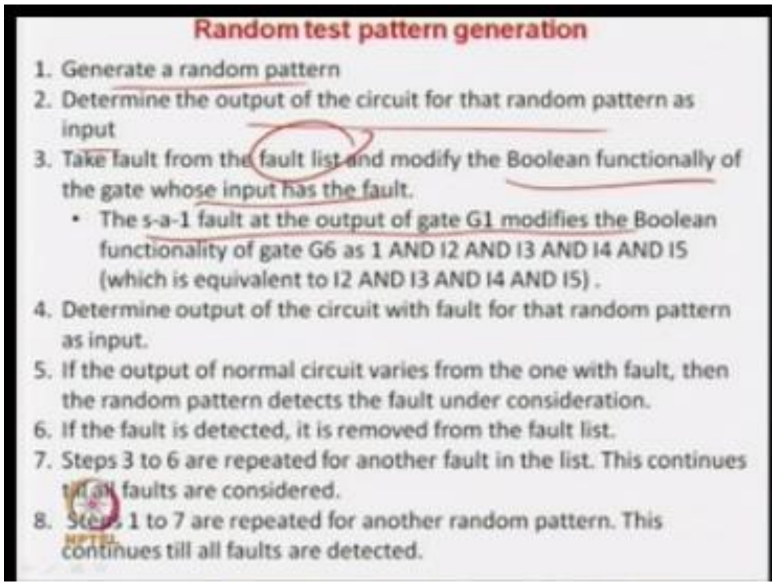
Do we require these three steps for all faults?
TPG would take significant amount of time.
However, one test pattern can test multiple faults.

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

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

Okay so it is long procedure that is problem, but now random pattern generation it is very simple you apply this pattern and you can find out that we can detect much more large number of faults in this case you can see 33 number of patterns has been for has been detected by this one as if know it seems that this is very good idea.

(Refer Slide Time: 21:56)

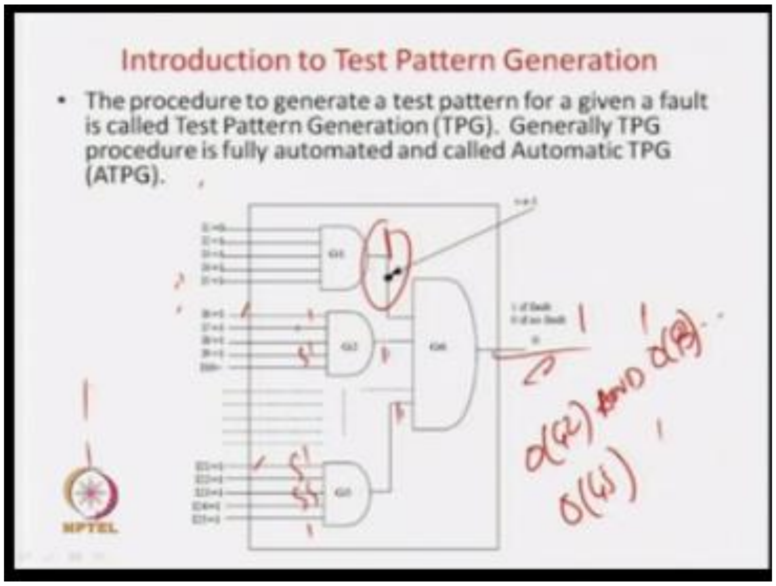


## Random test pattern generation

1. Generate a random pattern
2. Determine the output of the circuit for that random pattern as input
3. Take fault from the fault list and modify the Boolean functionally of the gate whose input has the fault.
   - The s-a-1 fault at the output of gate G1 modifies the Boolean functionality of gate G6 as 1 AND I2 AND I3 AND I4 AND I5 (which is equivalent to I2 AND I3 AND I4 AND I5) .
4. Determine output of the circuit with fault for that random pattern as input.
5. If the output of normal circuit varies from the one with fault, then the random pattern detects the fault under consideration.
6. If the fault is detected, it is removed from the fault list.
7. Steps 3 to 6 are repeated for another fault in the list. This continues till all faults are considered.
8. Steps 1 to 7 are repeated for another random pattern. This continues till all faults are detected.

That you can for a random pattern generation based rather than go for sensory propagate and justify so what is the idea of look at it so what is the basis algorithm you check a random pattern, any random pattern then you determine the output of the circuit in the random pattern in the normal stack at do not consider any fault, but determine the output of the stack at for that random pattern now take one fault from the fault list and modify the circuit okay so for example if the stack at one fault for the output of G1 if you take say for example the same circuit.

(Refer Slide Time: 22:27)



## Introduction to Test Pattern Generation

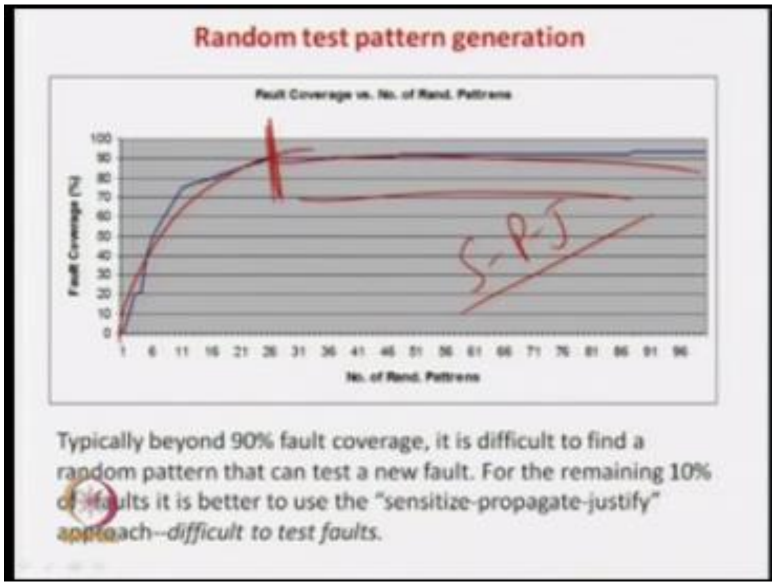
- The procedure to generate a test pattern for a given a fault is called Test Pattern Generation (TPG). Generally TPG procedure is fully automated and called Automatic TPG (ATPG).

So in this case if you are taking stack at one fault here okay so the output of this AND gate functionality will be output G2 and output of G3... output of G5 because the stack at one so this gate input has no effect so the Boolean function will be modified as O of G2 and O of G3 and O of G4 and G5 because this is modified because already stack at 1 so this has no control I will affect on the AND gate. So just modify it so it 1 that is already and of I2, I3, I4 and I5 this is equivalent to this one.

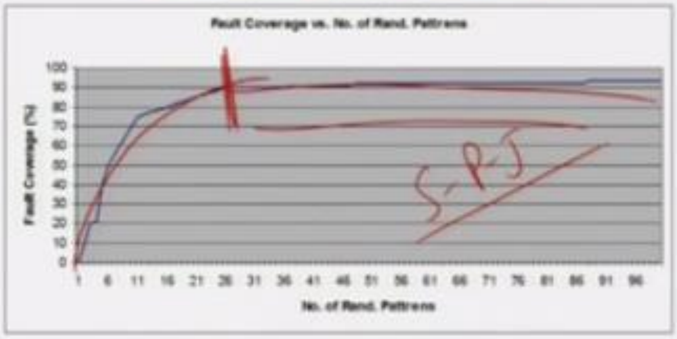
So Boolean functions is modified now you determine the output of the circuit that is chained the circuit with the fault for the random pattern for the same random pattern. If the output of the normal stack at various form the one with the fault then the random relate with circuit and the consideration so what we have done we have take the circuit we applied a random pattern and then you have recoded the output now we do now we change the circuit with the fault now apply the same random pattern then you find that the output various then that pattern random pattern detect the fault.

Now we next what we do we again with delete the fault and say that this is deleted then we take another fault and find out what is case the same random pattern detects another fault or not so let us see that the other fault in this consider till all the faults a listing right how it is if the random pattern goes and you can easily see that random pattern approach is very easy just have to apply the pattern and just you have to output of the circuit it does not require any kind of like sensation and one thing  I should point which will be discussing in details in the further lecture that always for sensory propagate and justify may not be successful that is you may be able to sensory the fault propagate the value with the output but you may not be able to justify the again you have to and find out another way of propagating the follow up may be there are multiple pass to propagate the fault to the output.

(Refer Slide Time: 24:20)



## Random test pattern generation



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

Say for example if there is a AND gate over here and fault can be propagated through here fault can be propagated through here fault so you may try out with this one then what may happen you will find out that you may not able to justify it or may not able to propagate it through this then you have try to this part and you have to try to this part.  In other words there can be requirement of lot of back tracks if you are going with the sensory propagate and justify approach.

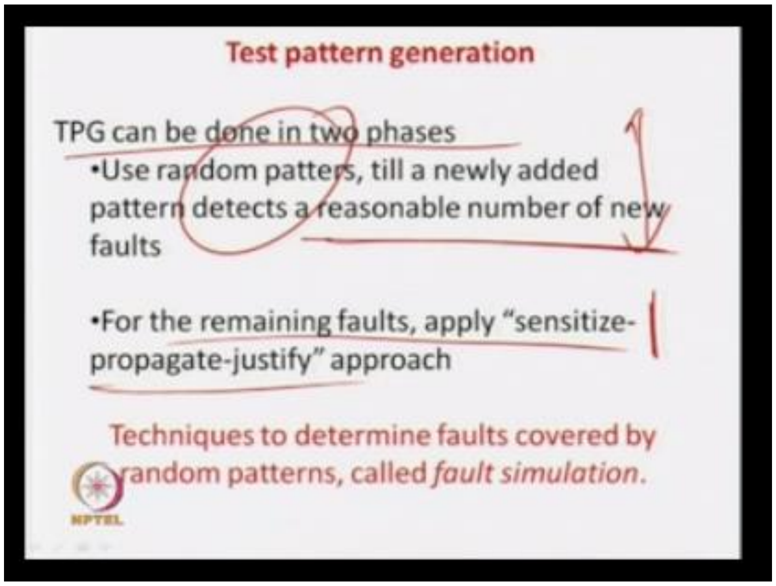
 So in that way the random pattern generation is very much official algorithm in which what case will what we do is that we apply the test pattern find out the output of the circuit and then we modify the circuit with the fault and see if there is any different that is very simple idea. Now question has been do you require sensory propagate and justify approach the answer is yes. So statically this graph has been found so what is there saying like if there are saying like if the circuit has 100.

Then what you do you apply the first random pattern then if detects 20 number of next random pattern you do it says the generators another 20 number of this keep on goes and then it saturates that means what up to around 90% of the faults you can get a very good fault covers part random test pattern that means you apply one pattern it detects 10 faults you apply another random pattern you get another 10 fault then another random pattern you get 8 if you keep on doing it after that this harm faults which are called difficult to test by applying random pattern you may not able to test these fault or in another words you apply 10 random patterns you detect 94.

For the rest 10 faults which we are calling difficult to test part. If you have apply and say more than 30 number of random pattern that is first random pattern you apply after 90 fault is detected. Other random pattern none of 10 faults are detected you keep on doing it say after 31 random pattern say the one of the fault gets detected and keep on doing it so you have to apply great number of random patterns to detect those 10 number of faults.

So whenever you find that the number of random patterns are getting saturation or you saying that it get saturated that is efficiency of random test pattern generation get certificated I mean saturated 90% of the faults for the rest 10% of the faults you find that after applying 10 random patterns one fault is get detected so after that it is actually become more difficult than the sensory propagate and justify approach then we do after you stop random generation approach that is when find the pre concede random pattern or 4 concocted random patterns have not able detect any new fault then what we do we stop that point and from this point we term them difficult to test faults and for them what we do we apply sensory propagate and justify approach of doing it.

(Refer Slide Time: 26:58)



## Test pattern generation

TPG can be done in two phases

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

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

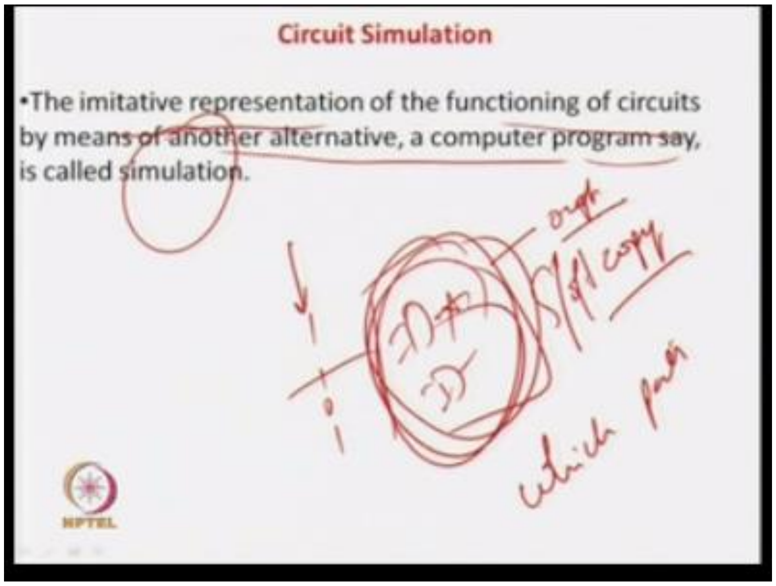
So you cannot say that the propagate and justify approach very difficult approach so you should throw way and we should always go for I mean what you called random pattern generation is not the case for the first few faults for the 90% of the faults random pattern is very good because you can detect last number of faults I mean just applying random pattern and verify which faults are getting detected, but once you get saturation after that the random pattern generation become insufficient because for one pattern you may not be able to detect anything and you have to keep on doing it till you get fault to get the fault.

So fault is later to get senator, propagate and justify so we can see very simple analogy so I think all you might have gone to fairs or may loss where you have balloons where there is a lot of boat where lot of balloons over there and you have to shoot with the gun so what you can do if there is a lot of balloons in the boat that you can random shoot and some of the balloons can basted that is true.  If I do not have very good aim so what I can do is that I can randomly fire the bullet and most of the balloons get blasted now why it happens because there is more number of solutions or more number of points and random firing gets answer.

The same things happens here more of faults are there so you can just apply the random patterns and you can detect the fault but when ten number of balloons remains very less the most of the balloons we have blasted and a few say around 5 or 6 balloons are reaming so at the time what seen happens at the time its I  very difficult then you have to aim and then find out how far is that how your holding your gun and all those precaution you have to take care and then only can blast the balloons so in same analogy here so may lot of faults are there you can randomly do all that.

But whenever the number of faults are coming to the less then what you have to do then you have open a very aimed approach which is called the sensory propagate and justify so the same analogy here so in first we will go so actually the test measure is done in two phases first we go for random paten when ever our a random new pattern detects a reasonable number of new faults when you finds that is not being done then we go for a sensory propagate and justify approach of taking the faults so this actually the second phase so which will be dealing later the in the part of the course but now we will see how we can efficiently do this.

(Refer Slide Time: 29:01)



## Circuit Simulation

- The imitative representation of the functioning of circuits by means of another alternative, a computer program say, is called simulation.

orgt
S/W copy
which part

So for what do you call for we have seen that for circuit the test pattern generation may random pattern so what we have to do, we have to take this circuit and give some input and get the output. So that means what there is the circuit is there, because but you have to know the circuit is not yet fabricated, so we are finding out which patterns have to be applied when the circuit is fabricated.

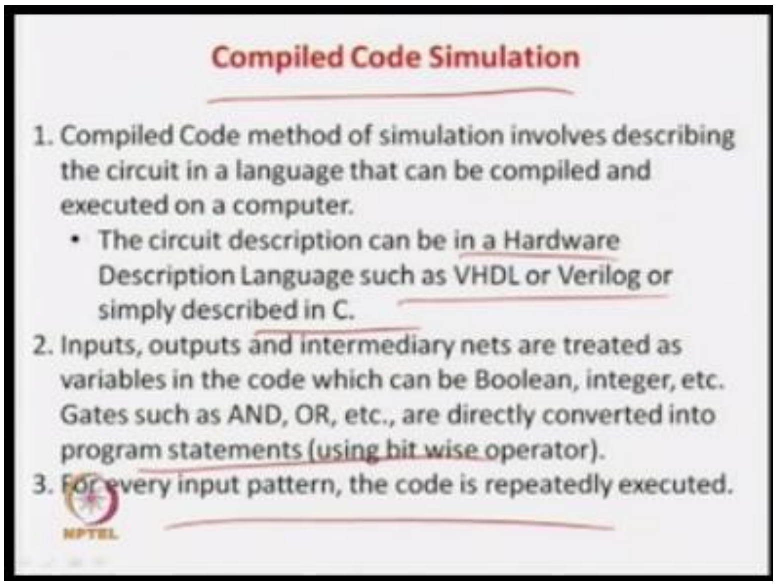
So all these test generation activity is being done when the circuit is being designed, fabrication has not being done. So we do not have the circuit in the hard form that is do not have the fabricated version of the circuit. So we have a say the circuit is soft copy so in a program or some other this is soft copy. So only soft copy is available, and then you have to find out which patterns.

So by random pattern so you are applying some random pattern 1s and 0s, so you have to find out the output. So that means what you have to simulate your circuit, so what do you mean by simulation of the circuit, the imitation the imitative representation of the functioning of the circuit by means of and the alternatives say our computer program is called simulation.

That is you do not have the hard copy of your circuit or you do not have the hard version of your circuit or the fabricated circuit you do not care, we apply some patterns and this circuit is represented in a computer program and you can generate the output. After that you can apply the fault and it will be safety. So in other words you do not have the hard copy, but you represent the circuit in the computer program.

That is very much required for fault test pattern generation by sensitize, propagate and justify approach.

(Refer Slide Time: 30:23)



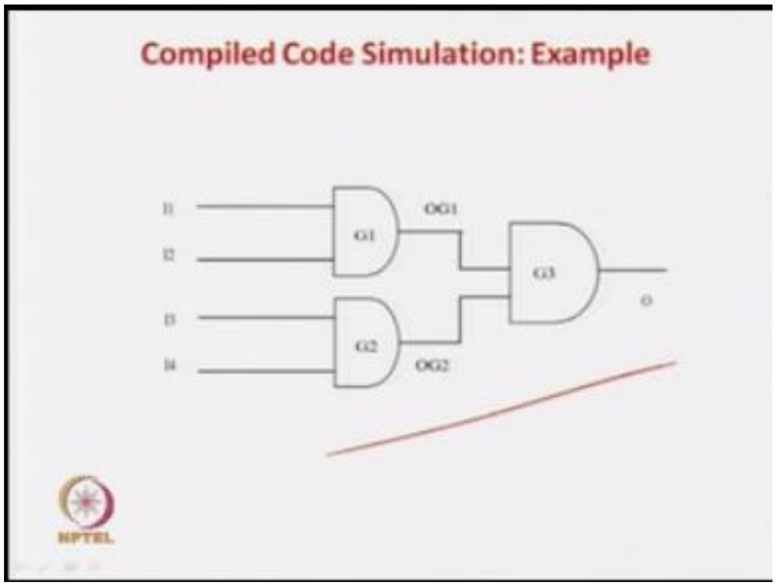
## Compiled Code Simulation

1. Compiled Code method of simulation involves describing the circuit in a language that can be compiled and executed on a computer.
   - The circuit description can be in a Hardware Description Language such as VHDL or Verilog or simply described in C.
2. Inputs, outputs and intermediary nets are treated as variables in the code which can be Boolean, integer, etc. Gates such as AND, OR, etc., are directly converted into program statements (using bit wise operator).
3. For every input pattern, the code is repeatedly executed.

So we will study in detail circuit simulation and also about fault simulation, so what do you mean by fault simulation, in fault simulation the circuit is there at the program and then fault is inserted in this circuit which is represented in the program and you find out what is the output corresponding for random. So first we will see something called a compiled code simulation, so what is the compiled code simulation, the whole circuit is represented as C program or any other program.

And you generate the output for input just on the C program, just like compiled code simulation involves this way the circuit in some language which can be compiled with your computer like it can be hardware language by Verilog, VHDL or it can be simple C. And then you have to give some input and you get the output.

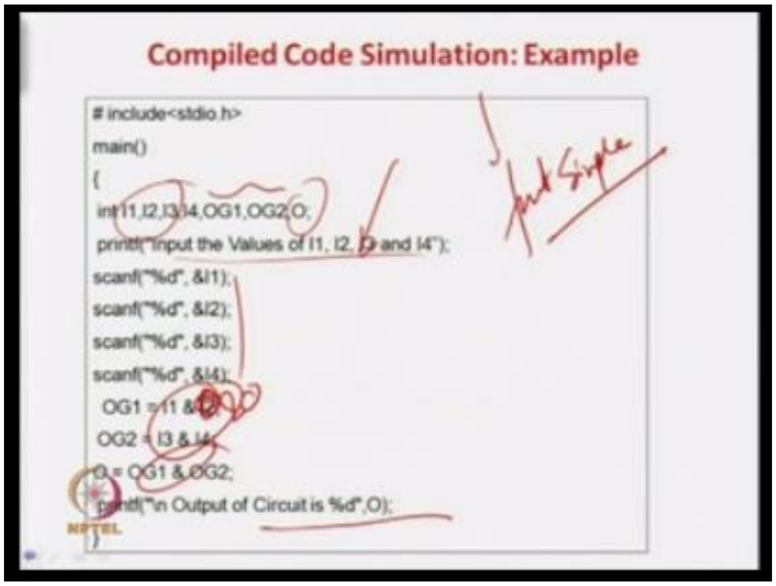
(Refer Slide Time: 31:03)



## Compiled Code Simulation: Example

As you can repeat it as long as you want. Say for example the very simple circuit four inputs and one output.

(Refer Slide Time: 31:08)



## Compiled Code Simulation: Example

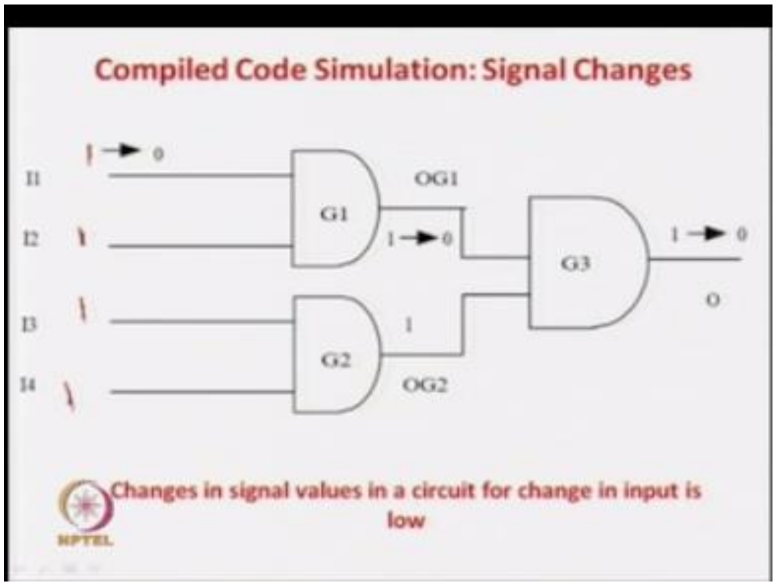
```
# include<stdio.h>
main()
{
 int I1,I2,I3,I4,OG1,OG2,O;
 printf("Input the Values of I1, I2, I3 and I4");
scanf("%d", &I1);
scanf("%d", &I2);
scanf("%d", &I3);
scanf("%d", &I4);
  OG1 = I1 & I2;
 OG2 = I3 & I4;
O = OG1 & OG2;
 printf("\n Output of Circuit is %d",O);
}
```

Now how do you do this, so very easy you can represent it in C, so you have lines like 1, 2, 3, 4 these are the input lines and these are the output lines they are temporary variables. Now you bring test input the values of this one, then you scan if and then you do the ending, and then final ending and the output of the circuit is so.

So different inputs you can give and different outputs you can get from the circuit. So these are very simple approach, now if you want to find out some fault maybe some stack at one fault is and they have some stack at 0 fault is there you can some, stack at fault is there for and to, and this can be ended with 0 and so forth. So just you can modify the circuit for the different kind of a fault.

And then you get the compiled code fault simulation is very simple, just you have to apply a, you know what you call fault, fault bring it.

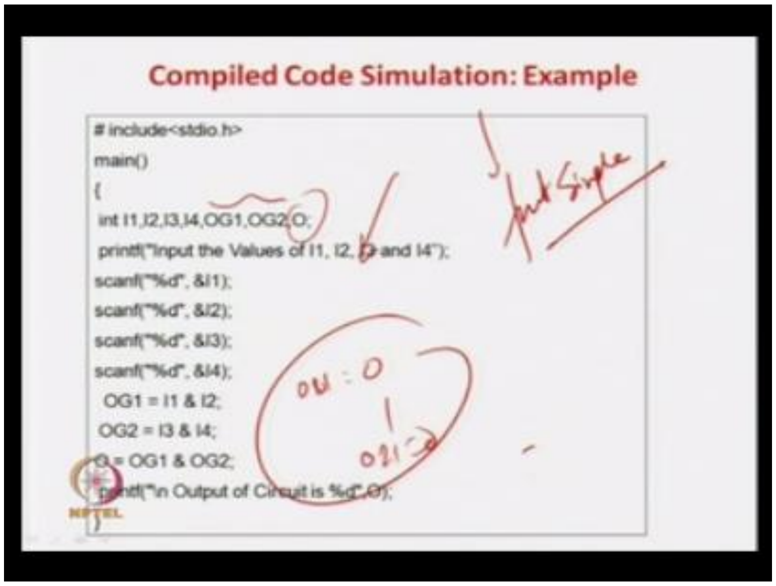
(Refer Slide Time: 31:52)


## Compiled Code Simulation: Signal Changes

Changes in signal values in a circuit for change in input is low

But now you see what happens say for example you first give input as 1111 and then you generate the output, so what will happen.

(Refer Slide Time: 32:01)



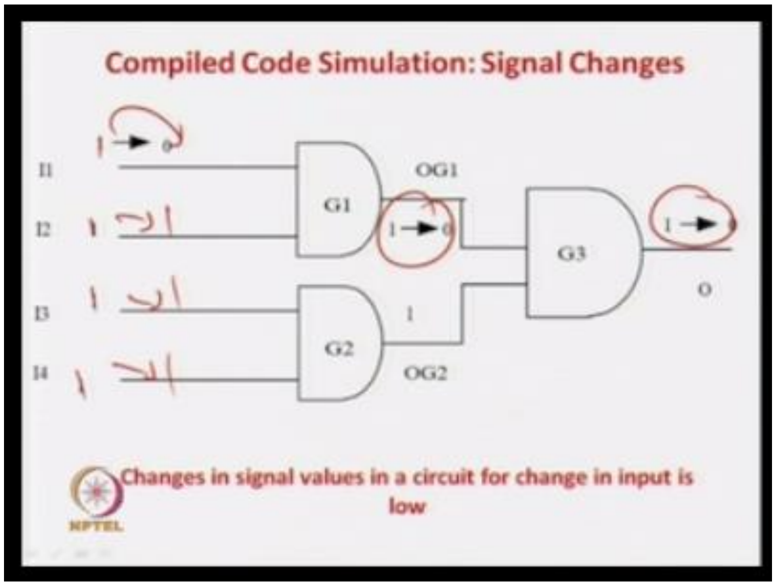
## Compiled Code Simulation: Example

```
# include<stdio.h>
main()
{
 int I1,I2,I3,I4,OG1,OG2,O;
 printf("Input the Values of I1, I2, I3 and I4");
scanf("%d", &I1);
scanf("%d", &I2);
scanf("%d", &I3);
scanf("%d", &I4);
 OG1 = I1 & I2;
 OG2 = I3 & I4;
 O = OG1 & OG2;
 printf("\n Output of Circuit is %d",O);
}
```

So if you give 1111 so first what will happen first this step is executed so it is 1 and 1 so G1=1 now then these things are also 11, then OG2 is also computed to be 1, and then this is done then O is generated to be 1. So it is will treat that computation, so fine this is done. Next what you do is that say next we apply 0111 that is the next random pattern.

So in C language what is going to happen, so again the same board will run and it will take three steps. So it will be 0 and 1 that is equal to 0, this is 1 and this is 0 and 1 is equal to 0. So answer is 0, but it again takes a three steps.

(Refer Slide Time: 32:37)

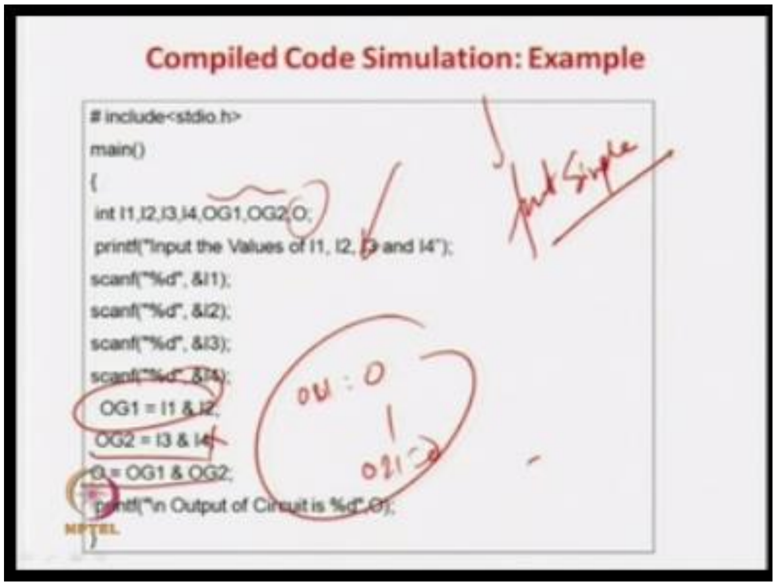


## Compiled Code Simulation: Signal Changes

Changes in signal values in a circuit for change in input is low

What you clearly observe this circuit with not required to do so much activity. But because in C this we are changing from 0 to 1, fine this is no change, this is no change, this is no change. So output here is changing from 1 to 0 and this one is changing to 0.

(Refer Slide Time: 32:53)

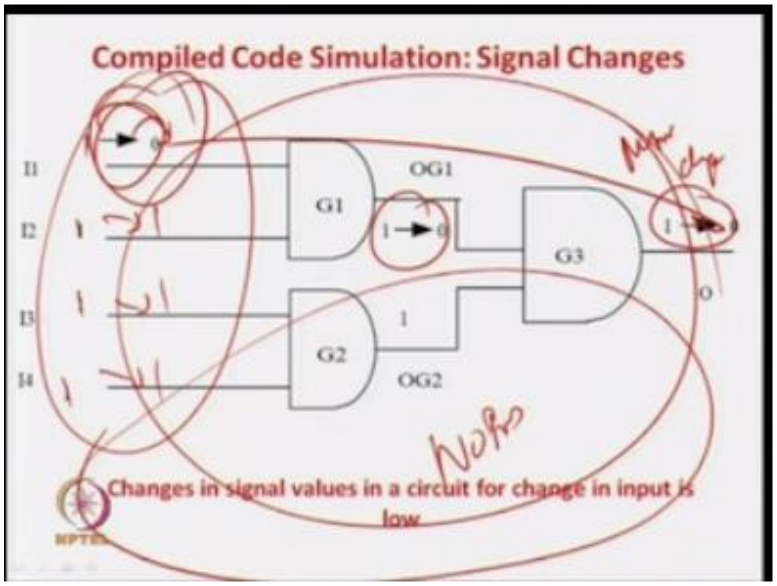


## Compiled Code Simulation: Example

```
# include<stdio.h>
main()
{
 int I1,I2,I3,I4,OG1,OG2,O;
 printf("Input the Values of I1, I2, I3 and I4");
scanf("%d", &I1);
scanf("%d", &I2);
scanf("%d", &I3);
scanf("%d", &I4);
 OG1 = I1 & I2;
 OG2 = I3 & I4;
O = OG1 & OG2;
 printf("\n Output of Circuit is %d",O);
}
```

So only one bit, two bit computation or 2 line computation is fine, that is you need to compute this obviously these you need not compute it was already saved obviously you need to compute this all.

(Refer Slide Time: 33:01)



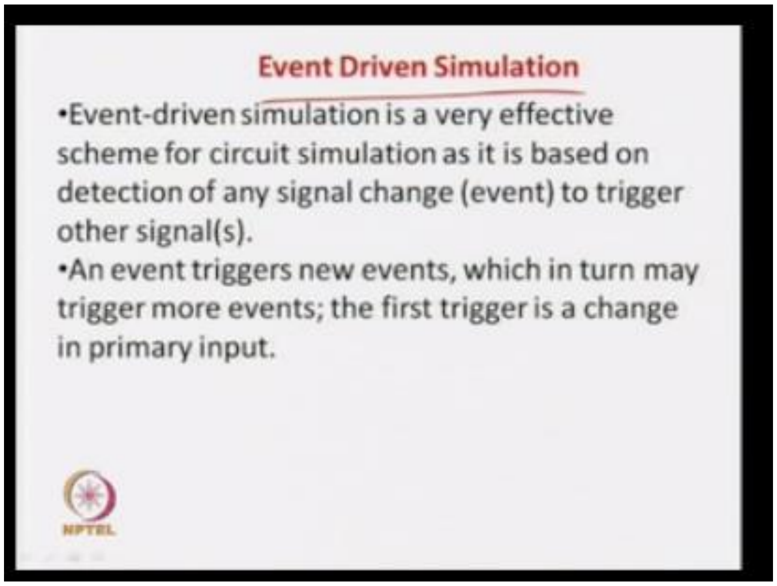
## Compiled Code Simulation: Signal Changes

Changes in signal values in a circuit for change in input is low

So that is what the circuit there are some changes, but if you are going for a compiled code simulation then the problem is that you have to have a simulation for the entire part of the circuit. In other words now as you consider the very big circuit and we say that there is no changes in other parts of the circuit, now only a minor change here which is reflected on the part minor change.

But now if you are going for a compiled code simulation then what will happen the whole circuit will have to be resimulated and which is actually going to give you a big problem that is unnecessarily just for a minor change your whole circuit you are going to go for a compiling, compile will change then you are executing everything is reevaluated. So even if a very small part of the circuit is being changed, but for the compiled code simulation then you are changing or you are recomputing or you are doing a random computation for other parts on the circuit.

(Refer Slide Time: 33:52)

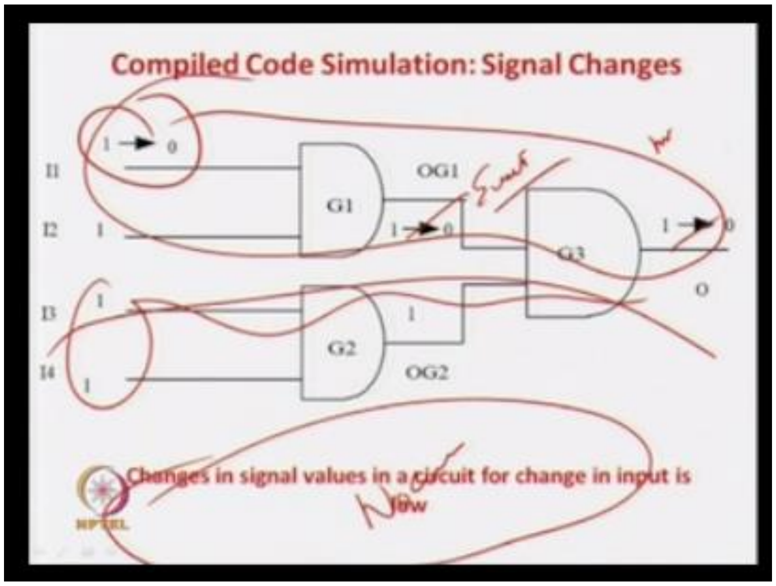


## Event Driven Simulation

- Event-driven simulation is a very effective scheme for circuit simulation as it is based on detection of any signal change (event) to trigger other signal(s).
- An event triggers new events, which in turn may trigger more events; the first trigger is a change in primary input.

So to avoid this one we call event driven simulation. So what is the idea of an event driven simulation.

(Refer Slide Time: 33:58)



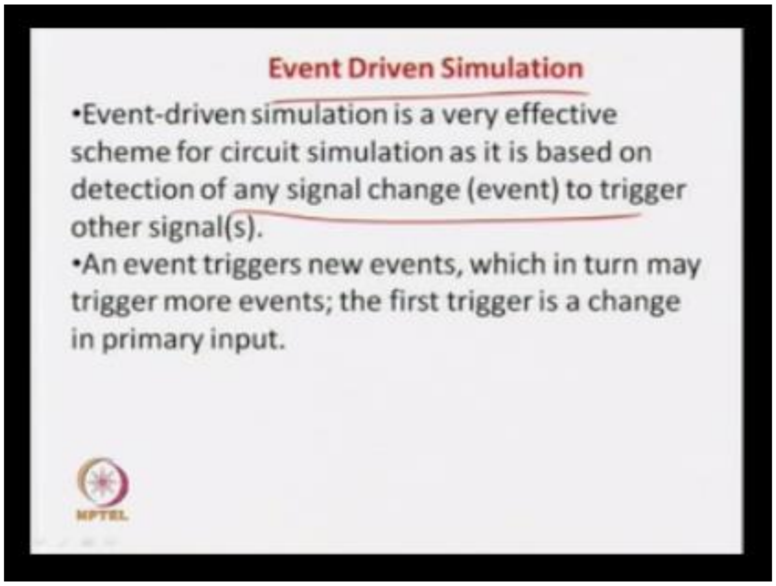
## Compiled Code Simulation: Signal Changes

Changes in signal values in a circuit for change in input is low

So the idea of a event driven simulation is something like this say for the same circuit we take so for the same circuit if you take in case of an event driven simulation what is happening is say this is the event okay. And there is no event in this start of the circuit maybe some other big circuit is there, there is also no event. So this part will not go for any kind of a computation. So what we will do only we will go for computation of this part of the circuit where there are some event change.

So that is actually called event driven these are some the events happening. So this is actually called event driven simulation.

(Refer Slide Time: 34:33)

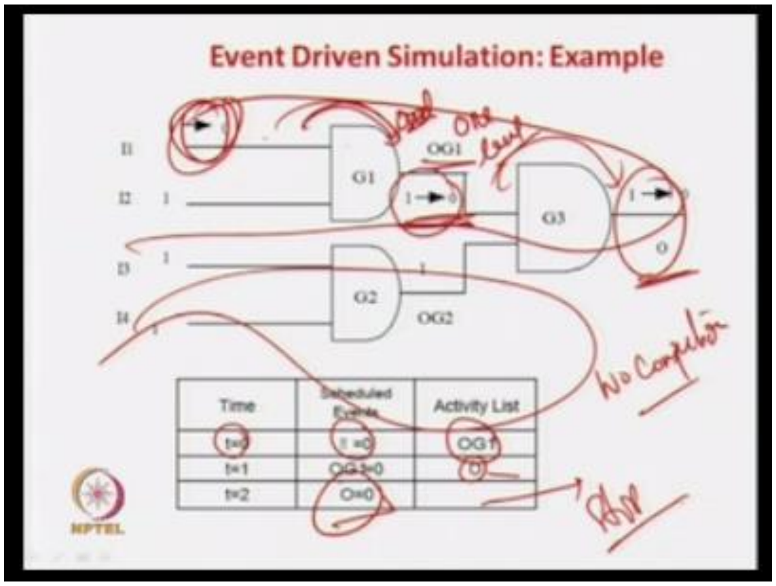


## Event Driven Simulation

- Event-driven simulation is a very effective scheme for circuit simulation as it is based on detection of any signal change (event) to trigger other signal(s).
- An event triggers new events, which in turn may trigger more events; the first trigger is a change in primary input.

So it will save lot of time in your computer execution of the fault test pattern generation is the random pattern. So event driven simulation is very efficient because it detects any signal changes and that triggers of the signals.

(Refer Slide Time: 34:46)



## Event Driven Simulation: Example

| Time | Scheduled Events | Activity List |
|---|---|---|
| t=0 | I1=0 | OG1 |
| t=1 | OG1=0 | O |
| t=2 | O=0 | |

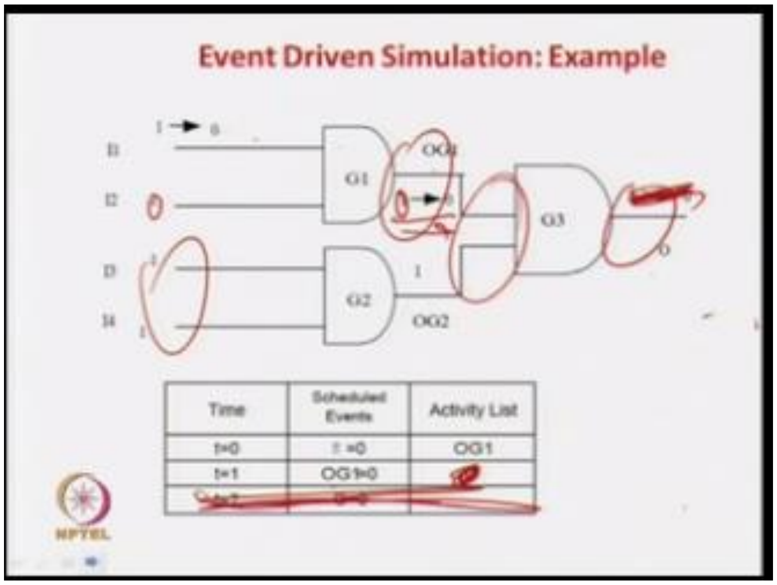
So it is done your trigger signal we will see with an example how it is done. So for example, the same circuit we take so this is a first change for time T=0 so one change is this one I=1 so this changes this. So now only the next level test patterns on next level signals from this gate, because this is reevaluated will come in the activity list.

So only OG1 will be in this activity list, now why is that, so this will be because the output of this gate is only Og1 and only this one level jump is allowed one level jump, only one level is allowed, so and there is a change here so only this can be the activity list. Okay, then fine so you reevaluate this, this is the case, so Og1 changes from 0 to 1, so OG1 changes from this one, so only the activity list will be output of this one.

Because only one level is allowed, so this changes there so this will come in the activity list this one is the case and then this available leads to be O=0 and nothing in the activity list and you stop. So you just require a very small number of computations to do it and you did not do any kind of a note computation required, no computation is required for this only we require this.

So what is the basic idea here, you have a circuit and you have some gates, so basic philosophy of this one let me discuss.

(Refer Slide Time: 36:02)



## Event Driven Simulation: Example

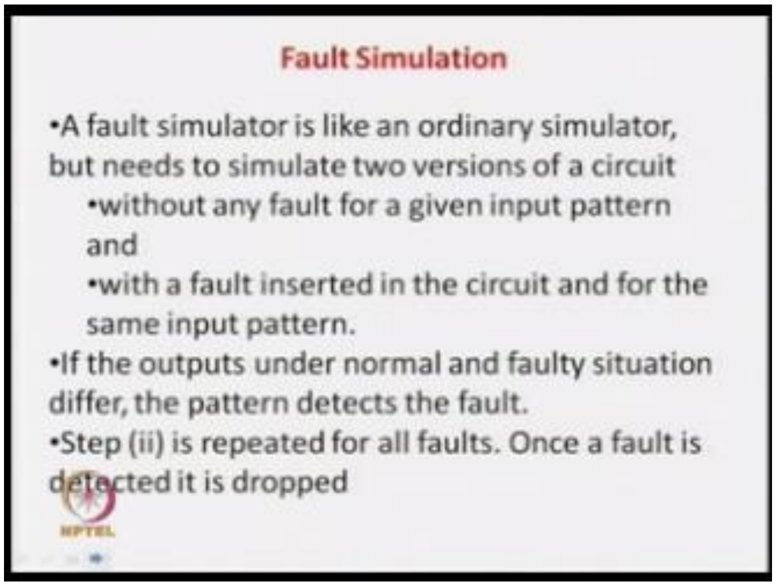
| Time | Scheduled Events | Activity List |
|---|---|---|
| t=0 | I1=0 | OG1 |
| t=1 | OG1=0 | [illegible] |
| ~~t=2~~ | ~~O=0~~ | |

So what we do is we have something like we have a gate like this some signal change is there, then output of this one will be in the activity list that is it. And again any change here will be again corresponding to this activity change. For example, if you have something like this say, say this had been the case say for example. Now in this case what would happen say this our initial pattern was 1011.

Now you change this comes and this one, now what will happen I=1 to 0 activity list is O=1. Now activity is 01 and a 0 and a 0 the activity list is 0. So initially it was also 0 now it is also 0, so the activity list will be empty in this case. Because it is 0 to 0, so no change is there everything is stopped over here, and this change is also not there. So in level two the propagation will stop.

But if you are taking a compiled code simulation then again you have to evaluate this, evaluate this, evaluate this and so forth and it is the wastage of time in computation so event driven simulation is helping a lot.

(Refer Slide Time: 37:09)



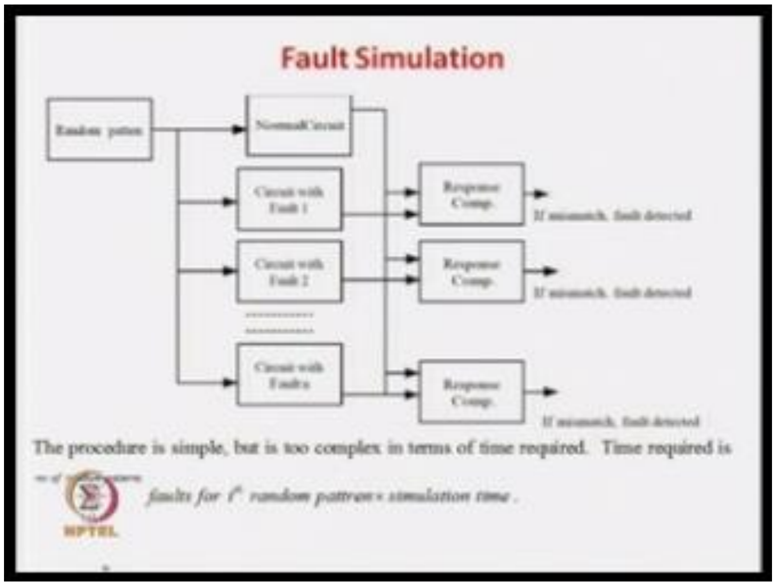
### Fault Simulation

- A fault simulator is like an ordinary simulator, but needs to simulate two versions of a circuit
  - without any fault for a given input pattern and
  - with a fault inserted in the circuit and for the same input pattern.
- If the outputs under normal and faulty situation differ, the pattern detects the fault.
- Step (ii) is repeated for all faults. Once a fault is detected it is dropped

Now we have to go for circuit simulation so do you mean by a circuit simulation in case of a fault, sorry we have seen about circuit simulation circuit simulation means what a circuit is there and then you are applying some inputs and you are finding out the output bit compile code simulation or event driven simulation. But for a fault simulation  but we know that for random it is pattern generation what you have to do  you have to put the fault in the  circuit and then you have to go for fault simulation, circuit simulation.

So fault simulation is nothing but if you have a circuit with the fault and then you have to simulate the circuit for the output for some given input you just find out the output is called  fault simulation you can also call fault circuit simulation also you can call, you can also call it fault circuit  simulation so fault simulation is like ordinary simulation but it has two versions one is without fault and one is with fault that is very simple idea so if the output of the faulty circuit is not matching with the output of the normal circuit so that is what is did I act then you know that the random pattern or the pattern is detecting the fault that is what is the idea of fault simulation.

(Refer Slide Time: 38:11)



The procedure is simple, but is too complex in terms of time required. Time required is [illegible] *faults for $i^{th}$ random pattern× simulation time* .

Let us see this is a block diagram which shows this so there is some random pattern now what happens so you are applying what do you call this circuit with fault 1, fault 2, fault n all the faults are there and this is a normal circuit output you are comparing this one and then if there is a mismatch this fault is detected, this fault is detected and you remove the and you will take the random pattern number 2 this how it is done. So if you are so how you are improving in it you are improving in using what you call the event driven simulation so if you are using event driven simulation so you are simulating circuit 1, circuit 2, circuit 3 which faults and this your also with fault 1, fault 2, fault 3 so even if we are doing with fault so there can be little amount of changes in the whole circuit.

So if you doing what do you call event driven simulation so you can save lot on the computations, so that is one way you are saving but only one another factor you are saving we are not doing another thing is that in one random pattern we are trying to see if one fault id detected, so there also we  get parallelized, so two way we are making in an efficient one is event driven which is already seen another is parallel that is what you are doing we are taking a circuit applying one test pattern random pattern and see if the fault is detected then with fault 2 and fault 3 and so forth.

So we need to also think if we can find out that given a random pattern how many faults it can detect in one goal that is one good way of doing it another is event driven because if you are going for compile code simulation then for even for a small number of fault I mean fault change or the input pattern change we have to go for a full code simulation, so that is one thing we have they are two factors you have to improve so one improvement is this event driven and that is parallelism so both of them we are going see in details.

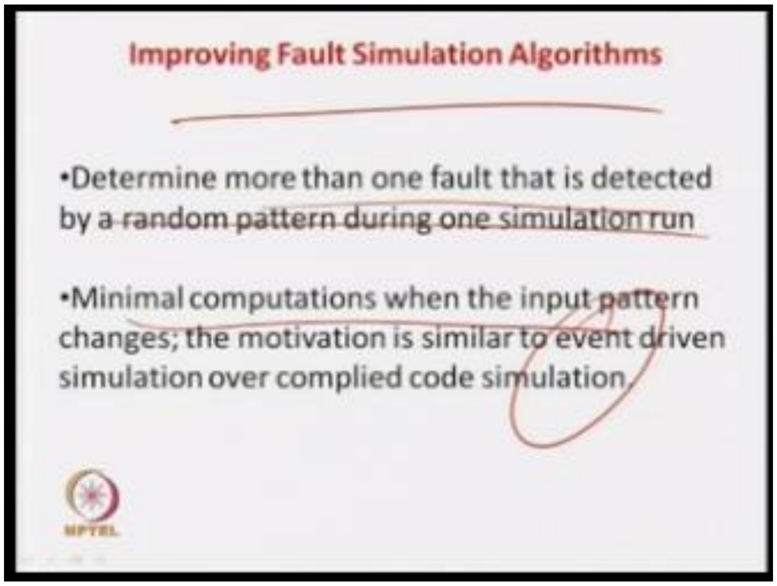
So but if you are not doing anything for this for even random pattern generation random pattern generation is also not very simple because the time is how much number of faults for the highest random pattern say for the $i^{th}$ random pattern till say for example we start with 100 faults, first pattern for the first pattern and how many faults we require to check for all 100s we require to check in pattern number 1.

Because 100 faults are there you have to check for 100 say for 10 patterns 10 faults get detected by the pattern very good, now pattern number 2 comes then again 90 remaining faults has to be tested further, say 20 faults get detected some by some digit then for pattern number 3 you have to go for 70, so similarly you go and say for the $10^{th}$ number for there is only 10 patterns remaining and say some on the one fault get detected then for the 11 pattern 9 faults has to be checked, 12 9 faults because this saturates say about to 100 patterns 9 faults has to be tested generally we stop at this point new things have saturated.

But you see for the first pattern 100 faults has to be checked for fault pattern to 90 faults ahs to be checked for and so on, so I mean if you are do not have a parallel faults simulator that is I mean if you can sum up paralyze the algorithm that is for a single pattern you can check whether among this 90 faults say 100 faults are there can you check parallel that where the all the faults are detectable by this random pattern would be a very good idea and also and among from that also you should be able to simulate your circuit for minimum requirement only those things are changed that only you should be taking into picture and not more.

So we say that the number of that is event driven simulation so whatever minimal changes  is there only that part has to be re-simulated  and other things are to be retained, okay and you should not work random I mean what do you call you should avoid redundant equation. So we say that the time for fault simulation is faults for the highest random test pattern into simulation time that is fault simulation normal value is always there and number of random patterns you are going for that, that is not a very small number.

(Refer Slide Time: 41:42)



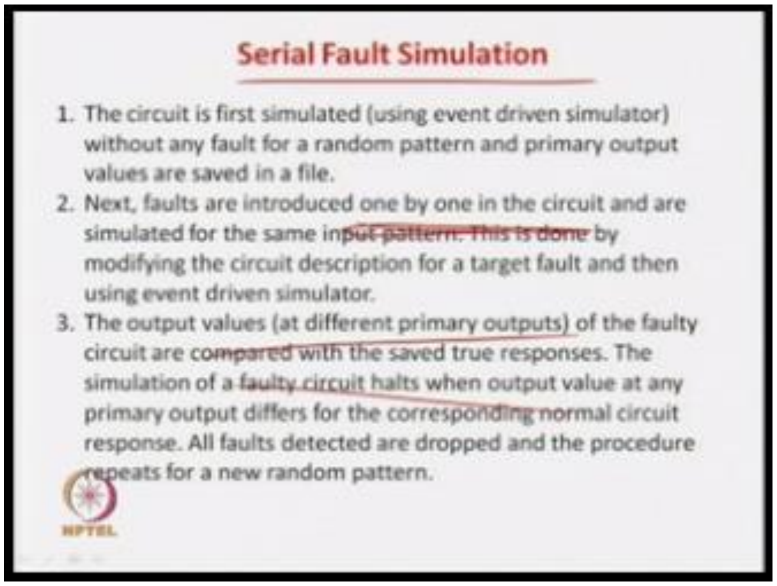
## Improving Fault Simulation Algorithms

- Determine more than one fault that is detected by a random pattern during one simulation run
- Minimal computations when the input pattern changes; the motivation is similar to event driven simulation over complied code simulation

Okay, so we have see so that is what is improving fault simulation algorithm that is minimum computation change that is event driven simulation so whenever there is a pattern change or wherever there is a fault change minimum changes are there then you have to go for only incremental simulation that is event driven simulation you should not simulate the whole circuit and determine more than one fault can be detected there random pattern, that is check is parallelized.

So if this two algorithm these two features we can build in then our things will be improve, so that is what we are going to do.

(Refer Slide Time: 42:09)



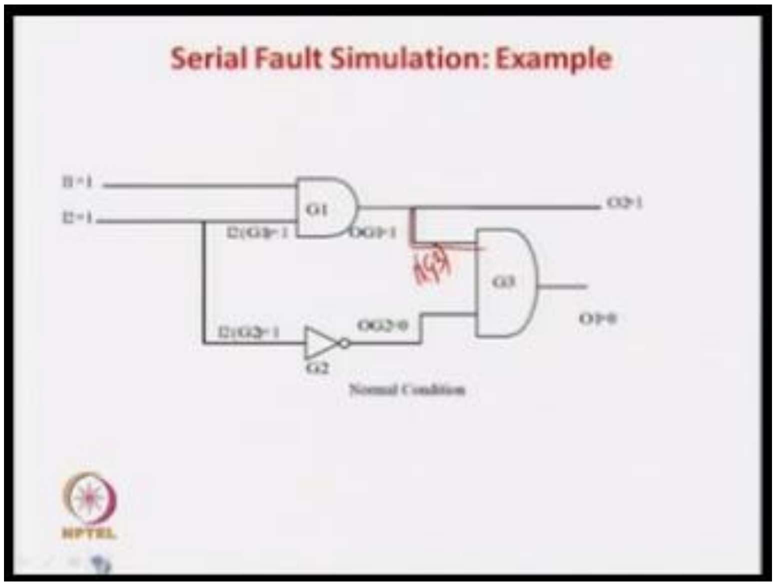
## Serial Fault Simulation

1. The circuit is first simulated (using event driven simulator) without any fault for a random pattern and primary output values are saved in a file.
2. Next, faults are introduced one by one in the circuit and are simulated for the same input pattern. This is done by modifying the circuit description for a target fault and then using event driven simulator.
3. The output values (at different primary outputs) of the faulty circuit are compared with the saved true responses. The simulation of a faulty circuit halts when output value at any primary output differs for the corresponding normal circuit response. All faults detected are dropped and the procedure repeats for a new random pattern.

So first we are going to see a very simple what fault simulation algorithm which is call the serial fault simulation then at  the very simplest one so in this case what did is done so it is nothing but you apply a pattern then you apply one fault the serial the serial you are going to check then you see whether you apply a pattern and with so apply a pattern then take one fault and see if the fault is detected by the pattern if so the fault is drop and the default is consider and these all been tested.

So then you take another fault see whether it is detected by the same random pattern and so on one by one you test for all the fault which is remaining and the when the faults have been checked for say you get that n number of faults has been detected so that you remove from the circuit that out of 100 sometimes faults have be removed so now for the 90 faults you repeat the same procedure for let us pattern.  So that is the  next fault introduce one by one that is a  very important thing it is one by one you are introducing the faults and doing the fault simulation. Where the output is different faults are detected and so forth,

(Refer Slide Time: 43:19)

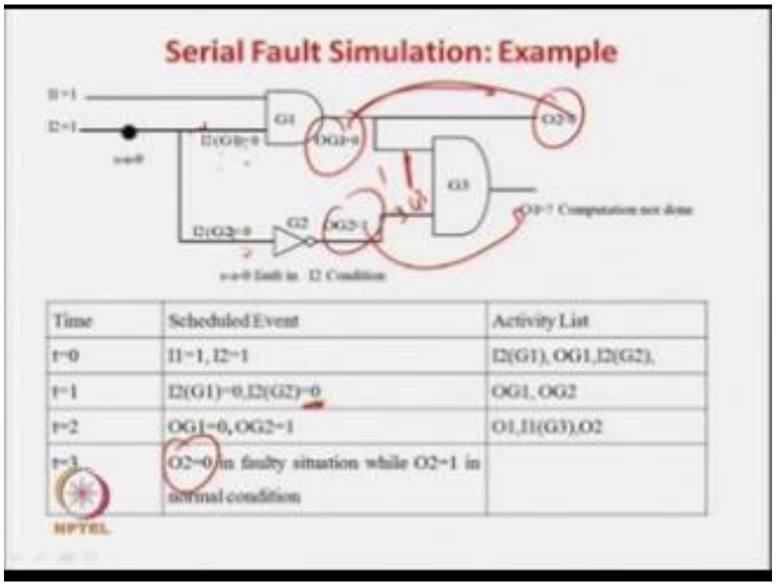


## Serial Fault Simulation: Example

Now for this simple serial pattern so that is the very important I mean that is the main idea is that one by one you are taking, but again what we have to remember that always we are going to use event driven simulation and even then we are going to improve on serial parallel detective and all those things slowly we will come into this. So let us study serial fault simulation with example, so let this be the circuit, this circuit you can see so one thing I should say here that this is input i1g all fan outs are different if you remember so this is i2g1 this is i2g2 this is input to of g2 some names we have given so this is output is o2 this is o1.

So in testing as we know all fan outs are different so I am putting a different name to this one also it is i1g3, so input 1 this is output g2 no problem so this is the 1.

(Refer Slide Time:  43:58)



## Serial Fault Simulation: Example

| Time | Scheduled Event | Activity List |
|---|---|---|
| t=0 | I1=1, I2=1 | I2(G1), OG1,I2(G2), |
| t=1 | I2(G1)=0,I2(G2)=0 | OG1, OG2 |
| t=2 | OG1=0, OG2=1 | O1,I1(G3),O2 |
| t=3 | O2=0 in faulty situation while O2=1 in normal condition | |

I1g3 this net okay, so now so they just stuck at 0 faults over here so what you have to do say for example, we do a random pattern is say 11, so what is the output of this circuit in normal case o2 equal to 1 o1 equal to 0 that is very simple because  11  so the answer is 1 so this is  1  and in this case it is 1 this is 0 so the answer is 0, so 10 is the result for the normal serial and this is the random pattern, right  so now  we are going for this one.

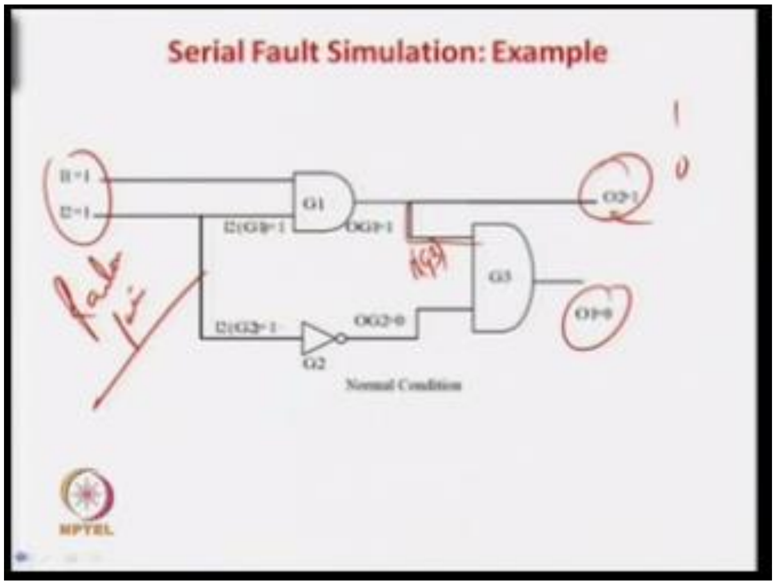
So the same random pattern when we are going to see whether this stuck at for the detectable or not, so this serial so we will take fault at a time. So this is one and one event driven simulation so as I told you, already told you that only one level is allowed so this is in the activity list and this is in the activity list, so only two things will be in the activity list. So what do you have here is i1=1 and i2=1 sorry, and this one will also be there because this is also in the list.

So activities I2G1, I2G1 is the single level OG1, OG1 same level and I2G2 is I2G2 because this is the signals we have and this is the one level change from the signals so one level change from this things so you are going to have these things are the activities because you are saying that this signal changes and the input is equal to one and input equal to one this to equal to 11 are going to have directly impact on these three points only this is one so these are in the activates now if you have this 1and 1 over here.

So you can compute this to be 0 because the stuck at fault this is 0 so I1 G1 is 0 and then this is also I2 G2 is also 0 this you can compute but this you cannot compute because in the second round this values dot so okay so what is the new activities this one in the new activities because of this and this was the remains in the activities thee could not be computed because in the first value this one was not known to us correct so now in the next activities though OG1 and OG2 now what happens now what we do is that so now we have the values over here so we have OG1 = 0.

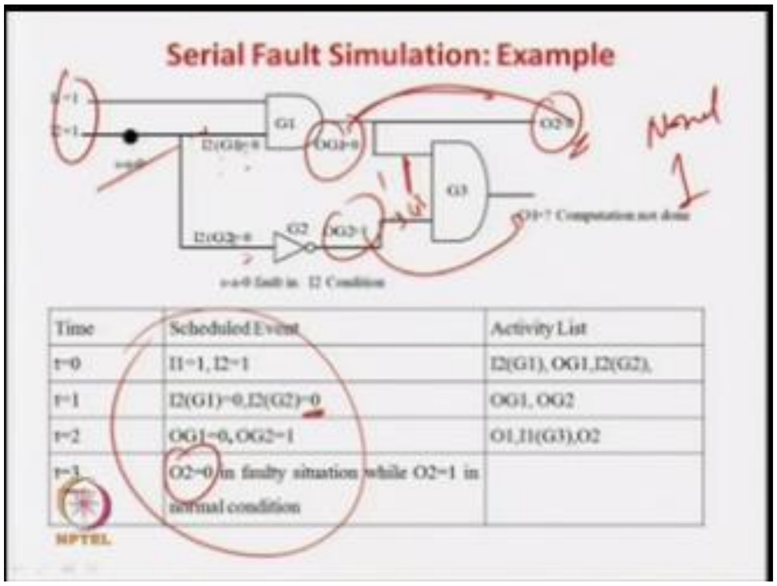
This one we know OG1 = 0 and O2G1 = 1 this things we know now what is in the activity list the activity list will be now 1 because of this one we have O1 because this value is known so this point is also known which I called I1G3 this is I1 G3 okay and O2 obviously this one will be there because this is there so on value jump will be there because one level jump is allow so this is you thing now you can easily see that if this one is a 0 so next jump O2 = 0 and if you look at the normal circuit.

(Refer Slide Time: 46:44)



O1 was a 1.

(Refer Slide Time: 46:47)

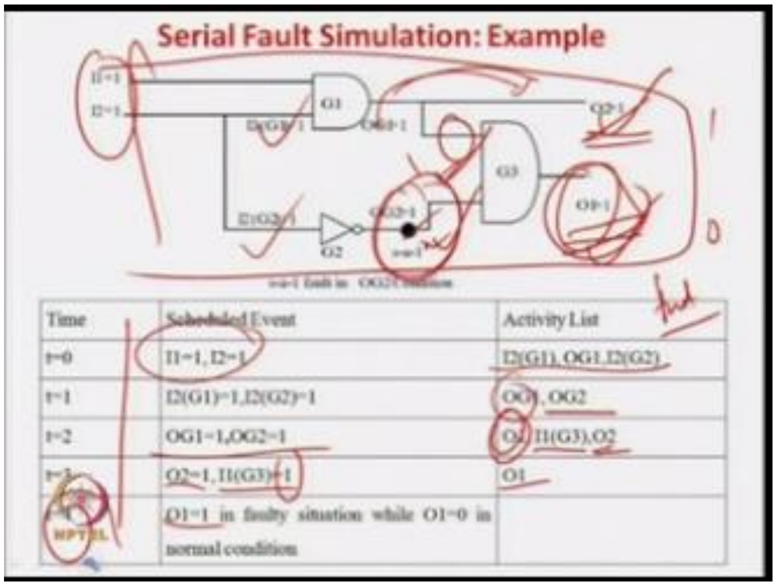


## Serial Fault Simulation: Example

| Time | Scheduled Event | Activity List |
|---|---|---|
| t=0 | I1=1, I2=1 | I2(G1), OG1,I2(G2), |
| t=1 | I2(G1)=0,I2(G2)=0 | OG1, OG2 |
| t=2 | OG1=0, OG2=1 | O1,I1(G3),O2 |
| t=3 | O2=0 in faulty situation while O2=1 in normal condition | |

So you can stop at this point say that 11 detects the this stuck at 0 fault because in stuck at 0 for this is 0 and normal case it is a 1 so it detects the 1 and easily is that so this may random patterns in machine so this we are doing event wise because we are doing only for those parts then were the signal chain now so one fault has been detected.

(Refer Slide Time: 47:10)


I1=1
I2=1
G1
I2(G1)=1
OG1=1
O2=1
G3
I2(G2)=1
G2
OG2=1
s-a-1
O1=1
1
0

## Serial Fault Simulation: Example

s-a-1 fault in OG2 location

| Time | Scheduled Event | Activity List |
|---|---|---|
| t=0 | I1=1, I2=1 | I2(G1), OG1, I2(G2) |
| t=1 | I2(G1)=1, I2(G2)=1 | OG1, OG2 |
| t=2 | OG1=1, OG2=1 | O2, I1(G3), O2 |
| t=3 | O2=1, I1(G3)=1 | O1 |
| | O1=1 in faulty situation while O1=0 in normal condition | |

Now let us see what next same pattern is there we are going for severe fault simulation so we are going for this fault okay another fault we have taken these are stuck at 1 fault over here so this one is already take I133 this is there for you okay so now again we are going to look at for all those things so now what happen so these one was say random pattern so now in this case is 11 so activity least of these three this point and this point this is the value so now you are going to be 11 over here.

You know this fault is here not here this time so you are going to get I1 and this one so these values are there see this values are here so we are having OG1 has this activity value or already it was there it will become to the new added this OG2 now you see so in this one and in normal case the outputs so busy but stuck at one OG2 will be one it is not 0 because the stuck at fault and OG3 into C I1 sorry I1 E3 is 1 because this values propagated so this values you get okay and sorry this is the pattern which was saying.

So OG1 and OG2 you get so OG! Is 1 and OG2 is also 1 because the stuck at fault over here the activity least in this case is output 1 you know that we have and then I1G3 you know these values this is also in the activity list and obviously O2 will be in the activity list now what you do now new values which you are obtaining now here is O2 so this O2 = 1 because the initially we having now we are having a value of one over here so O2 is = 1 and as well as you are also having this values from here to here it is 1.

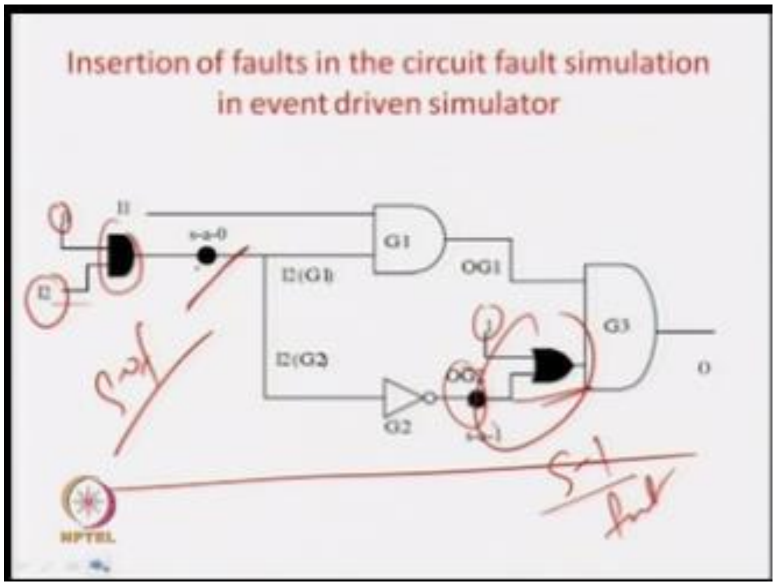
As well as from here to here it is 1 so I13 I1G3 1 that is this fan out is also in the value of 1 and in the activity least is now 1 because this signal is red okay so it was also there but initially we could not compute the value because this values was not ready value of this fan out was not ready so in the but you can call in the second step the value was not ready so you could not compute it but now in the third step this values of 1 is ready so in the value of sorry step number three these values is also ready this values so we cam compute O1 so now is in the activity list and the fourth stage.

O1 = 1 so if the fault is captured because in the normal case the answer was one and this answer was 0 so 40 is detected but there is a one difference we have to observe their care carefully here that here we require four steps to do and here we require only three steps so that is the beauty of compile simulation so we are going event wise whenever we detect that they is our difference within normal so let it we immediately stop so for this for we could not need not go for competition of this.

One which require force so we could have even you stop at the third stage and if I note that I am suggest the fault simulated some fault has been detect and we had that but for this one this one we have to go for stage number four why we are to go for stage number four because the fault is detected by this gate and not this output so where to go for stage number four but had you mean a combined got simulation then we are not going in the stepwise level we are get simulating the circuit totally and then you are comparing this one with this one.

So unnecessarily many times will be going for here also we will going for default for all circuits we have to go to the exhaustive level so here also we would have gone for unnecessarily T4 which is not require so for these circuit examples we can verify that always we will be at lot of gains.

(Refer Slide Time: 50:27)



Insertion of faults in the circuit fault simulation in event driven simulator

If you are going for a compile got simulation sorry you are going for a event driven simulation so this was one example in case of serial fault simulation so now one is small thing I would like to add before the it was two lectures that is we are always saying that these level is stuck at 0 this level is stuck at 1 then but algorithm level in a circuit measure how to insert a fault this is very simple so if you want to add a stuck at 0 fault of this net you put AND Gate with this is an this input was normally I2  you put AND Gate with one bit series so or else it is so this minutes are stuck at 0.

If you want to stuck at 1 fault over here then you this is your OG2 and you put a OR Gate with one input fixed at one so this will minute stuck at 1 fault and this is going to many cure stuck at 0 so these are you get mini okay so this was way simple but why do we require this because when you are using a fault simulator then you cannot say this is stuck at 0 this is stuck at 1 because writing it has stuck 1 will not be handle by a circuit simulator so what it, it will do you have repeat the stuck at 0 and stuck at 1by means of some gates and they will be.

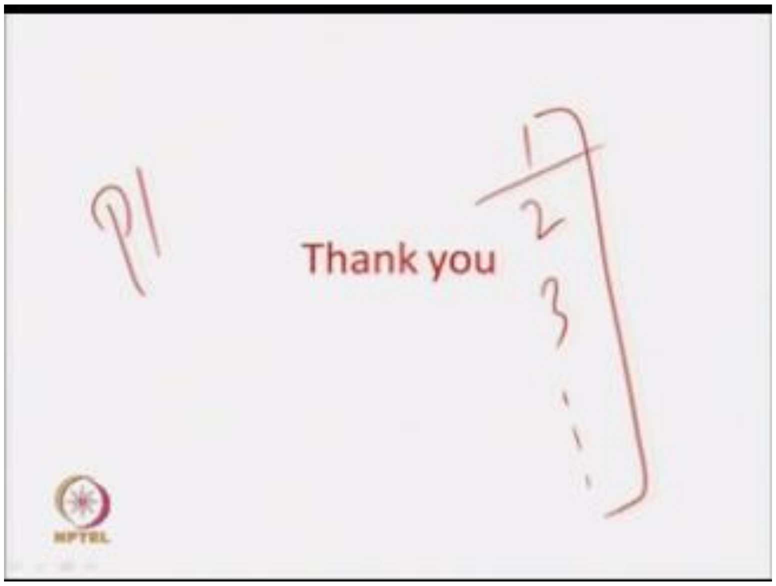
Mimicre is require and this mimicre will let use fault simulation for this part of circuit so with this we stop today and.

(Refer Slide Time: 51:37)

Thank you

So what we have it is remaining for the next two series of lectures in this fault simulators so here we have improved on fault simulation by we are not using compiled go so you can in the better technical call event driven simulation but still our faults with serial so we do fault1 fault 2 fault three and so on.

(Refer Slide Time: 51:51)



For a given random that is not got then tomorrow are in the next lecture what you are going to see is that how can you apt your pattern P1 and C in parallel can you check whether fault1 fault 2 fault 3 and fault 4 are detectably in one go by this test pattern so that we are going to see in the next class and say were another similar algorithm switch can enhance the performance of your fault switch in algorithms thank you.

**Kallal Barua**

**Kaushik Kr. Sarma**

**Queen Barman**

**Rekha Hazarika**

**CET Administrative Team**

**Susanta Sarma**

**Swapan Debnath**