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 - 2
Fault Simulation-2

Welcome to the 2nd lecture on Fault Simulation, as discuss in the last day that, our topic on fault simulation will be 3 hour lecture. So, in the last discussion, we have seen that, to generate test patterns for a circuit for a given fault models like stuck-at fault model what we have to do. There are two approaches, one approach was based on, what you call that a sensitized propagate and justify approach. So, in that case, given a fault we first have to sensitized the fault that is, if it is stuck-at 1 fault, we have to apply a 1 sorry we have to apply 0 and if it is the stuck-at 0 fault then, we have to apply 1.

Then, the value has to be propagated to some primary output and then, you have to justify the signals so that, the values can be propagated and sensitized. So then, we have seen that, this algorithm must be complex and if you want to apply to all the fault present, it will take a long time. Then, we have try to seen for an alternative approach, in which case we see that, whether what we do, we take a random pattern, we take an 1 test pattern and then, we apply in the circuit and then, try to verify what are the number of faults that is detectable by this one.

Then, if there is say for a given random pattern, we can find out the that, around 70 to 60 or 20 faults have been detected then, we drop them and then, we take another random pattern and try to do this procedure. So, what is the advantage of this procedure over the sensitized propagate justify approach. So, see for example, the primary importance of this one was that, in test pattern generation on any test pattern, there is a single pattern for multiple faults. In other words, single test pattern can be applicable to test more than one kind of a fault.

So, in this case what is happened, so if you are using sensitized propagated justify approach, so one fault can be tested for a pattern that is, one pattern will be generated for the fault. But, if you are using random patterns then, what happens, multiple faults can be detected by a pattern that you can analyze and find out. That is, one random pattern can given 3 or 4 or 10 or 15 faults it determine, can be brought down, can be detected and can be found out that, these are the patterns, this pattern can detect this, this, this and this faults.

So, the problem of multiple fault detection is also taking into picture, but say for example, like say one pattern, say pattern I detects 10 faults, fault 1, fault 2, fault 3. So, in sensitized propagate and justify approach, the same pattern may be generated when you are trying to do this for all the faults f 1, f 2, f 3 and f n, but for. So, you have to run the algorithm say 10 10 number of times, but if you see for a random pattern generation, when that pattern is applied and tested, for which faults are been detected.

Then, in the algorithm random test pattern generation algorithm, all the faults that is, f 1, f 2, f 3, f n will be determined in one go or in multiple number of goes, but for the same random pattern you will be able to find out that, so many faults are detected. So, we have saved in two cases, first the algorithm of random pattern generation is simple and secondly also, there is a if the same pattern detects a more number of faults then, do not you do not have to run the algorithm repeatedly.

Then, we have see that, so for some cases, sensitization random pattern generation is a much better approach than sensitized propagate and justify.

(Refer Slide Time: 03:29)

Design Verification and Test of
Digital VLSI Circuits
NPTEL Video Course

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

But, this is not the end of the story, because we have seen that, for if this a one random pattern applying a random pattern generates some new faults are detectable then, you apply next random pattern, there are few more number of faults detected and so forth. But, after some time, say around 90 percent of the fault coverage, so this saturates.

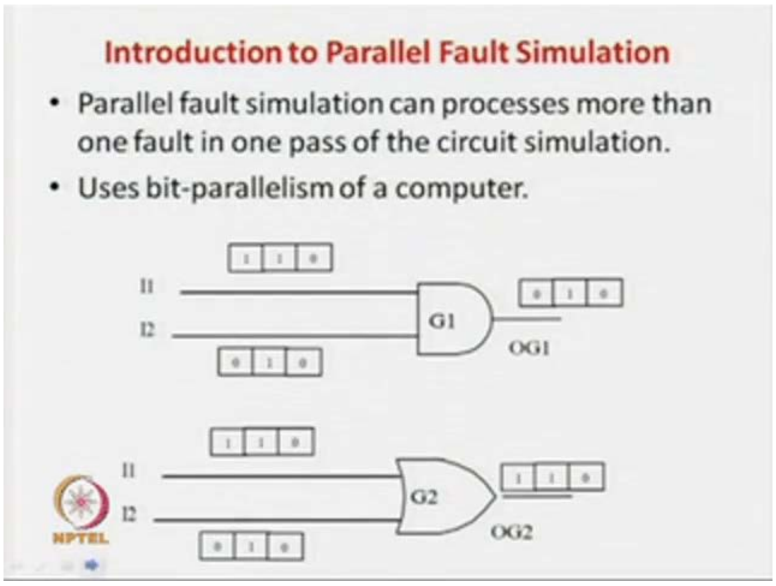
So, the after that, what happens if you apply a random pattern, no new fault will be detected and the procedure will keep on going till some 1 or 2 new faults will be detected and so forth. So, in that case, the number of random patterns are determining, whether a faults is detectable by or not, because more complex in sensitize propagate and justify approach. So, we term them as difficult to test faults and for those, we better apply sensitized propagate and justify approach.

But, for 90 percent of the cases, our random test pattern generation is a very good approach, that we have seen in the last lecture. Then, we have gone for that, if you want to use random pattern generation, a random test pattern generation procedure then, one very important part is that one that, there should be a procedure called fault simulation and circuit simulation. That is, given a pattern, you have to find out what is the normal value of the circuit that is, the value of the circuit output without any fault.

And then, you have to find out, what is the value of the circuit output with a fault that is, you have simulate this results then, you have to simulate the circuit, because you do not have the do not have hard copy of the circuit. And then, if the output differs in the normal case (( )), the faulty circuit then, we know that, the fault is detected by the random pattern.

So, fault simulation was a very important part of a random test pattern generation and we have seen the two types of fault simulation, one is event driven simulation and one is the simple compiled code simulation. And then, you have seen that, event driven simulation is very important, because it simulates only those part of the circuit, where there are changes in the signals.

(Refer Slide Time 05:13)



Introduction to Parallel Fault Simulation

- Parallel fault simulation can processes more than one fault in one pass of the circuit simulation.
- Uses bit-parallelism of a computer.

Then, we found out then, we have taken a very simple test pattern, what you call sorry fault simulation approach, this is sequence sorry sequential or serial or sequential based kind of fault simulation, which we call the serial fault simulation, in which case one test pattern on for a random pattern, you find out whether one fault is detectable or not then, you take another fault and find out, whether it is detectable or not and so forth. So, you take 1 by 1 fault at a time for each pattern and then, if the fault is detected, where the pattern you drop it and so, it proceeds.

Then, we find out that, this is not very efficient, because for each fault, in the most case we have to simulate the fault n number of times if there is n number of random patterns are tried. So now, then we saw that, there should be some improvement on this one and then, we told that, from the next lecture onwards, we will study some important other advanced fault simulation algorithms, which will parallelize the efforts like if the which is I mean, in case of serial, one fault at a time is considered to be test check that, whether the fault is detectable by the random pattern.

So, instead here what will do, we will take a one random pattern and then, try to verify more than one pattern sorry more than one fault can be detectable that is, verification that, whether more than one fault is detectable by that random pattern is possible or not, that we are going to do. So, there are many algorithms like parallel fault simulation then, we are actually we will see there are some other algorithms. So, as the name suggests, first we will try with the parallel fault simulation that then, detect the fault simulation is there and many others.

So, we will see three more algorithms, which are mainly towards the working on paralyzing the fault simulation approach. So, what is parallel fault simulation, as the name suggest you can see, parallel fault simulation actually can tell, what is the for a give a random pattern it can tell you, which are the faults that can be detectable in one run of the circuits. So, one run of the circuits means, one parallel simulation, one fault simulation of the circuit.

So, what you call, a simulation of a circuit means, you generally take some inputs and then, you travels to the output, this is actually can you can say that, I am scanning my circuit 1, this is call the scan of circuit. So, you are scanning the circuit 1 and in one scan you can determine, for a given random pattern, more than one number of fault that is, parallely you can parallelize your effort. You have to remember that, in serial fault simulation, only decision about only one fault can be done.

So, how is it based, these are it actually use a bit parallelism of a computer, so let me just elaborate this by simple example. So, whatever you take, if you are see our computer the hardware, it is a adder, multiplier, subtractor, which is in our processor. So, whenever we execute some C program or any other programming language, inside the processor, there is bitwise operation to finally do the operation. Because, we know that, our computer is may be a 32 bit machine or 64 bit machine, but generally in, but in the processor level, everything works at the bit level.

So, if you add 2 bit 2 3 bit numbers say 1 1 0 and 1 1 1 then, what happens basically that is, actually this is I mean, this is 7 and this is you know that is 6. So, if you do 6 plus 7 in a C language basically what happens, inside the processor this is what is happening, binary it happens, it is 1 plus 1 is 1 then, 1 1 1 carry 1 then, 1 and 1 in a something like this. So, basically what happens, what I mean to see is that, this hardware operation, this bit wise AND bit wise OR whatever, are happening by our hardware adder or hardware multiplier or hardware subtractor, whatever is available in your processor.

So, there is a parallel and we know that, our machine is 32 bit machine or 64 bit machine. So, there is 64 bit processor, so 64 bit ALU, so which may have 62 bit adder, which may have 62 bit AND ending operator bit wise ending operator, 62 bit bitwise OR operates, so all the operators are available. So, parallel fault simulation actually take what do you can call, takes the advantage of this architecture of the machine. Say for example, I want to parallelize the output simulation, I want to parallelize this simulation of the circuit, the circuit is very simple as an AND gate.

But, I what I want to do, I want to find out, I want to simulate the circuit for 3 inputs, one is 1 0, one is 1 1 and one is 0 0. So, if you take a, even if you take a I mean, what you call compile code simulation or even if you take a event driven simulation or whatever you do, basically then, it will if you run simple C code, so what will happen. First it will simulate for this one then, it will simulate for this one and then, it will simulate for this one.

But, if you can write a bit wise operation code that is, you know that, I have to simulate for all then, what you can do, you can say that, this is my first, these are my 1 bit input number and for 3 3 bits are there, bit 1, bit 2, bit 3, because for three instances, you have simulate your circuit. So, what you can do, instead of writing a FOR loop and then, simulating for 1 0, 1 1 and then, 0 0, you better represent it, this has number called 6 and this one you represent as the number called 1 and you do bit wise AND. So, if you are doing a bitwise AND, immediately you will get the answer 0 1 0, which is nothing but, 2.

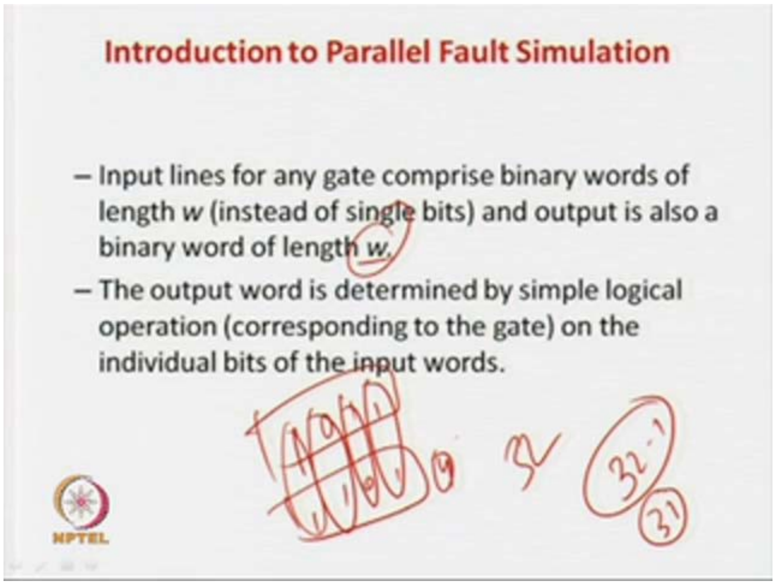
So, what happens, inside your circuit if anybody work doing what you call bitwise this one and bit wise AND with 1 0 1, internally it can be done parallely, because the internal hardware of your circuit or internal ending operation of your circuit, it can if it is the 32 bit processor, it can support 32 bit parallely, if it is 64 bit machine, it can support 64 parallel bitwise operation. So, in this case, in one go your output will be something like 1 0 1 and if you can get these three be parallely.

In other words what I am saying, so instead of giving three individual bits or three individual random patterns in three different goals, what we can do is that, we can think the three random inputs or n number of random inputs, you represent them as a vector you represent them as a vector of as a vector of binary numbers. And then, for example, if you are using 4 bits four numbers say for example, you have to simulate to find out the value of the circuits for four different inputs then, you have to write say 1 0 1 0 and this one may be 1 1 1 0 zero something like this.

Then, you represent this one as one vector and this one as one vector done, what you do then, you do a bit wise AND operation then, it will be done parallely. Instead of first to simulate for 1 1 1 then, 0 1 then, 1 0 and 0 0, so it will be 4, but if you are representing this this 4 inputs as in the 2 input cases as 1 1 as a vector of this size, 4 bit vector as binary numbers. And then, you do bit wise operation then, it will be in a parallel then, it can be done parallely in one scan of the circuit you can do it.

Similarly, this is for the AND gate, this is for the OR gate, same thing you can do, but now instead of AND, you do it bit wise OR operation.

(Refer Slide Time 11:54)



Introduction to Parallel Fault Simulation

- Input lines for any gate comprise binary words of length *w* (instead of single bits) and output is also a binary word of length *w*.
- The output word is determined by simple logical operation (corresponding to the gate) on the individual bits of the input words.

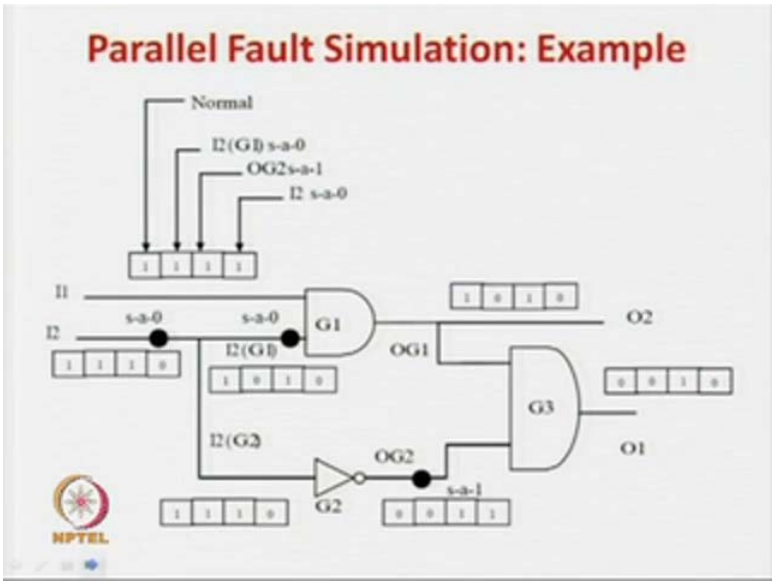
So, these are the basic idea a sorry of parallel fault simulation, so generally what happens, so if parallel fault simulation mainly depends on, what is the parallel efficiency of your computer. So, if your computer is 32 bit machine, so you can do 32 bit wise operation parallely then, 32 minus 1 actually we will see how and 31 faults can be simulated in one go that is, we will see how. So, it is actually called, we will see then, inputs lines of any gate comprise binary words, instead of single bits and output is also binary word of length of W, that is what I was discussing.

So, if your computer is having a, so what can be the maximum W that means, what it is saying that, now instead of each random pattern, so what we are trying to do, we are representing the whole thing is the binary vector. So now, what is the maximum length of the binary vector like for example, as I told you, so in the last example, we have seen four four bits like 1 1, 0 1, 1 0 and 1 1. So therefore, four numbers we have tried, this was one number, this was another number, this was one number, this was one number for simulating the circuit then, we represent them as two vectors.

So, the here the length of the each vector is 4, so one of the question is, what is the length you can go, the answer is, it is the length is maximum in a bit length of your machine. So, that is what actually is called W that is, the bit word length that is the W, so output will also be a vector now, instead of a individual individual output, it will be again a vector. If you call this W, what is the maximum length of W, the maximum length of W is the bit level parallel is a or bit level or the bit level parallelism supported by your complete architecture.

That is also can be roughly said that, if you measure 32 bit architecture, so it can go for 32 bit parallelism and the output is determined by simple logical operation or corresponding of the individual bits. They actually this can be done parallely, because we know that, a bit wise parallelism is in our machine. Now, we will see that, if W of a computer is a 32 then, for fault simulation, we can go only for 31 faults and 1 bit has to be deserve for normal operation that we will see.

(Refer Slide Time 14:02)



So that means, if you know that, if your computer is 32 bit machine then, what we can assume that, in one go, we can result about not 32, but 1 less than that, we can receive only about 31 faults in one scan of the circuits. So, we will see, this concerned by an example, because the concept is otherwise bit listening directly on the theory as I was mentioning, is not very helpful, because it creates confusion. So, we will directly go on for an example and then, we will see.

So, for this, let us assume assume that, W is equal to 4 here that is, your machine is 4 bit processor and only 4 bit numbers can be parallely XOR, AND, OR, NOT, etcetera can be done only 4 bit at a time, so this one. And as we already discuss, so in case of, if your machine W is 4 then, resulting about only 3 faults can be done in one go, not more than that. So, how it is represented, so what happens, so this let be this your circuit, so we know that, from our earlier discussion that, each individual line like this one, this one (Refer Slide Time: 14:57) and this one, the three fan outs.

They are individual lines like this one is one individual line, this one is one individual line and this one is one individual line. So, why they are individual lines, because fan-out lines will always be individual cases, next what we are doing. So now, what happens and the all the other needs obviously are individual lines, now corresponding to each individual line, you should have an array. So, what is the length of that array with a one dimension array, so what is the length of the array, the length of the array is W.

So here is 4, so we have 4 D arrays, you can see at all the points of your lengths of a circuit. Now, each places or each box or each individual element of the array represents some special thing. So, what is this, the first bit or the first box of your house correspond array corresponds to normal performance of the circuit that is, this is it tells the response, this array at this array, which corresponding to this net. So, this place states that, what is the value of this net, this place tells that, what is the value of this net when there is no fault in the circuit.

Next, there are three other boxes as I told you, because W is equal to 4 here, so 3 can use for fault simulation, so there are three boxes here, so each box corresponds to three different faults in the circuit. Say, this one the first box corresponds to I 2 G 1 stuck-at 0, so this one is I 2 0, I 2 G 1 stuck-at 0, so this box corresponds to this fault. Next box is I 2 OG 2 stuck-at 1, so where is O G 2 sorry I 2 G 1, this is I 2 G 1 stuck-at 0 just a minute, so this is I 2 sorry this sorry, this one is O G 2 sorry this one is O G 2. So, the next next this one is called the O G 2 stuck-at 1 fault, this one is stuck-at 1 fault and I 2 this one is I 2, so this one is stuck-at 0 fault, so this place corresponds to this one.

Now, this is for normal, this whenever this array is associated with line, first box implies that, what is the value of this net when the circuit is normal. Then, the second place tells that, I 2 G 1 it corresponds to this fault, so it states that, what is the value of this line whenever this fault is there. Second the third place tells that, O 2 stuck-at 1 is for this fault, so this this net this this box for this array states that, what is the value of this line whenever this there is a fault over here, the steady stuck-at 1 fault.

And the last box corresponds to stuck-at 0 fault, it also states that, what is the value of this net whenever there is a stuck-at 0 fault over here. So, and then again obviously, as you can see, there is an array at all the positions, so for example, this array what does it represent, this this place of this array represents, what is the value of these net these net when the circuit is normal. So, this box again states, what is the value of this net whenever there is a stuck-at fault here.

This box states that, what is the value of these nets sorry this (Refer Slide Time: 18:01) net, not this one, this fan-out net whenever there is stuck-at 1 fault here. Similarly, last box states that, what is the value of this net whenever there is fault at this net, so as you can see, so it is the what you call the this has means one one one dimension array at each net in the circuit. So, the whole each each this array at each point states, what is the value of that net at normal condition and each of the fault condition is represented parallely.

And you have to also one point I have missed, so we will start the fault simulation, so with this representation, we start. Then say for example, we give 1 1 at the input that means, this is our random pattern here is 1 1, so what is the random pattern, it is the 1 1. So now, for the random pattern 1 1, what is the value of this represents, this box represents what is the value of this net when the circuit is normal. As the random pattern is 1, so obviously, the value of this net is 1 at this point, so the first box will have 1.

Secondly, similarly for this net, so what is the value of this net when the input is 1 and this circuit is normal, the answer is 1, so that is why, this first two places in my array at I 1 and I 2 are 1 1. Now, let us see what happens, now let us the first box, the first box says that, here in this case, the first the first box say, the first box say that, what is the value of this net this net whenever there is a stuck-at 0 fault here and input is 1 obviously, random pattern.

So, this stuck-at 0 fault here has no effect on this line, so obviously, the answer will be 1 here, because if you apply a 1 over here then, it is stuck-at 0 fault here, this line is free of here. Similarly, this second box corresponds to this stuck-at 1 fault, so the stuck-at 1, one stuck-at fault here has no effect at this net. So, the value will be 1 here, because 1 applied as the input, similarly this stuck-at 0 fault has no effect here, so the answer will be 1 in this case.

So, this array will be all values will be 1 1 1 1 here, now let us see for the second one, now this one will be already 1 we know, so the circuit under normal condition, this net value is 1,.so it is 1. Now, you know that, this stuck-at fault has no effect on this net, no effect, so the value will be a 1, also this stuck-at fault has no effect here, so the answer will be 1. But now, the interesting thing is that just see, so now, just see here, actually the answer should be 1, because you are applying a 1, but this box corresponds to this stuck-at 0 fault, this stuck-at 0 fault and this stuck-at 0 fault has an effect at this line.

So, if this value will not be 1, but it will be a 0, which is shown here, so these two box are unaffected, because this fault and this fault do not have any effect on this net, only this fault has an effect. So, what is we are going to observe that, instead of 1, this value will be a 0, so this is how the input array has been filled up. Now, what you have to do, now we can easily see that, these two values can be filled up, because fan-out you just get propagated.

So, this will be directly propagated here without any changes, because this net because a this net, which is this normal case, if this answer is 1, this value of this net is 1, so it is done. So, this fault does not have any effected on this net., so it is a 1, same value is propagated from here, so again this fault do not have any effect on this, so this value is propagated here. And similarly, also this fault that is 0, so this fault has an effect here so that means, it is the stuck-at 0 fault here.

So, this one gets 0 here, because the 0 stuck-at fault, which is a also effected here at this net sorry this net is also affected, because of the stuck-at fault, so you get the answer a 0 over here. So, basically what happens, this box shows that, if the input is 1, this input is 1 then, under normal case, this net is going to have 1, under this stuck-at fault the answer is 1, under this stuck-at fault the answer is 1 and under this stuck-at 0 fault, the answer is 0.

So, only you can see that, this stuck-at 0 fault has an effect on this net is also obvious, so this array becomes now 1 1 1 0.

So now, in this case also, now when your propagate in this, so if you having 1 over here, so in the normal case, if this net is also going to be 1 correct then, this fault is having a effect on this one. So, this 0, this 1 gets converted to 0 over here, because if you are applying a 1 over here, so this is 1, because this fault is not having any effect here, but the stuck-at 0 fault is having a affected on this net. So now, in this, this 1 will not be propagated here, but it will be propagated as a 0.

But, this fault do not have any effect here, so 1 will be remaining a 1 and this fault this stuck-at 0 fault also has an effect at this net, so this 0 remains a 0 over here. So now, this array will be a 1 0 1 0 kind of thing, now what you have to do, now again let us see what happens. So, thus, just you have to run this is an or gate sorry NOR gate, so again you have to do a logical operations. So, as we know that, we have a parallel logical operators are available over here.

So, what you can do, so logic NAND, logic NOT of this array, so logic NOT should be not should be equal to 1, this is 0 0 0 and 1. So, this should have been the case here, here inverting this, but you see that, this is not a 0 over here, so this is correct, this is correct, but you see this 0 has been converted to 1. So now, why is that case, the case is so because, you should have been 0, because by this parallel NOT operation, you should not have been 0.

But, this net, this last of the third house corresponds to this thing stuck-at 1 and this net this net this fault has the effect on this net and which is converting the 0 to a stuck-at 1 fault and so, the answer here will be a 1. So, you are going to get as a 0 0 1 1, which is the answer over here, so this this only this is a this thing has been converted sorry this is a mismatch or what you can say is on consistent this one, you should have been a 0 by negation, but again this stuck-at fault is taking it is role, stuck-at 1 fault is taking the role and everything is getting changed over here.

So now, so this is your array at this point and now, you can see, again we can do a AND operation between these two, so in this case it is 1 0 1 0. So, this and this and this, you will get 1 0 1 0, because this is no fault over here, so no changes are there and you will a get consistent answer, so it is 1 0 1 0 and this one is 0 0 1 1. Now, again you can do a

AND operation here, again there is no fault at this area, so there is no effect will be here, so you can directly keep on doing the AND operation, so you are going to get 0 0 1 0.

Now, there will be only 1 here, so it we get, so the final answer 1 0 0 1, now so, what happens, this one interesting thing you have to observe that, we have gone only once from this part of the circuit to this part. First you have generated this array then, you have generated this array then, you have generated this array then, you have generated this array then, you have deal a negation. Obviously, at each state, you have to find out if there is any effect of the stuck-at fault in the presence of the array.

Otherwise you will just simple do a AND operation, OR operation, NOT operation whatever is the case. And then also, here also you get the answer, this will be just a propagation, that it is not different then, again you do AND operation. Now, you see VLAN and in one iteration of the circuit or one scan of the circuit, we are able to complete our operation, that is very important. Now you see, what does this two array signifies, so in one scan of the circuit, you have generated these two arrays say, array 1 which is for this output and array number 2 which for this output.

Now, you will see what happens, this is very important, see this corresponds to normal. So, it is says that, if your input is 1 1 then, the output at this net is 0 under the normal condition and it says that, if the answer is input is 1 1 at this net, the answer is 1, that is what is it say. Then, you says that, the second place says that, if the input is 1 1 and I 2 G 1 stuck-at 0 and this stuck-at 0 fault, the output of this net is 0. So, what it says that, for random pattern 1 1 at this net and this fault, the outputs are same.

So, this fault this fault is not detected by the random pattern 1 1 through this one, through this net, because the input is similar. Similarly, for the last fault that is, your I 2 stuck-at 0, so this being stuck-at 0, so what is says that, for random pattern 1 1, the output at this net under normal condition is 0 as well as for this fault is also 0, the fault cannot be detected. But, on the other hand, the third fault like O 2 stuck-at 1 this one, for the random pattern 1 1, here the output is 1 by this net.

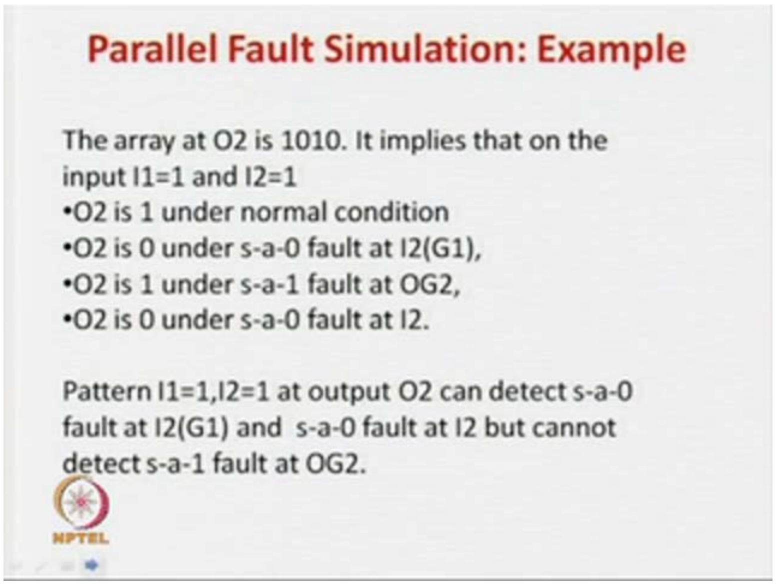
But, for the normal case, the answer is 0, so this fault is detected by the random pattern 1 1 detected at this, so that is what it signifies. But, there is another output over here you have to see, so in this case, this is the normal point and there is one point, which is a difference. So, this is different from this one, so this pattern, this output this output a 1 detects and this random pattern 1 1 detects this fault, so 1 fault is detected.

Now, again there is one more primary output you can see, so in this case you see, this is the output, the circuit at O O 2 when the random pattern is 1 1, it is 1 and there are two other faults and in two faults is the output is different. This is similar, if it is similar then, at O 2, this pattern cannot be detected, but these two faults I 2 G 1, it is I 2 G 1 this pattern and this fault and this fault and last one I 2 stuck-at 0. These two faults are giving a answer 0 0 when the input is 1 1 and the circuit under normal condition is 1 at O 2.

So, this random pattern 1 1 detects this fault and this fault at O 2 and this fault at O 1 for the random pattern 1 1. So, this experiment or this algorithm parallel algorithm what it does, it one scan of the circuit, it determines that, random pattern 1 1 detects this circuit, this stuck-at 1 fault at this output and this one and this one at the same. So, what is the emphasis, in one iteration this is possible, but if you are if you would have been normal random pattern generation procedure then, what you have to do then, you you have to do this three type, the three scans of the circuits are required.

So, one is for the normal then, you have to apply 1 1 then, you have to simulate this fault and you have to find out, whether this is different or this is different, one of the different then, you find that and detect it and the fault is drawn. Then, again you see for simulate faults simulate the circuit for this one and the P T then, again you fault simulate for this onr and you repeat. So, three times the circuit scanning is required, but here we did not require the three times of circuit scanning, we have gone only by one parallel procedure and we have got the answer.

(Refer Slide Time 29:12)



Parallel Fault Simulation: Example

The array at O2 is 1010. It implies that on the input I1=1 and I2=1
•O2 is 1 under normal condition
•O2 is 0 under s-a-0 fault at I2(G1),
•O2 is 1 under s-a-1 fault at OG2,
•O2 is 0 under s-a-0 fault at I2.

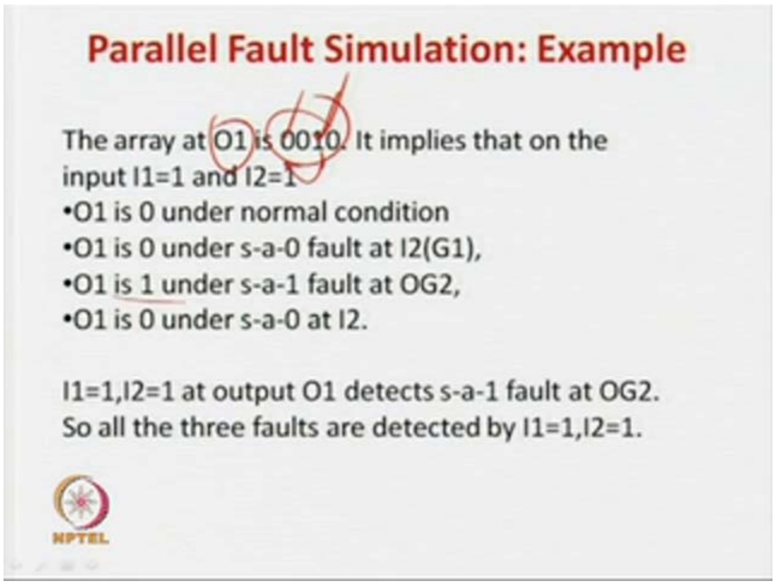
Pattern I1=1,I2=1 at output O2 can detect s-a-0 fault at I2(G1) and s-a-0 fault at I2 but cannot detect s-a-1 fault at OG2.

That is, in one iteration we have got the answer that is, 3 faults are detectable by the random pattern 1 1 and where it is detectable that is also we have achieved. So, this is what is parallelizing your serial fault simulation algorithm, so that is very this algorithm is very efficient, but one thing you have to remember that, your circuit is having 200 faults, still you have to repeat the procedure, because you can parallelize only up to that limit in, at which you have the parallel bit array or parallelism supported by a processor.

So, if your processor is 32 bit processing, parallel processing is possible then, you can parallelize upto 32 bits, so 31 number of faults you can detect in one go. So, if you have 200 faults that is, you have repeat at least 200 by 31 upper selling , so that mean number of times you have to repeat the procedure. Because, here you limited by the array size and the array sizes the maximum, which you can have what you by called I mean, the parallelism which is supported only procedure.

So, again which your viscous, so this is representing in the array, the array O 2 is this is the output then that means, what it says that, O 2 is one under normal condition, O 2 is 0 under this fault, O 2 is 1 under this fault and O 2 is this one. So, you can easily find out that, these two faults are detected and this fault is not detected.

(Refer Slide Time: 30:31)



Parallel Fault Simulation: Example

The array at O1 is 0010. It implies that on the input I1=1 and I2=1

•O1 is 0 under normal condition

•O1 is 0 under s-a-0 fault at I2(G1),

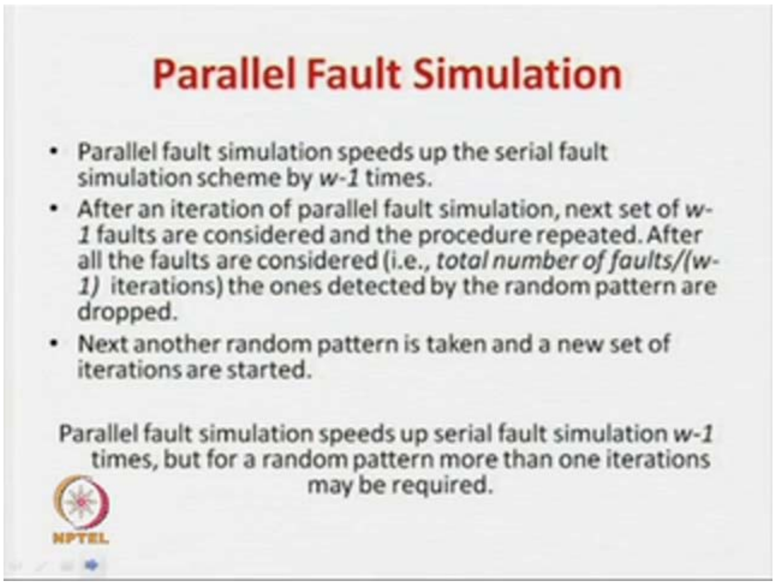
•O1 is 1 under s-a-1 fault at OG2,

•O1 is 0 under s-a-0 at I2.

I1=1,I2=1 at output O1 detects s-a-1 fault at OG2.
So all the three faults are detected by I1=1,I2=1.

What you have discuss, you have again writing showing in next, similarly for O 1 this was the answer, this is the normal condition and so, you can see that, there is only one point, which is the different. So, this pattern, so pattern 1 1 at O 1 detects only this thing corresponds to this fault. So, this is the only fault, which is detected, which you have already discuss, so this just again I have rewriting in the text form.

(Refer Slide Time 30:51)



Parallel Fault Simulation

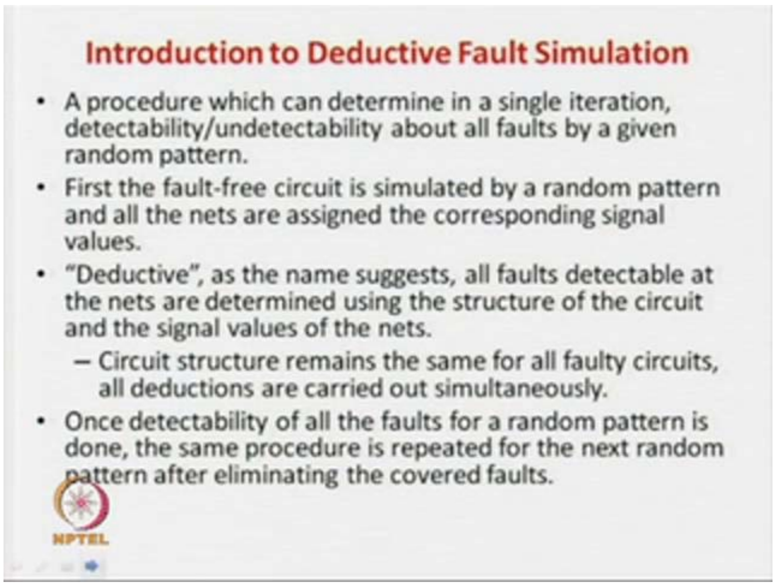
- Parallel fault simulation speeds up the serial fault simulation scheme by *w-1* times.
- After an iteration of parallel fault simulation, next set of *w-1* faults are considered and the procedure repeated. After all the faults are considered (i.e., *total number of faults/(w-1)* iterations) the ones detected by the random pattern are dropped.
- Next another random pattern is taken and a new set of iterations are started.

Parallel fault simulation speeds up serial fault simulation *w-1* times, but for a random pattern more than one iterations may be required.

So, what in summary, what we have seen, so we are seen that, in fault simulation we parallelize or speed up your fault simulation procedure W by n W minus 1 time, if you consider W is the bit wise parallelism, which is supported by your machine. So, what is the number of times you have to repeat, this is total number of faults by W minus 1 iteration, again you have to take a upper selling. So, again if a new pattern is applied, so after that what you have to do, so I mean, first step you have to apply one random pattern.

Then, you can try out your things for W minus 1 number of faults and then, again you have to repeat it, repeat it how many number of times, total number of faults by W minus 1 upper selling. And then, when you have done this for all number of faults, that is for this value then, you can what you can do then, you can drop the fault which you have been detected and then, again you take a new random pattern and repeat the whole procedure. This is also very simple similar to serial fault simulation, but only thing is that, in one go you can result about W minus n number of fault, so this way is a parallelism you have obtained.

(Refer Slide Time 31:53)



Introduction to Deductive Fault Simulation

- A procedure which can determine in a single iteration, detectability/undetectability about all faults by a given random pattern.
- First the fault-free circuit is simulated by a random pattern and all the nets are assigned the corresponding signal values.
- "Deductive", as the name suggests, all faults detectable at the nets are determined using the structure of the circuit and the signal values of the nets.
  - Circuit structure remains the same for all faulty circuits, all deductions are carried out simultaneously.
- Once detectability of all the faults for a random pattern is done, the same procedure is repeated for the next random pattern after eliminating the covered faults.

But now, you see what happens, so if your computer is very good, you have lot of parallelism impossible then, your answer you can get it very quickly. But, on the other hand, generally our modern computers are even a very fast computer, parallelisms are generally not more than 32 or 64 bit parallel bit wise operations are possible. So now, we will see another algorithm, which is actually call detective fault simulation, in which in one go, we can result about all (( )) faults in the circuit, whether your circuit whether your computer has parallelism, with your computer not has parallelism, that does not matter that much I mean, bit wise parallelism does not matter.

We will now handle it algorithmically, we now handle it in the hardware level of it, so the last parallel fault simulation you have taken, what you can say is the hardware advantage you have taken. That is, these processor has bit wise parallelism, which can parallelize your efforts, so that advantage has been taken by the parallel fault simulation algorithm. But now, this is the new algorithm, which we are going to study, this is the third fault simulation algorithm, in which case what we have doing.

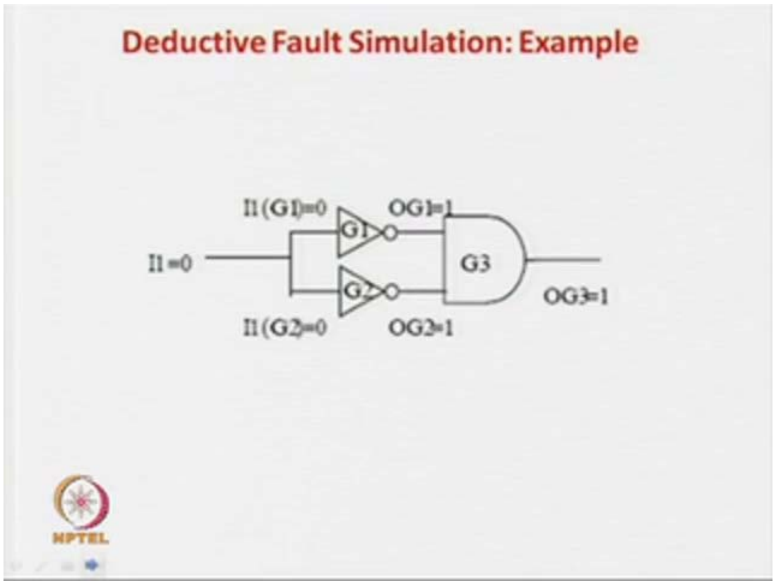
We are not going to look into the, we are not going to make a ourselves dependent on what you call the hardware of the computer. Hardware architecture of the parallelism for the computer, whether will make a algorithm in such a way that, in one go you can result about all the fault, that is our main first goal. And then, we are not going to use parallelism and all those bit level parallelism, all those things into picture, so that is our main goal.

So, we are going to see in details about the detective fault simulation, so this I mean, as I already told detected as the name suggest that is, it can determine in single iteration about the delectability and undetectability about all faults in by one random pattern, that is here. We are going to apply one random pattern and then, you are going to find out, whether all faults are detectable by that one, that is what is very important. So, in this case what is happen, the basic idea here is, so now, what is the motivation in this case.

They say that, this circuit structure remains same for all the faulty circuits that is, generally it does not happened that for a stuck-at fault. Few more AND gates have somehow come and come from the air and they are going to be regenerated or because of some stuck-at 0 fault, some of the AND gates will vanish, that is not going to happen. They say that, this circuit structure remains same for all the faults circuit, same in the idea means, number of gates or the looking of the structural identity of the circuit remains same.

So, that is why, they say that, why we cannot find out the algorithm, which can result about all the faults at a time in one go. Because, if the structure of the circuit changes then, you have to go for the differentiations of the circuit. So, that is the basic philosophy that, why this excuse me why it will thought that, this idea should, this algorithm or idea should be hold, should hold.

(Refer Slide Time 34:47)



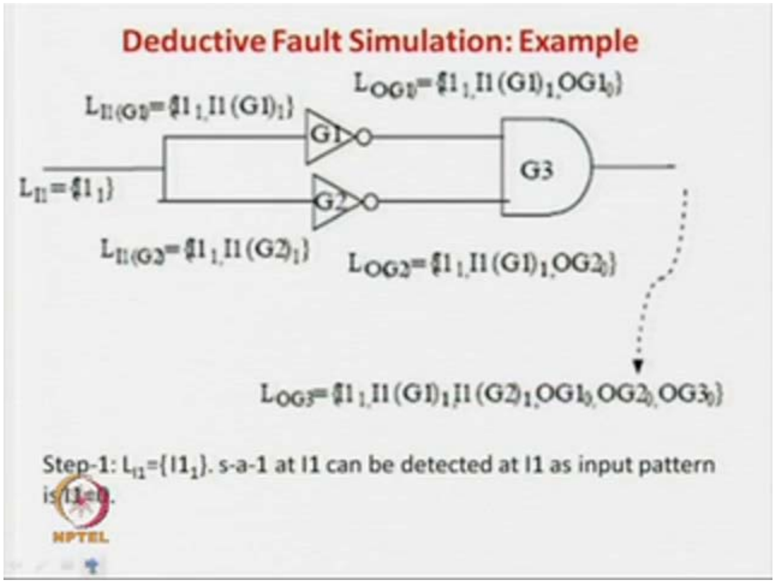
Deductive Fault Simulation: Example

So, let us see how it is going to happen, so then what basic procedure will be, will take a random pattern, determine idea about all the faults in circuit. That, which of the faults detectable by the pattern and which are not then again, which will be detected will be brought and then, a new new random pattern will be taken. So, even if you have 10000 faults in your circuit and you computer has 32 bit parallelism, but still your answer is going to be found out in only one iteration of the circuit, you did not go on a 1000 or 10000 mod W minus 1 upper selling.

So, that also sorry in a 10, even if this 10000 faults, you can find out the solution in one go, it did not have to wait for this number, you did not have to go for this number of iteration. Now, what is the philosophy behind it, they say that, the multiple iteration should only be required if your circuit structure is going to change for different faults, but here if the structure is same, the same algorithm should hold.

So, that is the idea of deductive fault simulation and deductive as the name suggest, is that to deduce about the faults which are detectable in one go for a random pattern, that is why the name called deductive fault simulation. So, what you do, here one thing you have to understand, we have taken a very simple circuit this one and first we have give this random pattern 0. So, if the random pattern I 1 G 1 it will be 0 over here, 0 over here, 1 1 and O G 3 output is 1, so that is very simple, first our normal fault simulation is done.

(Refer Slide Time 36:09)



Deductive Fault Simulation: Example

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

Step-1: $L_{I1}=\{I1_1\}$. s-a-1 at I1 can be detected at I1 as input pattern is I1=0.

Now, we are going to see for a deductive fault simulation of the circuit, so here one representation you should understand. So, if if this I am saying as I 1, so then I will represent L I 1 that means, it is the fault list of that net and when we say that, I 1 under score 1 that means what, it is detecting I 1 stuck-at 0. That is, for any net say, net is named is I then, L I is the fault list that is, for that given random pattern, which is the fault that is being detecting at that line, so that is L I.

Now, that is faults detected at I net by a random pattern and then, if you say that, I under score 1 that means, I stuck-at 1 is detected and if you say that, I 0 then, I stuck-at 0 is detected, that is what is the notation we going to follow. So, initially just again, you see I Y 1 equal to 0 has been given, that part you have to remember. So, if you see that, if 0 has been given, so obviously, a stuck-at 0 will be detected here.

So, write L of I 1 equal to I I 1 that is, this list will have detect only one fault for the random pattern that is, stuck-at 1, so that has been written over here. Now, this one, now when you are coming over here, so in this net for the random pattern, what are the faults will be detected. So obviously, stuck-at 0 will be detected here, you can also detect the stuck-at 0 fault over here, as well as a stuck-at 0 fault will also be able to detect here, because sorry stuck-at 1 fault you can also be detect here.

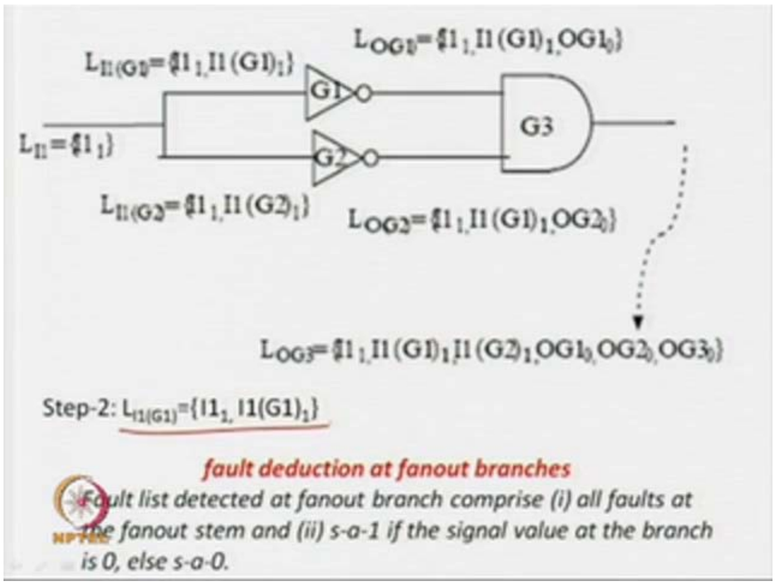
That is, I 1 G 1, this line is I 1 G 1, a stuck-at 0 1 stuck-at 1 fault at this net can also be detected at this line why, because you are applying 0 over here. So, if you are applying a 0 over here then, if this net is stuck-at zero, so the stuck-at 1, the answer will be 1, any normal case also will be 0. Again you are applying a 0 over here, if there is a stuck-at 1 over here, the value of this net will be equal to 0 is the normal case and in case of stuck-at 1, it will give a 1 over here.

So, I l of I 1 G 1 that is, this net will be equal to stuck-at 1 fault here, this thing this thing will be here as well as any stuck-at 1 fault will be detected over here. So, this means, this fan out, this value will be is propagated here as well as a new fault is also detected that is, this one. Again for this sub net of the fan out, again obviously, this one will be detected over here, because a stuck-at 0 fault will sorry a stuck-at 1 fault at this net is also detected here as well as a stuck-at 1 fault at this net, that is by I 1 G 2 1, this is about stuck-at 0 fault is detected because of this random pattern 0.

So, the list and one more thing you is obvious we have discuss many times that, a stuck-at 1 fault here is not detected at this line, similarly a stuck-at 1 fault is not detected at this line. So, this a I 1 G 1 1, it is I 1 G 2 1, so stuck-at 1 fault here is detected at this net and stuck-at 1 at of this net is detected at this net. So, you see this list is list 2 and this list 1 is these two, so that is what we are getting for this fault list, that is what a step 1 it is saying that, this one.

So, it is step 1 is saying that, stuck-at 1 can be detected at random pattern this one, this is link about this one, so this about (( )) this one is called sorry, this is about a step 2.

(Refer Slide Time 39:44)


$L_{I1(G1)}=\{I1_1, I1(G1)_1\}$
$L_{OG1}=\{I1_1, I1(G1)_1, OG1_0\}$
$L_{I1}=\{I1_1\}$
G1
G2
G3
$L_{I1(G2)}=\{I1_1, I1(G2)_1\}$
$L_{OG2}=\{I1_1, I1(G1)_1, OG2_0\}$
$L_{OG3}=\{I1_1, I1(G1)_1, I1(G2)_1, OG1_0, OG2_0, OG3_0\}$
Step-2: $L_{I1(G1)}=\{I1_1, I1(G1)_1\}$

fault deduction at fanout branches

Fault list detected at fanout branch comprise (i) all faults at the fanout stem and (ii) s-a-1 if the signal value at the branch is 0, else s-a-0.

So, step 2 is this one, which is already discussed about this one, similarly you can get it for this one. So, for fault it has the deduction, deduction one important rule has to be concluded, so fault detection at fan-out branch. So, what is that, fan-out fault list detected at the fan-out branch, this is a fan-out branch is all faults at the fan-out stem. So, whatever fault list is here, those things you can detected, that is a obvious as well as stuck-at 1 fault if the signal value of branch is 0, else 1, L 0.

So, it says that, so it is the just fan-out branch like this and so, whatever fault list whatever fault list is here, can also be detected in this fan-out as well as that along with that, you have to make a union of, if the value the signal value is 1, when you can detect a stuck-at 0 and if the signal value here is 0 then, you can detect a stuck-at 1 over here. So, this is the detective fault simulation rule at the fan-out, so detected fault simulation is says that, all fault at the fan-outs stem can be detected.

And a stuck-at 1 if the signal value of the branch is 0, if a stuck-at 0 fault in the detectable if signal value is obviously, signal is 1, that is obvious. So, this this is the rule you have to understand about this, so detective fault simulation is all about rules, so these are simple rule we have learnt about the fan-out branch. Now, next is, so this is there, now will go to next step, we will see what is the number of faults detected, but what are the faults that is detectable at this point.

Now, you see, so what happen, so this is a so we are applying we have applied a 0 over here. So, now you see, so if it is the stuck-at 0 over stuck-at 1 over here then, the answer is 1, normal case 0, fault case 1. Similarly, for this pattern, 1 1 0 1, so that is we are applying 0 fault over here 0 value over random pattern over here, stuck-at 1 fault is in this net. So, serial value is 0 1 0 1, 0 normal case, 1 fault, same thing here, here the answer will be 1 0 1 0.

That means what, so a stuck-at 1 fault here is also detectable at this line and also detectable at this line. So, that is why, this fault list has I 1 1 I 1 1 that means what, it shows that, for this random pattern, a stuck-at fault 1 is also detectable by this line and also detectable by this line, so we have this in the list. Next, the same 0 is applied and say, there is a stuck-at 1 fault at this line, this is stuck-at 1 fault at this line, so in this case the answer should be 0, fault case it is 1, so here is this case, is this case, so here there is no effect, so here it is 0 then, here it is a 1.

So, you can see a stuck-at 1 fault here, this one is also detectable here, so this this value is also propagated over here. Along with that, now if no fault is in this part of the circuit say, so now, 0 is there, so you are going to get 0 0. So, here both of them we are going to get a 1 over here, so now, in this case obviously, it can detect a stuck-at 0 fault here here, so G 1 that is O G 1 0, so that is in this list. So, this list if you observe has now three values, one is stuck-at 1 is this one, this stuck-at 1 this can be detected at this net.

Similarly, this net stuck-at 0, so this one is also available here and a new fault we are added is this net at stuck-at 0, so this one is added. Similarly, now if you look at this net, so this stuck-at fault will be propagated, this stuck-at 0 fault will be propagated and finally, a new fault that is, a stuck-at 0 over here that is, a O 2 stuck-at 0 is added over here. So now, again we can find out a rule for the inverter, so what is the rule for the inverter, the rule for the inverter is, so whatever fault list we have this one, you just propagate it here.

Because, it is detectable as well as along with that, that union, this list just get propagated here, this is just propagated here and this is a union, what is the union, if the signal value is 1 in case of normal then, you get a stuck-at 0 you can detect and if you 0 in case of normal then, a stuck-at 1 can be detected. In this case, I have the value is 1, so you can detect the stuck-at 0, so that is what we say, the fault detection of inverter is fault detector of an output of inverter comprise all faults at the input and stuck-at 1 of the signal value is 0 and else stuck-at 0.

So, that is, what you have doing in deducting fault simulation, in deductive fault simulation what is our idea, our main idea is that, we first start with a faulty sequence this one and then, you go from the next level, the next level, the next level then, we have some rules. So, what are these rules, these rules are called deductive fault rules, so in case of parallel fault simulation, we would be some bit wise operation, bit wise ANDing, bit wise ORing.

Here, we are not doing that, here what we are having, here we are having fault list and then, you are propagating the fault list by using the deductive rules. That is, here we have saying that, just propagate the value whatever can we detected here can we detect fan-out branch, along with that, add the some faults over here. Similarly, for the inverter, just do that and again add the faults over here and add the faults over here. So, this what is the done, so that is instead of doing some bit wise operation or bit wise this thing, here we are having some detective fault rules.

So, those have I using these fault rules, we are populating slowly by slowly our fault list, which can detected here. And finally, we will go to the primary output then, whatever will be the fault less in this case, are the faults which are detectable by this random pattern at this output. So, in what will happen in one go, we will able find out all the faults, which are the detectable by this random pattern. So, we will just see that, what is our, stil now you are see the rules for this fan-out branch. So, fan-out branch is inverters, now we will see for the AND gate which is going to be very important.

So now, what we are the fault list we have, these are the fault list for this net, these are the fault list for this net and then, you have to find out the fault list for this net. Now, let us try to deduce the rules of a sorry, this was for the inverter for the AND gate. So, we will just see you have see for the AND gate, so this is the list. So, we have to now find out for the AND gate, what are the fault list for the AND gates on deductive rule for the AND gate.

Now, you see, so this is we applied a 0 over here, so now, this is stuck-at 1, so if it is a stuck-at 1 here means, normal case 0, fault 1, normal case 0, fault 1, this is what you call 1 0 1 0. So, in this case, the answer is going to be 1 and 0, so obviously, this stuck-at 1 fault is detectable by this AND gate, so that is why, in this list we are having this one. Now, let us see for the another fault, let us see for this stuck-at 1 fault, so we are applying 0 over here, so it is 0 over here, 1 over here.

In this case what happened, we are applying a stuck-at 1 over here, so now, it was 0 now it was become 1, so it is 0 1, so it is 1 0 and in case the answer is 1 0. So, so in the normal case, the answer is 1, in the fault case the answer is 0, so this stuck-at 1 fault is also in this list. Now, let us see the other faults of this other net that is, other part of this fan-out branch. So, this is we applied 0 over here, so here here we are having stuck-at 1, so here it is 0, answer is 1, here it is stuck-at 1.

That means, normal 0, fault 1, 1 0 then, it is 1 0, so fault is again detected, so that is I 2 G 1 is also add this list. So now, let us see for the fault list over here, so now, for the fault list over here, is this one, so normal 0, 1 0 0, this is 1 1. So, let us see this is a stuck-at 0 fault over here, so that is on 0, normal 1, so it is 0 and a 1, again fault this stuck-at 0 fault, because this stuck-at 0 fault, this is 0, in the normal case that is 1. So, here we have the answer is 1 by the normal case, so it is this 1 in the output and again this stuck-at 0 fault is detected over here.

Similarly, you can easily verify that, if you have stuck-at 0 fault here this one, stuck-at 0 fault here, it will be 0 1, again this list will be there. So, this so all the faults here, here (Refer Slide Time: 48:01), here, here, everything is detectable by the AND gate for the random pattern 0. In the last, if there is no faults in the other part of the circuit excepting the output that, it is a 1 and a 1, in the answer is 1, so obviously, and the stuck-at 0 fault here that is, 1 0 is also detected over here.

So, you see in this random pattern what happens, we have detected all these faults and this is a new fault which is added, so we have to make a rule for the AND gate. So now, let us see, but this is the whatever I am discuss then, you have taken one fault and traversing the whole thing to find out you detected then, you want taking a fault at traversing the whole thing to be detected. So, so this is not a very good idea of doing it, so what we are try, what we have to do is, a bit different, whether we using the rules just like a what we are found out that, so this is the fault at this net.

Now, it is a fan-out branch then, this will have to be propagated then, this will have to be propagated, we not check, whether this fault is detectable by making 0 1, 1 0 and all those things. Just apply a deduction rules that, this one is the fan-out branch be propagated, now the signal value is 0, so stuck-at 1 will be deducted, stuck-at 1 will be deducted. Now, this is an inverter, the whole thing would be copy pasted over here and it has the value is a 0 over here sorry signal value is 1, so the stuck-at fault will be detected.

Similarly, for this case, this whole thing is propagated over here and this one that is, you take a case and you then, again propagate to this circuit is not a very good idea, but this (( )). Now, you will see, how can you get it for the AND gate, so for the AND gate is very simple to get, now you can see that, our normal signal value is 1 1. So now, in a AND gate, you take the case is an AND gate, so if it is 1 1 and this one, so if it a 1, we know that, this gate is a making the gate a pass filter kind of a thing.

So, if this part is 1, so whatever is happening in this can be reflected over here, this is a very good idea thing about AND gate that, if this line is 1 then, all the activities at this case get propagated. Similarly, if this is a 1 and this is a (Refer Slide Time: 50:03) 1 sorry if this is 1 then, also the other cases, so if this is a 1 then, again all the activities at this point get propagated. But, if it is 0 then, whatever happens here, it does not get passed through.

Similarly, for the other AND gate, other AND gate other input of the AND gate that is, in case of a AND gate in this case, so if this is 1 and this is 1, so if there is a fault or some activity is here, so this can be propagated here and you can observe it. Similarly, as this is also 1, so any type of activity, fault detection, propagation, etcetera at this point also will be reflected over here. So, it is a very helpful situation when if both the inputs of the AND gates are 1 then, what has happening is that, so if this is 1, so whatever is here will be propagated over here.

This is because, this input is 4, so this and what you mean by I 1 1, here it says that, the effect of this stuck-at fault here stuck-at 1 fault is available over here. So, this is 1 means, the available fault effect of this fault stuck-at 1 is available here, this fault stuck-at 0 fault effect is available here and SGC is 1, so the whole thing can be propagated over here. So, that is why, whatever is available over here, will come here, similarly you also have a 1 over here.

So that means what, any effect, so anything any fault effect here will be propagated here, because this is 1. So, this is 1 here means that is, this stuck-at 1 fault, this effect can be available here, so this 1 is there means, this stuck-at 1 fault here effect is available here. Similarly, this 1 here means, this effect of stuck-at 0 here is available in this, so as this one is 1, so the whole list will be available here, so this thing is come out. Along with that, here the value is normally in this case it is 1, so this stuck-at 0 fault will also be detected.

So therefore, the fault deduction rule for the AND gate with both input as 1 is all faults at both the input whatever, because means both of them are 1 1 then, what is going to happen, the fault effect of everything, the fault whenever this is 1, the whole thing will propagated, whenever this is 1, the whole thing effect. So, all faults at both the inputs will be propagated, along with that, stuck-at 1 fault at the AND gate output of the AND gate is detected, because the AND gate here is 1 1, so the answer is 1 so obviously and the stuck-at 0 fault will be detected.

So, this is one rule for the AND gate, there is one rule, which you have seen for this AND gate, whenever your input is what do you called 1 1. So now, you can easily see, we can just when this is one rule when the answer is 1 1, this is very simple that you propagate, this whole value here, whole value here you propagate it to the output. Now, we just going to see, so the four cases we have to analyze, today we will just see one more case, in which case here the input may be 0 and here the input may be 1.

So, what is your observation kind of, I think you are correctly guessed, so if says for some reason, the signal is 0 over here and 1 over here. So, I think your guess may be proper saying that, whatever list here, will be propagated here, that is absolutely true. Because, this is 1 is that will pass away, but if something is here, some list is here, that cannot be propagated, because this is 0 and it will block the propagation over here. And obviously, so only this list will be propagated as well as 0 1 means of an AND gate means what, it is the answer is 0, so one more stuck-at 1 fault will be there.

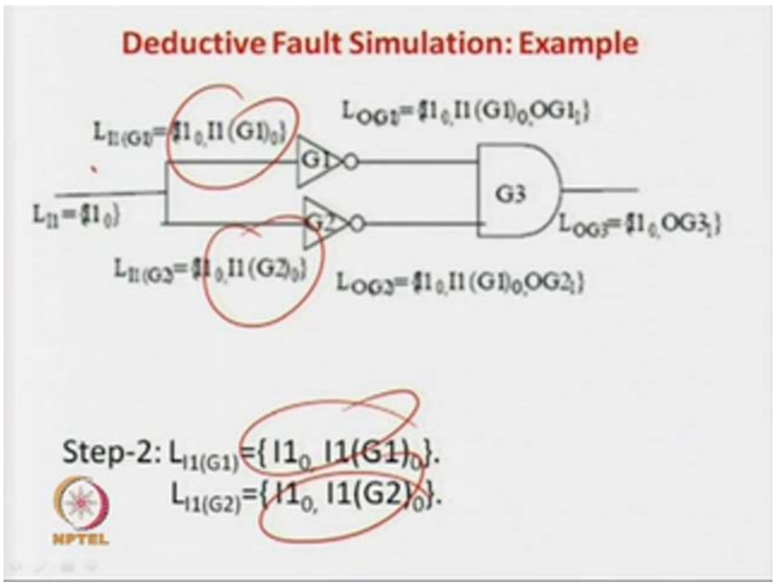
So, let us just see this a very close, so this is one of this case, so in this case, we are applying 1, so the answer is 0 0 sorry, we are seeing another case which is not 1 0, four cases you have to see, 1 1 we have seen, 1 0, 0 1, we have will be discussing that is, I told you that, whatever corresponding to this list will be will not be propagated, whatever corresponding to this list will be propagated and the vice versa for this pone. And in case of 0 0, we have to see, so what gets propagated.

So, all the combinations we will see, so if for 1 1, it is very simple, everything gets propagated, for 1 0 the list here will be propagated and for a 0 1, the list here will be propagated and for 0 0, nothing will be propagated something like this. But, it is not the case for let us see for 0 0 what gets propagated and then, other case 0 0 and 1 0 at the obvious, as we have seen, so we will see. So, it is not the case that, for 0 0 nothing will be propagated, something will be propagated we will see.

Because, by logic if both of them 0 0 then, I mean, this is blocking everything then, this also blocking everything then, nothing should be propagated over here, excepting only a stuck-at 1 fault will be deducted here at the output. But, this is not the case, but both the inputs 0 0 also something gets propagated, which is interesting to see and for 0 1 and 1 0, it is bit straight forward, so with that, we are going to see in the end. So now, this input is 1, so here it is 1 1 0 0, so both the inputs are 0 and the output is 0, normal case.

And you see what happens, so if it is a 0, so obviously, stuck-at input is 1 here sorry, random pattern is input, so stuck-at 0 fault is detected. So, this thing gets propagated here, this thing gets propagated here, as well as it is 0, it is 1 1 normal value. So, stuck-at 0 fault here and stuck-at 0 fault at these nets are detected, so this is in this list, so this step 1 is fault this one.

(Refer Slide Time 55:30)



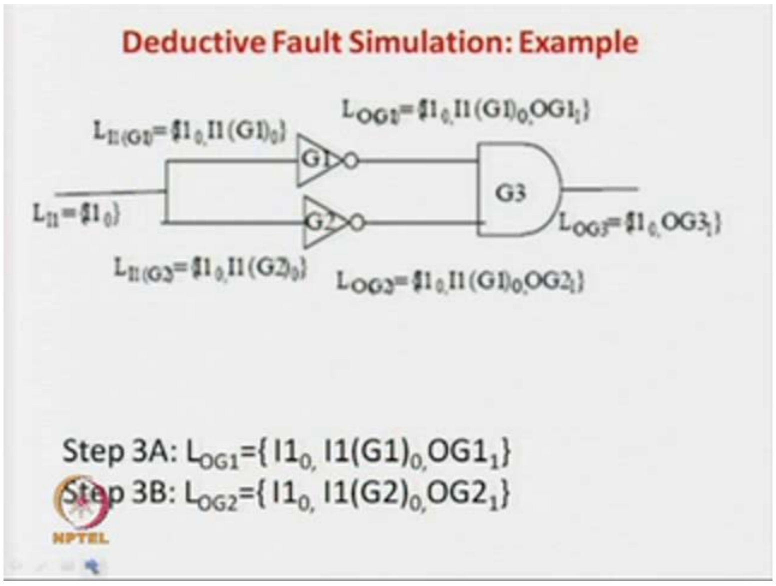
Deductive Fault Simulation: Example

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

Step-2: $L_{I1(G1)}=\{I1_0, I1(G1)_0\}$.
$L_{I1(G2)}=\{I1_0, I1(G2)_0\}$.

Now, step 2 says that, these are the two values at these two fan-out nets, which is very simple, because you are applying a what you called sorry you applying a 1 over here. So, stuck-at 0 fault is detected here, this is again 1 1, so again stuck-at 0 fault at this net and the stuck-at 0 fault at this nets are detected. So, this step is level 2 by the, what you called this fault deduction rule.

(Refer Slide Time 55:50)



Deductive Fault Simulation: Example

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

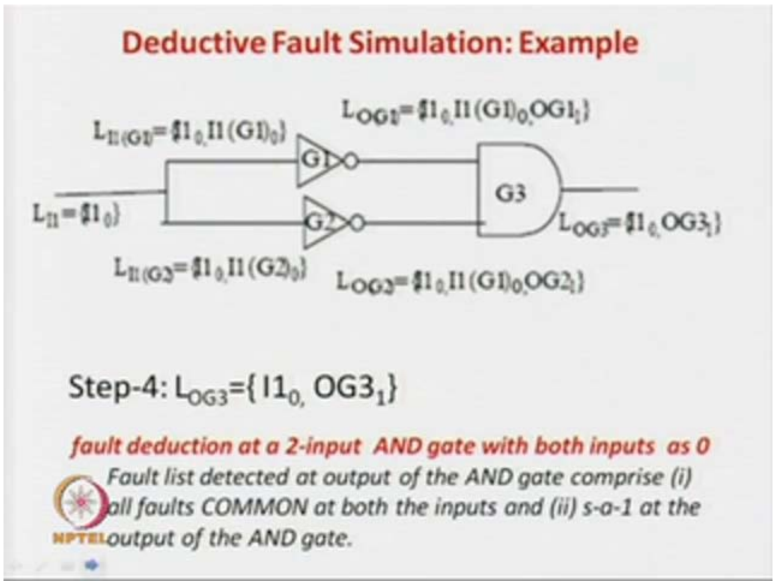
Step 3A: $L_{OG1}=\{I1_{0,}\ I1(G1)_{0,} OG1_1\}$

Step 3B: $L_{OG2}=\{I1_{0,}\ I1(G2)_{0,} OG2_1\}$

Now, on the inversion, so the answer it is 1, it is 0 sorry 1, so it is 1, so it is 1 1, now it is 0 0, so it means, 0 0 means and it is a inverter, so the whole list will be propagated here, the whole list will be propagated here as well as stuck-at 1 fault here will be detected as well as stuck-at 1 fault will be detected. So, this is one stuck-at fault at this net and at this one net, so this is the thing, which we are going to get, these are this is the things at the output of the inverter.

So, deductive fault simulation rule of the inverter already we know, is what, it is a whole list at the input along with that, if it is a 0 then, stuck-at 1 fault at this net and the other and vice versa, so this is the list over here.

(Refer Slide Time 56:38)



Deductive Fault Simulation: Example

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

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

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

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

Now, we have to find out the step we are trying to reach through is what, this is also a 0, this is also a 0. Now, what is the value at this output of the AND gate for faults are detected, so logically, I think we should know that, nothing should be propagated, because everything is blocking, obviously, so this is 0. So, one stuck-at 1 fault; obviously will be detected over here, that is true, so OG 3 1 is there, but how come another easier.

So, something has been also propagated that, you have to that is, I 1 0 is also here, so by our logical thinking that, 0 is blocking, this 0 is blocking, so this whole list should be block, this whole list should be block is correct, should have been there. But, still you find out that, so just let me again see, so what is there, so so both are blocking. So, this one should be block (Refer Slide Time: 57:24), this one should be block, this one should be block, this one should be block, because this one should be block, this one should be block, because this only this block, this only should be block.

Because, this list should be block, because this 1 0, now this list should be block because, this one is 0, so 0 0, the answer is 0. So, at least a stuck-at 1 fault here should be detected, which is correct, but you see one more element is appearing here that is, I 1 0 that is, this stuck-at 0 fault is detectable here. So, that is one thing which has come, which should not have been come logically, but let us see why it has come then, we can formulate the rule for AND gate with when both the inputs are 0 0.

So, we are applying a 1 over here, so this is a stuck-at 0 fault is detected, so 1 0 1 0 this is the case, this is 0 1 0 1, now you will see what happens, so normal case 0 0, fault case this is 1 1. So now, you can see, for this fault we are starting, so the answer is 0 1 and this fault is detected. So, one important interesting thing is happening is that and the fault as it is propagated, so even if the value is 0 0 under the normal case, but still is stuck 0 fault here is detected.

Now, one important thing you have to observe that, 0 is blocking, this 0 at this input and 0 is input at this one is totally blocking the flow here and obviously, this these things are not propagated and these things are not propagated, that is true. But, only these things has been propagated, this is because, this fault this fault this stuck-at 0 fault is effecting both in this and it is affecting both the inputs in one rule, that is why it is propagating the output and it is getting detected.

So, it is true that, this input 0 here, 0 here that is, we have a that is for this case of random pattern 1 that is, 0 0 1 sorry random pattern 1, so it is 1 random pattern 1, this is the random pattern 1, this 0 0, flow is stopped. So, these things are not propagated, these things are not propagated, but only these things are getting propagated, because this fault is affecting both the lines together.

So, the default deduction rule for a two input AND gate, when both the inputs are 0, is not that, nothing is propagated and only a stuck-at 1 fault at the output, it is not the case, only one more thing will be there, all faults common to both the inputs. Because, if there is this is 0, so one fault which is I mean, a propagated which is, so it is 0, normal case it is 0, so fault case it will be converted to 1. Now, if is another fault, which the same fault, which can also convert this one from 1 to 1 0 to 1, so the answer at the output will be 0 slash 1, so in that case, the fault will be detected.

So, let me just put it in the other way, before we a close for today, so that is, this is 0 0 in the normal case, so nothing is, so no fault in this net will be propagated and no fault in this net will be propagated, because this is the 0, the answer is always 0. But, let us think about fault, which can convert this to 1 and the same fault also convert this to 1, so the output will be 0 oblique 1 and the fault can be detected. So, this fault is actually doing the same thing, so this is actually affecting both the faults, both the both the inputs of the and gate together and the fault is getting detected, so which is the rules for this one.

So, we have seen the deductive fault simulation rule for the AND gate under two conditions, if both are 1 1 then obviously, the whole fault list from this input and this input should be out as well as we will be here, as well as a stuck-at 0 fault will be taken into picture. And for the both the inputs 0 0, it is very interesting, it is not that nothing will be propagated and only one stuck-at 1 fault will be there. But, if there is some fault, which is effecting both the lines in common then, that fault will be propagated here, all other things will it works and one stuck-at 1 fault will be taken.

So, there are two other combinations of a AND gate 0 1 and 1 0, which will see tomorrow lecture. And then, what also we will do, we will make table kind of a thing, that is will give you the rules for deductive fault simulation for AND gate, OR gate, NOT gate, fan-out and all. So that, whenever you have a circuit, you can apply the rules and you can go for a fault simulation by deductive mechanism. And in one go, we can find out, which are the faults which is detectable by a given random pattern in one shot. Following that, we will see the another fault simulation algorithm in the next class, which will be finishing our discussion.

Thank you.