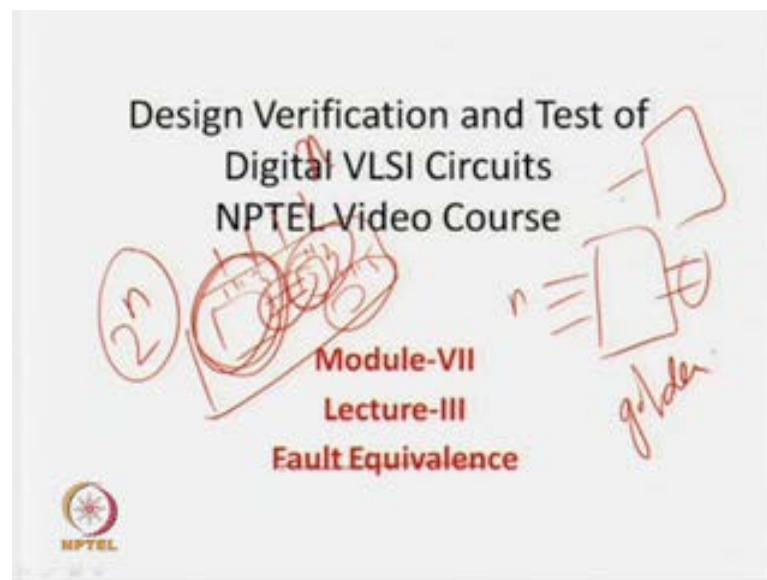


Design Verification and test of Digital VLSI Designs
Dr. Santosh Biswas
Dr. Jatindra Kumar Deka
Indian Institute of Technology, Guwahati

Module - 7
Lecture - 3
Fault Equivalence

Welcome to the third lecture on the testing part on Fault Equivalence. So, in the last lecture, what we had discussed was that for testing a circuit being functional or structural.

(Refer Slide Time: 00:32)

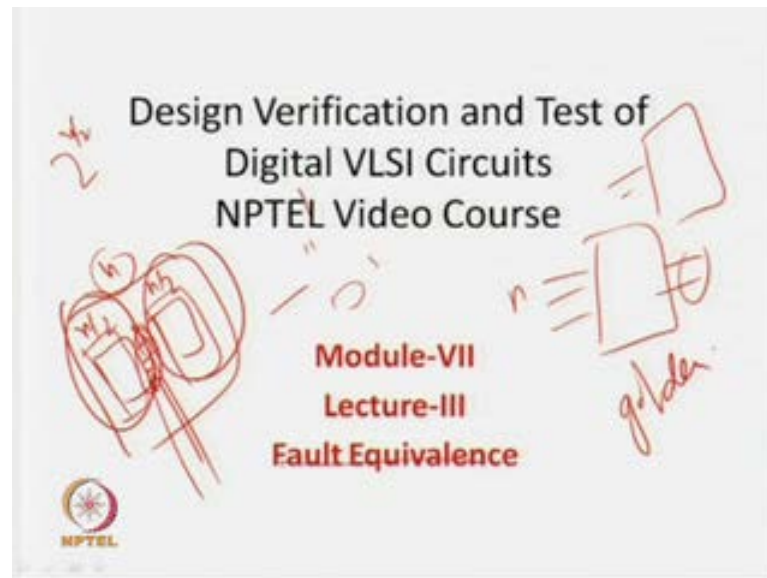


The idea is that we have given a circuit, and then we if there are n inputs kind of thing, so we have to apply some test patterns and we have to find out where there this output response matches with the golden response. Then our idea was that we have seen that if you go for a full functional testing then the odd number of inputs was 2 to the power n , which was quite high, and is almost impossible to apply to the high expensive test equipments, if you have to apply around 2 to the power n patterns.

Then we said that we can go for some kind of structural testing, so in structural testing whatever the idea was, that the whole big bigger circuit of say, which are n inputs. You break it up into small small modules, which I will say some 3 inputs, these will have 2 inputs and these are 5 inputs or some some number of inputs. And then you test this

functionally, you test this one functionally, you test this bug individually, functionally. And that you are about the functional test at the structural level, that is you are testing this guy or this smaller module functionally, but you are not checking whether the interface is proper or not. So, this is kind of a structural testing at the module level, with the module is a small blocks of circuit like this.

(Refer Slide Time: 01:44)



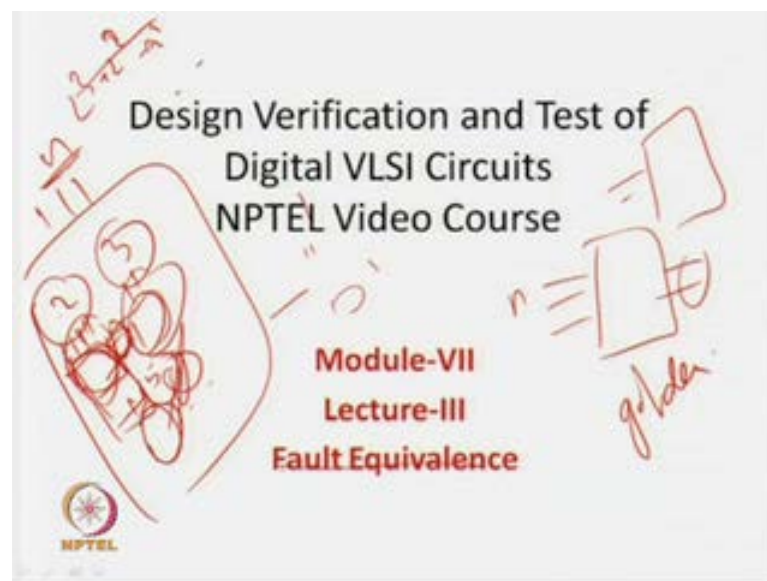
Now, the rather the question also we are asked in the end of the last lecture that, if this modules are quite large in number, then what is the idea? If this modules are quite large in number then, less number of interconnects has to be checked like, for example if you consider the case, where these are whole block, which have some $n \times n \times n$ inputs, then you break it up into say, two modules which is $n \times 2$ and $n \times 2$.

So, then less number of interconnects has to be test means, has to be verified or idea is that, you have to test this one individually, you have to test this functionally individually and then, the structural part is this block itself. Then, what happens, this less number of interconnects in there and also in the last lecture, we have seen, what do you call this in structural testing, if this interval for this inter module intermediate modules, you have to either control or you have to absorb.

So, if you have to control then, you have to put some 2 to 1 multiplex or you have to put a shift register kind of a thing. Similarly, if you have to absorb the intermediate lines then, you have to bring this peen out or you should have a shift register to absorb there.

So, if this number of intermediate lines are less so or the number of extra circuit is less but, if the intermediate lines are small that means, this modules intermediate modules, which are you are testing at structural level, the modules are large in size. And the and the ratio in which, n decreases may not be large, I mean that enough right for example, you can have n by 2 and n by 2. So, 2 to the power n by 2 is not a small number but, on the other hand, if you make these elements or what do you call this, the elements you want to test structurally are quite small.

(Refer Slide Time: 03:06)



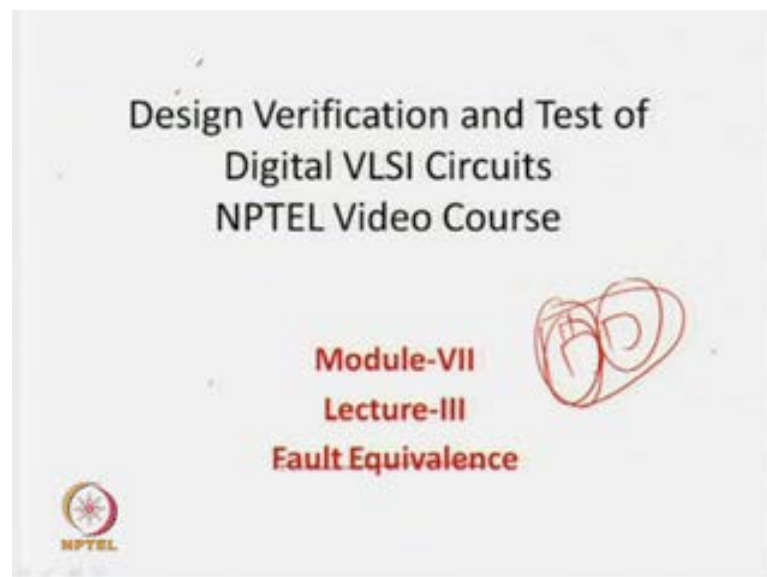
So, you have some 2 inputs, here you have 3 inputs and so on, some 5 inputs dot dot dot so, the whole n inputs here breaking up into small, small modules. And this one you are testing functionally, this one you are testing functionally and so forth so, now, lot of intermediate wires are move. So, you have to put more number of shift registers or more number of 2 is to 1 muses and peen outs to have controllability and absorbability intermediate lines.

But now, the what do you call this, modules or what do you call this, structural elements or structural box are now, having less number of pins. So, it is actually 2 to the power 3 plus 2 to the power 3 2 sorry plus 2 to the power 5 dot dot dot. So, the number of test patterns that need to be applied has quite less, it is 2 to the power 3 plus 2 to the sorry 2 to the power 2 plus 2 to the power 3 plus 2 to the power 5 and so on.

So, number of test patterns are smaller, because n you have broken down into small small numbers but, now the large number of intermediate wires are coming into picture. So, that is what, is the answer to the first question of the yesterday's I mean, last lecture in which, we have ask the question that, if the in this modules are quite large in number quite large in size or largely the more number of peen outs then, what is the problem. So, the problem is that, you have the n does not reduce in that amount so, you have to apply a large number of test patterns.

But, the interconnections were less so, less amount of what do you call this, 2 is to 1 multiplex or registers has to be applied. On the other hand, if your modules or on the or these modules are on which you are going to test functionally or which are the blocks of your structural testing are, I having very less number of inputs. Then, in the number of test patterns to be applied is less but, on the other hand, you have a large number of interconnects in between them again. So, the now, it is more number of 2 is to 1 multiplex are peen outs or shift registers are equal. So, these is a trade off, if you are doing this without a fault model but now, the next question was that, if then, what is fault model.

(Refer Slide Time: 05:01)

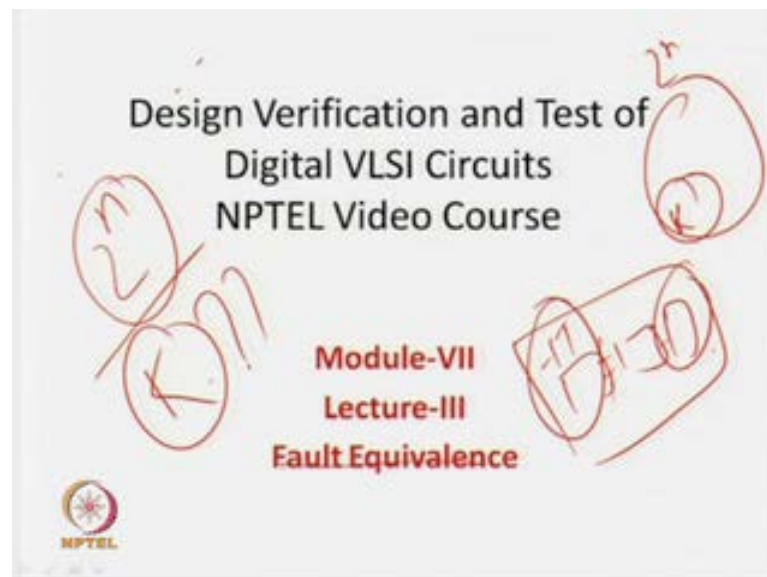


In case of fault model so, these are circuit and you have smaller modules in between them then, we do not test this one structurally. This is actually structural test with a with fault model, the previous one we discuss was the structural test without a fault model.

So, in fault model what we do, we just say that, we do not want to test this functionality of this, whether we want to find out, that the circuit should not have any fault from a fault risk.

So, we have seen lot of fault models like stuck-at fault, bridging fault, delay faults etcetera but, among them this stuck-at fault is the most widely accepted one. Because. it is simple to handle as well as it can give you an accuracy or with you call confidence of 99.9 percent plus. That, if you are doing structural testing at stuck-at fault then, the circuit is not having any defect, with this as accurate of 99.9 percent.

(Refer Slide Time: 05:47)



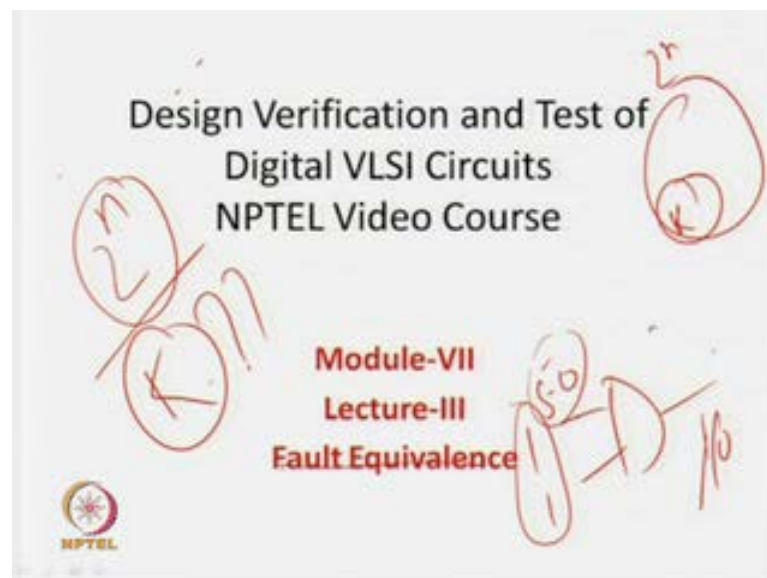
So, in this case what happen, even if you are in this case, if you are having a structural test with the functional structural test structural testing with fault model, if you have a bigger circuit like this then, you are breaking up into smaller modules then, we can have interconnects etcetera. Then, we are not bothered our functional testing of this block or functional testing of this block.

We are rather bothered that, this may should not have any stuck-at fault, this may should not have any stuck-at fault similar, this may should not have any stuck-at fault and this may have should not have any stuck-at fault and so on, as per the stuck-at fault model. And then, we have also shown that, for this it is such a good start that, you need not have any peen outs extra, neither you need to have any 2 is to 1 multiplex or neither you are

need to have any, what you called this shift register extra, if you are going to do a structural testing and using a fault model.

So, that is why the second question was, does really any stuck-at fault happen, the answer is no, really is not, stuck-at fault really may not happen in a circuit. But, what actually happens is, stuck-at fault is that, from 2 to the power n that is, all possible testing factors you can apply. We need to apply say, some k test patterns, which is feasible within the time. Now, the question is, which k test patterns you have to apply among this 2 to the power n, 2 to the power n is a larger set. So, full super set and then among them, we have to apply only some k amount of test patterns now, which k. Actually, stuck-at fault tells you which k to apply.

(Refer Slide Time: 07:06)

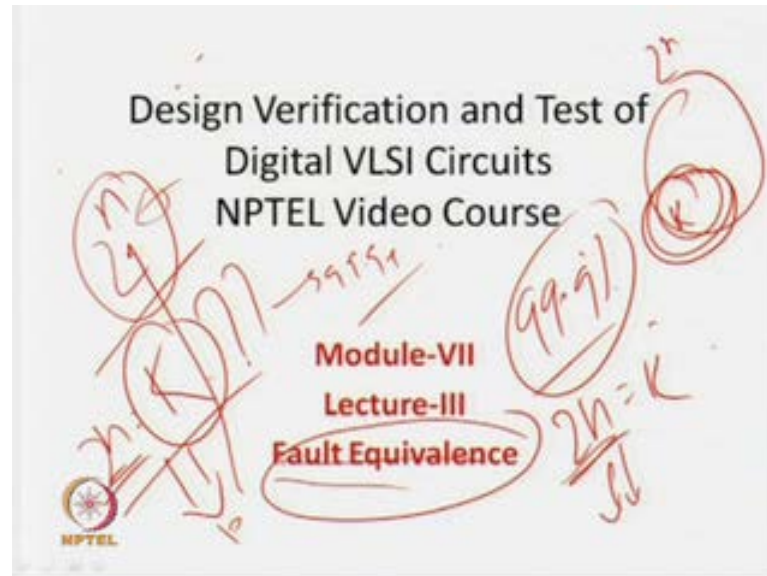


It will say that, apply stuck-at fault like for example, we have say that, if you are given an AND gate something like this. And then, you say that, stuck-at fault here 2 input AND gate. So, it say that stuck-at fault here means, the 2 input AND gates and a stuck-at fault, stuck-at-0 fault here, you want to test it. So, it says that, you will apply 1 1 and you verify the output to be 1.

If their answer is 1, it is correct else this stuck-at-0 fault is there. So, this 1 1 to trace the stuck-at-0 fault here, your applying a 1 1 so that means, this stuck-at fault is implying that, you apply this pattern. So, is stuck-at fault implies one test pattern. So, if we have n

n number of wires then, we have seen that utmost you can have 2^n number of stuck-at faults.

(Refer Slide Time: 07:43)



That means, 2^n number of patterns where, 2^n is equal to k , number of patterns. So, stuck-at fault is giving you a very good subset of 2^n , which if you apply then, you are going to say that, we are 99.9 percent sure that, your circuit do not have a defect. So, stuck-at fault as such do not appear but, they give you a guideline regarding, which subset of test patterns to apply that, can give you a very high confidence.

So, what will see to in today's lecture, in today's lecture we will see that, if there are n lines in this for in a circuit then, you can assume that, we have 2^n stuck-at faults that is, one stuck-at-0 and one stuck-at-1. But, today we will see that in fact, that is much much less than 2^n because, some faults are equivalent that is all that means, if you apply all test pattern more than one stuck-at faults will be tested.

That means, we will require really in really the number of test patterns equal to test stuck-at fault is much, much less than even 2^n . So, 2^n is extreme, fully functional test then, 2^n that is equivalent to test equivalent to your functional test sorry structural test with fault modelling comes to 2^n . 2^n means, once one line

stuck-0 stuck-2 so, 2. But then, will see today using for fault equivalence that, this 2^n further comes down to a much much lower number.

So, then, what happens is that, with the very low number of test pattern say, p in number we can have confidence of 2^n test factors with 99.9 percent plus accuracy. So, we will see again the beauty of stuck-at model that, not only 2^n , it is much less than 2^n test patterns has to be applied so, today we will see, how it is possible.

(Refer Slide Time: 09:19)

The slide is titled "Introduction" and contains the following text:

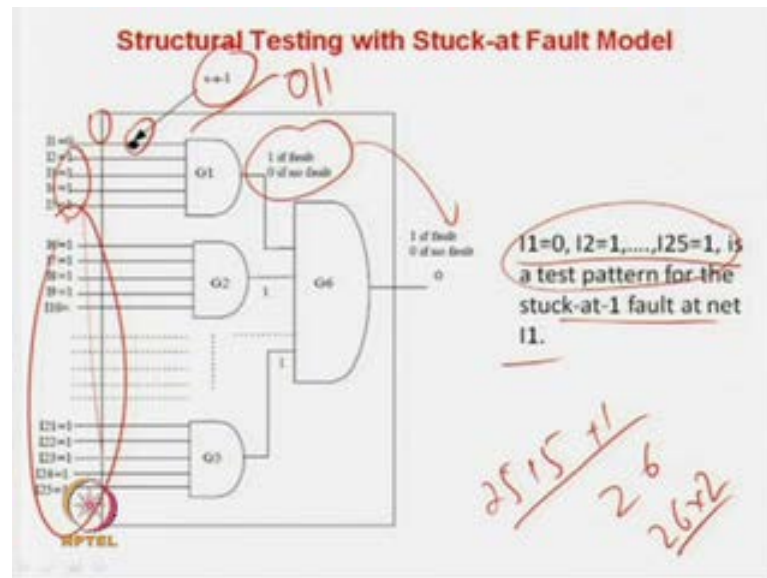
- If there are n nets in a circuit then there can be 2^n stuck-at faults
- The number of test patterns is liner in the number of nets in a circuit.
- The total number of test patterns required 2^n ?

Below the text is a circuit diagram of an AND gate. The top input is labeled "s-a-0" and "I1 = 1". The bottom input is labeled "s-a-0" and "I2 = 1". The output is labeled "s-a-0" and "0". There are handwritten red annotations: a circle around "2^n" in the first bullet point, a circle around "2^n" in the third bullet point, and arrows pointing from these circles to the output of the AND gate. Next to the output, it says "0 if fault" and "1 if no fault". The NPTEL logo is visible in the bottom left corner.

So, whether i told you so, if a circuit has 2^n bits, I mean 2^n nets so, we know that, even circuits has n nets. So, there can be 2^n stuck-at faults because, stuck-at-0 and stuck-at-1 so, it is linear. So, the question arises if you assume that, there is a what do you called, one test pattern is equal for each fault. So, number of test pattern equal to 2^n , which is much, much less than 2^n , that is fine. So, we are quite happy with it but, now we will see that, it is even much less than two even less than 2^n .

That is, number of stuck-at faults will be much much less than 2^n or in other words, the number of test pattern request to test all these 2^n faults are much much less than 2^n . That means, what one pattern will test multiple stuck-at faults, multiple stuck-at faults will be tested by a single pattern that, we will see, which is can be possible by fault equivalence.

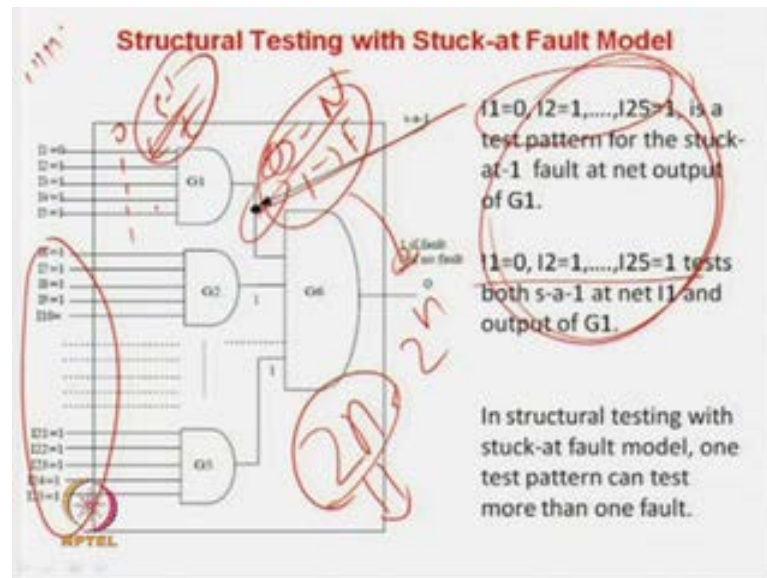
(Refer Slide Time: 10:06)



So, let us go to the yesterday's, our last day's example so, this is a circuit so, let there be a stuck-at fault here, stuck-at-1 fault. So obviously, we have seen, as the pattern to test this stuck-at-1 fault is 0 over here, if we have 0 over here, the fault gets sensitized. So, sensitized means, if the if if the no fault is there this test will be 0, if fault is there the net is one. And to visualize the effect of this point, we have to keep all the lines as 1. So, in the output of the AND gate, you will get 1, if there is fault and 0, if there is no fault.

Similarly, this propagation to the output has to be done so, we have to apply once at all these. So, this test pattern is equal to 0 this one, the test pattern for stuck-at-1 at net, this net so, you can see how many nets are there in this case. So, 25 nets are there and 26 27 29 to 30 So, with this 25 plus 5 plus 1 so, this is 26 nets are there. So, in this case, how many stuck-at fault should be there, it can be $2n$ that is, 26 into 2, what will see that it is much less than that. So, how do you do that, so, we see that this is stuck-at-1 fault here and the pattern to detect this is, this one that is, 0 at this net and 1 in others.

(Refer Slide Time: 11:13)



Now, you take and that case now, you take endless stuck-at fault here. Initially, it is our fault was here so, for that, our test pattern was this one and now again, we are considering another stuck-at fault at this point which is taken as stuck-at-1. Now, let us see what test pattern can be applied, there can be many patterns, which can be applied but, let us see one.

So, in this case, here is stuck-at-1 so, what answer should you should apply 0 over here now, if if the fault is there, this is zero and if it is normal sorry if the fault is there then, these. In the normal case, the answer should be 0 over here and 1 in case of fault. So, somehow you should apply 0 over here, similar this affect has to be delivered with this output. So, all these things should be 1 now, how to make 0 over here.

So, you can apply any one of this 5 lines are there so, accepting all 1's accepting all 1's accepting all 1's, you can apply any other pattern. So, if you apply all 1's over here then, the answer will be 1 and your fault will remain undetected. So, you can apply all 0's, you can apply 1 0 0 0 0 and so on. But, let us apply the pattern, which you applied at earlier so, let us apply the pattern 0 1 1 1 1 now, in this case, you see the answer is 0 over here, it is normal and fault.

So, the same pattern is actually testing stuck-at fault here, stuck-at-1 fault was tested here as well as the same pattern is also testing a stuck-at-1 fault here. So, in other words, this pattern is actually testing 2 faults in a group so that means, if we apply this test pattern

and the output you are getting 0 then, you know that, there are not a stuck-at-1 fault here and not a stuck-at-1 fault here. Hence, this circuit may have a stuck-at-1 fault here and stuck-at-1 fault here, either of there.

Now, it is very important point to ask for us to make, repeatedly i am saying, we have to keep it mind and we are not going for diagnostics, because we are testing millions of circuits in a run and if you find that, this circuit is not working fine, you through the circuit. And later on, you can apply to find out slowly that is that is not much of a constrain.

Because, you are then you you may be wanted to know, what when wrong, why there is a fault over here, why there is a fault over here, that is a diagnostic procedure, which we can do slowly on a very few number of sample chips, to find out well trained run. But, for mass scale testing, we do not really bother, whether the fault was here or whether the fault was here. We do not bother whether it was here or whether it was here, we just find out this is a fault and we are thrown the chip.

So, one test pattern that is 0 1 1 1 1 and all 1's is testing a stuck-at-1 fault here and a (()) stuck-at-1 fault here. So, we have seen that, one pattern at test, more than one stuck-at faults here so, will see that a much less than 2^n number of test patterns are equal to test all these stuck-at faults. So, stuck-at test faults that is, structural testing with stuck-at fault has giving so many benefits.

The first benefit is that, you need not have any extra pen outs or any extra multiplexes or any extra scan shift registers what they controllability and absorbability. Number of test patterns are required is much much less than 2^n that is, much much less than 2^n and also much much less than 2^n . Because, one test pattern is able to test multiple number of faults, so that is the beauty of stuck-at formula.

And at the same time, the accuracy, which has been followed statistically is about 99.9 percent plus that is, it is saying that, one test pattern that is, if you are doing a full structural stuck-at fault testing and you can be 99.9 percent confident, this circuit does not have any fault. So, that is the greatness of stuck-at fault model, now let us look at the formal definitions of fault equivalences.

(Refer Slide Time: 14:38)

Fault Equivalence For Single Stuck-At Fault Model

- Two stuck-at faults f_1 and f_2 are called equivalent iff the output function represented by the circuit with f_1 is same as the output function represented by the circuit with f_2 . Obviously, equivalent faults have exactly the same set of test patterns.

In the AND gate of Figure 1, all stuck-at-0 faults are equivalent as they transform the circuit to $O=0$ and have $I_1=1, I_2=1$ as the test pattern.

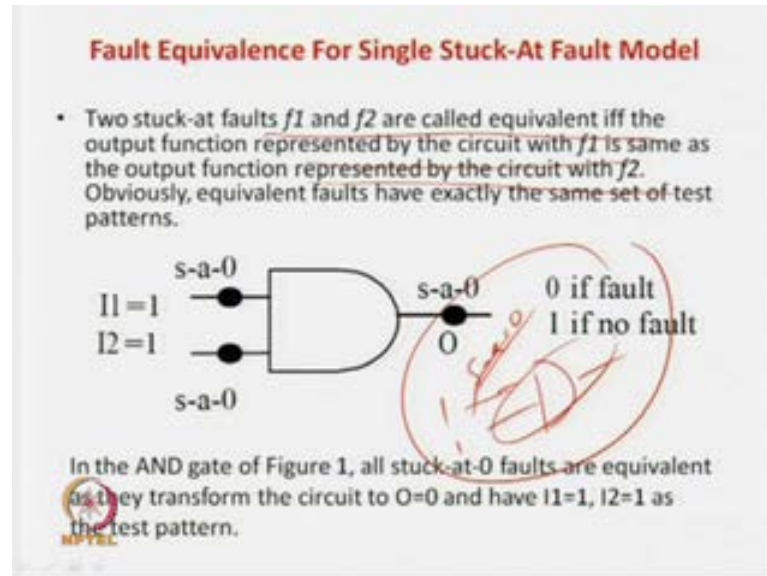
So, what is say, he saying that, if two stuck-at faults f_1 and f_2 are called equivalent, if the output function represented by the circuit f_1 is same as that represent by the f_2 . That means what, if as if you have a circuit say, some circuit x now, if there is stuck-at fault f_1 now, what is the output function, the output function may be something interms of see nets as well as this fault.

Now, if is another position for with the f_2 and if the output function remains same then, this two faults are call equivalent and obviously, under such a case, same test pattern to be applied. Let us take an example of a very simple language so, let us see that if you have a same stuck-at-0 fault over here. So, if you have a stuck-at-0 fault over here so, what is the function, this function at this point, the function at this point is 0. Now, if you have a stuck-at-0 fault over here not here then what is the function, the function is 0.

Now, if you have a stuck-at-0 fault at this point then, what is the function, the function is zero. So, you can say that, in a AND gate stuck-at-0 here, stuck-at-0 here and stuck-at-0 here are equivalent. Now, what is the test pattern to test this fault, it is 1 so, if you have AND gate if you apply a pattern 1 1 then, c is stuck-at faults are getting tested by 1 unit. So, that is what, in this case, their can be AND gate, their can be three lines So, 2 inputs and one this one so, into 2 you require you have 6 faults, 3 stuck-at-0 and 3 stuck-at-1. Now, how many test patterns require to be test, it is 6 now, you can see surprisingly, for stuck-at-0 fault, we require only one test pattern that is 1 1. Because, it is testing stuck-

at-0 here, stuck-at-0 here, stuck-at-0 here and because this stuck-at-0 faults at all the inputs of a AND gate are equivalent.

(Refer Slide Time: 16:18)



Because, the output function f of the AND gate with a stuck-at fault here or a stuck-at fault here or a stuck-at fault here is same and that is called 0. So, therefore, we can represent an AND gate in the single stuck-at model, as simple as, we can keep we can choose any point with a single stuck-at fault. All faults are not required, you can said that, i can have a struck-at-0 fault here then, automatically if you represent the AND gate this one this way with a stuck-at-0 fault that 1 input then, you need not bother about stuck-at-0 fault here, stuck-at-0 fault here.

Because, automatically if you are testing this stuck-at fault, you apply 1 1 and automatically other 2 positions are tested. So, for 2 input AND gate, the number of stuck-at fault that that needs to be tested is 1 so, it is much, much less than $2n$. So, we are always saving in this one now, you can also think about a 10 input AND gates say so, 10 input AND gate. So, number of input output lines are 11, 10 plus 1, 11 so, in number of test patterns number of stuck-at faults possible is 22, number of test pattern is 22. Now, if you see this stuck-at-0 fault.

(Refer Slide Time: 17:05)

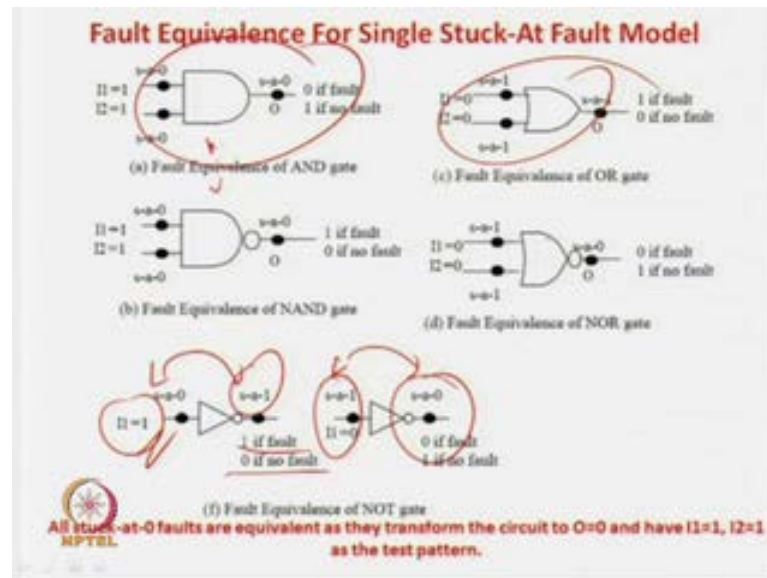
Fault Equivalence For Single Stuck-At Fault Model

- Two stuck-at faults f_1 and f_2 are called equivalent iff the output function represented by the circuit with f_1 is same as the output function represented by the circuit with f_2 . Obviously, equivalent faults have exactly the same set of test patterns.

In the AND gate of Figure 1, all stuck-at-0 faults are equivalent as they transform the circuit to $O=0$ and have $I_1=1, I_2=1$ as the test pattern.

So, if I said that, I consider a stuck-at-0 fault here then, what is the pattern to be applied, all 1's so, a stuck-at-0 fault here, at the first net first input all 1's, test the stuck-at-0 fault here- test the stuck-at-0 fault at the output and also stuck-at-0 fault at all the at all the 9 other inputs. So, for a for a n input AND gate, for even at least 2 inputs or dot dot dot, you can make even a n input AND gate so, number of stuck-at-0 for that needs to be considered is only one. So, if there are more number of input lines in a gate or a circuit, the fault equivalence place a major role. Because, more number of fault equivalence faults can be merged and the number of test patterns or number of fault that needs to be tested, remains much much less than 2^n that is the number of this thing.

(Refer Slide Time: 18:04)



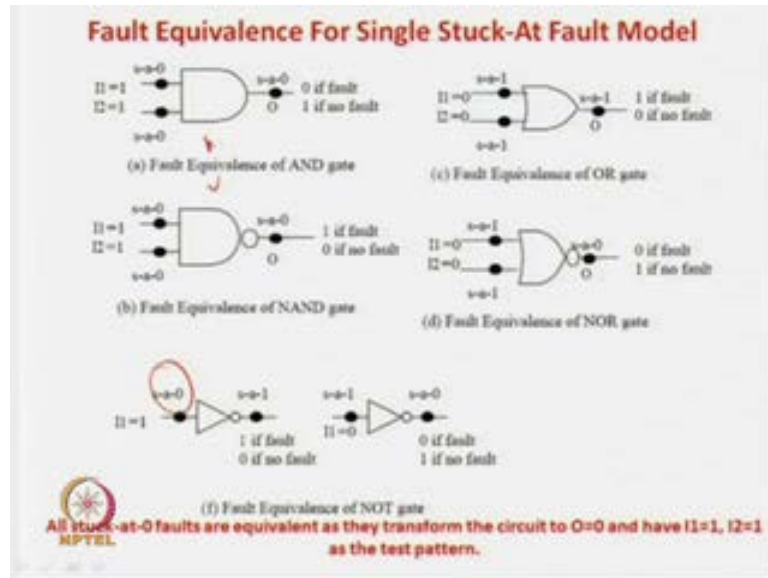
So, we have seen that, this happens for a AND gate now, let us see for other gates so, these are AND gates, you have already seen that, all these stuck-at-0 faults are equivalence. Now, let us see see a OR gate so, if you see a OR gates, this net is stuck-at-1 so, what is the pattern to be applied to test the OR gate, you have to apply a 0 over here. So, if you apply 0 0 if we stuck-at-1 the answer is 1, if there is a fault with the stuck-at fault and the answer is 0, this is know for you. You can very easily observe that, the same pattern is to be applied if these may occur at stuck-at-0 fault and similarly, if these (()) struck at zero fault.

So, for OR gate, this stuck-at-1 all these stuck-at-1 faults are equivalence and you can just keep any one of them, whichever you desire. So, you can see that, AND gate and OR gate are dual of each other. So, in case of AND gate, any one point you can have as a stuck-at-0 fault for but, for a OR gate for any other point, you can get a stuck-at-1 fault so, just a dual of each other. Now, if you go for the NAND gate, it is the always similar to AND gate so, if you have a stuck-at-0 fault here so, you have to apply 1 1.

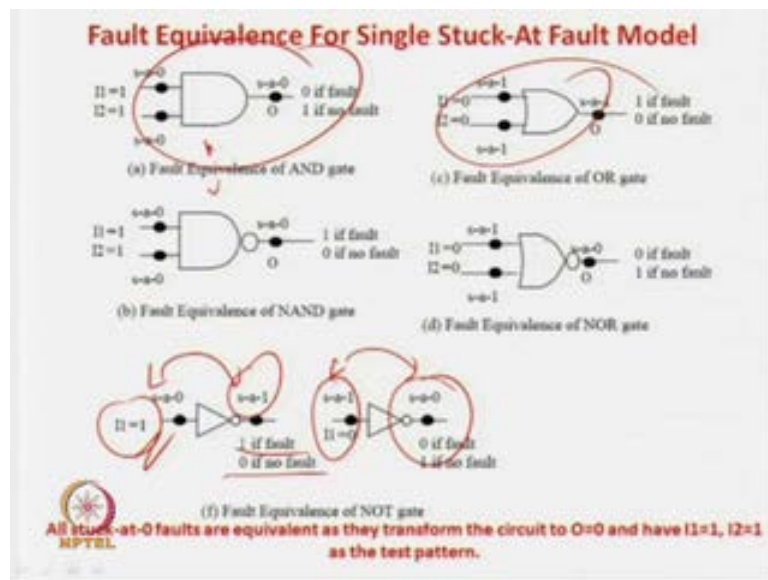
So, if the apart only the output get reversed so, you can see this fault equivalent concept of AND and NAND gate are same. So, you can see that, if this stuck-at-0 fault in a NAND gate, if you apply 1 over 1 then, the answer is 0, if there is no fault. But, the answer is 1 if there stuck-at-0 fault over here, the answer is 1, if there is a fault kind of a

thing. So, I mean, you can easily find out so, those things will we can this things are very quite obvious so, we are not going much into details.

(Refer Slide Time: 19:36)



(Refer Slide Time: 19:42)

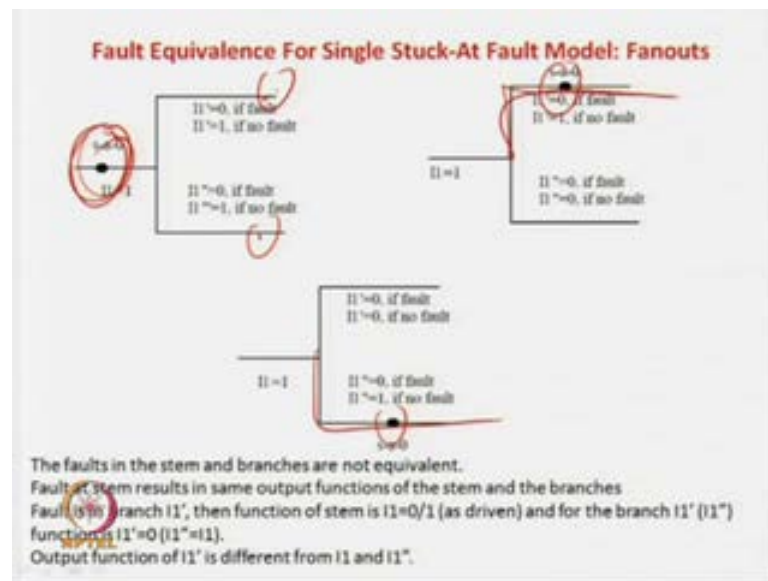


Now, also in the case of, what do you called, this inverter stuck-at-0 fault here. You need to apply 1 and for a stuck-at-1 fault here sorry and these actually, it you need to apply 1 over (()) stuck-at-0 fault for an inverter. And also if this stuck-at-1 fault at the output of a inverter, you have to apply 1 over there because, you can just see, if this stuck-at-0 fault

over here, you apply 1. So, if answer is 1 then, there is a fault and if there is a no fault, if the answer is 0, any one of them, you can consider.

Now, if you consider a stuck-at-0 fault over here then, you apply 1 over here, the answer is 1 if there is a fault then, the answer is 0 if there is no fault. So, for inverter this two step are equivalent similarly, a stuck-at-1 fault here is equivalent to a stuck-at-0 fault here, for the case of a inverter. So, when once you understand the concept of fault equivalence for AND gate and OR gate, other things are very much straight forward.

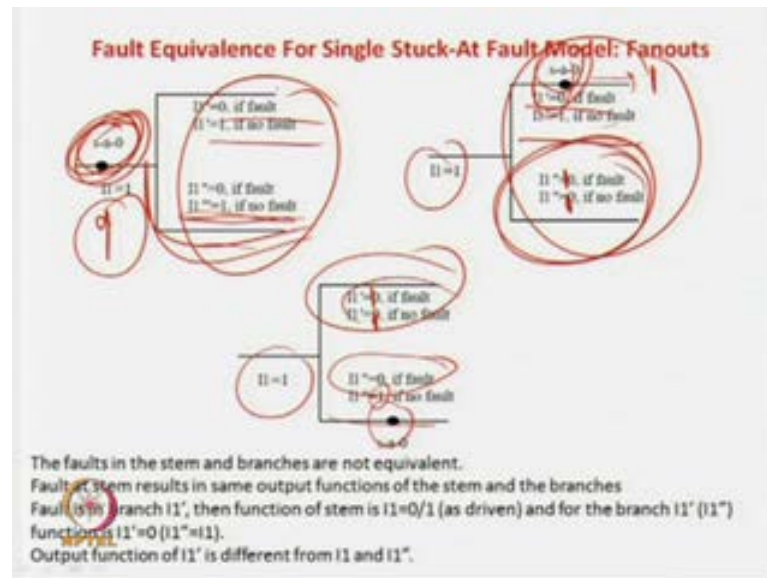
(Refer Slide Time: 20:32)



But, there is a very interesting fact, when you come to what you call the fanouts so, you can see this is a, as i already told you that, in case of fanouts, you have to consider stuck-at-0 fault here individually and the stuck-at-0 fault here individually and the stuck-at-0 fault here individually. Because, statistically we have seen that, if you take one circuit fault over here, you do not take another two circuit for there then, the accuracy was coming down to less than 99.9 percent plus or some numbers.

So, to against it, if you will found out that you have to consider stuck-at fault here, if you will found stuck-at fault here then, these two lines are obviously affected by the fault. But, if a stuck-at-0 fault is here, in only these net is, fanout is affected or what do you called is a affected by this fault and if this stuck-at-0 fault or stuck-at-1 fault here then, only this line is infected. But surprisingly, you see that, these two faults are, these faults are not equivalent, you should have been equivalent but, it is not.

(Refer Slide Time: 21:26)



So, you see why, let us consider stuck-at zero fault here so, if these stuck-at-0 fault here then what happens, in this net i_1 we call it, say it is 0. If the fault is that use stuck and if no fault is there, the answer is 1 obviously, if this stuck-at-0 then, you have apply a 1 to tested. Similarly, for i_1 double pan, this net what happens, if it is a 0 then, if there is a fault because, stuck-at-0. And if the i_1 is the value of 1 then, there is no stuck-at-0 fault here because, you are applying a 1 over here.

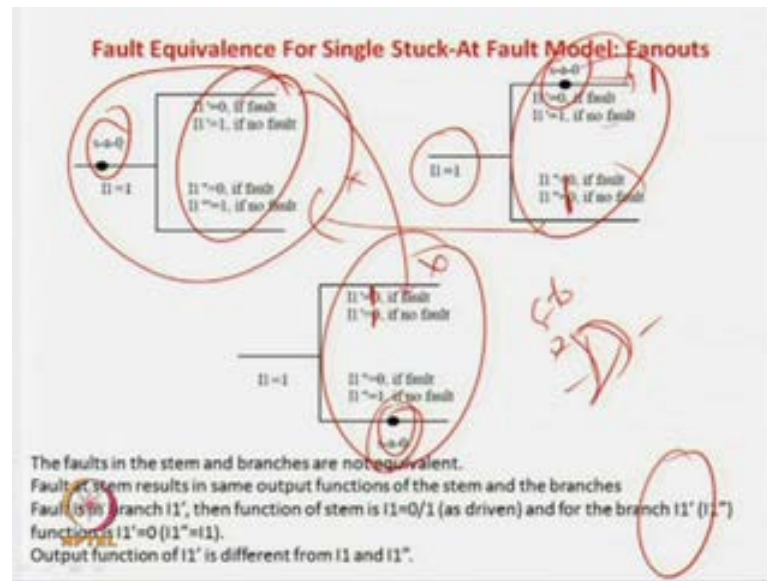
Now you see, if there is a stuck-at-0 fault here then what happens, to a (()) this stuck-at-0 fault, we have to apply 1 to apply 1 over here, you have to get 1 over here, that is very obvious. So, now what happens, you see if I have a 1 over here then, if there is fault is there then, you get the answer is 0. And if it is 1 then, there is no fault is obvious because, I put a 1 over here, I should get a 1 over here because, this stuck-at-0 fault, this is 0.

But, this fault is over here so, this output is not affected by this 1 so, what happens, it says that i_1' equal to 0 is fault and i_1' is equal to sorry it should be stuck-at-0 fault here, it is 1 sorry it is 1 sorry. So, it should be i_1' is equal to 1 if there is fault and the i_1' is 0 there is no fault because, this fault is not affecting this line in any way.

So, now, what happens is that so, the presence of this stuck-at fault so, whatever is the functionality here and with a stuck-at-0 fault here, whatever is the functionality at this

fault are not equivalence. Similarly, let us see so, in this case also, it should be 1 and sorry so, what it says that, if we apply a stuck-at-0 fault here as so, you apply 1 over here. Now, if we if there is a fault so, this will be 0 but if it is no fault so, one will be propagated at, the answer will be 1. But, this net is not affected this fault so, i 1 and this formula always be 1, with fault or without fault.

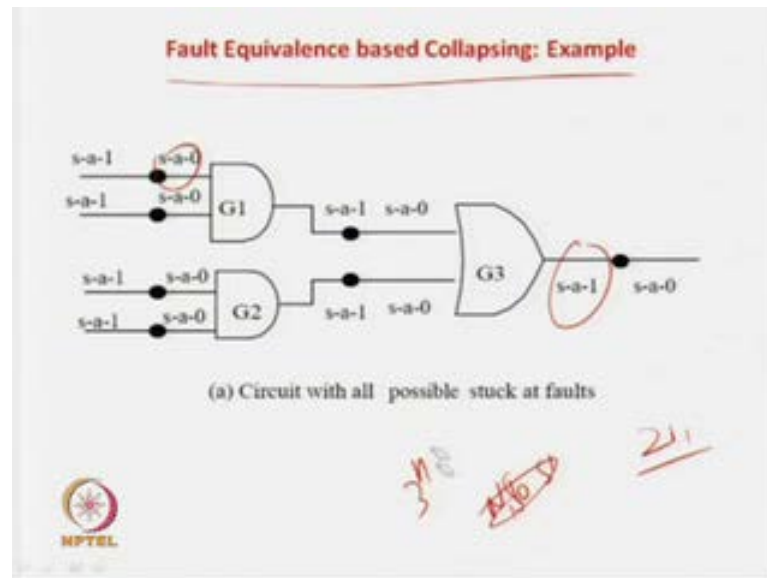
(Refer Slide Time: 23:23)



Now, you can see that, the output function by this, if the fault is here, the output function (()) 1's the output function this is the mistake sorry sorry (()) 1's. So, the output function be the stuck-at-0 fault here (()) sorry and a stuck-at-0 fault here and a stuck-at-0 fault 1 and not equivalent. Because, this one and this one, this one is not matching with this one similarly, this one is not matching with this one so, the output is different. So, even if the fanout is such a nice thing and look similar but, these faults are not equivalent so, you cannot have any merging between this fault.

Like in AND gate, we can easily merge this fault and you keep on only stuck-at-0 fault for NOR gate, you can easily keep a stuck-at-1 fault here. But, for a what do you call a fanout, you cannot do any such kind of margin because, this fault the output function this one is not equivalent to this one, with this output fault and this fault the output function this one is not equivalent to any one of this. So, that cannot be any kind of merging, that is what is can be explained here.

(Refer Slide Time: 24:17)



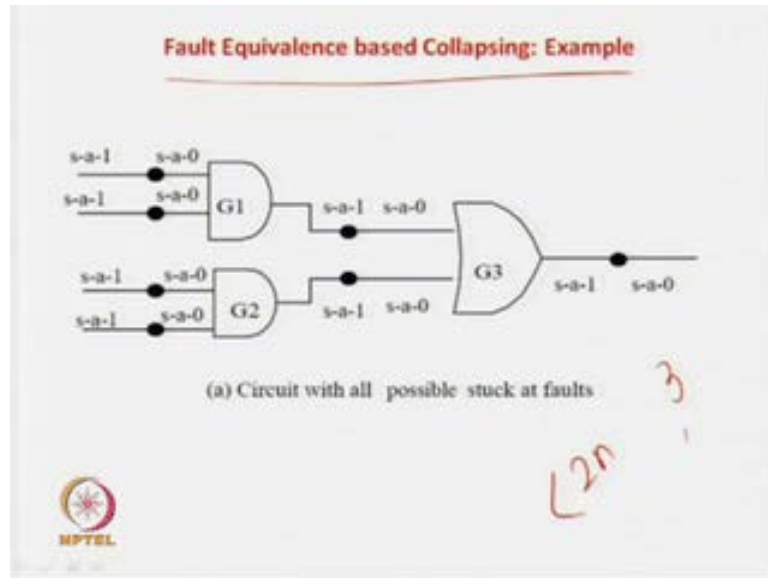
So, a fanout is very surprising, as I already told you so, no merging and all these things can be possible. Now, let us take a simple example and illustrate the concepts so, see we have a circuit having a fault equivalence so, this one. So, these are circuit now, these are AND gate so, first we have to put all the two n faults. So, again I told you, we have to remember that so, stuck-at faults when you are considering for each line, can be having a stuck-at-0 and stuck-at-1 and we are considering one fault at a time.

That means, if there are multiple faults can be also be possible is multiple that is, a stuck-at-0 fault here and a stuck-at-0 fault stuck-at-1 fault here. So, two faults can also occur simultaneously but then, the number of all possible faults will be 3^n , 3 to the power n . If you can see, because, this line can be normal, it can be stuck-at-0 and it can be stuck-at-1. So, in all so, the difficulty will be, we have three possibilities each net and then, the number of all possible test faults will be 3^n and the number of test patterns will be not 2 to the power n but, in the 3 to the power n .

And it is actually much more complex problem to be solved and in fact, we have seen that, this is called multiple stuck-at fault model. That, if you are considering more than one faults in ago so, we have seen that, if you are taking multiple stuck-at-0 fault model then, the confidence is some more 99.99 plus something. And if you are going for a single stuck-at fault model, then the confidence will be 99 point something plus and the

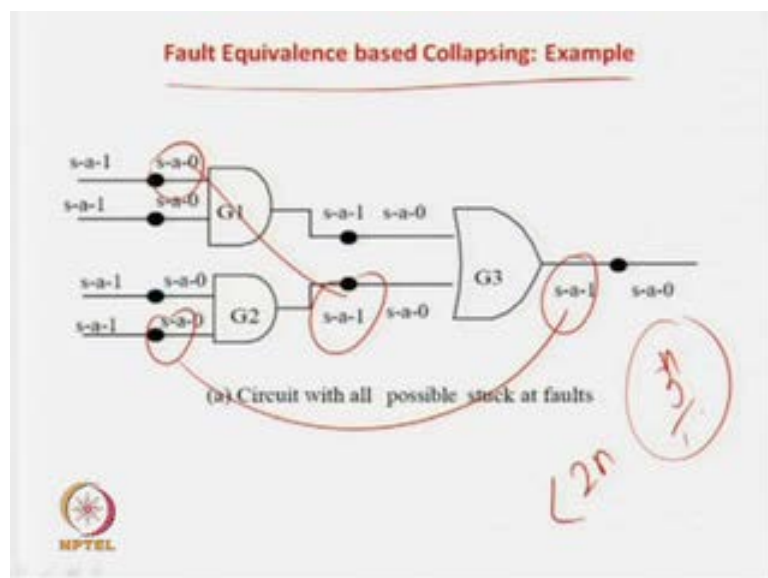
marginal difference is very, very less. That is, the amount of accuracy is very, very less and that does not merit investigation because, the amount of cost to be paid.

(Refer Slide Time: 25:50)



For testing a single stuck-at fault model, the number of pattern is even less than 2^n . In this case, 3^n faults are there and then, if collapsing is not that much.

(Refer Slide Time: 26:01)

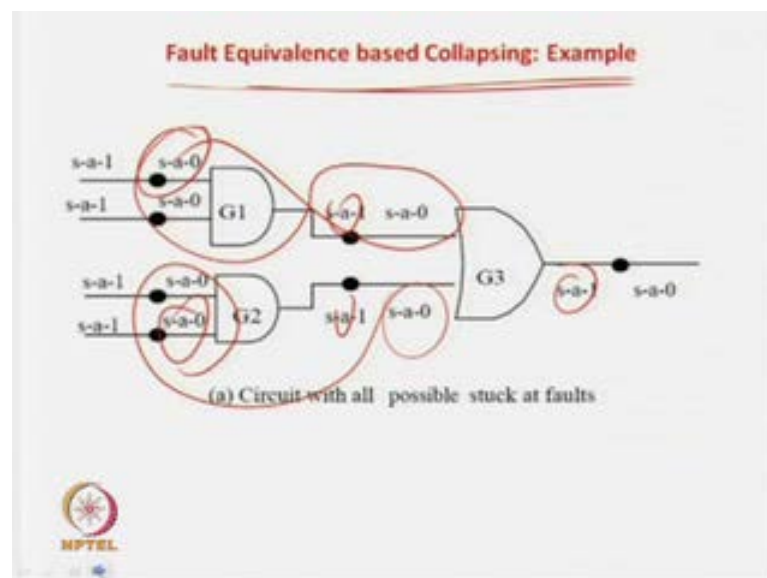


Because you can easily appreciate that, this one and this one combination and this one and this one combination to be equivalent or not. We have to do lot of, the algorithm will be quite complex, in this case algorithm is very simple, for AND gate, we know that

these 3 faults are equivalent because, the same functional area is there. But, if I ask you that whether these two these two combinations and this two combination are equivalent, the algorithm will be very, very complex.

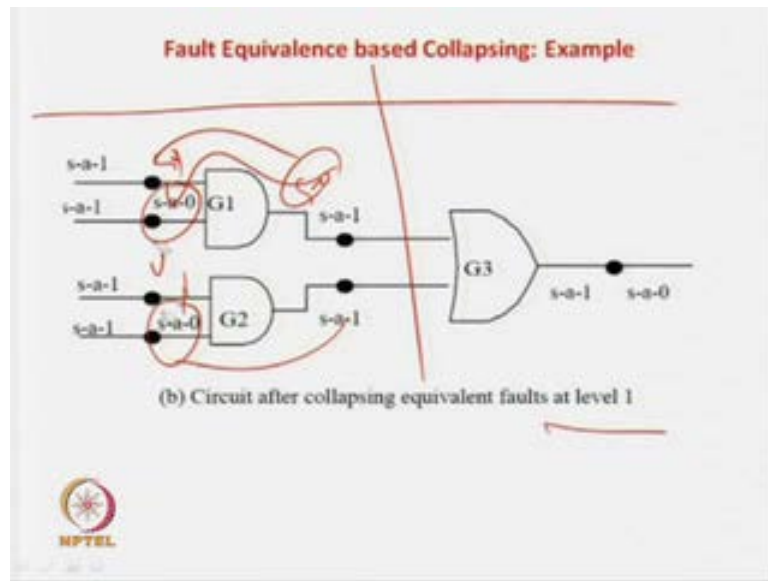
So, the amount of m paid amount of, what we call the cost paid, in terms of computation and the number of test patterns is the order of if they are 3 n faults. And the number of test patterns is some what very high order then the amount of cost paid for improvement of accuracy is not that high. So, therefore, if we found out statistically that, single stuck-at fault model, they are quite happy, so here also we are fault equivalence based collapsing will be doing and also, we are assuming the single stuck-at fault model.

(Refer Slide Time: 26:51)



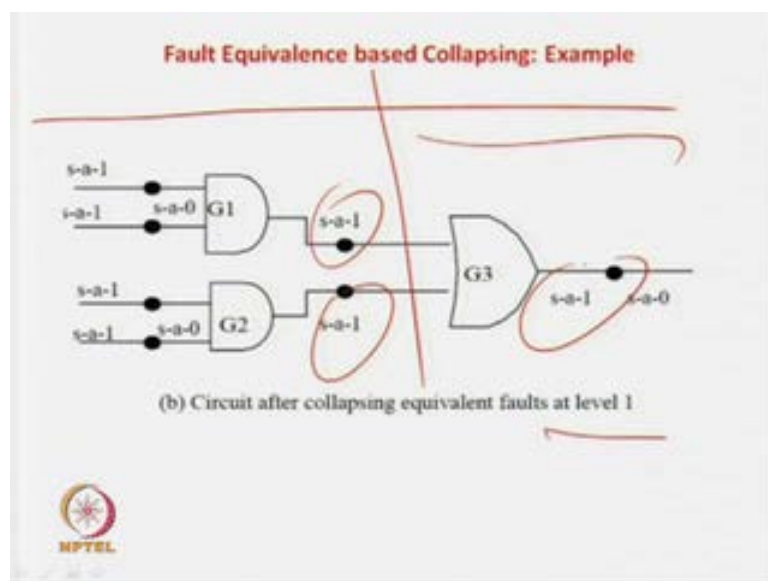
So, let us have this one so, we see that, this is the case with this AND gate so, you know that, any one stuck-at fault you can keep. So, let us try to this (()) we can keep any one of them similarly, for this case also, we can give any one of them and this is an OR gate. So, we have also seen that for OR gate, you can keep either any one of them, in this fault or this fault or this, any one of them, you can keep.

(Refer Slide Time: 27:13)



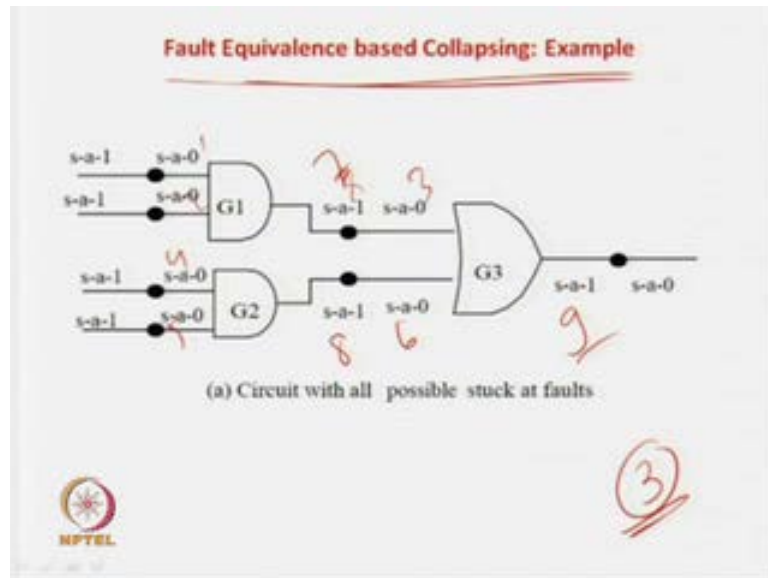
So, let us see what they have done so, we have seen, we have taken this stuck-at fault- we have kept, (Refer Slide Time: 27:14) this stuck-at fault we have kept and this if we have done on the in the level 1. So, we are going level wise so, the 1st level we have done so, in this case, this stuck-at-0 fault, this stuck-at-0 fault or these two are merged into this one these things are all merged here. And in this case, this stuck-at-0 fault and this stuck-at-0 fault are merged over here so, here remaining with only two stuck-at-0 faults as you can see over here.

(Refer Slide Time: 27:42)



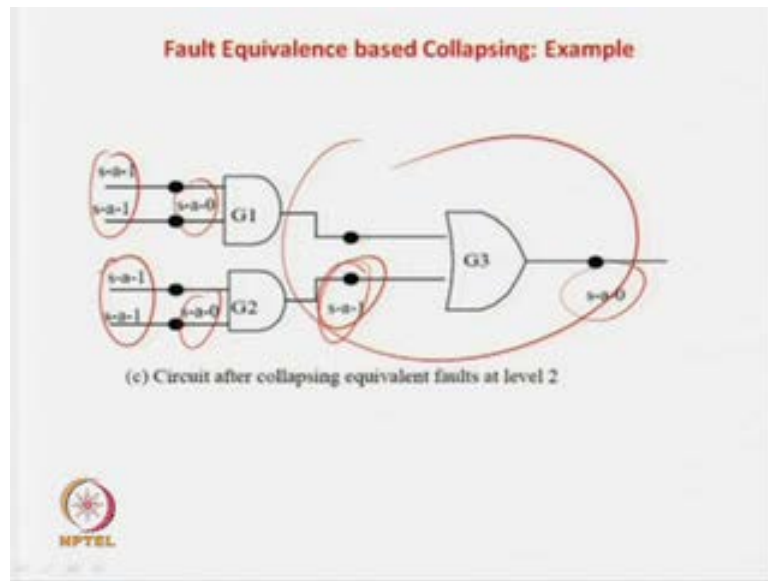
Now, we are going to level 2 so, you can see that. in OR gate, this this stuck-at-1 faults are equivalence, as we already discussed. So, any one of them you can keep so, let us see what they have done. So, we have kept only one stuck-at fault now, initially if you look at it so, we had so many faults. So, how many we have saved in this case.

(Refer Slide Time: 28:03)



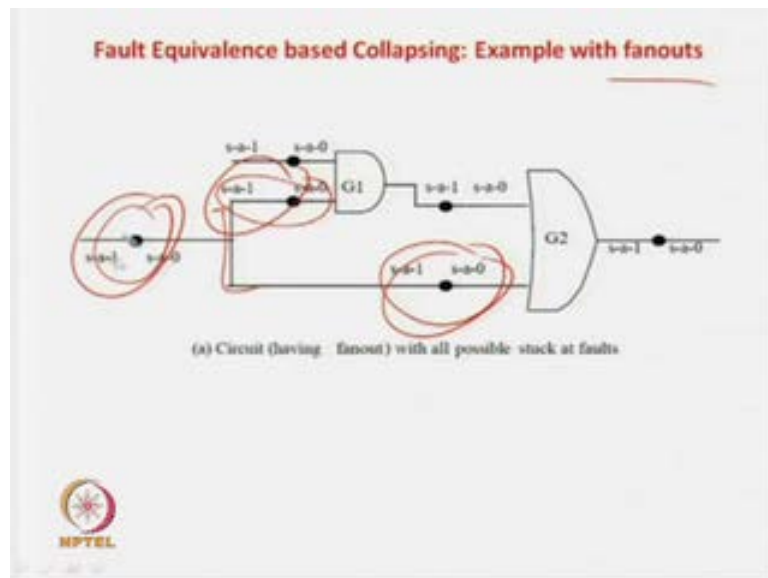
So, out of these three, 1 2 3 sorry this is 4 5 6 and this is say, 7 8 9. So, 9 faults were there, which you can be merging or something so, among these 9 faults, we have kept only three of them. So, it is one third reduction has been possible. So, out of these three we have kept 1, out of (Refer Slide Time: 28:23) these three we have kept 1, 4 5 6 kept 1 and 7 8 9 we have kept 1 so, one third fault reduction is possible by equivalence.

(Refer Slide Time: 28:32)



So, you can see over here so, this is level 2 level and finally, sorry we have only 1 fault here, 2 faults here, 3 faults here and some other faults like a stuck-at-0 fault here. And some stuck-at-1 fault here are still remaining, for which we will see what we can do ever in this.

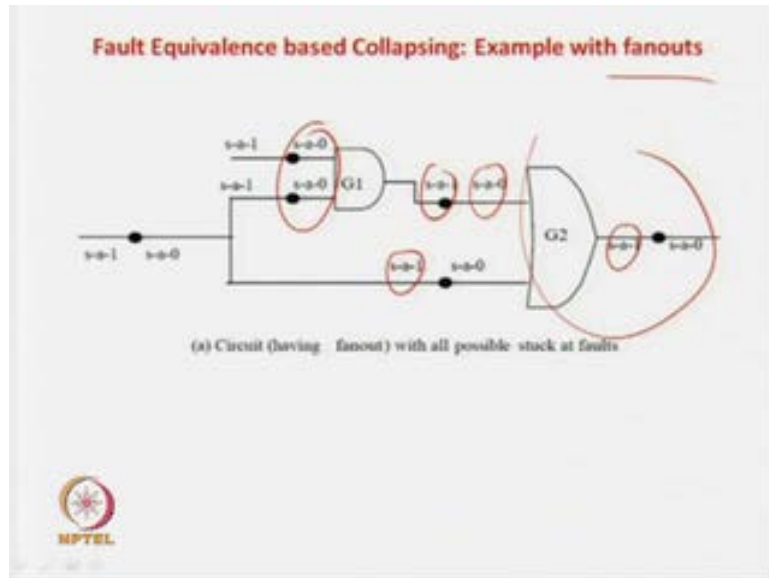
(Refer Slide Time: 28:52)



So, we have one third fault reduction now, let us take another example, when we are going to consider fanouts now, again you see, as i told you a very same thing you will be like the previous example but, now there is a stuck-at-0 1 fault, here is also stuck-at-0 1

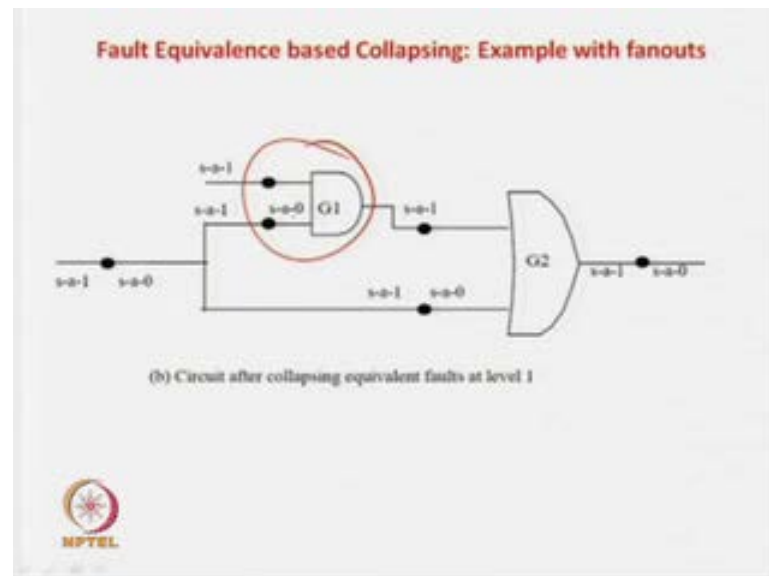
fault here and here also stuck-at-0 1 fault here and as it is a fanout branch. So, this one is independent, (Refer Slide Time: 29:11) this one is independent and this one is independent so, we have this one. So, these are circuit example now, let us investigate, what is possible in this case.

(Refer Slide Time: 29:25)

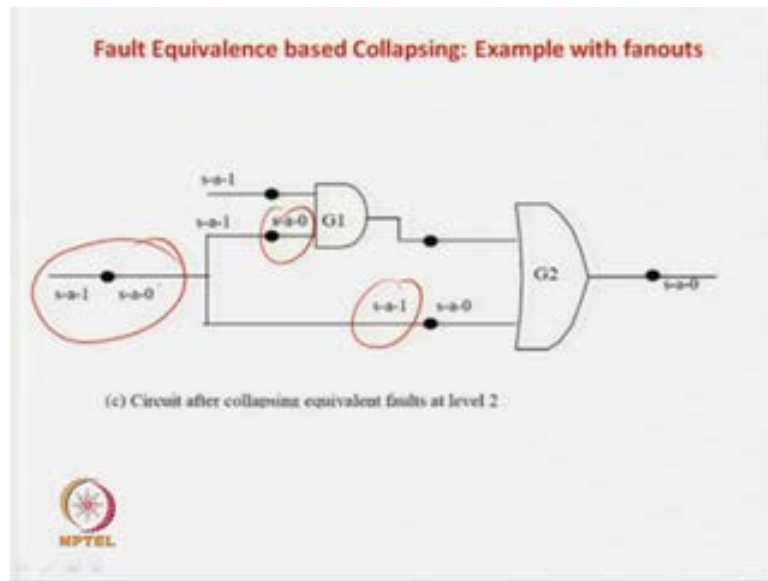


So, in this case, if you see this is a AND gate so, obviously, among them we can keep any one of them. Then, this is a OR gate so, you can keep this one and this one, and this one, you can keep any one of them correct so, this what will be the possible.

(Refer Slide Time: 29:41)

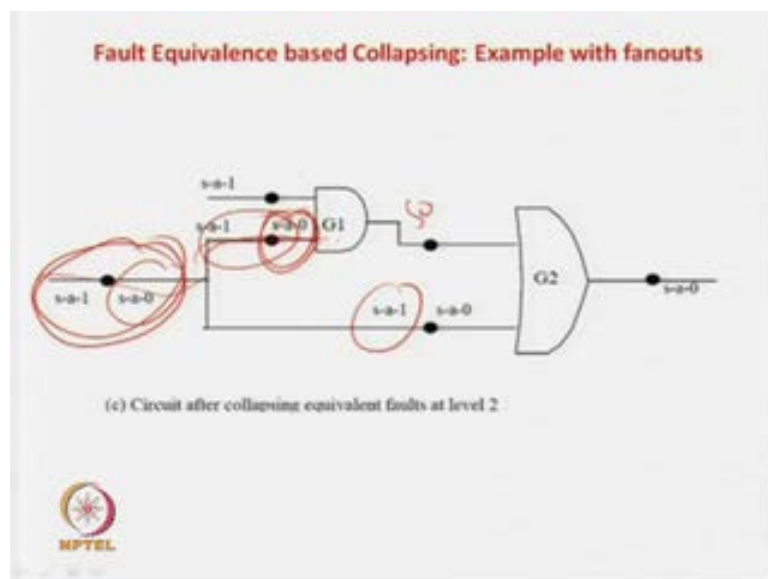


(Refer Slide Time: 29:51)



So, let us see, what we can achieve so, in this case, you see for this AND gate, they have kept only one stuck-at-0 fault, stuck-at-0 fault over here. And these are for level 1 and if you do for level 2 then, again you can keep only a stuck-at-1 fault here for this gate and 1 stuck-at-0 fault. What do you see in this case, we have not been able to eliminate the faults over here because, this net and this net is independent.

(Refer Slide Time: 30:07)

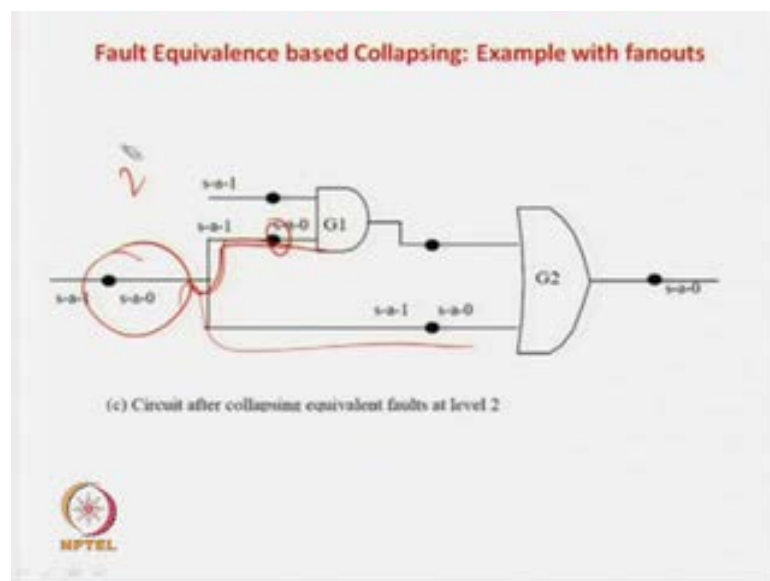


You can think you can say that, this stuck-at-0 fault and this stuck-at-0 fault are almost equivalent kind of a thing. But then, we can could have eliminated because, this is input

to this AND gate and this AND gate, a stuck-at-0 fault here, a stuck-at-0 fault (Refer Slide Time: 30:16) here, a stuck-at-0 fault here and this thing can be merge. But, you know that, this line and this line are independent in fanouts, in this stuck-at-0 fault, this line will be affected like for example, if i tell you, what is the very interesting thing to observe.

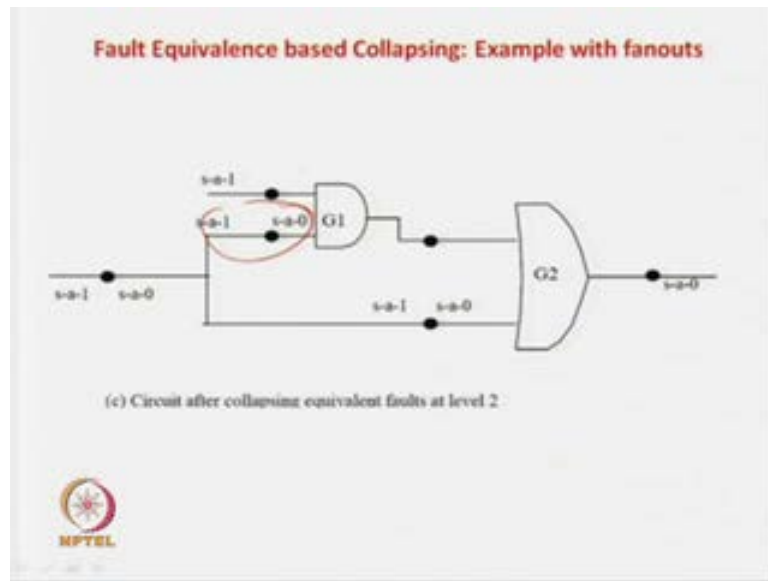
If i merge this stuck-at fault, this stuck-at fault there is an affect with gates, change you say that, if i say that if this stuck-at-0 fault here then, a stuck-at if a stuck-at-0 fault is here then, this line is affected as well as this line is affected. But, if there is only a stuck-at-0 fault here then, only these line is affected so, these two affects and we require to here two test pattern for this. So, these increase in the sometime also require a increase in number of test pattern because, we are going down from 2^n to a much less number. But, sometimes we are going too less in number then, may not be we are going to get the accuracy that was found out.

(Refer Slide Time: 31:04)

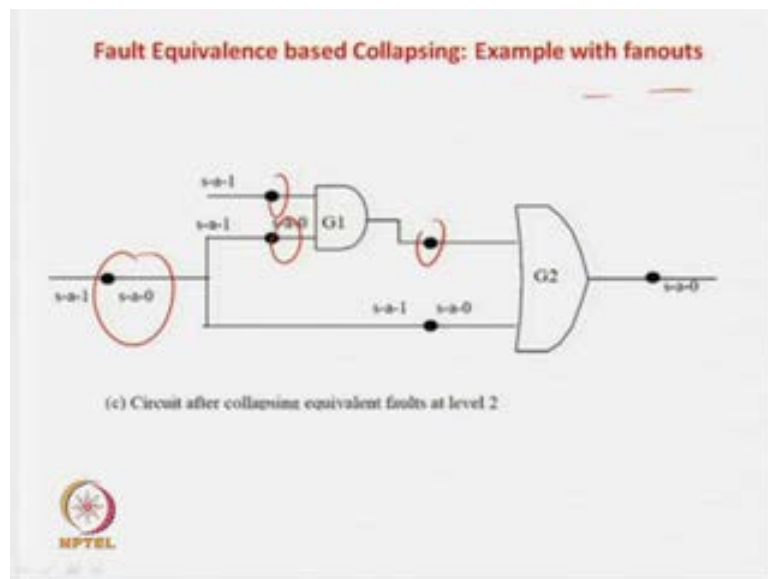


So, if you are keeping one more test fanout, as a independent branch so, we require another test pattern for this line. So, which we can see in (()) So, increase a number of test patterns, we have kept the fanouts lines are independent, which is giving a higher amount of accuracy.

(Refer Slide Time: 31:21)

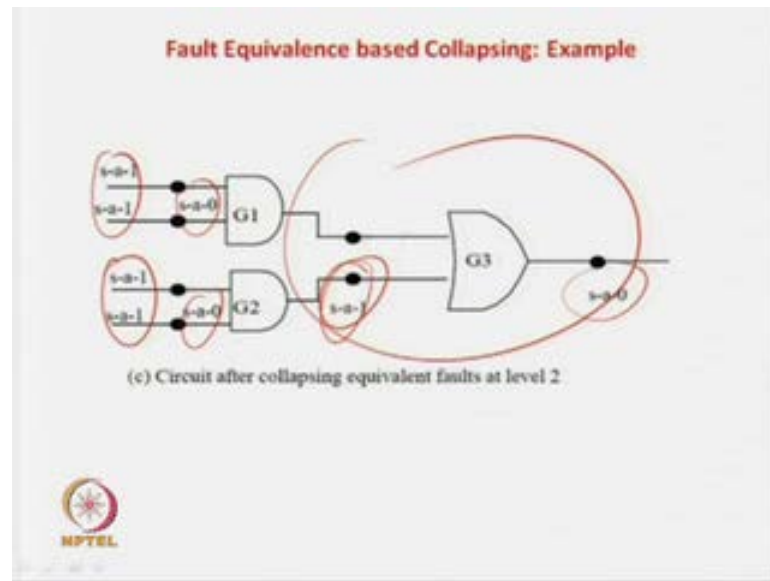


(Refer Slide Time: 31:26)

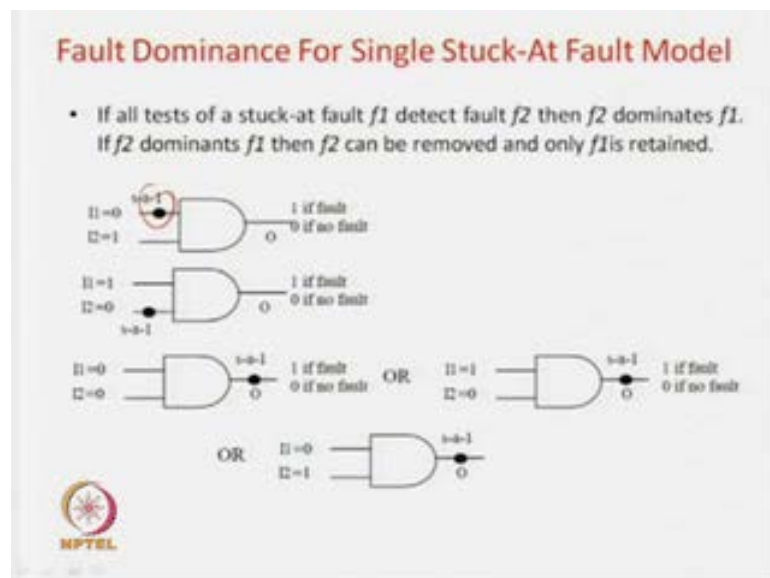


So, but, in the fault collapsing point of view, you can we have to use observe that, we have collapse this sorry we have collapse this stuck-at-0 fault here, (Refer Slide Time: 31:26) here and here. But, we could not do anything with this so, this one is actually, which we are having in the stuck-at fault model with fanouts but, still we have a lot of drop in the number of faults.

(Refer Slide Time: 31:57)

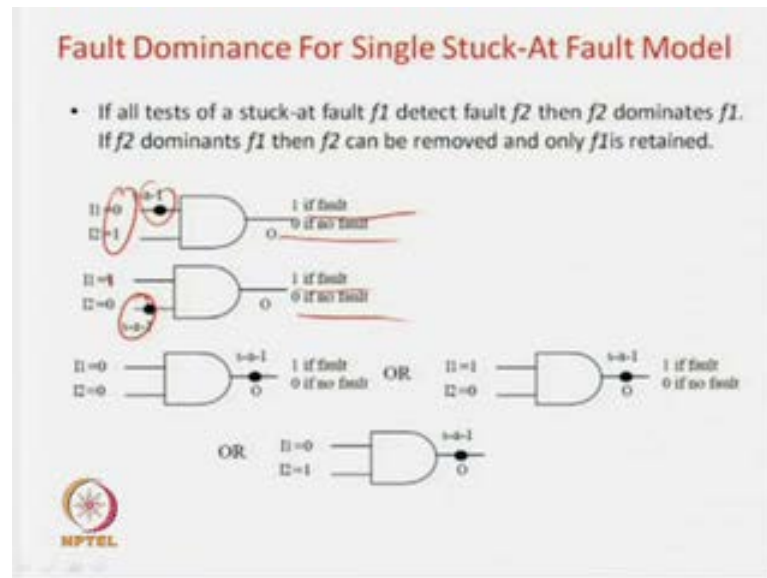


(Refer Slide Time: 32:09)

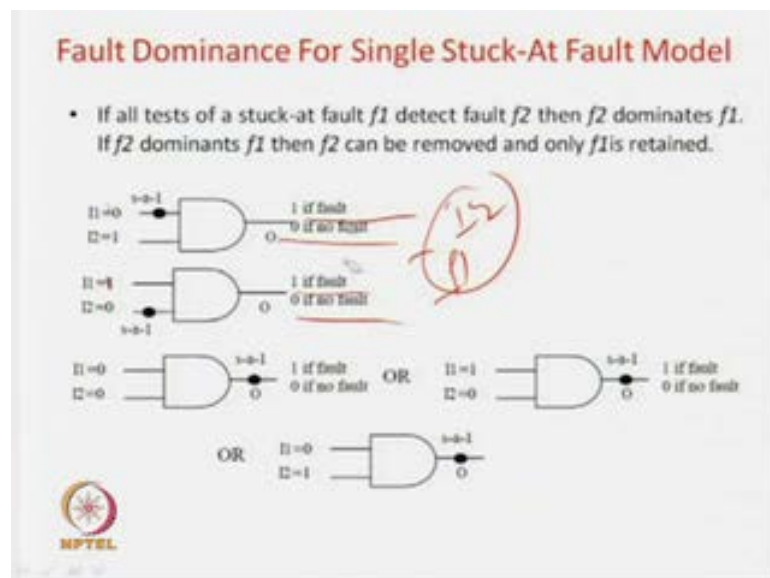


Now, we will see another example because, in the last case we have seen that, we could do a lot of things with, what do you called the stuck-at-0 fault. Now, you can see that, we have a 2 stuck-at-1 fault so far here, we have not do none anything in this case. Similarly, in this case also, we have in the last example, also we had some stuck-at-1 faults here and these are the 2 stuck-at fault for AND gate and some stuck-at-1 for the OR gate or something. So, what we have this going to see is that, some faults still remain so, can we do something with this. So, still we have seen only the collapse faults by equivalence.

(Refer Slide Time: 32:26)



(Refer Slide Time: 32:54)



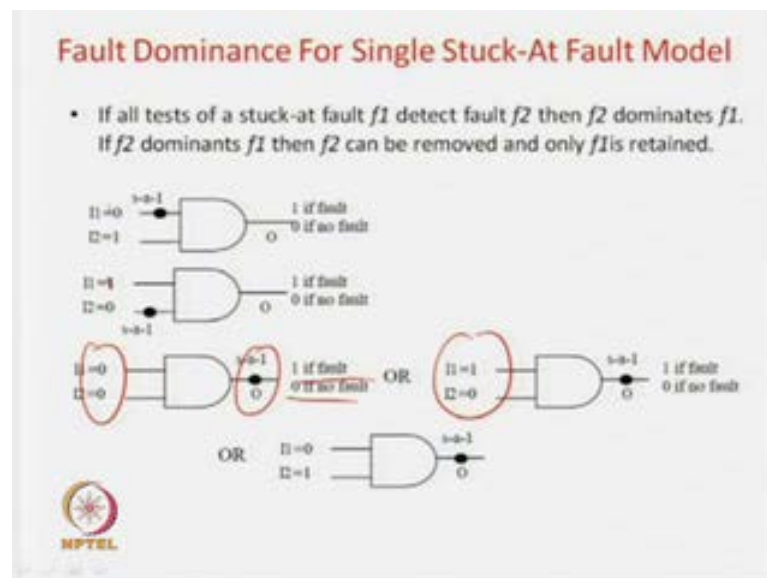
Let us see, if you can do something else, now you see this is the same AND gate. So, let us say us, stuck-at-1 fault is there so, what pattern here to apply, simply you have to apply a 0 over here and 1 over here. So, you get the answer 1 if there is fault, given in the stuck-at-1 point and you are applying 0 1 and if there is no fault then, what is going to happen, we are going to get a 0.

Now, let us say, you take the another net in this case, with that stuck-at-1 fault over here then, what we are going to apply, you are going to apply a 1 over here and a 0 over here.

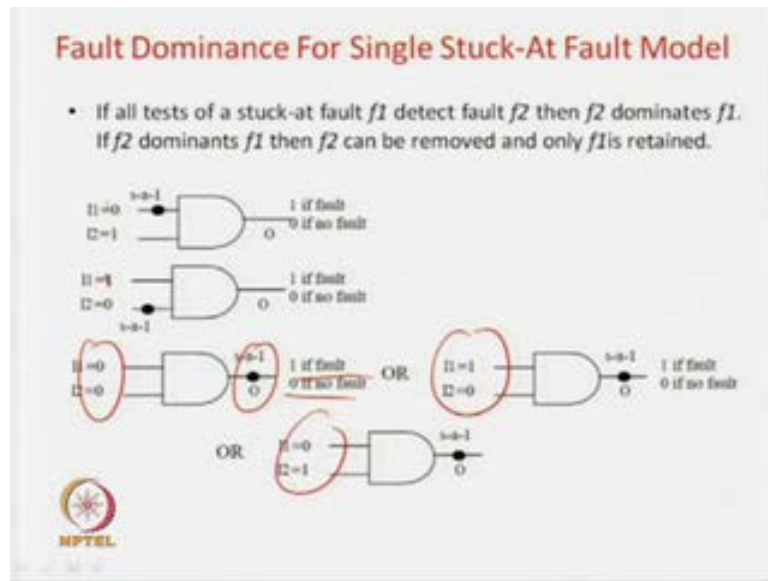
And your same answer 1 if there is fault and 0 if there is no fault that is fine but, you can easily see that, there are 2 different test pattern. And obviously, the function, if there is stuck-at-1 fault over here the function is i_2 .

If the stuck-at-1 fault over here then, what is the function, the function of this AND gate is i_2 because, whatever i_2 it will be passed and if there is a stuck-at-1 fault here so, what is the function of the AND gate, it is i_1 . Because, if it is stuck-at-1 then, whatever the value here will be passed so obviously, the two outputs are different and these faults are not equivalent. So, we cannot do anything using a equivalent equivalent faults collapsing or equivalent fault theory, if you are applying a, what you called this stuck-at-1 is concern.

(Refer Slide Time: 33:26)

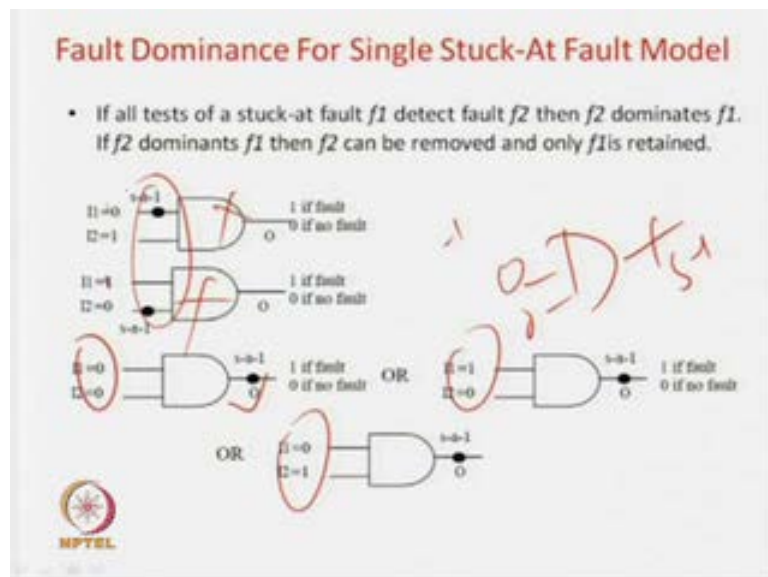


(Refer Slide Time: 33:41)

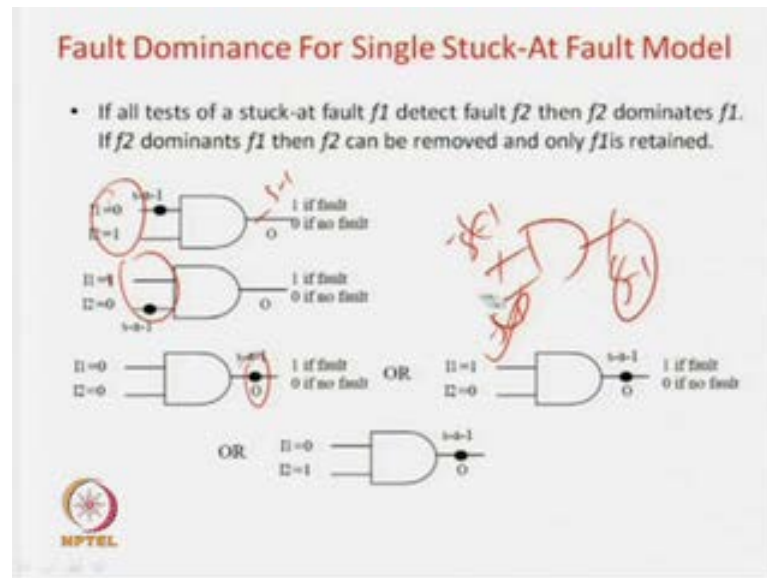


So, now, let us look at the stuck-at-1 fault at the output now, what you can apply, you can apply 0 0. So, you can apply 0 0, 1 if there is a fault because, it will be stuck-at-1 and 0 if there is no fault. Similarly, you can apply a 1 0, the answer is 1 if there is fault and the answer is 0 if there is no fault. Similarly, also you can apply a 0 1 that is, excepting 1 1 you can apply any other pattern.

(Refer Slide Time: 33:53)

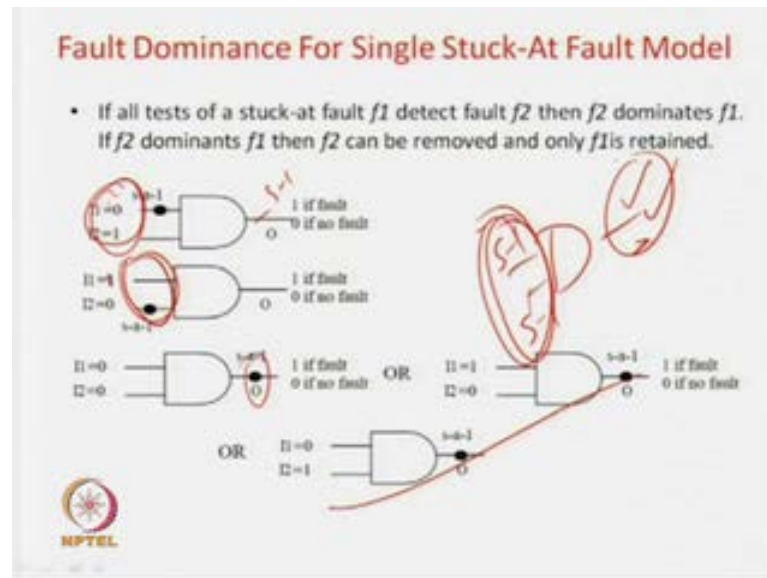


(Refer Slide Time: 34:32)



Now, the now say, what i do i just give you an idea say, let us think that, what we can do say for example, i give you say i say that, even the stuck-at faults or stuck-at-1 faults are not equivalent. But, still i assume that they are equivalent and i say that, i just take a stuck-at-1 fault over here. Then, what i can do, i can apply this pattern (Refer Slide Time: 34:04) this pattern, i can apply this pattern, i can apply this pattern or i can apply this pattern. Now, by chance so that, i apply the pattern 0 0 so obviously, this fault this thing gets tested but, 0 0 is not a test for this one and not a test for this one. So, these 2 faults are not tested so, you cannot merge or you cannot do anything of this but, now let me just changing the idea of this.

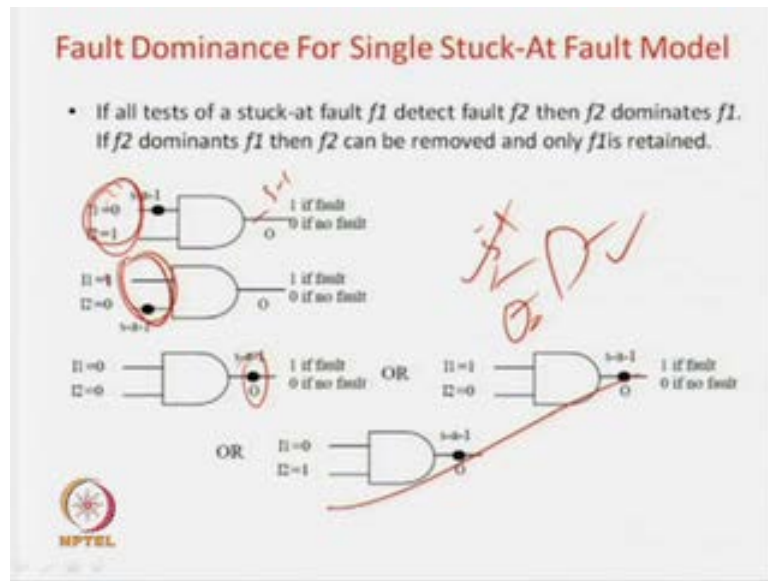
(Refer Slide Time: 35:08)



Instead of, collapsing this fault, I just keep this stuck-at-1 fault and this stuck-at-1 fault then, what happens you see then, I what I am doing to do is that, i apply I need to apply this pattern to test this one and I need to apply (Refer Slide Time: 34:43) this pattern to test this one. So, what happens, if I test this fault and then, I test this fault twice I am testing the stuck-at-1 fault just observe this. What happens, I am not considering stuck-at-1 fault here, I am just considering because, I saw that, I may land into problems if I consider stuck-at-1 fault here and if i totally do away with this one.

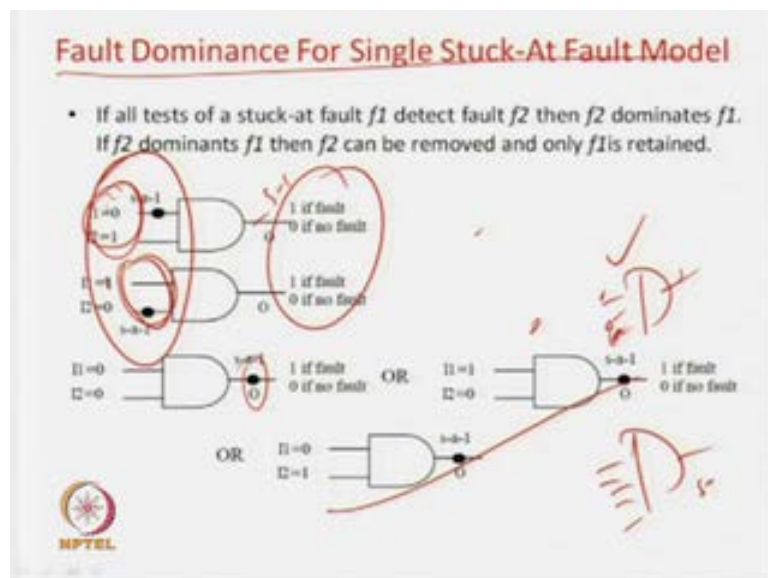
So, what i have done now, i have just changed my thought a bit then, what i have done, i have put a stuck-at-1 fault here and i have put a stuck-at-1 fault here. Now, what i am telling, to test a stuck-at-1 fault here you apply a 0 1 here, to test a stuck-at-1 fault here you apply a 1 0 over here. And by if you look at this one i have applying these two patterns and twice i am also testing so, automatically what happens, these two together dominates this one or we can say these two, together can covered this one but, one you cannot do it.

(Refer Slide Time: 35:39)



Because, if i say that, if i consider only a stuck-at-1 fault here then, what happens if i apply this one then, both of them are not tested. There is a worst case possibility, you have to always consider where, if i have only one stuck-at-1 fault here then, you are going to test this one, fine. Then, we are also going to test this one fine but then, this one will not be tested. Because, to test a stuck-at-1 fault here, one apply 0 1 and 0 1 is not a test for a stuck-at-1 fault over here.

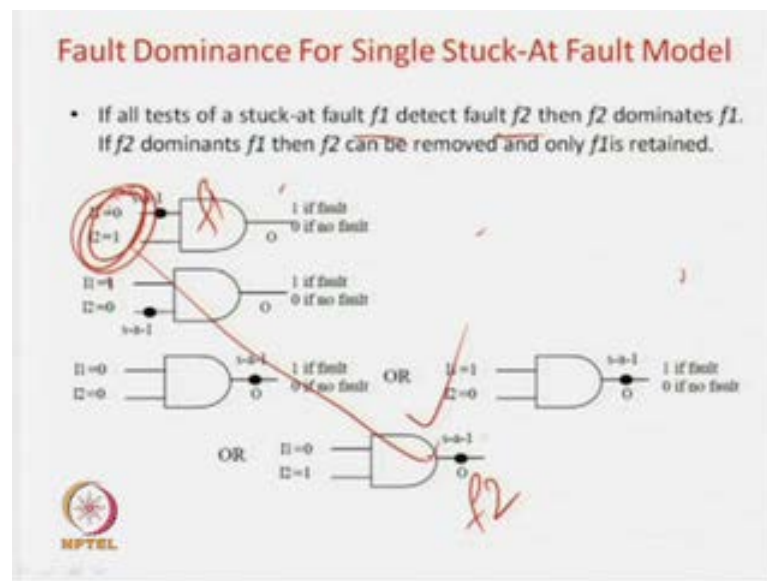
(Refer Slide Time: 36:04)



So, therefore, if we but, if we keep these two stuck-at faults at the inputs then, automatically the output stuck-at fault gate tested twice. For that is why, we can say that, this is another type of fault collapsing, which is called fault dominance. So, this is also you can reduce the number of faults from c to 2, in case of fault equivalence, what we have done, we have a 3 input and 2 input AND gate and then, we called from 3 is stuck-at-0 fault so, we have made it 2 1.

So, also they have seen that, if it is a 10 input AND gate then, 11 stuck-at-0 fault can be converted to 1. But, in case of stuck-at-1 fault AND gate, it is not that much reduction but, it is somewhat, at least you reduce the stuck-at fault at the output. That is, if you take this sorry if you take a stuck-at-1 fault at a input, stuck-at-1 fault at a input automatically stuck-at-1 fault at a output is tested. So, at least one fault you are reduce so, whatever you are reducing is good for us so, some reduction is there. Now, is but, this not as high as fault equivalence.

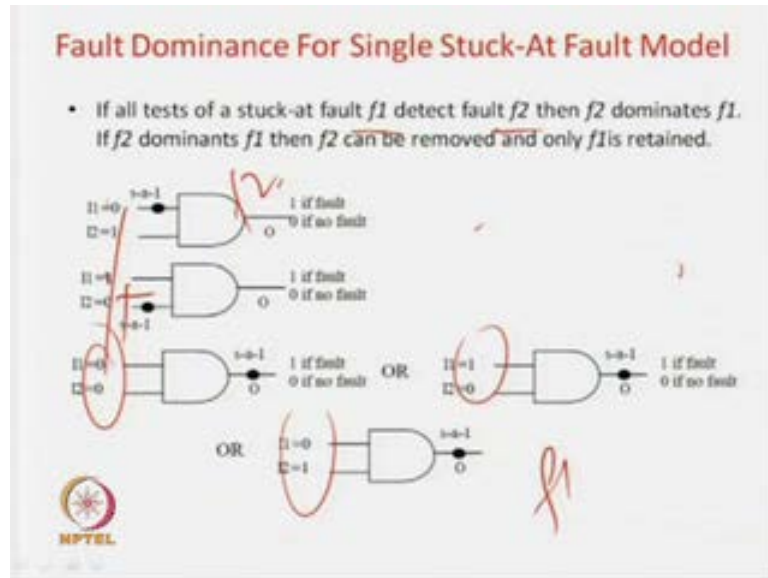
(Refer Slide Time: 36:54)



So, let us look at the formal definition, the formal definition says that, if all test for a stuck-at-1 fault, if all tests for a stuck-at-f 1 detect of fault f 2. So, for all test of stuck-at fault is f 1 so, all test of stuck-at-1 fault is fault f 1 so, all test for f 1 is what, only 0 1 because, this stuck-at-1 fault is has only one pattern so, that is 0 1. So, we are saying that, all test of pattern f 1 detect fault f 2 so, obviously, this detects f 2 is correct then, f 2 is dominating f 1 so, these f 2.

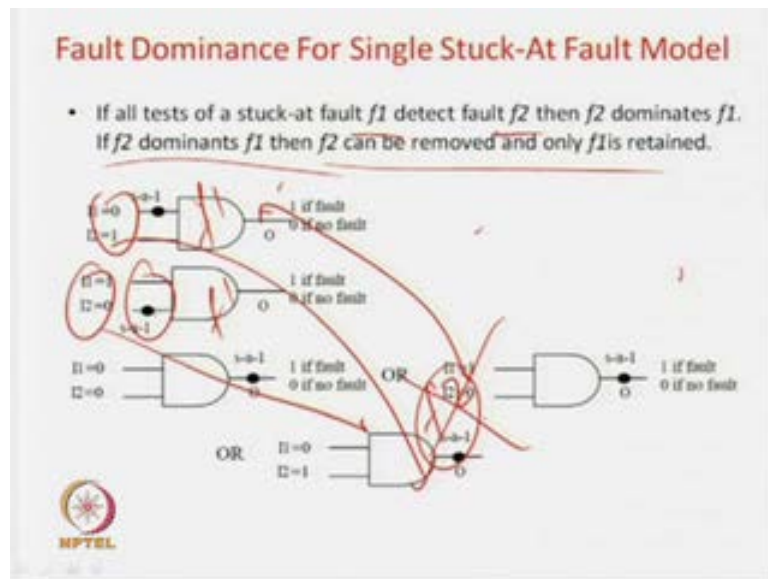
So, all test of f_1 , this is the all test of f_1 , only one pattern is there, these detecting f_2 then, f_2 dominates f_1 . So, in this case, f_2 is dominating f_1 but, you can easily see, the definition other way does not hold.

(Refer Slide Time: 37:40)



All test, all pattern, if i consider this one as f_1 , this one as f_2 then see what happens, all tests of f_2 . So, this is one test of f_2 , (Refer Slide Time: 37:47) this is one test of f_2 , this is one test of f_2 , all test of f_1 so, detect f . So, this one does not detect f_2 so, f_2 does not dominates f_1 .

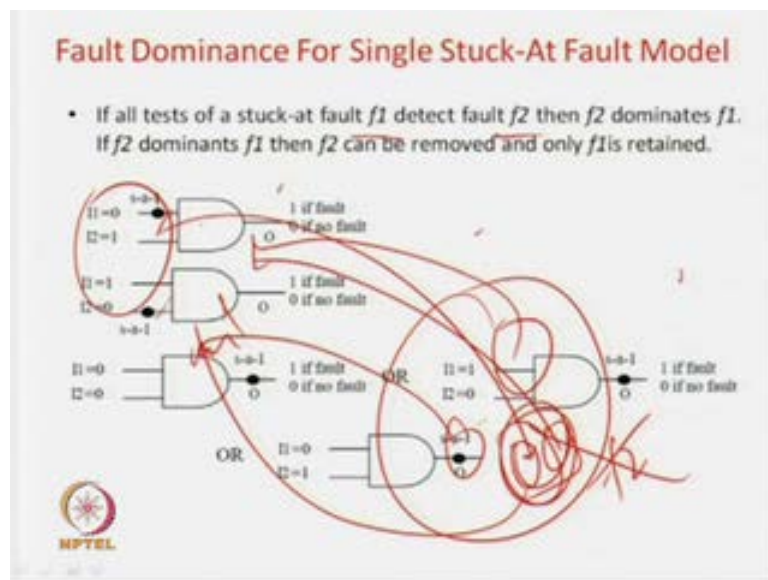
(Refer Slide Time: 38:02)



But, we can easily say that, all test of f_1 that is, single in number is detecting this one so, f_1 is detecting f_2 . So, f_2 is dominating this one so, f_2 is actually dominating this one. Similarly, all test of this guy, is only one pattern of this guy actually test this one so, this one also you can consider as f_1 . So, it is saying that, if all test of fault f_1 detect fault f_2 then, f_2 dominates f_1 .

So, f_1 is this fault, f_2 is dominating this one as well as this one now, if this is the case then, you can say that, if f_2 dominates f_1 then, f_2 can remove then only, f_1 is retained so, you just eliminate f_2 and you already retained f_1 . So, basically, the philosophy here is very simple, it says that, all patterns of this guy or this guy the test this one. So, you keep this one, this one and you need at this one.

(Refer Slide Time: 38:51)

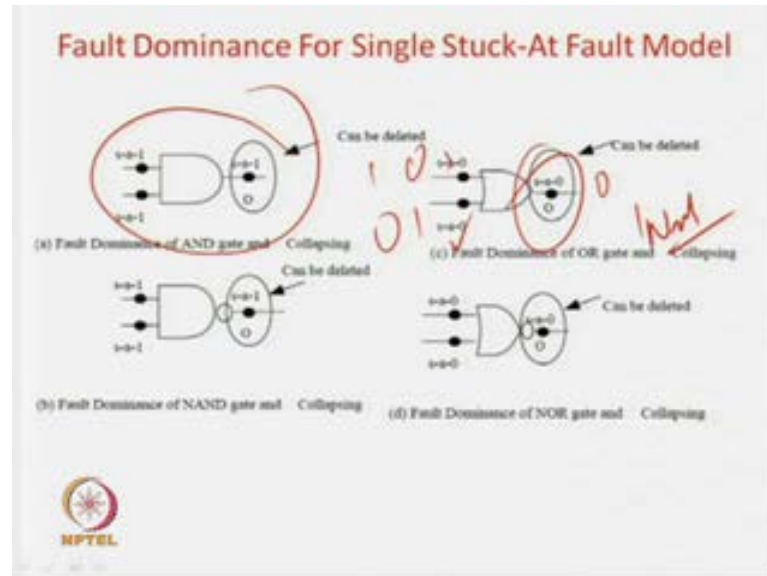


Because, if you do the other way round then, you may have a problem, it says that in this case, this f_2 this one pattern may detect this one and another pattern may detect (Refer Slide Time: 38:58) this one sorry this one. But, there is some other pattern lie in this case, which are not detecting these two. So, if somehow you test this f_2 using the remaining part then, this f_1 's will not be tested.

So, therefore, f_2 dominates f_1 means, whatever he available here can take place so, better you eliminate this one then, automatically this guy will be tested more than once. so, that is the concept of fault collapsing by fault dominance. So, in this case, is this AND gate what will going to happen, you are happening only one stuck-at-1 fault, one

stuck-at-1 fault here and automatically, this can be retained I mean, this can be eliminated.

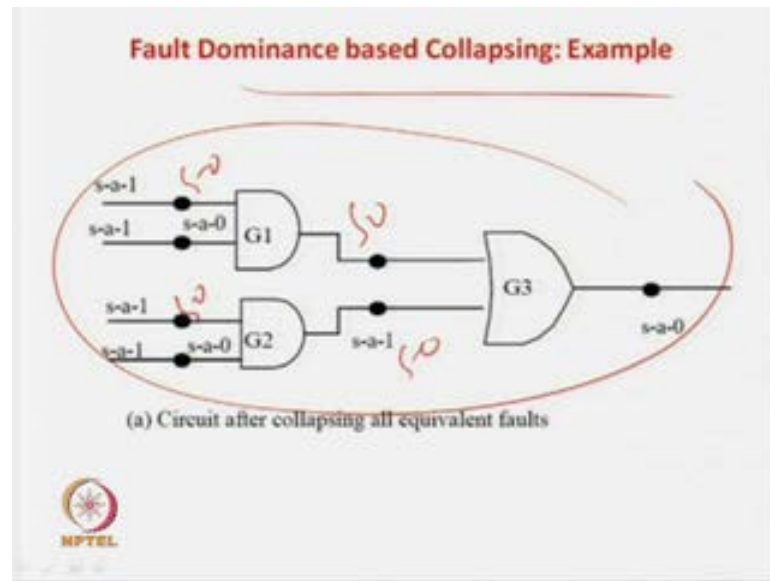
(Refer Slide Time: 39:34)



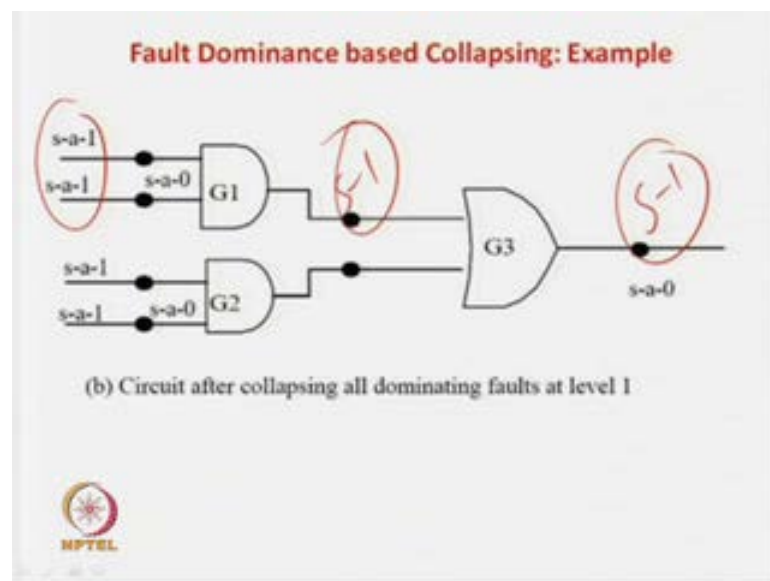
So, let us see the generalized example so, in case of so, AND gate this is the case so, one you can delete so, in case of OR gate, where i told you, you can easily verify that this is just the dual. So, in case of OR gate, it is stuck-at-0 so, to test a stuck-at-0 fault over here so, you have to apply 0 and 1 over here. To test this fault, you have to test this fault, you have to apply 0 over here and a 1 over here and obviously, if you are applying the 0 1 in this so, the answer is 0 if this is a fault and otherwise is normal.

So, in this case also, you are applying 0 and the 1 so, if you are applying sorry if you are applying something like 0 over here and 1 over here to test it then, the answer is a 0 if the fault is there and 1 if it is normal. So, i will easily observed that, 0 1 and 1 0 also test in this form so, in case of OR gate, you can easily eliminate down this. So, i mean inverter all this, all faults were collapsed by this inverter so, these only stuck-at, this is stuck-at-1 fault, which was easily eliminated or collapse by fault equivalence. When in case of fault dominance, we did not consider the inverter, again these things are remaining same for the AND and the OR gate, this is very obvious and you can easily understand.

(Refer Slide Time: 40:48)



(Refer Slide Time: 41:52)

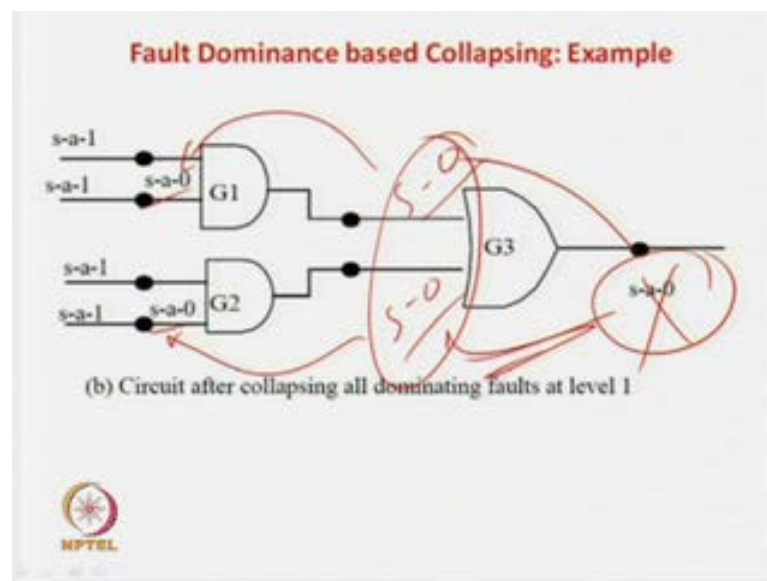


So, now, let us take another example, when you are going to see the fault dominance based collapsing. In the last example, we have got this circuit where, you are using where you are use only fault equivalence to do the collapsing. So, in this case, if you remember we had a stuck-at-0 fault here, we had a stuck-at-0 fault (Refer Slide Time: 41:06) here, stuck-at-0 fault here, stuck-at-0 fault here and also we had a stuck-at-1 fault here and a stuck-at-1 fault here.

So, this guy and this guy were collapsed and we got this one similarly, this guy and this guy was collapsed and we got this one. So, this guy and this guy got collapsed and we got this one, this was by equivalence. Now, we will see, what else because, some faults is still remaining, let us see what we can do by using the dominance. So, in the 1st level you can already in the 1st level you can see all the stuck-at-1 is already collapsed by equivalence with the AND gate but, these two can easily collapse this one.

Because, if you are testing these two, automatically these two get tested so, this stuck-at-1 fault is eliminated. And you can just also you have just remember the point that, this is a stuck-at-1 fault over here so, this is get eliminated by this one, as well as this is also getting eliminated by a stuck-at-1 fault by equivalence. So, a single fault can be getting eliminated or collapsed by both so, this one get this this stuck-at-1 fault gets, i mean collapsed by equivalence with OR gate as well as by dominance with this one. So, similarly, we we have this one is the final thing we have at level 1 now, if we go through this level, we have to just know that.

(Refer Slide Time: 42:26)



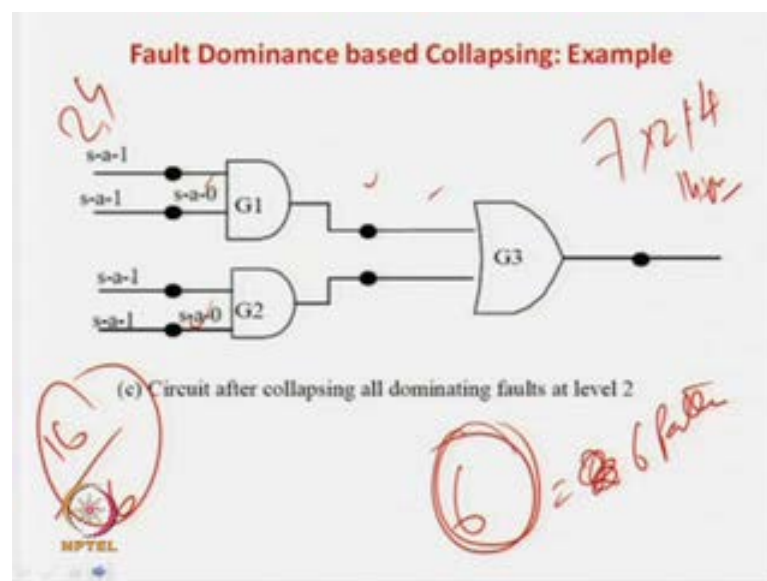
So, we have a stuck-at-0 fault here and stuck-at-0 fault here, this you have to understand. Now, who is having this stuck controlling, this stuck-at-0 fault, these two stuck-at-0 faults are actually being tested by this one, this is not shown here. Because, they have already collapsed but, still they are, we have to know that, if there is a stuck-at-0 fault here and stuck-at-0 fault here, these two stuck-at-0 faults are automatically getting tested.

So, if these two stuck-at-0 faults are getting tested by fault dominance of an OR gate, this one is also getting tested.

So, this you can collapse by virtue of these two and (Refer Slide Time: 42:51) these two are getting collapse by virtue of this one. So, finally, we are having a circuit like this so, this we just repeat, the last part was big tricky so, there is a stuck-at-1 fault here, which you can easily collapse by this one. But, now, also we had a stuck-at-1 fault over here sorry a stuck-at-0 fault over here, which you not finding out how to collapse.

Because, you are not seeing stuck-at-0 fault here but, you have to understand that, these two stuck-at-0 faults are here then, they are actually being tested by virtue of these two. So, these three gets collapsed by fault equivalence fault dominance and this one gets eliminated and only these two remains and again these two gets eliminated by fault equivalence of this one and this one.

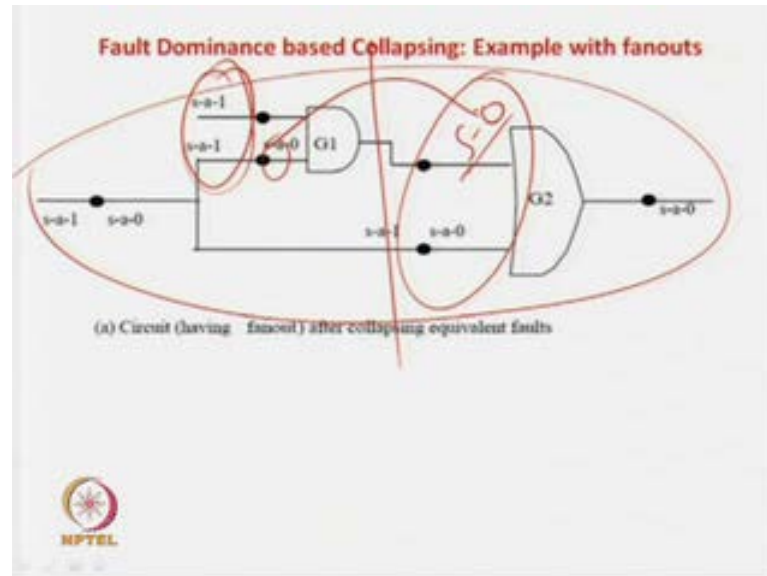
(Refer Slide Time: 43:30)



So, this is our final circuit we are having so, the number of faults here are 1 2 3 4 5 6 so, you have 6 faults here, which needs to be tested. So, we have 2 sorry we have 6 patterns required to be tested and the number of nets here are 1 2 3 4 5 6 7 so, 7 into 2, 14 faults were there. So, we we should have require 14 test patterns but, now with this collapsing, we have only 6 in numbers. So, we have reduce the number of test patterns from 2 to the power 4 because, 1 to 4 patterns are there from 16 to 6. So, this is a great achievement,

which you have done using stuck-at fault model and what you call, the fault collapsing equivalence and dominance.

(Refer Slide Time: 44:07)



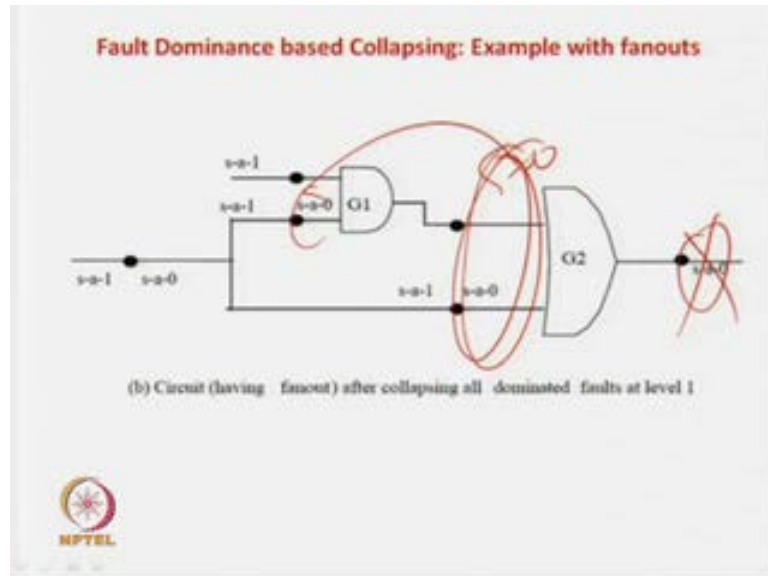
Now, let us again see the same example with both stuck-at fault with equivalence and dominance and we are looking at the fanout example. So, this all is own example with fanout so, this is have keep, we are already got what you can see this is circuit we have got after fault collapsing on equivalence. Now, let us see, what we can do now, if you see the level 1 so, these are the 2, this stuck-at-1 faults are there. Now, let us see, what we can have so, this this is a stuck-at-1 fault already there.

So, because the OR gate is collapsing so, OR gate is, this is also stuck-at-1 fault over here. So, this one, (Refer Slide Time: 44:47) this one and this one, this collapse all this stuck-at-1 faults and now, these are all collapse equivalence. So, let us see what we can do in level 1 so, we do not have any changes at this level but, let us the at this last level we can see that there is a change.

So, sorry So, this one we have level 1 so, this is stuck-at faults with equivalence collapsing whereas, you can see in level 1 that is, we are considering this as level 1, there we cannot have any fault collapsing with the dominance. Because, already, we have both the stuck-at-0, which is collapsing in the stuck-at-1 that is by dominance. So, only what is left is, in this level we may try something because, we know that, in case of an OR gate, these 3 stuck-at-0 faults get collapsed by these 2 stuck-at-0 faults. But, this stuck-

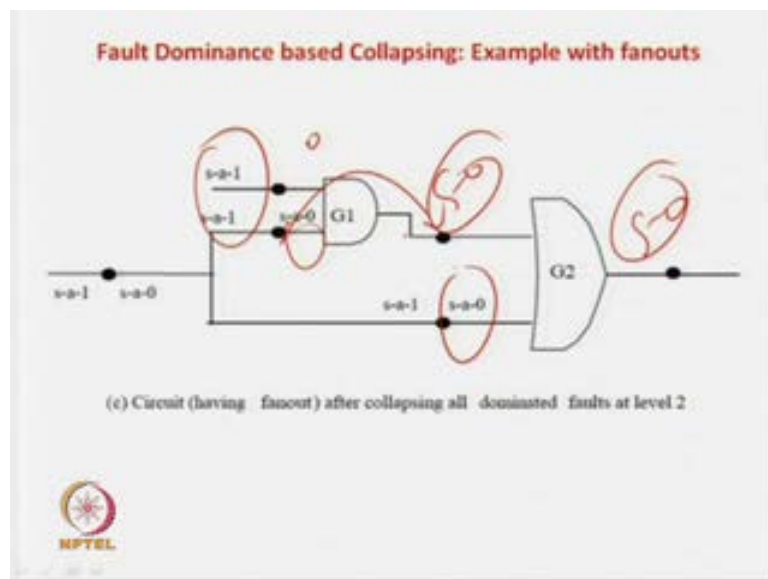
at-0 fault is not visualize here because, this guy is continuing over here that is what, is achieved here.

(Refer Slide Time: 45:42)



This is a stuck-at-0 over here so, these guys controlling this and these 3 stuck-at-0 faults are can be keeping these two. This can be eliminated by fault dominance and so, we are having a circuit like this.

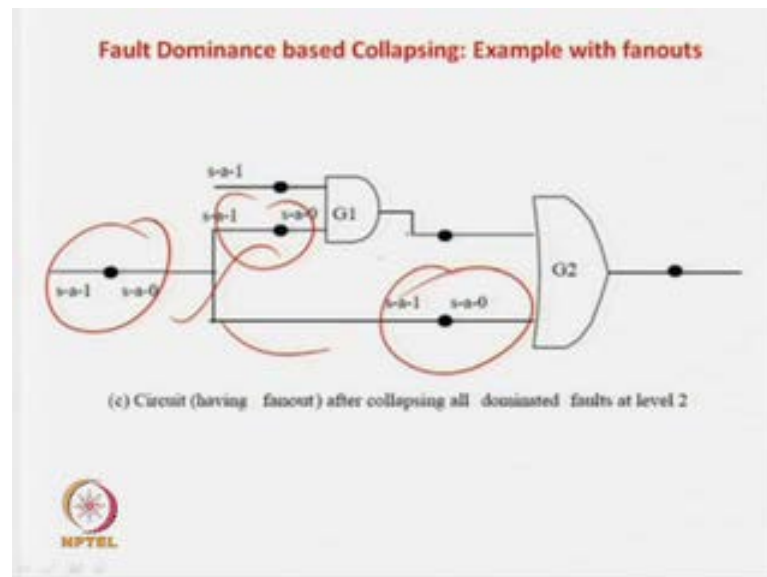
(Refer Slide Time: 45:53)



So, we are having this this stuck-at-0 stuck-at-0 this stuck-at-0 this stuck-at-0 and this stuck-at-0 these are taking care of by this one by equivalence. And you can say that, this

stuck-at-1 is equivalent dominated by this one and this is also eliminated and there should have been a stuck-at-0 fault here. So, which is actually again this is a stuck-at-0 implies a stuck-at-0 over here and these 3 guys by fault dominance actually it (\emptyset). So, you can also see that, also you have reduce the faults to a quite a large number.

(Refer Slide Time: 46:29)




But, now, we have already seen that but, this guys this guys and this guys we cannot do any kind of elimination. Because, we have seen that, these things are not equivalent in case of a, what you called fault dominance or fault equivalence. Fanout lines are not not equivalent or they are not dominated so, all these faults in individual fanouts we have to keep.

(Refer Slide Time: 46:46)

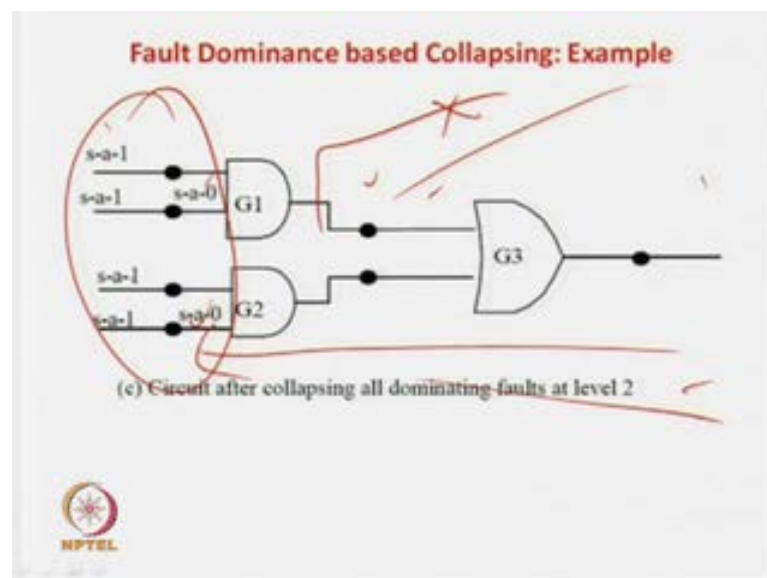
Check point theorem.

- A circuit with no fanouts, s-a-0 and s-a-1 faults is to be considered only at the primary inputs. So in a fanout free circuit test patterns are 2^x (Number of primary inputs).
- For circuit with fanout, checkpoints are primary inputs and fanout branches. Faults are to be kept only on the checkpoints.

So a test pattern set that detects all single stuck-at faults of the checkpoints detects all single stuck-at faults in that circuit.

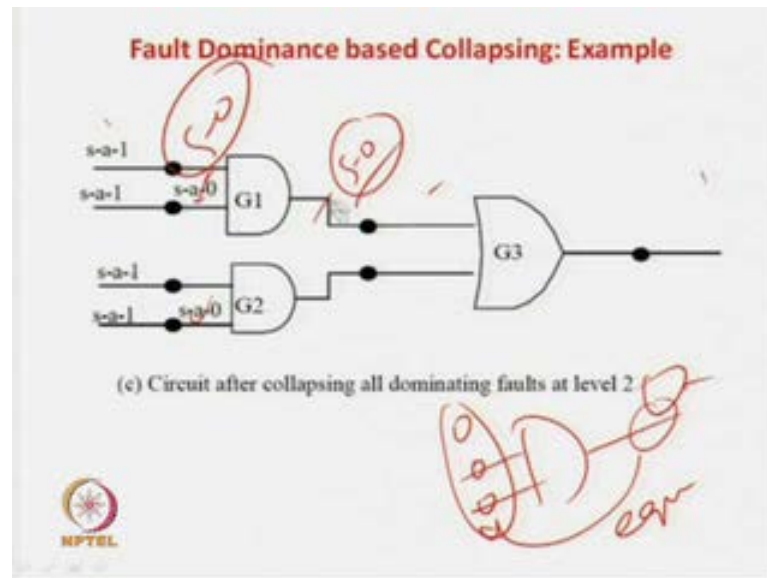


(Refer Slide Time: 46:58)



So, now, towards the n or you can say that, you have to understand a very interesting fact over here. So, what is the interesting fact, you can see here, last example if you see so, where are the stuck-at faults, it is a circuit without any fanout it is a circuit without any fanout. So, what happens, all these faults are actually getting clustered over the inputs, no internal lines is having any kind of stuck-at faults. Because, all have been collapsed and brought to the input level and this circuit does not have any kind of a fault, what you call sorry any kind of a fanout, there is no fanout.

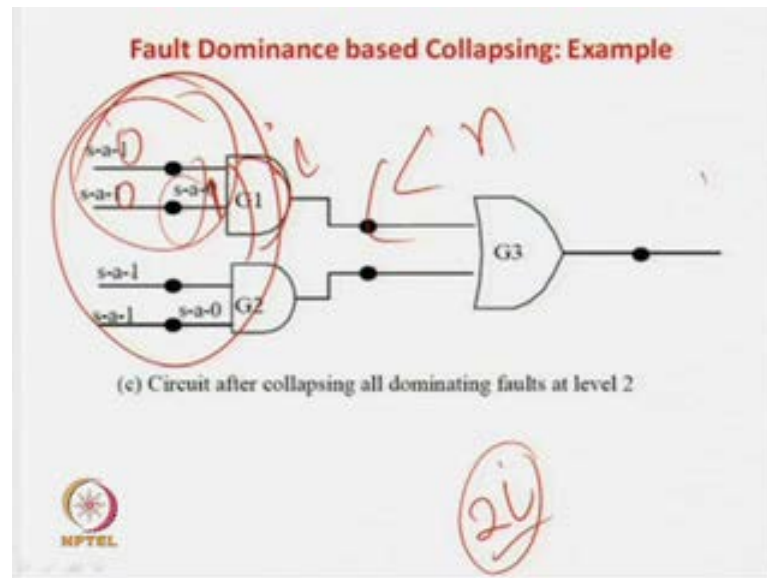
(Refer Slide Time: 47:33)



So, it is very obvious, why it happens because, for any gate like AND gate, OR gate, NOT gate or whatever so, the fault fault equivalence, the fault of the outputs can be brought over here by equivalence (\equiv). And by fault dominance what happens, the output faults stuck-at-0, stuck-at-1 can be brought by a equivalent number of faults at the sorry by by n number of faults at the inputs. Like this stuck-at-1 fault here, are brought together by the 2 stuck-at fault, 1 faults here, see dual thing happens for the OR gate.

And and for this thing, you can see there is a stuck-at-0 fault over here, though there is a stuck-at-0 fault over so, both of them get converted in this one. So, we can very easily see what happens, all the inputs are only having a stuck-at-0 fault and the internal lines do not have any other stuck or any other fault has to be considered.

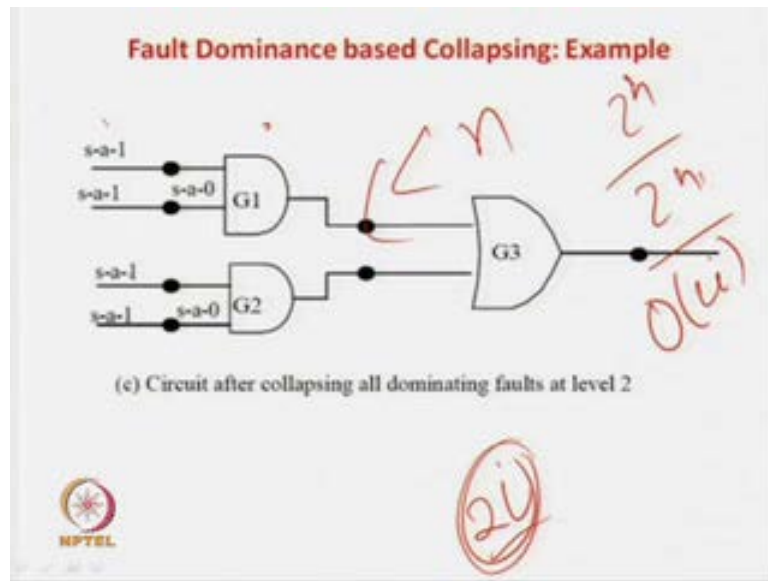
(Refer Slide Time: 48:17)



Because, by fault dominance and fault equivalence, all the faults get transfer to the inputs of i and if there is a no fanout then, this thing hold transitively and all outfall only the inputs of a circuit have falls. So, this gives very very less number, if a circuit internally has n nets so, you can know that, number of inputs will be much much less than the number of internal line so, let it be i .

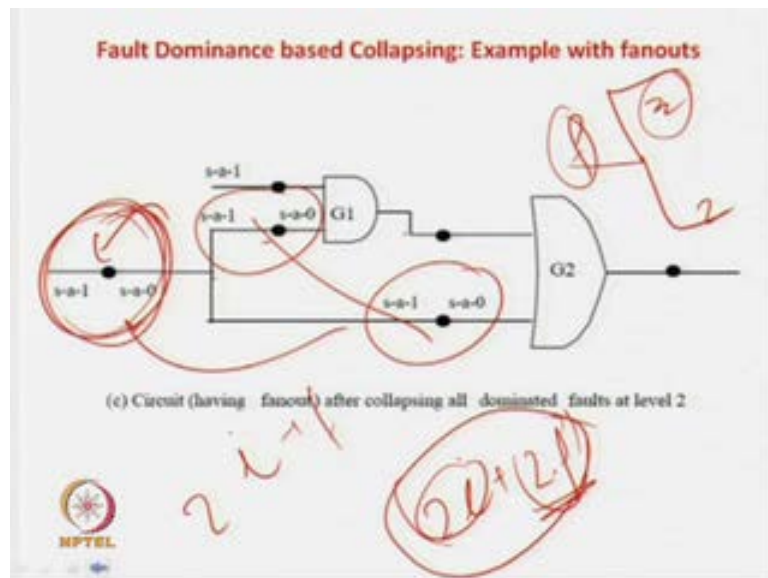
So, in the old case, only the 2^i number of stuck-at fault is to be considered, that 2^i also is a upper bound because, if it is a AND gate then, we will have only stuck-at-1 and only one stuck-at-0. And if it is the OR gate then, you will have the dual then, you have only the 2 stuck-at-0 and 1 stuck-at-1, in case is an OR gate and vice versa.

(Refer Slide Time: 49:02)



So, if the only the upper bound is 2^i where, i is the number of input lines and there will be also much some what less, depending on the OR gate or the input. So, we have come down from 2^n to the power n to 2^n and from 2^n to upper bound of 2^i where, i is the number of input. So, this is the beauty of stuck-at fault model.

(Refer Slide Time: 49:19)



But, if there is a if there is fanout then, the number the number slightly rises because, now, why it rises you can already see. Now, we have stuck-at faults at the inputs of the gates that is, 5 and also at the fanout branches because, this guys and this guys cannot be

brought over here because, they are not equivalent. So, in addition to 2^i that is, the number of inputs, again some 2^f or you can say 2^f number you have to add for the number of faults.

Because, they are the number of fanout branches, each fanout branches we have a stuck-at-0 fault and the stuck-at-1 fault. Sometimes they will be merged with, what you can call say, for example, in this case this is a fanout and this is a fanout so, this fanout is also the input of the gates and this fanout is also the input of the gates and this also some extra fault we have to have.

So, depending on the some fanout say, this is f so, we will have some faults here we will have some fault here but, these faults and these faults cannot be collapsed. So, and again two faults will remain so, if there are f fanout branches so, along with 2^i , also 2^f number of fanout get added and again this is the upper bound. So, if we have fanout, the number of faults or the number of test patterns in the in such a case is slightly higher than 2^i into number of inputs that is, the number of 2^i into number of fanout nodes, i mean this starts a fanout so, this will helps an extra faults. But, if if you are forgotten the concept of fanout then, these 3 faults could have been moves to this one so, this is actually called a check point theorem.

(Refer Slide Time: 50:48)

Check point theorem.

- A circuit with no fanouts, s-a-0 and s-a-1 faults is to be considered only at the primary inputs. So in a fanout free circuit test patters are 2^i (Number of primary inputs).
- For circuit with fanout, checkpoints are primary inputs and fanout branches. Faults are to be kept only on the checkpoints.

So a test pattern set that detects all single stuck-at faults of the checkpoints detects all single stuck-at faults in that circuit.

Handwritten note: $2^i + 2^f$

NPTTEL

We says that, if there is no fanouts then, the number of test patterns or the number of faults is 2^i into the number of primary inputs upper bound. And if there is a fanout then,

you have to put checkpoints and the primary inputs as well as the fanout cases. So, the beauty of stuck at fault is that, it detects it says that, if you have test pattern, i mean the beauty of stuck at fault model says that, and any check point is saying that if you have a fanout then, you put check point at the primary input as well as the fanout steps or the fanout branches, appropriate positions.

And just find out, what patterns you test it then, the number of faults are much much less than 2^n say, some 2^i plus 2^p kind, 2^f and a stuck where as, is the number of fanout kind of a thing. Then, a very very less number of patterns as well as very very less number of faults are required to be handle, which is much much less than again 2^n . And but, still the statistically it has been showed that, the confidence or the accuracy is as high as 99.9 percent plus.

(Refer Slide Time: 51:41)

The slide is titled "Questions and Answers" in red text. It contains two bullet points in black text. The first bullet point asks: "For what class of circuits, maximum benefit is achieved due to fault collapsing and when the benefits are less? What is the typical number for test patterns required to test these classes of circuits?". The second bullet point asks: "What faults can be collapsed by equivalence in case of XOR gate?". There are handwritten red annotations on the slide: a circled "10" next to the first bullet point, a circled "10" next to the second bullet point, and a vertical line connecting the two circles. In the bottom left corner, there is a logo for NPTEL.

So, in a last class, what we have seen, the beauty or structural testing in the fault models is there, we did not have any extra peens, we did not have any extra registers, we did not have any extra multiplexers as well as we have we can just apply some test patterns. And verify that, there is no stuck-at fault models in any of the nets and even the number of the nets say, n we do not require to apply even 2^n patterns, the number of patterns are even much less than 2^n .

Because, by fault collapsing by fault equivalence and fault dominance the number comes to as upper bound of 2^i plus 2^f , if there is fanouts and so, the test time becomes very

very less. Because, and because what you can call that, the idea is 2^{2n} , if n is the number of nets in the circuit there. Now, you required 2^n patterns but, by collapsing the number comes to $2^i + 2^f$ where, i and f are much much less than n . So, our test time is much much less because, the patterns are less so, we have solved a very big problem of test pattern or test time reduction.

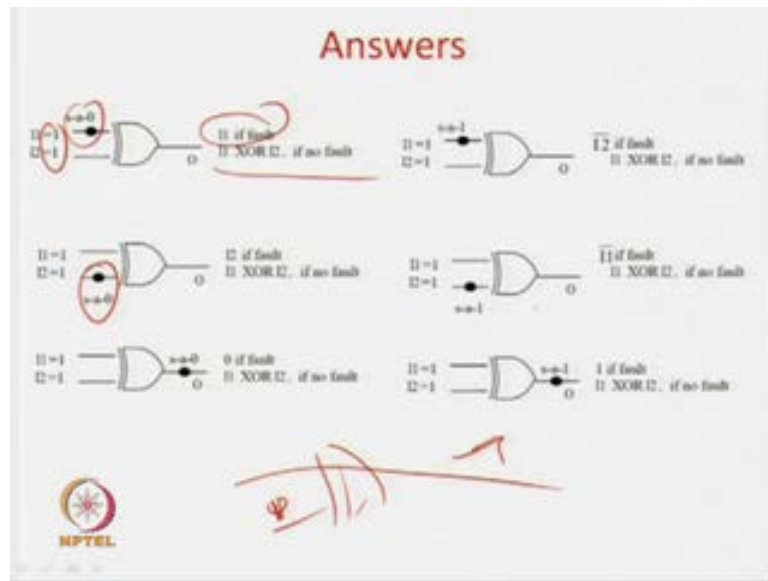
So, we have reduce the number of test patterns, yes having a accuracy of 99.9 percent plus and our test time remains very low here because, the number of patterns are less. So, we have achieved our target using structural testing with fault models and in the next sequence of lectures, we will see even at test, fault a stuck-at fault, how can you automatically find out a test pattern. Because, in our cases circuits are very simple having a AND gate, OR gate on 2 or 3 level of circuits were there.

What is the very complex circuit then, you have to find out, that also requires a algorithm to find out that, given a test or given a faults f internally a circuit, how do you find out a test pattern that test the fault so, that we will see in subsequent lectures. Now, if you are look at the question and answer session so, today we will ask you simple stuff. So that, we can see in the examples later. So, for what class of circuits maximum benefit is achieved due to fault collapsing and when benefits are and when benefits are less.

What is the typical number of test pattern required to test the circuits so, very simple, you can say that, if the number of if the circuits have very less fanouts or if the circuits not have any fanouts then, the number of faults are only equivalence only equal to order of 2 to the number of primary inputs. So, here you get a great benefit but, if the number of fanouts are very very large then, whole of the benefits goes away because, faults cannot be collapsed in the fanout.

So, the typical circuit, number of fanouts are never very high so, we always get a good amount of benefit, when you are what you call, you are taking a fault collapsing by fault equivalence and dominance. But, if you have a high number of fanouts then, slightly the benefit is less. The second question is, what is the what faults can be collapsed by equivalence, in case of an XOR gate. So, we have seen in our study AND gate, OR gate, inverter and NAND gate but, we have not seen XOR gate.

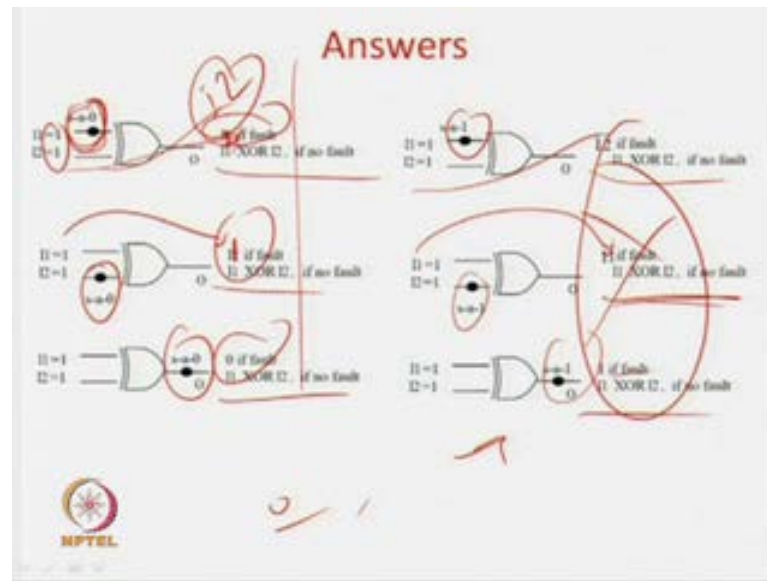
(Refer Slide Time: 54:29)



So, what is the case by, can you do anything by fault equivalence, the answer is no because, for fault equivalence the answer is to be that, with a fault here and with a fault here, the output function for this one and (Refer Slide Time: 54:37) this one should be equivalent. But, this is not the case of in case of XOR gate so, we cannot have any fault collapsing in this by equivalence.

You see, if i have a stuck-at-0 fault here so, what is the answer, the answer is i 1 sorry if i 1 so, to have a stuck-at-0 fault here, what you have to do, you have to apply 1 1. So, what is the function, if there is a stuck-at-0 fault here then, what is the answer, i 1 if there is a fault. So, if there is a fault over here so, 0 is actually propagating, i 1 is propagating and if there is no fault, the answer is i 1 XOR i 2 if there is no fault. Now, if there is a stuck-at-0 fault over here so what happens, we know that, in XOR gate is the controlled inverter. So, if one line is 0 then, XOR gate actually passes whatever, is in the output and if it is a 1 then, next one passes the inversion.

(Refer Slide Time: 54:30)



So, in this case, if it is stuck-at-0 fault over here, over here so, obviously, stuck-at-0 fault over here. So, what happens is actually, this i_1 gets pass through actually, you should write it say, i_2 (()) mistake from my side it was i_2 because, is a control inversion so, if you stuck-at-0 means, whatever is available here will pass (()). Now, in this case, if you see this stuck-at-0 fault over here then what happens, it is i_1 if there is a fault and and because, i_1 gets passed because, a controlled inversions stuck-at-0 so, it reduced.

If there is a no pass then, i_1 XOR i_2 so, this i_2 and i_1 are not equivalent so, that is a problem. Now, if you have a stuck-at-0 fault over here so, what is the answer, if i_1 XOR i_2 if no fault and is a 0 if there is a fault. So, you will see there, neither of them are equivalent now, in this case you see, if it is a stuck-at-1 fault over here then, what is the answer, is the control inverter so, one is the inversion so, i_2 will be inverter and it will be going over here.

And if there is no fault, it is simple in solving now, in this case, if you see stuck-at-1 fault over here then, i_1 will be inverter then, it will be here and is i_1 or i_2 , if there is no fault. And if the stuck-at-1 fault here 1 (()), there is no fault so, neither of them are equivalent So, we cannot have any kind of fault collapsing for an XOR gate. So, if your circuit has too many XOR gates then, we say that, it is a not a good design from a test ability point of view.

So, thank you and we come to the end of this lecture so, from in next lectures onwards, we will see, how to generate test patterns automatically even the faults.

Thank you.