Design Verification and Test of Digital VLSI Designs

Prof. Dr. Santosh Biswas
Prof. Jatindra Kumar Deka
Indian Institute of Technology, Guwahati

Module - 8
Fault Simulation and Testability Measures
Lecture - 4

Testability Measures (SCOAP)

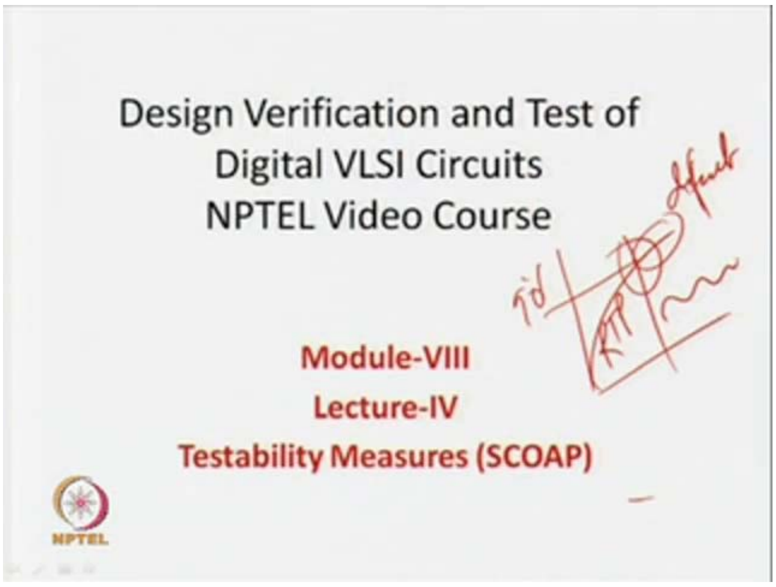
Welcome to the lecture 4 of module 8 so, in the last 3 lectures on fault simulation so, what we have basically seen that, that when we want to go for automatic test pattern generation that is, for a given fault, you want to find out which pattern detects it. So, we have found out that, there are two schemes basically, so one is based on random test pattern generation and one is based on sensitize propagate and justify approach.

So, what we have done in random pattern generation based approach; so in random test pattern generation based approach, what we have done. So, we have taken basically a fault, basically a random pattern, and then we have found out that, which are the faults that can be detected by the random pattern. So, that can be, we have seen four algorithms for that serial parallel detective and concurrent and then, the idea was that, that we can easily go upto say, around 90 percent of the faults say, which are easy to test.

So, for the 90 percent of the faults, you will find out that, for a new random pattern, the number of new faults that is deductible will be around say, 90 percent of the faults we can detected by the random pattern generation.

,

(Refer Slide Time: 01:21)



Design Verification and Test of
Digital VLSI Circuits
NPTEL Video Course

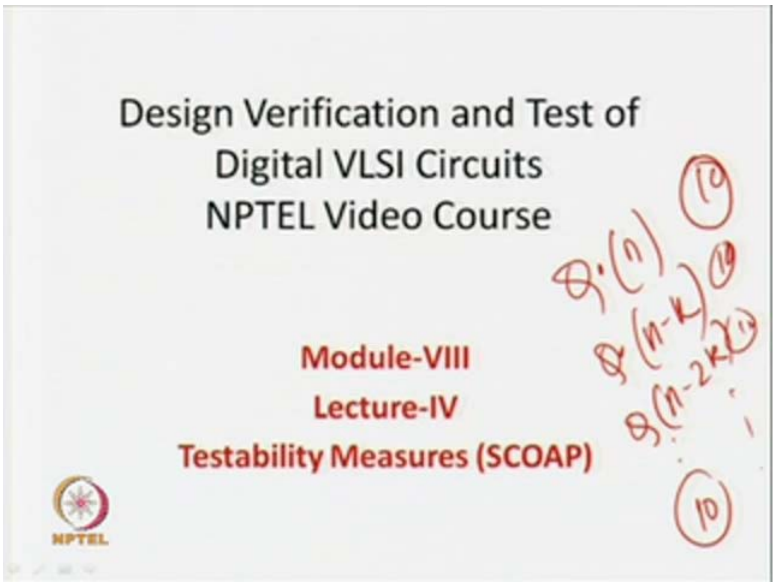
Module-VIII
Lecture-IV
Testability Measures (SCOAP)

But after this 90 percent, there are some faults which are actually call difficult to test faults. So, there are difficult faults and for that, your random patterns cannot detected that is, when if you apply random pattern after say, 90 percent of your fault coverage, we will find out that, it does not detect a new pattern or it may detect a very less number of new patterns.

And then, the next pattern random pattern you apply, it may not detect any other fault and this can go on. So, at that time, we stop the random test pattern generation and go for sensitize propagate and justify approach that is, this part of the curve that is. Now, we have seen that, for a, whatever be the complexity of the random test pattern generation, let us call it say, complexity of say Q.

,

(Refer Slide Time: 01:57)


Design Verification and Test of
Digital VLSI Circuits
NPTEL Video Course

Module-VIII
Lecture-IV
Testability Measures (SCOAP)

$Q\cdot(n)$
$Q(n-k)$
$Q(n-2k)$
10
10
10
NPTEL

Then, you have to multiply, in the first iteration you have to consider all the faults so, let us say that, we have some n number of faults. Now, again the complexity is Q and say it has detected k number of faults in the first iteration. Then, you can say that, next the complexity is n minus k, in the following iteration, which is the n minus say, 2 k. So, in next iteration also that, 2 k faults are there, we have to dot dot dot dot and say in the n, 10 faults remain, which are difficult to test and cannot to take to be random patterns.

Now, you see, this 10 faults we are considered here, this 10 faults we are considered here, this 10 faults we are consider here, dot dot dot. So, for all the iteration, the difficult to test faults were considered for random test pattern generation and the problem that happened was, that could not be detected. So, in the first go say, around there were n pattern, so n faults, so there were 100 faults say, n equal to 100. Then, when the first pattern you apply and you verify for 100 faults, this is (()) patterns 20 faults say gets detected.

Then, the next iteration you go for 80 and but the major that is, after every iteration, some of the faults are dropped. So, they are easy to test faults and there are adding a advantages to your random test pattern generation algorithm. But, you see some of the faults remain, the difficult to test faults remain and these faults keep on linger in all the iterations of your random test pattern generation.

,

(Refer Slide Time: 03:15)

Design Verification and Test of Digital VLSI Circuits NPTEL Video Course

Module-VIII

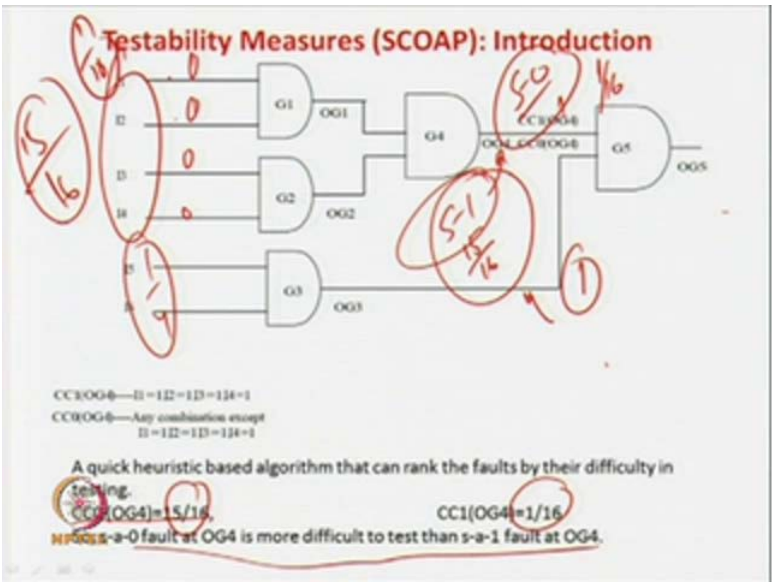
Lecture-IV

Testability Measures (SCOAP)

Easy
diff
NPTEL

So, we have thought that, if we can find out some algorithm or some heuristic, which can say that, these are easy to test fault and they are difficult to test fault. So, what we will do that, this difficult to test faults will not at all bring it into the paradigm for random test pattern generation. We will try with only easy to test faults and difficult to test faults, we can keep in a separate beam and go directly for sensitize propagate and justify approach.

(Refer Slide Time: 03:38)



Testability Measures (SCOAP): Introduction

CC1(OG4)—I1=I2=I3=I4=1

CC0(OG4)—Any combination except I1=I2=I3=I4=1

A quick heuristic based algorithm that can rank the faults by their difficulty in testing.

CC0(OG4)=15/16, CC1(OG4)=1/16

s-a-0 fault at OG4 is more difficult to test than s-a-1 fault at OG4.

So, what this will help, this will actually help in, I am optimizing the random test pattern generation algorithm because, you will be not trying in every iteration, the faults which

,

linger, that are difficult to test fault. Now, let us see, how can we find out some kind of algorithm or a heuristic, that contain you which the easy faults and which the difficult faults. So, let us start with an example so, these are simple circuit so, let us consider that, let me just tell you some formal notations will be using.

So, when we say that, C C 0 that is actually call combinational controllability of 0 and if you say, C C 1 then, we say that is combinational controllability of 1 then, we also put a line l that is net l. So, if we say that, C C 0 of O G 3 so, what does it mean, it will mean that, how difficult is to get this line to get the value zero combinational controllability of 0 , for this output, the name of this output is this one. So, if we say that, C C 1 so, if we say that C C 1 O G 3 that means, how difficult is to make this line to 1 and this net.

So, these are the two notations we are generally going to use and then, will add one more that will tell you later. So, let us consider this circuit now, select at this net say, we are considering a stuck at 0 fault and a stuck at 1 fault. Now, you just try to find out, which is the easy fault to test and which is the difficult fault to test. So, let us first consider that, we are going for the stuck at 0 fault so, if you are allowing a stuck at 0 fault then, you have to apply a 1 over here.

So, that is what is there and then also, we have to apply a 1 over here then, you can observe the value here correct so, this is your… So, if this line is stuck at 0 so, you will get the answer 1 0, this is 1 and the answer is 1 0 and you can detect it. So, in fact, you have to apply a 1 here and you have to apply a 1 here now so, you can say that, difficulty of one getting this stuck at 1, stuck at 0 fault here is combinational controllability of 1, you have to get at O G 3.

These you have to tell that, you have to make this line 1 and this line also you have to make it to 1. So, it is combinational controllability sorry combinational controllability of O G 4, because of this get and you have to control it to 1. So, if you add these two that is, you have to control these to 1 that is, combinational controllability of O G 3 plus combinational controllability of O G 4, this one. If you add it then, actually you can get the value of, what is the difficulty of testing this one.

So, let us see, just try to give some estimate so, in this case, you have to get a 1 so, what are the possible inputs here 0 0 0 1 1 0 1 1. So, how difficult is to make it to 1, it is you can add a number, that the difficulty in controlling this one is say, 1 by 4. Because, only

,

one because among these four patterns, you can apply only one of them will success in making this one as a 1. So, you can say that because, 1 1 is require to test it, make it a 1 so, you say that, 1 1 and say, you can say that, combinational controllability of 1 to O G 3 is actually one forth because, only one of the four combinations can do this one.

Now, in this case, you have to apply a 1 over here so, you see 1 2 3 4, four lines are there so, 16 possible combinations are there. And when you can get a 1, you can get 1 2 3 4 so, only 1 by 16 of a pattern can get your 1 over here. So, you can say that, it is 1 by 16 plus the order of difficulty is 1 by 4, something like this correct. So now, you can say that, for a stuck at 0 fault here so, for a stuck at stuck at 0 fault here, you remember that these are combinational complexity, 1 by 16 plus 1 by 4.

Now, let us try to do the same thing for this stuck at 1 fault kind of a thing, in the same net so, in the same net, we are going to do that. So, same net we are now, going to take a stuck at 1 fault, stuck at 1 fault means you have to apply a 0 over here obviously, this is controllability. So, this is you have to apply a 1 at this net and this is the stuck at 1 so, it is the answer is 1 slash 0 and the 1 slash 0. So now, to the how how difficult is to take a stuck at 1 fault here, it is 1 by 4 because, this line has to be made a 1, which is similar plus give to make this net 0.

So, again these 4 inputs are there so, there is 16 combination and 15 out of them that is, 0 0 0 0 to 0 1 1 1. Actually accepting 111, 1111 will give me the 1, you cannot does it accepting these combination, all other 15 combinations can give a 0 over here. So, you can say that, the difficulty of controlling or how you can get the value is 15 by 16 and 1 by 4. So, this 1 by 4 is common to both the stuck at 0 fault here, a stuck at 1 fault so, let us eliminate this.

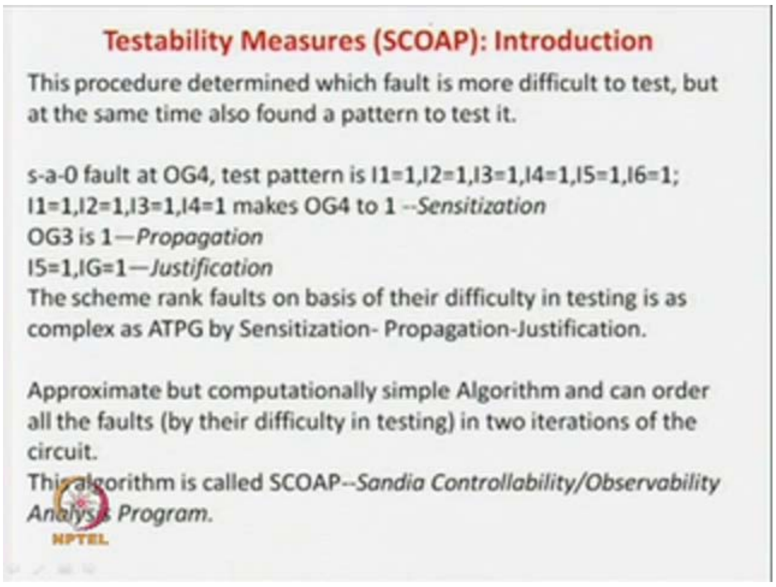
So, you can say that, combinational controllability of O G 4 to 0 is 15 by this one and combinational controllability of difficulty here is 1 by 16. So now, you can see what happens is there to test a stuck at because, this 1 by 4 here, this this this one get value 1 this net is same for all so, let us forget about it. Now, a stuck at 0 fault here needs a application of a 1 so, whose probability is 1 by 16. Now, the apply a there is 1 by 16 now, to test a stuck at 1 fault over here, the probability is 15 by 16 that is, 15 combinations out of 16.

,

So obviously, testing a stuck at 1 fault here is a easier than testing a stuck at 0 fault here so, if I able to apply a random test pattern generation then, I could have say that, you take this stuck at 1 for random test pattern generation. It is easier because, 15 out of 16 combinations can help you that is, and one fourth and one fourth that is an one fourth so, this is common for all. So, out of 15 16, 15 combinations can help you to detect the stuck at fault and only one out of 16 combination can help to detect a stuck at 0 fault, which is a very difficult problem.

So, better for this stuck at 0 fault, you go for a what do you call a, sensitize propagate and justify approach and for the other, you go for this stuck at 0 fault you go for a I mean, for sorry for the stuck at 1 fault, you go for a random test pattern generation. So, this stuck at 0 fault at O 4 is more difficult than a stuck at 1 fault at this one.

(Refer Slide Time: 09:48)



Testability Measures (SCOAP): Introduction

This procedure determined which fault is more difficult to test, but at the same time also found a pattern to test it.

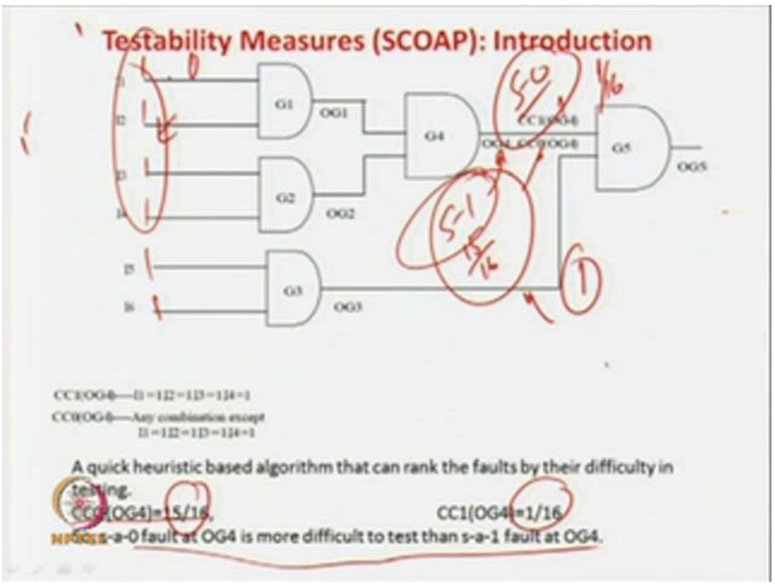
s-a-0 fault at OG4, test pattern is I1=1,I2=1,I3=1,I4=1,I5=1,I6=1;
I1=1,I2=1,I3=1,I4=1 makes OG4 to 1 --*Sensitization*
OG3 is 1—*Propagation*
I5=1,IG=1—*Justification*
The scheme rank faults on basis of their difficulty in testing is as complex as ATPG by Sensitization- Propagation-Justification.

Approximate but computationally simple Algorithm and can order all the faults (by their difficulty in testing) in two iterations of the circuit.
This algorithm is called SCOAP--*Sandia Controllability/Observability Analysis Program.*

So, by this algorithm kind of a thing, you can easily find out that, which is a easier to test fault and which is difficult to test fault. So, this is a basic notion, which is actually I will tell you, what I mean, what do you mean by SCOAP and all. What is the basic notion there, how can you determined that, which is the or you can say that, how to determine, which is a easy to test fault and which is a difficult to test fault. So here, we have found out a procedure but, you also have to observe that, at the same time we have also generated the test pattern.

,

(Refer Slide Time: 10:10)



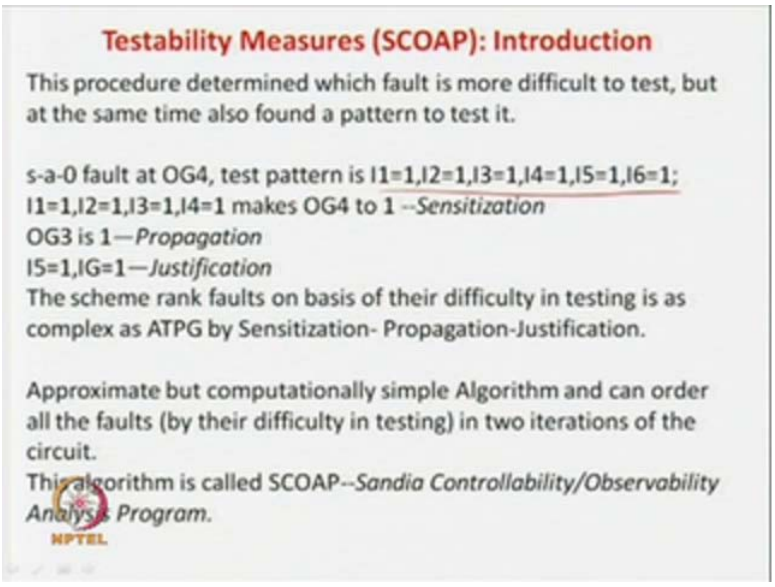
Testability Measures (SCOAP): Introduction

CC1(OG4)—I1=1,I2=1,I3=1,I4=1
CC0(OG4)—Any combination except I1=1,I2=1,I3=1,I4=1

A quick heuristic based algorithm that can rank the faults by their difficulty in testing.
CC0(OG4)=15/16. CC1(OG4)=1/16
s-a-0 fault at OG4 is more difficult to test than s-a-1 fault at OG4.

That is, what we have done, we have said that, you have this is a you have to apply a 1 over here. So, 1 and 1 is the test pattern and here to test a stuck at 0 fault, you have to apply 1 1 1 1, only 1 only 1 combination. And here you have to apply any one for the stuck at 1 fault here, any other combination other than this one.

(Refer Slide Time: 10:29)



Testability Measures (SCOAP): Introduction

This procedure determined which fault is more difficult to test, but at the same time also found a pattern to test it.

s-a-0 fault at OG4, test pattern is I1=1,I2=1,I3=1,I4=1,I5=1,I6=1;
I1=1,I2=1,I3=1,I4=1 makes OG4 to 1 --*Sensitization*
OG3 is 1—*Propagation*
I5=1,IG=1—*Justification*
The scheme rank faults on basis of their difficulty in testing is as complex as ATPG by Sensitization- Propagation-Justification.

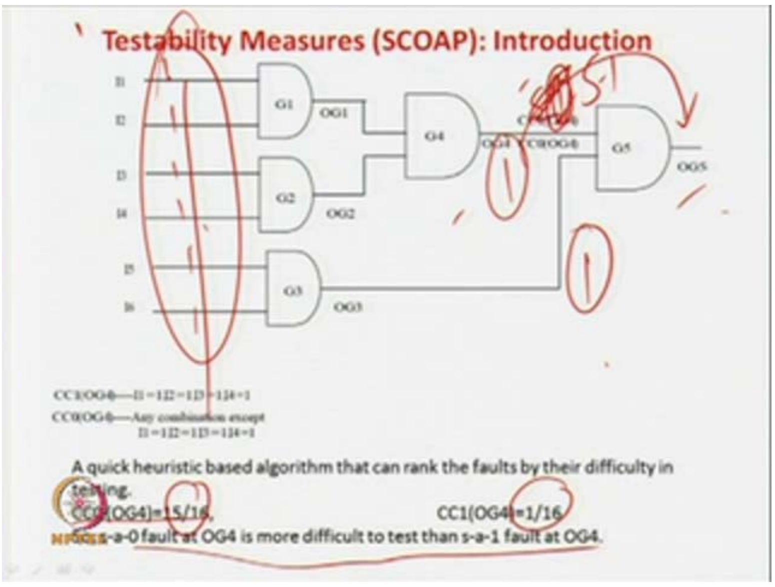
Approximate but computationally simple Algorithm and can order all the faults (by their difficulty in testing) in two iterations of the circuit.
This algorithm is called SCOAP--*Sandia Controllability/Observability Analysis Program.*

So, indirectly what we have done, indirectly we have generated test patterns for finding out which is a difficult to test fault and which is easy to test fault. So, this is not a very

,

good approach so, what we have seen that, for the stuck at 0 fault, the test pattern is this one say, 1 pattern these all 1’s.

(Refer Slide Time: 10:44)



Testability Measures (SCOAP): Introduction

CC1(OG4)—I1 = I12 = I13 = I14 = 1

CC0(OG4)—Any combination except I1 = I12 = I13 = I14 = 1

A quick heuristic based algorithm that can rank the faults by their difficulty in testing.

CC0(OG4)=15/16, CC1(OG4)=1/16

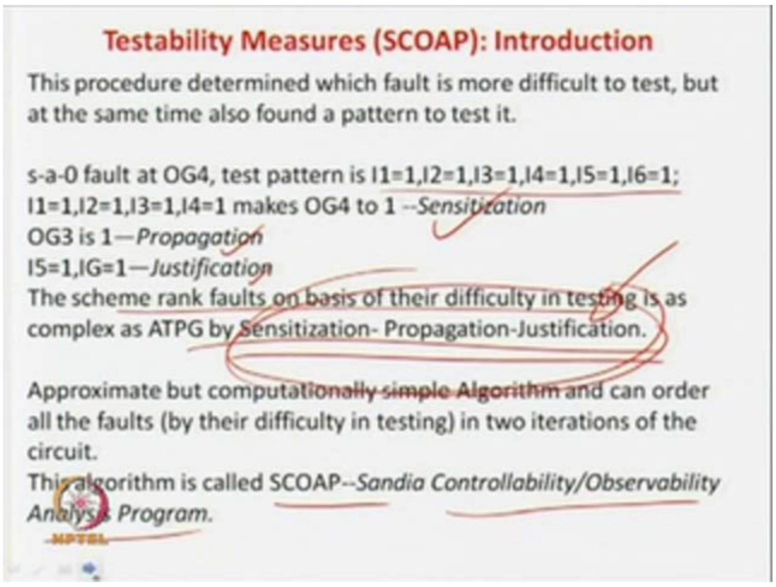
s-a-0 fault at OG4 is more difficult to test than s-a-1 fault at OG4.

Because, we have said that, to test a stuck at 1 fault let us take this case that, why we are saying that to find out which is a easier or which are difficult to test fault, we are applying all the patterns. Say, for this stuck at 1 fault, we have to apply a sorry for this stuck at 0 fault, we have to apply a 1 over here. So, if the answer is 1111 and for this one also we want to be propagated so, you have to 1 1. So, we have generated the test pattern by sensitize propagate and justify approach indirectly.

So, stuck at 0 you apply a 1 so, this sensitize now, this directly propagated to the output so, you have to apply a 1 over here and then, we are saying that, this is the pattern that has to be apply. And for the stuck at 1 0 1 fault so, what we have done, we have say that any other pattern other than this thing will hold.

,

(Refer Slide Time: 11:24)



Testability Measures (SCOAP): Introduction

This procedure determined which fault is more difficult to test, but at the same time also found a pattern to test it.

s-a-0 fault at OG4, test pattern is I1=1,I2=1,I3=1,I4=1,I5=1,I6=1;
I1=1,I2=1,I3=1,I4=1 makes OG4 to 1 --*Sensitization*
OG3 is 1—*Propagation*
I5=1,IG=1—*Justification*
The scheme rank faults on basis of their difficulty in testing is as complex as ATPG by Sensitization- Propagation-Justification.

Approximate but computationally simple Algorithm and can order all the faults (by their difficulty in testing) in two iterations of the circuit.
This algorithm is called SCOAP--*Sandia Controllability/Observability Analysis Program.*

So basically, what we have done so, we have gone for sensitized propagate and justify so, we are ranking of their faults based on difficulty of testing. But, we are also generating the test pattern so, that is not a very good thing we are doing, because of the complexity. So here, we are actually saying that, we are trying to find out, which is a easier fault and which is a difficult set of faults. For that, we are actually going for a sensitized propagate and justify approach is a very complex algorithm.

So, to solve a simpler problem, we are taking a bigger algorithm and trying to solve it so, that is actually not wrong. But here, our main emphasize that, we want to find out a very simple algorithm, may be it is a heuristic, may be it will give approximate results. But, our main goal was to find out that, which are the easier set of faults and which are difficult to test set of faults. Now, by doing this approach, which we have discuss in the last slide, what we are doing, we are doing a ATPG kind of a thing that is, we are using sensitized propagate and justify approach.

And by that, we are finding out which are easier fault and which is a difficult to test fault so, these are the (()). Because, we are generating the test pattern by, we have two algorithms, one is random test pattern, which is easier and which is low complexity and we have test sensitized propagate and justify approach, which involves more complexity. So, what we are doing is, we are using a more complex algorithm to find out which is a easier fault and which is a difficult to test fault, that does not have many meaning.

,

Because, now, our idea was that, we have to find out a very simple low complexity algorithm, which can tell you which is a easy fault and which is a difficult fault. So, difficult faults will through to sensitized propagate and justify approach and easier faults we will go by, what do you call the random pattern generation. But now, this this algorithm in the last slide, it it actually finds out which is a easier to test fault and which is a difficult to test fault.

But, for that you are using an algorithm, which is a very complex one that is, sensitized propagate and justify approach and then, you are finding out which is easier and difficult. So, I mean say that, why you want to classify difficult and easy, you apply sensitized propagate and justify approach and then, everything you will get difficult faults as well as you are going to get the test patterns. So, that high complexity algorithm of sensitized propagate and justify approach you are using to find out the which is a easier fault, which is difficult, it does not solve our purpose.

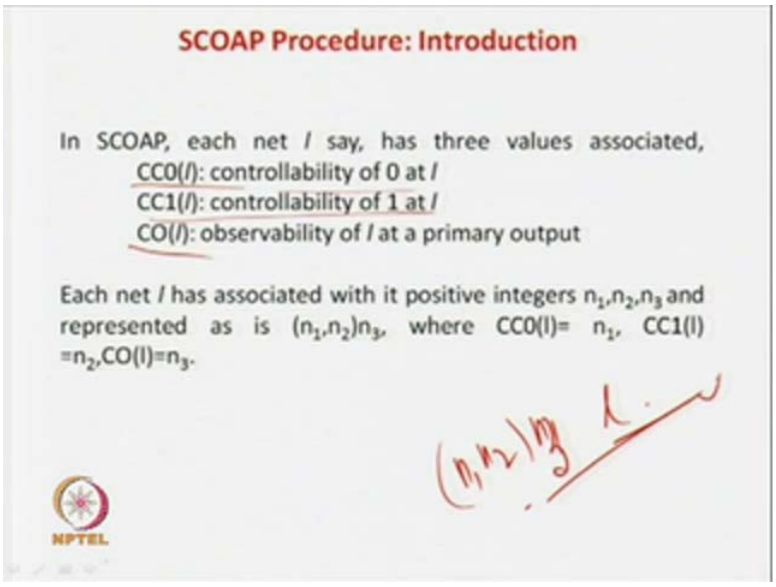
Our purpose is that, we have to use sensitized propagate and justify algorithm as minimum as possible and only for the difficult to test fault. So, we cannot use a complexity algorithm to find out, which is a easier fault and which is a difficult fault. We have to find out some easy to test faults and difficult to test faults using a very simple algorithm or may be but, it may give some approximate result that is possible.

Then, for the easier faults, we are going to use a simple algorithm that is, random test pattern generation and for the difficult to test faults, we can use the complex algorithm for sensitized propagate and justify so, this is the case. So, our so, our idea of we have to find out a simpler algorithm to find out, which is easier fault and test fault. So, we are going to study SCOAP, which is the algorithm, which is Sandia Controllability Observability Analysis Program.

So, it is a very simple algorithm as we will see, we does not required to go for sensitized propagate and justify. What only it will do, it will scan the circuit and it will approximately tell you, which is a easier to test fault and which is a difficult to test fault. But, it may be approximate and give you approximate result, but it can give you a clear idea that, which are easier and try to random on which are difficult, then go for sensitized propagate and justify so now, we will study this this algorithm in details.

,

(Refer Slide Time: 14:27)



SCOAP Procedure: Introduction

In SCOAP, each net *l* say, has three values associated,

CC0(*l*): controllability of 0 at *l*

CC1(*l*): controllability of 1 at *l*

CO(*l*): observability of *l* at a primary output

Each net *l* has associated with it positive integers n_1,n_2,n_3 and represented as is $(n_1,n_2)n_3$, where CC0(l)= n_1, CC1(l) =n_2,CO(l)=n_3.

So, before that, we see some notations so, C C 0 f were already we told you, it is controllability of 0 at net l and how difficult is to control the net l to 0. C C 1 is how difficult is to control the net at 1 and there is one more that is, combinational observability of l. That is, say for example, this is a net l and this is lot of other circuit and this is the output, primary output. Then, C C of O that is, that is C O of O l that is, the combinational observability of l that is, how difficult is to observe this at the primary output.

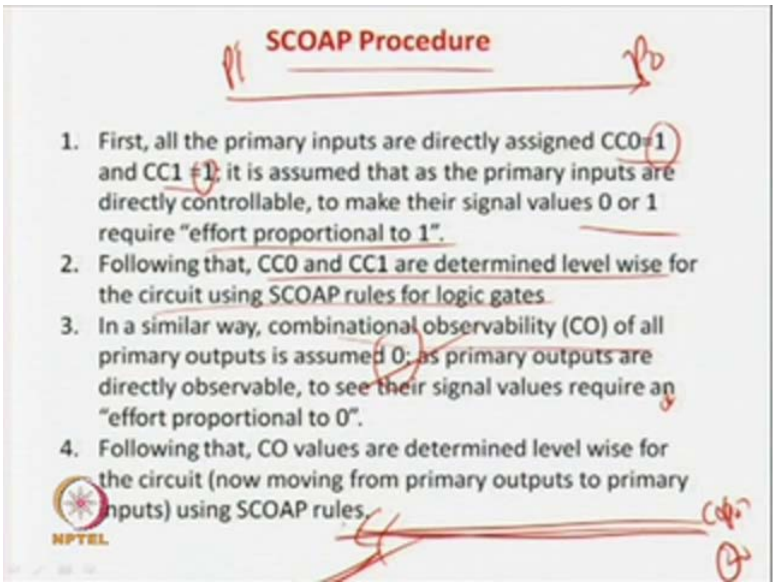
So, to test a circuit say, if it is a stuck at 0 fault say then, you have to apply a 1 over here so, we have to find that, combinational controllability of 1 at l, because if we apply a 1 then, stuck at 0 fault can be tested as well as this l value has to be observed at the primary output that is, C O of l. So, you have to add C C 1 l plus C O of l to get take a stuck at 0 fault. Now, if you want to a test for stuck at 1 fault, what you have to do, if you apply 1 to stuck at 1 fault then, you have to apply a combinational controllable 0 at l.

If stuck at 1 fault here, you have to apply a 0 now, this affect has to be propagated to the output, that you have to observe the value at the output. So, it is combinational observability of l at primary output, if you add them then, you get the difficulty of, how difficulty to take a test a stuck at 1 fault at the net l. So now, sometimes we use a short notation like say, n 1, n 2 and n 3 so, n 3 for a net l.

,

So, we say that, n n 1 stands for C C O of l, n 2 stands for C C 1 of l and C o stands for combinational observability and these are some short notation, which may sometimes use.

(Refer Slide Time: 15:58)



SCOAP Procedure

1. First, all the primary inputs are directly assigned CC0=1 and CC1 =1; it is assumed that as the primary inputs are directly controllable, to make their signal values 0 or 1 require "effort proportional to 1".
2. Following that, CC0 and CC1 are determined level wise for the circuit using SCOAP rules for logic gates
3. In a similar way, combinational observability (CO) of all primary outputs is assumed 0; as primary outputs are directly observable, to see their signal values require an "effort proportional to 0".
4. Following that, CO values are determined level wise for the circuit (now moving from primary outputs to primary inputs) using SCOAP rules.

Now, we discuss the SCOAP procedure so, what is the SCOAP procedure, this SCOAP procedure is first all primary inputs are assigned 0, C C 0 and C C 1. So, what is the idea so, all the primary inputs we assume that, it can be easily controllable. So, the difficulty of controllability of C C 0 that is, making a primary input 0 or primary input are 1, that is the assumption. That is, the assumed at primary signal is directly controllable and their observability effort is proportional to 1.

That is, if you have a AND gate say, these are all the primary inputs then, C C C C 0 is of this input is 1 and also for this one is 1 and controllability of 1 is also what is nets are 1 that is, primary inputs can be directly controllable and difficulty level is 1. Then, what we do, C C 1 and C C 0 are determined level wise using some logic rules. So, in the last lecture, we have seen that, for the detective fault simulation, there are some rules that is, from one level to another level in this circuit if you go, you have some rules.

So now, here will also study some of this what do you call, this combinational circuit rules will also study that is, SCOAP rules that, if you go for one level of this circuit to another level of this circuit, how the rules are applied. That is, if you know that, C C 0 of this net and C C 0 of this net then, what is the value of C C 0 of this net. If you know C C

,

1 and C C 1 of this and what is the value of C C 1 of this net that is, at level level propagation, how difficulty is to control and how difficulty is to observe.

So, all those things will study, that is the rules so now, primary inputs we apply C C of 0 and C C of 1 as 1 1, that is difficult to to control the primary inputs are 1 1 that is fine. Then, you go for the next level of the gates then, you have to apply some rules that will study. In the similar way so, you have to go to the the, similar rule if you apply then, we can go for the primary inputs, the primary outputs, every net will have the value of C C 0 and C C 1.

Similarly, we have to go for combinational observability, combinational controllability goes from primary inputs to primary outputs, this is the level of propagation. But, for the combinational observability, see we go for the reverse way. So, we assume that, the primary output lines are very easy to observe directly, if you prove you can observe so, difficulty level is 0. So, primary input controllability difficult is 1 and primary output observability is 0.

So, you may ask the question that, primary outputs are directly available as the output so, why you make it as 1 and similarly, the controllability, similarly the primary sorry what the question is that, the primary input controllability is 1 1, that is the first statement but, the observability of the primary output is 0. So, you may ask the question thereb why it is like that, that primary input is directly controllable if you are giving a 1 and the primary output directly observable but, you are not giving a value of 1, you are saying that is 0, why it is possible.

So now, here the assumption is there, heuristic is not an what you call, exact algorithm so, the assumption it is there that, to control line, you have to put some effort at least. So, there saying that is, primary input controllability are 1 but, the observation is more simpler, just you provide and you get the value. So, it is say that, observable difficulty is 1 less that is 0, there is just an heuristic you can also try out with controllability level 1 1 at the primary inputs and the observability at the primary outputs as also 1.

You can try out and find out that, you may get another portion of SCOAP but, the analysis is not different. That is, whatever SCOAP with primary input controllability of 0 0 and output observability is primary output observability is 0, the value of I mean, the (( )) of easy to test faults and difficult to test fault, you will get will be very similar you

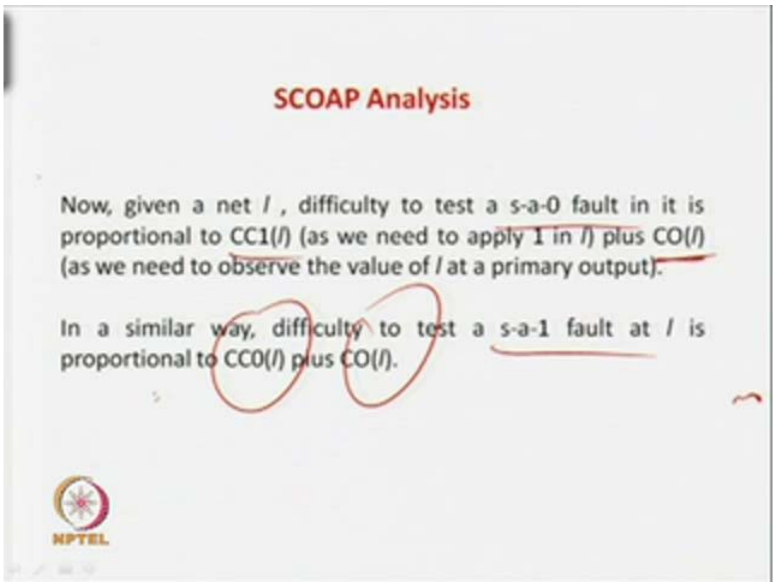
,

apply the logic that, controllability of primary inputs is 1 and also observability of primary outputs is 1.

So, there is not much difference in the results because, there heuristical algorithms and so, therefore, we go by the what is the proposed by the I mean, literature that is, difficulty to control the inputs is 1 and observation is more easier. So, primary output observability is a 0 following that, we know that, the primary output controllability values of all the primary outputs are 0.

So now, it is 0 basically now, we have to come out from this level to this level and we have to apply some rules. And then, you can find out the observability all the nets but, we have to turn out this circuit in the reverse direction.

(Refer Slide Time: 19:54)



SCOAP Analysis

Now, given a net *l* , difficulty to test a s-a-0 fault in it is proportional to CC1(*l*) (as we need to apply 1 in *l*) plus CO(*l*) (as we need to observe the value of *l* at a primary output).

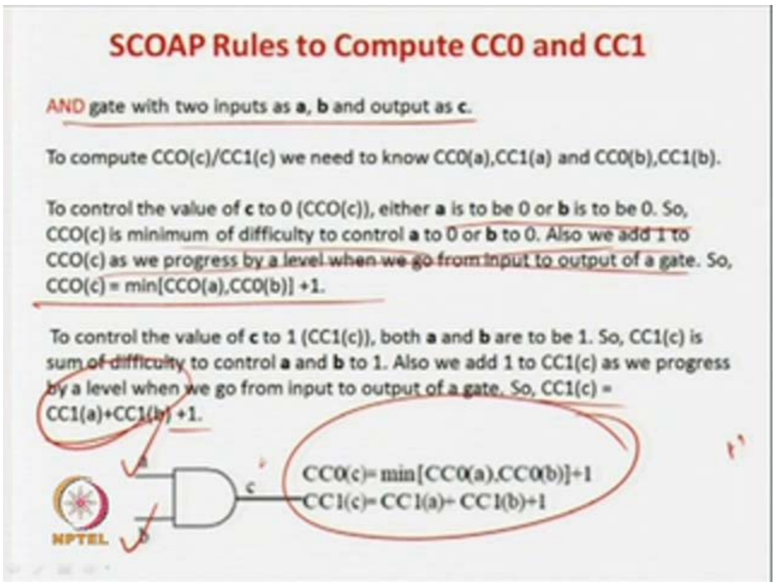
In a similar way, difficulty to test a s-a-1 fault at *l* is proportional to CC0(*l*) plus CO(*l*).

So, once you have traverse the circuit in two directions, this direction which start with combinational controllability of 0 of primary nets equal to 1 and then, we apply rules, rules, rules, rules and at every net will get C C 0 and C C 1. So, here also, C C 1 of the primary inputs are all 1’s and you the finally, for the primary output, you get the all the values of C C 0 and C C 1, you go by this way. And when you are going for the observability part of you, the primary output observability are we say that, primary output combinational observability of the primary output equal to 0.

,

And there are some rules, you keep on applying them and then, you get the value of primary observability of all the nets in the circuit so, primary outputs and the primary inputs. So now, as already discussed so, a given a net, difficult to test a stuck at 0 fault is you have to control the net to 1 and you have to observe the value of the output. So, it is C C 1 of l plus C of l similarly, you were stuck at 1 fault means, you have to apply a 0 so, difficult it to make that net to 0 and you have to observe the net at the output. So, this is a, we can find out the difficulty to test a Stuck at 1 fault and stuck at 0 fault.

(Refer Slide Time: 20:57)



SCOAP Rules to Compute CC0 and CC1

AND gate with two inputs as **a**, **b** and output as **c**.

To compute CC0(c)/CC1(c) we need to know CC0(a),CC1(a) and CC0(b),CC1(b).

To control the value of **c** to 0 (CC0(c)), either **a** is to be 0 or **b** is to be 0. So, CC0(c) is minimum of difficulty to control **a** to 0 or **b** to 0. Also we add 1 to CC0(c) as we progress by a level when we go from input to output of a gate. So, CC0(c) = min[CC0(a),CC0(b)] +1.

To control the value of **c** to 1 (CC1(c)), both **a** and **b** are to be 1. So, CC1(c) is sum of difficulty to control **a** and **b** to 1. Also we add 1 to CC1(c) as we progress by a level when we go from input to output of a gate. So, CC1(c) = CC1(a)+CC1(b) +1.

CC0(c)= min[CC0(a),CC0(b)]+1
CC1(c)= CC1(a)+ CC1(b)+1

Now, we will study about, what do you call these rules because, we said that the primary inputs controllability we know that, they are equal to 1, observability primary output is 0. But now, how to propagate so for that, we require some rules so, let us see of an AND gate with two input output as this one and this one. So now, say for example, these are the rules, see the rules related so now, see how difficult is to control this c value to 0. So, c value can be 0 when, when this is 0 or this is 0, if either of them is 0 so, we know that this value is 0.

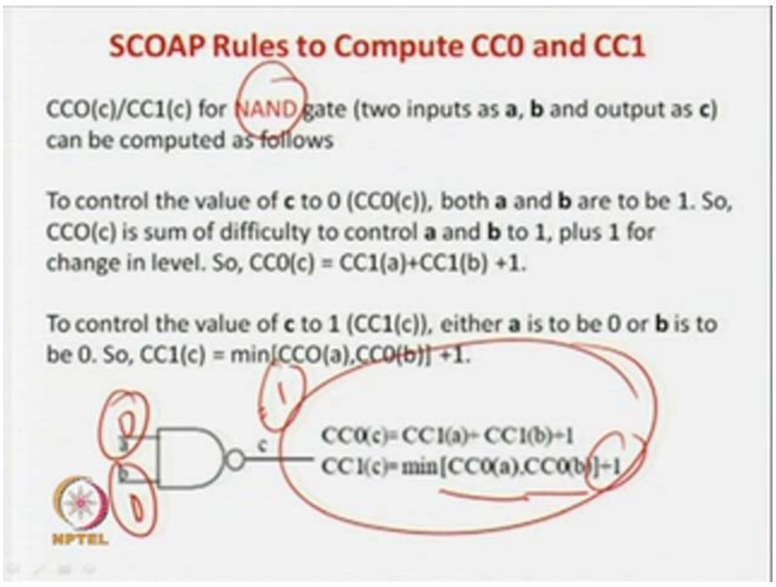
So, how difficulties to make this c to 0 so, it is minimum of combinational controllability of 0 of a or combinational controllability of b to 0 plus we have to add a 1, that is because of the level shift, this is from this level to this level. So, difficulties increasing we assume and we add a 1 but to make a c 0, how difficult it is, it is… If it is to make c 0, we have to apply either a 0 or b 0 so, if you if a is a to 0 is easier then, we say that, we will apply a 0 and we forget about b and the answer is 0. If b is easier to apply to 0 then, we apply b 0 and forget about a then, automatically c will be 0.

So, minimum of the difficulty of making a 0 comma b 0 is the difficulty of controlling c to 0. That is, whichever is easy, a is easier or b is easy to make 0, that will take and we will add 1. That is why, we say that, minimum of combination controllability of a and combination controllability of b that is, easier to make a as 0 or b as 0 is the equivalent to combination controllability of c to 0 and we have to add a 1 because, is a level shift. Now, let us see, what is the difficulty of making c as 1, making c as 1 is a more difficult problem because, a has to be 1 as well as b has to be 1, there is no choice or something.

So, it is how difficult is to make a as 1 this value, how difficult is to make b as 1, those value you have to add because, together you can make c equal to 1. And then, you have to add a 1 because, is a level shift so, this is the rule for a C C of 0 of a AND gate and C C of 1 of AND gate provided. Thus, we know the value of C C 0 of a, C C 1 of a, C C 0 of b, C C 1 of b, that is what is written. To control the value of c to 0, either a to be 0 or b to be 0 so, C C 0 is the minimum difficulty to control a to 0 or b to 0, we add 1 as we (()) level.

So, C C 0 is this one similarly, for C C 1 you have to add there because, both of the a and b has to be 1 and there is a 1 for level shift so, these are the two rules you have to remember for a AND gate.

(Refer Slide Time: 23:37)



SCOAP Rules to Compute CC0 and CC1

CCO(c)/CC1(c) for NAND gate (two inputs as **a**, **b** and output as **c**) can be computed as follows

To control the value of **c** to 0 (CCO(c)), both **a** and **b** are to be **1**. So, CCO(c) is sum of difficulty to control **a** and **b** to 1, plus 1 for change in level. So, CC0(c) = CC1(a)+CC1(b) +1.

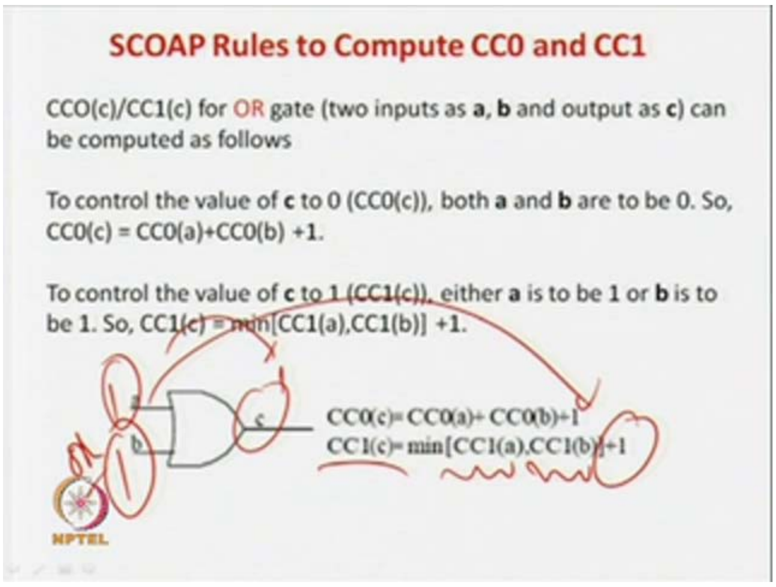
To control the value of **c** to 1 (CC1(c)), either **a** is to be 0 or **b** is to be 0. So, CC1(c) = min[CCO(a),CCO(b)] +1.

0
a
b
0
c
1
CC0(c)=CC1(a)+CC1(b)+1
CC1(c)=min[CC0(a),CC0(b)]+1
NPTEL

,

Now, we will see for a NAND gate so, how can we get a NAND gate to be 0 so, NAND gate is 0 if we have 1 and a 1 over here. So, difficult to do make c as 0 for the NAND gate we are looking at so, it is C C 1 of a, they have to make this one to 1 as well as you have to also make this one to 1. So, both the controllability difficulty you have to add plus 1, there is a level shift now so, this is the value of C C 0. Now, if we have to make c equal to 1 then, anything this is 0 or this is 0, any of them is 0 directly makes this one as 1. So, it is how minimum of this difficulty, minimum of this difficulty plus you have to add a 1 so, this makes your rules for the NAND gate so, you have to remember this rules.

(Refer Slide Time: 24:18)



SCOAP Rules to Compute CC0 and CC1

CC0(c)/CC1(c) for OR gate (two inputs as **a**, **b** and output as **c**) can be computed as follows

To control the value of **c** to 0 (CC0(c)), both **a** and **b** are to be 0. So, CC0(c) = CC0(a)+CC0(b) +1.

To control the value of **c** to 1 (CC1(c)), either **a** is to be 1 or **b** is to be 1. So, CC1(c) = min[CC1(a),CC1(b)] +1.

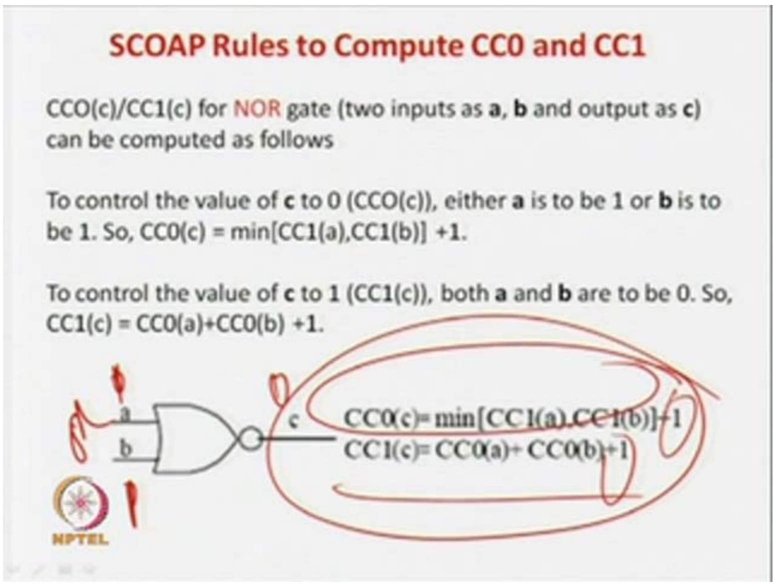
a
b
c
1
1
CC0(c)=CC0(a)+CC0(b)+1
CC1(c)=min[CC1(a),CC1(b)]+1

Now, you go for OR gate, so OR gate is always the dual of a AND gate so, let us see so, you want to get a 0 over here. So, you have to apply a 0 and a 0 over here because, any other combination make a 1. So, the combinational difficulty of 0 at this plus combinational difficulty of 0 at b plus 1 is a level shift, because both of them has to be 0 and 0 so, this is the value of C C 0 and this one. Now, to make C C equal to 1 then, there is any, either this one is 1 or this one is 1, any of them is 1 will get you this one.

So, is minimum of the difficulty of this one to be 1 and or difficulty of this one, if to take the minimum of them and you have to add a 1 for level shift. So, C C 1 of this one is minimum of difficulty to make a as 1 comma difficulty to (( )). So, minimum of them you have to take and 1 you have to add for the level shift so, these are the rules for a OR gate.

,

(Refer Slide Time: 25:08)



SCOAP Rules to Compute CC0 and CC1

CC0(c)/CC1(c) for NOR gate (two inputs as **a**, **b** and output as **c**) can be computed as follows

To control the value of **c** to 0 (CC0(c)), either **a** is to be 1 or **b** is to be 1. So, CC0(c) = min[CC1(a),CC1(b)] +1.

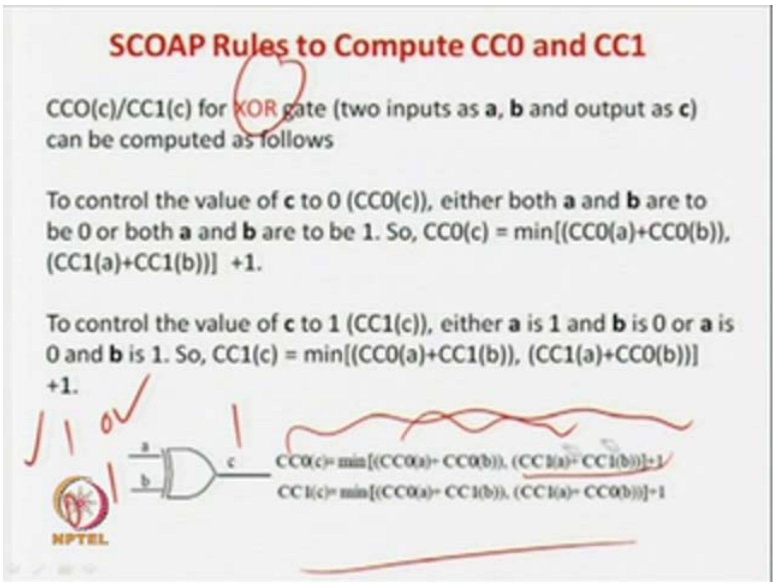
To control the value of **c** to 1 (CC1(c)), both **a** and **b** are to be 0. So, CC1(c) = CC0(a)+CC0(b) +1.

CC0(c)= min[CC1(a),CC1(b)]+1
CC1(c)= CC0(a)+ CC0(b)+1

1
a
b
c
0
1
0
0
0
NPTEL

Similarly, we can go for the NOR gate, so NOR gate whenever you get a 1, you get a 1 whenever you have 0 and 0. So, this is the value and whenever you get a 0, so it is either this one as a 1 or this one as a 1, so this is your rule. So, this is a very simple rule now, you can easily understand and 1 is for the level shift, so this for the NOR gate you have to remember.

(Refer Slide Time: 25:31)



SCOAP Rules to Compute CC0 and CC1

CC0(c)/CC1(c) for XOR gate (two inputs as **a**, **b** and output as **c**) can be computed as follows

To control the value of **c** to 0 (CC0(c)), either both **a** and **b** are to be 0 or both **a** and **b** are to be 1. So, CC0(c) = min[(CC0(a)+CC0(b)), (CC1(a)+CC1(b))] +1.

To control the value of **c** to 1 (CC1(c)), either **a** is 1 and **b** is 0 or **a** is 0 and **b** is 1. So, CC1(c) = min[(CC0(a)+CC1(b)), (CC1(a)+CC0(b))] +1.

CC0(c)= min[(CC0(a)+ CC0(b)), (CC1(a)+ CC1(b))]+1
CC1(c)= min[(CC0(a)+ CC1(b)), (CC1(a)+ CC0(b))]+1

Now, you see the interesting case is for a XOR gate that is, we require some fine time for explanation. So now, what you was there so, in case of the XOR gate, what you see that

,

say 0, we want to get a 0 over here. So, what are the possibilities so, to get a 0 over here so, we can apply a 0 0 over here or a 1 1 over here, any of this two combinations will apply a 0 over here correct. So now, what is the idea, any of them will have 0 so, either if this is here to apply then will get this one, if this is easier to apply then will apply this one.

So, either of these two, which is easier to do will apply and will get a 0 over here but, now, it is the size difference between the OR gate, in case of AND gate AND gate and OR gate. So, what was the idea so, if you want to get a 0 over here so, either you apply a 0 over here or we apply a 0 over here, either of them which is easier can be use to control this to 0. But, that is either this or this any one of them but, in case of an XOR gate, one pattern that is, input pattern at one input is not going to suffice.

Here, to consider the input at both the patterns and then only, you can say that which one is easier to test and which one is difficult to apply. So, that is, in case of AND gate, only one input, this difficulty of this level and difficulty of this level, if you OR it as you add it, you were going to get the answer as first combinational control is 0 and 1 at this output. But, for the XOR gate, it is somewhat different so, one input, you cannot use one input to get a 0 or a 1 directly.

For AND gate or a OR gate, one input if you control, you can easily get a 0 and a 1 at the output by controlling. For 1 actually for AND gate, you have to apply 1 1 but for a 0, you have to apply a 0 then, any of the inputs if you apply a 0, you can get the, where the output. But, for in XOR gate to apply a 0 or a 1 so, never you can say that, one one that is, one input can control this value. So, for XOR gate to get the value of 0 and 1 at the output, both the input values you have to know.

These unlike a AND gate where, one get 0 implies that, this is a 0, OR gate 1 is a 1, this we if you do not know, you know the answer is 1. So, this does not hold for a OR gate that is why, the controllability and observability for the XOR gate is be different so, to get a 0 over here, either of this we apply. So, you say that, C C 0 of a, C C 0 of b that is, difficulty of applying a 0 0, C C 1 of a, C C 1 of b that is, difficulty of apply 1 1.

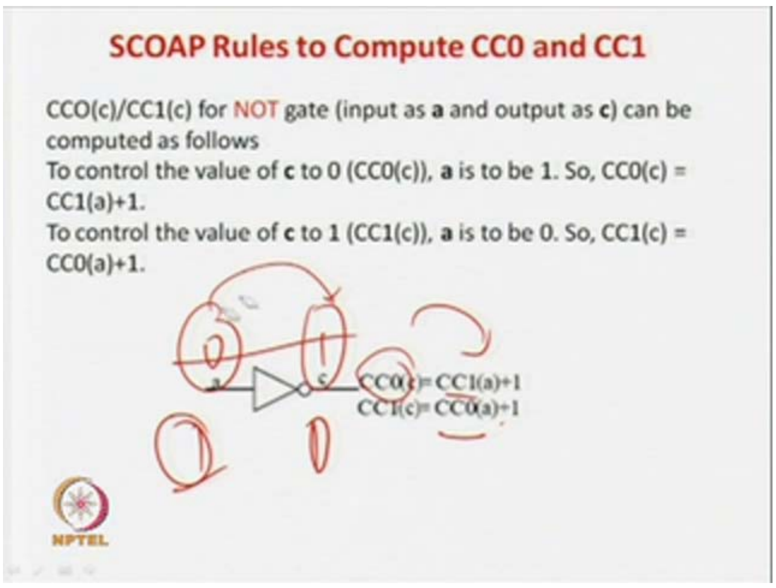
So, minimum of these two so, whichever pair is easier to apply, that minimum you have to take that is, actually can give you a 0 over here plus you add a 1 so, this is the controllability of XOR gate to 0. Now, if you want to apply a 1 at the output, how

,

difficult is it so, there are n two patterns 1 0 and 0 1 then, you get the answer as 1. So, how difficult is to apply the pattern, C C 0 sorry how difficult is to apply this pattern not this one, see this C C 0 of a and C C 1 of b, how difficult is to apply the pattern 1 0, this C C 1 of a and C C 1 of b.

So, easier of these two you will apply plus a 1 you have to go for a level shif.sSo, this for two formulas will give you, what is the a what do you call a it will give you what is the difficulty of C C 0 and C C 1 of XOR gate so, XOR gate is a bit difficult to do this.

(Refer Slide Time: 28:40)



SCOAP Rules to Compute CC0 and CC1

CC0(c)/CC1(c) for NOT gate (input as **a** and output as **c**) can be computed as follows

To control the value of **c** to 0 (CC0(c)), **a** is to be 1. So, CC0(c) = CC1(a)+1.

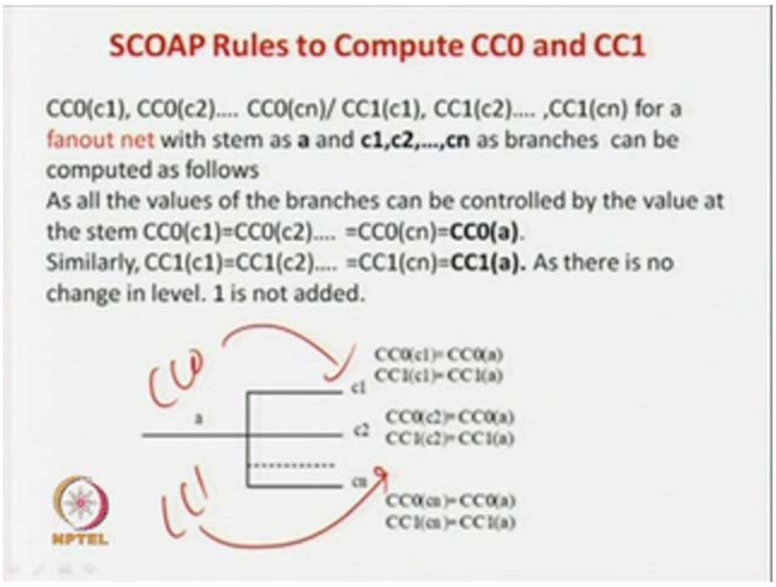
To control the value of **c** to 1 (CC1(c)), **a** is to be 0. So, CC1(c) = CC0(a)+1.

Now, for the inverter it is very simple, you get a 0 over here, you will get a 1 over here so, to get a 1 over here, what you have to do, difficult to apply as 0 over here plus level shift. Now, if you want to get a 1 over here sorry a 0 over here, you have to apply a 1 over here so, C C 0 of this one is you have to apply a 1 difficulty apply so, on a level shift.

So, inverters are very simple, want to get a 0 apply a 1, how difficult to apply a 1 that you find out, add a 1 because of level shift you are done. Similarly, for this case so, inverters are the very simple one and XOR gate, as you can think is the most complex one to think about this things.

,

(Refer Slide Time: 29:15)



SCOAP Rules to Compute CC0 and CC1

CC0(c1), CC0(c2)…. CC0(cn)/ CC1(c1), CC1(c2)…. ,CC1(cn) for a fanout net with stem as **a** and **c1,c2,…,cn** as branches can be computed as follows

As all the values of the branches can be controlled by the value at the stem CC0(c1)=CC0(c2)…. =CC0(cn)=**CC0(a)**.

Similarly, CC1(c1)=CC1(c2)…. =CC1(cn)=**CC1(a).** As there is no change in level. 1 is not added.

CC0(c1)=CC0(a)
CC1(c1)=CC1(a)
c1
a
CC0(c2)=CC0(a)
CC1(c2)=CC1(a)
c2
cn
CC0(cn)=CC0(a)
CC1(cn)=CC1(a)
NPTEL

Now, for a fan out so, say for example, you know that, C C 0 of a we know and C C 0 sorry 1 of a also you know then, if you if you control this two 1 0 then, automatically all these things will be 0. If you make this one to 1 then, automatically everything will be 1 so, so if the difficulty of this net to be control to 0 is C C 0 of a so obviously, the difficulty in controlling this one to 0 sorry we are starting with the 0. So, if you know that C C 0 of a is some value we know then, the 0 is applied then, all the nets will be 0.

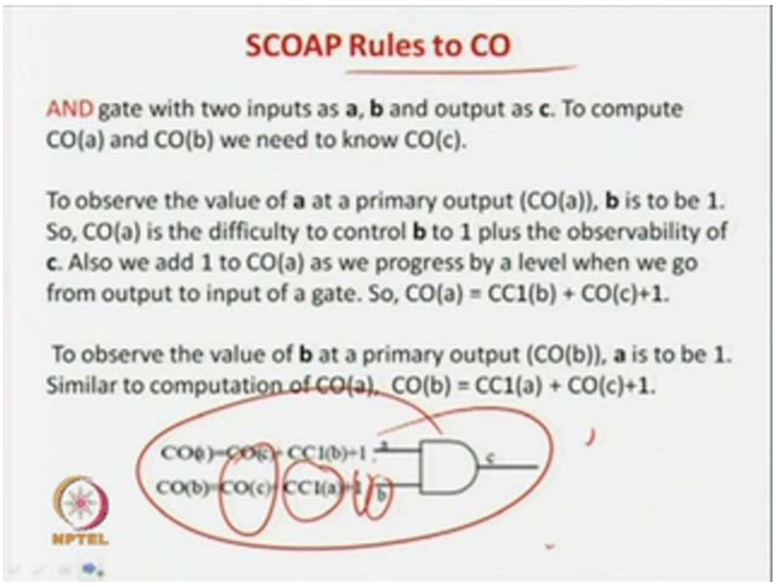
So, if you apply a 0 automatically everything will be 0 so, if we know the difficulty of controlling c c of 0 to C C of a to 0 then, automatically same value apply for c 1 of 0, C C 0 of c 2, C C 0 of c, everything will be same. Because, if you just apply 0, automatically it is a carried over and there is no level shift because, there is no gate so, need not add a 1. Similarly, if you apply a 1 over here then, we say that, C C 1 of a we know, this difficult you know similarly, apply a 1, 1 1 1 1 will go here.

So, C C of 1 of c 1 is equal to C C 1 of a same value, how difficulty is to make a as 1, same value of same. The determinant of difficulty will be will be use to apply a 1 over here, 1 over here, 1 over here because, you are directly apply this 1. You directly getting the value of n, there is no gate inter between so, we did not add 1 so, 1 is not required to be added, this 1 is never required to be added

,

And the next level so, directly you can say that, for the fanouts so, whatever is it difficulty of C C sorry C C 0 over here. The same thing will be applied C C 1, C C value will be applied here because, there is no level shift and nothing is there.

(Refer Slide Time: 31:12)



SCOAP Rules to CO

AND gate with two inputs as **a**, **b** and output as **c**. To compute CO(a) and CO(b) we need to know CO(c).

To observe the value of **a** at a primary output (CO(a)), **b** is to be 1. So, CO(a) is the difficulty to control **b** to 1 plus the observability of **c**. Also we add 1 to CO(a) as we progress by a level when we go from output to input of a gate. So, CO(a) = CC1(b) + CO(c)+1.

To observe the value of **b** at a primary output (CO(b)), **a** is to be 1. Similar to computation of CO(a), CO(b) = CC1(a) + CO(c)+1.

So, still now, we have seen the rules to control a line and also that is a control a line from 0 to 1 and sorry till now, we have seen how difficulty is to control a line to 0 and how difficult is to make control a line to 1, that is the level (()). That is, if you are getting a AND gate and if you know the controllability values of the inputs, how to calculate output OR get, NAND gate, NOR gate, XOR gate all the values we have found out. And if you know the C C 0 and C C 1 of the inputs, you can easily compute the value of C C 0 and C C 1 of the output.

Similarly, we have seen for the NAND gate sorry what the fan outs now, we will see similar rules for CO. That is, if you know that the observability at a primary output, how do you know that, how to control how the know the values of the primary input that is, reverse propagation. So now, let us see this case, say for example, we get the C C 0 primary output, may be whatever so, we know say, that we know that, combinational computability c, this value we know.

If is the primary output is 0 but say, we know the, it will be the not primary input also for the time been but, we just see that we know this value. So now, say that, you know that difficulty of observing this line is this one now, you find out now you need to find out, how difficult is to observe this line in the primary input. So, you see to you know that, to traverse the fault effect from this one to this one, we need to apply a 1 at b correct. So, if you apply a 1 at b so, we know that, this value can be forwarded to the output correct.

So now, what we say that so, to observe this a so, what we need to do, first we need to make the b 1. So, we know that, combinational controllability of 1 to b, that has to be done then, actually this has to be observe. So, if you apply a 1 then, a will be refracted to c correct. Now but, how difficult is to observe c because, a is now reflected from here to here by applying a 1 over here. So, if we say that, difficulty of combinational observabillity of a is equal to, we have to make this b as 1.

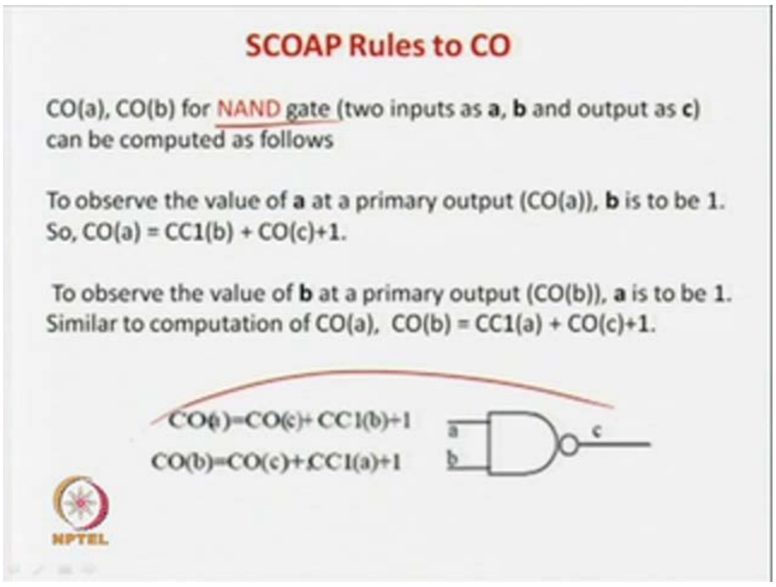
But, by making b as 1, a is now only propagated to c, is not propagated here to primary output. So, how difficult is to move it to a primary output, that we know is that, is equal to combinational observabillity of c. Because, combinational observabillity of c will tell you, how difficult observe the c at the primary output. So, we say that, combinational observabillity of a is equal to combinational controllability b to 1, you make b as 1, value of a is reflected at c.

And we know that, how difficulties to compute the value of c, observe the value of c at the primary output, we know that is c C C of 0 c. This one plus 1 you have to add because, there is the level shift so, that is the value. Combinational observabillity of a is equal to combinational controllability b to 1, observabillity of c plus 1 is the level shift, that is what is actually said in the line. Whatever explain we said that, to observe the value of a at a primary output, b is to be 1, so CO of a is the difficult to control b to 1 plus observabillity of c because, your observing a through c.

So, we add 1 to CO, as your progressing by a level from output input and so, we have to add this controllability of b, observabillity of c because, you have observing a through c and 1 for a level shift. And we have get the formula similarly, if you want to observes b Ss, what we have to do, we have to make a as 1. So, combinational controllability a as 1 plus you have to observe c because, b will be again observed as c. So, this is there and 1 for a level shift so, these are the rules for a observabillity of a AND gate.

,

(Refer Slide Time: 34:33)



SCOAP Rules to CO

CO(a), CO(b) for NAND gate (two inputs as **a**, **b** and output as **c**) can be computed as follows

To observe the value of **a** at a primary output (CO(a)), **b** is to be 1. So, CO(a) = CC1(b) + CO(c)+1.

To observe the value of **b** at a primary output (CO(b)), **a** is to be 1. Similar to computation of CO(a), CO(b) = CC1(a) + CO(c)+1.

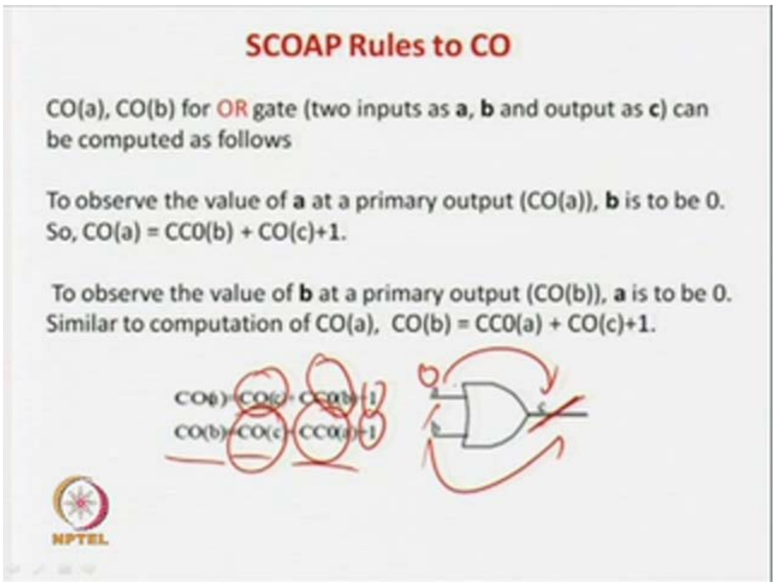
CO(a)=CO(c)+CC1(b)+1

CO(b)=CO(c)+CC1(a)+1

Now, if you look, let us look at the observabillity conditions for a NAND gate so, NAND gate say for example, if you want to observe this a then, what we have to do, you have to observe this as a. We know that, you have to apply a 1 because, if you apply a 0 over here obviously, it is non controllable and everything will be if you apply a 0, everything will be 1 and your whole purpose of test pattern generation is lost. So, we have to apply a 1 so, it is combinational controllable b to 1 is observed by a c. So, this is there and 1 for the level shift. Similarly for b, a has to be made 1, so combinational controllability a to 1 plus you have to observe a b via c. So, this is the case and 1 is added for the level shift so, these are your rules for observing a and b for a NAND gate.

,

(Refer Slide Time: 35:15)



SCOAP Rules to CO

CO(a), CO(b) for OR gate (two inputs as **a**, **b** and output as **c**) can be computed as follows

To observe the value of **a** at a primary output (CO(a)), **b** is to be 0. So, CO(a) = CC0(b) + CO(c)+1.

To observe the value of **b** at a primary output (CO(b)), **a** is to be 0. Similar to computation of CO(a), CO(b) = CC0(a) + CO(c)+1.

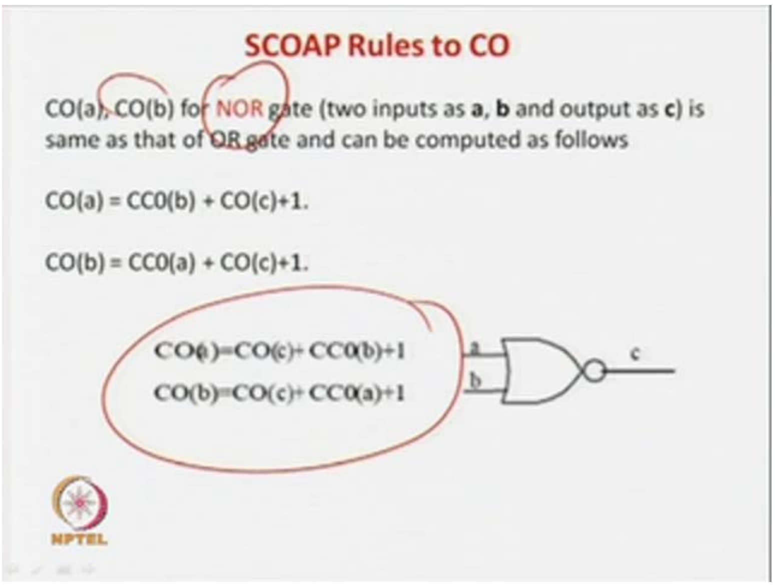
CO(a)=CO(c)+CC0(b)+1
CO(b)=CO(c)+CC0(a)+1
0

For the OR gate, we always now, that it is the dual of NAND gate so, let us see that, if you want to observer a then obviously, b has to be made 0. If b is 1 then, a c will become 1 and the whole purpose of the test pattern is lost. So, combinational controllability of b to 1 you have to b to 0 you have to find out, this is the case. And then, C C o of 0, controllability of c you have to take because, a is observed via c. So, that we have to add and plus 1 for the level shift.

Similarly, if you want to observe b if you want to observe b then, what you have to do? If you want to observe b the output, then a has to be made 0. This is the case, 1 for the level shift you have to do and you have to observe c. We know that, know the observabillity of c that is, c b is observed via c, so combinational controllability observabillity of b is difficult it to observe c plus difficulty to make a as 0 plus 1 for a level shift.

,

(Refer Slide Time: 36:04)



SCOAP Rules to CO

CO(a), CO(b) for NOR gate (two inputs as **a**, **b** and output as **c**) is same as that of OR gate and can be computed as follows

CO(a) = CC0(b) + CO(c)+1.

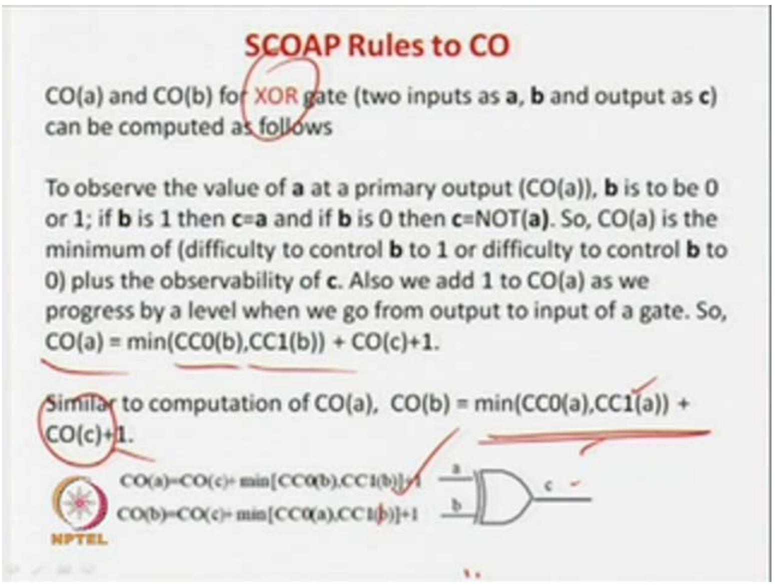
CO(b) = CC0(a) + CO(c)+1.

CO(a)=CO(c)+ CC0(b)+1

CO(b)=CO(c)+ CC0(a)+1

Similarly, for a NOR gate also, you can find out in the same way that, these are the rules for not going to details so, these are the rules for the NOR gate.

(Refer Slide Time: 36:17)



SCOAP Rules to CO

CO(a) and CO(b) for XOR gate (two inputs as **a**, **b** and output as **c**) can be computed as follows

To observe the value of **a** at a primary output (CO(a)), **b** is to be 0 or 1; if **b** is 1 then **c**=**a** and if **b** is 0 then **c**=NOT(**a**). So, CO(a) is the minimum of (difficulty to control **b** to 1 or difficulty to control **b** to 0) plus the observability of **c**. Also we add 1 to CO(a) as we progress by a level when we go from output to input of a gate. So, CO(a) = min(CC0(b),CC1(b)) + CO(c)+1.

Similar to computation of CO(a), CO(b) = min(CC0(a),CC1(a)) + CO(c)+1.

CO(a)=CO(c)+ min[CC0(b),CC1(b)]+1

CO(b)=CO(c)+ min[CC0(a),CC1(b)]+1

Now, we have to again observe carefully for the XOR gate because, XOR gate is the most complex one. Now, let us see, how it can be done say, for example, you have to observe this at this output so now, we all know that, if you if b is equal to 1 then, what is the value of c, c is equal to a bar. Because, we know that, XOR gate is a controlled inverter, if you apply 1 beat as a 1 then, value of you get as a a inversion. So, if you like

,

say, if you are apply a 1 over here and let a be 0 so, 0 and a 1 will get the value of a 1 over inversion so, the explain a prime.

So now, a is 1 then, 1 and a 1 you get a 0 at the output c that is, again equal to a prime so, if apply 1 beat so, that is why, we call what you called XOR gate is the control invento because, if the one input is 1, that the other input actually get inverted. Similarly, if you apply a 0 over here then, c will be equal to a so, that is the idea. So, either you get a or you get a prime, depending on whether you apply a 0 over here, 1 over here. So, c now, both of them solved for purpose because, our goal is that, we have to observe the, you can propagate it here.

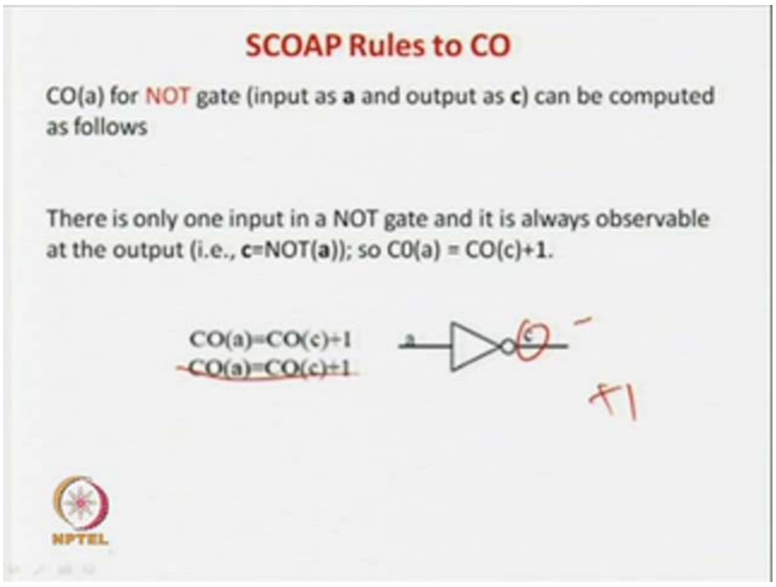
So, we can observe a also, we can observe a prime also, both of them solved for purpose, like this may lead to normal case 0, fault case 1 and this may lead to normal case 1 and fault case 0, both of them we need to our fault detection. So, propagation of a or a prime, both of them these (( )) for purpose so, how difficult is to observe a so, b is either 0 or 1 anything is possible. So, it is minimum of difficult to control minimum minimum difficulty of controlling of b to 0 or b to 1.

So, if it is easier to make b equal to 0, you take b then, you get c equal to a, if it is more easier to get b equal to 1 then, you will get a prime fine. Then, you apply 1, the choice is yours whichever is easier you apply, either you gets c equal to a or a prime. So, minimum of this one 0 or 1, whichever is easier to control apply plus again a will be observable by a c so, that is control observable of c is to there and 1 for a level shift. So, these are somewhat be tricky for the XOR gate compare to other gate observabillity and controllability both.

Now, if you say 1 to observe b of the output so, similarly, if a is 1 then, you get b prime, if a is equal to 0 then, you get b so, both of them is ok for us. So, if a is if a is easier to to make to 1 apply that, if a is easier to we get to 0 you do that. So, minimum of controllability of a to 0 and a to… so, whatever may be the case, we do that so, there is the slight mistake in this figure, just tell you. So, it is written correctly over here so, to control a to 0, it is minimum of C C 0 of b and C C this as 1.

So, this is actually, this is this is proper but, in this case to control b, this has to be a a now, this correct so, these are small type over here just note it. So, this will be actually this one, why because, if you want to propagate b so, b if you a a is easier to control make a, make a equal to 0 if one is easier to get at a, make it easy 1 of a. So, this is the case and 1 for the level jump and it is observable via c so, make this one as this one. So, there is a slight mistake so, this one also will be a correct.

(Refer Slide Time: 39:50)



SCOAP Rules to CO

CO(a) for NOT gate (input as **a** and output as **c**) can be computed as follows

There is only one input in a NOT gate and it is always observable at the output (i.e., **c**=NOT(**a**)); so CO(a) = CO(c)+1.

CO(a)=CO(c)+1

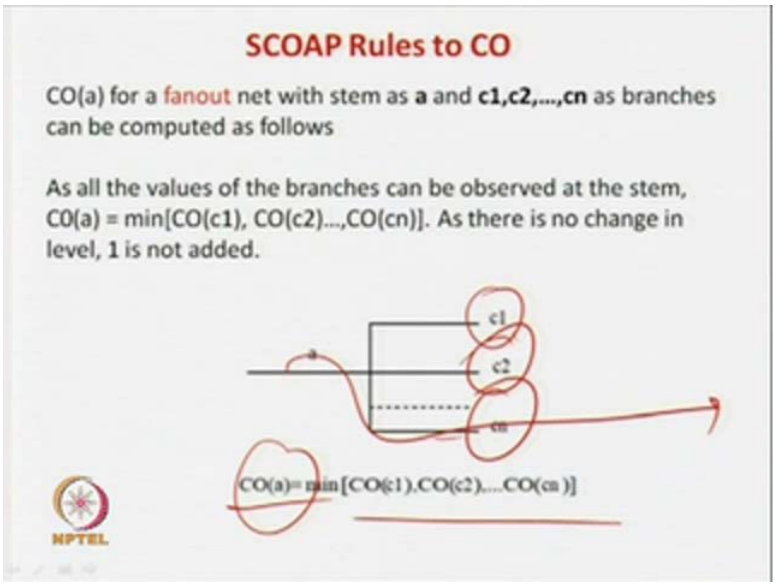
~~CO(a)=CO(c)+1~~

So, this how we go for the computation for a XOR gate now, for a inverter, inverter is the most simplest one so, how difficult is to observe this, this one because, this one you have to observe from here. So, it is very easy, if you will get a equal to 0 so, you will get a equal to 1 and it can be observe. Similarly, if you want to apply 1, you will get 0 and it can be observe so but, only thing you have to remember that, this one is observe by a c. So, difficulty of c is only mattering over here and this is a level job so, if you want to observe a, you have to observe c and plus 1 for the level job so, this is the case.

So, there is nothing, how many surface of a is c of 0 plus 1 and this thing is not true, same rule is over them. So, you have I mean because, for both for 0 and 1, this is you can say for 0 and 1 both the case is validated. Both the case means, if a equal to 0, a equal to 1 so, both the cases the same phenomena observe, (( )) not require. So, you have to observe c, you have to observe a plus 1 for the level job so, this is the most simplest one.

,

(Refer Slide Time: 40:50)



SCOAP Rules to CO

CO(a) for a fanout net with stem as **a** and **c1,c2,...,cn** as branches can be computed as follows

As all the values of the branches can be observed at the stem, CO(a) = min[CO(c1), CO(c2)...,CO(cn)]. As there is no change in level, 1 is not added.

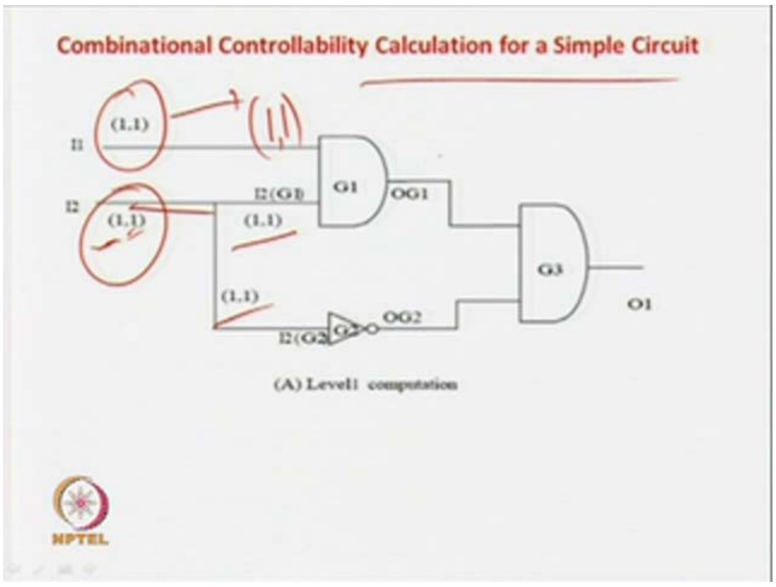
a
c1
c2
cn
CO(a)= min[CO(c1),CO(c2),...CO(cn)]

Now, for the now we go for what, we will go for the fanout so, fanout is the most interesting or you can say, fanout you have to deal in details. I mean, shown in detail by there will be some thought. So, you have to observe this one say, via this, via this, via this now, you think there is the very deep combinational cloud over here, the very deep combination cloud over here. But, here we directly goes to the primary output then, how difficulties to observe the line, how difficulties to observe this one because, you want to observe this.

So, this may be a very complex cloud over here, you have to observe it very difficult, here also some more difficult smaller cloud but, this is directly primary output. So, wherever is the minimum difficulties you have to take back, this is very simple. So, combination observability of a is minimum of observability of this, of this, this also whichever is minimum observability difficulty or whichever is easiest to observe among these three, that one is the case. So, what we write, if you see what we write, we write that combination observability of a is minimum of this, minimum of this or minimum of this, whichever is the easiest to observe, we will take take the output of a via this line so, this is your formula.

,

(Refer Slide Time: 41:57)



Combinational Controllability Calculation for a Simple Circuit

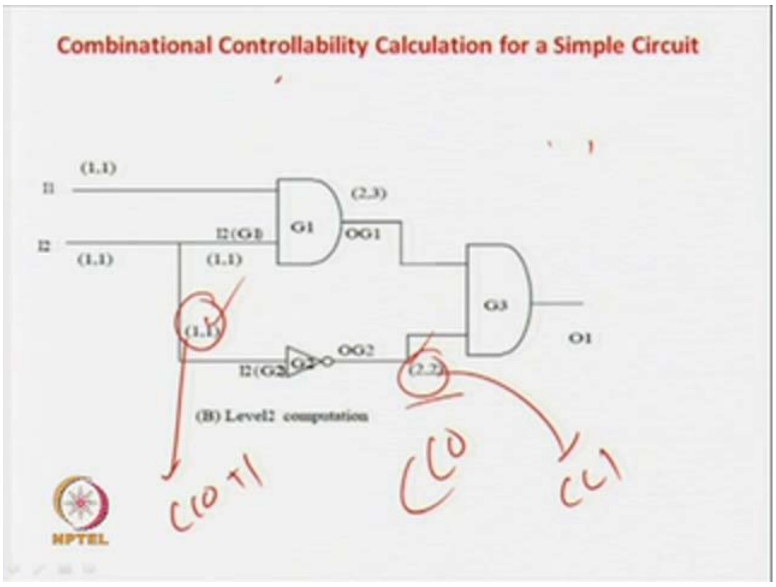
So now, till now, we have seen the rules for what so, we have seen the rules for combinational controllability of 0 1 for all the gates, we have also seen the combinational controllability sorry observability rules for all the gates. Now, what will see now, we will see, we will explode this for an algorithm for a for a simple circuit so, this is an example. So, we let us first see for combinational controllability of 0 and combinational controllability of 1.

So, the notation is 0, 0 means, this one is combinational controllability of 0 and this one is combinational controllability of 1 so, these we write it comma. So, let us see, how how we are going about this so, in this case, you see we write this one as 1 1, why because, making this line 0 or 1 that is, combinational controllability of 0, combinational controllability of a primary input is always 1. So, we write it as 1 1, that was difficult so, this is also 1 1.

Now, we have seen that for a fanout, if you apply 0 over here, everything will get 0 over here. If it is 1, everything will be 1 1 1 so, controllability of the stem is directly reflects over the output so, this is the stem. So, if the controllability of 1 and 0 sorry 0 and 1 are 1 1 so, they directly reflect over it, because whatever value you gives here directly gets it so, this is 1 and a 1. Now, this is 1 1, this thing I have written over here and this is 1 1 so now, how difficult these two so, this is level 1.

,

(Refer Slide Time: 43:17)



Now, you see so, this is 1 1 and this is 1 1 now, let us we will calculate for this and this level. So, how difficulties to get a 0 over here, this can be 0 or this can be 0 so, how difficulties to make this line to 0, 1. So, difficulties to make this line to 0 is 1 so, either of them, if you make as this difficulty you can say, combinational controllability of 0, combinational controllability of 1, both of them are 1 both of them are 1. Now, either this is 0 or this is 0, we will solve a purpose so, minimum of 1 comma 1 is 1 plus, there is a level shift so, we get this result 2.

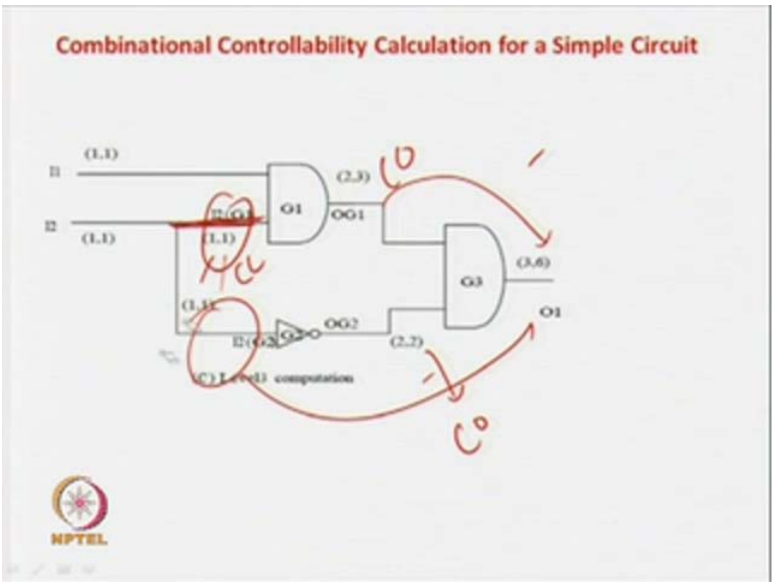
So, that is, one minimum of 1 comma 1, there combinational controllability of 0 at this and combinational controllability of 0 at this that is, this one or this one, minimum you have to take plus 1 for a level shift. So, minimum 1 1 is a 1 plus 1 is a one explicit so, 2 over here so, combinational controllability of 0 at this net is 2. Now, if you have to now let us see combinational controllability of this net to be 1 so, if you want to get this as 1 so, what you have to do, you have to apply a 1 over here and a 1 over here.

So, combinational controllability of 1 at this net is equal to 1 and combinational controllability of 1 at this net is 1. So, what is the, how can you get a 1 over here, it is combinational controllability of 1 at 1 net plus combinational controllability 1 at b net, this is a net plus 1. So, 1 plus 1, 2 plus 1 because, a level shift so, it is 3 so, we get what we get a, simply we get a 3 over here that is, it will combinational controllability of one at this one that is, 1 plus 1 plus this one plus 1 right.

,

Now, for the OR gate so, we want to do it so, we know that, 1 1 is the combinational controllability of 0 and 1 over here. So, what is the combinational controllability of this one, simply this one plus you have to apply 1 1 for the level shift. So, combinational controllability of 0 at this one is combinational controllability of 1 here plus 1 actually, 1 plus 1 is 2. So now, if you want to find out the combinational controllability of 1 here so, it is combinational controllability of 0 here plus 1 that is, 1 plus 1 is 2. So, we get the value of 2 2 so, this solves for the level 2.

(Refer Slide Time: 45:34)



Combinational Controllability Calculation for a Simple Circuit

Now, you have to compute for this one, which is also very simple so, you know that, we want to get a 0 over here. So, to get a 0 over here, you have to apply a either 0 over here, either a 0 over here so, what is the combinational controllability of this net. So, it is combinational controllability of this net is how much so, it is combinational controllability of this net to 0 is how much, it is 2 and combinational controllability of this net is how much, it is also 2. So, minimum of 2 and a 2 is 2 plus 1 for a level shift so, you get a 3 over here so, this combinational controllability of 0 at this net is 3.

Now, say we want to get a 1 at this net so, to get a one at this net, you have to apply combinational controllability of 1 here and a combinational controllability of 1 here. So, what is the combinational controllability of 1 at this net, it is 3 combinational controllability of this net is 2 so, it is 3 plus 2 plus 1, it is 6. So, combinational

,

controllability of 1 at this net is 6 so, this is how, we can find out the combinational controllability of the full circuit by traversing from this level to this level.

Now, you can easily analyze for some facts like say for example, to test a stuck at 0 fault here so, to test a stuck at 0 fault here, we have to apply a 1 over here. So, combinational controllability of applying a 1 here is 6 correct and observing this line, because we are called it is having stuck at 0 fault, as observation of this line does not matter. Because, is same for both, was stuck at 0 and stuck at 1, you can directly observe this now, you say stuck at 1.

So, to stabilize to stuck at 1 fault, you apply a 0 so, in this case, you have to apply a value, the durable level is c 3. Because, to take a stuck at 1 fault, you apply a 0 over here, to take a stuck at 1 fault sorry to test a stuck at 0 fault and we have to apply a 1. So, difficult to applying a 1 is 6, stuck at 0 you have to apply a 0 here, difficult to applying a 0 is 3. So, stuck at 1 fault testing is easy easy compare to testing a stuck at 0 fault at this net, here we have for this net is directly primary output.

So, the observability values are 0 so, we are not added so, this gives you some intrusion similarly, if you can consider this net so, stuck at 0 fault you have to apply a 1, stuck at 1 fault you have to apply a 0, difficulty of both of them are 2 2. So, they are equivalent to test over here because, observability is same for stuck at 0 fault then, stuck at 1 fault are the same. But, say for example, if I say that, I want to compare, what is the difficulty of testing a stuck at 0 fault here and testing a stuck at 0 fault here to sorry say, let if I say that, I want to test a stuck at 1 fault here and stuck at 1 fault here.

So, to do that, you have to apply a 0 over here, how difficult is to add 0 over here, it is 2 now, how difficult is to apply a 0 over at this net, it is again 2. But now so, you can say that, testing at stuck at 1 fault here and stuck at 1 fault is here but, that may not be the case why, because, you have to observe this line to determine a stuck at 1 fault here, you have to observe this line to determine a stuck at 1 fault here. Now, this line and this line are not same, as this lines are not same, so you have to also find out a also you have to add that, how difficulties to observe this line, this combination observability of this one and combination observability of this one.

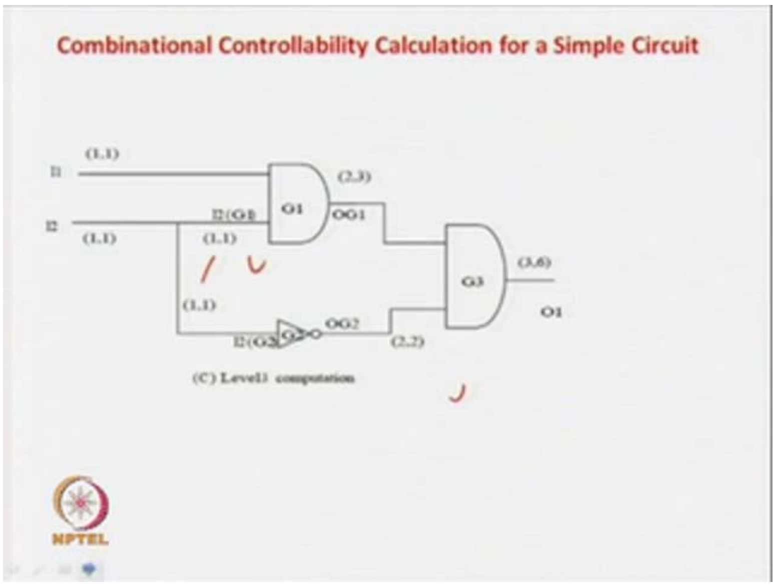
So, whichever will be higher, that fault will be will be difficult to test but now, for example, if you want to find out whether stuck at 0 fault is easier to test at this line or

,

stuck at 1 is easier to test at this line. So, if you want to compare the value difficulty or easiness of stuck at 0 and stuck at 1 at the single line then, obervability may not matter much. Because, same line we are comparing the values so, same line has to be observe so, observe combination observability of this net is same for stuck at 0 and stuck at 1, both.

But, difficulties only coming into matter but, if you want to compare one fault at this net and one fault at this net so, both controllability and observability will matter. Because, difficulty observing at this net and difficulty observing at this net may not be same.

(Refer Slide Time: 49:34)



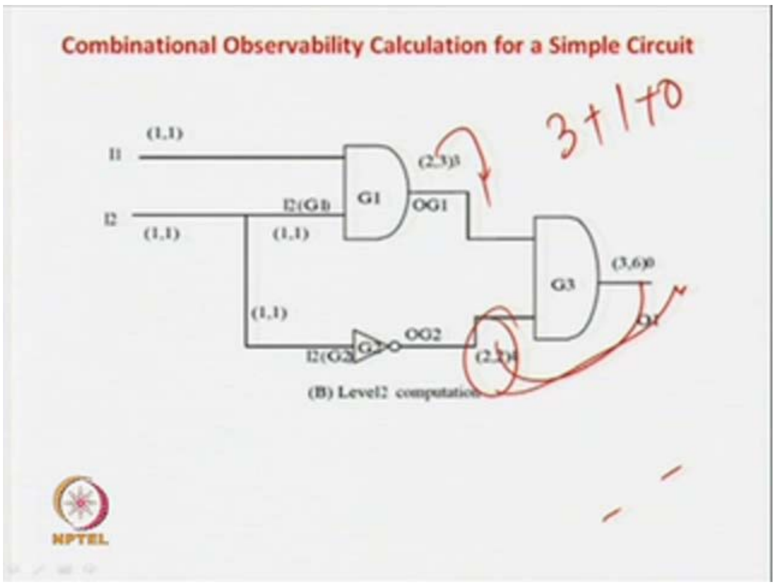
Combinational Controllability Calculation for a Simple Circuit

(C) Level3 computation

So now, we see for the same circuit now, how can we find out the combinational observability that is, we have to look for this reverse propagation. So now, you see observability of this one is 0, primary output now, this is the level 1 now, go to sorry now, we go to next level.

,

(Refer Slide Time: 49:48)



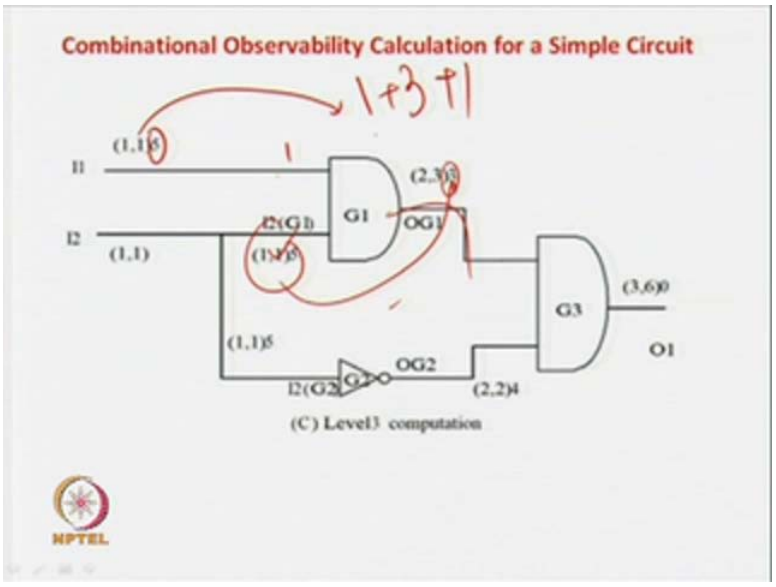
Combinational Observability Calculation for a Simple Circuit

So, next level see, you want to observe this net say, this net we want to observe say so, how difficult is to observe this net so, this net will be observe by this one. So, combinational observability of O 1, that is 0, that 0 has to be added right. Now, if you want to observe this line, this line has to be a 1 so, how difficult is to make this line as 1, this is 2 correct and then, making this line 1 as 2 and then, we have to add a 1 for the level shift.

So, observing this line difficulty is 3 now say, we want to observe this net, this net we want to observe so, how difficulty is to observe this net. So, difficult to observe this net is, is to make this net to as 1, this difficulty is 3 so 3 plus, level shift is there so, plus 1. But, this net you are observing through O 1, O 1 is a primary output so, difficulty observing O 1 is 0. So, 3 plus 1 plus 0 is 4. So, difficulty to observe this net at the output is a 4 correct now, these about level 2, you have done so now, will go to level 4 level 3.

,

(Refer Slide Time: 50:50)



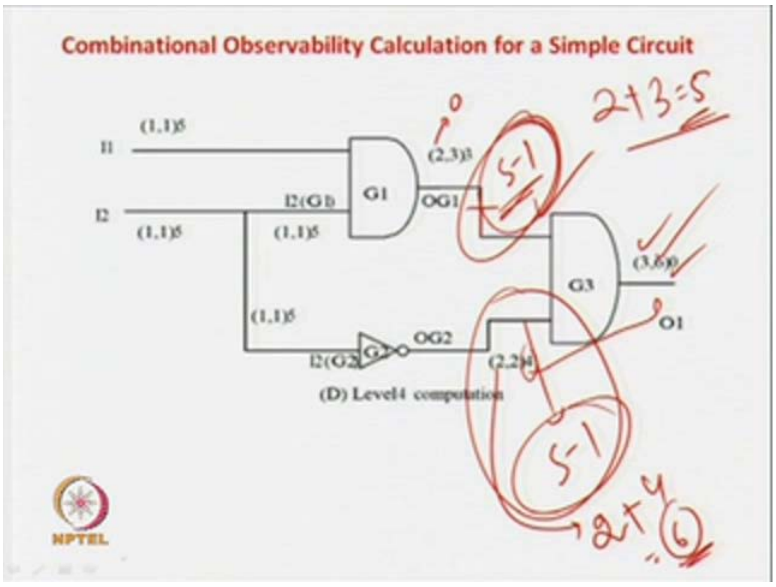
So, say for example, you want to observe this net you want observe this net, here is the observe this net is actually, how difficulty is to observe this net plus one. So, 4 plus 1 is 5 so, this is 5 now, you see that, you want to observe this net so, if you want to observe this net, you have to make this net to 1. So, how difficulty it is, it is 1 then, you have to observe this net via this net. So, this has to be added, the difficulty to observe this net is 3, 3 plus 1 so, how difficulty is to control.

So, to observe this net so, you have to apply a 1 over here so, the how difficulty is to apply a 1 over here is 1 so, you have added 1, how this has to be observed by a this 3 over here and one level jump. So, it is 3 plus 1 plus 1, it is 2 5 so, difficulty to observe this net is 5, this how we can calculate. Similarly, if you want to say, observe this net, this net you have to observe so, you have to make this net as 1. How difficulty is to make this as 1 is 1 then, if you have to observe this net, it will be via this net.

How difficulty to observe this net is 3 so, you have to add 3 plus one level jump 5 so, this net is 5. So now, if you have to calculate for all, accepting this net you we have to calculate.

,

(Refer Slide Time: 52:12)



So, we know that, of observability of this net is 5, observability of this net is 5 so, this fanout can be observe via this and this. So, minimum of them, difficulty of observing via this or via this will be observability of a and here, both them are equal. So here, the value of 5 so, had these been 4 for some reason then, which would have said that, observing this line via this line is easier because, it is 4 and this is 5. So, you would have written a 4 but now, in this case, what of them, both the stem, there is both the I mean, what do you call this, fanout and fanout are 5 so, this stem is will also have a 5.

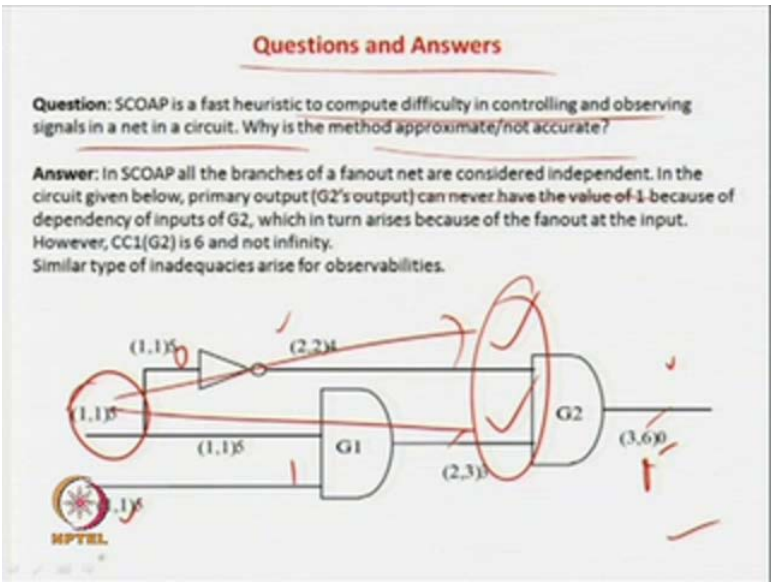
So now, this completes our enumerating the whole circuit with controllability of 0, controllability 1 and observability. Now, you can easily find out which is a easier fault to test and which is a difficult fault to test. Let us say that, we test for a stuck at 1 fault here so, to apply a stuck at 1 fault here, you have to apply a 0 over here. So, difficulty level is 2 and also, you have to observe this at the output. So, how difficulty is to observe this net at the output is 3 so, 2 plus 3 is 5.

So, difficulty for testing a stuck at 1 fault here is a 5 now, we say that, we want to test a stuck at 1 fault at this net to state a stuck at 1 fault at this net is, you have to apply a 0 difficulty level is 2. Now, you have to observe this net at the output so, difficulty to observe this net at the output is 4, so 2 plus 4 is 6. So, you can say that, stuck at 1 fault testing here is, that difficulty level is 5 and here, difficulty is 6.

,

So, this net here is easier fault to test that this net at stuck at 1 so, this net stuck at 1 fault difficulty is 5, this net stuck at 1 fault difficulty 6. So, this may be a easier fault than a this net.

(Refer Slide Time: 53:47)



Questions and Answers

Question: SCOAP is a fast heuristic to compute difficulty in controlling and observing signals in a net in a circuit. Why is the method approximate/not accurate?

Answer: In SCOAP all the branches of a fanout net are considered independent. In the circuit given below, primary output (G2's output) can never have the value of 1 because of dependency of inputs of G2, which in turn arises because of the fanout at the input. However, CC1(G2) is 6 and not infinity.
Similar type of inadequacies arise for observabilities.

So similarly you can find out for in one go, you can find out these a difficulty level of testing the faults by I mean, stuck at faults, you want to stuck at 0 fault if you want to test, it will be combinational controllability 1 plus observability. And the similar way for the, if you want to go for the stuck at 1, other fault stuck at 0 fault, it is combinational controllability of 1 plus observability. Similarly, for all the faults, you can find out in one scan of this circuit and one scan of this circuit.

This will give you combinational controllability values, the reverse can will give combinational observability values. For all cases, you find out what is the level of difficulty of the faults. Now, you can setup a threshold that, whatever is lower than the threshold will be easier to test fault and whatever a difficult to test faults in the higher than the threshold.

Now, based on this remarkation, we can easily find out, which are the easy to test faults and difficult to test faults; easier one, you will apply random pattern and difficult one, you can apply what do you call this sensitize propagated justify approach. But it is a heuristic, so I mean, there can be lot of inaccuracies and also, the heuristic you have to say properly. If you take a very high threshold, then it may happen that, you have a very

very less number of faults in difficult to test series and then, in random test patterns, in the end you might learn in problems.

But if you have a low threshold then, lot of faults will go to sensitize propagated justify approach, then the beautiful layers or the easiness that is given to you by the random pattern generation will be loss. So very careful, you have to set the threshold. Now, we go to the question answer section, in the question answer section, we are saying that, SCOAP is the heuristic, which computes this controllability and observability very fast way.

That means, there may be lot of inaccuracy say, in some inaccuracy, can we so, this (( )) inaccurate. So, can we give some example, why is it inaccurate, can you give me some example? So, in this answer, yes SCOAP is a, what do you called while approximate algorithm, the main reason of this inaccuracies are, it does not it in SCOAP, all the branches of a fanout are considered independent of each other that is, you can have say this is a fanout so, this is one branch, this is one branch.

So, they some difficulty or not but, this difficulty level, this difficulty level, this difficulty and this difficulty level are correlative. Because, this two outputs are dependent on this input because, these a fanout input and these are the fanout branches. Similarly, you can say that, in the same way you can say that, this guy is dependent on this one but because, they all belong to a common transitive fanout point. But to make the things easier, SCOAP does not consider this.

SCOAP considers that everything is independent and we solve the problem, this list to inaccuracy. So, let us see why so, let us see that, combinational controllability of C C G 1, let us find out. So, C C 1 of G 2 so, that is C C 1 of G 2 that is, how difficulties to make this line to 1. So, let us see so input, this is 1 1 primary inputs, the fanout will be 1 1 over here so, these are level shift so, it is 2 2, because in what you just add 1.

So now, in this case of AND gate so, how difficulty is to get a 1 over here, it is a 0 over here and it is 1 1 minimum of them plus 1. So, minimum of 1 plus 1, 1 comma 1 plus 1 is 2, how difficulty is to get a 1 over here. It is difficulty to this one, difficulty to get a 1 over here and 1 over the 1, which is 1 1 in this case, 1 plus 1 plus 3 is there and for the inverter, this one is actually 1 1 so, it is just a level shift 2 2. So, now, we have 2 2 over here.

,

So, how difficulty is to get a 1 0 over here, it is minimum of 2 and 2 plus 1 that is, 3 and again, how difficulty is to get a 1 over here. So, it is difficult to get a 1 over at this one, this is 2, how difficult to get 1 at this net is, 3 plus 1 that is, 6. So, we get the value of, control this net to control this net to 1, we have the value 6 but now, we will see this very interesting that, you can never get a 1 over here. To get a 1 over here, it it is saying that, combinational controllability of this is 1 is 6, that is being 6.

That means, you have to apply 1 over here, you have to get 1 correct now, to get a 1 and 1 over here, you have to get a 1 and 1 over here, to get a 1 over here is a inverter. So, you have to get 0 over here so, to get 0 over here, you have to get a 0 over here so, this is 1 and this is use a control. So, you can never get a 1 over here so, difficulty to get a 1 over here is not 6, it is infinity. But SCOAP will say that, it is having a finite level of difficulty that is, 6 that is not correct because, you can never get a 1 over here.

So, stuck at 0 fault testing at this point is impossible but, still SCOAP will say that, you can test it, the difficulty level is 6, observability is 0 so, it is very good at the level of 6, you can test it. But it should tell you in a accurate way that, difficulty of getting a 1 over here is infinity and that is why, stuck at 0 fault cannot be tested, because of the contradiction (( )). But SCOAP will not be able to tell you this because, these two inputs, as I told you, this input and this input it considers as independent.

But, they are not independent because, these two inputs are dependent on this common transit fanout point, this is dependent on these two. So, that it that it does not consider and therefore, it gives an inaccuracy but is a very fast algorithm because, in one forwards scan and one reveres scan, you can get out from the values for everything, which is very easy way to do and is a simpler algorithm. But sometimes, till there may be inaccuracy then, we have to leave the inaccuracy, because what to get the answer fast.

So, this brings us to the end of this module so, in the next class, we will go to another module where, we study in details about sensitize propagate and justify approach. Because in this lecture, what we have covered, we have covered in details random fault simulation algorithms then, we have seen an algorithm, how to find out which are easier to test fault and which are difficult to test fault, so this was all about random pattern generation.

,

Now, once you have separated the easy faults and difficult faults so, in the next series of lectures or the next module, we will take up algorithms, this is called D algorithm, Ford, Fano some other algorithms are there, which will solve in a formal way, test pattern generation by sensitize propagate and justify approach. So, that will seen in next class.

Thank you.

,