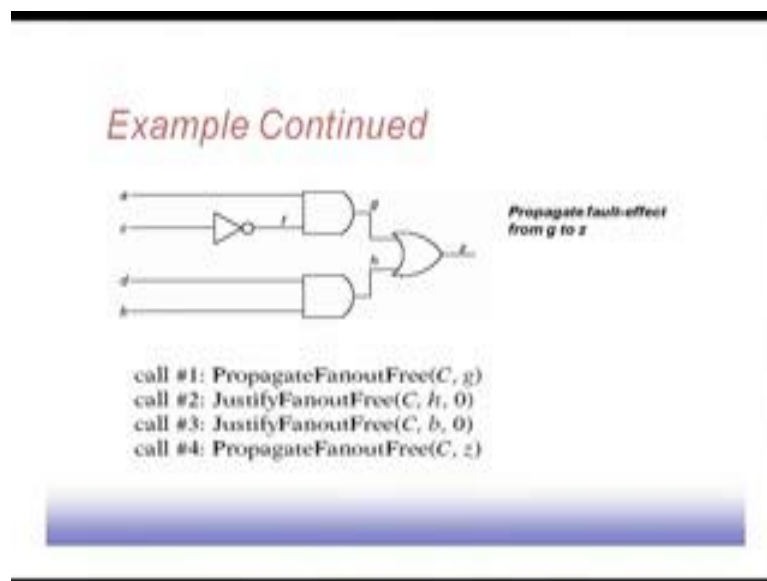


Digital VLSI Testing
Prof. Santanu Chattopadhyay
Department of Electronics and EC Engineering
Indian Institute of Technology, Kharagpur

Lecture - 19
Test Generation (Contd.)

We will continue with the propagation process. So, propagation process, we need to propagate this fault of g to the output z.

(Refer Slide Time: 00:21)




It will propagate, it will call this propagate fanout free C g, g has to be propagated. So, it will see that for propagating g, there is only one fanout. So, naturally, we need to consider that point. So, it will try to find that is OR gate. So, it requires that this h point must be set to 0. So, it will call this justify fanout free routine at C h 0. So, that this h point is set to 0 and that in turn will call this propagate; it will in turn call this justify routine. So, that this line b is at 0. So, it can also pick up line D to be equal to 0, but suppose it as picked up line b as the candidates. So, line b as put the value 0 justify to 0. So, that way it is satisfied, now this it will propel, then this it is passing through this OR gate. So, it will call this fanout; propagate fanout free C to z and that is z is a primary output. So, naturally this will; this is satisfied. So, they then the propagation process ends and we find that is pattern for testing this particular fault g.

(Refer Slide Time: 01:47)

D Algorithm

- Can handle arbitrary combinational circuits, with internal fanout structures
- Main idea: always maintain a non-empty D-frontier and try to propagate at least a fault effect to a primary output
- Initially, all circuit nodes are X, except for the fault site, where a fault effect (D or D-bar) is placed.




Now, we will discuss some more sophisticated test generation algorithms and the first one of in that category is known as D algorithm. So, it can handle arbitrary combinational circuits with internal fanout structures. So, previously whatever technique that we have discussed so that is true that is that can work well if there is there is no funout nodes fanout, there are only few fanout nodes, but if it has got a large number of fanouts then the D algorithm can be used. So, it the idea is that it is maintains a non empty D frontier here.

So, we will define what is a D frontier and try to propagate at least if fault effect we a primary output. So, it will try to propagate some fault effective some primary output and as a result in the process, it will create some more requirement for justification and then this initially all circuits notes at X except the fault site for the fault effect has its place. So, fault effect is D or D bar. So, D value we can put the desired fault so that we can get the complement of that as D bar.

(Refer Slide Time: 02:57)

D-Frontier and J-Frontier

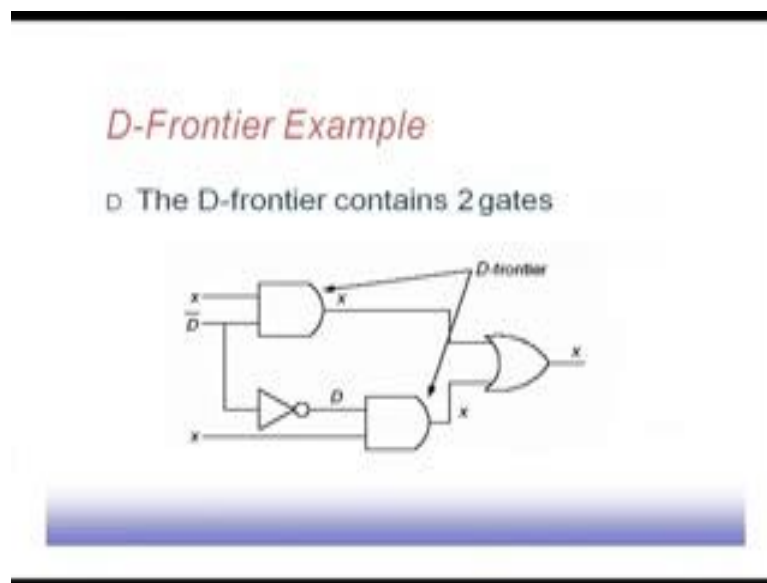
- D-Frontier: All gates whose outputs are X but has at least one D or D-bar at the input of the gates
 - Initially, the D-frontier consists of only 1 gate (output of the fault-site)
- J-Frontier: All gates whose outputs are specified by are not justified by the input assignments



D frontier, these are the 2 terms that you need to define D frontier and J frontier. So, D frontier, all gates whose outputs are X, but has at least 1 D or D bar at the input of the gates. So, that that is called a D frontier, initially D frontier consist of only one gate the output of the faulty site, but then it will in at any point of time. So, D frontier will have all those gates is output are X, but at least one D or D bar input is present. So, that is as if D is going to advance to that point.

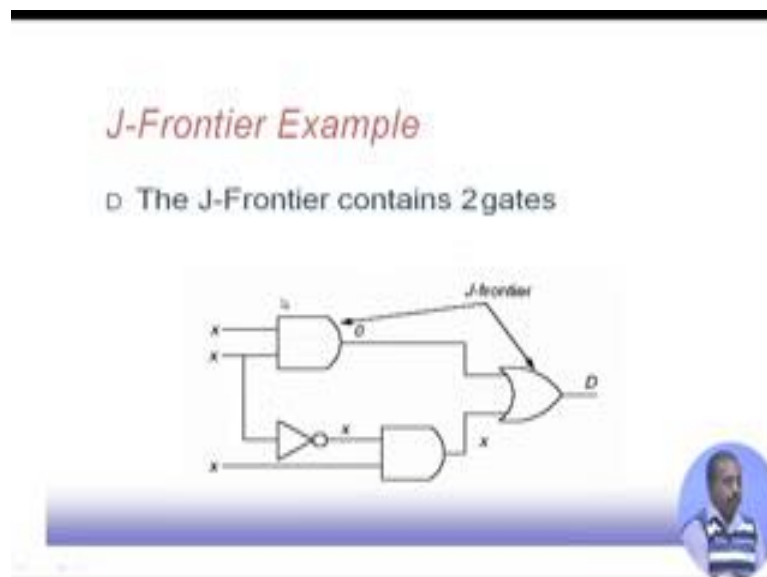
So, that is why it is called a D frontier then we have got J frontier; J frontier all those gates was outputs are specified by are not justified by the input assignment. So, the outputs are known outputs values are known, but inputs are not yet justified. So, input still remains at X. So, that is called J frontier. So, this D frontier and J frontier, they will be propagated in such a fashion that all the inputs of the circuit gates define and the fault gates excited and propagated to the output.

(Refer Slide Time: 04:11)



D frontier in this particular case, you see that this is a gate where we have got this line as D bar and this is a gate where we have got D as one of them. So, these 2 gates, they are at the D frontier. So, because their inputs are either D or D bar outputs are not yet defined. So, this is a D frontier.

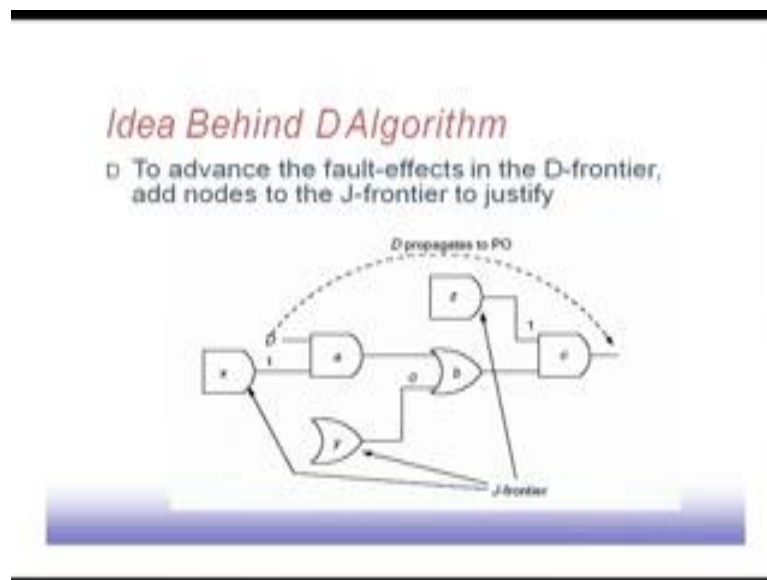
(Refer Slide Time: 04:35)



This D frontier contains these 2 AND gates, on the other hand this is the J frontier. So, J frontier will say that input is that output is specified, it is 0 or 0 1 or D output is specified, but at least some one input is not specified. So, in this, what this particular gate

none of the inputs as specified for this gate one input is specified, but the other one is not specified. So, these 2 gates they form the J frontier. So, basically for J frontier, we need to justify like if we are trying to justify this 0 then we have to see how this X values can be how this inputs can be set so that we can get a 0 here.

(Refer Slide Time: 05:33)



Similarly, this line has to be set to D this output has to be set to D. So, what is required is to set this X to some value so that output becomes D. So, in this particular case is X would be set to D that is true. So, these are the J frontier. So, we have got 2 things, one is the D frontier another is J frontier. So, what is the D algorithm? So, D algorithm, it will try to advance fault effects in the D frontier here. So, what is required is that any fault that has occurred in the circuit. So, that needs to be transported to some primary output through the circuits I need to transport these D to the primary output.

And in the process some other nodes are to be set to some values, so to propagate this D to this output, you see that we need a one at this point. So, this gates becomes J frontier gate because its output is specified, but input is not. Similarly if you take these if you once you are at this point. So, you see that this is the situation for this b has to be set to 0 this input has to be set to 0 because I want to propagate this D through this. So, this particular gates, its output is specified output is specified, but. So, this is forming a J frontier. Similarly at this point, D propagates to primary output back to back this z has z

for the gate z again this in output is at 1, but input is input is not yet specified. So, they remain at X. So, these gates they come to the J frontier.

Basically we are trying to propagate this D frontier to some primary output and in the process we need to we some J frontier gates they get created and those J frontier have to be justified.

(Refer Slide Time: 07:10)

D Algorithm

Algorithm 5 D-Algorithm(C, f)

```
1: initialize all gates to don't-cares;
2: set a fault-effect ( $D$  or  $\bar{D}$ ) on line with fault  $f$  and insert it to the D-frontier;
3:  $j$ -frontier  $\leftarrow \phi$ ;
4: result  $\leftarrow$  D-Alg-Recursion( $C$ );
5: if result  $\neq$  success then
6:   print out values at the primary inputs;
7: else
8:   print fault  $f$  is untestable;
9: end if
```

The D algorithm is like this that initialize all gates to do not cares set a fault effect D or D bar on the line with fault f and insert it into the D frontier. So, the line at which we are trying to test for which we are trying to test the fault. So, we put that line in the D frontier. So, I D or D or any value can be put. J frontier is initially null then it will recursively call this routine as a D algorithm recursions. So, we will the basically this D algorithm recursion what it will do? It is that it will try to propagate these D frontier to the primary output and in the process justify the J frontier. So, we will see this algorithm.

The result is success then we printout values of primary inputs because we have got the fault detector and if the result is unsuccess it is not successful then it is the fault f is declared to be un testable.

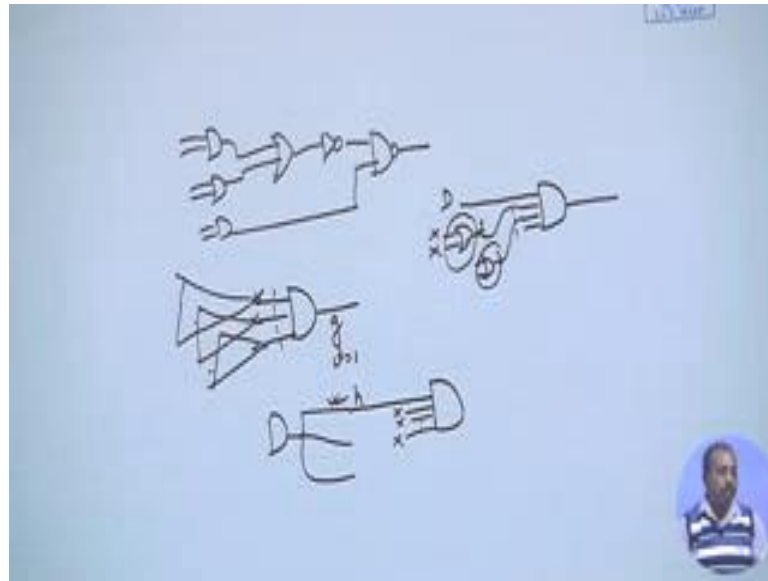
(Refer Slide Time: 08:09)



This is the D algorithm recursion, now if there is a conflict in any assignment or the D frontier becomes null in that case it is a failure. So, if there is any conflict in assignment that is occurring due to this D frontier progress or this D frontier there is no path by which these D can be propagated to the output D frontiers set is null set; that means, we cannot propagate this in the fault to some primary output. So, it is a failure case, second case is that we first propagate the fault effect is a primary output some primary outputs are available to which we can propagate the fault if no fault if it has reached a primary output then while not all gates in D frontier has been tried. So, we try to select a gate and gate g and we try to propagate this D frontier through this gate g .

So, how can I propagate? So, all unassigned inputs of g, there should be set to known controlling value. So, if it is AND gate then all the input should be set to 1, if it is OR gates all the inputs all those inputs should be set to 0. So, like that they should be set to some non controlling value and we add them to the J frontier.

(Refer Slide Time: 09:29)

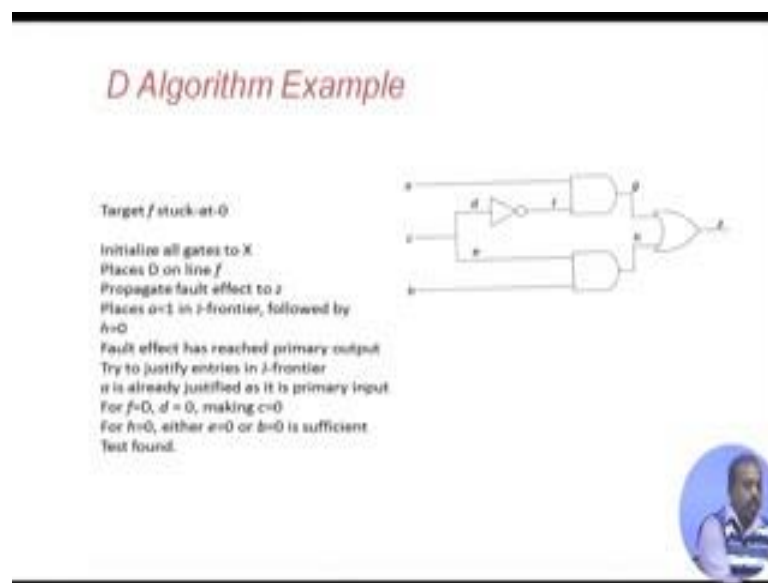


What we essentially mean is this is a gate and this input is at D and you want to propagate to this point. So, what we are doing? So, all these inputs they are to be put to 1; they are to be put to 1. So, if this input is coming from say another gate like this then this gate what happens is that its input are X and output is now becoming equal to 1 specified require required output is 1. So, this gate gets added to the J frontier. Similarly this gate if it is coming from say one and gate then this will get added to 1 J frontier over to J frontier. So, in this all the unassigned inputs of g they will be set to non controlling value and they will be added to the J frontier and then it will call this routine recursively to do those justifications. And after all if return success then this will return success otherwise it will return failure.

This will happen when the fault effect, no fault effect as a propagated to a primary output. So, as soon as a single fault if propagates to some primary output. So, it will come out of this if, so it will at least one primary output has been found to which the fault effect as propagated then we check the J frontier fault. So, the J frontier is null that means, you do not need to justify anymore primary inputs to justify all those propagation through this propagation of this particular fault to the primary output. So, in that case we are successful you have got the test pattern. Otherwise, suppose we have got a gate g which is in the J frontier of, which is in the J frontier.

Now, g has not been justified. So, we take an unassigned input of g that is j and set j equal to 1 and insert j to j equal to 1 to J frontier. So, if try to put j to be equal to 1 and it try to propagate. So, this will again call the recursive algorithm D algorithm recursion. So, for the success thing, this way it will try otherwise try the other assignment. So, you try with j equal to 1 if it fills we try with j equal to 0 we reset it to j equal to 0 and then again try out the other possibility. So, both of them fails then naturally this particular line cannot be set to this particular J frontier cannot be satisfied. So, that is a problem. So, it will return failure. So, this way this D algorithm recursion will D algorithm works.

(Refer Slide Time: 12:03)



We can better understanding it by means of an example, suppose this is the circuit that we have and we have targeting the fault f stuck at 0. So, this particular point stuck at 0. So, initially all gates are set to all gates are or lines in the circuit they are set to X and it replace D on line f . So, this D line is this line f is set to D and rest are all X. Next part is to propagate fault effect to z . So, if you want to propagate fault effect to z then what you need to do is that we. So, this line is d . So, this has to propagate through these g as a result this line has a equal to 1 is added to the J frontier. So, this line a equal to 1 that added to the J frontier. So, that D propagates from f to g .

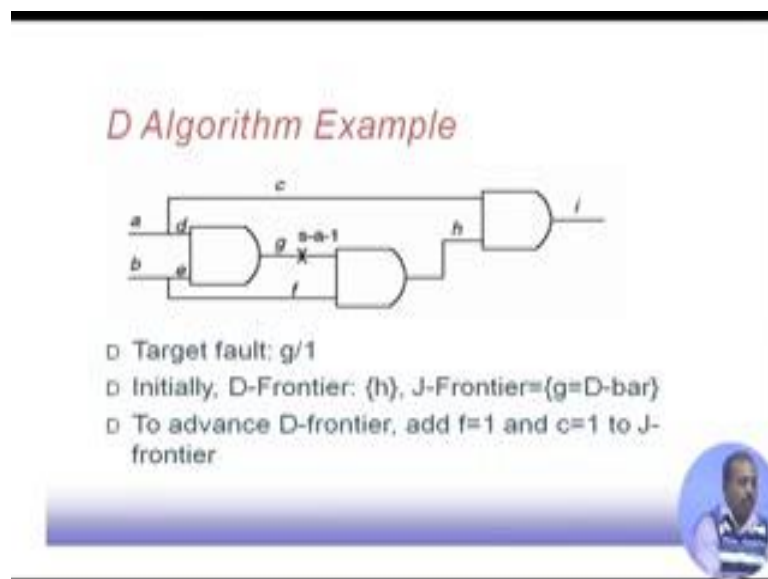
Now, after that these D as to propagate from g to z and for g to z propagation I need the line h to be set to equal to 0. So, that is this h equal to 0. So, that will be added to J frontier. So, at this point of time, we have got a equal to 1 and h equal to 0 has 2 gates in

the J frontier. Now fault effects has reached the primary output z. So, we try to justify the entries in the J frontier since a is already equal to 1. So, a is primary input and we can very easily set it to directly set it to 1. So, a is already justified.

Now for f to be equal to d, so we have to, we have to make this gate was there in the frontier because it is output was D and this is input was X. So, this was there in the frontier J frontier. So, we have to make it to be here to be we have to justify this. So, we have to justify getting your to make D equal to 0 for that we make C equal to 0. So, that way for f equal to d. So, I have to do this thing for h is equal to 0 for h is equal to 0. So, we can do either e equal to 0 or b equal to 0. So, either of them can be done.

So, accordingly, since the C line is already set to be equal to 0 equal to 0 is as equal to 0 is already done. So, you have found a test pattern where a is equal to a equal to 1, C equal to 0 and b can be do not care. So, that is the text pattern. So, this algorithm could successfully find it is pattern for this particular fault.

(Refer Slide Time: 14:42)

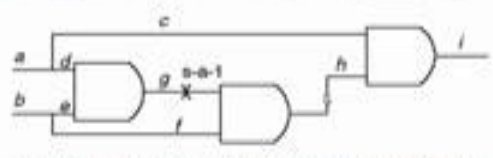


Now, let us try out another circuit. So, where in this case we are trying to C g stuck at 1. So, g stuck at 1 to find out. So, initially it is the D frontier is containing h because this gate has got D as input and h as output is output is X. So, this D frontier in h and J frontier, J frontier is having on this is; this gate is there. So, J frontier here will a g equal to D bar. So, this gate has to be justified. So, J frontier is g equal to D bar. Now if you want to advance this D frontier that is this is h, you want to advance then we have to add

f equal to 1 and C equal to 1 to J frontiers. So, f must be equal to 1. So, that if a propagates through this and this C equal to 1 has to be set so that this ultimate to propagates through this AND gate, they to be added.

(Refer Slide Time: 15:43)

D Algorithm Example (Cont.)



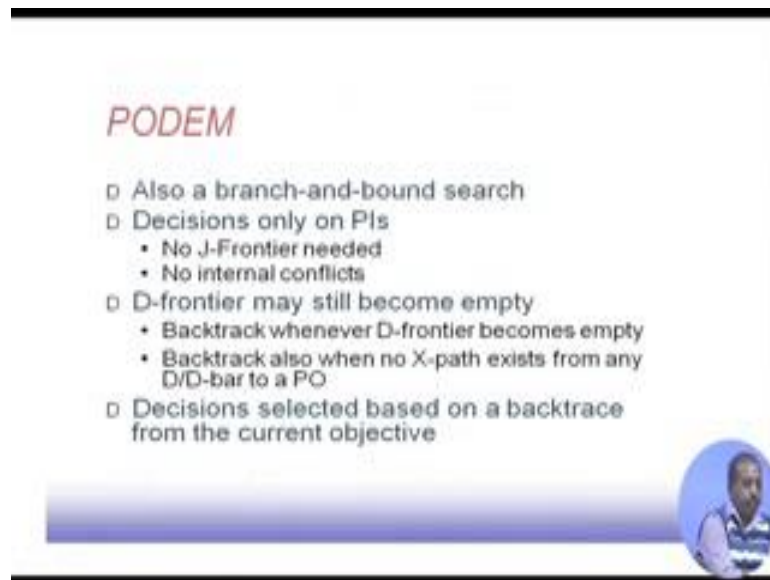
D Now justify every value in J-Frontier via branch-and-bound search

- Must not make D-frontier empty or conflict with other J-frontier values
- Otherwise backtrack

 D Result: g/1 is untestable


If we justify every value in the J frontier, now in a branch and bound such method that we have discussed previously. So, at the same time we must not make D frontier empty or conflict the J frontier value. So, that has to be there otherwise you are in to backtrack. So, you see that in this particular case it is not possible to satisfy all those constantly satisfy simultaneously that is if we want to put say this g equal to I am sorry, this f equal to one that will require that this b has to be equal to 1 then we cannot get a D bar at this point. So, that a justification is not possible. So, in this case, there is conflict will come. So, this g gets stuck at one this becomes untestable fault.

(Refer Slide Time: 16:36)



PODEM

- Also a branch-and-bound search
- Decisions only on PIs
 - No J-Frontier needed
 - No internal conflicts
- D-frontier may still become empty
 - Backtrack whenever D-frontier becomes empty
 - Backtrack also when no X-path exists from any D/D-bar to a PO
- Decisions selected based on a backtrack from the current objective



Next we will look into another algorithm which is also a branch and bound technique which is known as coding algorithm. So, this is here the decision are only on the primary input. So, we do not need to consider any J frontier only primary inputs have to be considered and there cannot be any internal conflicts. So, that way the conflict will occur only a primary input. So, there is no point trying to have a J frontier separately, we can directly, we can directly remember the primary input that needs to be justified for a particular case.

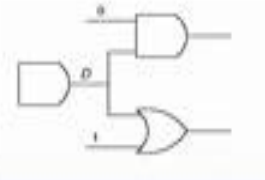
D frontier may still become empty, so that way if you cannot propagate some fault to the outputs. So, in that case the D frontier will become empty. So, we will be that with there will be a situation what the test cannot be found and backtrack will be backtracking will be required in one case D frontier becomes empty otherwise if there is no X path from any D or D bar to a primary output. So, X path means a path where all the inputs are all the lines are X so that we can put it to any of our desired values. So, that is the X path. So, decisions to be selected based on backtrack from the current objective function.

(Refer Slide Time: 17:54)


X-Path

□ The D in the circuit has no path of X's to any PO

- I.e., the D is blocked by every path to any PO



The diagram shows a logic circuit where a node 'D' is connected to a gate. The gate has two other inputs: one is labeled '0' and the other is labeled '1'. The output of the gate is not shown. This illustrates that the 'D' value is blocked from propagating to any primary output (PO) because the path is blocked by the presence of '0' and '1' values.




The D in the circuit has no part of Xs to any of the primary inputs. So, here you see this does D if you want to propagate. So, this input is already 0 and this input is already 1. So, D cannot be propagated to any of the primary outputs so that is why there is a say that there is no X path from to propagate these D to the primary output. So, D is blocked.

(Refer Slide Time: 18:22)

Getting the Objective

Algorithm 9 getObject(C)

```
1: if fault is not excited then
2:   return (g,  $\bar{v}$ );
3: end if
4: d = a gate in D-frontier;
5: g = an input of d whose value is x;
6: v = non-controlling value of d;
7: return (g, v);
```

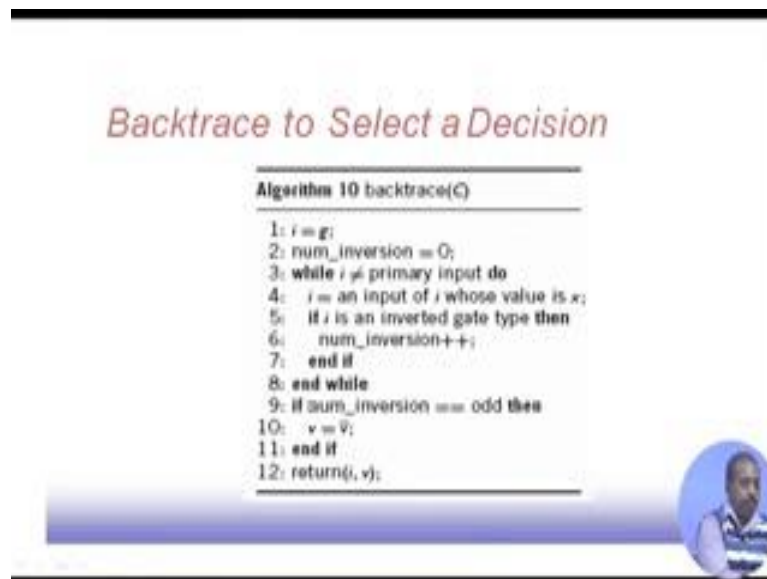


We will generate new object is, what is the, what is done is that a fault is not excited then we return $g \bar{v}$. So, we need to this is a new object that you have to justify. So, the line g put a value at D value. So, if the fault is not excited. If the fault is excited they already

excited then it will come to this part. So, we find a gate D which is in the D frontier and g, an input D is value is X then the objective becomes we have to set this line D to a to the non controlling value v. So, we find out for depending upon the gate that we have so this, what is the non controlling value for the gate and the type of the value that you need to set is v.

Basically what is required is that for gate g we need to put it here value v. So, this becomes a new objective to be justified. So, the fault is already excited now the fault has to be propagated through propagate to the D frontier and for that propagation. So, we need to select we need to determine a path through which the fault can be propagated and for that purpose, we identify a gate and we identify an input value input D is value is X and if you set X to that proper value then that fault will propagate to the towards the primary output. So, this is the new justification that we have to do, this objectives that to be justified.

(Refer Slide Time: 19:58)



Backtrace to Select a Decision

Algorithm 10 backtrace(C)

```

1:  $i = g$ ;
2:  $\text{num\_inversion} = 0$ ;
3: while  $i \neq \text{primary input}$  do
4:    $i = \text{an input of } i \text{ whose value is } x$ ;
5:   if  $i$  is an inverted gate type then
6:      $\text{num\_inversion}++$ ;
7:   end if
8: end while
9: if  $\text{num\_inversion} == \text{odd}$  then
10:   $v = 0$ ;
11: end if
12: return( $i, v$ );

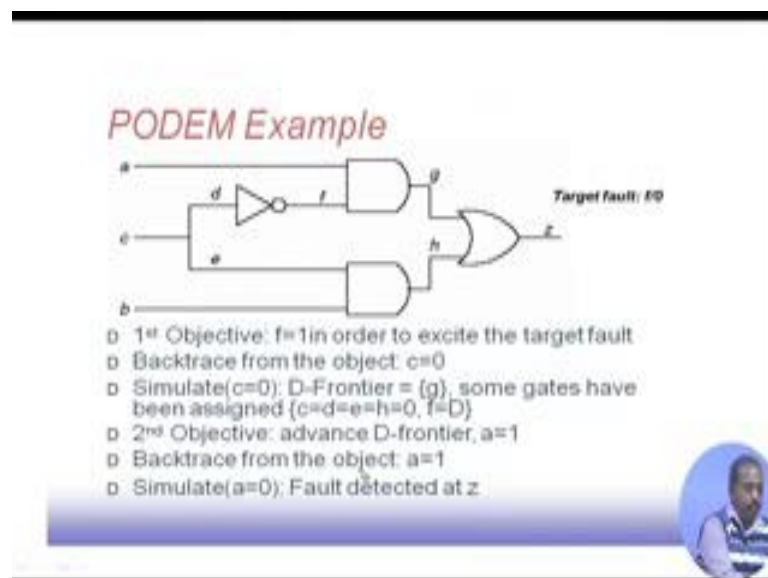
```

Now, this to select decision, it is like this that there is a back trace routine that can be followed. So, i is equal to g that for the gate for which we want to justify. Number of inversions is 0 and if i not equal to primary input that; that means, in that case i is an input of this is, i is modified to be an input of i, its value is x. So, we take up one of the inputs whose value is actually X now. So, that becomes a new i. So, if i is an inverted

gate i then number of inversion is implemented by 1. So, this way to it goes on and ends while and a number of inversion is odd then v equal to \bar{v} .

Basically what we are trying to do is for a particular input we are trying to see how can we reach a primary input corresponding to the, so the primary input can be reached by a number of inverted gates then we can say that number of inversions maybe odd it maybe even. If number of inversions is odd then we need to set the complement of the value and if the number of inversions is even then we need to set the same value. So, number of inversion is odd then v equal to \bar{v} otherwise it will be v . So, i basically for justifying this, i input to some proper to some value we need to justify this thing. So, i to be, this i is a primary input. So, we need to justify this primary input to this value.

(Refer Slide Time: 21:32)

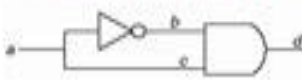


This is an example. So, here we targeted fault is f stuck at 0. So, for f stuck at 0, the first objective is f equal to 1 because we excite the target fault you need to the first generate this particular target that f equal to 1. Now for f equal to 1, we need to back trace from the object C equal to 0. So, that has to be we have to back trace. So, we simulate C equal to 0. So, simulates C equal to 0 that will have D frontier of D frontier to be equal to g because now this line is d . So, this is D frontier is this is this g some gates have been assigned. So, this is this is g c d e and h . So, they are all set to 0 and f has been set to be D . Now another objective is to advance D frontier for that advancing this D frontier up to

set a to b equal to 1 and back trace the object from the object a equal to 1. So, we simulate a equal to 0 and that way the fault is detected at line z.

(Refer Slide Time: 22:44)

Another PODEM Example



Target fault: b/0

- o 1st Objective: excite fault: b=1
- o Backtrace from objective: a=0
- o Simulate(a=0): b=D, c=0, d=0: empty D-frontier. Must backtrack
- o Change decision to a=1
- o Simulate(a=1): b=0, c=1, d=1, D-frontier still empty
- o Backtrack, no more decisions. Fault untestable.


Another example maybe like this that first objective is to excite the fault b stuck at 0 you want beat it. So, excite the fault b equal to 1. So, back trace from the objective so that is from this objective back trace and we get a equal to 0. So, this is the objective that we want to excite would b equal to 1 and 1 is back trace to the primary input we find that a has to be set to be equal to 0 then we simulate a equal to 0, if a equal to 0 is simulate and accordingly we find that b is getting d c is equal to 0 and d equal to 0 and then the D frontier becomes empty. So, we must back track then backtracking will change a to be equal to 1. So, this was the previous objective we see that with this we cannot achieve that thing; the d could not be propagated. So, we back track and change the decision a equal to 1.

Now we simulate with a equal to 1 and we see that b equal to 0 c equal to 1 and the d equal to 1. So, that in that case D frontier still empty because this d could not propagate to the primary output, but there is no more decision to be simulated. So, this fault is an untestable fault.

(Refer Slide Time: 24:01)

FAN

- ▷ Extend PODEM for an improved ATPG
- ▷ Concept of headlines to reduce the number decisions
- ▷ Multiple Objectives to reduce later conflicts



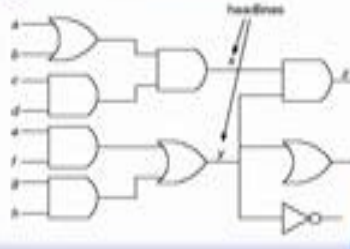
Another extension of this modern PODEM algorithm is the fan algorithm. So, this is fan is much more improved ATPG. So, it actually uses some concept known as headlines. So, headlines are used to reduce the number of decisions and multiple objectives to reduce later conflict.

(Refer Slide Time: 24:25)

Headlines

- ▷ Output signals of fanout-free cones
- ▷ Any value on headlines can always be justified by the PIs

We only need to backtrack to the headlines to reduce the number of decisions



So, what is the headline? So, a headline is nothing but a fanout free cone. So, this is a fanout free cone. So, this outputs that this particular portion, this X, this is a fanout free

cone and we have got this y also another fanout free cone. So, these are the fanout free cones x and y.

What is required is that when you are backtracking, you need to back trace only up to the headlines because once we are here. So, you know that this is a combinational logic. So, this can always be set to the desired value. So, this way we can say that this is can always be simulated to the desired value similarly. So, this is a headline. So, we can continue and come to this point. So, this way we can simulate only. So, whenever you are doing a back trace. So, if we find that we have reached a headline. So, you may do not need to back trace further. So, this is the idea of this fan algorithm. So, we will continue in the next class.