

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

Module - 9
Combinational Circuit Test Pattern Generation
Lecture -2
D-Algorithm-1

Welcome to the next lecture of module number 9 that is on D algorithms. So, but, the last few lectures, what we have seen is that for testing generally the two types of automatic test pattern generation algorithm if you look at it basically. So, that is random test pattern generation algorithm and other is by path sensitization that is by actually path sensitize, propagate and justify. So, by path sensitization method in the last lecture, we have discussed that they can be two types of them.

So, 1 is by Boolean difference and 1 is by the D algorithm that is by path sensitization, propagation and justification. Also then we have discussed that this that Boolean difference I am sorry that is that Boolean differentiation and all these for the very difficult problem in terms of complicity. Because, it is over Boolean difference then this the you have to go for derivatives and all these, so in a little and in number of these patterns that may result can be exponentially natural.

So, due to this exponential large complexity for this Boolean difference based scheme of this algorithms, we have generally shifted to sensitize, propagate and justify approach. Also what the few lectures and in over the last few weeks, we have mainly discussed that we have given several examples of sensitize, propagate and justify in which case, also in the last lecture we have seen one example. In which case what we do we actually this is the (()) pseudo path we apply a 1 and vice versa for (()) or the other way.

And then we propagate evaluate to some output and then we try to justify that that propagation and sensitization are justified by some important values. So, this was actually basically the D algorithm.

(Refer Slide Time: 01:54)



D algorithm are nothing but, sensitize, propagate and justify but, in this see this two lectures we actually have a detail look at them there is a very formal way defining the D algorithm. Because D algorithm as been seen is one of the most prominent or the background or the back bone of digital testing, because in next will see sequential circuits and several other bist architectures and all. For all of them D algorithm is the back bone, from D algorithm they have different other advanced versions of D algorithm like podium and fan.

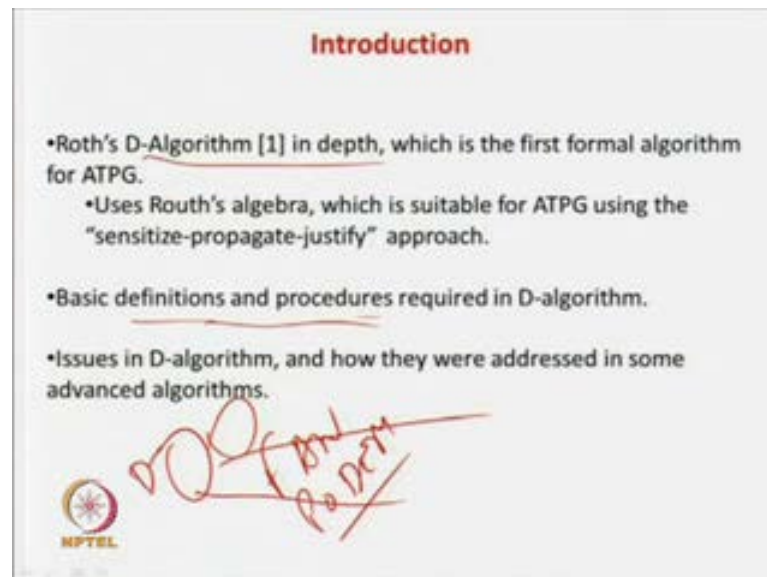
But, the basic bottom line or basic backbone, which we should learn in detail is the D algorithm, because it was the starting point for all sensitize propagate and justify the based D algorithm which still occupies the prominent role in digital testing.

So, in this 2 lectures what will do is that we will try to look at this algorithm here more formally. So, in the last lectures you give one example that for a given that circuit say gate like (\circ) then we apply here one, then we see that in normal case the answer is one the fault case the answer is 0. So, we could have written this as a it could be easily solve the generation problem but, we say that to replace it formally have to say that we represent it by some kind of a symbol like D.

So, that the because in Boolean algebra we cannot have a oblique b 0 or 1 or something like that. So, we have gone for the (\circ) algebra when you have symbol like d x and all. So, what is representing circuit algebra or the or that or circuit having false in a more

formal way. Similarly, for the D algorithm or sensitize propagate and justify with approach, till now whatever examples you have studied and all with the somewhat informal way. In today's lecture and in the next lecture 2 and 3 is not use will see this in a more formal way that is our basic approach.

(Refer Slide Time: 03:29)



The slide is titled "Introduction" in red. It contains four bullet points: "•Roth's D-Algorithm [1] in depth, which is the first formal algorithm for ATPG.", "•Uses Routh's algebra, which is suitable for ATPG using the 'sensitize-propagate-justify' approach.", "•Basic definitions and procedures required in D-algorithm.", and "•Issues in D-algorithm, and how they were addressed in some advanced algorithms." There is a red signature and the NPTEL logo at the bottom.

Introduction

- Roth's D-Algorithm [1] in depth, which is the first formal algorithm for ATPG.
- Uses Routh's algebra, which is suitable for ATPG using the "sensitize-propagate-justify" approach.
- Basic definitions and procedures required in D-algorithm.
- Issues in D-algorithm, and how they were addressed in some advanced algorithms.

NPTEL

So, first we have already seen about these Roth's algebra, so by in that Roth algebra will be used for this what you called this D algorithm. Then before we start discussion on the D algorithm, we see some basic definition and procedure that is very important, then we can go for formal algorithm. And finally, will see that some adverse algorithm like podium and fan.

So, this will see the basic of that they were all advanced versions of D algorithm then but, I already told you that D algorithm was the basic basic from there we have developed. So, many advanced algorithm but, this starting point was that this D algorithm for all these sensitize, propagate and justify their approach. So, see background what is the basic plan of the next 2 lectures.

(Refer Slide Time: 04:10)

D-Algorithm: Definitions and procedures


Definition 1: Singular cover

Singular cover of a logic gate is the minimal set of input signal assignments needed to represent essential prime implicants in the Karnaugh map of the logic gate, both for the case 0 and case 1.

Table shows singular cover for 2 input AND and 2 input OR gate.

AND	Inputs		Output	OR	Inputs		Output
	A	B			A	B	
	0	X	0		1	X	1
	X	0	0		X	1	1
	1	1	1		0	0	0

f(x) = (a+b)(c+d)



So, first before starting d algorithm as we told that will go for some of the definitions and procedures. So, first definition is the single singular cover, so what is the idea. So, let us first look at the definition and I will try to explain you, singular cover of a logic gate is the minimal set of input signal assignments needed to represent the essential prime implicants in the Karnaugh map of the logic gate both case 0 and 1. So, let us see what does it mean, so let us take the case 0.

So, let us say that, what do you mean by prime implicants already we have seen in our discussion during the design part of the course, because you know that this course is that design verification and test. So, in design module of this course, we have discussed in details what is a prime implicants. So, we see that whenever we have a Boolean function represent it as equal to say $a b + c d$ prime plus a something like this. So, if you say that prime implicants then this one should not re contained in any of them.

As well as it should contain an implicants which is nor neither in this nor that is this contains some mean term, which is in neither of the other implicants of the function. So, this is very informal kind of way or study but, already we have discussed it in details in the design model. So, in other way of this speaking that this implicants is the is monitory present in function representation, because it has the prime in term which is neither in this neither in this, some like this. So, it has to be it is actually something called as essential prime implicants.

(Refer Slide Time: 05:35)

D-Algorithm: Definitions and procedures


Definition 1: Singular cover

Singular cover of a logic gate is the minimal set of input signal assignments needed to represent essential prime implicants in the Karnaugh map of the logic gate, both for the case 0 and case 1.

Table shows singular cover for 2 input AND and 2 input OR gate.

AND	Inputs		Output	OR	Inputs		Output
	A	B			A	B	
	0	X	0		1	X	1
	X	0	0		X	1	1
	1	1	1		0	0	0

Handwritten notes in red: (a) b, (b) ab+ab, (c) ab+cd, (d) 1



Then if it is mandate to be present but, in case of prime implicants what is the idea that say it is like a b plus c d or something like that plus f some function. Then it should not be contain directly in any one of them, like if you can say that we have function like a b plus d sorry a b plus b then what does it mean that a b plus b. Then what does it mean that a b is actually contain a b, because a b is actually a b plus a prime implicants.

So, this a b containing b, so that we can say that a b is not require because b actually represents in case a a b. So, this is some of the definitions of prime implicant and essential prime implicants which we already discussed.

(Refer Slide Time: 06:13)

D-Algorithm: Definitions and procedures

Definition 1: Singular cover

Singular cover of a logic gate is the minimal set of input signal assignments needed to represent essential prime implicants in the Karnaugh map of the logic gate, both for the case 0 and case 1.

Table shows singular cover for 2 input AND and 2 input OR gate.

AND	Inputs		Output	OR	Inputs		Output
	A	B			A	B	
	0	X	0		1	X	1
	X	0	0		X	1	1
	1	1	1		0	0	0

So, basically I am saying that you have to first select the prime implicants. And then say for the case 0 then we have to find out that is the meaning meaning of set of input assignments you get to represent the essential prime implicants in case of 0 and 1. So, let us take the case of 0, say in the case of AND gate, say AND gate 0. Now you see how we can get 0 in case of an AND gate we can get the 0, if their answer is 0 0 0 1 and 1 0.

So, this is the few cases in which case we can say that the answer is 0 for AND gate now we can replace this by 0 x or x 0 be and, because this 1 0 and say sorry I am in these are the three cases, which represents by the output of the AND gate is 0.

(Refer Slide Time: 06:53)

D-Algorithm: Definitions and procedures

Definition 1: Singular cover

Singular cover of a logic gate is the minimal set of input signal assignments needed to represent essential prime implicants in the Karnaugh map of the logic gate, both for the case 0 and case 1.

Table shows singular cover for 2 input AND and 2 input OR gate.

AND	Inputs		Output	OR	Inputs		Output
	A	B			A	B	
	0	X	0		1	X	1
	X	0	0		X	1	1
	1	1	1		0	0	0

So, now from the represent these as an prime implicants case. So, we can x 0 and 0 x because this 0 0 included here these also here in this case it is 1 0 is included over here.

(Refer Slide Time: 07:09)

D-Algorithm: Definitions and procedures

Definition 1: Singular cover

Singular cover of a logic gate is the minimal set of input signal assignments needed to represent essential prime implicants in the Karnaugh map of the logic gate, both for the case 0 and case 1.

Table shows singular cover for 2 input AND and 2 input OR gate.

AND	Inputs		Output	OR	Inputs		Output
	A	B			A	B	
	0	X	0		1	X	1
	X	0	0		X	1	1
	1	1	1		0	0	0

So, this comprises 0 1 and sorry this this 0 x incorporates 0 0 and 1 0 correct.

(Refer Slide Time: 07:17)

D-Algorithm: Definitions and procedures

Definition 1: Singular cover *01, 00, 01, 12*

Singular cover of a logic gate is the minimal set of input signal assignments needed to represent essential prime implicants in the Karnaugh map of the logic gate, both for the case 0 and case 1.

Table shows singular cover for 2 input AND and 2 input OR gate.

AND	Inputs		Output	OR	Inputs		Output
	A	B			A	B	
	0	X	0		1	X	1
	X	0	0		X	1	1
	1	1	1		0	0	0

And if you take the other case, that is, if you take the case of 0 x, then it comprises 0 0 and 0 1. So, basically this 0 0 0 1 0 we do not required to represent, because of the case that it will no longer this are prime implicants case.

(Refer Slide Time: 07:30)

D-Algorithm: Definitions and procedures

Definition 1: Singular cover *f0, 01, b+a, b+a*

Singular cover of a logic gate is the minimal set of input signal assignments needed to represent essential prime implicants in the Karnaugh map of the logic gate, both for the case 0 and case 1.

Table shows singular cover for 2 input AND and 2 input OR gate.

AND	Inputs		Output	OR	Inputs		Output
	A	B			A	B	
	0	X	0		1	X	1
	X	0	0		X	1	1
	1	1	1		0	0	0

So, these are represent it as x 0 and 0 x, so it sorry it x 0 and 0 x. So, if that is actually you can say, if you think some terms of a and b then you can say that it is b prime plus a prime the thing you can write. Because this is 0 and the other part do not care that is x a

b prime plus a prime b this other part is is hyphen or which is do not care a part of a thing, that is why we represent x 0 and this 1.

(Refer Slide Time: 07:59)

D-Algorithm: Definitions and procedures

Definition 1: Singular cover

Singular cover of a logic gate is the minimal set of input signal assignments needed to represent essential prime implicants in the Karnaugh map of the logic gate, both for the case 0 and case 1.

Table shows singular cover for 2 input AND and 2 input OR gate.

AND	Inputs		Output	OR	Inputs		Output
	A	B			A	B	
	0	X	0		1	X	1
	X	0	0		X	1	1
	1	1	1		0	0	0

So, because if you take these two combinations are automatically in these two cases we have 0 0 0 1 and 1 0 this three are actually included in this two cover. So, now we can see that this two actually form a singular cover for the case 0 because you do not require to represent this one, this one, and this one (Refer Slide Time: 08:16). Because this two is comprised over here that is 0 x and this 1 actually 0010 are actually compressed in this formula. So, this two actually 0 x and x 0 becomes the singular cover for 0 of an AND gate for (0), so OR gate output is one when.

(Refer Slide Time: 08:36)

D-Algorithm: Definitions and procedures

Definition 1: Singular cover

Singular cover of a logic gate is the minimal set of input signal assignments needed to represent essential prime implicants in the Karnaugh map of the logic gate, both for the case 0 and case 1.

Table shows singular cover for 2 input AND and 2 input OR gate.

AND	Inputs		Output	OR	Inputs		Output
	A	B			A	B	
	0	X	0		1	X	1
	X	0	0		X	1	1
	1	1	1		0	0	0

Handwritten notes on the slide include: (X,X) with a dash, a Karnaugh map diagram, and circled 'X' marks in the OR gate table with labels 'A' and 'B'.

OR gate output is 1, when we have 0 1 or 1 0 or 1 1, so it is better to represent it has 1 x and x 1. Because these two actually form the cover, because this two are prime implicants, because this is neither contains in this this is neither contain in this this is 1 have one element in this which is not containing the other.

So, that is 1 x plus x 1 actually you write a function like this that is say for the OR gate you write it as a, a plus b that is actually b that is a as dash b that is here better write in this way then the function is something like a f equal to a. Then we do not write anything, then plus do not write anything then b that is actually 1 this in this one you do not care and this is do not care and x. So, this this are two if you take comprises all this three cases 0 1, 1 0 and 1 1, which is representing 1 in case of a OR gate.

(Refer Slide Time: 09:32)

D-Algorithm: Definitions and procedures

Definition 1: Singular cover

Singular cover of a logic gate is the minimal set of input signal assignments needed to represent essential prime implicants in the Karnaugh map of the logic gate, both for the case 0 and case 1.

Table shows singular cover for 2 input AND and 2 input OR gate.

AND	Inputs		Output	OR	Inputs		Output
	A	B			A	B	
	0	X	0		1	X	1
	X	0	0		X	1	1
	1	1	1		0	0	0

Handwritten notes: (X,X) and a Karnaugh map diagram with a circle around the 11 cell.

So, the in this two actually forms the singular cover for 1 in case of an OR gate. So, now if you look at it for the 1, if you look at it for the AND gate when you get the cases of 1. So, this only one combination that is 1 1 indicate the one the AND gate and 0 0 in the OR gate. So, this will actually from this singular cover for the OR gate and this is for sorry this will from this singular cover for 1 in case of AND gate and this is for the 0 in case of an OR gate there both well official.

(Refer Slide Time: 10:05)

D-Algorithm: Definitions and procedures

Definition 1: Singular cover

Singular cover of a logic gate is the minimal set of input signal assignments needed to represent essential prime implicants in the Karnaugh map of the logic gate, both for the case 0 and case 1.

Table shows singular cover for 2 input AND and 2 input OR gate.

AND	Inputs		Output	OR	Inputs		Output
	A	B			A	B	
	0	X	0		1	X	1
	X	0	0		X	1	1
	1	1	1		0	0	0

Handwritten notes: (X,X) and a Karnaugh map diagram with a circle around the 11 cell.

So, that basically what actual is singular cover says that, so if you want to get the output 0 in case of the AND gate. So, you can either use this or you can either use this similarly, this is for 1 and in for 1 in case of AND gate this is the combination require and for OR gate 0 this is the combination require. That is actually singular cover this is minimize set of inputs and this is actually other words, which is that what is the minimal set of input to represent a output 0 in the AND gate.

(Refer Slide Time: 10:24)

D-Algorithm: Definitions and procedures

Definition 1: Singular cover

Singular cover of a logic gate is the minimal set of input signal assignments needed to represent essential prime implicants in the Karnaugh map of the logic gate, both for the case 0 and case 1.

Table shows singular cover for 2 input AND and 2 input OR gate.

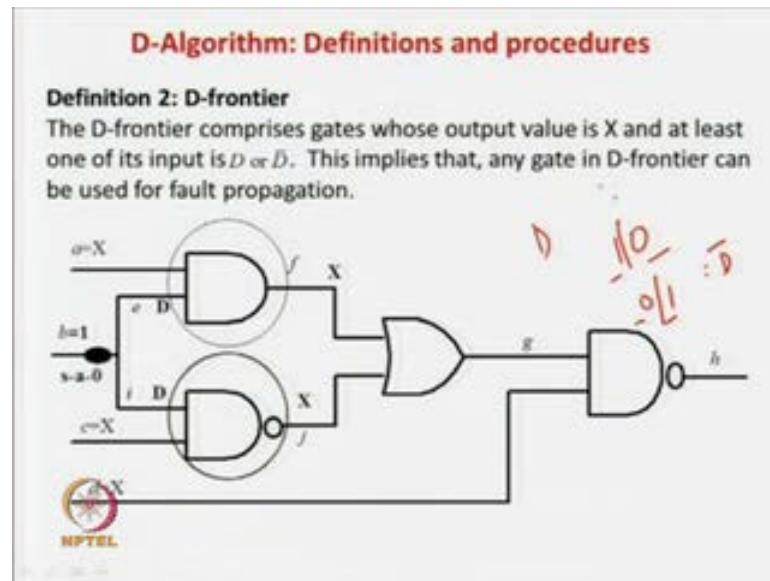
AND	Inputs		Output	OR	Inputs		Output
	A	B			A	B	
0	X	0	0	1	X	1	
X	0	0	0	X	1	1	
1	1	1	1	0	0	0	

Handwritten notes: (X,X) | (X,X) | 01 | 10 | 11

Handwritten diagrams: A circle with '1' and 'X' inside, and a circle with 'X' inside.

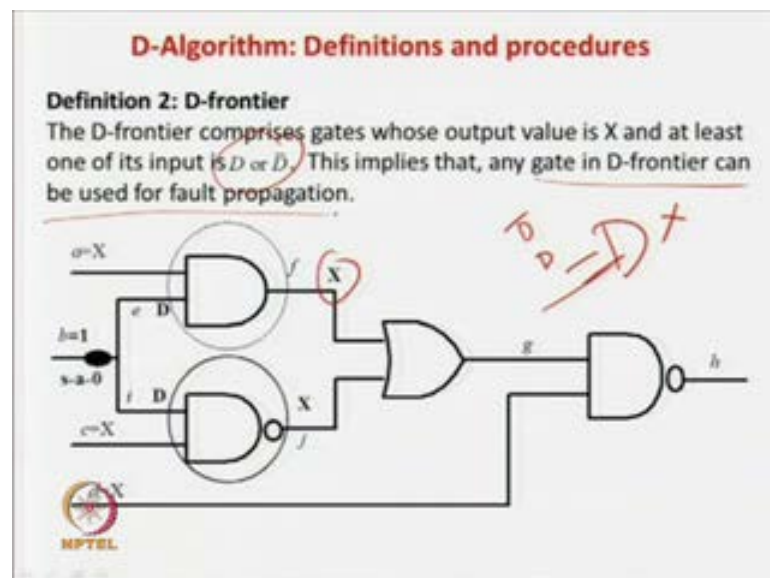
And say that only one input 0 that is the minimal set and you can also say that input is 0 other is do not care. So, these are the two minimal set, which represents the output to be 0 in case of an OR gate and we can be extra, so this is the basic definition of a singular cover.

(Refer Slide Time: 10:35)



Now, we go for what is second definition that is called D frontier. So, all this definition we are looking at will be required in the formal for formal representation of a D algorithm, so now what is the D frontier. So, we know that D basically tours introduce because the represents 1 0 that is normal case, fault case 0 1 that is actually we can also the D prime normal case and fault case. So, this was actually Roth's (\bar{D}) algebra, we have already seen the represents that was actually used for representing the fault in the circuit.

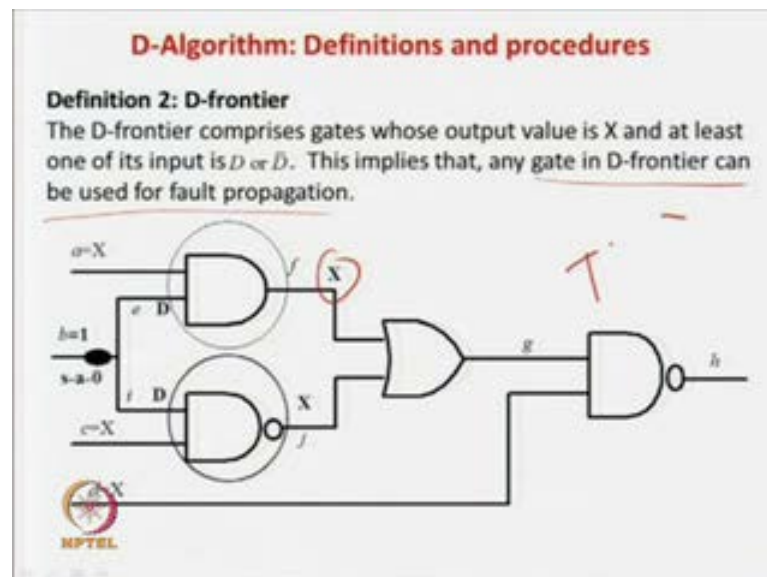
(Refer Slide Time: 11:11)



Now, see let us see, what is the D frontier. So, let us look into definition definition say that the D frontier comprises gates whose output value is x, we do not know the value x and at least one of it is input is D or D prime that is we have a D something like this. So, this s the either the D or may be D prime whatever and the output is x, if the one other input, I am not talking about the other input, decent plays that any gate in D frontier can be used for fault propagation.

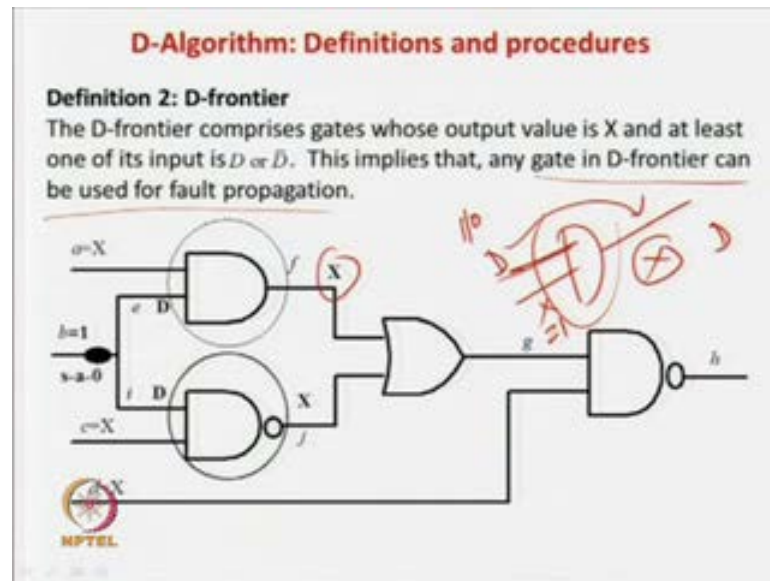
Let us now look at what is this what is meaning what does it is a definition because this is one of the most important definition or you can say notion e used in the D algorithm.

(Refer Slide Time: 11:38)



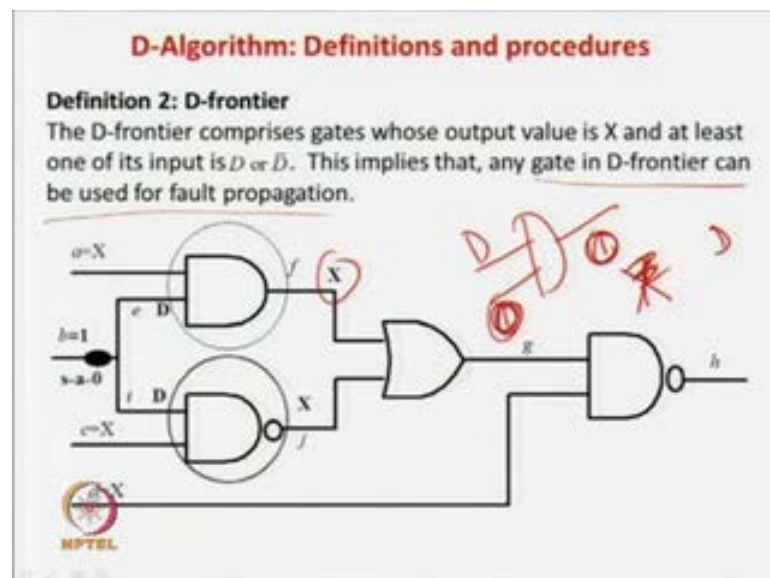
It says that for example, you have a gate like this where OR gate AND gate like this.

(Refer Slide Time: 11:41)



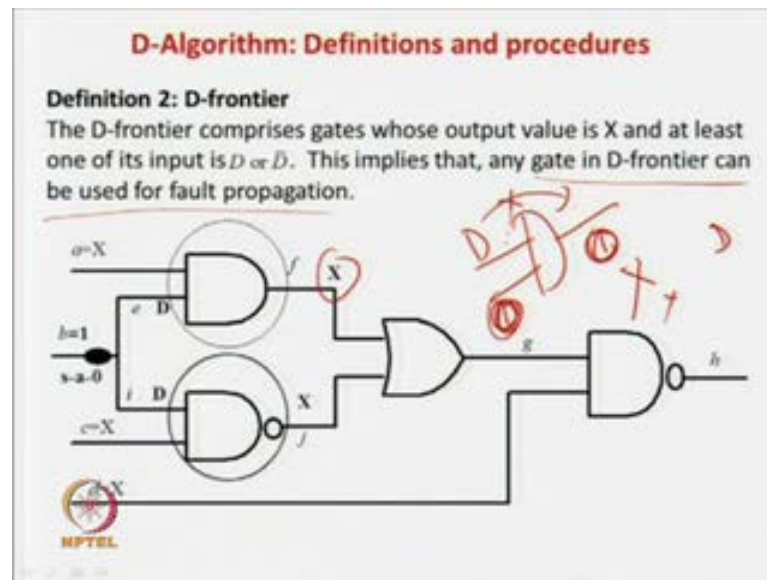
So and this is say initially all the nodes are say x, x, x we do not know say for the time being when on we say that this is d; that means, what normal case 1 and fault case 0. That means, line coverable of propagating the fault output, now if I know that this output is x; that means, as of now I do not know the value. That means, till we have a scope that this value can be D can be propagated over here if you apply a 1 over here; that means, this AND gate is capable of for fault propagation.

(Refer Slide Time: 12:12)



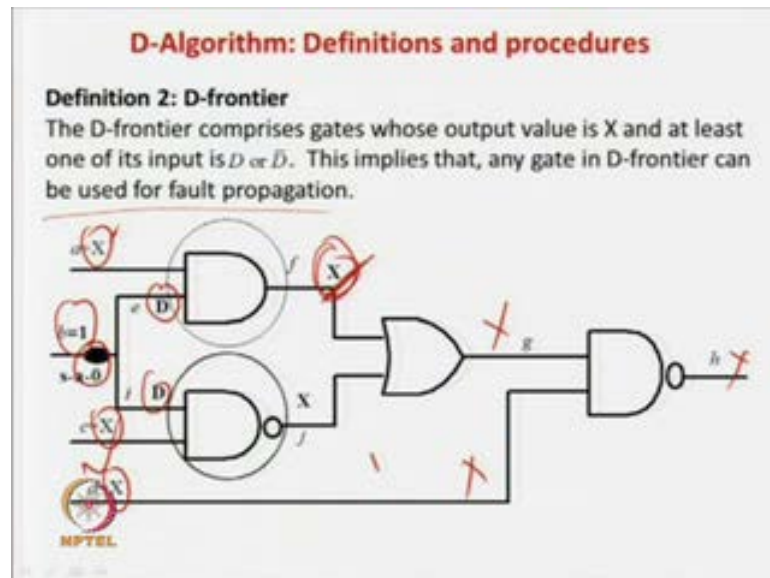
But on the other hand if you see there is a gates something like this say for example, this same AND gate we have D over here and the output here is 1 and the output here is 1 it is not x, it is not the x; that means, what now this is 1 this also become sorry this is 0 I am sorry this is 0 and this is not x this is not x.

(Refer Slide Time: 12:29)



Because if it is if it is 0 then this will be 0 and not x this is this will not be x and what is the case and D cannot be propagated over here. So, what does it means again in the D frontier can be used for propagating the affect of D fault that is D or D prime. Let us see this example of this circuit, so let us know that this is a sachet 0 faultier.

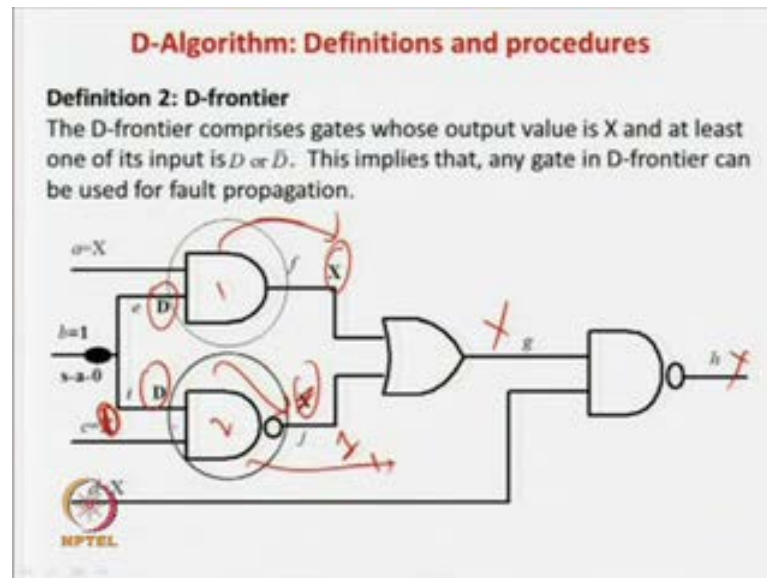
(Refer Slide Time: 12:44)



So, obviously we have sensitized you put a 1 over here now what do you do, so one more thing is that whenever you start circuit analysis you may all the input and all the net lengths as x initially. So, all these x as x because we see that we do not know the value. So, only whenever we start moving the value of the x will try to fill that from x to 0 or we can say that 1 0 D or D prime as as start computing the values, so it is a sachet 0, so your input is 1. So, this is D because normal case 1 fault case 0.

So, it say D over here and D over here and we know that input is x, x input, this is a input like this, because as of now we have not completed the value of x. So, the output of this AND gate is x, because if it is a x, if this is the x, if is 1 then you get the value here D and if this is x is 0 then you get the value 0, so that what we do not know the value of x. Similarly, this a NAND gate, so if the c is equal to 1, then we get a D prime here if c case is 0 then we get the 1 here, so as c is here in x over here, so this value is also x.

(Refer Slide Time: 13:47)



So, these are the two gates because the output here is x and x. So, these are the two gates which actually comprise the D frontier that is this fault affect can be propagated by this one or by this one. Because they are the actually comprised comprises the D frontier that is this gates will allow you that this gate 1 or 2 you can say comprises the D frontier. because they can allow you to propagate the further but, may be you think because of some reason other the x c has become a 0.

If c has become a 0, then immediately this will not b x it will be 1, then 2 will not be in fault D frontier because D can be propagated through this gate number 2. So, that is what actually we say that whenever I gat this in the D frontier then only it can be used for propagating the value of the output of the circuit am sorry I mean fault effect of the circuit.


(Refer Slide Time: 14:27)

D-Algorithm: Definitions and procedures

Definition 2: D-frontier
The basic idea is, X at the inputs can be appropriately selected so that D or \bar{D} can be propagated to the output of the gates.

In other words, faults can be propagated only through gates in the D-frontier.

Once the fault is propagated the gate is deleted from the D-frontier list.


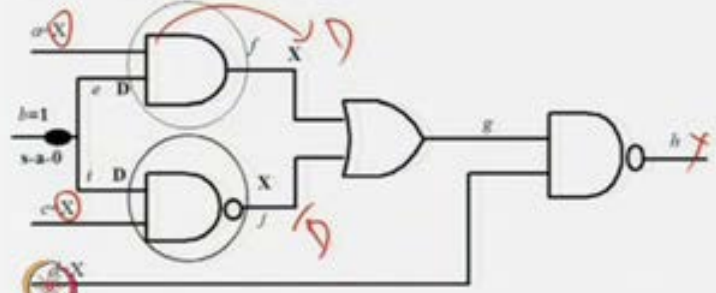


That is what is the basic idea, the basic idea is that x at the inputs can be appropriately selected. So, that D or D prime can be propagated to the output of the gates that is the basic idea that is...

(Refer Slide Time: 14:38)

D-Algorithm: Definitions and procedures

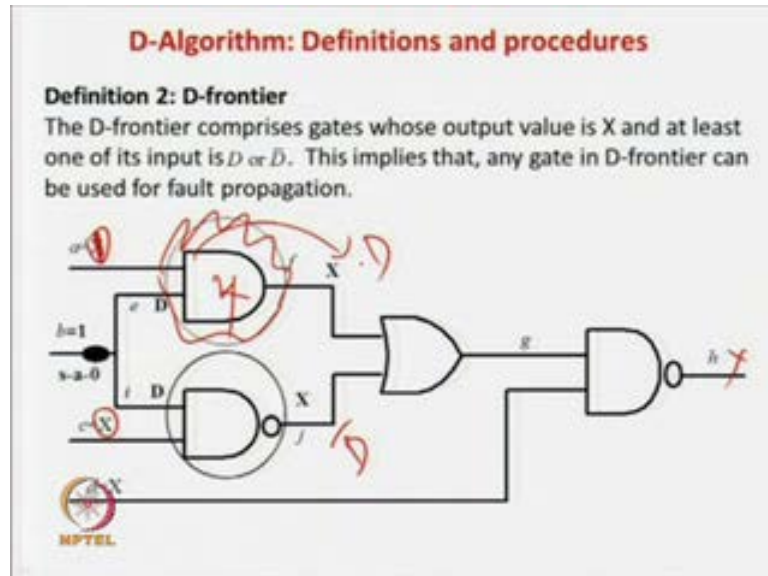
Definition 2: D-frontier
The D-frontier comprises gates whose output value is X and at least one of its input is D or \bar{D} . This implies that, any gate in D-frontier can be used for fault propagation.



So, these x this x can be controlled in such a way, so that these effect of D and here D prime can be propagate the output or the output of this gates which comprises the D frontier. So, we can if there is a gate in the d frontier then we have a scope that we can

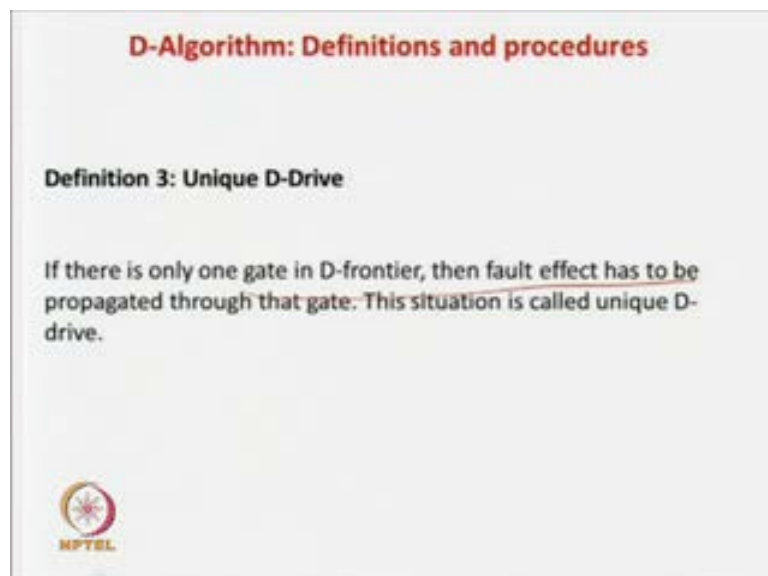
say this value approved lately or accordingly. So, that the fault effect can be propagated through the gates in the D frontier.

(Refer Slide Time: 15:04)



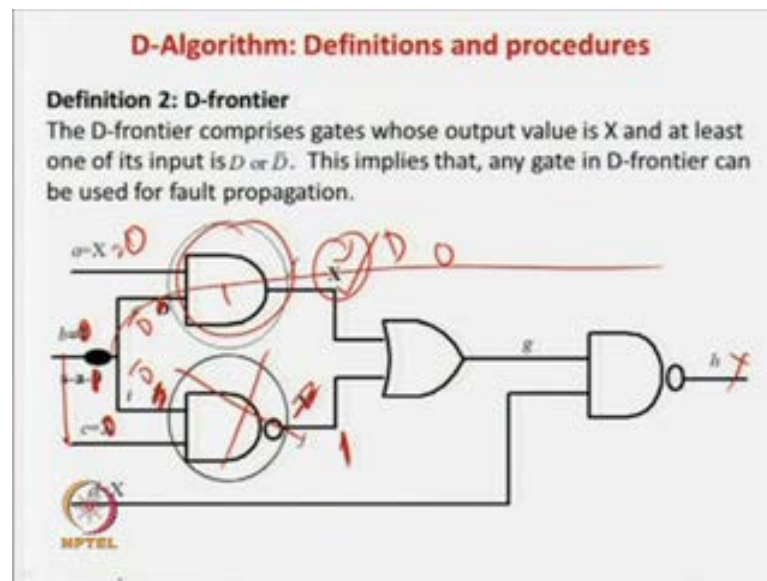
So, once it is very important, so once the fault is propagate for that this propagated that is say fault is propagated by making this as one then this one will not no longer be in the D frontier. Because already you have used this gates and the fault has been propagated. So, this cannot be again use for propagate the fault effect. So, it can be it is to be derivate from the D frontier that is the basic notion here this is second definition.

(Refer Slide Time: 15:22)



Now, there is next definition that is number 3, so that is called unique D drive. So, what is unique D drive then is actually called actually if there is only one gate in the D frontier then is actually, what do you called a unique D drive ,this situation because it can be then propagated through only one input. So, let us just see (Refer Slide Time: 15:38), what is a unique D drive the definition is very straight forward say for some example, say for due to some reason or something like this.

(Refer Slide Time: 15:46)



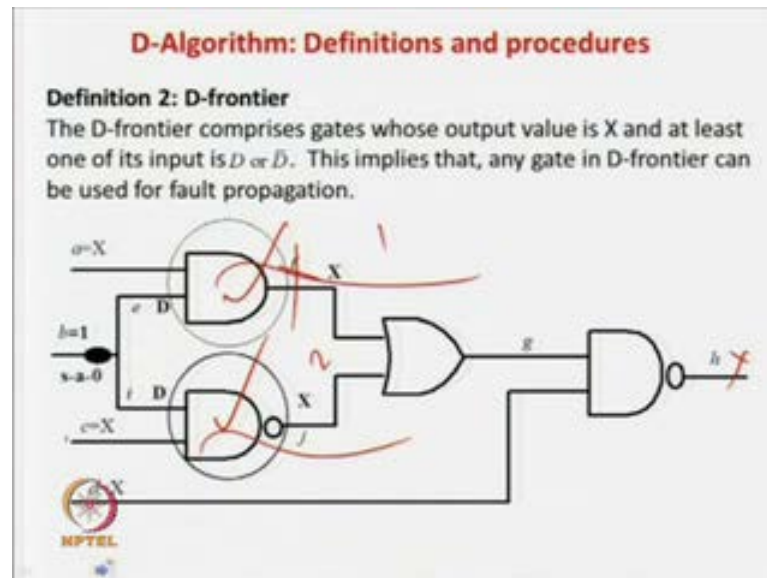
Let us say that this two lines were short some for some reason, then what and let us consider this for this sachet 1 or let us change this one; then what happens to sensitize sachet 1 would apply a 0 over here. So, the moment you apply a 0, so c will be 0. So, moment you apply 0 you will get the 1 over here but, in this case this is it will be it will not be D it will be a D prime and this will be a D prime, because normal case 0 fault case for s 1 and if you and this will be still 0, so this s 1, because this is 0.

Now, this AND gate output is still on x, because if you apply x equal to 1, then you will get the D over here if it is 0, you will get a 0 over here. So, there is a scope of propagating the fault this 1. But, is only one gate that is only this gate is D frontier, this gate not in the D frontier, so it has the unique gate, so which the fault effect have to be propagated.

So, this situation is called the unique D drive (Refer Slide Time: 16:34) there is only one D only one gate in the D, frontier which can be used for propagating the value of default

derivates. So, if for fault unique D drive then you will not have to think for for the choices just like I mean we have seeing the last example that always or with also be looking in details in this in this lectures or next lecture. That D algorithm like sensitize propagate and justify is always not a successfully gate, sometime you will if there is a like in the last example we have two two D frontiers.

(Refer Slide Time: 17:12)



Like this this one if you gone for the general case that is if you have just gone for the non modified case ten what is the idea. So, if I gone for non modified case then you have this two gates in the D frontier this gate and this gate in the D frontier.


Now, it is up to you, whether you see that this gate as the D frontier or this gate as the D frontier to propagate the value of the fault. Now, once you have selected this gate for all propagation then this is related but, there is option one and option two.

(Refer Slide Time: 17:29)

D-Algorithm: Definitions and procedures

Definition 3: Unique D-Drive

If there is only one gate in D-frontier, then fault effect has to be propagated through that gate. This situation is called unique D-drive.




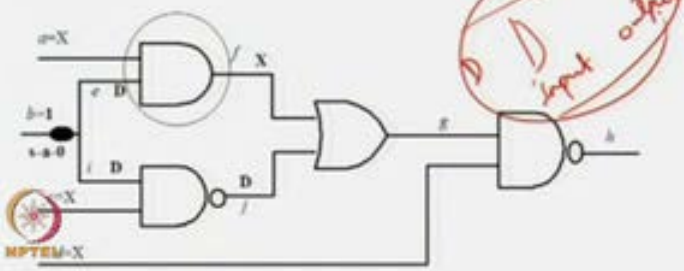
So, sometimes one or the option give you correct result correct result means inconsistency that is you know inconsistency, you can find the propagate default related to the output or you can also justify this. But, for some of the selection of default or there is a multiple choice available like in this case of two gates. So, you may land into a problem, because the propagation in one of the gate you may land to a inconsistency situation that is fault can be propagated but, it cannot be justify. So, in a unique D drive there is no option to think about the uses only one drive.

(Refer Slide Time: 18:03)

D-Algorithm: Definitions and procedures

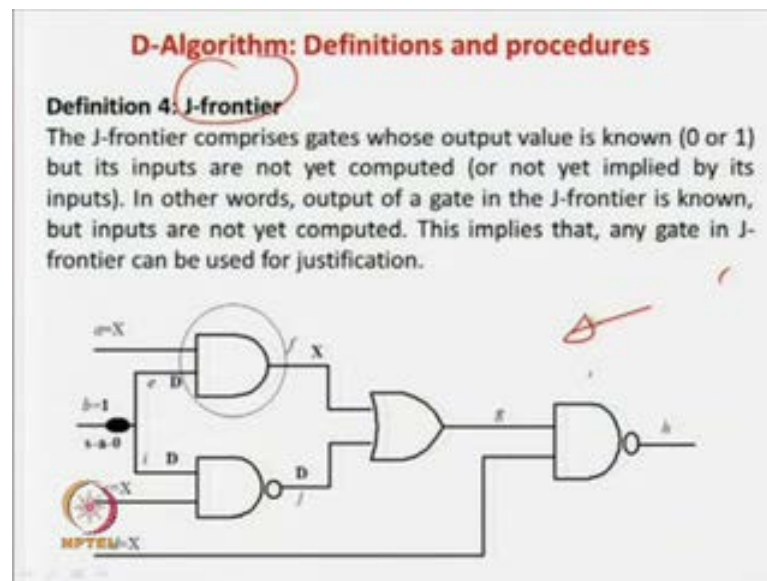
Definition 4: J-frontier

The J-frontier comprises gates whose output value is known (0 or 1) but its inputs are not yet computed (or not yet implied by its inputs). In other words, output of a gate in the J-frontier is known, but inputs are not yet computed. This implies that, any gate in J-frontier can be used for justification.



So, you have to just drive the value using the AND gate, so unique means there is no choice at that point. So, now let us look at the next definition that is definition four is about J frontier, so that J frontier that is mainly used in D algorithm one called the D frontier and one is J frontier. So, basically you remember this way, so D frontier means what that is you have propagating the value of the effect that is D effect from input to output, so that is what is basically you are doing.

(Refer Slide Time: 18:44)



So, if you are using a D frontier algorithm, so D algorithm may D frontier definition if you think. So, it is breaking the value of some D or D prime that is fault effect an you trying to propagate it to the output by gates having the output is x, so that is input to output. Now, this actually the propagation, so the D frontier is basically related to propagation, then something we know that to justification.

(Refer Slide Time: 18:48)

D-Algorithm: Definitions and procedures

Definition 4: J-frontier

The J-frontier comprises gates whose output value is known (0 or 1) but its inputs are not yet computed (or not yet implied by its inputs). In other words, output of a gate in the J-frontier is known, but inputs are not yet computed. This implies that, any gate in J-frontier can be used for justification.

So, once you are say for example, propagated the value up to here, now this line this later and this later properly justify. So, that the propagation and sensitization and successful. So, J D algorithm and sorry J frontier as we see now is basically related to the justification procedure and it works backward it works for primary output input, so backward direction. Now, let us first read the definition and then we look the meaning and implication, so the J frontier what we saying. So, j frontier what we are saying, J frontier comprise gates whose output value is known either 0 or 1 but, it is input or not yet compute or not yet implied by it is inputs.

(Refer Slide Time: 19:24)

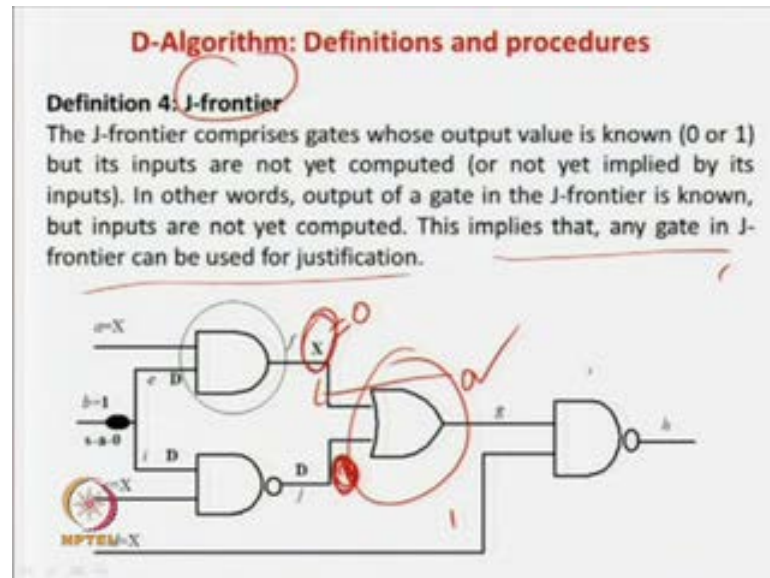
D-Algorithm: Definitions and procedures

Definition 4: J-frontier

The J-frontier comprises gates whose output value is known (0 or 1) but its inputs are not yet computed (or not yet implied by its inputs). In other words, output of a gate in the J-frontier is known, but inputs are not yet computed. This implies that, any gate in J-frontier can be used for justification.

So, in other words the output of a gate in the J frontier is known but, inputs are not yet compute that is it implies that any gate in J frontier can be used for justification. That means, what say for some reason you know the value of this 1 is to be 1 or 0 1 whatever you know that I have to get at this 1, so you should not write 0 oblique 1.

(Refer Slide Time: 19:41)



So, you know that I must get the value of 0 in this I must get the value of 1 in this as something like this but, now this is node but, that this is already known but, this inputs are say x this is x and this is also x the inputs are not yet computed. But, not it implied by those that is the inputs are what you know or at least one of it input is not yet known, at least we know they say 0. So, this we know that 0 for some but, this is x; that means, to justify the value of x here.

We know that x is to be completed to 0 that is, in these gate what happens the output is known either one or both of the inputs are still x. That means, this gates still as the scope that is 0 can be justified over here. So, let us will take this bigger example later we just derive the simple gate and sustain the concept.

(Refer Slide Time: 20:26)

D-Algorithm: Definitions and procedures

Definition 4: J-frontier

The J-frontier comprises gates whose output value is known (0 or 1) but its inputs are not yet computed (or not yet implied by its inputs). In other words, output of a gate in the J-frontier is known, but inputs are not yet computed. This implies that, any gate in J-frontier can be used for justification.

So, let us take this case, so we say that the output is here is 1 and with for that I mean let us assume that inputs are x. So, we know this gate can be in J frontier because you know the output is the 1 now for make this successful we require this 1 to be 1 and this 1 to be r 1, so because this may be for requirement.

(Refer Slide Time: 20:51)

D-Algorithm: Definitions and procedures

Definition 4: J-frontier

The J-frontier comprises gates whose output value is known (0 or 1) but its inputs are not yet computed (or not yet implied by its inputs). In other words, output of a gate in the J-frontier is known, but inputs are not yet computed. This implies that, any gate in J-frontier can be used for justification.

This may be requirement because see it may be drive in some sachet 0 fault. So, it is massed at just that it fault can be sensitized. So, this x equal to 1 and this x equal to 1 you can make that it can be successful.

(Refer Slide Time: 21:03)

D-Algorithm: Definitions and procedures

Definition 4: J-frontier

The J-frontier comprises gates whose output value is known (0 or 1) but its inputs are not yet computed (or not yet implied by its inputs). In other words, output of a gate in the J-frontier is known, but inputs are not yet computed. This implies that, any gate in J-frontier can be used for justification.

Now, let us see for some for reason both these inputs are 0 and 0 1 for some reason you can think because of some other integration some other set, so it is 0. Now, this is 1 and this is not x and this is also these things are not x, then this cannot may be J frontier. What is called as J frontier because this is sachet 0 and this inputs are not x so; that means, there is no scope that you can control this gate. So, get the value of a 1 over here. So, this for this gate we justification procedure is stopped.

(Refer Slide Time: 21:32)

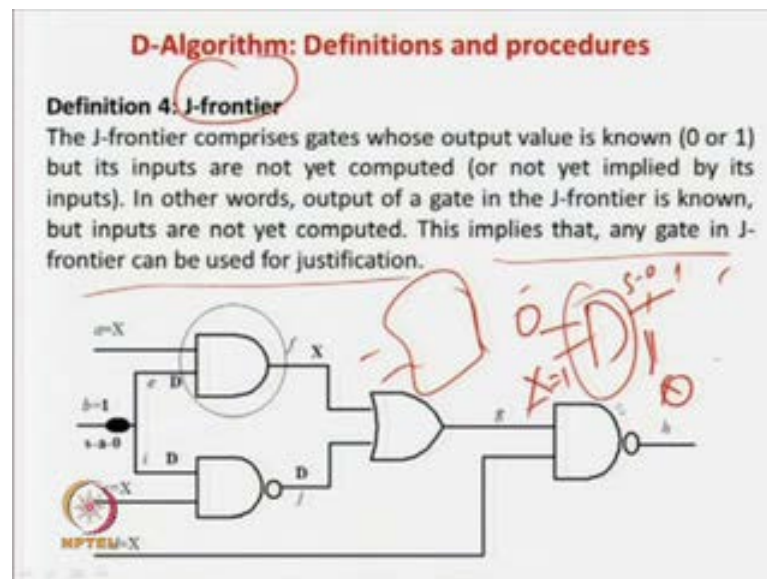
D-Algorithm: Definitions and procedures

Definition 4: J-frontier

The J-frontier comprises gates whose output value is known (0 or 1) but its inputs are not yet computed (or not yet implied by its inputs). In other words, output of a gate in the J-frontier is known, but inputs are not yet computed. This implies that, any gate in J-frontier can be used for justification.

In other words say for example, they can have a gate like this, this is a sachet 1 and this is 0, you have to apply see you know this is x and this is 1. So, this gates still get J, frontiers because we know that still this x is with you, you can control this x and you can apply 0. So, that you can successfully control the gates justifies this value. So, there is one 1 x; that means, sometimes it can be possible to justify and sometimes it may not be like.

(Refer Slide Time: 21:57)



For example where in again in the J frontier always does not mean that it will be successfully justification but, it may be like in last it was not but know in this case sachet 0, so you have to apply a 1. So, this is 0 and this is x for some reason this is 0, because of some other reason like fault sensitization and some other case because of some other I mean in this some other inputs are there some reason this is 0, so this is x. So, this is in J frontier because this is the x.

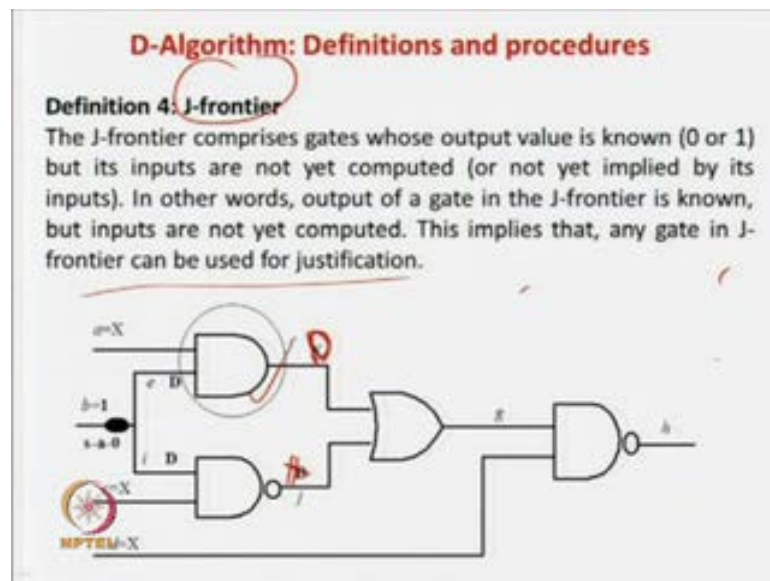
So, you may think there is a scope that is the x can be control, so that the output is 1. So, you try to apply this as 1 but, because this may be input is 0. So, you can never get a 0 over here. So, this gate is in the J frontier but, it is not successfully allow you to propagate the, I mean justify the value of the output of the AND gate.

So, gate is in the J frontier, so at least one input is 0, 1 input is x of the gate but, it does not just imply always to be successful in you called justifying the output of the gate. But, at least over inputs are not known the output is known that is output is known means that

is the requirement. And inputs are not known that we have to compute that inputs from where from where the requirements of this justification algorithm you have to find out that 1 also.

So, if you have again to the output is known but, inputs are not yet known and they have to be computed. So, you can think that gate can be used for justification, so that gate falls is in J frontier.

(Refer Slide Time: 23:19)



In this case say what do you say just take again example this case if you see, what do you want to over here. So, output of the gates are known say for example, in this case let us think that this is 0 D, because this gate D has to propagated over here correct. So, also this gate has to be propagated over here. Now, if you look at this gate, so what do you know we know that in this case say for example, because of the D frontier. So, this usually both of these were in d frontier somebody has selected that this should be D frontier and not this, so let us keep it has an x, so we are not yet selecting. So, let us the things that is we have been selected has the D frontier this is not in selected. So, once it is selected for the D frontier.

(Refer Slide Time: 24:03)

D-Algorithm: Definitions and procedures

Definition 4: J-frontier
The J-frontier comprises gates whose output value is known (0 or 1) but its inputs are not yet computed (or not yet implied by its inputs). In other words, output of a gate in the J-frontier is known, but inputs are not yet computed. This implies that, any gate in J-frontier can be used for justification.

So, what going to happen, so this 1 will be x will be converted to D and because it non inverting gate and sorry this is not D this is x because this may be not selected over d frontier for the fault propagation this one. Now, what happens, so in this last just look at this gate. So, output is known to be D 1 of the inputs we know is to be D other input is x. So, this gate will lie in the J frontier, because now this is D this is D and we need that we need to find out what is d x.

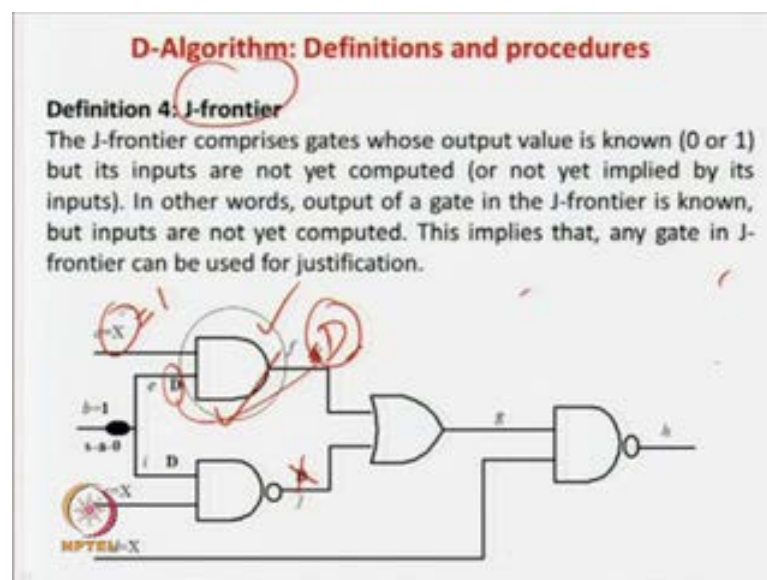
(Refer Slide Time: 24:48)

D-Algorithm: Definitions and procedures

Definition 4: J-frontier
The J-frontier comprises gates whose output value is known (0 or 1) but its inputs are not yet computed (or not yet implied by its inputs). In other words, output of a gate in the J-frontier is known, but inputs are not yet computed. This implies that, any gate in J-frontier can be used for justification.

That means, this were going backward and still these gate has a scope or justifying the output of D over here. So, obviously, you know that x equal to 1. So, see that x equal to 1 then this input actually justifies the propagation of this one, so this gate falls in the J frontier. So, what we have done initially we know that if just look at this case this is also x let us think, this is wrong for the type of is x actually. So, initially everything is x so you apply sachet 0 1, so D and D over here. So, see that this for the D frontier this one also D frontier this one is also D frontier because because in these two gates you can propagate this value of this D through this suppose both of them are D frontier. So, let us that this one is the d frontier.

(Refer Slide Time: 25:08)

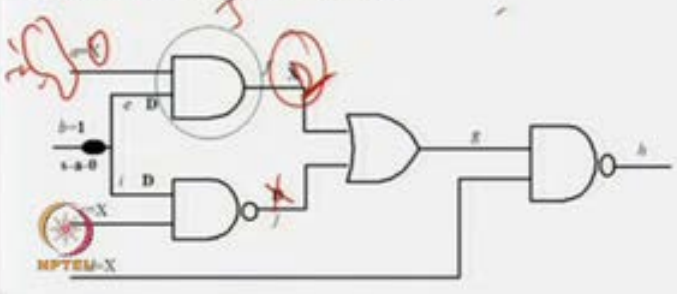


So, what just you see that the value of D coming over here, now this x is gone these two remains in the x because this is not selected as a the D frontier now once you selected the d from the output is known, so once the output is known this input is x. So, you can basic idea for frontier you can select this appropriately this will be 1 or 0 whatever the case, so that this output is justified over here. So, this actually this gate as the input is 1. So, it is actually you can used for justification of your approach. So, this actually faults in the J frontier, if actually this is actually gate is taken as the D frontier.

(Refer Slide Time: 25:40)

D-Algorithm: Definitions and procedures

Definition 4: J-frontier
The J-frontier comprises gates whose output value is known (0 or 1) but its inputs are not yet computed (or not yet implied by its inputs). In other words, output of a gate in the J-frontier is known, but inputs are not yet computed. This implies that, any gate in J-frontier can be used for justification.

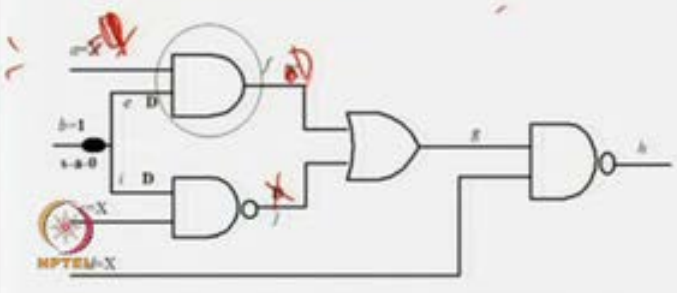


So, the output is 1 and the input you can compute for sometimes they mean all should not think that way, that this gate is in the J frontier. So, it can be used for propagation that is high or sometimes again not also taking in the J frontier can do your job, because of some other reasons of some other inputs say that these are automatically D comes out. So, in this case what happens this in J this gate may not be in the J frontier but, still you have actually it is justify your outputs.

(Refer Slide Time: 26:26)

D-Algorithm: Definitions and procedures

Definition 4: J-frontier
The J-frontier comprises gates whose output value is known (0 or 1) but its inputs are not yet computed (or not yet implied by its inputs). In other words, output of a gate in the J-frontier is known, but inputs are not yet computed. This implies that, any gate in J-frontier can be used for justification.




So, this is also be in d case, also for some times it may also happen that this x is 0 for some reason then this case J cannot be propagated over here in that case, this does not remain in a J frontier and it does not sorry get the purpose. So, again being in D frontier you can say many things. So, this gates in D frontier; that means, you can say that I can use sorry this gate is the D frontier; that means, you can use this for justification but, again in not D frontier does not imply much. So, again not in D frontier say a equal to 1 then it will help you to justify automatically this is 0, the gate is not in the D frontier because as it exist 0 or 1 already pre defined then both inputs are known and output is also known.

So, you got result in the J frontier, so sometimes it may help you if it is 0 that will not help you because they cannot be propagate and if it is 1 it can help you because D can be propagated. Then it does not remain in a J frontier, so again not in J frontier may or may not help you that depends on the situation.

(Refer Slide Time: 27:09)

D-Algorithm: Definitions and procedures

- Assume that it was decided that fault effect D be propagated via j .
 - So, $j=D$ by forward implication.
 - Also, $f=0$ for propagating D to g .
- Now, the encircled gate is in the J-frontier, because the output is known as 0 and its inputs are X, which can be decided in *justification step*.
- The basic idea is, X at the inputs can be appropriately selected so that output is justified. In other words, during justification gates in J-frontier can only be considered. Once the inputs are justified, the gate is deleted from the J-frontier list.



But, if gate is in d frontier because this is known and this is x, then if the gate is in sorry j frontier most probably j frontier. So, if gate is in j frontier then it can be used to find out that whether it can be used for justification or not. So, the gate not in J frontier it may help or may not help again the gate is in J frontier; that means, you can study that it can be used for justifying your approach. So, sometimes it may be successful, sometimes it

may not be successful but, whatever gates are in J frontier that can be used for justifying your inputs.

So, what do you do basically we propagate the fault, so J D frontiers and then it reach the primary output and then once you reach the primary output default then you have to find out which are the gate which are in J frontier. And then we have try to justify using the J frontier, so will take some complete examples in next lectures, so which will allow you to understand the concept better. So, whatever which we are saying this D I mean, this last example which we see that assume that this was decided at the fault effectively propagate via J.

So, that is we are actually affecting the in this same example actually we are may not say same idea is there we may not I mean go for this one this same except the you have discussed about the J frontier and the d frontier. So, this thing is this I mean elaborated the in a detail manner over here, so this example is detailed over in this slide, so you can see.


(Refer Slide Time: 28:20)

D-Algorithm: Definitions and procedures

Procedure 1: Implication
The implication procedure comprises 3 steps

1. Compute all the values of the signals that can be determined uniquely from already given signal values (of some nets). If many choices are available, for example as in singular cover, any one of them can be considered.
2. Maintain J-frontier and D-frontiers
3. Check for consistency and stop in case of inconsistency (i.e., contradictory signal values are implied by the implication procedure).

Basically, implication procedure is similar to a simulation process where values of all nets (and finally primary outputs) are determined starting from primary inputs.

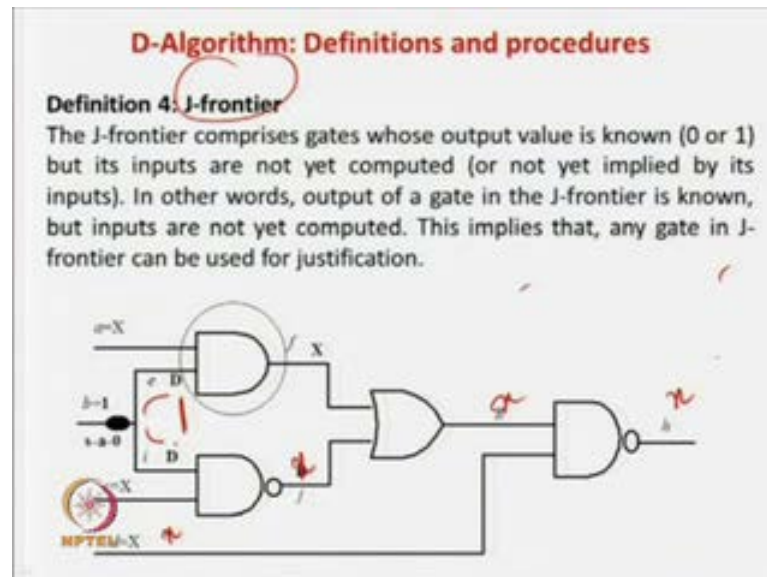
 NPTEL

Now, we go to the D algorithm that is another some definition, now we see and some procedure that is some algorithm or some sub states of this one. So, first sub state will be implication. So, implication actually procedure or for module after D algorithm it comprises three steps compute all the values of this signals that can be detect uniquely

from the given signal below some values if many choices are available for example, as singular cover any one of them can be considered.

So, what it is then that is residing in next step then we come back maintain J and D frontiers and check for consistency and stop in case of inconsistency that is contradiction, basically the implication procedure is in is similar simulation process, where values of all nets are determined starting from the primary inputs. So, what basically implication does, so we try to find out compute the all values of the signals that can be determined uniquely from the given signal in your some case.

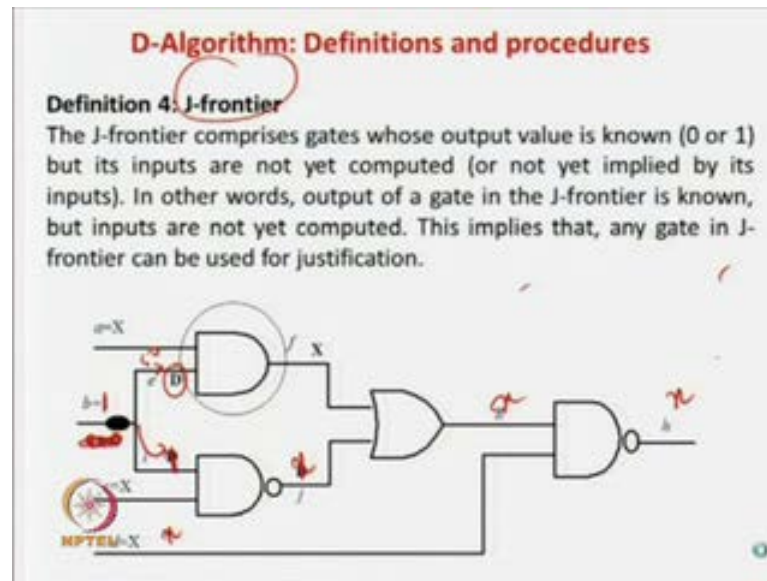
(Refer Slide Time: 29:26)



That is for one example we take this one so; that means, what, so initially everything is x initially we you start your design or start your implication procedure and you think this x over there. Then what we do say for example, we apply a D 1 D over here this sachet we know that here 1 is must to be apply; now once D is 1 is apply till what we know that immediately this two are D and (()).

But, we do not know the value of c d and a because they have to be determine by using D frontier and J frontier but, immediately, we know that this is D and D. So for example, also if you think that there was another say also let us take the other way, so let us not take the fault over here.

(Refer Slide Time: 30:02)



Let us take default c here sachet 0 like this, so what happens we know that d 1 has to be apply. So immediately, 1 d as apply we know that this is d, so this is the implication procedure. And also if to sensitize this we apply a 1 similarly, we know that this is. So, that is but, other are cannot compute because others we have to know the D frontier and the J frontier. So, suppose some of the values by the sensitization process some other mechanism.

What happens that once you set the values for some reason like in this case say the value of 1, then automatically this is sachet 0, so it will be 0 and this net does not have any fault because we know that in case of (()) out circuits. So, this net and this nets are, actually independent of each other. So, if the ne number I is independent of net number e then actually this value can be directly incorporate from here. Now, white can be incorporated because we know that d equal to 1 then this I will be equal to 1 because the fault is in the other branch of the other (()) output.

So, this actually comprises your what do you called this this form in this is implication procedure that is circuit initially we have some all the nets will be set as x. Now, next what you have to do you have to just find out that, what do you have to find out that some of the net values either than directly compute. Like in this case one can be directly computed over here, this was the implication but, the value of f value of g for the value of a this cannot be computed directly.

Because for that what do, you have to do you have to first find out whether you will use because if you having a sachet 0 fault at e, then what happens then your f this gate actually this is the only gate this will only become your D frontier then you have to actually propagate the value d through f. So, once you propagate the value of the d through f, then you know that this is this then your this gate will become your J frontier and then you have to find out that with the what is the value of x.

So, x you cannot it is difficult to directly get it by implication, because by implication what will do is that, what we do is that by implication you just find out the values which can be known directly. That was the first step we discussed that compute the values of all this signals and can be determined uniquely from the given signal values of some nets if any choices are available. For example, you can get the singular cover many choices are like for example, like for example, you have AND gate this output is actually 0 for some other net. So, we know that the inputs can be or these implication can be x 0 or 0 x.

(Refer Slide Time: 32:27)

D-Algorithm: Definitions and procedures

Procedure 1: Implication
 The implication procedure comprises 3 steps $0 + = D 0$

1. Compute all the values of the signals that can be determined uniquely from already given signal values (of some nets). If many choices are available, for example as in singular cover, any one of them can be considered.
2. Maintain J-frontier and D-frontiers
3. Check for consistency and stop in case of inconsistency (i.e., contradictory signal values are implied by the implication procedure).

Basically, implication procedure is similar to a simulation process where values of all nets (and finally primary outputs) are determined starting from primary inputs.

NPTEL

So, you can take any one of them that is what you say that is the multiple choices available you can take any one of them and this is may be considered. So, then for once you have done this implication then you can also find out that what are the J frontier and what are the D frontiers. Basically you will find out what is D frontiers that is how you will propagate the value then you can get the J frontier sort of it.

So, say some inconsistency, inconsistency means that you want the value to be one for some net and because of this, what is the implication and all you get the value to be 0 then actually say inconsistently you have to stop. So, basically implication procedure, so this is your procedure with the values of nets or determines it from the primary inputs. So, it just like a simulation, simulation what we do know the inputs and we try to compute the outputs. So, we are seeing in last few in last 2, 3 lectures back even your simulation you have seen compile good simulation we have seen.

So, in that case what we do we start the, we give some input to the circuit and we try to find out what is the output of the circuit that is the basically the idea. So, in this case also what happens we try to, because they fault in this circuits. So, what we do is that, we first find out the sensitized for sensitized fault, then initially all this circuit nets having x then you sensitize default then we try to find out which gates of the circuit can be determined 0 or 1 or D or D prime by this value.

So, once some of the nets are been determined this is step one, then if this multiple choice you can take any choice of them like x 0, 1 x, 0 1 0 0 whatever if some multiple choice are there like 0 x, x 0 1 x, x 1. So, like this for AND gate and not gate getting that is primary input cover we have seen.

So, any once of the choices you can take and then we try to find get the D frontier and the J frontier all this things you can find out then you can go for justification and all. So, we try to find out as many values of net uniquely as possible.


(Refer Slide Time: 34:13)

D-Algorithm: Definitions and procedures

Procedure 1: Implication
The implication procedure comprises 3 steps

1. Compute all the values of the signals that can be determined uniquely from already given signal values (of some nets). If many choices are available, for example as in singular cover, any one of them can be considered.
2. Maintain J-frontier and D-frontiers
3. Check for consistency and stop in case of inconsistency (i.e., contradictory signal values are implied by the implication procedure).

Basically, implication procedure is similar to a simulation process where values of all nets (and finally primary outputs) are determined starting from primary inputs.





But, having step in maintain the J frontier and the D frontier that is very important and also there is some inconsistency like say you have applied two values.

(Refer Slide Time: 34:19)

D-Algorithm: Simulation and Implication

Simulation	Implication
All the signal values are determined uniquely	All signals may not be determined uniquely
Value assignment moves from inputs to outputs of a circuit	Signal assignments propagate both towards primary inputs and primary outputs. For example, to test a s-a-1 fault at net l say, we need to have implication in two directions (i) backwards, to primary inputs to make $l=0$ and (ii) forward, to primary outputs to propagate D.
There is no inconsistency	Inconsistency may arise, when for a given net l (which is not the fault location) different signal values (0 or 1) need to be assigned.





Then I mean once you say that if it is to be 0 and by I mean what we called implication the value 1 and vice versa. So, you have to stop at the same (()). So, this implication is actually this you can called as a very important module of the D algorithm. So, when we will get the d algorithm in detail, so will in more formal we see how it feeds to. So, basically this is the heart of the D algorithm that is first uniquely determined what the

value of this signals maintains J frontier and D frontier and then in case of D algorithm we apply two more points that is there is n number D frontiers.

We have to take if the n gates in D frontier then you have to take any one of them and then you have to go for the J frontier justification and if you keep on doing it the D frontier is the primary output. So, basically will see that this implication is basically is the heart of D algorithm. So, but this slides difference between the simulation and implication let us see, so in each case all this in case of simulation, all the values are determined uniquely.



Because these some primary inputs; obviously, circuit will have some primary output there is no choice for and gate if your input is 0 0 you will get the answer is 0, if your input is 1 1 you get the answer 1. So, there nothing called x 0, x 1 all these things are not in simulation, simulation you have simple inputs you get simple output. But, in case of the implication some signal may not be determined uniquely, why because already we have seen that for a AND gate if the output is 0 so you take x 0 or 0 x. So, we can also take 0 0 but, we generate this 0 0, because there is a problem which we will see later, that we will see later why we will not taking. So, that is one thing, so in case of implication some of the values may be x we are not determined uniquely.

(Refer Slide Time: 35:51)

Simulation	Implication
All the signal values are determined uniquely	All signals may not be determined uniquely
Value assignment moves from inputs to outputs of a circuit 	Signal assignments propagate both towards primary inputs and primary outputs. For example, to test a s-a-1 fault at net <i>l</i> say, we need to have implication in two directions (i) backwards, to primary inputs to make <i>l</i> =0 and (ii) forward, to primary outputs to propagate D.
There is no inconsistency 	Inconsistency may arise, when for a given net <i>l</i> (which is not the fault location) different signal values (0 or 1) need to be assigned.

So, the value of assignments used for inputs and output for simulation. So, simulation primary inputs you give 1011 in the circuit and then you compute the output, so that is very simple.



(Refer Slide Time: 36:05)

Simulation	Implication
All the signal values are determined uniquely	All signals may not be determined uniquely
Value assignment moves from inputs to outputs of a circuit 	Signal assignments propagate both towards primary inputs and primary outputs. For example, to test a s-a-l fault at net I say, we need to have implication in two directions (i) backwards, to primary inputs to make $I=0$ and (ii) forward, to primary outputs to propagate D.
There is no inconsistency 	Inconsistency may arise, when for a given net I (which is not the fault location) different signal values (0 or 1) need to be assigned.

So, in case of implication what we will do it is not a unidirectional flow that will not flow from straight this 1 to input and output. But, there what we will do we first find out the some points we find out some some fault, fault is there then sensitize it then we find out the D frontier. So, D frontier will propagate the value then once d frontier propagate the values some J frontiers may be created, for in J frontier you have to justify the value of I mean justify the inputs.

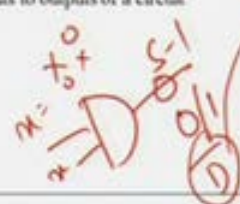

And then again and use a D frontier is stated as similarly, we take the d value the output and in every step one, one step which we are propagating the d value at the same time we are trying to justify also that the D frontier take the values and propagated. So, it tries to justifies the values of the input of gates in the J frontier. So, it is actually it propagate both ways primary inputs and outputs. For example, take as sachet 1 on net a, you have two implications backward to make the primary inputs is 0 and for output is the primary output to D.

(Refer Slide Time: 36:54)

D-Algorithm: Simulation and Implication	
Simulation	Implication
All the signal values are determined uniquely	All signals may not be determined uniquely
Value assignment moves from inputs to outputs of a circuit 	Signal assignments propagate both towards primary inputs and primary outputs. For example, to test a s-a-1 fault at net l say, we need to have implication in two directions (i) backwards, to primary inputs to make $l=0$ and (ii) forward, to primary outputs to propagate D.
There is no inconsistency 	Inconsistency may arise, when for a given net l (which is not the fault location) different signal values (0 or 1) need to be assigned.

That is what actually that is say if you have AND gates, so this one. So, this is actually sacht 1, so you apply a 0 over here. So, this is a forward direction you can say, so because we can considered these to be D frontier. So, it will be actually D prime in this case because normal case 0 fault case from case sacht 1. Now, again once this is 0 this is d what will be the output it is d in normal case 0, fault case 1 sorry D prime normal case 0 and fault case 1.

(Refer Slide Time: 37:23)

D-Algorithm: Simulation and Implication	
Simulation	Implication
All the signal values are determined uniquely	All signals may not be determined uniquely
Value assignment moves from inputs to outputs of a circuit 	Signal assignments propagate both towards primary inputs and primary outputs. For example, to test a s-a-1 fault at net l say, we need to have implication in two directions (i) backwards, to primary inputs to make $l=0$ and (ii) forward, to primary outputs to propagate D.
There is no inconsistency 	Inconsistency may arise, when for a given net l (which is not the fault location) different signal values (0 or 1) need to be assigned.

So, I mean I am just sorry, so in, so we are talking about the AND gate. So, in the AND gate this is sachtet 1. So, you apply a 0, you need a 0 over here. So, this normal case it is 0, fault case it is 1, so this is D prime. So, we know the value of the output of d prime and both the inputs are x. So, it has to be output should be 0. So, you can have x 0 or 0 x anyone one of them is fine.

So, that is what saying that in case of simulation we if you have 0 0 the input of the AND gate we get the output as 0. So, the unidirectional flow but, in case of implication this is both way, first we have sensitize this faults then this actually becomes D frontier kind of a thing, so you will get the d again we justify this 1. So, because it becomes J frontier, so it is actual moving backward and forward. So, backward is may be primary input 0, for what we propagate the primary outputs this d prime has to be propagate to some primary outputs.

So, in case of implication sometimes we have to sensitize the default and then move the default where is the output is forward direction and backward direction because for D frontiers are done to get J frontiers. So, you have to take the value backward and this is and that is up to for justification.

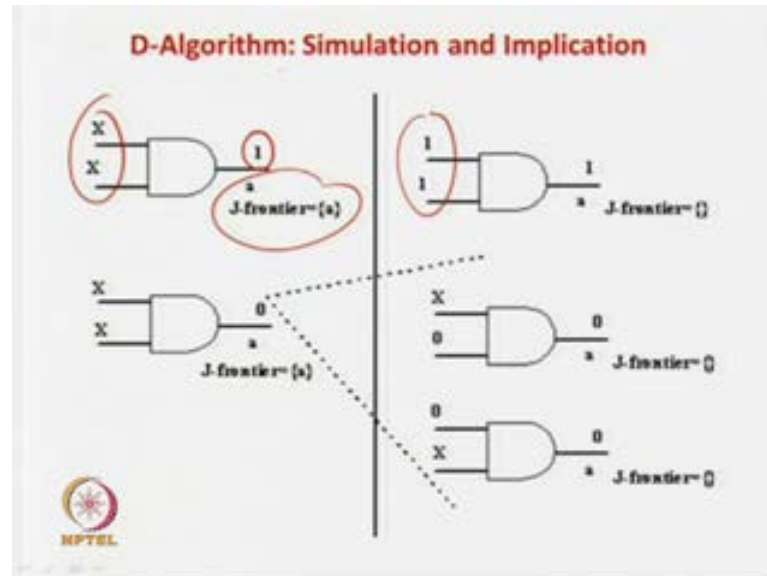
(Refer Slide Time: 38:33)

Simulation	Implication
All the signal values are determined uniquely	All signals may not be determined uniquely
Value assignment moves from inputs to outputs of a circuit	Signal assignments propagate both towards primary inputs and primary outputs. For example, to test a s-a-1 fault at net l say, we need to have implication in two directions (i) backwards, to primary inputs to make $l=0$ and (ii) forward, to primary outputs to propagate D.
There is no inconsistency	Inconsistency may arise, when for a given net l (which is not the fault location) different signal values (0 or 1) need to be assigned.

So, what is the difference between simulation and implication another difference is in case of simulation there is no inconsistency, because you give a input and the output you

can determined uniquely right. And there is no question of inconsistency but, in case of a implication then the lot of inconsistency can be there.

(Refer Slide Time: 38:44)



Because if the same net you have to assign a 0 and you have to assign a 1. So, say net involved in propagate your default value and you say that you have to assign a 1 and but, for justification case it says that you apply 0 such cases we have already seen in the last lecture also be seeing in details in next lecture. So, that is your inconsistency but, in case of simulation, there is no questions of any kind of inconsistency, because some inputs are there you get the output a unique output.

So, now let us see some example of definitions and the procedure you have seen because some of bit complex. So, we just try to illustrate that with in some examples. So, in this case say this is the AND gate, so this output is 1. So, when the output is known then the inputs are x and x. So, the output is known then we say that this is actually here J frontier J frontiers means, I already told you that output is known but, both the inputs or single input may be x.

(Refer Slide Time: 39:44)

D-Algorithm: Definitions and procedures

Definition 1: Singular cover

Singular cover of a logic gate is the minimal set of input signal assignments needed to represent essential prime implicants in the Karnaugh map of the logic gate, both for the case 0 and case 1.

Table shows singular cover for 2 input AND and 2 input OR gate.

AND	Inputs		Output		OR	Inputs		Output
	A	B				A	B	
	0	X	0			1	X	1
	X	0	0			X	1	1
	1	1	1			0	0	0

So, we know that the output is 1 both of them must be 1 1 that can be covered from the first in what do you called this first. I mean this slide we have shown this second slide that is on singular cover that is 11 in case of an AND gate this is the singular cover.

(Refer Slide Time: 39:53)

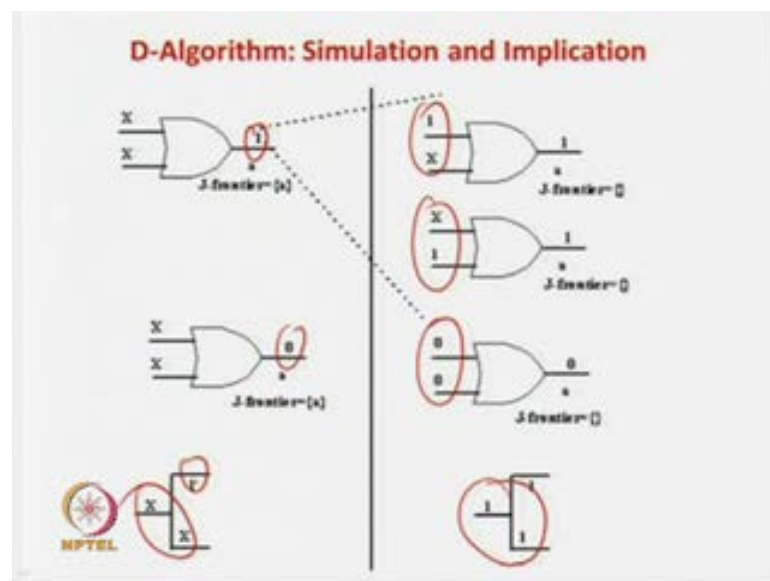
D-Algorithm: Simulation and Implication

The diagram illustrates the simulation and implication process for an AND gate. It shows four scenarios of input assignments (X, 0, 1) and their corresponding outputs (1, 0). Red circles highlight the 'J-frontier' for each output. A vertical line separates the two sides, and a dashed line indicates the flow of information from the output back to the inputs.

So, that is actually we have to know this table when you are doing this J frontier and D frontier this nets. So, 1 1 for this one to defeat this gate from the J frontier because I mean we have we got this 1 and we have justified this 1 and 1. So, this gate cannot be used anymore for justification.

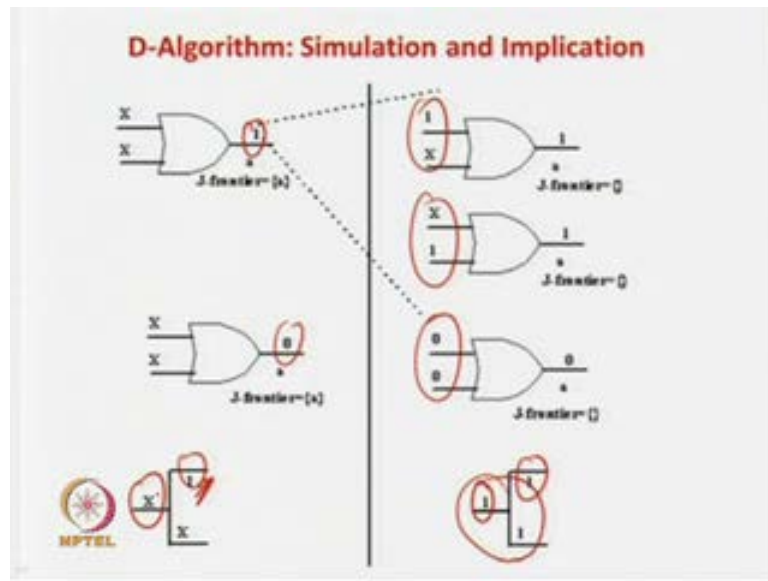
Now, if you see that this and gate output is 0, so this two values of this singular cover. So, you can take any one of them for justification in a backward direction because we already seen that in case of J frontier that is by implication is backward traversal and forward traversal, J frontier corresponds to backwards traversal. So, you can use any one of them and then you have to delete these list from the J frontier, because this already been used.

(Refer Slide Time: 40:28)



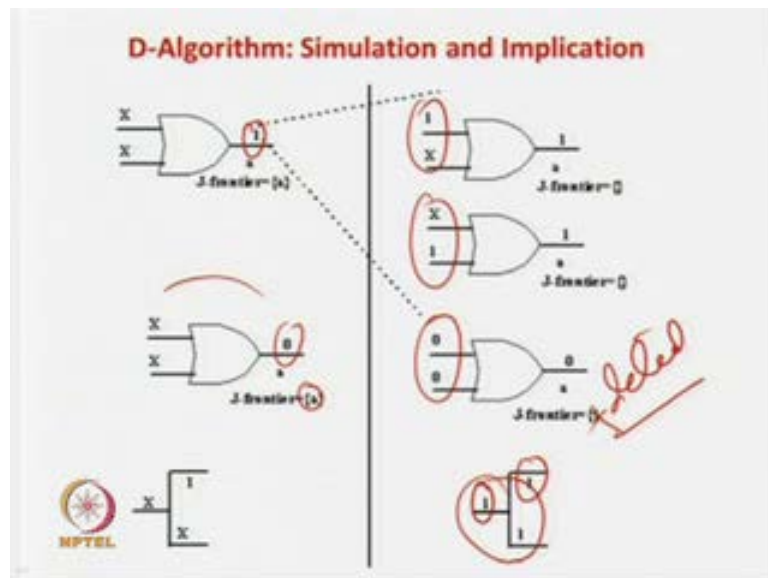
Similar you can take for the OR gate, so OR gate 1; that means, there are two choices, so you can take any one of them. So, OR gate 0 there is only one choice, so after that is on the case and this is for fan out is very simple, so if this is 1. So, then this implications, so this one both of them has to be a 1 1, so this from this fan out is very simple, because may this one is to be 1.

(Refer Slide Time: 40:51)



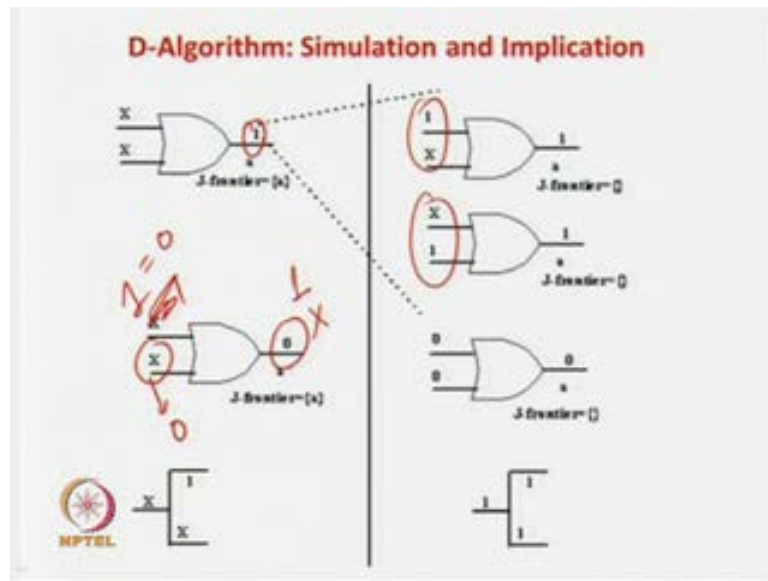
Then you know that actually says that one of this output is known then you have to determined this 1. So, if this output is known; obviously, this 1 will be 1 because this J frontier definitions say the output is 1 and the input 1 input or many inputs can be x. So, in this case in this case you can see that this is the only one input because the fan output.

(Refer Slide Time: 41:10)



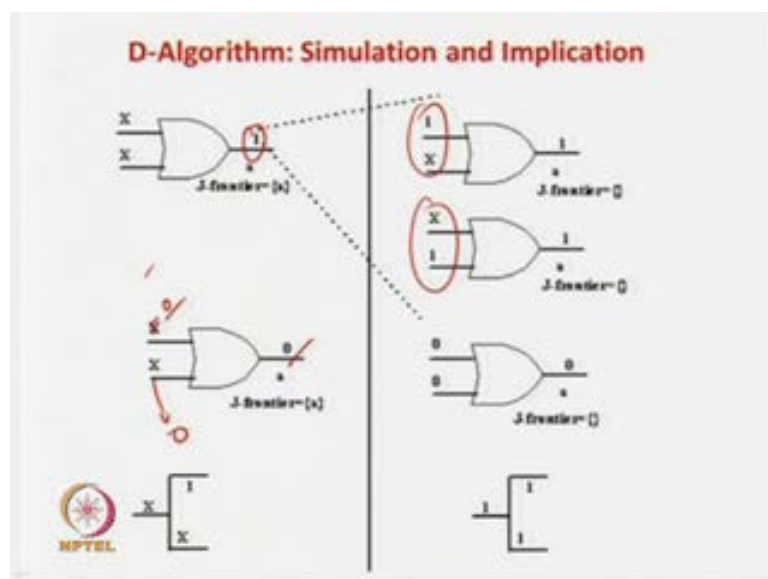
So, this can be set like this and one more thing you have to know that this a, so wherever this justification has been done, so this has to be deleted. So, this gate has to be deleted from the j frontier because this gate already has been used for justification.

(Refer Slide Time: 41:25)



So, sometimes here both the value of x , so it can be justify sometime it may not be possible like for some reason as I told you it may be one then what happens it says the this x this is not x this is 1. Still it will be for J, frontier because here still this 0 x 0 because to justify the values of this 0 then what we will do this x try to make it 0 but, then it is actually inconsistency below making this x 0 will not help because this also leads to be 0 but, scope is not there and then this actually net will become in going to inconsistency. So, already we discussed that always J frontier I mean J frontier have a scope to actually justify the output but, always it may not be helpful.

(Refer Slide Time: 42:01)




But if in the case this x is 0, then this same gate still is in the J frontier with the outputs is 0 and input is the case. So, this input you make it to 0 and it should be solving your purpose.

(Refer Slide Time: 42:15)

D-Algorithm: Simulation and Implication

The second gate has output of 0 and so by singular cover we have two options for assigning values to the inputs; the two options are shown in the right side.

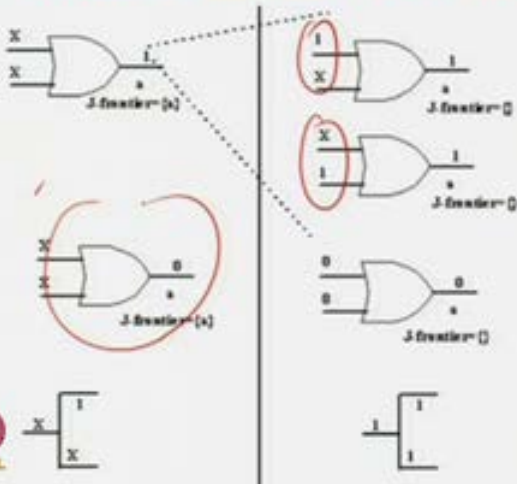
It may be noted that we could have also kept another option where both the inputs are kept 1. However, this is avoided because if both inputs are fixed then flexibility gets reduced and chances of inconsistency increases.



But always a gate in a J frontier may not solve your purpose that may be inconsistency but, still a gate in the J frontier means some scope that it can be used for backward propagations or something backward propagation and can be may be used for justification which may give you successful but, also may lead to inconsistency.

(Refer Slide Time: 42:33)

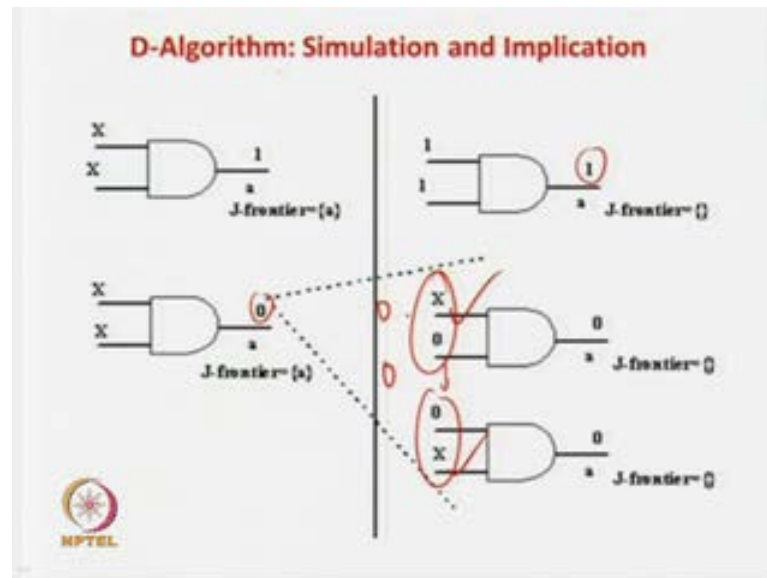
D-Algorithm: Simulation and Implication



The diagram illustrates the D-Algorithm simulation and implication. It shows a sequence of gates and their J-frontier states. On the left, a gate with inputs X and X and output 1 is shown with J-frontier={x}. Below it, a gate with inputs X and X and output 0 is shown with J-frontier={x}. On the right, a gate with inputs 1 and X and output 1 is shown with J-frontier=(). Below it, a gate with inputs X and 1 and output 1 is shown with J-frontier=(). Below that, a gate with inputs 0 and 0 and output 0 is shown with J-frontier=(). At the bottom, a truth table for the gate is shown with inputs X and X, and outputs 1 and 0.

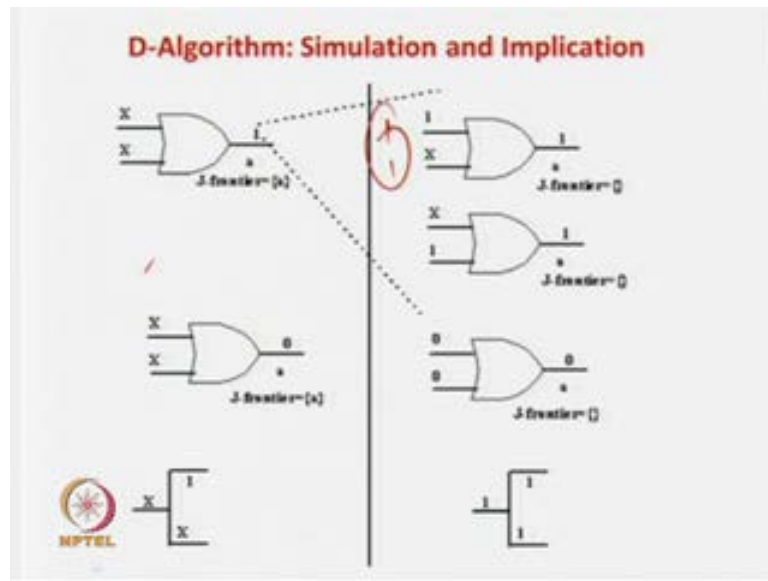
Now, so, this is what is been seeing this second gate has like this saying that this second gate sorry what is does say. So, the second gate has output of 0, so by singular cover we have two options for assigning values to the inputs, two options are shown in the right side. So, it is sorry we are talking about this gate actually this two gates, so you see that the output is one kind of a thing over here.

(Refer Slide Time: 42:58)



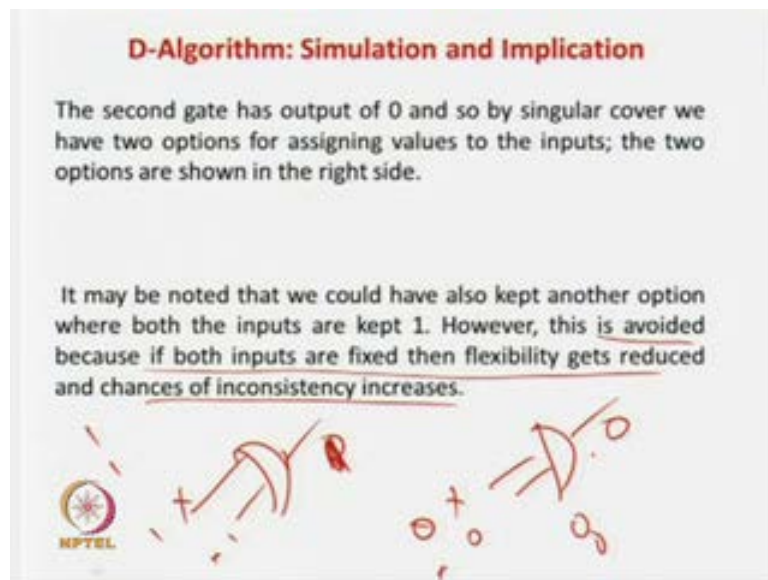
So, here one is also the case here, so that we are talking about this general case that is second gate over here it is second AND gate, so it is saying that the output is 0 over here, so there we are having two choices. So, which one to use or you can also take this third choice it is 0 and 0, so why not used. So, this is either used this or this it is not saying that you do not use 0 0 now why in this case, so that this is one portion definite in our mind.

(Refer Slide Time: 43:13)



Similarly, in this case 1 1 is 1 choice which is left which you are not using. So, why we are not using that is very important we will see.

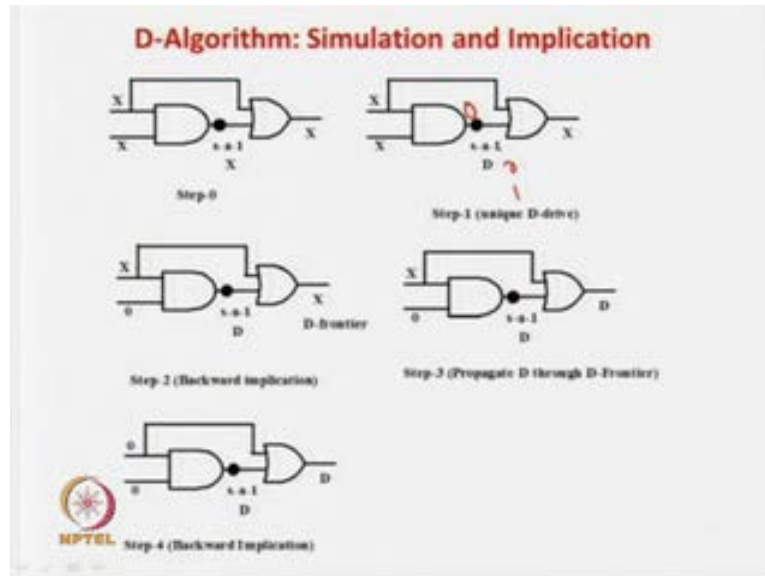
(Refer Slide Time: 43:23)



So, we could have another option to keep the input has 1 but however, this is avoided if both the inputs are fixed then the flexibility is lost and cases of inconsistency increase. That is what they are saying that if this is 0 output of an and gate is 0 you take x 0 or 0 x if is for an OR gate sorry if is an OR gate like this, if the output is 0 sorry output is 1 sorry output is 1 of an your OR gate.

So, you take $x = 1$ or $1 \times$ the y they do not try to avoid $1 \ 1$ and $0 \ 0$ here in this case. Now, why it is said it says that if you have a chord if it is covered by J frontier for back propagation I mean in backward track I mean you justifying using the J frontier gates.

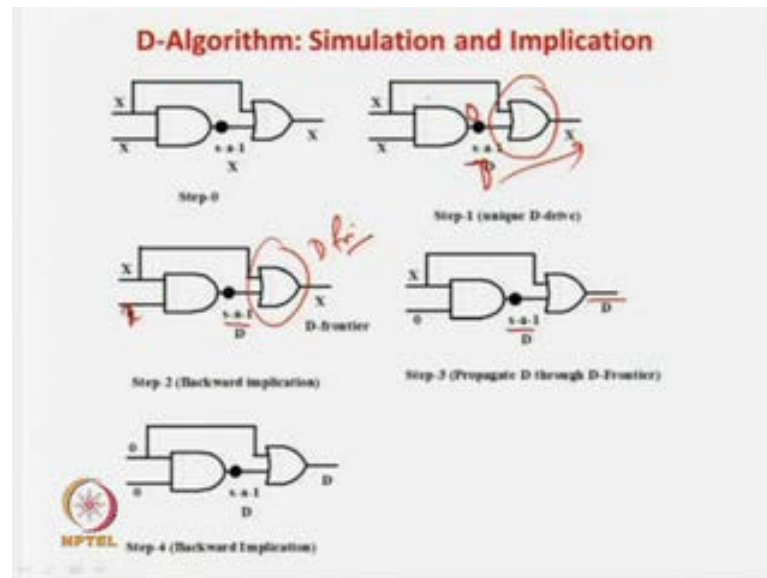
(Refer Slide Time: 44:08)



So, they say that if you put the value $0 \ 0$ if you put the value $1 \ 1$ then actually you are trying to make some of the things hard chorded. Some of the nature you are keeping hard chorded and then what happens hard chord the nets to much then the flexibility is lost and the chances of inconsistency are higher.

So, we will see that by an example, so what is the example over here, so is show that this is a AND gate, so small sorry circuit, so this sachet 1. So, initially all the nets will be 0 this steps 0, so in step in one we have sachet 1. So, just sachet 1; obviously, have to apply a 0 over here. So, 0 means normal case 0 fault case 1, so it is it will be a sorry sachet 1, so normal case 0 fault case 1. So, in this case sorry it will be a your D prime.

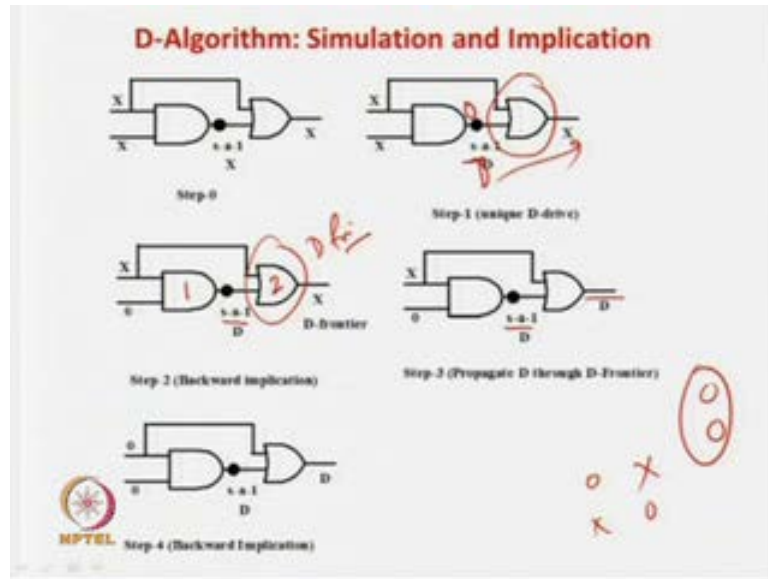
(Refer Slide Time: 44:46)



So, is a D prime over here, now this is there, now what is, so the is only unique say D frontier. So, this is the only only gate where d affected and it propagated, so the unique d cover, so it called unique d drive so; obviously, this is to be taken, so sachet 1. Now, this x, so this only this gate is only D frontier, so you will have you have to propagate, now in this case let us see, so this is the propagation.

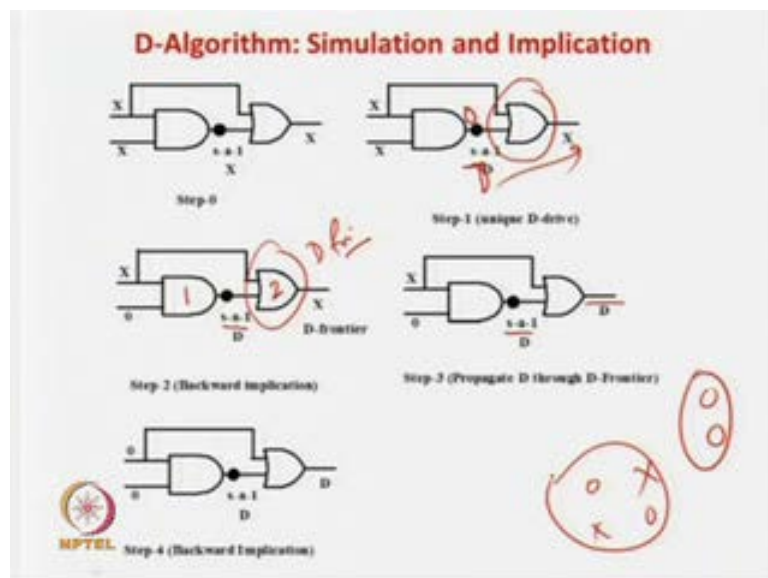
Now, in this case if you look at, so this is the only D drive. So, this is the only D frontier now this gate for this gate you know the output and both the inputs are, so it is d and the both the inputs were x if you look at this this output is re prime we know and both the inputs are 0 sorry both the inputs are x, so both the inputs are x, so for x.

(Refer Slide Time: 35:34)



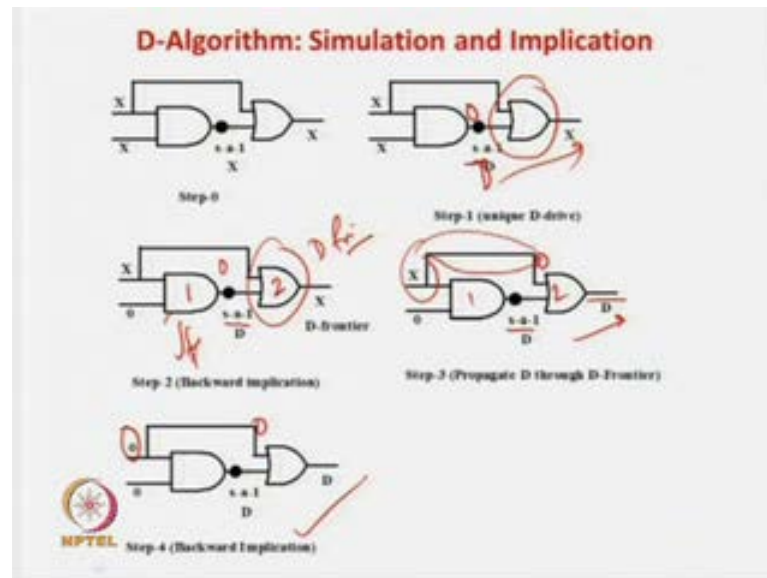
So, this what and you have to get the output this is say the gate 1, say this is gate two. So, output of the gate 1 should be 0, now there are three choices basically x 0 0 x . So, these are the two choices allowed by definition, so we have also asking the question that if you take both of them 0 what is the problem. So, let us go for the choice x 0 will see the problem later if you use this. So, x 0 may be there, so here usually the choice x 0 because the output is.

(Refer Slide Time: 46:11)



So, just a minute, so yeah, so let this be the case, so $x = 0$. So, let us first look at this two choices and the other choice problems and all will see later that is heart chording of this signals will see later.

(Refer Slide Time: 46:22)



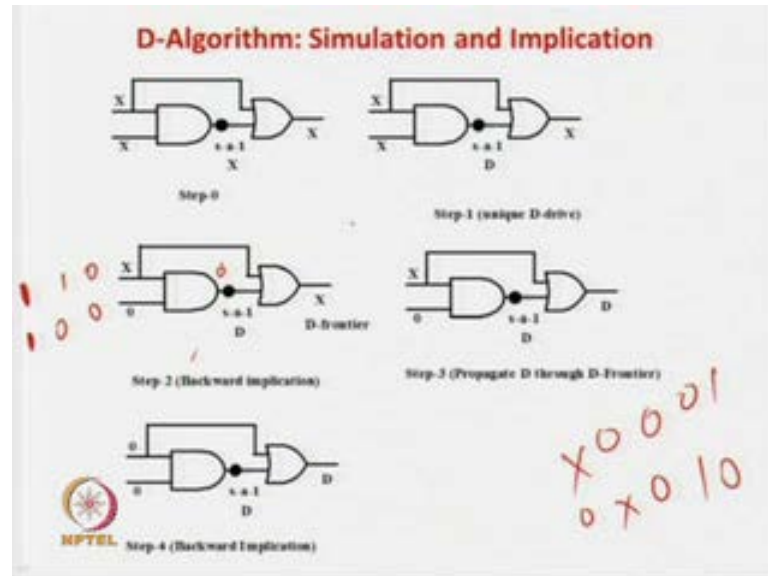
So, now in this case this is d' , so $x = 0$ the choice they have taken, now that is this is in the J frontier this gate is in J frontier. So, the output is d' , so and we require a 0 over here, so this inputs are not known that is the x and \bar{x} , so you can take $x = 0$ or $0 = x$. So, they have taken $x = 0$, so now this is done now this is x and this is the case now this is the gate, so now this is d' correct. So, this is a d' over here, so now this this affect have been propagated from this 1, so this gate 1, so gate two gate two was the j frontier.

So, immediately this affect d' has been propagated to this 1. So, this gate is deleted from the D frontier, so now we have to this net this this part is now will become your j frontier because we know that that this values has to be a 0 because for an or gate this value will propagate area. So, this value has to be 0 now this net this fan out net will become a J frontier. So, we know that by this j frontier rule of the fan nets, so this will has to be a 0 this is to be a 0.

So, now if this in backward implication and it is done, so this way it solves a problem of your implication that is that is already we have seen. So, this is simple D algorithm basic

idea of the D algorithm we have shown here now we see that they were not inconsistency in this case.

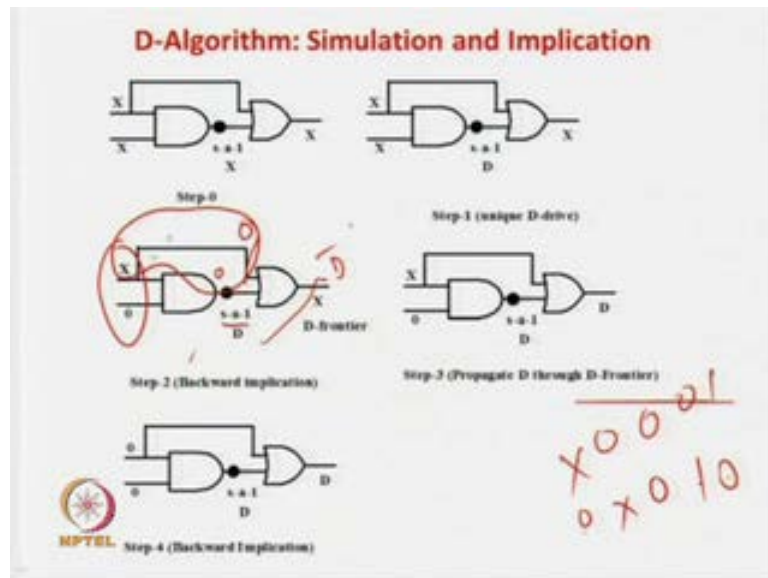
(Refer Slide Time: 47:43)



Now, so this was one example, so will see later that what is the problem or what may be the problem if you try to hard chord you said of this $x\ 0$ if you put this case or if you put the case $1\ 0$ this also possible also you can have take you could have taken the case has because we require anywhere this d prime. So, only we require 0 over here. So, $0\ x$ is the case $0\ 0$ is 1 case $1\ 0$ you can take or also you can take $0\ 1$. So, anyone of this hard chording sorry, so this is one.

So, any one of the choice like the choice is $x\ 0\ 0\ x\ 0\ 0$ you can also take $0\ 1$ and $1\ 0$. So, any of these choices like $1, 2, 3, 4, 5$ is apply over here you can get the value of 0 over here correct.

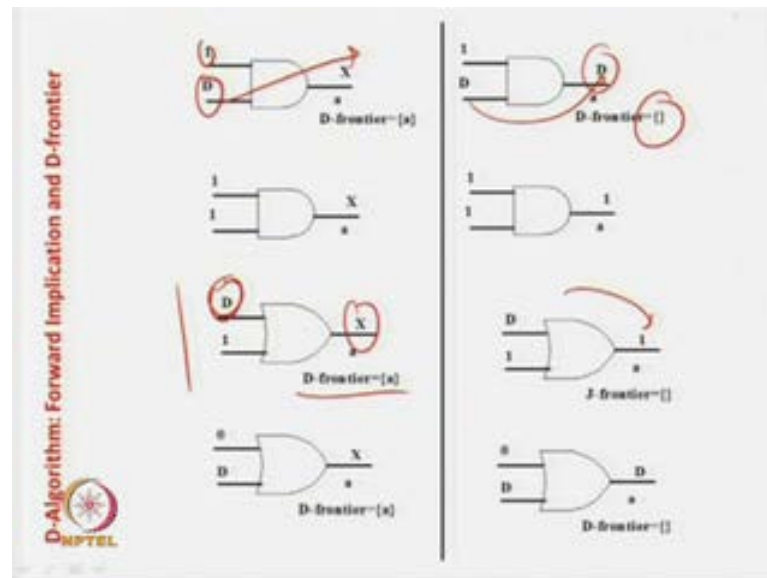
(Refer Slide Time: 48:25)



That is the output of this gate you can get this same that mean this is this, this, this, this, this (Refer Slide Time: 48:23) any of the choices you apply in this case you get a 0 over here. So, you will get a d prime over here, now this will become your D frontier. So, you have to propagate the values, so it will become d frontier, so it will propagate the value here. So, once this I mean d d D prime is propagated here this will no longer remains in the d frontier then you require a 0 over here and then this will become your J frontier and x will gets some value kind of a thing.

So, in this this example we have seen that there is no inconsistency because we have use the case of x 0 or 0 x whatever that was been allowed. So, later we see what may be the problems if you try to use the strict cases like 0 0 1 0 0 1 instead of keeping the x that is will see later.

(Refer Slide Time: 49:04)

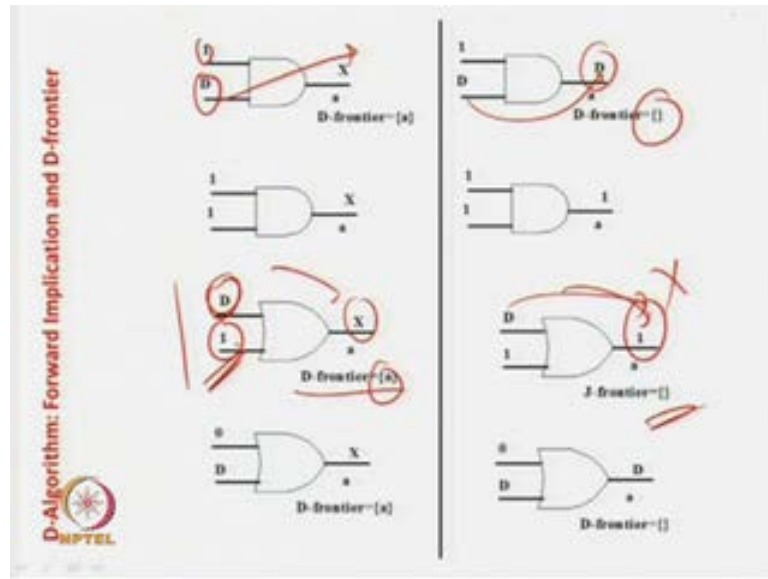


But it was just scheme just example the elastic or the implication procedure words and also miniature version of mini of a very broad version of D algorithm some more definitions are actually example of the definitions or procedure we have used. So, say for example, in this case same like it is a example of D frontier. So, I mean if it is a d and this is a 1 . So, this is the unique, so you can drive this 1 from here. So, the output is x and you know the input is input is d . So, you can easily value the drive this value of this 1 from here, so again the D frontier has to be is lost, so this we are propagating the value then this is the lost.

So, other things are very straight forward this is just a value computation this another example we can also very simply says that say 0 1 and the d . So, the output will be say for example, the output is actually in this case output is the d frontier is this 1 . So, either D frontier because this d is over there this output is x . So, you can easily propagate the value as 1 but, in this case again this is very this very important point you have to note that this a d frontier like J frontier.

Something always all gates in D frontier always all gates in the J frontier may not help because you have seen many examples of J frontiers. In which case this gates were in J frontier but, is not helping similarly this also happen in case of d frontier like in this case is the OR gate.

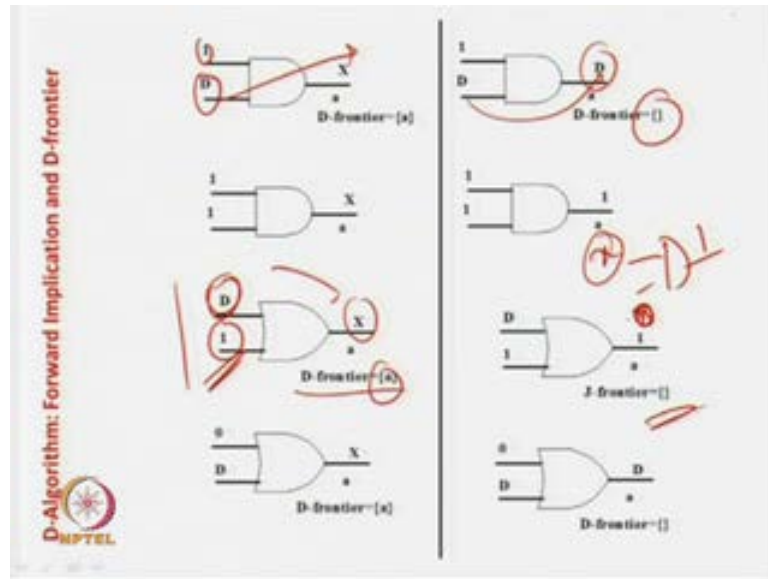
(Refer Slide Time: 50:28)



So, this is the d, so d the values can be propagated the output is x, so the value of d can be propagated. So, it can be assume then another input is 1 because of some reason, so this gate is in D frontier to the output is 1. So, the value of this 1 is not propagated and again this gate will be lost from the D frontier. So, you you considered these gate to be in d frontier because by definition the output is x the input is this. So, there is a scope of propagating he value but, because this input is 1, so nothing could have been done.

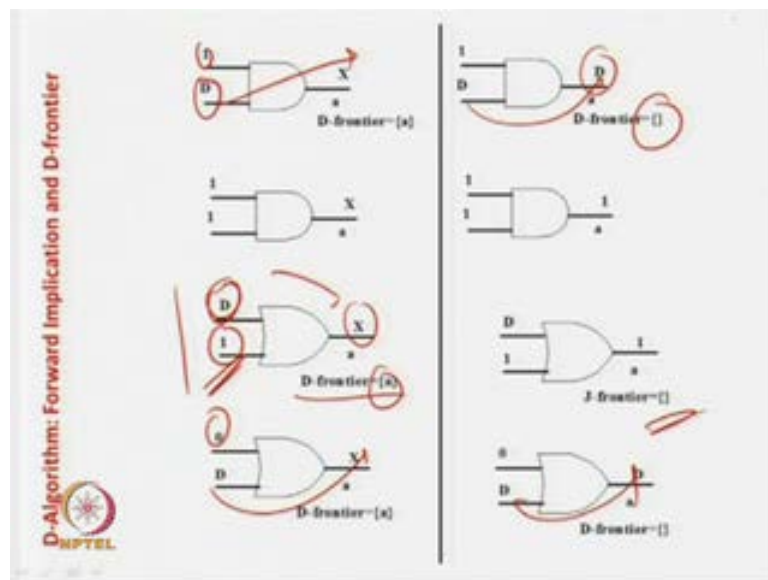
So, as this gate was just to that this gate was in D frontier or it could not do much. So, why could not do much because the value of d was not propagated, because of this one but, still we have by definition such case will be in D frontier but, when you are be trying to propagate the value the output will find already the output is not propagating. So, such type of gate are examples of gates in D frontier which is not of much, similarly, also by definition we have also seen that some cases of gates in the J frontier which we thought that some of the value affect I mean the value could be justify but, it could not be.

(Refer Slide Time: 51:22)



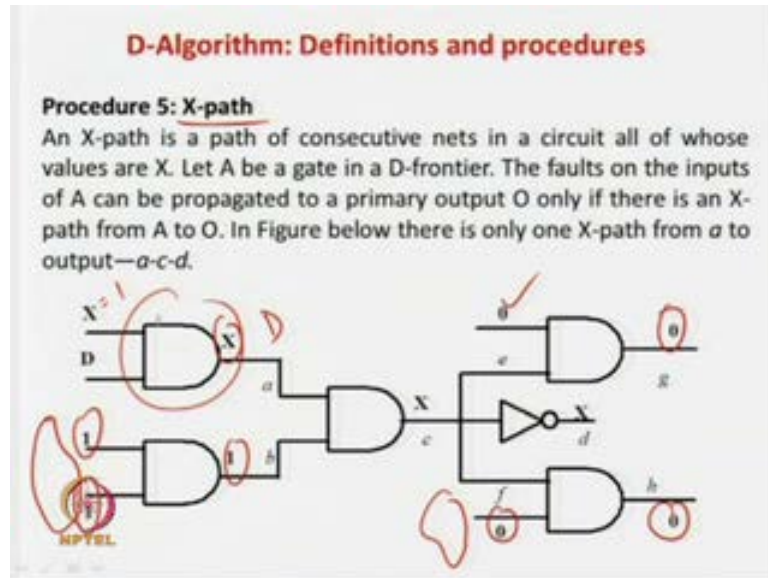
Because of this for example, like already many times we discuss in lectures that it is we require a 0 over here sorry we require a 1 over here. So, this is was x and this was 0, so this was x. So, and one input is 0 the output required is 1. So, we also considered this gate is in D frontier because these some scope that we may say to get the values 1 or in fact it is not be successful, because this one input is 0 which will make the output of the gate as 1. So, this was one example, which a gate is in J frontier but, not helping it as similar this is an example of gate in D frontier but, not helping in the propagation.

(Refer Slide Time: 51:54)



So, this is request to be, so if it is 0 then the d affect will be propagated and this 1 will help in this 1 correct.

(Refer Slide Time: 51:58)



So, I mean sometimes the lots of gates were be in, so in other words the lots of gates in J frontier and D frontier but, they always not be successful or always not be I mean lead to fault propagation in case of D frontier and always they ay not lead to some successful justification in case of J frontier. So, there will be lots of gates in J and D frontiers but, some may be successful some may not be successful that is what is the shown were the elastration and this example.

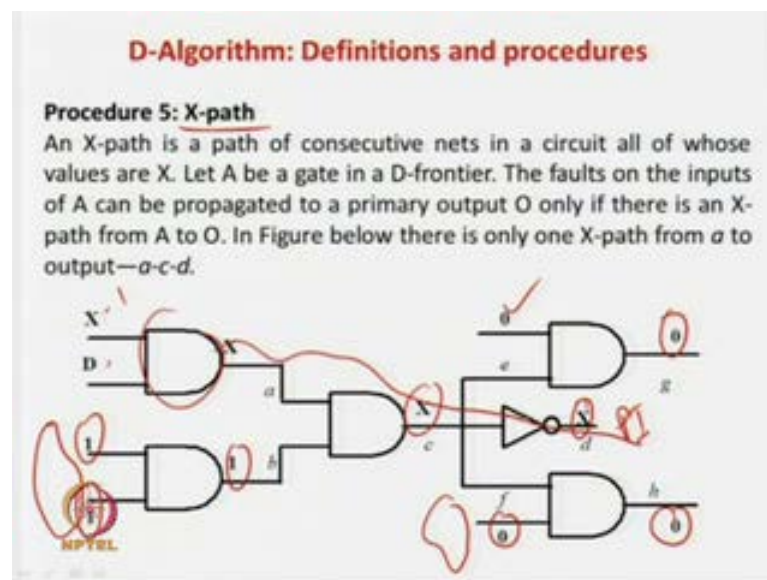
So, before I mean with the help of this definition, so this one more definition that is actually called the x path. So, as we already discuss that on the inputs of your nets circuit. So, initial we mark it as x then by by some if you know the values of some of the nets and that is the implication we try to find the values. So, for example, let us have this circuit, so let us know that this some for some reason we know that let us assume that we know the values 1 1, so by implication this 1 will be 1.

So, let be this x and d, so the output will be x, so this is the case for and by some reason let us assume that we know the value of this 1 to be a 0. So, the output is 0 to be in known by implication. So, this value let us assume for by some means we assume that they are not primary input may be by some either combination and logic as input or

because of some other reasons we know the value to be 0 by implication also we know the value to be 0 directly.

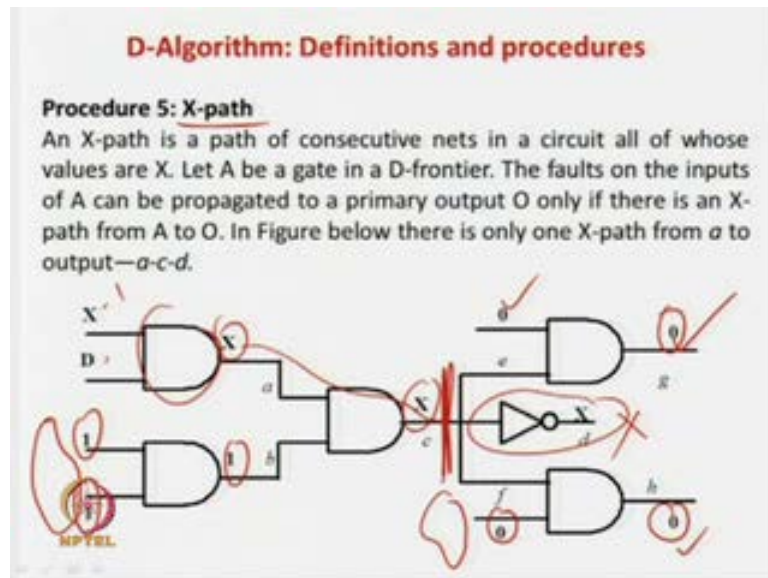
Now, in this case you see that this only 1 value as x. So, in this case you can consider that this is the D frontier because this because this is output is x you can propagate you can propagate the value of this 1 can be d. So, once it is d then this gate will become J frontier then you have to get the value as x as 1 and so forth. In our implication procedure goes forward but, for time being, so what is say that.

(Refer Slide Time: 53:36)



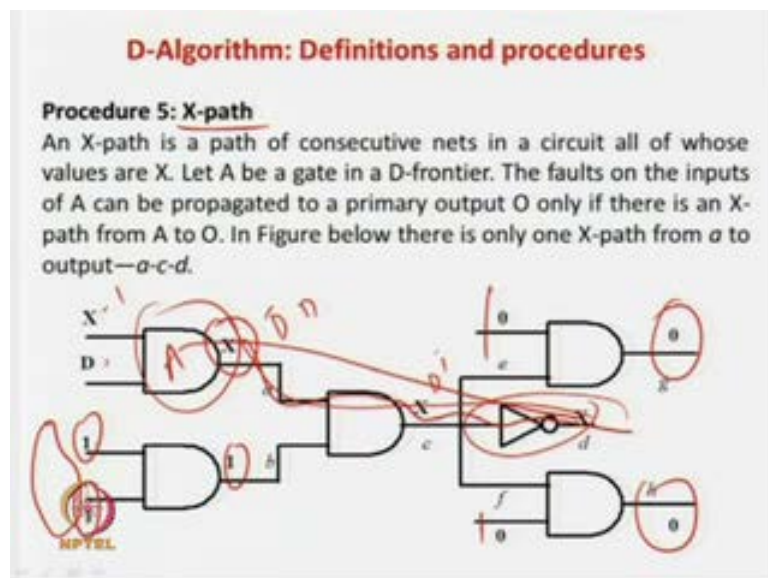
In this case if you look at it quickly, so this 1 x over here, so if it is x, so then 1 more x over here and this is invertors, so this is one more x over here and we considered that these are the primary output. So, there is a path from this d frontier there is a path from here to the primary output where everybody is on x, s, his at least one path, so if there is no path like this.

(Refer Slide Time: 53:56)



Say for example have this this gate not been there, so have this gate not been there then what will happen there is no path from this D frontier which is all come arrange to the x gate that is output of the gates. So, if it will come here some other this 1 is also 1 and this 1 is also 1. So, after this fault propagation will definite the (()) here. So, that is in other words what we have saying that there is a D frontier or in this example we are getting a single D frontier.

(Refer Slide Time: 54:20)



You say that from this D frontier this at least one path where all the outputs of the gates are x; that means, this is the only path which can be used for propagating fault definition because either this x can be D or D prime this x can be x or D prime this x can be converted into D and D prime. So, this for this will propagate for only to this path but, this path and this path the outputs are not x so; that means, if it cannot be anything and this x path is not there. So, fault propagation will be terminated at this path or this path. So, it says that x path is an path of consecutive nets in this circuit to all good values are x.

So, let a be in D frontier because this is the gate a staying in D frontier the outputs or the inputs can be propagated to the primary output o, only this is x path from a to o that is saying that this is the input and this is the output. So, from the gate with the D frontier that is fault effect is there. So, that effect can be propagate to the primary output if and only if all these path of gates were outputs all are x, because this x can be converted into d or d prime and default effect can be propagated.

(Refer Slide Time: 55:23)

D-Algorithm


Input: Circuit netlist and one stuck at fault with its location.

Output: Primary input values and expected values at the primary outputs

Step-1: Initialize all the signal values of the circuit to X.

Step-2: Sensitize the fault location, i.e., if s-a-0 is the fault, apply D in the fault location, else if s-a-1 is the fault, apply \bar{D} in the fault location.

Step 3: Determine/update gates in D-frontier and J-frontier, due to signal changes from X to 0,1, D or \bar{D} . If there is no gate in D-frontier and D/ \bar{D} has not reached any primary output, backtrack().

 NPTEL

But, if there is no gates no path like this, then we know that the output then the idea is that you cannot propagate the value to the output and actually in fact your lost by this one. So, that is that is all the definition we require for knowing the D algorithm and next what we will study actually the D algorithm in detail that is your circuit net list and

primary output values and expected values at the primary outputs. That is circuit net list and one stuck at fault and the output.

And the primary input values require test the defaults and what is the expected the primary output that is your Automatic Test Pattern you call this ATP. So, in next lectures using all the definitions we have studied today will try to elaborate and look formally into this D algorithm using this definition and the procedure which are discussed to be.

So, using that will try to formally adequate D algorithm and then you have to be do then in very I mean what you call in details we take an examples. And try to find out how D algorithm basically works on that algorithm using implication J frontier and D frontier propagation. So, that is what we have for today in next lecture we go in elaborate D algorithm D algorithm with example,

Thank you.