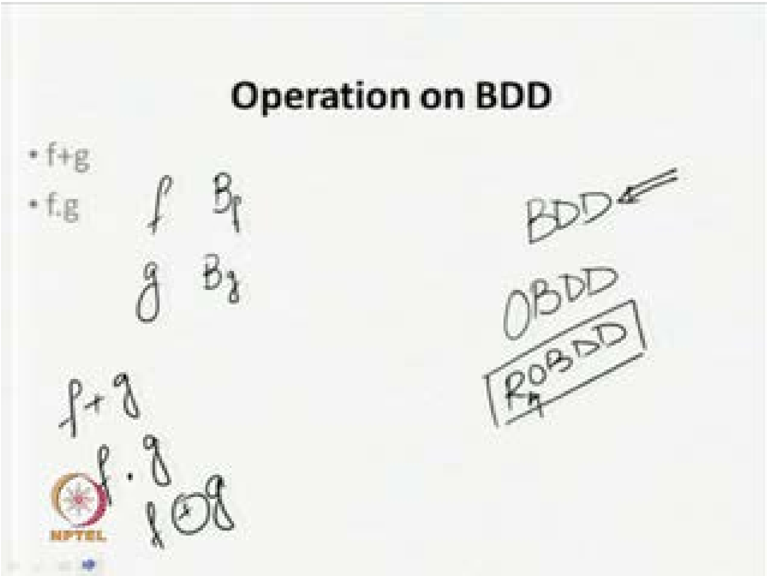
Design Verification and Test of Digital VLSI Designs
Prof. Dr. Santosh Biswas
Prof. Dr. Jatindra Kumar Deka
Indian Institute of Technology, Guwahati

Module - 6
Binary Decision Diagram
Lecture - 3
Operation on Ordered Binary Decision Diagram

(Refer Slide Time: 00:40)

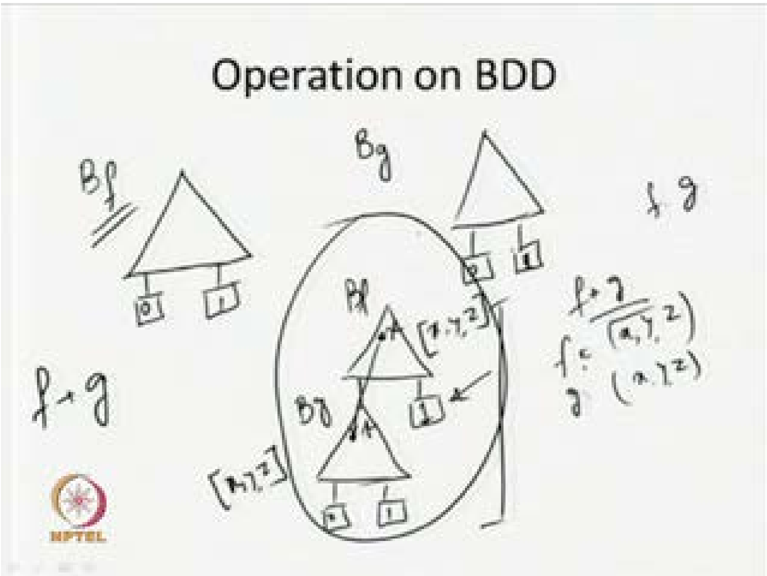


We are discussing about a data structure called binary decision diagram, and these binary decision diagram is used to represent any Boolean function. Now, today we are going to see some operation that can be perform on ordered binary decision diagram, because we have already discussed about say BDD we have introduced. What is your BDD? Binary decision diagram, then we have talked about OBDD - ordered binary decision diagram. In ordered binary decision diagram, we are going to maintain a particular ordering of that variables.

And after that we have talked about ROBDD - reduced ordered binary decision diagram; that means, this is the reduced form order reduced BDD of a given particular Boolean function which follow a particular ordering of that variables. And we have seen on property at the ROBDD of a given function is always unique; that means, it is the canonical representation of a given Boolean function. So with a particular variable ordering we are going to get a unique structure, unique BDD structure for that particular function. If we sense the BDD sorry, if we sense the variable ordering that we may get another structure.

Now, say when I am having say two function f and g, then the BDD representation of f will be your B f and the BDD representation of g will be your B g. Now what happens we can perform the operation f plus g or you can perform the operation f dot g or like that f expressive or g like that. When we have talked about BDD - binary decision diagram at that time we have saying that we can perform those operation on BDD ourselves.

(Refer Slide Time: 02:18)



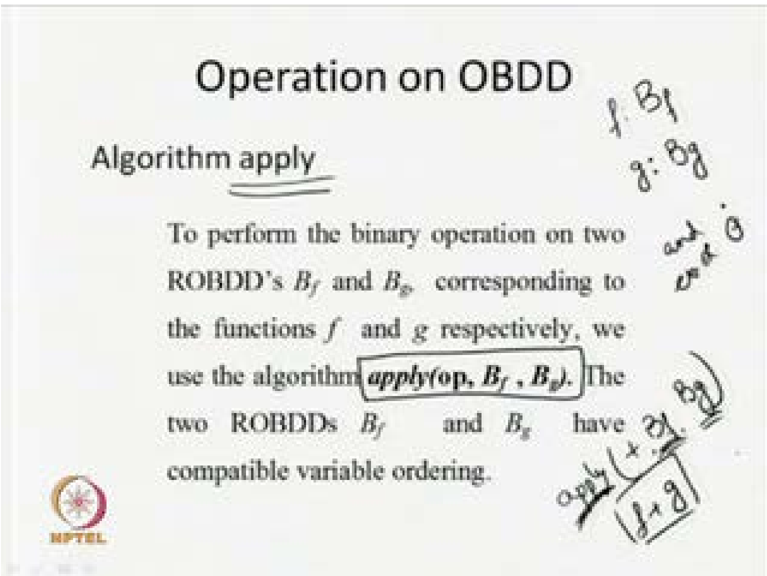
So what we have seen on that particular case? We have seen that if I am having a function B d f and the BDD representation of your f is B f then B f I can say that this is the structure and eventually we are having the terminal node 0, and 1 similarly I am having for the BDD B g of a function g, again I can say that I am having a BDD representation of this function and these two other terminal nodes.

Now when I am going to perform say f plus g, then what will happen? You can perform these operation on BDD what we can say that this is your B f. So if your function the if you value of the function f is 1 then f plus g will be 1 but, if value of f is 0 then what happens? You have to now look for a valuation of your function g. So in that particular case what happens? We have seen that in this particular 0 node we are going to put this particular B g g BDD B f, B g and 0 and 1 these are terminal nodes. So one if f is 1 then we are going to get that a functional value as 1 and if f is 0 then we are going to look for the what is the valuation of g depending on that I am going to get the functional value for f plus g.

Similarly, we can go for f dot g also but, what is the problem that we are going to get over here, you just see that f is a function say which is having some variable say x, y, z. Similarly, g is also function which is having say variable x, y, z. I am going to say send is triple, say you just said BDD B, f is an order variable, ordered binary decision diagram; that means, it is having the variable ordering x, y, z. Similarly, that B g is also having variable ordering say x, y, z but when we construct this particular BDD in that particular case what will happen?

We are not going to get this ordering of the variable x, y, z because somewhere here x is appearing and somewhere here also x is appearing. So in one particular path x is appearing twice, which is the valuation of your ordered binary decision diagram. So; that means, if we are simply going to construct the BDD for f plus g by taking the BDD B g which your B f then we are not going to get an ordered BDD; that means, the resultant BDD is not an ordered BDD.

(Refer Slide Time: 05:32)

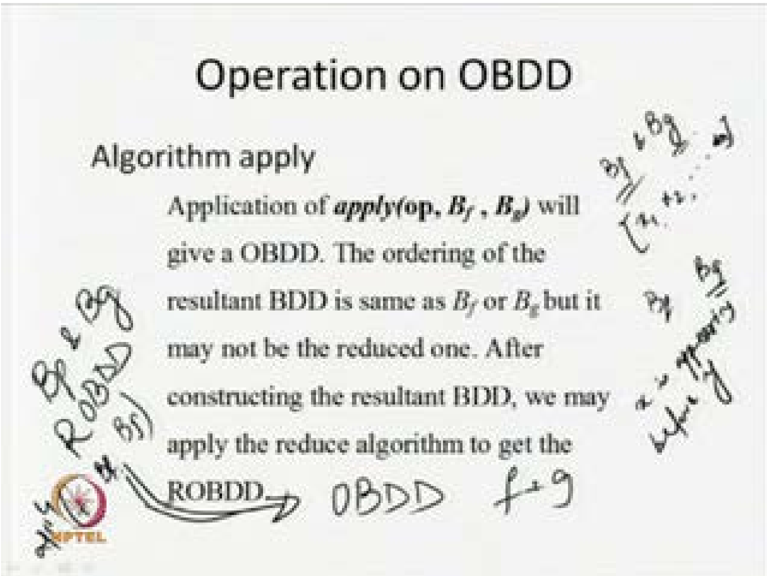


So for that simply we cannot use this particular operation that we use for your BDD. So for ordered BDD we have to handle it differently. So today's lecture we are going to see how we can perform those particular Boolean operation on the BDD's or in particular ordered binary decision diagram. Now just see that what we are going to see in this particular case so, the operation for any kind of operation, we are going to use an algorithm called apply. This apply algorithm will be used to perform any Boolean operation on BDD's. So that means, if we are applying these things on BDD's we are going to get some resultant BDD which is going to give us the evaluation of this particular Boolean operator.

So the apply algorithm is going to take to BDD say for f I am having B f one BDD representation of f and I am having say B g BDD representation of g, then the algorithm apply will take this particular form it is having three parameter apply operation B f and B g. So you can give any binary operator over here and two BDD's. So, B f and B g are BDD representation and f and g.

Now, if I use this particular function apply say plus B f, B g in this particular case what will happen? We are going to perform the all operation on Boolean function f and g. So f plus g and after that we are going to represent this things with our BDD's or ROBDD's. So in this particular case we are having the BDD representation of Boolean function f and BDD representation of g. So will use this particular apply algorithm to perform this particular addition or all operation.

(Refer Slide Time: 07:53)



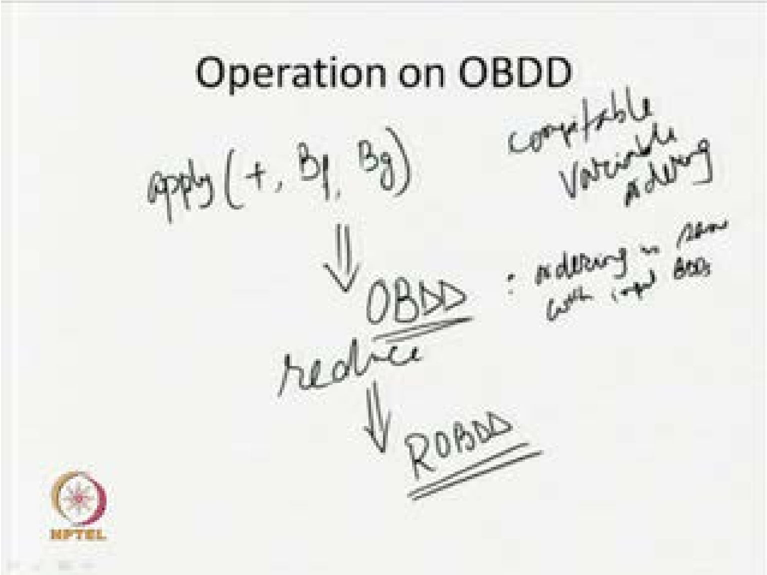
Similarly, we can use and we can use excel operation like that in this particular operator we can use any operator and this apply algorithm is going to perform or going to give us the BDD which is going to represent this particular function f plus g or f dot g. Now where I am going to talk about your f g or f B f there is other BDD's. So these are ordered binary decision diagram and these two binary decision diagram must have compatible variable ordering.

So what does it means incase of compatible variable ordering both the BDD's must have same variable ordering; that means, if one is having same variable ordering x, 2 x two like that x n they are both B f and B g should follow this particular same variable ordering. So; that means, what we say that compatible variable ordering say if I am having two BDD B f and B g. So in B f say if x is appearing before y in the ordering than that should be satisfy in B g also, x must appear before y in B g also, so for all variables if it satisfy this particular requirement then we can say that these the variable orderings are compatible for these two BDD's. So one primary requirement for use of this apply algorithm means that both the BDD's must have compatible variable ordering.

So when we are going to use this particular apply operation we are going to give two BDD B f and B g and just say that these two are ROBDD reduced binary decision diagram, after application of this particular operator say we are going to get, say going to perform that plus B f and B g this particular apply algorithm will return what it will return? It will give me an ordered BDD which represent f plus g and the ordering of that variable is same with the ordering of B f and B g. So the resultant BDD is also an ordered BDD and ordering of the resultant BDD is the ordering of the two input BDD's but, after application of this particular apply operator whatever BDD we are getting it may not be a reduced one.

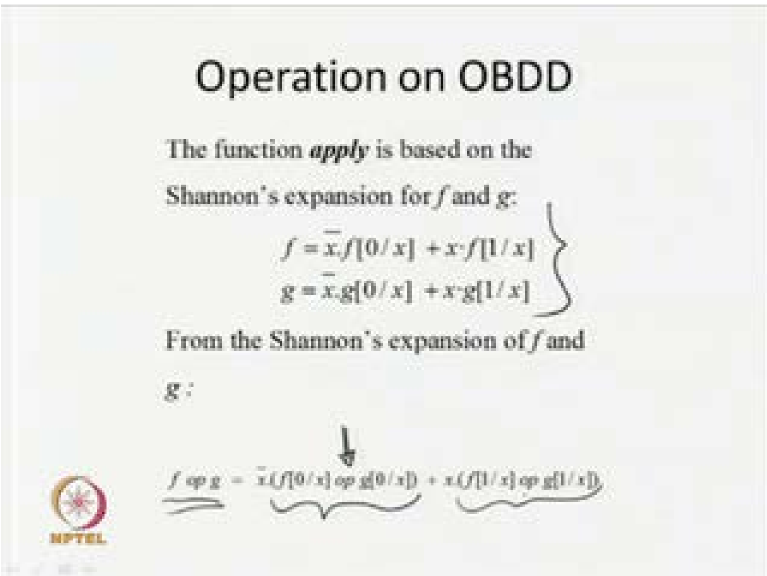
So after apply algorithm what we can do? We can use the reduced algorithm to get the reduced ROBDD reduced order binary decision diagram. So usually what happens? Now we are going to use apply algorithm apply so, some operators say plus B f and B g. We have does (( )) compatible variable ordering, the resultant of this particular operation will give me an ordered BDD. So in this particular case the ordering is same with input BDD. Whatever ordering we are having in input BDD it is going to give me the same ordering we are going to get an BDD which is having the same ordering.

(Refer Slide Time: 10:20)



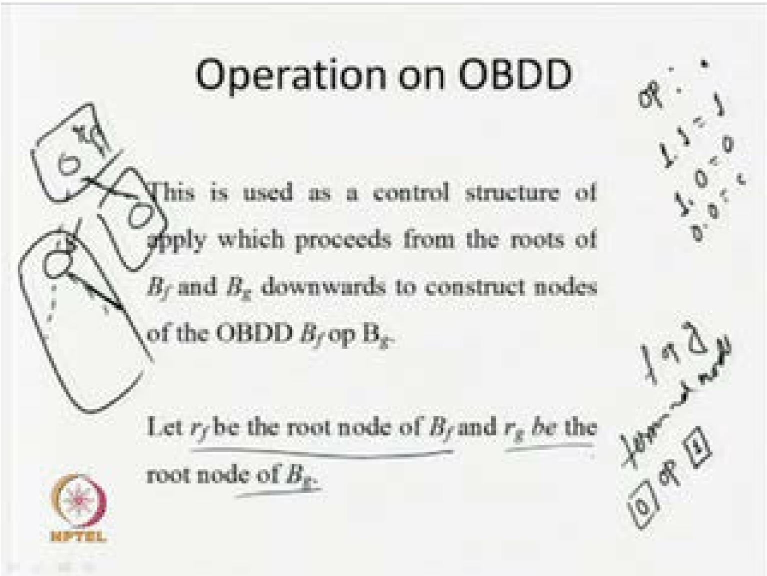
But the order BDD that we are getting may not be a reduced one after performing this apply operation. So two this particular resultant BDD's we are going to use the reduce algorithm. In last class I think we have discussed about this particular reduce algorithm. So after application of this particular reduce algorithm whatever BDD we are getting it will be your ROBDD reduced ordered binary decision diagram. Let me you say if we are having two function f and g we can represent these two function with the help of two ROBDD's and basic requirement is that the ordering of these two BDD's must be same; that means, there is suitable compatible variable ordering.

(Refer Slide Time: 12:25)



After application of this particular apply algorithm we are going to get a ordered BDD which represent this particular function f of g. That resultant BDD may not be a reduced one. So we use we are going to use the reduce algorithm to get a ROBDD of this particular output OBDD’s. Now how we are going to perform this particular apply operation. So this is basically again based on our Shannon expansion on this say. I have already mentioned that the construction of BDD depends on the Shannon expansion of a Boolean function. So Shannon expansion we know that it is representing with the help of this expansion f is a function. So f can be represented by x bar f you are going to evaluate f where x is replaced by 0 plus x dot f is replaced by 1 similarly, g x bar g.

(Refer Slide Time: 14:31)

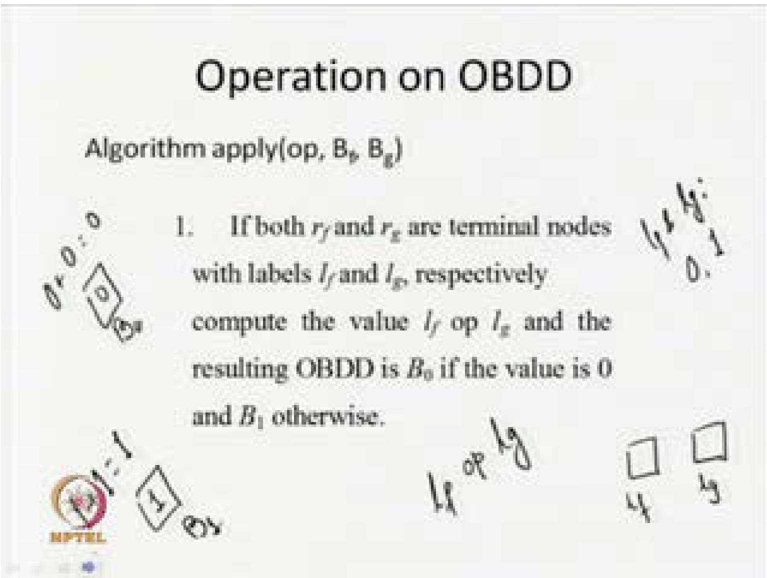


We are going to evaluate g by replacing x equal to 0 plus x and going to evaluate g by replacing x equal to 1. So these are the two Shannon expansion of our given function. Now if we are going to perform many operation f of g, you just see that we can rearrange this particular expansion and what we are going to do x bar, then this part we are going to perform this op operation, that operation on the evaluation of function f replacing x by 0 and evaluation of function g replacing of x by 0. So we can perform this particular operation on this sub function because already we have taken the decision of x equal to 0 similarly, for x equal to 1 I can go into look for this particular thing. So this operation you just see that while constructing our BDD we are going to apply this particular Shannon expansion recursively after one variable to the other variable.

So that same Shannon expansion will be used for performing a operation in this particular apply algorithm. So from root node we are going to perform and going down to the leaf nodes step by step so it is some sort of your top down approach, we are going to start with root nodes or given BDD's will travels it down in the problem fashion and eventually we are going to get the root nodes sorry terminal nodes or a leaf nodes and when we are coming to this particular terminal nodes then what will happen? In terminal nodes we are going to get the terminal values say in this particular case we are saying that f of g.

Now when we are coming down to terminal nodes then in case of terminal nodes we are having a value either 0 or 1. Now when I am coming to this particular terminal nodes then I can apply this particular operation because I know that if I am my operation is your say dot then I know that 1 dot 1 is equal to 1. I know for other combination 1 dot 0 is equal to 0 or may be 0 dot is equal to 0. So like that we are going to construct the resultant BDD and when we are going to get the terminal nodes then we can eventually apply the operation on those particular terminal nodes.

(Refer Slide Time: 16:19)

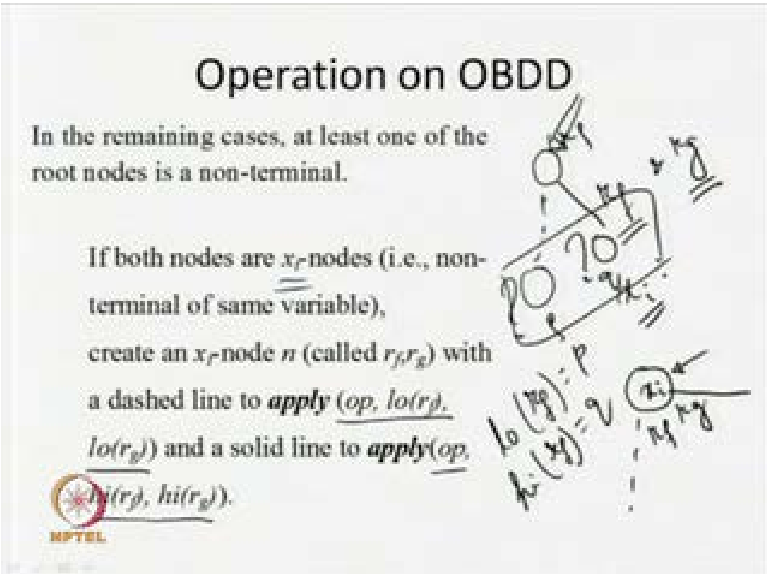


Now just see that we are going to do it recursively, let r f be the root node of our BDD B f and r g be the root node of BDD B g; that means, we can say that this is my root node r f and we are having the BDD's something like that. Now I can apply any labels say once I take the decision operate here then what will happen? I have to look for these particular two sub BDD's; that means, when I am coming to this particular sub BDD's then I will say that this is my root node r f. So that is why in general I am saying that let r f be the root node of B f and r g be the root node of your B g.

Now what your first step apply algorithm apply of B f and B g. If both r f and r g are terminal nodes with label it l f and l g. So now first condition we are saying that when we are coming to the terminal nodes, this is your two terminal nodes say this is I am saying that label this is your l f and this is your l g. So l f and l g these two can have values either 0 and 1 nothing else because these are the terminal nodes and these are the value. Then compute the l f of l g; that means, in this particular case when we are coming to this thing then we are going to perform the l f operation l g and the resultant BDD will be B 0 if the result is 0 or it is B 1 if the result is your 1; that means, I can say that 1 plus 1 is equal to 1. So in that particular case I am going to get the BDD one which is your B 1.

Similarly, if it is your 0 plus 0 then it is equal to 0, then I am going to get the resultant value as 0 which is the BDD B 0. So when we both the nodes are terminal nodes, then we are having the valuation of this particular function. So we can apply the operation on this terminal nodes and the resultant BDD's will be either B 0 or B 1. So this is the first case we are going to consider if both are terminal nodes.

(Refer Slide Time: 17:52)

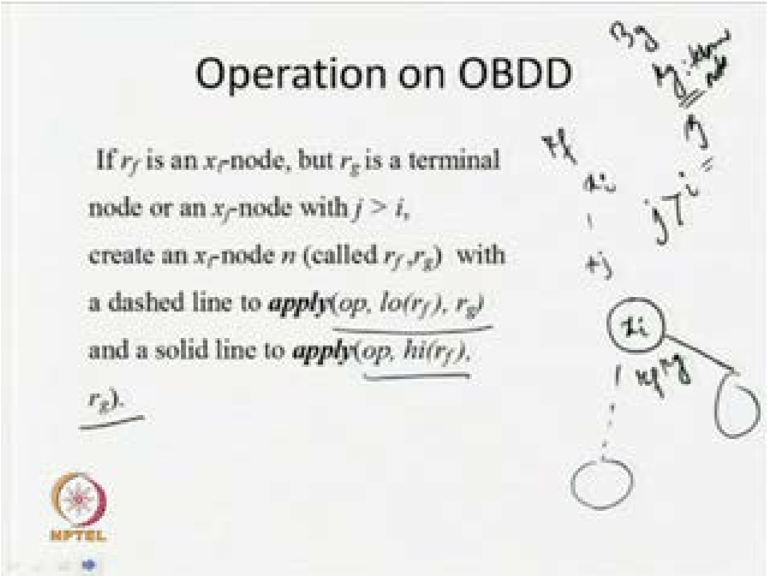


Now if it is non-terminal nodes, then what will happen? At least one of the nodes will be your non-terminal nodes. So when I am going to talk about say r f and r g if both are non-terminal nodes then at least one of them will be your non-terminal nodes. Now first case we are going to consider that if both nodes are x i nodes. So both are x i nodes means both are your both nodes are your non-terminal nodes and they are labeled with these variable x i.

Now we are going to create a node x i; that means, since now both the function on depends on this particular variable x i, we are going to create a node x i and we label this particular node or we are going to call this particular node r f and r g because in my given BDD's one node is r f and second node is r g. So we are going to label a node called r f and r g and since both this particular nodes are labeled by x i we are going to construct a x i node and after that when I am having this particular x i node these x i node has two is this, one is this and one is your sorry this line will said that x i equal to 0. So with then we say that x i equal to 1.

So in that particular case we are going to apply this particular operation for your dash line op l f, l o r f and l o r g apply op f hi of r f and hi of r g. So basically what will happen? If this is my r f node, then I know that I am having two function already we have defined. One is your lo r f and second one is your hi r f. What it will return basically it will return a nodes that is pointed by dashed line in case of l o say if it is say that this is my say p q; that means, l o r f will return p and h i r f will return q; that means, now when I am going to perform the operation on this particular r f and r g, next time what will happen by the looking into their definition we have to take decision on these two nodes p and q that is why we are applying this things op l o r f, l o r g and in solid line apply o f op hi r f hi r g. So what about we are having so this is what is this dashed line and what is the solid line, what is the decision we have to look for those particular sub BDD's. So that is why we are saying that we are going to apply the operation on lo of r f and lo of r g and in case of hi o r f and hi of r g.

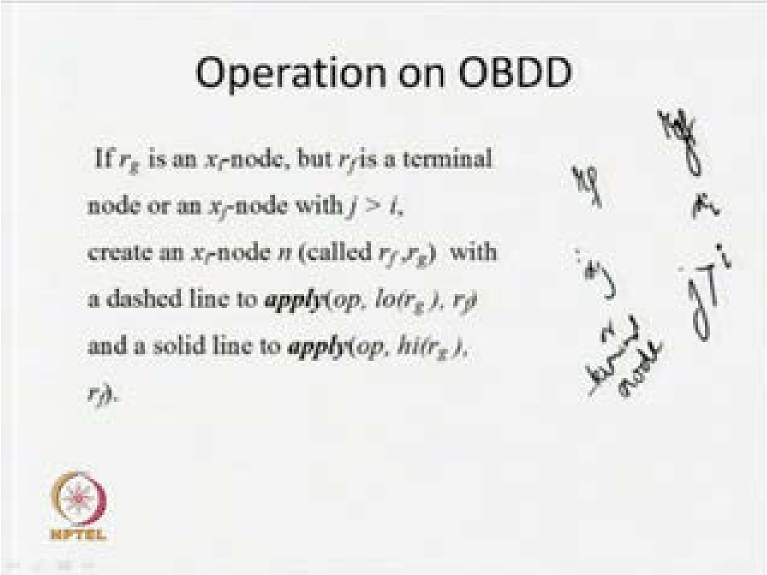
(Refer Slide Time: 20:52)



So if both are labeled with same variable x I, then we are going to follow this particular rule. Now on the other hand in the second case what will happen? We may have one non-terminal nodes or one terminal nodes also, so in this particular case say r f is having an x i node and r g is having an x j node, where say j is greater that i. So in while we are traversing it from top to bottom when we come to this particular scenario one is x i and one is x j and j is greater than i basically, it says that this particular BDD B g will be independent of this particular variable x i on that computation part, because we are traversing it from top to bottom and uniformly we are going down in one part we are getting x i but, second part we are not getting x i we are getting x j.

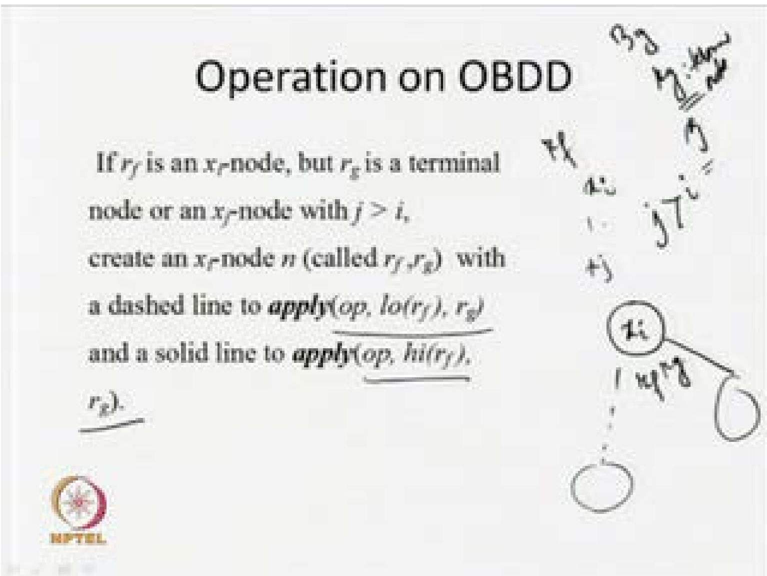
So what does it means? That means, your B g is independent of x i in that particular evaluation path. So for that what will happen now the resultant BDD now may be dependent on x i because one of the function is given on x i. So we are going to create an x i node at that particular point and again similarly, it will be labeled by r f and r g because these are my root nodes and similarly, we are going to construct this particular two node. So what will be the position over here? You just see that we are going to now call apply this particular operation op lo of r f and r g because r g is independent of your x i. So; that means, again I have to look for down the line somewhere I am going get x j in your r f at that time I am going to see what will be the situation.

(Refer Slide Time: 23:45)

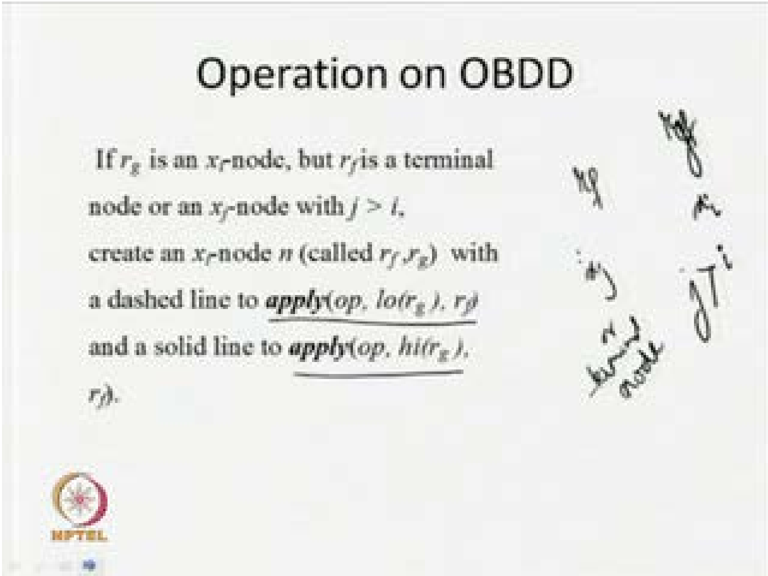


So that is why we are going to apply l o r f and r g because r g is your independent of i and in case of your solid line we are going to apply op hi of r f and r g. So this is the way I am going to construct my resultant BDD. So here r f is an x i node but, r g is a terminal node or it is an x j node. Here I am talking about it is an x j node, j is greater than i or it may happen that r g may be your terminal node also done the (( )) will be seven will see in that case what we are going to do. So this is the second scenario and third scenario is when I am seeing that if r g is the x i node but, r f is a terminal node or x j node, where j is greater than 1.

(Refer Slide Time: 23:59)



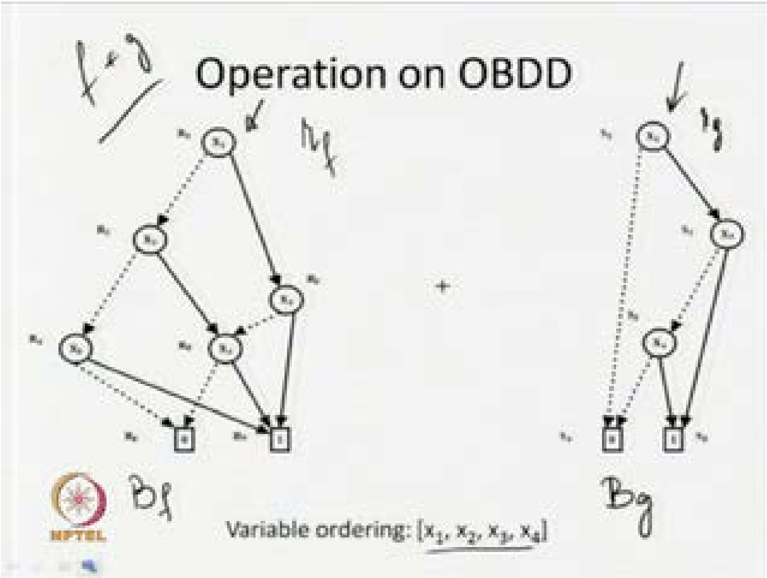
(Refer Slide Time: 24:02)



So this is basically if you see this is symmetric to this particular condition only but, now situation is different what we are saying that r f it is a x j node or terminal node and r f sorry r g is an x i node and here it says that j is greater than i. So this is similar situation like the previous one but, in that particular case we are saying that r f is an x i node but, now we are saying that r g is an x i node. So this is the similar situation so, here that the similar way we are going to treat it so in dashed line we are going to say that op l o r g and r f and apply op hi of r g and r f.

So see that this a symmetric so, that is why we are taking in the previous case we are taking l o r f and r g. Now we are going to take l o r g and r f. So these are the rules that we are having over here, now just see that we are going to take two ROBDD’s, we are going to traverse it from top and it will be a top down approach, the both are terminal nodes then we are going to apply the operation over there because we know the values either it will be 1 of 1 or 0 of 0 or 1 of 0 or 0 of 0 we may have four condition. In all other cases one of the node will be your non-terminal nodes.

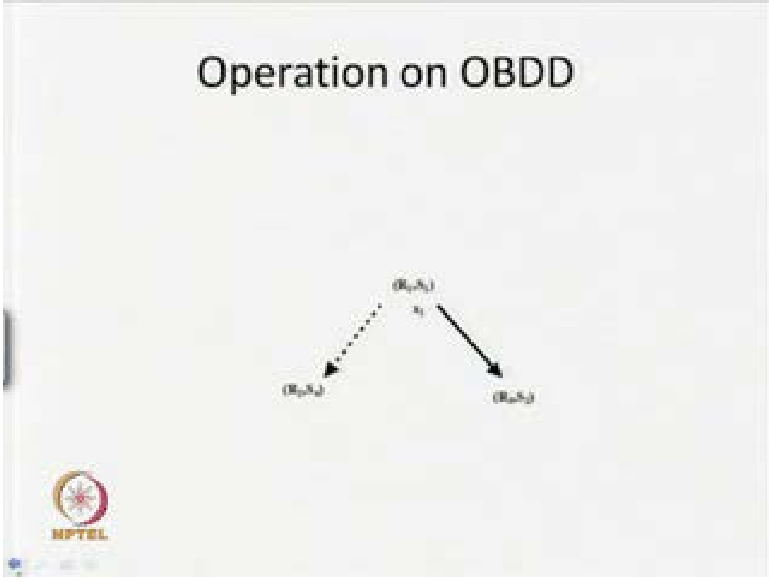
(Refer Slide Time: 25:59)



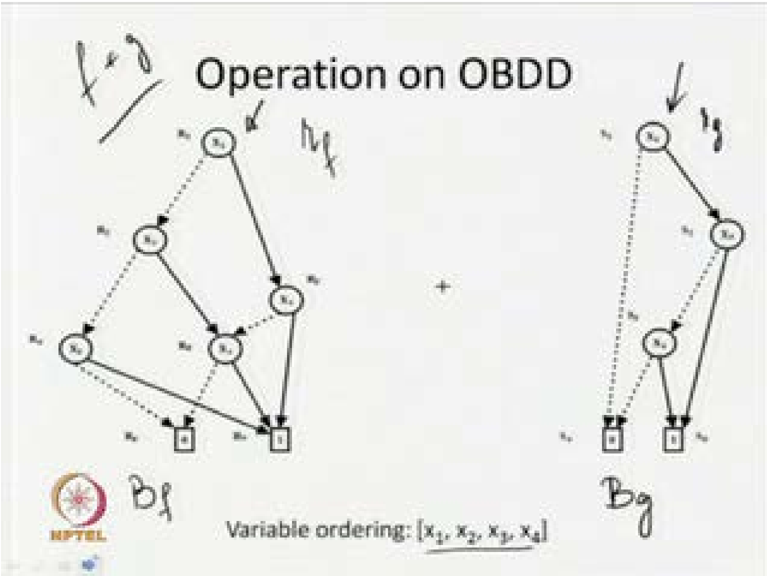
One situation is opening in both the non-terminal nodes are having the same level. We are how we have seen what we are going to do. In the second case one is a non-terminal nodes, second may be a terminal nodes or another non-terminal node, another non-terminal nodes but, there leveled is not same. So we have seen how it is going to or how we are going to handle this issue. Now this apply algorithm we are going to now see with a help of an example, just see that this is one BDD set of function your f say can say that this is your B f and say this is the BDD for function g, we can say that this is your B g. Now both are having the same variable x 1 to x 4 and the variable ordering is your x 1, x 2, x 3, x 4.

By looking into this construct of this two BDD's we can find that we can see that they are having the compatible variable ordering they are following the (( )) ordering. Now I want to perform f plus g. So this is that is why I am saying that this BDD plus this BDD is going to give me f plus g. Now what we are going to do? Now will doing start from the top so I will considered these two are my root nodes say as my nomenclature this is your r f and this is your r g. Now we are having that name nomenclature over here r 1, r 2, r 3, r 4, r 5, r 6, r 7, s 1, s 2, s 3, s 4, s 5.

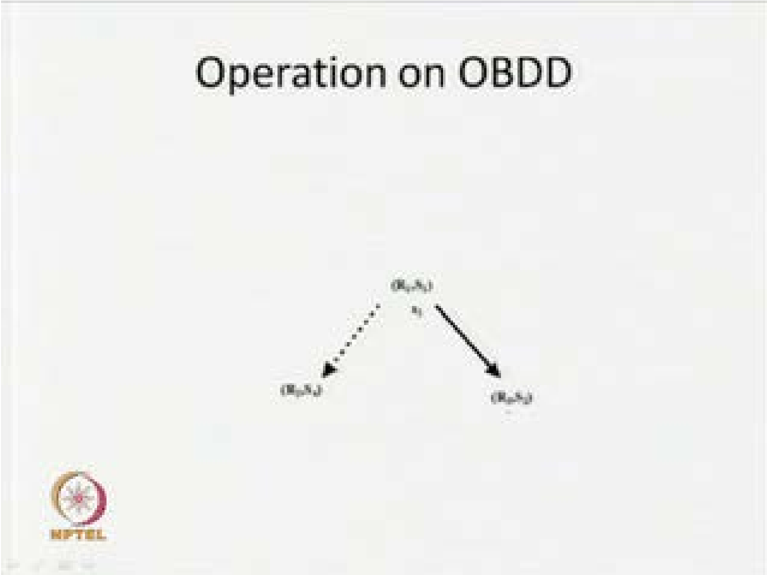
(Refer Slide Time: 27:14)



(Refer Slide Time: 27:18)

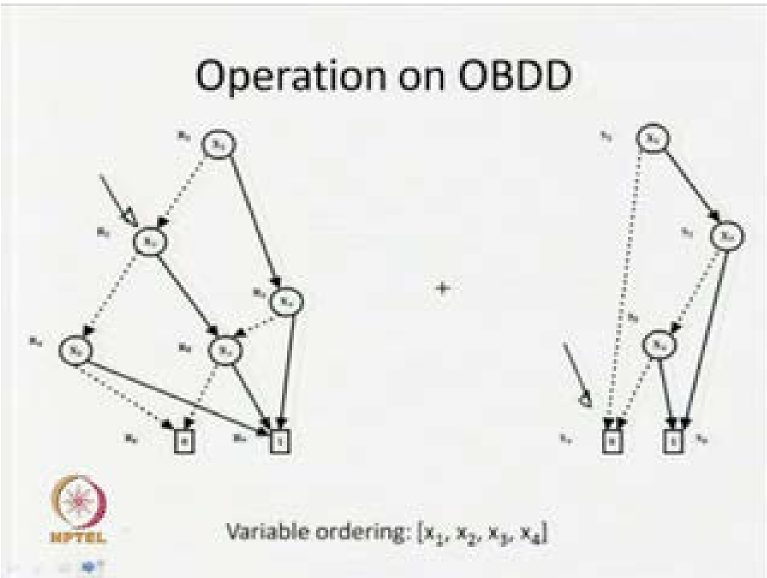


(Refer Slide Time: 27:42)

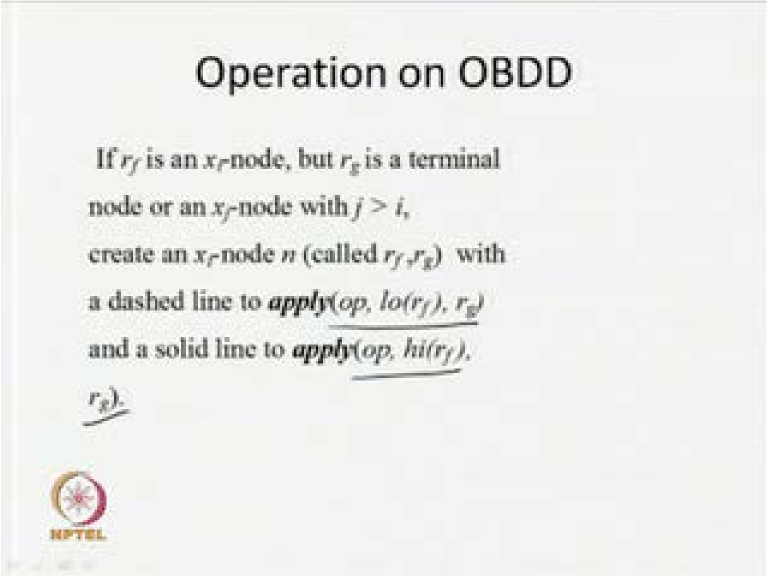


Now see what we are going to construct first we are going to take this particular root node. Now I am going to take r 1 s 1 and you just see that both are having the same variable label x 1 and x 2. So we are going to create an x 1 node. So lo of your r 1 is going to r o, r 2 and lo of x 1 is going to s 4 similarly, hi of r 1 is going to x 2 and hi of s 1 is going to s 2. So that is why we are constructing this particular structure r 1 s 1 it is labeled with your x 1 now it is r 2 x 4 and r 3 s 1.

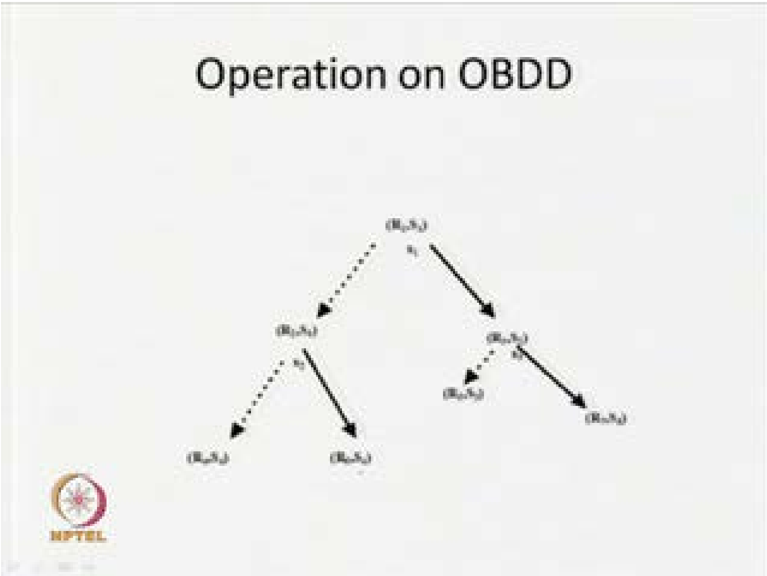
(Refer Slide Time: 27:56)



(Refer Slide Time: 28:44)

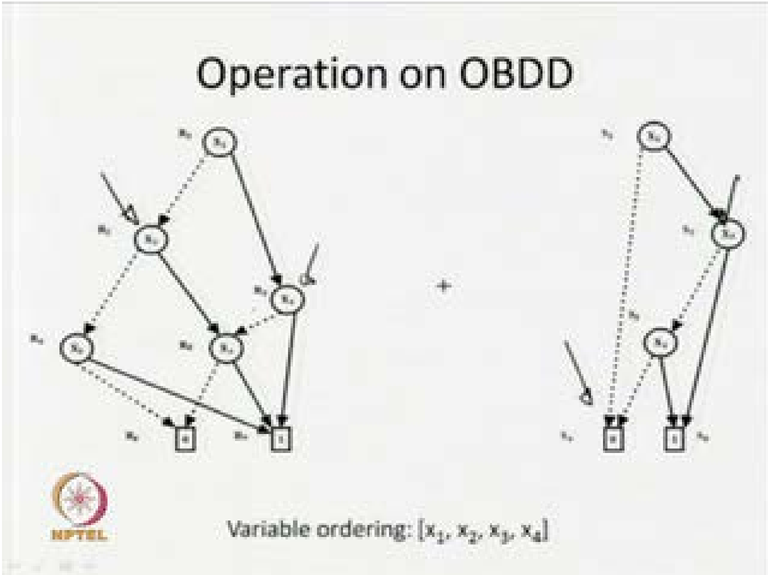


(Refer Slide Time: 28:54)

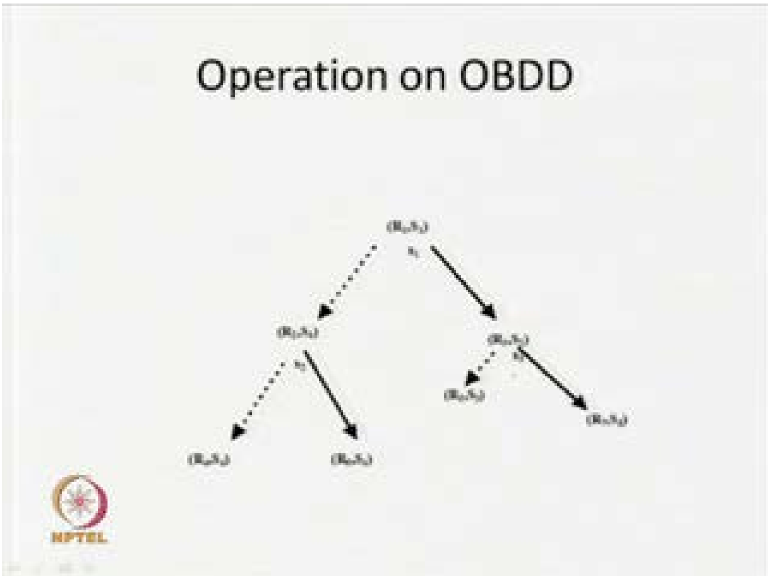


So this is the scenario that we are having. So you are coming to this particular thing r 3 x 1. Now after that what will happen, now; that means, I am going to look for if I am going to follow this particular dashed line, this will be my root node and this one will be my another root node. Now I am having this particular partial structure now in this particular case now r 2 is a root node and your s 4 is a root node. So one is your non-terminal node second one is a terminal node. So in this particular case what we are going to do? Will for dashed line will follow this lower this one and it will remain as it is for your this things hi 1 if will follow the structure and it will remain as it is. So in this particular case if you look into it, we are going to apply this particular rule already we have discussed that lo of r f r g hi of r f r g. So we are going to get this structure form r to s 4 we are getting r 4 s 4 r 5 s 4.

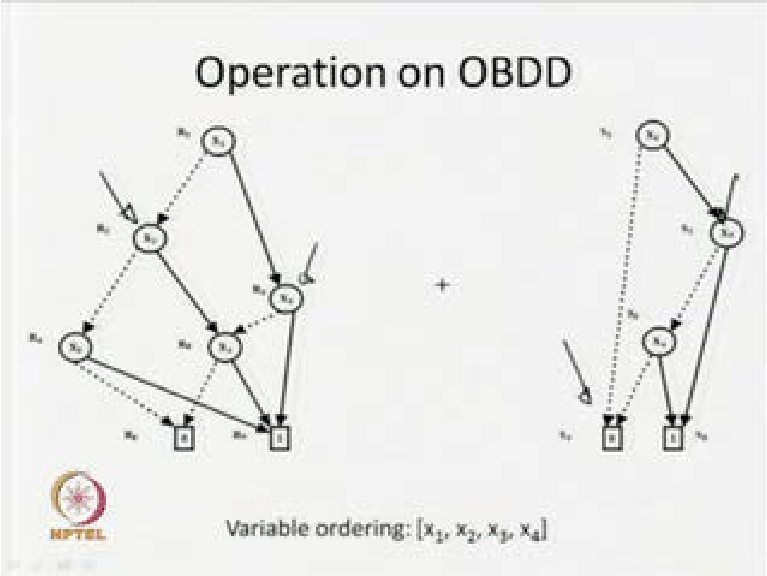
(Refer Slide Time: 29:04)



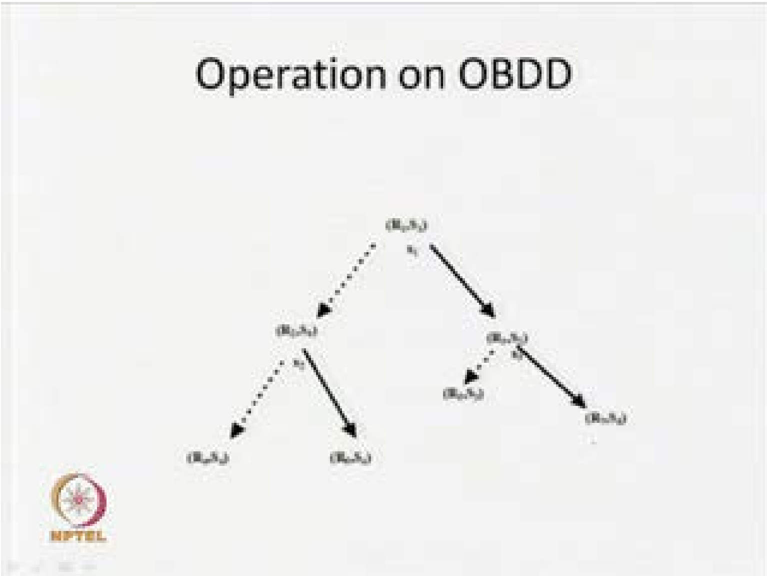
(Refer Slide Time: 29:20)



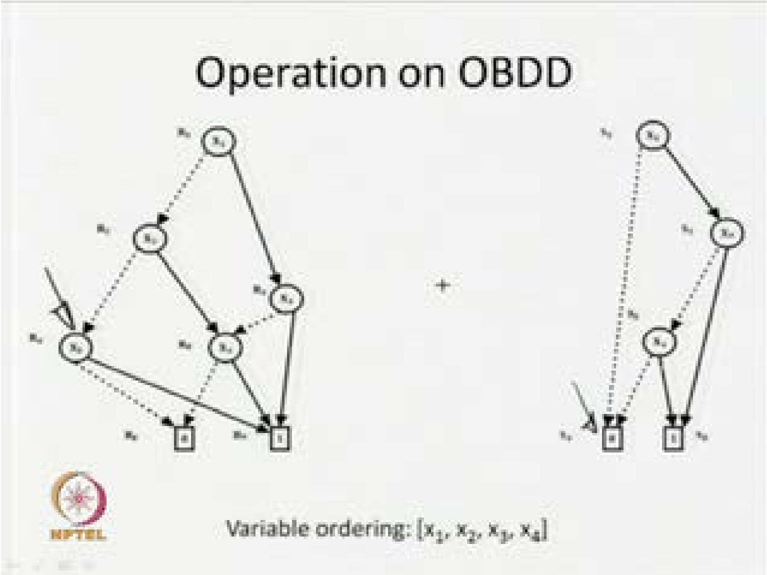
(Refer Slide Time: 29:32)



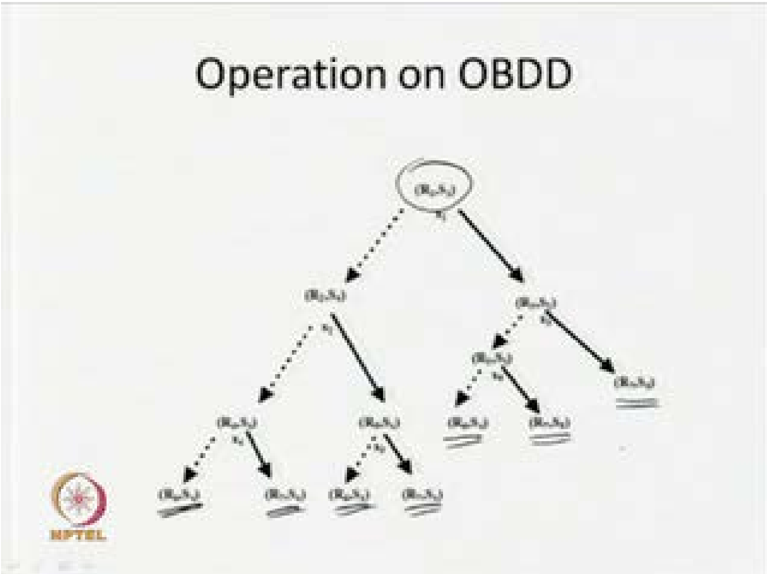
(Refer Slide Time: 29:45)



(Refer Slide Time: 30:00)

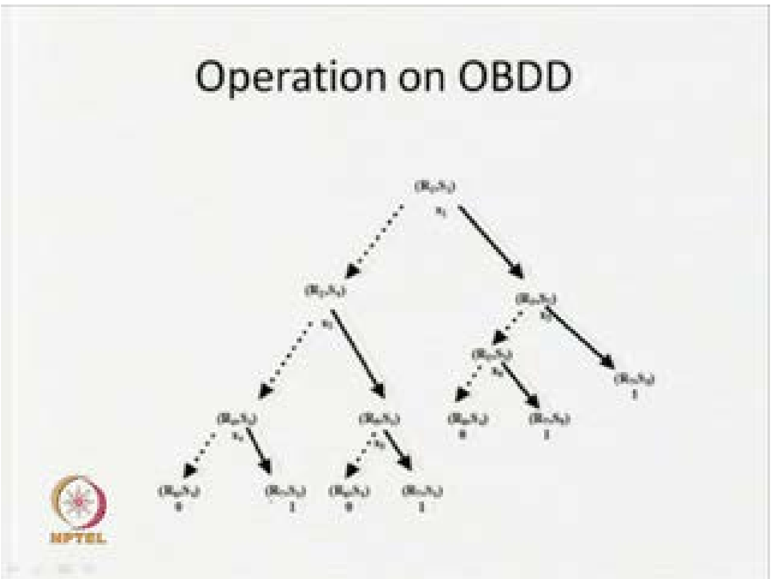


(Refer Slide Time: 30:14)

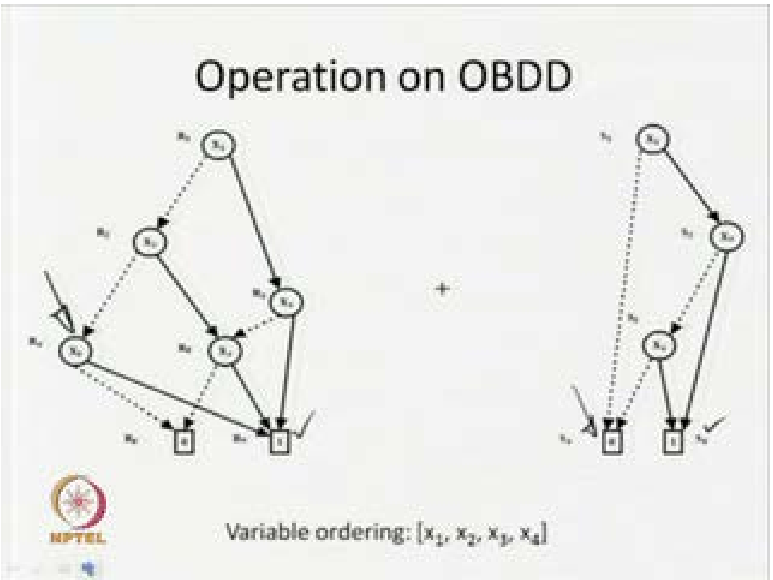


Similarly, if we look into the other side for one so this is my root node and this is my root node. So we are going to construct an s 2 r 3. So this is your s 2 r 3 and these are the dashed line is coming to r 5. So both are non-terminal nodes and we are going to get an x 3 node over here because you just see that this is your x 2 and this is your x 3 and this is also x 3. So I am going to create an x 3 node over here and we are getting this particular two dashed line and solid line. Now r 4 s 4 and r 3 s 4 and r 5 s 3 and r 2 s 6. So in this particular case now by considering these as a root node this one will be one root node and this one will be the another root node. So we are going to construct this particular dashed line and solid line similarly, here we are getting what may be that this thing so we are going to have this scenario. So now we are coming up to this particular label.

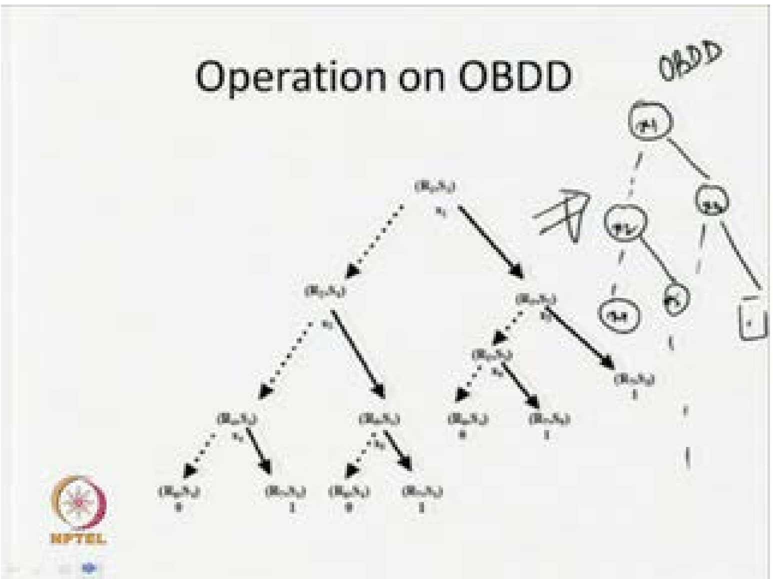
(Refer Slide Time: 30:21)



(Refer Slide Time: 31:00)



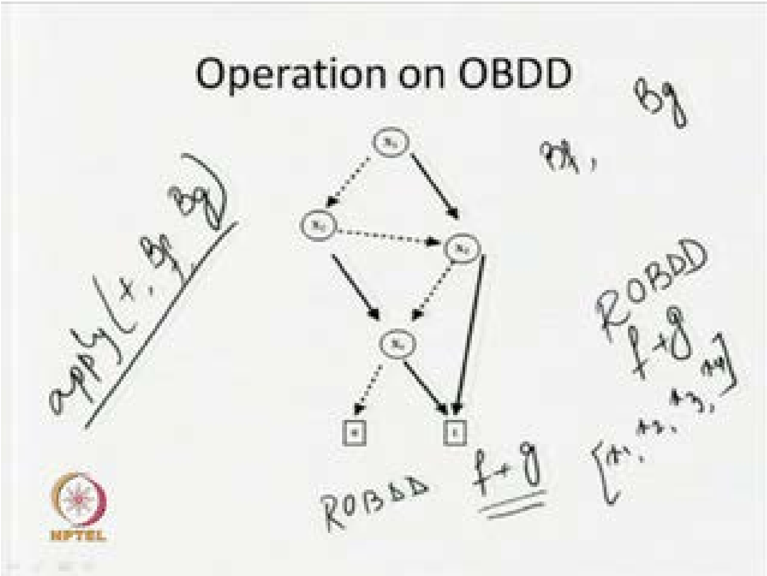
(Refer Slide Time: 31:09)



Now when we are construct you say from top r 1 s 1 we are starting it and coming down this. Now you just see that all those particular nodes whatever we are getting over here, all these things are non-terminal nodes. Now I am going to apply the binary operator on this particular terminal nodes. Now while I am applying this particular terminal nodes you just see that 0 plus 0 will be 0 something like that 1 plus 0 will be 1. So basically, we just see that this is your r 4 s 4; r 7 s 4; r 7 s 4; r 7 is 1 s 4 is 1. So 1 plus 1 it is going to give me 1. So in this particular case I am going to get 1.

So eventually, I am getting this particular structure. Now these structure now what will happen? Now I can construct the BDD from this particular structure I am getting say this is your x 1, this is the dashed line, this is solid line. In dashed line I am getting x 2, in solid line I am getting x 3. Now in this particular dashed line I am getting your x 4 and in this solid line I am getting this particular x 5 like that I am getting this particular dashed line and this solid line. So this solid line will come to 1 so like that I can construct the BDD. So I am getting a BDD and this is basically OBDD, I am getting this OBDD from this particular structure.

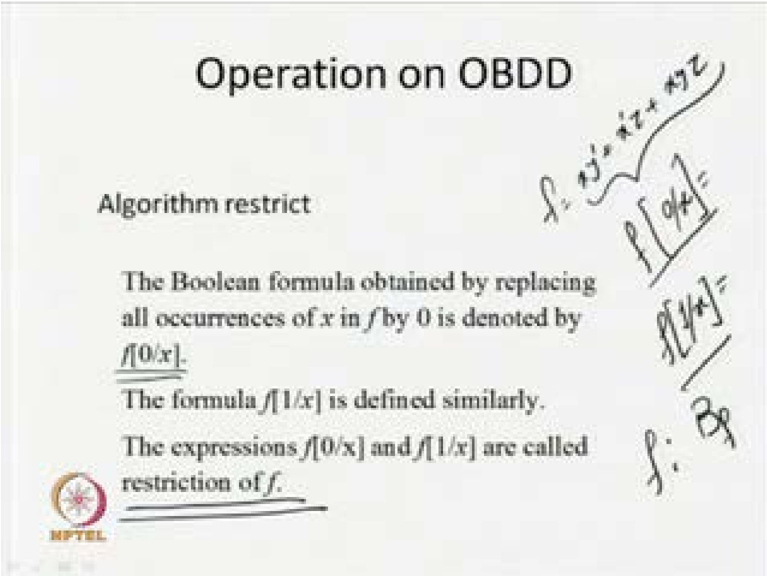
(Refer Slide Time: 32:19)



Now after getting this particular ordered binary decision diagram we can see that it may not be a reduced one. So after that we are going to apply this particular reduce algorithm and after applying of this particular reduce algorithm we are getting this particular structure and this is the ROBDD that we are getting over here. So eventually we are getting ROBDD for f plus g and ordering of the variable which same with ordering of your B f and B g, this is basically x 1, x 2, x 3, x 4, where B f is the order BDD, ROBDD of f and B g is the ROBDD of g. So what happen we have use this particular apply algorithm plus B f B g and we are getting this particular ROBDD's, BDD for f plus g.

So like that if we are having two ROBDD's we can always use some binary operation on this two structure and we can eventually get the ROBDD's to represent this particular Boolean operation. So this is one important algorithm and will be using this algorithm while going to look for the uses of ROBDD's.

(Refer Slide Time: 33:34)



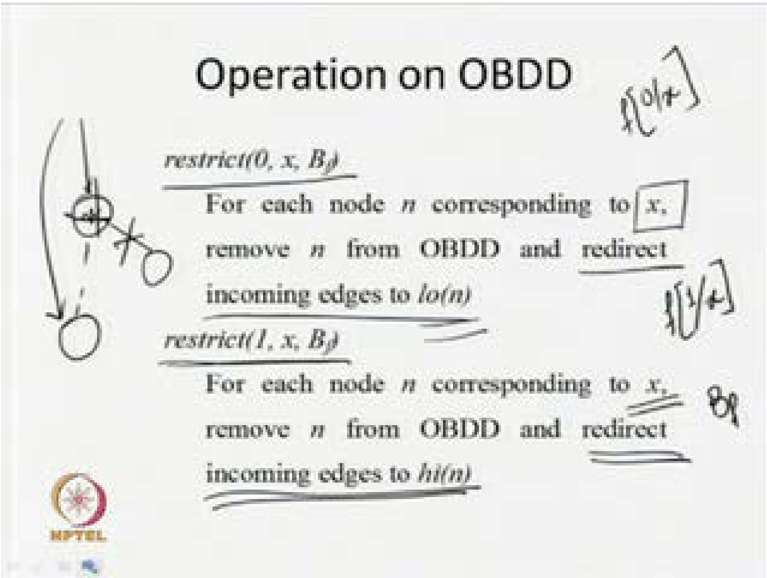
Now another issues we are going to see that we have seen the apply algorithm. Now we are going to look for another issues we say that a Boolean formula obtain by replacing all occurrence of x by f x or in f by 0 is denoted by f x replaced by 0; that means, if I am having any Boolean function say f is equal to say x y bar plus x dash z plus x, y, z something like that I am having a Boolean function. So basically this Boolean function depends on the valuation of x, y and z. x may be either 0 or 1 and similarly, y and z but, we can say that what will be the valuation of these particular function when I replace all x by 0. So this is basically we are denoting this particular things all x replaced by 0 what is the functional value.

Similarly, we can say that what is the functional value? If I replace all x by 1; that means, we are putting a restriction on one variable I am saying that x will be always 0 only or will say that x will be always 1 only, you just look for the valuation of these particular function depending on the other variable. So this is the way we can look into it and we can say that this is basically known as our restriction of f. We are going to say that we are going to restrict the valuation of function with respect to some variables. I am having a variable of function with depends on separate variables but, I am saying that restrict that particular function for one particular variable to be 0, or restrict a function for a particular variable to be 1 only.

So we are going to say this is the restriction of this particular function f. So we are going to have restriction of this function f, f x is replaced by 0 or x is replaced by 0. Now if we are having BDD representation of f say B f whether we can directly restrict it or not, we can apply this particular restriction on it or not.

So here I can I know that if I am going to restrict it for x equal to 0 then we find something like that it will be a z on may be because x states will be zero other two terms will be zero. So similarly, if I am having the BDD representation of this Boolean function f but, can I apply this particular restriction on those particular Boolean function or not.

(Refer Slide Time: 36:05)



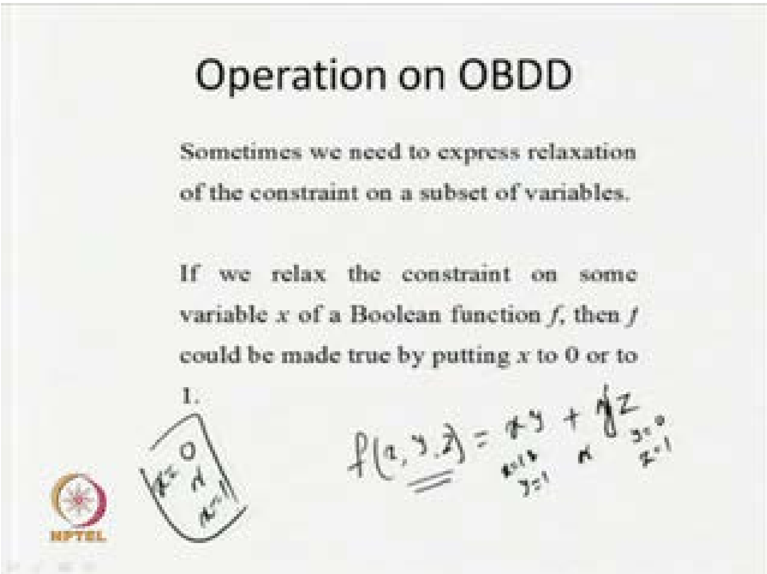
So it is very simple, so what happens? We are going to say that this is restrict 0 x B and this is basically nothing but, f replacing all variable of all x to be 0 and restrict 1 x B f this is nothing but, restriction of this Boolean function f replacing occurrence of all x by 1.

So if I am having the BDD representation of this particular Boolean function as your B f it may be reduce or that BDD also, then with the help of this restrict operation or restrict algorithm what we can do? We can construct the x f replaced by 0. So what we have going to do in this particular case we are going to look all nodes corresponding to this particular label x.

So we are going to look for all corresponding nodes that are label x and will delete all such n and we will redirect the incoming edges to lo of n because why we are redirecting the incoming edges to lo of n? Because we are restricting x to be 0 so basically, if I am having x node over here and say this are your low and this is your high so, you may have several. So when I am going to restrict x to be 0; that means, x would be always 0. So this evaluation we are not going to consider so in this particular case we are going to remove this particular node and whatever incoming edges we are having all those to in the redirected to this particular lo node.

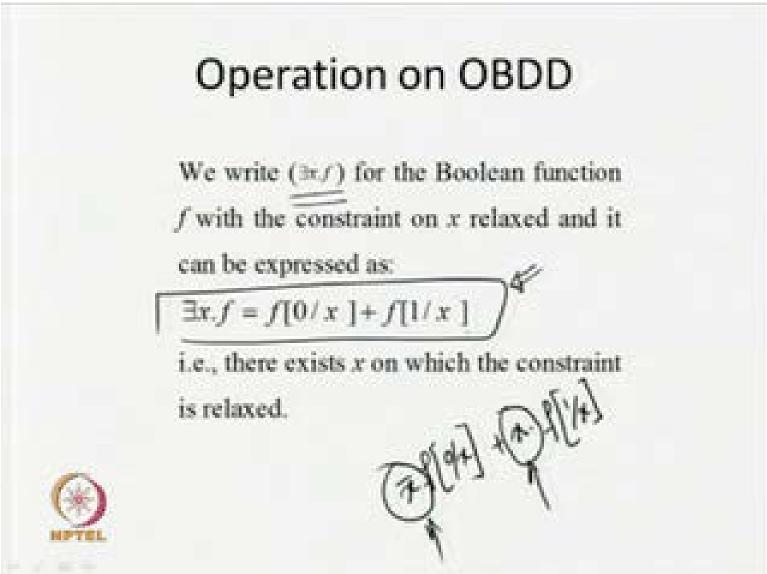
Similarly, we are going to use this particular restrict 1 x B f. So the pre pane is similar now we are going to remove all those particular node label by x and we are going to redirect the incoming edges to high of n. So this is the way we can construct ROBDD's for our restrict operation also, here restricting the function where x will be 0 and restrict the function x will be equal to 1. So we can if we are having a BDD representation of a Boolean function we can construct the restrict o 0 x B f by moving the all x node from this particular BDD and redirecting the incoming edges to the lo of x.

(Refer Slide Time: 38:37)



Similarly for restrict 1 x B f, so we are talking about some function say if I am having f x, y, z so, something like that x y plus y z. So in this or sub y bar z. So in this particular case what will happen these are function depends on these three variable. So we are saying that we are sometime we restrict that particular function on a particular variable that variable will be either equal to 0 or that variable will be equal to 0, some time we want to relates the constant some variable. What is this particular function? Basically we are saying that functional value will be 1 provide that x is equal to 1 and y is equal to 1 or y is equal to 0 and z is equal to 1. So the function value will be 0 provide that it is satisfy this particular constant. Sometimes we want to relates the constant on some variable say you just look into particular function or look in to the function that derive from this particular Boolean function where it is independent of one particular variable; that means, we are going to relates the constant of this particular function on a particular variable.

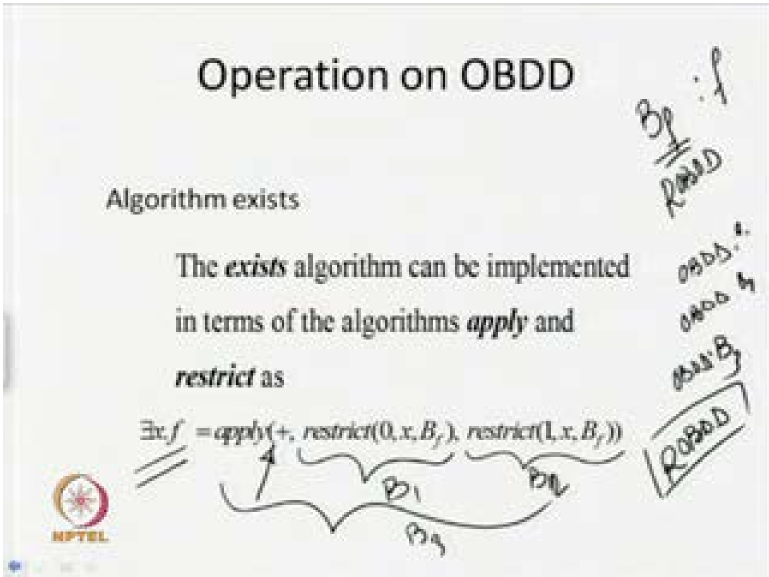
(Refer Slide Time: 40:20)



So you can relates the constant of some variable of x for this particular Boolean function; that means, the functional value will give me the evaluation one either x equal to 0 or x equal to 1 whatever may be the value of x, that functional value that like function is going to give me valuation one; that means, the valuation will depends on other variables. So in that particular case we are going to say that we are relaxing the function on a particular variable. Now generally we write these things with the symbol there exists f when we say that we are relaxing this particular function on this particular variable x; that means, we said that exist a variable x where the constant the x relaxed. Now this is expressed as with this expansion there exist f is say that we are going to relax the constraint on x for this particular function f, this is f either x will be replaced by 0 or x will be replaced by 1; that means, x can whatever may be the value of x.

Now we are going to look for the functional value of this given function. So it is very simple what is the Shannon expansion basically we have going to say that if x equal to 0 then we are going to say that evaluation of f with respect to 0 or x equal to 1, evaluation of this particular function by x equal to 1. Now I want to relax the function either x equal to 0 or x equal to 1; that means, we are not considering the value of this particular x over here. So we are dropping this two x bar and x so we are saying that f of x equal to 0 plus f x equal to 1 so, whatever value whatever maybe the value of x now going to look for a functional value of the given function. So this is relaxing on this particular variable x. So we are saying that there exists an x on which the constraint is relaxed we evaluation this particular function with the help of this expression sometime we relax the function on some variable.

(Refer Slide Time: 42:12)

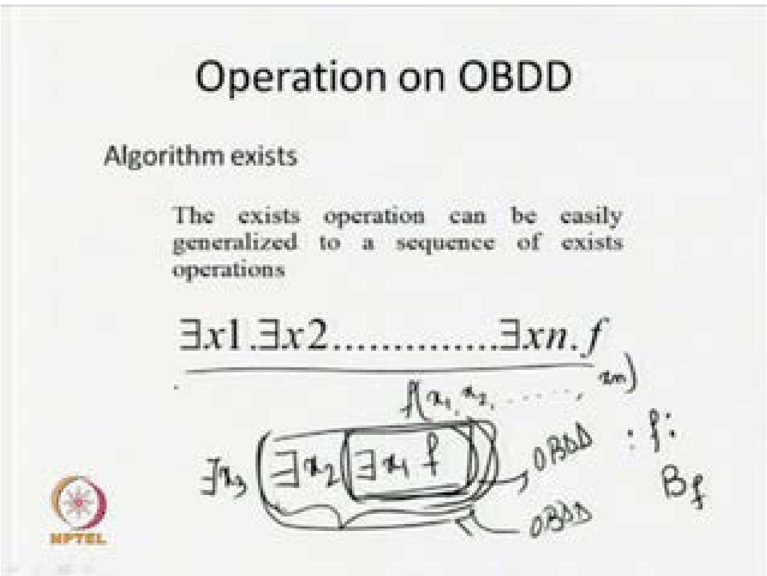


So relax the constraint on some variable so this is the way we are going to look. So with this expansion we can evaluate this particular relaxation. Now if we are having the BDD representation of this particular Boolean function so, where that can we use this particular relaxation on this particular BDD’s or not. So if we are having the BDD representation B f of a given function f then how to find the relaxation on a given variable x, already we have seen this restrict and operation restrict operation and restrict algorithm, with the help of restrict algorithm we are restricting the function to a particular valuation either x is equal to 0 or x is equal to 1 but, in case on exist or relaxing on that particular constraint it may be either 0 or it may be either 1. So that is why we can evaluate this particular there exist x f with the help of this particular expression.

Say using this particular relax operation, restrict operation, restrict x to be equal to 0 and restrict x to be equal to 1. So in case of reaction we are going to say that x may be either zero or 1 so that is why you are using this particular or operation plus operation. So we are using apply plus restrict 0 x B f and restrict 1 x B f. Now you just see that we are using this particular restrict x to be 1, restrict x to be 0. So if I am having say ROBDD of B f as your b g so after restricting I am going to get one OBDD's say and second restricts operation will go to give me another OBDD but, the variable ordering to this two resultant of OBDD's will be same with this particular B f. Now we are going to apply this particular operation plus in the apply algorithm.

So the resultant will be another OBDD. So I say that this is your B 1 and say this is B 2 so I am getting OBDD B 1 and B 2 after performing this apply operations say I am going to get another BDD say B 3. So this is the b 3 now it may not be reduce one. So to the resultant BDD I am going to use the reduce algorithm. So eventually I am going to get the ROBDD. So the order of the variable of this particular BDD's same with the order of this particular BDD B f. So this is the way that we can find out the restriction on some variable.

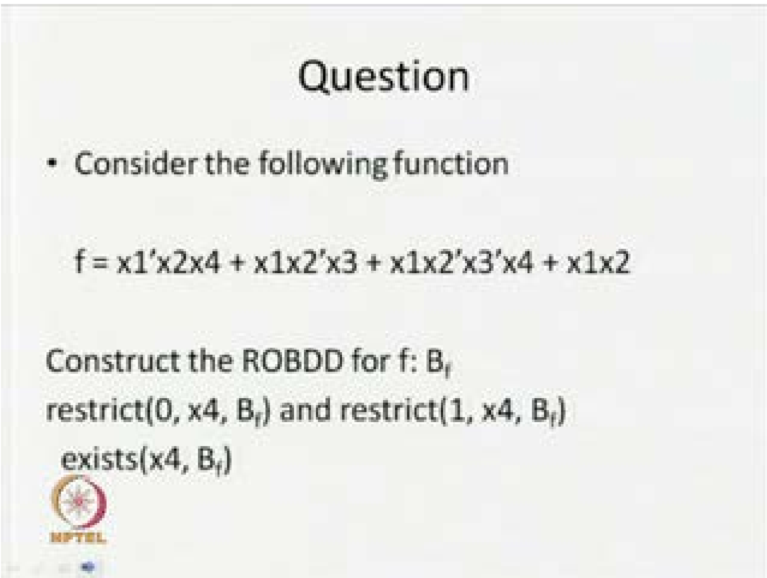
(Refer Slide Time: 44:58)



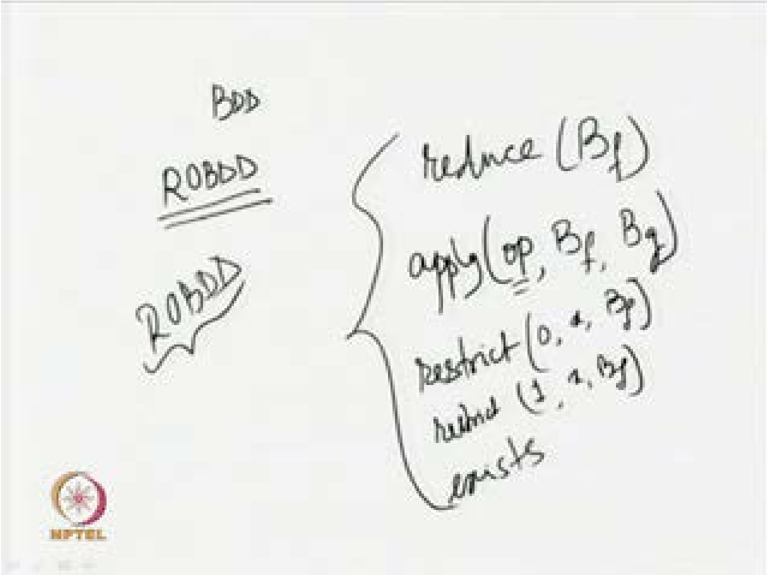
So this way we can find out the restriction of this particular given function f now this particular exist operation can be nested also or it can be given as a sequence also, basically if say that if I am having an function f on some variable x 1, x 2 something like that x n. So what will happen say if I am having there exist x 1 f; that means, we are relaxing this particular function on variable x 1. So we have going to get one function f which is now independent of x. Now we getting one function what will happen to this particular function I can I can put a relaxation on the second variable x 2; that means, whatever resultant function I am getting I am going to now put another relaxation on this x 2. So again I am going to get the Boolean function again I can put a relaxation on this particular Boolean function say on that thought variable exist.

Like that we can keep on relaxing it on some function. So we can use as a sequence so we are going use this particular expression that it is relaxing as on say x n and x n minus 1 like that. So what will happen if f is going to represented by the order BDD B f. So this particular result these exist x 1 f 1 will give me another OBDD. So to this particular OBDD I am going to relax on x 2 I will apply this same operation over here and which is give me another OBDD. So like that I can keep on applying this particular operation on this particular resultant OBDD.

(Refer Slide Time: 46:38)



(Refer Slide Time: 46:41)

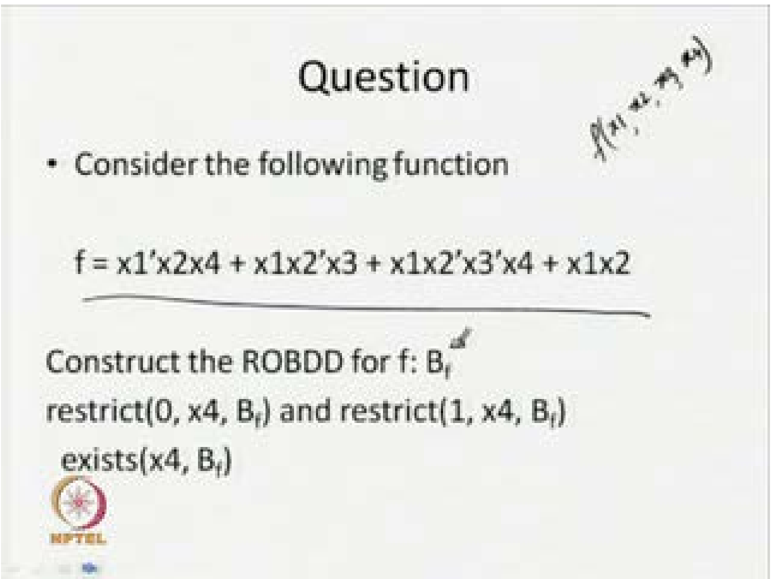


So we have seen some algorithm till now. So first what happens we have seen that BDD binary decision diagram then eventually we have come to the r o b d d ROBDD that reduced ordered binary decision diagram. Now what will happen we are going to get a unique representation of this particular r o b d d ROBDD for a given function. So now to what does particular r o b d d ROBDD representation we have seen some algorithm. So what are the algorithms? First algorithm we have talked about reduce, with the help of this particular reduce algorithm we can get a reduction of a given variable say B f.

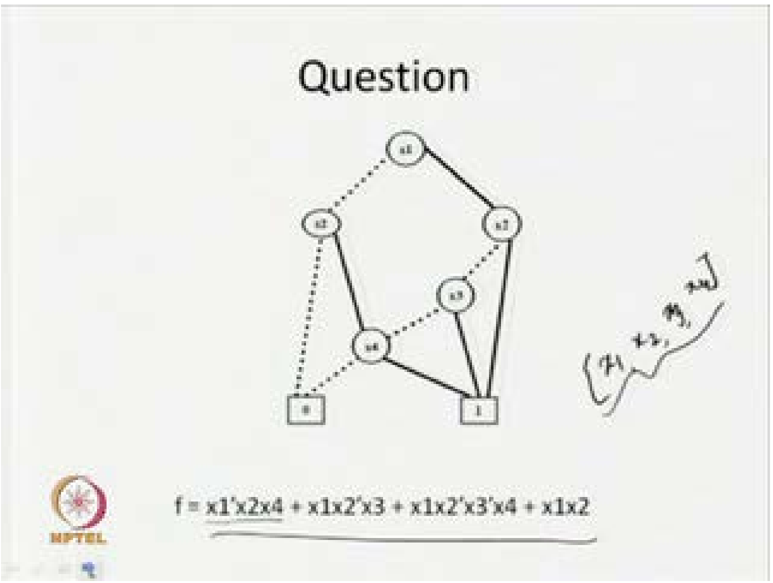
So it will going to return this things then we have seen another algorithm which is your apply, with this particular apply we can use any binary operation on two 2 BDD's B f and B g and we have seen that this particular apply operation is going to give us an BDD which is going to talk about the f of g and one main constraint on this particular apply algorithm is that B f and B g suit have compatible variable ordering and the whatever the resultant BDD we are getting that BDD.

Will have also have the similar ordering with B f and B g but, the output BDD may not be a reduce one. So we can use after that we can apply this particular reduce algorithm to get the ROBDD, we have seen another this things algorithm which is your restrict we can restrict the function on some variable to be 0 or 1. So I can say that restrict 0 B x B f or we can say that restrict 0, 1 x B f. So this is the restrict operation that we are getting over here and we have seen another algorithm which is exist basically we are going to relax the function on a given variable; that means, we are relaxing the constraint on a given variable or it may be series of variable. So these are the algorithm that we have seen which can be directly used on your ROBDD and resultant we are going to get another ROBDD as a result where the ordering of the output BDD will be same with the input BDD.

(Refer Slide Time: 49:01)



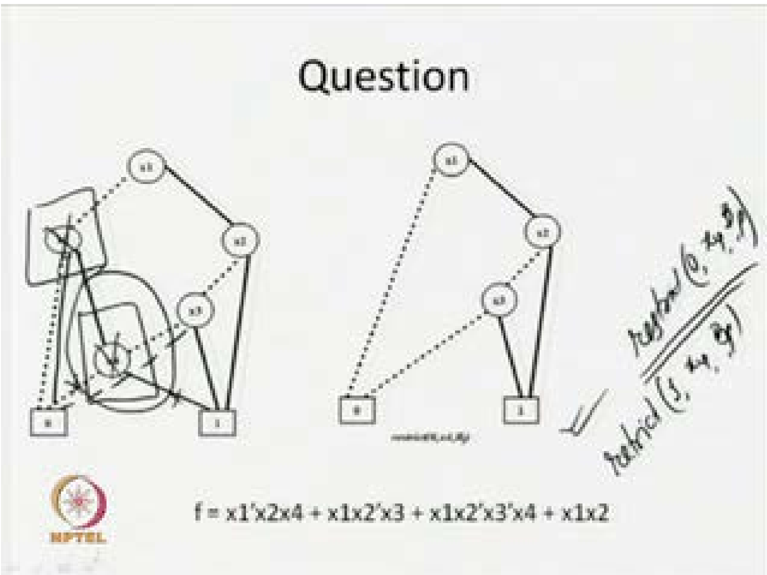
(Refer Slide Time: 50:17)



Now just look in the simple question over here I am saying that consider this particular Boolean function. So I am expressing a Boolean function now, construct the ROBDD's for this particular function f say we are going to say that B f. So I am giving a function f which depends on four variable x 1, x 2, x 3, x 4. First we are going to say that construct the ROBDD for B f reduced ordered binary decision diagram, then look for a structure of restrict 0 x 4 B f; that means, we are restricting x 4 to be 0 or restrict 1 x 4 B f or restricting the B f where x 4 to be 1; that means, we are restricting the function f where x 4 is equal to 1. So construct the BDD's and after that we look for this particular function exist x 4 B f; that means, we are relaxing the constraint on the variable x 4 for in the given function x. Just see how, what that BDD B f will look like, you can use the Shannon expansion to construct this particular BDD yeah I will just give you the resultant BDD. So this is the ROBDD's for this particular given function and ordering that we are going to consider is your x 1, x 2, x 3, x 4.

By using the Shannon expansion we can construct the OBDD and after the apply reduce algorithm to get the ROBDD and eventually will get this particular structure. I am saying that which are given variable ordering we are going to get the unique representation. So if this is your ordering you have always going to get this structure. So what is the first term x 1 prime x x 2 x 4; that mean, x 1 is equal to 0, x 2 equal to 1, x 4 equal to 1, the function will give me the value 1 similarly, x 1, x 2 bar x 3, x 1 x 2 bar x 3 it will give me evaluation one x one x 1 x 2 bar x 3 bar x 4.

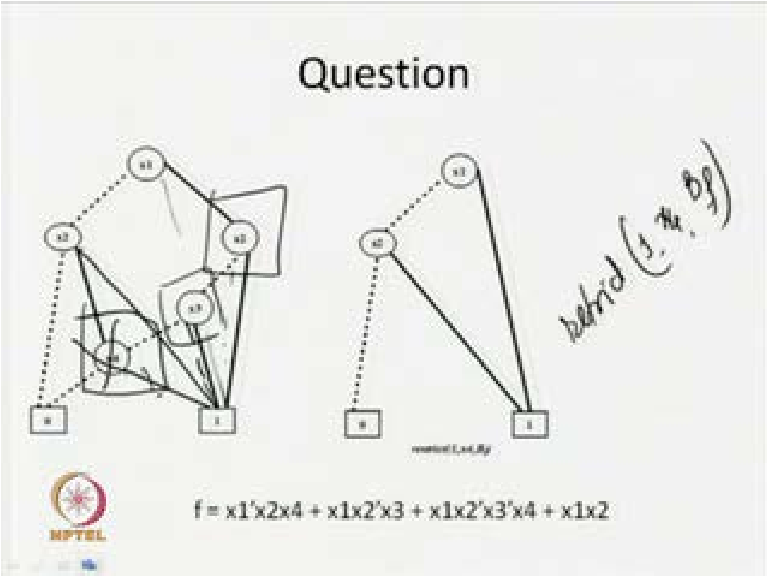
(Refer Slide Time: 51:44)



So this is another one value x and x 1 and x 2. So this is the OBDD representation of this particular given function and you can check that none of the relax rule can be applied over here. So you can say that this is the ROBDD of this particular given function. Now what I am talking about, first we are saying that apply the restrict operation over here restrict 0 x 4 B f. So what is the rules since we are going to restrict this function on a variable x 4 so we are going to look for all exponents and we are going to remove this particular x exponent from this particular BDD's. So we are going to remove this particular exponent. When we are going to remove this exponent? Then it is outgoing as will also be removed. So these two will be removed and now we are going to restrict it x 4 to be equal to 0.

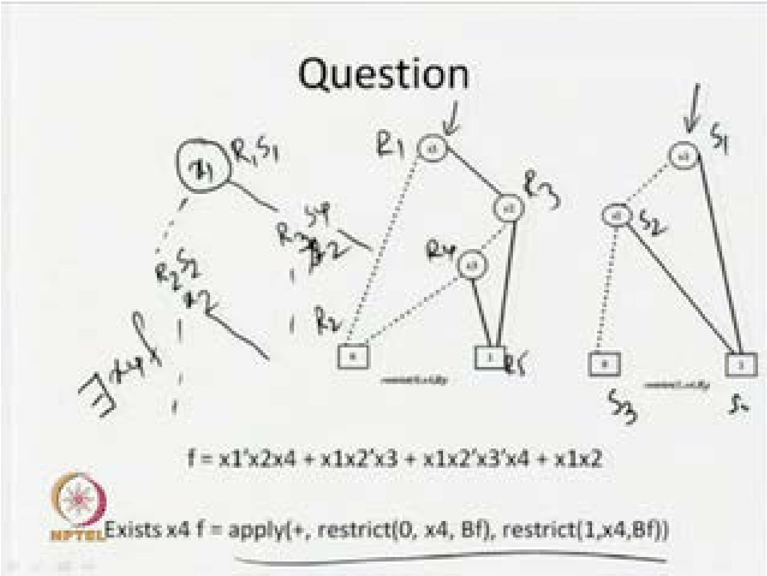
So we going to see that one x 4 all incoming x will be restricted 3 director to this particular dashed line. So this dashed line is redirecting in to 0; that means, this dashed line will be redirected to this particular 0 and x 4 is equal to 0 it is redirected to 0 that is why these are will be redirected to this particular 0. So; that means, after removing this particular portion we are getting this particular BDD and if you look into it this particular now this may not be a reduced one. Now we are going to look for the were the reduction can be possible or not what we will find? That will find that this x 2 is an redundant node we are having redundant test. So we can remove this particular x 2 also so this is application of the reduction algorithm. So eventually we are getting this particular BDD, x 1 is 0 then it is coming to 0, x 1is equal to 1 then going to talk take decision on x 2 then 0 and 1 then we going to take decision on x 3. So this is the ROBDD for restrict 0 x 4 x 3 now similarly, try to construct the BDD for your restrict one x 4 B f.

(Refer Slide Time: 53:52)



So in that particular case what will happen? Similar rules we are going to apply, what we are going to do? Will remove this particular x 4 and whatever incoming as we have that will be redirected to this particular one as of the particular x 4. So; that means, that one would straightly will come over here and this dashed line will come over here. So we are getting removing this things. So whatever resultant BDD we are getting it may not be a reduced one. Now just try see the rules now when I am coming to this particular x 3 then we are getting that it is having a redundant test. So we can remove it, after removing it what will happen that incoming as will be redirect to over here, again when I am going look into x 2 will find that this is another redundant test we are going to remove it. So after removing it that one as solid as we will be redirected to this particular one so eventually, I am getting this particular steps also this is basically steps for your restrict one x 4 B f.

(Refer Slide Time: 55:17)

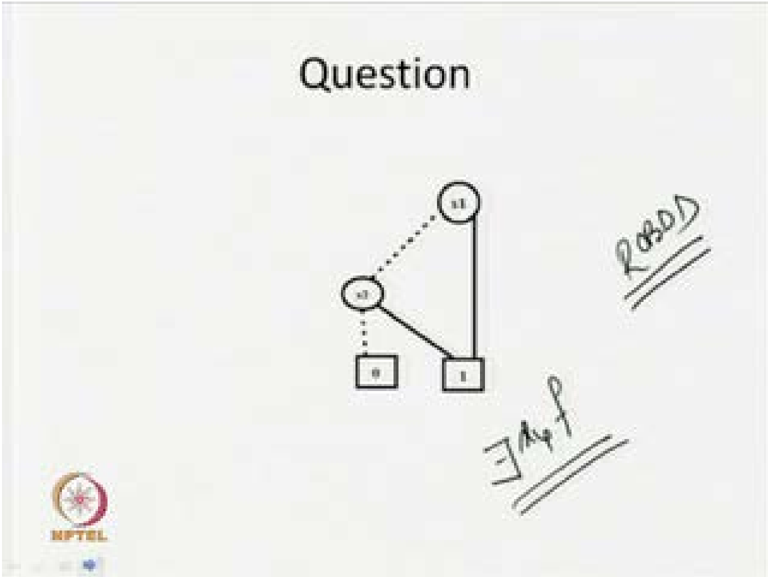


So here this is the we know the BDD's for this particular given function and we have not getting the BDD's for OBDD's called or in fact I can say that ROBDD's for restrict one x 4 B f and restrict zero x 4 B f. Once I am giving this particular two BDD's then always I can look for the exist there exist x 4 f; that means, and it is going to evaluate these things with the help of these particular expression a flip flop this restrict x 4 to be 0 and restrict x 4 to be 1. So; that means, I am having these two operation for restrict 0 and restrict 1. So in that particular case what will happen? The root node is your x 1 node, so in that particular case I am going to create an x 1 node and so what is the levels I can say that this is your r 1, r 2, r 3, r 4, r 5 say this is your s 1, s 2, s 3, s 4.

So I am going to create a x 1 node and I am just calling that this is your r 1 s 1. Now this dashed line will go and we are going to construct a solid line. So dashed line where it will go, low op this is r 1, so this is your r 2 and this is your low op your s 1 is your s 2 and in your solid line hi of r 1 is your r 3 and hi of s 1 is your s 4.

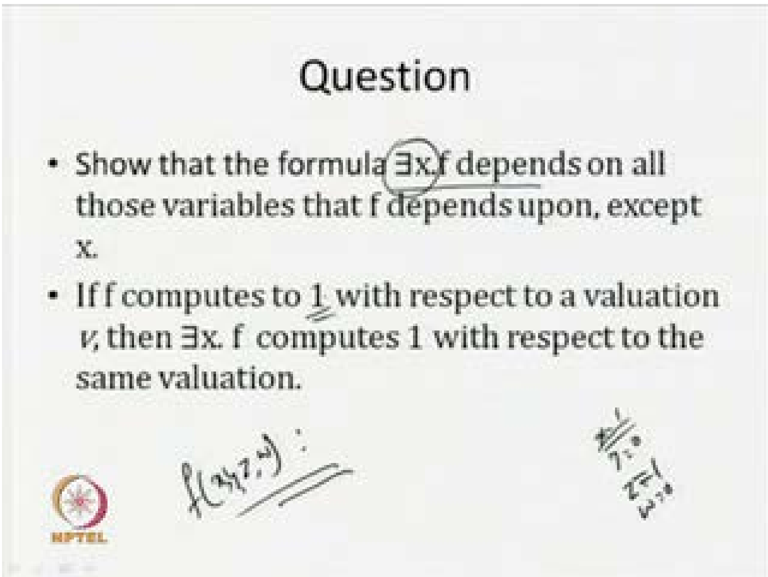
Now you just see that this is your r 2 s 2 and r 3 s 4. Now you also see that r 2 is terminal nodes, s 2 is a non-terminal node; that means, now since it is a non-terminal node so we are going to create a node x 2 similarly, here r 3 is a non-terminal node but, s 4 is a terminal node. So we are going to create an x 2 node at that particular point. Now similarly, I have to look for what is the solid and dashed line. So like that we can now use this particular apply algorithm and you can construct this particular BDD's.

(Refer Slide Time: 57:51)



Now after that we are going to get a resultant BDD it may not be a reduce one, now after that apply reduce algorithm to get the ROBDD's or this particular exist x 4 f. Now what will the structure look like? So you can find that eventually will get this particular ROBDD you test it, then this is your there exist x 4 f. The ROBDD representation of this particular given restriction they are relaxation there exists x 4 f.

(Refer Slide Time: 58:31)



So what will happen? After getting this BDD we try to construct it and what happen we are going to get you apply the reduce algorithm and eventually you are going to get this particular ROBDD.

Now just have a quick look on this particular problem so question, show that the formula there exist x f depends on all those variables, that f depends upon except x. So we are having a function f it depends on some variables say x 1 to x n also, some variables x, y, z like that. Now there exist x f what does it means? We are relaxing this particular function on some variables. So now basically what it is asking? That there exist some x f depends on all those variables that f depends upon except x. Now you just try to get the feeling what that there exist and takes and see whether this is correct or not. If it is correct it you establish it or if it is not correct then also establish it or given contra example as a very simple one, second one you seen that if f computes to 1 with respect to a valuation v say if I say that if f is a function x, y, z, w like that.

I can have the valuation of this particular function say x equal to 1, y equal to 0, z equal to 1 and say w equal to 0 say with respect to this valuation I am having going to have some valuation of these function. So if these valuation compute once then what it is saying that then there exist f computes 1 with respect to the same valuation.

Now what we have to show that, there exist x f will also compute 1 with respect to the same valuation. I think you might have a got the feelings yes indeed it is going to compute once why? Because there exist x f in this particular case we are going to relates the constraint of x. So the x equal to 0 or x equal to 1 it is immaterial. So if for a particular valuation if f computes 1 and there exist x you have to also compute to 1 with this particular same valuation, because there exist x f is to immaterial of this particular x is the x equal to 1 and 0, see x equal to 1 is going to give me one so on relaxation also it will be one. So with this I will complete my lecture here today.