

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

NPTEL

NPTEL ONLINE CERTIFICATION COURSE
An Initiative of MHRD

VLSI DESIGN, VERIFICATION & TEST

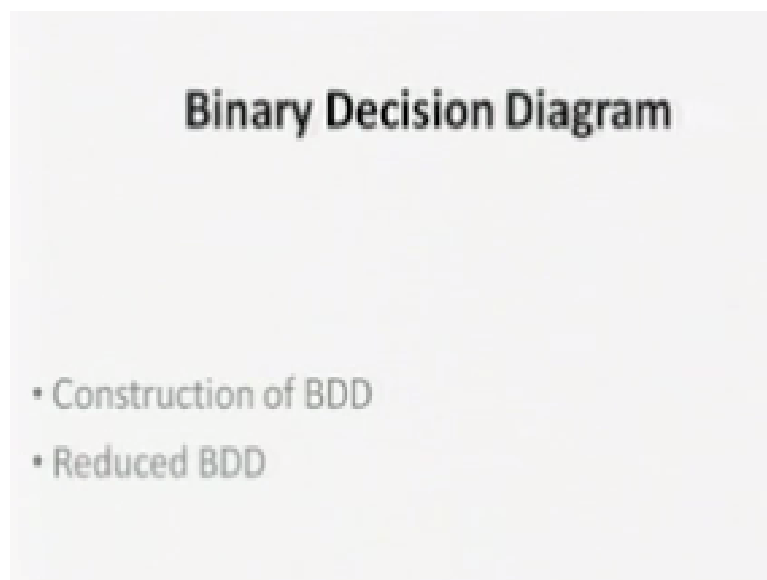
Prof. Jatindra Kr. Deka
Department of CSE
IIT Guwahati

Module VI: Binary Decision Diagram

Lecture II: Ordered Binary Decision Diagram

Ok in last class, we have introduced the function called Binary Decision Diagram with the help of binary decision diagram we can represent any Boolean function or expression. We have seen how to construct this particular binary decision diagram.

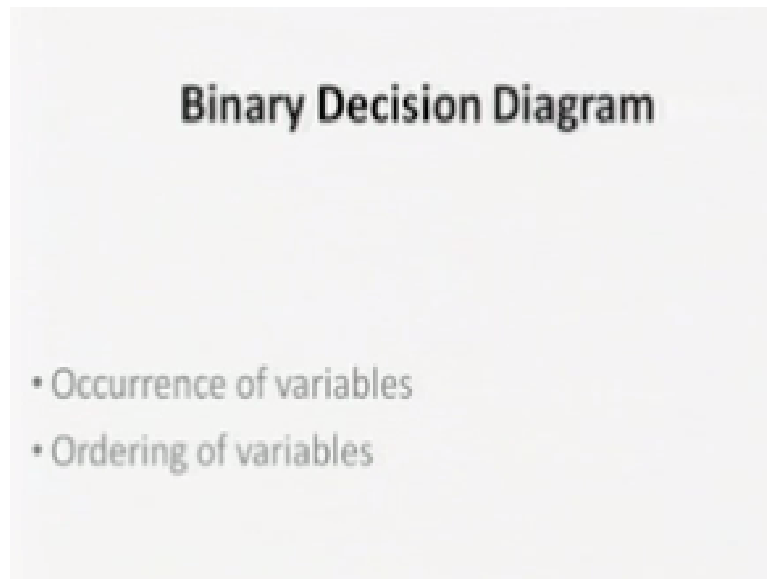
(Refer Slide Time: 00:43)



So construction can be done from your truth table, where we are going to get binary decision or you can use the senon expression or Boolean expression to construct the (BDD) Binary Decision Diagram and after that we have seen some reduction rule.

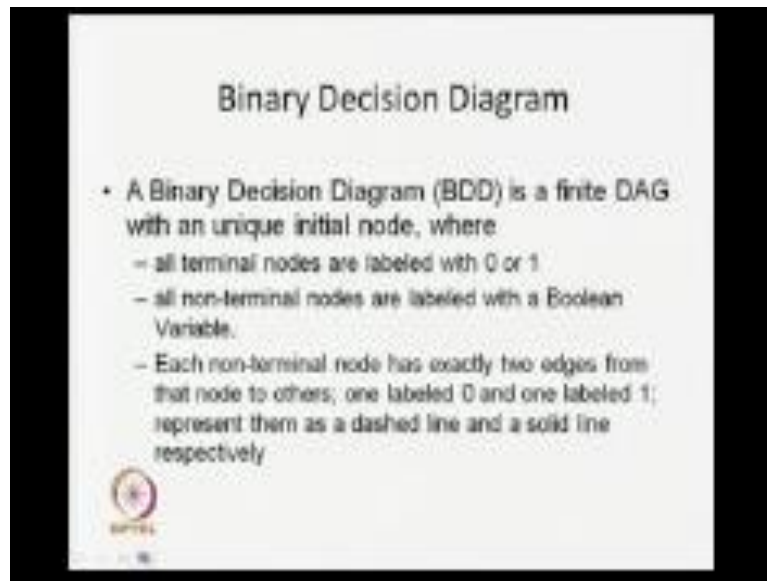
We have three reduction rules; one elimination of duplicate terminals, one of redundant nodes, and one of duplicate terminals. With the help of these three reduction rules we can reduce the BDD and we get a reduced binary decision diagram. In most of the cases we have seen that we are going to get a compact representation of a boolean function, if you use a reduced BDD (reduced binary decision diagram) but when you look into the construction of all BDD.

(Refer Slide Time: 01:40)



We will see that we are not putting any restriction on the ordering of variables and again, we are not giving any restriction on the occurrence of variables in a particular part. It may appear in many places because if you look into the definition.

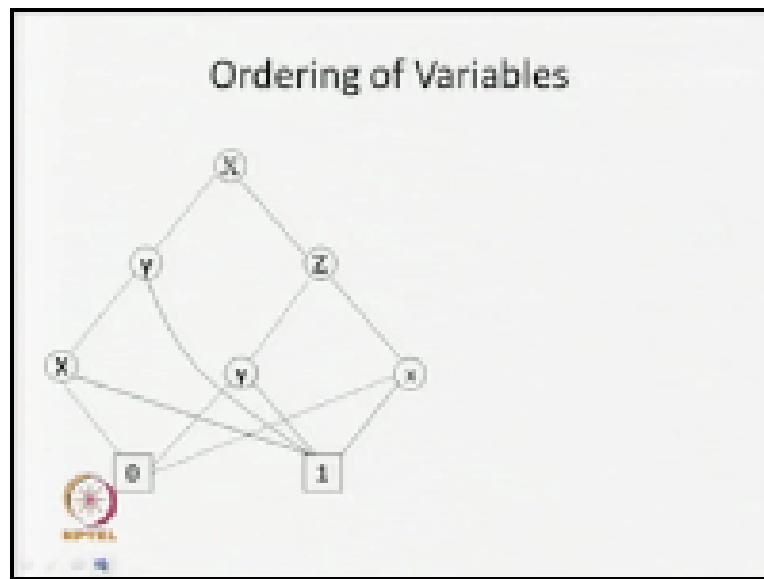
(Refer slide time; 01:59)



We are saying that a Binary Decision Diagram is a finite DAG with a unique initial node, where all terminal nodes are labeled with 0 or 1. All non terminal nodes are labeled with a Boolean variable. So if we are having a Boolean Expression, we are having some particular about that boolean expression and the non terminal nodes will be labeled by this particular boolean variable, and each non terminal nodes have been throughout going as a one represented by dashed line with special indicate, it is the Evaluation of the variable is 0 and another one is given by solid line which is the valuation of this particular variable is 1 ok.

So this the definition of Binary Decision Diagram and we can construct the binary decision diagram of any boolean expression and why we are taking about this can you look in to the definition of this particular BDD, this is not anything about your occurrence the variable, how many time its acquire? So it is not keeping any particular ordering, so if you look in to this particular binary decision diagram definition.

(Refer Slide Time: 03:15)



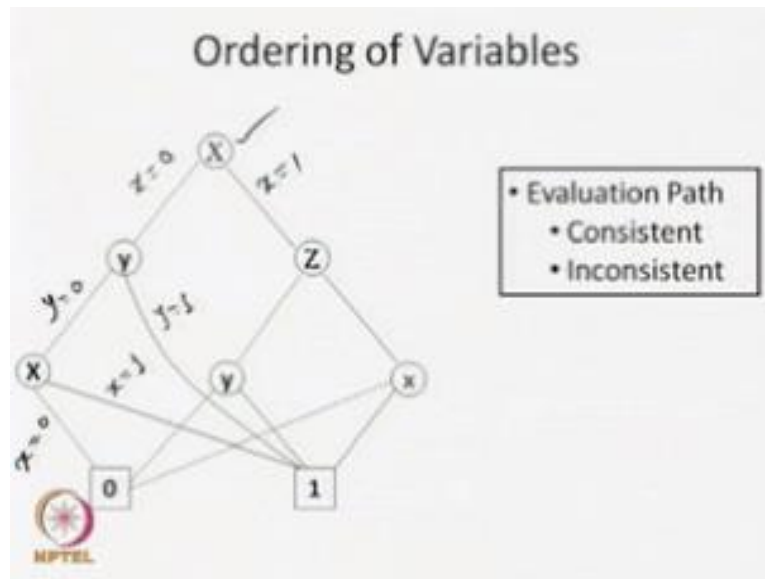
Then we will say that this is the binary decision diagram ok, this is simple that we having three variable. So this is the function of three variable of $f(x, y, z)$, so x may be a over here then this line indicates the illusion of this x is 0 and indicates the illusion of this particular x is 1 like that y and z are another variable and these are the two terminals one is 0 and second one is 1.

If you look in to it you see that here, in this particular case, this particular x is appearing twice in this particular part. There is for definition, we do not have restriction. Similarly, you can say that y is also appeared in two different position but they are in the two different parts. Now in this particular case you can look in to it, then by looking in to this particular variable BDD.

Now we are going to see some problems, we are taking in that particular case we are going to have a note on of consistent of evaluation path and inconsistent of evaluation path. So when we draw this particular BDD, some of the part may be inconsistent ok. I will say 5 in inconsistent, so they evaluation of Boolean function will be done to consistent part only ok.

Now you see that I am saying that this is x variable x here I am taking X=0 and this particular part I am taking X=1. When X=0 and I am going to take decision of Y, this particular part Y=0 and this particular part y=1 and after when I come to this particular node then again I am going to take decision on X and I am going to say that if I follow this part then x=0 and if I follow this particular path then X=1.

(Refer Slide Time: 05:03)

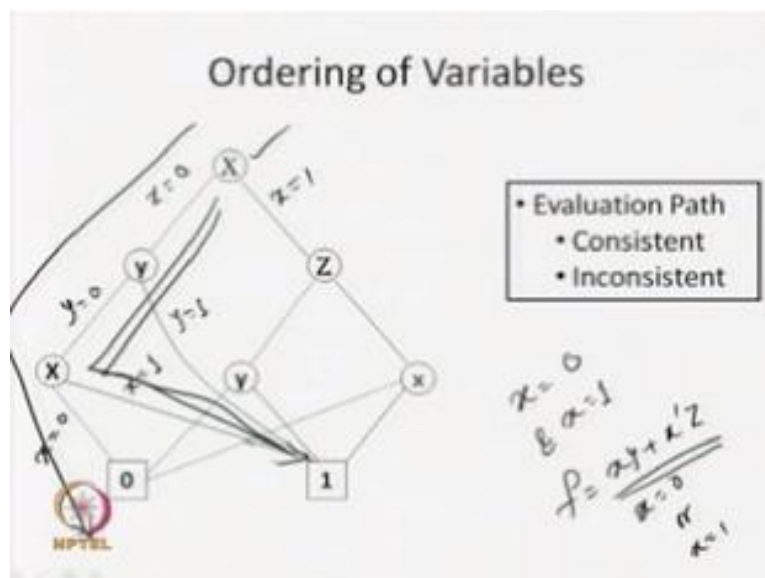


So this is where we can see and ultimately going to get the function, so if I follow this particular path then X=0 and y=0 and we are going to say that functional boolean is 0 and if X=0 And y =0. It is independent in this particular evaluation part, and if I look into this particular part of evaluation ok single double line, now whatever I say that x=0 and evaluation of Y=0 and X=1.

In this particular part, you can see that evaluation of X is taken as equal to 0 and it is also taken as X=1 and when we look for any particular part of evaluation and any instead you can have only one evaluation for one variable, just in that say $f=xy+x'z$. Now in this particular evaluation I am going to look for the evaluation of the Boolean function. I am going to say that evaluation of X=0 or I am going to say that X=1.

So for this two different combination, I am going to get two different evaluation but in this particular part what will happen the values is treated as $X=0$ as well as $X=1$, which is not possible. So in this case we are going to say that this is an inconsistent, so that means we have notes on inconsistent part of BD, if we look into the basic definition of BDD.

(Refer Slide Time: 06:40)



The evaluation of the function, that is evaluation of function has to be done through consistent part only that means we should not do in inconsistent part and due to this particular problem, we are trying to eliminate this particular problem and in this particular case what we are going to do? We are going to put some restriction on occurrence of the variables and in the particular case we are going to put particular order of set of variables and after putting this particular order of variables and now we are going to get ordered BDD ordered Binary decision Diagram.

(Refer Slide Time: 07:15)

Ordered BDDs (OBDDs)

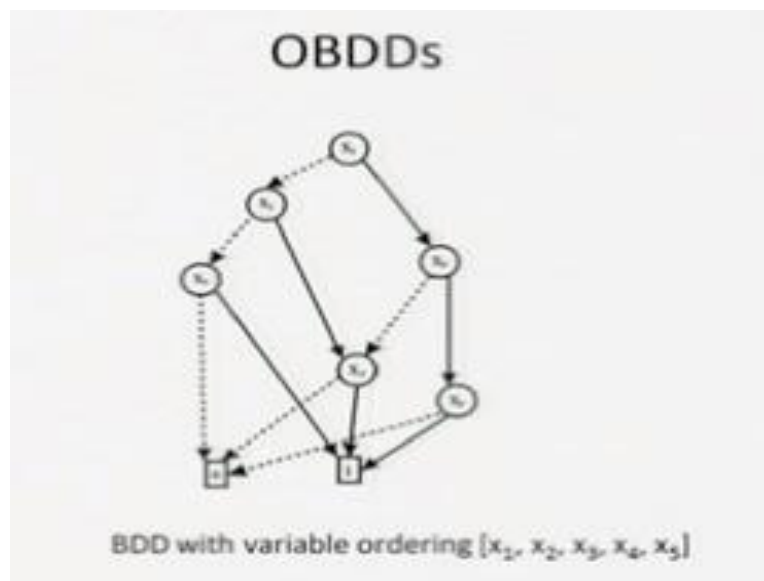
- Let $[x_1, x_2, \dots, x_n]$ be an ordered list of variables without duplication and let B be a BDD all of whose variables occur somewhere in the list.
- We say that B has the ordering $[x_1, x_2, \dots, x_n]$ if all variable labels of B occur in that list and, for every occurrence of x_i followed by x_j along any path in B , we have $i < j$.

So in case of Ordered Binary Decision diagram, that means variable will come in a particular ordered. Let Consider that $X_1, X_2,$ and X_n be an ordered list of variables but without duplication and let BDD all are whose variable occur somewhere in the list. Now we are considering the BDD, where all the variables are of this particular list occurring somewhere in this particular BDD and we are saying this ordered BDD where duplication is not allowed.

We said that B has ordering that X_1, X_2, \dots, X_n . If all variable labels occur in the list and for every occurrence of X_i followed by X_j along any path in B , we have $i < j$, that means if we are going to follow this particular ordering say X_1 to X_n . So when we draw this particular BDD then X_1 must always come before X_2 or may be X_1 always come earlier point of X_n you should follow in this particular order.

And when we are following restricting this particular order of BDD, then you will found that on particular part the variables will appear only ones, that means since it appears only ones the multiplicity of Evaluation that $X=0$ and $X=1$ the where we see the earlier example we will not talk about that, that means in case of ordered of BDD we are going to following the particular order of the variable ok. Just see simple example.

(Refer Slide Time: 08:47)



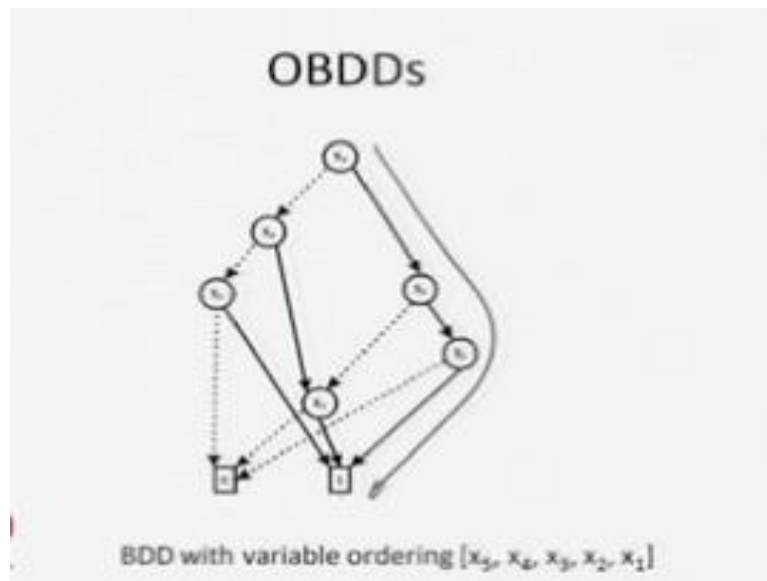
You see that, I am having BDD of what we are having 5 variables and we considering this particular ordered variables X_1, X_2, X_3, X_4 and X_5 , this is the ordering of the variable. When we mention, this particular ordering what we have to see that X_1 must always occur before X_2, X_3, X_4 and X_5 similarly X_3 must always occur before X_5 in any part in this particular variables.

So if you look at any part you will find that this is X_1, X_2, X_4 , and then we are having devaluation value. So we are saying that this is particular ordering X_3 is not coming before X_2 in any of the part similarly X_5 is not coming before X_1 in any of the evaluation. So in this particular case we are going to say ordered Binary Decision Diagram, so that means the variables are following a particular ordering.

Since they are following a particular ordering the duplication or occurrence of a multiple times of a variable in a particular part is not allowed ok. So this is where we say this is called Ordered Binary Decision Diagram and similarly this is another Binary Decision Diagram Ordered Binary decision diagram.

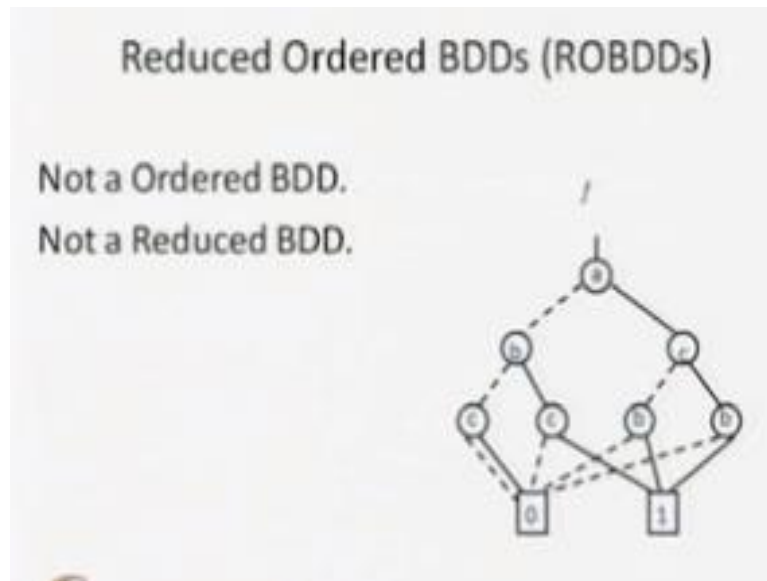
In this particular case, we are following a particular order $[X_5, X_4, X_3, X_2, X_1]$ that means, when you look in the evaluation part, in this particular part the variables must occur in this particular order that means X_5 will come first then X_4 , then X_3 , then X_2 and then X_1 to the terminal node 1.

(Refer Slide Time: 10:22)



So that means it will follow this particular order. Now when we are using this particular order this is in diagram. Then variable will appear in a particular order in any execution method or evolution part of this BDD, so appearing of particular variable multiple times in a particular part is abounded that means the inconsistent part is abounded that means all evaluation part are of inconsistent so this is the Ordered Binary Decision Diagram. So we are going to follow a particular order of the variable.

(Refer Slide Time: 10:59)



If you consider this particular order of ROBDDs, I think in the last class and also, I have done the particular BDD. Since this level over here, so in this particular class I am saying that this is not a ordered ROBDD. Why it is not an ordered BDD of a particular order? So if I follow this particular part than the ordering is a, b, c. This is the ordering of a variable but if I follow this particular order our evaluation part then I am going the get the evaluation part a, c, b ok.

You now considered A is appearing before b, c but in this particular part c is appearing before b and b is appearing c. So that means it is not following any particular ordering, so that means it is not ordered BDD. Secondly you are saying that this not a reduced BDD, why? You can see some instances look into this particular node C, we are having $c=0$, $c=1$ if it evaluating to 0 that means this is redance, so this redance can be removed.

Secondly, if you look into this particular two nodes $b=0$ it evaluates to 0 b here also $b=0$ it evaluates to 0, in this case $b=1$ evaluate=1, $b=1$ evaluate to 1 that means deserve, you consider duplicate non redance, so this two non residence can merged to one non redance. So since, we can apply the reduction rule to this particular BDD.

So it said to be not a reduced BDD. So this is not ordered because we are not having a similar ordering BDD and it is not reduced BDD okay. Just a simple example I am that giving now see the Impact of the chosen variable.

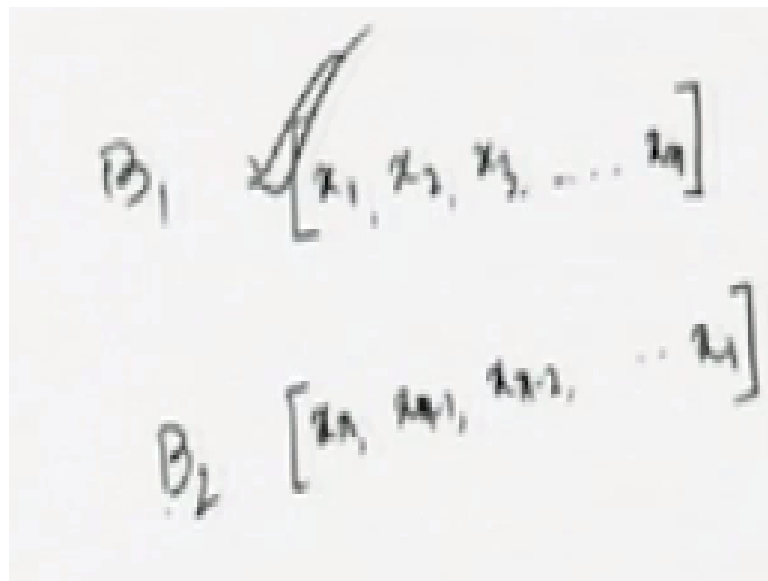
(Refer Slide Time: 12:49)

Impact of the chosen variable ordering

- In general the chosen variable ordering makes a significant difference to the size of the OBDD representing a given function.

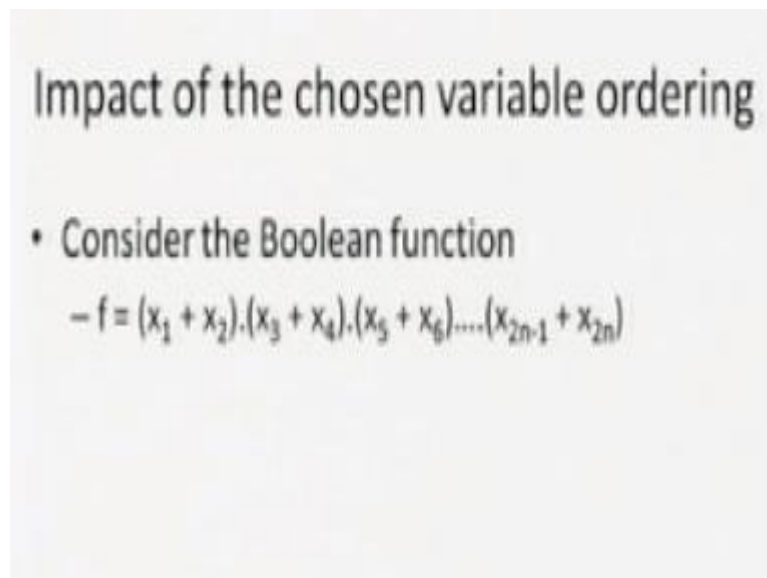
Now consider that I am going to say that, we are going to put a variable ordering $x_1, x_2, x_3, \dots, x_n$ this may be one variable ordering. Secondly I can say that $x_n, x_{n-1}, x_{n-2}, \dots$ like that up to x_1 . So with the help of this particular variable ordering, I can get a BDD and say that zero BDD B_1 and the help of this variable ordering I can have the BDD representation on the given function and I say that is your B_2 .

(Refer Slide Time: 13:34)



Now we are getting this two BDD, this are representing the same Boolean function. Now after that we will use the reduction of these things we are going to get Reduced Ordered Binary Decision Diagram okay. In this particular case, we have to see what is the size of B1 and B2? This may happen that the size may vary in both the cases, one may be bigger one and one may be a smaller one. So that means the ordering of variable is going to matter a lot or the size of the BDD ok. I will explain these things with the help of the one simple example.

(Refer Slide Time: 14:13)



Impact of the chosen variable ordering

- Consider the Boolean function
– $f = (x_1 + x_2).(x_3 + x_4).(x_5 + x_6)....(x_{2n-1} + x_{2n})$

Consider this particular Boolean Function we are saying that $f = (x_1 + x_2).(x_3 + x_4).(x_5 + x_6)....(x_{2n-1} + x_{2n})$ that means we are having y and variables. So this is a simple function.

(Refer Slide Time: 14:34)

Impact of the chosen variable ordering

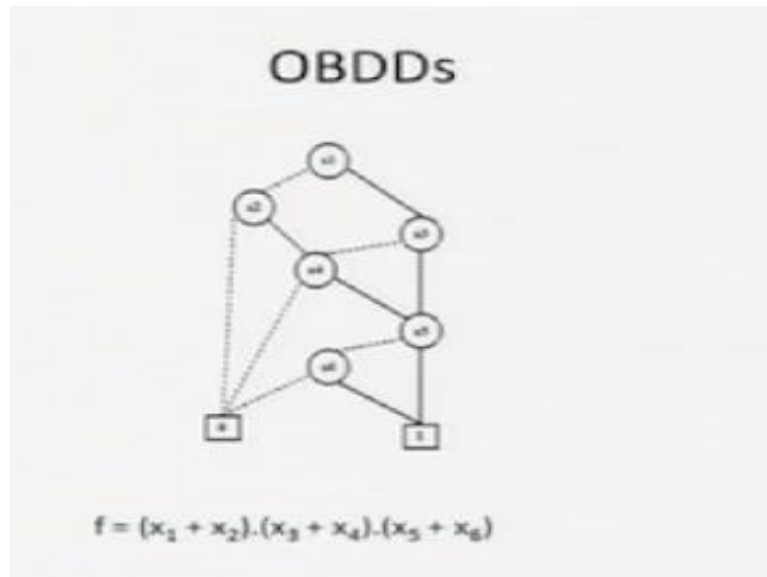
- Consider the Boolean function
– $f = (x_1 + x_2).(x_3 + x_4).(x_5 + x_6)....(x_{2n-1} + x_{2n})$
- If we chose the variable ordering $[x_1, x_2, x_3, x_4, \dots]$, then we can represent this function as an OBDD with $2n+2$ nodes.
- If we chose the variable ordering $[x_1, x_3, x_5, \dots, x_{2n-1}, x_2, x_4, x_6, \dots, x_{2n}]$, the resulting OBDD requires 2^{n+1} nodes.

Now in this particular case, if I chose a variable ordering $X_1, X_2, X_3, X_4, \dots$ that is one particular variable ordering. Then in the particular case what happens we can draw the BDD or the particular function and the ordered BDD will have n and variable plus 2 nodes. So we can construct the BDD for the particular Boolean function and that particular Boolean function will have n and variables plus two nodes and will see that no more reduction can be possible and we can say this is a Reduction Ordered Binary Decision Diagram.

On the other hand if we choose a variable ordering or a varying something like that $x_1, x_3, x_5, \dots, x_{2n-1}, x_2, x_4, x_6, \dots, x_{2n}$ that means all the odd ones I am going to put first and then even scripton put in data. So this is also possible by variable ordering. So if you use this particular variable ordering than we will see that total number of nodes that we will be having is 2 to the power of n plus 1.

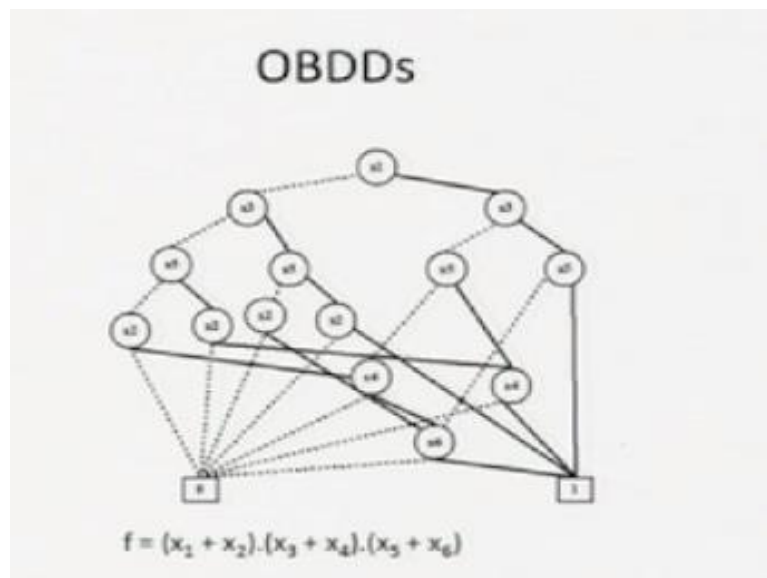
So if you are having n and variable the total number of your this things, that nodes that will appear in the BDD nodes to 2 to the power n plus 1. And the variables it is twice and plus nodes. So you can see that you have a constantly difference above, size of the BDD on variable ordering.

(Refer Slide Time: 16:04)



So this is the OBDD I have drawn. This basically the same function $X_1, X_2, X_3, X_4, X_5, X_6$ and the variable ordering that we are having is your X_1, X_2, X_3, X_4, X_5 , and X_6 . So in this particular case I am getting this particular BDD and I am having 6 non terminal nodes then two terminal nodes. Now if I send the variable ordering then we are going to get this particular representation.

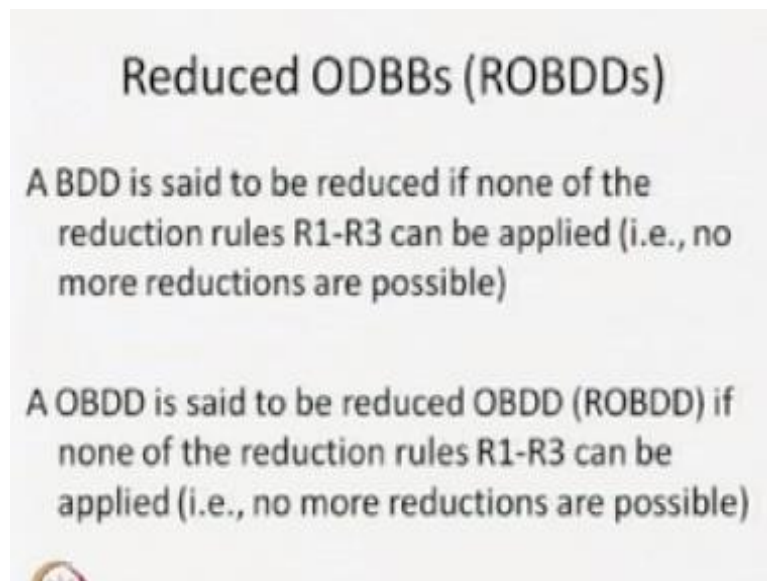
(Refer Slide Time: 16:35)



So here the variable ordering basically I am taking $x_1, x_3, x_5, x_2, x_4,$ and x_6 . This is the variable ordering x_1, x_3, x_5, x_2, x_4 and x_6 like that. So in all the evaluation part you will find that this following this particular order variables. Now when I am having this variable ordering then you see that I am getting a BDD ok.

So this is same function, I am representing one ordered BDDs ordered, ordering is from x_1, x_2 to x_6 and the same boolean function and I am representing with another order BDD where the order is different, you have seen that the size is more. So the size is basically size of the BDD difference on your on the area will order and if you look into this particular things no more reduction is possible of the particular BDDs and on the other hand this BDD is also no more reduction is possible. So this is basically Reduced Ordered Binary Decision Diagram.

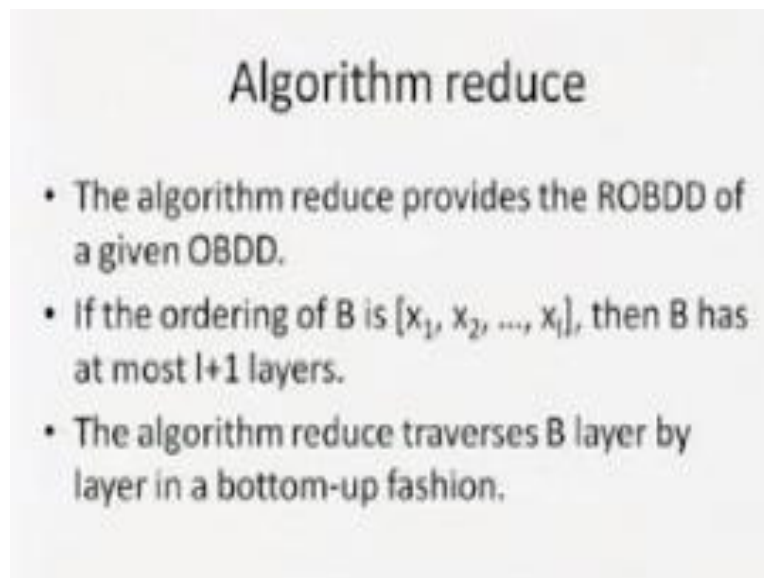
(Refer Slide Time: 17:40)



So that is why we are saying that a BDD is said to be reduced if none of the reduction rule R1-R3 can be applied ok and similarly for ordered BDD ordered binary decision diagram we are going to say that it is a reduced ordered binary decision diagram ROBDD if none of the reduction rules can be applied and if ordered ok. So eventually we are coming to reduced ordered binary decision diagram.

So it is a binary diagram of having a particular order binary decision diagram and thirdly it is a reduced one no more deduction rule is possible, so you are going to see this is a Reduced Ordered Binary Decision Diagram. And eventually we are going to walk with ROBDD which is having all the referenced prospects for us.

(Refer Slide Time: 18:29)



Algorithm reduce

- The algorithm reduce provides the ROBDD of a given OBDD.
- If the ordering of B is $[x_1, x_2, \dots, x_i]$, then B has at most $i+1$ layers.
- The algorithm reduce traverses B layer by layer in a bottom-up fashion.

Now in this particular case you just see that already I have mentioned about this particular ordering of variable depending on the ordering of variables. Now size of ordered BDD will vary in some cases we are going to get a compact representation and in some case we are going to get slightly bigger representation of ordered BDD or bigger reduced ordered BDD.

Now how to get the proper variable ordering so that we can get a compact representation of this particular function so that means we have to look for a proper variable ordering but to get a proper variable ordering to get a compact representation of the function in BDD is the hard problem.

We do not have any matter or we do not have any algorithm to say that this particular variable ordering is best variable ordering and it will give us a compact representation of a Boolean function so this a hard problem, it is difficult to find out the particular based variable ordering, so this is open problem peoples are still working on it whether it can be solved or not.

Since it is open problem, it is a hard problem so currently we are using some realistic and with the help of the series, we will try to find out some variable ordering which are going to give us some compact representation. It may not be the minimal one but the BDD what we are going to get may not be the minimal one but it may give us some compact representation.

We apply some realistic, and by using this realistic we try to find some variable ordering and after that apply this particular variable ordering to construct a BDD and in the most of the cases we have found it we are getting a size of BDD of representation of the function okay. Now we have introduced a data structure called BDD by the decision diagram which is used to represent Boolean function and later on we have seen ordering of the variable and the reduction of the particular ordered binary decision diagram.

We are eventually coming to Ordered Binary Decision Diagram and we use an ROBDD to represent the Boolean function. Now after that, since we are having this particular BDD represent the Boolean function we may need some algorithm or we need some beneath some method to work with that particular BDD. Now once you succeeded, I am giving you that Boolean function and you have come up with the variable ordering are you have seen a particular variable ordering then you are going to construct a BDD Ordered BDD for order particular function.

You can very well construct it with the help of senon of the boolean function, but eventually whatever you are getting it, it is reduced one or not you have to check it or if it is not a reduced one you have to construct a reduced BDD Reduced Binary Decision Diagram and already you have mention that you have three rules to go for a reduce BDD and we can apply the three rules to the initial BDD.

Now we are going see an algorithm, where we are putting this particular rule in algorithm form and we will say or we will get called algorithm reduce and we will apply this particular three rules to solve any algorithm reduce BDD. So basically what will happen in this particular algorithm reduce we are going to get a BDD or basically we are going to get a Ordered BDD as a input and the output we are going to get Reduced Ordered BDD and the variable ordering of the output BDD will be same as with the input BDD.


So we will look into this particular Algorithm, so if we are going to look an ordering of some variable say X_1, X_2, X_l then you are having the l variables then we can have at most $l + 1$ layer in BDDs. So we are going to have $l + 1$ layer variables at most. But in some cases in BDD it may be less because that particular function may be independent of that particular variable

The algorithm traverses this particular, the algorithm traverse the B that binary decision diagram layer by layer in the bottom of this also. So we are going to have a representation about and this algorithm traverse from the bottom layer to the top layers. So that means we will start from terminal nodes and after that we will traverse by to the route nodes and eventually we are going to apply some rules, so that we can get the reduced BDD okay.

(Refer Slide Time: 23:45)

Algorithm reduce

- We assign an integer label $id(n)$ to each node of B.
- $id(n)$ equals to $id(m)$ iff, the subOBDDs with root nodes n and m denote the same Boolean function.

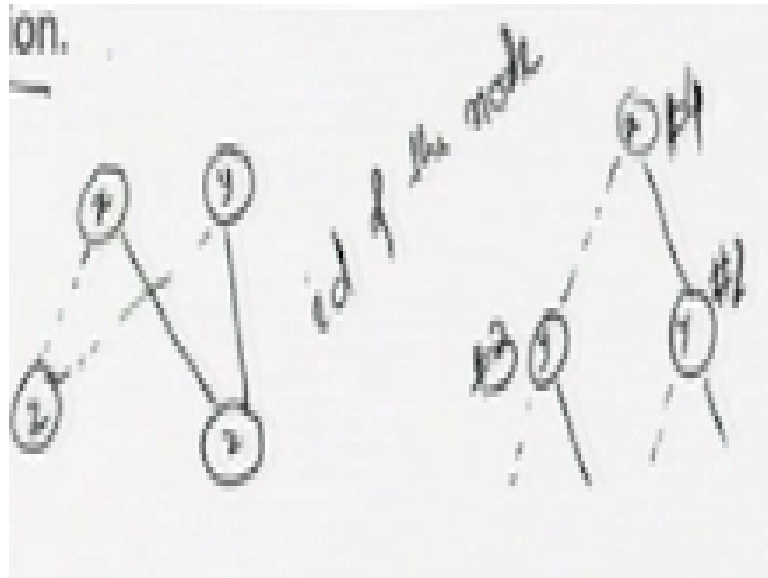


Now here in this particular case, we are going to assign some integer label to each node. So first one is the leveling of the node, so if I am having some may be BDD like that say this is your x and variable y . So we are going to have some variable, so in this particular case we are going to assign some integer label to each node. I can say that, I am assigning this level 3 to A and I am assigning that and this is a levee 1 2 to 8 and I am going to say that this is a level 4 assigned to this level.

So this is a integer level of this particular nodes and we are going to say that it is basically the ID of this particular node. So basically we are going to a Id, id is nothing but a it is the integer level of each node and we can say that id of the particular node and will be equal to id of and if the sub OBDDs with the root nodes n and m denotes the same boolean function.

So basically, if sub OBDDs denotes the same node of boolean function we are going to give them the same root node what you call integer level, just I will say that this is one node say level with X and other node say Y. So which evaluates to 0 layer.

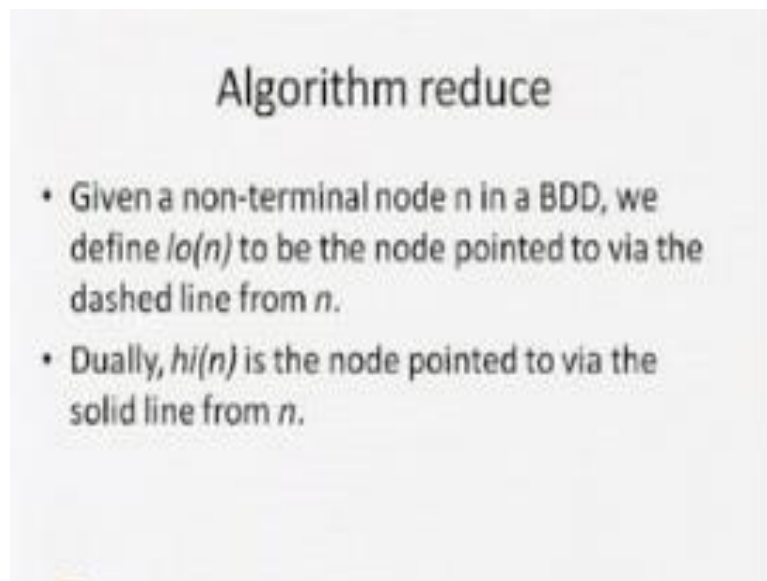
(Refer Slide Time: 25:05)



Now in this particular case, what will happen you just see that and after that you may have this particular BDD okay? Now apart from this particular node and I am talking about this m, so once you take $X=0$ it is going to look for this particular sub BDD similarly for this particular sub OBDD $s y=0$. So in this particular case $X=0$ they are going to have the same sub node.

So one both are having the same sub OBDD, that means both these particular nodes are in the same sub formula for $X=0$ and $X=1$. So in this particular case, we are going to give a same level to this node and m, so we are going to give some level or id and this level will be the same for this particular node. So they are going to have the same sub BDDs then the level will be the same, this is the rule to assign the level to each and every node.

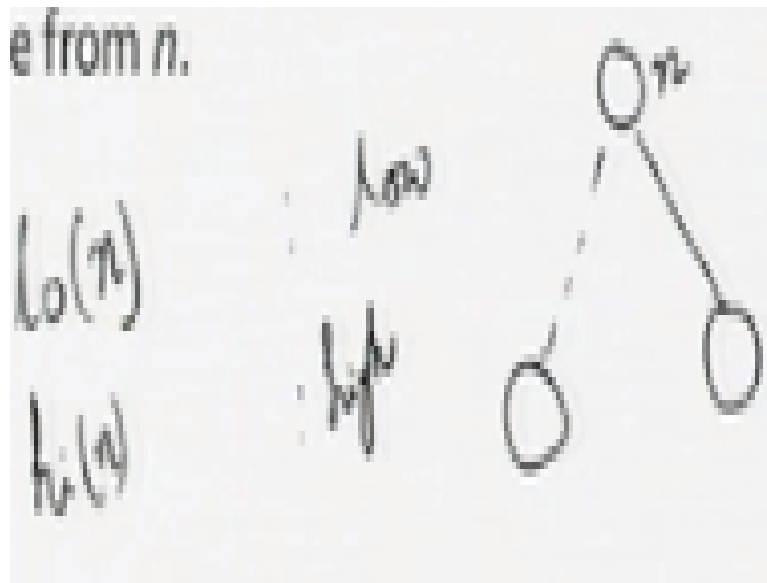
(Refer Slide Time: 26:16)



The slide is titled "Algorithm reduce" and contains two bullet points. The first bullet point states: "Given a non-terminal node n in a BDD, we define $lo(n)$ to be the node pointed to via the dashed line from n ." The second bullet point states: "Dually, $hi(n)$ is the node pointed to via the solid line from n ."

And analog one, we are going to define which is specific two functions one is your low and one is your high. So for every non terminal node we define a function called $lo(n)$ this is basically low to the node pointing to hire the best line of n and similarly we can have $hi(n)$ which is basically high. So if I am having in any node say n over here so we can have that this line and this solid line ok.

(Refer Slide Time: 26:52)




So in this particular case the level of this particular node is 3 and the level of this particular node is 4, so in this particular low it is going to return and this particular level because it is low to be the node pointed to higher the base time so it is pointing in this particular nodes. Similarly high will return the level 4 because it is pointing the level 4. So basically low lo and high hi are the two functions, it is going to give me the nodes pointed by this lines and notes pointed by the solid line respectively.

Now with the help of this things we are going constrain this particular reduced algorithm. So first task is to label it is an app menu.

(Refer Slide Time: 27:42)

Algorithm reduce

- Labeling of terminal nodes:
 - Assign the first label (say #0) to the first 0-node it encounters.
 - All other terminal 0-nodes denote the same function as the first 0-node and therefore get the same label.
 - Similarly, the 1-nodes all get the next label (say #1)
- Reduction Rule (eliminate duplicate terminals)



Now next task is to label the nodes. So how we are going to label the nodes? So label the first level 0 to the first 0 node it encounters. So basically we are going to follow from the terminal nodes and follow in the bottom of x nodes. So first we are going to get some node, 0 node we may have one 0 node and more 0 node level by 0. Then we are going to assign the final level 0 to that particular node.

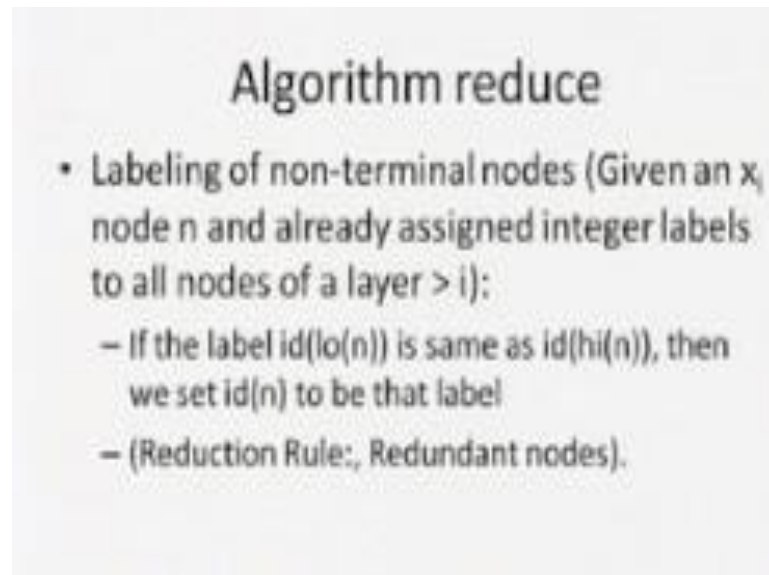
Similarly, we are having some nodes which are level terminal node, which are valued to 1 and we are going to assign level 1 to that particular node. So we are having the more 0 node, we are going to keep the same level to those particular 0 node or we going to keep 0, level 0. And we may have more 1 node to the entire one node we are going to get level 1.

So basically this is the starting of all labeling algorithm and we are starting from the terminal nodes and we are going to follow the bottom of nodes. Now we are going to follow the bottom nodes. Now what we can say, now in this particular case we are giving the level 0 and 1. Now let on what we are going to do, the nodes which are having the same level will be merged together.

Basically this is the second case about the algorithm. First we are going to note each and every node and next we are going to merge the nodes which are having the similar level. So in this particular case all 0 nodes will be level merged to 1 node and 1 node will be merger to 1 node level by 1. So this is nothing but the reduction rule that we have elimination of duplication terminals.

So that means we are going to take care of this duplication terminals ones. So removal of duplication terminals of all 0 nodes will be merged to 1 and all 1 node will be merged to 1 ok.

(Refer Slide Time: 29:40)



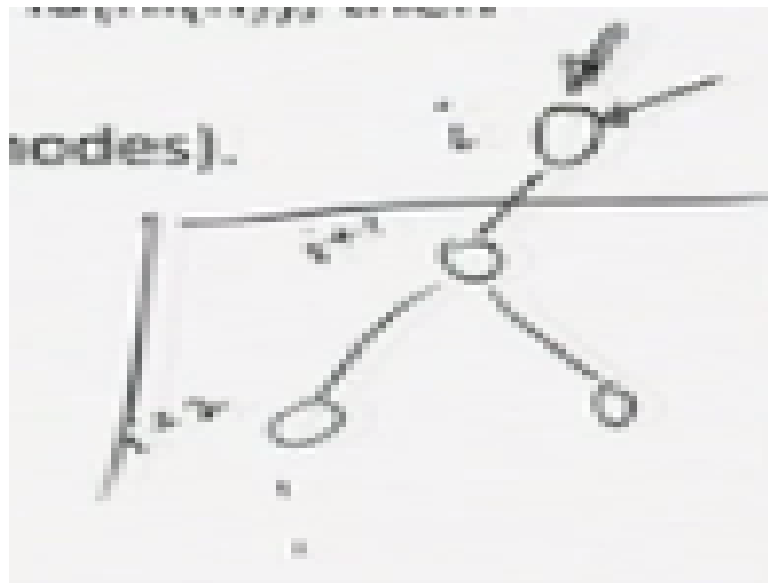
Algorithm reduce

- Labeling of non-terminal nodes (Given an x_i node n and already assigned integer labels to all nodes of a layer $> i$):
 - If the label $id(lo(n))$ is same as $id(hi(n))$, then we set $id(n)$ to be that label
 - (Reduction Rule: Redundant nodes).

Now ones we label this particular terminal node, next we are going to follow the bottom of nodes and we are going to get from one terminal node. Labeling of non terminals nodes, given a X_i node and already assigning integer labels to all nodes of a layer i is layer is get 1 that means 1, we follow that particular labeling. So this is I am coming level i then $i+1$, then $i+2$ like that I have to compute the level.

When we come to level this particular node then what will happen? All the nodes below this particular level must be marked ok. You should have the entire integer labeling of those nodes below this particular nodes then only we can get the nod. Since we are following the bottom approach, so when we read this particular node then the entire node below this particular node will be ordering level.

(Refer Slide Time: 30:39)



If the label $id(lo)$ and is same as $id(hi)$ then we said id and to be end line of that level, that means you can say that, if I am giving the particular node over here and n is low and high. Both are coming to 1 particular node say m , so in the particular case say i will be of low n say this is I can have say some level 3. So id of low and equal to 3 similarly id of i of $n=3$ so if both are same, then what will happen we are going to keep the same level to this particular nodes.

So we can say that this is also level 3. So what basically it means that now, in the next place we are going to merge them together. So these will be merged together because they are having the same level and it is said that this is nothing but the reduction rule of the redundant node. Removal of redundant node because here I am having some redundant test for both 0 and 1, it is going to give a sub OBDD.

So if the $id(lo)$ and $id(hi)$ is same then and we will get the level of these particular n ok. So this basically the leveling putting the integer level and basically it is going to help us to use this integer level with the reduction rule, removal of redundant rule ok.

Next we are going to say that already I have said that if I am coming to that x_i node then all the nodes below that level is already marked. Now if there is another node m such that m and n have the same variable x_i that means I am having a node n and I am having a node m ok and both the level which m variables say x and y .

Now what will happen in this particular case, if $id(lo(n))=id(lo(m))$ that means this is low ok. So some label having oh sorry. This is lo similarly i and i coming to the same node or it may cover to the define node but both are having the same level. So in this particular case say low and low n and low m is coming to the same sub BDD and similarly high and high n and high m are coming to the same BDD. Then id of n will be equal to id of m.

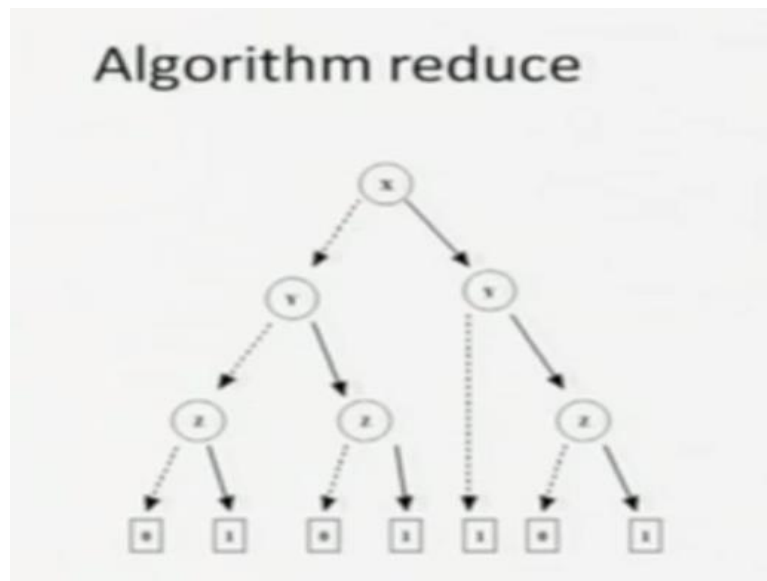
So if I am having a already say id of this particular node say m equal to 5 and then I am going to say the same level to this particular node also and it is 5. So if you see this thing later on, we are going to merge them together ok that means we are, this is basically correspondence to the rule of removal of your duplicates nodes .Basically this two are duplicates because they are having the same sub BDD below that we are going to evaluate that same sub function.

So we can merge them together, so this is basically nothing but the removal of this particular duplicates node ok. So in this particular case if this is not getting these two nodes, so I am coming to the particular node xi and the entire node below this particular node is level and if it is following this two particular rule say, this is the first one and this is the second one.

This is not following these particular two cases then what will happen otherwise we said id when to be the next and use integer level. So here starting with integer level 0 for the 0 node, then we are starting with integer 1, then if they are not equal then I will go to next unused 2, then next unused number 3 and like that. So if it is not following these two particular rules that we have already mentioned then say id n to be the next unused numbers.

So next unused number will be given with the, it means it is an new node. We are getting and it will be appeared or presented in the ROBDD, so ones we are having this things.

(Refer Slide Time: 35:06)



Then what we are going to do, then already I have mentioned on that the next place what we are going to do? We are going to merge the nodes which are having similar level ok. That means that can be removed. So just we are going to look an example, this is an BDD we are constructing x, y, z and this is basically BDD also. So if you look in to this how we are going to start it?

First we are going start the terminals nodes, so in this particular case we are going to keep 0 to all 0 nodes and assign 1 to all one 1 nodes. This is the first rule labeling the terminal nodes 0 and 1. Then we are going to look for this particular case, say in this particular case let 0 is going to 0 nodes and 1 is going to 1 node.

Since this two are depends we get a new label and now we are coming to this nodes say this two are having the definite and it should get a new label. Secondly already we are having the z node. So we will compare these two nodes and a low of this particular z is same as low if this particular 1 and high of particular z, z is high of these one. So these nodes will get a same level over her because they are having same BDDs for 0 and 1.

And similarly when I come to this particular node z level by z again we will find that they are having in to different nodes 0 and 1. So it should get a define level. If I consider about this node and what will happen? The behavior is same, so it is going get this particular level.

So now first leveling the terminal node then we are coming to one level up and after leveling this particular node we go the next level. So since I am having 3 variables so total we are having 4 levels. Your distance what you call non terminal nodes, forty variable and one level is the terminal nodes. Similarly when we come to this particular node you just see that lo of this particular node is pointing to that level to lo nodes and hi of this particular node is 11 to this particular node to level 2.

So that means in low or your this node is same with the high of this node that means this is removable order redundant. So since this are same this particular node is going to get this particular node so it will be 11 to level 2. Now we come to this particular low then you will find that low is coming to level 1 and high is coming to level high. That means low and high is not giving the same representation, they are giving the different representation different sub BDDs function.

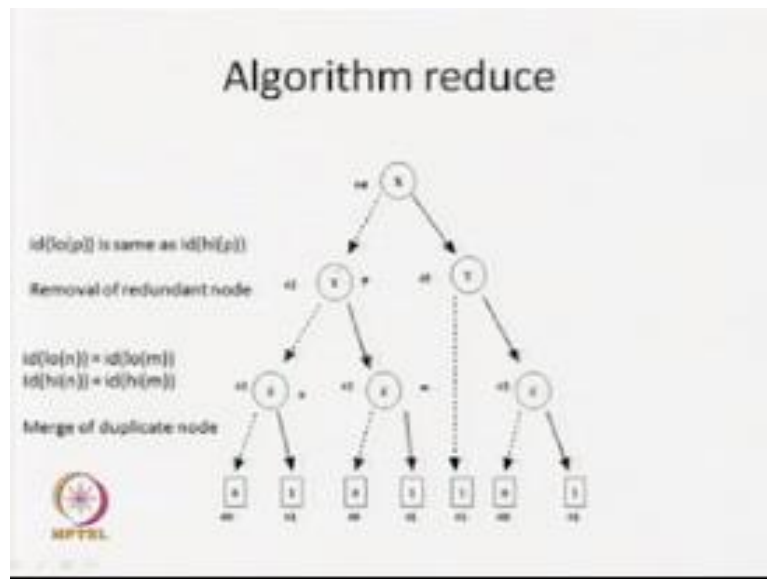
So it is bound to get a new level. Similarly again behavioral of this two nodes are not similar. Second third condition will also be, so it is going to get a new level and we are going to bound it by 3. So it assigning the level 3. Now when I am coming to these particular things then low of this particular node is 2 and high of this particular node is your 3.

So id of low is 2 and id of high is 3. That means they are having definite sub function. So it is going to get a new level, so you are starting from these particular terminal nodes and following a bottom of approach and coming to these particular nodes and we have level all this particular nodes.

Now in these particular nodes, now next one is your merging of all duplicates nodes. If it is id of low n and is equal to id of low m and id of high n equals to id of high m, so this the scenario that we are going margins, merging the duplicates nodes in this particular case is 3 nodes will be merged together.

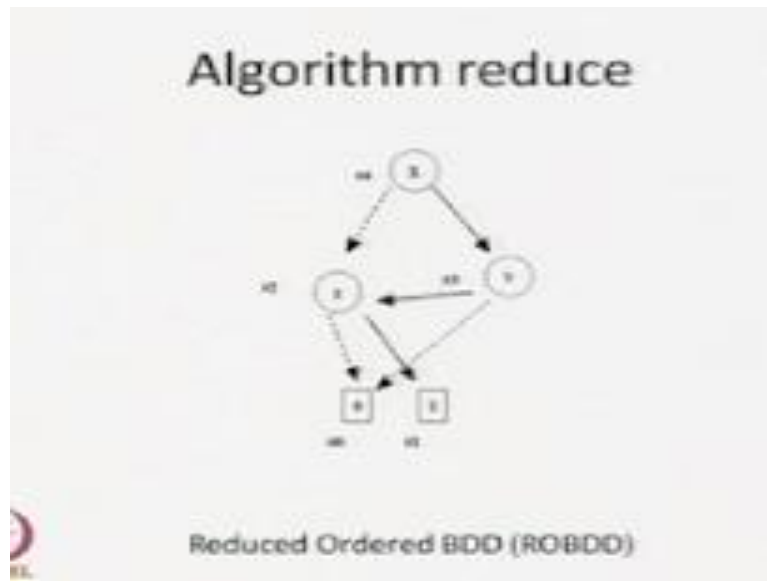
And secondly you are having another one removal of redundant node. So if id of lo(p) is same as id of h (p), so this is said we are saying this is the level to p and this is level n and m. So that is why I am saying that l and m so that can be merged and now since for this particular node p low and high, so we can remove this thing.

(Refer Slide Time: 39:25)



So basically these are the rules that we are going to apply, so in this particular case now we can merge this particular nodes having the same level and eventually we can going to get this particular reduce BDD.

(Refer Slide Time: 39:36)




This is the Reduce Ordered Binary Decision Diagram of the BDD that we will study. And with the help of this reduce algorithm, we can use and reduce the BDD and eventually we are going to get the Reduced Ordered Binary Decision Diagram. So that means we are having an algorithm called Reduce, where we are going to give an input BDD. So input is your Ordered BDD and output we are going to get is ROBDD Reduced Ordered Binary Decision Diagram.

And the variable or the ordering of the output BDD is the same as the variable ordering input. So now you just see that we can construct a BDD for given any Boolean function and after that after construction you can use this particular algorithm to get the Reduced Ordered Binary Decision Diagram.

(Refer Slide Time: 40:43)

Reduced Ordered BDDs (ROBDDs)

- The reduced OBDD, representing a given function f , is *unique*.
- That is to say, let B_1 and B_2 be two reduced OBDDs with *compatible variable ordering*. If B_1 and B_2 represent the same Boolean function, then they have identical structure.
- The order in which we applied the reductions does not matter.
- OBDDs have a canonical form, their unique ROBDDs.



Now this is some properties, you can see that of reduced Ordered Binary Decision Diagram (ROBDDs). So the reduced Ordered Binary Decision Diagram representing a given function f is unique, so if you are going to look at the particular function f and we are going to concentrate the reduced ordered binary decision diagram of that particular function.

Then this particular binary of Reduced Ordered Binary Decision Diagram will be unique with respect to that particular variable ordering because if you sense the variable ordering then size will vary, because already we have seen that means we are going to get different representation different BDD.

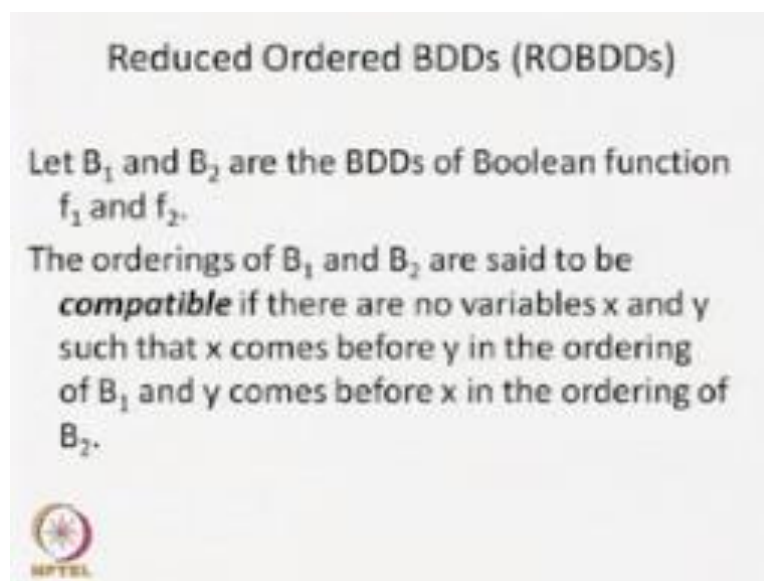
So with the particular variable ordering we are going to get a unique representation. Secondly we are having a notion of work notion of compatible variable ordering. If you consider two BDD b_1 and b_2 , if they are having the same variable ordering then we can say that they are having compatible variable order ok. And this two BDD say B_1 and B_2 if they are having a compatible variable ordering and if they are representing the same Boolean function.

Then you are going to get a unique representation if the particular variable ordering so if two BDD are having compatible variable ordering and secondly they are representing same Boolean function then this two will be identical.

So again, that is why you are saying that for a particular variable ordering we are going to get a unique BDD, ROBDD representation ok, that is why we are saying that ROBDD representation of any function is going to give us a canonical representation of that particular function and secondly, we have already seen that how to apply this particular function Reduced algorithm, we are going to get a reduced ordered.

On other hand you can apply particular reduction rule also and the ordering in which you are applying this particular reduction is immaterial we are going to get is reduced ordered binary decision diagram. So this is the unique property that we are having, that we are going to get a unique representation of the Boolean function with respect of particular variable ordering that is why we said that ROBDD keeps us the canonical representation Boolean function ok, this is the main piece of ROBDD and we can use this particular BDD.

(Refer Slide Time: 43:19)

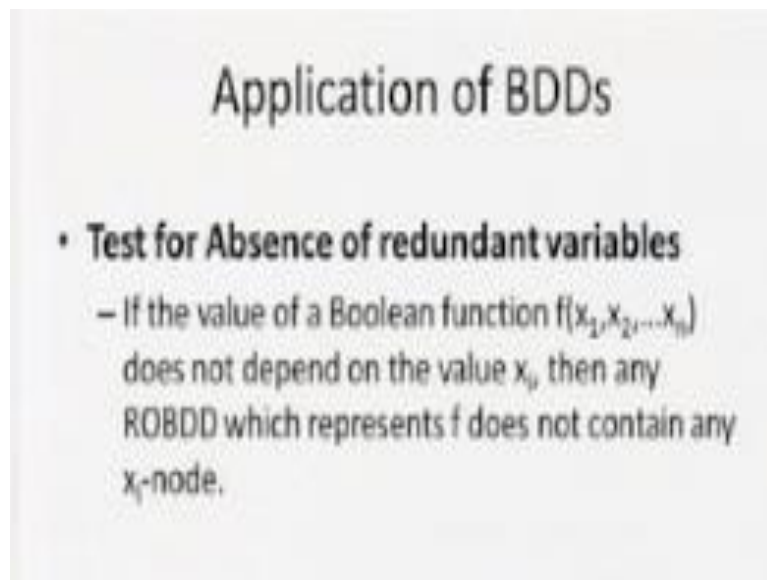


Now similarly Reduced Ordered Binary Decision Diagram saying that, if we are having two BDDs B_1 and B_2 . And this B_1 and B_2 are representing Boolean function F_1 and F_2 . So we are having one Boolean function F_1 and we are saying that this is a BDD representation B_1 and we are having other Boolean function B_2 and BDD representation B_2 .

The orderings of B1 and B2 are said to be compatible, if there are no variables x and y , such that x comes before y in the ordering of B1 and y comes before x in the ordering of B2. So we do not have such type of scenario than we can say this is a compatible variable ordering. That means in B1 if say x is coming before y then in B2 also x must come before y .

So y comes before x in B2 then we are not going to going say that this two are having compatible variable ordering. So now we can look for any variable ordering and look for two BDDs. We will say that these two BDDs will have compatible variable ordering if they are having a same variable ordering ok.

(Refer Slide Time: 44:39)



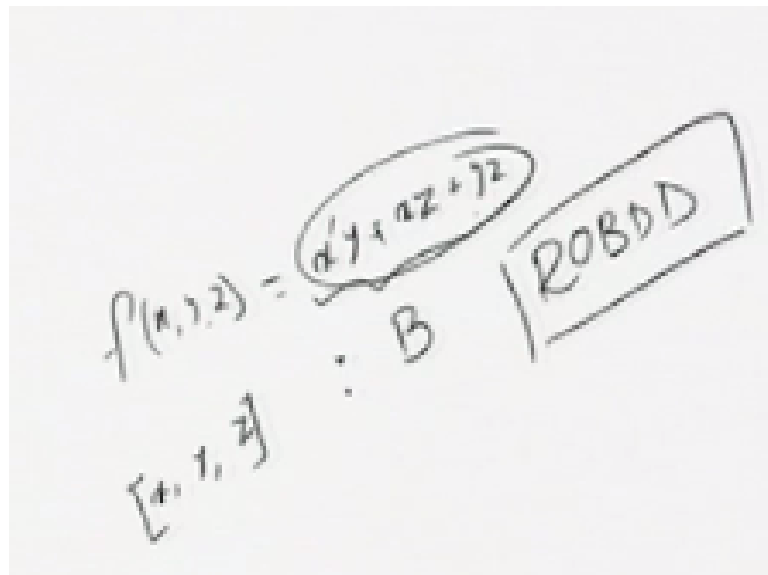
Now we can have any Boolean function say i am having $f(x,y,z)=x'y +xz +yz$. Now if we are having such type of function, then I always keep some 2 variable orders say I am going to choose a variable ordering x, y, z and I can represent this particular function with the help of BDD b and if we apply a reduction rule or reduce algorithmic to this particular B then we are going to a ROBDD.

So once we are getting this ROBDD then what happens? It is basically nothing but we are representing this particular Boolean function ok. We are representing this Boolean function with the help of ROBDD.

Now once we have this particular ROBDD representation of this Boolean function. Now what we can do? Already we have mentioned that in most of the cases, we are going to get the compact representation of Boolean function and after that we do some operation and we can do some manipulation on those particular BDDs also. So far any function we are going to have the BDD representation.

Now this BDD representation are going to helps us to take some decision very quickly, so what are that particulars decision? So first one I am talking about Test for Absence of redundant variables, so say if I am giving one particular function can you say that this functions is redundant of this is independent of some variable say if I am giving this particular function $x'y + xz + yz$.

(Refer Slide Time: 46:24)



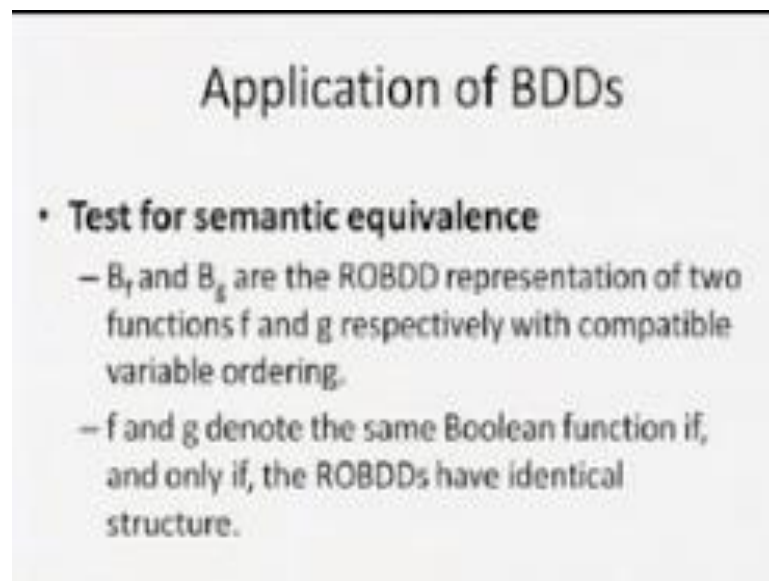
Can you say, tell me whether it is independent of some variable? This is small function we can see it and say that it may be independent or may not be independent but if we are having a BDD representation then by looking in to the structure of BDD easily we can say that whether it is independent of some variable or not. So how we can say that if the value of a Boolean function say $f(x_1, x_2, \dots, x_n)$ does not depend on the particular variable say x_i then any ROBDD which represents f does not contain any x_i no ok.

So if I am having a BDD representation of a function say x is coming something like that, say this is y , this is 0 and this is 1 . So this a function $f(x, y, z)$ if I look at the BDD. In this particular BDD we have found that this particular BDD is not having this particular variable

z. So you can very well say that this function is independent of this function of this particular variable z.

So test for absence of redundant variable is easily checked by looking in the structure of this particular ROBDD because ROBDD is going to give us a unique representation of a particular variable ordering. So all particular variables is not appearing in this particular structure we considered it is independent of this particular variable.

(Refer Slide Time: 48:00)



Application of BDDs

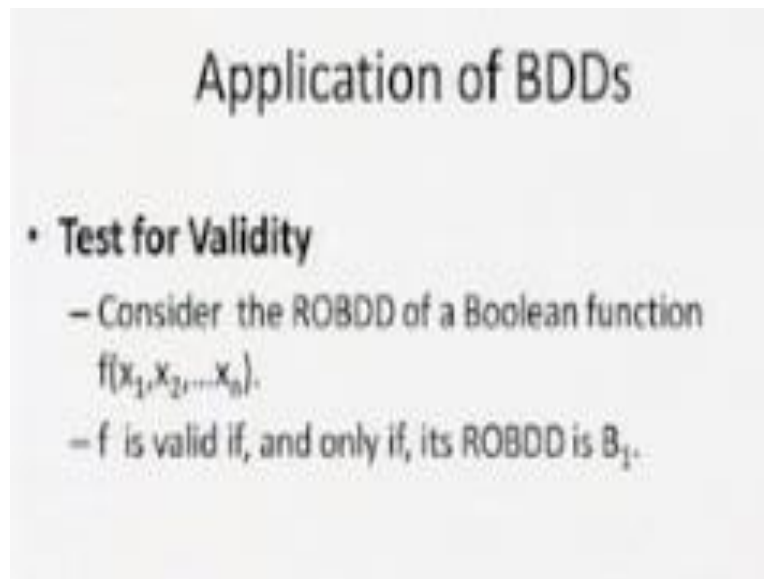
- **Test for semantic equivalence**
 - B_f and B_g are the ROBDD representation of two functions f and g respectively with compatible variable ordering.
 - f and g denote the same Boolean function if, and only if, the ROBDDs have identical structure.

Similarly we can say that test for semantic equivalence because with respect to particular variable ordering it is going to give me a unique BDD representation. Now if we are giving two functions say $F1$ and $F2$ and I am asking whether $F1$ is equivalent to $F2$ to 1 if they representing in some Boolean function. Then what will happen we are going to do some Boolean manipulation of this particular $F1$ and we try to check whether we can get $F2$ to $F1$.

Now if we construct now BDD $B1$ for $F1$ and BDD for $F2$ with a particular variable ordering that means they should have a compatible variable ordering. Now once we construct this particular BDD for function $F1$ and $F2$ and eventually if you find that both are having a identical structure.

Then we can say that both F1 and F2 are representing the Boolean function. That means F1 and F2 are equivalent. So semantic equivalent can be checked very easily, ones we construct or ones we can have the BDD for this two function. If there are having the identical BDD representations, with respect to compatible variable ordering then we can say that this two functions are equivalent.

(Refer Slide Time: 49:10)



Similarly we can say that the test for validity. We know when we said that the particular variable Boolean function is valid or not. For all valuation of this particular variable it if evaluates to 1 then we can say that this is a valid function. So in this particular case now if we construct a BDD and if your BDD is your B_1 then you can always say that it is a valid function.

Basically we can say that BDD is B_1 if it is only terminal nodes then we are going to say that this is your BDD B_1 . So if I am giving any function f now construct the ROBDD and with respect to particular variable ordering and if ROBDD is happen to be here the BDD b_1 then we can say that this is a valid function.

So you just consider, if I am giving a long expression to check whether it is valid or not but if I am having a BDD presentation and BDD presentation is B_1 then we can say that this is a valid function.

(Refer Slide Time: 50:19)

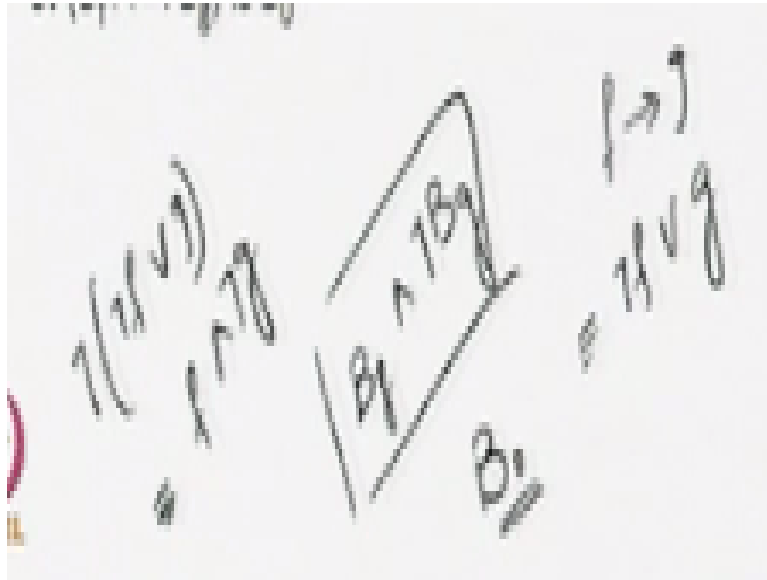
Application of BDDs

- **Test for Implication ($f \rightarrow g$)**
 - We can test whether f implies g by computing the ROBDD of $(B_f \wedge \neg B_g)$
 - f implies g if, and only if, the resultant ROBDD of $(B_f \wedge \neg B_g)$ is B_0

Similarly test for implication, whether f implies g or not. So f implies this is something like that f implies g is equivalent not of f or 0. So in this particular case what I am going to do? I am going to have a take the negation of these things. Negation of these things, so negation of f implies. If I take negation of that means what happens? I am going to get a not of f or g applying the demerges g what am I going to get? This is your f and g .

That means I am going to construct a BDD, bf and not of sorry this is not of bg . And if this particular BDD is your b is 0 because I am taking the negation you just see that, that means negation is a conduction. That means if negation is a conduction that processes the variable. So in that particular case if this is equal to your b_0 now you can say that f implies g .

(Refer Slide Time: 51:15)



So this is also but how to do this operation we are going to check. We will discuss this things in our next class.

(Refer Slide Time: 51:23)

Application of BDDs

- **Test for Satisfiability**
 - A Boolean function $f(x_1, x_2, \dots, x_n)$ is satisfiable if it computes 1 for at least one assignment of 0 and 1 values to its variables.
 - The function f is satisfiable if, and only if, its ROBDD is not B_0 .

Similarly we can say that test for satisfiability. And we say that a function is satisfiable, if for at least one valuation it gives the value as 1 then we say that it is satisfied. That means if I have a function $f(x, y, z)$ then at least for one valuation it should give me the functional value as 1. Then in the particular case we are going to say that this function is satisfiable.

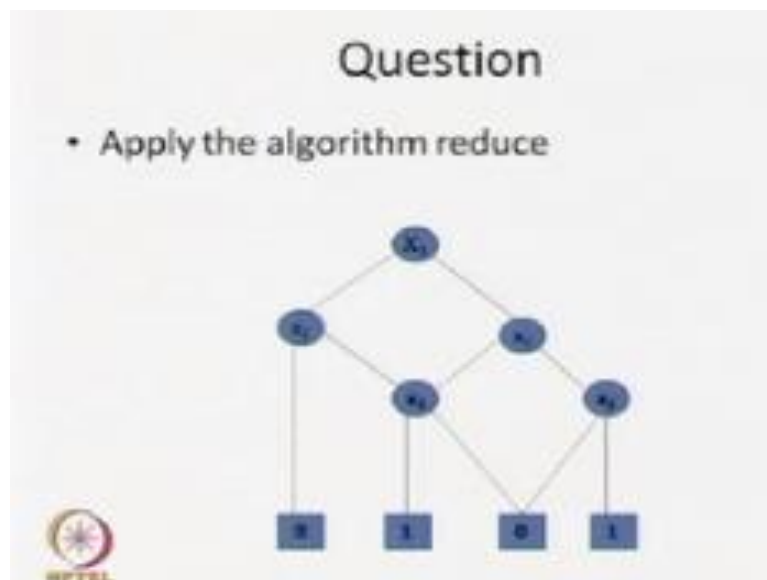
Now to check it, that this function is satisfiable? Now how to check it, now you have look for all valuation or if you are having a two table representation and we are having any one np as 1, then we can say that it is satisfiable. So if I am having a BDD representation this particular function and if this BDD representation is not equal to B_0 , if it is not B_0 then we can say that this is a satisfiable function.

That means, if it is B_0 for all valuation it will go B_0 that means for all valuation we are going to get 0. So if the result in BDD is not 0 or not equal to B_0 I can say that it is satisfiable because for some valuation it is going to get the terminal 1. So by looking into the structure, looking into the result and ROBDD because ROBDD is your unique representation of function and which is a conical representation function if the ROBDD is not equal to B_0 .

Then we can say that that function is satisfiable. you just see that now if you are having boolean function and if we can represent this boolean function with the help of ROBDD with respect of particular variable.

Then some decision can be taken very quickly like whether it is valid or not, whether it is satisfiable or not, whether this function is independent or not of some variable or not, whether do the function are equalivalent or not. So such kind of decision can be taken very quickly. So this are the advantage of using ROBDD.

(Refer Slide Time: 53:22)



Ok, now one question I am saying that apply the algorithm reduce to reduce this particular BDD. So I am giving an OBDD Ordered Binary Decision Diagram and you have to apply the algorithm to this. So it is having three variable x_1 , x_2 , x_3 and the ordering is your x_1 , x_2 , and x_3 . So this is the order we are having and this is the given BDD.

Now apply the algorithm to reduce this particular BDD. So in this particular case what will happen? I am going to label this nodes. In this particular case what will happen all 0 nodes will be labeled with 0, and all 1 node with be labeled with 1. Now we are labeled this particular terminal nodes and will go up. In this particular case when I come this particular label so it is one node is coming to 1 and 0 node is coming to 0.

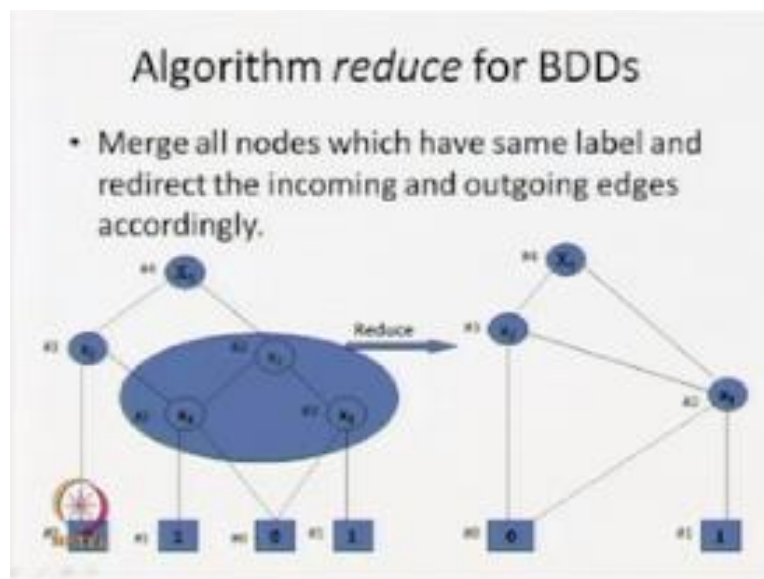
That is your sub BDD that is not same so we are going to have a ordering as 2. Similarly this is also a X_3 0 is going to 0 and 1 is going to 1. So this is not same with 0 and 1. Secondly this two are having the same level x_2 same variable x_3 and x_3 . So sub BDD is 0 for both the 0 going to 1 terminals for both the one if it is going to 0 terminals.

So it will also go to get a same label with this particular x so both are labeled with the x_2 . Similarly if I am coming to particular node then 0 is going to level 0 and 1 is going to level 1. That means there are sub BDD it is going to get a different level so I am going to label it with 3. When I come to this particular level with 0 we are going to this particular level 2 with 1 also we are coming to particular level 2.

So in this particular case we are going to get the same level with this two node which we are going to get x_2 . Now when I come to this particular node 0 is coming to your 3 and 1 is coming to your level. So this are different, so in this particular it is going to get a new level and we are going to say that we are going to give a level 2. Now this is slow we are starting from the terminal nodes going in the bottom of node and coming to the terminal nodes this is root nodes and all leveling is over.

So in the particular case now next page is your merging. So this three nodes are having the same level so we can merge together and ultimately we are going to get this particular ROBDD.

(Refer Slide Time: 55:59)



So merge all the nodes which have the same levels. So this three are the same level we are going to reduce it to 1 nodes and accordingly we are redirecting the input and output. So ultimately we are going to get this particular reduced BDD.

(Refer Slide Time: 56:23)

Question

- Consider the following function
– $f(x,y,z) = xz + xz' + x'y$
Is it independent of any variables.

So another question consider the following Boolean function is $f(x,y,z)=xz+xz'+xy$. It is independent of any variables. So this a function x , y and z and it is having involved in any particular x , y , and z . now you have to see whether it is independent of any variable or not. I think you can simply construct this particular BDD, this is your x if I am giving the ordering.

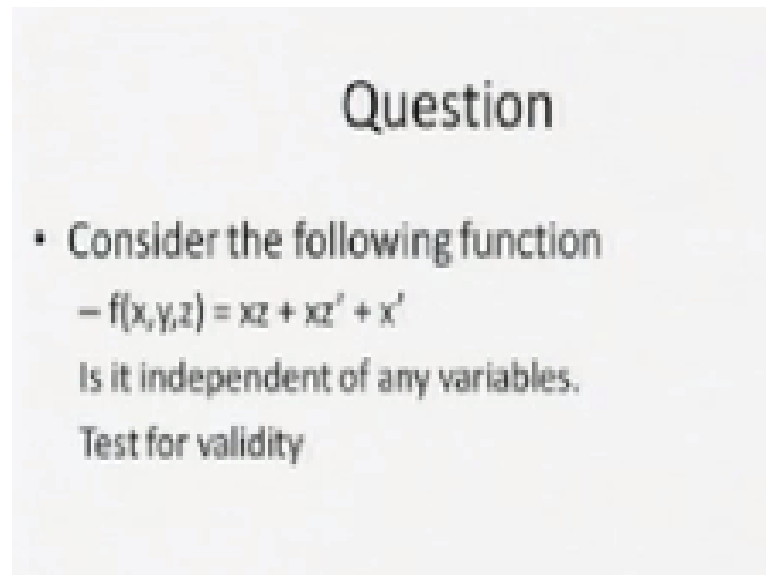
So x is 0 I have to take the decision on y if $x=1$, then I have to take decision on z ok. Now if y is 0 then this is 0, and $y=1$ then I am having 1. Similarly if it is x then z if $z=1$ then what will happen? $X=1$, so in this particular case my function is 1 and x is 0 1 and $z=0$ then my function is 0. $X=1$ and $z=0$ oh sorry, this is not so in both the cases it will be 1.

$X=1$ $z=1$ it is going to give 1, $x=1$ $z=0$ it is going to give us 1. So in this particular case I am having the circular x then this so I can remove this one, so in this particular case eventually this is come to this particular point so this is the result in BDD and in this particular result in BDD z is not appearing. So we can say that this particular function is independent.

So this is small function I can we can do the Boolean manipulation but what I am going to get $X(z+z')+x'yx+x'y$ that means what I am going to get $xz+=1$ so this $x+z'y$ which, so we can say that this function is independent of this variable z , But if I give a bigger expression such type of manipulation may not be that simple.

So if you have that BDD representation from this BDD presentation we can say that this function is independent of that variable.

(Refer Slide Time: 58:53)



Question

- Consider the following function
– $f(x,y,z) = xz + xz' + x'$
Is it independent of any variables.
Test for validity

So look into this particular function. It is a very small function and simple function. $xz + xz' + x'$ it is independent of any variables, is it test for validity? When I consider this is valid the result is BDD is your 0. To construct it again I can say that, I am going to take decision of your, this is simple. From this equation itself I can say that it is y is not there so it is independent of y ok.

So I am going to take x, x may take a value 0 and x may take value as 1. So when I am going to construct it when x part is equal to 0 then my value is 1 ok. Now x is =1 to take decision on this particular z, so when z is 1 then xz is going to give me 1 and when z is 0 $x1z0$ is going to give me 0. So in this particular case I am going to have this.

Now eventually, what will happen you just see that now this is a redundant test, so I can remove it what I am going to get x sorry this is terminal. So I am removing it so 1 is will come over here and you see x is here this is redundant x is equal to 0 is 1 $x=1$ so what happens from here I can say that whether I am going to get this particular BDD also.

What is the BDD I am getting this is the BDD1 that means for all valuation, function is going to give me. Since my result in BDD is B1 so, what I can say that this is a valid function.

Secondly by looking into this function I have said that this is independent y. Now after looking into this particular BDD what can I say? It is independent of X, Y and Z. That means this function is independent of all these particular three variables. So by looking into this particular function we can take decision very easily.

So this is a small function I am giving but if I am giving you a bigger expression by inspecting this function it will be difficult to say it is valid but it is independent of some variables, but when we get ROBDD by looking into the structure of this particular ROBDD we can take many more decisions quickly ok. With this I end up my lecture today. Thank you.

Centre for Educational Technology

IIT Guwahati

Production

Head CET

Prof. Sunil Khijwania

CET Production Team

Bikash Jyoti Nath

CS Bhaskar Bora

Dibyajyoti Lahkar

Kallal Barua

Kaushik Kr. Sarma

Queen Barman

Rekha Hazarika

CET Administrative Team

Susanta Sarma

Swapan Debnath

