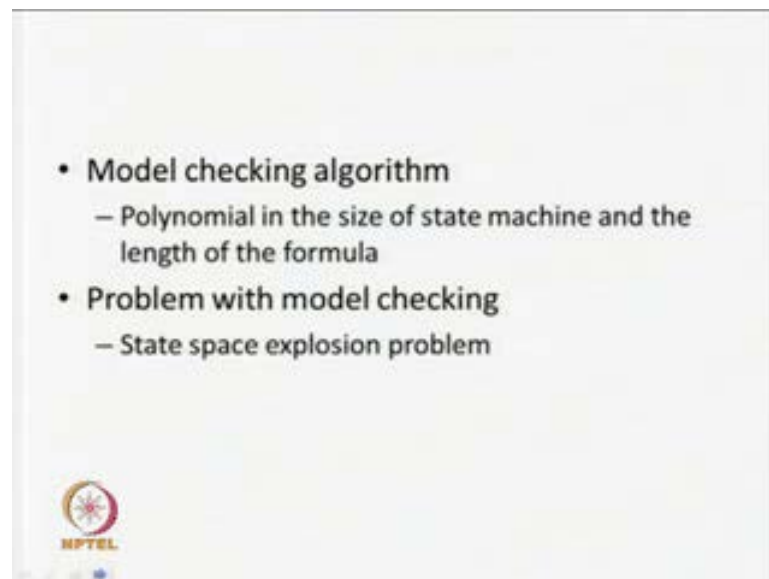


**Design Verification and Test of Digital VLSI Design**  
**Prof. Dr. Santosh Biswas**  
**Prof. Dr. Jatindra Kumar Deka**  
**Indian Institute of Technology, Guwahati**

**Module - 6**  
**Binary Decision Diagram**  
**Lecture - 1**  
**Binary Decision Diagram: Introduction and Construction**

Today I will start a new topic call binary decision diagram or in short BDD. What we have seen till now that we have seen about a verification of a system for that we need to have a model of the system and we need a specification language. So, we have talking about CTL computational tree logic to given of specification and we have taken our model as a finite stack machine. Now after that we have discuss about the CTL model checking algorithm. Now, if we give a model of our system and if we give a specification then we apply our model checking algorithm to check why that the given specification is true in our system or not.

(Refer Slide Time: 01:36)

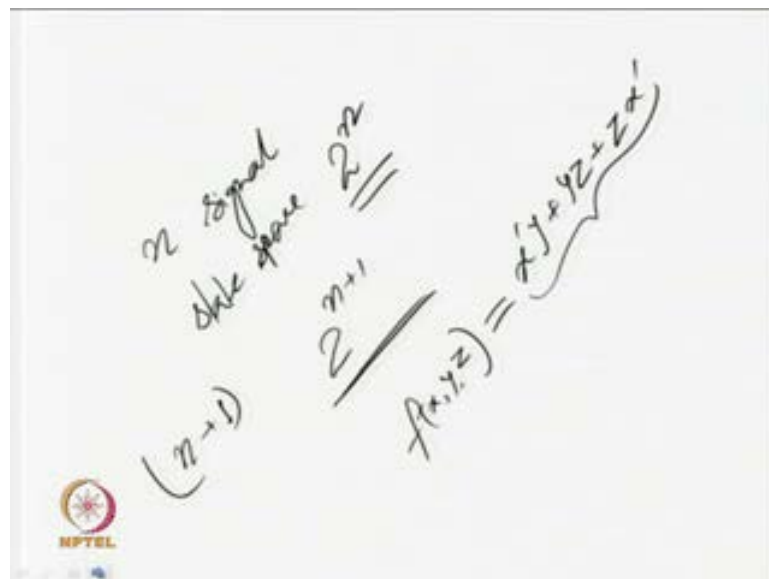


So, if you look into this particular model checking algorithm you will find that this is somewhat similar to a craft over slave algorithm. So, we try for the Intergraph and will check with a it is true will some state on node, and ultimately the model checking algorithm return the state set of states where the formula is true. Again when we have discuss this particular model checking algorithm we founded, we are having a

polynomial terminal algorithm to, to the CTL model checking which is polynomial in the size of our keep get structure this is basically polynomial in the size of does graph state space, that we have giving into it and also it is in the polynomial in the size of our given formula, what a length of our given formula. So, we have seen that we are having a polynomial them algorithm and we are having a automatic metrology, automatic algorithm, we can develop this algorithm, we can improvement this things and automatically we can check with the given formula is true in a given model or not.

Manual interfusion is not (( )) so we are happy with them, but we have one more problem with this particular model checking algorithm. This is known as your state space explosion problem; what is this state space explosion problem basically you just see that, if we are walking with a system where we are having encounter fairy about. So, or n control signals that means, those n control signals can be dataries over atomic proposition so number of define combination will be your 2 to the power n.

(Refer Slide Time: 02:52)



So, for n signals state space is your 2 to the powder n, because we are having 2 to the power differ n 2 to the powder and define combinations. Now say while we are designing it phosphination we have studied with n signals that we have formed that it cannot be design properly we need one more signals. So, will in cooperator one more signal in our design and eventually the total number of signal will become n plus one and states space will become knot and 2 to the power n plus 1.

So, in this particular you can see that the state space is in explosions that means, it explosions to the number of signals that we have. So, for a smaller system it is fine but for a bigger system it is going to give as problem. So, that why you are now looking into some at a way to represent our system. So, fault depth BDD is one solid circuit with the help of BDD or we can taken this particular state space explosions problem.

Now that is why, we have going to talk about BDDs. What is BDD? It is a binary decision diagram which is a data structure to implement or to stow any Boolean function. So, if say if we are having a Boolean function of t variable x y z then function x y z can be represented with some Boolean explosions x bar y plus y z plus z x y. So, if we are having such take a Boolean explosion, then this Boolean explosion can be stow or represented with the help of binary decision diagram.

So, we have going to look into this particular binary decision diagram in this particular module. And let on we are going to see, how that finite state machine or the state transition graphs will be implemented or represented with the help of BDD and will see how the model checking algorithm can be in cooperator of BDDs. One we use BDDs for our model checking algorithm then we say this is your symbolic model checking. So, eventually we will go to symbolic model checking algorithm where the system will be represented with the help of BDD binary decision diagram.


(Refer Slide Time: 05:15)

**Binary Decision Diagrams (BDD)**

- Based on recursive Shannon expansion

$$f = x f_x + x' f_{x'}$$

- Compact data structure for Boolean logic
  - can represents sets of objects (states) encoded as Boolean functions
- Canonical representation
  - Reduced ordered BDDs (ROBDD) are canonical

 NPTEL

Now what is your BDD? So I have already mention that BDD is use to represent and in Boolean function. So, it is basically BDD is based on recursive Shannon expansion we know that any Boolean expansion can be expansion as a Shannon expansion. So, this Shannon expansion is given it like that,  $f$  is equal to  $x \cdot f_x$  plus  $\bar{x} \cdot f_{\bar{x}}$ . So, what does it means, so  $x$  is a importable so expanding in this particular function of  $x$  and we are having the evaluation value of  $x$  and we have going to treat  $x$  equal to 1 plus  $\bar{x}$  that means we have going to treat the variable  $\bar{x}$  and the functional value when we have going to treat this value variable  $x$  equal to 0. So, with the help of this Shannon expansion we can represent and the Boolean function and the BDD is based on this particular Shannon expansion. What is BDD; it is compact data structure for Boolean logic.

So eventually will find that it is compact to a optic presenting our Boolean logic or Boolean function. On the other have you can say that we can represent our states space also with the help of BDDs because states space can be represented with the help of a combination of some Boolean function and one more advantage of BDD is that it is a canonical representation. So, it is reduce order BDDs are canonical in form that means, we have going to get a unit representation of a Boolean function if we use ROBDD reduce order binary decision diagram. First will talk about binary decision diagram then will go to order binary decision diagram and finally, will come to reduce order binary decision diagram. When we talk about reduce order binary decision diagram you have going to get a unit representation of a Boolean function. That is why, will say that this is the canonical representation of a Boolean function.

So, now I am going to give one example of this particular Shannon expansion. So, you just see that I am going to talk about this particular function  $f$  is equal to  $a \cdot c$  plus  $b \cdot \bar{c}$  ok. That means  $f$  is a function of  $t$  variable  $a \cdot c$  plus  $b \cdot \bar{c}$  and already I have mention that the Shannon expansion is your  $f$  is equal to  $x \cdot f_x$  plus  $\bar{x} \cdot f_{\bar{x}}$  that means, this  $f_x$  says that the value of this given function when  $x$  equal to 1. So, this is basically we have going to have this function  $f$  so we have going to say  $f_a$  will be the functional function 1 we have going to treat  $x$  equal to a equal to 1.

(Refer Slide Time: 07:19)

### Shannon Expansion

$$f = ac + bc$$

$$f_{a'} = f(a=0) = bc$$

$$f_a = f(a=1) = c + bc$$

$$f = af_a + a'f_{a'}$$

$$= a(c+bc) + a'(bc)$$

$$f = xf_x + x'f_{x'}$$

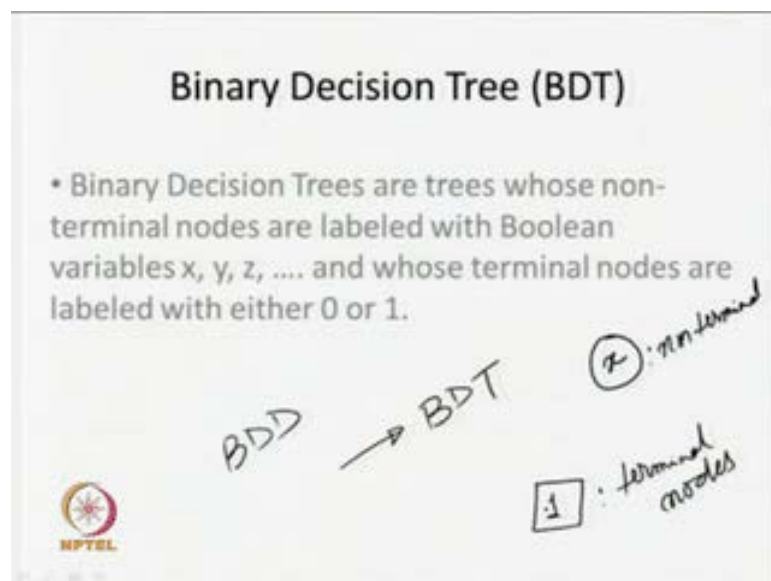
So, if  $a$  will we have going to get as your  $c$  plus  $b$   $c$ , because when we treat  $x$   $a$  equal to 1 then one dot  $c$  will be equal to  $c$  plus  $b$   $c$ . Similarly,  $f$   $a$  what is the function when we are thing that value of  $a$  is equal to 0 so  $f$   $a$  bar is equal to  $f$  when  $a$  equal to 0 so when we put  $a$  equal to 0 then this particular trance become 0 0 plus  $b$   $c$  so eventually you can going to get  $b$   $c$ .

So,  $f$  of  $a$  is equal to  $c$  plus  $b$   $c$   $f$  of  $a$  by is your  $b$   $c$ . so that is why this particular function can be represented is your  $f$  of  $a$  into  $b$   $f$   $a$  plus  $a$  this into  $f$  of  $a$  this so this is nothing but  $a$  into  $f$  of is your  $c$  plus  $b$   $c$  and  $f$  of is your  $b$   $c$  so  $a$   $a$  bar dot  $b$   $c$ . Now basically what we can see that, if we have going to the simply thing that we are having this three variable  $a$   $b$   $c$ . Initially we have going to look for the valuation of  $a$ ,  $a$  can either take 0 or 1 just I am going to represent it something like that say this is the variable  $a$ . So, this maybe your  $a$  equal to 0 and this is your  $a$  equal to 1.

So, in this particular case already I have the valuation of  $a$ ,  $a$  either  $a$  will be equal to 0 or  $a$  will be equal to 1, so what we need to evolver. This is the function based on this variable  $b$  and  $c$ , because already we have taken the decision on  $a$ . Now this particular function that when  $a$  equal to 0 you see that the function will be your  $b$   $c$  that means, I am going to prop about the function  $b$   $c$  over here. So, I can say that this is your  $g$  and when  $f$  is equal to  $s$  and I am going to say that  $c$  plus  $b$   $c$  is the value. So, I can say that  $c$  plus  $b$   $c$  is the value and I say that this is the function  $h$ .

Now my next (( )) is to evaluate this function  $g$  and  $h$  again will use this particular Shannon expansion say if you consider that variable  $b$ . Then what will happen I will take as a  $b$  is equal to 0 and  $b$  equal to 1 similarly, for  $h$  also we have going to talk about  $b$  is equal to 0 and  $b$  equal to 1 and eventually what function will remains it will on the value of variable  $c$ . So, in this way recursively we have going to get Shannon's expansion and we have going to get a some sort of your decision tree or decision diagram and it will be represent with this particular form.

(Refer Slide Time: 10:50)

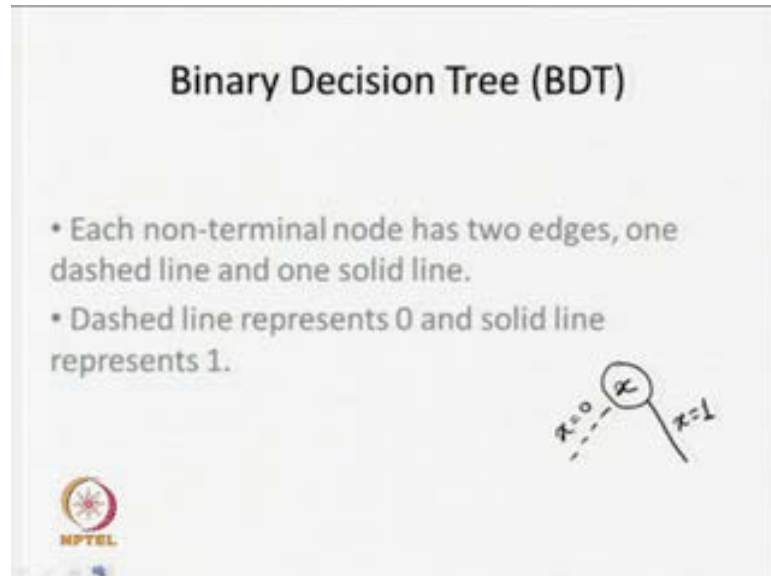


Now we have going to talk about these particular BDD so binary decision diagram but before going to binary decision diagram, I am going to talk about BDT binary decision tree. So, it is a tree so we have saying that binary decision trees are trees whose non terminal nodes are leveled with Boolean variables. Say if you having a function of  $x y z$  then non terminal nodes will be represented by those particular Boolean variables and the terminal nodes will be represented by the Boolean values or Boolean constant either 0 or 1. So, one we have going to draw, draw BDT or BDD we use this particular cycles to represent non terminal nodes.

And we use this particular square box to represent terminal nodes ok. So, in case of non terminal nodes it will be labeled with that variables of that particular (( )) say if  $x$  is a variable say I will labeled this particular non terminal nodes with  $x$  and if it is a terminal

nodes either Boolean function value will be 0 or 1. So, I can say that if I present write 1 over here it says the, this value of this particular terminal node is 1.

(Refer Slide Time: 12:30)




So, we have going to draw a tree in the tree we are having terminal nodes and the non terminal nodes. The level of non terminal nodes are the variables of the function and the label of our terminal nodes or Boolean constant either 0 of 1 which will be the functional value of a given function. Now in BDD's of BDT each non terminal nodes are having two edges ok. So, if decision non terminal node x it is having two edges one is will be represented by your dash line and second one will be represented by the solid line; that this line says that, this is representing the valuation of x equal to 0 and solid line says that, this is the valuation of x with value x equal to 1. So, I know that we are having only two possible either x will be 0 or x will be 1. So, when I am going to treat the valuation of a particular variable x then it is having two outgoing as that, this is basically says it is the value of a x equal to 0 and x equal to 1 will represented by this particular solid line. So, I have treat non terminal nodes are having two outgoing as s.

(Refer Slide Time: 13:29)

### Binary Decision Tree

a	b	c	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Truth Table → BDT



Now that is why you can say that, so in of BDT we can something write something like that so this is say variable x this is variable y this is variable z. So, when variable x equal to 0, we are having a valuation of x 0 then we are going to look for what will be the value of y. If value are x equal to 1, then we have going to say what is the value of y over here like that we are going to construct the complete tree it is the tree ok. Now I am going to give an example.


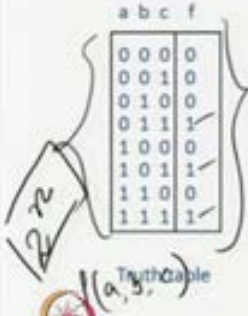
(Refer Slide Time: 14:08)

### Binary Decision Tree

a	b	c	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

$f = ac + bc$

Truth Table → BDT



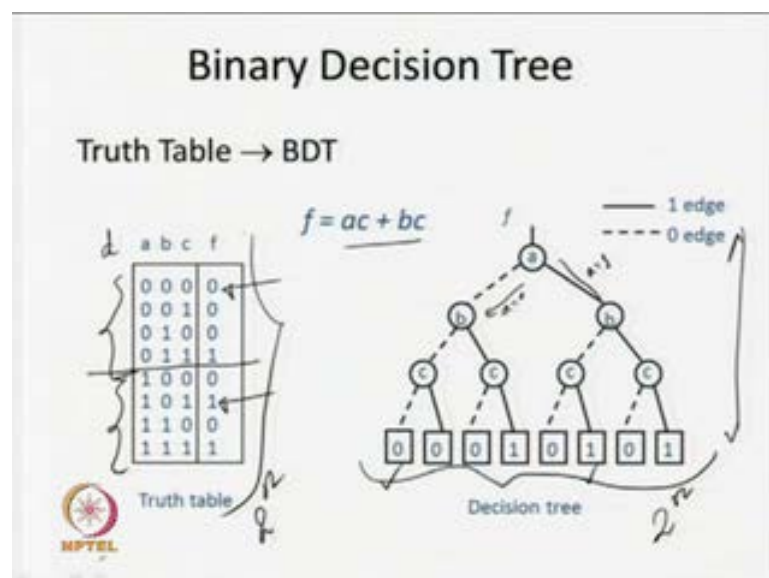


Just say that we have going to take a function  $f$  which involve three variables so  $f$  is a function of these three variable  $a, b, c$ . All of you know that the Boolean function can be represented very well with the help of truth table. So, I can have this particular truth table representation of my Boolean function  $f$   $a, b, c$ . So, what you know it is that, it says that when  $a$  is equal to 0  $b$  is equal to 0  $c$  is equal to 0 then my functional value is 0.

Similarly, if I say that if my  $a$  is equal to 1  $b$  is equal to 0 and  $c$  is equal to 1 then my functional value is 1. So, in truth table we have going to represent all possible combination of these tree variables and you know that what will be a size of this particular truth table. It is 2 to the power  $n$  we are having 2 to the power  $n$   $n$  trees in this particular truth table if  $n$  is the number of variable. So, here I am having 3 variables only so I am having 2 to the power 3 which is 8  $n$  3 so these are the 8 possible combination that we can have with respect to  $a, b$  and  $c$ , because (( )) 0 of 1 similarly,  $b$  and  $c$ .

Now this particular function I am representing with the help of your truth tables. Now we can construct the BDD for this particular function BDT binary decision tree for this particular function looking into this particular truth tables. So, if you look into it eventually we have going to get there it is having these 3 terms, so if you look into it then eventually what will happen 3 in terms will be there and when I am going to reduce it then eventually I am going to get this particular function  $f$  is equal to  $a c$  plus  $b c$ .

(Refer Slide Time: 15:56).



Now what we are going to do, now we are going to get the BDT for this particular function so this is the truth table I am having this is the function  $f$ . Now I am having 3 variables  $a$ ,  $b$  and  $c$  I have to see one  $a$  is equal to 0 and  $a$  equal 1 so when  $a$  is equal to 0 then I am going to look for this particular point because  $a$  is equal to 0 now  $b$  and  $c$  can have either 0 and 1. When  $a$  equal to 1 then I will having these particular part of the truth table. So, here I am representing one non terminal nodes labeled with  $a$  that means, I am going to take a decision on  $a$  when  $a$  is equal to 0. I will follow this particular dash line when  $a$  is equal to 1 then I will follow this particular solid line.

Now when I am taking a decision on  $a$  either coming to these part or these part. So, remaining portion will depends on  $b$  and  $c$ 's. Now again when I will come to  $b$  then  $b$  can be either 0 and 1, similarly  $c$  can be either 0 and 1 now eventually I am getting  $c$  so  $c$  will be either 0 and 1 it is having only 3 variables so I have already taken the final decision. Now, I am going to represent those terminal nodes, which is talked about these particular functional values.

Now you just able to one thing say when  $a$  is equal to 0,  $b$  is equal to 0,  $c$  is equal to 0 then my functional value is 0. So, that is why this terminal node is 0. So, when I say that  $a$  equal to 1,  $b$  is equal to 0,  $c$  equal to 1 then my functional value is 1, because it says that when  $a$  equal to 1,  $b$  equal to 0,  $c$  equal to 1 then this is the functional value 1. So, I am putting this 1 in this particular polynomial node.

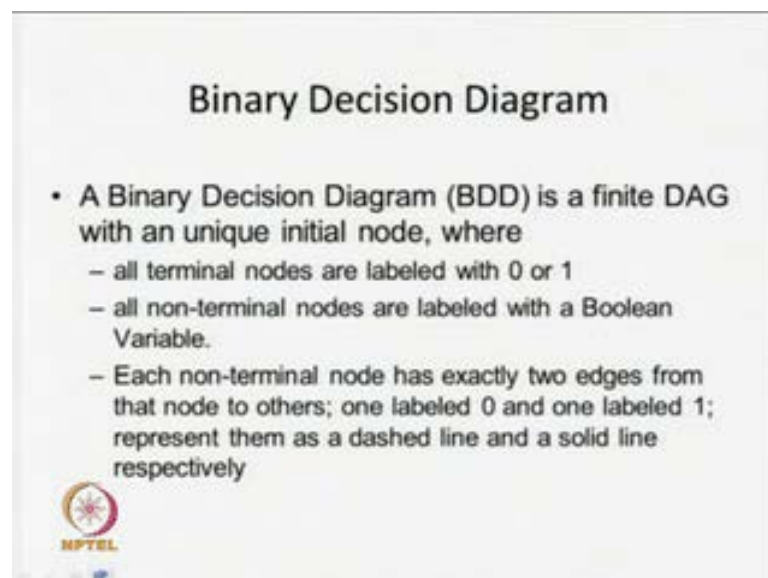
So you just see that by reversing all those particular edges we can get the functional value of this particular Boolean function. So, that is why I can say that from truth table I can state your construct these particular BDT ok. So, this is the function  $f$  is equal to  $a + b + c$  if you reduce this particular representation. We have going to get this one and this is the decision tree that we are getting over here.

Now you just see that, you are having a truth table I am getting a BDT binary decision tree. What advantage we are getting by representing this particular truth table with the help of BDT? If you see or if you look into it at such we are not getting any advantage, because you see that when I am representing it with the help of truth tables then what happen, I am getting total  $2^n$  if I am having  $n$  number of variables. Now when I am drawing this particular BDT binary decision tree again you will find that in node or terminal node we are having  $2^n$  define terminal nodes. Because

this is going to give the functional values for different combinations and height of this particular tree will be  $n$  basically level and because you are having  $n$  variable.


So again the way you are having the expansion grow of in truth table, because if I increase by one more variable said  $d$  then what will happen, we are going to get 16 different entries  $2$  to the power  $6$ . For there also this BDT the label of this BDT will be increase by 1 and I have to take decision on the and eventually we have going to get 16 different terminal nodes. So,  $x$  as if you look into the binary decision tree and your truth table we are not getting any advantage. In both the cases we are having expansion grow up.

(Refer Slide Time: 19:42)



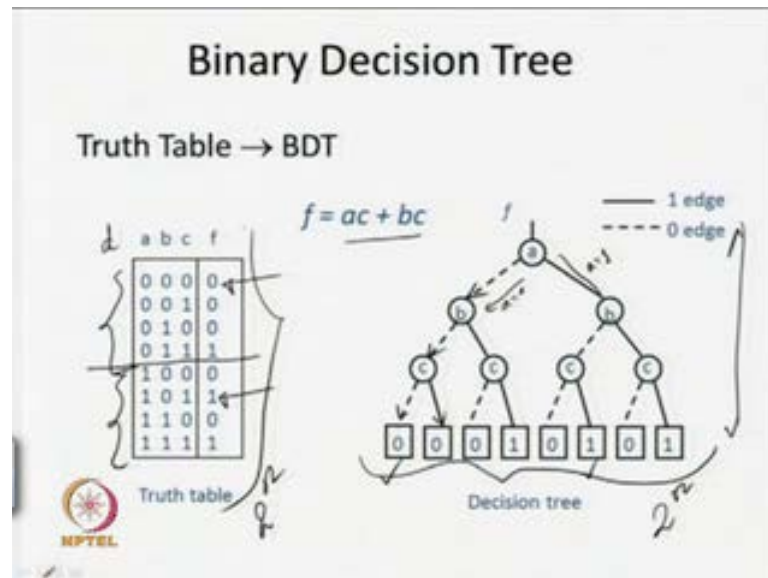
**Binary Decision Diagram**

- A Binary Decision Diagram (BDD) is a finite DAG with an unique initial node, where
  - all terminal nodes are labeled with 0 or 1
  - all non-terminal nodes are labeled with a Boolean Variable.
  - Each non-terminal node has exactly two edges from that node to others; one labeled 0 and one labeled 1; represent them as a dashed line and a solid line respectively

  
NPTEL

So that is why this is the such idea I am giving you that how I am going to use this particular data structure to represent my function. So, instead of binary decision tree we are going to look for binary decision diagram. Which is binary decision diagram, it is very similar to your binary decision tree, but now my representation will be a graph instead of a tree I think you know what is the difference between trees and graphs. So, we have going to get a graph now when we have going to talk about the BDT. So, this is a dag director a cycle graph.

(Refer Slide Time: 20:11)



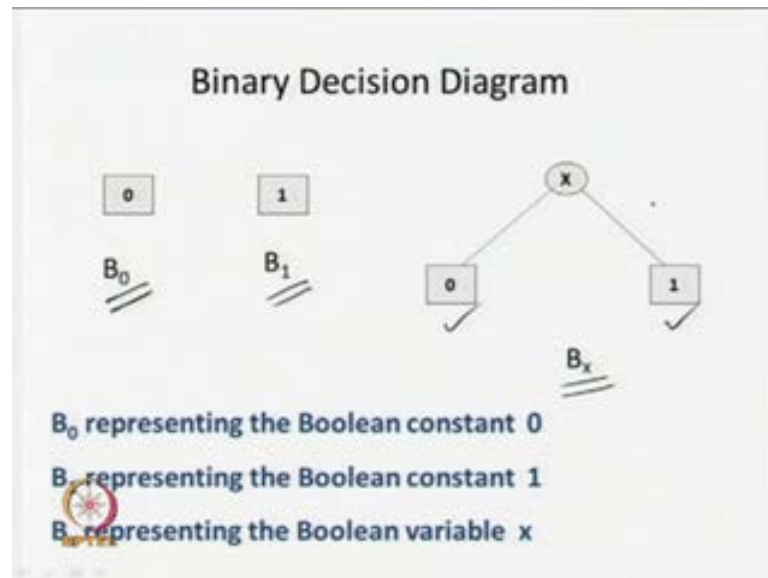
So, because we have going to talk about the direction also in this particular case though explicitly we have not mention it that means, when a is equal to 0 I will follow this particular path, b is equal to 0, c is equal to 0 we have going to follow and when c is equal to 1, we have going to follow this 1. So, this is some sort of your directed (( )). Now that is why you are saying that what is your BDD; this is a directed acyclic graph; that means, it is a directed to, but it should not have any cycle. And a similar way now we can talk about BDD also, so what happens in BDT we are seeing that we having terminal nodes and non terminal nodes in BDD also will we having terminal nodes and non terminal nodes.

In case of non terminal nodes that will be label by your Boolean variables, terminal nodes will be label by Boolean values 1 or 0 and every non terminal nodes are having 2 outgoing edges 1 as will be represented by your dash line and second one will be represented by your solid line; dash line says that, the value of this particular value is taken as 0 and solid line says that, the value of this particular variable is taken as 1. So, you just see that this is similar to your BDT only, but it is now we have going to get a graph instead of a three.

Now what is the primitive of our BDD, primitive BDD binary decision diagram. So, in this particular case we have going to get 3 primitive binary decision diagram and with the help of this 3 primitive binary decision diagram we have going to construct the

binary decision diagram of any given Boolean function. So, this is your  $B_0$  it is a terminal node label by 0, it says that the Boolean value is equal to 0. So, this is  $B_0$  is representing these particular BDD, which represent the Boolean constant 0.

(Refer Slide Time: 21:24)



Similarly, we are having one binary decision diagram BDD  $B_1$ , which represent the Boolean constant 1; that means, this is such a box with an one over here. So, it is a primitive BDD it says that, this is the Boolean constant 1  $B_0$  says that, this is the Boolean constant 0 and another BDD we are having which represent the Boolean any Boolean variable; so if you label by  $x$  then we have going to say this is the BDD for the Boolean variable  $x$  and we say this is your  $B_x$ . So, what it says that  $x$  it is having 2 outgoing edges, 1 dash, second one is your solid dash line is pointing towards the Boolean constant 0 and solid line is pointing towards the Boolean constant 1; that means when  $x$  is equal to 1, then my functional value is 1. So, this is their representation of BDD representation of our any Boolean variable  $x$ . So, this is order 3 primitive BDDs and with the help of these 3 primitive BDD, we have going to construct the BDDs of any given Boolean function.

(Refer Slide Time: 23:06)

**Shannon Expansion → BDD**

$$f = ac + bc$$
$$f = x f_x + x' f_{x'}$$

- $f_a = f(a=0) = bc = g$
- $f_a = f(a=1) = c + bc = h$

Now just see that what I am going to talk about, now we are saying that the BDD is nothing but our BDD can be constructed or represented with the help of Shannon expansion. Again consider these particular same Boolean function  $f$  equal to  $a$   $c$  plus  $b$   $c$  and the Shannon expansion is your  $f$  is equal to  $f$  of  $x$   $f$   $x$  plus  $x$  dash of  $f$  of  $x$  dash; that means, functional value of  $a$  when  $x$  is treated is equal to 1 and your functional value of  $f$  when  $x$  is equal to 0. That means, I can say that already I have say that  $f$  of  $a$  equal to 0 will be your  $b$   $c$ , because  $a$  equal to 0 will be your  $b$   $c$ , because  $a$  equal to 0 I am going to have  $b$   $c$  so I say that, this is your  $b$   $c$  and when  $f$   $a$  is equal to 1 then  $f$  of  $a$  is your  $c$  plus  $b$   $c$ , because  $a$  is equal to 1, 1 into  $c$  equal to  $c$  not a functional depends on  $c$ .

So,  $c$  plus  $b$   $c$  so when I am saying that, this is a sub function  $g$  and second is sub function  $h$ . So, this particular information we have going to represent with the help of this particular partial BDD. So, we are having a terminal nodes  $a$  either  $a$  have value 0 or  $a$  have value 1, so when  $a$  is equal to 1 then I am having the sub function that we have to evaluate this  $c$  plus  $b$   $c$  this is  $h$  and when  $a$  equal to 0 then sub function we are having  $g$  which is your  $b$   $c$ . So, this is the partial BDD for this particular given function after that taking the decision on  $a$ .

Now what we have to do next, now either we have to take decision on  $b$  or decision on  $c$ . Now if I am going to take decision on  $b$ , then I am going to label these 2 things will  $b$ . So, in this particular case now again I will apply this particular Shannon expansion for

this particular function g or this particular function h; this way we can construct our BDD also

(Refer Slide Time: 25:06)

**Shannon Expansion → BDD**

$f = ac + bc$                        $f = x f_x + x' f_x'$

- $f_{a'} = f(a=0) = bc = g$
- $f_a = f(a=1) = c + bc = h$
- $g_{b'} = (bc)_{|b=0} = 0$  ←
- $g_b = (bc)_{|b=1} = c$
- $h_{b'} = (c+bc)_{|b=0} = c$
- $h_b = (c+bc)_{|b=1} = c$

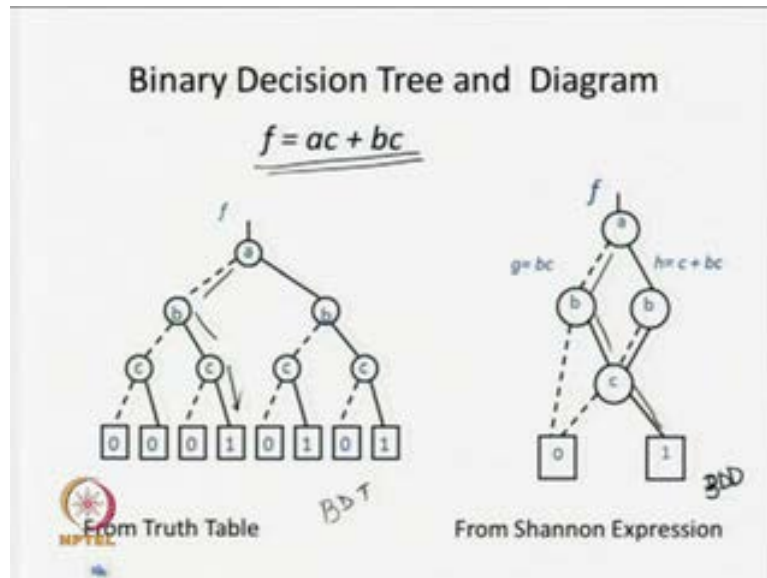
So eventually, now what will happen you just see now we have going to apply this particular Shannon expansion for all the variable. So, we are getting g and h here I am having the function g and here I am having the function h. Now will apply Shannon expansion on g and h, so if g if b equal to 0 then what will happen the functional value is 0 and when b equal to 1 then g b will be your c, because if it is your b equal to 1, 1 dot c will be c.

Similarly, in case of h if it is your, your b equal to 0 then it will be equal to c and when b equal to 1 then it will be c plus c which is equal to c. Now you just see that, now what we are trying now when b equal to 0 that functional value is 0 so we are writing doing this particular h and saying that, when b equal to 0 functional value is 0. When b equal to 1 then functional value is c so that is why I am doing this particular c node c and pointing from b to c with the help of solid line.

Similarly, in case of your function your h when b is equal to 0 the functional value is c, so this dash line is coming to c and when b equal to 1 then again the functional value is c that is why this solid line is coming to this particular node c. Now we are having this particular 3 function one is already getting the constant. Now you just see that this sub functions are depends on your c, if c is equal to 0, functional value is 0, c is equal to 1

that functional value is 1. So, for that from c I am having this particular dash line which is coming to 0 and I am having this particular solid line which is pointing towards this particular terminal node 1. So, eventually I am getting these particular BDD. So, this is the BDD of this particular given function  $a c + b c$ .

(Refer Slide Time: 27:24)



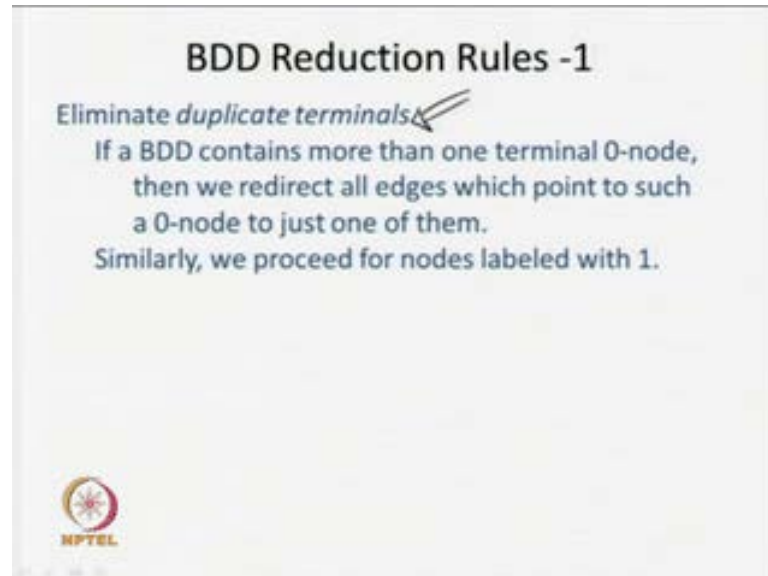
Now you just see what we have got now. We are taking this particular function  $f$  is equal to  $a c + b c$ , so from truth table I am getting this particular diagram which is your BDT basically binary decision tree. And by using this particular Shannon expansion I am getting this particular diagram, which is I am going to say this is your BDD binary decision diagram. So, both this diagrams are now representing the same Boolean function, because you can refer this particular graph and then eventually you can find out the valuation of this particular function say, if  $a$  equal to 0  $b$  equal to 1  $c$  equal to 1 functional value is 1. So, when  $a$  equal to 0  $b$  equal to 0  $c$  equal to 1 my functional value is 1, so from both this diagram will consider the same.

So, here we are having BDT and we are having BDD, but both are represent this function. So, here I am going to get some sort of your compact representation of this particular given function. Now they are equivalent both of them are representing the same function so whether can we exceed this particular BDD from this particular BDT, it may be possible. So, see what we are going to do so we can use some reduction rules and



with the help of this particular reduction rule; we can try to reduce our BDD or we can try to reduce our BDT.

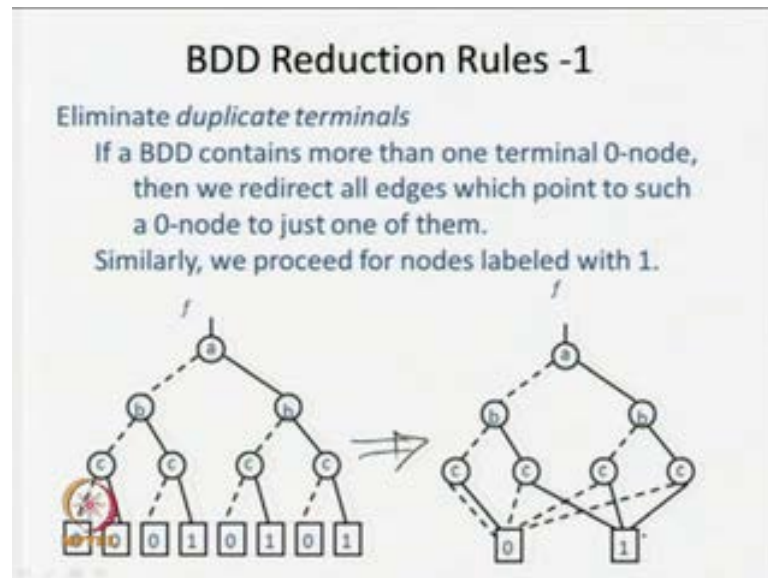
(Refer Slide Time: 28:56)



So, what is that reduction rule first reduction rule we are saying that eliminate of duplicate terminals. So, if we are having duplicate terminal nodes we can eliminate it. Say if we are having more number of terminal nodes which are going to represent the Boolean constant 0, then we can represent those particular terminal nodes with 1 terminal node. if we are having several terminal nodes which are representing Boolean constant 1 we can represent those particular several terminals with the help of 1 terminals (( )) represent this particular Boolean function. So, that is why the first rule we have saying that eliminate of this particular duplicate terminals.

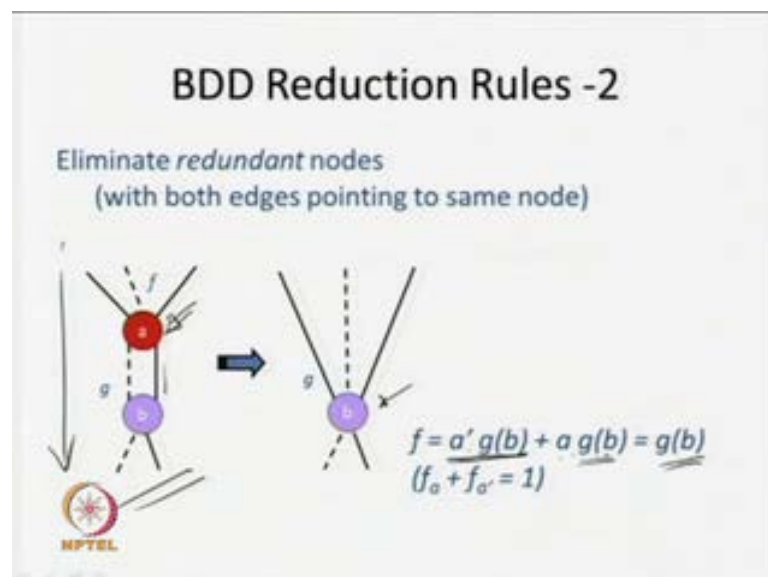
So, if we apply these things to our BDT then what will happen you just see so in case of these things what will happen, now all terminals node will be combine to 1 and that incoming edges will be redirected to this particular new terminal nodes. Now what happen you just see that this is the BDT that we had now we are having several terminal nodes some are represented by 0 or some are represented by 1.

(Refer Slide Time: 29:46)



Now all 0s will be combined and represented by one terminal nodes and all 1 will be combined together there will be represented by 1 terminal nodes 1. And whatever incoming as is the coming to those particular terminal nodes they will be redirected to this particular terminal nodes. So, after eliminating this particular duplicate terminals what I am getting from this BDT we are getting some sort of BDD over here.

(Refer Slide Time: 31:06)



So, now at least we can see that, the number of nodes are let us here, because instead of 2 to the power n nodes terminal nodes we getting only 2 terminal nodes. So, here we had 8

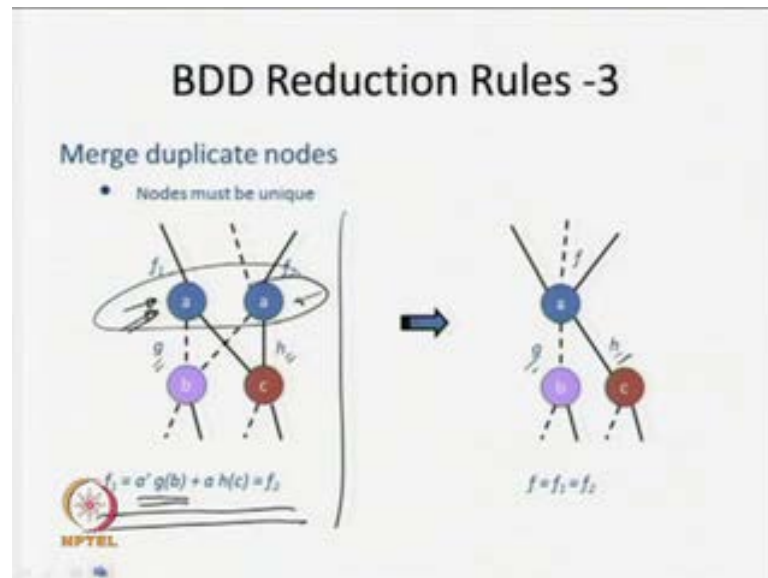
terminal nodes, because we are having 3 variables so these 8 terminal nodes will be represented by 2 terminal nodes. So, it is for any function if we have a function of  $n$  variables then in BDT, we have going to get  $2^n$  terminal nodes, but in BDD we can represent this  $2^n$  terminal nodes by only 2 terminal nodes so there is a massive reduction. So, we need to have only 2 terminal nodes when we talk about BDD.

This is one reduction that is why eliminate of duplicate terminals. So, second reduction rule we have going to say that eliminate of redundant nodes; that means, with both edges pointing to the same node. We have going to see, why to we have going to talk about the redundant nodes. So, eliminate of redundant nodes what we are going to say about these things. Now you just see that, we are having this particular partial BDDs just consider this one say we are taking some decision where coming to way. When  $a$  equal to 0 it is pointing to this particular non terminal nodes, when  $a$  equal to 1 again this pointing to this particular non terminal nodes. So, now you just see that, when  $a$  is equal to 0 then 0 into that functional value depends on  $b$ .

So, that is why I am saying that, when  $a$  equal to 0 then  $a$  this in to  $g \cdot b$  dot remaining portion. Similarly, when  $a$  equal to 1 again we are getting that  $a \cdot g \cdot b$ , because already we know  $a$  equal to 1 and this is the remaining portion that we have to evaluate. So, this is basically what happen you say that,  $a \cdot g \cdot b$  plus  $a \cdot g \cdot b$  which is equal to  $g \cdot b$ ; that means, now you just see that what about function, that I am going to have it is basically redundant or irrelevant on the decision of this particular  $a$  in this particular path. So, in this particular path so that is why what I am saying that we have going to eliminate this particular redundant node. So, if we are having such type of redundant nodes we have both 0 and 1 edges pointing to the same sub function then we can eliminate these things. So, we eliminate these particular nodes and all the incoming edges will be directed to this particular node.

So in this particular way we can eliminate some of redundant nodes. After eliminating this particular redundant nodes my size of your BDD will get reduce. That means, we have going to see the compact representation of our Boolean function. So, first redundant, first reduction rule is your elimination of duplicate terminals,, second deduction rule is your eliminate of your redundant nodes.

(Refer Slide Time: 33:33)

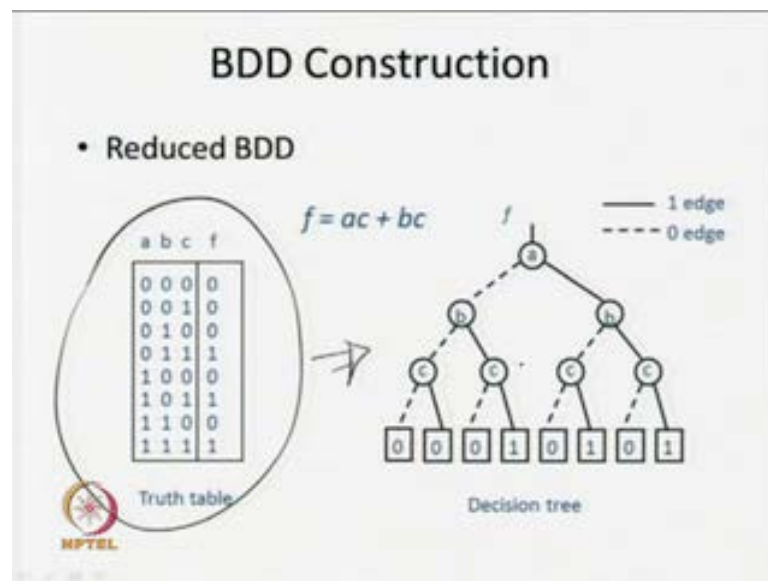


Now may we have one more rules which is your reduction rules 3, which is nothing but call you are merging of your duplicate nodes. So, if we are having some duplicate nodes then we can merge them to get a and we have going to get a some sort of simple or reduce BDD. Now you consider this particular partial BDD so what happens in this particular case you just see that I am coming to this particular 2 nodes. Now I have to take the decision on a so f a we evaluating and in this particular node I am going to take the decision on a. So, what will happen in this particular case if a is equal to 0 then I am coming to this particular node b, when a equal to 1 I am coming to this node c that means, this is a dash into g b. So, this is basically coming to this particular dash line is going to say a dash dot g b.

And when I follow this particular solid line then I am going to get a dot h c, that sub function that we are having in this particular node is o h so a dot h. So, when I consider this particular node a, this particular node I am getting this particular function. Similarly, I am coming to this particular node and whatever we have evaluate over here is a state f. Now again we just see that, when a is equal to 0 we are coming to that same node b and when that means we are going to look for same function g and when a equal to 1 where coming to the same function f. So, in this particular case the partial evaluation of the given function for f 1 and f 2 both these function are same. So, we are saying that f 1 equal to f 2 which is your add a g b plus a h c.

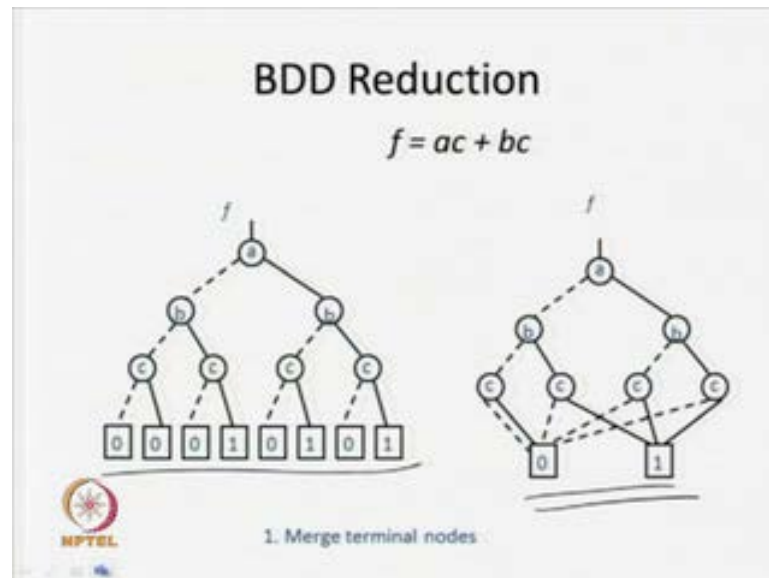
So, in this particular case we can say that these are basically some sort of duplicate nodes that we have in our BDD. So, what we can do know, we can eliminate these merge these 2 nodes to a single node. So, in this particular case both  $f_1$  and  $f_2$  will be equal to your  $f$  so you can say that merging these 2 nodes to 1 and these are the sub function that we are having when it is equal to a equal to 1. We are going to evaluate the sub function  $h$  and when it is equal to 0 then we have going to evaluate this particular sub function 0. So, like that we can merge this particular duplicate nodes and eventually we have going to get a some sort of your reduce or compact representation of our given Boolean function.

(Refer Slide Time: 36:05)



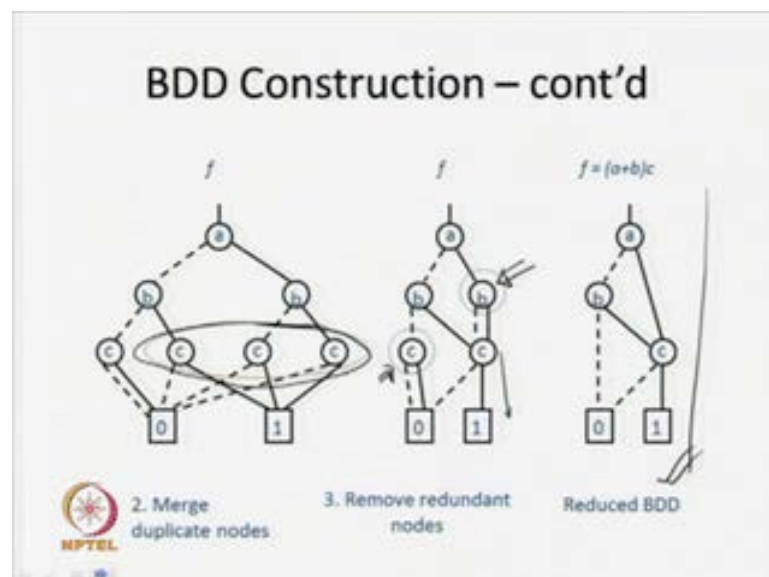
So we can have these particular 3 nodes. Now you just see that, first we are getting this particular truth table you are looking into a function representing by truth table and we are drawing this particular binary decision tree from this particular truth table. So, it is having a exponential glow up. Now what we are going to see, we have time to reduce it. When we have going to reduce it according to the past rule we have going to merge terminal nodes. So, we try to merge those particular terminal nodes, so we are getting 2 terminal nodes one is 0 and second one is 1 and we have going to redial a those incoming edges to these 2 terminal nodes.

(Refer Slide Time: 36:26)



So, after application of first deduction rule we are reducing the terminal nodes to only 2. So, you may have 2 to the power n terminal nodes in binary decision tree, but eventually we have going to get 2 terminal nodes in our binary decision diagram, with the help of first reduction rule merging a terminal nodes we have getting only 2 nodes once we are having this particular 8 nodes.

(Refer Slide Time: 37:13)

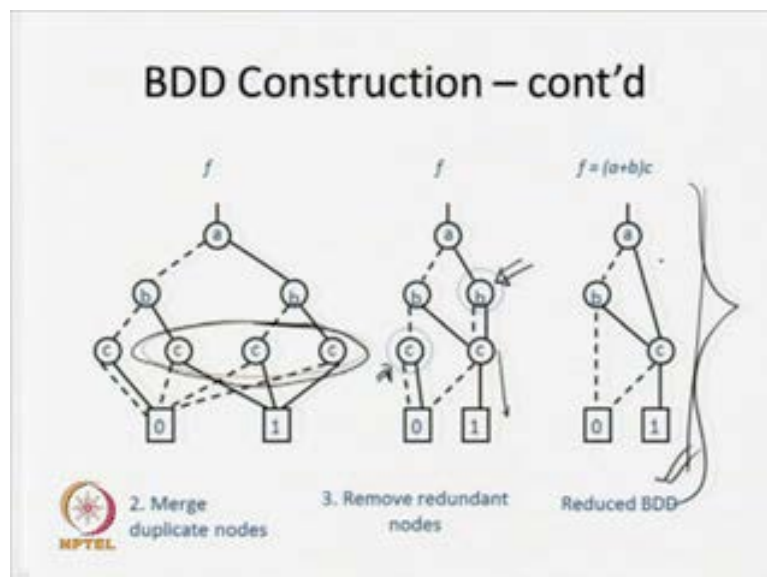


Now next you just see that after getting this particular some sort of your reduction, now we are going to look inspect those particular nodes. Now you just see these particular 3

node c. When c equals to 0 the function value is 0 when c is equal to 1 the function value is 1. So, nature of these 3 nodes are same so these are basically some sort of duplicate decision that we are having. So, in this particular case we are going to merge these 3 nodes true 1 particular node c and the incoming edges will be redirected. So, form b it is 1 coming to c, so b equal to 1 coming to c, b is equal to 0 coming to c, b equal to 1 coming to c. So, these are the 2 solid line so we are redirecting those particular edges. So, we are getting some sort of reduction over here, when we are merging the duplicate nodes so by this particular reduction rule merging of the duplicate nodes.

So after that we have going to see whether we are having any redundant nodes or not so in this particular case you just see that if you look this particular node b. What will getting, when b equal to 0 I am coming to see when b equal to 1 where coming to c. That means, whether b equal to 0 or 1 it is immaterial we have going to evaluate this particular some functions. Similarly, when I look in to this particular c, again it c equal to 0, the functional value is 0; c equal to 1, functional value is 0. So, again it is immaterial on the decision of these particular c so what happens now, we have going to get this particular diagram BDD; where we are removing this 2 redundant nodes so eventually we are getting this particular BDD.

(Refer Slide Time: 39:55)

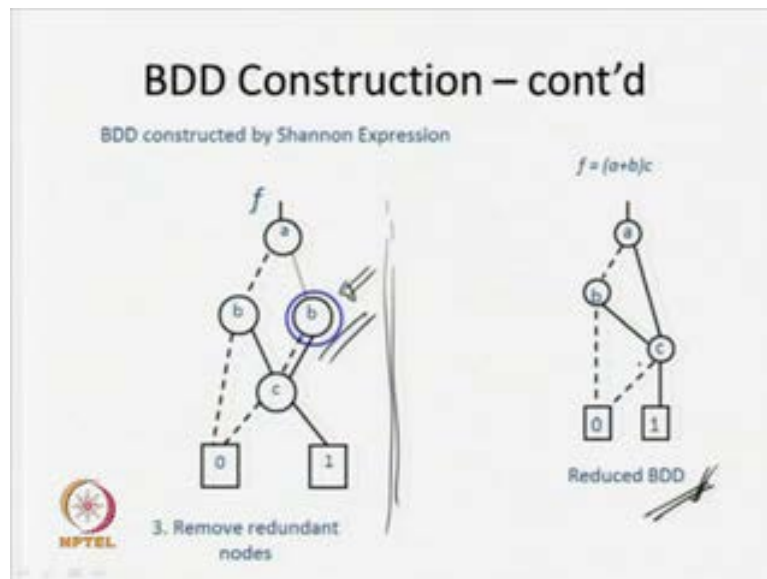


So, we have applied your this reduction rule, remove a all your non terminal nodes, merging of duplicate nodes and remove of redundant nodes after removing of redundant

nodes it may happen that some of some duplicate node may arise again one here. So, what we have going to do, will again apply this particular merging of duplicate nodes; after merging of duplicate nodes what will happen, again some redundant may arise, again we have going to remove those particular redundant node. So, in this way what happens we are going to repeat these particular 2 rules and try to eliminates more and more redundant nodes and merges duplicate nodes.

So, eventually we have coming to this particular BDD. So, you just see that, this is what we have talked about here that that we are looking into the BDT and you have applying those particular deduction rules eventually we are coming to a representation. And you just see that no more deduction is possible, because a, b, c these are a 3 nodes. There is no redundant node, because all 0s and 1 is going to deferent direction. We do not have any duplicate also, because this is the levels are deferens so since are levels are deferens the question of duplication is not arise over here.

(Refer Slide Time: 40:34)



Now this is from BDT, we have coming to this particular reduce BDD. Similarly, we have the same function we have taken and you have constructed the BDD with the help of your Shannon expression. So, this is your constructing this particular BDD with the help of Shannon expression. So, after constructing the particular BDD will see whether it can be reduce or not. So, first rule talks about the redundant remove well of your

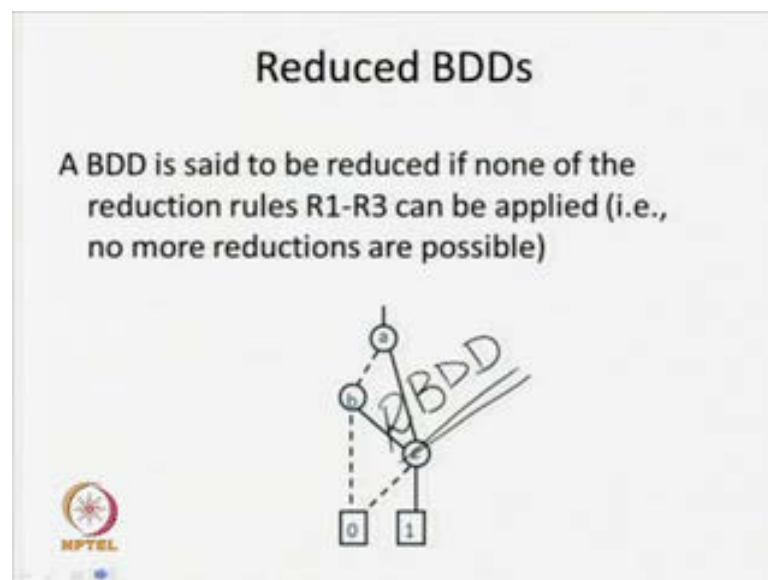


terminal nodes. So, since we have using that Shannon expression so there will not be any your redundant terminal nodes, because you have going to have only 2 terminal nodes.

So, basically we have to look for a other 2 rules merging of duplicate nodes and remove well of redundant stats. So, when I am coming to this particular diagram BDD we have founded this particular node b. It is doing some short of your redundant test on, because b equal to 0 and 1 is going to have the same sub function. So, we can remove these particular redundant nodes and ultimately we have coming to this particular BDD.

So whatever you have going to do if we are starting from your BDT or you apply Shannon expression to get a your BDD; eventually we can apply those particular reduction rules and finally, you come to a particular BDD. So, we have coming to this particular BDD and no more reduction is possible over here. So, by looking into this things what happens, now you have going to have the notion of your reduced BDD reduce binary decision diagram, which is say within as your RBDD reduce binary decision diagram.

(Refer Slide Time: 42:02)

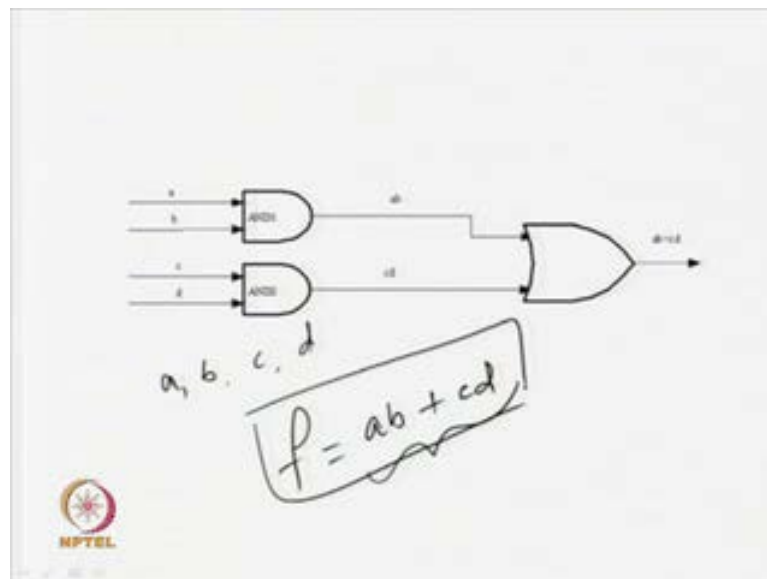


When you say that a binary decision diagram will be reduced one will say that, it will be reduced one. If none of the reduction rule r 1 and r 2, r 3 can be apply. So, we are having 3 rules r 1, r 2 and r 3 if none of this reduction rule can be applied anymore; that means, no more reduction is possible. So, if no more reductions are possible at that point we have going to say that this is the reduce BDD we have going to getting it.

So, we say this is RBDD or say reduce BDD. So, this is the notion so when we have coming to that reduce BDD that means, no more reduction is possible and we are getting a representation of my given Boolean function. So, you just see that we are representing of our Boolean function with the help of a decision diagram which is known as your BDD binary decision diagram. It can be reduce to get a reduce BDD and in reality we have found that in most of the cases we have going to get a compact representation of our Boolean function.

So, initially what happens when you talk about BDT we will find that we are having a exponential blow up, we are having total  $2^n$  number of terminal nodes. If we are having  $n$  variables, but we can apply the Shannon expression also to construct a BDD for a given function, in when we are using for a Shannon expression at least we can assumed that, we have going to get a 2 terminal nodes and we may have some none terminal node.

(Refer Slide Time: 44:42)

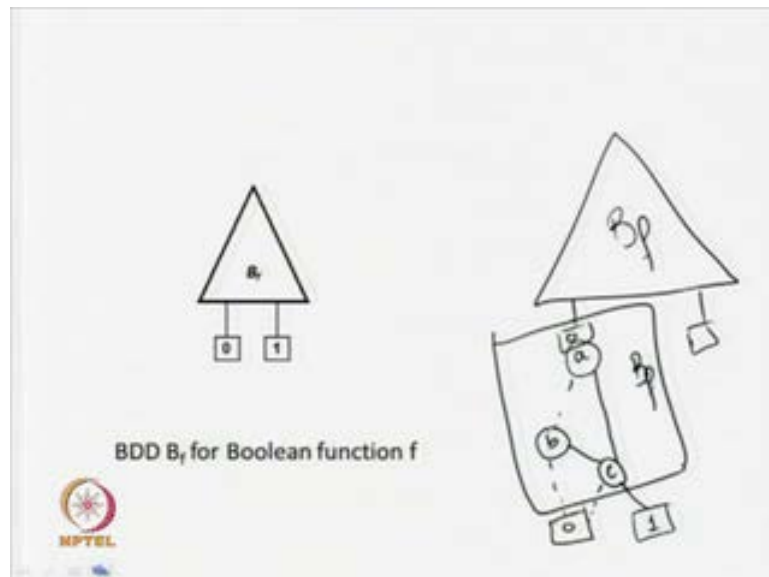


When we are getting the initial BDD with the help of Shannon expression then we try to use the reduction rule, removal of your redundant test and marginal of duplicates none terminals. By repeated that applied of this two rules we have going to get compact representation of our given Boolean function. So, when we are having a compact representation of our Boolean function, we have going to use this things while representing our Boolean function and it is going to help us to up to (( )) with the state

space expression problem. Will see how we have going to represent our state space with the help of binary decision diagram.

Now say if I am going to have any Boolean logic circuit. Just for expression I am saying that say I am having 2 and gates over here and 1 or gates that input variables are you are a, b, c, d. So, if it is the four input variable and if this is the circuit given to me, than what happens I can write the function for this particular function is your this is nothing but some of product. That means, a into b plus c into d you just see that if we are having any Boolean circuit and logic circuit we can always right the Boolean expression for this particular circuit. And once we have this particular Boolean expression than what will happen always we can try to construct the BDD for this particular Boolean function.

(Refer Slide Time: 45:50)

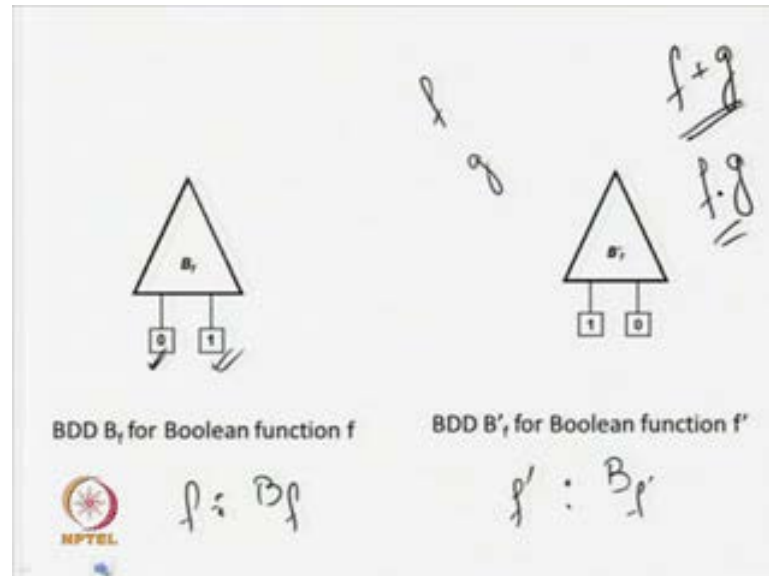


We just see once we have constructing the BDD, than we can use those particular BDD in our walk. Now you just see now we are having some BDDs just we are representing this particular BDDs something like that in symbolic way. This triangular we are writing that this is the  $B_f$  BDD of the Boolean function  $f$ . Just look into the previous example say I am having say a either this is b, if b equal to zero than I am having the functional value s 0 if b equal to 1 I am coming to see if a equal to 1 c is equal to 0 functional value is 0 and c equal to 1 my functional value is 1.

So, this is the BDD, that already we have seen that is a simple BDD. Now this particular tree than early we represent with the help of this particular triangular and we say that,

this is the function BDD power function  $Bf$  and eventually it is having this particular 2 terminal nodes 0 and 1. So, this is basically we have representing binary decision diagram  $Bf$  this is the graph that we have an eventually we having this 2 terminal nodes.

(Refer Slide Time: 47:07)



So, if we are having a BDD of  $Bf$ , then we can do some other operation on this particular BDD, BDD also. Now you just see that if I am having a function  $Bf$  say  $f$  I am having some function than the compliment of  $Bf$  is your return by your  $f$  bar. Now if the BDD of this function is your  $Bf$  than I am going to get the BDD for this particular compliment of this function  $B \bar{f}$ . Now if we are having the BDD of a particular function to get the compliment is very simple, because we know that if it is compliment basically if  $x$  is equal to 0, than compliment of  $x$  is 1 if  $x$  is equal to 1 that compliment of  $x$  is equal to 0.

So, what we are going to do, simply we are going to inter sense this particular terminal nodes. So, if it is 0 than 0 will be replace by 1 and 1 will be replace by 0. So, this is the way that we can say that if I am having given Boolean function if we know the BDD representation of this particular Boolean function than compliment can be very easily find out. What the simply inter sense the terminal nodes, 0 will be replace by 1 and 1 will be replace by 0 so this is very simple.

Secondly if we are having some Boolean function generally than what we are going do, we are going to performs on Boolean operation also. Say if I am having one function  $f$  and one function  $g$  of say same variable of define variable than what happens, I can

perform Boolean function  $f$  plus  $g$  or I can perform Boolean function  $f$  dot  $g$ . So one is conjunction, second one is rejection. Now say if I am having 2 Boolean function we can do it. If I given any 2 Boolean function to you, always you can find out the some of this 2 Boolean function of product of this Boolean function this junction and conjunction. Now if I give the BDD representation of these 2 functions in state giving a original function I will give the BDD representation of this 2 function. With I can you construct  $f$  plus  $g$  or  $f$  dot  $g$  again if you look into it this are very simple very simple you can do it. You just see that that, this is basically representing  $f$  plus  $g$  that first diagram. Why I saying that this first diagram representing  $f$  plus  $g$ , because  $f$  plus  $g$  if you say that if anyone is one than output is one.

(Refer Slide Time: 49:06)



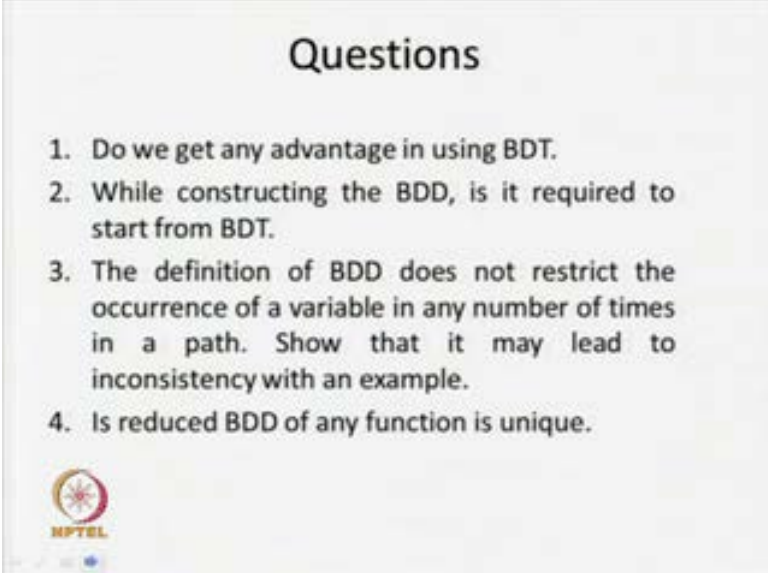
So first I am important thing  $g$  this is your important thing  $f$  this is your  $Bf$ , if  $f$  is equal to 1 than what will happen  $f$  plus  $g$  will be 1. So, if  $f$  is equal to 1 I am just taking this decision my functional value is 1. Now similarly, if now  $f$  is equal to 0, so  $0$  plus  $g$  so this is basically depends on  $g$ , if  $g$  is equal to 1, my function value is 1, so  $g$  is equal to 0 my functional value is 0. So, that means in this 0 terminal nodes just I am going to again this particular  $b g$ . So, this particular whole diagram is going to give me  $f$  plus  $g$  so just see that construction  $f$  plus  $g$  is also very simple.

So,  $f$  dot  $g$  is other way round if  $f$  is equal to 0, then the value of  $f$  that  $g$  will be equal to 0, if  $f$  is equal 1 that the function value will depend on your value of  $g$ , because  $f$  dot  $g$   $f$

is equal to 1,  $1 \cdot g$ . So, basically if  $g$  is equal to 0, then my value is 0, if  $g$  is equal to 1, my value is 1. So, that is why now what I am doing first  $b \cdot f$  is representing this particular Boolean function  $f$  if  $f$  is 0, so this is the result my function value is your 0, if the  $f$  is 1, then what happens I am going to a loop order value of  $g$ . So, I will just simply plug in this particular BDD of  $b \cdot g$  for function  $g$  in that one terminal nodes of this particular  $Bf$  and eventually it is going to give a practice particular and let what I can do again since I can say that these  $(( ))$ .


So, that means I can rewrite this particular  $s_2$  this things and I can rewrite this particular  $s_2$  this particular again after having these 2 BDD, we are getting an BDD if caught I can go for deduction of these particular  $b$  BDDs course. Now you just see that we have seen one particular refer structure none as your BDD binary decision diagram, we can represent our Boolean function with the help of your BDD binary decision diagram. And I mention that from result also we have found that in most of the cases, we have going to get a compact representation or any given Boolean function. Now so look for some questions very simple question I am giving it to you. First question I am saying that, do we get any advantage in using BDT binary decision tree?

(Refer Slide Time: 51:27)



### Questions

1. Do we get any advantage in using BDT.
2. While constructing the BDD, is it required to start from BDT.
3. The definition of BDD does not restrict the occurrence of a variable in any number of times in a path. Show that it may lead to inconsistency with an example.
4. Is reduced BDD of any function is unique.

 NPTEL

So we have saying that we have going to look for a data structure, which is going to give as a compact representation of your Boolean function. I have mentioned about BDT binary decision tree. Now whether do you get any advantage of your BDT, so if you look

into it the construction of BDT and the BDD. That it looks like h as we are not getting any advantage why, because it is simple mapping or simple transmission of my truth table to BDT. In truth table we are having the problem with your exponential bro up; if I am having  $n$  variables I am going to get 2 to the power  $n$  define an entrance.

So, in BDD also we have going to get 2 to the power  $n$  terminal nodes and whether label will be a is equal to your  $n$ . So, as a simple example you see that if I am going to have a function which is having 8 variables. Then the total number of vantage in our truth table will be your 2 the power 8 which is your 256, when we are going to look into the BDT, we are going to get 256 terminal nodes. And the level of this particular BDT will be 8, because for each variable we are having 1 label. So, actually we are not getting any advantage, but for clarity just I am mentioning what is BDT and but from BDT we can use the reduction rule and we can get the BDD.

Now second question you just see that, while constructing the BDD is it required to start from BDT? Again I think it is a very simple I am sorry, after lessoning to my lecture you can variable say that this is no, because BDT can be again pureed as a repeated application of our Shannon expanse. That means, when we can construct the BDT and apply the reduction rule, we are having 3 reduction rule, this is your eliminate of terminal nodes, duplicate terminal nodes, removal of redundant state and merging of your duplicate non terminal nodes. You can apply those things and you can find out those particular BDT or reduce BDD on the other hand we can use the Shannon expansion if it we have going to use this particular Shannon expansion on each variable and you can get the initial BDD.

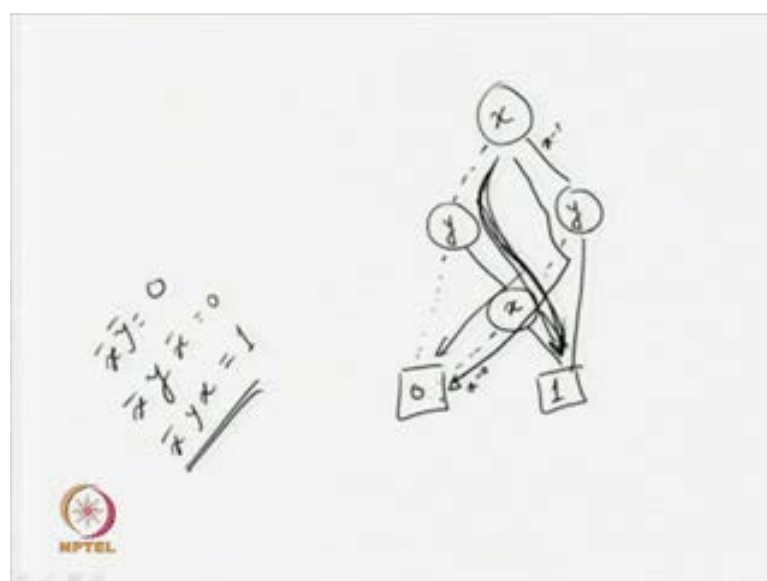
So, after application of Shannon expansion whatever BDD we are getting it may not be a reduced one, but of course we are having only 2 terminal nodes, one representing the constant 0 and second representing the constant 1. So, after getting this particular binary decision diagram we can apply the other 2 reduction rule, one is your removal of redundant test and second one is your merging of duplicates non terminals.

So, reputedly we have going to apply this particular rules and finally, we have going to get a reduce BDD; when we said that it is reduce BDD when none of the reduction rule can be affect further. Now third question you just see that, the definition of BDD does not restrict the occurrence of a variable in any number of times in a path. Show that it

may lead to inconsistency with an example. Now you just see that I am defining the BDD binary decision diagram or reduce binary decision diagram and when we are defining it, we are saying that it is a DAG directed acyclic graphs, which is having non terminal nodes and terminal nodes. Non terminal nodes are having or it is labeled with your variables and terminal nodes are labeled with constant 0 and 1. And every non terminal nodes are having 2 outgoing has as one is representing 0, which is given by dashed line and second one is representing the value 1 which is given by the solid line. Now I am saying that since this is the definition of BDD so it is not (( )) ask to a use the same variable in many different places and they can come in any order and they can come in any order.

So if we use this particular basic definition and you can construct, whether it is give as any problem, if it is going to give as any problem then so it with an example. So, now what I can say, now already just said now one particular BDD say this is your x, x equal to 0 and x equal to 1 I am getting y. So, if y is equal to 0 say I am having this functional value 0 if y is equal to 1 say I am drawing something like that x. So, I can draw this particular BDD as per our definition of BDD, it is correct BDD. So, I am having 4 non terminal nodes, have a non terminal nodes, having to outgoing (( )) I am having 2 terminal nodes and the non terminal nodes are leveled with does variable x y and z x. We are having it is a function of 2 variables say x and y.

(Refer Slide Time: 56:04)





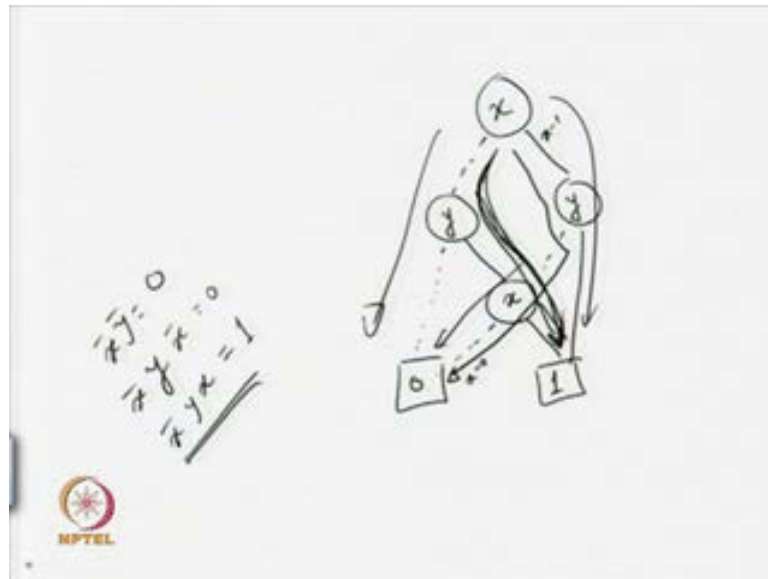
So, now you just that in this particular case when  $x$  equal to 0,  $y$  is equal to 0, my functional value is 0 that means  $\bar{x}, \bar{y}$  is equal to 0. When  $x$  equal to 0,  $y$  equal to 1, then what will happen again I am going to take decision on  $x$ . If  $x$  equal to 0, then I am going to get 0, and when  $x$   $y$  when  $x$  equal to 1, then I am going to get 1. Now you just see that in this particular case, I am having  $\bar{x}, \bar{y}$  is 0, basically I am looking for this particular execution part or this particular part valuation actually  $x$  equal to 0,  $y$  equal to 1, and  $x$  equal to 0.

When I look into this particular valuation basically I am looking that valuation of  $x$  equal to 0, valuation of  $y$  is equal to 1, value  $x$  equal to 1. Now you just see that in this particular part now valuation of  $x$  is taken as 0 as well as 1, but you know that if we are going to consider any Boolean function if you look into any Boolean variable. At any instance of time that Boolean variable can take only one value, which is known as your 1 value it is either 0 or 1, but it cannot take both the value 0 and 1 together.

So, if you look into this particular part what will happens the value of  $x$  is taking 0 as well as 1. So, this is basically in consistence we are getting an inconsistency so we are having an inconsistent path in this particular representation. So, though our basic definition is of BDD is not restricting about the occurrence of variables, in any number of times, in any places, but we may face problem. So, in that particular case what happens we have to come up with the notion of your consistent parts and when we are going to look for the evaluation of this particular Boolean function, it should be evaluated through the consistent part only. We should not evaluated through inconsistent part.

So that is why, we are having this is as one in consistent part, because here  $x$  equal to 0 and  $x$  equal to 1 which is in consistence similarly, we are going to get another inconsistent part  $x$  equal to 1,  $y$  equal to 0 and  $x$  equal to 0. So, here  $x$  equal to 1 and  $x$  equal to 0 this is also inconsistency. So, we cannot have any valuation through this particular path. So, this is the problem that we are having so if we look into the basic definition of BDD then valuation of this particular Boolean function have to be done over consistent path only.

(Refer Slide Time: 59:48)



So this is a consistent, but we do not have any problem  $x$  equal to  $0$ ,  $y$  equal to  $0$ , we are going to get this particular value. Similarly, this is also another consistent, but  $x$  equal to  $1$ ,  $y$  equal to  $1$ . We do not have any so valuation have to be define over consistent path we may have inconsistency.

(Refer Slide Time: 60:00)

### Questions

1. Do we get any advantage in using BDD.
2. While constructing the BDD, is it required to start from BDT.
3. The definition of BDD does not restrict the occurrence of a variable in any number of times in a path. Show that it may lead to inconsistency with an example.
4. Is reduced BDD of any function is unique.

Another question we have talking about is reduce BDD of any function is unique. So we are talking about BDD we are going to get the after application of reduction role we are going to get the reduce BDD.

(Refer Slide Time: 60:25)

**Shannon Expansion → BDD**

$f = ac + bc$                        $f = xf_x + x'f_x'$

- $f_{b'} = f(b=0) = ac = g$
- $f_b = f(b=1) = ac + c = h$
- $g_c = (ac)_{|c=0} = 0$
- $g_c = (ac)_{|c=1} = a$
- $h_c = (ac+c)_{|c=0} = 0$
- $h_c = (ac+c)_{|c=1} = 1$

Now whether the reduce BDD representation of any function is unique. At says if you look into it let see so this is the function that we are working out  $f$  is equal to  $a c$  plus  $b c$  so a  $b c$  and I am getting this particular BDD and no more reduction is possible so it is an reduce BDD of the given function. Now the same function I can do in another way. Now what will happen? Now here what we are doing basically, we applying this particular Shannon expansion in different way.

First I am applying the one particular variable  $b$ . So, this is  $b$  after taking the decision of  $b$  I am going to take the decision on  $c$  that means I am using the Shannon expansion on that variable  $c$  and finally, we have going to apply the Shannon expansion on  $a$ . So, you just see that now I am having  $b c a$  so this is another reduce BDD on this particular 3 variable  $a b c$ . Now again you just see that I think it is a reduced one you cannot apply any more reduction in this particular thing. So, we are getting reduce BDD so same function I am taking  $f$  is equal to  $a c$  plus  $b c$ . I am getting 1 BDD this way and another BDD what I am getting a  $b$ ,  $b$  is equal to 0, I am going to get 0, if  $b$  is equal to 1 in difference on  $c$  and 1  $c$  is equal to 0 1. So, these are the 2 RBDD both are representing the same function  $f$  is equal to  $a c$  plus  $b c$ .

So, that means what we can say that, the BDD representation or reduce BDDs are not unique for a given function. And formed is we have seen, but here we have seen that the order the Shannon expansion that we have used for constructing this BDD is in the order

of a, b, c in the variable. And here we have applied the Shannon expansion on the variable in the ordered b c a. May be due to this division we are getting different representation. In next class we have going to talk about those particular issues about the ordering of this particular variable, but you just see that if you simply talk about BDD and if you talk about RBDD reduce binary decision diagram, then this is not unique this is one simple example I will give you I will stop here today.