INDIAN INSTITUTE OF TECHNOLOGY GUWAHATHI

NPTEL

NPTEL ONLINE CERTIFICATION COURSE
An Initiative of MHRD

VLSI Design, Verification & Test

Prof. Jatindra Kr. Deka
Department of CSE
IIT Guwahati

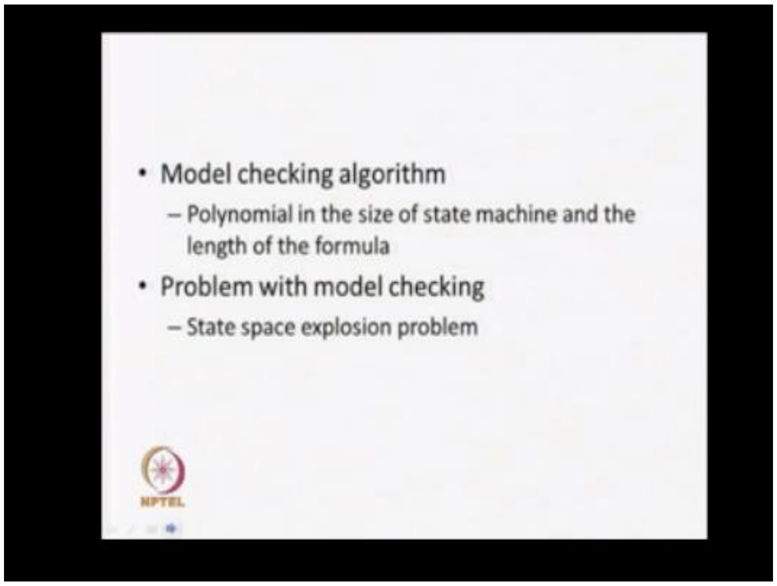
Module VI: Binary Decision Diagram

Lecture I: Binary Decision Diagram: Introduction and construction

Today I start a new topic called Binary Decision Diagram, called in short BDD. Okay, what you have seen till now, that you have seen about the application of the system, product, we need to have a model of the system and we need a specification language, so we are talking about CTL composition logic 2, given our specification and we have taken our model as a final estimation. Okay now after that we have discuss about the CPL model seeking algorithm, now if you give model of our system and if you give a specification , then we have a platform for model seeking algorithm to check whether the given specification is 2 in our system or not.

So if you look into this particular model seeking algorithm, you will find that this is somewhat similar to a craft referred algorithm, okay. So it oppose the entire craft and will check whether it is 2 in some state or not and ultimately the model seeking algorithm returns the state, state of states where the formula is 2. Again well we discussed this particular model seeking algorithm, we found that we are having a polynomial dynamic algorithm, who the CPL model seeking, which is polynomial in the size of our quickest structure which is basically polynomial in the size of craft that you are giving into it and also it is in the polynomial in the size of the given formula or length of our given formula.

(Refer Slide Time: 02:04)

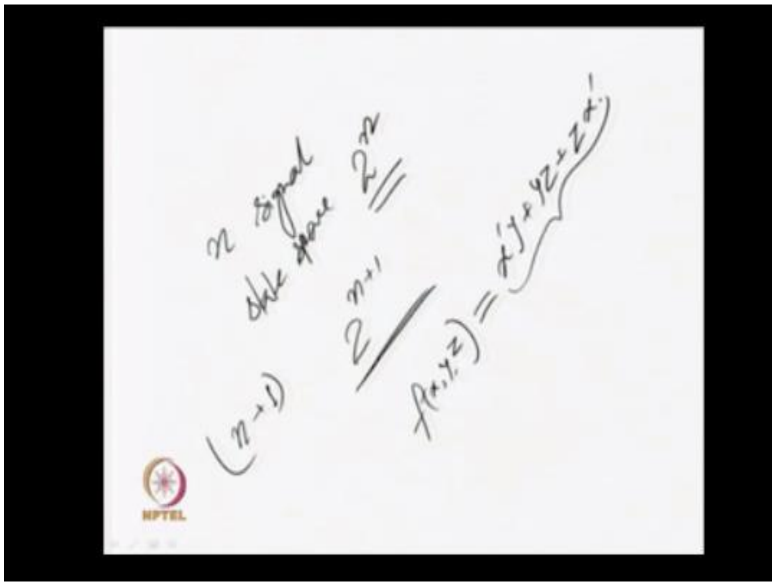

- Model checking algorithm
  - Polynomial in the size of state machine and the length of the formula
- Problem with model checking
  - State space explosion problem

So we have seen that we have a polynomial dynamic algorithm, and we have a automatic metrology, automatic algorithm, we can develop the algorithm, we can implement these things, we can check whether given formula is 2 in the given module or not, meanwhile in the abrasion it's equal, so you happy with that. But you have one more problem with this particular model seeking algorithm, this is known as your state space explosion problem. What is this state space explosion problem, basically future seeking, if we are working with a system where we are having n control variables, okay.

So all n control signals that means those n control signals can be taken out for propagation, so number of different combinations will be 2 to 4^n, so for n signals, state space is 2^n because you rare having 2^n difference combinations. Now say while you are designing it post nature you have started with n signal, but you have founded it cannot be, design property we need one more signals, so we incorporate one more signal in our signal and eventually the number of signal will become n+1 and state space will become no2^{n+1}.

So in this particular, you can see that the state space is in exponential in nature, that means exponential to the number of signals that we have, so for a smaller system is fine but for the bigger system it's going to be the problem, so that’s why we are now looking into the somewhere around to refrain our system, so for that BGD is one solution, with the help of BGD or we can tackle this particulars that space explosion problem, okay. Now that why you are going to talk about BDD, okay what is BDD, it's a Binary Decision Diagram which the data structure to implement or to slow any Boolean function.

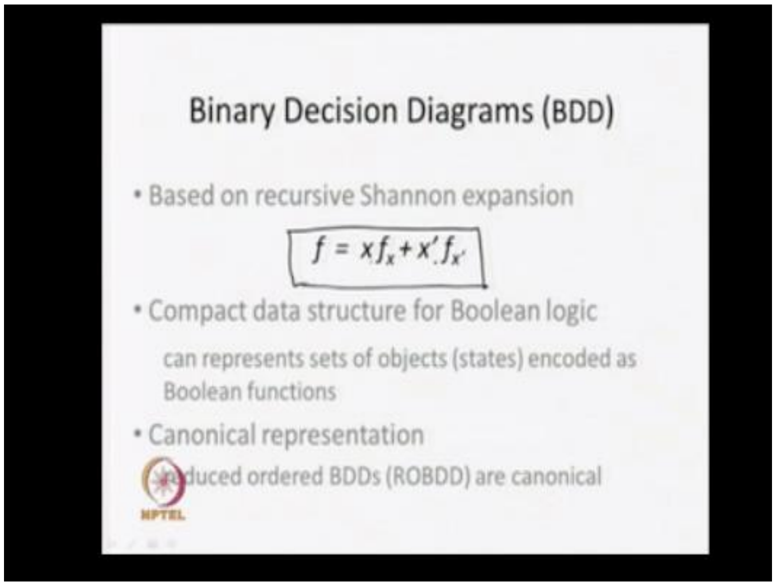
(Refer Slide Time: 04:50)


n signal
state space 2^n
$(n+1)$ 2^{n+1}
$f(x,y,z) = x'y + yz + zx'$

SO if we having a Boolean function of 3 variables, x, y, z then function f (x, y, z) represented with some Boolean expression x’y + yz +zx’, so if we having sub level Boolean expression, then this Boolean expression can be stored or represented with the help of binary decision diagram, so we are going to look into this particular binary decision diagram, in this particular module. And that all we are going to see how the final step matching in the state condition map will be implemented or represented with the help of BDD and will see how the models is working, can be incorporated or BDD’s.

Well we use BDD’s for our model seeking algorithm, then we say this symbolic models again, so eventually will go to symbolic model seeking algorithm again where the system will be represented with the help of BDD, Binary Decision Diagram. What is your BDD? So I have already mentioned that BDD is used to represent any Boolean function, so it is basically, BDD is based on recursive Shannon expansion, we know that any Boolean expression can be represented in an expansion, so this Shannon expansion is given withy expression.

(Refer Slide Time: 05:52)



Binary Decision Diagrams (BDD)

- Based on recursive Shannon expansion

$$\boxed{f = xf_x + x'.f_{x'}}$$

- Compact data structure for Boolean logic

  can represents sets of objects (states) encoded as Boolean functions

- Canonical representation

  reduced ordered BDDs (ROBDD) are canonical

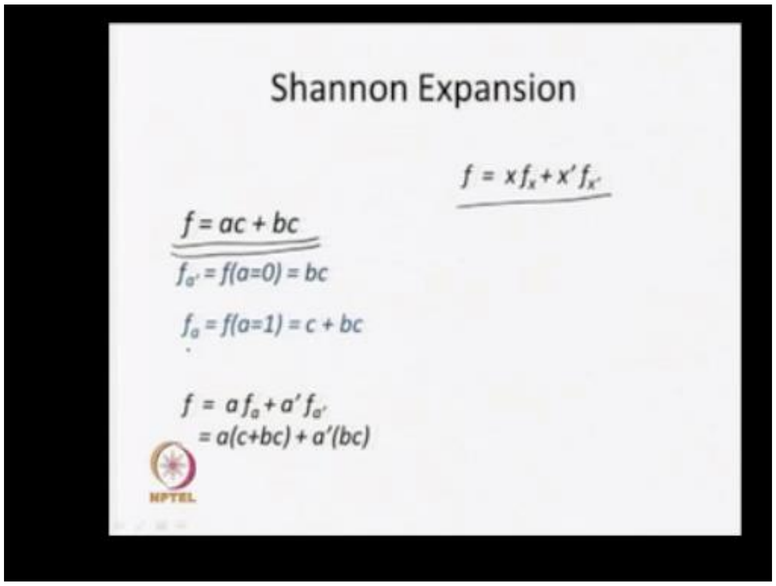
$F=xf_x+x'.f_{x1,}$ so what does it mean, so x is the finite variable, so you are expanding this particular function of x and we are having the evaluation value of x1, you are going to put x=1+x', that we are going to put the variable x' and the functional value when we are going to keep this value variable x=0, so with the help of the Shannon expansion you can represent and the Boolean function, and the BDD is based on this particular Shannon expansion. What is BDD? It is the complete data structure for the Boolean logic, so eventually we'll find that it is the compact way of representing our Boolean logic or Boolean function.

On the other hand you can say that, you can represent our state space also with the help of BDD, because state space can be represented with help of combination of some Boolean function. And one more advantage of BDD is that Canonical representation, so it is reduce all the BDD's on canonically, that means you are going to get a unit representation on the Boolean function, if you use our OBDD reduce our Binary Decision Diagram, first we talk about Binary Decision

Diagram, then we talk about Order Binary Decision Diagram and will come to reduced order binary decision diagram.

When we talk on the reduced order binary decision diagram you are going to get a unit representation of a Boolean function, that’s why this is called as a canonical representation or an Boolean function. So now I’m going to give one example of this particular Shannon expansion, so here just I'm going to talk about this particular function f=ac+bc, okay. That means f is function of key variable ac+bc and already I have mentioned that the Shannon expression is $F=xf_x+x'.f_{x1,}$ that means the effect say that the help of the even function when x=1, so this is basically we are going to have this function f.

(Refer Slide Time: 08:11)



Shannon Expansion

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

$\underline{\underline{f = ac + bc}}$

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

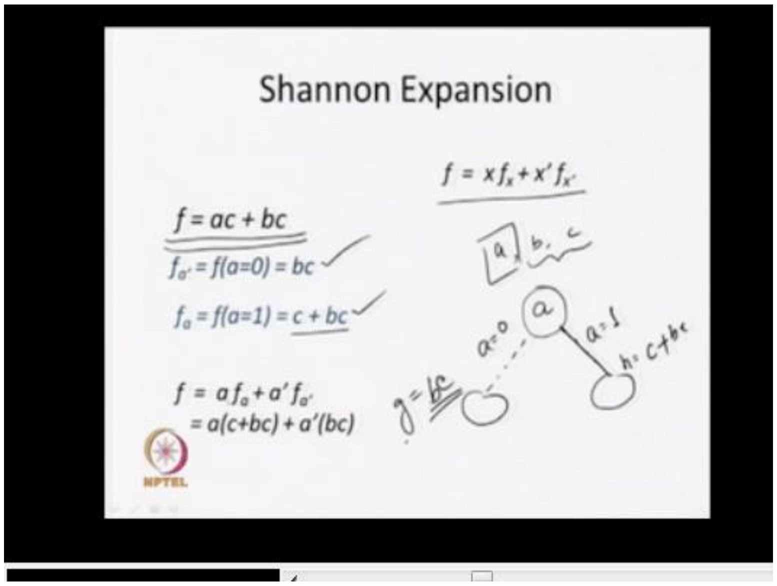
$f_a = f(a=1) = c + bc$

$f = af_a + a'f_{a'}$
$= a(c+bc) + a'(bc)$

So you are going to say fa is the function, well we are going to take x= a=1, so fa is going to be your c+bc, that which is x=1, then 1 that is equal to bc, similarly fa’ is the function where you think that the Lo (a) =0, so f(a’)=f1, a=0, so one put a=0, when these particular trans become 0, 0+bc, so eventually you are going to get bc, so f(a=1) is c+bc, f(a=0) is your bc, so that s why these particular function can be represented as your f(a*bfa+a), this into f(a), so this nothing but the a*f(a) is c+bc anf f(a’) is your bc, a’.bc, now which you can see that which you are going to implement having these three variable, a, b, c.

Initially you are going to look for the representation of a, a can undertake either 0 or 1, just I'm going to represent with something like that, this is the variable f, so this is may be you’re a=0, and this is you’re a=1, so in this particular case already I have the illusion of a, a either a will be equal to 0, or a =1, so what we need to evaluate which in the function is based on this variable b and c, because we have already taken the decision on a, this particular function when a=0, you see that the function will be your bc, that means I'm going to prove, about the function bc, I can say this is your g and when f=s and I'm going to say as c+bc is the hollow, so I can say c+bc is the hollow and this is the function.

(Refer Slide Time: 10:17)



Shannon Expansion

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

$f = ac + bc$

$f_{a'} = f(a=0) = bc$ ✓

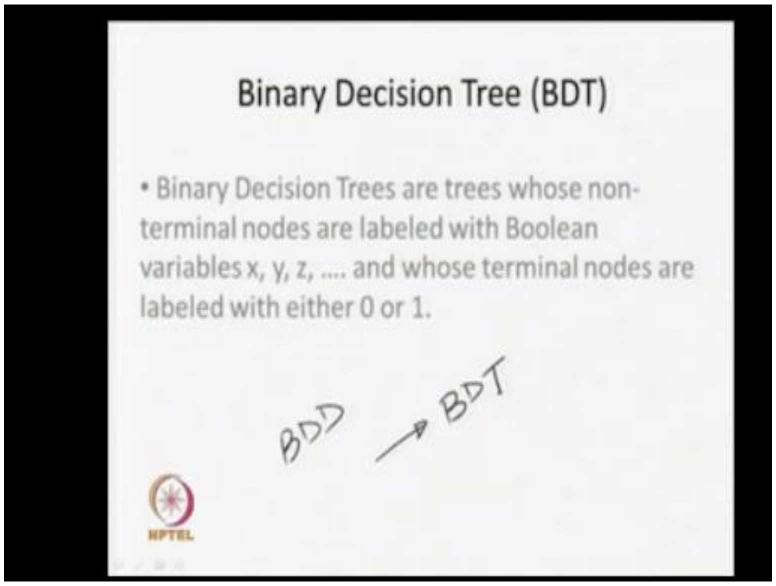
$f_a = f(a=1) = \underline{c + bc}$ ✓

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

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

Now my next on is to evaluate this function g and h, whether we use this particular Shannon expansion, if we consider the variable b, then what will happen, I will get b=0 and b=1, similarly for dates also we are going to talk about b=0 and b=1, eventually what function it will match, it will be on the fellow of variable c. So in this recursion you are going to get Shannon expansion and we are going to get some sort of a decision 3 or this decision diagram represented in this particular from.

(Refer Slide Time: 11: 09)



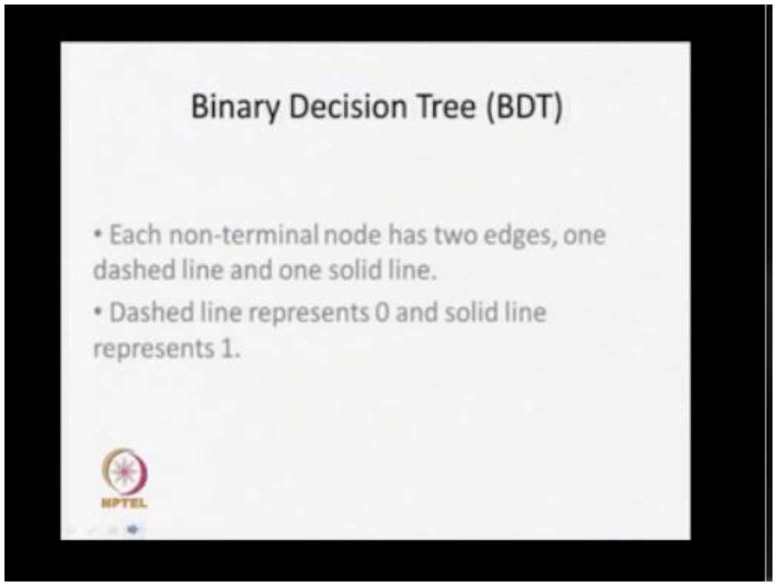
Binary Decision Tree (BDT)

- Binary Decision Trees are trees whose non-terminal nodes are labeled with Boolean variables x, y, z, ... and whose terminal nodes are labeled with either 0 or 1.

Now we are going to talk about this particular BDD, Binary Decision Diagram but before going to binary decision diagram, I'm going to talk about BDT, Binary Decision Tree, so this is a tree, we are using a binary decision trees are trees whose non-terminal nodes are Boolean variables, so we are having a function of x, y, z, then non terminal nodes will be represented by the particular Boolean variables, and the terminal nodes will be represented by the Boolean hallows or the Boolean constants, either 0 or 1.

So while we are going to draw BDT or BDD we use these particular circles to represent non terminal nodes and use this particular square box to represent terminal nodes, okay. So in case of non terminal node it can be level meter variables or depend what you say, in maximum variables will evolve this particular non terminal nodes and if it is the terminal node, either Boolean function will be 0 or 1, so I can say that if I represent write 1, it is the hallow of this particular terminal node is 1, so we are going to draw a three inductor in a terminal nodes and non terminal nodes, now level of non terminal nodes are the variables of the function and the level of the terminal nodes are Boolean function either 0 or 1, which will be the functional hallow of the given function.

(Refer Slide Time: 12:40)

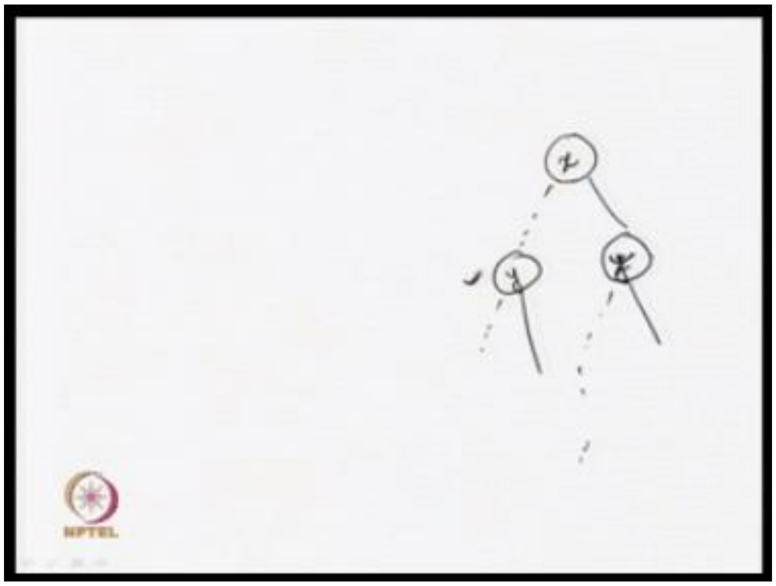


Binary Decision Tree (BDT)

- Each non-terminal node has two edges, one dashed line and one solid line.
- Dashed line represents 0 and solid line represents 1.

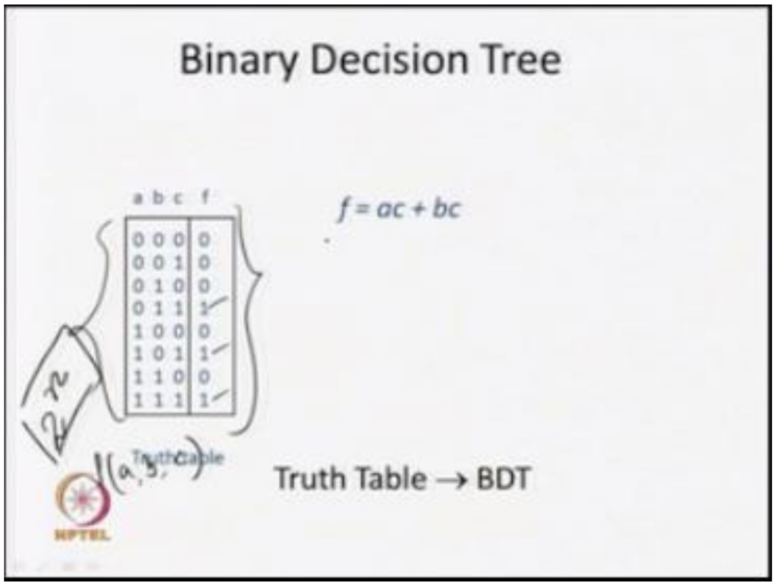
Now in BDD or BDT's, each non terminal nodes are 2 x's, it is having two s's, one s will be replaced by your gas line and second one will be replaced by solid line. That base line says that this is representing that valuation of x = 0 and solid line say that is the valuation of x with value x =1 so I know that we are having only two possible rates at x= 0 or x will be 1 so when I am going to treat the valuation of particular x it is having two out going as that this basically says it is value of x = 0 and x =1 is represented by this particular solid so I applied none terminal nodes are having two outgoing as this look here.

(Refer Slide Time: 13:35)



Now that is why you can say that so in our BDT you can something like something like that so this is a variable x this is variable y this is variable z so one variable so this one variable x= 0 we are having valuation x0 that we are going to look for what will be the value of y if variable of x = 1 then we are going to say what is the variable of y what you like it we are going to construct a complete tree it is a tree okay.

(Refer Slide Time: 14:14)



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 ✓

$f = ac + bc$

2^n

$f(a, b, c)$ Truth table

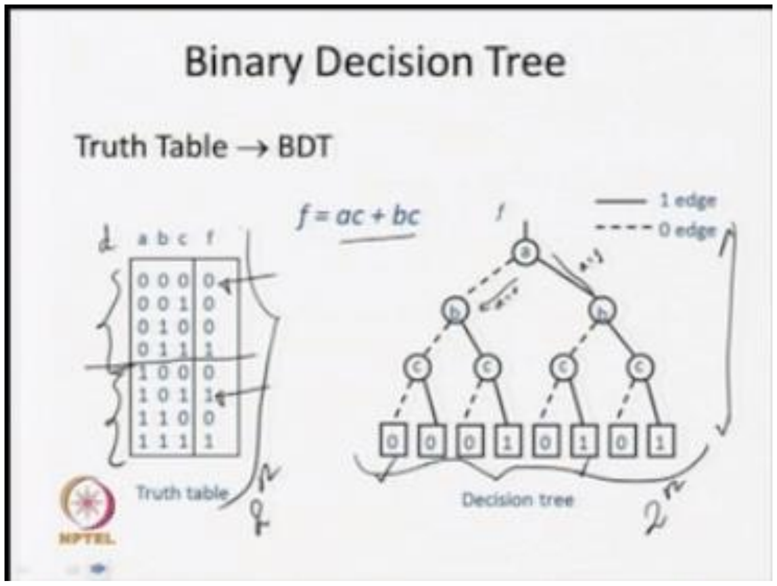
Truth Table $\rightarrow$ BDT

Now I am give an example this is it we are going to take a function f which involve tree variable so f is a function of this tree variable a, b ,c all of you know that the Boolean function can be represented that very well with the help of truth tables so I can have this particular truth table representation of my Boolean function f a, b, c so what you know it says that when a= 0 b =0 c=, 0 then my functional variable is 0 similarly if I say that if my a=1, b= 0 and c=1 then my functional value is 1 so in truth table we are going represent all possible combination of this tree variable and you know that what will be size of this particular truth table it 2^n we are having 2^n increase in this particular truth table if n is a number of variables.

So here I am having 3 variable only so I am having 2^3 which is 8 entries so these are 8 possible combinations that we can have with respect to a b and c because a can take either value 0 of 1 similarly b and c now this particular function I am representing with the help of your Boolean truth tables now we can construct the BDT for this particular function BDT binary decision tree or this particular function looking into this particular truth tables so if you look into it eventually you are going to get that it is having these 3 terms.

So if you look into then eventually what will happen 3 mean terms will be there and when I am reduce it then eventually I am going to get this particular function f = ac+ bc okay.

(Refer Slide Time: 16:02)



Binary Decision Tree

Truth Table → BDT

$f = ac + bc$

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

Decision tree

Now what we are going to do now we are going to get the BDT for this particular function so this is truth table I am doing this is the function f now I am having three variables a, b and c I have to say 1 a= 0 and a=1 so when a is = 0 then I am going to look for this particular point because a= 0 now b and b and c can either 0 and 1 when a= 1 then I will having these particular part of the truth table.

So here I am representing 1 none terminal nodes leveled with a that means I am going to take a decision on a when a= 0 I will follow this particular base line when a= 1 then I will follow this particular solid line okay now when I am talking a decision on a either coming this part or this part so remaining position will depends on b and c now again when I will come to b then b can either 0 and 1 similarly b can either 0 and 1 now eventually I am getting c so c will be either 0 and 1.

Since it is having only 3 variables so I have already taken a final decision now I am going to represent those terminal nodes which talk about this particular function algorithms now you just refers one this say when a= 0, b=0 c=0 then my functional value is 0 so that is why this terminal node is 1 so when I say that a=1 b=0 c=1 then my functional value is 1 because it says that when a= 1 b =0 c =1 then this is the functional value 1 so I am putting this one in this particular terminal.

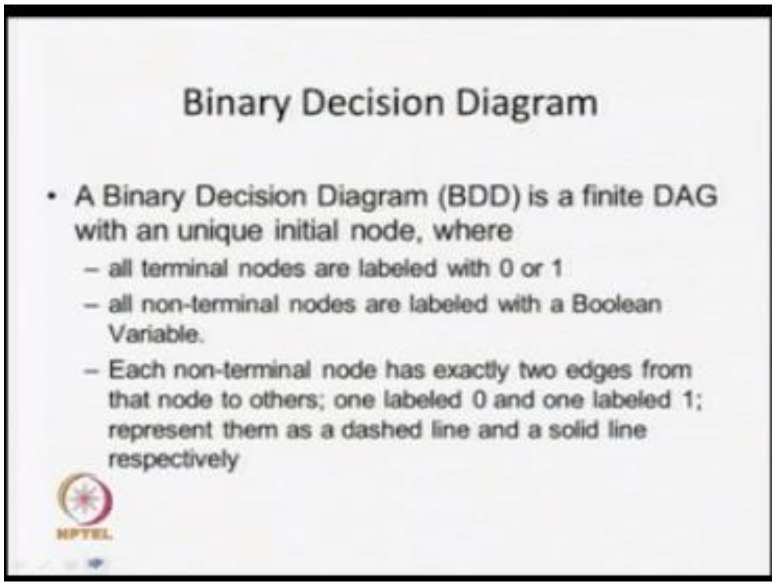
So you just see that by the above seen all those particular as this we can get the functional below of this particular been an function so that is why I can say that from truth table I can state your construct this particular BDT okay so this is the function f = ac + bc if you reduce this particular representation you are going to get this one and this is the decision tree that we are getting over here.

Now you just see that we are having a truth table I am getting an BDT binary decision tree what advantage we are getting by representing this particular truth table with the help of BDT if you see and if you look into it at such we are not getting any advantage because you see that when I am reprinting it with the help of truth tables then what happens 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 we will find that in leap node or terminal nodes we are having 2^n different terminal nodes because this is going to give the functional values for different combination and high of this particular 3 will and basically level and because you are having n variable so again the way we are having the exponential grow up truth table because if I increase by one more variable say d then what will happen we are going to 16 different n place 2^6 for that also this BDT the level of this BDT will be increased by 1 and we have to take decision on the and eventually we are going to get 16 different terminal nodes.

So access if you look in to that binary decision tree and your truth table we are not getting any point as in both case we are having exponential blow up so that is why this is the 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 okay.

(Refer Slide Time: 19:48)

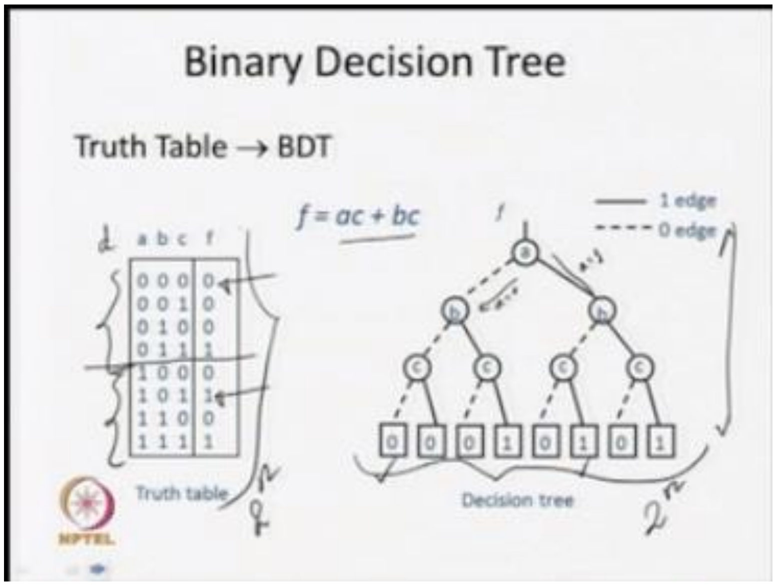


Binary Decision Diagram

- 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

It is very similar to your binary decision tree but now my representation will be graph instead of a tree I think you know a what is the difference between trees and graphs so we are going to get a graph now when we are going to talk about the BDT so this a DAG directed a cycle graph so because we are going to talk about the direction also in this particular case though explicitly.

(Refer Slide Time: 20:17)



Binary Decision Tree

Truth Table $\rightarrow$ BDT

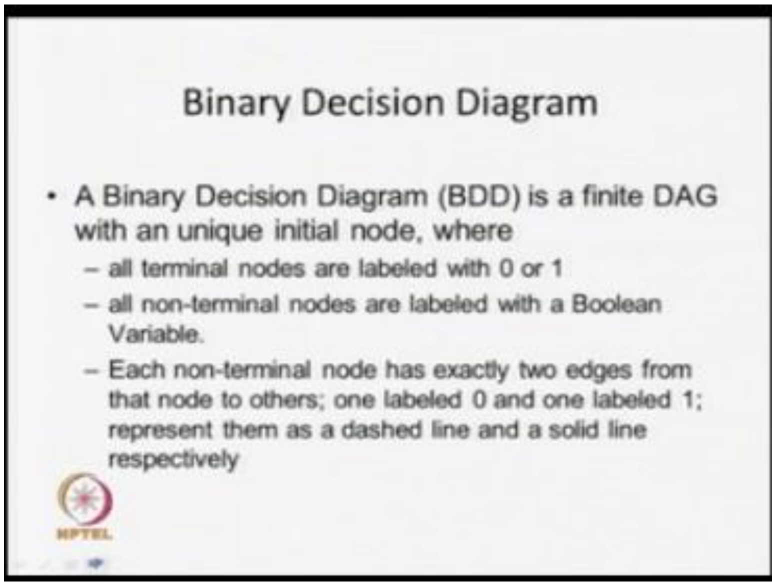
$f = ac + bc$

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

We have not mention it that means that means when a= 0 I will follow this particular path b =0 c=0 we are going to follow and when c=1 we are going to follow this particular so this is some sort of our directed path now that is why you are saying that what is your BDD this is a directed a cyclic graph.

(Refer Slide Time: 20:35)



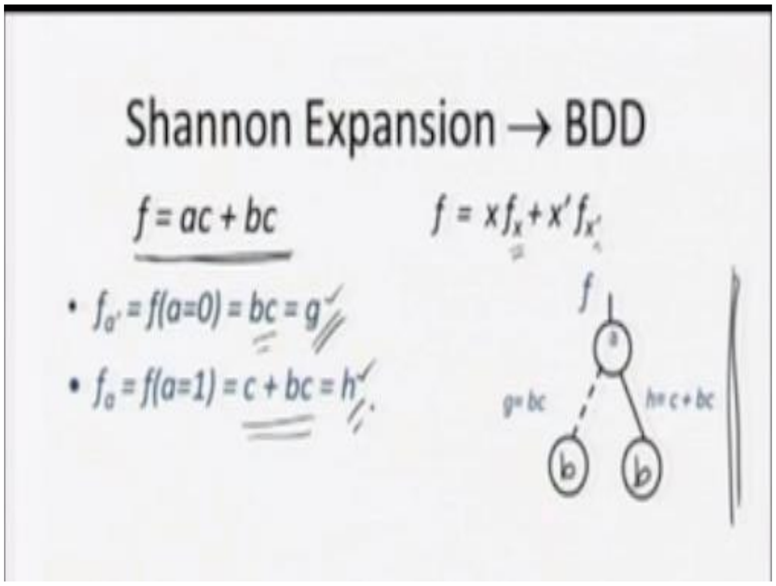
Binary Decision Diagram

- 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

That means it is directed graph but it should not have any cycle and a similar way now we can talk about BDD also so what happens in BDD we are saying that we are having terminal nodes and non terminal node in BDT also we will have terminal nodes and none terminal nodes in case of non terminal nodes that will be leveled by your Boolean variables terminal nodes will be leveled by Boolean values 1 or 0 and every non terminal nodes are having two outgoing as this 1 as will be represent by your based line and second one will be represent by soil line.

Base 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 say that this is similar to your BDT only but it is now we are going to get a graph in discrete of a tree.

(Refer Slide Time: 21:31)



Shannon Expansion → BDD

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

- $f_{a'} = f(a=0) = bc = g$
- $f_a = f(a=1) = c + bc = h$

Now what is the primitive of our BDT primitive BDT binary decision diagram so in this particular case we are going to get 3 primitive binary decision diagram and with the help of these 3 primitive binary decision diagram we are going to construct the binary decision diagram of any Boolean function so this is your B0 it is a terminal node leveled by 0 it say that the Boolean value is = 0 okay so this B0 is representing this particular BDT which represent the Boolean constant 0 similarly we having 1 binary decision diagram BDD b1 which represent the Boolean constant 1.

That means this is the such a box within one over here so it is a primitive BDD it says that this the Boolean constant 1 B0 say that this the Boolean constant 0 and another BDD we are having which represent the Boolean any Boolean variable so if we leveled by x then we are going to say this is the BDD for the Boolean variable x and we say this is your BX so what it says that x it is having 2 outgoing as this 1 test second one is your solid base line is pointing towards the Boolean constant 0 and solid line is pointing towards the Boolean constant 1 that means 1 x = 1 then my functional value is 1 so this is the representation of BDD representation of our any Boolean variable x so these are the three primitive with it is and help of this three primitive with it is we are going to construct that BTC is of any event Boolean function.

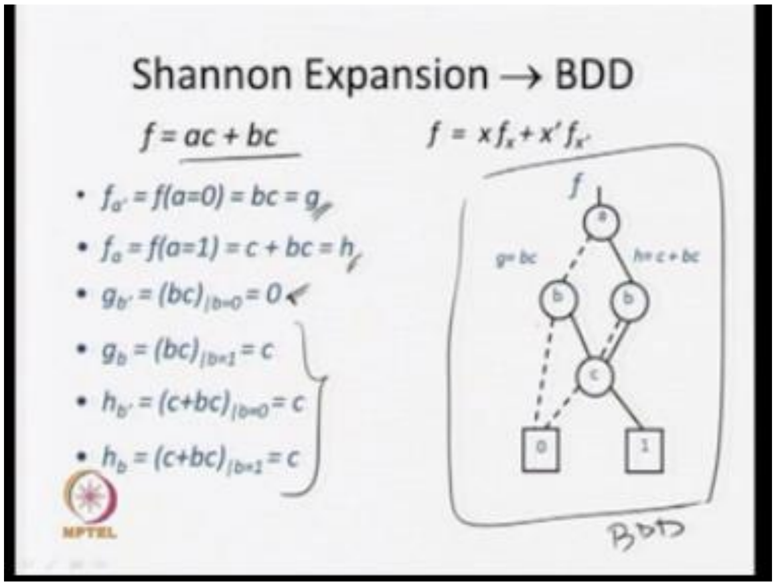
Now just say what I am going to talk about, now we are saying that the BDD is nothing but or BDD can be constructed or represented with the help of Shannon expansion again consider this particular same Boolean function F = ac + bc and the Boolean Shannon expansion is your f = f(x) f(x) + x ‘fx ‘that means functional value are there point x is theta’ = 1 and your functional value are there 1xc = 0.

That means I can say that already I have said that f(a) =0 will be your BC because A = 0 I am going to have BC so I said that this is your BC and when FA = 1 then f(a) is your c+ bc because a = 1, 1 x c = c not a functional depends on c so c + bc so one I am saying that this is a sub function g and second is sub function a, so this particular information we are going to represent with the help of this particular partial BDD.

So we are having a familiar notes A either A have value 0 or A have value 1 so when A = 1 then I am having the sub function that we have to evaluate this c + bc this is H and when A = 0 then sub function we are having these which is your BC so this is the partial BDD for this particular given function after taking that this is your Non A now what we have to do next now either we have to talk this is a non B or this is a non C.

Now if I am going to take this is a non B then I am going to level these two things with B okay, so in this particular case now again I will apply this particular Shannon expansion for this particular function G or this particular function H these where we can construct our BDD also.

(Refer Slide Time: 25:15)



Shannon Expansion → BDD

$f = \underline{ac + bc}$ $\qquad f = xf_x + x'f_{x'}$

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

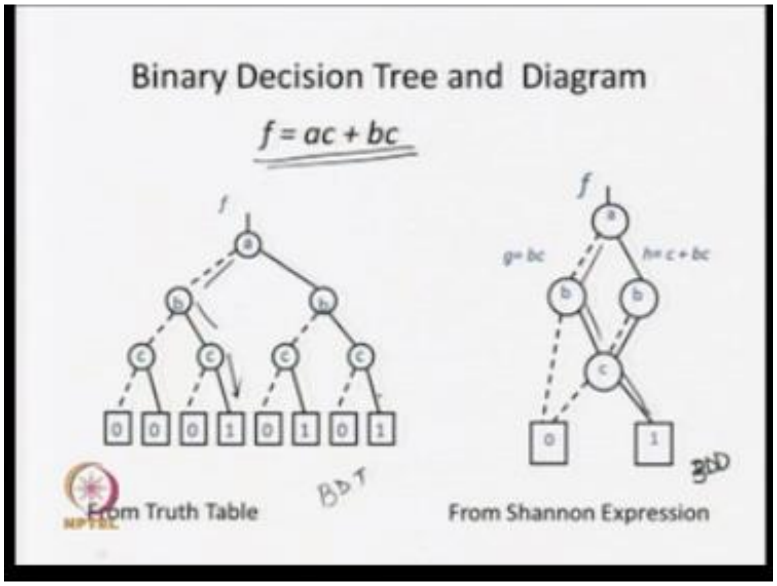
So eventually now what will happen you just see now we are going to apply this particular Shannon and expansion for all the variable so we are getting G and H okay here I may be having the function G and here I am having the function H now we will apply Shannon expression on expansion on G and H so if G, if B = 0 then what will happen the function of value is 0S and when V =1 then GB will be your C.

Because if it is your B = 1, 1.c will be C similarly in case of H if it is your B = 0 then it will be equal to C and when B = 1 then it will be C + C = C now you just see that now what we are trying now when B = 0 the functional value is 0 so we are writing during this particular S I am saying that when B =0 function al values if 0 when B =1 then functional value is C so that is why I am drawing this particular C.

Not C and pointing from B to C with the help of solid light similarly in case of your function your H when B = 0 the functional value is 6 so this base line is coming to C and when B = 1 then again the functional value let us say that is why this solid line is coming to this particular node C, Now we are having this particular three function one is already getting the constant now you just see that.

This sub functions are depends on your C if C= 0 functional value is 0 C = 1 the functional value is 1 so further from C I am having this particular base line which is coming to 0 I am having this particular solid length which is pointing towards this particular terminal node 1, so eventually I am getting this particular BDD okay, so this is the BDD of this particular given function ac + bc now you just see what we have got now.

(Refer Slide Time: 27:32)


Binary Decision Tree and Diagram

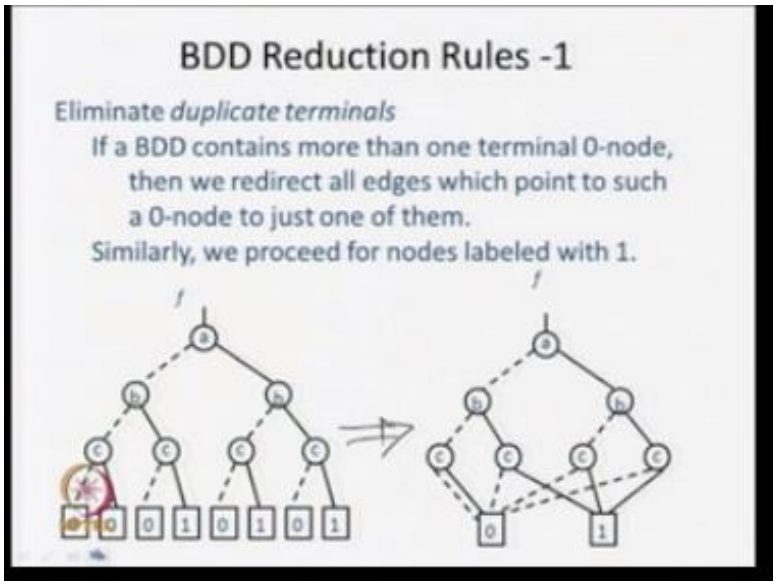
$f = ac + bc$

We are taking this particular function f = ac + bc so from truth table I am getting this particular diagram which is your BDD basically binary this is one tree okay and by x using this particular Shannon expansion I am getting this particular diagram with this I am going to say this is your BBD binary this is one diagram, so both this diagram are now representing the same Boolean function, okay.

Because you can traverse this particular graph and gradually you can find the valuation of this particular function say if A =0, B = 1, C = 1 functional value is 1, so when A = 0, B =1, C = 1 my functional value is 1 so from both this diagram we can see this, so here we are having BDD and we are having BDD but both are depending this dimension, so here I am going to get some sort of your compact representation of this particular given function, okay.

Now they are equivalent both of them are representing the same function so whether can you achieve this particular BDD from this particular BDD it may be possible, so see what we are going to do.

(Refer Slide Time: 28:51)



BDD Reduction Rules -1

Eliminate *duplicate terminals*

If a BDD contains more than one terminal 0-node, then we redirect all edges which point to such a 0-node to just one of them.

Similarly, we proceed for nodes labeled with 1.

So we can use some reduction rules and with the help of this particular reduction rule we can try to reduce our BDD or you can try to reduce or BDD 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 this, say if we are having more number of terminal nodes which are going to represent the Boolean constant 0.

Then we can represent this particular terminal nodes with one terminal nodes if we are having several terminal nodes which are representing Boolean constant 1 we can represent those particular several terminal with the help of one terminals which will represent this particular

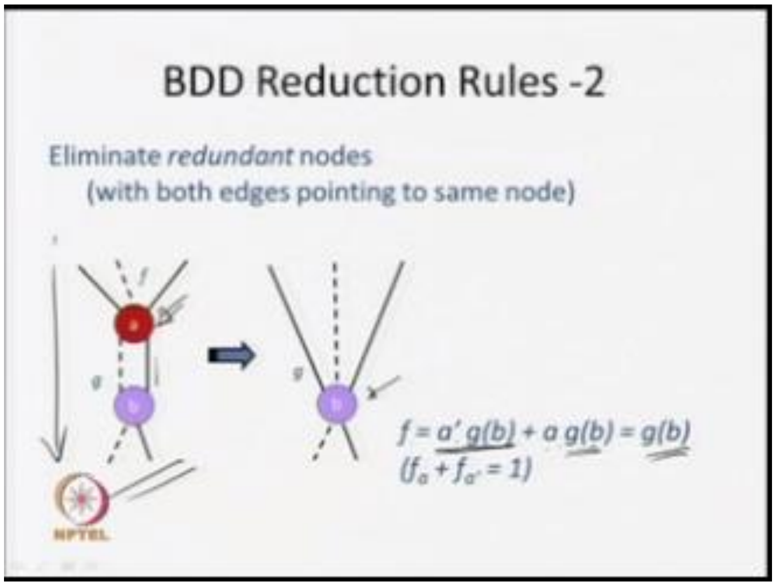
Boolean rule, so that is why the past rule we are saying that eliminate of this particular duplicate terminal.

So if we apply this thing to our BDD then what will happen you just see, so in case of this things what will happen now all terminals node will be combined to one and that incoming as well will be redirected to this particular new terminal nodes, okay. Now what are that you just see that this the BDD that we had now we are having several terminal nodes some are represented by 0 or some are represented by 1, now all 0 will be combined and represented by 1 terminal nodes and all 1 will be combined together they will be represented by one terminal nodes 1.

And whatever incoming as is the coming to those particular terminal nodes they will be redirected to this particular terminal node, so after eliminating this particular duplicate terminals what I am getting from this BDD we are getting some sort of BDD over here okay, so now at least we can see that number nodes are less here because instead of 2^n nodes terminal nodes we are getting only two terminal nodes.

So here we had 8 terminal nodes because we are having three variable so this side terminal nodes will be represented by 2 terminal nodes so it is for any function if you are having a function of N variables then in BDD we are going to get 2^n terminal nods but in BDD we can represent this 2^n terminal nodes by only two terminal nodes so give raiser must be reduction, okay. So you need to have only two terminal nodes. When we talk about BDD okay this is one reduction rule say eliminate of duplicate terminals.

(Refer Slide Time: 31:17)



BDD Reduction Rules -2

Eliminate *redundant* nodes
(with both edges pointing to same node)

$f = \underline{a' g(b)} + a\, \underline{g(b)} = \underline{g(b)}$
$(f_a + f_{a'} = 1)$

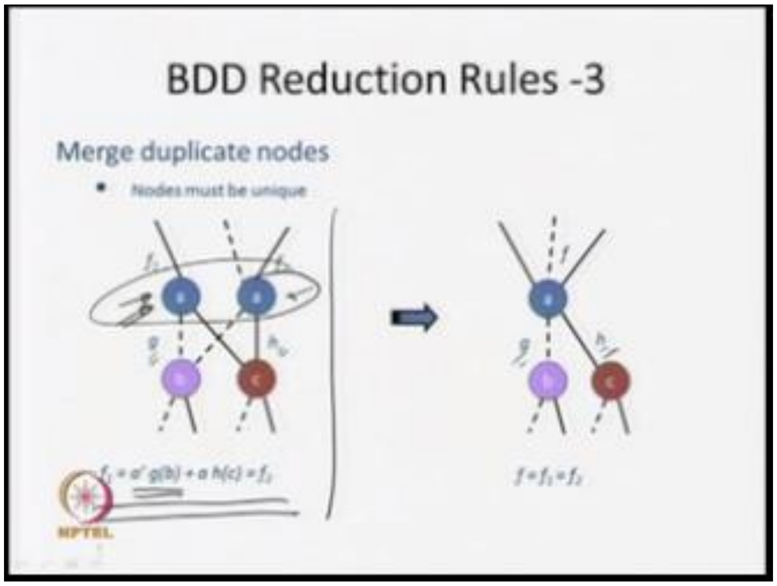
So second reduction rule we are going to say that eliminate of redundant nodes that means, if both as a pointing to the same nodes we are going to see why do we have to talk about the redundant node so eliminate of redundant nodes what we are going to say about these things now you just. See that we are having this particular partial BDD's just consider this one say we are taking some decision we are coming to with when A = 0 it is pointing to this particular non terminal nodes.

When A = 1again this pointing should this particular terminal non terminal nodes okay so now you just see that when A = 0 then 0 into the functional value depends on B so that is why I am saying that when A = 0 then As x Cb that mean many person similarly when A = 1 again we are getting that A .GB because already we know A =1 and this is the remaining portion that we have to evaluate so this is basically what happens you say that.

A that a'g(b) + ag(b) which is equal to g(b) that means now you just see that what about function that I am going to have it is basically redundant or irrelevant on that decision of this particular A in this particular part so in this particular part so that is why what I am saying that we are going to eliminate this particular redundant nodes so if we are having such type of redundant nodes where both 0 and 1 is juts pointing to the same sub function then we can eliminate these things, so we eliminate this particular nodes and all the incoming as just will be directed to this particular node. So in this particular way we can eliminate some of the redundant nodes okay, after eliminate this particular redundant nodes my size of your BDD will get reduced, okay.

That means we are going to ask the compact representation of our Boolean function, so first redundant first reduction rule is your elimination of duplicate terminals second reduction rule is your eliminate of your redundant nodes. Now may have one more rules.

(Refer Slide Time: 33:39)



BDD Reduction Rules -3

Merge duplicate nodes

- Nodes must be unique

Which is your reduction rule tree which is nothing but call your merging of your duplicate nodes so if we are having some duplicate nodes then we can merge them together and we are going to get some sort of simpler or we do us 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 two nodes now I have to take the decision on A so fa we are evaluating and in this particular node I am going to take the decision on h.

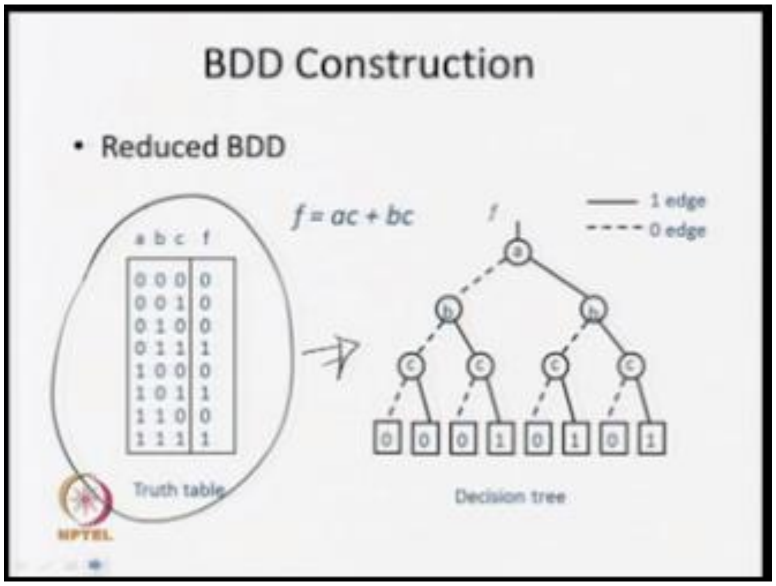
So what will happen in this particular case if h=0 then I am coming to this particular node b when a=1 I am coming to this node c that means this is a thus into gb so this is basically coming to this particular dash line is going to say a′.gb and when I follow this particular solid line then I am going to get a.hc that sub function that we are having in this particular node is your h so a,hc 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 what about we are involve with the nobody you say f, now I can use just see that when a=0 we are coming to that same node b and when that means we are going to look for the same function g and when a=1 we are coming to the same function h, so in this particular case the partial evaluation of the given function for f1 and f2 both these function are same.

So we are saying that f1=f2 which is your a.gb+ahc so in this particular case you can say that these are basically some sort of duplicate nodes that we have in our BDD, so what we can do now we can eliminate these merge these two nodes where single nodes, so in this particular case both f1 and f2 will be equal to your f so we can say that merging these two nodes to work and these are the sub function that we are having when it is equal to a=1 we are going to evaluate this sub function h and when it is equal to 0 then we are going to evaluate this particular sun functions.

So like that we can merge these particular duplicate nodes and eventually we are going to get a some sort of your reduce or compact representation of our given Boolean function, okay.

(Refer Slide Time: 36:11)



BDD Construction

- Reduced BDD

$f = ac + bc$

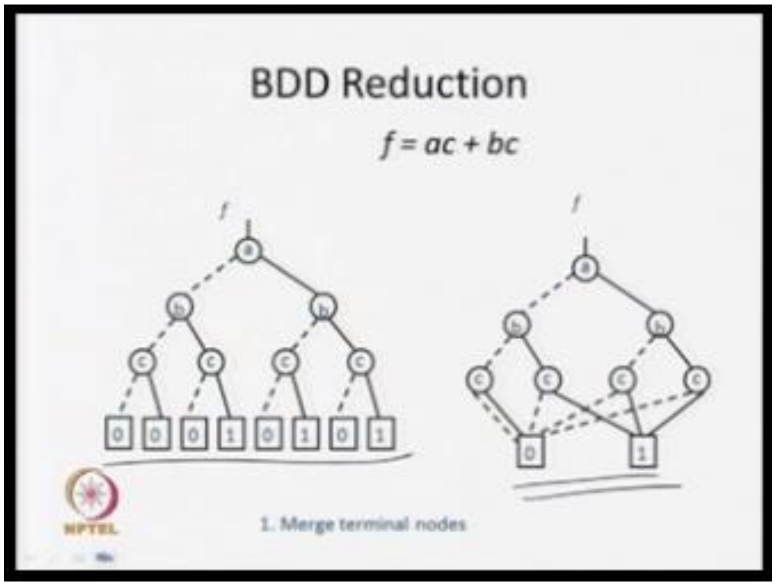
a	b	c	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Truth table

Decision tree

So we can have these particular tree rules, now you just see that first we are getting this particular truth table we 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 an exponential grow up.

(Refer Slide Time: 36:31)



BDD Reduction

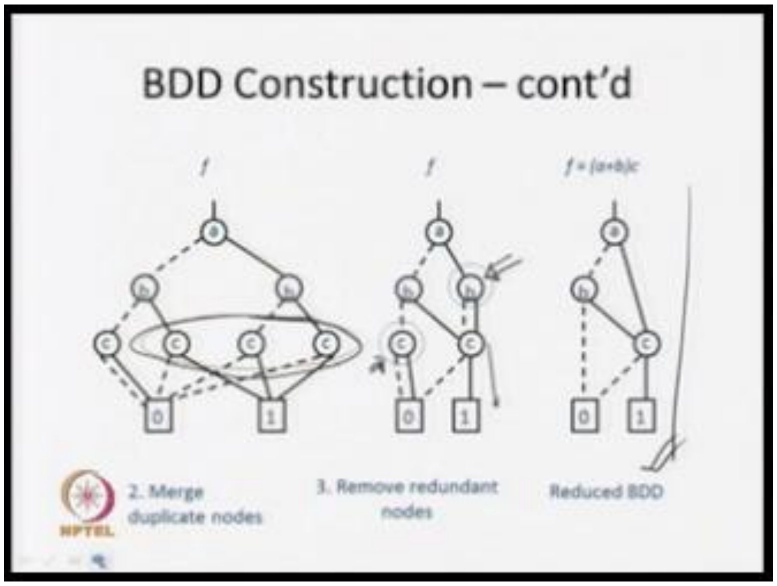
$f = ac + bc$

1. Merge terminal nodes

Now what we are going to see we are trying to reduce it okay, when we are going to reduce it according to the first rule we are going to merge the terminal nodes so we try to merge those particular terminal nodes so we are getting two terminal nodes one with 0 and second one is 1 and we are going to redirect those incoming as just to these two terminal nodes so after application of first reduction rule we are reducing the terminal nodes to only two, okay.

So you may have 2^n terminal nodes in binary decision tree but eventually we are going to get two terminal nodes in our binary decision diagram with the help of first reduction rule merging up terminal nodes we are getting only two nodes instead of having these particular eight nodes, okay.

(Refer Slide Time: 37:22)



BDD Construction – cont'd

Okay, 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 this particular tree nodes c when c=0 the function value is 0 when c=1 function value is 1, so NSR of this tree 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 this tree nodes to one particular nodes c and the incoming as just will be redirected.

So from b it is one coming to c so b=1 coming to c b=0 coming to c b=1 coming to c so these are to dash line as a line we have so lid 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 why this particular reduction rule merging after duplicate nodes, okay. Now so after that we are going see whether we are having any redundant nodes or not.

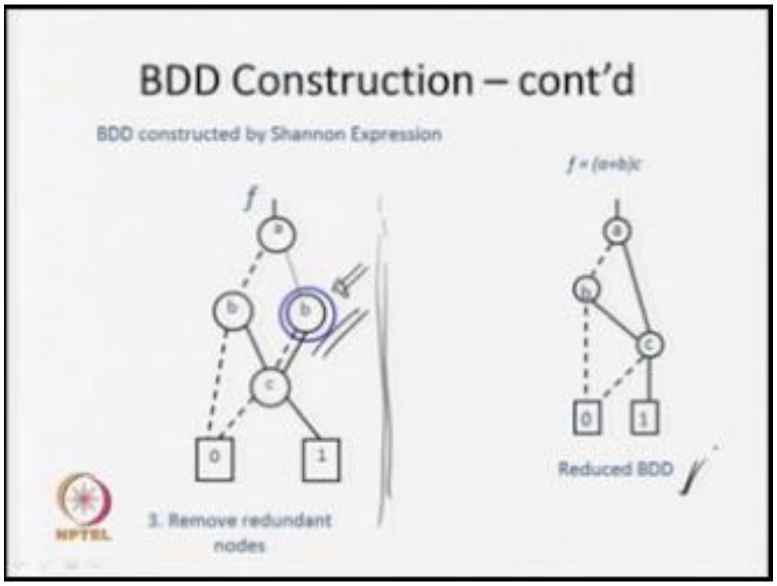
So in this particular case you just see that if you look these particular node b what we are getting when b=0 I am coming to c when b=1 we are coming to c that means whether b=0 or 1 it is immaterial we are going to evaluate these particular sub function. Similarly when I look into this particular c I can if c=0 the functional value is 0 c=1 functional value is 0, so again it is immaterial on the decision of this particular c.

So what happens now we are going to get this particular diagram BDD where we are removing these two redundant nodes, so eventually we are getting these particular BDD, okay so we have applied your this reduction rule removal of your non terminal nodes merging of duplicate nodes and remove up redundant nodes after removing of redundant nodes it may happen that some duplicate node may arise again one here.

So what we are going to do we will again apply this particular merging of duplicate nodes after merging of duplicate nodes what will happen again some redundancy may arise then again we are going to remove those particular redundant nodes. So in this way what happens we are going to repeat this particular two rules and try to eliminate more and more redundant nodes and merge duplicate nodes, so eventually we are coming to this particular BDD, okay.

So you just see that this is what we have talked about here that we are looking into the BDD and we are applying those particular reduction rules and eventually we are coming to a representation and you just see that no more reduction is possible because abc these are the tree nodes there is no redundant nodes because all 0s and 1 has a going to different direction we do not have any duplicate also because the levels are different so since the levels are different the question of duplication is not arise over here, okay. Now this is from BDD we are coming to this particular reduced BDD.

(Refer Slide Time: 40:40)


BDD Construction – cont'd

BDD constructed by Shannon Expression

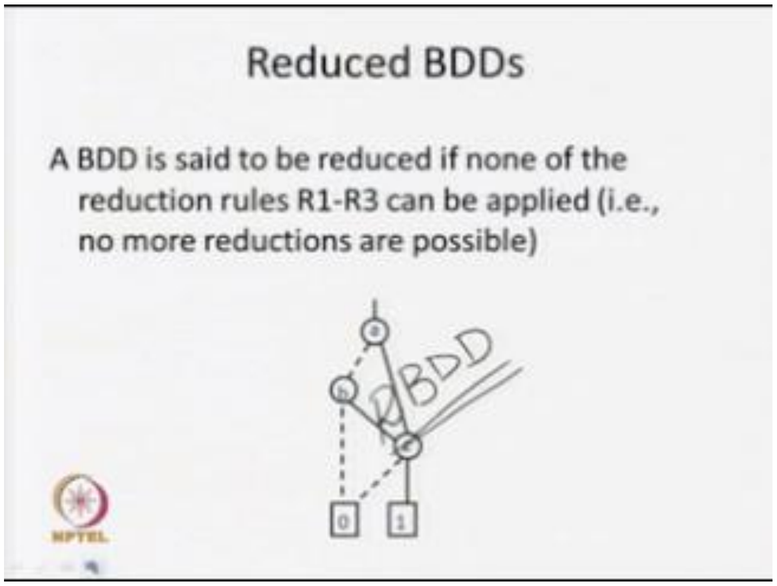
3. Remove redundant nodes

Similarly we have the similar same function we have taken and we have construct the BDD with the help of your Shannon expression so this is we are constructing this particular BDD with the help of Shannon expression so after constructing those particular BDD we will see whether it can be reduced or not, so first rule talks about the redundant removal of your terminal nodes so since we are using that Shannon expression so they will not be any your redundant terminal nodes because we are going to have only two terminal nodes.

So basically we have to look for the other two rules merging of duplicate nodes and removal of your redundant types, so when I am coming to this particular diagram BDD we have found that this particular node b it is doing some sort of your redundant based on it because b=0 and 1 is going to have the same sub function. So we can remove this particular redundant nodes and ultimately we are coming to this particular BDD, so whatever you are going to do if you are starting from your BDD or apply Shannon expression to get it your BDD eventually you can apply those particular reduction rules and finally we come to a particular BDD, okay.

So we are coming to this particular BDD and no more reduction is possible over here.

(Refer Slide Time: 42:07)



Reduced BDDs

A BDD is said to be reduced if none of the reduction rules R1-R3 can be applied (i.e., no more reductions are possible)

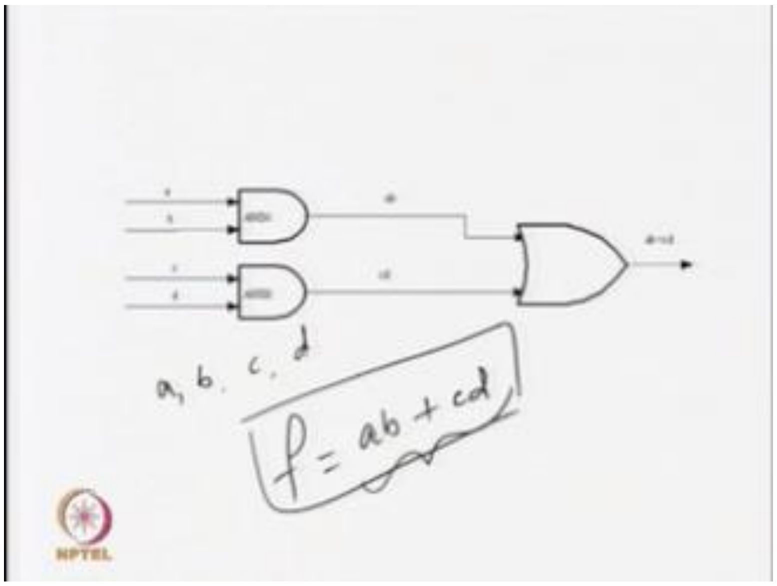
So by looking into these things what happens now we are going to have the notion of your reduce BDD, reduce binary decision diagram which is written as your RBDD reduce binary decision diagram when you say that a binary decision diagram will be a reduced one we will say that it will be a reduced one okay, if none of the reduction rule R1 and R2, R3 can be applied, okay so we are having three rules R1, R2 and R3 if none of these reduction rule can be applied any more that means no more reduction is possible.

So if no more reductions are possible at that point we are going to say that this is the reduce BDD we are getting it. So we say this is RBDD or say reduce BDD, okay. So this is the notion so when we are 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 our BDD representing our bullion function with the help of the decision diagram which is known as you BDD binary decision diagram it can reduce to get a reduce BDD and in reality we are found that if most of the cases we are going to get a complete representation of our bullion function.

So initially what happens when we talk about BDD you will find that we are having a exponential grow up we are having total 2^n numbers of terminal notes if we are having invariable but we can applied a xenon expansion also to constrict the BDD for a given function in one we are using the xenon x function at least w can assume that we are going to get only two terminal notes and we may have some non daminal notes.

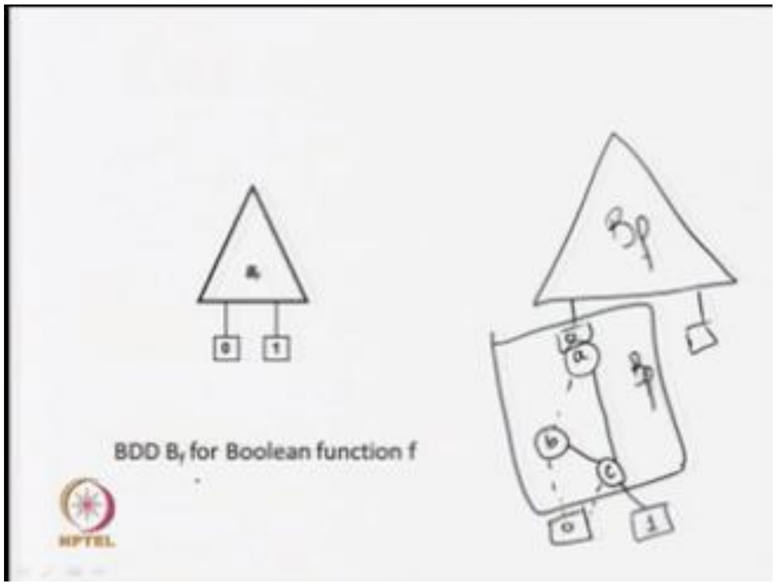
When we are getting the initial BDD with the help of xenon x expansion then we have try to use the reduction rule removal of you reddened test and margins of duplicate non daminals by repeated the applied of this two rules we are going to get a compact with the presentation above given bullion function, so when we are having a compact presentation of a bullion function we are going to use this things of representing of bullion function and it is going to help us to tickle with the states for expression problem will see how we are going to represent our state space with the help of binary decision right.

(Refer Slide Time: 44:51)



So basically we are going to have this now say if I am going to have any bullion logic cycle just for expansion I am saying that I am having two angers over here and one are gets the input variables are your a b c d so if is the 4 input variable and if this is the circuit given to me then what happen that I can like the function for this particular function is your basically this is nothing but some of product it means a x b+ c x d you just see that if we are having any bullion circuit analog circuit we always write the bullion expression of this particular circuit and once we have this particular bullion expression then what will happen always you can try to contract the BDD for this particular bullion function okay. You just see once we have constructing the BDD then we can use those particular BDD in our work okay.

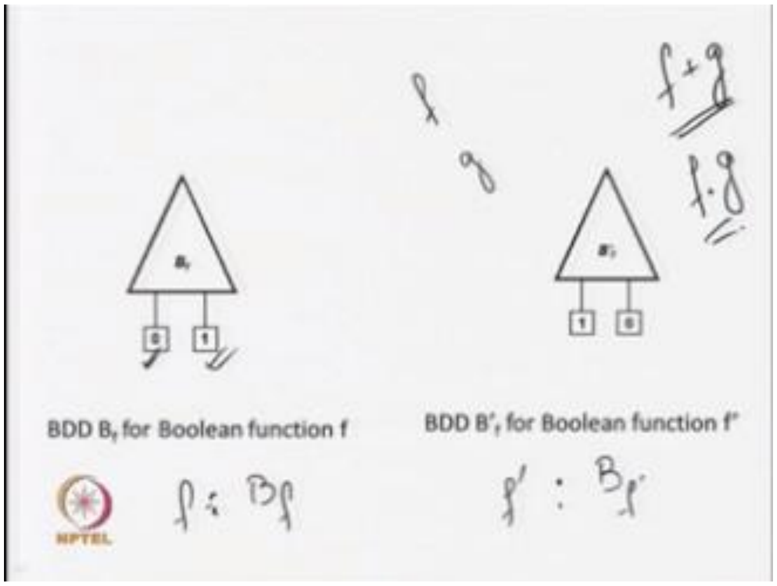
(Refer Slide Time: 45:51)



Now you just see now we are having some BDD this we are representing this particular BDD something like that in symbolic where this triangular we are writing that this is the BP, BDD of the bullion function I have so just look in to the previous examples I am having say A like the this is b if b = 0 the I am doing the function value as 0 if b = 1 I am coming to see if a = 1 c = 0 functional value is 0 and c =1 my functional value is 1 so this is the BDD that already we have seen that this is the simple BDD.

Now this particular B generally we represent with the help this particular triangle and we say that this is the function BDD for function Bf and eventually it is having this particular two terminal notes 00 and 1 so this is basically we are representing binary decision diagram BF this is the graph that we have and eventually having this two terminal diagrams. So if we are having a BDD of Bf then you can do some other operation on this particular BDD also.

(Refer Slide Time: 47:14)

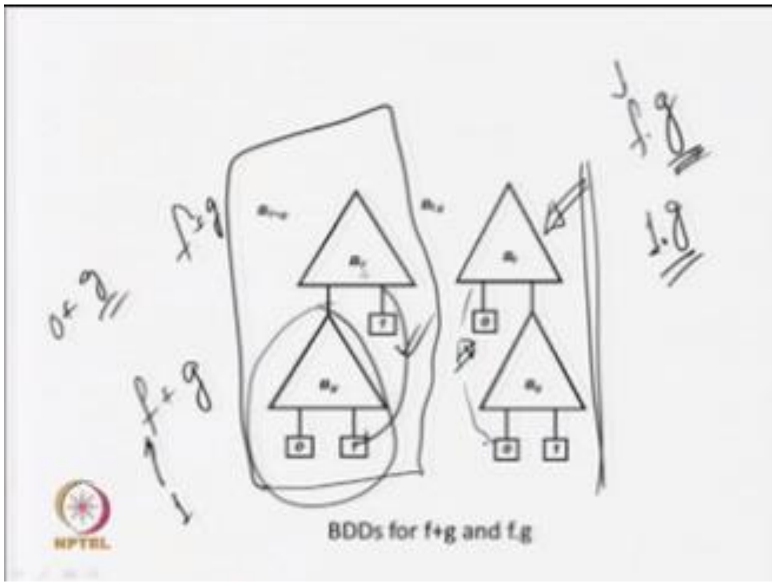

f + g
f · g
f
g
0
1
1
0
BDD B_f for Boolean function f
BDD B'_f for Boolean function f'
f : Bf
f' : Bf'
NPTEL

Now you just see that if I am having a function BF say F I am having some function then the complement of YF is your written by your F1 okay now if the BDDC of this function is your BF then I am going to get the BDD for this particular complement of this function BF now if we are having the BDD of a particular function to get the complement is very simple because w know that of it is complement is basically if x = 0 and complement of x is 1 if x = 1 that complement of x =0 so what we are going to do simply we are going to inter since this two particular terminal notes so if it is 0 then 09 will be replace by 1 and 1 will be replace by 0 okay.

So this is the way we can said that if I am having a given bullion function if we know that BDD representation of the particular bullion function then complement can be very easily find out okay what just simply inter since the terminal note 0 will be represent by 1 and one will be represent by 0 so this is very simple, secondly if we are having some bullion function generally then what e are going to do we are going to perform some bullion operation also say if I am having one function f And one function g after same variable of different variable then what happens I can perform bullion function f + g or I can perform bullion function F + G okay.

So one is conjunction second one is j, now say if I am having two bullion function we can do it 9f I give any two bullion function we can do it if I give any two bullion function to you always you can find out the some of this two bullion function or product of this two bullion function this junction or that, now if I give you the repeated representation of this two functions instead I give you the original function in will give you the BDD representation of this two. But again can you construct f + c or F dot G again if you look in to it this is very simple a simple way you can do it.

(Refer Slide Time: 49:12)



BDDs for f+g and f.g

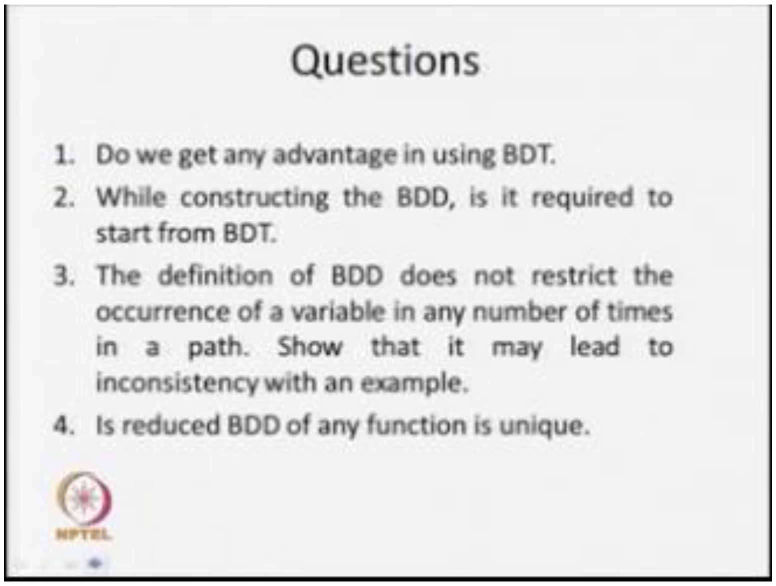
You just see that it is basically representing F + G that is first diagram why I am saying that this first diagram representing f+ g because f + g if you said that if anyone is one then output is one so first algorithm g this is your if you think this is you VF if F =1 then what will happen f + g will be one so if f = 1 I am just taking this decision this is my function value is one now similarly if you know f = 0 so 0 + G so this a is basically depends on G if g = 1 my functional value is one so g = 0 my functional value is 0.

So that means in this 0 terminal notes just I am going to plug in this particular BG so this particular whole diagram is going to give me f + g so just see that constrict and all f + g is also very simple so f + g is all the way wrong if f =0 then the value of f dot g will be =0 if f = 1 that the function value will depend on your value of g because f dot g f =1 dot g so basically if g =0 then my value is 0 if g =1 my value is 1.

So that why you know what I am doing first BF is representing this particular bullion function f if f is 0 so this is the result my function value is 0 if f is one then what happen I am going to a look for the value of this so I will just simply plug in this particular BDD of BG for function G in the one terminal notes of this particular BF and eventually it is going to get this particular. And later on what I can do? Again since I can say that this to prove can be mark so that mean I can re direct this particular s to this things and I can re direct this particular s2 this particular one.

Again after having this two BDD we are getting NBDD if it required I can go for detection of this particular BDD okay now you just see that we have seen one particular refer striation known as your BDD binary decision diagram we can represent our bullion function with the help of your BDD binary decision diagram and I can mention that our form result also we have found that in most of the cases we are going to get a compact with representation or any given bullion function.

(Refer Slide Time: 51:36)



Questions

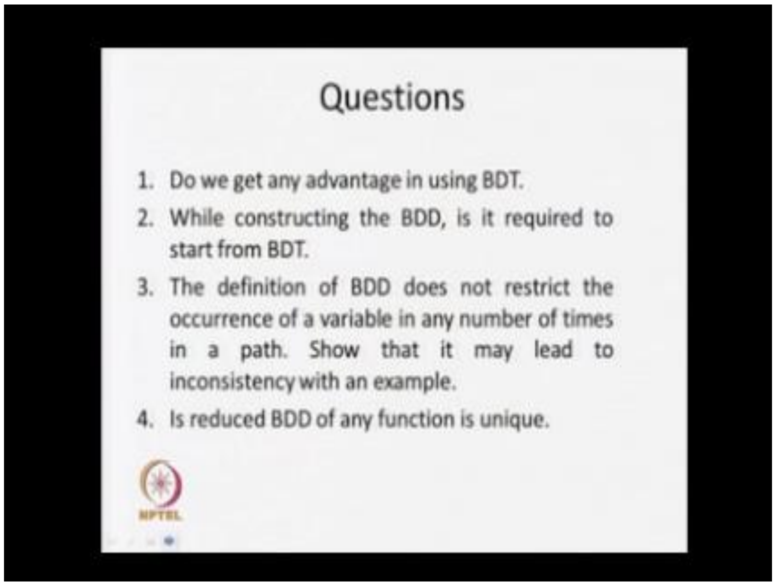
1. Do we get any advantage in using BDT.
2. While constructing the BDD, is it required to start from BDT.
3. The definition of BDD does not restrict the occurrence of a variable in any number of times in a path. Show that it may lead to inconsistency with an example.
4. Is reduced BDD of any function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

Now just look for some questions very simple question I am giving to you first question I am saying that do you get any advantage ion using BDT binary decision tree so we are saying that we are going to look for a data structure which is give as a compact representation of your bullion function I have mentioning about BDT binary decision tree now whether do you get any advantage of you BDT so if you look in to in it the contraction of BDT and a BDT that it looks like excess you are not getting any advantage why?

Because it is simple mapping or simple transmission of my true table to BDT in truth table we are having a problem with your exponential grow up if I am having N variables I am going to get two to through go N defined a n trace so in BDT also we are going to get to by N terminal notes and whether level will be is equal to N.

So the simple example is see that if I am going to have a function which is having 8 variables then the total number of Np is not suitable will be your 2 8 which is your 256 when we are going to look in to the BDT we are going to get 256 terminal notes and the level of this particular BDT will be 8 because for each variable we are having one level so XL we are not getting any advantage but for crating the simultaneously what is BDT and but from BDT we can use the reduction rule and we can get the BDT now second question is that while constructing the BDD is it required to start form BDT

(Refer Slide Time: 53:05)



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.

Again I think it is very simple and, and it after listening to my lecture you can very well set up this new. Because BDT can be pure as a repeatedly application of ours on an expansion okay, that means one who constructs the BDT and apply the deduction rule we have three deductions rule this is your eliminate of your terminal. Typically terminal notes or removable of resultants and merging of your duplicate non terminals we can apply those things and we can find out those regular BDT or radio BDT.

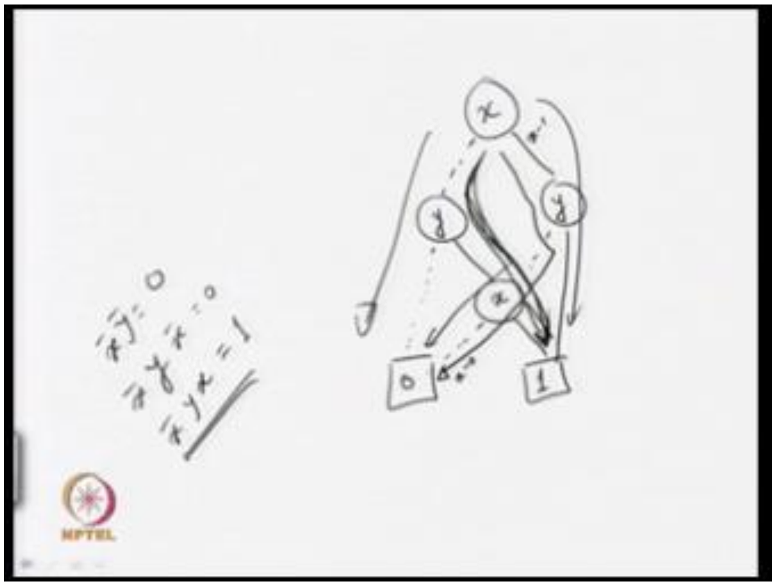
On the other hand we can use the central expansion if we are going to use this particular expansion on each variable and we can get the initial value of the BDT. So after application of central expansion whatever digit we are getting it may not be reduced on but off Course we are having only two terminals only presenting the constant 0. And second we present the constant one so after getting this particular binary decision diagram we can apply the other two deductions. One is your removal of return test and second one is your merging of duplicate non terminals.

So repeatedly we are going to apply this particular rules and finally we are going to get a radio speedily. One is that it is with non reduction role can be applied further okay. Now first question is that definition of BDT does not restrict the occurrence of the variable in any number of times in a part okay. So that it may lead to in consistency with an example. Now you itself see that I am defining the BDT binary definition diagram of reduce binary design diagram.

And when we are defining it we are saying that it we are saying that it is a deck directed a cycle gulps which is having non terminal node and terminal notes non terminal notes are having or it is leveled with your variables and terminal notes are leveled with constant 0 and 1 and every non dyminal notes are having two out going access one is represented 0 which is given by display and second one is representing the value 1 which is given by the solid guide.

Now I am saying that since this is the verification of BDD so it is not restrict ask to use the same variable in many different places okay and they can come in any order and they can coming any in, so if we use this particular basic definition I am using the constrict whether it is going to give as any problem if it is going to give any problem then so it we can exampled so now what I can say okay.

(Refer Slide Time: 56:08)



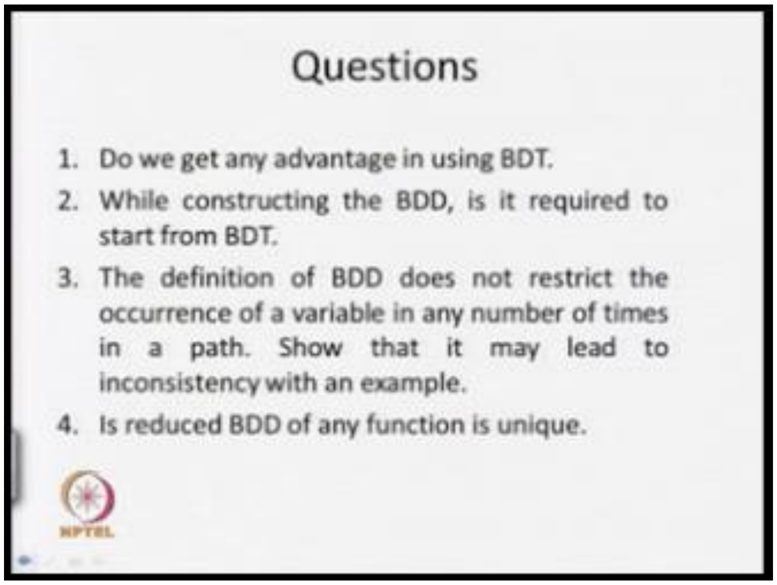
Now I will just say, now one particular BDD so this is your x, x =0 and x =1 I am getting y if y =0 say I am having this functional value 0 if y = 1 say I am trying something like this x, so I can draw this particular BDD as for our definition of BDD it is a correct BDD so I am having four non terminal notes having non terminal nodes with outgoing errors and two terminal nodes and non terminal nodes are level with variable, x, y, z, okay, so your having the functions of 2 variables x and y. So now use of circle, in this particular is x=0, y-0, the functional variable is x’y’=0, when x=0, y=1, then what will happen, then again I'm going to decision on x, if x=0, then I'm going to get 0 and when xy1x=1 then I'm going to put 1.

So now use of pseudo in this particular case, I'm having x’yx’ is 0, basically I'm looking for this particular execution part, previously x=0, y=1, when I look into this particular execution then I'm looking that illusion of x =0, illusion of y=1, illusion of x=1, now use of pseudo in this particular part, now evaluation of xi taken as 0 as well as 1, but you know that if we are going to consider any Boolean function, if we are look into any Boolean function variable at any instance of time that we have to able to take only hallow, which is known as your one hallow which is either 0 or 1.

But it cannot take both the 0 and 1, so if you look into this particular part what will happen that value of x is taken 0 and as 1, so this is basically inconsistence, you are getting a inconsistence, so you are having a inconsistence part in this particular representation, so though of basic definition of BDD is not restricting about the occupants of variable in any number of times in any pressures, but we any face problem, so in the particular case what happen, 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 divided to consistent part, so that is why we are having this as one inconsistent part because here x=0 and x=1 which is inconsistent similarly we are going to get another inconsistent path x=1 y=0 and x=0 so here x=1 and x=0 this is also inconsistent.

So we cannot have any valuation to this particular path, okay so this is the problem that we are having so if you look into the basic that we know BDD then valuation of this particular Boolean function have to be done over consistent path on, so this is the consistent path we do not have any problem x=0, y=0 we are going to take this particular value, okay. similarly this is also another consistent path x=1, y=1 we do not have any, so valuation I have to be defined over consistent path we may have inconsistent.

(Refer Slide Time: 1:00:08)

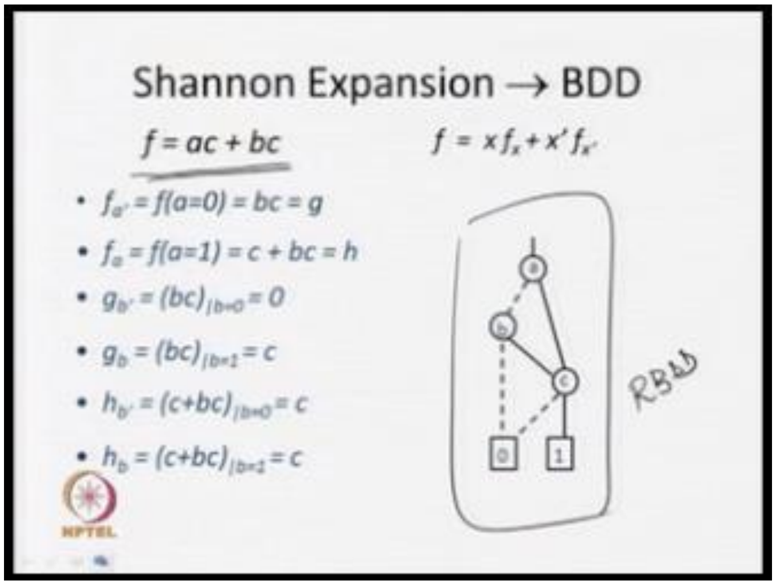


Questions

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.

Another question we are 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 rule we are going to get the reduce BDD now where that the reduce BDD representation of any function is unique as just that if you look into it let say.

(Refer Slide Time: 1:00:32)



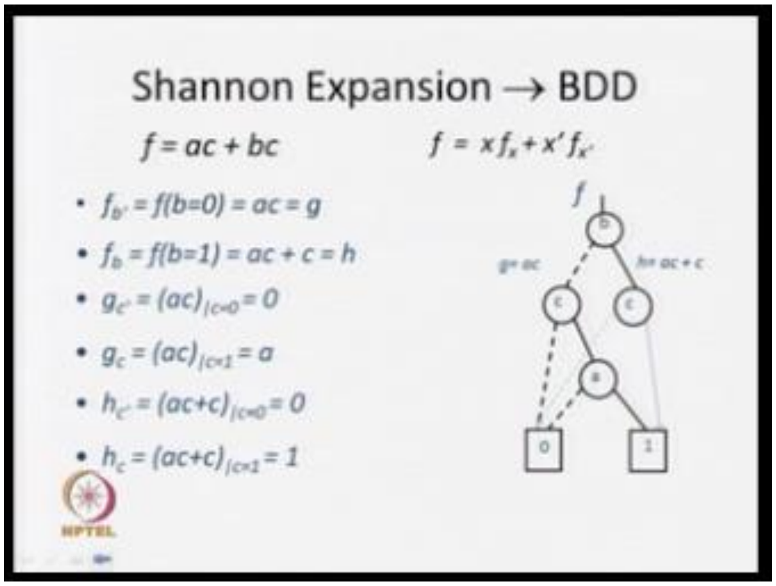
Shannon Expansion → BDD

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

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

So this is the function that we are working out f=ac+bc so a,b,c and I am getting these particular BDD and no more reduction is possible so it is an reduce BDD of the given function.

(Refer Slide Time: 1:00:51)



Shannon Expansion → BDD

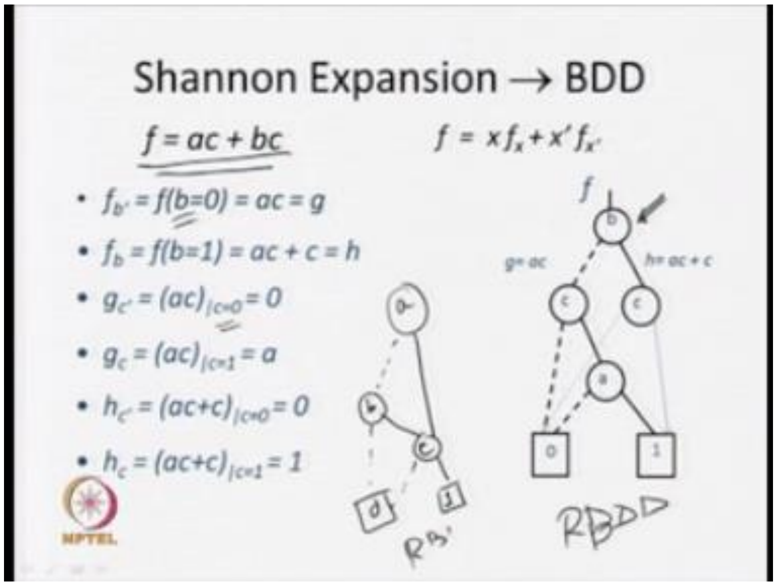
$f = ac + bc$ $\qquad$ $f = x f_x + x' f_{x'}$

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

f
b
g= ac
h= ac + c
c
c
a
0
1
NPTEL

Now the same function I can do in another way.

(Refer Slide Time: 01.01:01)



Shannon Expansion → BDD

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

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

Okay now what will happen now here what we are doing basically we are applying the particular Shannon expansion in define to it, first I am applying this on particular variable B so this is B after taking that decision of B I am going to take the decision on C that means I am using the Shannon expansion on the variable C and finally we are going to apply the Shannon expansion on A.

So you just see that now I am having BCS so this is another reduced BDD okay on this particular three variable ABC now again you just see that I think it is a reduced one you cannot apply anymore reduction in this particular thing so we are getting reduced bit it is a same function I am taking F = ac + bc okay I am getting one BDD this way and another BDD what I am getting AP, if P = 0 I am going to get 0.

If B = 1 it depends on C and one C = 0 and so these are the two RB DD both are representing the same function f = ac + bc so that means what we can say that the BDD representation or reduced BDD are not unique or a given function and from this we have seen what that is because we have seen that the other the Shannon expansion that we have used for constructing this BDD is in the order of ABC in the variable.

And here we have applied the Shannon expansion on the variable on the order BCA may be due to this reason we are getting different representation okay in next class we are going to talk about those particular issues about the ordering of this particular paragraph, okay but you just see that if you simply talk about BDD and if you talk about RBDD reduced binary digital diagram. Then this is not unique this is one simple example I am giving okay I will stop here today.