

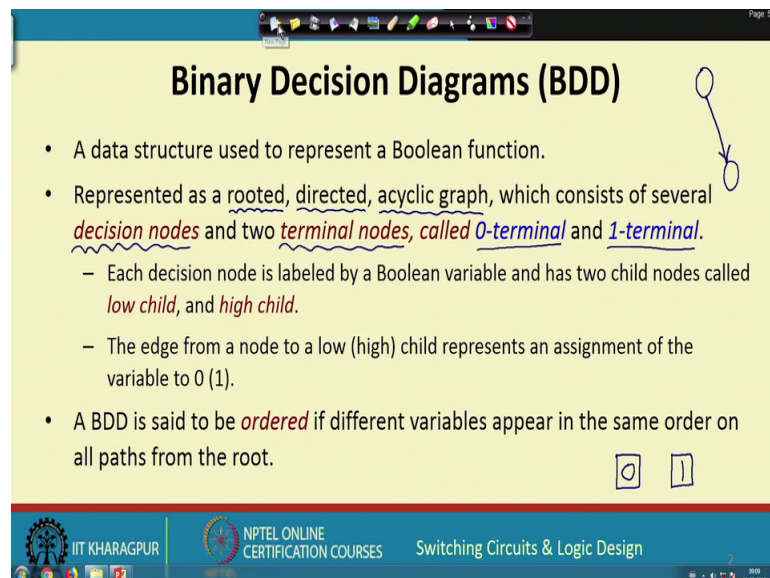
**Switching Circuits and Logic Design**  
**Prof. Indranil Sengupta**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Kharagpur**

**Lecture – 27**  
**Binary Decision Diagrams (Part I)**

So far we have seen different ways in which we can represent a switching function. We looked at the truth table; we looked at various algebraic forms, like the sum of product, the product of sum and so on. Now, over the next few lectures we shall be looking at some of you can say unconventional, but very effective ways of representing switching functions which have many applications. So, in many applications we do not represent a function just by truth table or by an expression, but by using one of these representations that we shall be discussing.


Now, the first such representation we shall be discussing is called binary decision diagram. So, this is the first part of our lecture.


(Refer Slide Time: 01:16)



**Binary Decision Diagrams (BDD)**

- A data structure used to represent a Boolean function.
- Represented as a rooted, directed, acyclic graph, which consists of several decision nodes and two terminal nodes, called 0-terminal and 1-terminal.
  - Each decision node is labeled by a Boolean variable and has two child nodes called low child, and high child.
  - The edge from a node to a low (high) child represents an assignment of the variable to 0 (1).
- A BDD is said to be ordered if different variables appear in the same order on all paths from the root.



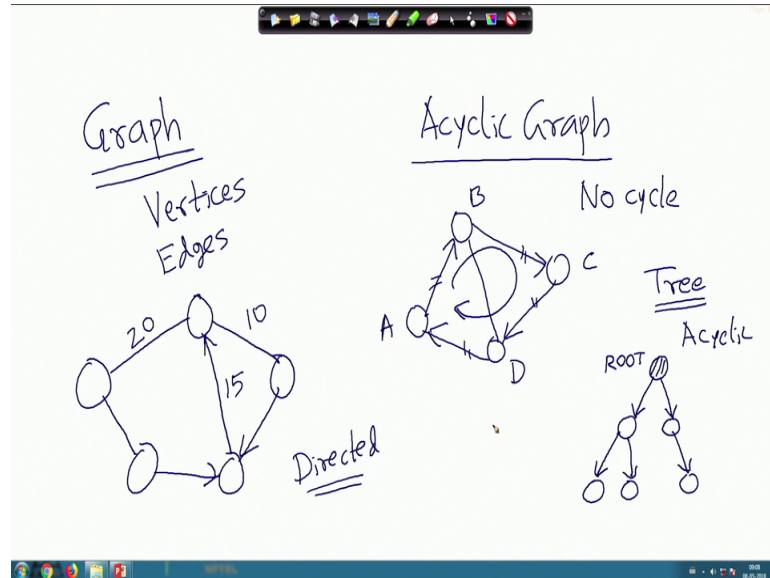


IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Switching Circuits & Logic Design

Let us try to understand first what is a binary decision diagram? So, as you can see from here from this slide binary decision diagram in short we referred to as BDD, is essentially data structure. We call it, a way to represent some information to represent a switching function or a Boolean function. BDD is nothing but some kind of a graph.

Now, for those of you who were not familiar with graph let me give you a brief introduction to a graph.

(Refer Slide Time: 01:57)



As you see when you referred to a graph, a graph essentially consist of a set of vertices and a set of edges. So, an example graph is these are the vertices typically they are represented by circles and there are edges, this edges can connect circles. Now, there are many applications where you can represent the information as a graph. Let us say this circles can indicates some cities or towns and these lines can indicate the roads that are connecting them. So, these edges can also have some labels; for example, in the example as sited the distance in kilometers can be the level of the edges, right like this and so on. And, there are some graphs where these edges can have directions. Like some of the roads may be one way roads, right. So, you cannot drive in both directions, so such a graph is called a directed graph.

Now, there is another term which we use is called an acyclic graph. See, a cycle in a graph means something like this. Suppose, I have a graphs so, let us add directions to the edges. So, you see there is some kind of a cycle like this there is a directed edge, if you consider these four edges together starting from this vertex A, I can go to B, I can go to C, I can go to D, then I come back to A, this is a cycle. Now, an acyclic graph is one where there is no cycle, right.

Now, a special type of a graph is called a tree which is of course, acyclic and a tree looks like this. So, I am just giving an example of a tree, this tree does not have any cycle and this edges can have directions again may not have direction and there is one special node which is referred to as the root which is considered to be at the top of the tree. So, these are some definitions or concepts that we shall be using in our definition of a binary decision diagram.

Now, let us see, binary decision diagram is an acyclic graph, there are no cycles, the edges are directed. Well, we may not be showing the arrows in the diagrams, but as a matter of convention so, any edge that connects a node which is on the top to a node at the bottom will be assumed to have a direction like this top to bottom, and there will be a special node which will be designated as root, root at node.

Now, the vertices can be of two types in a BDD. So, here we shall be see with example some vertices are called decision nodes, some vertices are called terminal nodes. Now, the terminal nodes are marked as 0 and 1, there is one terminal node called 0 there is one terminal node called 1, there can be several such. These are called 0-terminal and 1-terminal. They indicate the logic value 0 and the logic value 1.

(Refer Slide Time: 06:21)

**Binary Decision Diagrams (BDD)**

- A data structure used to represent a Boolean function.
- Represented as a rooted, directed, acyclic graph, which consists of several *decision nodes* and two *terminal nodes*, called *0-terminal* and *1-terminal*.
  - Each decision node is labeled by a Boolean variable and has two child nodes called low child, and high child.
  - The edge from a node to a low (high) child represents an assignment of the variable to 0 (1).
- A BDD is said to be ordered if different variables appear in the same order on all paths from the root.

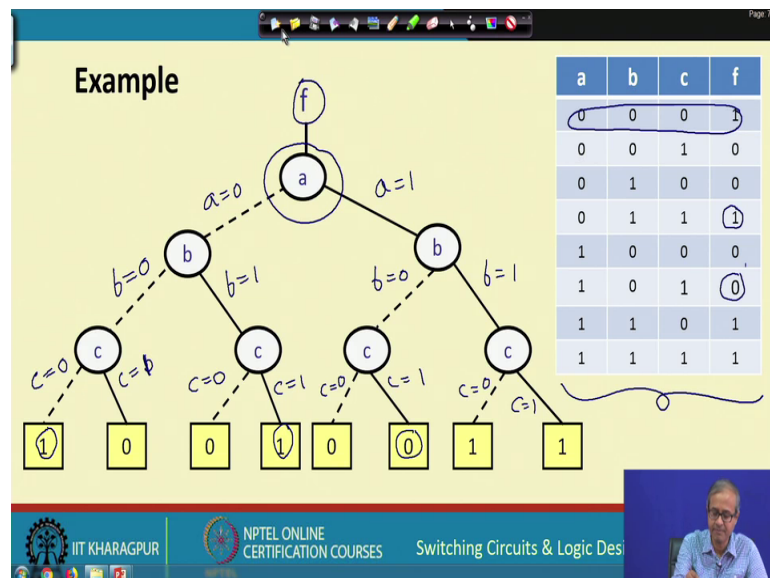
Talking about the decision node a decision node looks like this a decision node is labeled by a Boolean variable. Let us say I have a variable a, so, the vertex the decision node will be labeled by a and there will be two child nodes. Well, the nodes that connect to it

below they are called child, this is called the parent. So, there can be two child node let us say one here and one here. So, I am showing two different kind of edges one show it as dotted and one showed as solid. So, the left one I call it as the low child, the right one I call it as the high child, this is the convention. It has two child every decision node will be labeled with a variable a and will be having two child's left and right; left indicates the dotted line, right indicates the solid line.

They are called low child and high child and low child and high child actually indicates so, will be going to the left direction whenever a is equal to 0 and will be going to the right direction whenever a equal to 1. So, the edge the two edges represent the assignment of the variable here a to either 0 or 1.

Now, this we shall see later. So, in a BDD we will be having several such nodes, they will be labeled with variables. Now, if starting from the root that is a special node called the root. So, if you look at the different decision nodes that come in between. So, if the variables appear in the same order in whatever path we traverse from the root then we say that it is an ordered BDD, ordered binary decision diagram. This we shall see later, ok.

(Refer Slide Time: 08:15)



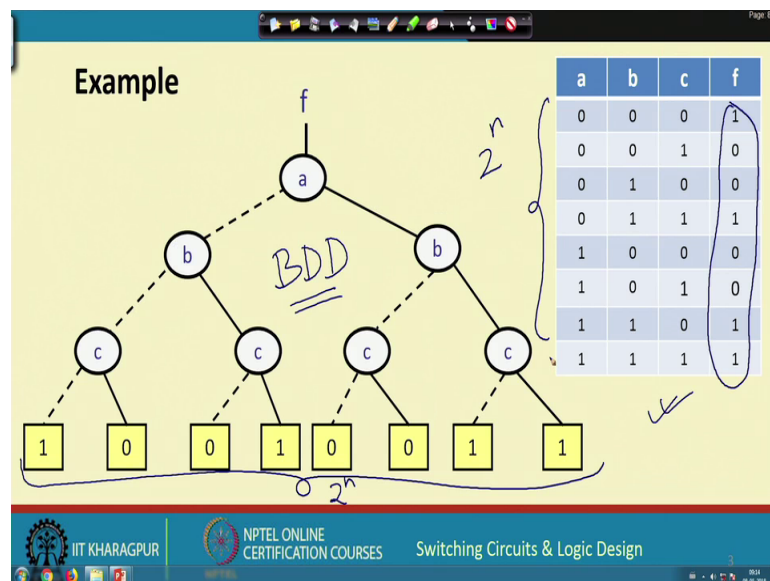
Let us now look at how a binary decision diagram looks like. Well, here we consider a three variable function represented by a truth table like this. So, you see these are the output 1 0 0 1 0 0 1 1 and a, b, c are the three variables. Now, this is a classic kind of a

binary decision diagram where the root node is on top, this is the root and the root indicates the function  $f$ , ok. Now, I have two paths left and right. So,  $a$  will be having a low child and a high child, this corresponds to  $a$  equal to 0, this corresponds to  $a$  equal to 1.

Similarly, at the next level I have nodes which are marked by  $b$ . So, in the left this indicates  $b$  equal to 0, this indicates  $b$  equal to 1, this also is  $b$  equal to 0,  $b$  equal to 1 and similarly, at the lowest level there are nodes mark  $c$ , similarly they will indicate the values of  $c$  equal to 0 and 1, like this.

So, now, you see starting from the root if you traverse along any path for example, if I traverse the left most path, I traverse the edges  $a$  equal to 0,  $b$  equal to 0,  $c$  equal to 0, I arrive at the terminal node 1. This corresponds to the first row of the truth table 0 0 output is 1. So, you consider one case where the output is output is 1 again let us say 0 1 1 0 1 1 I come here; let us take a 0 case let us consider this 1 0 1, 1 0 1.

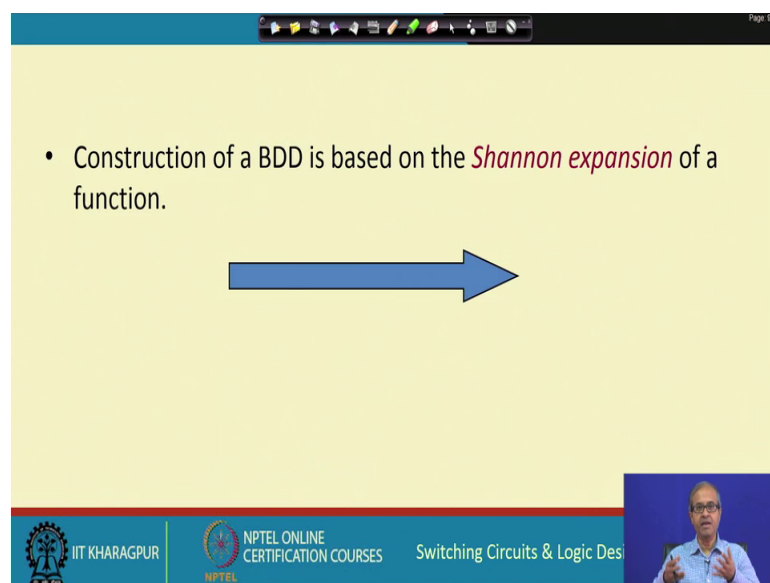
(Refer Slide Time: 10:21)



So, you see essentially what we do here is that if we look into the output column of the truth table 1 0 0 1 0 0 1 1, we simply copy this as the terminal nodes at the lowest level. This will give us one representation for this function, this is what we referred to as the binary decision diagram.

Now, one problem with this representation is that because the size of the truth table is exponential for an  $n$  variable function the number of rows is  $2^n$  because of the same reason the number of terminal nodes here will also be  $2^n$ . So, this size of this binary decision diagram will be exponential in size, exponential in  $n$ . So, it is pretty large. So, we shall see that there exists some techniques using which we can reduce the size of the binary decision diagram. So, it need not be exponential in size we can reduce the size of the BDD to a great extent we shall see some examples in this regard, ok. Let us move on alright.

(Refer Slide Time: 11:40)



• Construction of a BDD is based on the *Shannon expansion* of a function.

Earlier we talked about the Shannon's decomposition theorem. So, we again revisit that same thing Shannon's expansion we are calling here of a function. Now, the idea here is that we can construct a BDD from a given function by repeated applications of the Shannon's decomposition theorem or Shannon's expansion theorem.

Now, recall earlier when we talked about the multiplexer realization of a function representing a function using two to one multiplexers, there also we did the same thing. We successively decomposed a function into sub functions by applying the Shannon's decomposition theorem, we got smaller and smaller functions and every decomposition was map to a two to one multiplexer. Here the concept is almost identical every decomposition will be corresponding to one decision node of the BDD and we repeatedly do it we get the complete BDD, let us see.

(Refer Slide Time: 12:52)

**Shannon Expansion**

- Given a Boolean function  $f(x_1, x_2, \dots, x_i, \dots, x_n)$
- Positive cofactor  
 $f_i^1 = f(x_1, x_2, \dots, 1, \dots, x_n)$
- Negative cofactor  
 $f_i^0 = f(x_1, x_2, \dots, 0, \dots, x_n)$
- Shannon's expansion theorem states that  
 $f = x_i' f_i^0 + x_i f_i^1$   
 $f = (x_i + f_i^0)(x_i' + f_i^1)$

The diagram shows a node labeled  $x_i$  with two branches: the left branch is labeled  $x_i=0$  and leads to  $f_i^0$ ; the right branch is labeled  $x_i=1$  and leads to  $f_i^1$ . The root node is labeled  $f$ .

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Switching Circuits & Logic Des

Take an example. Suppose, I have an  $n$  variable function  $f$  the variables are  $x_1$  to  $x_n$  in general there is a variable  $x_i$ . So, we talked about this earlier also. So, we define something called a positive co-factor. This we denote as  $f_i^1$  with 1 in this superscript which means the same function where this  $x_i$  is replaced by 1, this is referred to as the positive co factor with respect to variable  $i$ . So, in the same way we can define a negative co factor negative co factor is very similar which is denoted by  $f_i^0$ , where this variable  $x_i$  is replaced by the constant value 0, right.

Now, Shannon's expansion theorem we have already seen. Now, here we have showing two forms of expansion one of that we have seen earlier. So, in Shannon's expansion theorem we can write the function  $f$  like this. So, when you expand a function with respect to  $x_i$  I can write  $x_i'$ . That means the compliment of  $x_i$  multiplied by the negative co factor with respect to  $x_i$  plus the variable  $x_i$  and the positive co factor. Now, there is an alternate representation also which you have not talked about earlier, where this is something like product of sums. This same function we can write like  $x_i + f_i^0$  and  $x_i' + f_i^1$ . But, of course, we shall be using the first representation only in our examples and illustrations.

Now, one thing you see the why we expanded here this one. So, we can map it into a decision node of the BDD, where the node will be labeled by  $x_i$  and the left and right



this is this corresponds to  $x_i$  equal to 0, this corresponds to  $x_i$  equal to 1, the low and right child's they will be corresponding to  $f_i 0$  and  $f_i 1$ .

So, basically and this of course, this will represent the function  $f$ . So, every decomposition of the Shannon expansion can be mapped into a decision node of the BDD or a decision vertex, this is the basic idea.

(Refer Slide Time: 15:48)

**How to construct BDD?**

$$\begin{aligned}
 f &= ac + bc + a'b'c \\
 &= \underline{a'}(b'c' + bc) + \underline{a}(c + bc) \\
 &= \underline{a'}(b'c' + bc) + \underline{a}(c)
 \end{aligned}$$

This is the first step.  
The process is continued for all input variables.

Low child:  $b'c' + bc$   
High child:  $c$

NPTEL ONLINE CERTIFICATION COURSES  
Switching Circuits & Logic Des

So, function example here consider a three variable function like this. Here in this step we are decomposing this function with respect to the variable  $a$ . So, we write  $a$  multiplied by the negative co-factor, replace  $a$  by 0 you get  $b'c' + bc$  plus  $a$  multiplied by the positive co-factor replace  $a$  by 1, this become  $0c + bc$ . Now,  $c + bc$  can be simplified into only  $c$  and this cannot be simplified any further. So, this decomposition as I said can be conceptually mapped into a decision node, where this node is labeled  $a$  the low child refers to this function  $b'c' + bc$  and the right child refers to this function  $c$ , ok.

This is of course, the first step and we repeatedly go through this, next we will be doing let us say using variable  $b$ . Next we will be doing using variable  $c$  and so on. Let us see how we do it.



(Refer Slide Time: 17:12)

$f = ac + bc + a'b'c'$   
 $= a'(b'c' + bc) + a(c + bc)$   
 $= a'(b'c' + bc) + a(c)$

Expand by  $a$

$b'(c) + b(c)$        $b'(c) + b(c)$

Expand by  $b$

$c'(1) + c(0)$      $c'(0) + c(1)$      $c'(0) + c(1)$      $c'(0) + c(1)$

Expand by  $c$

Variable ordering:  $a, b, c$

$b$   
 $a$   
 $c$   
 $c$   
 $a$   
 $b$   
 $\vdots$

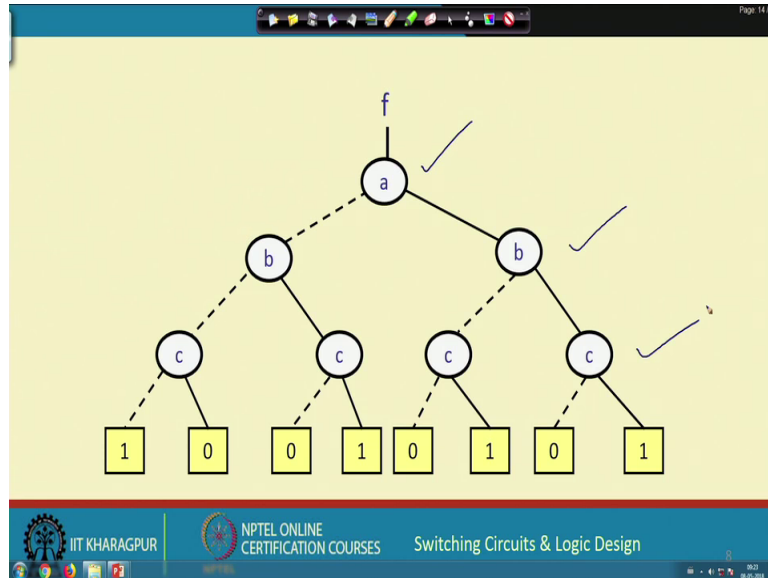
Well, here the whole step is shown. The first step we have already seen, this step we have already seen in the previous slide. So, where we have expanded the function with respect to variable  $a$  and we have got two sub functions  $b'c' + bc$  and only  $c$ . In the next step let us suppose we are trying to expand with respect to  $b$ . So, this  $b'c' + bc$  plus  $bc$  if you do a Shannon expansion with respect to  $b$  you do it in a similar way  $b'$  multiplied by the negative co factor with respect to  $b$ , replace  $b$  with  $0$  this all becomes  $0$  it becomes only  $c'$ , only  $c'$  and  $b$  multiplied by positive co-factor, replace  $b$  by  $1$ , it becomes only  $c$ .

Similarly, for this variable  $c$  you do the same thing with respect to  $b$ ,  $b'$  multiplied by replace  $b$  by  $0$ . There is no  $b$ , so, it remains  $c$ ,  $b$  into positive co factor replace  $b$  by  $1$ , no  $b$  so, it remains  $c$  and in the last step we expand you see. Now, we have got four sub functions  $c'$ ,  $c$ ,  $c$  and  $c'$ . So, you see the size of the sub functions are getting smaller and smaller as you proceed. So, in the last step we expand by  $c$ , you see  $c'$  can be written like this  $c'$  multiplied by the negative co-factor replace  $c$  by  $0$  it becomes  $1$ , plus  $c$  negative co the positive co-factor replace  $c$  by  $1$ ,  $c'$  becomes  $0$ , right, but if it is  $c$ , then  $c'$  multiplied by replace  $c$  by  $0$  it is  $0$   $c$  into replace  $c$  by  $1$ , it is  $1$ . So, in this way I get this.

Now, you recall when you do the expansion here I have shown  $a, b, c$  I can do it in any order. I can first use  $b$  then  $a$  then  $c$  or I can use first  $c$  then  $a$  then  $b$  and so on. So, here I

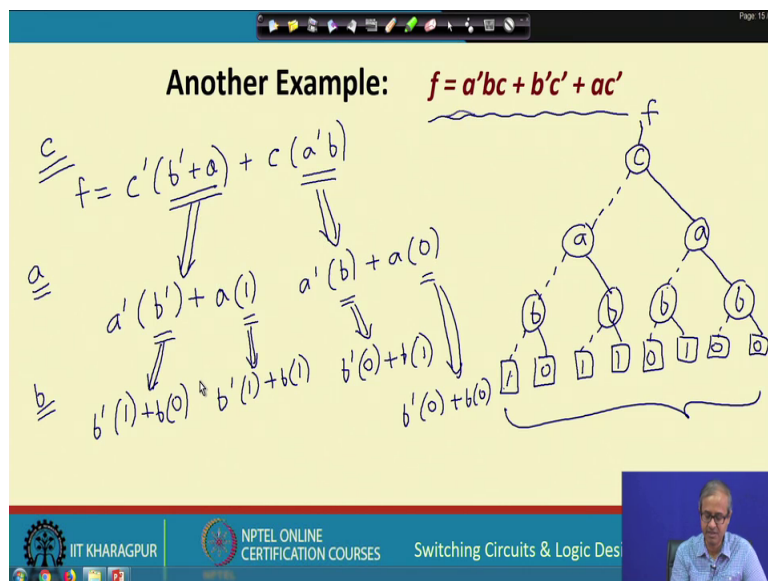
have illustrated only one particular order, the variable ordering is a, b, c. So, at the end I get some constants or terminals you see 1 0 0 1 0 1 0 1 remember this 1 0 0 1 0 1 0 1.

(Refer Slide Time: 19:44)



So, I can directly map this into a BDD like this 1 0 0 1 0 1 0 1. So, so, my order of expansion was first I use variable a. Next level I use variable c, third level I was used variable c b and c right, this is how we had carried out the expansion.

(Refer Slide Time: 20:10)



Let us take another example, let us work this out. Suppose, we have a function like this a bar bc plus b bar c bar plus ac let us follow some particular I am not may say a, b, c let us

say first we expand with respect to variable  $c$ , let us see. So, it will be the function will be  $c$  bar the negative co-factor replace  $c$  by 0, this will become 0, it will be  $b$  bar or  $a$ ,  $b$  bar or  $a$ ,  $b$  bar or  $a$  plus  $c$  into replace  $c$  by 1, the last two will become 0, it will become only  $a$  bar  $b$   $a$  bar  $b$ . So, I have got two sub functions  $b$  bar plus  $a$  and  $a$  bar  $b$ .

In the next step let us expand with respect to  $a$ , let us say. So, this function if we expand with respect to  $a$ , this will be  $a$  bar into replace  $a$  by 0 it will be only  $b$  bar plus  $a$  into replace  $a$  by 1 this will be something plus 1, it will be 1, right. In the same way if you expand this with respect to  $a$ , it will be  $a$  bar replace  $a$  by 0. So, it will be only  $b$  plus  $a$  replace  $a$  by 1 this will be 0. So, now, we have four sub functions  $b$  bar, 1,  $b$  and 0. Now, we are left with  $b$ , ok, we will be expanding with respect to  $b$ .

So,  $b$  bar can be expanded as  $b$  bar replace  $b$  by 0 this will be 1 plus  $b$  replace  $b$  by 1, this will be 0 and here we have a constant one. So, if we expand by  $b$ ,  $b$  bar replace  $b$  by 0, there is no  $b$ , this remains 1 plus  $b$  into replace  $b$  by 1, it remains 1 and for  $b$  it will be  $b$  bar replace  $b$  by 0, this will be 0 plus  $b$  replace  $b$  by 1, this will be 1 and for 0 this will be  $b$  bar into 0 plus  $b$  into 0. So, now, we have 1 0 1 1 0 0 0 0.

So, now, our BDD will look like this will be having the root node  $c$  at the top level this will represent the function  $f$ , at the next level will be having  $a$ ,  $a$ , and  $a$  this will be the negative edge 0 and this will be positive edge. Next level will be having  $b$ ,  $b$ ,  $b$ ,  $b$  and  $b$ . So, this will be negative, this will be positive, this will be negative, this will be positive and in the last level will be having  $c$  no, not  $c$  ok,  $c$   $a$   $b$  you have taken already.

So, last level you have the terminal nodes. So, if you take the terminal nodes. So, we have 1 and 0. So, you will be having 1 out here, 0 out here like this then 1 and 1, 1 out here, 1 out here like this 0 and 1 0 out here, 1 out here and 0 0. So, we have obtained the binary decision diagram for this function. So, given any variable ordering you can construct the BDD by systematically decomposing the function using Shannon's law, fine.

(Refer Slide Time: 24:42)

**Variable Ordering (OBDD)**

- The size of a BDD is determined both by the function being represented and the chosen ordering of the variables.
  - For some functions, the size of a BDD may vary between a linear to an exponential range depending upon the ordering of the variables.
- An example:
$$f(x_1, \dots, x_{2n}) = x_1x_2 + x_3x_4 + \dots + x_{2n-1}x_{2n}$$

Variable ordering:  $x_1 < x_3 < \dots < x_{2n-1} < x_2 < x_4 < \dots < x_{2n}$   
BDD requires  $2^{n+1}$  nodes to represent the function.

Variable ordering:  $x_1 < x_2 < x_3 < x_4 < \dots < x_{2n-1} < x_{2n}$   
BDD requires  $2^n$  nodes to represent the function.

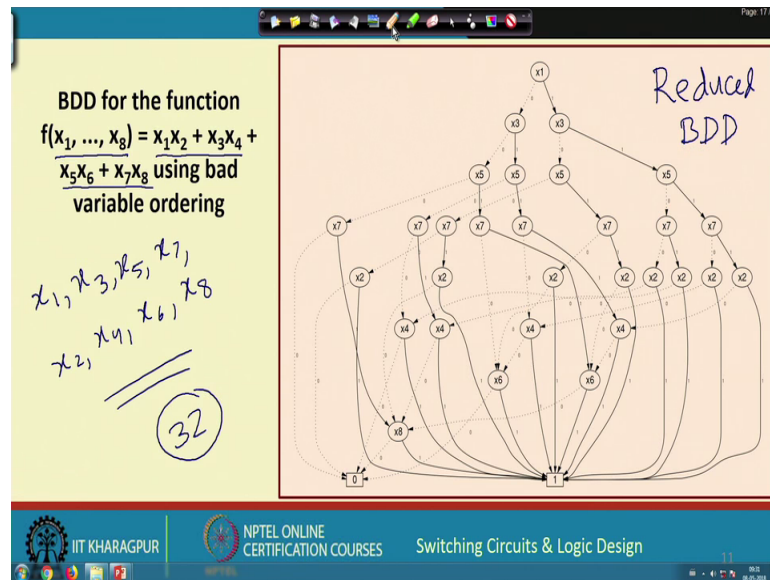
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Switching Circuits & Logic Des

Now, let us talk about variable ordering. The point to note here is that the size of a binary decision diagram is determined not only by the function you want to represent, but greatly also on the ordering of the variables. You see in general for most of the functions BDD gives a very compact representation for a function we shall see some examples later, but what we are trying to point out here is that the ordering of the variable plays a great role. We shall take an example where the ordering if we change the size of the BDD can differ to a great extent. Well, of course, here we shall be showing you the reduced or minimized form of BDD which we have not discussed so far, just we shall be showing you pictorially how the reduced version looks like, later on we shall see how to arrive at the reduced version.

Let us see, let us take an example we say that it can be proportional to the input variables, it can be proportional to the power of an input variable is called exponential. Let us take an example. This is a very classic example. Suppose, I have a function which looks like this, but the variables are paired the product terms are like this  $x_1 x_2$  or  $x_3 x_4$  or  $x_5 x_6$  like this. So, if I use expansion if I use variable orderings like this first I choose the odd number variables  $x_1, x_3, x_5$  and so on. Then I choose the even number of variables then it can be shown that I will be requiring exponential number of nodes in the reduce or minimum BDD representation.

But, if we use a variable ordering which is just  $x_1, x_2, x_3, x_4$  in that order then we will be requiring only  $2^n$  nodes. This is a very classical example which is given in all textbooks which is used to show that variable ordering is very important, ok.  $2^n$  to the power  $n$  is very large, it grows very quickly with respect to  $n$ , but  $2^n$  is proportional to  $n$ .

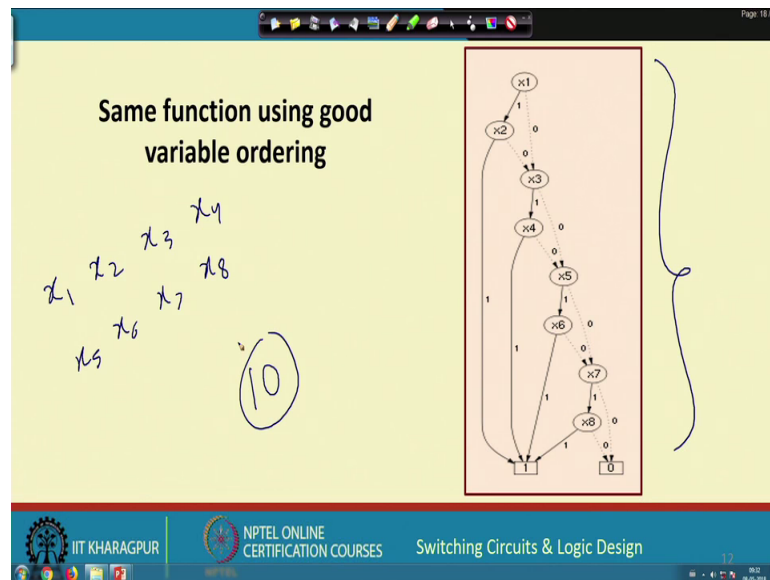
(Refer Slide Time: 27:13)



Let us take a very specific case of that function, where there are eight variables. So,  $x_1, x_2, x_3, x_4, x_5, x_6$  and  $x_7, x_8$ . First we considered the bad variable ordering; that means, we expand with respect to this first the odd number of variables  $x_1, x_3, x_5, x_7$ , then the even number variable  $x_2, x_4, x_6$  and  $x_8$ . So, you see here the entire BDD shown of course, this is not the conventional BDD. This is the reduced version of the BDD. So, we shall see later how to reduce a BDD in size.

But the point to note is that you see there are large number of nodes in this representation if you count how many 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 20 23 27 30 and 2 terminal nodes total 32 nodes are there in the BDD representation, if we have a variable ordering like this, right.

(Refer Slide Time: 28:31)



But, this same function if we use an alternate variable ordering like here the same function we are using a variable ordering like this  $x_1$ , then  $x_2$ , then  $x_3$ , then  $x_4$ , then  $x_5$ ,  $x_6$ ,  $x_7$ , and  $x_8$ ; then we have you see such a compact representation of the function where the number of nodes are 1 2 3 4 5 6 7 8 9 10 only 10 nodes are there. So, this is a very classic example which is used to illustrate. But, the ordering of the variable is very important and most of the BDD generation and manipulation tools they try to find out very good ordering of the variables, so as to minimize or reduce the size of the function.

(Refer Slide Time: 29:24)

Slide 19/19:

- Important point to note:
  - It is essential to find a good variable ordering when using the OBDD data structure in practice.
  - The problem of finding the best variable ordering is NP-hard.
  - Several heuristics for variable ordering have been proposed.

Video inset: A small video window showing a speaker in a blue shirt.

Footer: IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Switching Circuits & Logic Design

So, the important point to note is that just and mention this just now, that it is very important to find a good variable ordering and we have a binary decision diagram here which is called ordered binary decision diagram, where to ordering of the variable is defined. Of course, finding the best ordering is not easy this is this NP-hard is a class of problems which are consider to be computationally very complex. Why? If there are  $n$  number of variables the number of possible orderings can be factorial  $n$  and factorial function grows very rapidly with  $n$ , right. That is why this are very difficult problem. But, there are a number of rules or heuristics these are called which are used to find a good variable ordering in general.

So with this, we come to the end of this lecture. Now, in the next lecture we shall be looking at how we can reduce the size of the BDD to the form in the two examples that we showed. The conventional BDD looks like a tree, how we can reduce the size of the tree and make the BDD more compact, such rules we shall be discussing in our next lecture.

Thank you.