

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

Lecture – 28
Binary Decision Diagrams (Part II)

So, we continue with our discussion on Binary Decision Diagram. If you recall in our last lecture we said what a Binary Decision Diagram is we also talked about something called ordered BDD ordered binary decision diagram, where the order in which we are expanding variable that is fixed from, or with respect to any path from the route that you follow that.

Let us say if I use an order a b c so, will be expanding along all the paths in that same order a b and c. Now, in this next lecture on binary decision diagrams, we shall be talking about some rules using which we can reduce the size of the BDD,

(Refer Slide Time: 01:02)

Reduced Ordered BDD (ROBDD)

- An ordered binary decision diagram is said to be *reduced ordered BDD* (ROBDD) if the following two graph reduction rules are applied:
 - ✓ - Merge any isomorphic subgraphs.
 - ✓ - Eliminate any node whose two children are isomorphic.
- The advantage of an ROBDD is that it is *canonical* (unique) for a given function.
 - There is exactly one ROBDD for a given variable ordering.
 - This property makes it useful in functional equivalence checking.

The slide includes two diagrams of Binary Decision Diagrams (BDDs). The left diagram shows a node 'a' with two children, '0' and '1', which are isomorphic. The right diagram shows a node 'a' with two children, '0' and '1', which are also isomorphic. A node 'b' is shown above 'a' with children '0' and '1'. A node 'x' is shown to the right of 'b'.

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

Because as I said the size of the BDD as such is exponential in nature, because the number of leaf nodes the terminal nodes that will be equal to 2 to the power n the same size as the output of the truth table.

So, we have to do something to reduce the size, because exponential is a really large number. Suppose I have a 20 variable function, how much is 2 to the power 20? 2 to the

power 20 is 1 million; for 30 variable function how much is 2 to the power 30? It is 1 billion. So, that number grows very rapidly with n all right, today we talk about functions with 100 variables and even more. So, we just cannot represent a binary decision diagram in a conventional way we need some rules for reduction.

Let us see so, reduced order BDD that is what we call this is referred to as ROBDD, well we know that what is a binary decision diagram, if the ordering of the variables is constant along all the paths we called it as ordered BDD, now we apply some rules to reduce the size so, what we get is reduced ordered BDD. So, we have this reduced ordered BDD, where two reduction rules are applied systematically, these are the two rules we shall be illustrating.

We shall be merging any isomorphic sub graphs what is isomorphic sub graphs, isomorphic sub graphs are two graphs which are identical in nature like, let me take an example suppose I have a part of my BDD which looks like this, there is another part of my BDD where also the same structure is formed. So, I say that these two are isomorphic, if I find two such sub graphs which are identical isomorphic we can merge them into one.

This is one rule and we can eliminate any node whose two children are isomorphic. So, what I mean is that suppose I have a decision on b at a higher level so, one of its children is pointing here other is pointing here. So, what it says that both are pointing to basically the same thing right this to indicate the same thing. So, there is no need for using b , so if b is 0 then also will be doing this, if b is 1 then also will be doing this. So, we can simply eliminate b from this graph this is so, called elimination rule we shall be seeing in this more detail.

Now, the interesting thing about ROBDD is that it is a canonical form. Now, you recall for you we talked about some of product and product of some representation, we talked about canonical representations canonical is some sort of unique representation, this reduced order binary decision diagram with respect of particular variable ordering is canonical, which means there exists a single unique ROBDD presentation ok.

Now, because there is exactly one ROBDD for a given function, this is a very useful property using which we can use it in many applications, one such application is functional equivalence checking like.

(Refer Slide Time: 05:15)

Reduced Ordered BDD (ROBDD)

- An ordered binary decision diagram is said to be *reduced ordered BDD* (ROBDD) if the following two graph reduction rules are applied:
 - Merge any isomorphic subgraphs.
 - Eliminate any node whose two children are isomorphic.
- The advantage of an ROBDD is that it is *canonical* (unique) for a given function.
 - There is exactly one ROBDD for a given variable ordering.
 - This property makes it useful in functional equivalence checking.

$f_1 \equiv f_2$

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

We have two functions f_1 and f_2 and we are not very sure whether they are the same function or not, we want to check whether they are equivalent or not. So, what you can do we can create the ROBDD of the two functions and, then check whether the 2 ROBDD are identical or not fine.

(Refer Slide Time: 05:35)

Some Properties of ROBDD

- The following properties hold in a ROBDD:
 - Uniqueness:** No two distinct nodes u and v are labeled with the same variable name and have the same low and high successor.
 - Non-redundant:** No variable node u has identical low and high successor.

Diagrams illustrating the properties:

- 1. A node labeled 'a' with two children, '0' and '1', representing a non-redundant node.
- 2. A node labeled 'a' with two children, '0' and '1', representing a non-redundant node.
- 3. A node labeled 'a' with two children, 'a' and 'b', representing a redundant node.

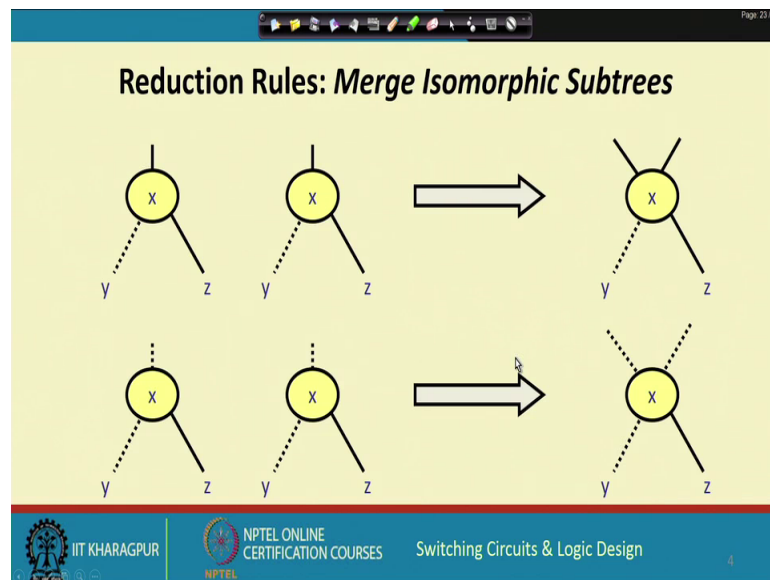
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Switching Circuits & Logic Design

So, some properties of ROBDD as I have just now said, there are two main properties; one is uniqueness no two distinct nodes can be labeled with the same variable name and have seen low and high successor. Just the example that you took earlier a 0 1, the same

example let me show once more. And again a 0 and 1 we see let us say this is my node a and this is my node b, it says that you cannot have two such nodes even we labeled with the same variable here a and, have the same low and high success low successor is 0 in both the cases and high successor 1 in both the cases.

So, if you have such then you will be merging them into 1, there will be unique copy of such structure existing. And non redundant is other one it is says no variable node can have identical low and high successor. So, a particular node cannot have let us say both the dotted and the solid arcs pointing to the same node in the next level, this means that a not required irrespective of the value of a you will come to be anyway ok these are the two main properties of ROBDD that needs to be satisfied.

(Refer Slide Time: 07:17)

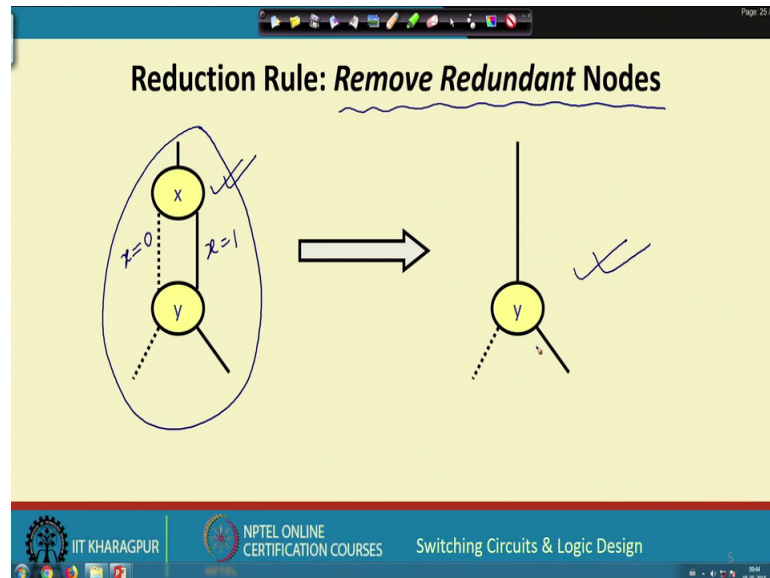


So, this is explained diagrammatically here, first is the deduction rule where we say that if we find isomorphic sub graphs merge them. So, two cases are shown here there can be similar cases other similar cases also. If you find that the two sub graphs like this, two nodes labeled with the same variable x and x same low and high successors child y z y z, then you and you see they are coming from two different places ok.

So, then you merge them into a single node and around places from they are coming so now, they will be to incoming input stages one from this one other from this one, but you use is a single copy of this single copy of this. Similar this stage maybe solid this stage

maybe dotted also. So, this is other example so, if we have this thing then you can merge this like in this way same way merge this two and two edges will be coming like this.

(Refer Slide Time: 08:58)



So, you can have the other cases also I have not shown all like for example, one of them can be dotted and the other can be solid. So, in that case when you merge the first one will be dotted and this one will be solid right, or the other way around this one solid this one dotted ok. So, you can have all such cases of merging. The other case is to remove the redundant nodes, like I already mentioned this if you have a scenario like this.

Where there is a decision node labeled with x it says, if x equal to 0 you come here if x equal to 1 come here, now you see this decision does not make any sense, because you are anyway coming to y in both the cases, which says you can all together eliminate this node x and get a reduced representation like this. So, these two rules I mean if you apply repeatedly on a binary addition diagram, then the size of the BDD will be going on reducing step by step and finally, we shall be getting representation which is called the ROBDD, or the reduced order BDD representation of the function.

(Refer Slide Time: 10:08)

Construction of ROBDD: an example

Given function: $f(x_1, x_2, x_3) = x_1 \cdot x_2 + x_1' \cdot x_3 + x_1 \cdot x_2' \cdot x_3$

$x_1 > x_2 > x_3$

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

Let us take an example complete example, for constructing the reduced order BDD. So, we considered a three variable function like this a function of three variables ok.

(Refer Slide Time: 10:26)

Diagram illustrating the construction of a Reduced Order BDD (ROBDD) from a full BDD for the function $f(x_1, x_2, x_3) = x_1 \cdot x_2 + x_1' \cdot x_3 + x_1 \cdot x_2' \cdot x_3$.

The diagram shows the full BDD (left) and its expansion (middle), followed by the reduction process (right) resulting in the ROBDD (bottom right).

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

So, if you use the method for constructing BDD that we discussed earlier, you have the function here let us say I expand the variables, in this order first x_1 then x_2 then x_3 so, I am not showing the steps you can use this expansion step to get the functional representation so, I am only showing the final one. So, once you do this the BDD you get will look like this, this is your initial form non reduced version of the BDD right.

Now, you see you will be apply in those two rules that I mentioned in a repetitive fashion, you will be merging nodes whose left and right childs, or low and low and high childs are identical, or if you get isomorphic sub graphs merge them. Now, with respect to the original BDD 1 observation, you can immediately see that in the terminal part there are so, many 0 and so many one nodes. So, why not merge all the 0 nodes into 1 and why not merge all the 1 nodes into 1 that is the first step we do ok.

So, you see here in this example there are 1 2 3 0 nodes and 5 1 nodes. So, the first step you do is something like this, we use a single copy of this 0 node and a single copy of the 1 node, you see the 0 nodes are pointed from where the dotted h for x 3, you see from dotted x 3 comes here dotted h from this x 3 still comes here, this x 3 also comes here and also from here, this also comes no this 3 only ok.

And the 1 the they write this solid arrow for this x 3 the solid arrow from x 3 coming here, the solid arrow from this x 3 also solid arrow, solid arrow from this x 3 solid arrow and both dotted and solid arrow from this x 3 right. So, this is the first step of reduction, now in this step you can make some observations, well the first thing is that you see if you look at this part, x 3 both the dotted and solid edges are pointing to the same node.

So, as per as our reduction rule we can eliminate x 3 and, another thing also you see if you look at just look at these two nodes this x 3 and this x 3, you see these are isomorphic why because in this x 3 the low child points to 0 and high child points to 1, here also low child points to 0 high child points to 1 so, they can be merged.

So, after this reduction step we get something like this, you say exactly what I said this x 3 was having two parallel edges pointing to 1 so, here we have eliminated that three this x 2 is now with a one edge is directly pointing 2 1 x 2 is directly pointing to 1, and these 2 x 3s have been merged into 1, this is x 3 and input edges was see one was coming from x 2 the dotted line in the solid line, now from x 2 both the dotted line in the solid lines are now pointing to x 3 right. So, after this reductions so, you get this.

Now, in this version you see now again who have the scope for reduction, you look at this part of the graph, this x 2 has both the dotted and the solid, solid arrows pointing to the same node. So, x 2 can be eliminated ok. So, you get finally, after eliminating x 2 a BDD like this. So, you can understand what is the purpose of carrying out this reduction, because in the original BDD there were 8 terminal nodes in the lowest level 8 for 12, 14,

15 total 15 nodes. And after reducing you are left with only five nodes for larger function the reduction can be even more, this is the basic purpose of carrying out reduction and getting order BDD.

So, what we get finally, here this is your ROBDD, for this particular variable ordering that reduce $x_1 \times x_2 \times x_3$, but if you use some other variable ordering, then you can get some other structure for ROBDD fine.

(Refer Slide Time: 16:02)

The slide is titled "Some Benefits of BDD" and lists three main points:

- Checking for tautology is trivial.
 - BDD is a constant 1.
- Complementation.
 - Given a BDD for a function f , the BDD for f' can be obtained by simply interchanging the terminal nodes.
- Equivalence check.
 - Two functions f and g are equivalent if their BDDs (under the same variable ordering) are the same.

Hand-drawn diagrams on the slide include a BDD node labeled 'f' with a downward arrow pointing to a terminal node '1', and a circled 'f' followed by an equals sign and a '1'.

The slide footer contains the IIT KHARAGPUR logo, NPTEL ONLINE CERTIFICATION COURSES, and the course title "Switching Circuits & Logic Design".

So, some benefits of ROBDD are pretty straight forward to understand, you see you have a function f now I want to check, if the function is equal to 1 for all assignments of input variables such a function is called a tautology, now for a BDD it is very easy to check whether a function is tautology.

Because the ROBDD of that function will simply be like this, it is straight away point to the node 1 right, not only dot also dash. So, checking for tautology simply f will be pointing directly to 1, the second thing is that complementation well you look at the ok.

(Refer Slide Time: 17:08)

Page 33/33

Some Benefits of BDD

- Checking for tautology is trivial.
 - BDD is a constant 1.
- Complementation.
 - Given a BDD for a function f , the BDD for f' can be obtained by simply interchanging the terminal nodes.
- Equivalence check.
 - Two functions f and g are equivalent if their BDDs (under the same variable ordering) are the same.

f'
=
=

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

Let us take an example we did not try to illustrate, suppose I have a BDD representation let us say I have x_1 , I have x_2 , I have x_3 here and I have 0 here, and I have 1 here, let us say x points to it is, suppose I have a function like this is f .

Now, what I am saying is that suppose I have a BDD of a function f , now I want to get a generate a BDD for the complement function should, I have to go through the process again and construct the BDD. It says it is not required if you have the BDD for a function f , just interchange 0 and 1 make this 0 as 1 make this 1 as 0.

So, for all those cases where the function was getting the value 0, now it will get the value 1 and for all cases where it was getting 1, now it will get 0 just the reverse right. And thirdly these equivalents whether two functions are identical or not it is easy because, here you have seen that ROBDD is canonical. So, if you construct the ROBDD of the two function and show that they are the same, then you can say that the two functions are equivalent, or otherwise the functions are not equivalent ok.

(Refer Slide Time: 18:44)

Use of BDD in Synthesis

- BDD is canonical for a given variable ordering.
- It implicitly uses factored representation:

$h(x+x')$
 $x'h + xh = h$

$ah + bh = (a+b)h$

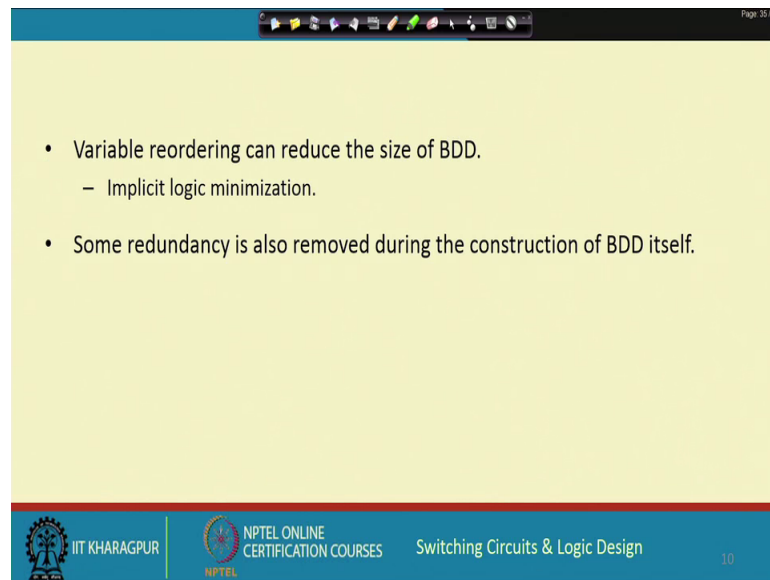
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Switching Circuits & Logic Design

Now, means you can use this ROBDD in various ways for synthesizing circuits also. So, we shall be taking some examples say earlier I mentioned that the 2 2 1 multiplexer the way we designed a function using such multiplexers. And the way we realize a BDD implement of BDD using repeated Shannon decomposition they are very similar. So, a BDD node and a 2 2 1 multiplexer they have a one two one correspondence.

Let us take some examples so, in synthesis what I mean to say is that you see some of the reduction rules that we are applying, we already mentioned that one is this rule, where if a node is pointing both edges to the same node I can eliminate that node well. In terms of switching algebra what it does that mean, you see if this entire sub function below, this represents a sub function h . Then this BDD represents $x \bar{h}$ or $x h$ the left hand side so, if you take h common it will be x or $x \bar{h}$ which is equal to 1 it get is eliminated only h .

So, this kind of a algebra minimization is implicitly carried out during this transformation. Similarly the other one there are two identical or isomorphic sub graphs, suppose they represents h this means plus bh . So, if I merge them like this it means a or b and h they also have the same. So, again using some rules of the switching algebra we do this kind of minimization. So, we are doing the same thing implicitly using this graphical data structure of BDD.

(Refer Slide Time: 20:54)



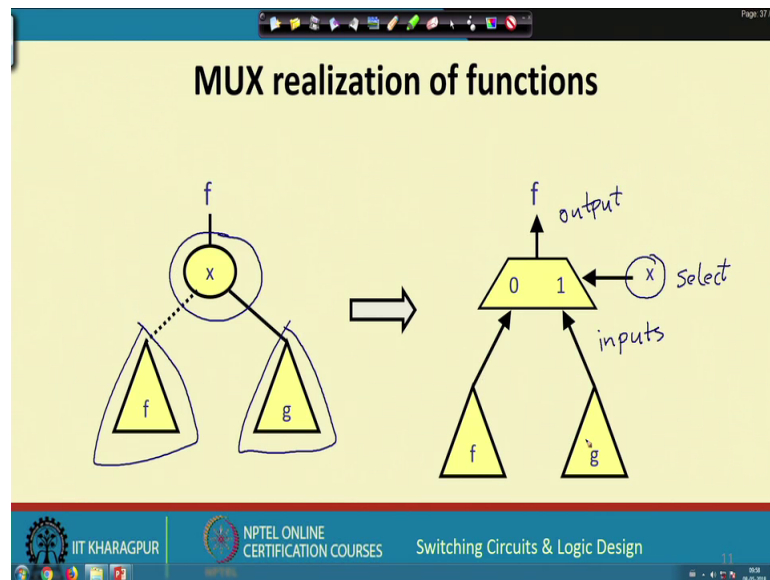
- Variable reordering can reduce the size of BDD.
 - Implicit logic minimization.
- Some redundancy is also removed during the construction of BDD itself.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Switching Circuits & Logic Design | 10

So, so let us make one point here this already we have mentioned that variable ordering can reduce the size of the BDD ok. Now, this variable re ordering is also one way to minimize the logic implementation, implicitly. So, we are not talking or thinking about gates circuits minimizing, we are thinking about the function the BDD representation, we are trying to find out what kind of variable ordering can give us a BDD which is smaller.

Now, a smaller BDD may mean a smaller circuit. So, we are thinking about a correspondence like that and, during the construction of the BDD itself as we are seen some redundancy are implicitly removed. So, the idea is that when we use a BDD, we are already carrying out some minimization function minimization while you are constructing the BDD, while you are applying the reduction and merging rules to get the ROBDD, where already applying some minimization right ok.

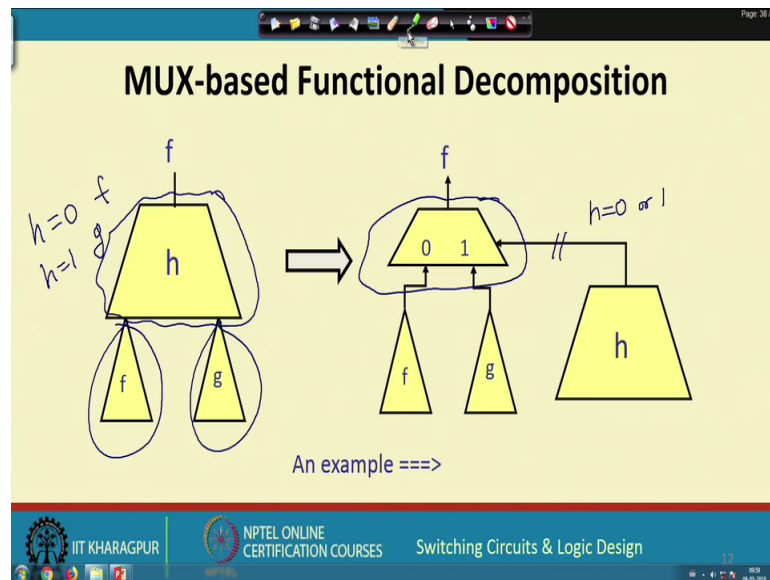
(Refer Slide Time: 22:19)



Now, let us talk about the multiplexer realization of functions, you think of a scenario like this decision node in a BDD, labeled x it is low child represents a function f , the high child represents the function g . Now, if I want to implement this decision node by a multiplexer while a multiplexer symbolically sometimes denoted like this, like trapezium this is your select line.

This is your multiplexers select line, this is your multiplexer output and this side are the inputs and 0 1 indicates, if x equal to 0 which is selected f is selected x equal to 1 g is selected. So, this is a exactly what the decision node means, if x is 0 you come this side, if x is one you come this side. So, this can be directly mapped to a multiplexer and you can repeatedly do this is f and g can be further decomposed you can have more multiplexers generated for f and g fine.

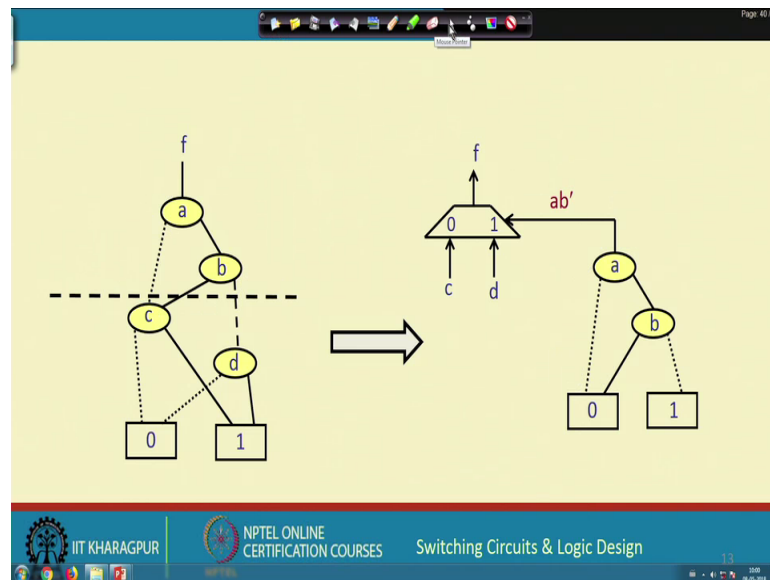
(Refer Slide Time: 23:43)



Let us consider some scenario like this, well you have two sub functions f and g well I am showing some segments of BDD and, you can have a decision block here which can be represented by h it means if some condition h is true, then you go to g if h is false, then you come to f ; that means, if h is 0 then f if h is 1 then g .

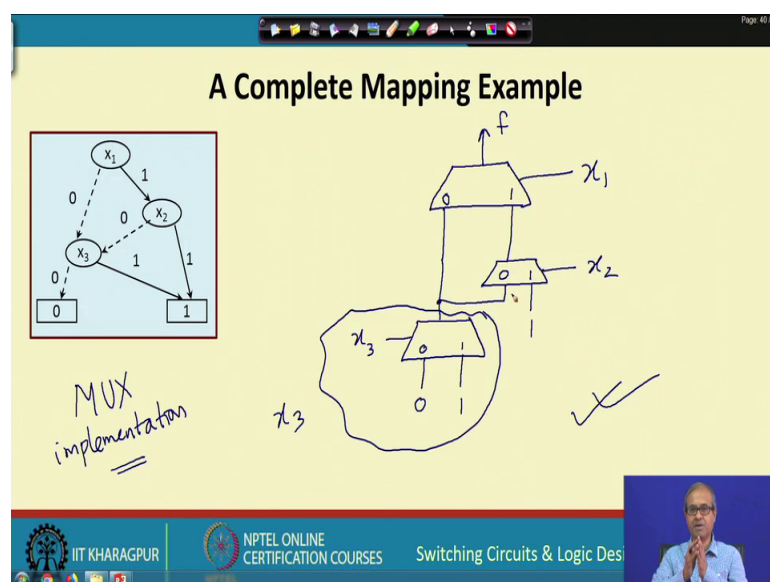
Now, if we have a scenario like this if you can identify this h , then you can again realize this using a multiplexer what this select line is generated from h , because depending on h equal to 0 or 1, you will be selecting either f or g if it is 0 if selected, if it is 1 g is selected right.

(Refer Slide Time: 24:51)



Let us look at a concrete example here, considered a scenario like this where you can say this part indicates h . And this is our c and d or our means inputs so, you can have a multiplexer like this see can be fed to one side, d can be fed to the other side and this h whatever it is this h network, this can be used as the select line. So, you can have a mapping for the multiplexer realization like this ok.

(Refer Slide Time: 25:37)



Let us work out a complete mapping example with respect to the BDD that we had taken as example earlier, this was the ROBDD that were generated from the function. Now,

what we want to know now is that let us have a complete multiplexer based implementation for this function. So, how do we do this? We start from the root side, let us take x_1 we take 1 multiplexer here; the output of the multiplexer generates the function f .

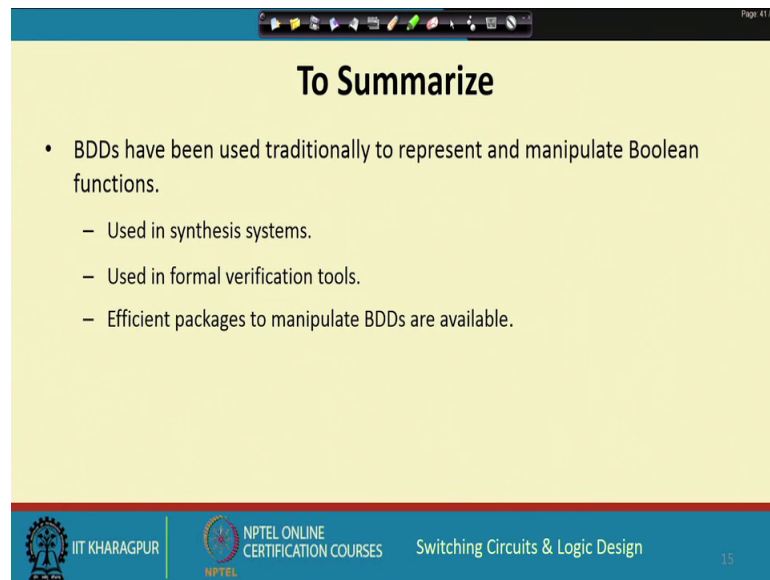
The select line is connected to x_1 , now there are 2 0 and 1 now we will be having two inputs here, one is for select line 0 other is for select line one let us show them like this. Now, if it is select line 0 where do you go you come to here x_3 so, there will be another multiplex out here, this will be selected by x_3 . So, again they will be a 0 input and a 1 input and for x_3 , this 0 input is connected to 0 and the one input is connected to 1. So, you can directly connected it to 0 you can directly connect it to 1.

But for the other case here you have x_2 so, here you have a multiplexer, which will be selected by x_2 and the 0th input will be connected to x_3 so, this one so, x_3 node is indicated by this. So, this same thing will go here 0 input and the one input is connected to 1 so, one input is connected to 1. So, you see this BDD using a pure multiplexer realization you can have something like this.

But of course one thing you understand, that this may not be the best possible realization for example, if you look at this part what does this function mean, extremely select line 0 here and 1 here, it means if x_3 is 0 you send 0 if x_3 is 1 you send one this is as good as x_3 , you can eliminate this multiplexer and you can directly connect x_3 here ok.

So, such minimization are possible, but in general for a large BDD for every decision node you can directly map them into a 2 2 1 multiplexer. So, this is a very convenient and easy way to map a BDD to a multiplexer network. This is one of the ways in which synthesis of circuits using multiplexer kind of a network can be carried out ok.

(Refer Slide Time: 29:04)



The slide is titled "To Summarize" and contains a bulleted list of applications for Binary Decision Diagrams (BDDs). The list includes: BDDs have been used traditionally to represent and manipulate Boolean functions; BDDs are used in synthesis systems; BDDs are used in formal verification tools; and efficient packages to manipulate BDDs are available. The slide footer includes the IIT Kharagpur logo, NPTEL Online Certification Courses logo, the course title "Switching Circuits & Logic Design", and the page number "15".

- BDDs have been used traditionally to represent and manipulate Boolean functions.
 - Used in synthesis systems.
 - Used in formal verification tools.
 - Efficient packages to manipulate BDDs are available.

So, to summarize we have talked about BDDs, now BDD has many applications, they have been traditionally used to represent and manipulate switching functions of Boolean functions in many way, they are used to generate circuits which are called synthesis, they are used to verify the operations of circuits, there is a branch called formal verification where they are very heavily used.

And there are many software packages which are available already in internet which you can download and use for manipulation of BDDs. So, you can create a BDD for a function you can minimize them in various ways you can do (Refer Time: 29:46) given to BDDs, you can find the and of the 2 BDDs or you can complement a BDD lot of such operations can be carried out, these are all supported by the BDD tools. So, with this we come to the end of this lecture. So, what the last couple of lectures we talked about binary decision diagram, which is a very important data structure that is used to represent and manipulate switching functions.

Thank you.