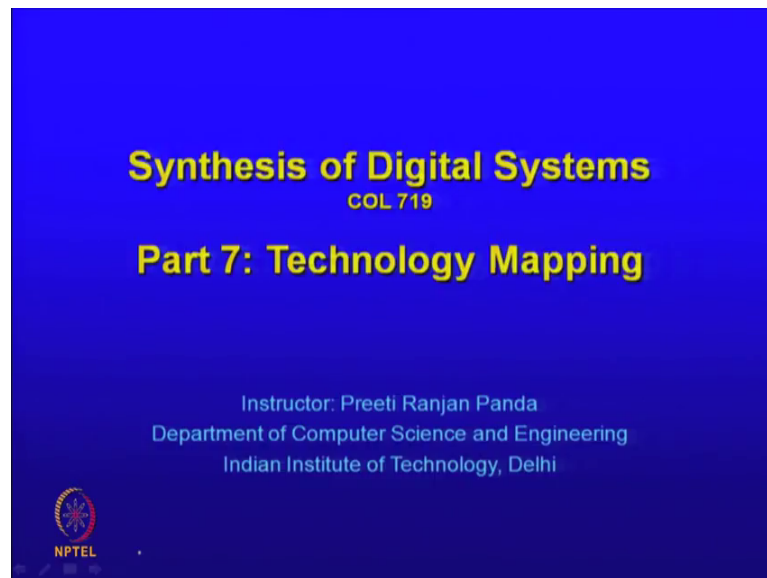


Synthesis of Digital Systems
Dr. Preeti Rajan Panda
Department of Computer Science & Engineering
Indian Institute of Technology, Delhi

Lecture – 24
Multi-level Logic Synthesis: Technology Mapping

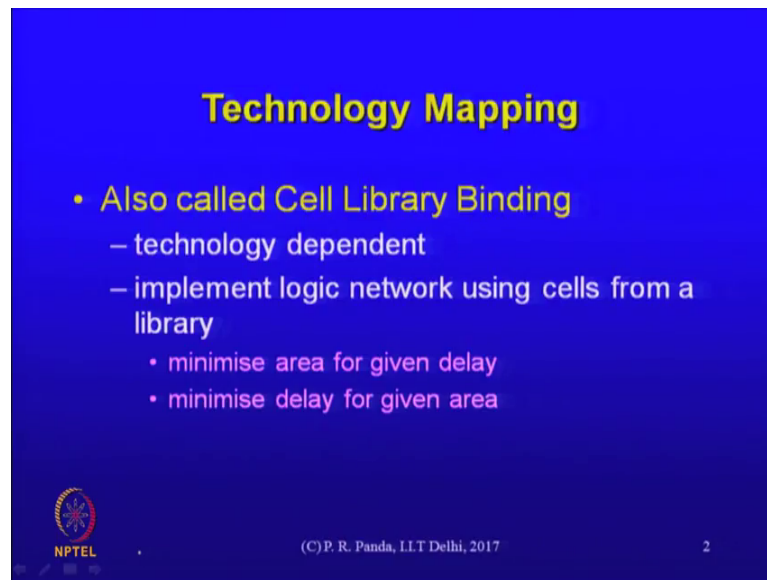
Technology in mapping is the second phase of multi level logic synthesis. What we did so far is the first phase in which we assumed some simple library we did not go into the properties of the cells in the library. These arguments were in terms of literals which are a little higher levels of abstraction.

(Refer Slide Time: 00:44)




Now once you have done those optimizations there is the need to realize a logical net list that we have come up with in terms of physical cells chosen from a standard cell, library or any library so that is what the technology mapping.

(Refer Slide Time: 01:10)



Technology Mapping

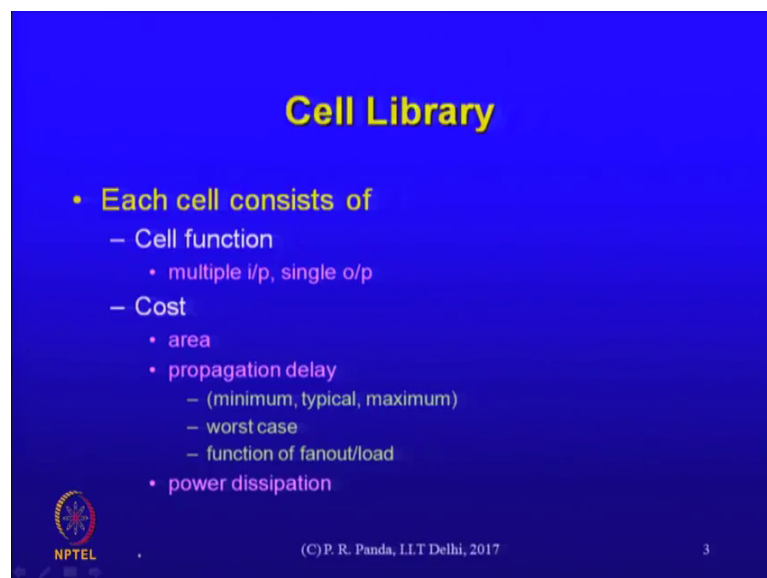
- Also called Cell Library Binding
 - technology dependent
 - implement logic network using cells from a library
 - minimise area for given delay
 - minimise delay for given area

 (C) P. R. Panda, IIT Delhi, 2017 2

It consists of its also called cell library binding. Because these are cells that you choose from an individual library and realize a logical net list in terms of those cells, this is the technology dependent part of logic synthesis.


So, what it means is that in the first phase you pretty much keep the same irrespective of what is the target library, but the second phase is target library dependent here too you could specify different objective functions, it could be minimized area for a given delay, or minimize delay for an area constraint and so on.

(Refer Slide Time: 01:57)



Cell Library

- Each cell consists of
 - Cell function
 - multiple i/p, single o/p
 - Cost
 - area
 - propagation delay
 - (minimum, typical, maximum)
 - worst case
 - function of fanout/load
 - power dissipation

 (C) P. R. Panda, IIT Delhi, 2017 3

So, there are two inputs one is their optimized logic network that we came up with, but the other important input here is a cell library where each cell consists of information that should be useful to us in making that decision. So, the function of that cell has to be captured in some way so that when we see some logical gate we ought to know whether this cell is suitable for that or not.

We can assume for example, that these are simple multiple input cells with a single output, but beyond that various parameters can be included in the cell that helps us in various synthesis decisions. This would include area and propagation delay of various types, it could be that you have one delay and that is some average delay or it could be a maximum delay, it could be a minimum, typical, worst case.


It could also in general of course, be a function it need not be a number because the propagation delay through a cell of course, consists of other things beyond that cell itself right that specifically it is a function of fan out of that cell. A number cannot really be hardcoded as the delay of a gate because there are other dependencies, but those other fan out kind of information at least partially the load is available to us when we look at the entire net list.

So, we could capture the propagation delay in terms of some intrinsic properties of that cell itself, but also some external properties and how the delay depends whether it is delay or power or whatever the function is. And how it depends on the external parameters those could also be encoded as a property of those cells so that we can take the appropriate decisions when we decide whether or not to use the cell.

(Refer Slide Time: 04:18)

Mapping Problem

- Find an equivalent network whose internal nodes are cell instances
 - minimising an objective function



(C) P. R. Panda, IIT Delhi, 2017

4






Our mapping problem can then be find an equivalent network you start with a network, but you find an equivalent network whose internal nodes ultimately are all cell instances and some objective function has to be optimized in the process.


(Refer Slide Time: 04:37)

Technology Mapping

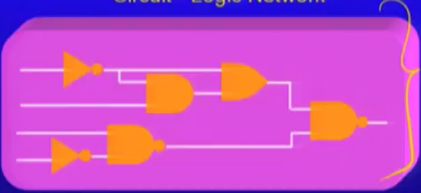
Implement given circuit with library cells minimising cost

Cell Library

Cell	Cost
	1
	5
	3
	2
	6



Circuit - Logic Network



(C) P. R. Panda, IIT Delhi, 2017

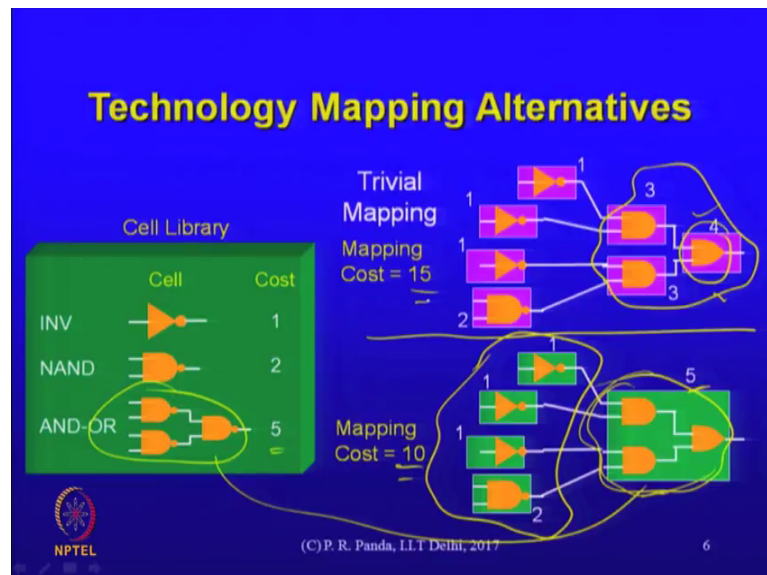
5

Let us just quickly illustrate technology mapping with an example. So, what we have here is a typical logic network. So, far we do not know anything about the physical properties these are just logical gates there are chosen. So, as far as technology mapping is concerned this is just a specification of the functionality of that circuit. Now I need to

realize that in terms of elements that are chosen from cell library associated with each of those cells let us just put a cost this is a simplification of course, the real cost might actually not be a constant for example, it could be a function.

But if I choose an inverter let us say the cost is 1, if I choose this AND gate followed by or that cost is 5 and so on. This is an annotation of the cost of each of the individual cells our problem in technology mapping would then be find a way of realizing that logic network in terms of elements selected from that is a library in a way that the cost is minimized total cost is minimized.

(Refer Slide Time: 06:03)



A trivial solution should not be difficult we can just have each of these elements of the network realized in terms of some combination of cells from the library that cannot be hard because these are gates. And of course, each individual gate is capable of being implemented with some combination of cells a nand gate is good enough for us to realize anything any combinational function. So, specifically an or can be implemented a nand can be implemented an inverter can be implemented everything can be implemented of course.

So, the trivial one would be you just perform that translation locally take every gate and just find out which is the library element that can realize that gate. If there are more than 1, then you just select the one that has lower cost I could do that, but of course, I would like to do this in a way that the cost is minimized.

Problem of course, in the trivial mapping is that alternatives may be there. The solution chosen was a local one which is each individual gate we chose the component with the lowest cost, but what we skipped is that it is possible to sometimes combine several gates and realize that combination of gates in terms of maybe 1 cell from the library ok.

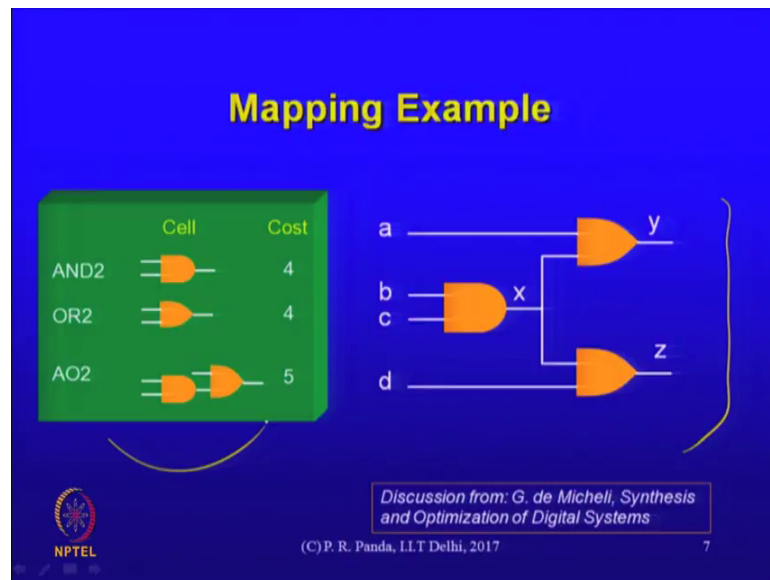
And the costs in the process might be different it might be different from the sum of the costs of realizing the individual gates in terms of individual library elements. So, it could be that I have this and or library element and there is an and or structure there in my logic network also.

So, I can through a matching process come to the conclusion that at least that part of this circuit could be realizable in terms of something that I can take from the library and in the process I come up with a cost that is the cost that is annotated here that cost could be smaller than the sum of the costs. If each of these gates were to be individually realized in terms of some other components that is expected of course, somebody has designed an and or gates means that it is most probably a more area efficient whatever the cost metric is if it is area.

Then it might have been laid out in a way that is more compact than if you lay out 2 discrete structures of those gates and then wire them up that is an expectation. When there is a manually designed component that is more complex, you can expect that its implementation would be smarter than if you take discrete components and put them together ok.

So, if I do that then all these 3 gates are realized in terms of one component and these gates are still the same. So, my total cost then would be the sum of these costs which is then the original mapping was 15. So, you can say there is some leeway here for an optimization function to work with.

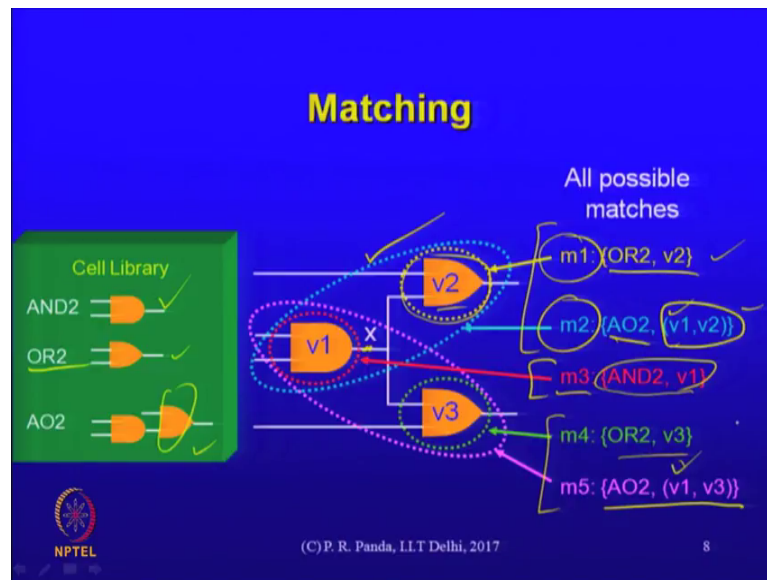
(Refer Slide Time: 09:58)



Let us try to formalize this idea of technology mapping by taking an example, and building up the logic which essentially forms the algorithm for doing technology mapping efficiently. This discussion is straight from the book De Micheli's synthesis and optimization of digital systems. So, take a look at that for more details, but this is the same example that is discussed there and we can take a look here just 3 gates.

And I have a small library here in terms of whose elements I would like to realize that logic network ok. What is nontrivial about this example its just that I have multiple choices that is all, but when there are multiple choices how to resolve it and how to minimize the total cost is the question.

(Refer Slide Time: 11:06)



But first of course, I need to make sure I realize some implementation and then there is a question of doing it in an efficient way minimizing the cost. So, the first part is let us just make sure that we come up with the right set of matchings or realizations of those gates so that we are able to at least generate a netlist generate some netlist.

So, here are the set of matchings that we can identify take one of these gates like that OR gate, and ask the question of which of these cell library elements are suitable for realizing that OR gate v 2 this is good that OR gate of course, is already there and I can use. But in this example that and or is also good because this part of it is matching our v 2, but as it turns out one of these inputs is the and of 2 other inputs right one of the inputs to the OR gate is the and. And in fact we do have such a structure at the input of v 2.

So, there are 2 matching is possible to just cover v 2 it could be that I use that OR 2 gate, for v 2 that is one matching. Alternatively I could select that and OR 2 gate which will together cover both v 1 and v 2. So, both of them are possibilities each of these we are calling a separate match that is m 1, m 2 refers to v 2 being matched by that and OR 2 gate.

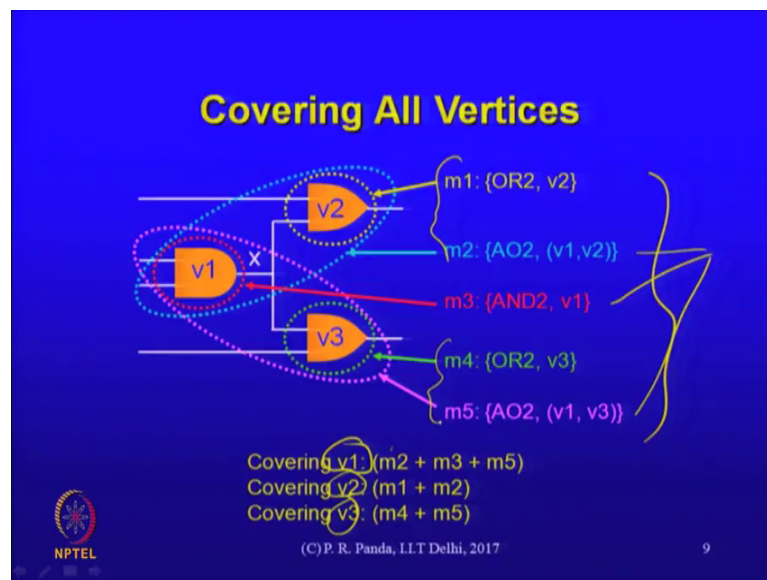
In fact, the combination of v 1 and v 2 both of these that is this ellipse both of these are together matched by 1 and OR 2 component we can do this for all the gates that we have between them the optimal solution must lie. So, question of course, is what set I should choose what subset of those matchings I should choose because I do not have to choose

all of them if I have chosen m_1 there is no need to also choose m_2 fine. So, for v_1 what are the choices there is a AND gate; 2 input AND gate that is all I do not have other choices.

So, there is a matching m_3 that matches v_1 with AND 2 when I say matching of v_1 what I mean is the output is fixed these are single output gates possibly multiple inputs, but single output. So, whatever I choose that at the end must give an output that is the output of the gate I am trying to match so that is how I am defining my matches. So, for v_1 there is only one match independently if I were to generate that output of v_1 .

Similarly, for v_3 I would have 2 choices one is I could use OR 2 as I did with v_2 , but I could also use that and OR 2 which would cover both v_1 and v_3 . So, these are all my matchings I can enumerate all the matchings for the gates taken 1 at a time ok. So, I have 2 matchings for v_2 , one for v_1 and 2 for v_3 . When I say for v_3 I just mean the output realizing the output of that gate of course v_1 is also occurring here and is also occurring here so that is my enumeration of the matches.

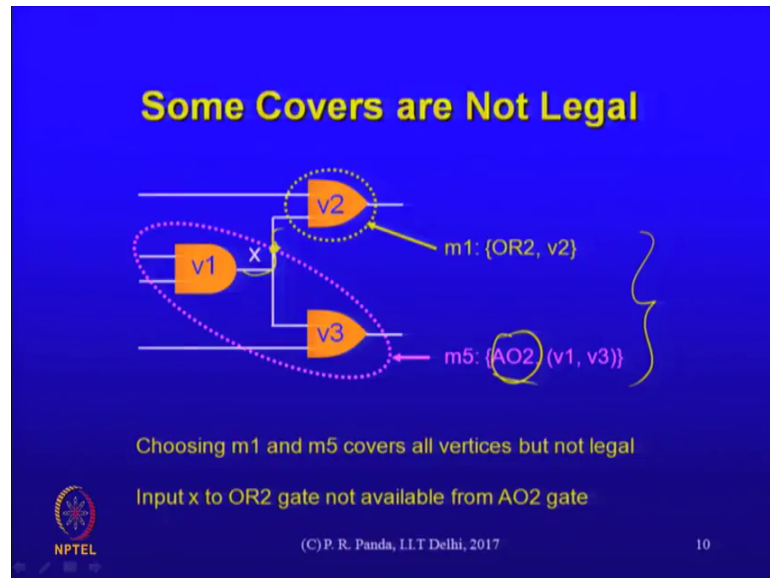
(Refer Slide Time: 15:10)



As part of my mapping process I must have the requirement that all the gates must be covered. So, covering all the vertices, are all the gates, it means that I have to choose some subset of these matchings all the matchings I have enumerated here, but it would be redundant to use all of them in my actual technology mapping because there is redundancy among them.

How do I identify those redundancies covering of v 1 is possible if I select either m 2 or m 3 or m 5 any one of them is right because each of them covers v 1 any one of them I could actually select and realize my v 1. Similarly covering of v 2 means that I could use either of these 2 covering of v 3 means I could use either of these 2 my requirement of course, is that all 3 must be covered.

(Refer Slide Time: 16:32)

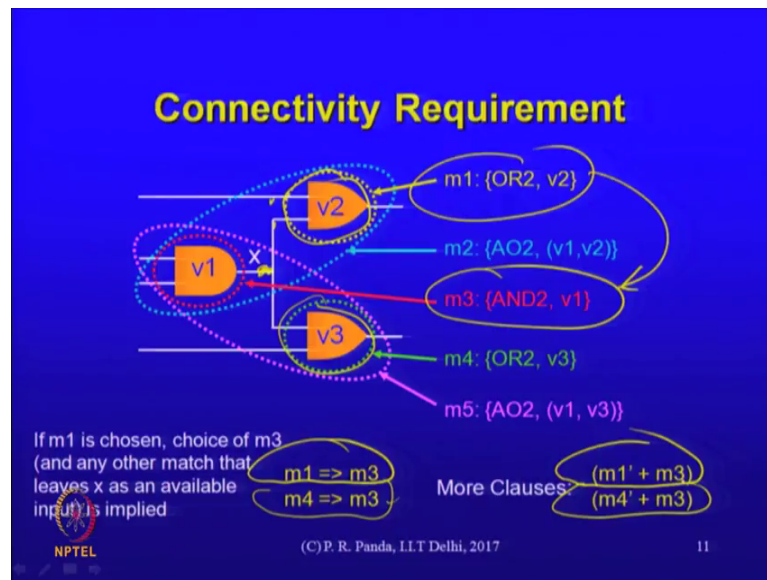


There is one thing to note here there are some constraints some covers might not be good enough given the connectivity that is there in the netlist that can be illustrated by us considering that pair of matches m 1 referred to v 2 being matched with that OR gate just that 2 input OR gate, m 5 referred to v 1 and v 3 together being covered by that and all.

So, between these 2 they actually cover all the 3 vertices, but that library element and OR 2 does not really give us that output of the and function as an explicit output of the cell it might not the designer has designed it in some way he might give, but it might not give.

Particularly if our assumption is that all the elements have 1 output then of course, it is not there we could relax that, but in general it might be there it might not be there when you pick up some element from a library that specific output internal output that you are looking for might not actually be present. So, if it is not present then that does not lead to a legal cover even though all the nodes are actually included in it. So, I need to somehow take care of that.

(Refer Slide Time: 18:08)



So, a few other clauses I may need to put in some clauses I have already identified these are Boolean clauses that first said any one of these matches must be true, must be selected. But this connectivity is an additional requirement which we can identify as follows. If m 1 is chosen right if I choose this then the choice of m 3 is implied. Do you see why? I actually have multiple choices, but as it turns out that m 2 is not a choice and in fact, I should since these 2 are the inputs that I need.

So, for producing the x I need to choose the appropriate matching if I were to use m 5 then that x would be hidden. So, I can come up with a new implication that says m 1 implies m 3. Similarly I can say m 4 the choice of m for that OR gate implies m 3. Now this is a new requirement set of new requirements that come about from the need to establish connectivity properly in the selected maps.

How do you encode that in terms of a Boolean function the earlier part we know how to encode as a Boolean function we already did. We had these clauses that said m 1 plus m 2 plus m 3 which means that from those 3, one of them must be chosen that's fine, but this implication how do you capture that as a Boolean function?

(Refer Slide Time: 20:28)

$\overline{m_1} \Rightarrow \overline{m_3}$

m_1	m_3	$m_1 \Rightarrow m_3$
0	0	1
0	1	1
1	0	0
1	1	1

$\overline{m_1} \overline{m_2}$
 $\overline{m_1} + m_2$

NPTEL ETSC, IIT DELHI

Are you familiar with this? What logic captures the Boolean function m_1 implies m_3 both are Boolean values m_1 is a Boolean value what does it tell you if it is true then?

Student: (Refer Time: 20:46).

First of all are we clear that these m_1 , m_2 , m_3 , m_5 all of these are just Boolean variables right if it is true it means that you have selected that matching. If it is false it means that you have omitted that matching and out of all the matchings that we have enumerated we have to select some subset that subset must cover all the gates, but it must also satisfy the connectivity requirements that is my problem. So, these are Boolean variables m_1 implies m_3 . What Boolean function is associated with that implication it must correctly capture my requirement we can write out the truth table it must cannot be that difficult sorry.

Student: (Refer Time: 21:55).

So, anyway there are only 4 conditions. So, we can analyze each one of them to find whether the suggested implication is violated where is it violated where is it holding if its violated we will call it false. We will call the implication false right otherwise it is true what is there on the output column where is it violated for which combination of m_1 and m_3 is it valid violated.

Student: Explicit the first time (Refer Time: 22:27).

Violated means let us just call it 0 if it is not violated everything is fine then we will call it 1. So, which ones would be 0 here.

Student: (Refer Time: 22:39) first second 0 (Refer Time: 22:41) first and third (Refer Time: 22:45).

This is 1 of course.

Student: (Refer Time: 22:48).

m_1 implies m_3 must mean that if m_1 is 1 then m_3 must be 1 that is my requirement that is violated by which of these combinations. It cannot be violated by these 2 because I do not care about the conditions where m_1 is false statement was if m_1 is 1, then m_3 must be 1. So, if m_1 is 0 then it is not even captured by that constraint it must be that it is true yeah it is not violated so we will call it fine where is it violated actually.

Student: (Refer Time: 23:32).

It is only here that it is valid this is of course, fine that is my truth table and that function is just that so if you have m_1 being true and m_2 being false then that is what we do not like. So, the function implication function is essentially just this yeah m_1 , m plus if this is the standard logical function to capture implication I can do that. So, if I say m_1 implies m_3 then the clause that is inferred from there is m_1 prime plus m_3 .

If I say m_4 implies m_3 then the clause that is implied from there is m_4 prime plus m_3 . Any questions on this implication this is this is a standard logical function and it its realization would be in terms of such a Boolean logic

(Refer Slide Time: 24:58)

Overall Requirements

- All vertices must be covered
- Connectivity requirements for each match must be satisfied

$$(m_2 + m_3 + m_5)(m_1 + m_2)(m_4 + m_5)(m_1' + m_3)(m_4' + m_3) \neq 1$$

- Find solution (truth assignment to all variables) such that
 - Equation is satisfied
 - Boolean Satisfiability Problem
 - Cost is minimised

NPTEL (C) P. R. Panda, IIT Delhi, 2017 12

So, then our overall requirements are just these 2 one is that all the vertices must be covered all the connectivity requirements must be met. And how do we make that happen we derived those implications and those implications lead to other clauses that must also be satisfied in addition to those covering related clauses. So, what do I have I had these clauses being inferred from the covering requirement because every gate translated to 1 clause here, for one of the gates I could have chosen either m_2 , or m_3 , or m_5 .

So, this was that AND gate that I had for the other 2 gates I had m_1 plus m_2 , I had m_4 plus m_5 . So, these are my covering requirements these are my connectivity requirements and all of them must be met this refers to that gate that it corresponds to the gate that it was derived from that clause must be satisfied which means that either m_2 , or m_3 , or m_5 must be true. For the second gate m_1 , or m_2 must be true for the third gate m_4 , or m_5 must be true all of these 3 must be true for me to cover all the gates of the netlist.

Similarly, all of these must be true for me to satisfy all the connectivity implications and therefore, I take the product of all of those clauses that must be true means that all clauses must be true that could be a way to capture our logical requirements for a legal covering. This does not capture the cost this only captures the feasibility because multiple solutions might be there for this then finding a solution refers to what giving a truth assignment finding what value for m_1 , m_2 , m_3 , m_4 , m_5 first of all results in that equation being satisfied.

This is actually nothing, but the Boolean satisfiability problem given a Boolean expression the satisfiability problem is to assign values to the variables such that that expression is satisfied ok. So, this is what this is the cost part of it we still have to get to, but from a feasibility point of view that satisfiability problem needs to be solved of course, we have already seen how to solve this.

Student: Yes sir or obedient (Refer Time: 27:56).

Yeah if you were to build the a (Refer Time: 27:59) for that function then the path from the true node. All the way to the top gives us a set of assignments that will lead to that functioning satisfied, so that is the overall idea.

(Refer Slide Time: 28:11)

Matching Approaches

- **Structural Matching**
 - check for isomorphism
 - general graph isomorphism intractable
 - tree-matching easier
- **Boolean Matching**
 - more general
 - e.g., $(a'b' + b'c + ab)$ matches $(a'b' + ac + ab)$
 - more complex

The slide includes a diagram with yellow arrows pointing from the 'Boolean Matching' section to the structural matching sub-points, and a yellow oval around the example expressions. The NPTEL logo is in the bottom left, and the copyright notice '(C) P. R. Panda, IIT Delhi, 2017' and page number '13' are in the bottom right.

So, a couple of things here one is we have to enumerate those matches, but I informally pointed out that you take one of those gates, and two of the library elements could match that gate or 3 of the library element could match that gate. We somehow have to formalize that a little bit that is the first part that in fact is the easier part.

Because not that many elements will match libraries would have 10s, or 100s of elements and some subset of them will match it cannot be too hard to find that matchin. But in general that matching could be structural which means that I could just check for an isomorphism.

If both of these were represented as graphs both my netlist is represented as a graph and those library elements are also represented as small graphs some of them are just single node graphs, but otherwise they are 2 or 3 or such a small number of nodes. Then what we are looking for could be a check for isomorphism even though the general graph isomorphism is an intractable problem the I kind of isomorphism we are talking about here where one of those patterns is actually very small might not be too expensive.

But even then the matching could be better if you assume that the structure that you have is a tree rather than a general graph we will make that assumption even though the general netlist is not a tree it is of course, there is a combinational function and the dag is what is the standard structure representing that netlist.

It turns out that many of these algorithms are difficult if you want to solve them optimally assuming it is a dag, but if you assume that it is a tree then efficient solutions are there. So, one standard heuristic could be you take that dag and just cut some of the edges of the dag and make it a tree artificially even though it was not a tree originally you make it in a way that other things would still be satisfied.

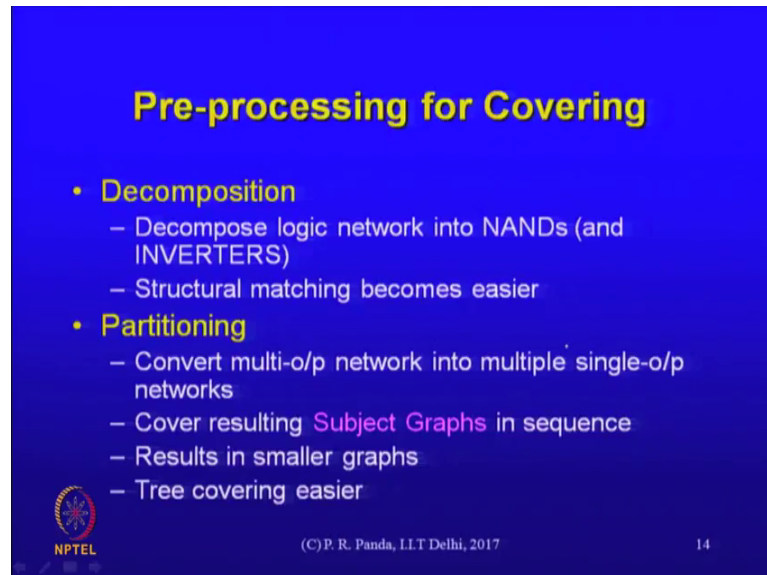
But ah, but then you can run your algorithm matching and mapping algorithms by assuming that there are tree or you divide a graph into multiple trees and apply the optimization algorithm on the individual trees that is the approach that is taken here. The other thing to note here is that structural matching is not the only kind of matching that is involved the more general matching is what is called a Boolean matching where you have some function here that in general could match some other function like that ok.

Maybe this is there in the library and that one is not that this is there in your logic network and you are trying to find out elements from the library that are matching and in fact, this element is there, but establishing that these 2 are equivalent is a different problem though because structurally they are not equivalent. You have a prime b prime plus b prime c plus a b I have a prime b prime plus a c plus a b turns out that both are equivalent if your other truth table you only see that they are equivalent, but it is not immediately obvious of course, you can build the bdds for each of them and then trivially establish that it is equal.

But then that does bring construction of the b d d into the picture here. So, this is in general the more accurate thing to do the Boolean matching, but often we do not do this

because it might be expensive we can do with just structural matching here and give up some optimality in the process ok.

(Refer Slide Time: 32:26)



Pre-processing for Covering

- **Decomposition**
 - Decompose logic network into NANDs (and INVERTERS)
 - Structural matching becomes easier
- **Partitioning**
 - Convert multi-o/p network into multiple single-o/p networks
 - Cover resulting **Subject Graphs** in sequence
 - Results in smaller graphs
 - Tree covering easier

NPTEL (C) P. R. Panda, IIT Delhi, 2017 14

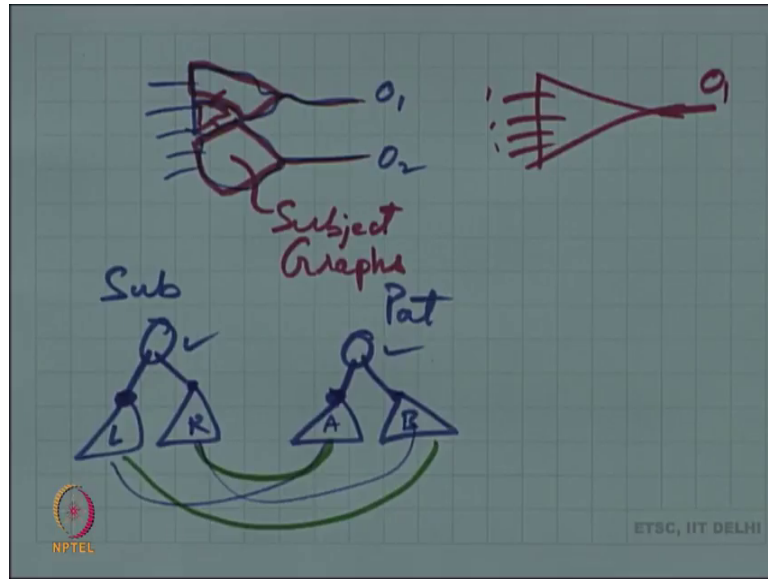
So, I would like to perform this covering and we could do some initial pre processing just to make the algorithm simple.

Let us do a decomposition of that logic network into just nands remember it is just a specification of the functionality. So, far what is there in that logic network which we want to realize in terms of the cell library gates that is just the functionality you could restructure it and whatever way you want as long as its equivalent.

So, specifically I can convert it into just 2 input nands it is always possible and if there were inverters then its fine its a special case of the nand gate of course,, but that will help us since we are relying on structural matching it will actually help us in performing this structure matching because it is simpler you could generalize that to other kinds of gates also, but the basic strategy could be illustrated using just this.

So, you take whatever the network that was given and decompose it into nand gates just 2 input nand gates the other preprocessing I might want to do is a partitioning of an original multi output network into multiple single output networks. Whats happening here?

(Refer Slide Time: 34:03)



So, suppose you have some logic like that and some other logic like these are the inputs and those are the outputs. The matching and covering process we want them to be trees. So, these multiple outputs unfortunately might be overlapping in terms of the logic that is there, but you can divide that into multiple graphs that might be independently it might be considered please for us to just to simplify the covering process later on.

So, how do you divide this that could be 1 tree, this could be 1 tree and that part here could be a tree now its possible that there are multiple outputs. So, from here there is a single output from here there is a single output, but this one I have to see how many outputs are there it is possible that there are multiple outputs between that.

Student: (Refer Time: 35:25).

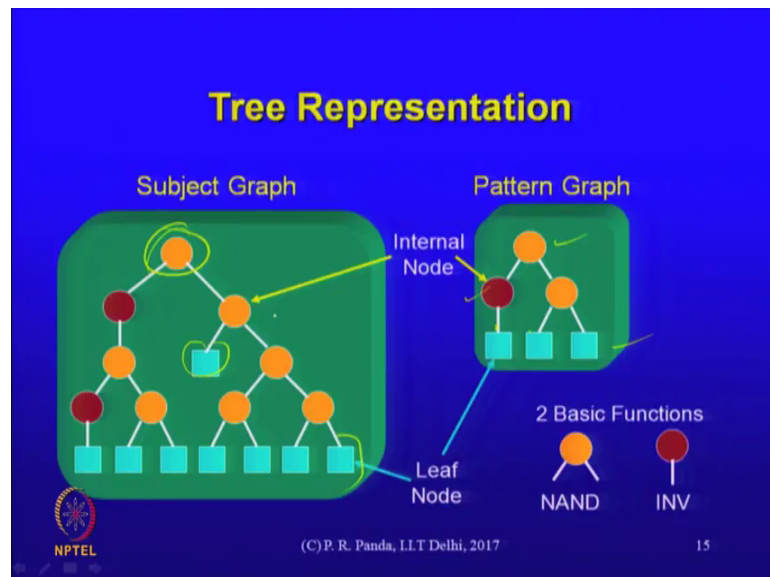
Right if that is the case then I can split that further. So, in general the kind of structure that we want to cover is this I have one output and I have a bunch of inputs and this is a tree structure that I have there are some nice properties of these algorithms that work on trees we can guarantee optimality in terms of the covering.

So, where optimality is broken is in the process of dividing that dag in 2 trees and that is hard to do there may be many choices which one is the best is not clear, but a tree itself can be covered in an efficient way ok. So, these are what we are calling subject graphs that need to be covered. Once you have made this partition each of these problems is

independently solved you take each of those trees and do the covering yourself they are anyway not overlapping here in the logic.

So, you realize them in terms of different gates what it also means is that if there was some logic from one tree, that could have been shared with some logic of the other tree in terms of an efficient realization of gates that you are giving up in the process of doing this partitioning into trees. But there is some efficiency that results one is that individual algorithms are efficient, but also the tree is smaller this is a bunch of smaller trees. Therefore, you can expect that those algorithms will converge faster alright.

(Refer Slide Time: 37:09)



So, we have a subject graph and a pattern graph. The subject graph comes from the network that we want to cover where does the pattern graph comes?

Student: (Refer Time: 37:22).

From the cell library this is the graph that corresponds to an individual cell. So, the purpose of the matching process is to find whether the particular cell under consideration can be used or not does it match that node of the subject graph or not that is the question.

So, the way we represent this is now there are 2 kinds of nodes under consideration here remember we broke everything down into a nand gates and inverters maybe. So, these are nand gates all of these and nand gates and these brown nodes are inverter right.

Then these are the leaf nodes we just put these dummy nodes in to help us in the algorithms, but in the subject graph that would correspond to the primary inputs of the combinational logic that we are trying to cover. In the pattern graph it just means that this is a partial covering right and whatever corresponds to those leafs you may now have further cell library elements that would match there so right.

So, that is what I have 3 kinds of nodes one is the 2 input nodes other is a single input node and the third one is a leaf node that is all that I have. And there is only one kind of node to remember there is only nand gates so they are all actually equivalent orange node is a 2 input nand gate there is nothing more complex in terms of the functionality.

(Refer Slide Time: 39:07)

Tree Matching -1

```

MATCH (u, v) {
  u - Pattern Graph node
  v - Subject Graph node
  if (u is leaf) return TRUE
  ...
}

```

Subject Graph

Pattern Graph

Leaf of Pattern Graph reached.
Subtree rooted at v will be matched by different pattern.

NPTEL
(C) P. R. Panda, IIT Delhi, 2017
16

So, how do I go about matching suppose I am at a particular node u, and a node v, node u from the pattern graph and node v from the subject graph I am trying to establish whether the matchings is or not. So, we start off with the simpler cases saying if u is a leaf then it is ok. Why is it ok? There is a traversal that is happening this implies a recursive kind of a process I might end up at the leaf I start off with the roots of both the structures.

But I might end up somewhere and the leaf tells us essentially the termination condition for the matching. We are saying that if we reach the leaf of the pattern node then everything is fine why is everything fine if I am trying to match u with v it means that the v itself can then be matched by some other library element perhaps. But so far

everything is fine if I reach the leaf it means that for example, both of these have been correctly matched that is how I will land up with the leaf of the pattern graph. So, it must be this is a termination condition for the recursion ok.

(Refer Slide Time: 40:37)

Tree Matching - 2

```
MATCH (u, v) {  
  if (u is leaf) return TRUE  
  else {  
    if (v is leaf) return FALSE  
    ...  
  }  
}
```

Leaf of Subject Graph reached.
Subtree rooted at u will never be matched

(C) P. R. Panda, IIT Delhi, 2017

What if v is a leaf that is u an internal node and v from the subject graph is a leaf this is not why?

Student: (Refer Time: 40:59) u have some other nodes.

Yeah u is an inverter there for example, that is an inverter here, but I do not have a need for that inverter v is already at the primary input for the subject graph. So, there is nothing to do I have come down the wrong part here. So, matching will not occur anymore if I reach the leaf of the subject graph and while I am still there at the internal node of the pattern graph of course, if both of them are leaves then what happens.

Student: (Refer Time: 41:32).

Then it is fine I have recovered that.

Student: Not yet.

Sure look at that algorithm. Yeah if you is a leaf mate means that irrespective of whether v is a leaf or not and we have already covered that so what condition is left. Of course, the interesting thing is still left the real matching is still left which is both are internal

nodes. How do we proceed if both are internal nodes then the functionality is limited they are only inverters or nand gates. So, it cannot be that what do we do. So, I am somewhere here and I am somewhere here it does not matter where perhaps I am here and I am trying to check whether the matching should proceed further or not how do I go about it.

Student: if they are match then they go to their (Refer Time: 42:27).

If they are match means.

Student: (Refer Time: 42:30) number of nodes (Refer Time: 42:23) of rooted circle (Refer Time: 42:25).

Number of nodes.

Student: Yes (Refer Time: 42:40) because that will. So, we had a case where these not will that an internal leaf.

Yeah.

Student: (Refer Time: 42:53). So, if number of nodes are not matching in the 2.

But if number of nodes are you know you expect the number as you start from the root you expect a mismatch this is a small thing remember the pattern is small the subject graph is large. So, the matching is if it is partial it is just that is it legal or not for example, it matches does this much that is fine right this part of the subject graph is going to be matched by that cell then I continue the matching with maybe I find some other match here.

Maybe I find some other match there that is fine we just want to know whether there is a match at the root of that graph or wherever I am in my traversal is there a match that is rooted at that node that is what we are looking. So, in the process I landed up here say, and I landed up here, and so that is my u that is my v and I want to know whether these 2 match or not these 2 means that is the start of the match right then there is a sub tree under them that I have to continue the matching. How do I go about this first of all I can compare these 2 nodes. What will you tell us what kind of comparison I can do?

Student: (Refer Time: 44:10).

Yeah there are only 2 kinds of nodes. So, what are the possibilities both can be inverters in which case what can I do what can I say about the matching if both are inverters they are matching.

Student: (Refer Time: 44:23).

We can recurse further to their children because at this level it is ok. So, I just call match down further into the next the child node that is fine. What other things can happen both might not be inverters both might be.

Student: NAND gates

Nand gates then what I do?

Student: (Refer Time: 44:44).

Then it is fine so, but specifically what can I do so that is my subject and that is my pattern. So, I discovered that both are 2 input nodes which means that this is fine. So, what do I have now I have a sub tree here, sub tree here, I have a sub tree here, I have a sub tree here. How do I continue the match these are both matching this is just a structural match then.

Student: (Refer Time: 45:20).

Right I just continue the match for these 2, so these 2 nodes. So continue the matching means that if a false value is returned it should be propagated all the way back right. So, that is what so these two can be matched similarly.

Student: Start only (Refer Time: 45:40).

Right if the left has not matched I must abandon the match and just return false to the parent that you try.

Student: (Refer Time: 45:49).

Anything else the sum property of the nand function that I can use to do this a little more extensively. Left is matched against left and right is matched against right well what is the significance of left and right.

Student: (Refer Time: 46:13).

This is a commutative function right. Therefore, I must also check these 2 right I must check these 2; who knows maybe those 2 if they match then it is fine right that is fine right so that is what I should be doing.

Student: Therefore, In a and In b that will (Refer Time: 46:34) In a (Refer Time: 46:34) if they are not.

Fine if they are not matching you can terminate this match at an appropriate time of course, what if I have an inverter on one side and a nand gate on the other side. There is a pattern node and a subject node right u and v. So, if one is an inverter the other is in the nand.

Student: the terminate (Refer Time: 47:06).

We terminate yeah its falls.

(Refer Slide Time: 47:08)

Tree Matching - 3

```
MATCH(u, v) {
  if (u is leaf) return TRUE
  else {
    if (v is leaf) return FALSE
    if (degree(v) ≠ degree(u))
      return FALSE;
    if (degree(v) = 1) {
      return MATCH(child(u), child(v))
    } else {
      return (MATCH(left(u), left(v))
        & MATCH(right(u), right(v)))
        | (MATCH(left(u), right(v))
        & MATCH(right(u), left(v)))
    }
  }
}
```

Subject Graph Pattern Graph

NPTEL (C) P. R. Panda, IIT Delhi, 2017 18

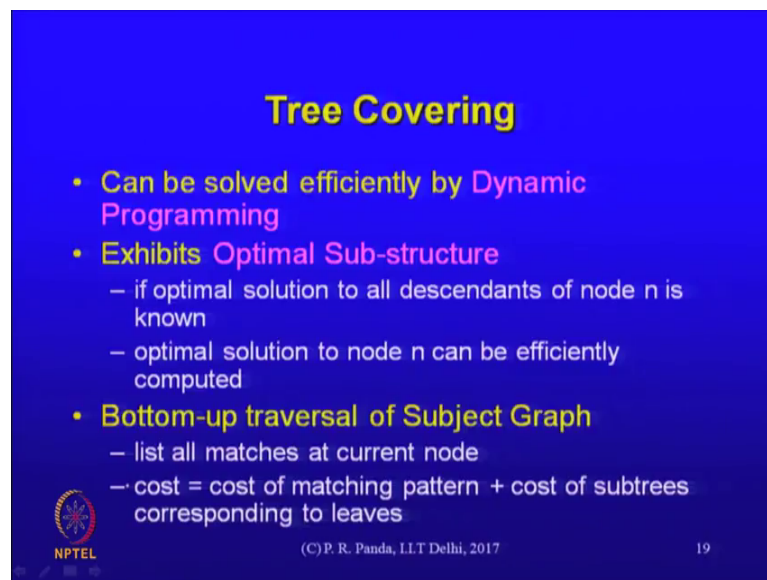
So, that is all that my algorithm has these are the terminating conditions ok. So, what we are checking here one is a visa leaf then I return false one is if u is a leaf I return true if their degrees are not equal then I return false that is also the other simple case.

If the degree is 1 both are inverters then you just match their respective children. Otherwise because of the commutativity property you can match left with left and right

with right or left with right and right with left that is all that you need to do. This is an algorithm that looks expensive because of all these recursions, but in practice it might not be just because the pattern graphs might not be too complicated these are just individual cells in a library right.

So, the algorithm is general, but it may turn out that most of the time we are terminating penny fast actually ok.

(Refer Slide Time: 48:18)



Tree Covering

- Can be solved efficiently by Dynamic Programming
- Exhibits Optimal Sub-structure
 - if optimal solution to all descendants of node n is known
 - optimal solution to node n can be efficiently computed
- Bottom-up traversal of Subject Graph
 - list all matches at current node
 - cost = cost of matching pattern + cost of subtrees corresponding to leaves

NPTEL (C) P. R. Panda, IIT Delhi, 2017 19

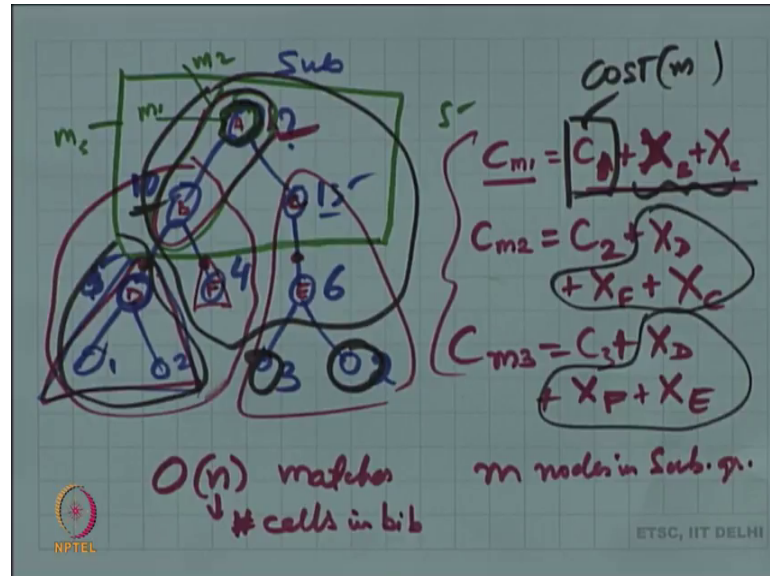
That is magic then there is the question of covering the matching actually is the easier part in this there is not much to do the covering is something that can be efficiently solved by strategy called dynamic programming are you familiar with this does not matter.

So, we will try to develop the argument from scratch this refers to algorithms applied to problems that have a particular property called an optimal substructure means that if optimal solution to all descendants. Now this is a descendant referring to a search strategy in general, but in descendant what it means in our case also should be clear there is a 3 kind of a structure.

So, there is a parent child relationship descendent can be defined in those terms. If the optimal solution is known to all the descendants of a node then that optimal solution to that node can be efficiently calculated that is the property its not always true for all

problems it is not necessarily true, but for this tree covering problem this property holds let us see why it should hold that is my tree right.

(Refer Slide Time: 49:43)



So, in general that tree of course, can be large and such a structure is there. So, what this implies in terms of strategies we could do a bottom up traversal of the subject graph. This is my subject graph we are trying to cover it up with patterns, our strategy could be for each of the nodes we list all the matches this we have already seen how to list essentially I will run match for all the cell library elements.

And the cost that I choose for that node referring to the cost of covering the sub tree you see why this tree assumption was needed here. This argument does not hold if it is a dad in general, but for a tree it is fine the idea of a sub tree makes sense only in a tree. If it turns out that there is an overlap between that part and some other part of the structure then all of this argument does not hold.

Student: Does not hold.

So, I find the cost here for that sub tree irrespective of what is there in the rest of the tree. So, my argument in terms of this optimal substructure is if I have annotated the cost let us say I annotated costs 1, 2, 5, 4, 10, 3, 2, 6, 15 something like that.

Each of these costs refers to the cost of covering that sub tree that is rooted at that node total cost of covering in a way that is optimally computed if that optimal cost we are

somehow computed how it is computed we will have to see, but if I knew the optimal costs of covering all the descendants optimal solution to all the descendants of a node then the optimal solution here can be efficiently calculated that property.

Let us argue that it holds for our problem what is the argument here our cost is each of those gates is associated with the cost each matching is associated with a cost, but of course, there are multiple matchings that are possible. So, at this level there are multiple matches that are possible across all those matches I should be able to efficiently compute the total cost, and so that I can choose the optimal cost for each of them the minimum one I would choose.

How do I do that assuming that all of this is known that we have to establish, but if they were all known then it cannot be a hard problem the optimal tree covering problem because there are a limited number of matches. How many matches are there first case for a given node when I say match we are taking one node there, and we are saying does it match a particular cell library. If there are n cells in the library then a maximum of n .

Student: (Refer Time: 53:15).

N such matches have to be done right. Some of them will come out to be true, some will come out to be false, those that are false of course, they can be thrown out there is no need to look at them further, but maximum of n of them will be true.

Student: (Refer Time: 53:29).

Right. So, if I know that some out of them max of n of course, I know that, but some m out of them will be true those are the ones that I should choose for optimal for computation of the optimal cover how do I do that. So, let us say 5 such cases are there for each of them I can I know where it matched right. So, one of them maybe was just a match of that one node other one maybe was a match of these 2 nodes third one could be a match of these three. So, let us see those are my 3 matchings and one that is a second matching and that is a third matching all of them are they are successful matches then what could be my optimal covering strategy.

Student: Compute the cost.

They are compute the cost how do I compute the cost I just compute the cost.

Student: (Refer Time: 54:31).

For m_1 matching cost, for the m_2 matching, cost for the m_3 , matching and the best one of them is actually optimal. Let us argue what is the cost for m_1 matching? M_1 matching I am selecting that node which means so, what would be the total cost of selecting that node at this level it would be let me give some names here I call it A, B, C, D and so on that would be the cost of selecting m_1 just that will be the cost of A, just that single match which is that is just the cost of that node plus.

Student: (Refer Time: 55:18).

Let me call this instead of C let me just call it D, we call it X this is actually the cost here is what I want to calculate is the subtree rooted at.

Student: (Refer Time: 55:39).

At that node so that is what this is plus $X + c$ that is my total cost that is how I am counting costs right at the point is that it is valid to count costs in this way some other problems it might not be valid to count the cost in that way, but that is all if I were to choose the m_1 match my total cost at the root of that tree would be this I can do this for the others if I were to use m_2 then?

Student: (Refer Time: 56:19).

Yeah I have to first take that pattern that pattern has a cost that is associated with it let me call it c one just implying that it is the cost of the first match whatever that cost is here it turns out to be one node, but in the second case its actually 2 nodes. So, well anyway it is one cell library element whatever it is that is the cost of that library element single library element plus whatever is left out. So, this part of it is left out and that part of it is left out.

Student: (Refer Time: 56:57) 4 node 4.

You are right that is also left out I will call it out and I would just say this is that C_2 plus X of D.

Student: (Refer Time: 57:13).

X of F.

Student: Plus.

Plus x of this is this is a C that is what it is right similarly I can say the cost of using that m 3 would be whatever is the cost of that library element corresponding to m 3 plus.

Student: (Refer Time: 57:40).

Right so, that that is and that right my cost computation and whichever is the minimum out of these I will just annotate as the optimal cost. Here this is a guarantee of optimality of course, it was assuming that my computation of all these sub tree nodes are also optimally computed, but that is not hard to establish right.

So, I start from the leaf at the leaf there is no cost and from the leaf level to the next level. As I go upwards I can always apply this in a algorithm and make sure that wherever I am as long as I have the cost computer optimal cost computed for the children I am able to cost to compute the optimal cost for the parent.

Student: So, we have address how we can optimally calculate the cost of all the various option able to us.

Right.

Student: (Refer Time: 58:53) there are only 3 match.

Yeah.

Student: We can have a very big graph and these matchings can be huge so for.

Yeah

Student: Simple match we said we have at the.

Up to n.

Student: n (Refer Time: 59:07).

Yeah right up to n matches could be.

Student: (Refer Time: 59:12).

Where n is the number of cells in library.

Student: This for single.

That is for single word right.

Student: Now they can first is large number of nodes in the graphs.

Yeah.

Student: (Refer Time: 59:25).

So, let us say there are m nodes in the in the subject graph.

Student: And when we are matching (Refer Time: 59:32) permutation of (Refer Time: 59:35) first AND 2 and then 3 (Refer Time: 59:40).

Yeah.

Student: So, this itself is a huge problem to optimization.

But we just argued at the time of the matching is that the matching is relatively you expect it to be relatively simple task. Because the nodes of library typically are not very complicated logic nodes what kind of logic is there in a library that is complex maybe there is an x or maybe there is an and or it is still a few gates its not very large.

The logic nodes are not very large you could have adders and so on that are treated differently that would not be subject to this kind of optimization. But the stand typical standard cell OR gate array cell libraries are not very complexed we did say that the match algorithm you have to be careful about you take some graph and try to match it with some other graph it is hopeless of course, it can become exponential easily, but the number the subject graph can be very large m can be very large, but the pattern graph you do not expect to be large therefore, you expect it that matching process to terminate fast.

Student: The number of matches on the cell that is probability is very less if we get a large number of.

Yes, even though this n is the worst case n could be 100s, but the number that too you do not expect a very large number those things actually help us cut in this algorithm itself is efficient. This is an optimal algorithm considering what the assumptions have been made the crucial assumption is that I have a tree structure how you get that tree structure is an open problem you can get some tree structure, but establishing that this is the best tree structure is not easy.

Student: Yeah (Refer Time: 61:27).

Validating it is correct is not an issue validating that that is the best way of cutting is the issue you can cut it any way you want if you find there are multiple outputs cut it further. So, it is that process is not difficult you can get a valid cut the problem is to choose the best cut.

Student: So, in this tree (Refer Time: 61:58) although we are converting everything to and (Refer Time: 62:02).

Yeah.

Student: Can we have some other early termination conditions so that maybe we need not enter match itself. Because when we convert things in tune and inverters the complex cells.

Student: At least you would have to go down 2, 3 levels to conclude that it does not match to certain nodes in the graph ok.

Right.

Student: So, because that that will have a so for example, we may have a cells of 6 inputs, or 7 inputs these are gates.

Yeah.

Student: In that case matching is still possible.

Student: So, can we have some early termination itself let us say can we have. So, because that that will have a reasonable for example, we may have cells of 6 inputs or seven inputs these are there.

Yeah

Student: In that matching is still possible so can we have some early termination condition itself let us say

You could remember we left out that Boolean matching.

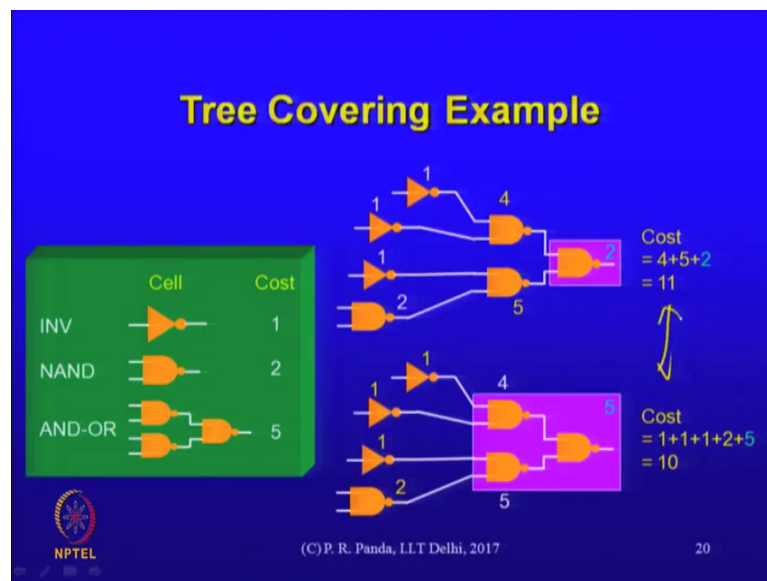
Student: Right.

We said that it is the complicated, but In fact, that is the direction you could be a little smarter the structural matching is a little dumb in that sense you could make that.

Student: (Refer Time: 62:51).

Matching a little more sophisticated that way we actually worked out everything. So, you hopefully that do not need the further analysis the examples of course, is fine.

(Refer Slide Time: 63:09)



These two costs are different for reasons that you know understand perfectly.

(Refer Slide Time: 63:15)

Tree Covering Algorithm

```
TREE_COVER (V, E) {
  COST (v) = -1  $\forall$  internal vertices v
  COST (u) = 0  $\forall$  leaves u
  while ( $\exists$  node with -ve cost) {
    select any v  $\in$  V whose children have COST  $\geq$  0
    M (v) = set of all matching Pattern Graphs at v
    COST (v) =  $\min_{m \in M(v)} (\text{COST}(m) + \sum_{x \in L(m)} \text{COST}(x))$ 
    L (m) = vertices of Subject Graph
              corresponding to leaves of m
  }
}
```

NPTEL (C) P. R. Panda, IIT Delhi, 2017 21

And the tree covering algorithm could then look like something that you can easily fill up. Now we can leave this open it helps I am not sure, but here is a template.

Student: (Refer Time: 63:40) in exam.

For this a variation of it we could fill in the exam. So, let me have some costs initially perhaps I can give them a negative cost or something just to indicate that the costs have not yet been computed ok. And at the leaves I indicate a cost of 0 leaves of the subject draft it is right because you did not need nothing to match. While some nodes are still there with a negative cost let me select some node whose children all have cost greater than or equal to 0. What is the meaning of this.

Student: The first level just about the leaf we are selecting that vertex.

Just that why, but this is a loop that is going over all the nodes cost greater than or equal to 0 is checking. What from the initialization.

Student: Either the cost.

Has it been computed or it is (Refer Time: 64:47). Yeah that is alright. So, in both the cases you can proceed if it is not done then you is more complex, but if it is done among other things if all the children you have computed the cost then there is enough

information for us to go ahead right. So, if that is the case then $m \vee$ could be the set of all the matching pattern graphs at v right that is what we had enumerated.

My cost computation here is just one line that is what we did in 10 minutes that we looked at that example. We are saying find the minimum among all the matches right selecting 1 match at a time m of the following cost of m itself cost of that cell corresponding to m itself, plus the sum of all the sub trees that are rooted at the children l of m is just the vertices of the sub graph corresponding to leaves of m . When you select m there will be some bunch of sub trees that you now have to account for right.

So, those sub trees costs you would have to add that is exactly what we were doing when we said the you have c_1 plus x_1 plus x_2 plus x_3 and so on. Here this is essentially the cost of m right for that m and these are the other cost right. So, the leaf matching costs you see why there is the requirement that at each node we are annotating the cost of the entire sub tree rooted at that node right it is because I may come back to it later. I may choose some match here which matches all of this, if I do that then I need the that cost, that cost, and that cost essentially I need the optimal cost for every descendant in order for this algorithm to pick up all the right notes.

I do not know actually when I start out how many matches will be there and what is the complexity of the cells maybe it actually covers 10 of these nodes then there will be a large number of leaves I need to know the optimal cost at all of those leaves. So, that is the reason for maintaining the total cost at each of the nodes it corresponds to the sub tree that is rooted at that. The point is once you do that computation any node under it is also something for which you know the optimal costs.