Synthesis of Digital Systems

COL 719

Part 6: Technology Mapping

Preeti Ranjan Panda

Department of Computer Science and Engineering

Indian Institute of Technology Delhi

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

Cell Library

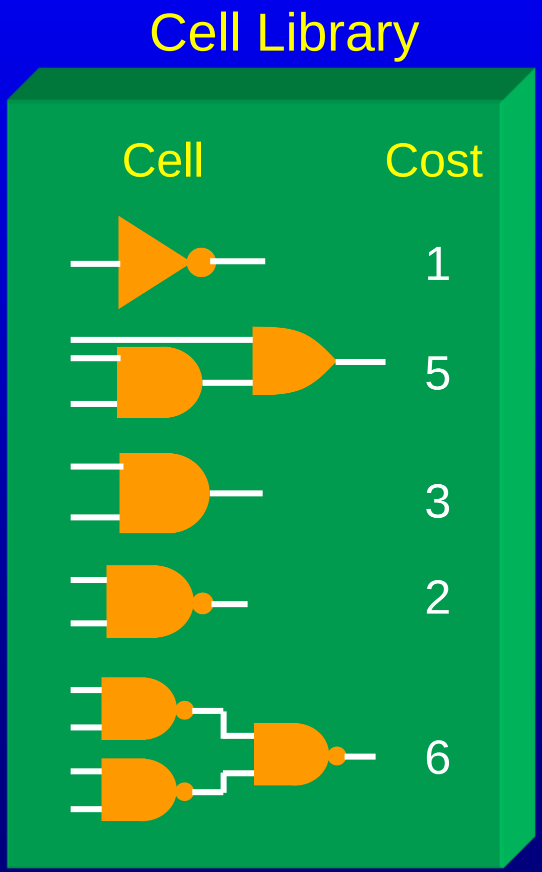
- Each cell consists of
  - Cell function
    - multiple i/p, single o/p
  - Cost
    - area
    - propagation delay
      - (minimum, typical, maximum)
      - worst case
      - function of fanout/load
    - power dissipation

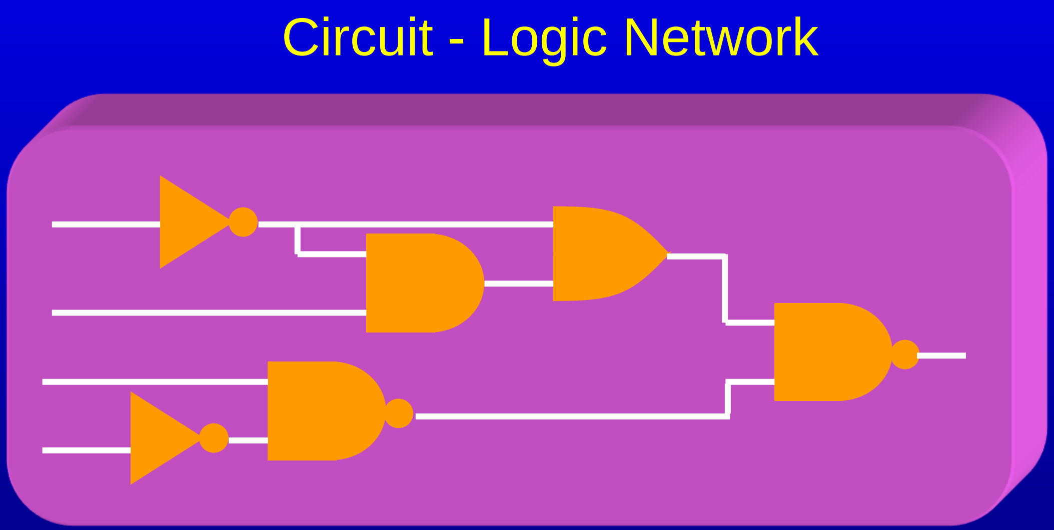
Mapping Problem

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

Technology Mapping

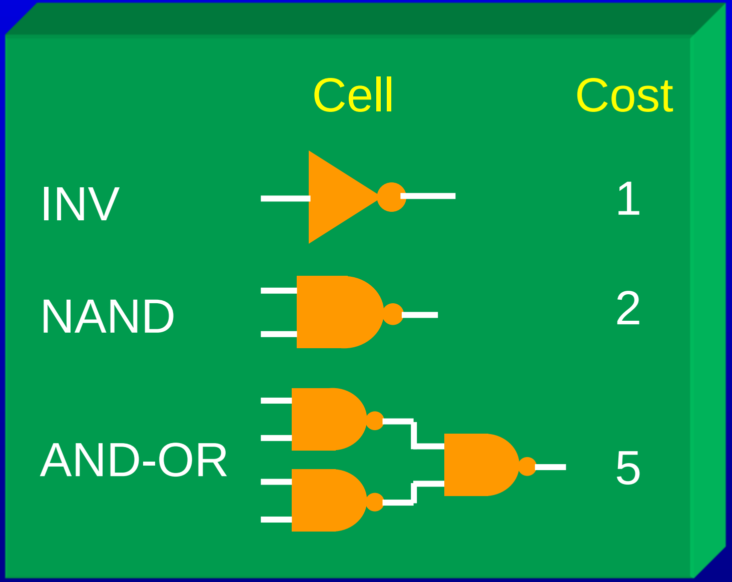
Implement given circuit with library cells minimising cost





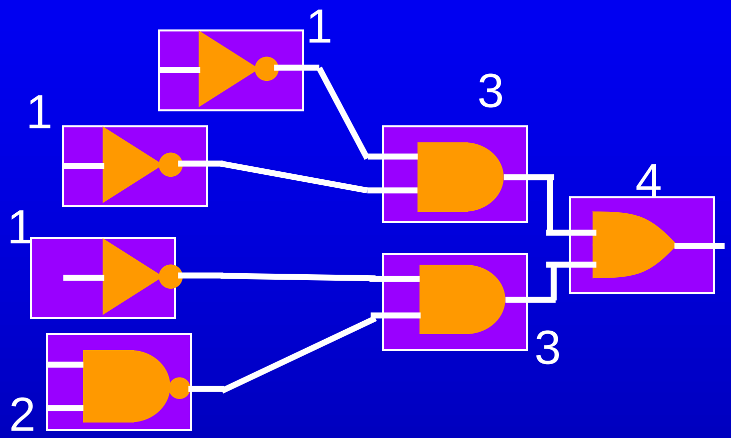
Technology Mapping Alternatives

Cell Library

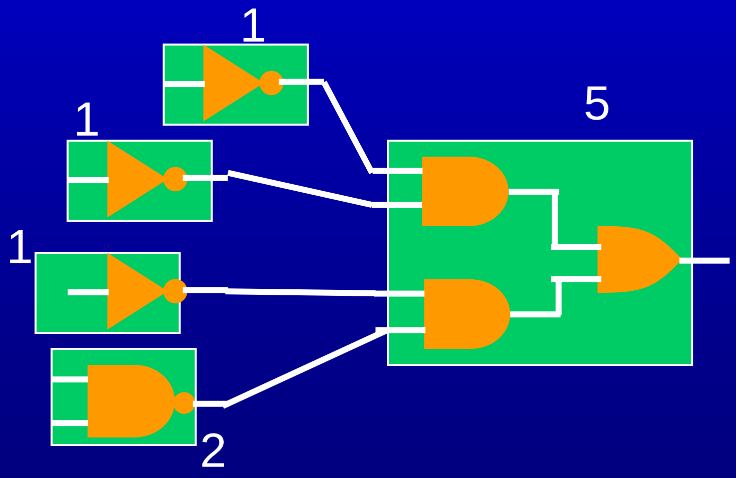


Trivial Mapping

Mapping Cost = 15

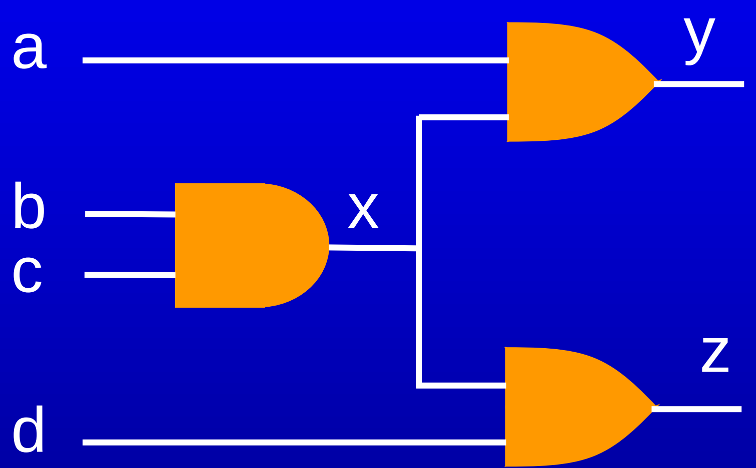


Mapping Cost = 10



Mapping Example

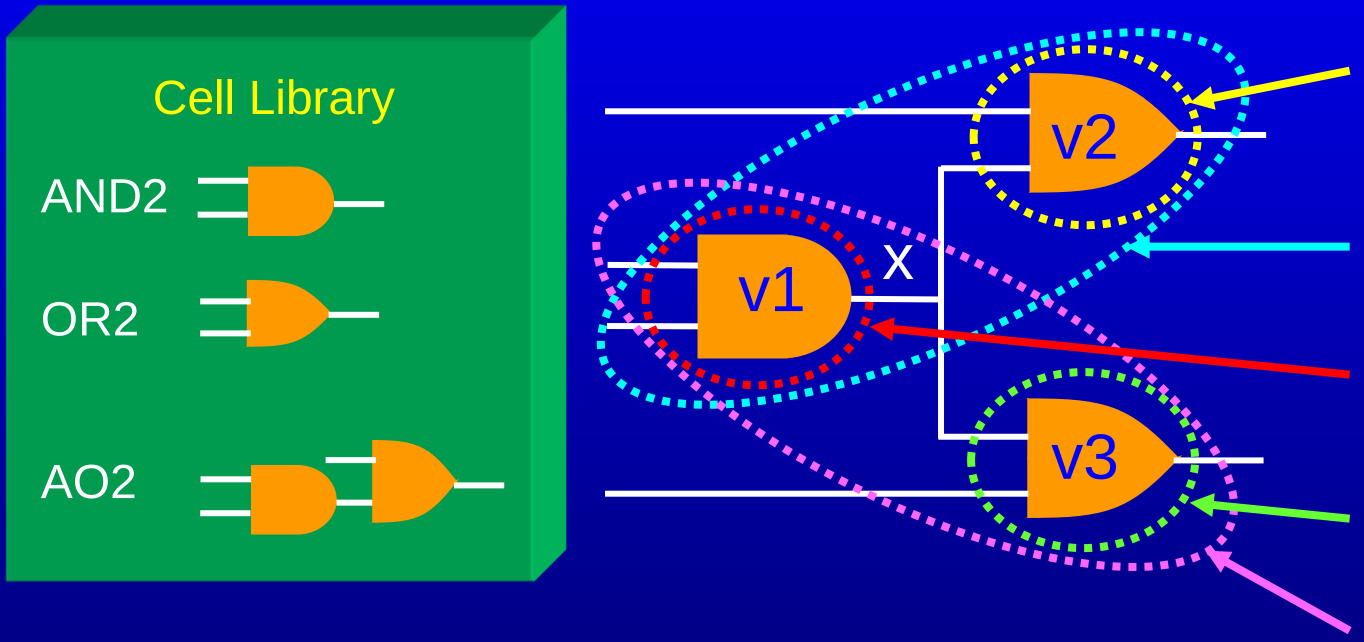
	Cell	Cost
AND2		4
OR2		4
AO2		5



Discussion from: G. de Micheli, Synthesis and Optimization of Digital Circuits

Matching

All possible matches



m1: {OR2, v2}

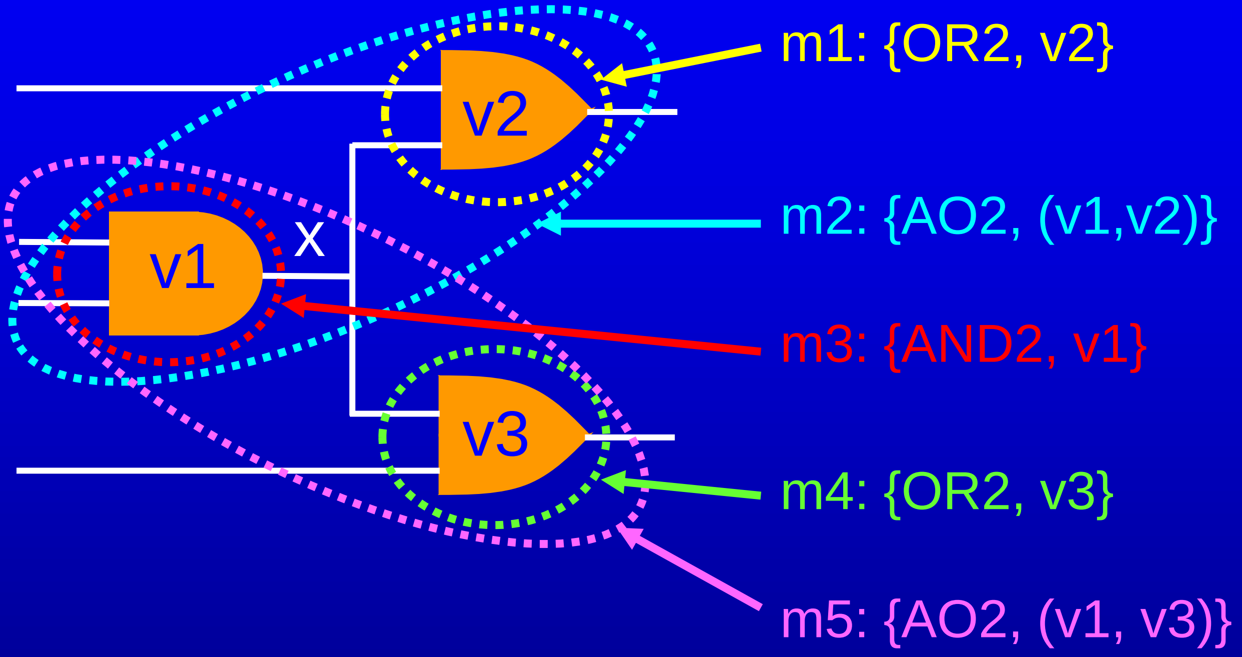
m2: {AO2, (v1,v2)}

m3: {AND2, v1}

m4: {OR2, v3}

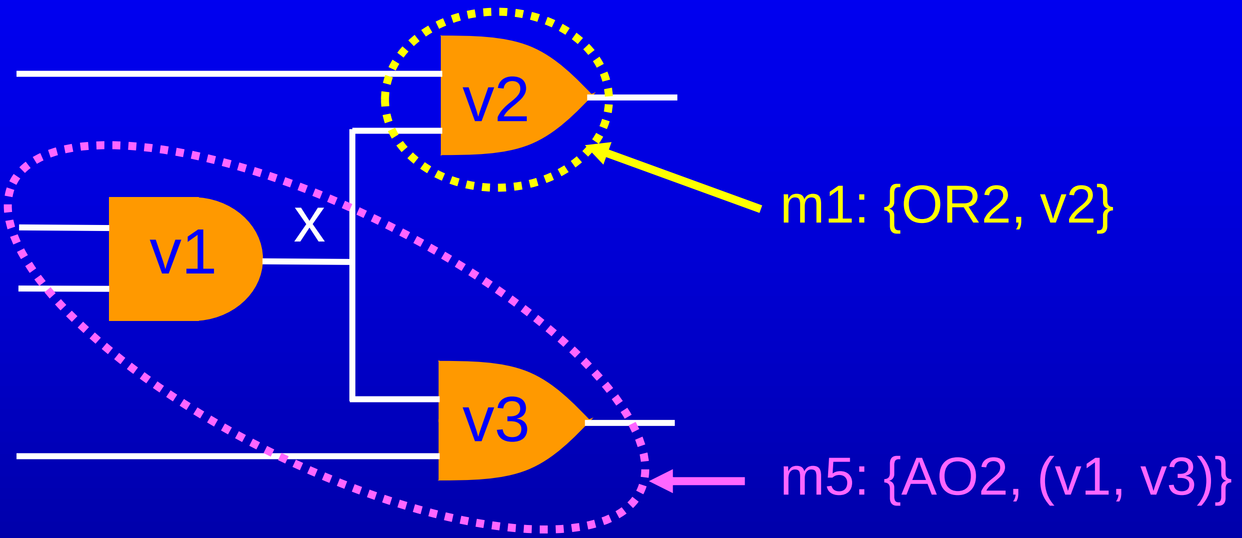
m5: {AO2, (v1, v3)}

Covering All Vertices



Covering v1: (m2 + m3 + m5)
Covering v2: (m1 + m2)
Covering v3: (m4 + m5)

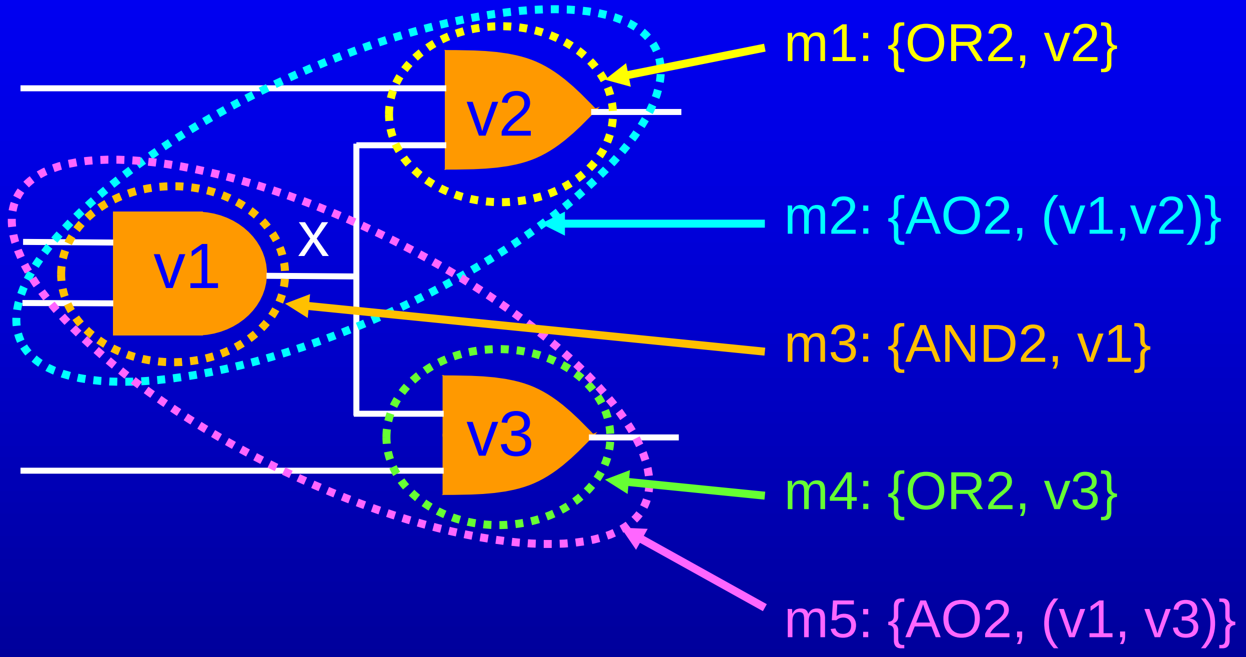
Some Covers are Not Legal



Choosing m1 and m5 covers all vertices but not legal

Input x to OR2 gate not available from AO2 gate

Connectivity Requirement



If m1 is chosen, choice of m3 (and any other match that leaves x as an available input) is implied

m1 => m3
m4 => m3

More Clauses: (m1' + m3)
(m4' + m3)

Overall Requirements

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

$$\boxed{(m2 + m3 + m5)\,(m1 + m2)\,(m4 + m5)\,(m1' + m3)\,(m4' + m3) = 1}$$

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

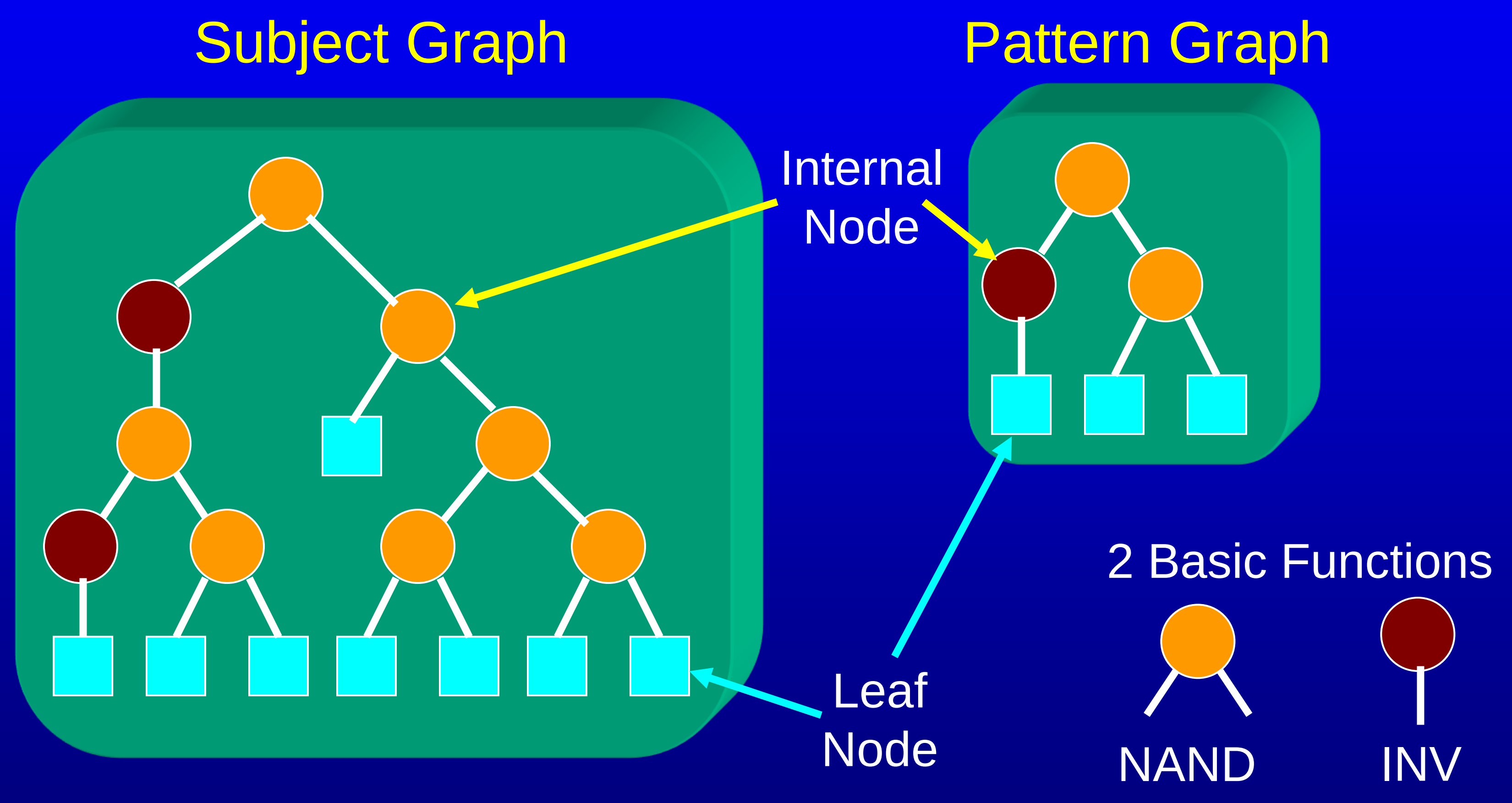
Matching Approaches

- 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

Pre-processing for Covering

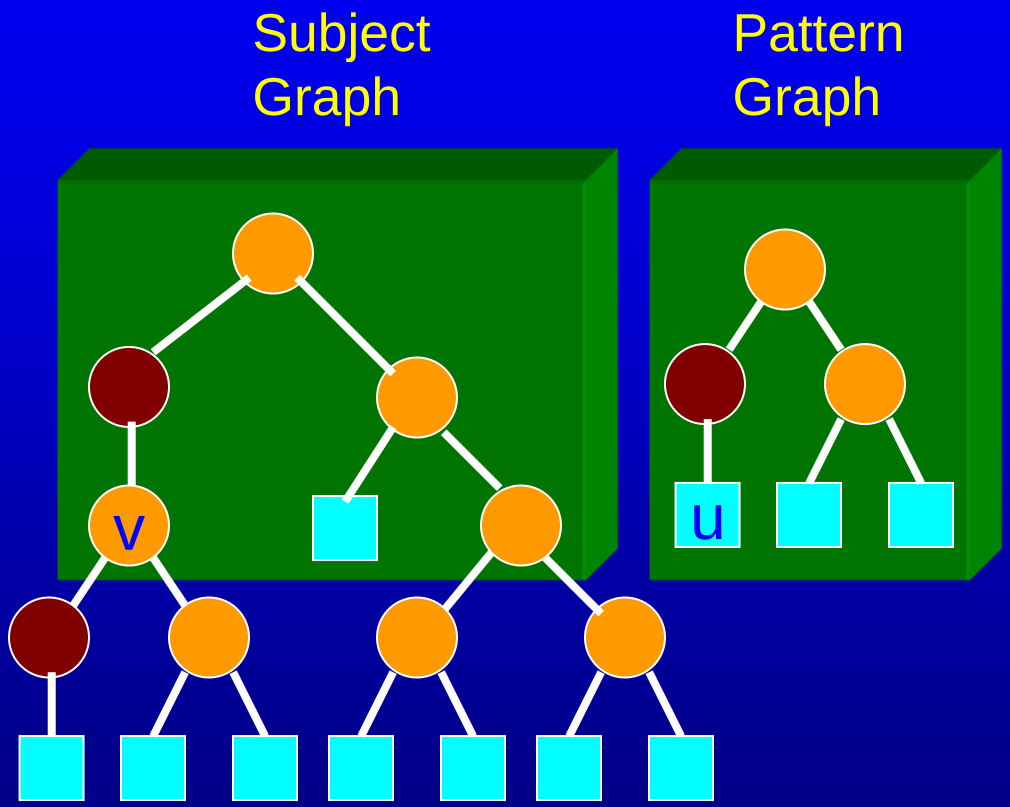
- Decomposition
  - Decompose logic network into NANDs (and INVERTERS)
  - Structural matching becomes easier
- Partitioning
  - Convert multi-o/p network into multiple single-o/p networks
  - Cover resulting Subject Graphs in sequence
  - Results in smaller graphs
  - Tree covering easier

Tree Representation



Tree Matching -1

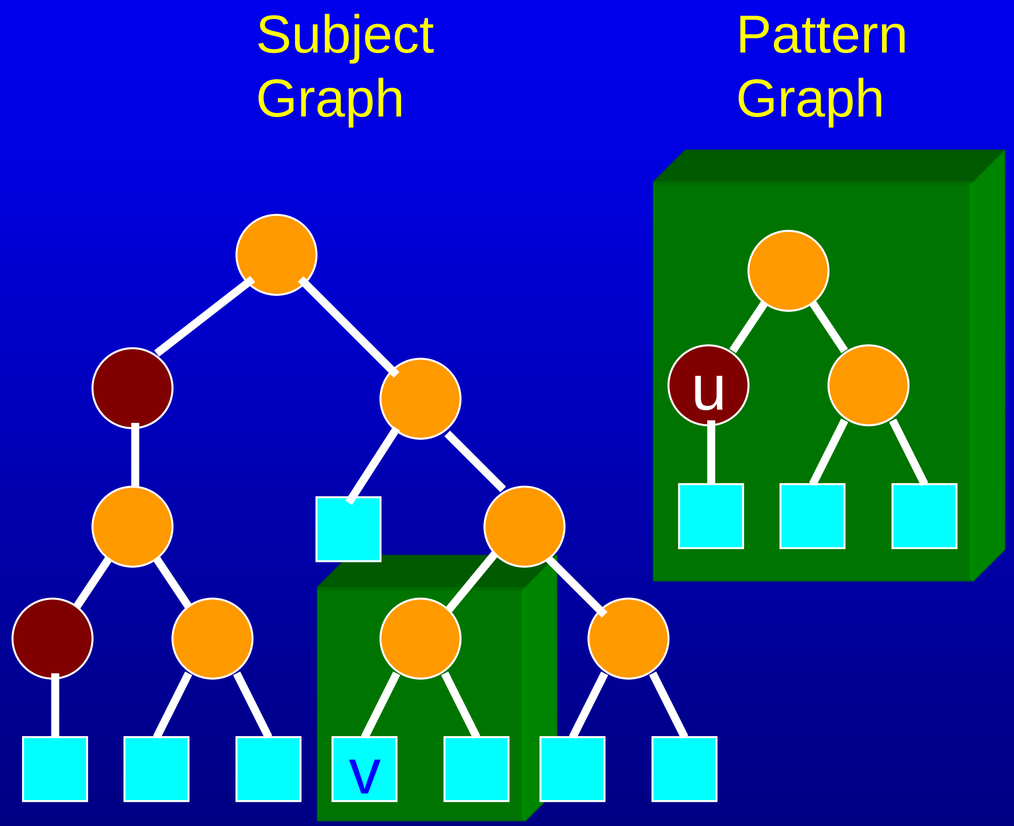
```
MATCH (u, v) {
  u - Pattern Graph node
  v - Subject Graph node
  if (u is leaf) return TRUE
  ...
}
```



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

Tree Matching - 2

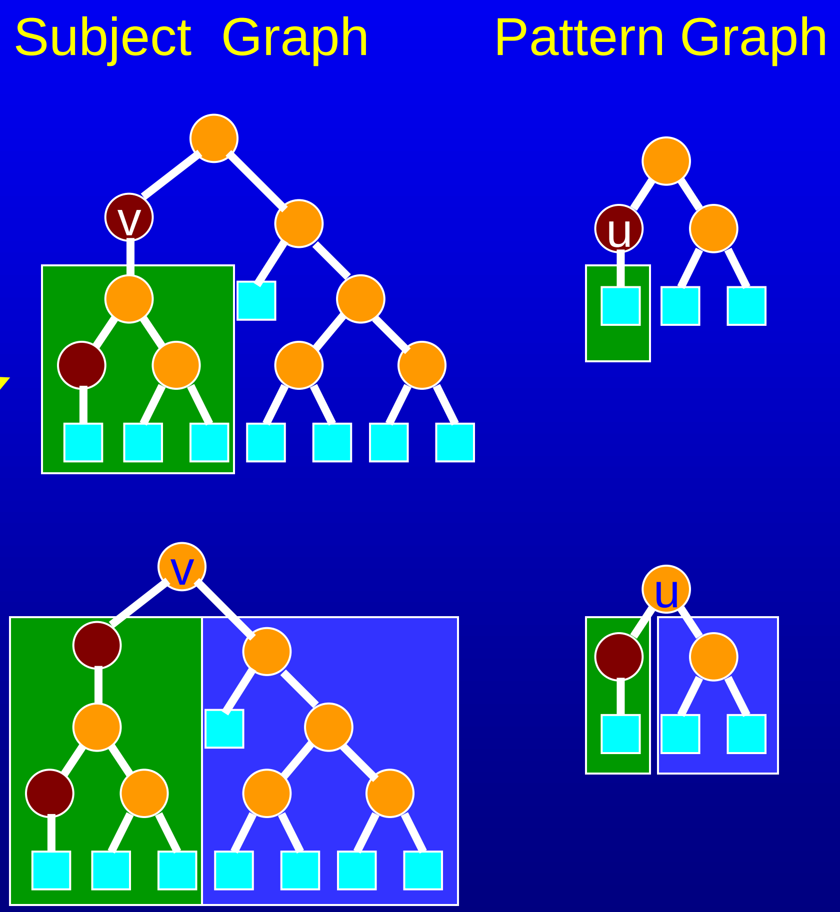
```
MATCH (u, v) {
  if (u is leaf) return TRUE
  else {
    if (v is leaf) return FALSE
      ...
}
```



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

Tree Matching - 3

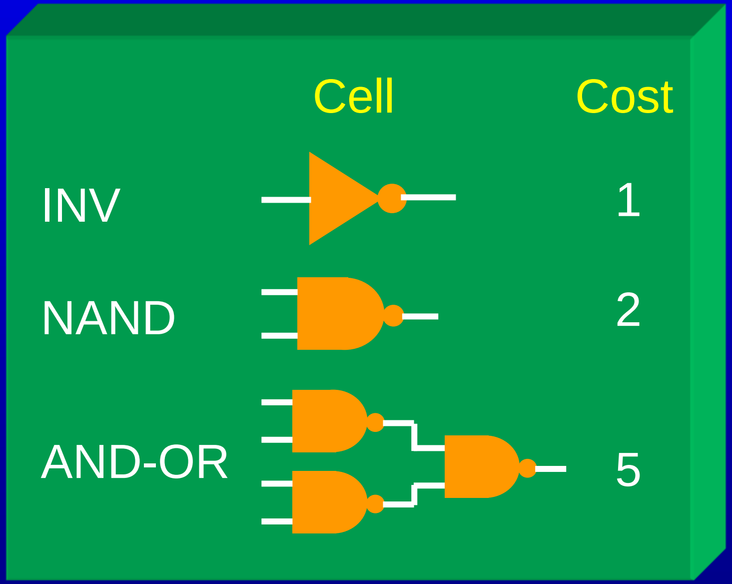
```
MATCH (u, v) {
  if (u is leaf) return TRUE
  else {
    if (v is leaf) return FALSE
    if (degree (v) ≠ degree (u))
      return FALSE;
    if (degree (v) = 1) {
        return MATCH (child (u), child (v))
    } else {
        return (MATCH (left (u), left (v))
                  & MATCH (right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

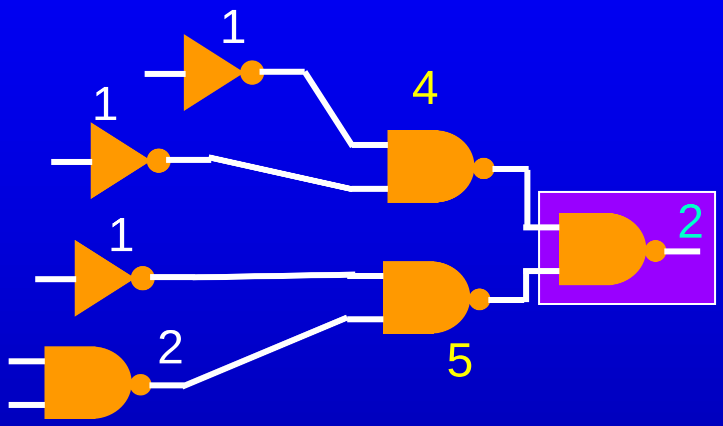


Tree Covering

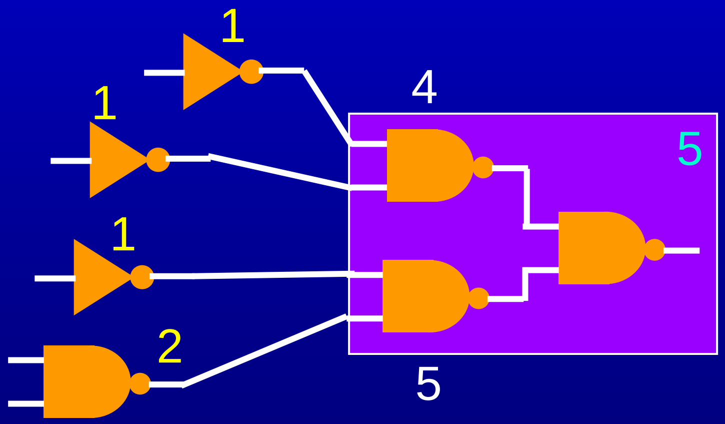
- Can be solved efficiently by Dynamic Programming
- Exhibits Optimal Sub-structure
  - if optimal solution to all descendants of node n is known
  - optimal solution to node n can be efficiently computed
- Bottom-up traversal of Subject Graph
  - list all matches at current node
  - cost = cost of matching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

Tree Covering Example





Cost
= 4+5+2
= 11



Cost
= 1+1+1+2+5
= 10

Tree Covering Algorithm

```
TREE_COVER (V, E) {
  COST (v) = -1 ∀ internal vertices v
  COST (u) = 0 ∀ leaves u
  while (∃ node with -ve cost) {
    select any v ∈ V whose children have COST >= 0
    M (v) = set of all matching Pattern Graphs at v
    COST (v) = min_{m ∈ M(v)} (COST (m) + Σ_{x ∈ L(m)} COST (x))
      L (m) = vertices of Subject Graph
                  corresponding to leaves of m
  }
}
```