VLSI System Design
Part III : Technology Mapping (1)

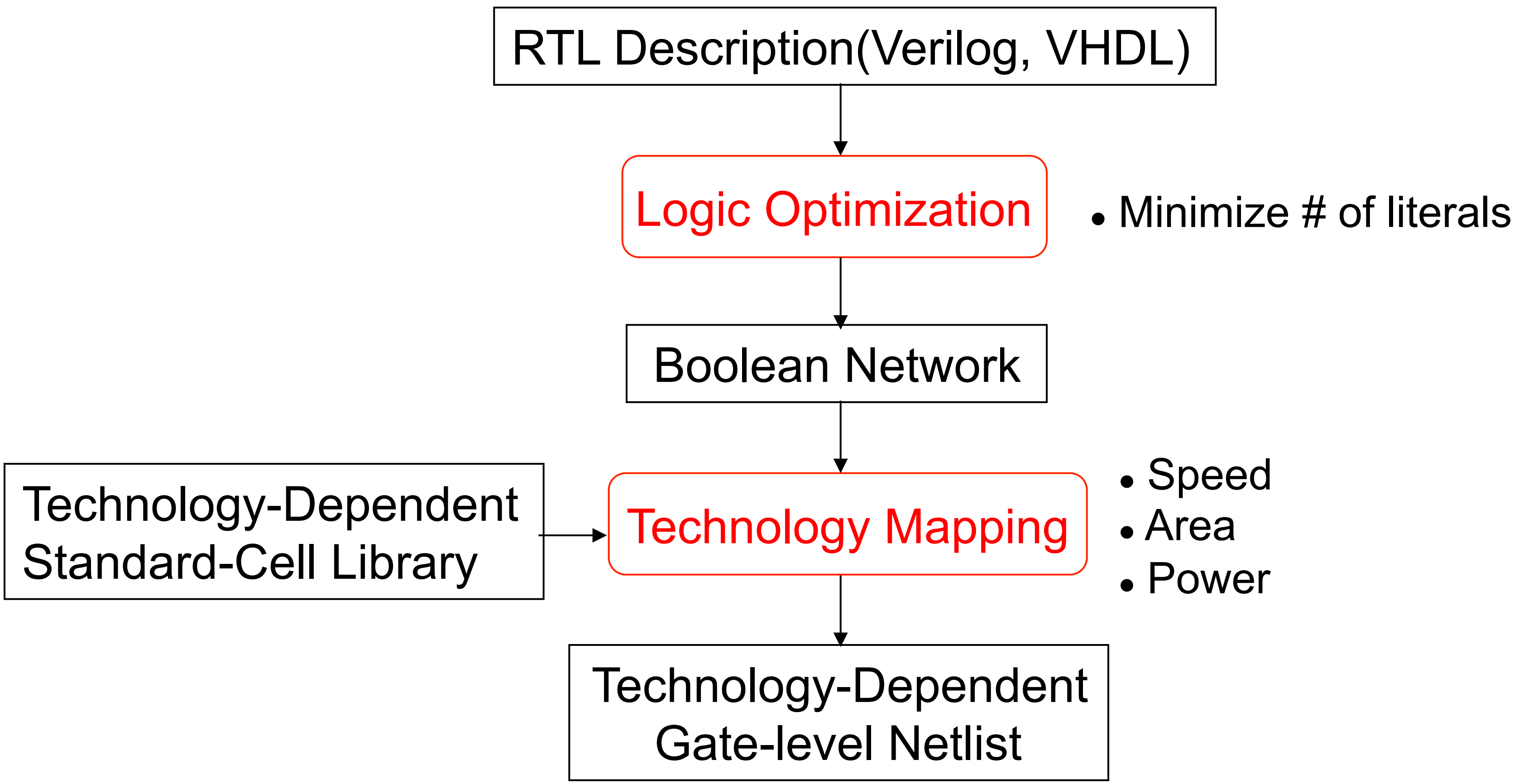
Lecturer : Tsuyoshi Isshiki

Dept. Communications and Computer Engineering,

Tokyo Institute of Technology

isshiki@vlsi.ce.titech.ac.jp

Logic Synthesis Flow



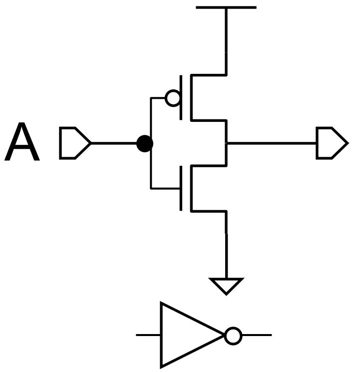
Technology Mapping and Circuit Cell Library

1. *Technology Mapping* transforms the Boolean Network into netlist composed of predefined circuit cells (mask layout for the cells provided).
2. Circuit cell types
   a. Functionality
      - Primitive cells：INV, AND, OR, NAND, NOR
      - Compound cells：XOR, AND-OR, MUX, TBUF
      - Storage cells：LATCH, FF

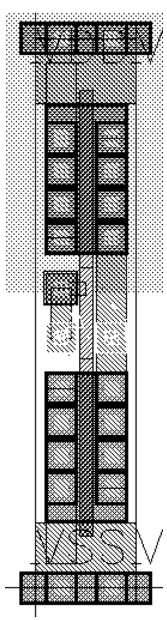
        \- Options : Positive/negative clock, asynchronous/ synchronous set & reset, clock enable

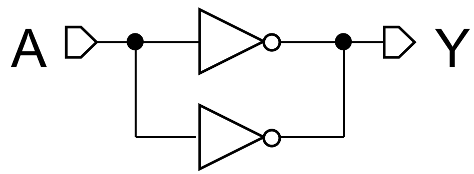
   b. Drive Power

INV Cell (Schematic / Layout)

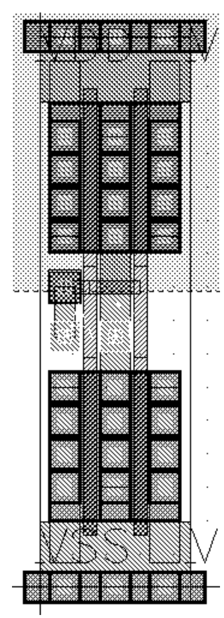


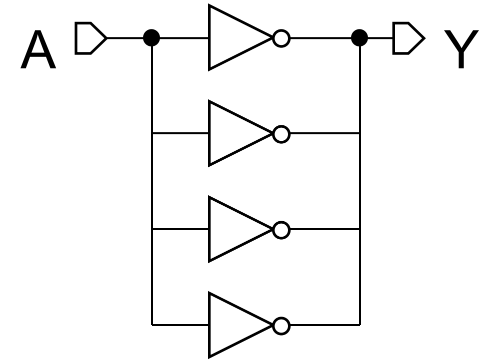
Power :×1



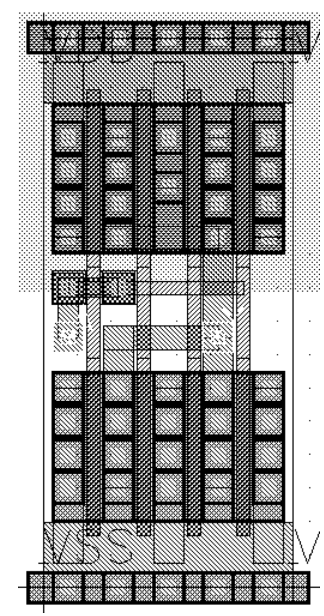


Power :×2

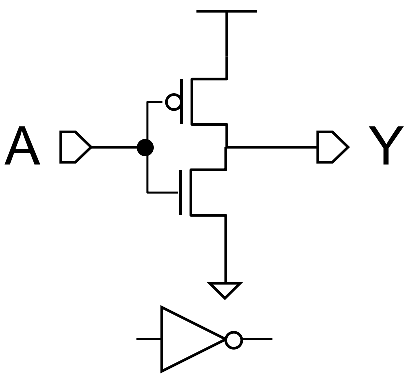




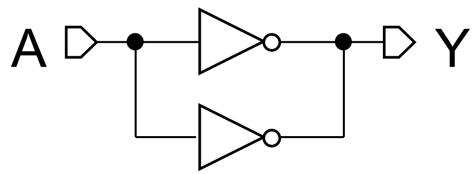
Power :×4



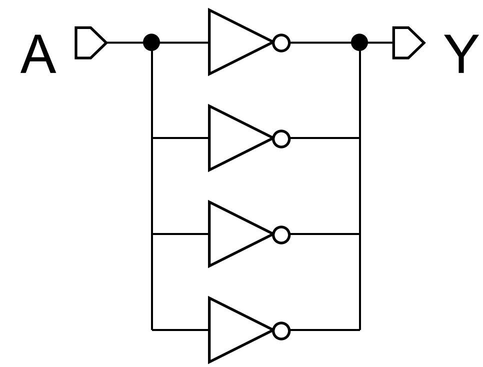
INV Cell (Schematic / Layout)



Power :×1



Power :×2



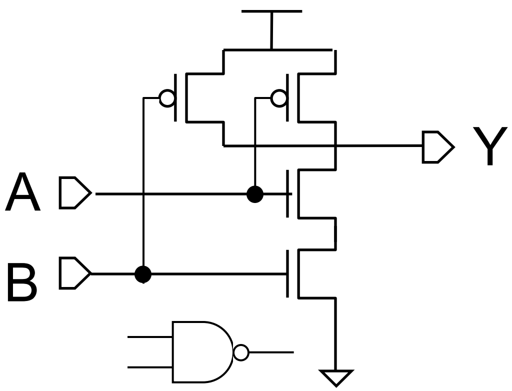
Power :×4

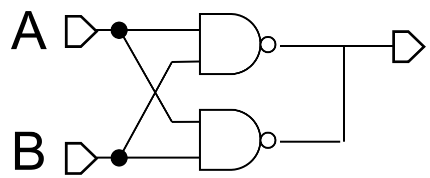
Size : 3.0 x 16.5 um
Load :
  A : 0.025pF
Internal delay :
  A=>Y(rise) : 0.042 ns
  A=>Y(fall) : 0.039 ns
Output transition delay :
  Y(rise) : 1.534 ns/pF
  Y(fall) : 0.715 ns/pF

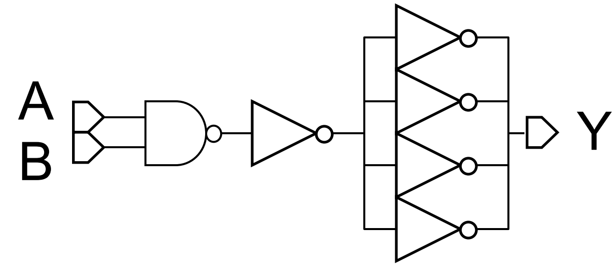
Size : 7.5 x 16.5 um
Load :
  A : 0.050pF
Internal delay :
  A=>Y(rise) : 0.035 ns
  A=>Y(fall) : 0.033 ns
Output transition delay :
  Y(rise) : 0.754 ns/pF
  Y(fall) : 0.355 ns/pF

Size : 13.5 x 16.5 um
Load :
  A : 0.100pF
Internal delay :
  A=>Y(rise) : 0.035 ns
  A=>Y(fall) : 0.034 ns
Output transition delay :
  Y(rise) : 0.374 ns/pF
  Y(fall) : 0.176 ns/pF

NAND2 Cell (Schematic / Layout)



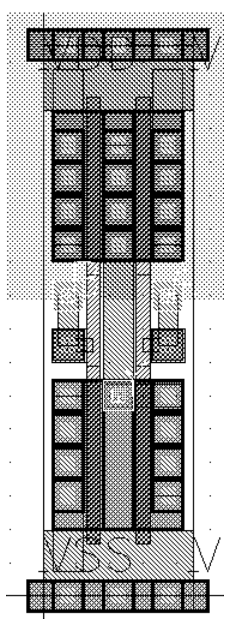


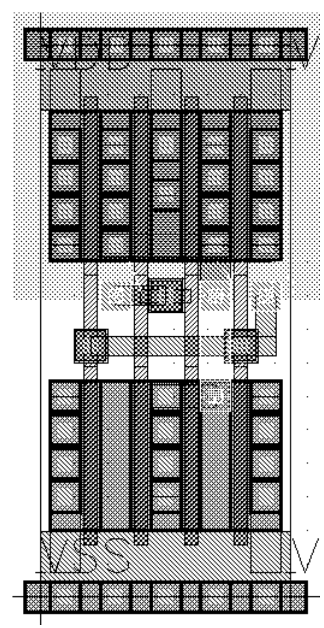


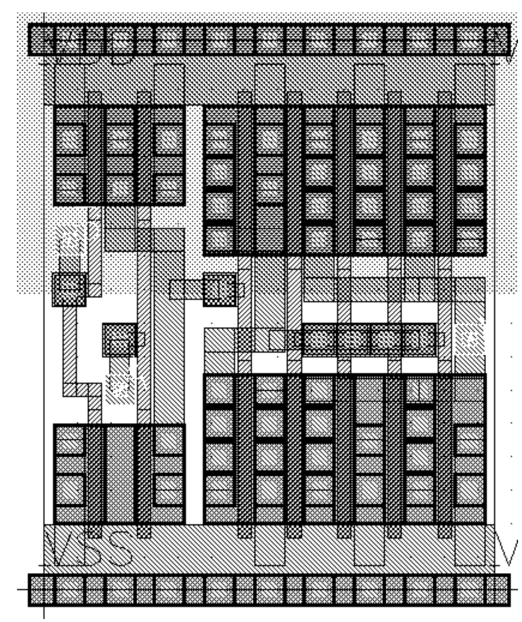
Power :×1

Power :×2

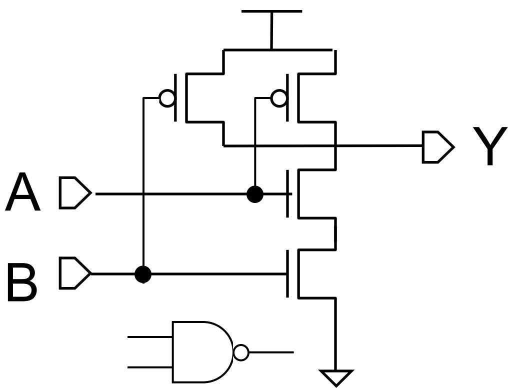
Power :×4



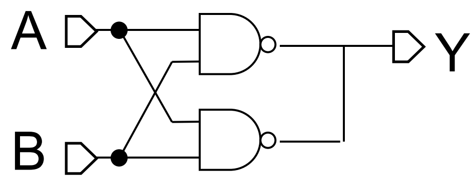




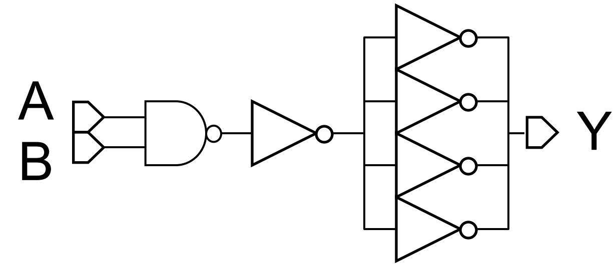
NAND2 Cell (Schematic / Layout)



Power :×1



Power :×2



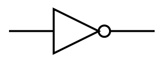
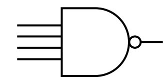
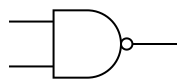
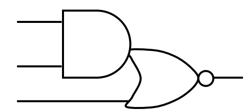
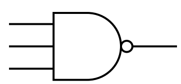
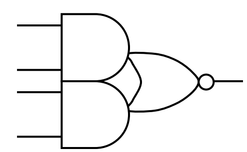
Power :×4

Size : 4.5 x 16.5 um
Load :
A : 0.025pF
B : 0.024pF
Internal delay :
A=>Y(rise) : 0.055 ns
A=>Y(fall) : 0.051 ns
B=>Y(rise) : 0.073 ns
B=>Y(fall) : 0.060 ns
Output transition delay :
Y(rise) : 1.532 ns/pF
Y(fall) : 1.153 ns/pF

Size : 7.5 x 16.5 um
Load :
A : 0.050pF
B : 0.050pF
Internal delay :
A=>Y(rise) : 0.048 ns
A=>Y(fall) : 0.050 ns
B=>Y(rise) : 0.068 ns
B=>Y(fall) : 0.058 ns
Output transition delay :
Y(rise) : 0.755 ns/pF
Y(fall) : 0.577 ns/pF

Size : 13.5 x 16.5 um
Load :
A : 0.017pF
B : 0.017pF
Internal delay :
A=>Y(rise) : 0.299 ns
A=>Y(fall) : 0.325 ns
B=>Y(rise) : 0.327 ns
B=>Y(fall) : 0.340 ns
Output transition delay :
Y(rise) : 0.373 ns/pF
Y(fall) : 0.205 ns/pF

Cell Library Example

Cell name	cost	symbol	Cell name	cost	symbol
INV	2		NAND4	5	
NAND2	3		AOI21	4	
NAND3	4		AOI22	5	

Area Optimal Technology Mapping Flow

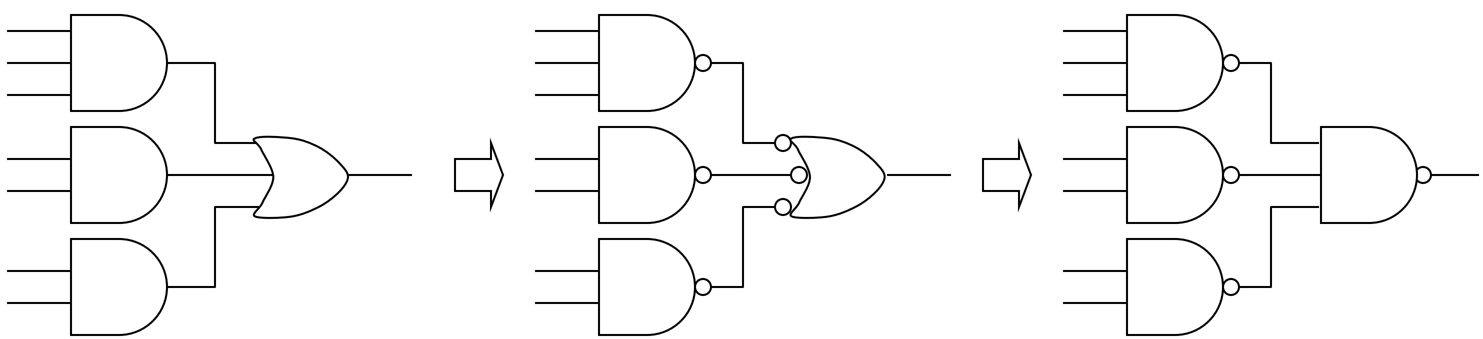
1. Transform the optimized Boolean Network into NAND network
2. Decompose NAND network into trees
   - Fan-out : # of destination pins for output pin of a node
   - Tree : DAG (directed acyclic graph) where all nodes have a fan-out of 1

     → Fast algorithms exist for solving the Optimal Tree Covering Problem
3. Transform each NAND-tree into NAND2-tree
   - Balanced NAND2 decomposition
4. For each NAND2-tree, obtain the optimal tree covering in terms of circuit area by *dynamic programming*

Transformation of Boolean Network to NAND Network

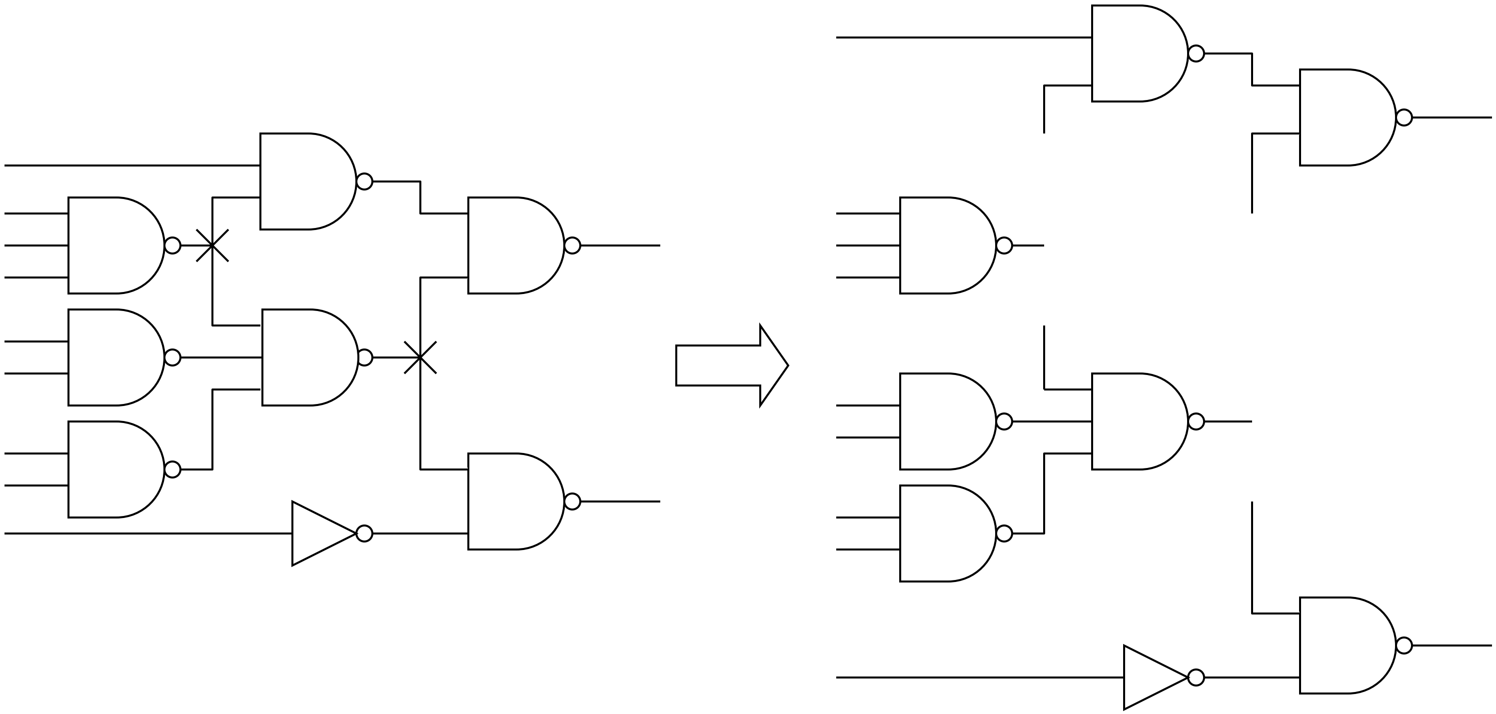
- At each node of Boolean Network, covert the sum-of-product form into NAND-NAND form

$$F = abc + de + fg$$
$$= (\overline{\overline{abc}}) + (\overline{\overline{de}}) + (\overline{\overline{fg}})$$
$$= \overline{(\overline{abc})(\overline{de})(\overline{fg})}$$



DAG-to-Tree Decomposition

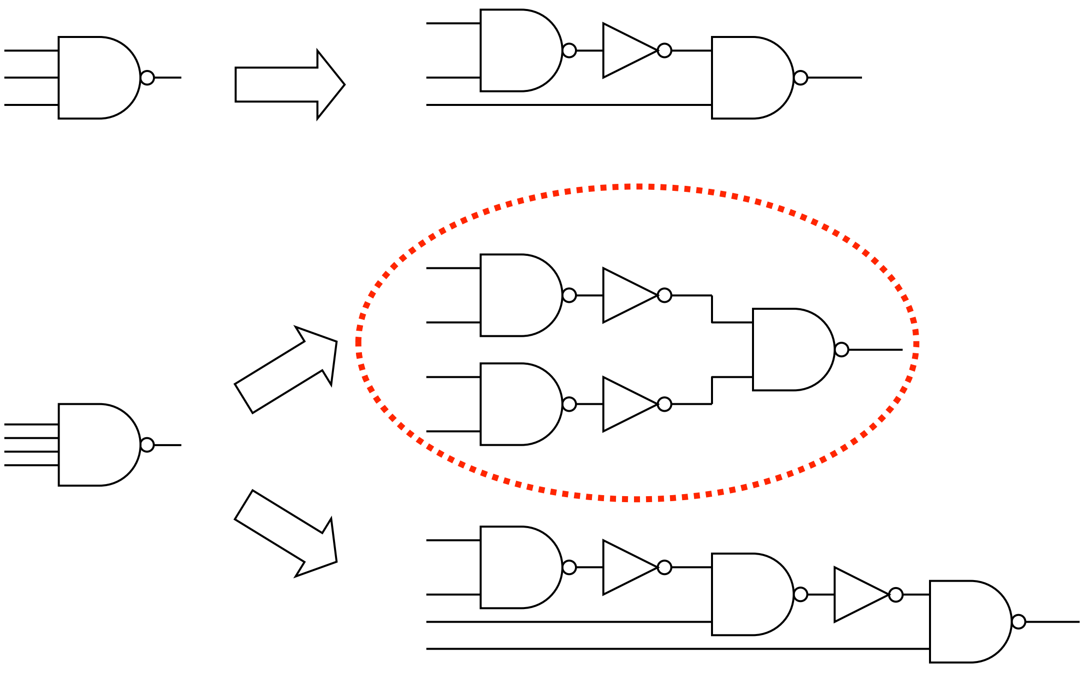
- If the gate output has a fan-out of more than 1, disconnect all pins from the arc (net).



"Forest of Trees"

NAND2 Decomposition (NAND2-Tree)

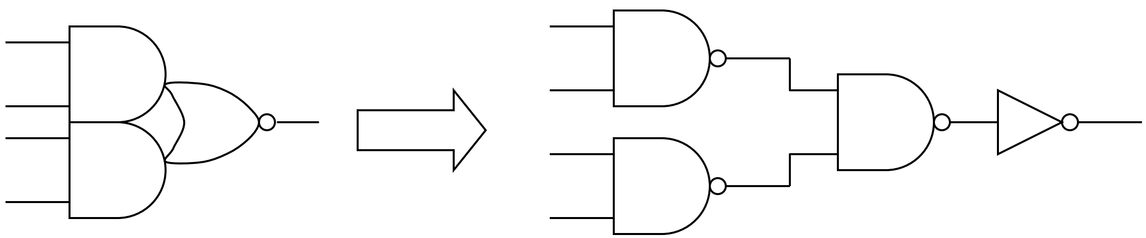
- For each NAND gate on the NAND-tree, decompose into NAND2 gates
- If there are multiple decomposition solutions, choose the tree with the smallest height（balanced tree decomposition）



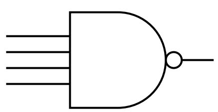
Cell Patterns

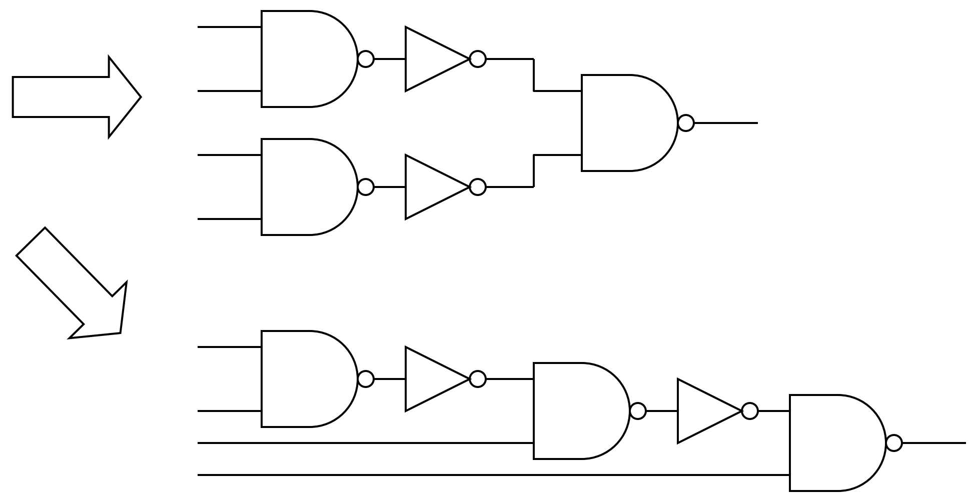
- For each cell in the library, enumerate all functionally equivalent NAND2-trees and register them as *cell patterns*.

AOI22



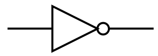
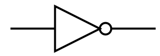
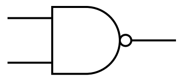
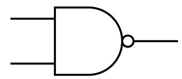
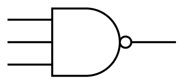
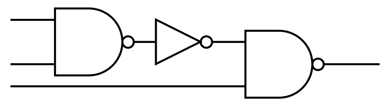
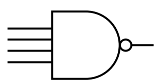
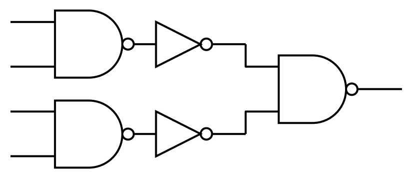
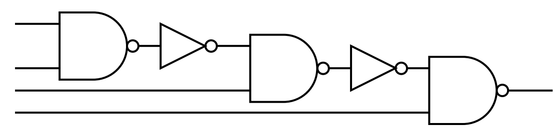
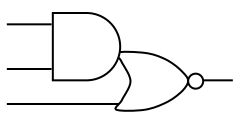
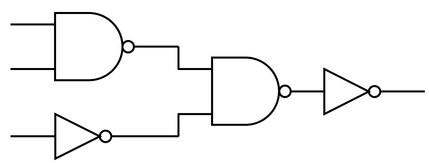
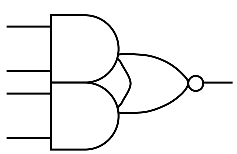
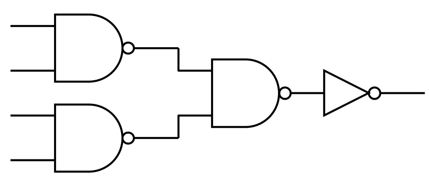
NAND4





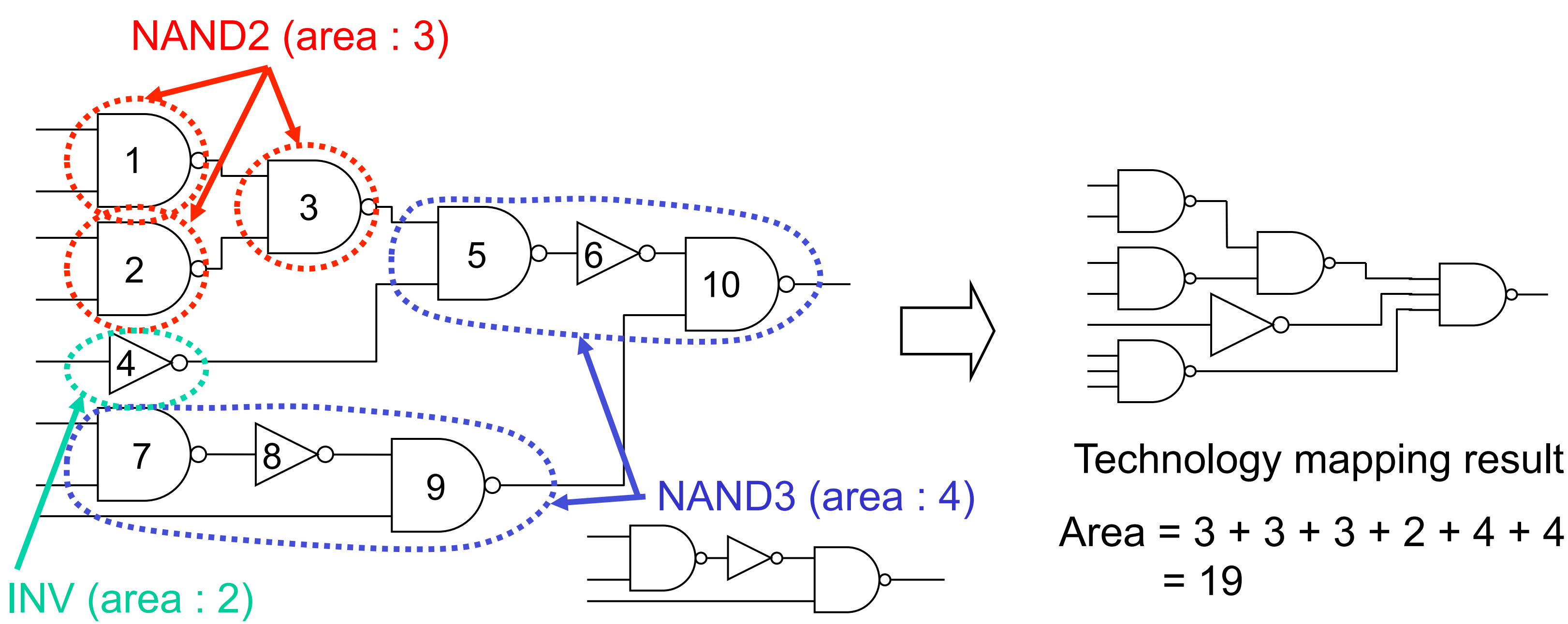
If there are multiple NAND2-tree patterns, enumerate all patterns

Cell Library Example

Cell name	cost	symbol	Primitive DAG（NAND2+INV representation）
INV	2		
NAND2	3		
NAND3	4		
NAND4	5		 
AOI21	4		
AOI22	5		

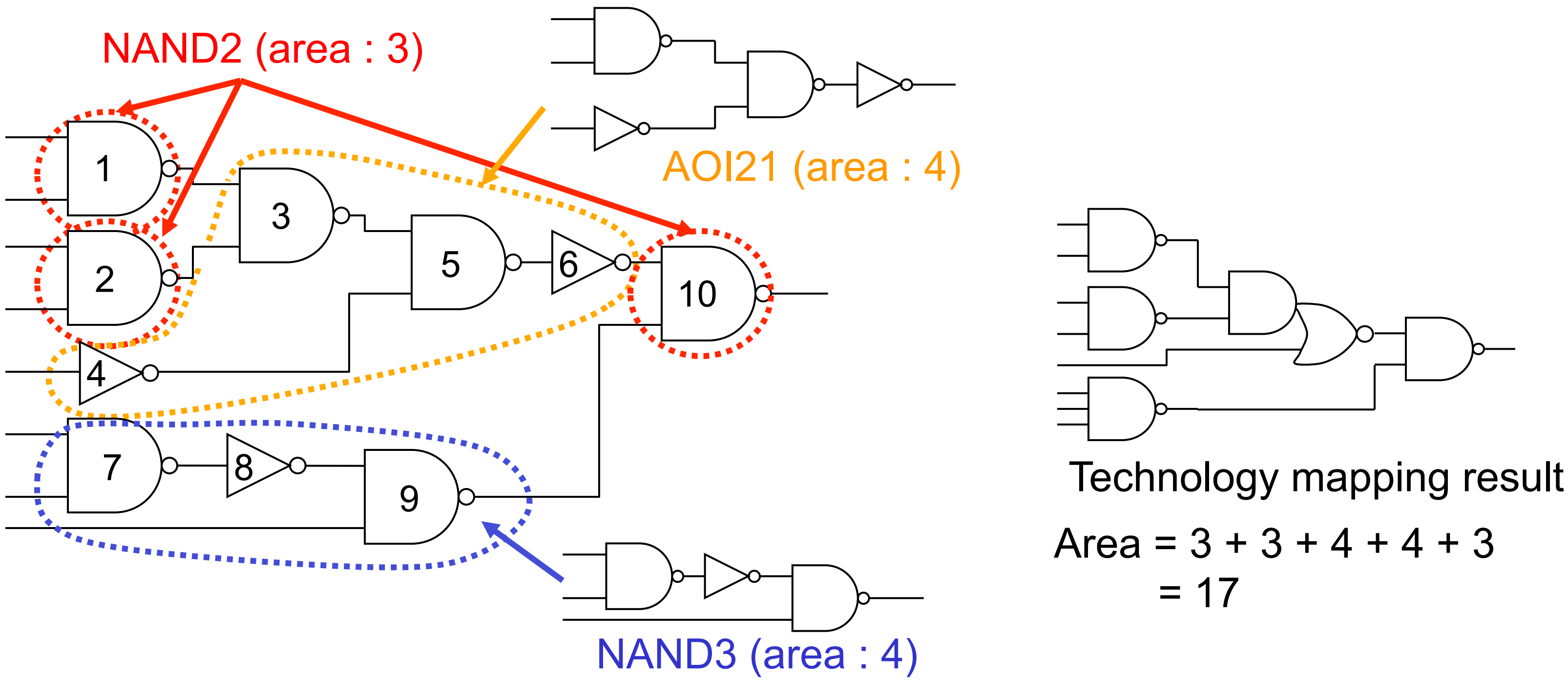
Technology Mapping as Tree Covering Problem (1)

- Cover the NAND2-tree with registered cell patterns with the minimum cost (circuit area, speed, etc.)
- Each node must be covered by exactly one pattern



Technology Mapping as Tree Covering Problem (2)

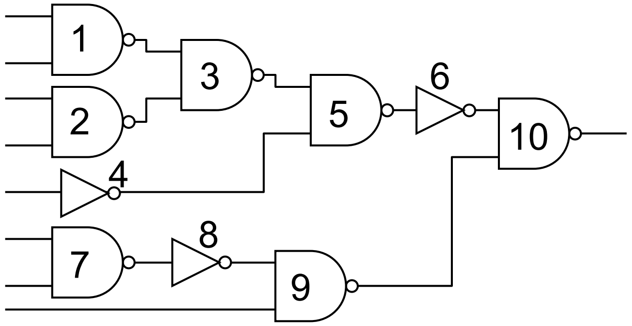
- Cover the NAND2-tree with registered cell patterns with the minimum cost (circuit area, speed, etc.)
- Each node must be covered by exactly one pattern



Area = 3 + 3 + 4 + 4 + 3
= 17

Tree Covering Approach (1)

- Definition of *tree* graph :
  - Each node consist of several *child* nodes and a *parent* node.
  - A *root* is a node with no parent node. (only one root in a tree)
  - A *leaf* is a node with no child nodes.
- Divide the covering problem on tree *T* into smaller covering problems on the *subtrees* of *T*.
  - Recursively solve the covering problem on the subtrees rooted at each node of *T* and store the optimal covering cost at each node.
  - Start from the leaf nodes and continue towards the root
  - *Here, assume that the covering cost is circuit area*

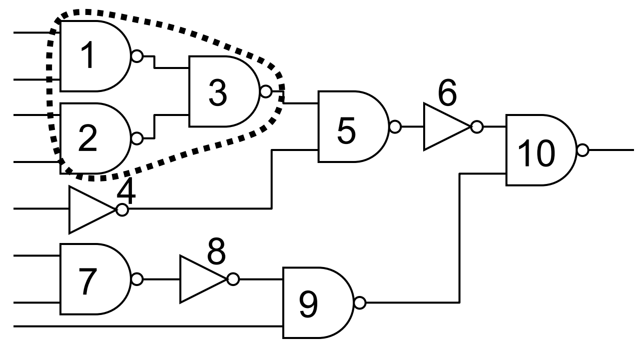


Target NAND2-tree

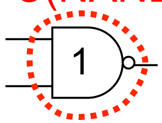
Tree Covering Example (1)

- $C\,(p_j)$: cost of cell pattern p_j
- $C_{opt}(i)$: optimal covering cost of the subtree rooted at node i
- $C_{map}(i\,,p_j)$: optimal covering cost of the subtree rooted at node i when p_j is used to cover i

  $\rightarrow$ $C_{opt}(i) = MIN\{C_{map}(i,\, p_j)\}$

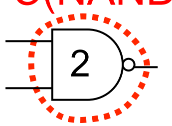


C(NAND2) = 3



$C_{map}(1, \text{NAND2}) = C(\text{NAND2}) = 3$
$C_{opt}(1) = C_{map}(1, \text{NAND2}) = 3$

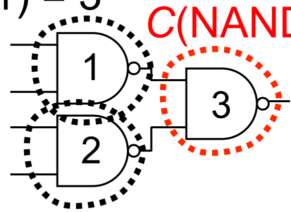
C(NAND2) = 3



$C_{map}(2, \text{NAND2}) = C(\text{NAND2}) = 3$
$C_{opt}(2) = C_{map}(2, \text{NAND2}) = 3$

$C_{opt}(1) = 3$

C(NAND2) = 3

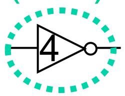


$C_{opt}(2) = 3$

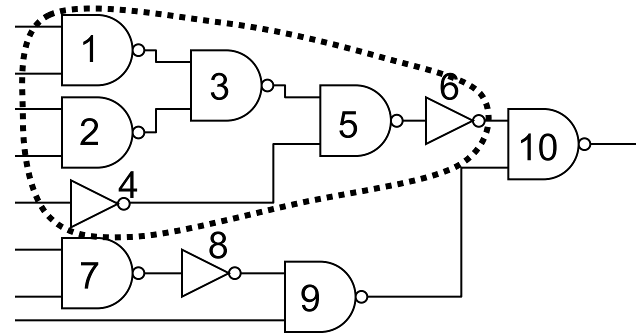
$C_{map}(3, \text{NAND2}) = C_{opt}(1) + C_{opt}(2) + C(\text{NAND2}) = 9$
$C_{opt}(2) = C_{map}(2, \text{NAND2}) = 3$

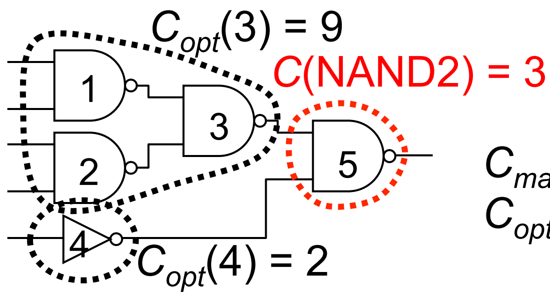
Tree Covering Example (2)

C(INV) = 2

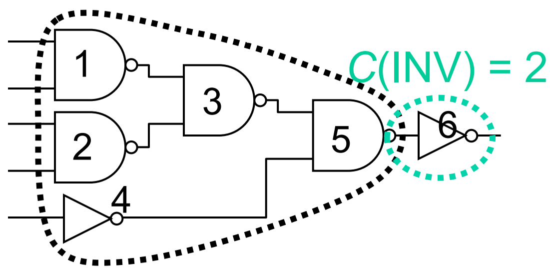


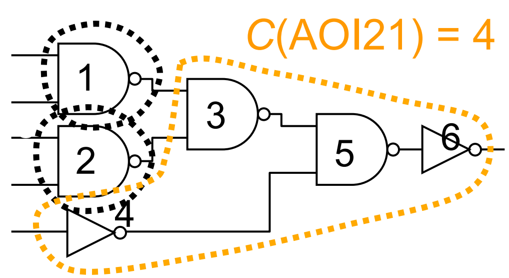
$C_{map}(4, \text{INV}) = C(\text{INV}) = 2$
$C_{opt}(4) = C_{map}(4, \text{INV}) = 2$





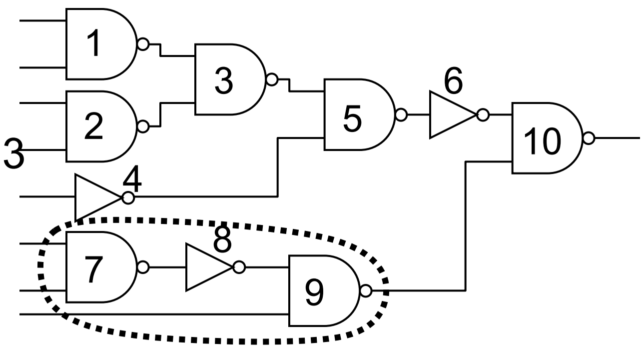
$C_{map}(5, \text{NAND2}) = C_{opt}(3) + C_{opt}(4) + C(\text{NAND2}) = 14$
$C_{opt}(5) = C_{map}(5, \text{NAND2}) = 14$



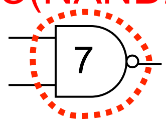


$C_{map}(6, \text{INV}) = C_{opt}(5) + C(\text{INV}) = 16$
$C_{map}(6, \text{AOI21}) = C_{opt}(1) + C_{opt}(2) + C(\text{AOI21}) = 10$
$C_{opt}(6) = C_{map}(6, \text{AOI21}) = 10$

Tree Covering Example (3)

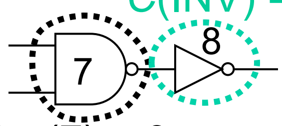


C(NAND2) = 3



C_{map}(7, NAND2) = C(NAND2) = 3
C_{opt}(7) = C_{map}(7, NAND2) = 3

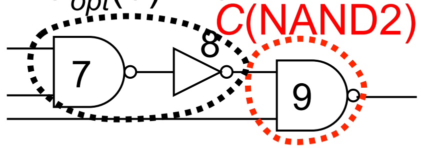
C(INV) = 2



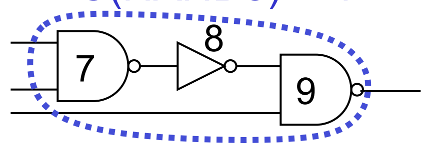
C_{opt}(7) = 3

C_{map}(7, INV) = C_{opt}(7) + C(INV) = 5
C_{opt}(8) = C_{map}(8, INV) = 5

C_{opt}(8) = 5
C(NAND2) = 3

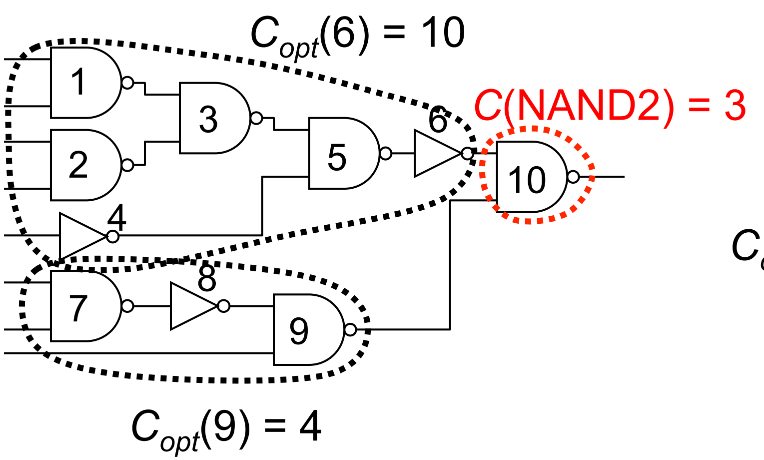


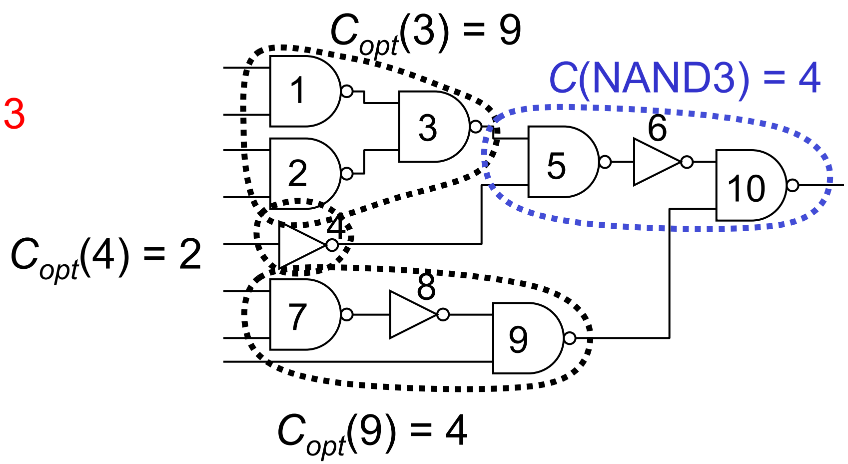
C(NAND3) = 4



C_{map}(9, NAND2) = C_{opt}(8) + C(NAND2) = 8
C_{map}(9, NAND3) = C(NAND3) = 4
C_{opt}(9) = C_{map}(9, NAND3) = 4

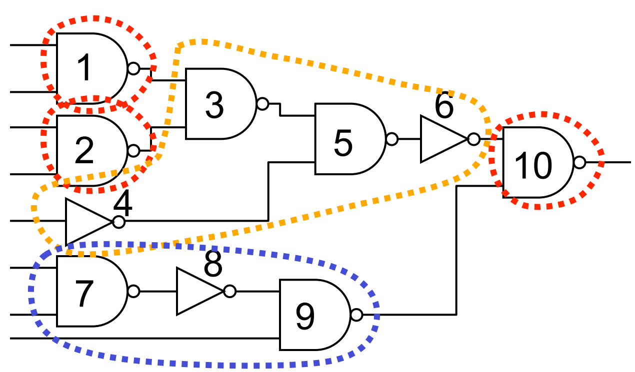
Tree Covering Example (4)





$C_{map}(10, \text{NAND2}) = C_{opt}(6) + C_{opt}(9) + C(\text{NAND2}) = 17$
$C_{map}(10, \text{NAND3}) = C_{opt}(3) + C_{opt}(4) + C_{opt}(9) + C(\text{NAND3}) = 19$
$C_{opt}(10) = C_{map}(10, \text{NAND2}) = 17$

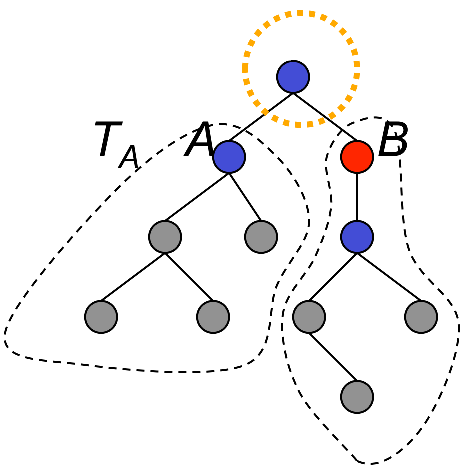
Optimal tree cover →

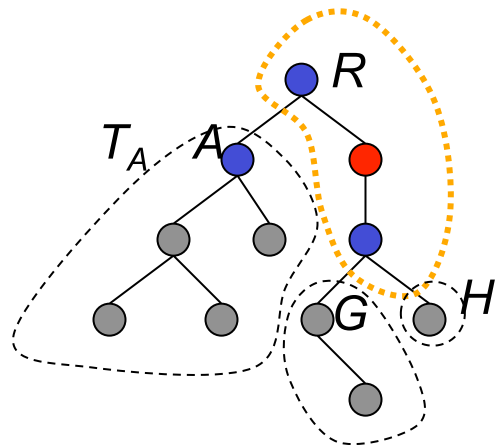


Principle of Optimality for Area-Optimal Tree Covering

- Area-optimal covering cost using pattern p_i can be derived from the optimal covering costs of the subtrees whose roots are connected to the leaves of p_i.
- Area-optimal covering on the subtrees rooted at each node needs to be calculated only once and stored.

  → *Dynamic Programming* : Deriving the solution to a problem from a set of solutions on its subproblems.



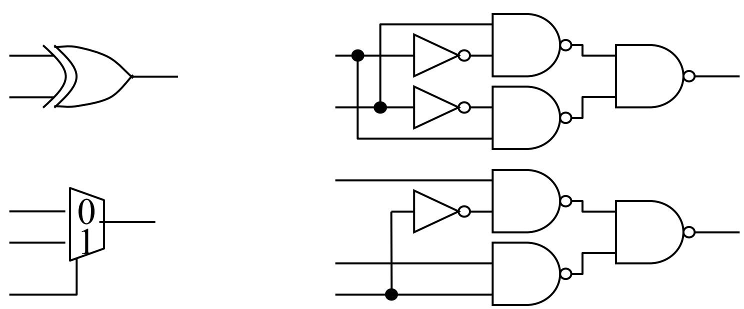


$C_{map}(R, p_1) = C_{opt}(A) + C_{opt}(B) + C(p_1)$ $C_{map}(R, p_2) = C_{opt}(A) + C_{opt}(G) + C_{opt}(H) + C(p_2)$

$$C_{opt}(R) = MIN\{C_{map}(R, p_1) + C_{map}(R, p_2)\}$$

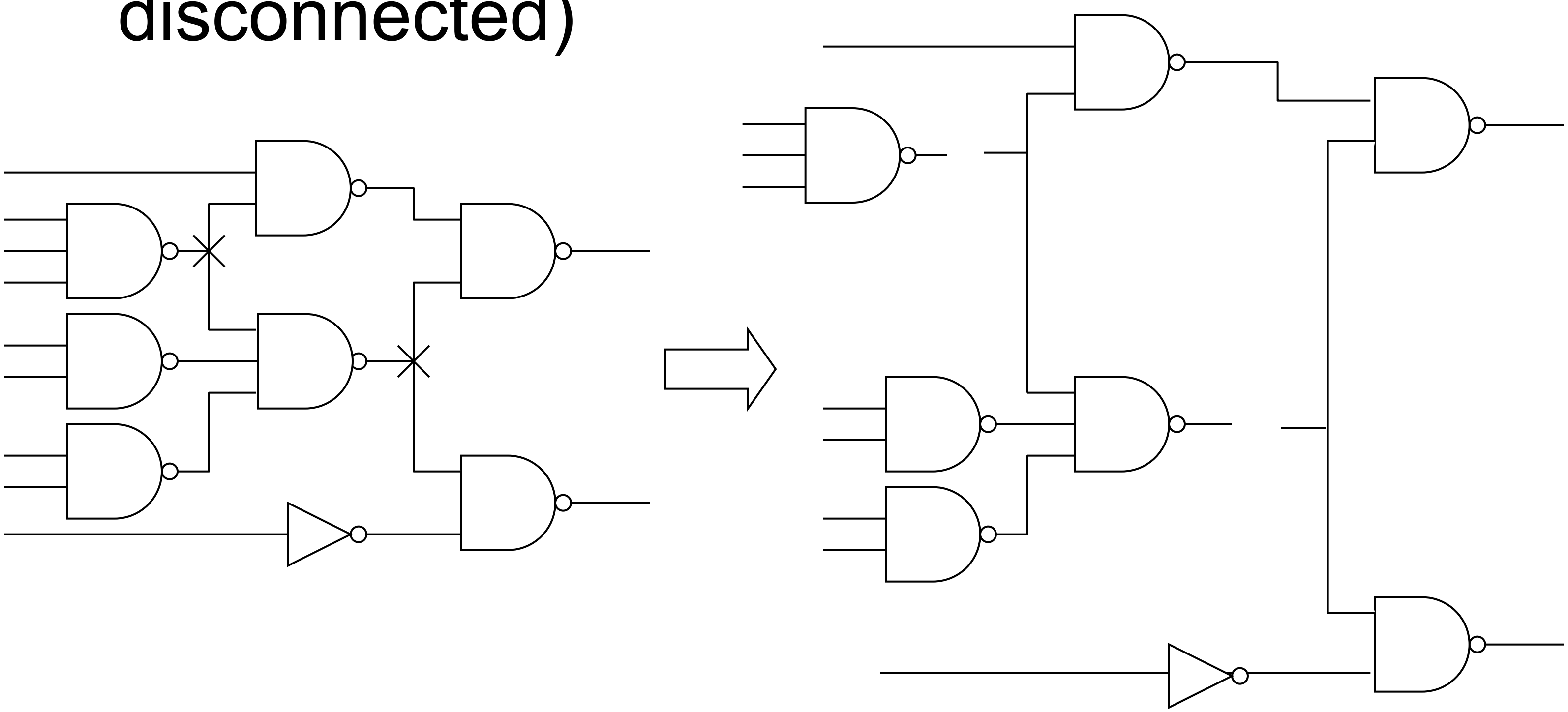
Problems Arising in DAG Decomposition (1)

1. Gates such as multiplexer and EXOR cannot be expressed in NAND2-trees
   - Instead of decomposing the network into trees, decompose into *Leaf-DAG*
     - Leaf-DAG：primary inputs are allowed to have multiple fan-outs.
     - Tree-covering algorithm can still be applied to Leaf-DAG



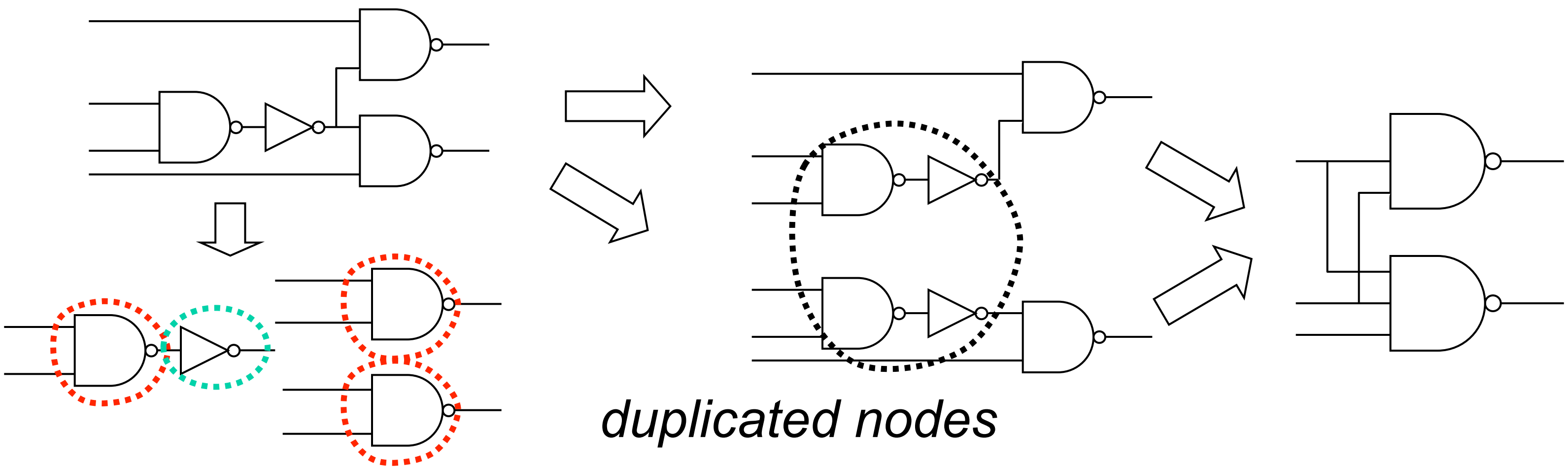
Leaf-DAG Decomposition

- If the gate output has a fan-out of more than 1, disconnect only the output pin from the net. (in tree decomposition, all pins were disconnected)



Problems Arising in DAG Decomposition (2)

2. The solution space for the overall objective of "DAG covering" is restricted by decomposing the target DAG into a tree or a leaf-DAG. Therefore opportunity for deriving the optimal DAG covering can be lost with the decomposition.
3. Single-cone decomposition : At each primary output, exact a "cone" which includes all paths to the primary inputs.



Tree decomposition :
Optimal covering cost = 11

Single-cone decomposition :
Optimal covering cost = 8