Logic Synthesis
Technology Mapping

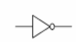
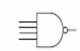
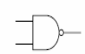
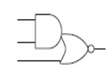
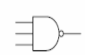
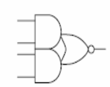
Presenter:
Biện Quang Hoàng
Lương Ngọc Nhơn

Contents

1. Technology Libraries

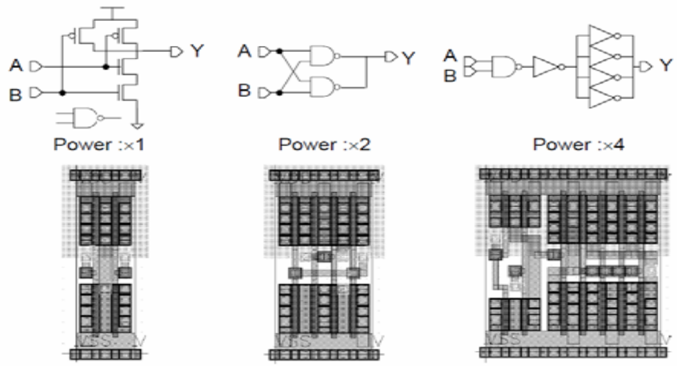
- Gate is primitive element
- Gates are inverter, NAND , NOR gate and complex gates: NOR, XOR gates
- Technology library consists of a finite collection of gates

Gate library example

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

Library gates

NAND2 Cell (Schematic / Layout)



2. What is Technology Mapping (TM)?

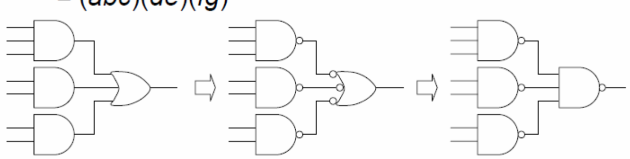
Implement Boolean network using gates of a library

- The goal : optimal use gates of library to produce circuit
- Satisfy delay less, minimum area , heat

Transformation of Boolean Network to NAND Network

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

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



The role of TM

- Is to Choice gates to implement equations
- Is Not to reduce number of levels of logic
- Is NOT to change circuit structure

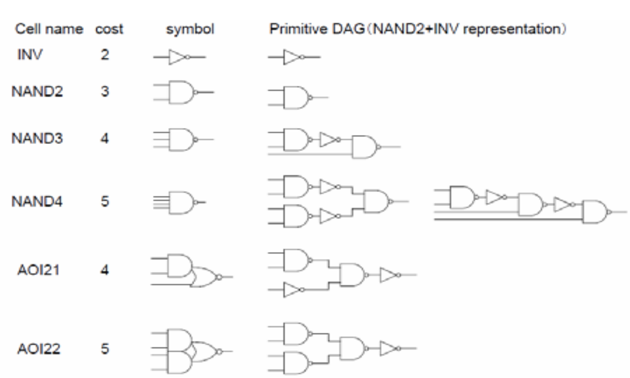
3. Graph Covering

- TM is based on graph covering
- Cover is a collection of a pattern graphs
- Each node in Boolean network can be replaced by NAND gate
- Each gate is a form in Figure7.6

Subject DAG

- Realization Or Pattern for each library gate in terms of 2-input NAND and inverter
- Realization or Pattern is a primitive DAG
- Form of Boolean network is a subject DAG

Primitive DAG



Boolean network to NAND network

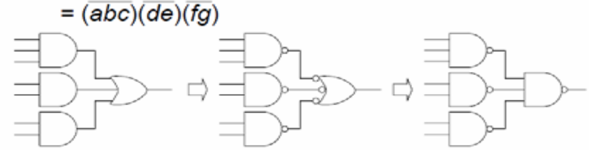
Transformation of Boolean Network to NAND Network

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

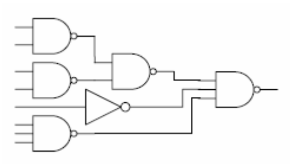
$F = abc + de + fg$

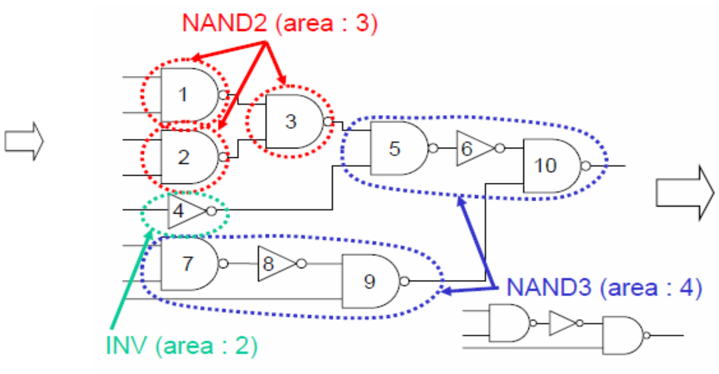
$= \overline{\overline{(abc)}} + \overline{\overline{(de)}} + \overline{\overline{(fg)}}$

$= \overline{\overline{(abc)}\,\overline{(de)}\,\overline{(fg)}}$



A NAND network and do TM



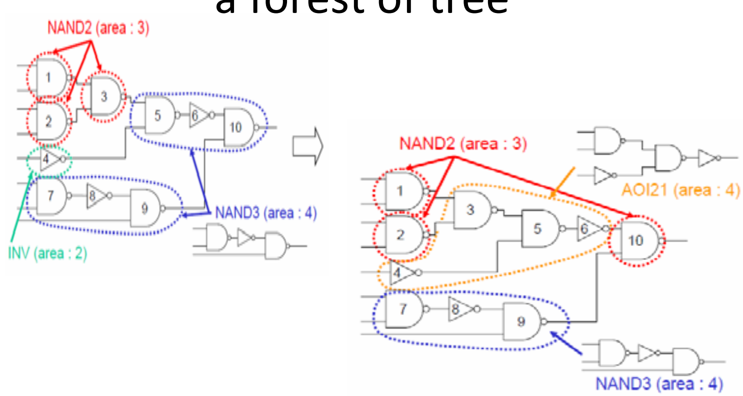


4. Part 7.8: TM by Tree Covering

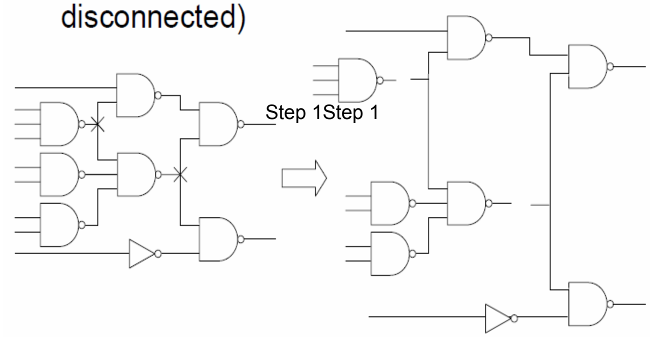
- A tree is a DAG
  - Tree output: root
  - Tree input: leaves

- Step 1: Partition subject graph into a forest of tree
- Step 2: Decomposition

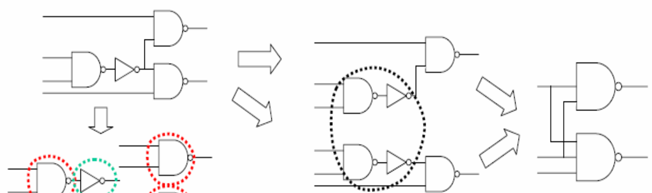
Step 1: Partition subject graph into a forest of tree



- 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)



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.



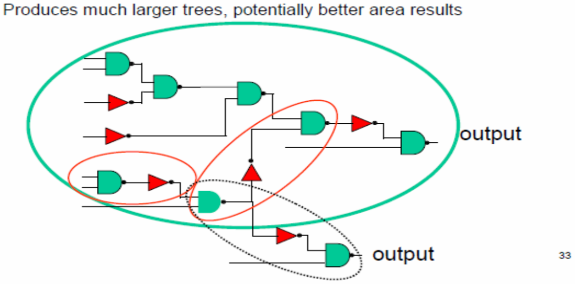
Tree decomposition :
Optimal covering cost = 11

Single-cone decomposition :
Optimal covering cost = 8

Single-cone partition

Single-cone partition:

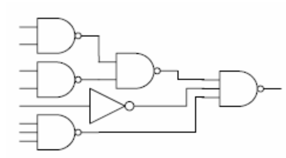
- from a single output, form a large tree back to the primary inputs;
- map successive outputs until they hit match output formed from mapping previous primary outputs.
  - Duplicates some logic (where trees overlap)
  - Produces much larger trees, potentially better area results



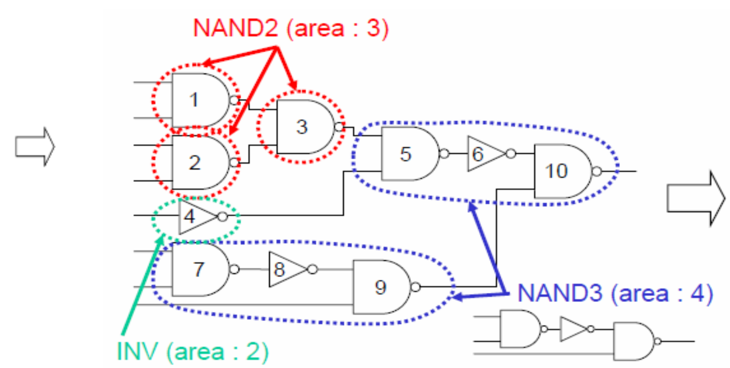
33

Step 2: Decomposition

- Each node in Boolean network can be replaced by NAND gate
- Each node in a NAND tree is replaced by n-input NAND tree is decomposed into a NAND2-tree
-



3-input NAND is decomposed into NAND2

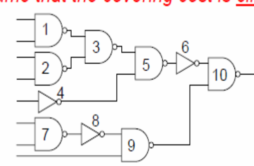


5. Optimal Tree Covering

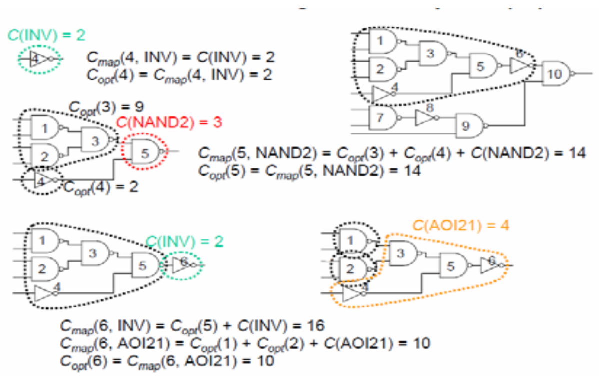
- Find minimum area cover
- If the subject DAG and primitive DAG's are trees, then an efficient algorithm to find the best cover exists
- •Based on dynamic programming

Tree Covering Approach (1)

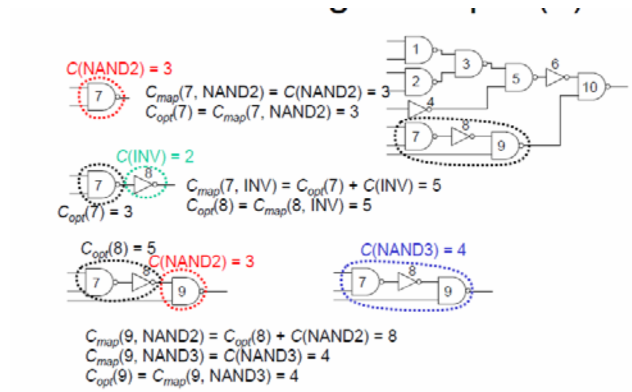
- Definition of *tree* graph :
  - Each node consist of several *child* nodes and a *parent* node.
  - A *root* is a node with no parent node. (only one root in a tree)
  - A *leaf* is a node with no child nodes.
- Divide the covering problem on tree *T* into smaller covering problems on the *subtrees* 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*



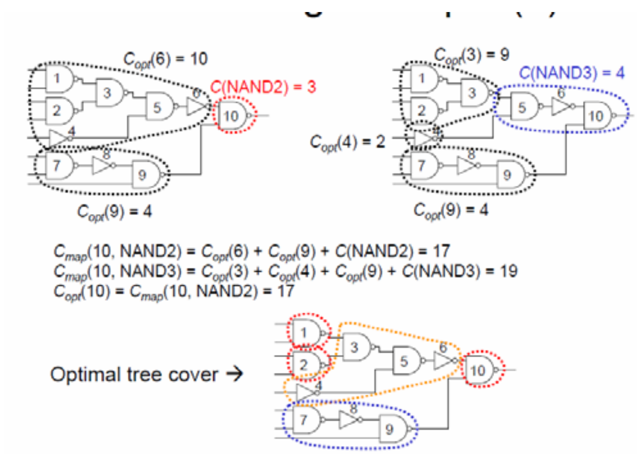
Target NAND2-tree



- The same tree but Cmap(6, AOI21) min < Cmap(6, INV)
- → choose covering of Copt(6)



- Cmap(9, NAND3) min → choose this cover



- Cmap(10, NAND2) min → choose this cover

6. Q&A

7. Reference

- Chapter 7.7 and 7.8 of **Logic Senthesis– SrinivasDevadas, AbhijitGhosh, Kurt Keutzer**
- **VLSI System Design Course**
- **Tsuyoshi Isshiki**
  Dept. Communications and Integrated Systems , Tokyo Institute of Technology
- http://www.vlsi.ss.titech.ac.jp/~isshiki/VLSISystemDesign/top.html