CMPE 212
Principles of Digital Design

Lecture 9

Synthesis of Combinational Circuits

February 22, 2016

www.csee.umbc.edu/~younis/CMPE212/CMPE212.htm

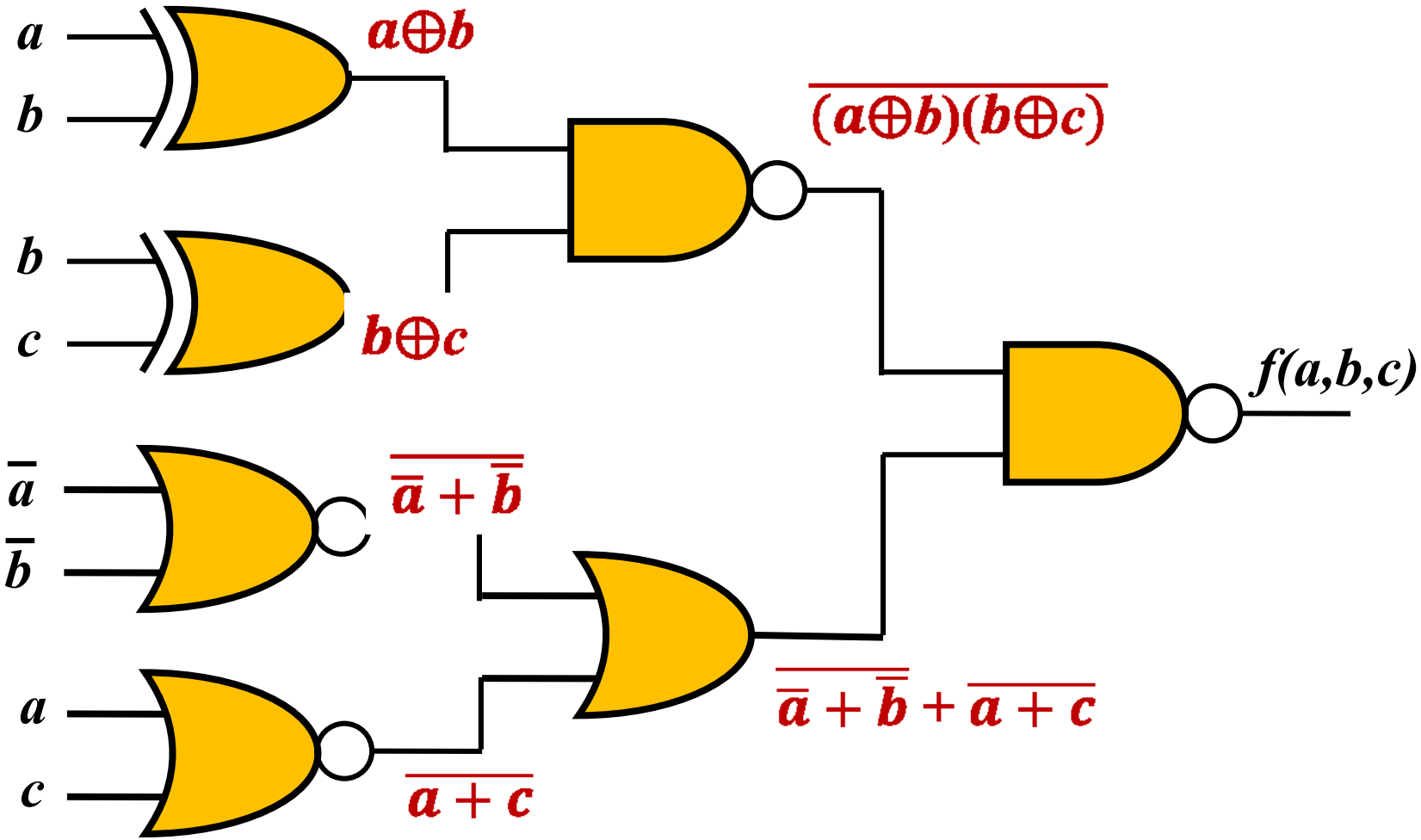
Lecture's Overview

- ***Previous Lecture:***
  - ➔ Canonical form of switching functions
    (conversion from simplified to canonical form)
  - ➔ Analyzing switching circuits using algebraic methods
    (Truth table and Derivation of logic function)
  - ➔ Analysis of timing diagram
  - ➔ Effect of physical characteristics
    (propagation delay and power dissipation)
- ***This Lecture***
  - ➔ Synthesis of combinational logic circuits

Analysis of Combinational Circuits

- Digital circuits are designed by transforming a word description of a function into a switching equation and then a circuit
  - ➔ Digital circuit analysis is the opposite process
- A digital circuit can be described by:
  1) Switching function (algebraic method)
  2) Hardware design language module
  3) Truth tables
  4) Timing diagram
- Analysis of a logic circuit is used to:
  - determine that its behavior of matches specifications
  - transform the circuit to a different format to optimize the implementation

Example: Algebraic Method



$$f(a,b,c) = \overline{\overline{(a \oplus b)(b \oplus c)}.\ (\overline{\bar{a} + \bar{b}} + \overline{a + c})}$$

Algebraic Simplification

$$f(a,b,c) = \overline{\overline{(a \oplus b)(b \oplus c)}.\ (\overline{\overline{a} + \overline{b}} + \overline{a + c})}$$

$$= \overline{\overline{(a \oplus b)(b \oplus c)}} + \overline{(\overline{\overline{a} + \overline{b}} + \overline{a + c})}$$

$$= (a \oplus b)(b \oplus c) + (\overline{a} + \overline{b})(a + c)$$

$$= (a\overline{b} + \overline{a}\,b)(b\overline{c} + \overline{b}\,c) + \overline{a}a + \overline{a}c + \overline{b}a + \overline{b}c$$

$$= a\overline{b}b\overline{c} + a\overline{b}\overline{b}c + \overline{a}bb\overline{c} + \overline{a}b\overline{b}c + 0 + \overline{a}c + \overline{b}a + \overline{b}c$$

$$= 0 + a\overline{b}c + \overline{a}b\overline{c} + \overline{a}c + \overline{b}a + \overline{b}c$$

$$= \overline{a}b\overline{c} + \overline{a}c + \overline{b}a + \overline{b}c$$

Consensus

$$= \overline{a}b\overline{c} + \overline{a}c + \overline{b}a \quad = \overline{a}(b\overline{c} + c) + a\overline{b}$$

$$= \overline{a}(b + c) + a\overline{b} = \overline{a}b + \overline{a}c + a\overline{b} = (a \oplus b) + \overline{a}c$$

Truth Table Method

- Build the truth table for the circuit and then derive a simplified switching function using SOP or POS
- Example:

to simplify the previous circuit

$$\prod M(0,1,7)$$

a	b	c	$\overline{(a \oplus b)(b \oplus c)}.\ (\overline{\overline{a} + \overline{b}} + \overline{a + c})$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

$$f(a,b,c) = (a+b+c)(\overline{a}+\overline{b}+c)(\overline{a}+\overline{b}+\overline{c}) = (a+b+c)(\overline{a}+\overline{b})$$

$$= a\overline{a} + a\overline{b} + b\overline{a} + b\overline{b} + \overline{a}c + \overline{b}c \quad = a\overline{b} + b\overline{a} + \overline{a}c + \overline{b}c$$

$$= a\overline{b} + b\overline{a} + \overline{a}c + \overline{b}c \quad = a\overline{b} + b\overline{a} + \overline{a}c \quad = (a \oplus b) + \overline{a}c$$

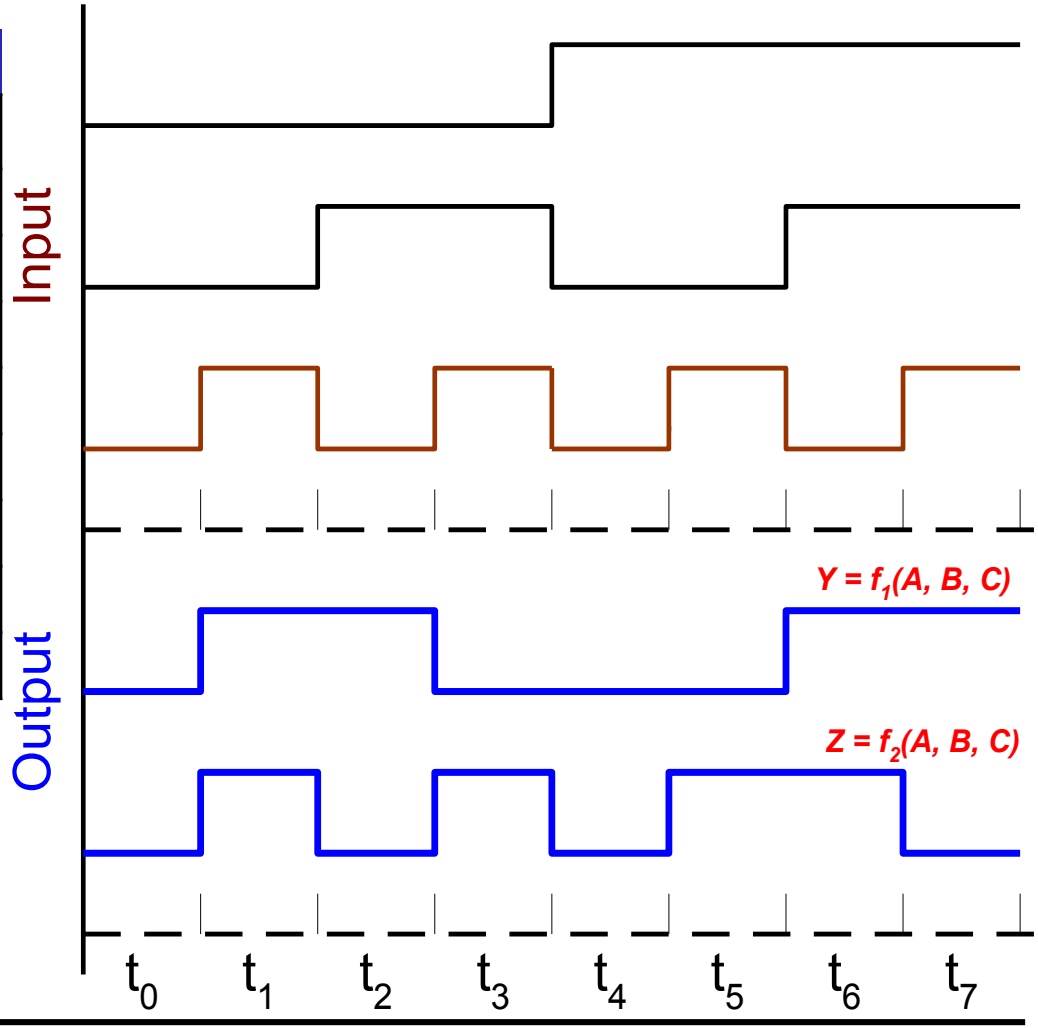
Timing Diagram

- Apply a sequence of input and observe corresponding output
- Useful for analyzing the propagation delay:

Time	Input (A, B, C)	Output	
		$f_1(A, B, C)$	$f_2(A, B, C)$
t_0	000	0	0
t_1	001	1	1
t_2	010	1	0
t_3	011	0	1
t_4	100	0	0
t_5	101	0	1
t_6	110	1	1
t_7	111	1	0

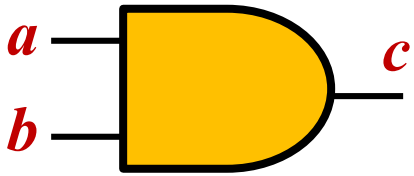
$$f_1(A,B,C) = \sum m(1,2,6,7)$$

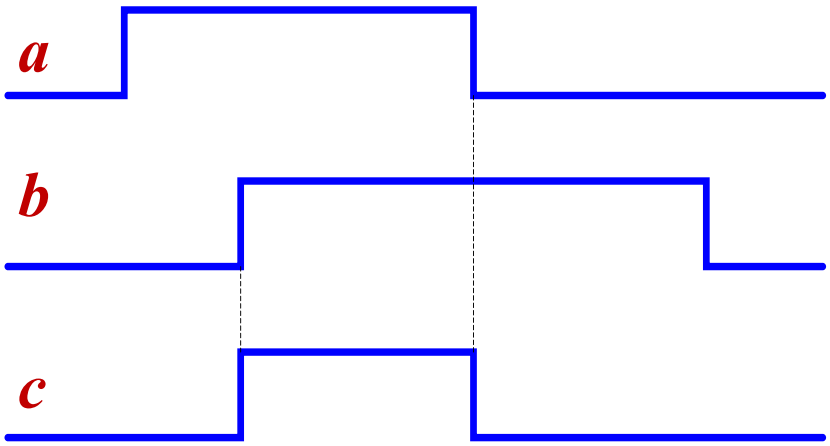
$$f_2(A,B,C) = \sum m(1,3,5,6)$$



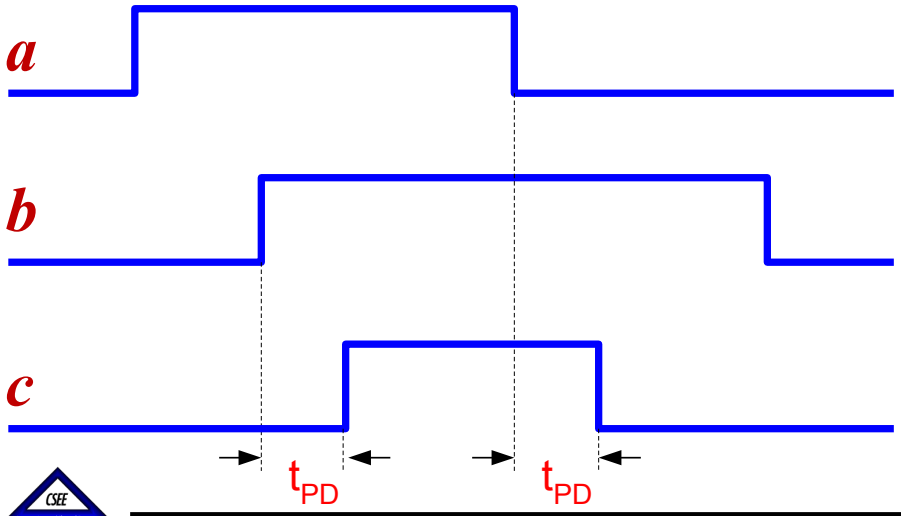
Propagation Delay

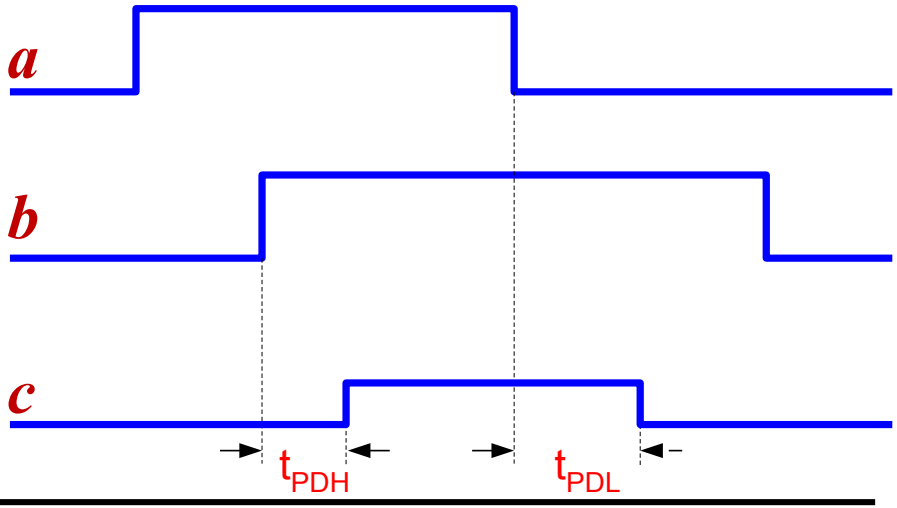
- Propagation delay is time for the output to become ready after the input gets changed





- Propagation delay depends on the microelectronics technology and size
- Rising and falling time may differ
- $t_{PD} = ½ (t_{PDH} + t_{PDL})$



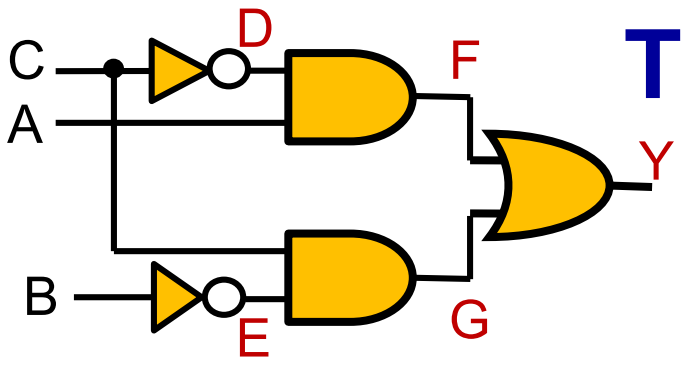


Important Physical Characteristics

- Physical characteristics vary depending on the microelectronics technology used in the design and fabrication
- There is a trade-off between speed, power dissipation and cost

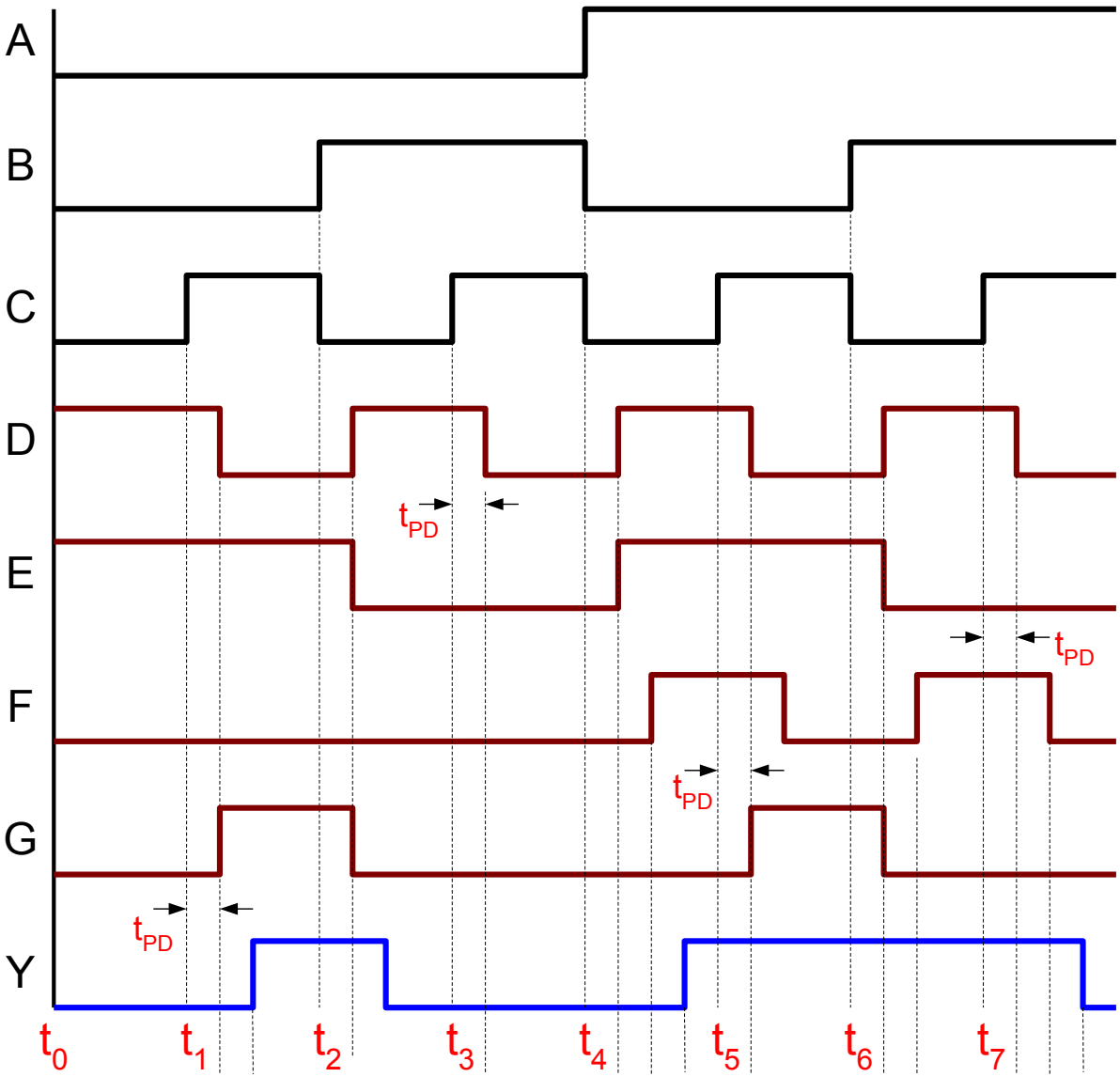
Logic family	Propagation Delay t_{PD} (ns)	Power Dissipation Per Gate (mW)	Technology
7400	10	10	Standard TTL
74H00	6	22	High-speed TTL
74L00	33	1	Low-power TTL
74LS00	9.5	2	Low-power Schottky TTL
74S00	3	19	Schottky TTL
74ALS00	3.5	1.3	Advanced low-power Schottky TTL
74AS00	3	8	Advanced Schottky TTL
74HC00	8	0.17	High-speed CMOS

Timing Diagram with Delay



ABC	Y=f(A,B,C)
000	0
001	1
010	0
011	0
100	1
101	1
110	1
111	0

$$f(A,B,C) = \sum m(1,4,5,6)$$



Logic Synthesis

Definition: To design a logic circuit such that it meets the specifications and can be economically manufactured:

- Performance – meets delay specification, or has minimum delay.
- Cost – uses minimum hardware, smallest chip area, smallest number of gates or transistors.
- Power – meets power specification, or consumes minimum power.
- Testablility – has no redundant (untestable) logic and is easily testable.

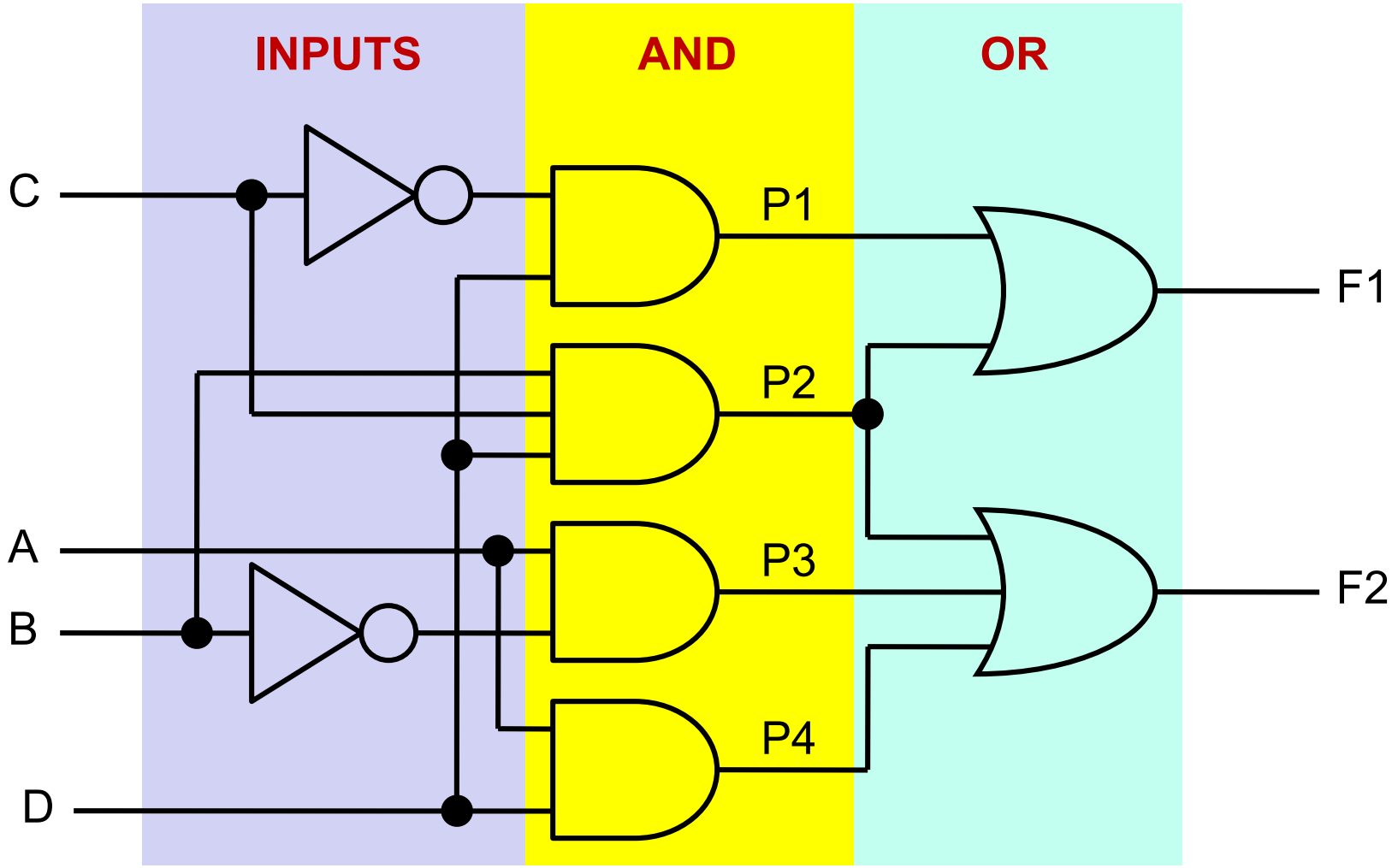
Procedure:

1. Minimization – Obtain MSOP or MPOS. This is also known as two-level minimization because the result can be implemented as a two-level AND-OR or NAND-NAND or NOR-NOR circuit.
2. Technology mapping – Considering design requirements, transform the minimized form into one of the technologically realizable forms:
   - Programmable logic array (PLA)
   - Field programmable gate array (FPGA)
   - Others . . .

Slide contents are courtesy of Vishwani D. Agrawal

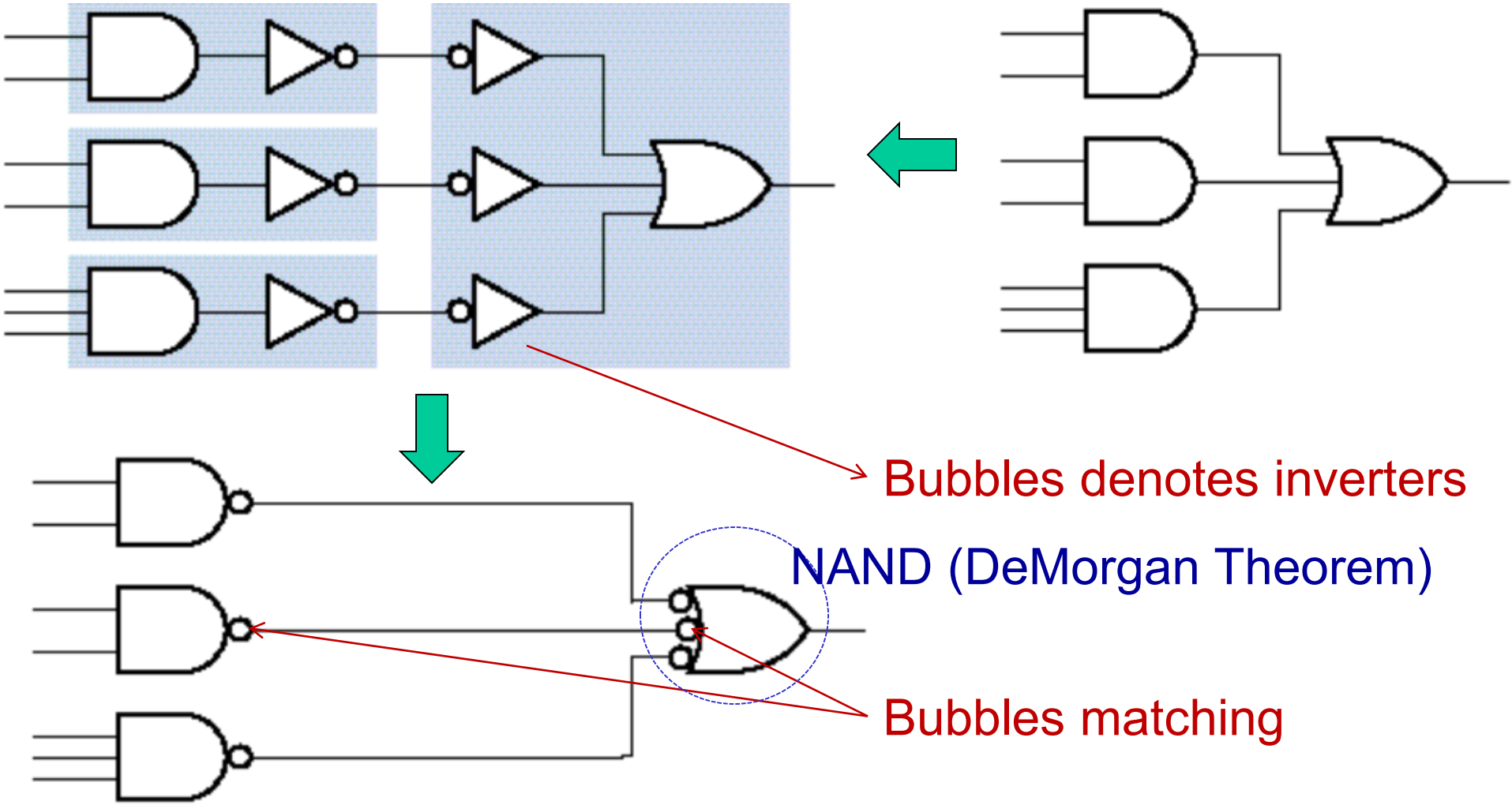
Two-Level AND-OR Implementation

- Also known as technology-independent circuit.



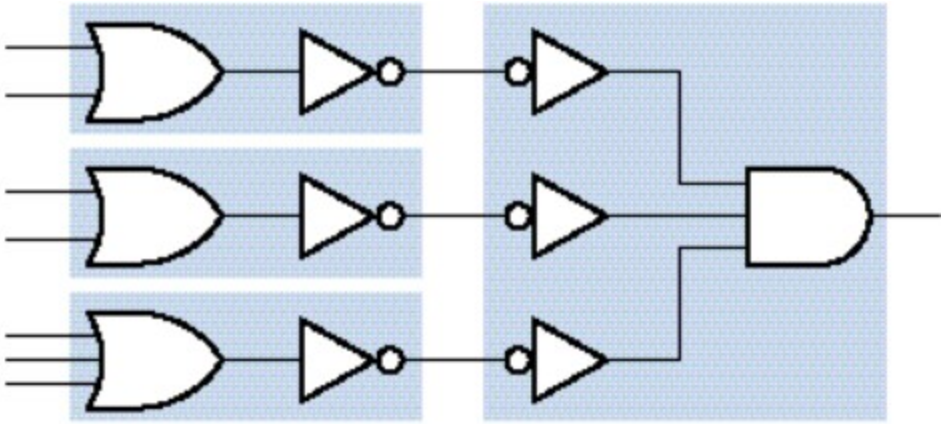
Slide contents are courtesy of Vishwani D. Agrawal

Bubbles Matching and Gate Conversion

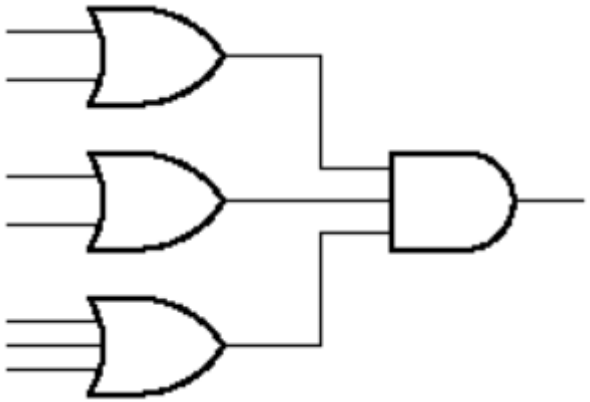


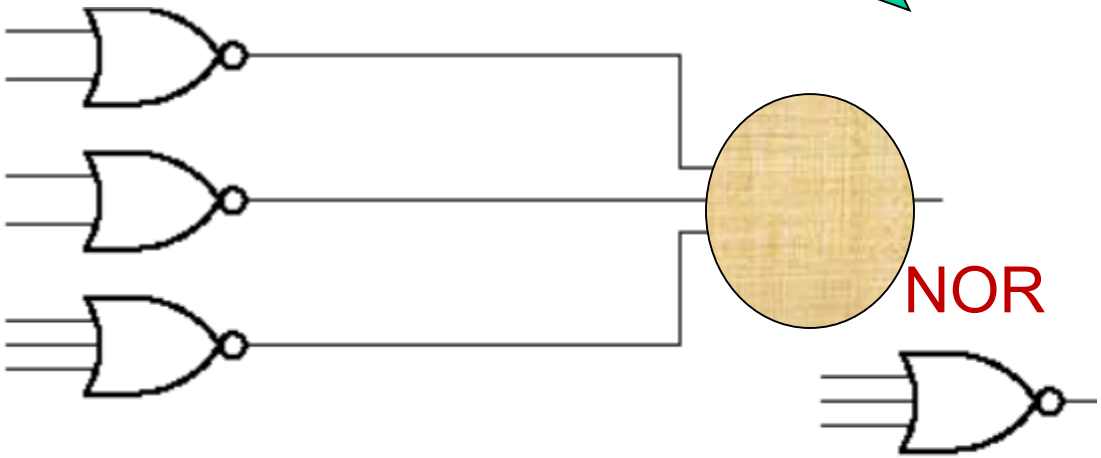
Implementing NAND (and NOR) gates requires fewer transistors than AND (and OR) gates➔ Convert AND-OR to NAND-NAND or NOR-NOR

OR-AND Conversion to NOR-NOR



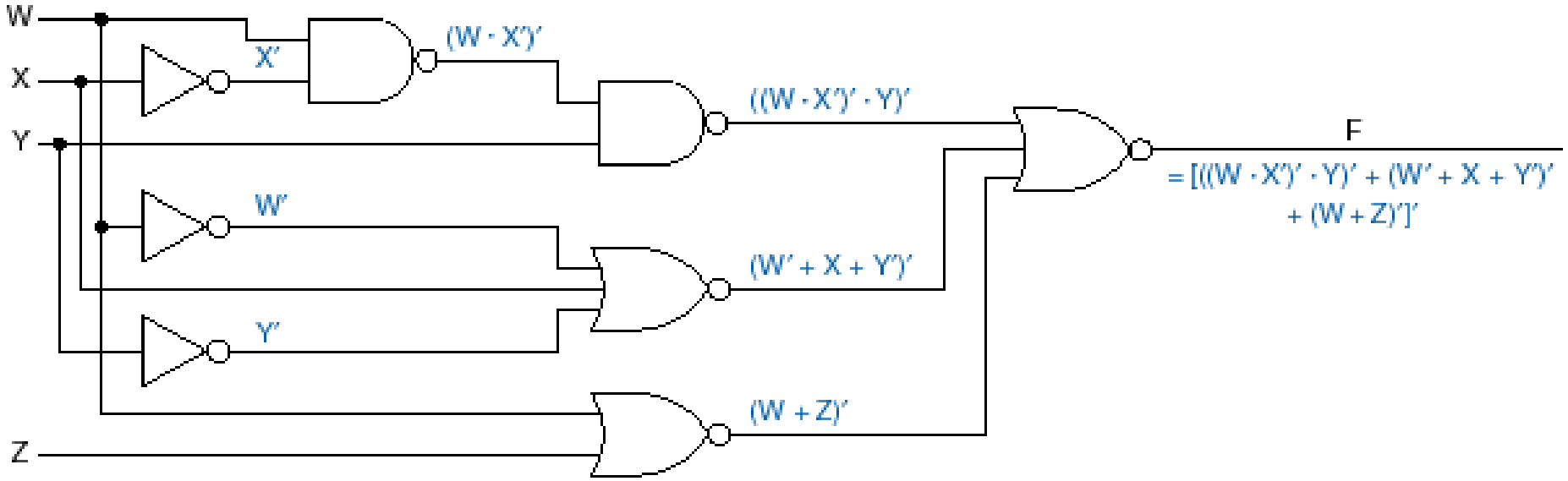
Bubble matching



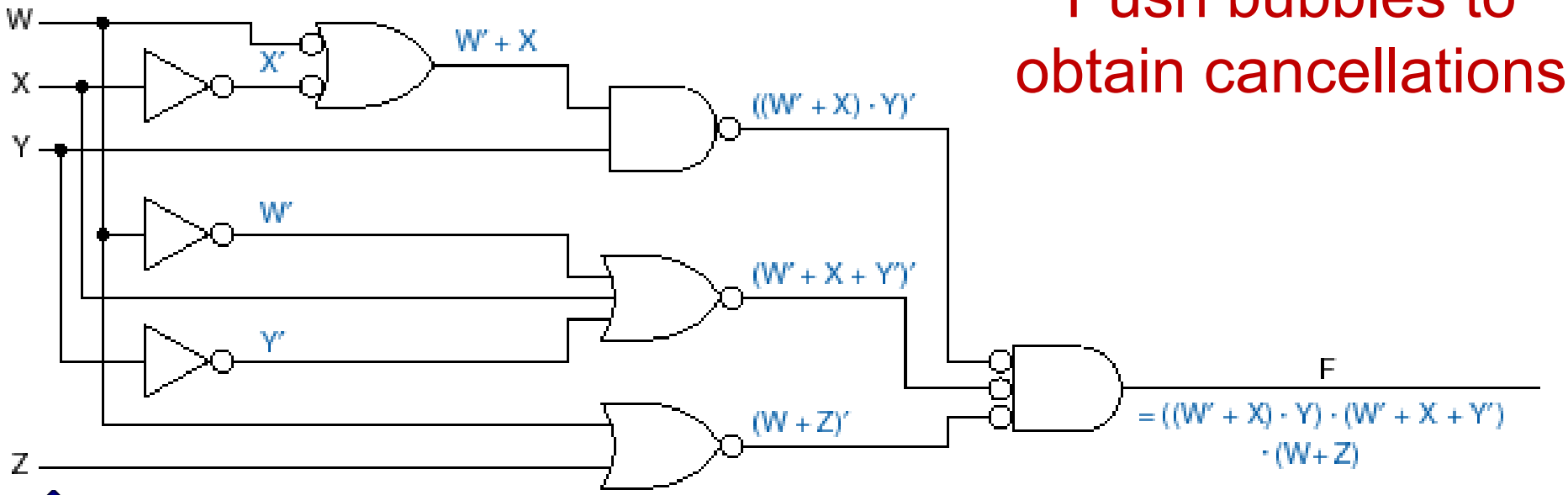


Can convert POS expression to 2-Level NOR design

Inverse Process



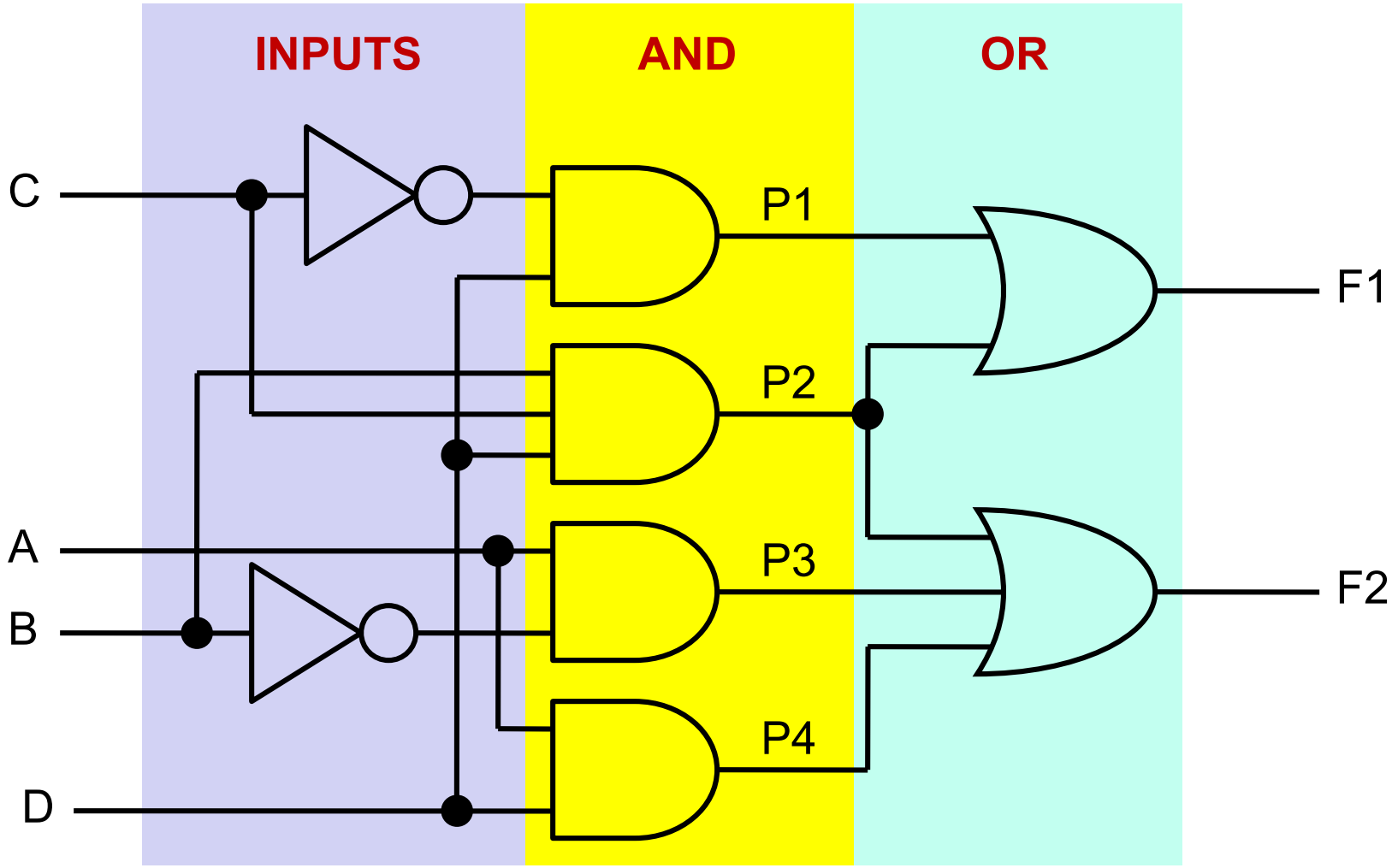
Push bubbles to obtain cancellations



Picture courtesy of Dhananjay Phatak

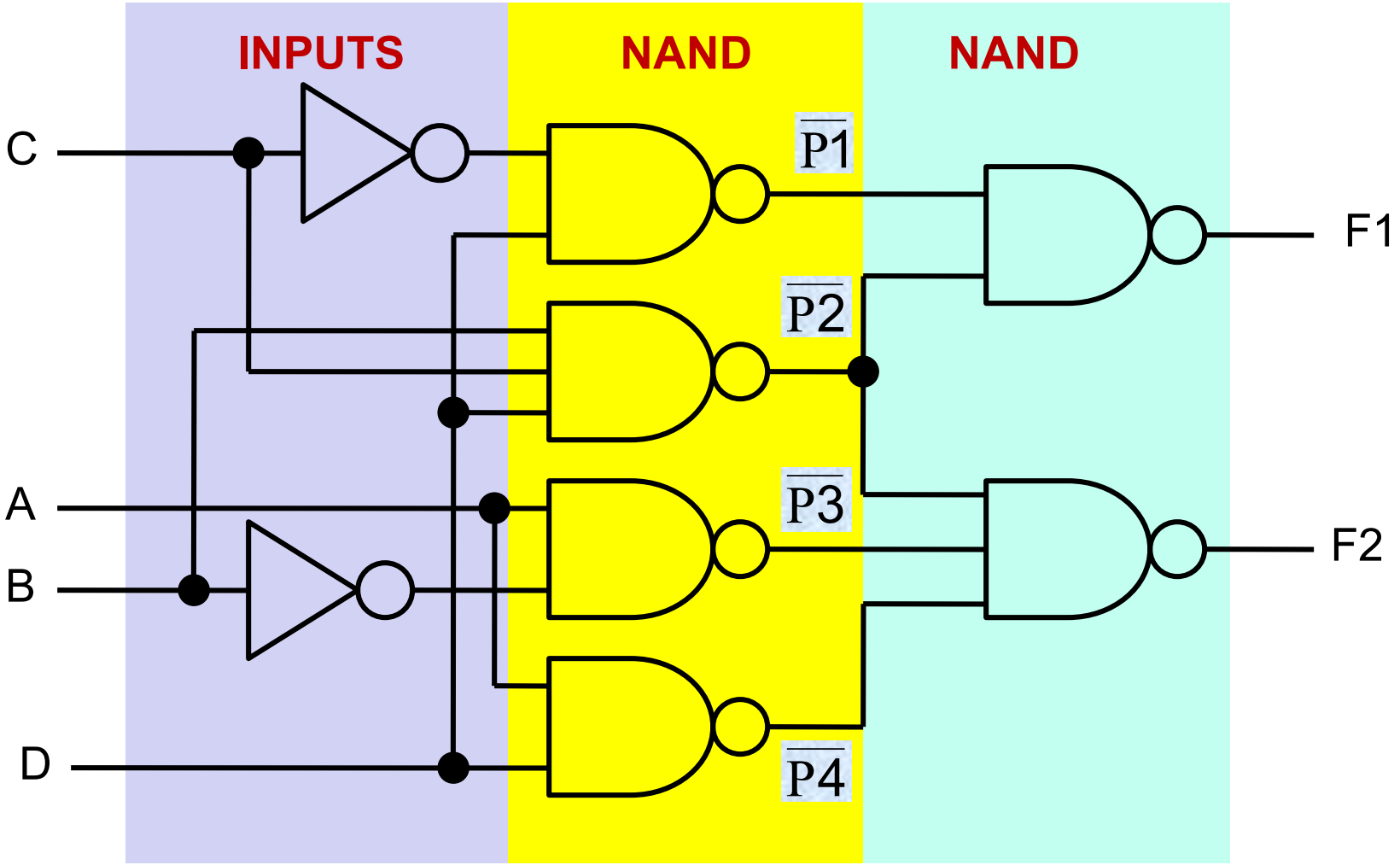
Two-Level AND-OR Implementation

- Also known as technology-independent circuit.



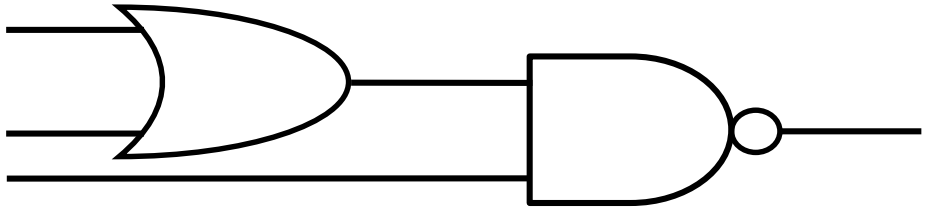
Slide contents are courtesy of Vishwani D. Agrawal

NAND-NAND Implementation

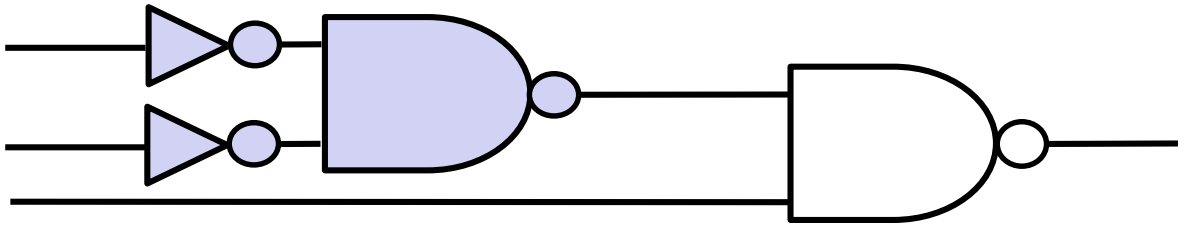


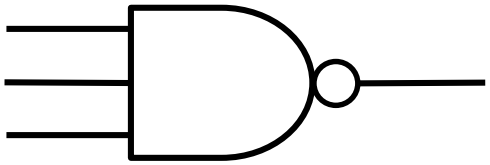
Slide contents are courtesy of Vishwani D. Agrawal

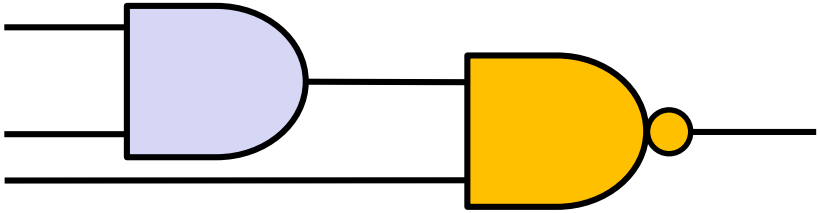
Popular Cells

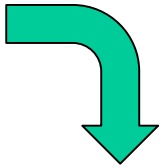


OR-AND-Inverter Cell



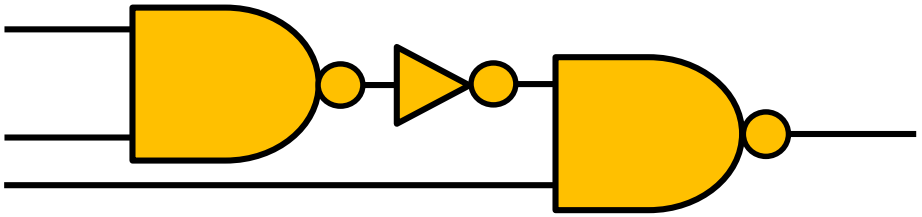




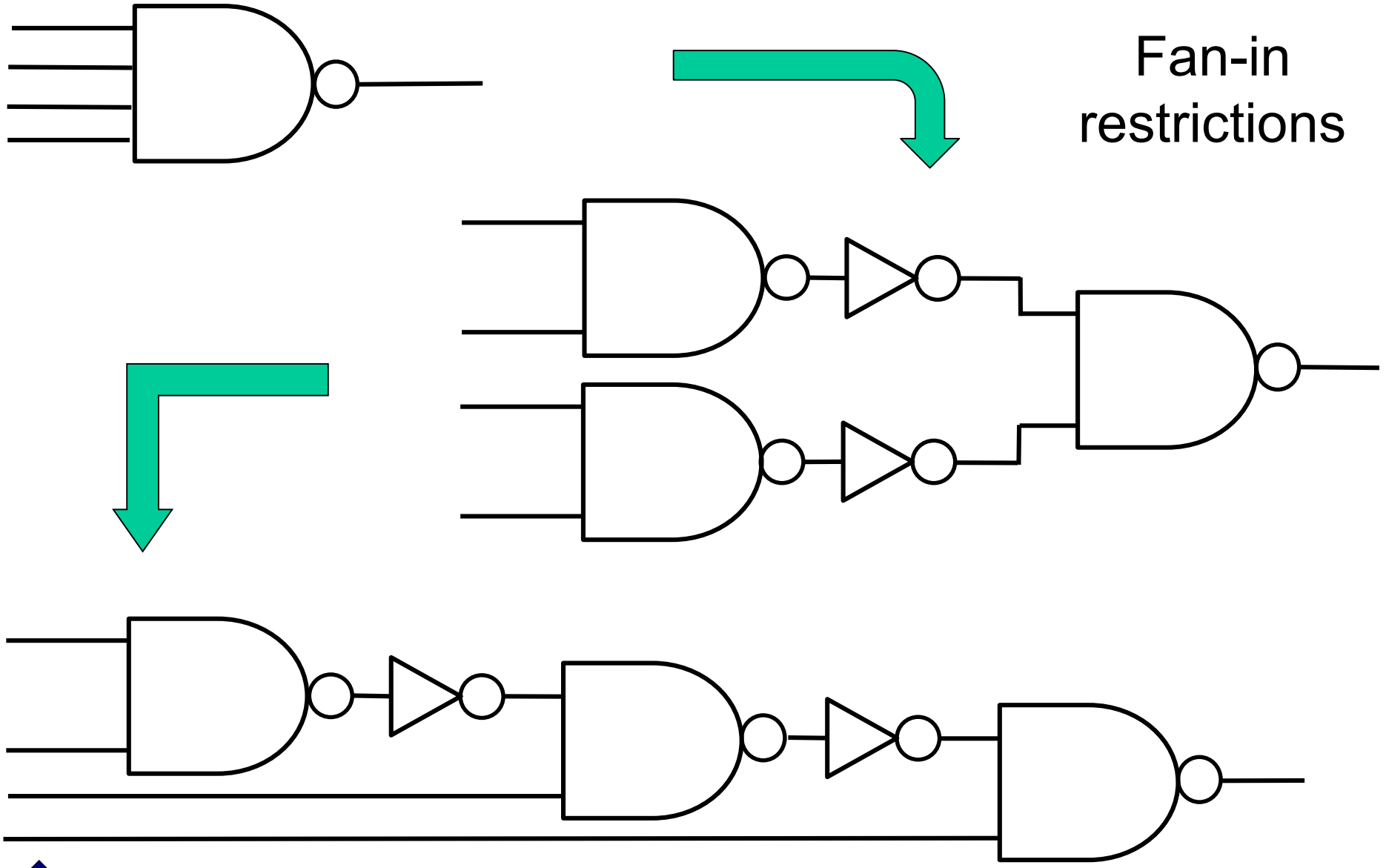


NAND3 Cell

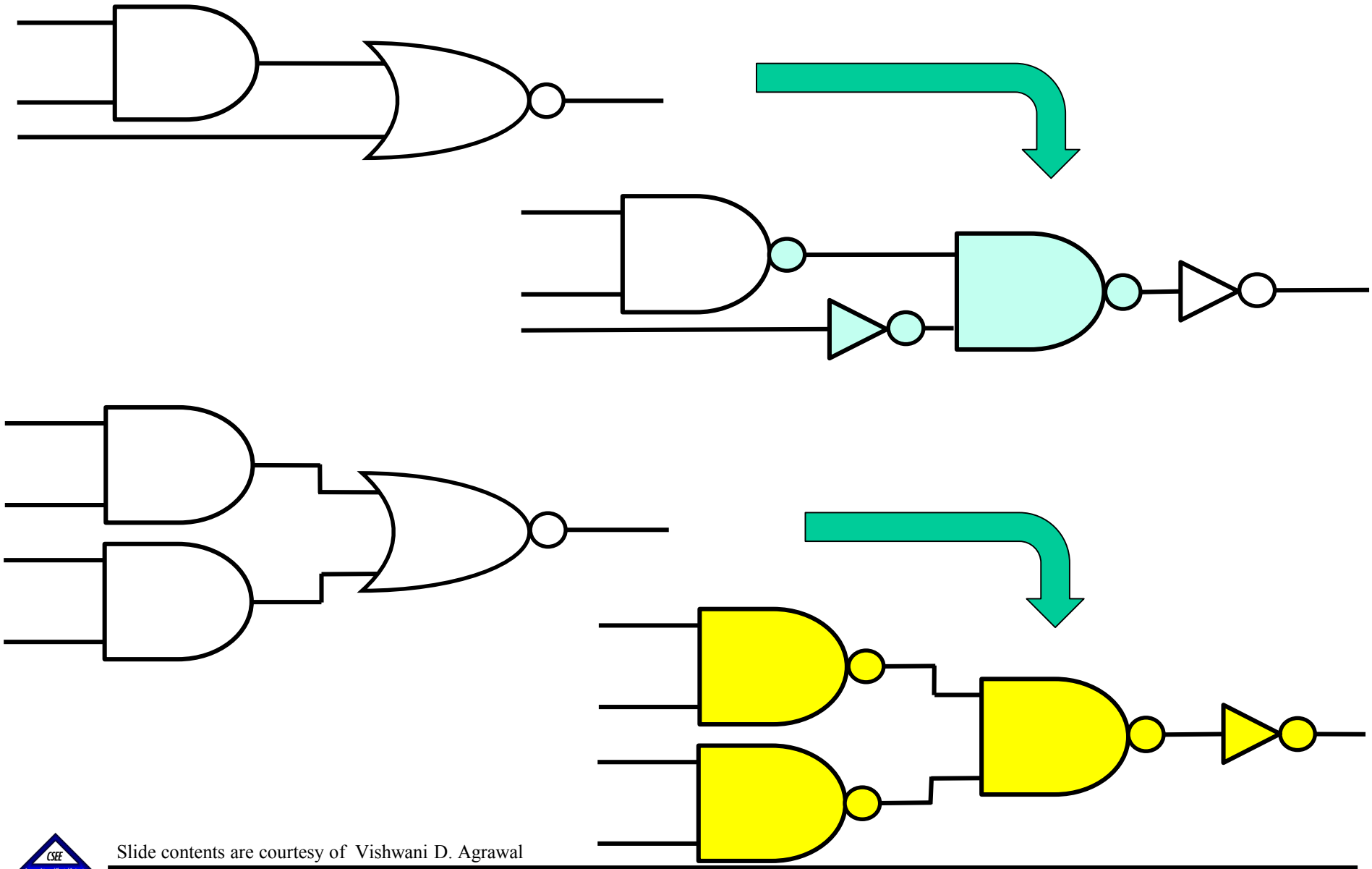
Fan-in restrictions



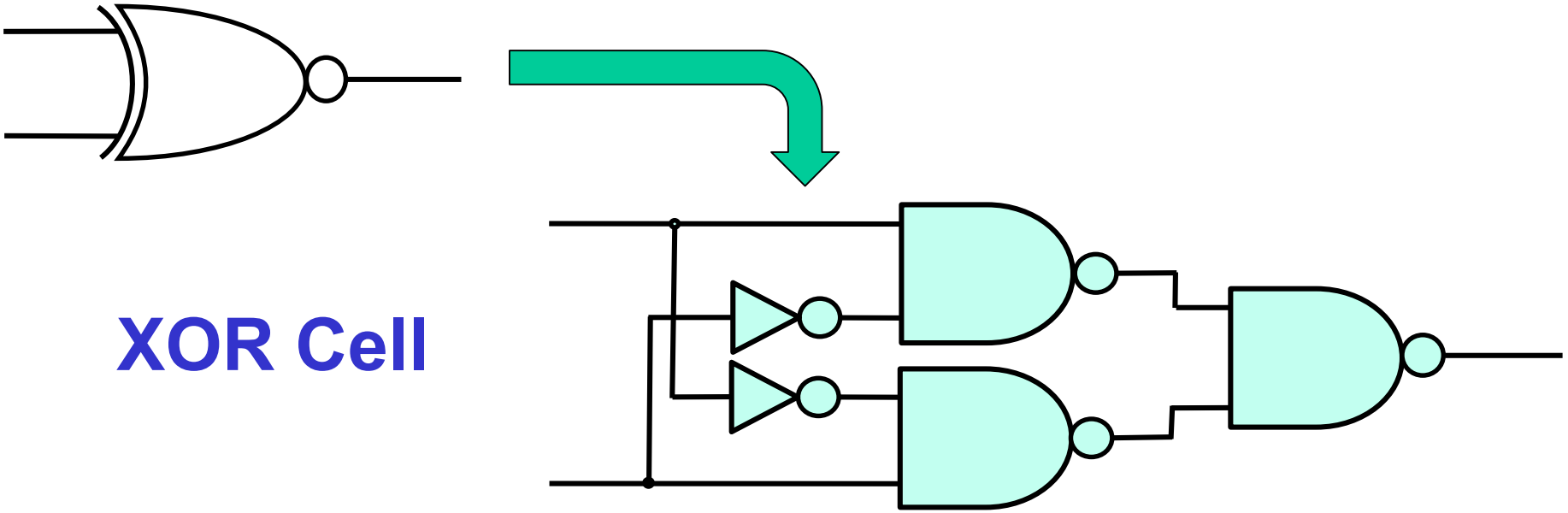
4-Input NAND Cell



AND-OR-Inverter Cell



Technology Mapping



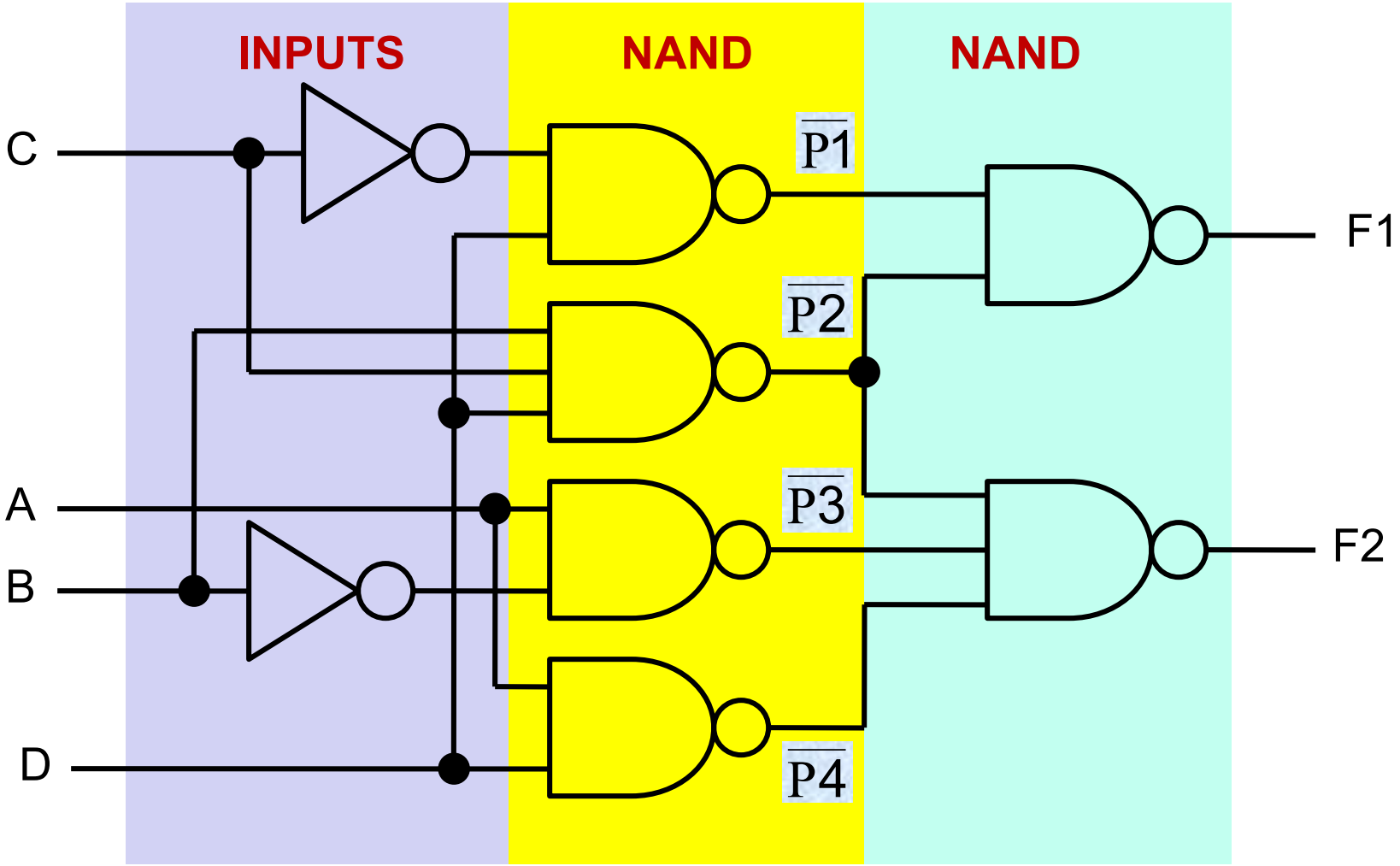
XOR Cell

Procedure

1. Obtain SOP
2. Convert to two-level AND-OR circuit.
3. Transform to two-level NAND-NAND circuit.
4. Transform to two-input NAND and inverter tree network.
5. Perform an optimal pattern matching to obtain a minimum cost tree covering.

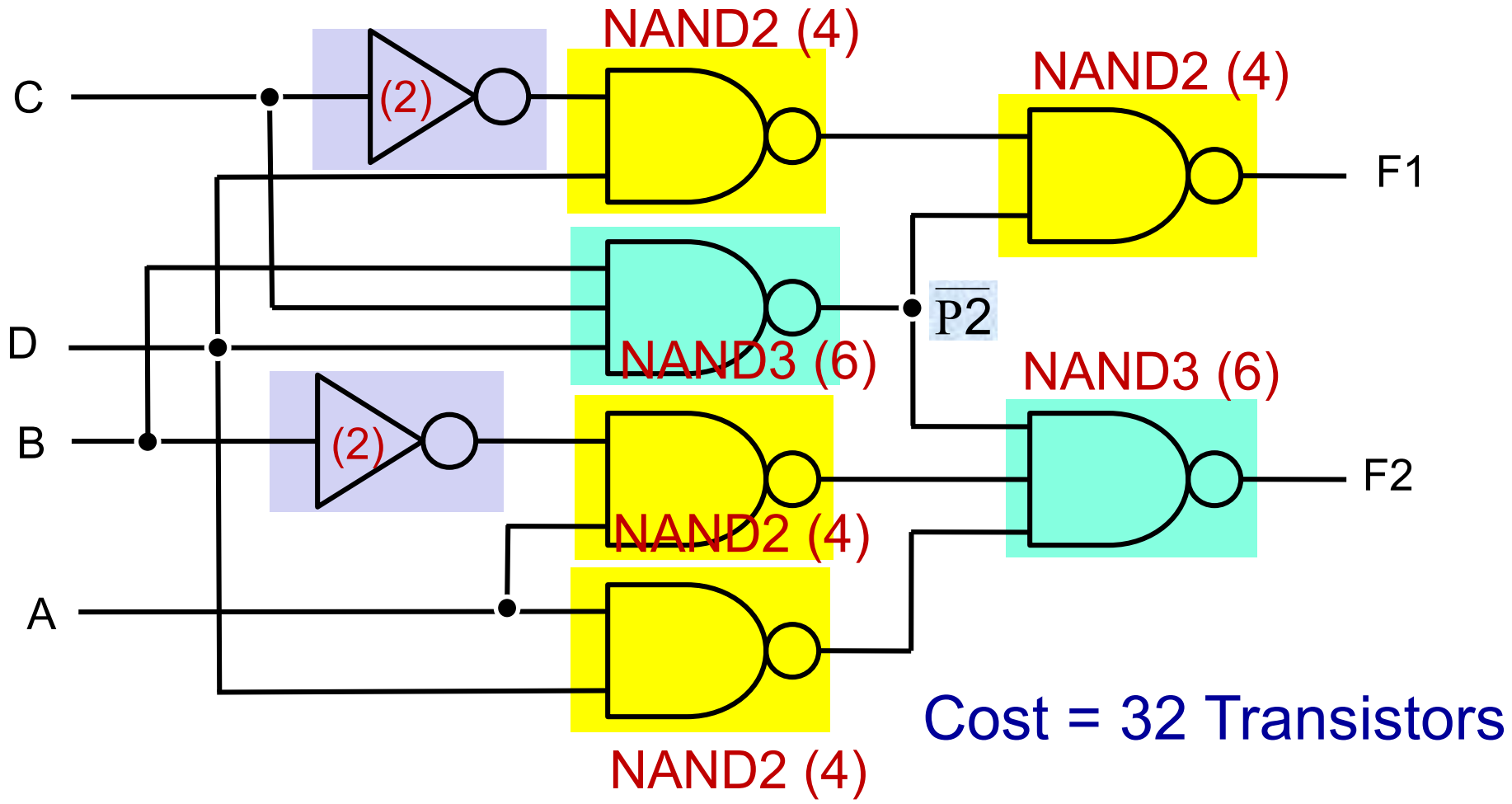
Slide contents are courtesy of Vishwani D. Agrawal

Example: 2 Level NAND Circuit

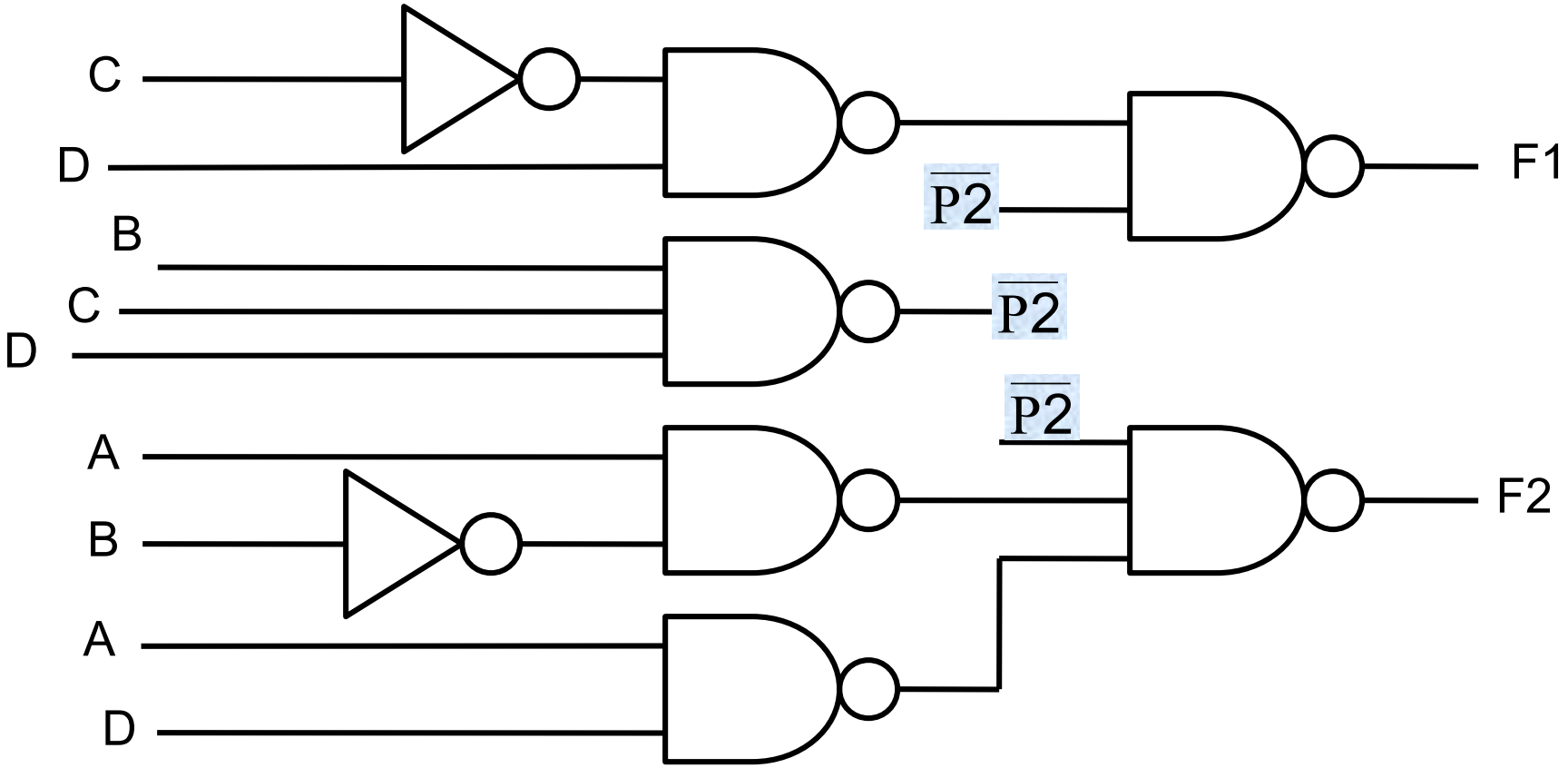


Slide contents are courtesy of Vishwani D. Agrawal

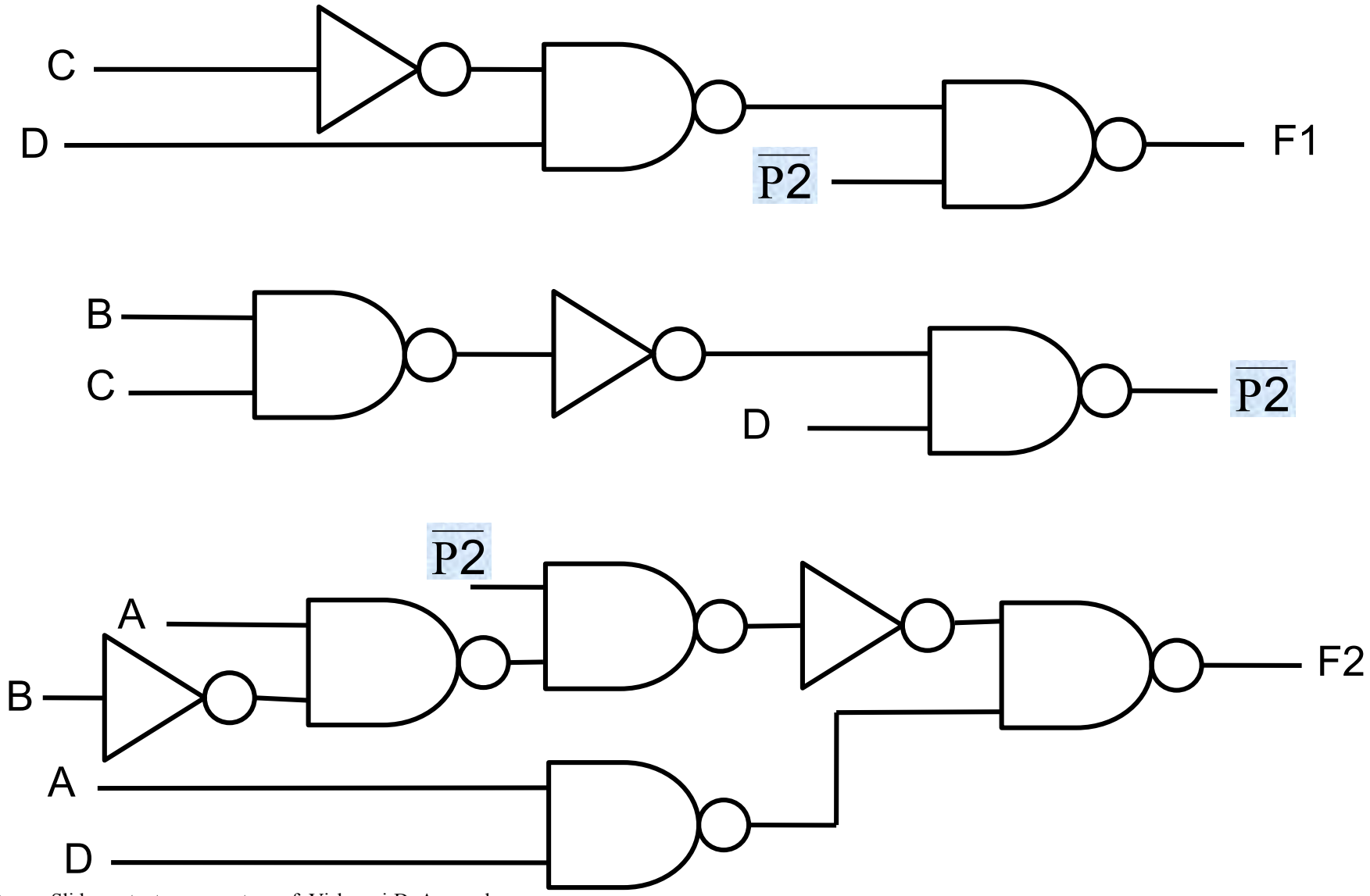
A Simple Technology Mapping



Splitting into a Forest of Trees



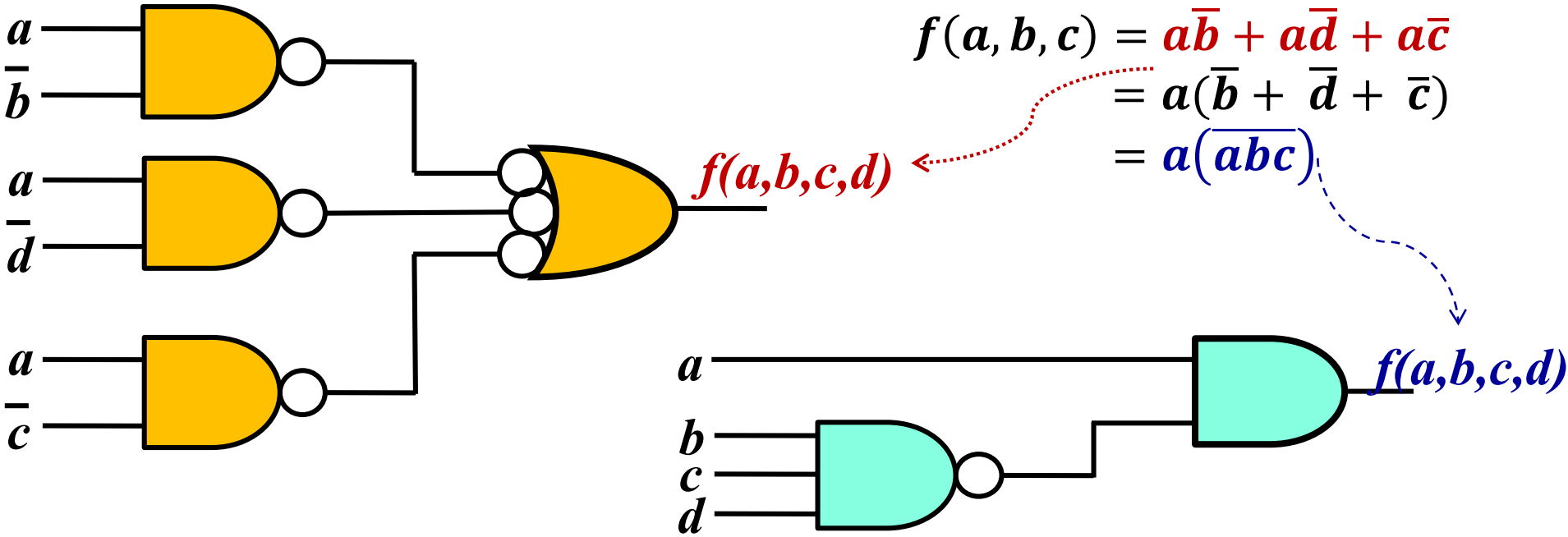
Two-Input NAND Trees



Slide contents are courtesy of Vishwani D. Agrawal

Factoring

- Factoring is a technique to deal with fan-out constraints
- Can also minimize the required hardware
- Very difficult to perform in multi-level logic realization



Redundancy and duplicate logic may be unavoidable for complex fan-out issues, i.e. regenerate the same signal multiple times

Conclusion

- ***Summary***
  - ➔ Logic Synthesis
    (Definition, Procedure)
  - ➔ Gate Type Coversion
    (Role of DeMorgan Theorem, Bubble matching)
  - ➔ Popular Cell Library
    (AND-OR, AND-OR-INVERT, OR-AND, OR_AND_INVERT)
  - ➔ Technology Mapping
    (Splitting forest to trees, Fan-in constraints, Factoring)
- ***Next Lecture***
  - ➔ Simplification of Switching Functions

Reading assignment: Section 2.5 in the textbook