NAND and NOR implementation Lecture 4

Dronacharya Group of Institutions

Overview

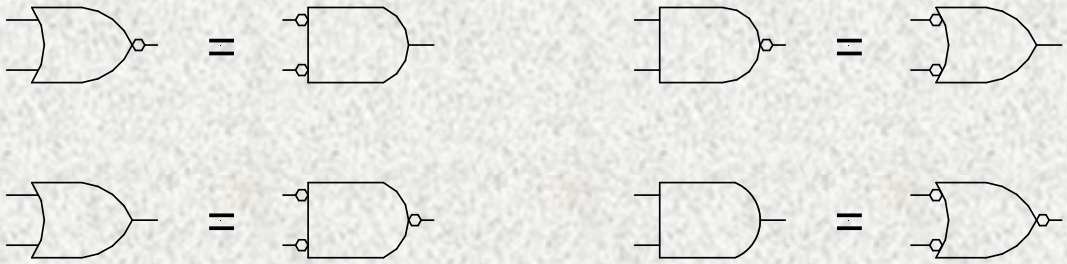
- Developing NAND circuits from K-maps
- Two-level implementations
  - Convert from AND/OR to NAND (again!)
- Multi-level NAND implementations
  - Convert from a network of AND/ORs
- Exclusive OR
  - Comparison with SOP
- Parity checking and detecting circuitry
  - Efficient with XOR gates!

NAND-NAND & NOR-NOR Networks

DeMorgan's Law:

$(a + b)' = a' b'$ $\quad$ $(a\ b)' = a' + b'$

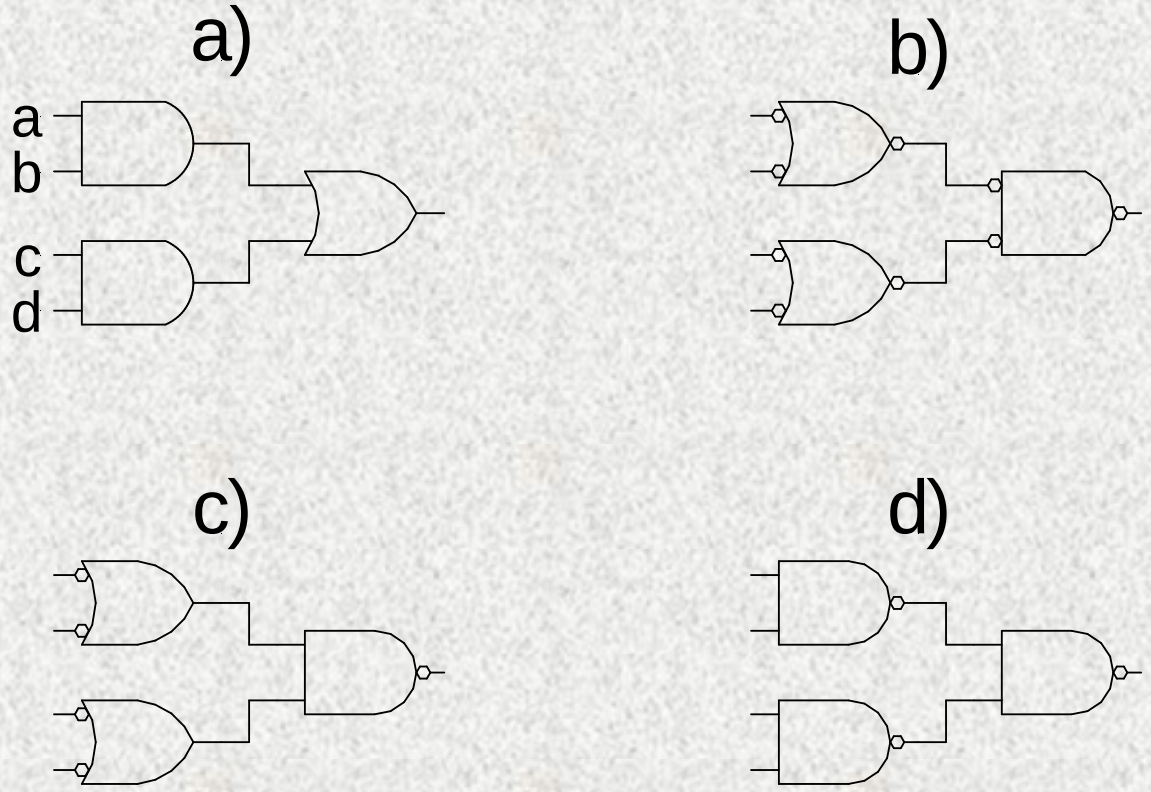
$a + b = (a' b')'$ $\quad$ $(a\ b) = (a' + b')'$



push bubbles or introduce in pairs or remove pairs.

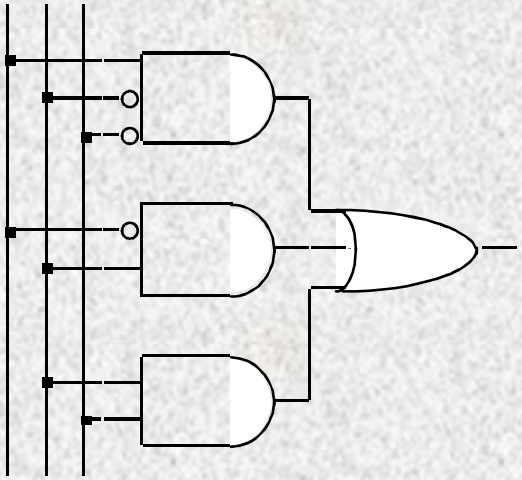
NAND-NAND Networks

- Mapping from AND/OR to NAND/NAND

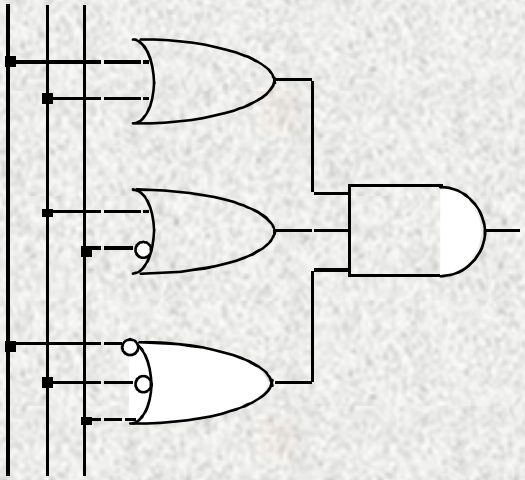


Implementations of Two-level Logic

- Sum-of-products
  - AND gates to form product terms (minterms)
  - OR gate to form sum

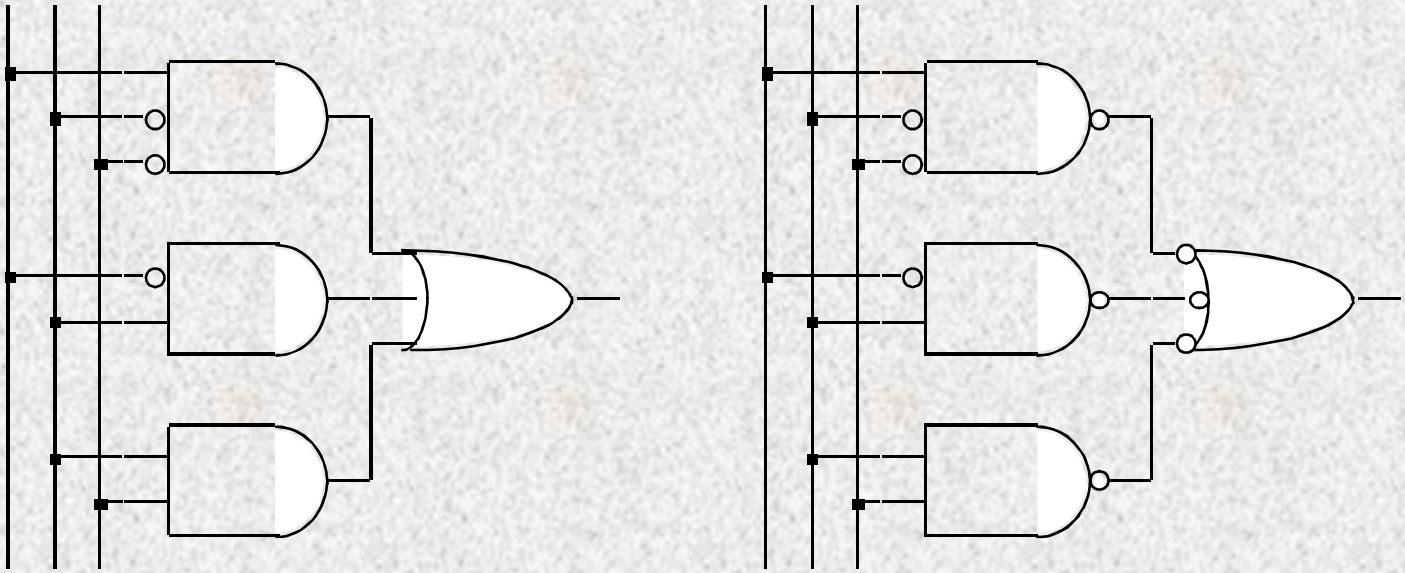


- Product-of-sums
  - OR gates to form sum terms (maxterms)
  - AND gates to form product



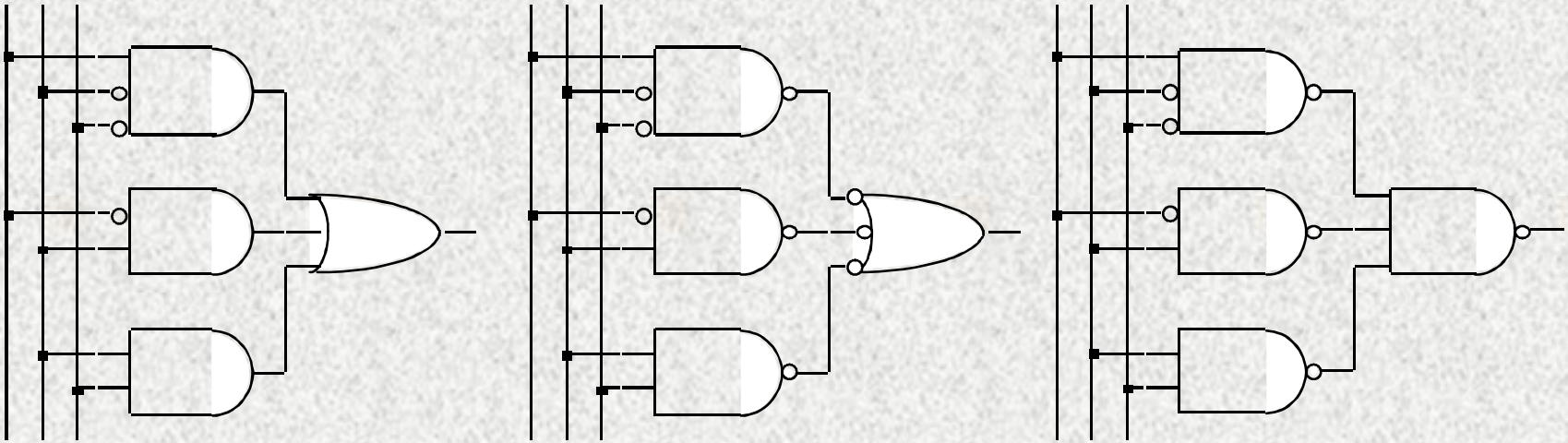
Two-level Logic using NAND Gates

- Replace minterm AND gates with NAND gates
- Place compensating inversion at inputs of OR gate



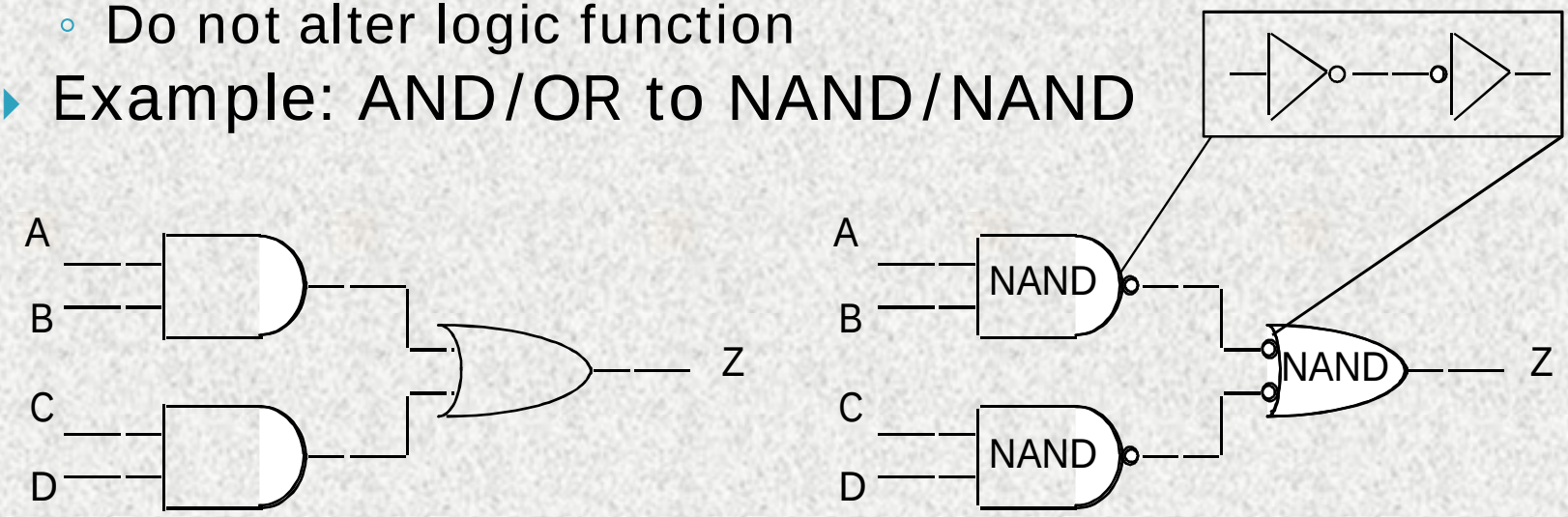
Two-level Logic using NAND Gates (cont'd)

- OR gate with inverted inputs is a NAND gate
  - de Morgan's: A' + B' = (A • B)'
- Two-level NAND-NAND network
  - Inverted inputs are not counted
  - In a typical circuit, inversion is done once and signal distributed



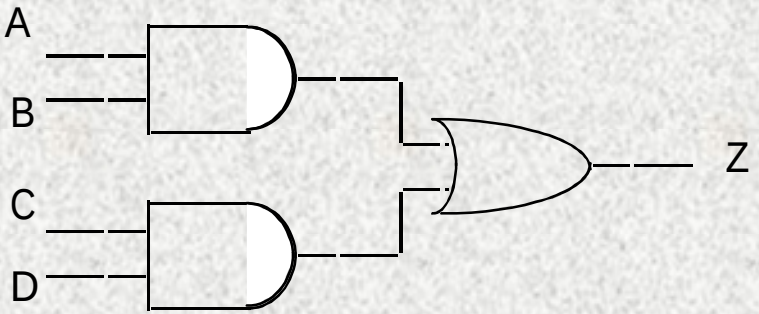
Conversion Between Forms

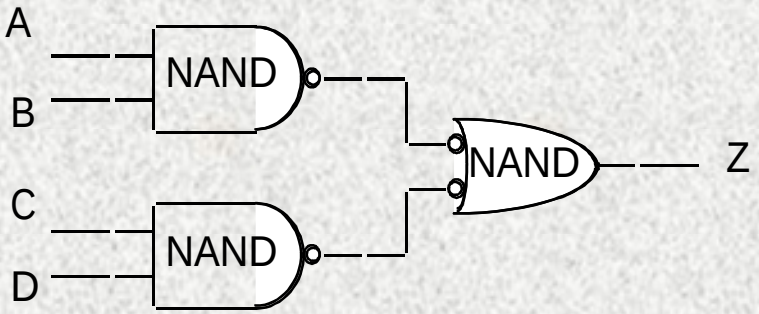
- Convert from networks of ANDs and ORs to networks of NANDs and NORs
  - Introduce appropriate inversions ("bubbles")
- Each introduced "bubble" must be matched by a corresponding "bubble"
  - Conservation of inversions
  - Do not alter logic function
- Example: AND/OR to NAND/NAND



Conversion Between Forms (cont'd)

- Example: verify equivalence of two forms

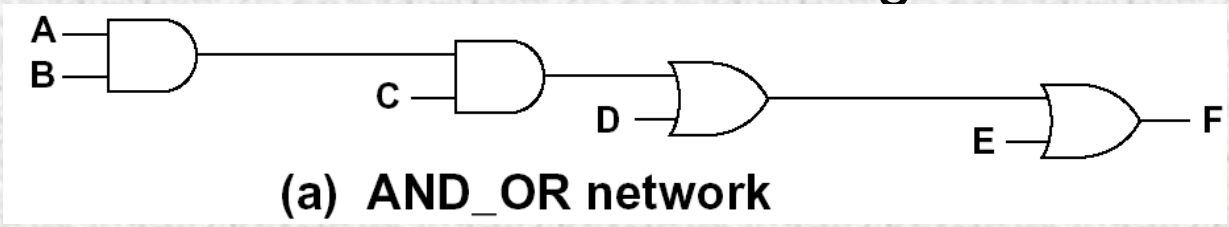




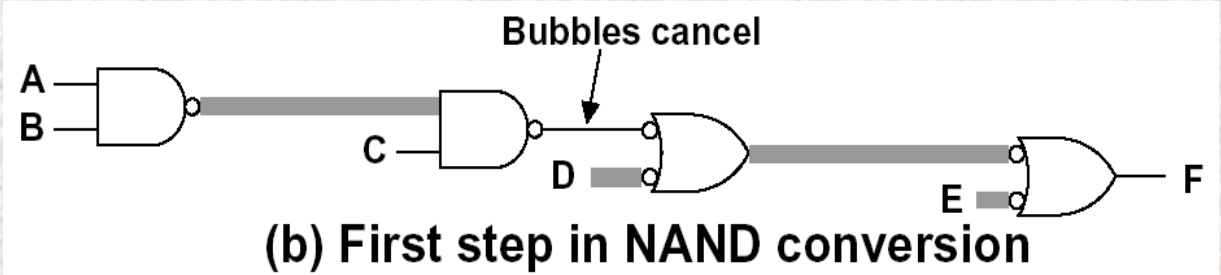
$$Z = [\ (A \cdot B)' \cdot (C \cdot D)'\ ]'$$
$$= [\ (A' + B') \cdot (C' + D')\ ]'$$
$$= [\ (A' + B')' + (C' + D')'\ ]$$
$$= (A \cdot B) + (C \cdot D) \checkmark$$

Conversion to NAND Gates

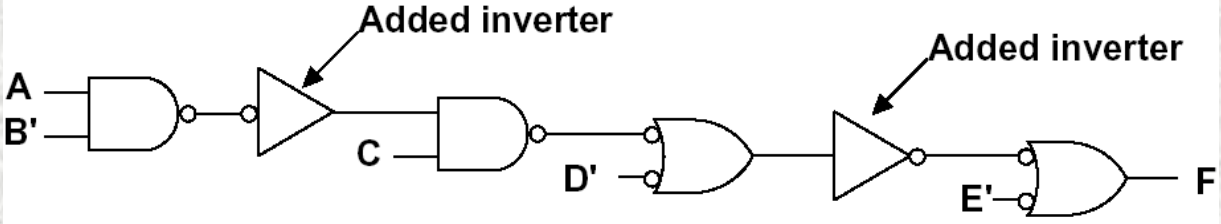
- Start with SOP (Sum of Products)
  - circle 1s in K-maps
- Find network of OR and AND gates



(a) AND_OR network



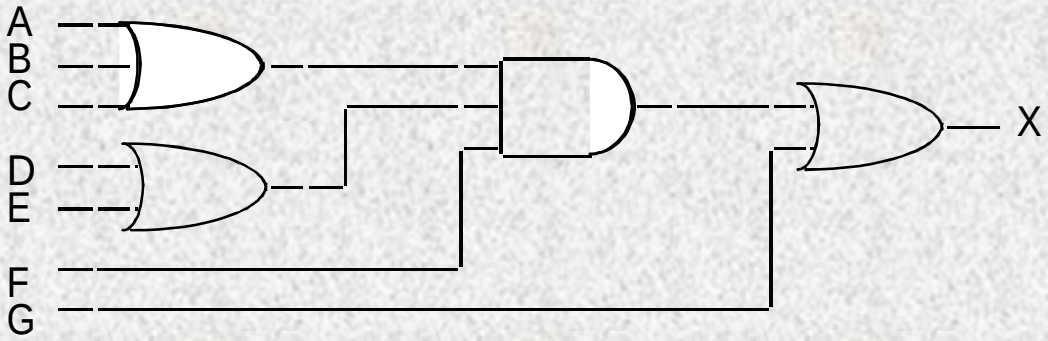
(b) First step in NAND conversion



(c) Completed conversion

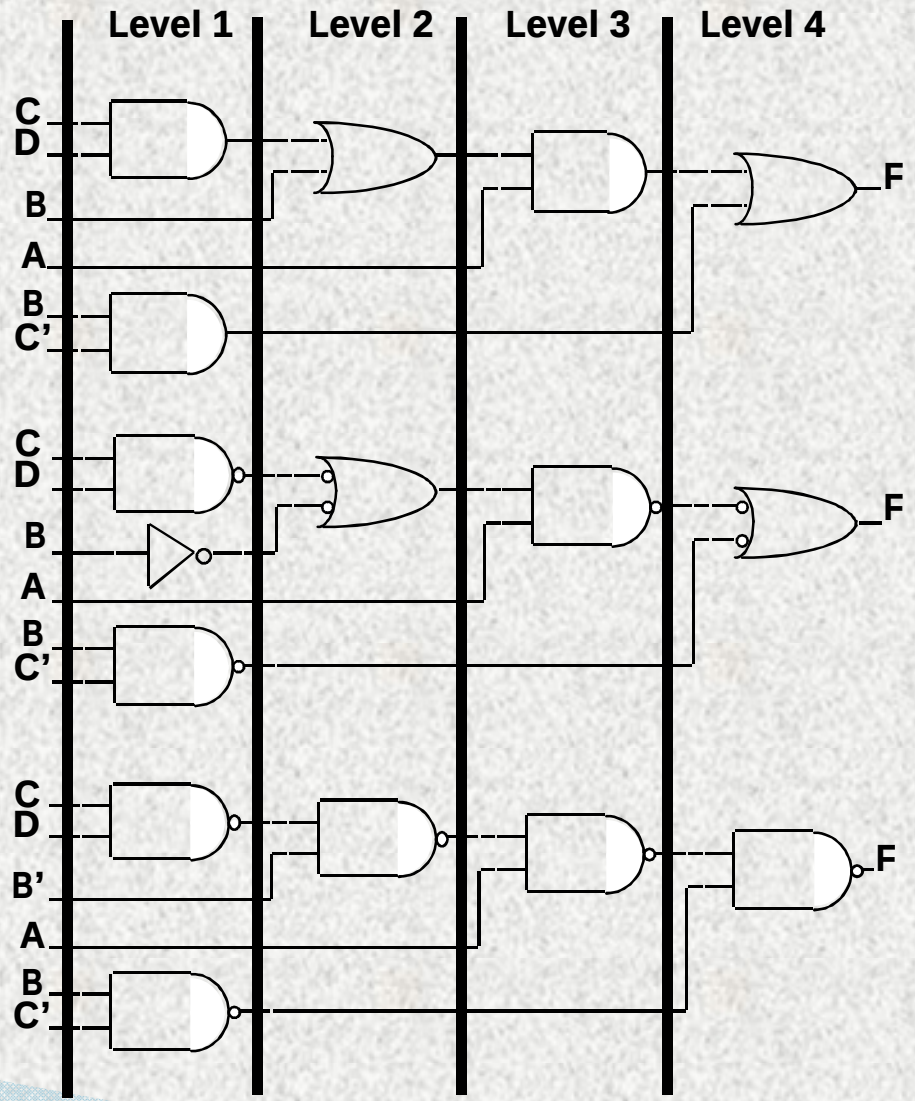
Multi-level Logic

- x = A D F + A E F + B D F + B E F + C D F + C E F + G
  - Reduced sum-of-products form – already simplified
  - 6 x 3-input AND gates + 1 x 7-input OR gate (may not exist!)
  - 25 wires (19 literals plus 6 internal wires)
- x = (A + B + C) (D + E) F + G
  - Factored form – not written as two-level S-o-P
  - 1 x 3-input OR gate, 2 x 2-input OR gates, 1 x 3-input AND gate
  - 10 wires (7 literals plus 3 internal wires)



Conversion of Multi-level Logic to NAND Gates

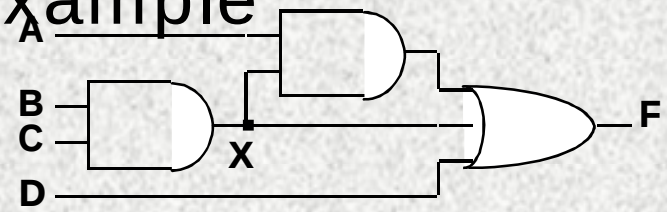
- F = A (B + C D) + B C'



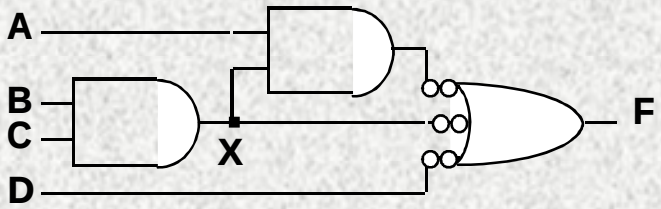
Conversion Between Forms

- Example

(a)



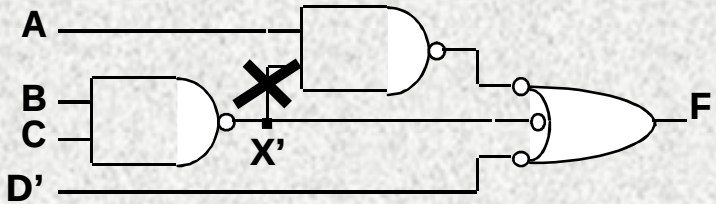
Original circuit



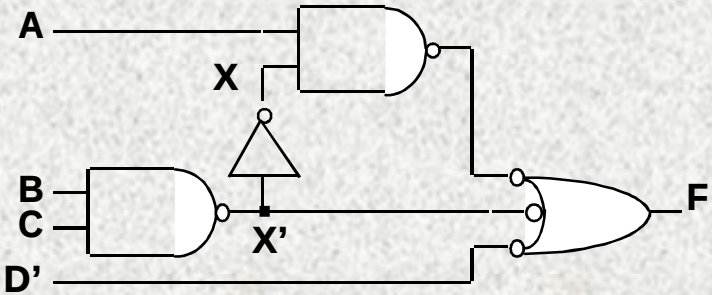
(b)

Add double bubbles at inputs

(c)



Distribute bubbles
some mismatches

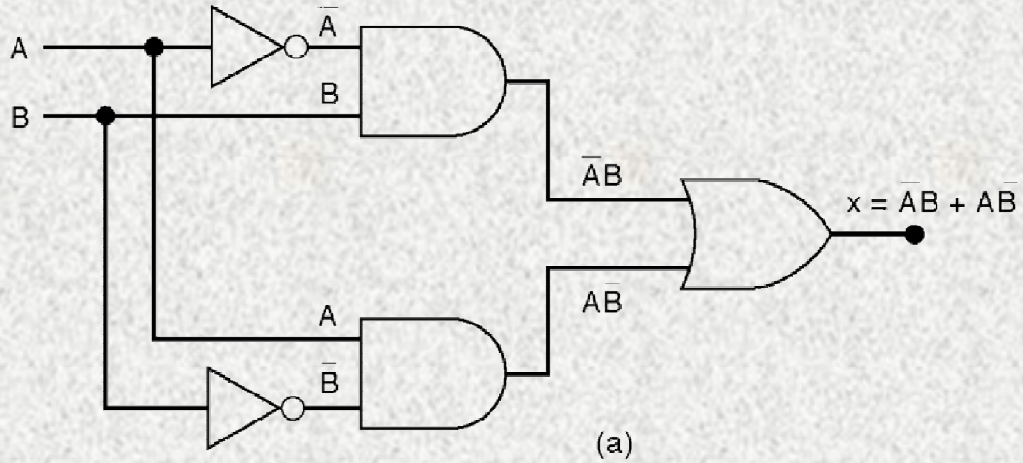


(d)

Insert inverters to fix mismatches

Exclusive-OR and Exclusive-NOR Circuits

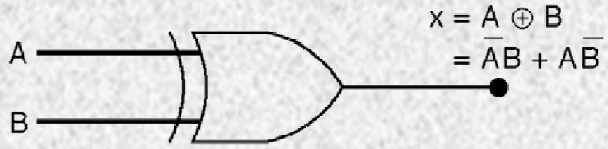
Exclusive-OR (XOR) produces a HIGH output whenever the two inputs are at opposite levels.



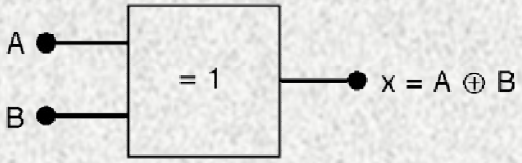
(a)

A	B	x
0	0	0
0	1	1
1	0	1
1	1	0

XOR gate symbols



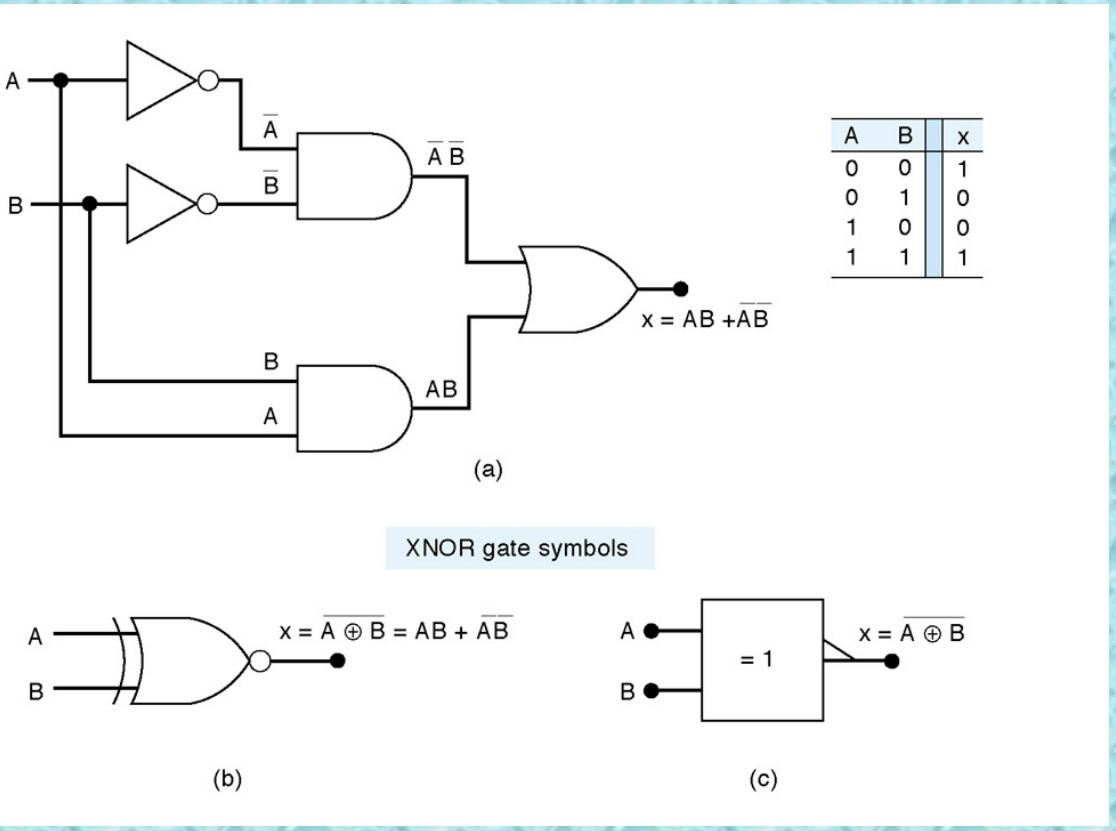
(b)



(c)

Exclusive-NOR Circuits

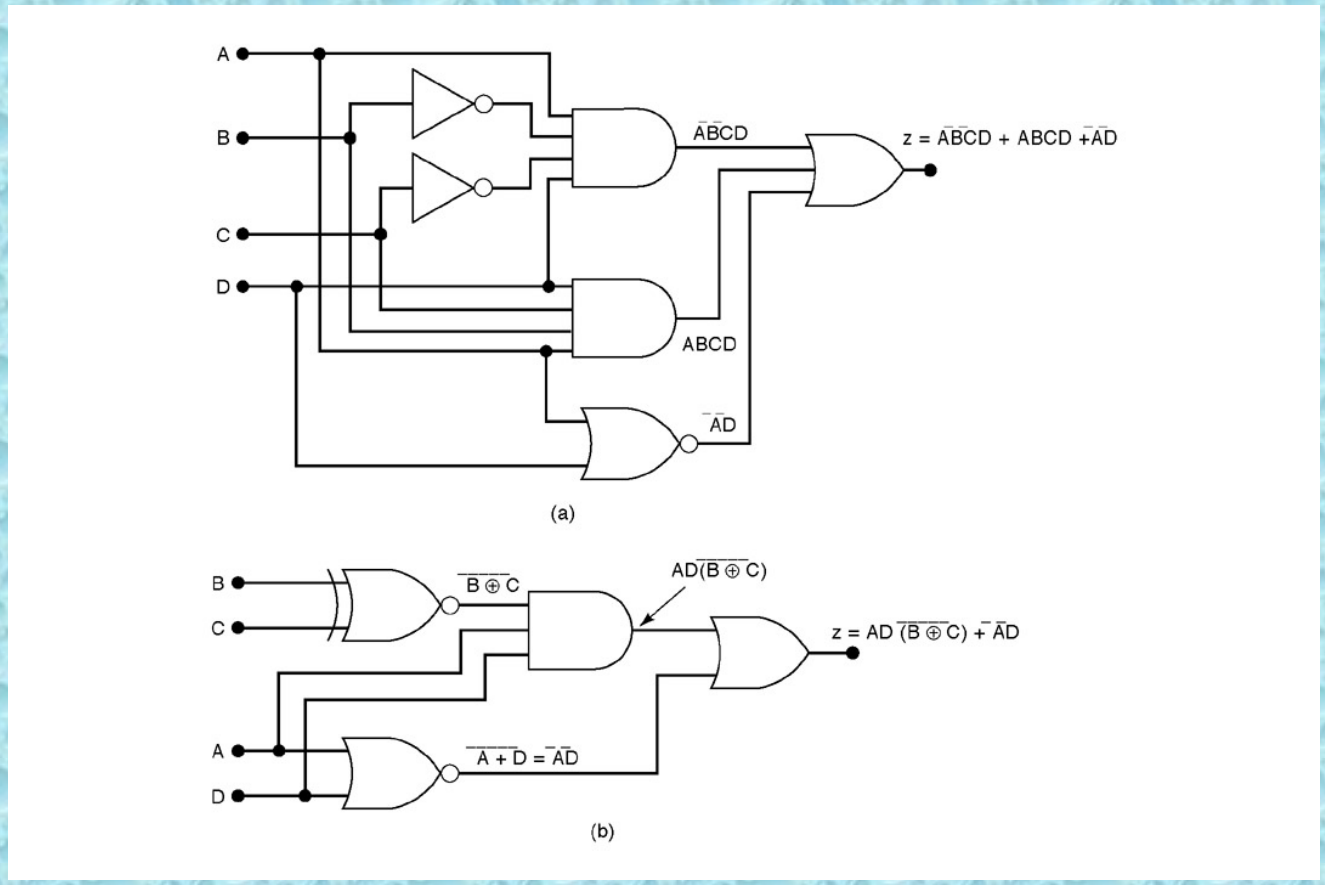
Exclusive-NOR (XNOR) produces a HIGH output whenever the two inputs are at the same level.



A	B	x
0	0	1
0	1	0
1	0	0
1	1	1

Exclusive-NOR Circuits

XNOR gate may be used to simplify circuit implementation.



XOR Function

- XOR function can also be implemented with AND/OR gates (also NANDs).

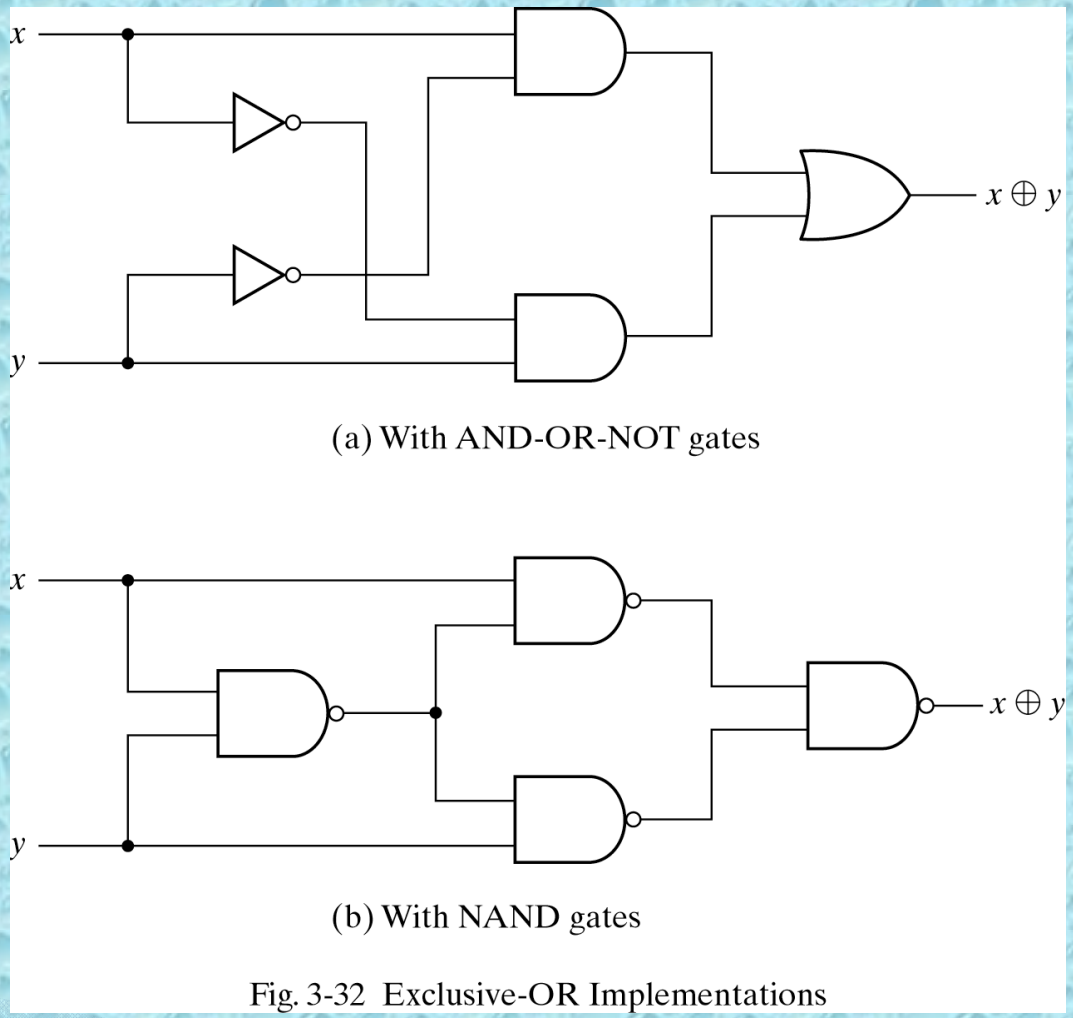


Fig. 3-32 Exclusive-OR Implementations

XOR Function

- Even function – even number of inputs are 1.
- Odd function – odd number of inputs are 1.

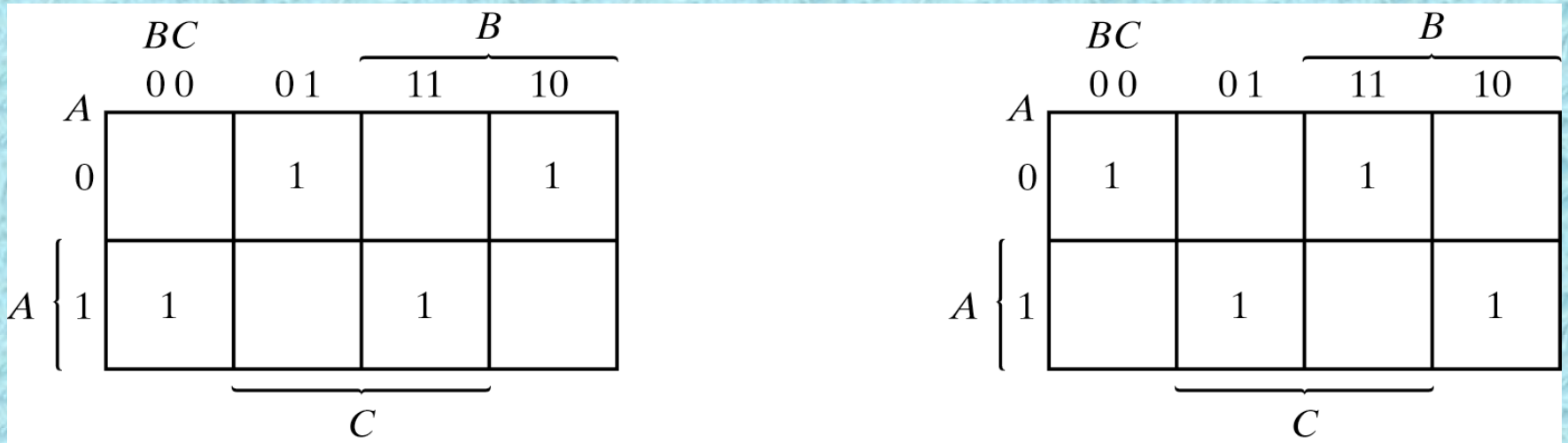
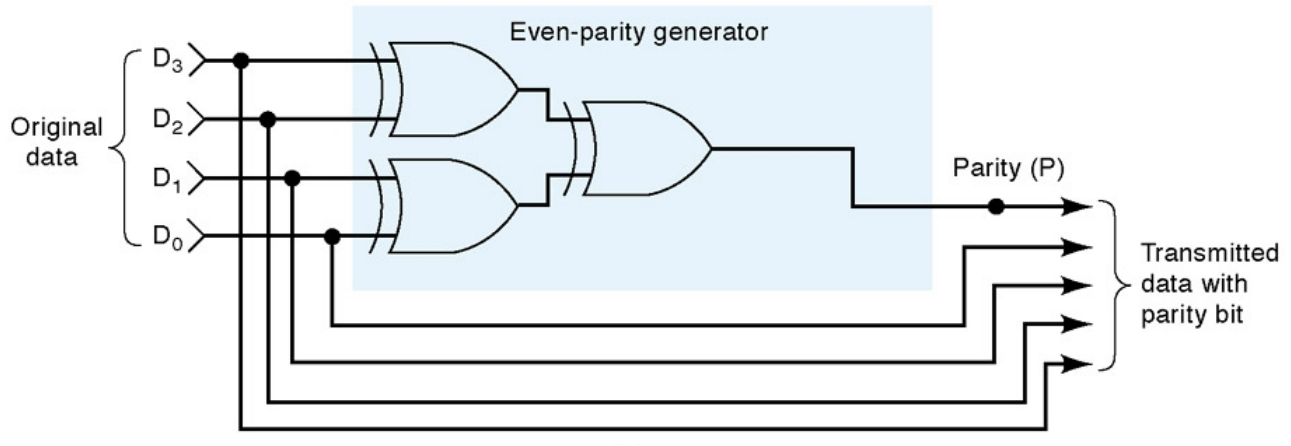
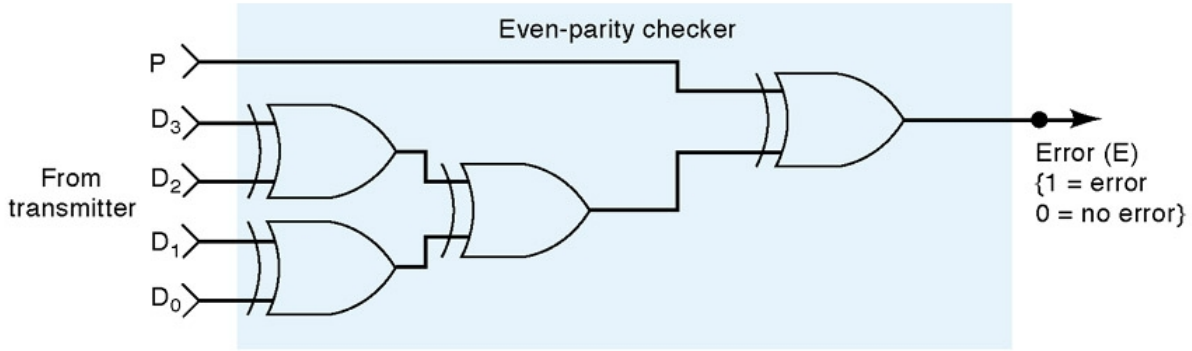


Fig. 3-33 Map for a Three-variable Exclusive-OR Function

Parity Generation and Checking



(a)



(b)

XOR gates used to implement the parity generator and the parity checker for an even-parity system.

Summary

- Follow rules to convert between AND/OR representation and symbols
- Conversions are based on DeMorgan's Law
- NOR gate implementations are also possible
- XORs provide straightforward implementation for some functions
- Used for parity generation and checking
  - XOR circuits could also be implemented using AND/Ors
- Next time: Hazards