DIGITAL ELECTRONICS CIRCUIT(BCA 103)

DEPARTMENT OF COMPUTER SCIENCE

PROGRAMME: BCA

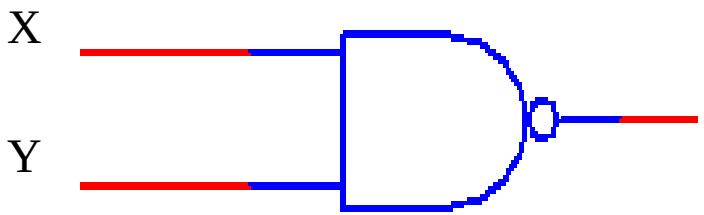
CENTRAL UNIVERSITY OF ODISHA

KORAPUT

NAND AND NOR IMPLEMENTATION

- Digital circuits are frequently constructed with NAND or NOR gates rather than with AND and OR gates.
- The NAND gate is said to be a *universal* gate because any logic circuit can be implemented with it.
- To show that any Boolean function can be implemented with NAND gates, we need only show that the logical operations of AND, OR, and complement can be obtained with NAND gates alone.
- The complement operation is obtained from a one-input NAND gate that behaves exactly like an inverter.
- The AND operation requires two NAND gates.
- **A convenient way to implement a Boolean function with NAND gates is to obtain the simplified Boolean function in terms of Boolean operators and then convert the function to NAND logic.**

NAND Gate

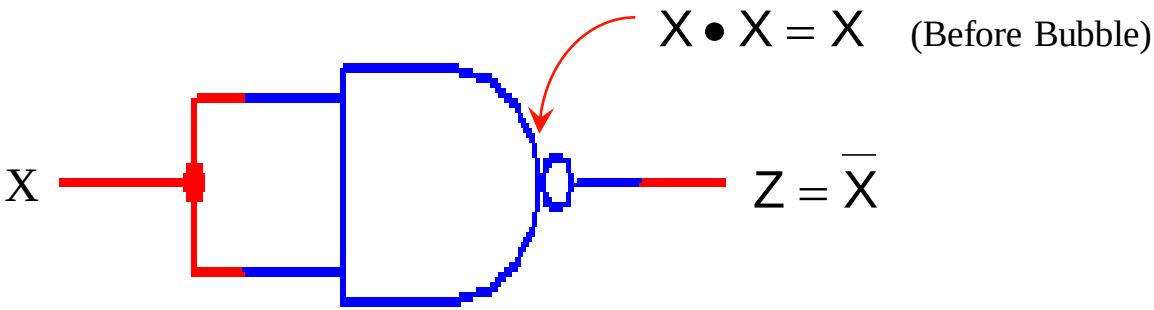


$$Z = \overline{X\,Y} = \overline{X} + \overline{Y}$$

X	Y	Z
0	0	1
0	1	1
1	0	1
1	1	0

NOT Implementation Using NAND

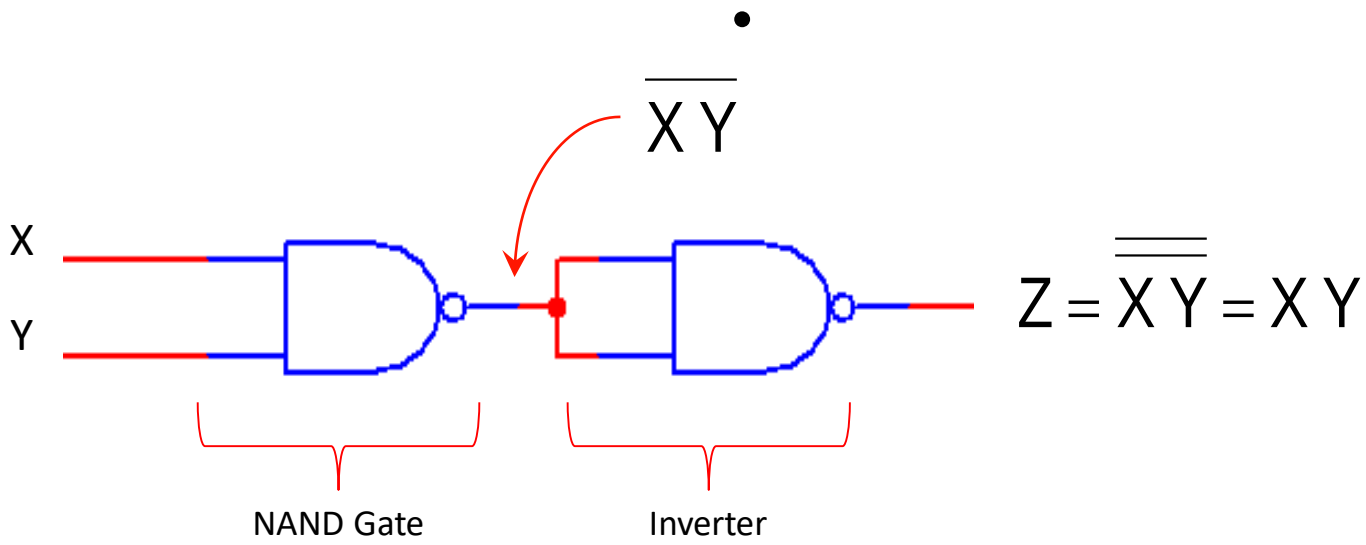
•



X	Z
0	1
1	0

Equivalent to Inverter

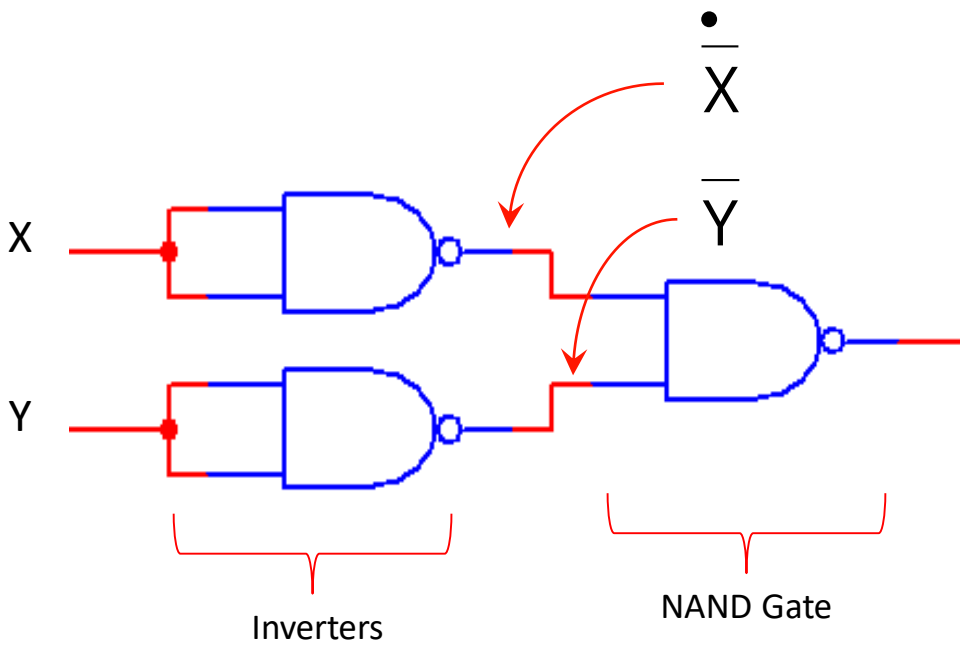
AND Implementation Using NAND



X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

Equivalent to AND Gate

OR Implementation Using NAND



$$Z = \overline{\overline{X}\,\overline{Y}} = \overline{\overline{X}} + \overline{\overline{Y}} = X + Y$$

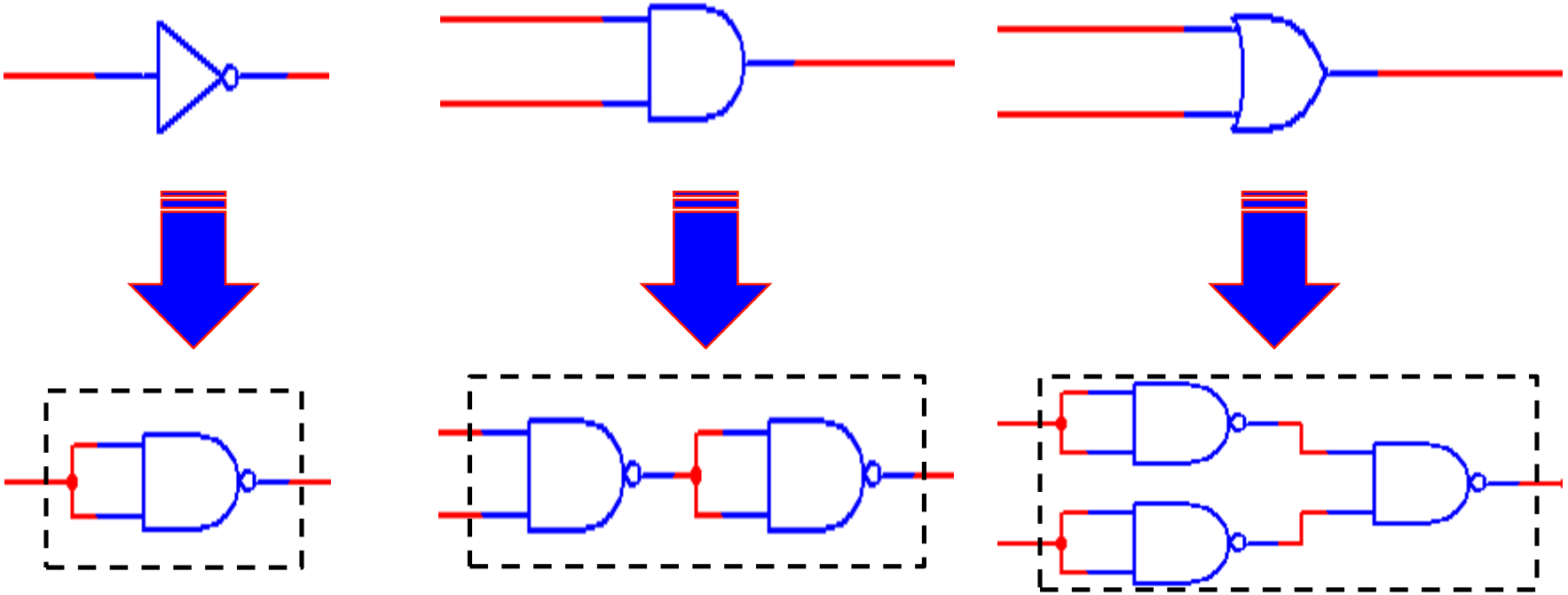
X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

Equivalent to OR Gate

NAND Gate Equivalent to IAO Gates

•

INVERTER AND OR



Two-Level Implementation - NAND

- The implementation of Boolean functions with NAND gates requires that the functions be in sum-of-products form.
- **Process for NAND Implementation:**

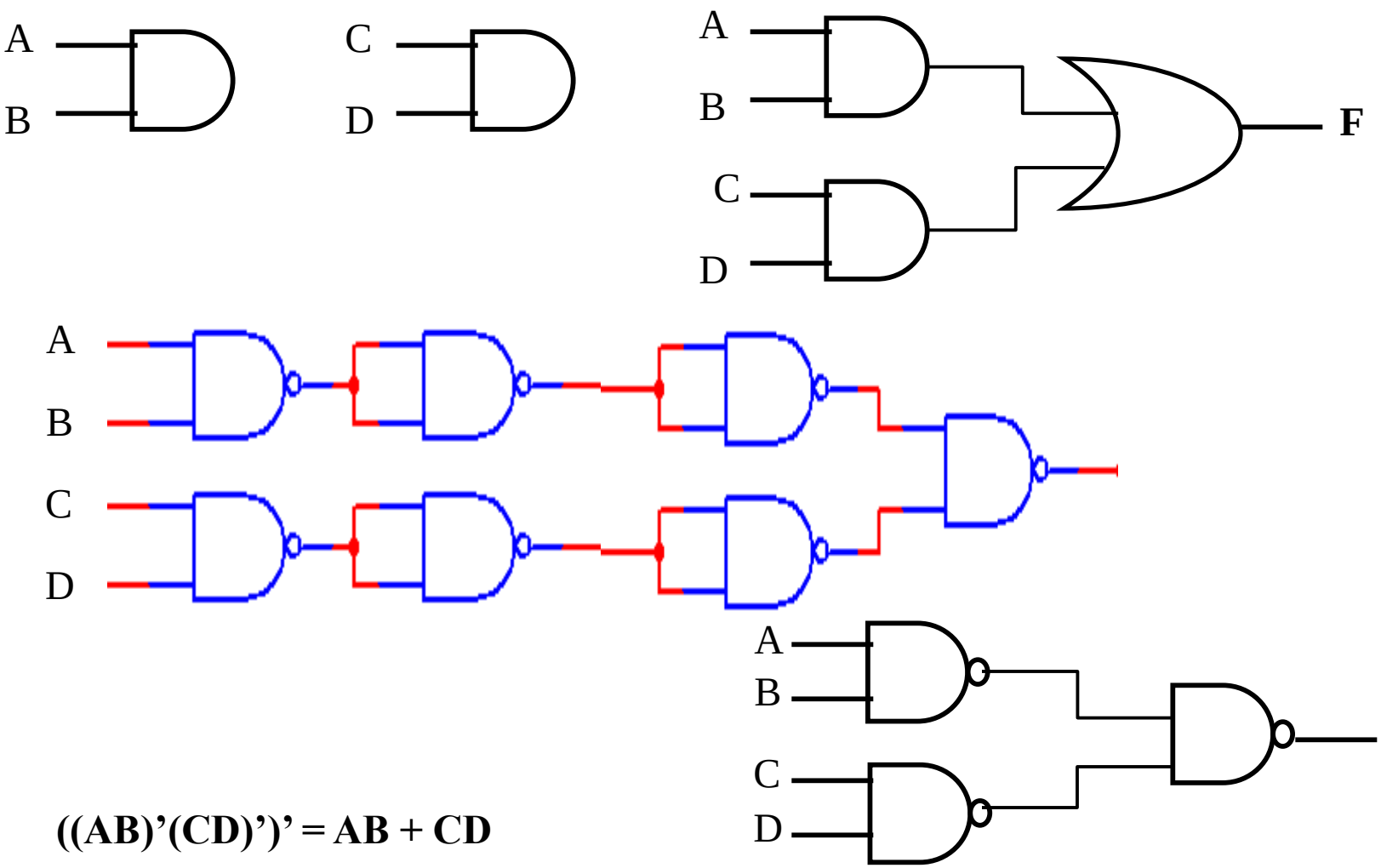
1. If starting from a logic expression, implement the design with AND, OR, NOT logic.
2. In the AND, OR, NOT implementation, identify and replace every AND,OR, and NOT gate with its NAND equivalent.
3. Redraw the circuit.
4. Identify and eliminate any double inversions (i.e., back-to-back inverters).
5. Redraw the final circuit.

Two-Level Implementation - NAND

- The implementation of Boolean functions with NAND gates requires that the functions be in sum-of-products form.
- **Process for NAND Implementation:**

1. Simplify the function and express it in sum-of-products form and implement the design with AND, OR, NOT logic.
2. Draw a NAND gate for each product term of the expression that has at least two literals. The inputs to each NAND gate are the literals of the term. This procedure produces a group of first-level gates. Identify and eliminate any double inversions (i.e., back-to-back inverters).
3. Draw a single gate using the AND-invert or the invert-OR graphic symbol in the second level, with inputs coming from outputs of first-level gates.
4. A term with a single literal requires an inverter in the first level. However, if the single literal is complemented, it can be connected directly to an input of the second-level NAND gate.

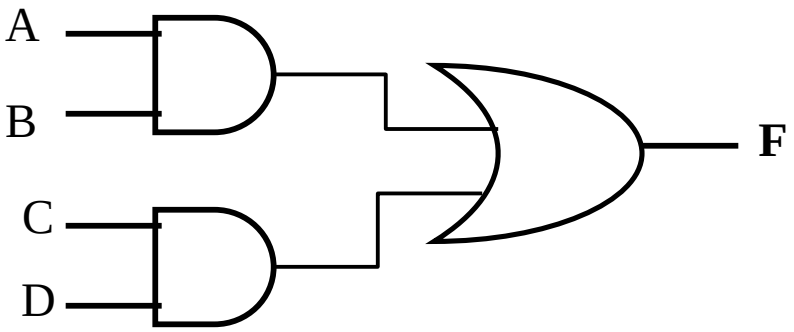
- Ex: **AB + CD**



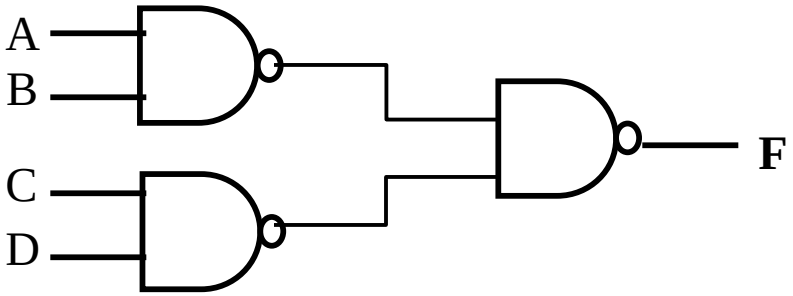
((AB)'(CD)')' = AB + CD

Ex- AB + CD

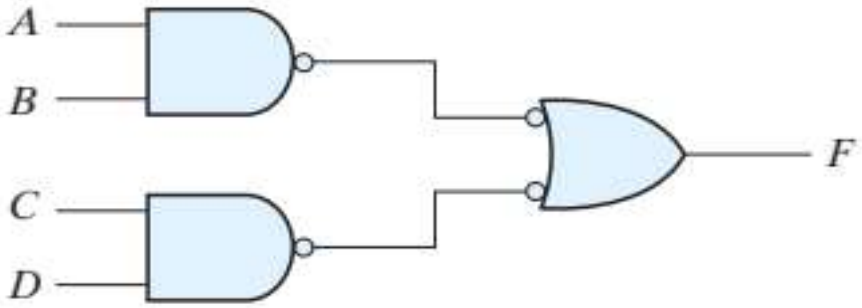
First Way



Second Way



Third Way



Example

Implement the following Boolean function with NAND gates: *F* (*x*, *y*, *z*) = (1, 2, 3, 4, 5, 7)

The first step is to simplify the function into sum-of-products form.

x \ yz	00	01	11	10
0	x'y'z'	x'y'z	x'yz	x'yz'
1	xy'z'	xy'z	xyz	xyz'

x \ yz	00	01	11	10
0	m_0	m_1	m_3	m_2
1	m_4	m_5	m_7	m_6

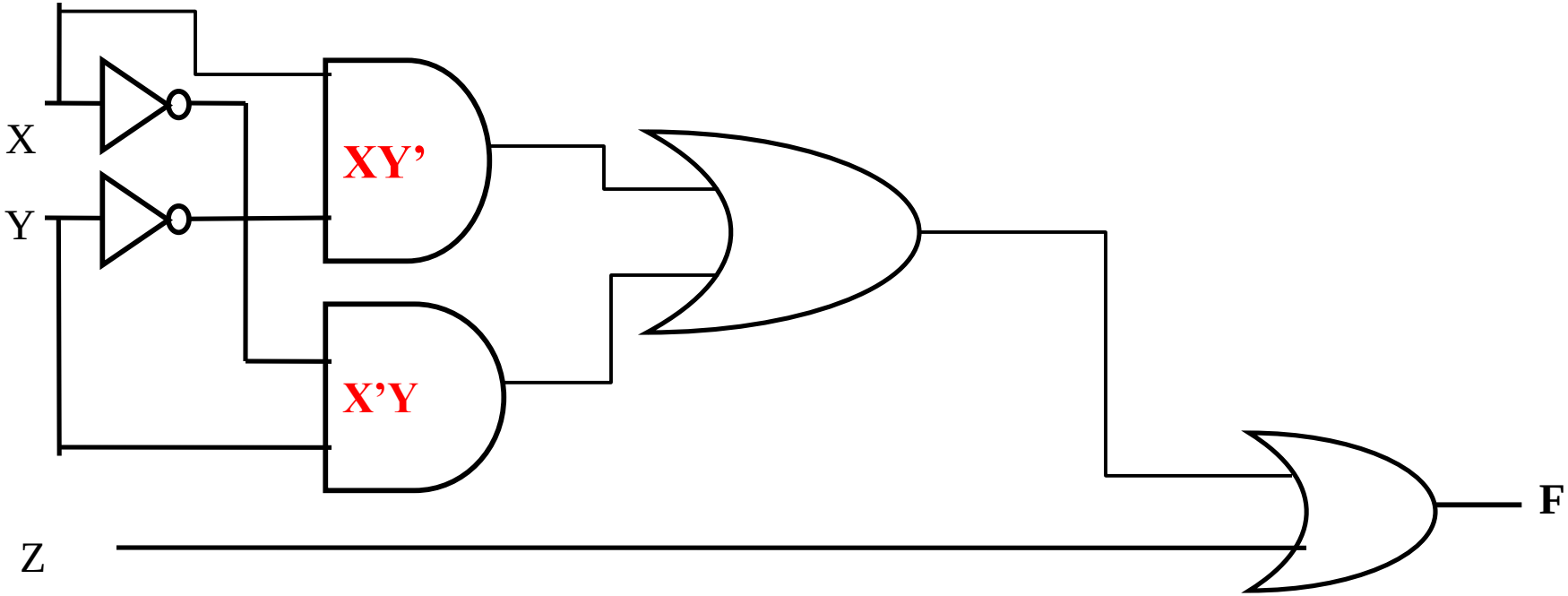
x \ yz	00	01	11	10
0	0	1	1	1
1	1	1	1	0

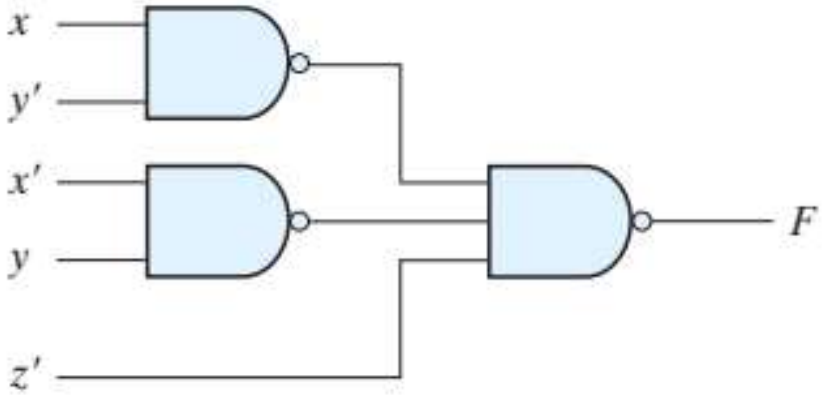
x \ yz	00	01	11	10
0	0	1	1	1
1	1	1	1	0

- **i) x'y'z + x'yz + xy'z + xyz = z ii) x'yz + x'yz' = x'y iii) xy'z' + xy'z = xy'**

  Hence, F = xy' + x'y + z

F = xy’ + x’y + z

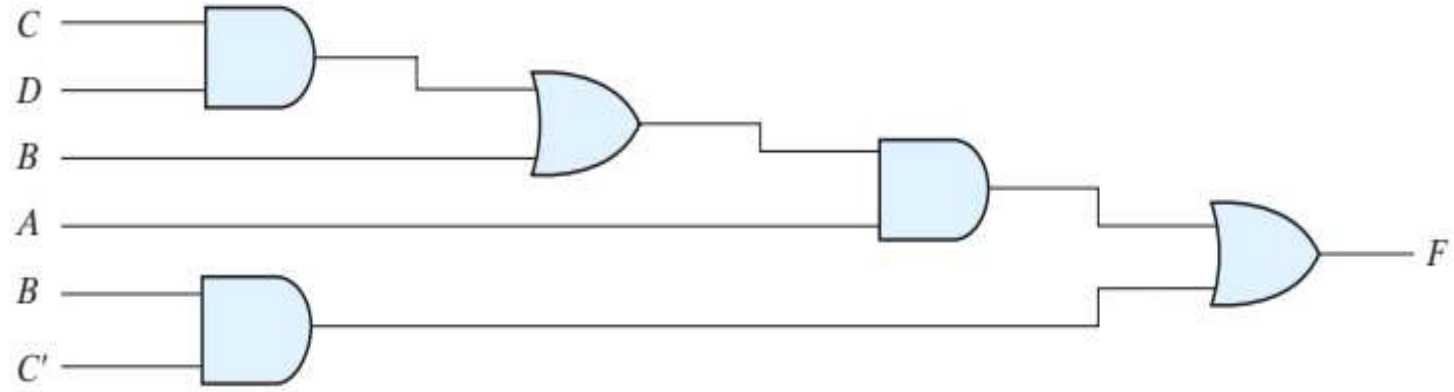




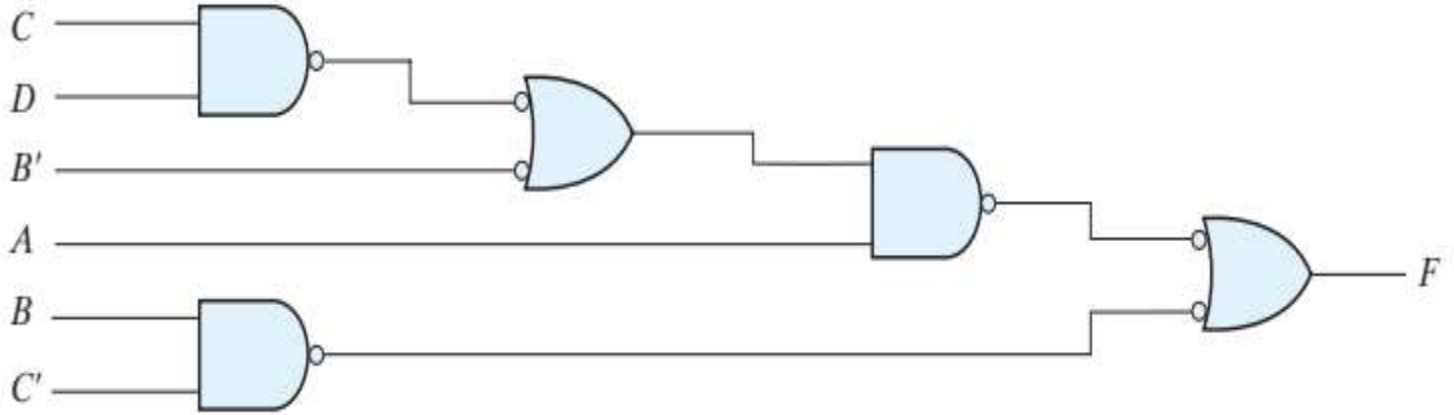
Multilevel NAND Circuits

- The general procedure for converting a multilevel AND–OR diagram into an all-NAND diagram using mixed notation is as follows:
  1. Convert all AND gates to NAND gates with AND-invert graphic symbols.
  2. Convert all OR gates to NAND gates with invert-OR graphic symbols.
  3. Check all the bubbles in the diagram. For every bubble that is not compensated by another small circle along the same line, insert an inverter (a one-input NAND gate) or complement the input literal.

- **Example: F = A (CD + B) + BC’**
-

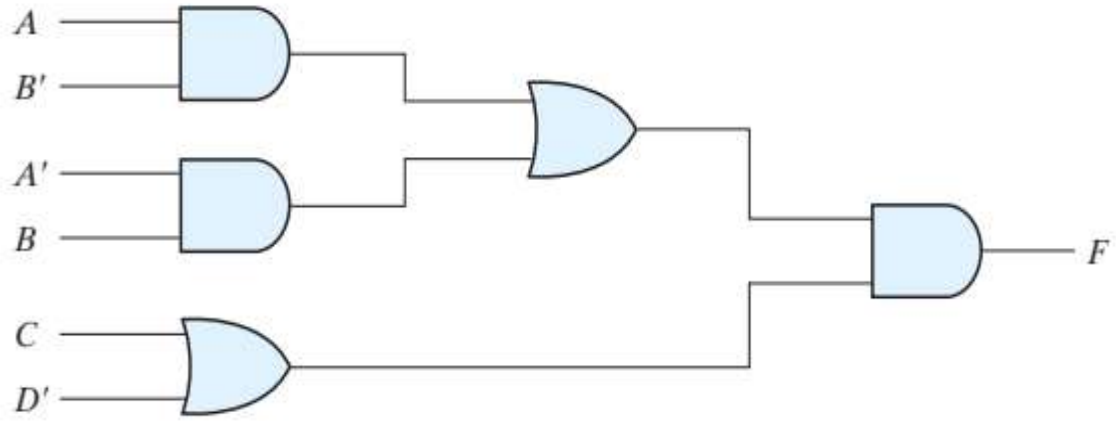


(a) AND–OR gates

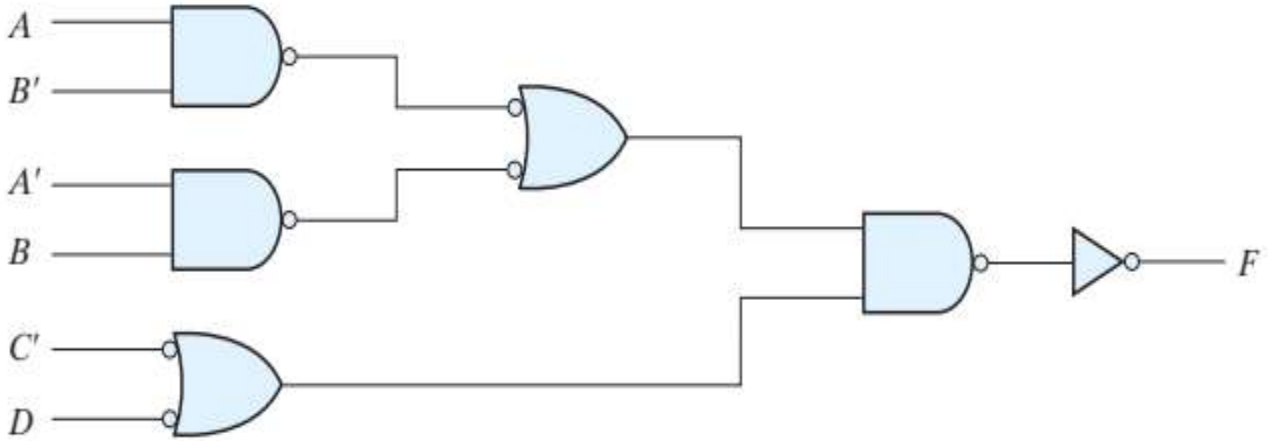


(b) NAND gates

- **Example: $F = (AB' + A'B)(C + D')$**



(a) AND–OR gates

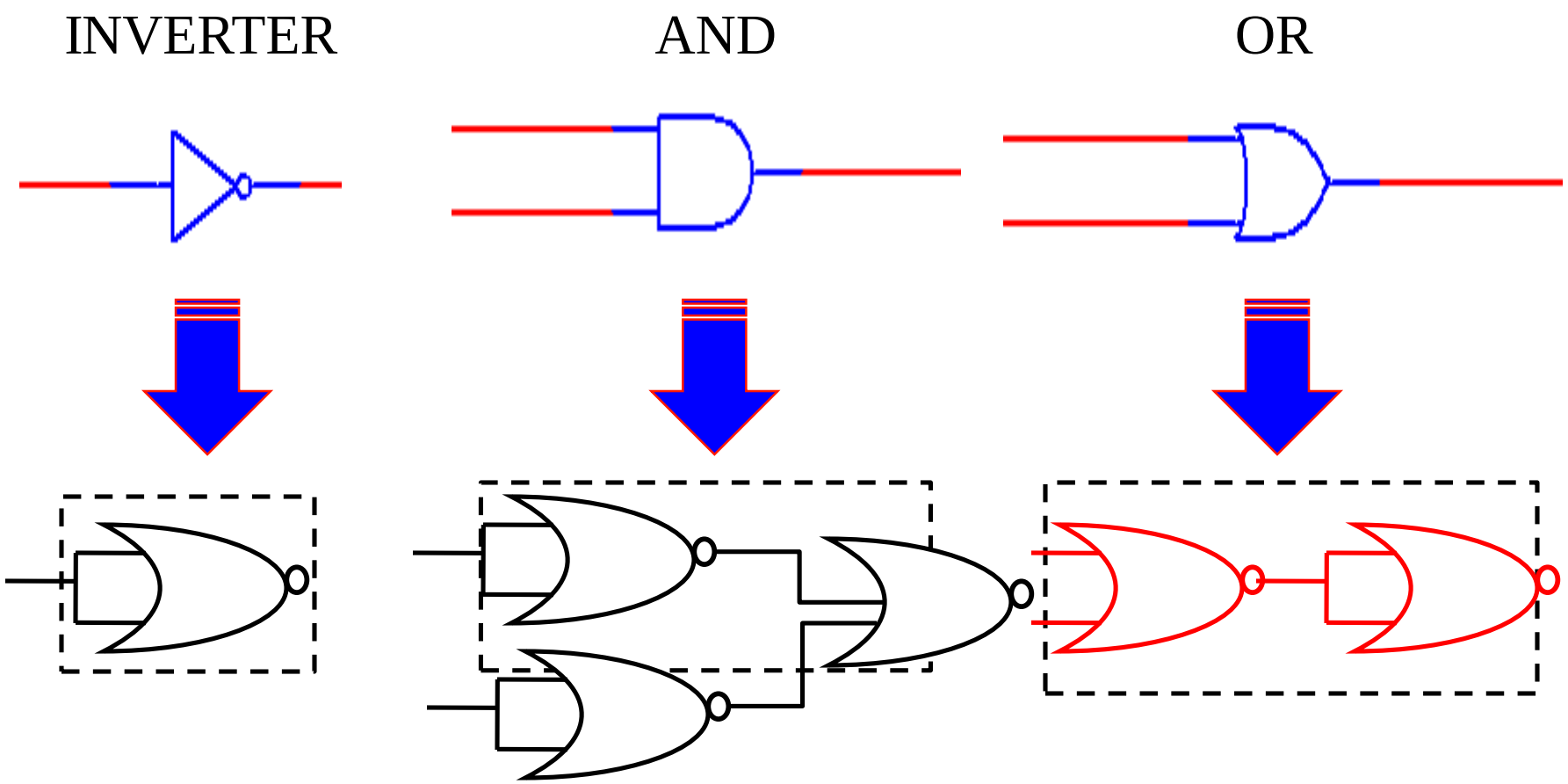


(b) NAND gates

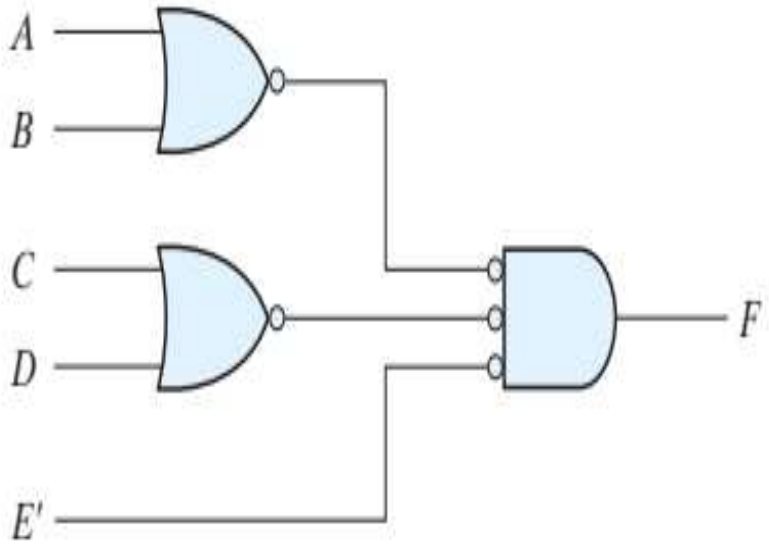
NOR IMPLEMENTATION

- The NOR operation is the dual of the NAND operation.
- Therefore, all procedures and rules for NOR logic are the duals of the corresponding procedures and rules developed for NAND logic.
- The NOR gate is another universal gate that can be used to implement any Boolean function.
- The complement operation requires one input NOR gate that behaves exactly like an inverter.
- The OR operation requires two NOR gates, and the AND operation is obtained with a NOR gate that has inverters in each input.

NOR Gate Equivalent to IAO Gates



- **Ex: $F = (A + B)(C + D)E$**



- **Implementing $F = (AB' + A'B)(C + D')$ with NOR gates**