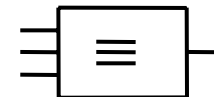
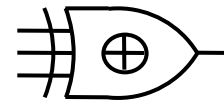
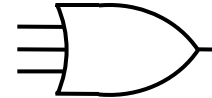
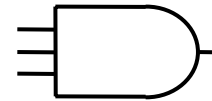


Types of Gates

- Gates already studied

- AND
- OR
- Inverters
- Exclusive-OR
- Equivalence

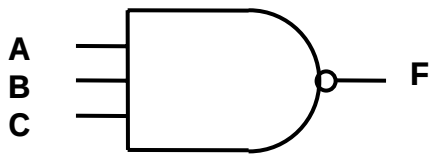


- **NAND and NOR Implementation**

Digital circuits are frequently constructed with NAND or NOR gates rather than with AND or OR gates. NAND and NOR gates are easier to fabricate with electronic components and are the basic gates used in all IC digital logic families

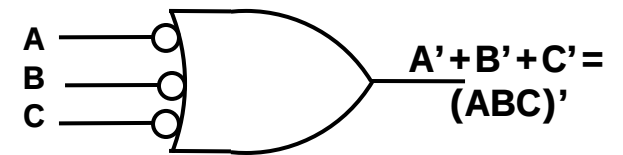
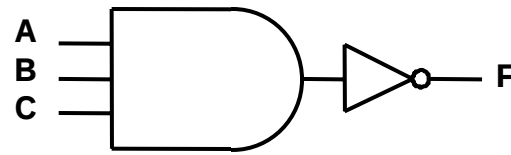
NAND and NOR Gates

- NAND gate is AND gate followed by an inverter



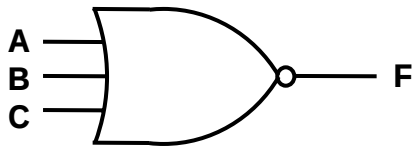
NAND

$$F = (A B C)' = A' + B' + C'$$



Alternate NAND Symbol

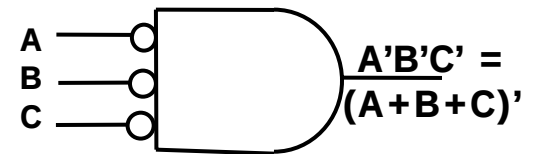
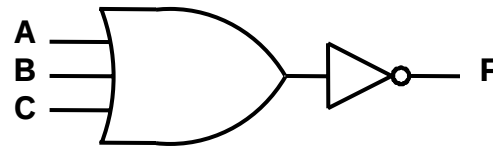
- NOR gate is OR gate followed by an inverter



NOR

$$F = (A + B + C)' = A'B'C'$$

$$F^D = (A'B'C')^D = A' + B' + C'$$

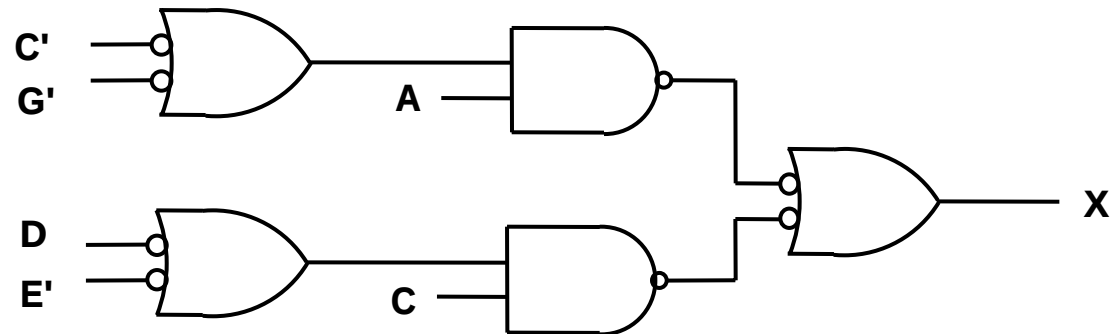
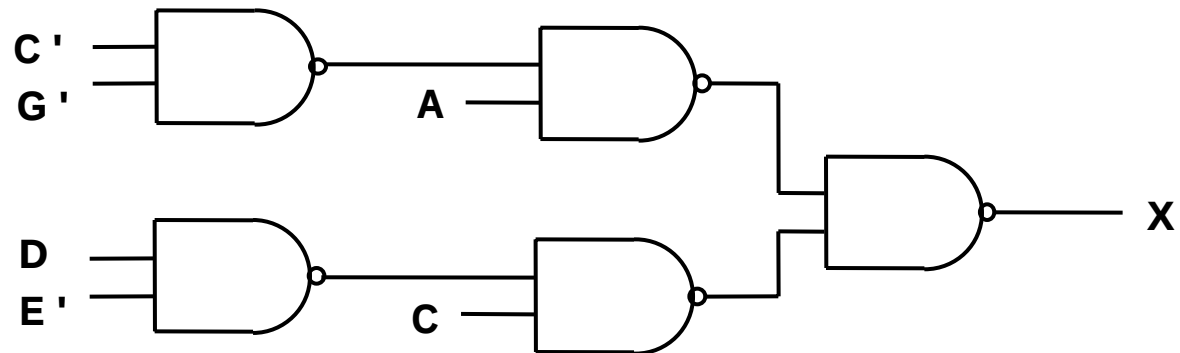


Alternate NOR Symbol

NOR is the dual of NAND

Alternative Gate Symbols

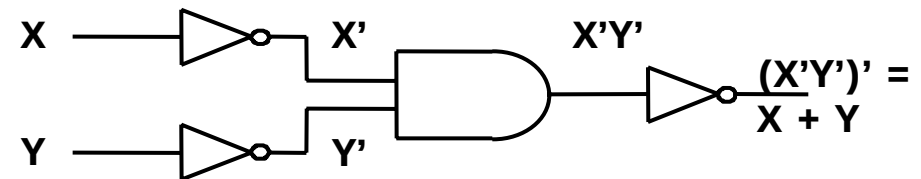
Which is easier to understand?



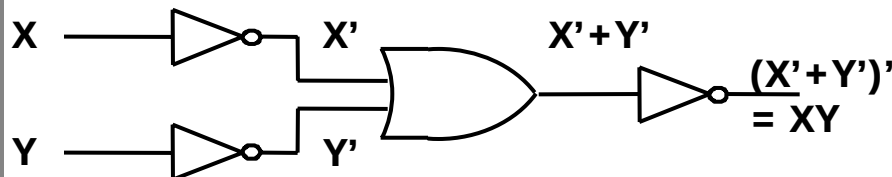
Functionally Complete

- A set of gates is functionally complete if any Boolean expression can be realized with this set of gates
- AND, OR, and inverter is functionally complete
- Any set of gates which can implement AND, OR and inverter is also functionally complete

- Is the set (AND, inverter) functionally complete?
- AND and inverter are implemented.
- What about the OR function ?



- Is the set (OR, inverter) functionally complete?
- OR and inverter are implemented.
- What about the AND function ?

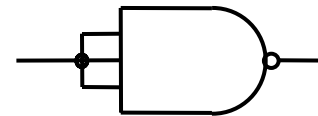


- Is the set (AND, OR) functionally complete?
- AND and OR are implemented.
- What about the inverter function ?
- The set (AND, OR) is not functionally complete

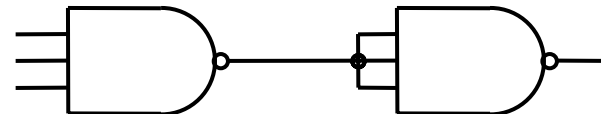
Functionally Complete

- Is the set (NAND) alone functionally complete?

Implement Inverter



Implement AND



Implement OR

Use AND and Invert

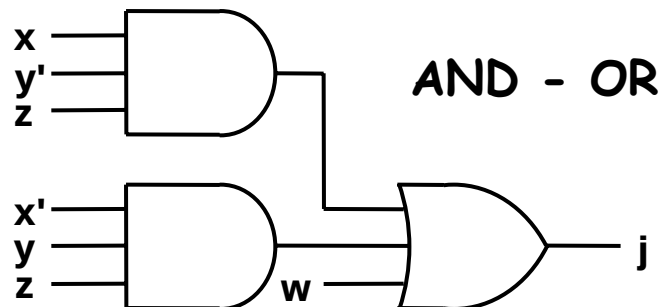
- The set (NAND) alone is functionally complete
- In a similar way, (NOR) alone is functionally complete

Two Level Networks

NAND – NAND Implementation

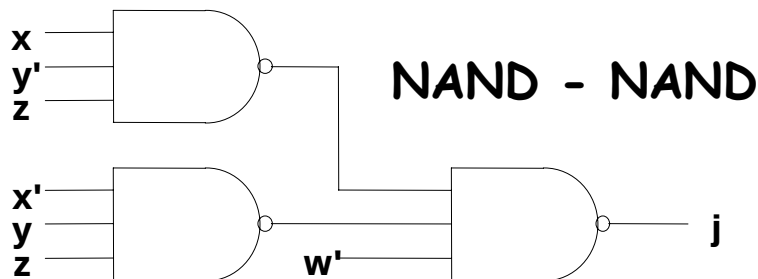
Implement:

$$j = xy'z + x'yz + w$$



$$j = [((xy'z) + (x'yz) + w)']'$$

$$j = [(xy'z)'(x'yz)'w']'$$



- Minimum 2-Level NAND-NAND Network
- NAND - NAND is one of the most commonly available types of IC
- Find the minimum sum of products for a function F
- Draw the corresponding 2-level AND-OR network
- Replace all gates with NAND gates
- Invert any inputs to level - 1 gates

Example

Realize F in a minimum 2-level NAND-NAND network

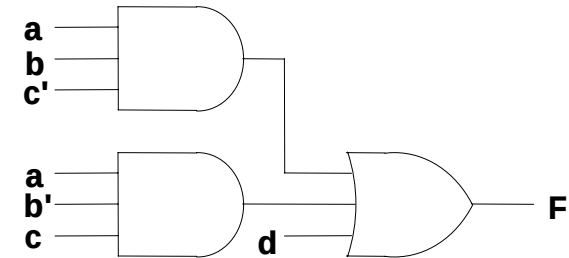
$$F(a, b, c, d) = \sum m(3, 5, 7, 9, 10, 12, 13) \\ + \sum d(0, 1, 6, 11, 15)$$

Find the essential prime implicants

a b \ c d		a b			
		00	01	11	10
00	c d	X		1	
01	c d	X	1	1	1
11	c d	1	1	X	X
10	c d		X		1

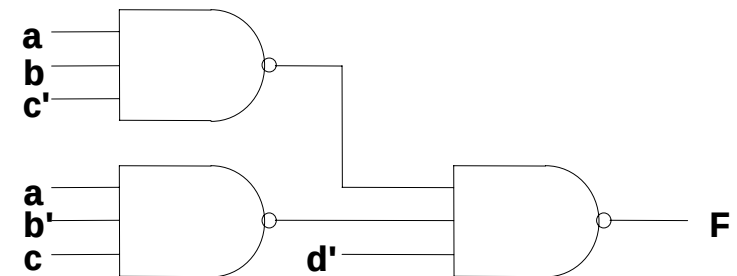
$$F = d + abc' + ab'c$$

Draw the Network



Convert all gates to NAND

Invert Level-1 inputs



Two Level Networks

NOR - OR & OR - NAND Implementation

Implement:

$$j = xy'z + x'yz + w$$

$$j = [(xy'z)']' + [(x'yz)']' + w$$

$$j = (x' + y + z')' + (x + y' + z')' + w$$

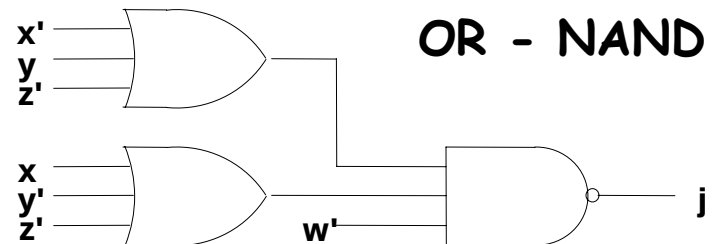
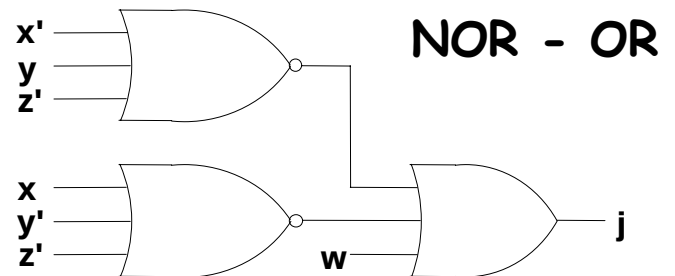
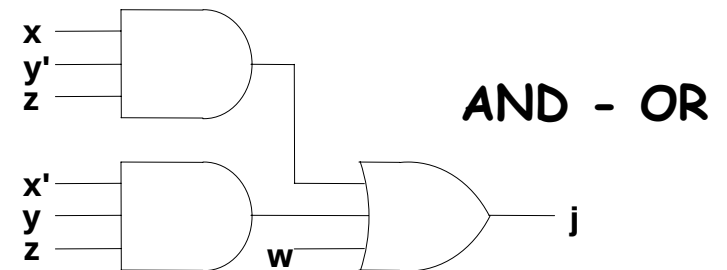
1. Convert level-2 AND gates to NOR
2. Invert inputs to level-2 NOR gates

$$j = [((xy'z) + (x'yz) + w)']'$$

$$j = [(xy'z)'(x'yz)'w']'$$

$$j = [(x' + y + z')(x + y' + z')w']'$$

1. Convert level-1 gates to NAND
2. Convert level-2 gates to OR
3. Invert all inputs to all gates



Two Level Networks

Product of Sums

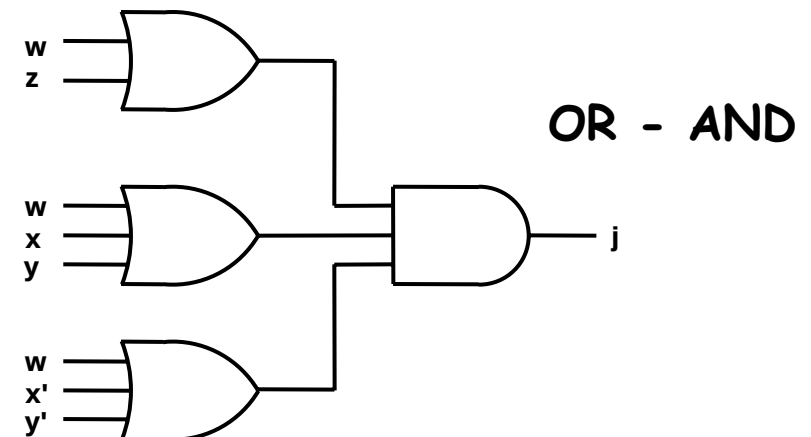
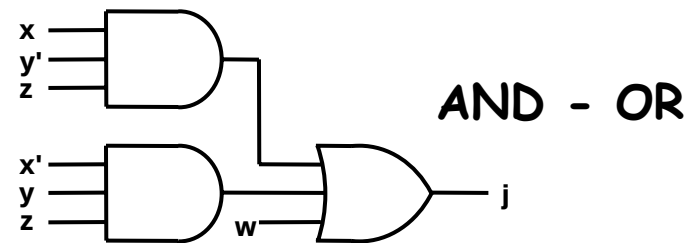
Convert: $j = xy'z + x'yz + w$

$j' = w'z' + w'x'y' + w'xy$

- Convert from sum of products to product of sums form
 - Algebraically
 - Using a Karnaugh Map

		w x			
		00	01	11	10
y z	00	0	0	1	1
	01	0	1	1	1
	11	1	0	1	1
	10	0	0	1	1

$j = (w + z)(w + x + y)(w + x' + y')$

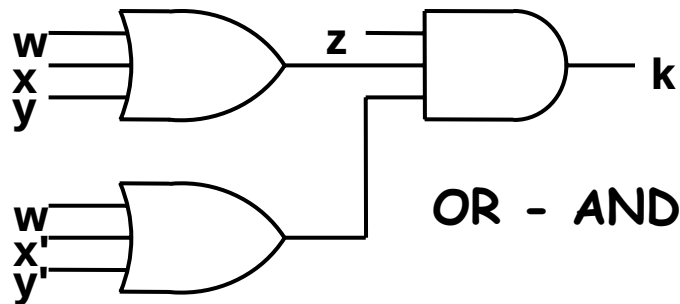


Two Level Networks

NOR – NOR Implementation

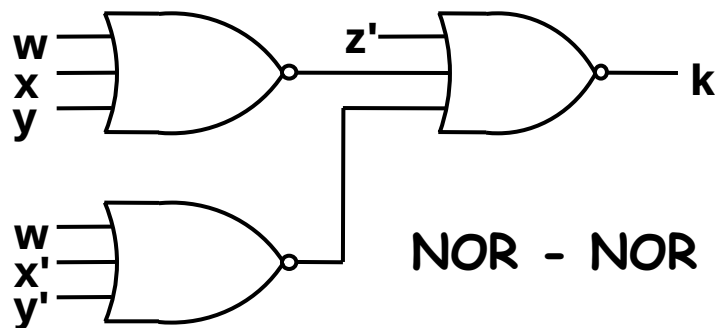
Implement:

$$k = z (w+x+y) (w+x'+y')$$



$$k = [(z(w+x+y) (w+x'+y'))']'$$

$$k = [z' + (w+x+y)' + (w+x'+y')']'$$



- Minimum 2-Level NOR-NOR Network
- NOR - NOR is also a very commonly available type of IC gate
- Find the minimum product of sums for a function F
- Draw the corresponding 2-level OR-AND network
- Replace all gates with NOR gates
- Invert any inputs to level - 1 gates

Example

Realize F in a minimum 2-level
NOR-NOR network

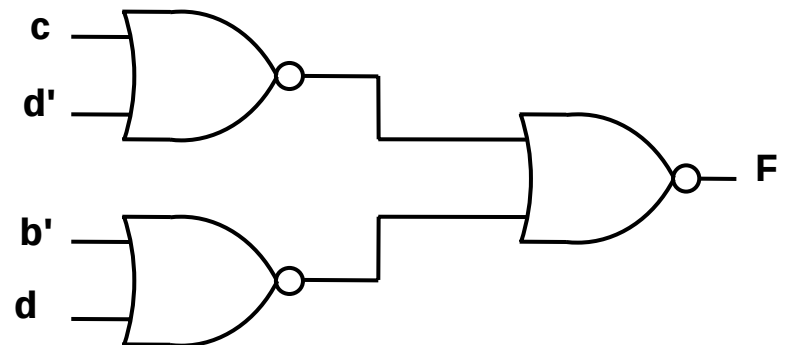
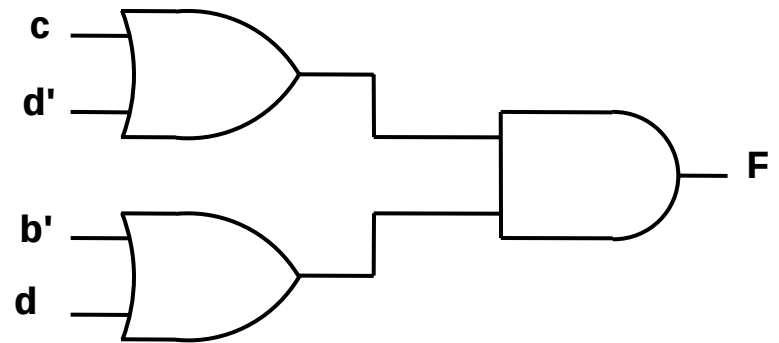
$$F(a, b, c, d) = b'd' + cd$$

		a b			
		00	01	11	10
c d	00	1	0	0	1
	01	0	0	0	0
	11	1	1	1	1
	10	1	0	0	1

Find F' in sum of products form

$$F' = c'd + bd'$$

$$F = (c + d')(b' + d)$$



Two Level Networks

NAND - AND & AND - NOR Implementation

Implement:

$$k = z (w+x+y) (w+x'+y')$$

$$k = z[(w+x+y)']'[(w+x'+y')']'$$

$$k = z (w'x'y')' (w'xy)'$$

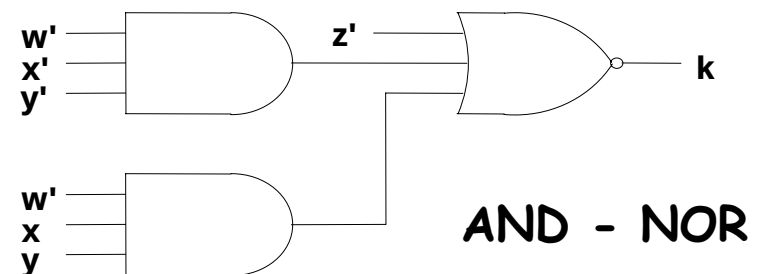
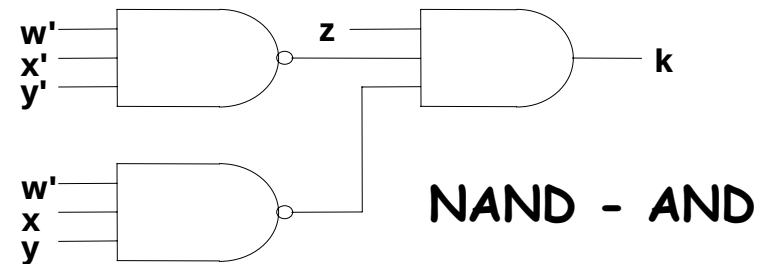
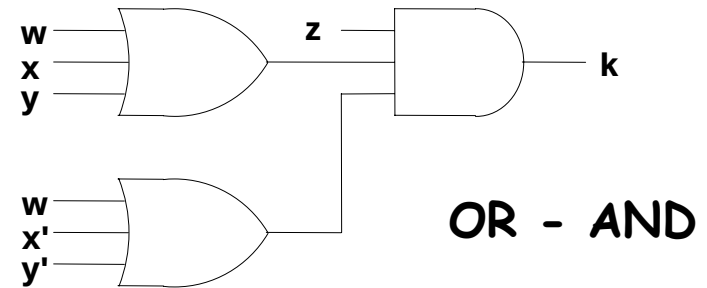
1. Convert level-2 OR gates to NAND
2. Invert inputs to level-2 NAND gates

$$k = [(z(w+x+y) (w+x'+y'))']'$$

$$k = [z' + (w+x+y)' + (w+x'+y')']'$$

$$k = [z' + w'x'y' + w'xy]'$$

1. Convert level-1 gates to NOR
2. Convert level-2 gates to AND
3. Invert all inputs to all gates



Two Level Networks

	Level - 1			
Level - 2	AND	NOR	NAND	OR
AND	no	OK	no	OK
NOR	no	OK	no	OK
NAND	OK	no	OK	no
OR	OK	no	OK	no

no means the 2-level network is not functionally complete

Sum of products forms

Product of sums forms

1. Convert level-1 gates to NOR
2. Convert level-2 gates to AND
3. Invert all inputs to all gates

		Level - 1			
Level - 2		AND	NOR	NAND	OR
AND	no		OK	no	OK
NOR	no		OK	no	OK
NAND	OK	no		OK	no
OR	OK	no		OK	no

1. Convert all gates to NOR
2. Invert inputs to level-1 NOR gates

1. Convert level-2 OR gates to NAND
2. Invert inputs to level-2 NAND gates

1. Convert level-2 AND gates to NOR
2. Invert inputs to level-2 NOR gates

1. Convert all gates to NAND
2. Invert inputs to level-1 NAND gates

1. Convert level-1 gates to NAND
2. Convert level-2 gates to OR
3. Invert all inputs to all gates

Wired Logic

- Wired logic implements Boolean functions in other two-level forms. Examples are open collector TTL NAND gates and ECL NOR gates. The logic performed by the circuit in (a) is given by

$$F = (AB)' \cdot (CD)' = (AB + CD)'$$

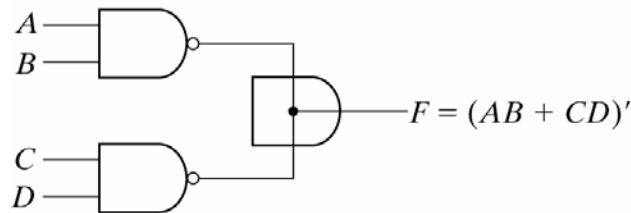
and is called an AND-OR-INVERT function.

- The two forms NAND-AND and AND-NOR are equivalent forms and can perform the AND-OR-INVERT function
- Similarly, the NOR output of ECL gates can be tied together to perform a wired-OR function. The logic performed by the circuit in (b) is given by

$$F = (A + B)' + (C + D)' = [(A + B)(C + D)]'$$

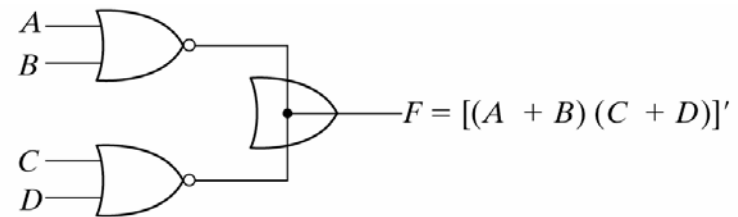
and is called an OR-AND-INVERT function.

- The OR-NAND and NOR-OR forms perform the OR-AND-INVERT function.



(a) Wired-AND in open-collector
TTL NAND gates.

(AND-OR-INVERT)



(b) Wired-OR in ECL gates

(OR-AND-INVERT)