The material covered in this class will be as follows:

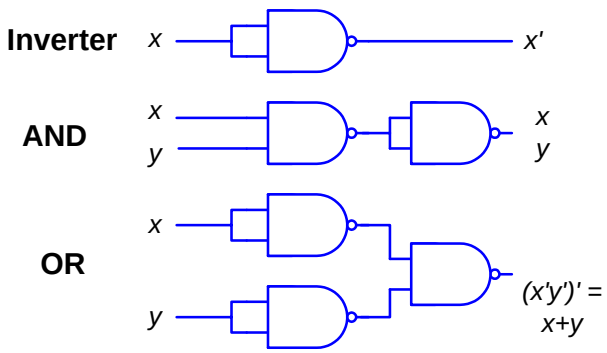
- ⇒ NAND and NOR Implementation
- ⇒Two level implementation
- ⇒Multilevel NAND circuits
- ⇒Two level and multilevel NOR implementation
- ⇒Other two-level implementations

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.

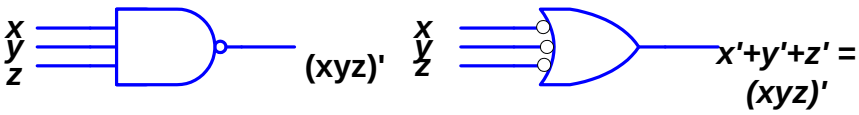
NAND Circuits

The basic AND, OR, and NOT gates can be implemented using NAND gates only.



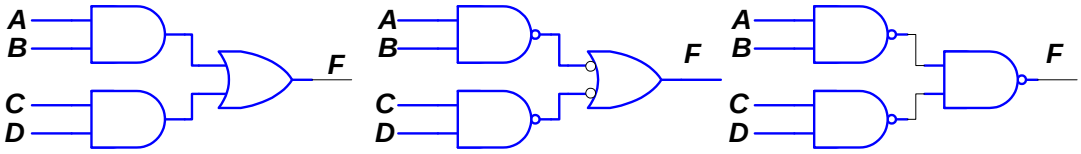
Logic Operations with NAND gates

Two equivalent graphic symbols for NAND gate are shown below:



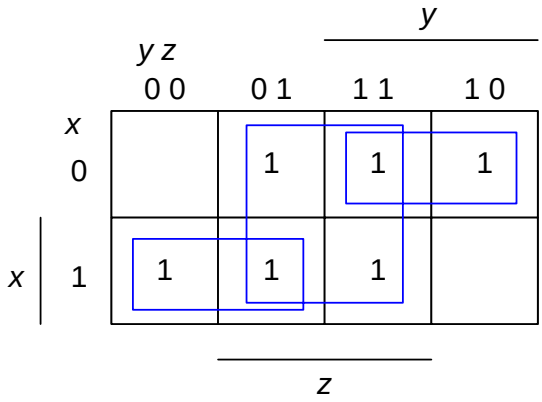
Two Graphic Symbols for NAND gate

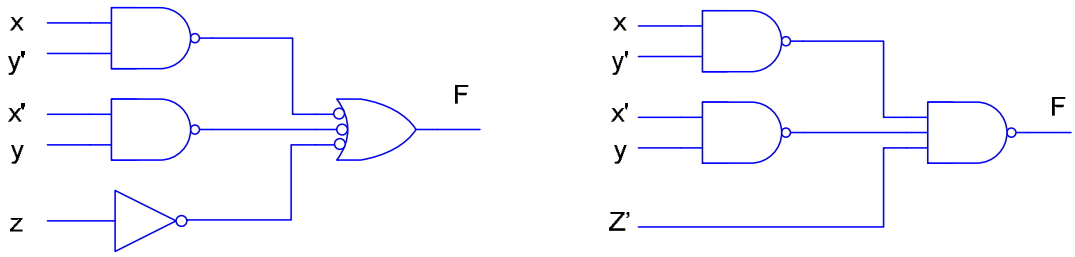
Example 1: Implement F = AB + CD using NAND gates



Example 2: Implement $F = \sum(1,2,3,4,5,7)$ using NAND gates

Simplification of the function gives F = *xy'+ x'y + z*

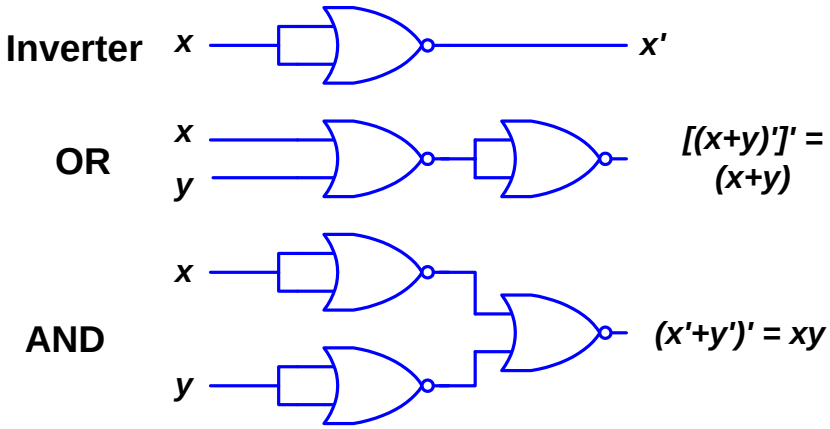




Two-level NAND implementation for F = xy +'x'y + z

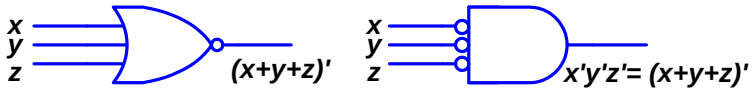
NOR Implementation

The basic AND, OR, and NOT gates can be implemented using NOR gates only:



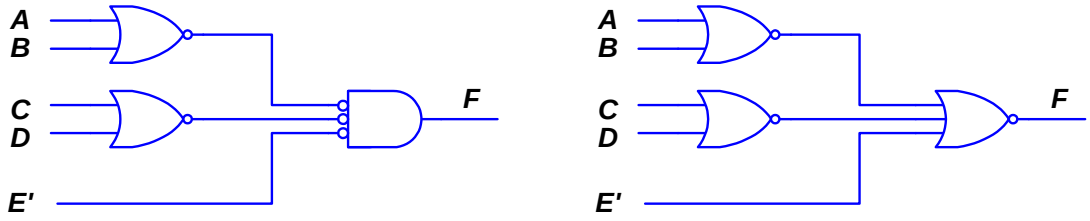
Logic Operations with NOR gates

Two equivalent graphic symbols for the NOR gate, are shown below:



Two graphic symbols for NOR gate

Example: Implement $F = (A+B)(C+D)E$ using NOR gates.



OTHER TWO-LEVEL IMPLEMENTATIONS

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

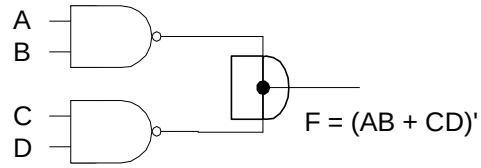
$$F = (AB)'.(CD)' = (AB + CD)'$$

and is called an 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

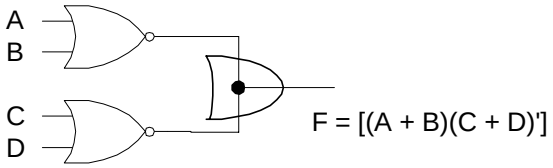
$$F = (A + B)' + (C + D)' = [(A + B)(C + D)]'$$

and is called an OR-AND-INVERT function.



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

(AND-OR_INVERT)



(b) Wired-OR in ECL gates

(OR-AND-INVERT)

There are 16 possible arrangements of two level implementations. Eight are degenerate and will not be considered.

Nondegenerate Forms

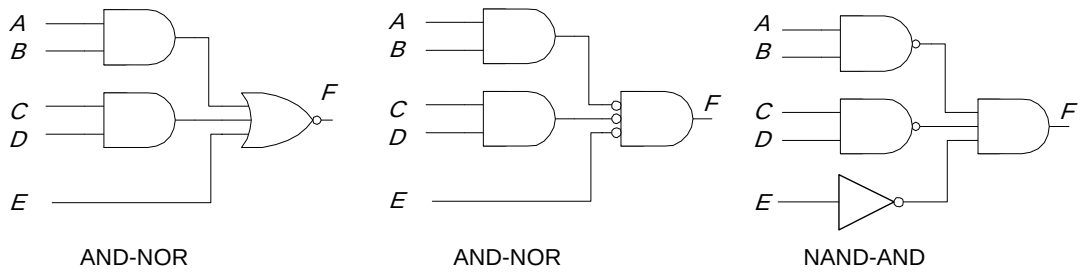
The eight nondegenerate forms are as follow:

AND-OR	OR-AND
NAND-NAND	NOR-NOR
NOR-OR	NAND-AND
OR-NAND	AND-NOR

The first gate listed in each of the forms constitutes a first level implementation. The second gate listed is a single gate placed in the second level. Note that any two forms listed in the same line are duals of each other. The first four forms listed above have been investigated previously. The remaining four forms are investigated next.

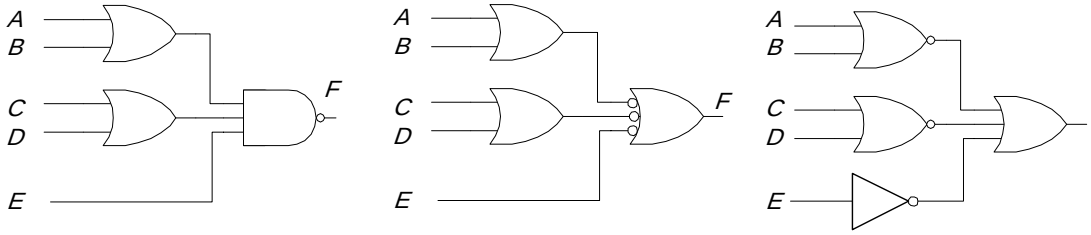
AND-OR-INVERT Implementation

The two forms NAND-AND and AND-NOR are equivalent forms and can be treated together. Both perform the AND-OR-INVERT function as shown in the circuit below which implements the function $F = (AB + CD + E)'$.



OR-AND-INVERT Implementation

The OR-NAND and NOR-OR forms perform the OR-AND-INVERT function. This is shown in the circuit below which implements $F = [(A + B)(C + D)E]'$.



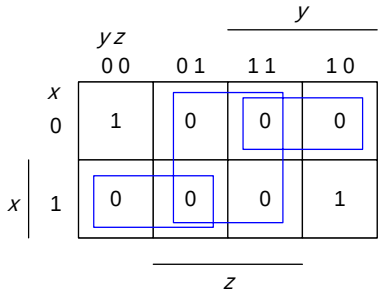
OR-NAND OR-NAND NOR-OR

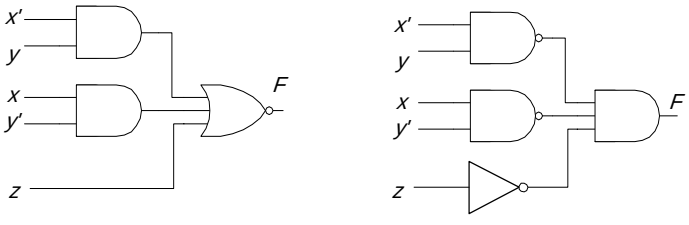
Example: Implement the function given by the map below with the four two-level forms discussed above.

Map simplification in SOP form

$F = xy'z' + xyz'$

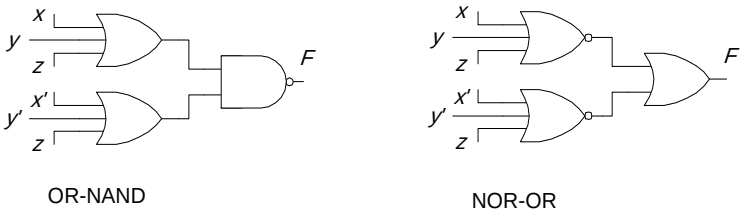
$F' = x'y + xy' + z$





AND-NOR NAND-AND

$F = (x'y + xy' + z)'$



OR-NAND NOR-OR

$F = [(x + y + z)(x' + y' + z)]'$