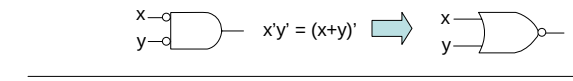
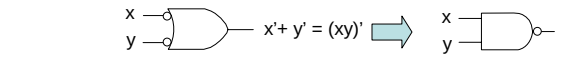
Sections 3.7-3.9
Combinational Circuit Implementation

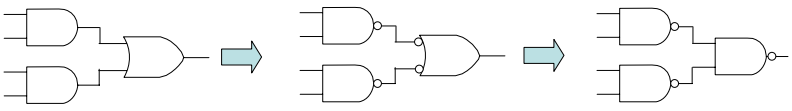
Bubble notation

To keep diagrams simple, we can represent the complement operation with just the circle part of the inverter gate.

Among other things, this helps us visualize ways to convert between gate types:



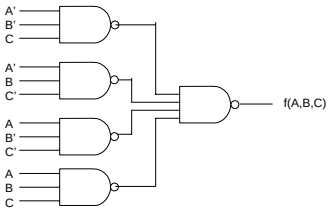




Review – 2-level NAND implementations

We can go directly from an SOP expression to a 2-level NAND circuit

f(ABC) = A'B'C + A'BC' + AB'C' + ABC



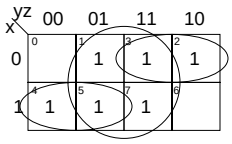
Note that a 2-level implementation does not necessarily use the minimum number of gates, nor can we constrain the number of inputs to the gates

General process for 2-level NAND circuits

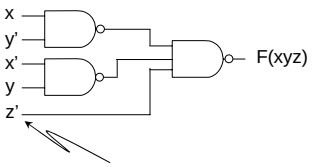
1. Simplify the function into SOP form
2. Use a first-level NAND for each term that has at least 2 literals
3. Use a second-level NAND to combine all terms
4. Invert any term with one literal in the first level.

Example 3.10

F(xyz) = Σm(1-7)



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

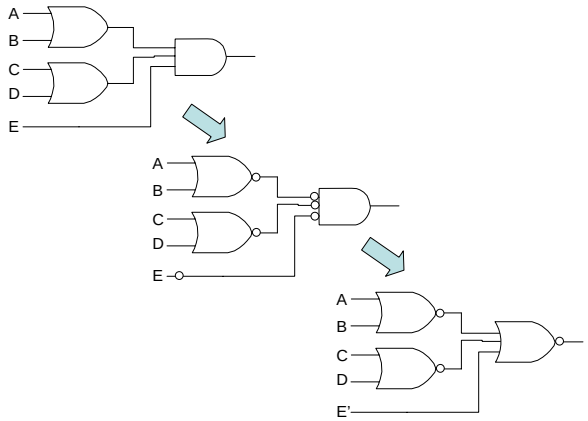


Note that the term with a single literal has to be complemented.

2-Level NOR circuits

To implement a two-level circuit using NOR gates, start with POS form

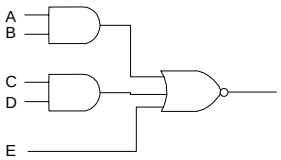
F(ABCDE) = (A+B)(C+D)E



AND-OR-Invert Circuits

Start with the complemented form of an SOP expression:

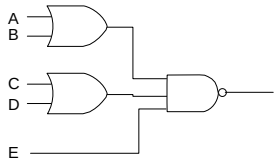
F(ABCDE) = (AB + CD + E)'



OR-AND-Invert

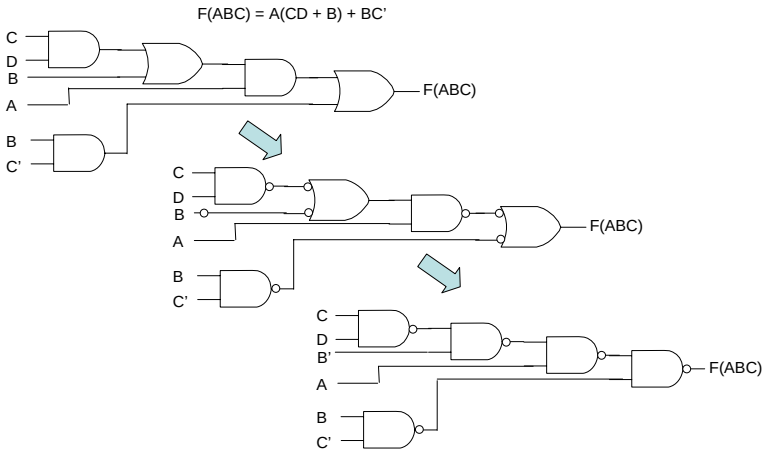
Start with the complemented form of the POS expression:

F(ABCDE) = ((A+B)(C+D)E)'



Multi-level implementations

For a number of possible reasons (reducing # of gates, reducing # of gate inputs, producing partial results,…) you may want to have more than 2 levels.



XOR gates - review

- A XOR B = AB' + A'B
- Some characteristics:
  x XOR 0 = x
  x XOR 1 = x'
  x XOR x = 0
  x XOR x' = 1
  x XOR y' = x' XOR y = (x XOR y)'

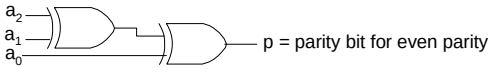
• XOR implements the "odd" function:

a XOR b XOR c XOR … = 1 IFF an odd # of (a,b,c,…) equals 1

XORs in Parity Encoding

- Recall that we can add a "parity bit" to a data item that lets us detect some errors
  - "odd parity" == set the parity bit so that the overall # of 1's is odd
  - "even parity" == set the parity bit so that the overall # of 1's is even

Even parity generator for 3-bit data



Even parity checker for 3-bit data