

Technology Mapping

To reduce implementation cost and turnaround time, designers use gate-arrays.

These gate-arrays contains only m -input NAND and NOR gates where m is usually 3.

Technology mapping is the process where we convert a schematic (expression) with AND, OR, and NOT gates to NAND and NOR gates.

The conversion is based on the following rules:

Rule 1: $xy = ((xy)')'$

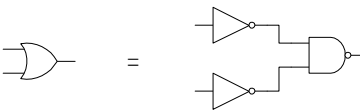
Rule 2: $x + y = ((x + y)')' = (x' y')'$

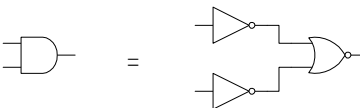
Rule 3: $xy = ((xy)')' = (x' + y')'$

Rule 4: $x + y = ((x + y)')' = (x' y')'$

Rule 5: $x'' = x$

Rule 1: 

Rule 2: 

Rule 3: 

Rule 4: 

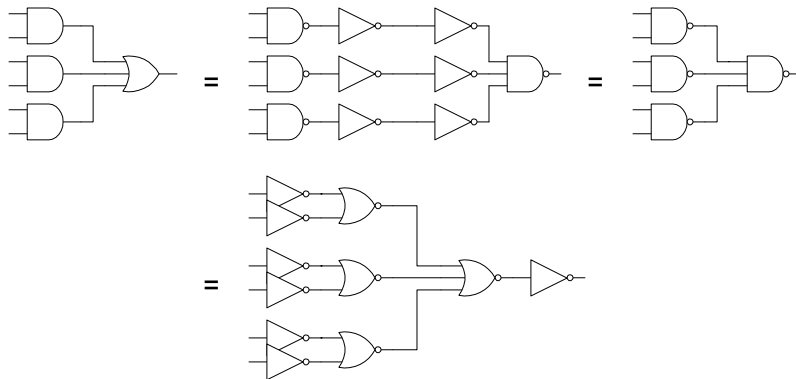
Rule 5: 

Replace AND and OR gates with NAND gates by using Rules 1 and 2.

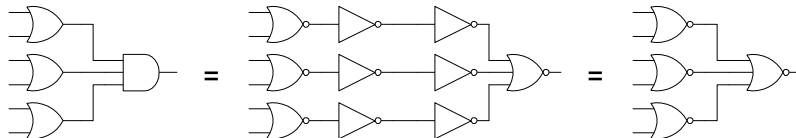
Replace AND and OR gates with NOR gates by using Rules 3 and 4.

Eliminate double inverters whenever possible by using Rule 5.

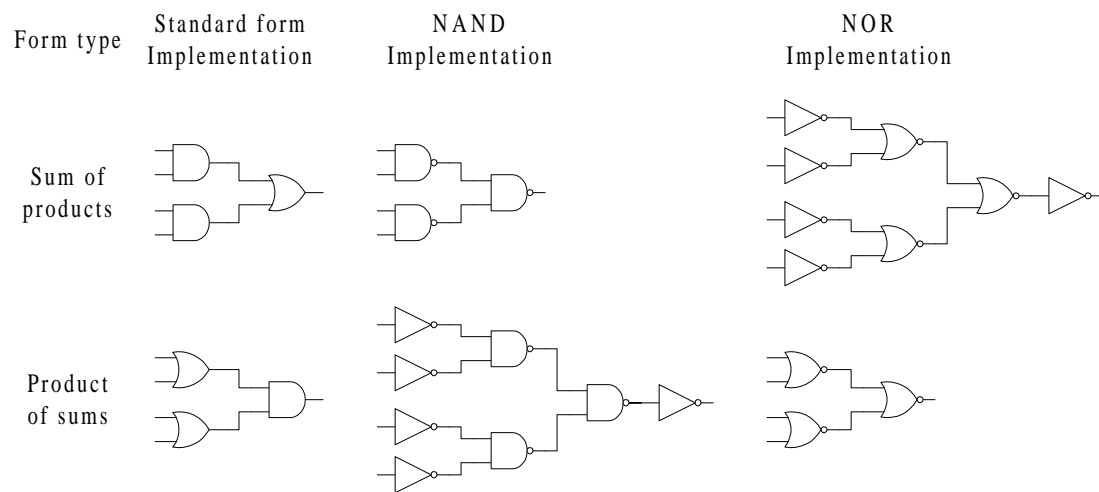
Example: Derive the NAND and NOR implementations of the carry function $c_{i+1} = x_i y_i + x_i c_i + y_i c_i$



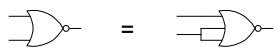
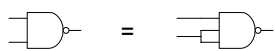
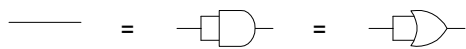
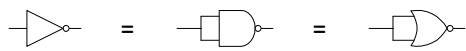
A better NOR implementation is to start with the product-of-sums expression $c_{i+1} = (x_i + y_i) (x_i + c_i) (y_i + c_i)$



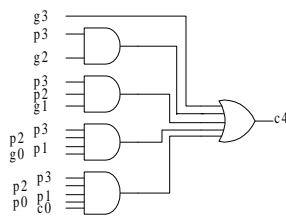
Translation of Sum of Products and Product of Sums to NAND and NOR schematics:






Four more identities:



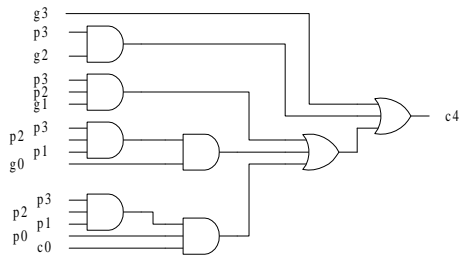
Example: Implement the following schematic using 2 and 3-input NAND gates:



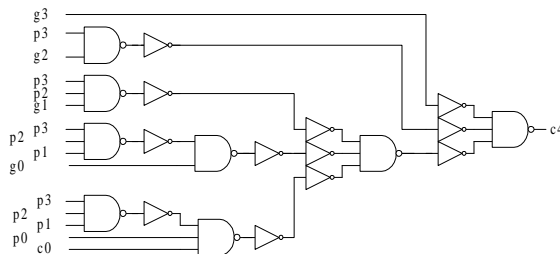
Rule 1:  =  

Rule 2:  =  

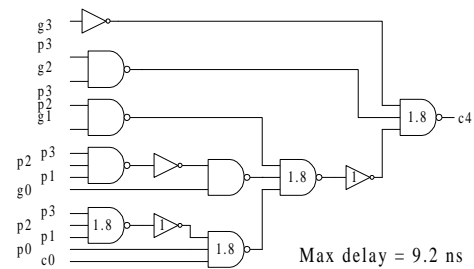
AND-OR implementation



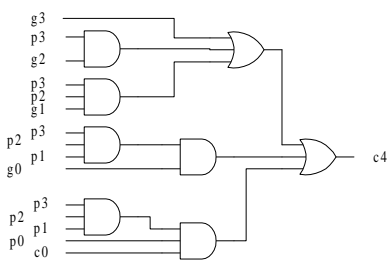
using only 2 & 3-input gates



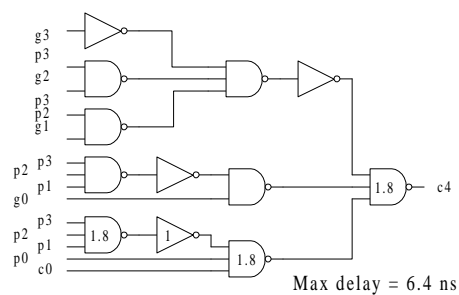
convert to NAND implementation



NAND implementation

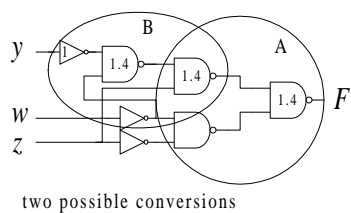
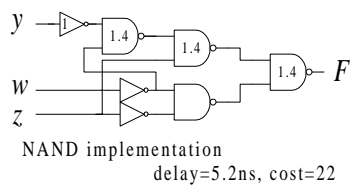
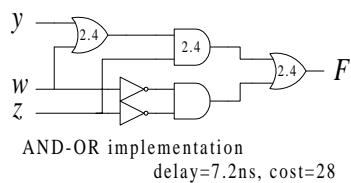


optimized decomposition




optimized NAND implementation

Example: Convert the expression $w'z' + z(w + y)$ into a logic schematic using any of the gates from the library defined in Tables 3.14, 3.15 and 3.16.



Rule 1:  = 

Rule 2:  = 