Canonical Boolean function representation

Five canonical representations are described in the textbook:

- truth table
- canonical sum
- minterm list
- canonical product
- maxterm list

Representations 4 and 5 are *duals* of 2 and 3.

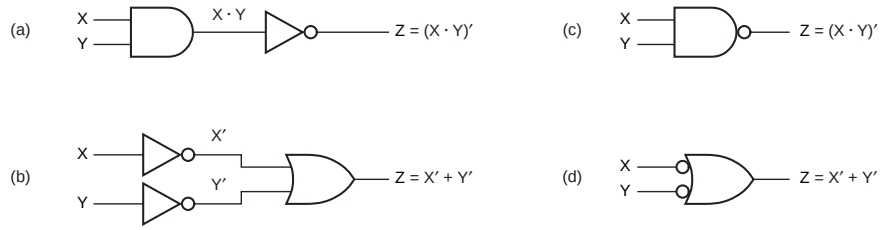
Most people find it difficult to think in terms of products of sums, but CAD programs manage quite nicely.

Terminology

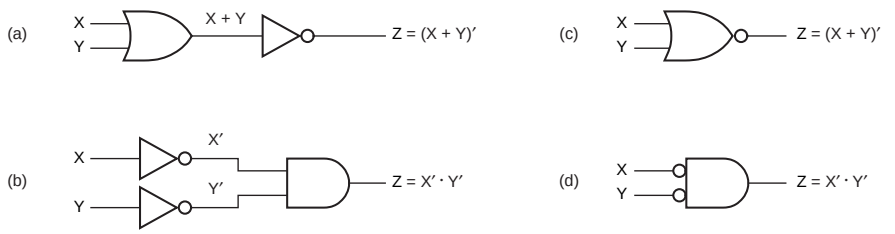
- Constant: 0 (F) or 1 (T)
- Literal: a variable or its complement (negation): X, Y′, /Z
- Product term: AND of one or more literals: X, /Y * Z, /W * /X * Y * Z
- Sum term: OR of one or more literals: X, /Y + Z, /W + /X + Y + Z
- Normal term: a product or sum term in which no variable appears more than once; that is, if a variable occurs, its complement does not, and vice versa.
- Minterm: a normal product term that includes all variables that are primary inputs; that is, every variable or its literal is in the term but not both.
- Maxterm: a normal sum term that includes each primary input or its complement. Dual of minterm.
- Sum of products: sum (OR) of product (AND) terms.
- Product of sums: product (AND) or sum (OR) terms.
- Canonical sum: sum of minterms
- Canonical product: product of maxterms

DeMorgan's theorems

$(X \cdot Y)' = X' + Y'$:



$(X + Y)' = X' \cdot Y'$:



Duality

A circuit computes dual logic functions, depending on the convention: positive logic vs. negative logic.

Positive logic AND = negative logic OR:

X	Y	Z
LOW	LOW	LOW
LOW	HIGH	LOW
HIGH	LOW	LOW
HIGH	HIGH	HIGH

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

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

Positive logic OR = negative logic AND:

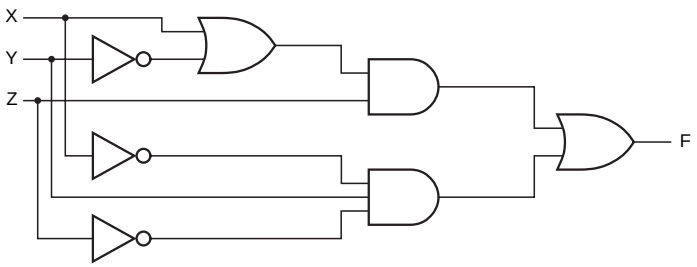
X	Y	Z
LOW	LOW	LOW
LOW	HIGH	HIGH
HIGH	LOW	HIGH
HIGH	HIGH	HIGH

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

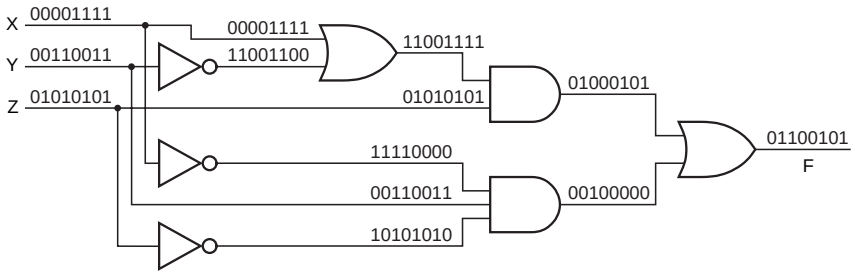
X	Y	Z
1	1	1
1	0	0
0	1	0
0	0	0

Combinational circuit analysis

Circuit to be analyzed:

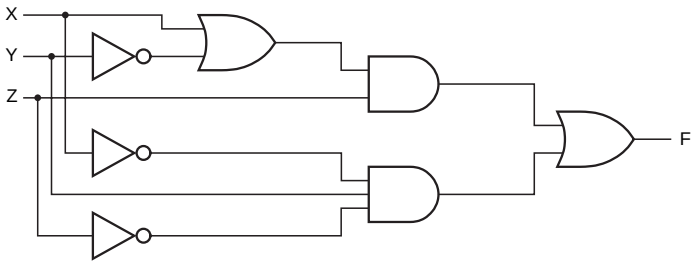


Analysis output: truth table

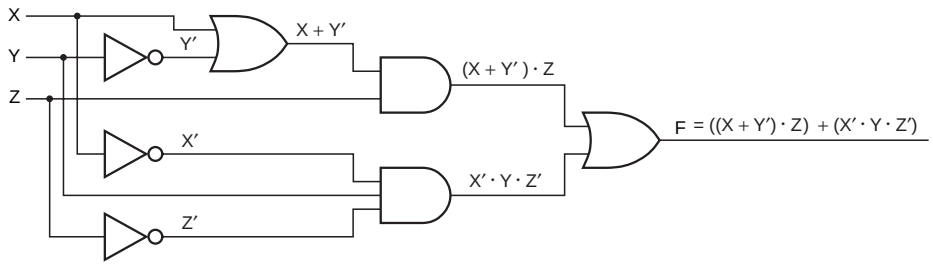


Combinational circuit analysis (2)

Circuit to be analyzed:

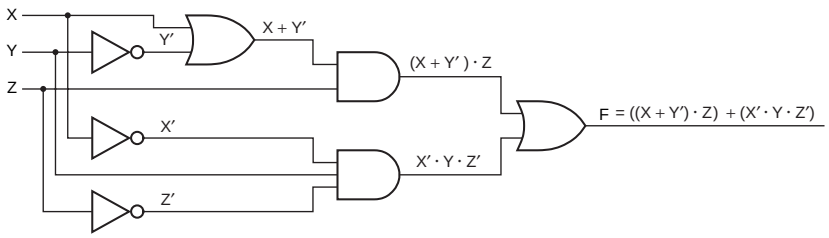


Analysis output: formula

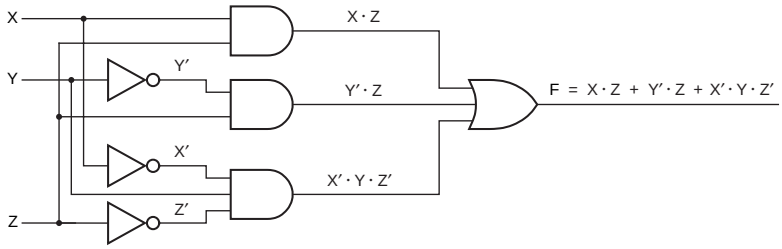


Circuit reimplementation: sum of products

Sample circuit:

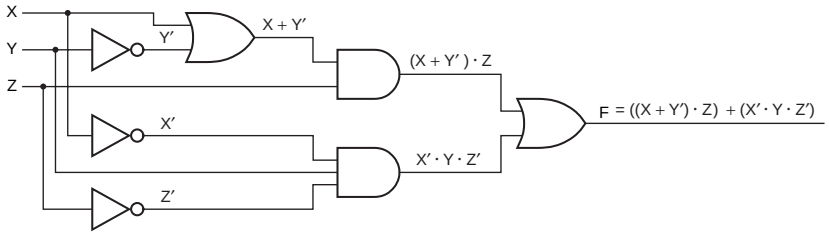


Convert F to sum of products by "multiplying out':

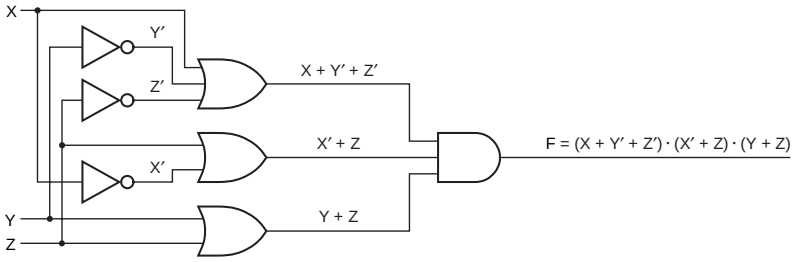


Circuit reimplementation: product of sums

Example circuit:

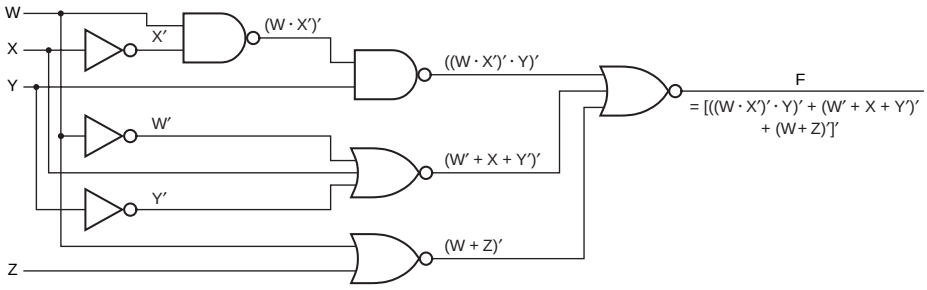


Convert F to product of sums by "adding out":

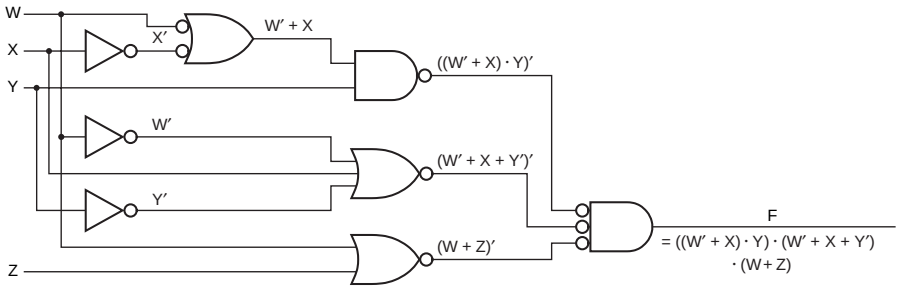


Analysis of NAND-NOR circuit

Example circuit:

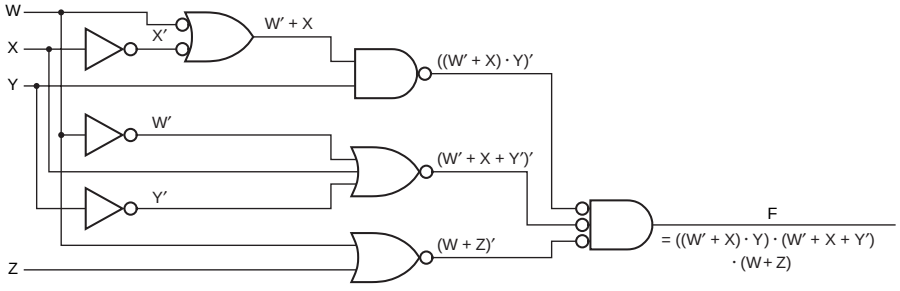


Use DeMorgan's theorem:

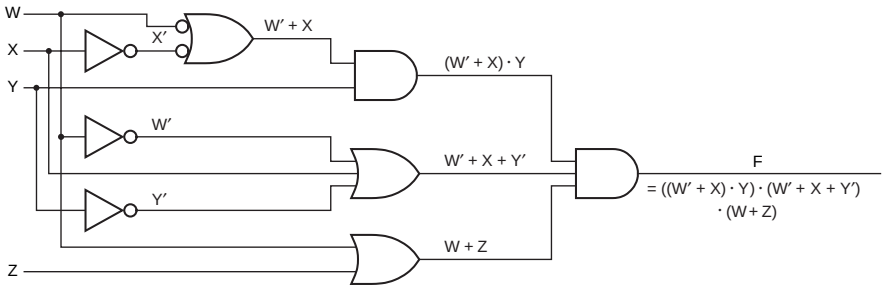


Analysis of NAND-NOR circuit (2)

Circuit modified using DeMorgan:

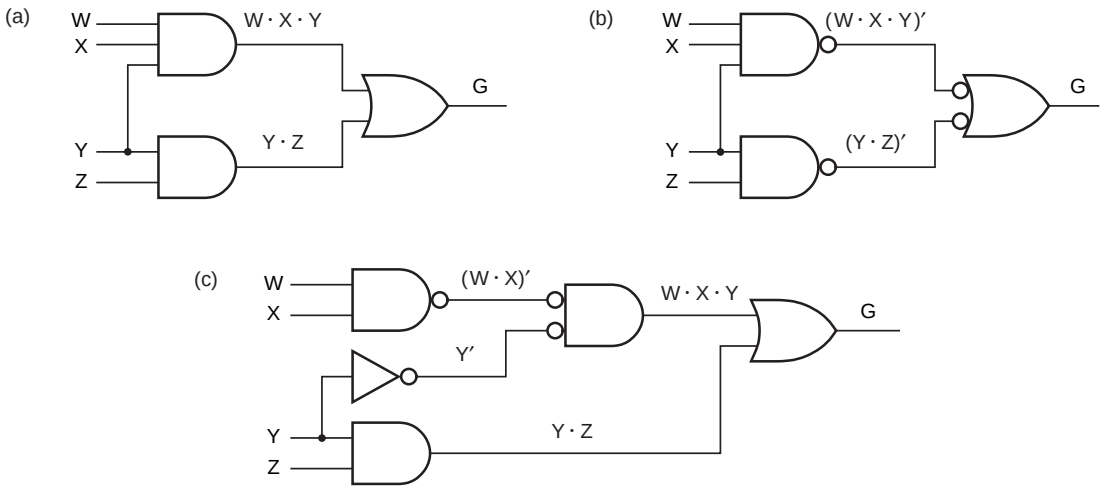


Replace NOR-NOR by OR-AND:



AND-OR ⇒ NAND-NAND

Three circuits for G(W,X,Y,Z) = W · X · Y + Y · Z:
(a) AND-OR (b) NAND-NAND (c) ad hoc.



Mystery combinational circuit

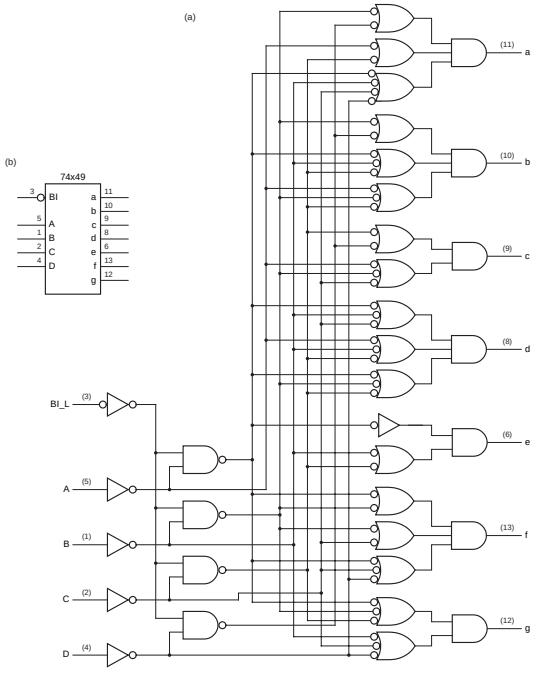


Table 5-21 Truth table for a 74x49 seven-segment decoder.

Inputs					*Outputs*						
BI_L	D	C	B	A	a	b	c	d	e	f	g
0	x	x	x	x	0	0	0	0	0	0	0
1	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
1	0	0	1	0	1	1	0	1	1	0	1
1	0	0	1	1	1	1	1	1	0	0	1
1	0	1	0	0	0	1	1	0	0	1	1
1	0	1	0	1	1	0	1	1	0	1	1
1	0	1	1	0	0	0	1	1	1	1	1
1	0	1	1	1	1	1	1	0	0	0	0
1	1	0	0	0	1	1	1	1	1	1	1
1	1	0	0	1	1	1	1	0	0	1	1
1	1	0	1	0	0	0	0	1	1	0	1
1	1	0	1	1	0	0	1	1	0	0	1
1	1	1	0	0	0	1	0	0	0	1	1
1	1	1	0	1	1	0	0	1	0	1	1
1	1	1	1	0	0	0	0	1	1	1	1
1	1	1	1	1	0	0	0	0	0	0	0