MINTERMS

A binary variable may appear either in its normal form (x) or in its complement form (x').

Now consider two binary variables x and y combined with an AND operation. Since each variable may appear in either form, there are four possible combinations: xy, x'y, xy' and x'y'. Each of these four AND terms is called a minterm, or a standard product.

In a similar manner, n variables can be combined to form 2^n minterms.

With n variables forming an OR term, with each variable being primed or unprimed, provide 2^n possible combinations, called maxterms, or standard sums.

A minterm of n variables is a product of n literals in which each variable appears exactly once in either true or complemented form, but not both. A literal is a variable or its complement.

The minterm associated with each input combination is the & (AND), or product, of the input variables, while the maxterm is the | (OR), or sum, of the inverted input variables

Each minterm equals 1 at exactly one particular input combination and is equal to 0 at all other combinations

Each maxterm equals 0 at exactly one of the 8 possible input Combinations and is equal to 1 at all other combinations

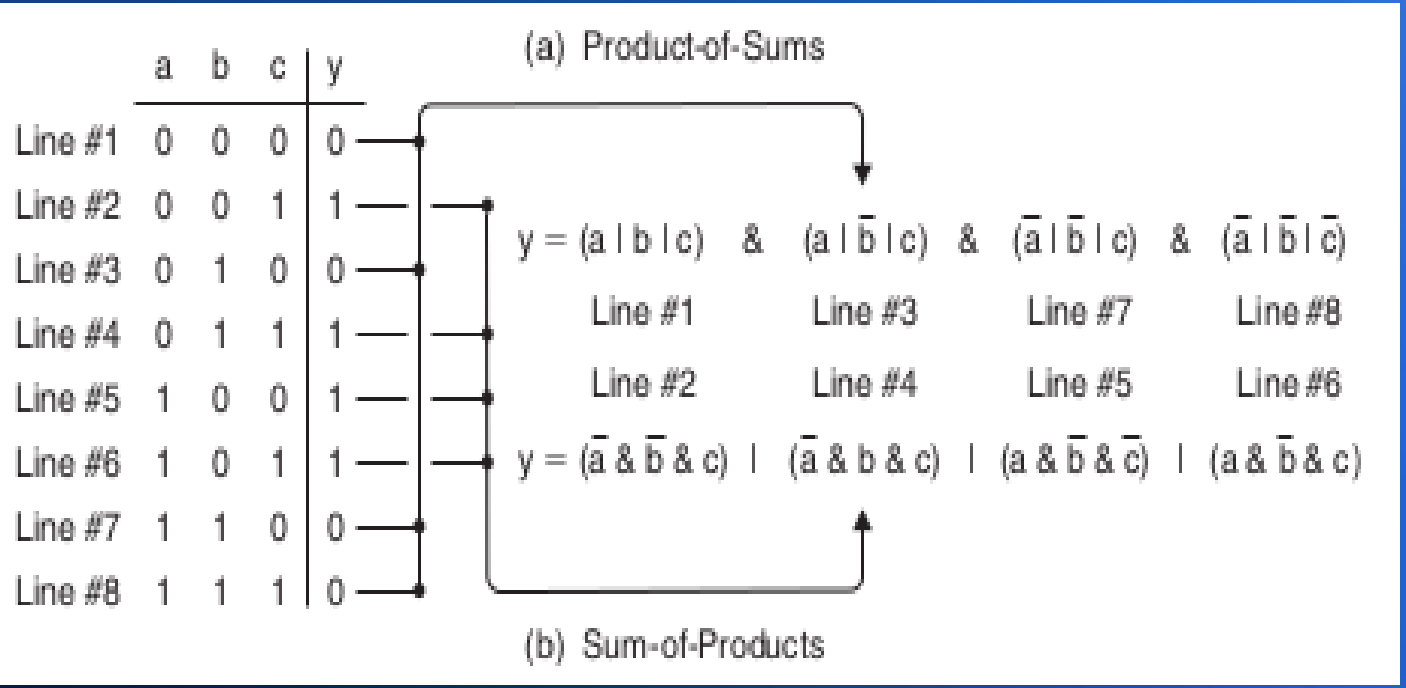
a	b	c	Minterms	Maxterms
0	0	0	$(\bar{a} \,\&\, \bar{b} \,\&\, \bar{c})$	$(a \mid b \mid c)$
0	0	1	$(\bar{a} \,\&\, \bar{b} \,\&\, c)$	$(a \mid b \mid \bar{c})$
0	1	0	$(\bar{a} \,\&\, b \,\&\, \bar{c})$	$(a \mid \bar{b} \mid c)$
0	1	1	$(\bar{a} \,\&\, b \,\&\, c)$	$(a \mid \bar{b} \mid \bar{c})$
1	0	0	$(a \,\&\, \bar{b} \,\&\, \bar{c})$	$(\bar{a} \mid b \mid c)$
1	0	1	$(a \,\&\, \bar{b} \,\&\, c)$	$(\bar{a} \mid b \mid \bar{c})$
1	1	0	$(a \,\&\, b \,\&\, \bar{c})$	$(\bar{a} \mid \bar{b} \mid c)$
1	1	1	$(a \,\&\, b \,\&\, c)$	$(\bar{a} \mid \bar{b} \mid \bar{c})$

Minterms and Maxterms

min(0,0) = 0
min(0,1) = 0
min(1,0) = 0
min(1,1) = 1
So minimum is pretty much like logical AND.

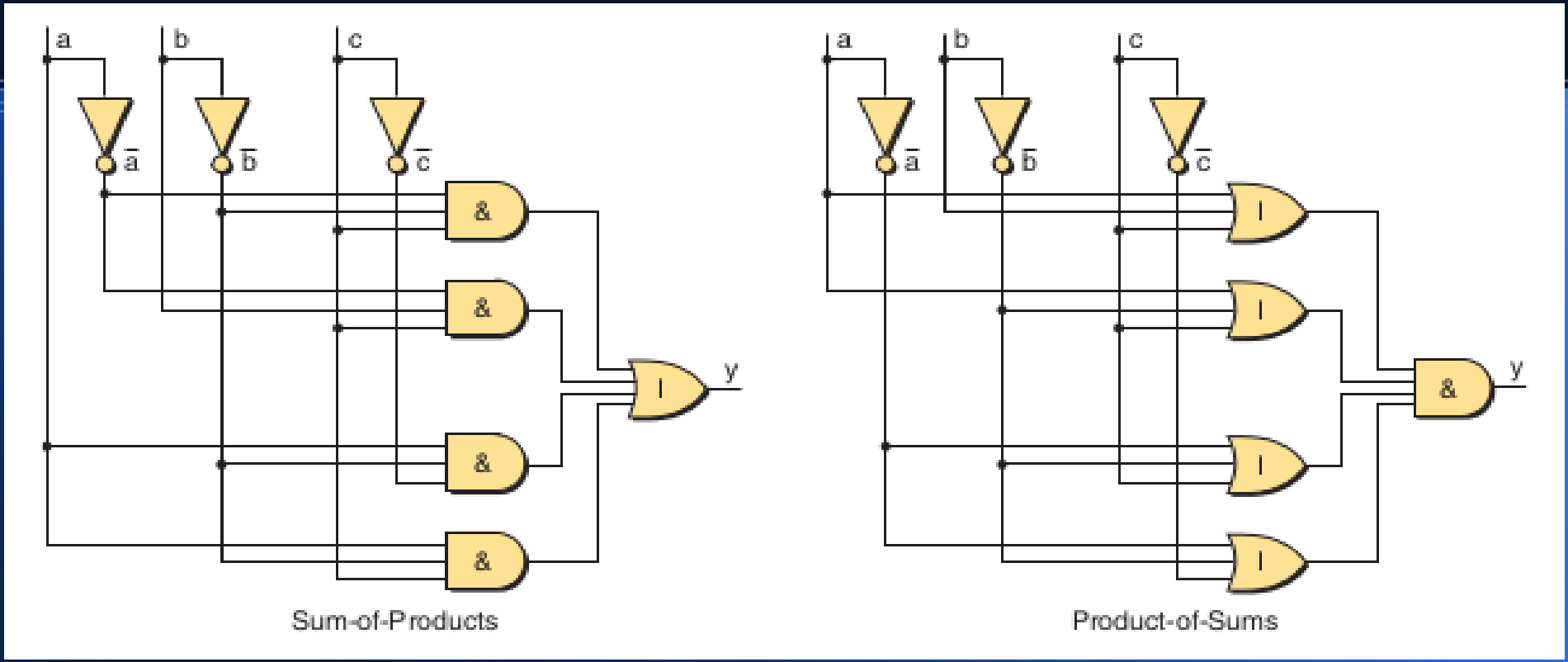
max(0,0) = 0
max(0,1) = 1
max(1,0) = 1
max(1,1) = 1
So maximum is pretty much like logical OR.

The minterms corresponding to each line in the truth table for which the output is a logic 1 are extracted and combined using | (OR) operators; this method results in an equation said to be in sum-of-products form



The maxterms corresponding to each line in the truth table for which the output is a logic 0 are combined using & (AND) operators; this method results in an equation said to be in product-of-sums form

Sum-of-products versus products-of-sums (implementations)



In general, a two-level implementation is preferred because it produces the least amount of delay through the gates when the signal propagates from the inputs to the output. However, the number of inputs to a given gate might not be practical.

For a function whose output is logic 1 fewer times than it is logic 0, it is generally easier to extract a sum-of-products equation.

By comparison, if the output is logic 0 fewer times than it is logic 1, it is generally easier to extract a product-of-sums equation.

The sum-of-products and product-of-sums forms complement each other and return identical results.

Furthermore, an equation in either form can be transformed into its alternative form by complementing it and then using a DeMorgan Transformation

Minterms and Maxterms for Three Binary Variables

			Minterms		Maxterms	
x	y	z	Term	Designation	Term	Designation
0	0	0	$x'y'z'$	m_0	$x + y + z$	M_0
0	0	1	$x'y'z$	m_1	$x + y + z'$	M_1
0	1	0	$x'yz'$	m_2	$x + y' + z$	M_2
0	1	1	$x'yz$	m_3	$x + y' + z'$	M_3
1	0	0	$xy'z'$	m_4	$x' + y + z$	M_4
1	0	1	$xy'z$	m_5	$x' + y + z'$	M_5
1	1	0	xyz'	m_6	$x' + y' + z$	M_6
1	1	1	xyz	m_7	$x' + y' + z'$	M_7

x	y	z	Function f_1	Function f_2
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$f_1 = x'y'z + xy'z' + xyz = m_1 + m_4 + m_7$$

$$f_1 = (x + y + z)(x + y' + z)(x' + y + z')(x' + y' + z)$$
$$= M_0 \cdot M_2 \cdot M_3 \cdot M_5 \cdot M_6$$

$$f_2 = x'yz + xy'z + xyz' + xyz = m_3 + m_5 + m_6 + m_7$$

$$f_2 = (x + y + z)(x + y + z')(x + y' + z)(x' + y + z)$$
$$= M_0 M_1 M_2 M_4$$

The summation symbol Σ stands for the ORing of terms; the numbers following it are the indices of the minterms of the function.

$$F = A'B'C + AB'C + AB'C + ABC' + ABC$$
$$= m_1 + m_4 + m_5 + m_6 + m_7$$

$$F(A, B, C) = \Sigma(1, 4, 5, 6, 7)$$

The product symbol Π, denotes the ANDing of maxterms; the numbers are the indices of the maxterms of the function.

$$F = (x + y + z)(x + y' + z)(x' + y + z)(x' + y + z')$$
$$= M_0M_2M_4M_5$$

$$F(x, y, z) = \Pi(0, 2, 4, 5)$$

To convert from one canonical form to another, interchange the symbols Σ and Π and list those numbers missing from the original form.

$F(A, B, C) = \Sigma (1, 4, 5, 6, 7)$

$F'(A, B, C) = \Sigma (0, 2, 3) = m0 + m2 + m3$

De Morgan's law

$F = (m0 + m2 + m3)' = m0' \ m2' \ m3' = M0 \ M2 \ M3 = \Pi (0, 2, 3)$

Express the complement of the following functions in sum-of-minterms form:

(a) $F(A,B,C,D) = \Sigma(2,4,7,10,12,14)$

(b) $F(x,y,z) = \Pi(3,5,7)$

(a) $F(A, B, C, D) = \Sigma(2, 4, 7, 10, 12, 14)$

$F'(A, B, C, D) = \Sigma(0, 1, 3, 5, 6, 8, 9, 11, 13, 15)$

(b) $F(x, y, z) = \Pi(3, 5, 7)$

$F' = \Sigma(3, 5, 7)$

Convert each of the following to the other canonical form:

(a) $F(x,y,z) = \Sigma(1,3,5)$

(b) $F(A,B,C,D) = \Pi(3,5,8,11)$

(a) $F(x, y, z) = \Sigma(1, 3, 5) = \Pi(0, 2, 4, 6, 7)$

(b) $F(A, B, C, D) = \Pi(3, 5, 8, 11) = \Sigma(0, 1, 2, 4, 6, 7, 9, 10, 12, 13, 14, 15)$

Canonical Form

In a mathematical context, the term canonical form is taken to mean a generic or basic representation.

Canonical forms provide the means to compare two expressions without falling into the trap of trying to compare “apples” with “oranges.”

The sum-of-products and product-of-sums representations are different canonical forms.

Thus, in order to compare two Boolean equations, both must first be coerced into the same canonical form; either sum-of-products or product-of-sums.

The strength (and purpose) of the canonical form is that it allows us to devise a simple algorithm for designing any circuit automatically, starting from a set of Boolean equations and finishing with an integrated circuit.

An integrated circuit, called a programmable logic array, supports this design methodology.

However, we also note that the canonical form is not the smallest representation of most circuits, and consequently not the best implementation according to the size criterion.

An expression is said to be in sum-of-products form when all products are the products of single variables

SOP $AB' + CD'E + AC'E'$

NOT SOP $(A + B)CD + EF$

multiply out $(A + BC)(A + D + E)$

multiply out $(A + BC)(A + D + E)$

$$(A + BC)(A + D + E) = A + AD + AE + ABC + BCD + BCE$$
$$= A(1 + D + E + BC) + BCD + BCE$$
$$= A + BCD + BCE$$

$X = A, \qquad Y = BC, \qquad Z = D + E$

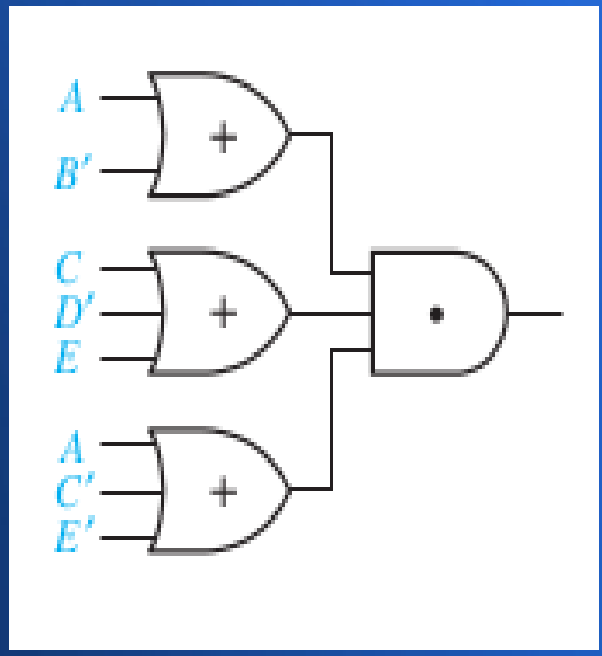
$$(X + Y)(X + Z) = X + YZ = A + BC(D + E) = A + BCD + BCE$$

(second distributive rule)

An expression is in product-of-sums (POS) form when all sums are the sums of single variables

$$(A + B')(C + D' + E)(A + C' + E')$$

A product-of-sums expression can always be realized directly by one or more OR gates feeding a single AND gate at the circuit output



A sum-of-products expression can always be realized directly by one or more AND gates feeding a single OR gate at the circuit output