Chapter 2: Combinational Logic Circuits

Binary Logic

- Binary logic deals with binary variables, which take on two discrete values
- Associated with binary variables are three basic logical operations

Operation:	AND	OR	NOT
Expression:	xy or x.y	x + y	x′

Truth table:

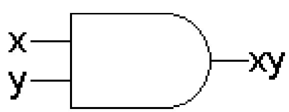
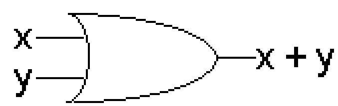
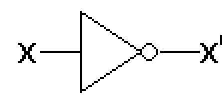
x	y	xy
0	0	0
0	1	0
1	0	0
1	1	1

x	y	x+y
0	0	0
0	1	1
1	0	1
1	1	1

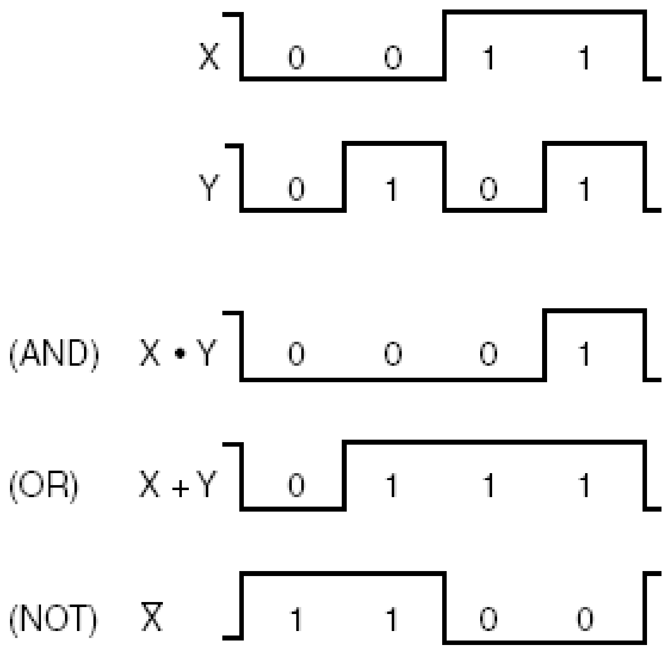
x	x′
0	1
1	0

Logic Gates

- Each of basic operations can be implemented in hardware using a logic gate
  - Symbols for each of the logic gates are shown below
  - These gates output the AND, OR, and NOT of their inputs

Operation:	AND	OR	NOT
Expression:	xy or $x \bullet y$	$x + y$	x'
Logic gate:			
	AND gate	OR gate	NOT gate (inverter)

Timing Diagram



(b) Timing diagram

Fig. 2-1 Digital Logic Gates

Gates More Than Two Inputs

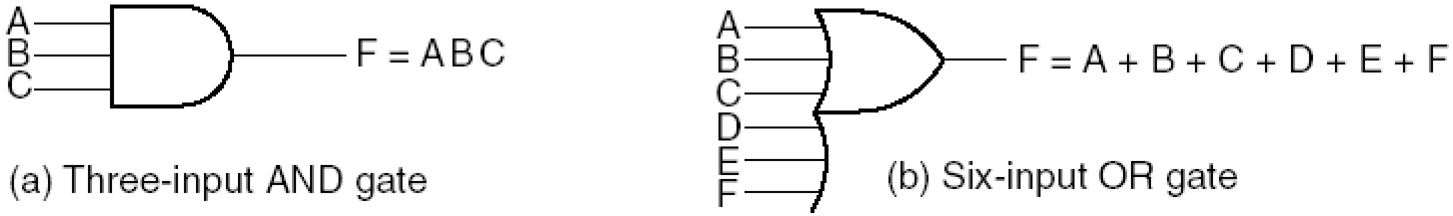


Fig. 2-2 Gates with More than Two Inputs

Boolean Function

- We can use basic operations to form complex Boolean functions, e.g.,

$$f(x,y,z) = (x + y')z + x'$$

- Some terminology and notation:
  - f is the name of the function
  - (x,y,z) are the input variables, each representing 1 or 0. Listing the inputs is optional, but sometimes helpful
  - A literal is any occurrence of an input variable or its complement. The function above has four literals: x, y', z, and x'

- Precedences are important, but not too difficult
  - NOT has the highest precedence, followed by AND, and then OR.
  - Fully parenthesized, the function above would be kind of messy:

$$f(x,y,z) = (((x +(y'))z) + x')$$

Truth Tables

- A truth table shows all possible inputs and outputs of a function
- Remember that each input variable represents either 1 or 0
  - Because there are only a finite number of values (1 and 0), truth tables themselves are finite
  - A function with n variables has 2^n possible combinations of inputs.
- Inputs are listed in binary order—in this example, from 000 to 111.

$f(x,y,z) = (x + y')z + x'$

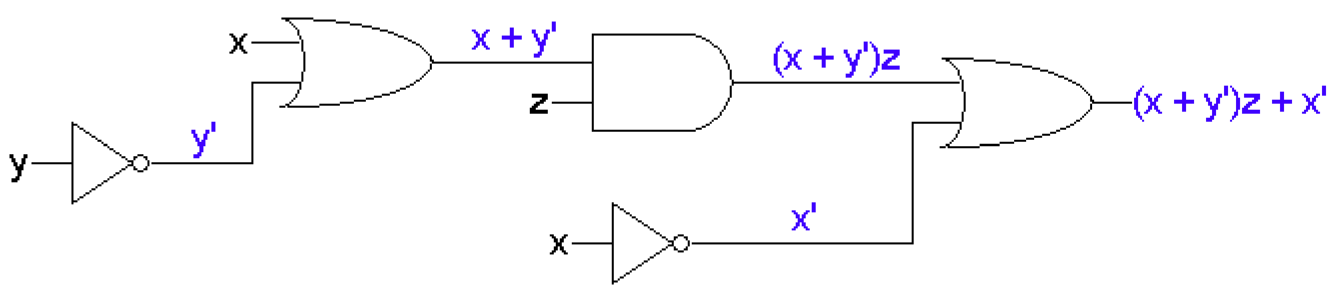
$f(0,0,0) = (0 + 1)0 + 1 = 1$
$f(0,0,1) = (0 + 1)1 + 1 = 1$
$f(0,1,0) = (0 + 0)0 + 1 = 1$
$f(0,1,1) = (0 + 0)1 + 1 = 1$
$f(1,0,0) = (1 + 1)0 + 0 = 0$
$f(1,0,1) = (1 + 1)1 + 0 = 1$
$f(1,1,0) = (1 + 0)0 + 0 = 0$
$f(1,1,1) = (1 + 0)1 + 0 = 1$

x	y	z	f(x,y,z)
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Expressions and Circuits

- *Any* Boolean expression can be converted into a circuit by combining basic gates
- The diagram below shows the inputs and outputs of each gate
- The precedences are explicit in a circuit. Clearly, we have to make sure that the hardware does operations in the right order!

$$(x + y')z + x'$$



Boolean Algebra

- A Boolean algebra requires
  - A set of elements **B**, which needs *at least* two elements (0 and 1)
  - Two binary (two-argument) operations OR and AND
  - A unary (one-argument) operation NOT
  - The axioms below must always be true
    - The magenta axioms deal with the complement operation
    - Blue axioms (especially 15) are different from regular algebra

1. $x + 0 = x$	2. $x \bullet 1 = x$	
3. $x + 1 = 1$	4. $x \bullet 0 = 0$	
5. $x + x = x$	6. $x \bullet x = x$	
7. $x + x' = 1$	8. $x \bullet x' = 0$	
9. $(x')' = x$		
10. $x + y = y + x$	11. $xy = yx$	Commutative
12. $x + (y + z) = (x + y) + z$	13. $x(yz) = (xy)z$	Associative
14. $x(y + z) = xy + xz$	15. $x + yz = (x + y)(x + z)$	Distributive
16. $(x + y)' = x'y'$	17. $(xy)' = x' + y'$	DeMorgan's

Comments on the Axioms

- The left and right columns of axioms are DUALs
  - exchange all ANDs with ORs, and 0s with 1s
- The duality principle of Boolean algebra: A Boolean equation remains valid if we take the dual of the expression on both side of the equal signs

1. $x + 0 = x$	2. $x \bullet 1 = x$	
3. $x + 1 = 1$	4. $x \bullet 0 = 0$	
5. $x + x = x$	6. $x \bullet x = x$	
7. $x + x' = 1$	8. $x \bullet x' = 0$	
9. $(x')' = x$		
10. $x + y = y + x$	11. $xy = yx$	Commutative
12. $x + (y + z) = (x + y) + z$	13. $x(yz) = (xy)z$	Associative
14. $x(y + z) = xy + xz$	15. $x + yz = (x + y)(x + z)$	Distributive
16. $(x + y)' = x'y'$	17. $(xy)' = x' + y'$	DeMorgan's

Are these axioms for real?

- We can show that these axioms are true, given the definitions of AND, OR and NOT

x	y	xy
0	0	0
0	1	0
1	0	0
1	1	1

x	y	x+y
0	0	0
0	1	1
1	0	1
1	1	1

x	x'
0	1
1	0

- The first 11 axioms are easy to see from these truth tables alone. For example, $x + x' = 1$ because of the middle two lines below (where $y = x'$)

x	y	x+y
0	0	0
0	1	1
1	0	1
1	1	1

Is X+YZ = (X+Y)(X+Z)?

X	Y	Z	X+Y	X+Z	YZ	X+YZ	(X+Y)(X+Z)
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	1	1	1	1
1	0	0	1	1	0	1	1
1	0	1	1	1	0	1	1
1	1	0	1	1	0	1	1
1	1	1	1	1	1	1	1

DeMorgan's Theorem

- We can make up truth tables to prove (both parts of) DeMorgan's law
- For (x + y)' = x'y', we can make truth tables for (x + y)' and for x'y'

x	y	x + y	(x + y)'
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

x	y	x'	y'	x'y'
0	0	1	1	1
0	1	1	0	0
1	0	0	1	0
1	1	0	0	0

- Since both of the columns in blue are the same, this shows that (x + y)' and x'y' are equivalent

$$\overline{X_1 + X_2 + \ldots + X_n} = \overline{X}_1 \overline{X}_2 \ldots \overline{X}_n$$

$$\overline{X_1 . X_2 \ldots X_n} = \overline{X}_1 + \overline{X}_2 + \ldots + \overline{X}_n$$

Simplification with Axioms

- We can now start doing some simplifications

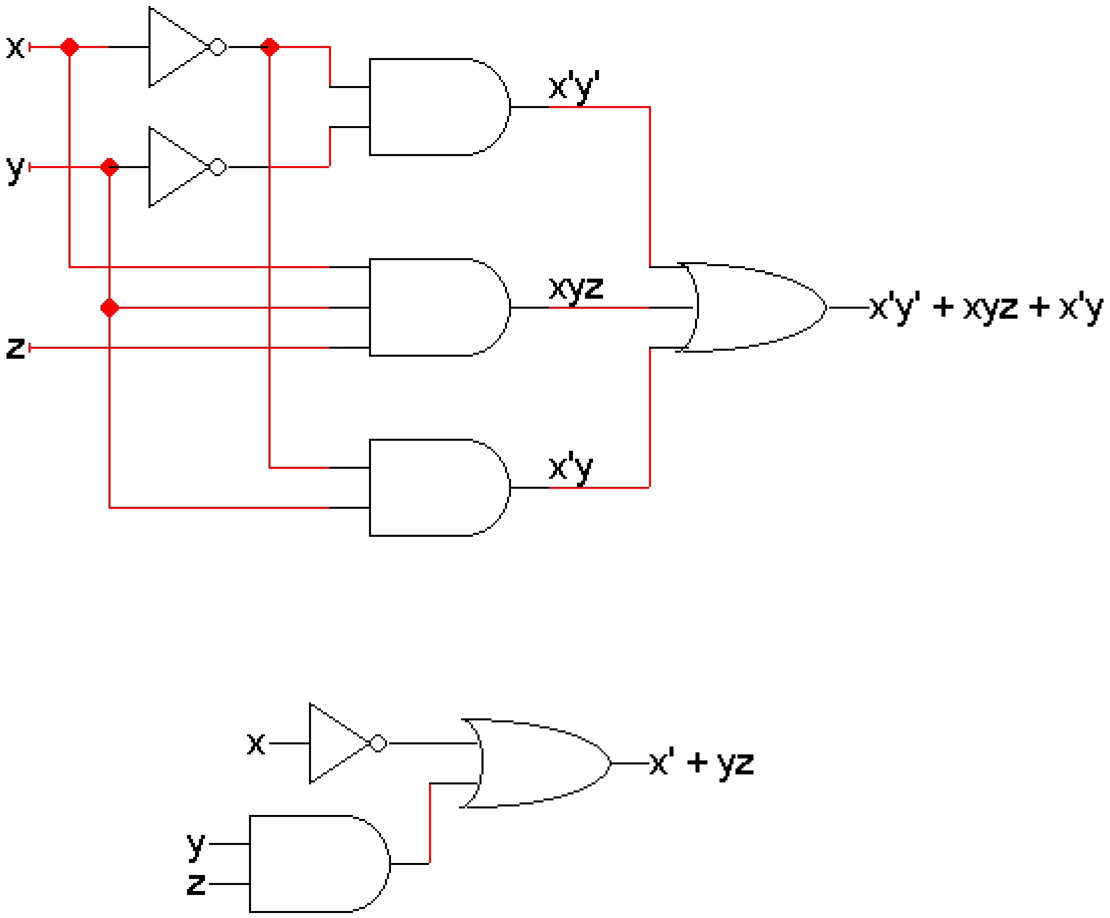
$x'y' + xyz + x'y$

$= x'(y' + y) + xyz$	[Distributive; $x'y' + x'y = x'(y' + y)$]
$= x' \bullet 1 + xyz$	[Axiom 7; $y' + y = 1$]
$= x' + xyz$	[Axiom 2; $x' \bullet 1 = x'$]
$= (x' + x)(x' + yz)$	[Distributive]
$= 1 \bullet (x' + yz)$	[Axiom 7; $x' + x = 1$]
$= x' + yz$	[Axiom 2]

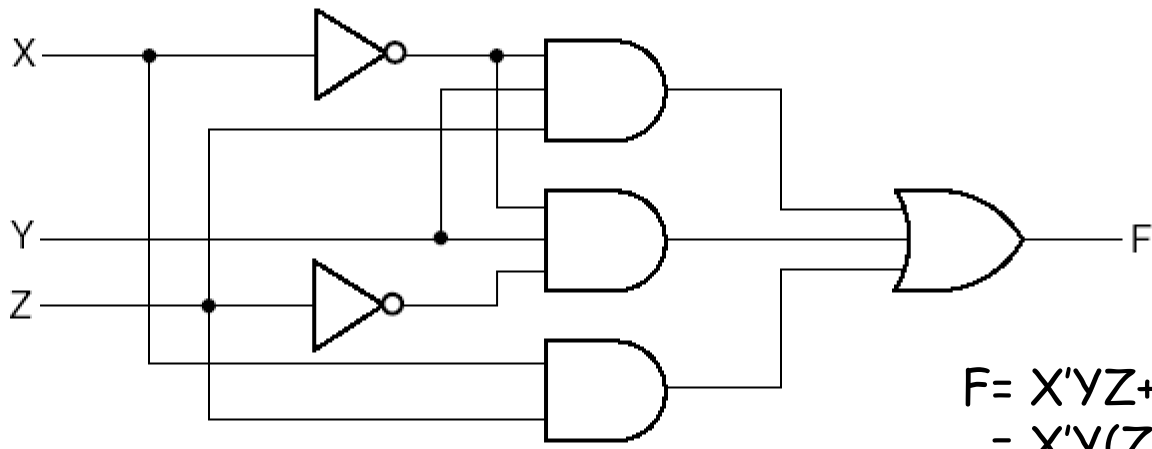
1. $x + 0 = x$	2. $x \bullet 1 = x$	
3. $x + 1 = 1$	4. $x \bullet 0 = 0$	
5. $x + x = x$	6. $x \bullet x = x$	
7. $x + x' = 1$	8. $x \bullet x' = 0$	
9. $(x')' = x$		
10. $x + y = y + x$	11. $xy = yx$	Commutative
12. $x + (y + z) = (x + y) + z$	13. $x(yz) = (xy)z$	Associative
14. $x(y + z) = xy + xz$	15. $x + yz = (x + y)(x + z)$	Distributive
16. $(x + y)' = x'y'$	17. $(xy)' = x' + y'$	DeMorgan's

Let's compare the resulting circuits

- Here are two different but *equivalent* circuits
- In general the one with fewer gates is "better":
  - It costs less to build
  - It requires less power
  - But we had to do some work to find the second form

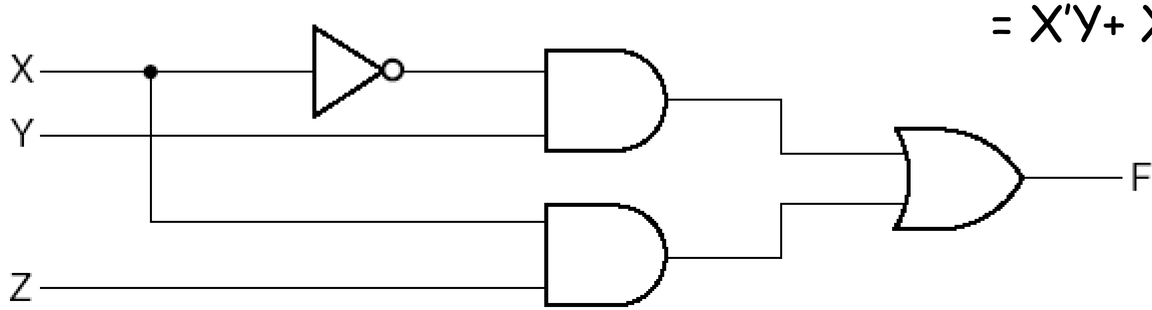


Another Example



(a) $F = \overline{X}YZ + \overline{X}Y\overline{Z} + XZ$

$$
\begin{aligned}
F &= X'YZ + X'YZ' + XZ \\
&= X'Y(Z+Z') + XZ \quad (14) \\
&= X'Y \cdot 1 + XZ \quad (7) \\
&= X'Y + XZ \quad (2)
\end{aligned}
$$



(b) $F = \overline{X}Y + XZ$

Fig. 2-4 Implementation of Boolean Function with Gates

Some More Laws

- Here are some more useful laws. Notice the duals again

1.	$x + xy = x$	4.	$x(x + y) = x$
2.	$xy + xy' = x$	5.	$(x + y)(x + y') = x$
3.	$x + x'y = x + y$	6.	$x(x' + y) = xy$

Consensus Theorem

- $XY + X'Z + YZ = XY + X'Z$ or its dual

  $(X+Y)(X'+Z)(Y+Z) = (X+Y)(X'+Z)$

- The proof of the theorem:

$$
\begin{aligned}
XY + X'Z + YZ &= XY + X'Z + YZ(X + X') \\
&= XY + X'Z + XYZ + X'YZ \\
&= XY + XYZ + X'Z + X'YZ \\
&= XY(1 + Z) + X'Z(1 + Y) \\
&= XY + X'Z
\end{aligned}
$$

The Complement of a Function

- The complement of a function always outputs 0 where the original function output 1, and 1 where the original produced 0
- In a truth table, we can just exchange 0s and 1s in the output column(s)

$f(x,y,z) = x' + xyz'$

x	y	z	f(x,y,z)
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

x	y	z	f'(x,y,z)
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Complementing a Function Algebraically

- You can use DeMorgan's law to keep "pushing" the complements inwards

$$f(x,y,z) = x(y'z' + yz)$$

$$\begin{aligned} f'(x,y,z) &= (\, x(y'z' + yz)\,)' && [\text{ complement both sides }] \\ &= x' + (y'z' + yz)' && [\text{ because } (xy)' = x' + y' \,] \\ &= x' + (y'z')'\,(yz)' && [\text{ because } (x + y)' = x'\,y' \,] \\ &= x' + (y + z)(y' + z') && [\text{ because } (xy)' = x' + y'\text{, twice}] \end{aligned}$$

- You can also take the dual of the function, and then complement each literal
  - If $f(x,y,z) = x(y'z' + yz)$…
  - …the dual of f is $x + (y' + z')(y + z)$…
  - …then complementing each literal gives $x' + (y + z)(y' + z')$…
  - …so $f'(x,y,z) = x' + (y + z)(y' + z')$

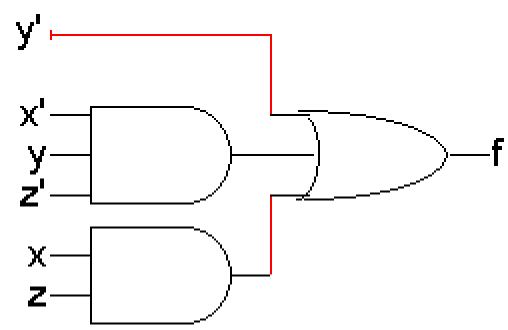
Standard Forms of Expression

- We can write expressions in many ways, but some ways are more useful than others

- A sum of products (SOP) expression contains:
  - Only OR (sum) operations at the "outermost" level
  - Each term that is summed must be a product of literals

$$f(x,y,z) = y' + x'yz' + xz$$

- The advantage is that any sum of products expression can be implemented using a two-level circuit
  - literals and their complements at the "0th" level
  - AND gates at the first level
  - a single OR gate at the second level



- This diagram uses some shorthands…
  - NOT gates are implicit
  - literals are reused

Minterms

- A minterm is a special product of literals, in which each input variable appears exactly once

- A function with n variables has 2^n minterms (since each variable can appear complemented or not)

- A three-variable function, such as f(x,y,z), has $2^3 = 8$ minterms:

$x'y'z'$	$x'y'z$	$x'yz'$	$x'yz$
$xy'z'$	$xy'z$	xyz'	xyz

- Each minterm is true for exactly one combination of inputs:

Minterm	Is true when…	Shorthand
$x'y'z'$	x=0, y=0, z=0	m_0
$x'y'z$	x=0, y=0, z=1	m_1
$x'yz'$	x=0, y=1, z=0	m_2
$x'yz$	x=0, y=1, z=1	m_3
$xy'z'$	x=1, y=0, z=0	m_4
$xy'z$	x=1, y=0, z=1	m_5
xyz'	x=1, y=1, z=0	m_6
xyz	x=1, y=1, z=1	m_7

Sum of Minterms Form

- Every function can be written as a sum of minterms, which is a special kind of sum of products form
- The sum of minterms form for any function is *unique*
- If you have a truth table for a function, you can write a sum of minterms expression just by picking out the rows of the table where the function output is 1.

x	y	z	f(x,y,z)	f'(x,y,z)
0	0	0	1	0
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	0	1
1	0	1	0	1
1	1	0	1	0
1	1	1	0	1

$$\begin{aligned} f &= x'y'z' + x'y'z + x'yz' + x'yz + xyz' \\ &= m_0 + m_1 + m_2 + m_3 + m_6 \\ &= \Sigma m(0,1,2,3,6) \end{aligned}$$

$$\begin{aligned} f' &= xy'z' + xy'z + xyz \\ &= m_4 + m_5 + m_7 \\ &= \Sigma m(4,5,7) \end{aligned}$$

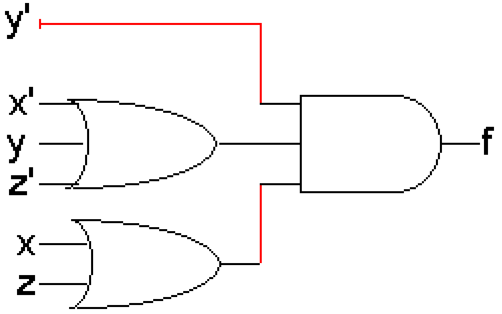
f' contains all the minterms not in f

The Dual Idea: Products of Sums

- A product of sums (POS) expression contains:
  - Only AND (product) operations at the "outermost" level
  - Each term must be a sum of literals

$$f(x,y,z) = y' (x' + y + z') (x + z)$$

- Product of sums expressions can be implemented with two-level circuits
  - literals and their complements at the "0th" level
  - *OR gates* at the first level
  - a single *AND gate* at the second level



- Compare this with sums of products

Maxterms

- A maxterm is a *sum* of literals, in which each input variable appears exactly once
- A function with n variables has 2^n maxterms
- The maxterms for a three-variable function f(x,y,z):

$x' + y' + z'$	$x' + y' + z$	$x' + y + z'$	$x' + y + z$
$x + y' + z'$	$x + y' + z$	$x + y + z'$	$x + y + z$

- Each maxterm is *false* for exactly one combination of inputs:

Maxterm	Is *false* when…	Shorthand
$x + y + z$	x=0, y=0, z=0	M_0
$x + y + z'$	x=0, y=0, z=1	M_1
$x + y' + z$	x=0, y=1, z=0	M_2
$x + y' + z'$	x=0, y=1, z=1	M_3
$x' + y + z$	x=1, y=0, z=0	M_4
$x' + y + z'$	x=1, y=0, z=1	M_5
$x' + y' + z$	x=1, y=1, z=0	M_6
$x' + y' + z'$	x=1, y=1, z=1	M_7

Product of Maxterms Form

- Every function can be written as a *unique* product of maxterms
- If you have a truth table for a function, you can write a product of maxterms expression by picking out the rows of the table where the function output is 0. (Be careful if you're writing the actual literals!)

x	y	z	f(x,y,z)	f'(x,y,z)
0	0	0	1	0
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	0	1
1	0	1	0	1
1	1	0	1	0
1	1	1	0	1

$$f = (x' + y + z)(x' + y + z')(x' + y' + z')$$
$$= M_4\, M_5\, M_7$$
$$= \prod M(4,5,7)$$

$$f' = (x + y + z)(x + y + z')(x + y' + z)$$
$$(x + y' + z')(x' + y' + z)$$
$$= M_0\, M_1\, M_2\, M_3\, M_6$$
$$= \prod M(0,1,2,3,6)$$

f' contains all the maxterms not in f

Minterms and maxterms are related

- Any minterm m_i is the *complement* of the corresponding maxterm M_i

Minterm	Shorthand	Maxterm	Shorthand
$x'y'z'$	m_0	$x + y + z$	M_0
$x'y'z$	m_1	$x + y + z'$	M_1
$x'yz'$	m_2	$x + y' + z$	M_2
$x'yz$	m_3	$x + y' + z'$	M_3
$xy'z'$	m_4	$x' + y + z$	M_4
$xy'z$	m_5	$x' + y + z'$	M_5
xyz'	m_6	$x' + y' + z$	M_6
xyz	m_7	$x' + y' + z'$	M_7

- For example, $m_4' = M_4$ because $(xy'z')' = x' + y + z$

Converting Between Standard Forms

- We can convert a sum of minterms to a product of maxterms

From before	f	$= \Sigma m(0,1,2,3,6)$
and	f'	$= \Sigma m(4,5,7)$
		$= m_4 + m_5 + m_7$
complementing	$(f')'$	$= (m_4 + m_5 + m_7)'$
so	f	$= m_4'\, m_5'\, m_7'$ [DeMorgan's law]
		$= M_4\, M_5\, M_7$ [By the previous page]
		$= \Pi\, M(4,5,7)$

- In general, just replace the minterms with maxterms, using maxterm numbers that don't appear in the sum of minterms:

$$f = \Sigma m(0,1,2,3,6)$$
$$= \Pi\, M(4,5,7)$$

- The same thing works for converting from a product of maxterms to a sum of minterms

Summary

- So far:
  - A bunch of Boolean algebra trickery for simplifying expressions and circuits
  - The algebra guarantees us that the simplified circuit is *equivalent* to the original one
  - Introducing some standard forms and terminology
- Next:
  - An alternative simplification method
  - We'll start using all this stuff to build and analyze bigger, more useful, circuits