Boolean Algebra

Digital Logic

School of Information Technology, KMUTT

Huntington Postulates

- 1a) closure w.r.t. + 1b) closure w.r.t. •
- 2a) $x + 0 = x$ 2b) $x \cdot 1 = x$
- 3a) $x + y = y + x$ 3b) $xy = yx$
  - commutative
- 4a) $x(y+z) = xy + xz$ 4b) $x+yz = (x+y)(x+z)$
  - distributive
- 5a) $x + x' = 1$ 5b) $x \cdot x' = 0$
- 6) Let B be a set of Boolean elements, there exist at least two elements $x, y \in B$ s.t. $x \neq y$

Duality

- Every algebraic expression deducible from the postulates of Boolean algebra remains valid if the operators and identity elements are interchanged.
  - AND ⇔ OR
  - 0 ⇔ 1
- Example:
  - Postulate 2a) $x + 0 = x$, thus 2b) $x \cdot 1 = x$
  - Postulate 4a) $x(y+z) = xy + xz$, thus by duality ➔ 4b) $x+yz = (x+y)(x+z)$

Other Basic Theorems

- Theorem 1: Idempotent
  (a) $x + x = x$ (b) $x \cdot x = x$
- Theorem 2
  (a) $x + 1 = 1$ (b) $x \cdot 0 = 0$
- Theorem 3: Involution → $(x')' = x$
- Theorem 4: Associative
  (a) $x+(y+z) = (x+y)+z$ (b) $x(yz) = (xy)z$
- Theorem 5: DeMorgan
  (a) $(x+y)' = x'y'$ (b) $(xy)' = x'+y'$
- Theorem 6: Absorption
  (a) $x + xy = x$ (b) $x(x+y) = x$

Boolean to Gate Mapping

- When a Boolean expression is implemented with logic gates
  - Each *term* requires a gate
  - Each *variable* in a term becomes an input to the gate
  - We call a variable within a term a "*literal*"
  - Example: $xy' + x'z$
    - Number of terms = ____________________
    - Number of variables = ____________________
    - Number of literals = ____________________

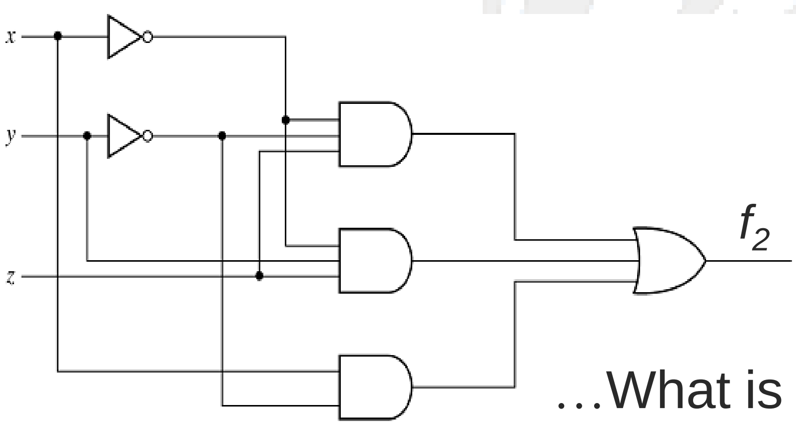
Simplifying Boolean Expressions

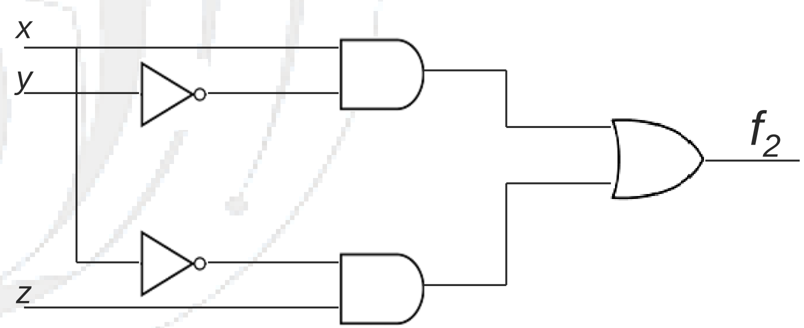
- Recall $f_2 = x'y'z + x'yz + xy'$, which can be simplified as

  $= x'z(y'+y) + xy'$ → Post. 4a)—distributive

  $= x'z(1) + xy'$ → Post. 5a)

  $= x'z + xy'$





…What is the physical implication?

Consensus Theorem

$$xy + x'z + yz = xy + x'z$$

- Given a pair of terms for which a variable appears in one term and its complement in another, the consensus term is formed by multiplying the two original terms together, leaving out the selected variable and its complement.
  - the consensus of ab and $a'c$ is bc
  - abd and $b'de' \rightarrow (ad)(de') \rightarrow ade'$
  - $ab'd$ and $a'bd' \rightarrow (b'd)(bd') \rightarrow 0$
- In using the consensus theorem to simplify Boolean expressions, *the consensus term is the eliminated term*

Simplify $a'b' + ac + bc' + b'c + ab$ (using Consensus Theorem)

Simplification by Adding Redundant Terms

- Often, redundant terms can be inserted into a Boolean expression so as to eliminate and/or combine with other terms—simplifying the expression as a result
- Redundant terms…
  - Adding xx'
  - Multiplying $(x+x')$
  - Adding consensus term
  - etc.

Simplify $wx + xy + x'z' + wy'z'$

$wx + xy + x'z' + wy'z'$

$= wx + xy + x'z' + wy'z' + \ldots\ldots\ldots$

→ by consensus theorem

$=$

$=$

$=$

$= wx + xy + x'z'$

Minterms and Maxterms

- Minterms—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
  - Each minterm consists of *n* variables with an AND operation
  - For *n* variables, there exist a total of 2^n possible minterms
- Maxterms—similar to minterms, but combined with an OR operation instead

Min/Maxterms for three 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

Note: 1. There are 8 minterms and maxterms (because $n = 3$)
2. Each variable in a minterm is primed when the corresponding bit is 0
3. Each variable in a maxterm is primed when the corresponding bit is 1
4. $m_i = (M_i)'$

Canonical Form

- Boolean functions expressed as
  - a sum of minterms, or
  - a product of maxterms

are said to be in *canonical form*

- To express a Boolean function in canonical form, simply
  - OR all minterms that produce a '1' in the function, or
  - AND all maxterms that produce a '0'

From the truth table, write f_1, f_2 in canonical form

Using minterms:

f_1=

f_2=

Using maxterms:

f_1=

f_2=

x	y	z	f_1	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

Minterm Expansion

Express $f(a,b,c) = a + b'c$ in a sum of minterms

a is missing b and c, thus

- $a = a(b+b') = ab + ab'$
- $ab(c+c')+ab'(c+c') = abc+abc'+ab'c+ab'c'$

$b'c$ is missing a, thus

- $b'c(a+a') = ab'c+a'b'c$

Answer:

$f = a'b'c+ab'c'+ab'c+abc'+abc$

$= (\qquad) = \sum(\qquad)$

Maxterm Expansion

- Express $f = xy + x'z$ in a product of maxterms
  - Distributive law → $a+bc = (a+b)(a+c)$
    - $xy+x'z \quad = (xy+x')(xy+z) \quad = (x'+x)(x'+y)(x+z)(y+z)$
      $\quad = (x'+y)(x+z)(y+z)$
  - Each OR term is missing one variable
    - $(x'+y) = (x'+y+zz') = (x'+y+z)(x'+y+z')$
    - $(x+z) =$
    - $(y+z) =$

Answer:

- $f = (x+y+z)(x+y'+z)(x'+y+z)(x'+y+z')$
  $= (\qquad) = \prod(\qquad)$

Conversion between Canonical Forms

For a Boolean function *f* expressed in one canonical form, the other canonical form may be obtained by:

- $m \Leftrightarrow M$ *or* $\sum \Leftrightarrow \prod$
- present indices ➔ missing indices

- Examples:
  - $f = m_1+m_4+m_5+m_6+m_7 = M_0M_2M_3$
  - $g = \prod(0,2,4,5) = \sum(1,3,6,7)$
  - $h(x,y) = \sum(0,1) = \prod(\qquad\qquad)$

Standard Forms

- A canonical form requires that

  1) the expression be written as a
     - sum of products, or
     - product of sums

  2) each product/sum term has the same number of literals, and each variable appears exactly once

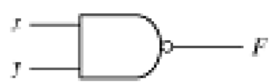
- When Requirement (2) does not hold true, a canonical form becomes a *standard form*

- Examples:
  - $f_1 = y' + xy + x'yz'$ ➔ sum of products (sop)
  - $f_2 = x(x'+z)(x'+y+z')$ ➔ product of sums (pos)
  - $f_3 = ab + c(d+e)$ → why is it not in a standard form?

Other Logic

- So far, we have focused on AND, OR, NOT
- Other logic operators also exist, e.g.
  - NAND, NOR, XOR, XNOR, Buffer
- NAND—AND followed by NOT (Inverter)

NAND

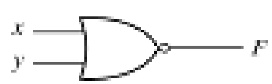


$F = (xy)'$

x	y	F
0	0	1
0	1	1
1	0	1
1	1	0

- NOR—OR followed by NOT (Inverter)

NOR



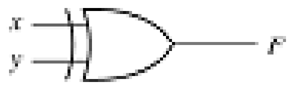
$F = (x + y)'$

x	y	F
0	0	1
0	1	0
1	0	0
1	1	0

XOR, XNOR, Buffer

- XOR—Exclusive OR

Exclusive-OR (XOR)

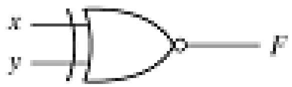


$F = xy' + x'y = x \oplus y$

x	y	F
0	0	0
0	1	1
1	0	1
1	1	0

- XNOR—Exclusive NOR or Equivalence

Exclusive-NOR or equivalence

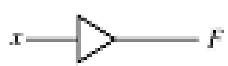


$F = xy + x'y' = (x \oplus y)'$

x	y	F
0	0	1
0	1	0
1	0	0
1	1	1

- Buffer—A transfer function

Buffer



$F = x$

x	F
0	0
1	1