CSE 205: Digital Logic Design

Boolean (Binary) Logic

- Deals with binary variables and binary logic functions
- Has two discrete values
  - 0, False, Open
  - 1, True, Close
- Three basic logical operations
  - AND (.); OR (+); NOT (‘)

- We need to define algebra for binary values
  - **Boolean Algebra:** Developed by George Boole in 1854

Boolean Algebra

- Why study Boolean Algebra?
  - To find the simplest circuit implementation with the smallest number of gates or wires.

- The simpler that we can make a Boolean function, the smaller the circuit that will result.
  - Simpler circuits are cheaper to build, consume less power, and run faster than complex circuits.

- With this in mind, we always want to reduce our Boolean functions to their simplest form.

ALGEBRA

- What is an algebra?
  - Mathematical system consisting of
    - Set of elements
    - Set of operators
    - Axioms or postulates: facts that can be taken as true; they do not require proof

Axiomatic definition of Boolean Algebra

- A Boolean algebra requires
  - A set of elements ***B***, consisting of two elements (0 and 1)
  - Two binary operations OR and AND
  - The axioms below must always be true

1.	$x + y \in B$	$x \bullet y \in B$	Closure
2.	$x + 0 = x$	$x \bullet 1 = x$	Identity
3.	$x + y = y + x$	$x \bullet y = y \bullet x$	Commutativity
4.	$x(y + z) = xy + xz$	$x + yz = (x + y)(x + z)$	Distributivity
5.	$x + x' = 1$	$x \bullet x' = 0$	Complement
6.	At least 2 elements: $x,y \in B$ such that $x \neq y$		Cardinality

Axiomatic definition of Boolean Algebra

- Based on axiom #5, we can develop a unary (one-argument) operation NOT

x	x'
0	1
1	0

Axiomatic definition of Boolean Algebra

- The distributive laws

x	y	z	$y+z$	$x \cdot (y+z)$	$x \cdot y$	$x \cdot z$	$(x \cdot y)+(x \cdot z)$
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

Duality principle

- If an expression is valid in Boolean algebra, the dual of that expression is also valid.

- To form the dual of an expression, replace all + operators with . operators, all . operators with + operators, all ones with zeros, and all zeros with ones.
- Form the dual of the expression

  $a + (bc) = (a + b)(a + c)$
- Following the replacement rules…

  $a(b + c) = ab + ac$

DUALITY PRINCIPLE

- The left and right columns of axioms are duals
  - exchange all ANDs with ORs, and 0s with 1s

1.	$x + y \in B$	$x \bullet y \in B$	Closure
2.	$x + 0 = x$	$x \bullet 1 = x$	Identity
3.	$x + y = y + x$	$x \bullet y = y \bullet x$	Commutativity
4.	$x(y + z) = xy + xz$	$x + yz = (x + y)(x + z)$	Distributivity
5.	$x + x' = 1$	$x \bullet x' = 0$	Complement
6.	At least 2 elements: $x,y \in B$ such that $x \neq y$		Cardinality

Basic theorems of Boolean Algebra

- In addition to the axioms, additional laws can be derived; they are called theorems of Boolean Algebra
- These theorems are useful in performing algebraic manipulations of Boolean expressions

1.	$x + x = x$	$x \bullet x = x$	Idempotency
2.	$x + 1 = 1$	$x \bullet 0 = 0$	
3.	$yx + x = x$	$(y + x) \bullet x = x$	Absorption
4.	$(x')' = x$		Involution
5.	$x + (y + z) = (x + y) + z$	$x(yz) = (xy)z$	Associative
6.	$(x + y)' = x'y'$	$(xy)' = x' + y'$	DeMorgan's

Proof of $x+x=x$

- We can only use Huntington postulates:

Post. 2: (a) $x+0=x$, (b) $x \cdot 1=x$
Post. 3: (a) $x+y=y+x$, (b) $x \cdot y=y \cdot x$
Post. 4: (a) $x(y+z) = xy+xz$,
(b) $x+yz = (x+y)(x+z)$
Post. 5: (a) $x+x'=1$, (b) $x \cdot x'=0$

- Show that $x+x=x$.

$$\begin{aligned} x+x &= (x+x)\cdot 1 && \text{by 2(b)} \\ &= (x+x)(x+x') && \text{by 5(a)} \\ &= x+xx' && \text{by 4(b)} \\ &= x+0 && \text{by 5(b)} \\ &= x && \text{by 2(a)} \end{aligned}$$

- We can now use Theorem 1(a) in future proofs

Proof of $x \cdot x = x$

- Similar to previous proof

Post. 2: (a) $x+0=x$, (b) $x \cdot 1=x$
Post. 3: (a) $x+y=y+x$, (b) $x \cdot y=y \cdot x$
Post. 4: (a) $x(y+z) = xy+xz$,
(b) $x+yz = (x+y)(x+z)$
Post. 5: (a) $x+x'=1$, (b) $x \cdot x'=0$
Th. 1: (a) $x+x=x$

- Show that $x \cdot x = x$.

$$
\begin{aligned}
x \cdot x &= xx+0 && \text{by 2(a)} \\
&= xx+xx' && \text{by 5(b)} \\
&= x(x+x') && \text{by 4(a)} \\
&= x \cdot 1 && \text{by 5(a)} \\
&= x && \text{by 2(b)}
\end{aligned}
$$

Proof of x+1=1

- Theorem 2(a): $x + 1 = 1$

  $x + 1 = 1 \cdot (x + 1)$ by 2(b)

  $=(x + x')(x + 1)$ 5(a)

  $= x + x'\,1$ 4(b)

  $= x + x'$ 2(b)

  $= 1$

Post. 2: (a) $x+0=x$, (b) $x\cdot 1=x$
Post. 3: (a) $x+y=y+x$, (b) $x\cdot y=y\cdot x$
Post. 4: (a) $x(y+z) = xy+xz$,
(b) $x+yz = (x+y)(x+z)$
Post. 5: (a) $x+x'=1$, (b) $x\cdot x'=0$
Th. 1: (a) $x+x=x$, (b) $x.x =x$

- Theorem 2(b): $x \cdot 0 = 0$ by duality
- Theorem 4: $(x')' = x$
  - Postulate 5 defines the complement of x, $x + x' = 1$ and $x\,x' = 0$
  - The complement of x' is x is also $(x')'$

Absorption Property (Covering)

Theorem 6(a): $x + xy = x$

$x + xy = x \cdot 1 + xy$ *by*	2(b)
$= x\,(1 + y)$	4(a)
$= x\,(y + 1)$	3(a)
$= x \cdot 1$	Th 2(a)
$= x$	2(b)

Post. 2:	(a) $x+0=x$,	(b) $x\cdot 1=x$
Post. 3:	(a) $x+y=y+x$,	(b) $x\cdot y=y\cdot x$
Post. 4:	(a) $x(y+z) = xy+xz$,	
	(b) $x+yz = (x+y)(x+z)$	
Post. 5:	(a) $x+x'=1$,	(b) $x\cdot x'=0$
Th. 1:	(a) $x+x=x$,	(b) $x.x =x$
Th. 2:	(a) $x + 1 = 1$,	(b) $x.0 =0$

- Theorem 6(b): $x\,(x + y) = x$ by duality
- By means of truth table (another way to proof)

Absorption Property (Covering)

Theorem 6(a): $x + xy = x$

x	y	xy	$x+xy$
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

DeMorgan's Theorem

- Theorem 5(a): $(x + y)' = x'y'$
- Theorem 5(b): $(xy)' = x' + y'$
- By means of truth table

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

Consensus Theorem

$$AB + A'C + BC = AB + A'C$$

- The consensus or resolvent of the terms AB and A'C is BC
- It is the conjunction of all the unique literals of the terms, excluding the literal that appears unnegated in one term and negated in the other

Consensus Theorem

1. $xy + x'z + yz = xy + x'z$
2. $(x+y)\cdot(x'+z)\cdot(y+z) = (x+y)\cdot(x'+z)$ -- (dual)

- **Proof:**

$$\begin{aligned} xy + x'z + yz &= xy + x'z + (x+x')yz \\ &= xy + x'z + xyz + x'yz \\ &= (xy + xyz) + (x'z + x'zy) \\ &= xy(1+z) + x'z\,(1+y) \\ &= xy + x'z \end{aligned}$$

Boolean Function

- A Boolean function expresses the logical relationship between binary variables and is evaluated by determining the binary value of the expression for all possible values of the variables.

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

- Some terminology, notation and precedence:
  - f is the name of the function.
  - (x,y,z) are the input variables, each representing 1 or 0.
  - A literal is a single variable within a term, in complemented or uncomplemented form. The function above has four literals: x, y', z, and x'.
  - NOT has the highest precedence, followed by AND, and then OR.

Boolean Function

- A Boolean function can be represented in a truth table.

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

$$\begin{aligned}
f(0,0,0) &= (0 + 1)0 + 1 &= 1\\
f(0,0,1) &= (0 + 1)1 + 1 &= 1\\
f(0,1,0) &= (0 + 0)0 + 1 &= 1\\
f(0,1,1) &= (0 + 0)1 + 1 &= 1\\
f(1,0,0) &= (1 + 1)0 + 0 &= 0\\
f(1,0,1) &= (1 + 1)1 + 0 &= 1\\
f(1,1,0) &= (1 + 0)0 + 0 &= 0\\
f(1,1,1) &= (1 + 0)1 + 0 &= 1
\end{aligned}$$

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

Complement of a Function

- The complement of a function always outputs 0 where the original function outputted 1, and 1 where the original produced 0.

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

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	0
1	1	0	0
1	1	1	1

$\Rightarrow$

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	1
1	1	1	0

Gate Implementation of a Function

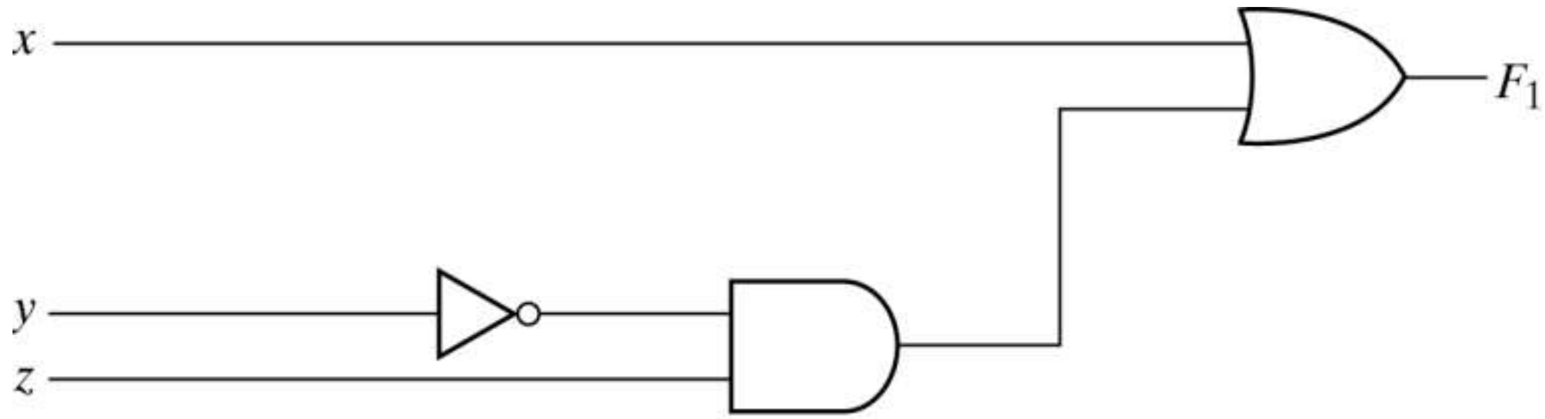
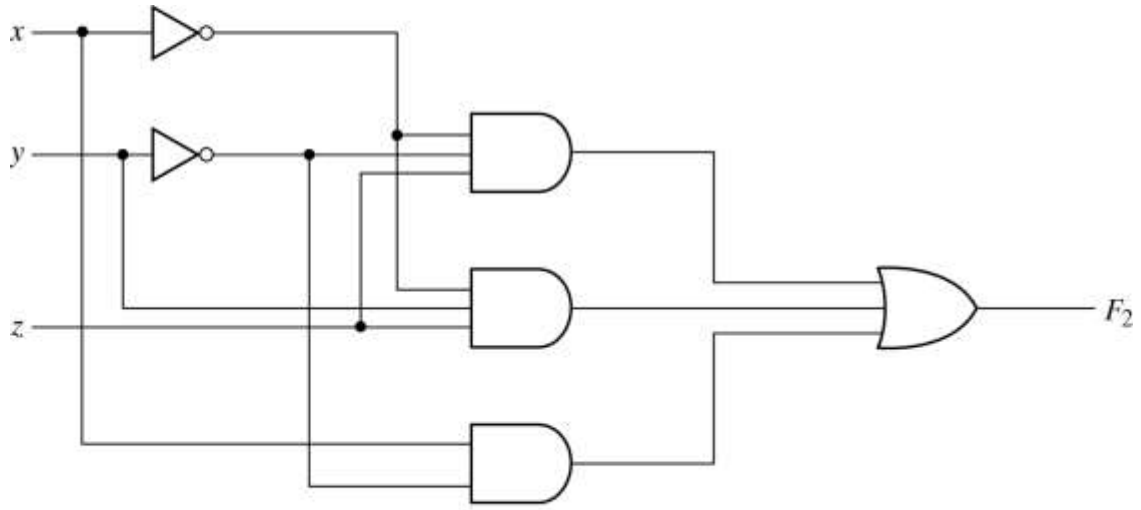
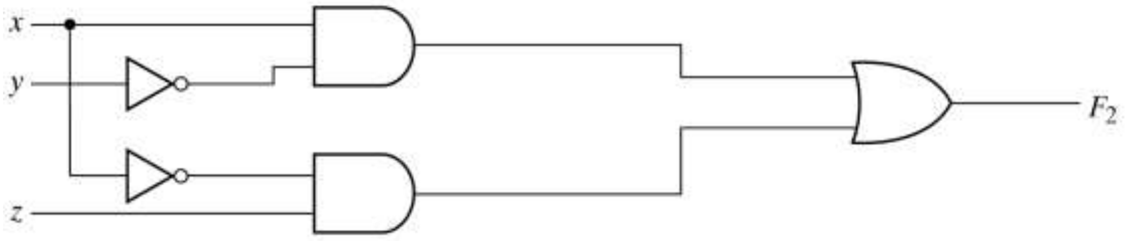


Fig. 2-1 Gate implementation of $F_1 = x + y'z$

Gate Implementation of a Function



(a) $F_2 = x'y'z + x'yz + xy'$



(b) $F_2 = xy' + x'z$

Fig. 2-2 Implementation of Boolean function F_2 with gates

PRACTICE

- Simplify the following Boolean expressions to a minimum number of literals:
  - $xyz + x'y + xyz'$
  - $(A + B)'(A' + B')'$
- $xyz + x'y + xyz' = xy(z + z') + x'y = xy + x'y = y$
- $(A + B)'(A' + B')' = (A'B')(A\ B) = (A'B')(BA) = 0$

Complement of a Function

- Applying DeMorgan's theorems:

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

$f'(x,y,z) = (x(y'z' + yz))'$ [complement both sides]
$= x' + (y'z' + yz)'$ [because $(xy)' = x' + y'$]
$= x' + (y'z')'\,(yz)'$ [because $(x + y)' = x'y'$]
$= x' + (y + z)(y' + z')$ [because $(xy)' = x' + y'$, twice]

Complement of a Function

- By taking the dual of *f* and complementing 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 $f'(x,y,z) = x' + (y + z)(y' + z')$

Practice

- Find the complement of the following expressions:
  - $xy' + x'y$
  - $[(x' + y + z')(x + y')(x + z)]$
- $F' = (xy' + x'y)'$

  $= (xy')'(x'y)'$

  $= (x' + y)(x + y')$

  $= xy + x'y$‘
- $F' = [(x' + y + z')(x + y')(x + z)]'$

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

  $= xy'z + x'y + x'z'$

Minterms

- A minterm is an AND term in which every variable or its complement in a function occurs once.
  - F(x,y) has 4 minterms $x'y'$, $x'y$, xy', xy

- An *n variable function has* 2^n *valid minterms*

- A minterm equals 1 at exactly one input combination and is equal to 0 otherwise
  - Example: $x'y'z' = 1$ only when x=0, y=0, z=0

Maxterms

- A maxterm is an OR term in which every variable or its complement in a function occurs once
  - F(x,y) has 4 maxterms $x'+y'$, $x'+y$, $x+y'$, $x+y$

- An n variable function has 2^n valid maxterms

- A maxterm equals 0 at exactly one input combination and is equal to 1 otherwise
  - Example: $(x+y+z) = 0$ only when $x=0$, $y=0$, $z=0$

Example: Three Binary Variables

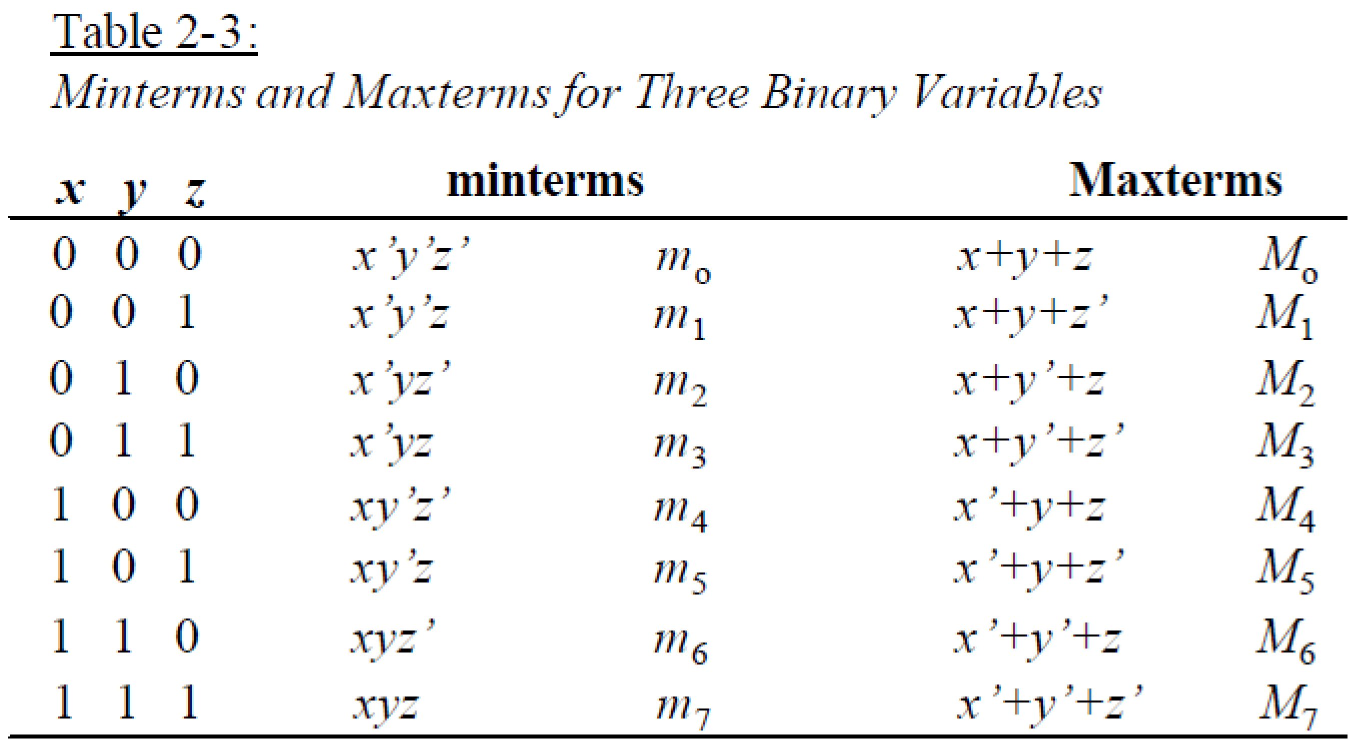


Table 2-3:

Minterms and Maxterms for Three Binary Variables

x	y	z	minterms		Maxterms	
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

Canonical Form

- Any boolean function that is expressed as a **sum of minterms** or as a **product of maxterms** is said to be in its canonical form.

Sum of Minterms

- A Boolean function can be expressed algebraically from a given truth table
  - by forming a minterm for each combination of the variables that produces a 1 in the function and
  - then taking the OR of all those terms.

Sum of Minterms

x	y	z	F_1	F_2
0	0	0	0	1
0	0	1	1	0
0	1	0	0	1
0	1	1	0	1
1	0	0	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

- $F_1(x, y, z) = \sum(1,4,5,6,7) = m_1 + m_4 + m_5 + m_6 + m_7$

  $= x'y'z + xy'z' + xy'z + xyz' + xyz$

Sum of Minterms: Example

- $F = x + yz$
  $= x(y + y')(z + z') + (x + x')yz$
  $= xyz + xyz' + xy'z + xy'z' + xyz + x'yz$
  $= x'yz + xy'z' + xy'z + xyz' + xyz$
  $= \Sigma(3,4,5,6,7)$
- Or convert the expression into truth-table and then read the minterms from the table

Product of Maxterms

- A Boolean function can be expressed algebraically from a given truth table
  - by forming a maxterm for each combination of the variables that produces a 0 in the function, and
  - then taking the AND of all those maxterms.

Product of Maxterms

x	y	z	F_1	F_2
0	0	0	$\boxed{0}$	1
0	0	1	1	0
0	1	0	$\boxed{0}$	1
0	1	1	$\boxed{0}$	1
1	0	0	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

- $$\begin{aligned} F_1(x, y, z) &= \prod(0,2,3) = M_0 \times M_2 \times M_3 \\ &= (x + y + z)(x + y'+z)(x + y'+z') \end{aligned}$$

Product of Maxterms: Example

- $F = xy + x'z = (xy+x')(xy+z)$
  $= (x+x')(y+x')(x+z)(y+z) = (x'+y)(x+z)(y+z)$

  $x'+y = x'+y+zz' = (x'+y+z)(x'+y+z')$
  $x+z = x+z+yy' = (x+y+z)(x+y'+z)$
  $y+z = y+z+xx' = (x+y+z)(x'+y+z)$

  $x+yz = (x+y)(x+z)$

  $F = (x+y+z)(x+y'+z)(x'+y+z)(x'+y+z')$
  $= M_0 M_2 M_4 M_5 = \prod(0, 2, 4, 5)$

- Or convert the expression into truth-table and then read the maxterms from the table

Conversion between Canonical Forms

- Conversion between minterms and maxterms

  $m_0 = x'y'z' = (x+y+z)' = (M_0)'$

  - In general, $m_i = (M_i)'$

- Sum of minterms → Product of maxterms:

  $f = \Sigma(0,1,2,3,6)$

  $f' = \Sigma(4,5,7) = m_4 + m_5 + m_7$

  $(f')' = (m_4 + m_5 + m_7)'$

  $f = m_4' \, m_5' \, m_7'$ [DeMorgan's law]

  $= M_4 \, M_5 \, M_7 = \prod(4,5,7)$

Conversion between Canonical Forms

- In general, to convert from one canonical form to another, interchange the symbols Σ and ∏ and list those numbers missing from the original form.

  - *Example:* $f = \Sigma(0,1,2,3,6) = \prod(4,5,7)$

Practice

- Express the following function (four variables: *A*, *B*, *C*, *D*) as a sum of minterms and as a product of maxterms:
  - $F = B'D + A'D + BD$

- $F = \sum(1, 3, 5, 7, 9, 11, 13, 15)$

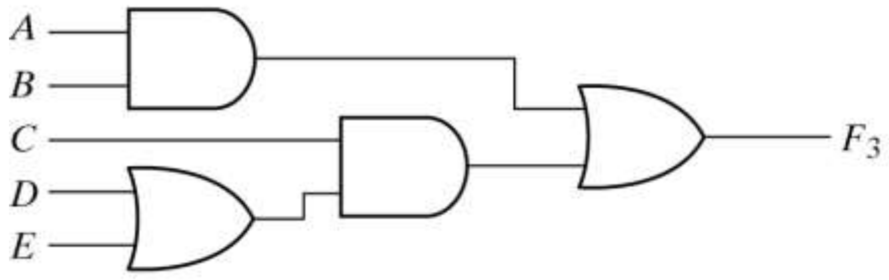
  $= \prod(0, 2, 4, 6, 8, 10, 12, 14)$

Standard Form

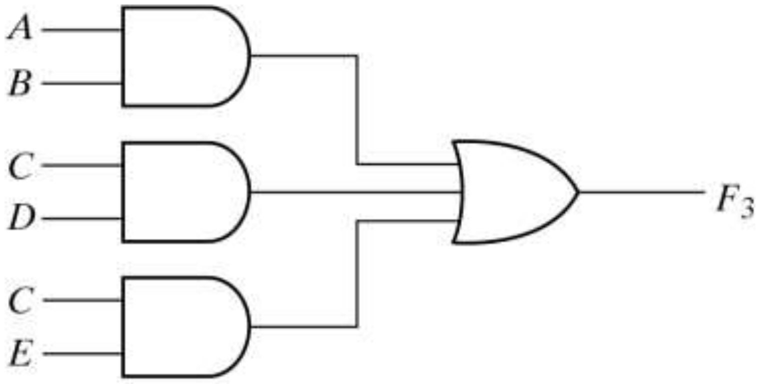
- Any boolean function that is expressed as a **sum of products (SOP)** or as a **product of sums (POS)**, where each product-term or sum-term may contain one, two, or any number of variables, is said to be in its **standard form**.

- SOP: $f(x,y,z) = xy + x'yz + xy'z$
- POS: $f(x,y,z) = (x' + y')(x + y' + z')(x' + y + z')$

Non-Standard and Standard Forms



(a) $AB + C(D + E)$



(b) $AB + CD + CE$

Fig. 2-4 Three- and Two-Level implementation

Practice

- Convert the following expression into sum of products and product of sums:
  - $(AB + C)(B + C'D)$

- $(AB + C)(B + C'D)$

  $= AB + BC + ABC'D + CC'D$

  $= AB(1 + C'D) + BC$

  $= AB + BC$ (SOP form)

  $= B(A + C)$ (POS form)

Canonical and Standard Forms

- Canonical forms
  - Sum of minterms (SOM)
  - Product of maxterms (POM)
- Standard forms (may use less gates)
  - Sum of products (SOP)
  - Product of sums (POS)
- $F = ab + a'$ (already **sum of products:SOP)**
- $F = ab + a'(b+b')$ (expanding term)
- $F = ab + a'b + a'b'$ (it is **canonical form:SOM)**

Other Logic Operations

- 2^n rows in the truth table of n binary variables.
- 2^{2n} functions for n binary variables.
- 16 functions of two binary variables.

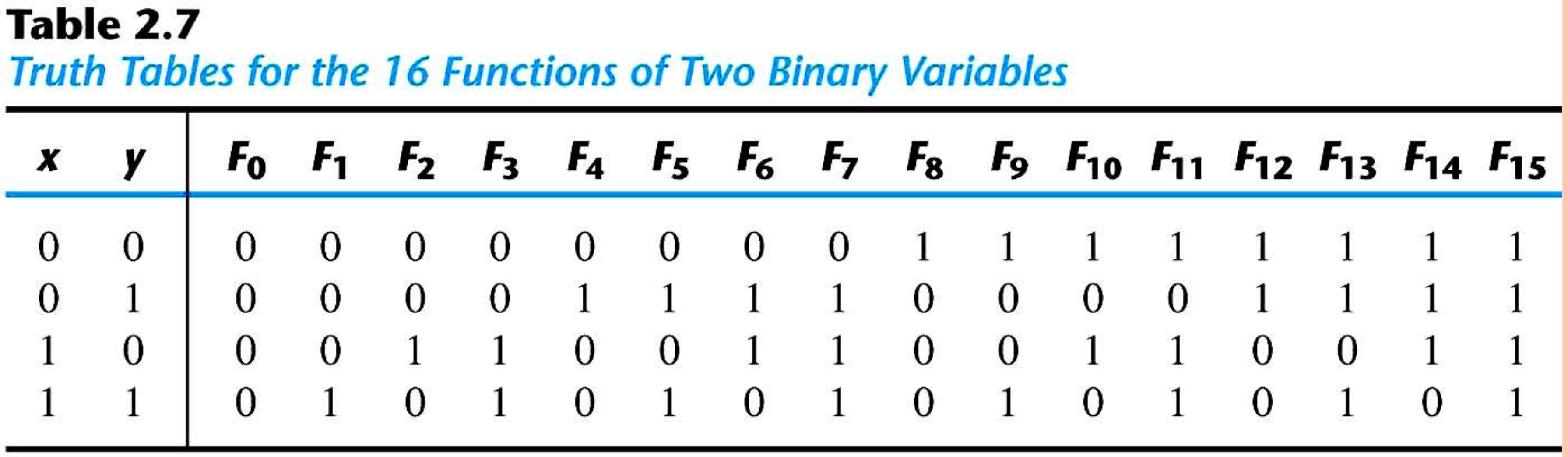


Table 2.7
Truth Tables for the 16 Functions of Two Binary Variables

x	y	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Boolean Expressions

Table 2.8
Boolean Expressions for the 16 Functions of Two Variables

Boolean Functions	Operator Symbol	Name	Comments
$F_0 = 0$		Null	Binary constant 0
$F_1 = xy$	$x \cdot y$	AND	x and y
$F_2 = xy'$	x/y	Inhibition	x, but not y
$F_3 = x$		Transfer	x
$F_4 = x'y$	y/x	Inhibition	y, but not x
$F_5 = y$		Transfer	y
$F_6 = xy' + x'y$	$x \oplus y$	Exclusive-OR	x or y, but not both
$F_7 = x + y$	$x + y$	OR	x or y
$F_8 = (x + y)'$	$x \downarrow y$	NOR	Not-OR
$F_9 = xy + x'y'$	$(x \oplus y)'$	Equivalence	x equals y
$F_{10} = y'$	y'	Complement	Not y
$F_{11} = x + y'$	$x \subset y$	Implication	If y, then x
$F_{12} = x'$	x'	Complement	Not x
$F_{13} = x' + y$	$x \supset y$	Implication	If x, then y
$F_{14} = (xy)'$	$x \uparrow y$	NAND	Not-AND
$F_{15} = 1$		Identity	Binary constant 1

Digital Logic Gates

- Boolean expression: AND, OR and NOT operations
- Constructing gates of other logic operations
  - The feasibility and economy;
  - The possibility of extending gate's inputs;
  - The basic properties of the binary operations (commutative and associative);
  - The ability of the gate to implement Boolean functions.

Summary of Logic Gates

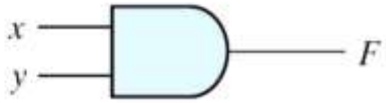
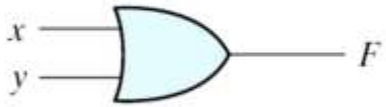
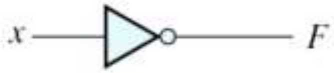
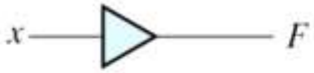
Name	Graphic symbol	Algebraic function	Truth table
AND		$F = xy$	$\begin{array}{cc\|c} x & y & F \\ \hline 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{array}$
OR		$F = x + y$	$\begin{array}{cc\|c} x & y & F \\ \hline 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{array}$
Inverter		$F = x'$	$\begin{array}{c\|c} x & F \\ \hline 0 & 1 \\ 1 & 0 \end{array}$
Buffer		$F = x$	$\begin{array}{c\|c} x & F \\ \hline 0 & 0 \\ 1 & 1 \end{array}$

Figure 2.5 Digital logic gates

Summary of Logic Gates

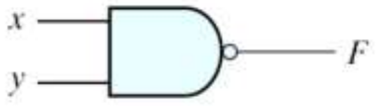
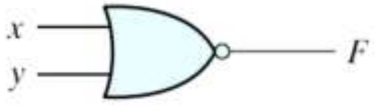
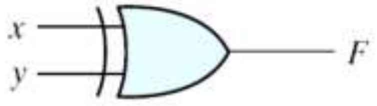
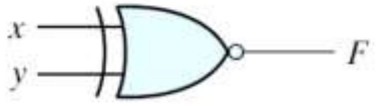
			x	y	F
NAND		$F = (xy)'$	0	0	1
			0	1	1
			1	0	1
			1	1	0
NOR		$F = (x + y)'$	0	0	1
			0	1	0
			1	0	0
			1	1	0
Exclusive-OR (XOR)		$F = xy' + x'y$ $= x \oplus y$	0	0	0
			0	1	1
			1	0	1
			1	1	0
Exclusive-NOR or equivalence		$F = xy + x'y'$ $= (x \oplus y)'$	0	0	1
			0	1	0
			1	0	0
			1	1	1

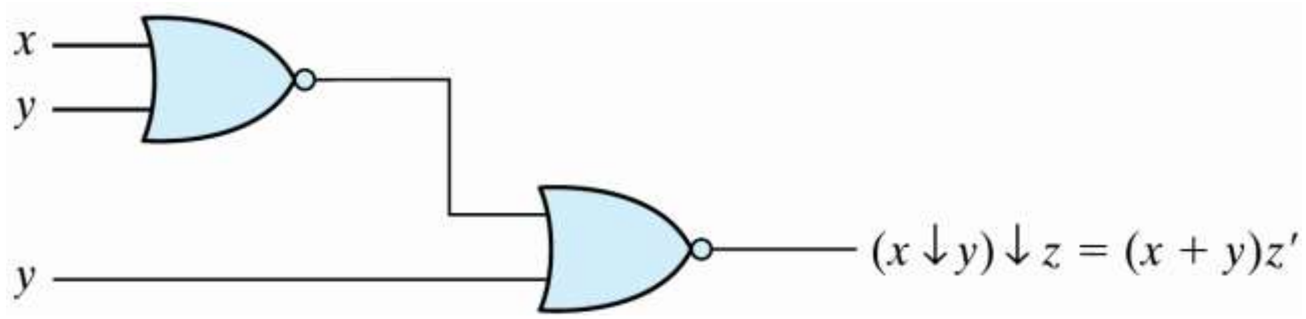
Figure 2.5 Digital logic gates

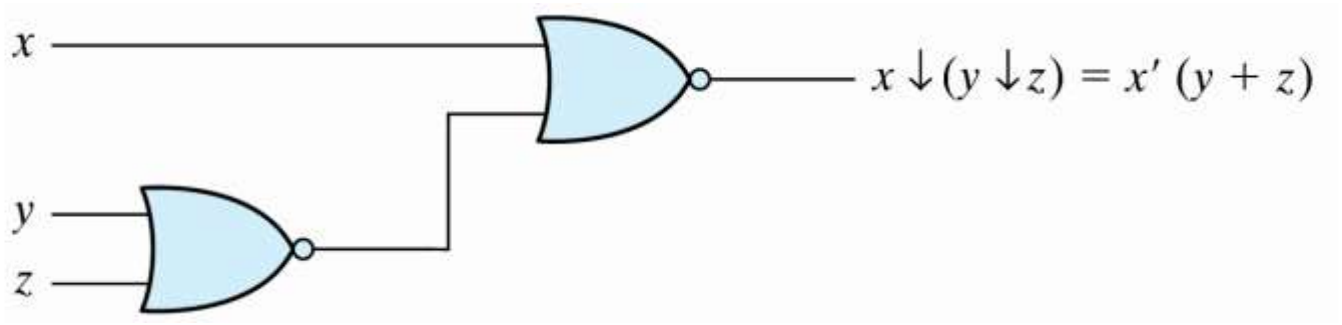
Multiple Inputs

- Extension to multiple inputs
  - A gate can be extended to multiple inputs.
    - If its binary operation is commutative and associative.
  - AND and OR are commutative and associative.
    - OR
      - $x+y = y+x$
      - $(x+y)+z = x+(y+z) = x+y+z$
    - AND
      - $xy = yx$
      - $(x\ y)z = x(y\ z) = x\ y\ z$

Multiple Inputs

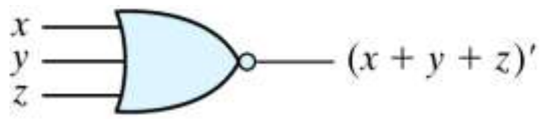
- NAND and NOR are commutative but not associative → they are not extendable.



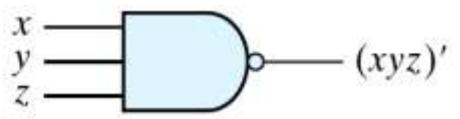


Multiple Inputs

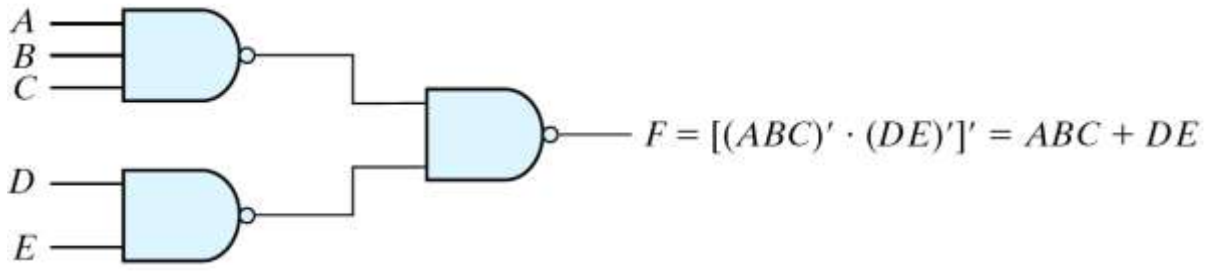
- Multiple NOR = a complement of OR gate, Multiple NAND = a complement of AND.
- The cascaded NAND operations = sum of products.
- The cascaded NOR operations = product of sums.



(a) 3-input NOR gate



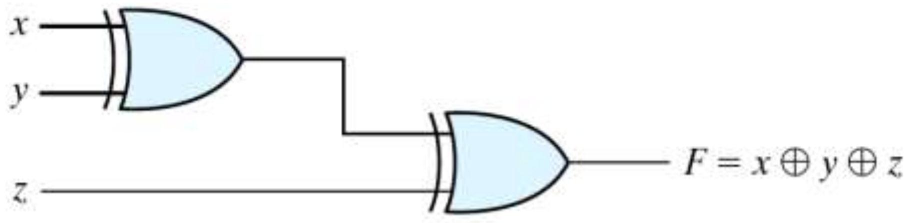
(b) 3-input NAND gate



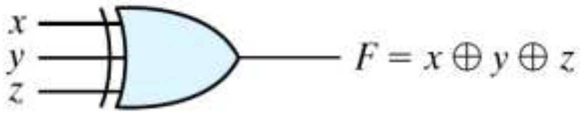
(c) Cascaded NAND gates

Multiple Inputs

- The XOR and XNOR gates are commutative and associative.
- Multiple-input XOR gates are uncommon
- XOR is an odd function: it is equal to 1 if the inputs variables have an odd number of 1's.



(a) Using 2-input gates



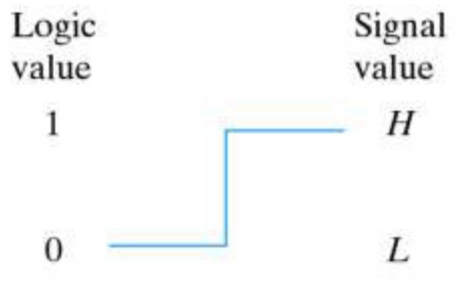
(b) 3-input gate

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

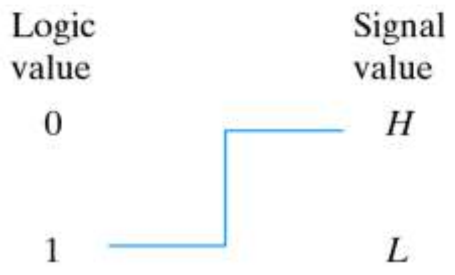
(c) Truth table

Positive and Negative Logic

- Positive and Negative Logic
  - Two signal values <=> two logic values
  - Positive logic: H=1; L=0
  - Negative logic: H=0; L=1
- Consider a TTL gate
  - A positive logic AND gate
  - A negative logic OR gate
  - The positive logic is used in this book



(a) Positive logic

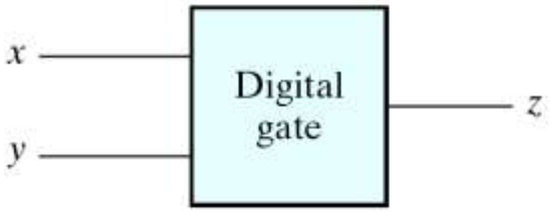


(b) Negative logic

Positive and Negative Logic

x	y	z
L	L	L
L	H	L
H	L	L
H	H	H

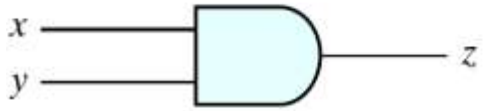
(a) Truth table with H and L



(b) Gate block diagram

x	y	z
0	0	0
0	1	0
1	0	0
1	1	1

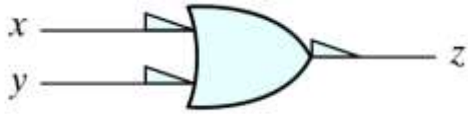
(c) Truth table for positive logic



(d) Positive logic AND gate

x	y	z
1	1	1
1	0	1
0	1	1
0	0	0

(e) Truth table for negative logic



(f) Negative logic OR gate

Syllabus

- Chapter 2 (Excluding Section 2.9)