Logo

# Discrete Mathematics

Dr. Han Huang

South China University of Technology

Logo

# Boolean Algebra

**Section 1.6**

# What is Boolean Algebra?

- A minor generalization of propositional logic.
  - In general, an *algebra* is any mathematical structure satisfying certain standard algebraic axioms.
    - Such as associative/commutative/transitive laws, *etc.*
  - General theorems that are proved about an algebra then apply to *any* structure satisfying these axioms.
- *Boolean algebra* just generalizes the rules of propositional logic to sets other than **{T,F}**.
  - *E.g.*, to the set {0,1} of base-2 digits, or the set $\{V_L, V_H\}$ of low and high voltage levels in a circuit.
- We will see that this algebraic perspective lends itself to the design of *digital logic circuits*.

Claude Shannon's Master's thesis!

Logo

# Example

我们可以把一个计算机看作一个代数系统。

设此计算机的字长是 32 位。有定点加、减、乘、除及逻辑加、逻辑乘等多种运算指令，则此代数系统的集合 S 是由 $2^{32}$ 个二进制数组成。其运算就是上述的运算，而这些运算是封闭的，因此构成一个代数系统。

# Boolean Algebra

- Sections:
  - § 1 – Boolean Functions
  - § 2 – Representing Boolean Functions
  - § 3 – Logic Gates
  - § 4 – Minimization of Circuits

Logo

# Boolean Functions

- Boolean complement, sum, product.
- Boolean expressions and functions.
- Boolean algebra identities.
- Duality.
- Abstract definition of a Boolean algebra.

# Complement, Sum, Product

- Correspond to logical NOT, OR, and AND.
- We will denote the two logic values as **0**:≡**F** and **1**:≡**T**, instead of **False** and **True**.
  - Using numbers encourages algebraic thinking.
- New, more algebraic-looking notation for the most common Boolean operators:

$$\bar{x} :\equiv \neg x \qquad x \cdot y :\equiv x \wedge y \qquad x + y :\equiv x \vee y$$

Precedence order→

# Boolean Functions

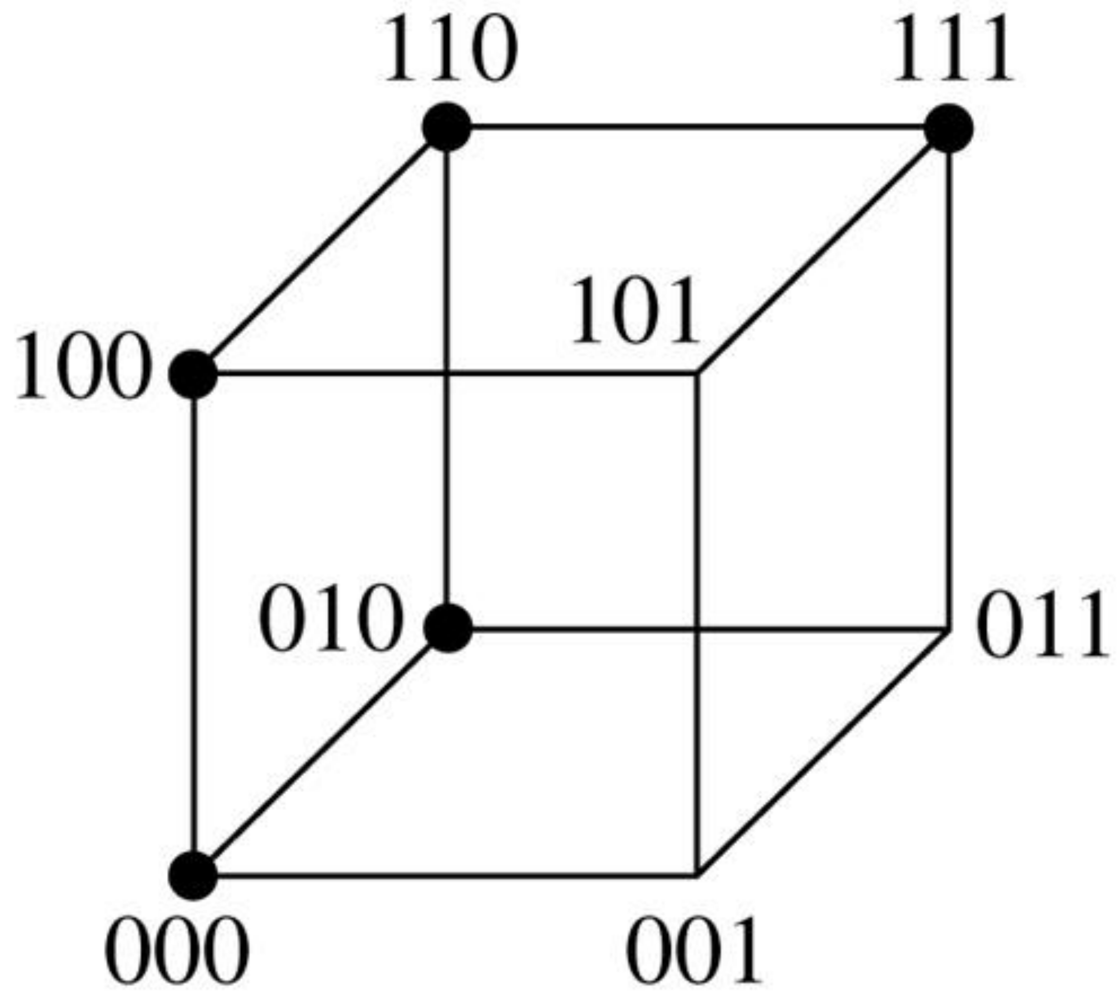
- Let $B = \{\mathbf{0}, \mathbf{1}\}$, the set of Boolean values.
- For all $n \in \mathbf{Z}^+$, any function $f: B^n \rightarrow B$ is called a *Boolean function of degree n*.
- There are $2^{2^n}$ (wow!) distinct Boolean functions of degree $n$.
  - B/c $\exists$ $2^n$ rows in truth table, w. 0 or 1 in each.

| Degree | How many | Degree | How many |
|---|---|---|---|
| 0 | 2 | 4 | 65,536 |
| 1 | 4 | 5 | 4,294,967,296 |
| 2 | 16 | 6 | 18,446,744,073,709,551,616. |
| 3 | 256 | | |

Logo

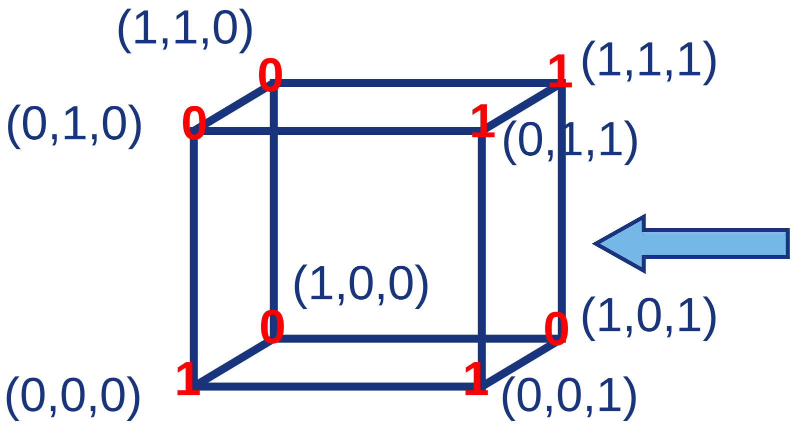
# Boolean Expressions

- Let $x_1, \ldots, x_n$ be $n$ different Boolean variables.
  - $n$ may be as large as desired.
- A *Boolean expression* (recursive definition) is a string of one of the following forms:
  - Base cases: **0**, **1**, $x_1, \ldots,$ or $x_n$.
  - Recursive cases: $\overline{E_1}$, $(E_1E_2)$, or $(E_1+E_2)$, where $E_1$ and $E_2$ are Boolean expressions.
- A Boolean expression represents a Boolean function.
  - Furthermore, *every* Boolean function (of a given degree) can be represented by a Boolean expression.


110
111
101
100
010
011
000
001

# Hypercube Representation

- A Boolean function of degree *n* can be represented by an *n*-cube (hypercube) with the corresponding function value at each vertex.




(*a*, *b*, *c*)

$$\overline{a+b}+bc$$

## Boolean equivalents, operations on Boolean expressions

- Two Boolean expressions $e_1$ and $e_2$ that represent the exact *same* function $f$ are called *equivalent.* We write $e_1 \Leftrightarrow e_2$, or just $e_1 = e_2$.
  - Implicitly, the two expressions have the same value for *all* values of the free variables appearing in $e_1$ and $e_2$.
- The operators $\overline{\phantom{x}}$, $+$, and $\cdot$ can be extended from operating on expressions to operating on the functions that they represent, in the obvious way.

Logo

# Some popular Boolean identities

- Double complement:
  $x = x$
- Idempotent laws:
  $x + x = x$, $x \cdot x = x$
- Identity laws:
  $x + \mathbf{0} = x$, $x \cdot \mathbf{1} = x$
- Domination laws:
  $x + \mathbf{1} = \mathbf{1}$, $x \cdot \mathbf{0} = \mathbf{0}$
- Commutative laws:
  $x + y = y + x$, $x \cdot y = y \cdot x$
- Associative laws:
  $x + (y + z) = (x + y) + z$
  $x \cdot (y \cdot z) = (x \cdot y) \cdot z$
- Distributive laws:
  $x + y{\cdot}z = (x + y){\cdot}(x + z)$
  $x \cdot (y + z) = x{\cdot}y + x{\cdot}z$
- De Morgan's laws:
  $(x \cdot y) = x + y$, $(x + y)$
- Absorption laws:
  $x + x{\cdot}y = x$, $x \cdot (x + y) = x$

← Not true in ordinary algebras.

also, the Unit Property: $x + x = \mathbf{1}$ and Zero Property: $x \cdot x = \mathbf{0}$

# Duality

- The *dual* $e^d$ of a Boolean expression $e$ representing function $f$ is obtained by exchanging $+$ with $\cdot$, and $0$ with $1$ in $e$.
  - The function represented by $e^d$ is denoted $f^d$.
- **Duality principle:** If $e_1 \Leftrightarrow e_2$ then $e_1^d \Leftrightarrow e_2^d$.
  - **Example:** The equivalence $x(x+y) = x$ implies (and is implied by) $x + xy = x$.

# Boolean Algebra, in the abstract

- A general *Boolean algebra* is *any* set $B$ having elements $\mathbf{0}$, $\mathbf{1}$, two binary operators $\wedge$,$\vee$, and a unary operator $\neg$ that satisfies the following laws:
  - Identity laws: $x \vee \mathbf{0} = x$, $x \wedge \mathbf{1} = x$
  - Complement laws: $x \vee \neg x = \mathbf{1}$, $x \wedge \neg x = \mathbf{0}$
  - Associative laws: $(x\vee y)\vee z = x\vee(y\vee z)$, $(x\wedge y)\wedge z = x\wedge(y\wedge z)$
  - Commutative laws: $x \vee y = y \vee x$, $x \wedge y = y \wedge x$
  - Distributive laws: $x\vee(y\wedge z) = (x\vee y)\wedge(x\vee z)$, $x\wedge(y\vee z)=(x\wedge y)\vee(x\wedge z)$.

Note that *B* may generally have other elements besides **0**, **1**, and we have not fully defined any of the operators!

# More about Boolean algebras

- *Any* Boolean algebra can be proven to satisfy all the theorems of "ordinary" Boolean algebra!
- An example of another Boolean algebra:
  - For any set $U$, let $B = 2^U$, $\mathbf{0}=\varnothing$, $\mathbf{1}=U$, $\vee=\cup$, $\wedge=\cap$, and $\neg=\overline{\phantom{x}}$ (set complement).
    - Then, $(B,\mathbf{0},\mathbf{1},\wedge,\vee,\neg)$ is a Boolean algebra!
- Boolean algebras can also be defined in terms of *lattices* (in chapter 7, though we skipped it).
  - A poset where every pair $x$,$y$ has a lub and a glb.
  - A complemented, distributed lattice is a Boolean alg.

# Representing Boolean Functions

- Sum-of-products Expansions
  - A.k.a. Disjunctive Normal Form (DNF)
- Product-of-sums Expansions
  - A.k.a. Conjunctive Normal Form (CNF)
- Functional Completeness
  - Minimal functionally complete sets of operators.

# Sum-of-Products Expansions

- **Theorem:** Any Boolean function can be represented as a sum of products of variables and their complements.
  - **Proof:** By construction from the function's truth table. For each row that is 1, include a term in the sum that is a product representing the condition that the variables have the values given for that row.

Show an example on the board.

# Literals, Minterms, DNF

- A *literal* is a Boolean variable or its complement.
- A *minterm* of Boolean variables $x_1,\ldots,x_n$ is a Boolean product of $n$ literals $y_1\ldots y_n$, where $y_i$ is either the literal $x_i$ or its complement $x_i$.
  - Note that at most one minterm can have the value 1.
- The *disjunctive normal form* (DNF) of a degree-$n$ Boolean function $f$ is the unique sum of minterms of the variables $x_1,\ldots,x_n$ that represents $f$.
  - A.k.a. the sum-of-products expansion of $f$.

Logo

# Conjunctive Normal Form

- A *maxterm* is a sum of literals.
- CNF is a *product-of-maxterms* representation.
- To find the CNF representation for $f$,
- take the DNF representation for complement $\neg f$,

$$\neg f = \sum_i \prod_j y_{i,j}$$

- and then complement both sides & apply DeMorgan's laws to get:

$$f = \prod_i \sum_j \neg y_{i,j}$$

Can also get CNF more directly, using the 0 rows of the truth table.

# Functional Completeness

- Since every Boolean function can be expressed in terms of $\cdot, +, \overline{\phantom{x}}$, we say that the set of operators $\{\cdot, +, \overline{\phantom{x}}\}$ is *functionally complete*.
- There are smaller sets of operators that are also functionally complete.
  - We can eliminate either $\cdot$ or $+$ using DeMorgan's law.
- NAND $|$ and NOR $\downarrow$ are also functionally complete, each by itself (as a singleton set).
  - *E.g.*, $\neg x = x|x$, and $xy = (x|y)|(x|y)$.

# Reversible Boolean Logic

- A *reversible* Boolean function of degree $n$ is a bijective function $f{:}B^n \leftrightarrow B^n$.
  - Also corresponds to a permutation of $B^n$.
- Reversible unary and binary Boolean operators are bijective operators on $B$ and $B^2$, respectively.
  - Unary $f{:}B \leftrightarrow B$, binary $f{:}B^2 \leftrightarrow B^2$.
  - It turns out that *no* set of reversible unary and binary Boolean operators is functionally complete!
  - However, there are many ternary reversible operators that are functionally complete, even as singletons.

# A little Quantum Logic

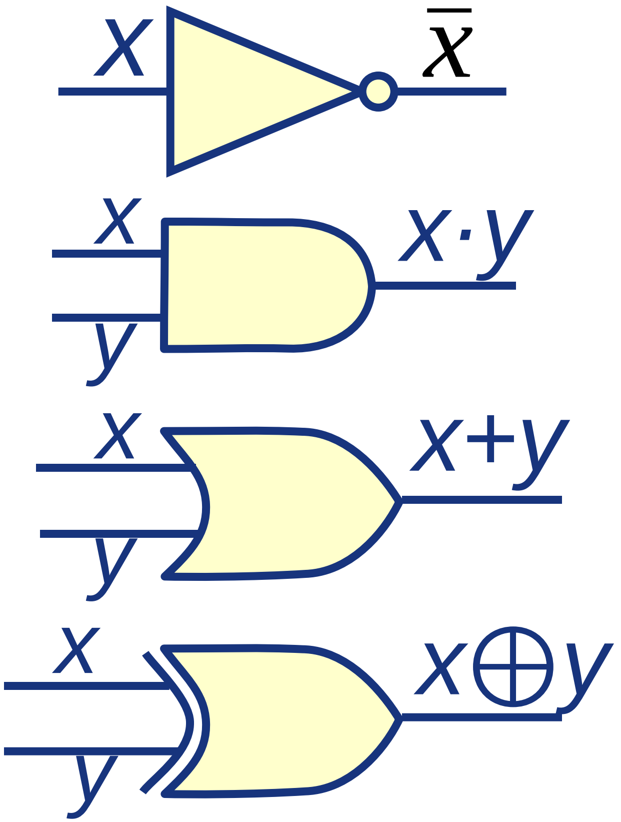
- A *quantum Boolean function* is a bijective and linear function $f{:}\mathbf{C}^{2^n}\leftrightarrow\mathbf{C}^{2^n}$.
  - That is, it maps vectors of $2^n$ complex numbers (one for each $n$-bit string of Boolean values) reversibly and linearly.
  - Any reversible Boolean function corresponds to a quantum Boolean function where a string in $B^n$ is represented by $c=1$ for that string, $c=0$ for all others.
- Any quantum Boolean function can be built out of *quantum operators* operating on just **C** and $\mathbf{C}^2$.
  - Quantization removes the need for ternary gates!

# Logic Gates

- Inverter, Or, And gate symbols.
- Multi-input gates.
- Logic circuits and examples.
- Adders, “half,” “full,” and *n*-bit.

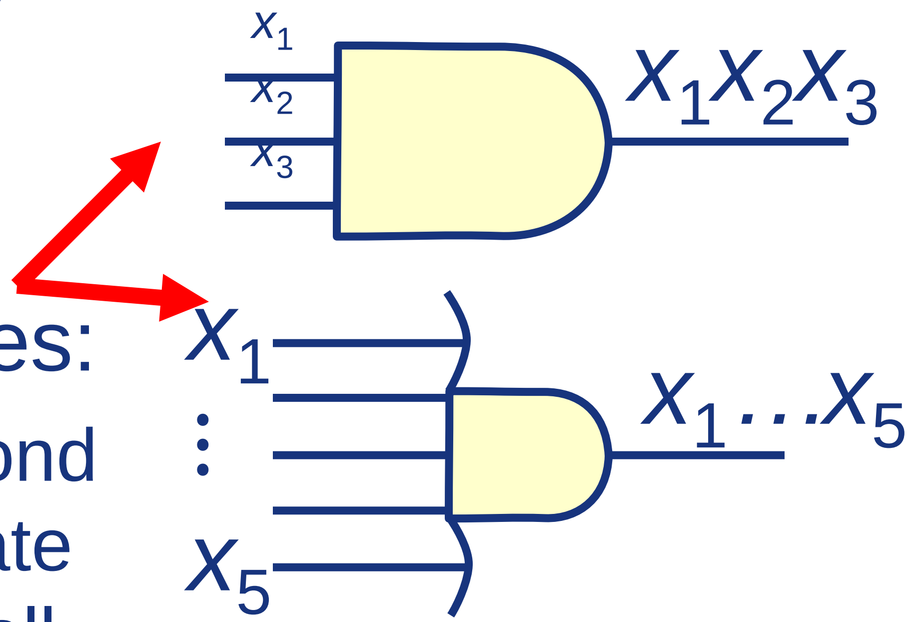
# Logic Gate Symbols

- Inverter (logical NOT, Boolean complement).
- AND gate (Boolean product).
- OR gate (Boolean sum).
- XOR gate (exclusive-OR, sum mod 2).



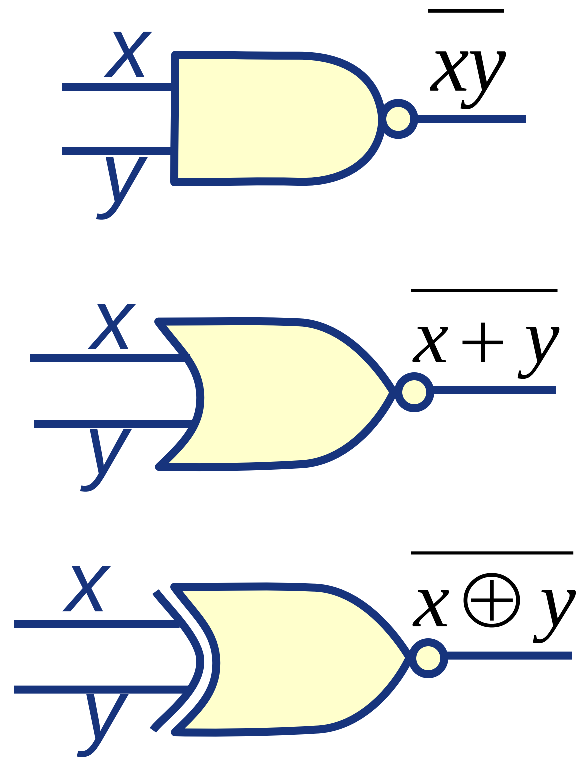
# Multi-input AND, OR, XOR

- Can extend these gates to arbitrarily many inputs.
- Two commonly seen drawing styles:
  - Note that the second style keeps the gate icon relatively small.

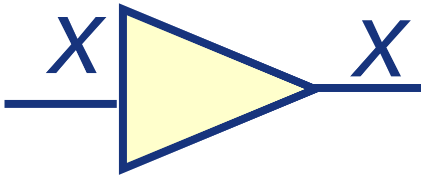


# NAND, NOR, XNOR

- Just like the earlier icons, but with a small circle on the gate’s output.
  - Denotes that output is complemented.
- The circles can also be placed on inputs.
  - Means, input is complemented before being used.

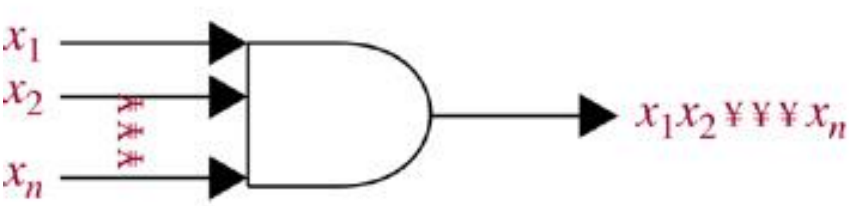


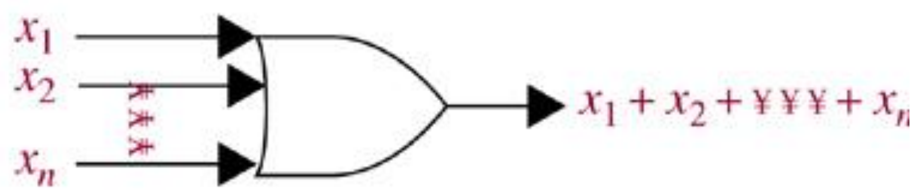
# Buffer

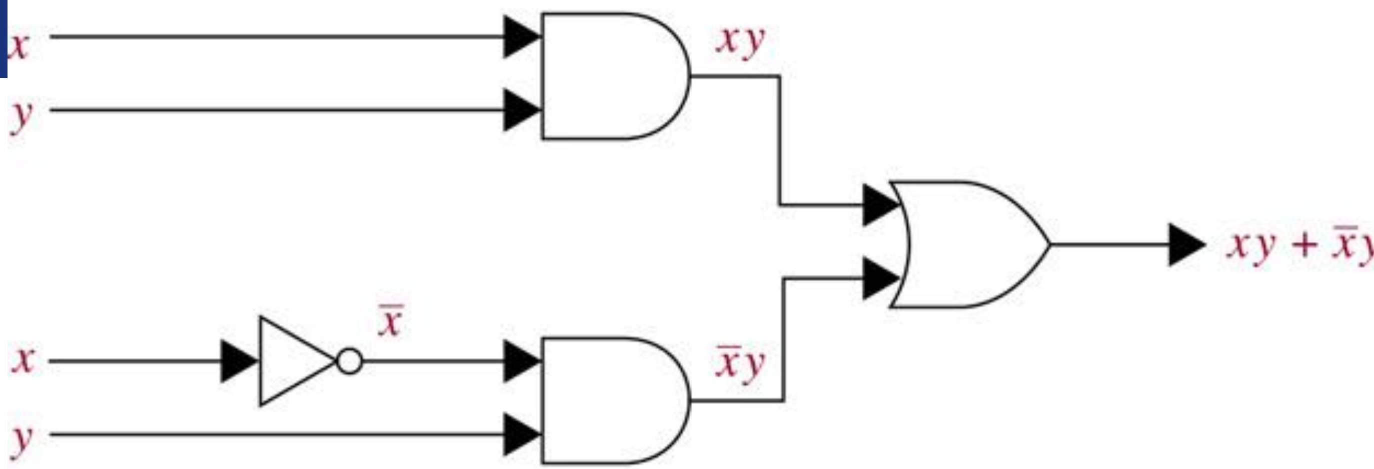


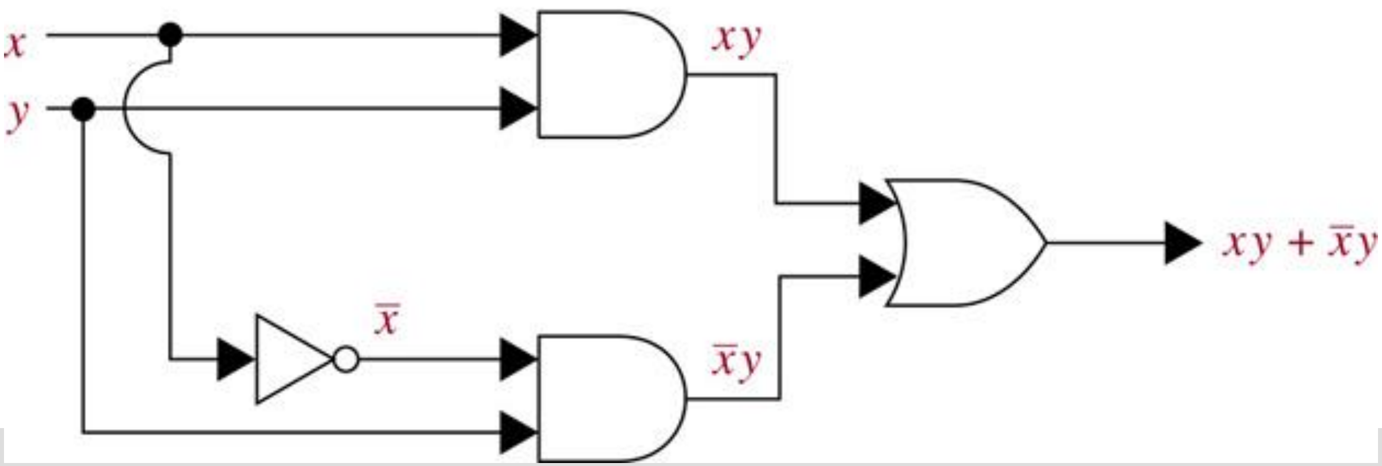

- What about an inverter symbol *without* a circle?
- This is called a *buffer*. It is the identity function.
- It serves no logical purpose, but…
- It represents an explicit delay in the circuit.
  - This is sometimes useful for timing purposes.
- All gates, when physically implemented, incur a non-zero delay between when their inputs are seen and when their outputs are ready.


(a) Inverter (b) *OR* gate (c) *AND* gate


$x_1$
$x_2$
$x_n$
$x_1x_2$¥¥¥$x_n$

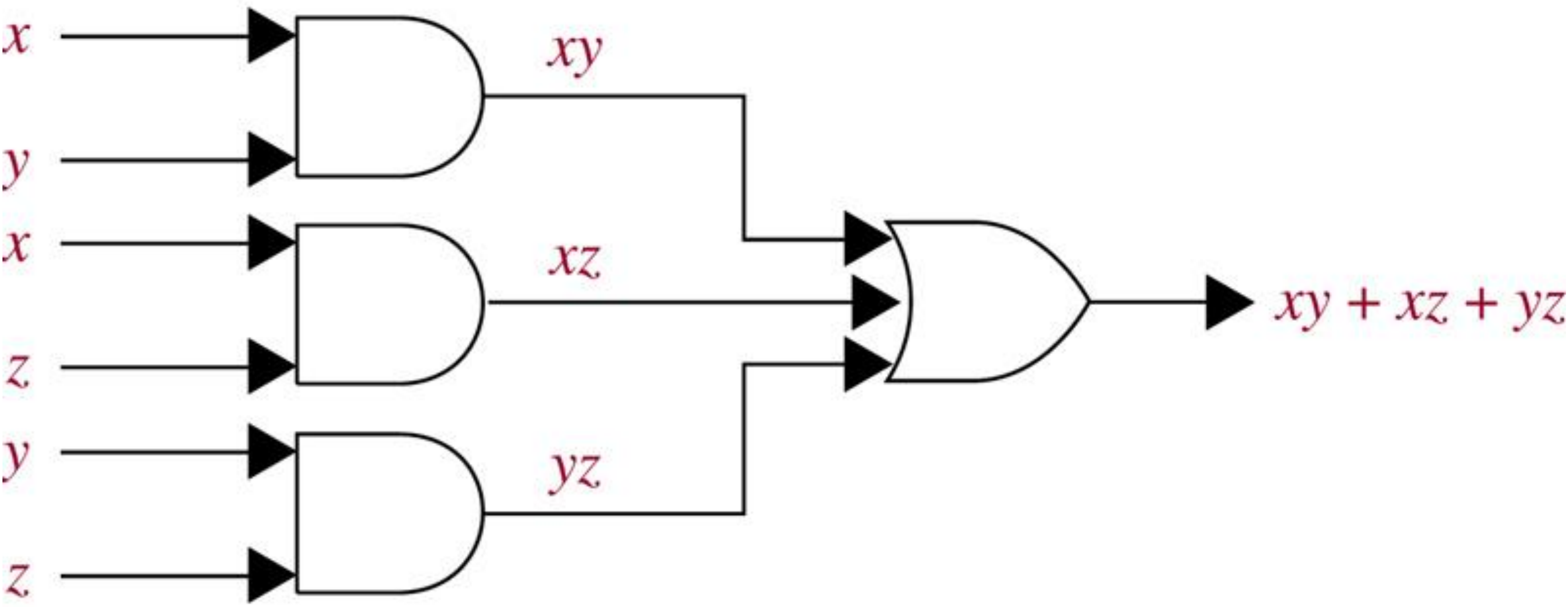

$x_1$
$x_2$
$x_n$
$x_1+x_2+$¥¥¥$+x_n$

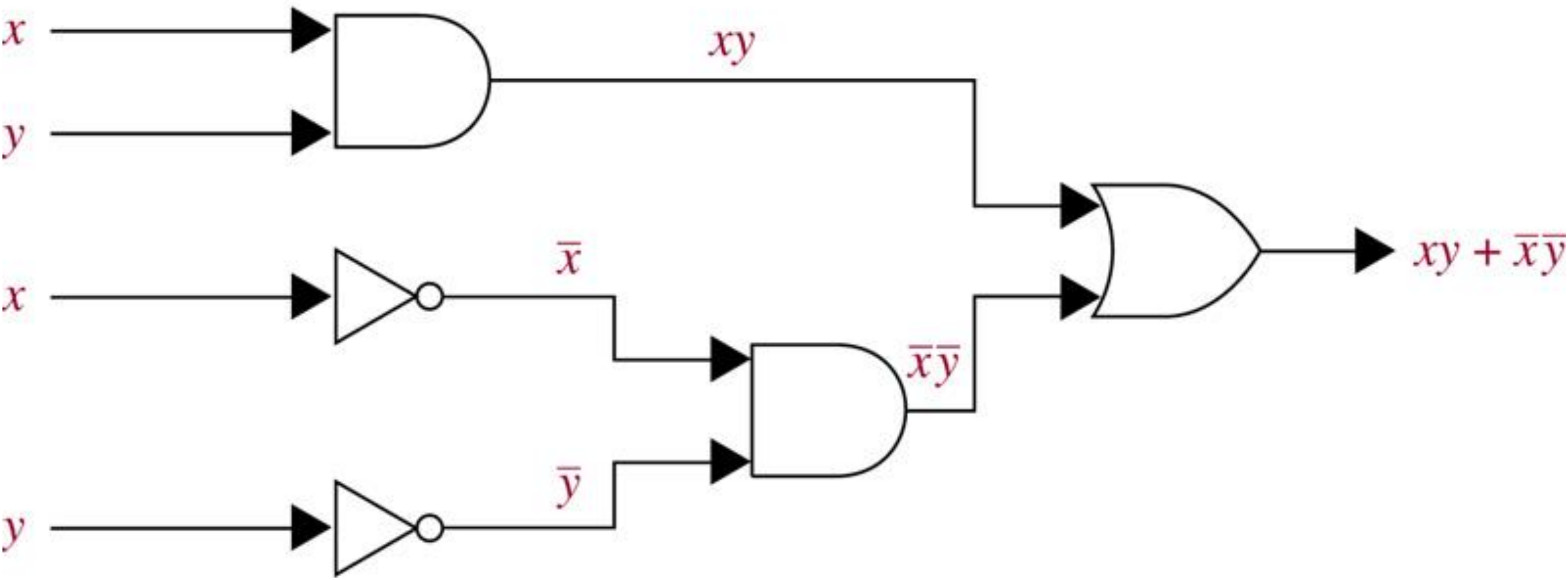

x
y
xy
x
$\bar{x}$
y
$\bar{x}y$
$xy + \bar{x}y$


x
y
xy
$\bar{x}$
$\bar{x}y$
$xy + \bar{x}y$

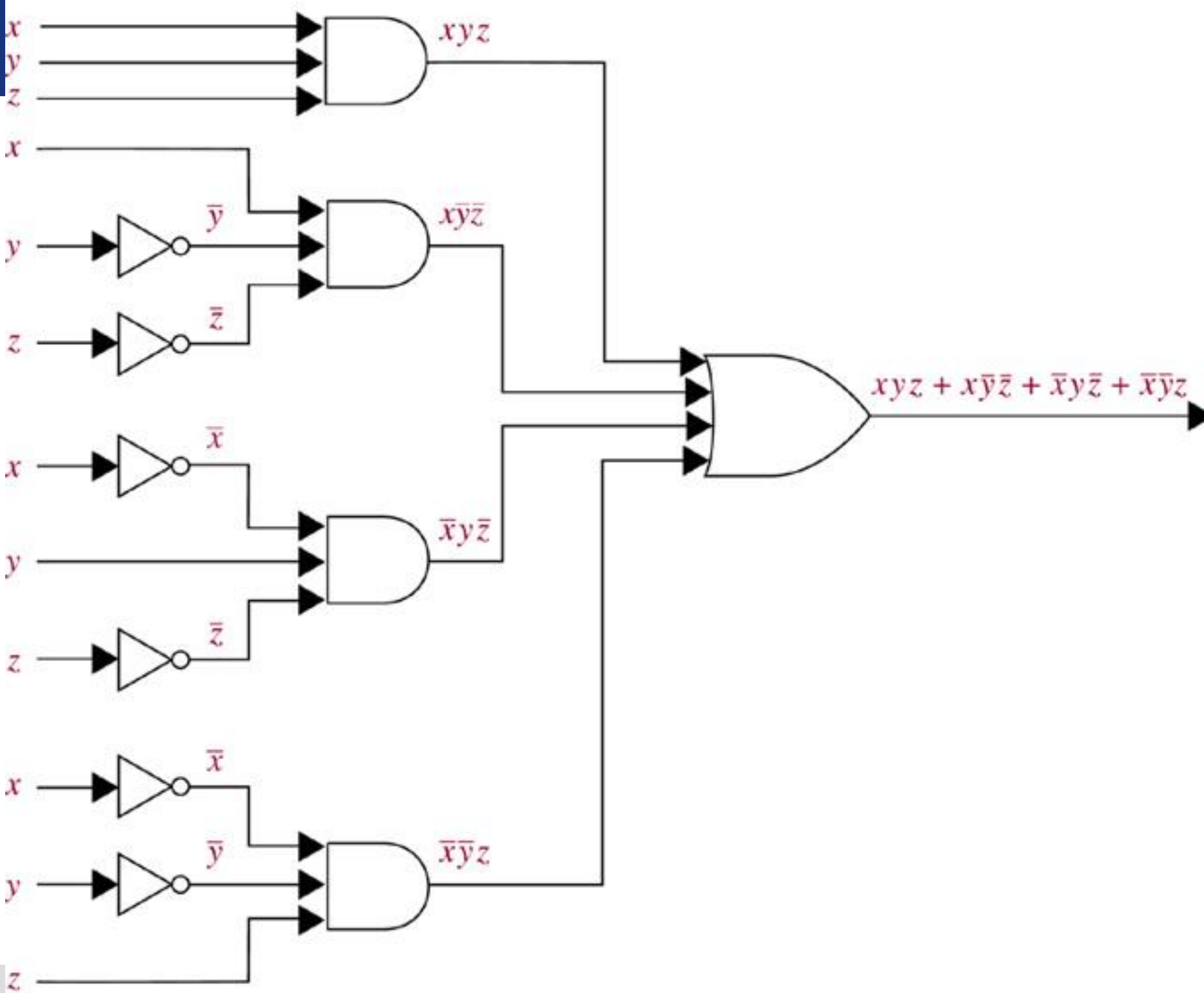
Logo







$x$
$y$
$xy$
$x$
$\bar{x}$
$y$
$\bar{y}$
$\bar{x}\bar{y}$
$xy + \bar{x}\bar{y}$

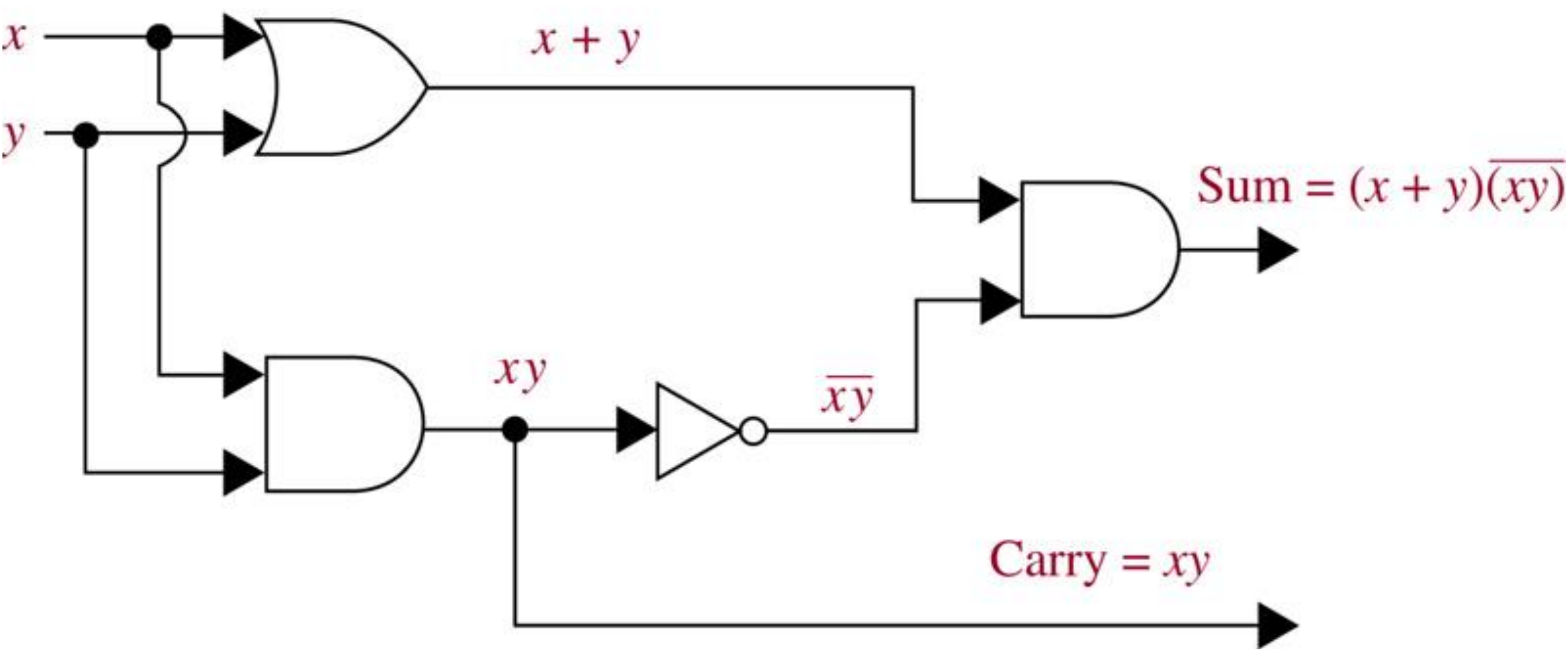

x
y
z
xyz
x
y
ȳ
xȳz̄
z
z̄
x
x̄
y
x̄yz̄
z
z̄
x
x̄
y
ȳ
x̄ȳz
z
xyz + xȳz̄ + x̄yz̄ + x̄ȳz

# Combinational Logic Circuits

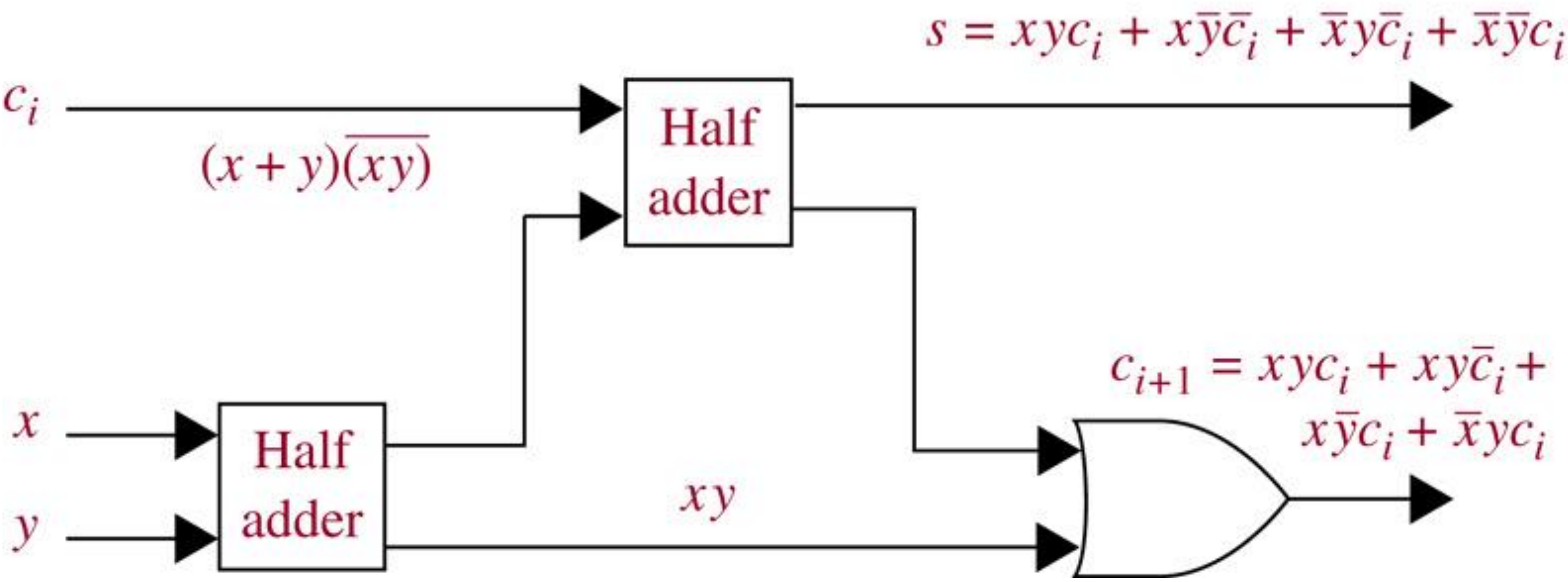
- **Note:** The correct word to use here is "combinat**ion**al," **NOT** "combinat**ori**al!"
  - Many sloppy authors get this wrong.
- These are circuits composed of Boolean gates whose outputs depend only on their most recent inputs, not on earlier inputs.
  - Thus these circuits have no useful memory.
    - Their state persists while the inputs are constant, but is irreversibly lost when the input signals change.

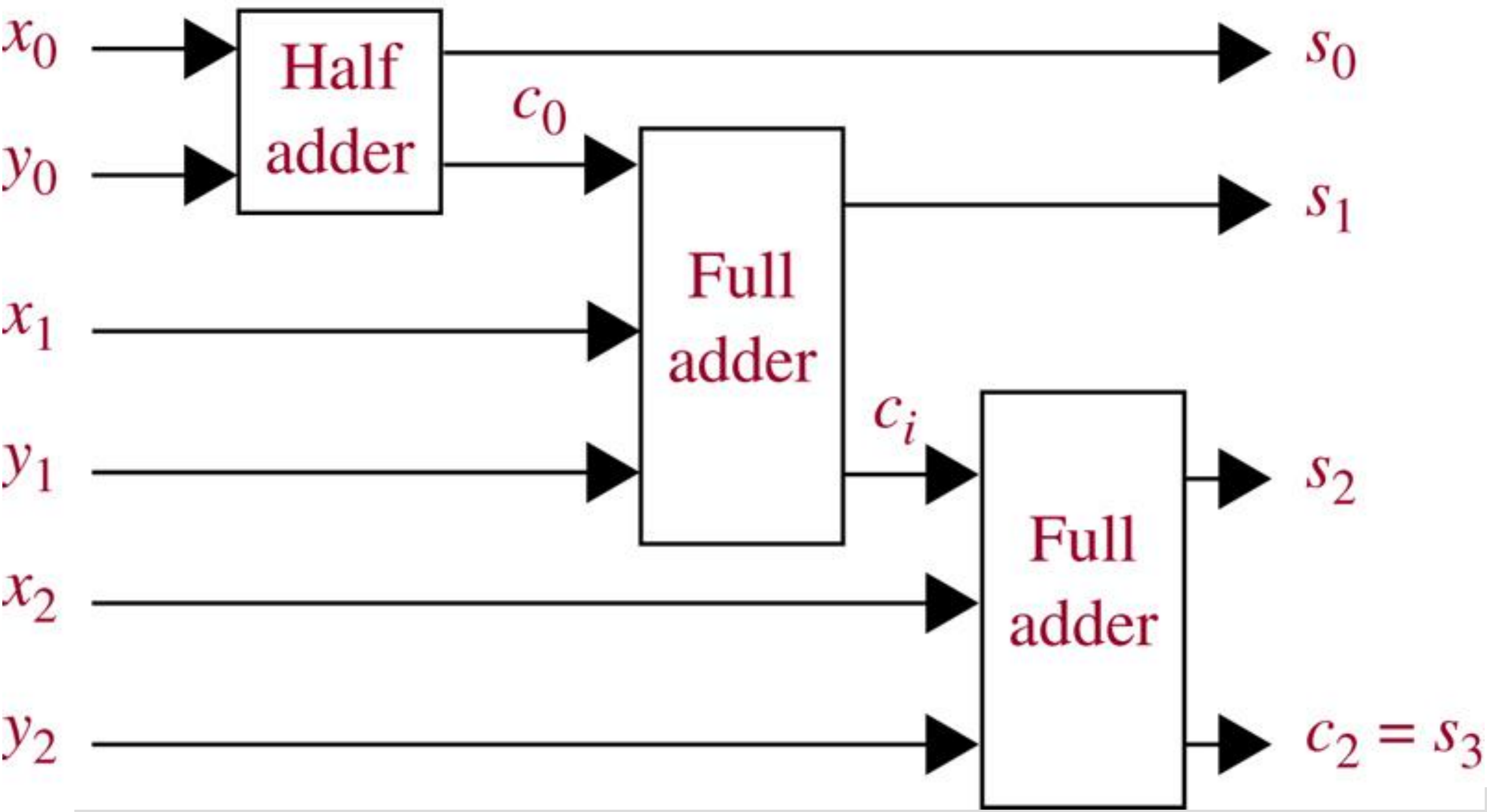
# Combinational Circuit Examples

- Draw a few examples on the board:
  - Majority voting circuit.
  - XOR using OR / AND / NOT.
  - 3-input XOR using OR / AND / NOT.
- Also, show some binary adders:
  - Half adder using OR/AND/NOT.
  - Full adder from half-adders.
  - Ripple-carry adders.

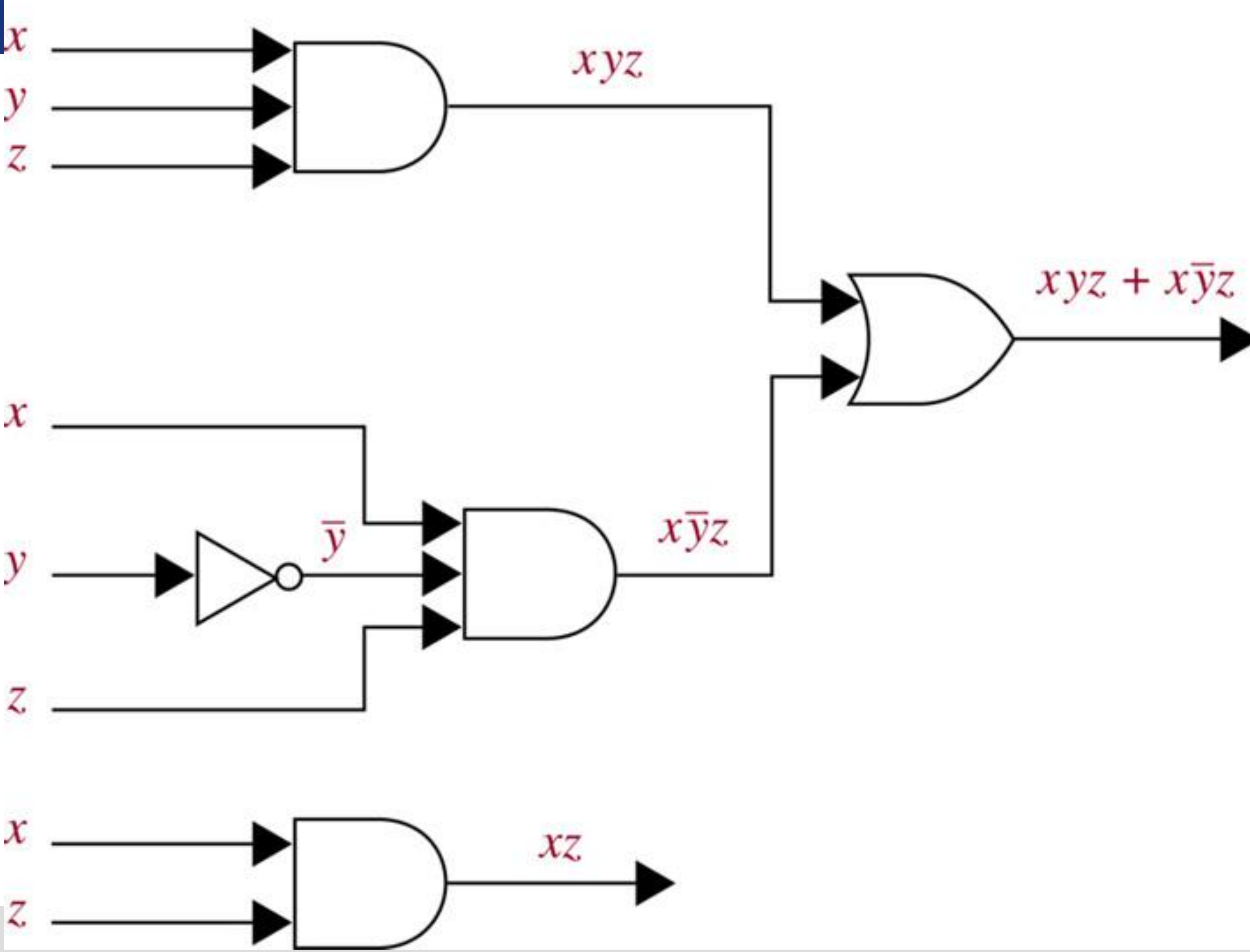


x
y
x + y
xy
$\overline{xy}$
Sum = $(x + y)\overline{(xy)}$
Carry = xy


$s = xyc_i + x\bar{y}\bar{c}_i + \bar{x}y\bar{c}_i + \bar{x}\bar{y}c_i$
$c_i$
$(x + y)\overline{(xy)}$
Half adder
$c_{i+1} = xyc_i + xy\bar{c}_i + x\bar{y}c_i + \bar{x}yc_i$
$x$
$y$
Half adder
$xy$



$x_0$
$y_0$
Half adder
$c_0$
$s_0$
Full adder
$s_1$
$x_1$
$y_1$
$c_i$
Full adder
$s_2$
$x_2$
$y_2$
$c_2 = s_3$


x
y
z
xyz
xyz + x$\bar{y}$z
x
$\bar{y}$
y
x$\bar{y}$z
z
x
z
xz

# Minimizing Circuits

- Karnaugh Maps
- *Don't care* conditions
- The Quine-McCluskey Method

# Goals of Circuit Minimization

- (1) Minimize the number of primitive Boolean logic gates needed to implement the circuit.
  - Ultimately, this also roughly minimizes the number of transistors, the chip area, and the cost.
    - Also roughly minimizes the energy expenditure
      - among traditional irreversible circuits.
  - This will be our focus.
- (2) It is also often useful to minimize the number of combinational *stages* or logical *depth* of the circuit.
  - This roughly minimizes the *delay* or *latency* through the circuit, the time between input and output.

# Minimizing DNF Expressions

- Using DNF (or CNF) guarantees there is always *some* circuit that implements any desired Boolean function.
  - However, it may be far larger than needed!
- We would like to find the *smallest* sum-of-products expression that yields a given function.
  - This will yield a fairly small circuit.
  - However, circuits of other forms (not CNF or DNF) might be even smaller for complex functions.

| | $y$ | $\bar{y}$ |
|---|---|---|
| $x$ | $xy$ | $x\bar{y}$ |
| $\bar{x}$ | $\bar{x}y$ | $\bar{x}\bar{y}$ |

| | $y$ | $\bar{y}$ |
|---|---|---|
| $x$ | 1 | |
| $\bar{x}$ | 1 | |

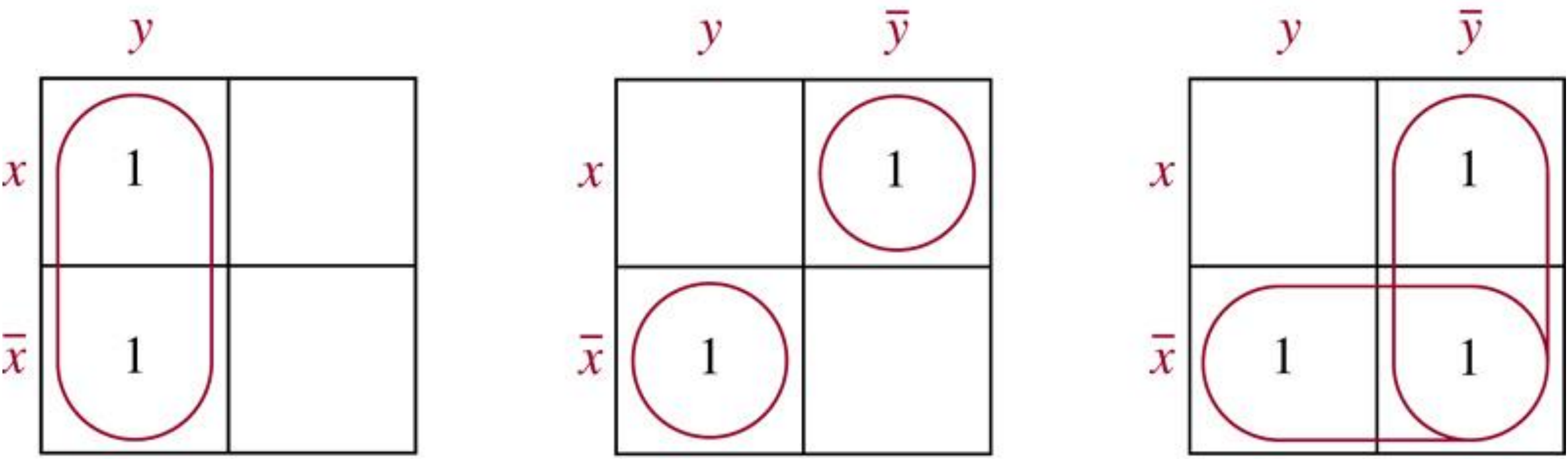
(a)

| | $y$ | $\bar{y}$ |
|---|---|---|
| $x$ | | 1 |
| $\bar{x}$ | 1 | |

(b)

| | $y$ | $\bar{y}$ |
|---|---|---|
| $x$ | | 1 |
| $\bar{x}$ | 1 | 1 |

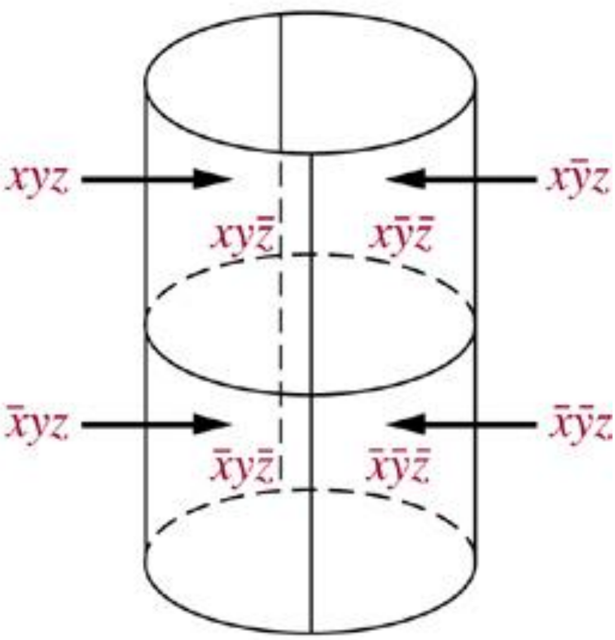
(c)




(a) (b) (c)

| | $yz$ | $y\bar{z}$ | $\bar{y}\bar{z}$ | $\bar{y}z$ |
|---|---|---|---|---|
| $x$ | $xyz$ | $xy\bar{z}$ | $x\bar{y}\bar{z}$ | $x\bar{y}z$ |
| $\bar{x}$ | $\bar{x}yz$ | $\bar{x}y\bar{z}$ | $\bar{x}\bar{y}\bar{z}$ | $\bar{x}\bar{y}z$ |

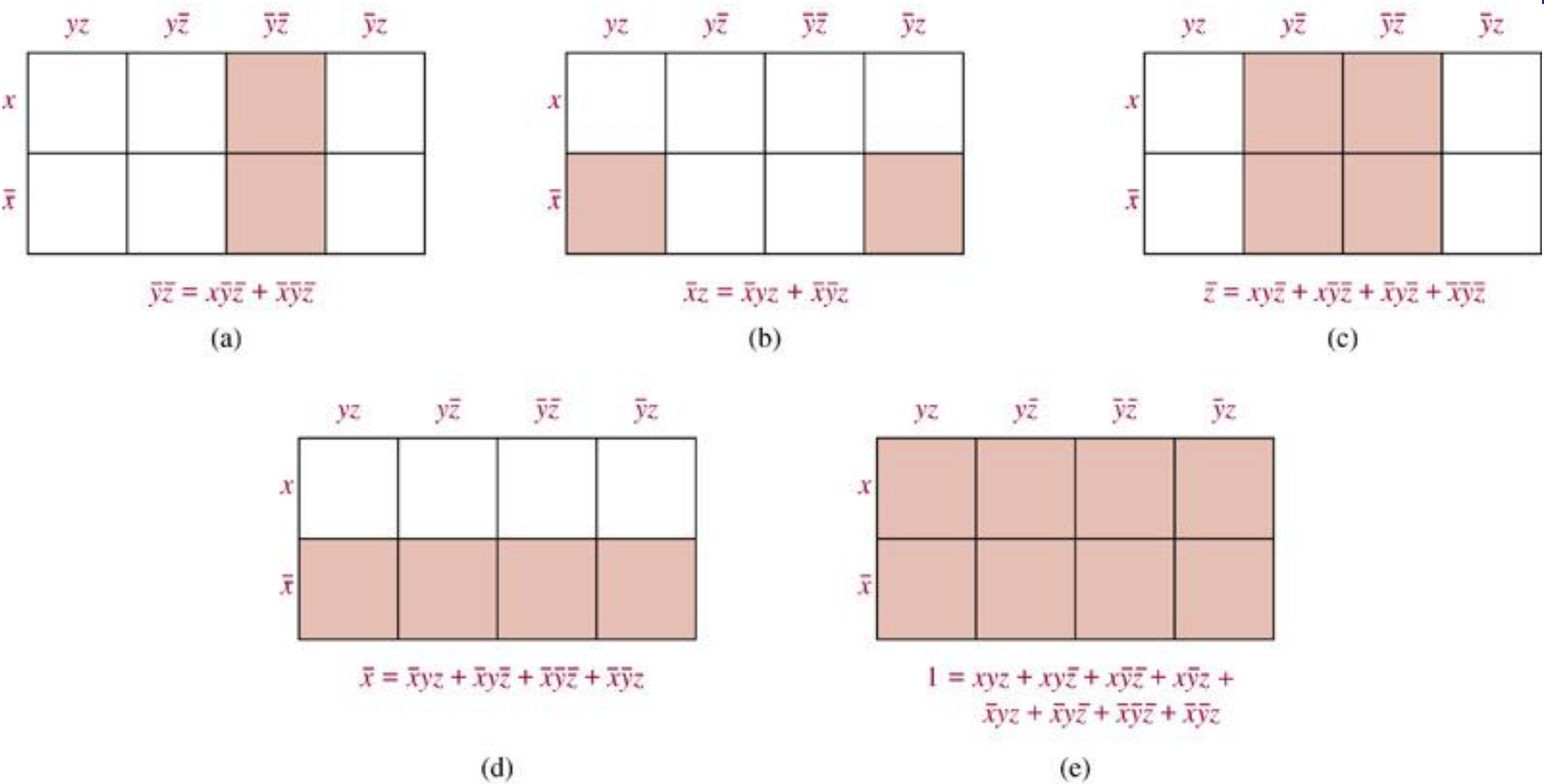
(a)




(b)

Logo




yz
$y\bar{z}$
$\bar{y}\bar{z}$
$\bar{y}z$
$x$
$\bar{x}$
$\bar{y}\bar{z} = x\bar{y}\bar{z} + \bar{x}\bar{y}\bar{z}$
(a)
$\bar{x}z = \bar{x}yz + \bar{x}\bar{y}z$
(b)
$\bar{z} = xy\bar{z} + x\bar{y}\bar{z} + \bar{x}y\bar{z} + \bar{x}\bar{y}\bar{z}$
(c)
$\bar{x} = \bar{x}yz + \bar{x}y\bar{z} + \bar{x}\bar{y}\bar{z} + \bar{x}\bar{y}z$
(d)
$1 = xyz + xy\bar{z} + x\bar{y}\bar{z} + x\bar{y}z +$
$\bar{x}yz + \bar{x}y\bar{z} + \bar{x}\bar{y}\bar{z} + \bar{x}\bar{y}z$
(e)

| | $yz$ | $y\bar{z}$ | $\bar{y}\bar{z}$ | $\bar{y}z$ |
|---|---|---|---|---|
| $x$ | | 1 | 1 | |
| $\bar{x}$ | 1 | | 1 | |

(a)

| | $yz$ | $y\bar{z}$ | $\bar{y}\bar{z}$ | $\bar{y}z$ |
|---|---|---|---|---|
| $x$ | | | 1 | 1 |
| $\bar{x}$ | 1 | | 1 | 1 |

(b)

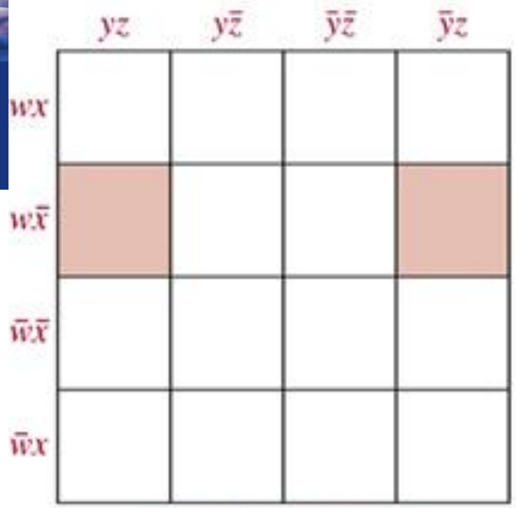
| | $yz$ | $y\bar{z}$ | $\bar{y}\bar{z}$ | $\bar{y}z$ |
|---|---|---|---|---|
| $x$ | 1 | 1 | 1 | 1 |
| $\bar{x}$ | 1 | | 1 | 1 |

(c)

| | $yz$ | $y\bar{z}$ | $\bar{y}\bar{z}$ | $\bar{y}z$ |
|---|---|---|---|---|
| $x$ | | 1 | 1 | |
| $\bar{x}$ | | | 1 | 1 |

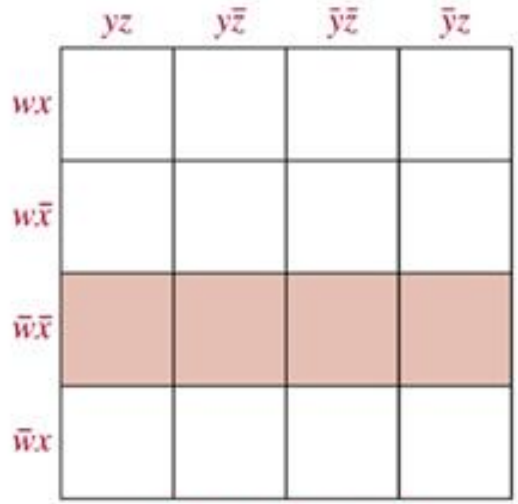
(d)

| | $yz$ | $y\bar{z}$ | $\bar{y}\bar{z}$ | $\bar{y}z$ |
|---|---|---|---|---|
| $wx$ | $wxyz$ | $wxy\bar{z}$ | $wx\bar{y}\bar{z}$ | $wx\bar{y}z$ |
| $w\bar{x}$ | $w\bar{x}yz$ | $w\bar{x}y\bar{z}$ | $w\bar{x}\bar{y}\bar{z}$ | $w\bar{x}\bar{y}z$ |
| $\bar{w}\bar{x}$ | $\bar{w}\bar{x}yz$ | $\bar{w}\bar{x}y\bar{z}$ | $\bar{w}\bar{x}\bar{y}\bar{z}$ | $\bar{w}\bar{x}\bar{y}z$ |
| $\bar{w}x$ | $\bar{w}xyz$ | $\bar{w}xy\bar{z}$ | $\bar{w}x\bar{y}\bar{z}$ | $\bar{w}x\bar{y}z$ |



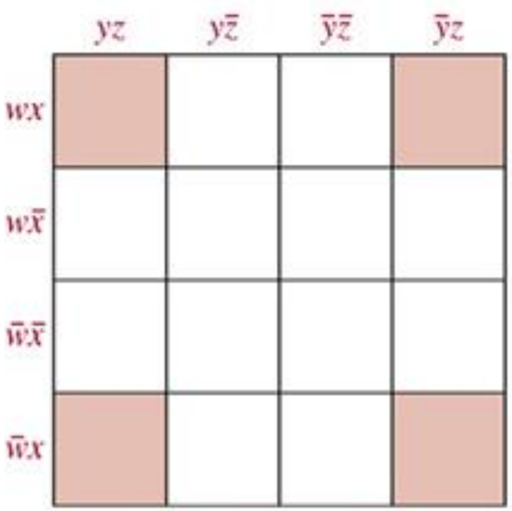

$w\bar{x}z = w\bar{x}yz + w\bar{x}\bar{y}z$

(a)



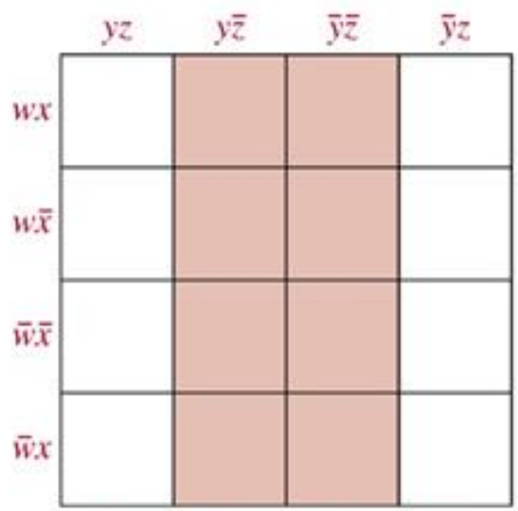

$\bar{w}\bar{x} = \bar{w}\bar{x}yz + \bar{w}\bar{x}y\bar{z} + \bar{w}\bar{x}\bar{y}\bar{z} + \bar{w}\bar{x}\bar{y}z$

(b)




$x\bar{z} = wxyz + wx\bar{y}z + \bar{w}xyz + \bar{w}x\bar{y}z$

(c)

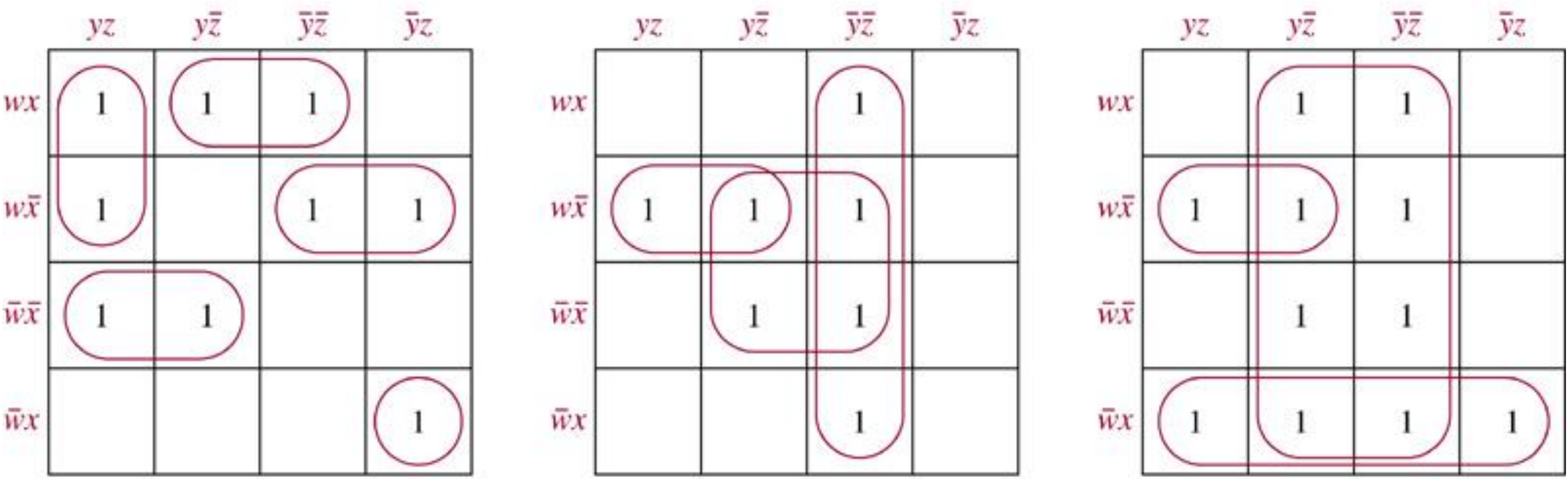



$\bar{z} = wxy\bar{z} + wx\bar{y}\bar{z} + w\bar{x}y\bar{z} + w\bar{x}\bar{y}\bar{z} + \bar{w}\bar{x}y\bar{z} + \bar{w}\bar{x}\bar{y}\bar{z} + \bar{w}xy\bar{z} + \bar{w}x\bar{y}\bar{z}$

(d)

Logo






(a) (b) (c)

| | $yz$ | $y\bar{z}$ | $\bar{y}\bar{z}$ | $\bar{y}z$ |
|---|---|---|---|---|
| $wx$ | $d$ | $d$ | $d$ | $d$ |
| $w\bar{x}$ | $d$ | $d$ | 1 | 1 |
| $\bar{w}\bar{x}$ | | | | |
| $\bar{w}x$ | 1 | 1 | | 1 |

(a)

| | $yz$ | $y\bar{z}$ | $\bar{y}\bar{z}$ | $\bar{y}z$ |
|---|---|---|---|---|
| $wx$ | $d$ | $d$ | $d$ | $d$ |
| $w\bar{x}$ | $d$ | $d$ | 1 | 1 |
| $\bar{w}\bar{x}$ | | | | |
| $\bar{w}x$ | 1 | 1 | | 1 |

(b)

| | $yz$ | $y\bar{z}$ | $\bar{y}\bar{z}$ | $\bar{y}z$ |
|---|---|---|---|---|
| $wx$ | $d$ | $d$ | $d$ | $d$ |
| $w\bar{x}$ | $d$ | $d$ | 1 | 1 |
| $\bar{w}\bar{x}$ | | | | |
| $\bar{w}x$ | 1 | 1 | | 1 |

(c)

| | $yz$ | $y\bar{z}$ | $\bar{y}\bar{z}$ | $\bar{y}z$ |
|---|---|---|---|---|
| $wx$ | $d$ | $d$ | $d$ | $d$ |
| $w\bar{x}$ | $d$ | $d$ | 1 | 1 |
| $\bar{w}\bar{x}$ | | | | |
| $\bar{w}x$ | 1 | 1 | | 1 |

(d)

**TABLE 1**

| $x$ | $y$ | $F(x, y)$ |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |

**TABLE 2**

| $x$ | $y$ | $z$ | $xy$ | $\bar{z}$ | $F(x, y, z) = xy + \bar{z}$ |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 |

**TABLE 3** **The Boolean Functions of Degree Two.**

| $x$ | $y$ | $F_1$ | $F_2$ | $F_3$ | $F_4$ | $F_5$ | $F_6$ | $F_7$ | $F_8$ | $F_9$ | $F_{10}$ | $F_{11}$ | $F_{12}$ | $F_{13}$ | $F_{14}$ | $F_{15}$ | $F_{16}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |

**TABLE 4 The Number of Boolean Functions of Degree *n*.**

| *Degree* | *Number* |
|---|---|
| 1 | 4 |
| 2 | 16 |
| 3 | 256 |
| 4 | 65,536 |
| 5 | 4,294,967,296 |
| 6 | 18,446,744,073,709,551,616 |

Logo

**TABLE 5 Boolean Identities.**

| *Identity* | *Name* |
|---|---|
| $\overline{\overline{x}} = x$ | Law of the double complement |
| $x + x = x$<br>$x \cdot x = x$ | Idempotent laws |
| $x + 0 = x$<br>$x \cdot 1 = x$ | Identity laws |
| $x + 1 = 1$<br>$x \cdot 0 = 0$ | Domination laws |
| $x + y = y + x$<br>$xy = yx$ | Commutative laws |
| $x + (y + z) = (x + y) + z$<br>$x(yz) = (xy)x$ | Associative laws |
| $x + yz = (x + y)(x + z)$<br>$x(y + z) = xy + xz$ | Distributive laws |
| $\overline{(xy)} = \overline{x} + \overline{y}$<br>$\overline{(x + y)} = \overline{x}\,\overline{y}$ | De Morgan's laws |
| $x + xy = x$<br>$x(x + y) = x$ | Absorption laws |
| $x + \overline{x} = 1$ | Unit property |
| $x\overline{x} = 0$ | Zero property |

Logo



**TABLE 6** **Verifying One of the Distributive Laws.**

| $x$ | $y$ | $z$ | $y+z$ | $xy$ | $xz$ | $x(y+z)$ | $xy+xz$ |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Logo

**TABLE 1**

| $x$ | $y$ | $z$ | $F$ | $G$ |
|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |

Logo



**TABLE 2**

| $x$ | $y$ | $z$ | $x+y$ | $\overline{z}$ | $(x+y)\overline{z}$ |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 |

**TABLE 1**

| $x$ | $y$ | $F(x, y)$ |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |

Logo

**TABLE 2**

| $x$ | $y$ | $z$ | $F(x, y, z)$ |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 |

Logo

**TABLE 3**
**Input and Outpu tfor the Half Adder.**

| *Input* | | *Output* | |
|---|---|---|---|
| $x$ | $y$ | $s$ | $c$ |
| 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 |
| 0 | 0 | 0 | 0 |

Logo

**TABLE 4**
**Input and Outpu tfor the Full Adder.**

| Input | | | Output | |
|---|---|---|---|---|
| $x$ | $y$ | $c_i$ | $s$ | $c_{i+1}$ |
| 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 |

**TABLE 1**

| *Digit* | *w* | *x* | *y* | *z* | *F* |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 2 | 0 | 0 | 1 | 0 | 0 |
| 3 | 0 | 0 | 1 | 1 | 0 |
| 4 | 0 | 1 | 0 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 | 1 |
| 6 | 0 | 1 | 1 | 0 | 1 |
| 7 | 0 | 1 | 1 | 1 | 1 |
| 8 | 1 | 0 | 0 | 0 | 1 |
| 9 | 1 | 0 | 0 | 1 | 1 |

**TABLE 2**

| *Minterm* | *Bit String* | *Number of 1s* |
|---|---|---|
| $xyz$ | 111 | 3 |
| $x\overline{y}z$<br>$\overline{x}yz$ | 101<br>011 | 2<br>2 |
| $\overline{x}\,\overline{y}z$ | 001 | 1 |
| $\overline{x}\,\overline{y}\,\overline{z}$ | 000 | 0 |

Logo



**TABLE 3**

| | | | Step 1 | | | Step 2 | | |
|---|---|---|---|---|---|---|---|---|
| | *Term* | *Bit String* | | *Term* | *String* | | *Term* | *String* |
| 1 | $xyz$ | 111 | (1,2) | $xz$ | 1–1 | (1,2,3,4) | $z$ | – –1 |
| 2 | $x\overline{y}z$ | 101 | (1,3) | $yz$ | –11 | | | |
| 3 | $\overline{x}yz$ | 011 | (2,4) | $\overline{y}z$ | –01 | | | |
| 4 | $\overline{x}\,\overline{y}z$ | 001 | (3,4) | $\overline{x}z$ | 0–1 | | | |
| 5 | $\overline{x}\,\overline{y}\,\overline{z}$ | 000 | (4,5) | $\overline{x}\,\overline{y}$ | 00– | | | |

**TABLE 4**

| | $xyz$ | $x\bar{y}z$ | $\bar{x}yz$ | $\bar{x}\,\bar{y}z$ | $\bar{x}\,\bar{y}\,\bar{z}$ |
|---|---|---|---|---|---|
| $z$ | X | X | X | X | |
| $\bar{x}\,\bar{y}$ | | | | X | X |

**TABLE 5**

| *Term* | *Bit String* | *Number of 1s* |
|---|---|---|
| $wxy\overline{z}$ | 1110 | 3 |
| $w\overline{x}yz$ | 1011 | 3 |
| $\overline{w}xyz$ | 0111 | 3 |
| $w\overline{x}y\overline{z}$ | 1010 | 2 |
| $\overline{w}x\overline{y}z$ | 0101 | 2 |
| $\overline{w}\,\overline{x}yz$ | 0011 | 2 |
| $\overline{w}\,\overline{x}\,\overline{y}z$ | 0001 | 1 |

Logo



**TABLE 6**

| | | | Step 1 | | | Step 2 | | |
|---|---|---|---|---|---|---|---|---|
| | Term | Bit String | | Term | String | | Term | String |
| 1 | $wxy\bar{z}$ | 1110 | (1,4) | $wy\bar{z}$ | 1–10 | (3,5,6,7) | $\bar{w}z$ | 0 – –1 |
| 2 | $w\bar{x}yz$ | 1011 | (2,4) | $w\bar{x}y$ | 101 – | | | |
| 3 | $\bar{w}xyz$ | 0111 | (2,6) | $\bar{x}yz$ | –011 | | | |
| 4 | $w\bar{x}y\bar{z}$ | 1010 | (3,5) | $\bar{w}xz$ | 01–1 | | | |
| 5 | $\bar{w}x\bar{y}z$ | 0101 | (3,6) | $\bar{w}yz$ | 0–11 | | | |
| 6 | $\bar{w}\,\bar{x}yz$ | 0011 | (5,7) | $\bar{w}\,\bar{y}z$ | 0–01 | | | |
| 7 | $\bar{w}\,\bar{x}\,\bar{y}z$ | 0001 | (6,7) | $\bar{w}\,\bar{x}z$ | 00–1 | | | |

**TABLE 7**

| | $wxy\bar{z}$ | $w\bar{x}yz$ | $\bar{w}xyz$ | $w\bar{x}y\bar{z}$ | $\bar{w}x\bar{y}z$ | $\bar{w}\bar{x}yz$ | $\bar{w}\bar{x}\bar{y}z$ |
|---|---|---|---|---|---|---|---|
| $\bar{w}z$ | | | X | | X | X | X |
| $wy\bar{z}$ | X | | | X | | | |
| $w\bar{x}y$ | | X | | X | | | |
| $\bar{x}yz$ | | X | | | | X | |

Logo

# End of Section 1.6