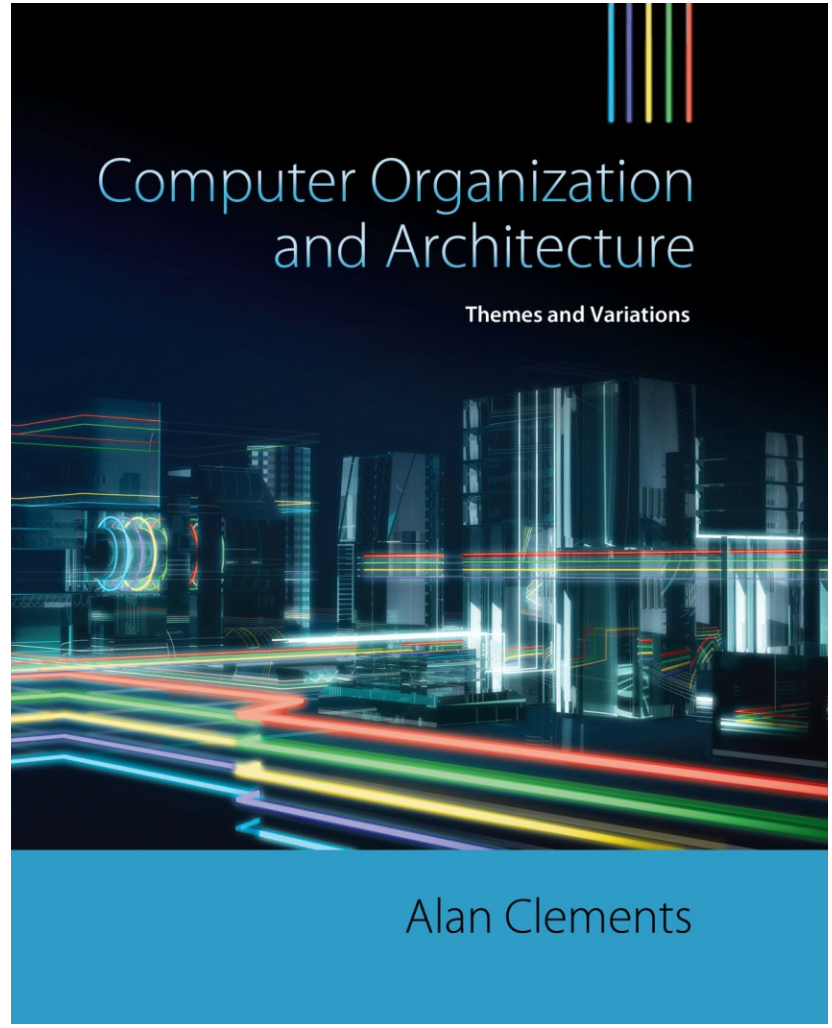
Part 5

CHAPTER 2

Computer Arithmetic and Digital Logic



1

These slides are provided with permission from the copyright for CS2208 use only.
The slides must not be reproduced or provided to anyone outside the class.

All downloaded copies of the slides are for personal use only.
Students must destroy these copies within 30 days after receiving the course's final assessment.

CENGAGE Learning

Comparing AND and OR Gates

TABLE 2.10 Truth Table for AND and OR Gates with Both Constant and Variable Inputs

AND → 1 iff all input = 1		OR → 0 if all input = 0	
Constant	Variable	Constant	Variable
$0 \cdot 0 = 0$	$A \cdot 0 = 0$	$0 + 0 = 0$	$A + 0 = A$
$0 \cdot 1 = 0$	$A \cdot 1 = A$	$0 + 1 = 1$	$A + 1 = 1$
$1 \cdot 0 = 0$	$A \cdot \bar{A} = 0$	$1 + 0 = 1$	$A + \bar{A} = 1$
$1 \cdot 1 = 1$	$A \cdot A = A$	$1 + 1 = 1$	$A + A = A$

Derived Gates NOR, NAND, Exclusive OR

- **NOR**, **NAND** and **XOR** are gates that can be derived from basic gates.
  - a **NOR** gate is an **OR** followed by an **inverter**.
  - A **NAND** gate is an **AND** followed by and **inverter** and
  - An **XOR** gate is an **OR** gate whose output is true *only if an odd number of its input is true.*

This is B not C

+ if have 3 inputs, all = 1 : output = 1.

TABLE 2.11 Truth Table for the NOR Gate, NAND Gate, and Exclusive OR Gates

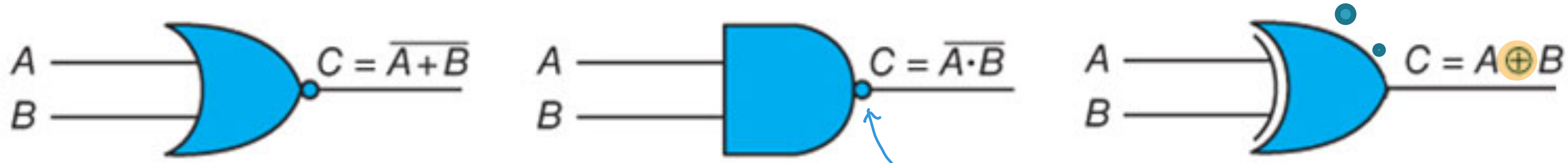
A	B	$C = \overline{A + B}$	A	B	$C = \overline{A \cdot B}$	A	B	$C = A \oplus B$
0	0	1	0	0	1	0	0	0
0	1	0	0	1	1	0	1	1
1	0	0	1	0	1	1	0	1
1	1	0	1	1	0	1	1	0

(a) The NOR gate (b) The NAND gate (c) The XOR gate

- These gates (**NOR**, **NAND**, and **XOR**) are used extensively in digital circuits and have their own symbols.

There is no bubble here.
The book added it incorrectly.

FIGURE 2.19 Three derived gates

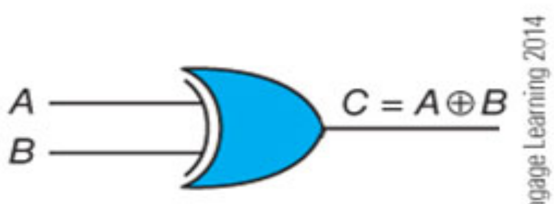


means inversion

(a) NOR gate (b) NAND gate (c) Exclusive OR gate

Exclusive OR

- The **Exclusive OR** function is written as **XOR** or **EOR**.
- The **Exclusive OR** is represented by $\oplus$ (e.g., $C = A \oplus B$).
- A two-input **XOR** gate can be constructed by *two* **inverters**, *two* **AND** gates and *one* **OR** gate, as shown in Figure 2.20.
  ($F = A \oplus B = A \,.\, \overline{B} + \overline{A} \,.\, B$)



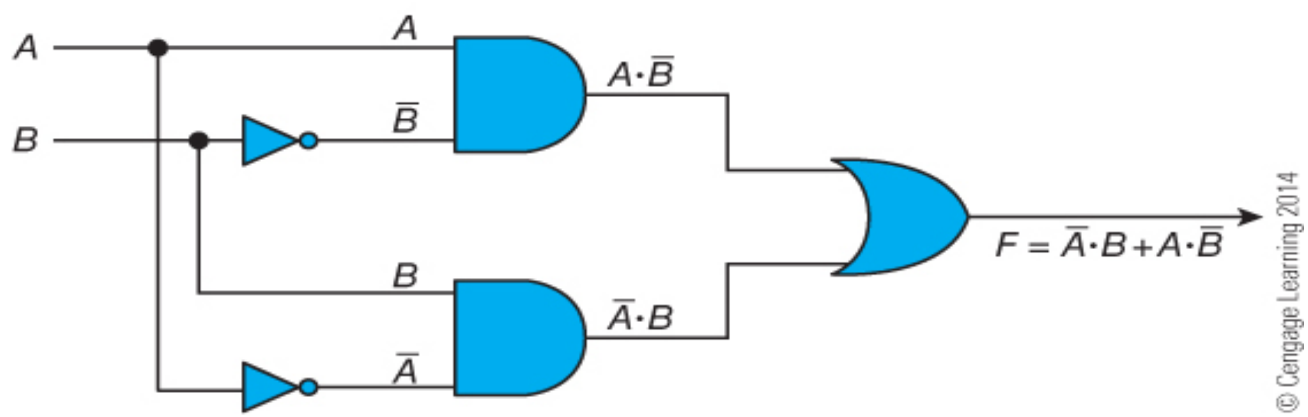
(c) Exclusive OR gate

Exclusive OR Gates

A	B	$C = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

(c) The XOR gate

FIGURE 2.20 Constructing an XOR circuit from AND, OR, and NOT gates



Three Input Exclusive OR

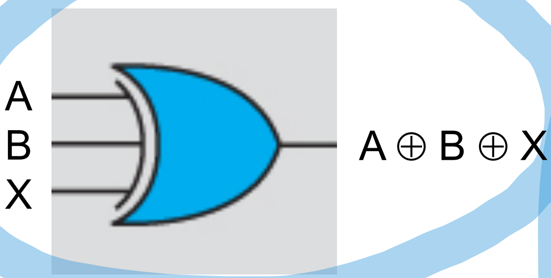
- A three-input **XOR** gate can be constructed with *two* **XOR** gates, each with two-inputs
- $C = A \oplus B = A\,.\,\overline{B} + \overline{A}\,.\,B$
- $A \oplus B \oplus X = C \oplus X = C\,.\,\overline{X} + \overline{C}\,.\,X$

$$\overline{C} = \overline{(A\,.\,\overline{B} + \overline{A}\,.\,B)}$$
$$= \overline{A}\,.\,\overline{B} + A\,.\,B$$

$$= (A\,.\,\overline{B} + \overline{A}\,.\,B)\,.\,\overline{X} + (\overline{A}\,.\,\overline{B} + A\,.\,B)\,.\,X$$

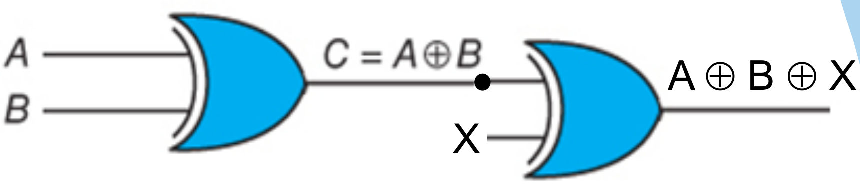
$$= A\,.\,\overline{B}\,.\,\overline{X} + \overline{A}\,.\,B\,.\,\overline{X} + \overline{A}\,.\,\overline{B}\,.\,X + A\,.\,B\,.\,X$$

3 ways to express the same thing.



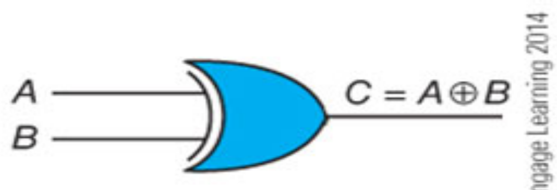
A	B	X	A ⊕ B ⊕ X
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

A	B	C = A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0



Inversion Bubbles

- By **convention**, the triangle in inverters are often omitted from circuit diagrams and the *bubble notation is used*.
- *A small bubble is placed at a gate's input to indicate inversion.*
- In the circuit below, the *two* **AND** gates form the product of (**NOT** A) **AND** B and A **AND** (**NOT** B), i.e., $\bar{A} . B + A . \bar{B}$
- This circuit implements **XOR**



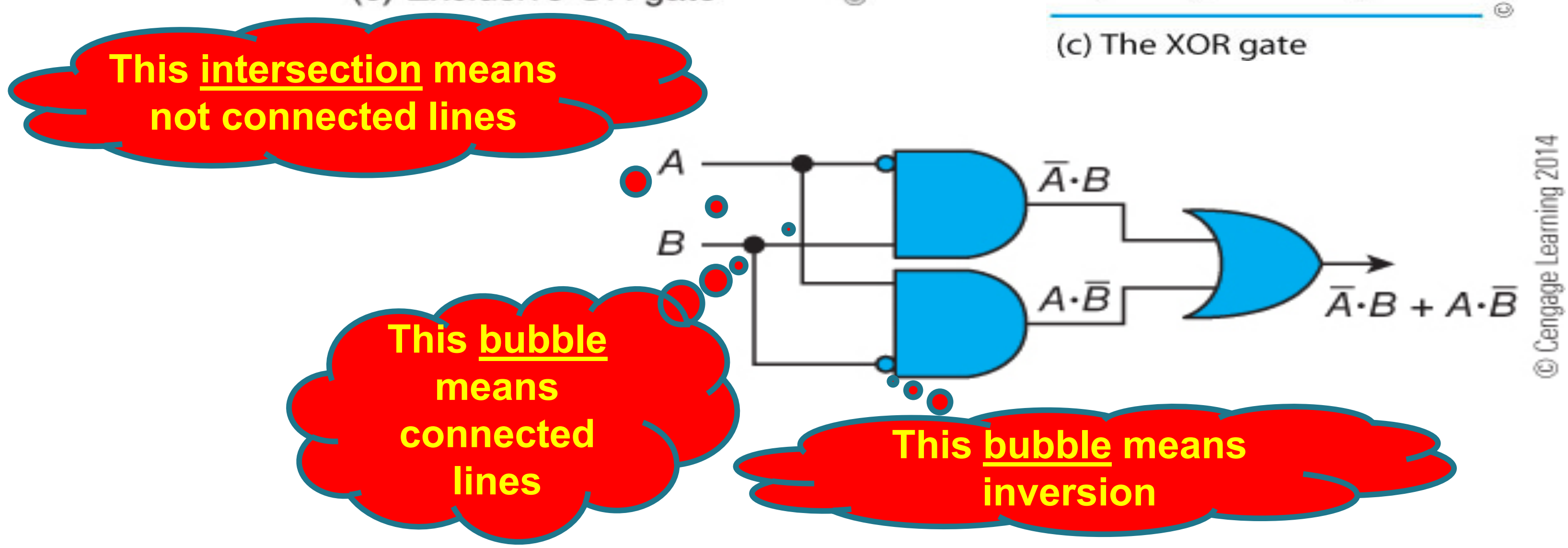
(c) Exclusive OR gate

Exclusive OR Gates

A	B	C = A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

(c) The XOR gate

© Cengage Learning 2014

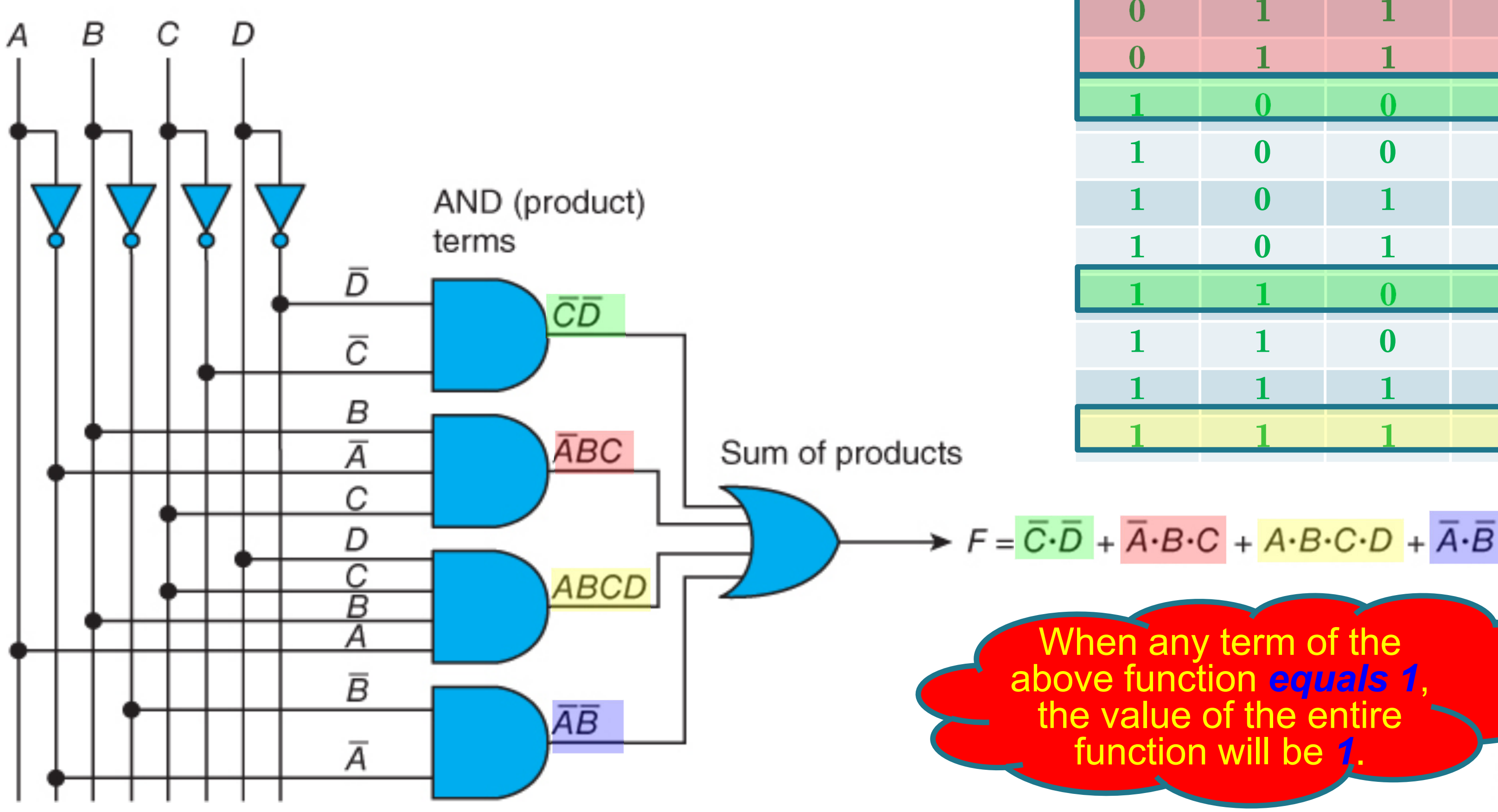


The *sum of products* truth table is identified by its 1's as output

Example of a Digital Circuit

- This is called a ***sum of products*** circuit.
- The output is the **OR** of **AND** terms
- Lines that cross each other ***without a black dot*** at their intersection are ***not connected*** together
- lines that ***meet at a dot*** are **connected**.

FIGURE 2.17 The generic AND-OR circuit



A	B	C	D	O/P
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

When any term of the above function ***equals 1***, the value of the entire function will be ***1***.

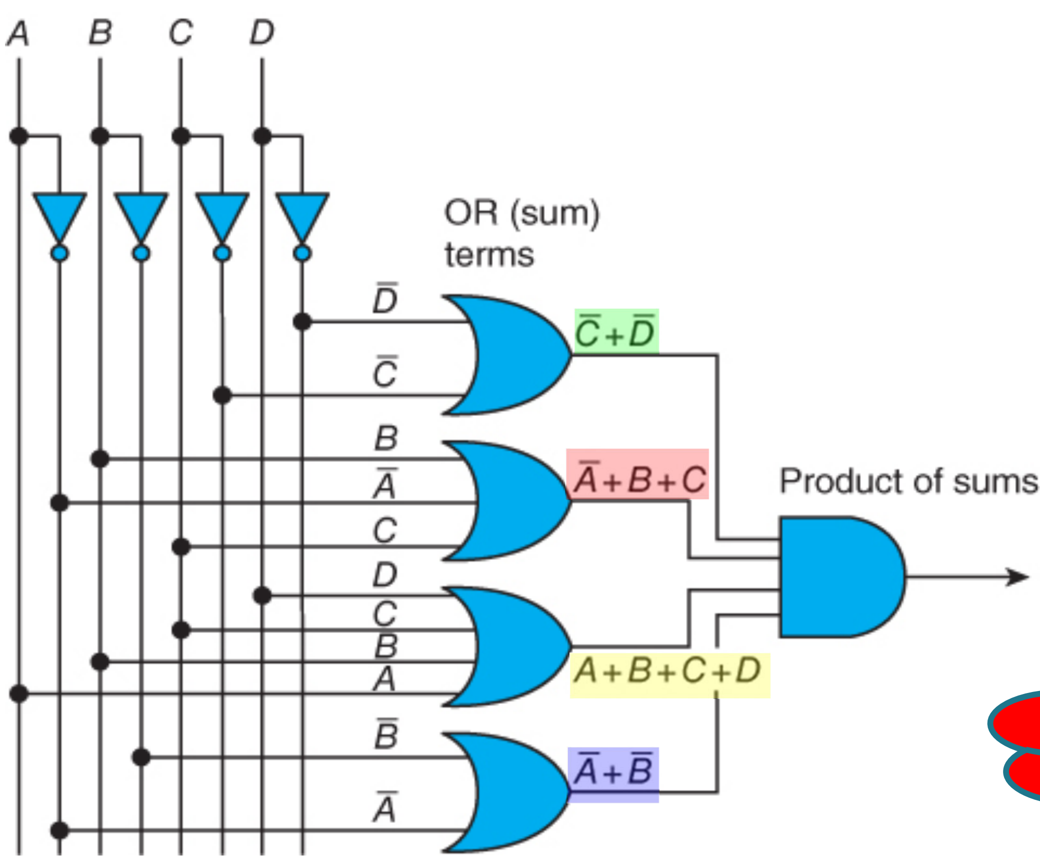
Example of a Digital Circuit

The *product of sums* truth table is identified by its 0's as output

- This is called a ***product of sums*** circuit.
- The output is the **AND** of **OR** terms

A	B	C	D	O/P
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

FIGURE 2.18 The generic OR-AND circuit



$$F = (\bar{C}+\bar{D})(\bar{A}+B+C)(A+B+C+D)(\bar{A}+\bar{B})$$

When any term of the above function ***equals 0***, the value of the entire function will be ***0***.

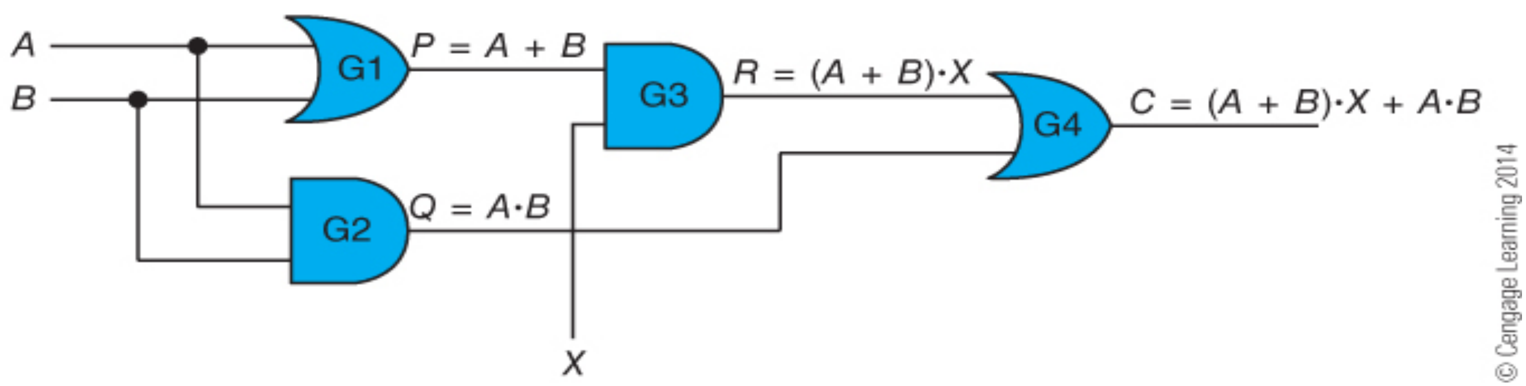
Boolean Algebra Follows Normal Algebraic Laws

- X + Y = Y + X (Commutative law)
- X . Y = Y . X (Commutative law)
- X + (Y + Z) = (X + Y) + Z (Associative law)
- X . (Y . Z) = (X . Y) . Z (Associative law)
- X + Y . Z = (X + Y). (X + Z) (Distributive law)
- X . (Y + Z) = X . Y + X . Z (Distributive law)
- $\overline{X+Y} = \bar{X}.\bar{Y}$ (De Morgan's law)
- $\overline{X.Y} = \bar{X}+\bar{Y}$ (De Morgan's law)
- $X+\bar{X}.Y = X+Y$

More Example of a Digital Circuit

- Figure 2.21 describes a circuit with
  - four gates, labeled **G1**, **G2**, **G3** and **G4**.
  - three inputs A, B, and X, and
  - an output C.
  - It also has three intermediate logical values labeled P, Q, and R.
- We can treat a gate as a ***processor*** that operates on its inputs according to its logical function;
  - For example, the inputs to gate **G3** are P and X, and its output is P · X.
  - Because P = A + B, the output of **G3** is (A + B) · X.
  - Similarly, the output of gate **G4** is R + Q,
  - Because R = (A + B) · X and Q = A · B, the output of gate **G4** is (A + B) · X + A · B.

FIGURE 2.21 A circuit with four gates



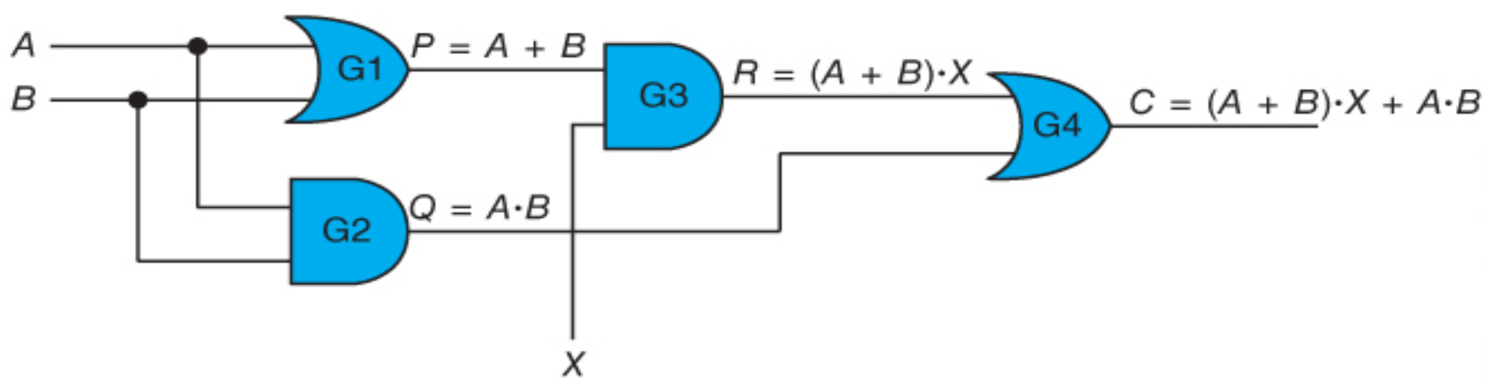
More Example of a Digital Circuit

- Table 2.12 gives the truth table for Figure 2.21.
- Note that the ***output corresponds to the carry out of a 3-bit adder***.

TABLE 2.12 Truth Table for Figure 2.21

Inputs			Intermediate Values			Output
X	A	B	$P = A + B$	$Q = A \cdot B$	$R = (A + B) \cdot X$	$C = Q + R$
0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	1	0	1	0	0	0
0	1	1	1	1	0	1
1	0	0	0	0	0	0
1	0	1	1	0	1	1
1	1	0	1	0	1	1
1	1	1	1	1	1	1

FIGURE 2.21 A circuit with four gates



The Half-Adder and Full-Adder

- Table 2.13 gives the truth table of a *half-adder* that adds bit A to bit B to get a sum and a carry.

A single-bit full-adder is a logical circuit that performs an addition operation on three one-bit binary digits

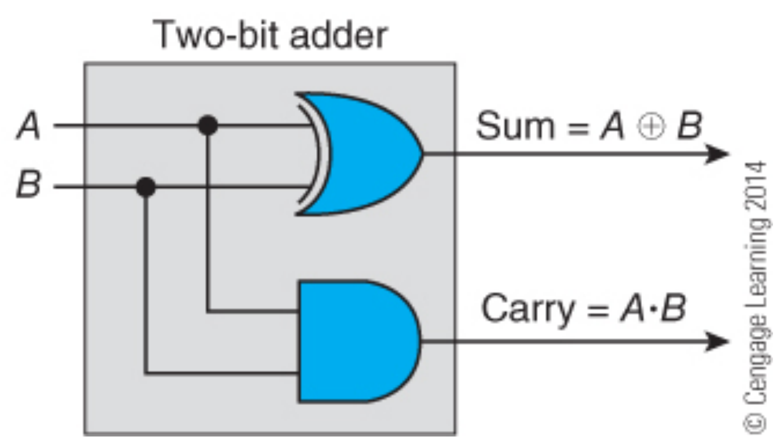
- Figure 2.22 shows the possible structure of a two-bit adder.
  - The carry bit is generated by **AND**ing the two inputs.

TABLE 2.13

Truth Table of a Half Adder

A	*B*	Sum	*C*
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

FIGURE 2.22 The two-bit adder (the half adder)



Full-Adder Circuit

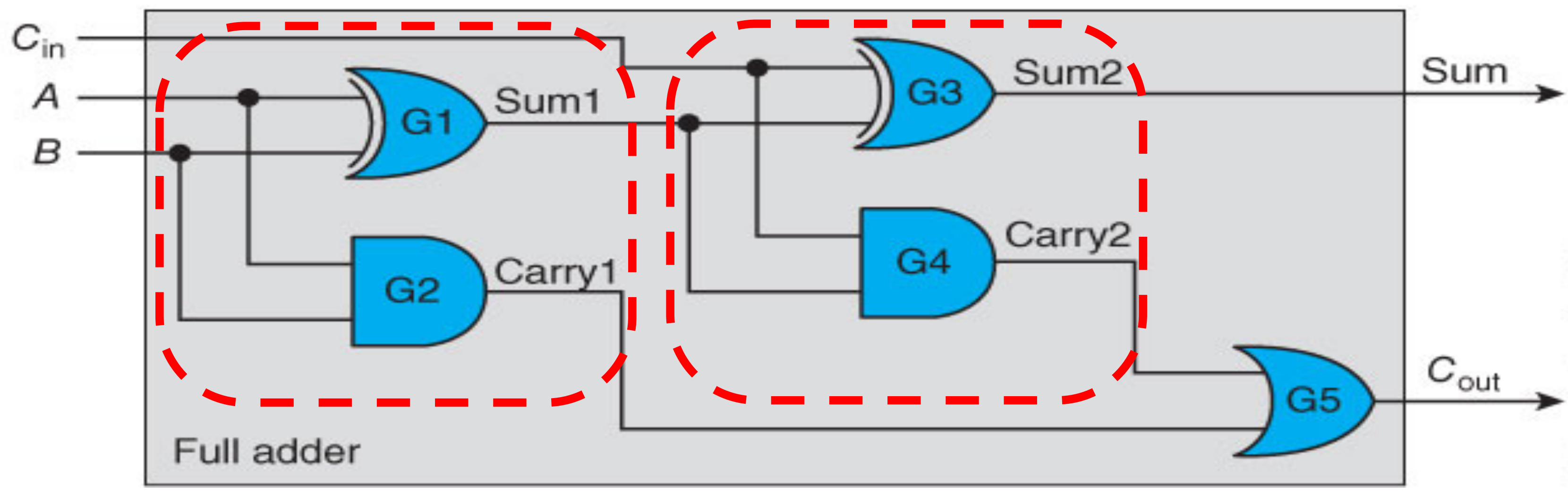
- Figure 2.3 gives the possible circuit of a *one-bit full-adder*.
  - Consists of *two half-adder* and a *one* ***OR*** gate

$$\text{Sum} = (A \oplus B) \oplus C_{in}$$
$$= (A \cdot \overline{B} + \overline{A} \cdot B) \cdot \overline{C_{in}} + \overline{(A \cdot \overline{B} + \overline{A} \cdot B)} \cdot C_{in}$$

$$C_{out} = A \cdot B + (A \oplus B) \cdot C_{in}$$

A	B	C_{in}	Sum	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	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

FIGURE 2.23 The full adder



Full-Adder Circuit

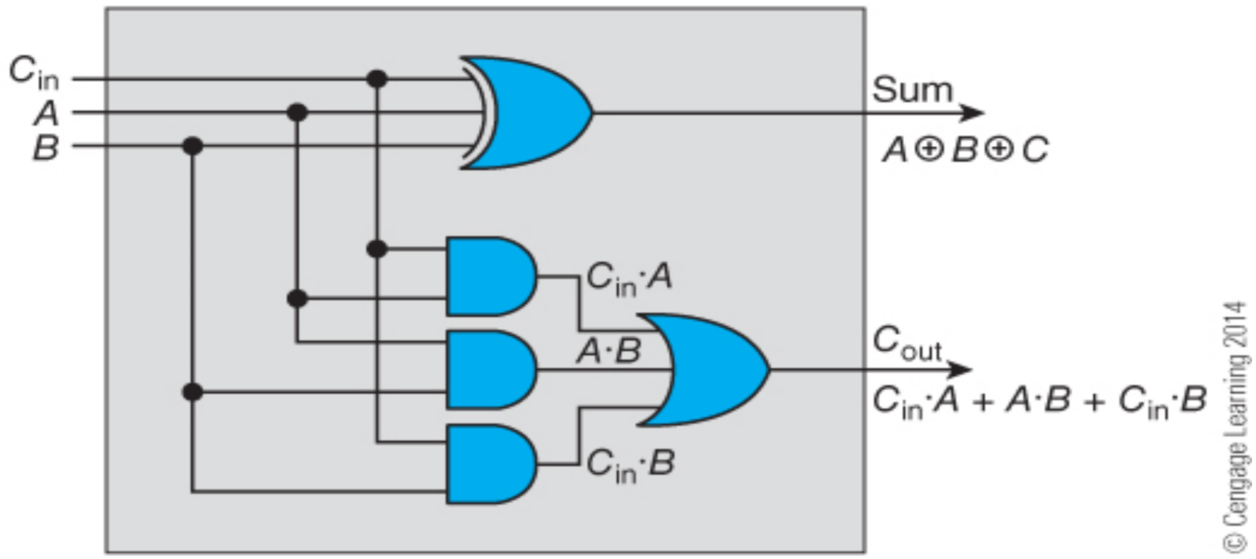
- Figure 2.3 gives an alternative circuit of a *one-bit full-adder*.

$$\text{Sum} = (A \oplus B) \oplus C_{in}$$
$$= (A \cdot \overline{B} + \overline{A} \cdot B) \cdot \overline{C_{in}} + \overline{(A \cdot \overline{B} + \overline{A} \cdot B)} \cdot C_{in}$$

$$C_{out} = C_{in} \cdot A + A \cdot B + C_{in} \cdot B$$

A	B	C_{in}	Sum	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	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

FIGURE 2.24 Alternative full adder circuit



Full-Adder Circuit

Sum $= (A \oplus B) \oplus C$

$= (A.\overline{B} + \overline{A}.B).\overline{C} + \overline{(A.\overline{B} + \overline{A}.B)}.C$

Using De Morgan's law: $\overline{X + Y} = \overline{X}.\overline{Y}$

$= (A.\overline{B} + \overline{A}.B).\overline{C} + (\overline{(A.\overline{B})}.\overline{(\overline{A}.B)}).C$

Using De Morgan's law: $\overline{X.Y} = \overline{X} + \overline{Y}$

$= (A.\overline{B} + \overline{A}.B).\overline{C} + ((\overline{A} + \overline{\overline{B}}).(\overline{\overline{A}} + \overline{B})).C$

Using property $\overline{\overline{A}} = A$

$= (A.\overline{B} + \overline{A}.B).\overline{C} + ((\overline{A} + B).(A + \overline{B})).C$

Using Distributive law: $X.(Y + Z) = X.Y + X.Z$

$= (A.\overline{B} + \overline{A}.B).\overline{C} + ((\overline{A} + B).A + (\overline{A} + B).\overline{B})).C$

Using Commutative law: $X.Y = Y.X$

$= \overline{C}.(A.\overline{B} + \overline{A}.B) + (A.(\overline{A} + B) + \overline{B}.(\overline{A} + B)).C$

Using Distributive law: $X.(Y + Z) = X.Y + X.Z$

$= (\overline{C}.A.\overline{B} + \overline{C}.\overline{A}.B) + ((A.\overline{A} + A.B) + (\overline{B}.\overline{A} + \overline{B}.B)).C$

Using property $\overline{X}.X = 0$

$= (\overline{C}.A.\overline{B} + \overline{C}.\overline{A}.B) + ((0 + A.B) + (\overline{B}.\overline{A} + 0)).C$

Using inversion property: $X + 0 = X$

$= (\overline{C}.A.\overline{B} + \overline{C}.\overline{A}.B) + (A.B + \overline{B}.\overline{A}).C$

Using Commutative law: $X.Y = Y.X$

$= (\overline{C}.A.\overline{B} + \overline{C}.\overline{A}.B) + C.(A.B + \overline{B}.\overline{A})$

Using Distributive law: $X.(Y + Z) = X.Y + X.Z$

$= \overline{C}.A.\overline{B} + \overline{C}.\overline{A}.B + C.A.B + C.\overline{B}.\overline{A}$

Using Commutative law: $X.Y = Y.X$

$= A.\overline{B}.\overline{C} + \overline{A}.B.\overline{C} + A.B.C + \overline{A}.\overline{B}.C$

A	B	C	Sum
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

do not need to know derivation in this course
↳ 2209.

Full-Adder Circuit

$C_{out} = (A + B)\cdot C + A\cdot B$ ***From Figure 2.21***

expanding

$C_{out} = C\cdot A + A\cdot B + C\cdot B$ ***From Figure 2.24***

$C_{out} = A.B + (A \oplus B).C$ ***From Figure 2.23***

$C_{out} = A.B + (A.\overline{B} + \overline{A}.B).C$

Using Distributive law

$C_{out} = A.B + A.\overline{B}.C + \overline{A}.B.C$

Using Distributive law

$C_{out} = A.(B + \overline{B}.C) + \overline{A}.B.C$

Using $X + \overline{X}.Y = X + Y$

$C_{out} = A.(B + C) + \overline{A}.B.C$

Using Distributive law

$C_{out} = A.B + A.C + \overline{A}.B.C$

Using Distributive law

$C_{out} = A.B + (A + \overline{A}.B).C$

Using $X + \overline{X}.Y = X + Y$

$C_{out} = A.B + (A + B).C$

Using Distributive law

$C_{out} = A.B + A.C + B.C$

Using Commutative law

$C_{out} = C\cdot A + A\cdot B + C\cdot B$ ***As in Figure 2.24***

Using Commutative law:

$C_{out} = A\cdot C + B\cdot C + A\cdot B$

Using Distributive law

$C_{out} = (A + B)\cdot C + A\cdot B$ ***As in Figure 2.21***

prove 2 expressions are equivalent

NOT part of this course.

A	B	C	C_{out}
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Full-Adder

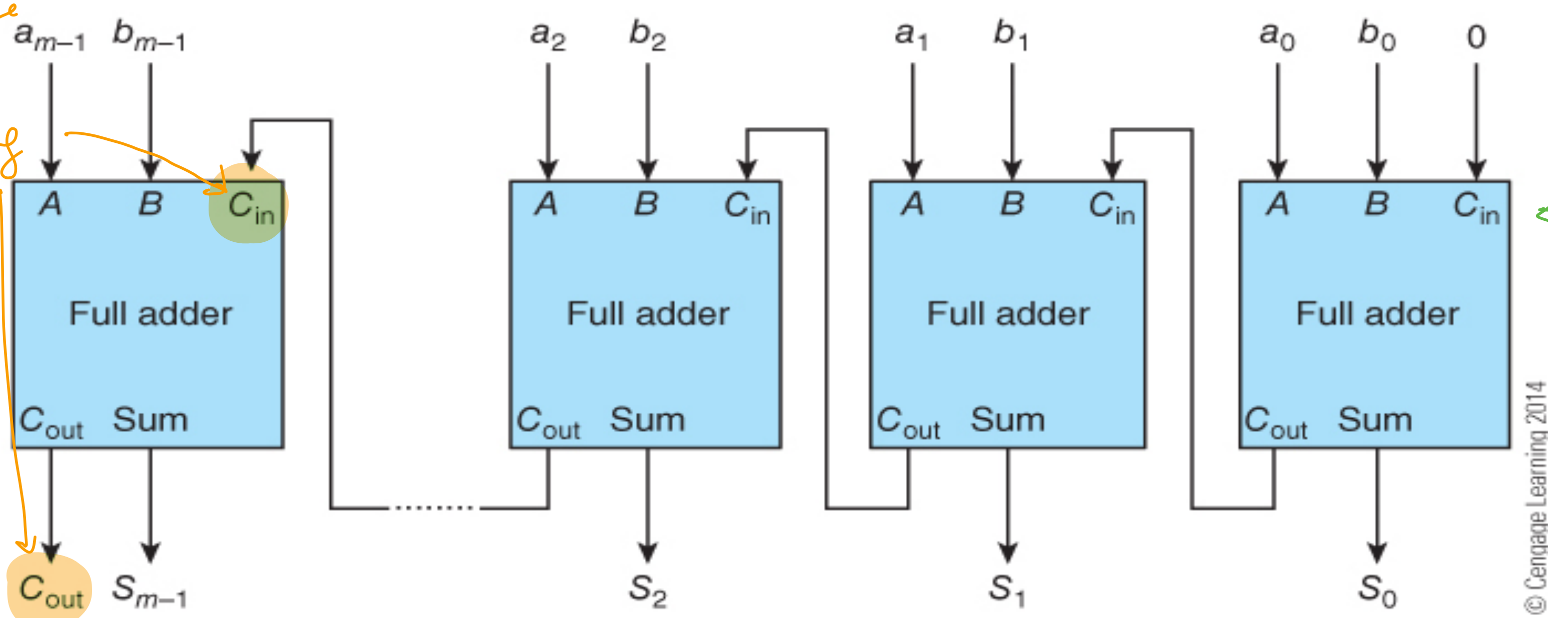
- We need *m* *full-adder* circuits to add two *m*-bit words *in parallel* as Figure 2.25 demonstrates.
- The m_i *full-adder* adds bit a_i to bit b_i, together with a *carry-in* from the stage on its right, to produce a sum_i and a *carry-out* to the stage on its left.

FIGURE 2.25 The parallel adder

know if there is overflow by examining

No overflow if they are the same.

Put XOR with inverter. (would output 1 if both are 0 or both are 1)

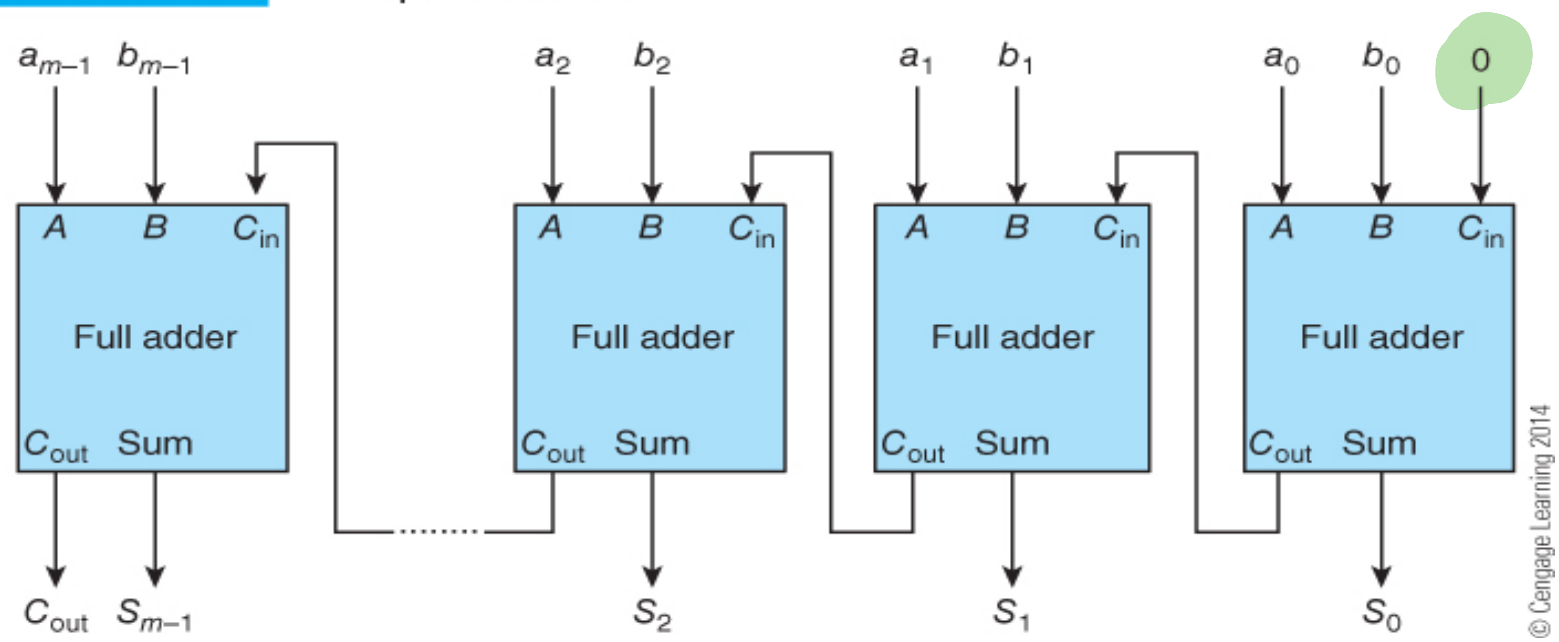


← least significant bit.

Full-Adder

- This circuit is called a parallel-adder because all the bits of the two words to be added are presented to it at the same time.
- The circuit is *not truly parallel* because bit s_i cannot be correctly produced until the *carry-in*$_i$ bit has been calculated by the *previous stage*.
- This is a *ripple through* adder because addition is not complete until the carry bit has ***rippled*** through the circuit.
- *True parallel-adders* use high-speed *look-ahead carry* circuits to produce all carry bits at once, hence speeding up the addition operation.

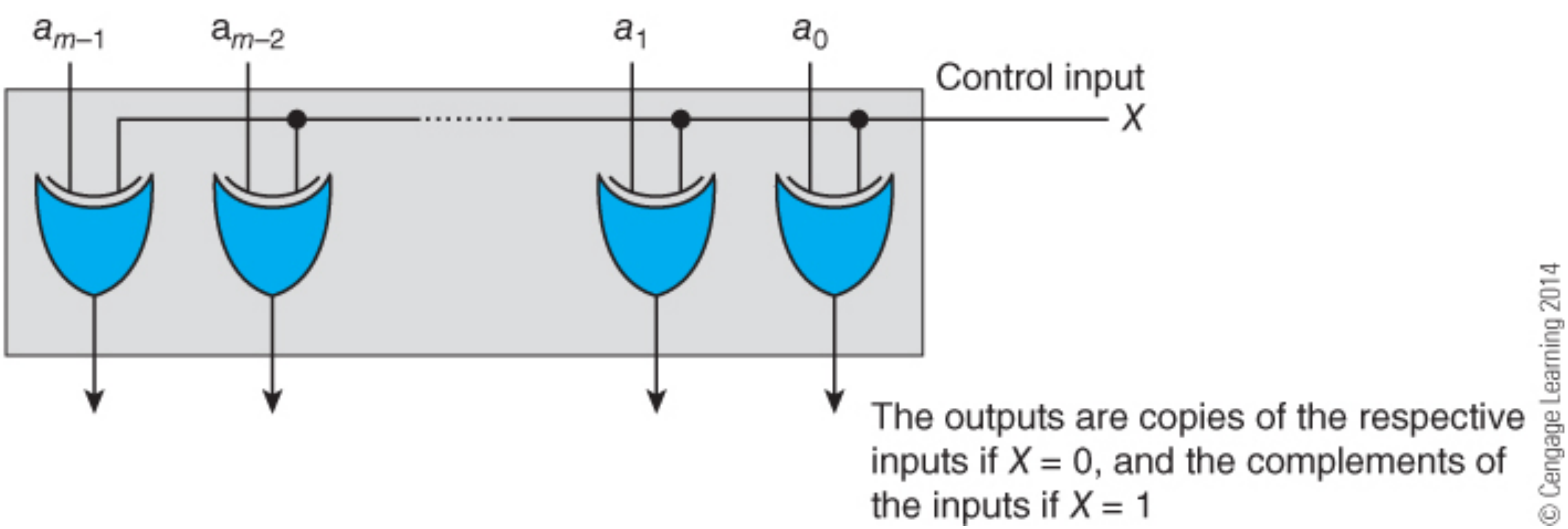
FIGURE 2.25 The parallel adder



Programmable Inverter

a	X	a ⊕ X
0	0	0
0	1	1
1	0	1
1	1	0

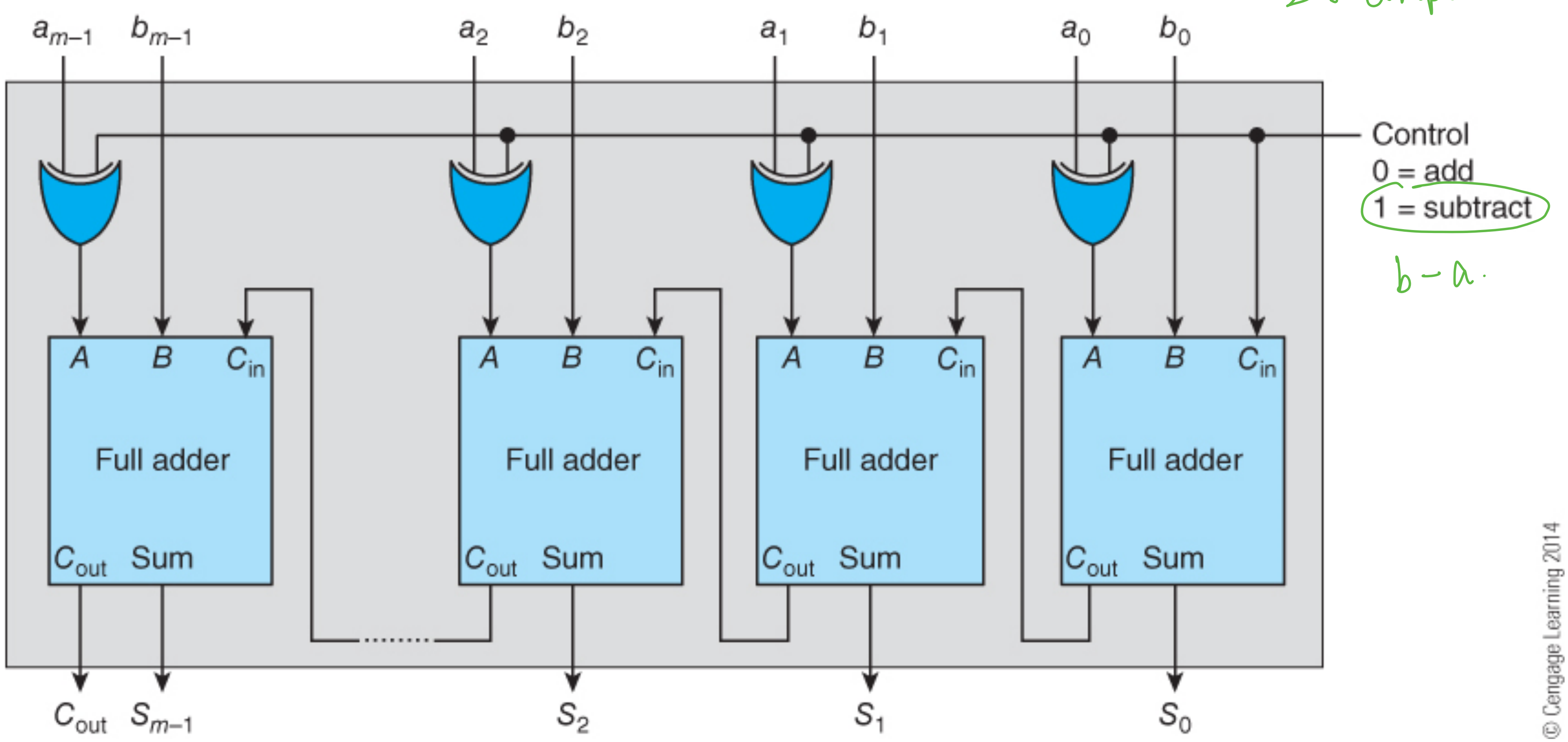
FIGURE 2.26 The programmable inverter



Full-Adder/Subtractor

FIGURE 2.27 The adder/subtractor

2's complement.

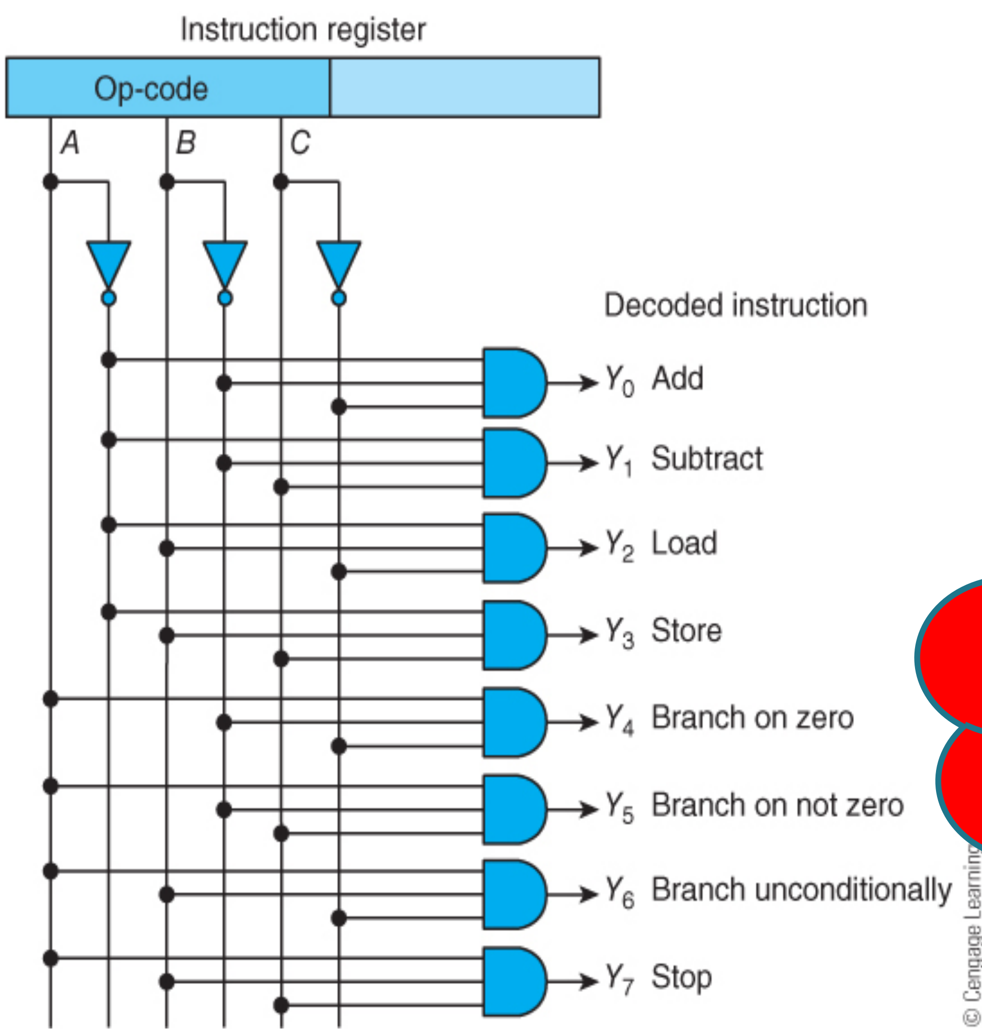


b - a.

The Decoder

- Figure 2.29 has ***three*** inputs A, B, and C, and ***eight*** outputs Y0 to Y7.
- The ***three*** inverters generate the complements of the inputs A, B, and C.
- Each of the ***eight AND*** gates is connected to ***three*** of the six lines .
  - each of the ***three*** variables appear in either its true or complemented form.

FIGURE 2.28 Application of a decoder



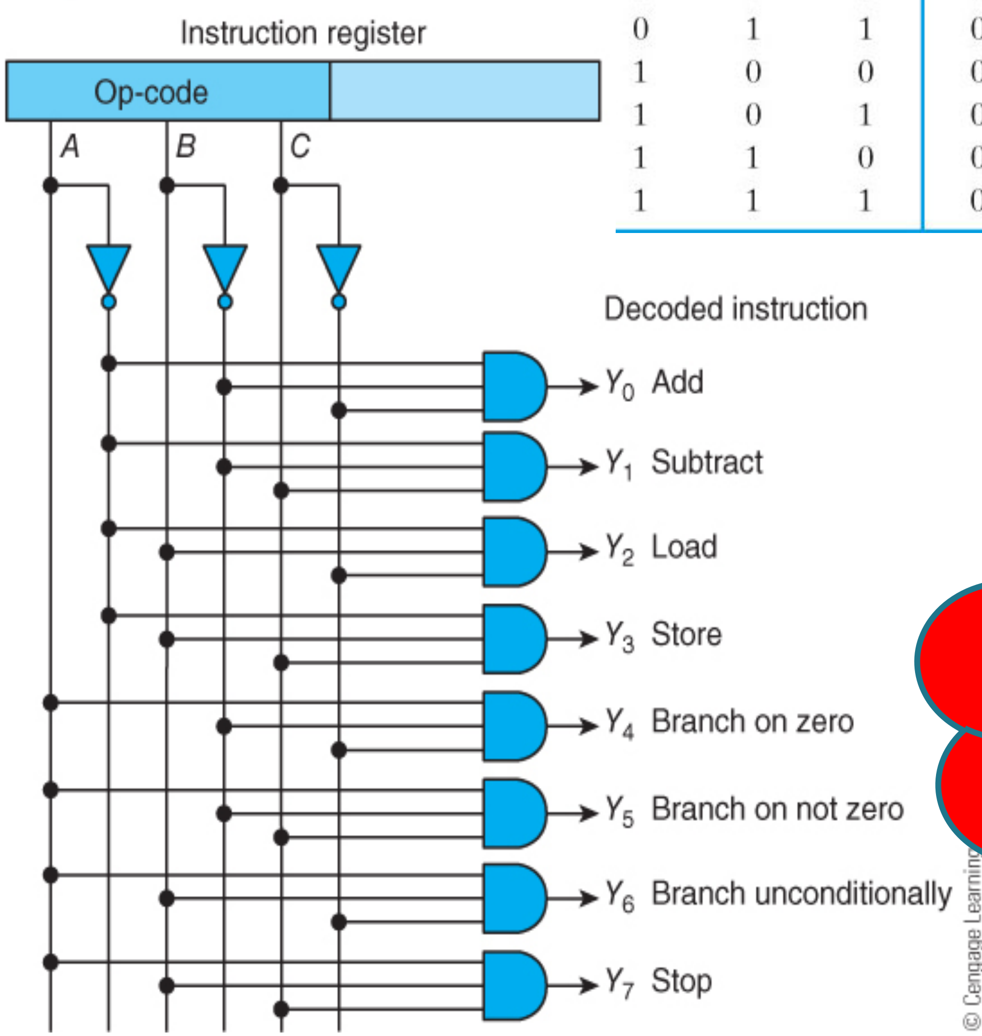
A ***decoder*** is combinational logic circuit that converts binary information from the n-bits coded input to a maximum of 2^n unique outputs.

The Decoder

TABLE 2.15 The Decoder

Inputs			Outputs							
A	*B*	*C*	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

FIGURE 2.28 Application of a decoder

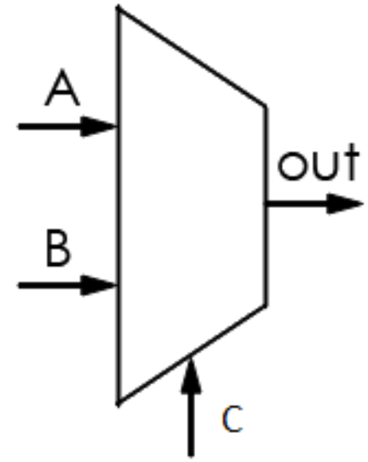


A ***decoder*** is combinational logic circuit that converts binary information from the n-bits coded input to a maximum of 2^n unique outputs.

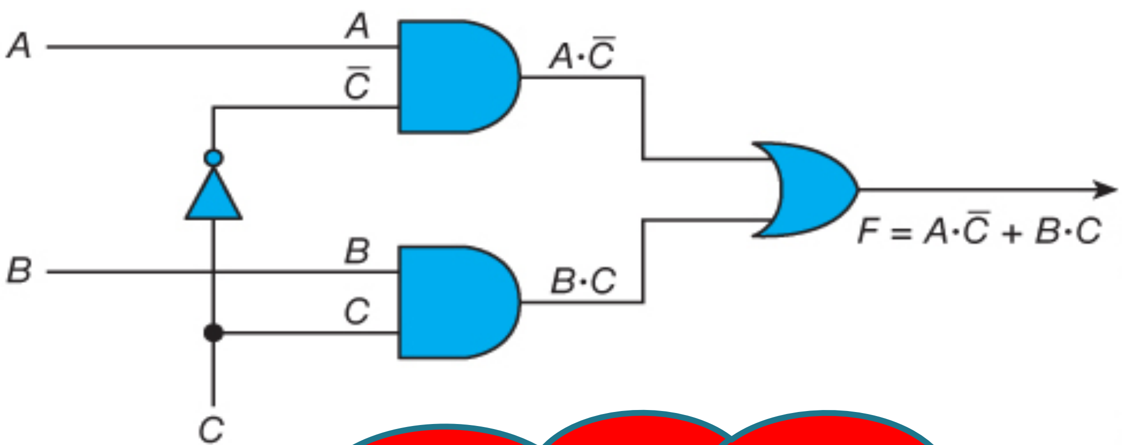
The Multiplexer

- When C = 0, the output is A
- When C = 1, the output is B
- C works as a selector to select either A or B to go

controler selects one to pass.



Alternative representation of the two-input multiplexer



if control = 0, information from A will pass

if control = 1, B will pass.

Truth table

C	A	B	S
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

A ***multiplexer*** is combinational logic circuit that has up to 2^n binary input lines and n select lines, where the n select-lines are used to forward one of the input values to the output line.

The Multiplexer

- When C = 0, the output is A
- When C = 1, the output is B
- C works as a selector to select either A or B to go

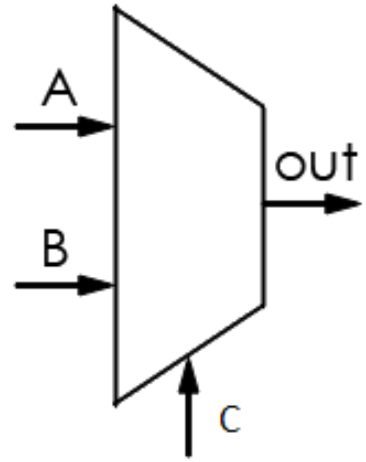
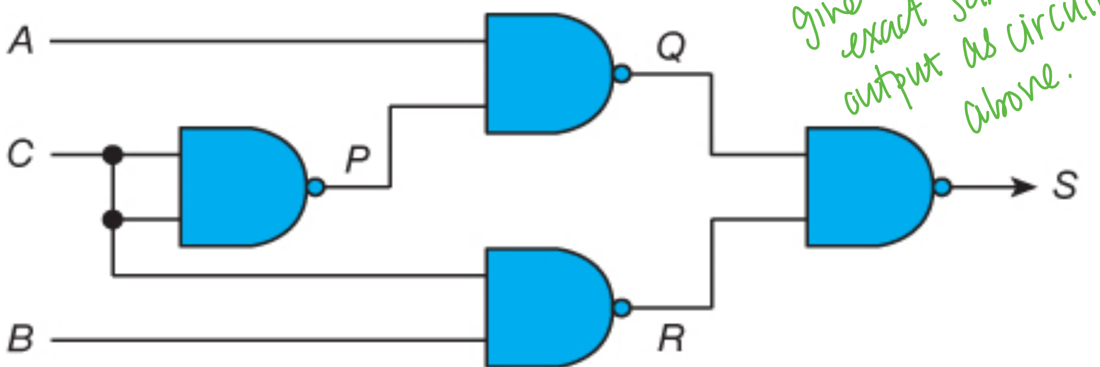


FIGURE 2.29 The two-input multiplexer and its truth table



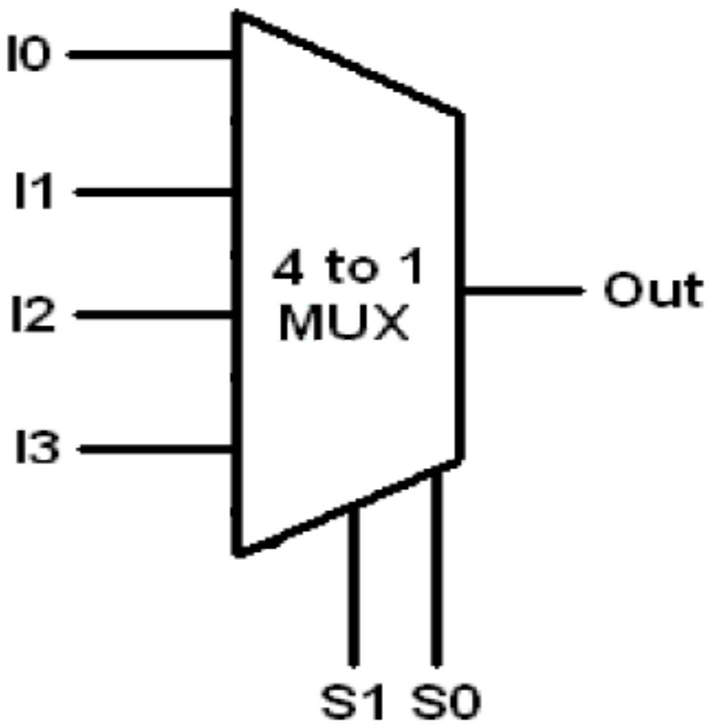
give exact same output as circuit above.

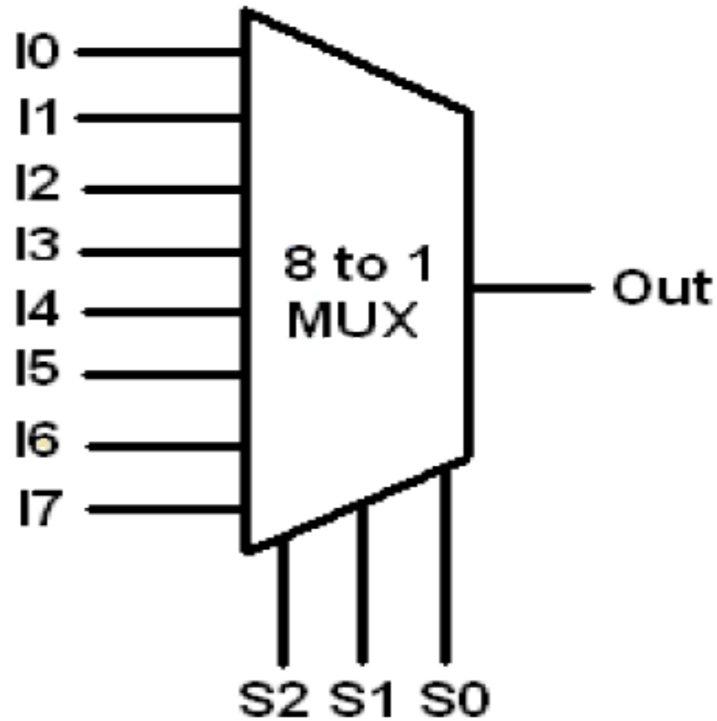
Truth table

C	A	B	P	Q	R	S
0	0	0	1	1	1	0
0	0	1	1	1	1	0
0	1	0	1	0	1	1
0	1	1	1	0	1	1
1	0	0	0	1	1	0
1	0	1	0	1	0	1
1	1	0	0	1	1	0
1	1	1	0	1	0	1

A ***multiplexer*** is combinational logic circuit that has up to 2^n binary input lines and n select lines, where the n select-lines are used to forward one of the input values to the output line.

The Multiplexer

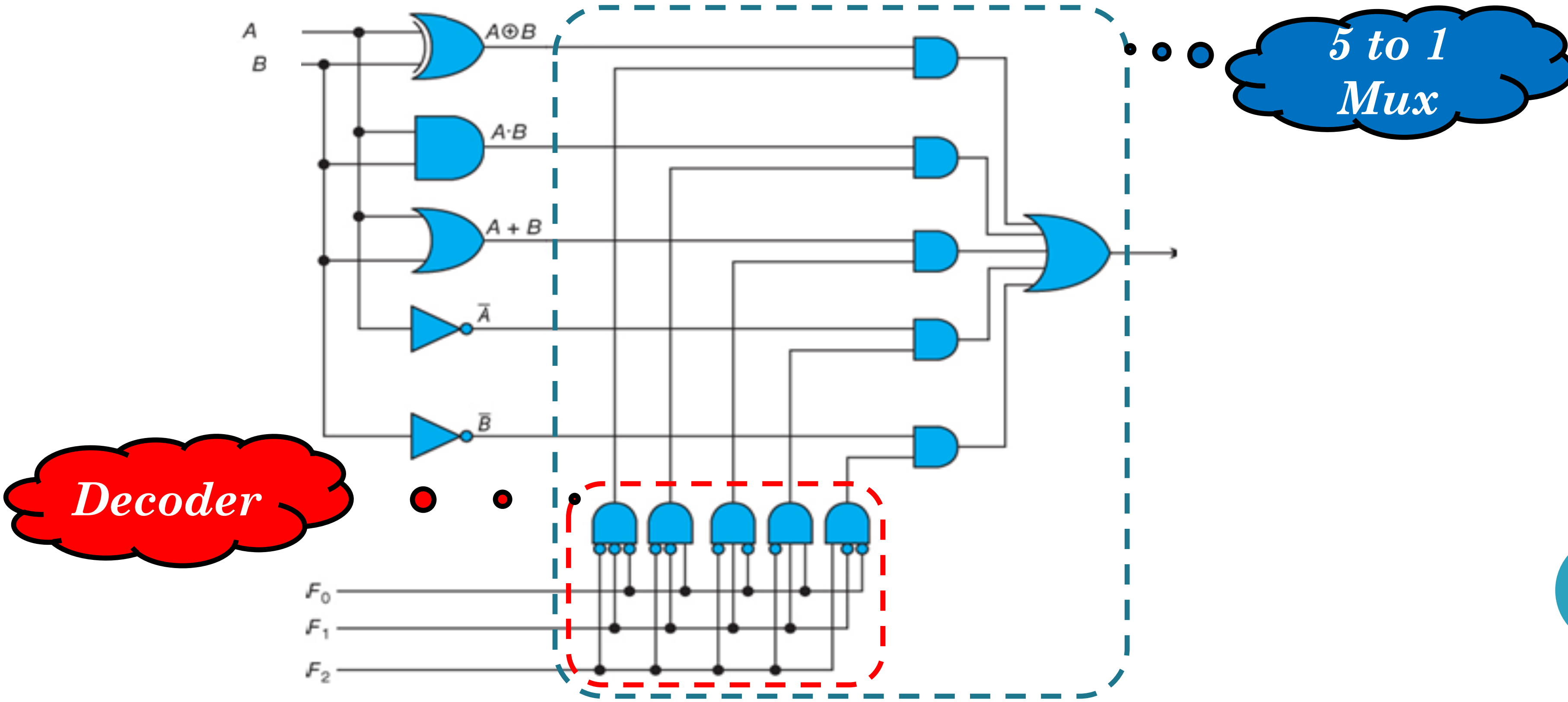


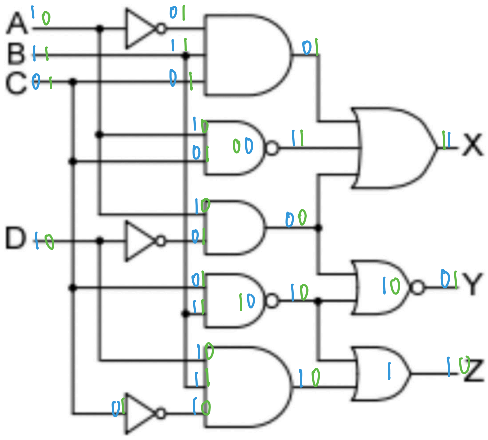


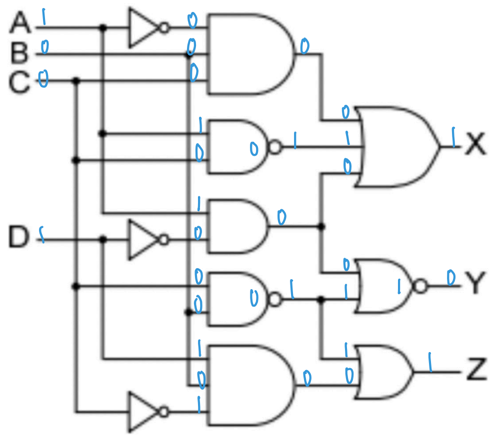
A ***multiplexer*** is combinational logic circuit that has up to 2^n binary input lines and n select lines, where the n select-lines are used to forward one of the input values to the output line.

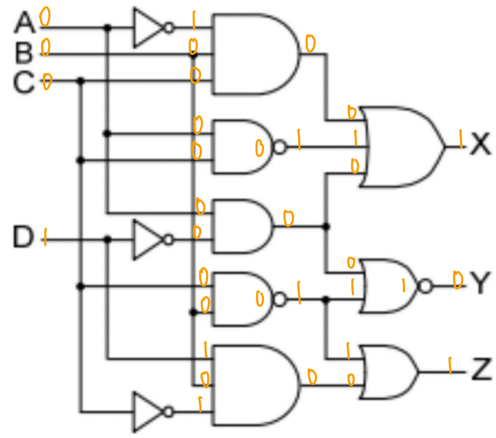
One Bit of an ALU

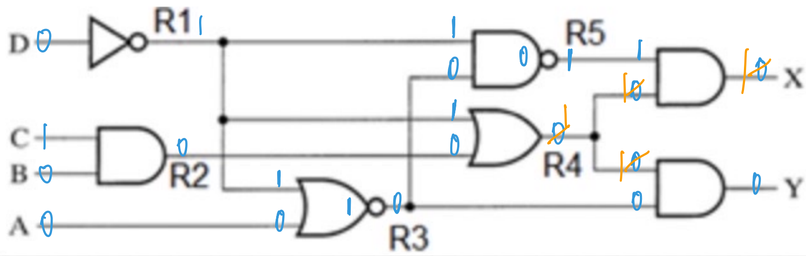
- This diagram describes one-bit of a primitive ALU that can perform five operations on bits A and B (**XOR**, **AND**, **OR**, **NOT A** and **NOT B**).
- The function to be performed is determined by the *three-bit control signal* F2,F1,F0.
- The five functions are generated by the five gates on the left.
- On the right, five **AND** gates are used to gate the selected function to an **OR** gate to produce the output.



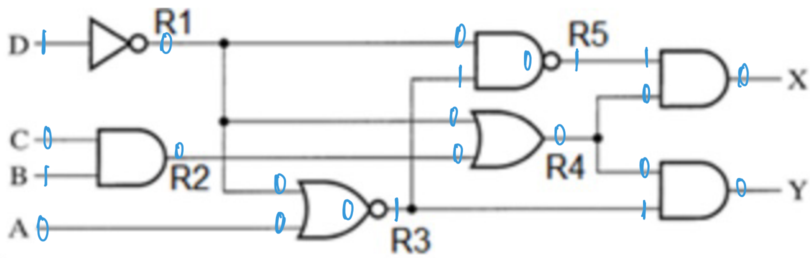

A
B
C
D
X
Y
Z


A
B
C
D
X
Y
Z

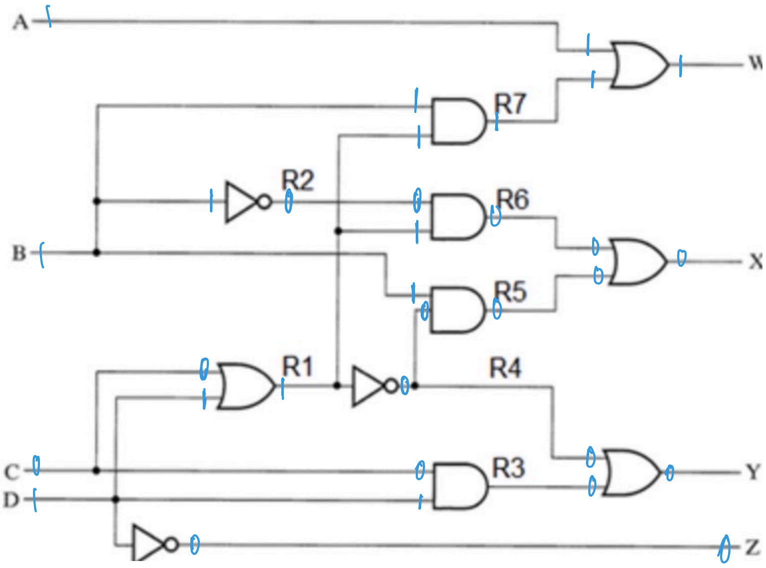

A
B
C
D
X
Y
Z

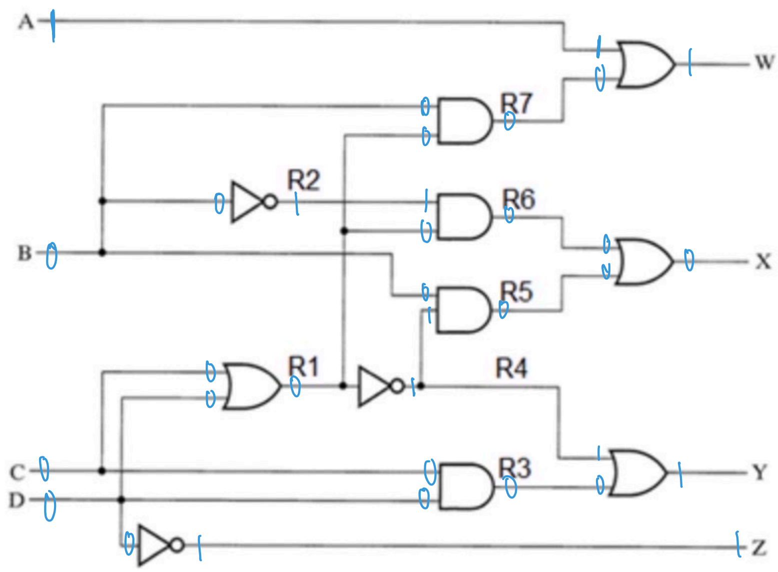

D
C
B
A
R1
R2
R3
R4
R5
X
Y

1


D
C
B
A
R1
R2
R3
R4
R5
X
Y

2

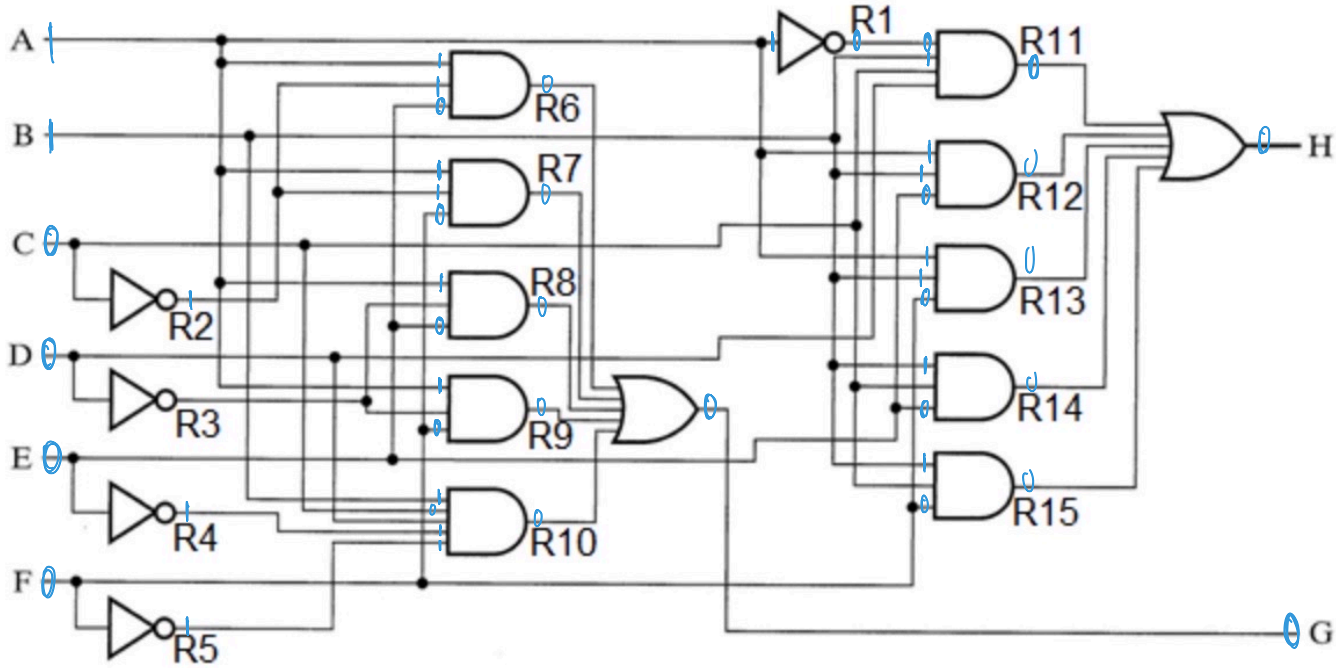

A
B
C
D
R1
R2
R3
R4
R5
R6
R7
W
X
Y
Z


A
B
C
D
R1
R2
R3
R4
R5
R6
R7
W
X
Y
Z

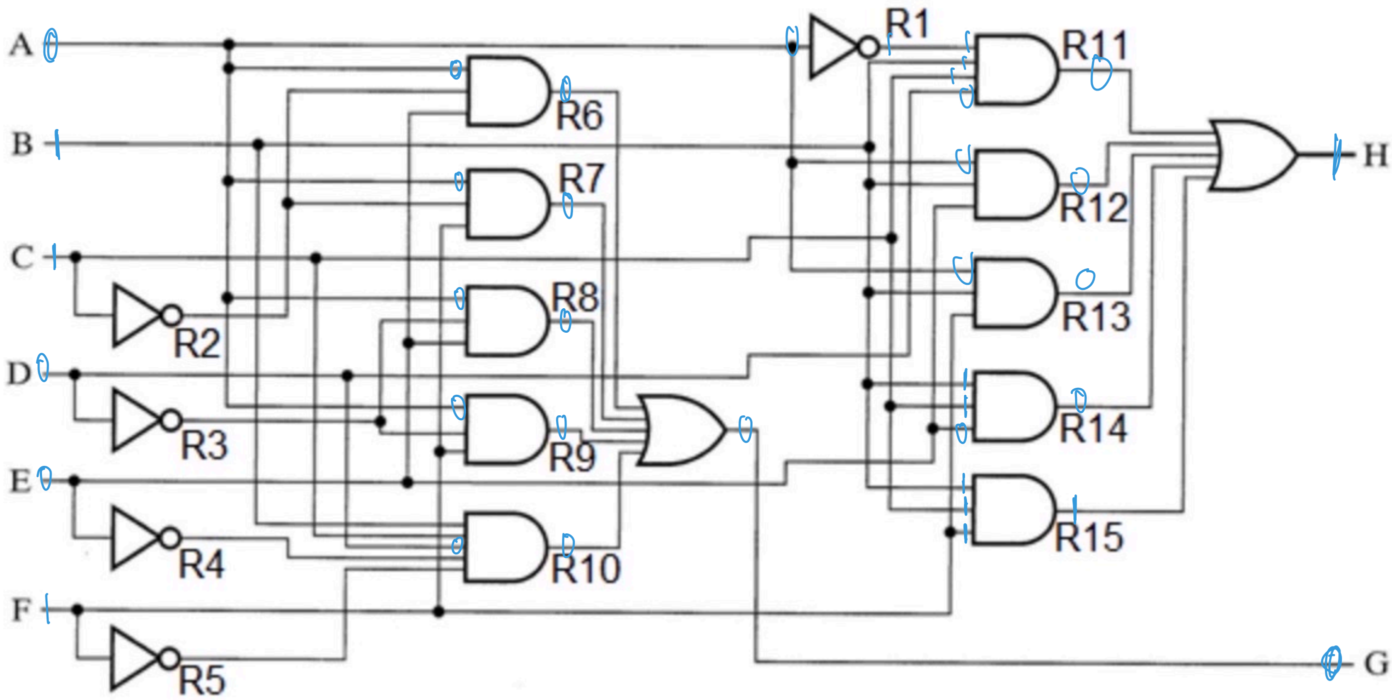
3

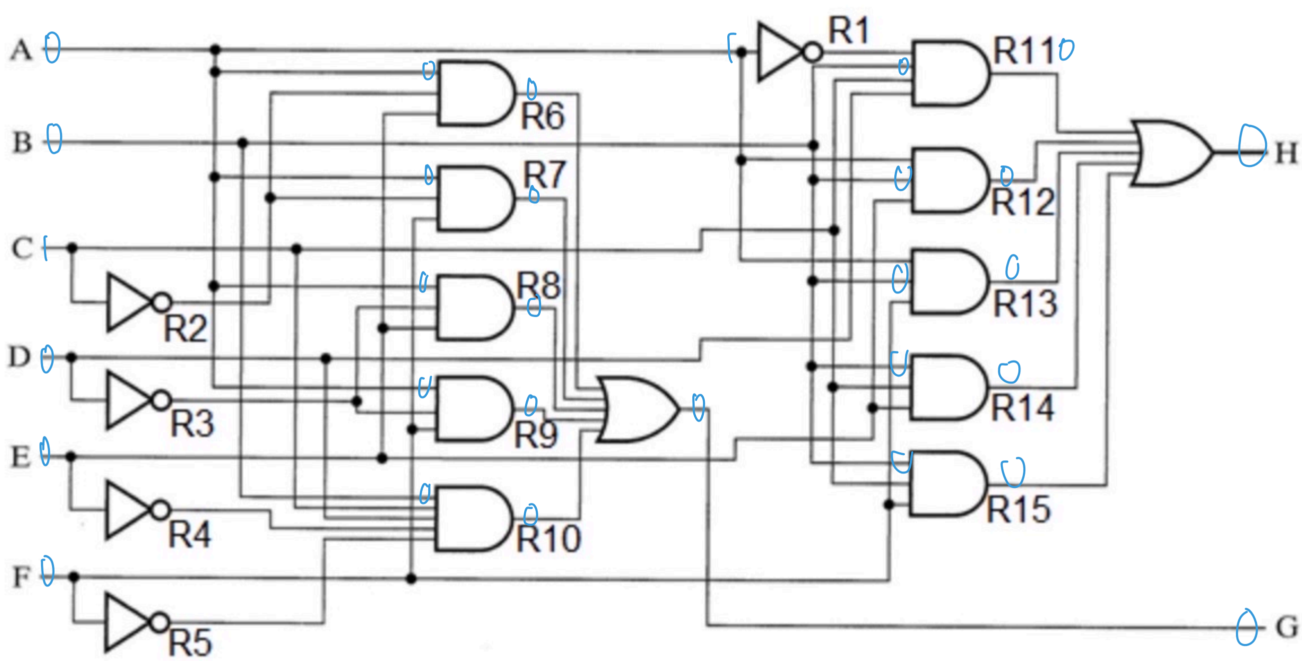
4

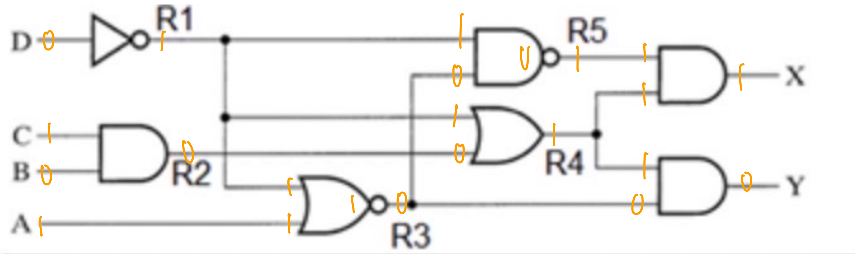
5


A
B
C
D
E
F
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13
R14
R15
H
G

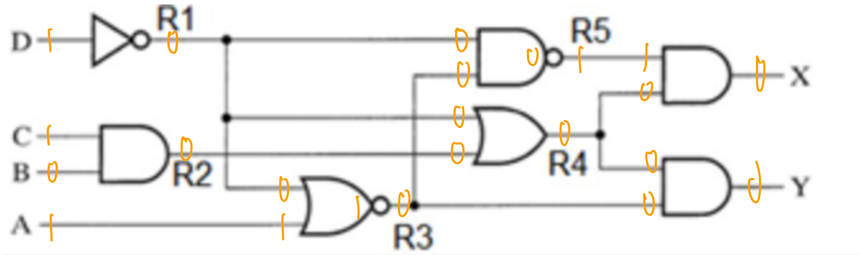
b


A
B
C
D
E
F
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13
R14
R15
H
G

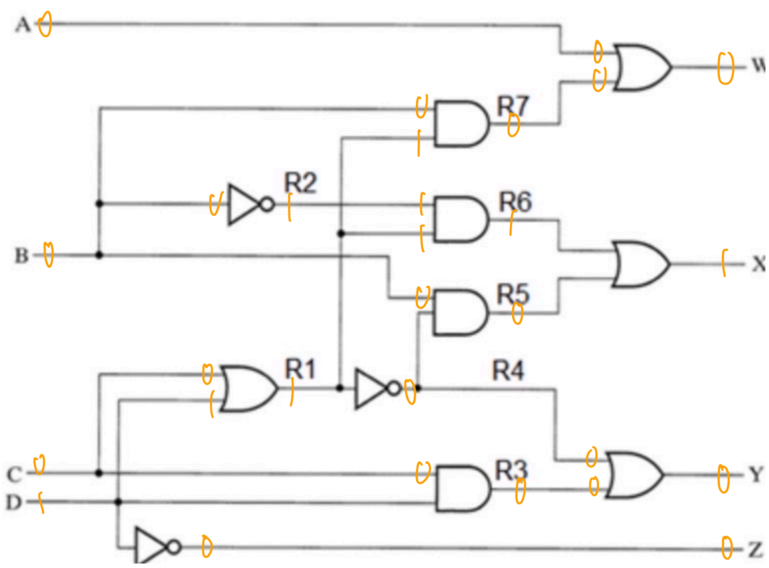

A
B
C
D
E
F
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13
R14
R15
H
G

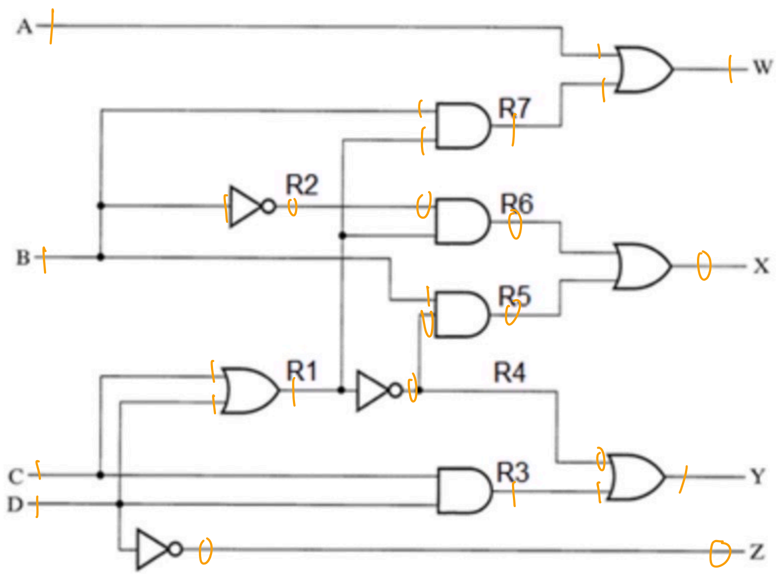

D
C
B
A
R1
R2
R3
R4
R5
X
Y

1


D
C
B
A
R1
R2
R3
R4
R5
X
Y

2

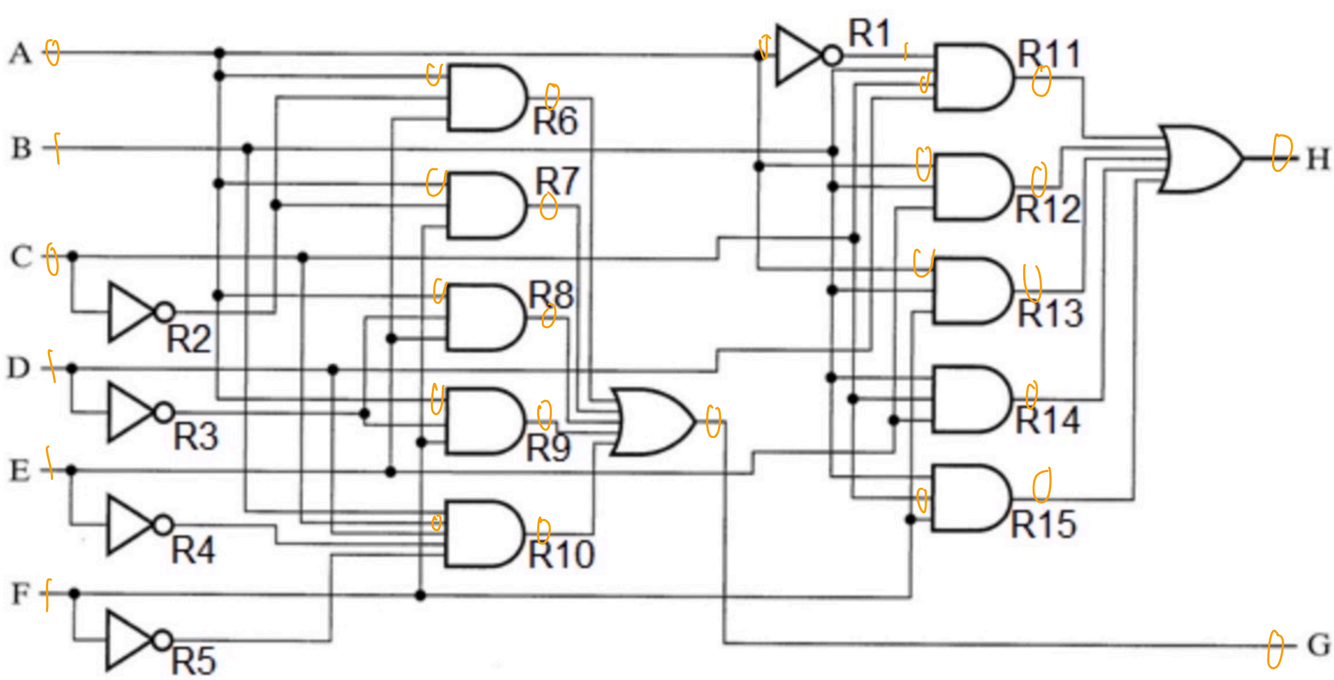

A
B
C
D
R1
R2
R3
R4
R5
R6
R7
W
X
Y
Z


A
B
C
D
R1
R2
R3
R4
R5
R6
R7
W
X
Y
Z

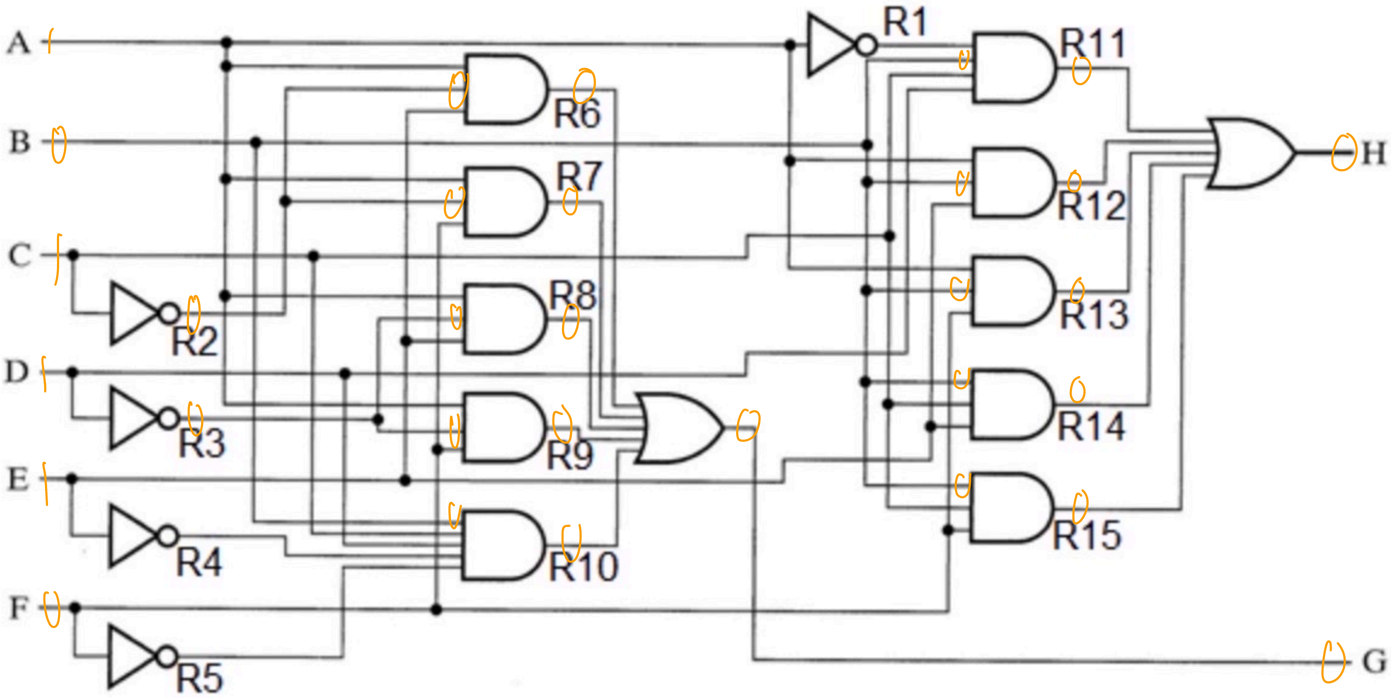
3

4

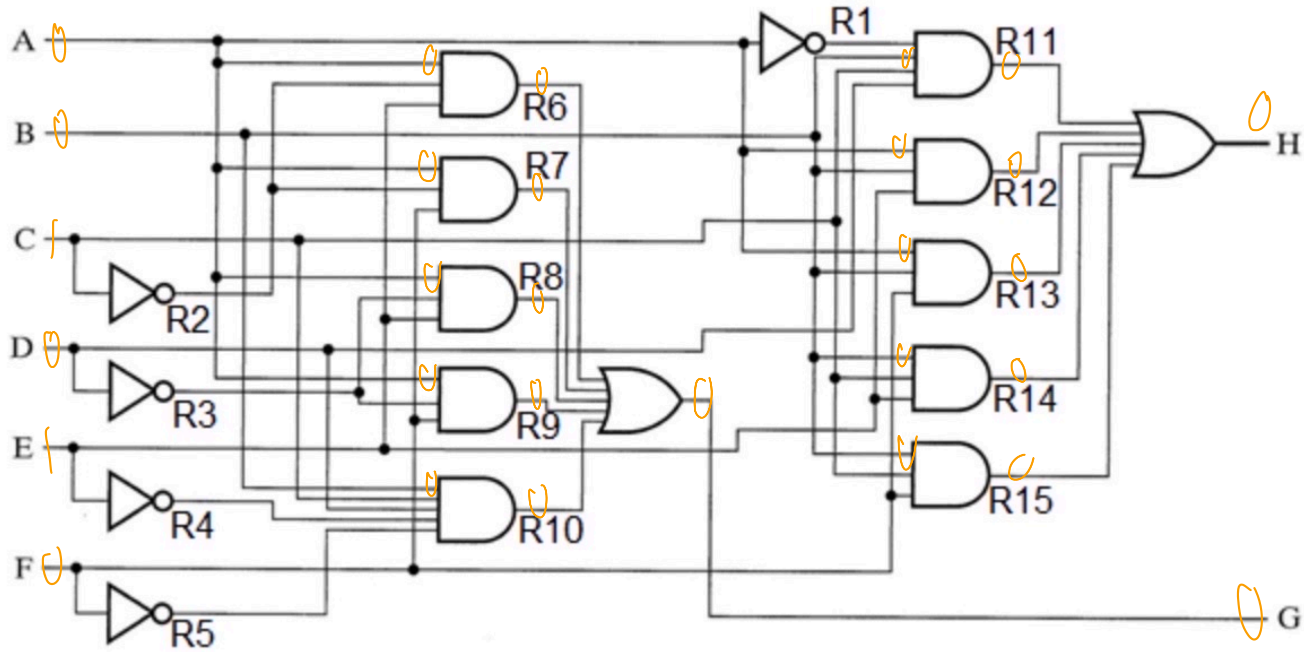
5


A
B
C
D
E
F
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13
R14
R15
H
G

6


A
B
C
D
E
F
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13
R14
R15
H
G

7


A
B
C
D
E
F
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13
R14
R15
H
G