Boolean algebra

- Three basic operations
  - Binary OR +
  - Binary AND •
  - Unary Complement ′

Figure 10.5

Precedence	Operator
Highest	Complement
	AND
Lowest	OR

Ten properties of boolean algebra

- Commutative
- Associative
- Distributive
- Identity
- Complement

Duality

- To obtain the dual expression
  - Exchange + and •
  - Exchange 1 and 0

Commutative

$$x + y = y + x$$

Commutative

$$x + y = y + x$$

$$x \cdot y = y \cdot x$$

Associative

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

Associative

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

$$(x \cdot y) \cdot z = x \cdot (y \cdot z)$$

Distributive

$$x + y \cdot z = (x + y) \cdot (x + z)$$

$$x \cdot (y + z) = x \cdot y + x \cdot z$$

Identity

$$x + 0 = x$$

$$x \cdot 1 = x$$

Complement

$$x + x' = 1$$

$$x \cdot x' = 0$$

Idempotent property

$$x + x = x$$

$$x \cdot x = x$$

Zero theorem

$$x + 1 = 1$$

$$x \cdot 0 = 0$$

Absorption property

$$x + x \cdot y = x$$

$$x \cdot (x + y) = x$$

Consensus theorem

$$x \cdot y + x' \cdot z + y \cdot z = x \cdot y + x' \cdot z$$

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

De Morgan's law

$$(a \cdot b)' = a' + b'$$

$$(a + b)' = a' \cdot b'$$

Complement theorems

$$(x')' = x$$

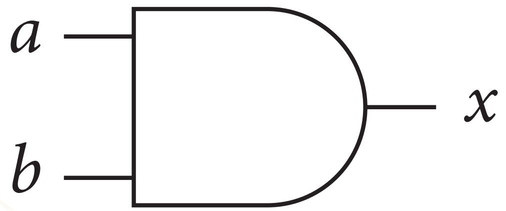
$$1' = 0$$

$$0' = 1$$

Logic diagrams

- An interconnection of logic gates
- Closely resembles the hardware
  - Gate symbol represents a group of transistors and other electronic components
  - Lines connecting gate symbols represent wires

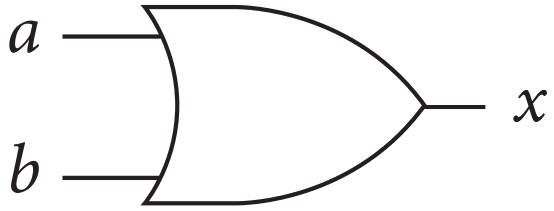
Figure 10.6



$x = a \cdot b$

a	b	x
0	0	0
0	1	0
1	0	0
1	1	1

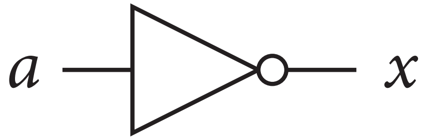
(a) AND gate.



$x = a + b$

a	b	x
0	0	0
0	1	1
1	0	1
1	1	1

(b) OR gate.

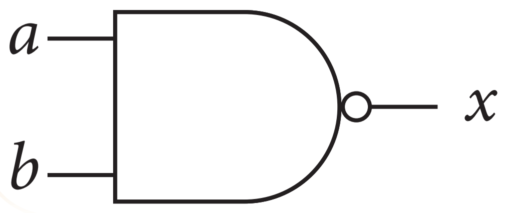


$x = a'$

a	x
0	1
1	0

(c) Inverter.

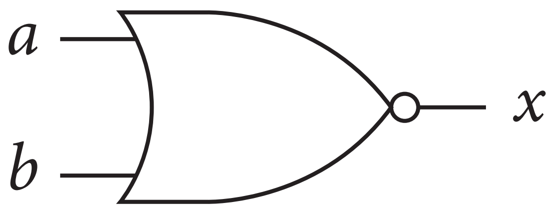
Figure 10.7



$x = (a \cdot b)'$

a	b	x
0	0	1
0	1	1
1	0	1
1	1	0

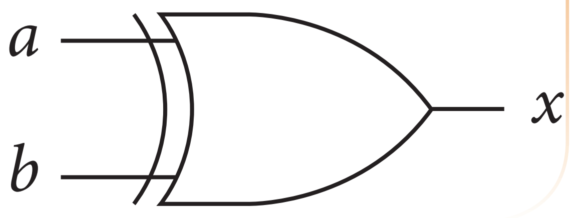
(a) NAND gate.



$x = (a + b)'$

a	b	x
0	0	1
0	1	0
1	0	0
1	1	0

(b) NOR gate.

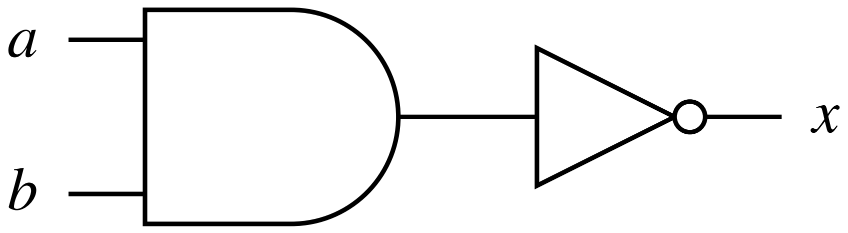


$x = a \oplus b$

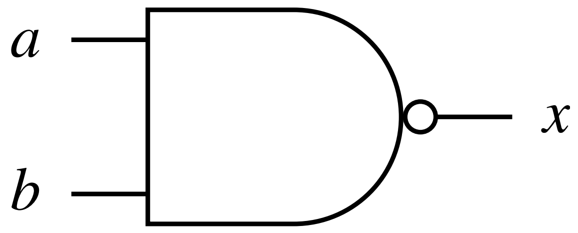
a	b	x
0	0	0
0	1	1
1	0	1
1	1	0

(c) XOR gate.

Figure 10.8



(a) AND inverter.

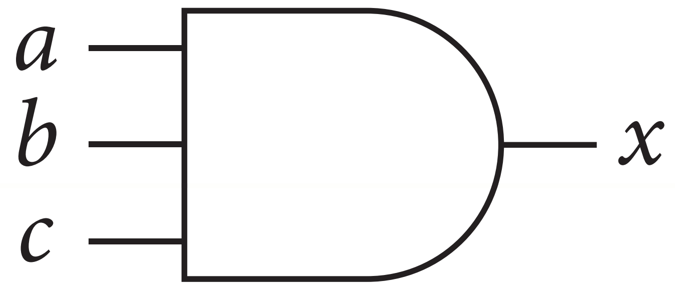


(b) NAND.

Figure 10.9

Precedence	Operator
Highest	Complement
	AND
	XOR
Lowest	OR

Figure 10.10



$$x = a \cdot b \cdot c$$

a	b	c	x
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Boolean expressions and logic diagrams

- AND gate corresponds to AND operation
- OR gate corresponds to OR operation
- Inverter corresponds to complement operation

Figure 10.13

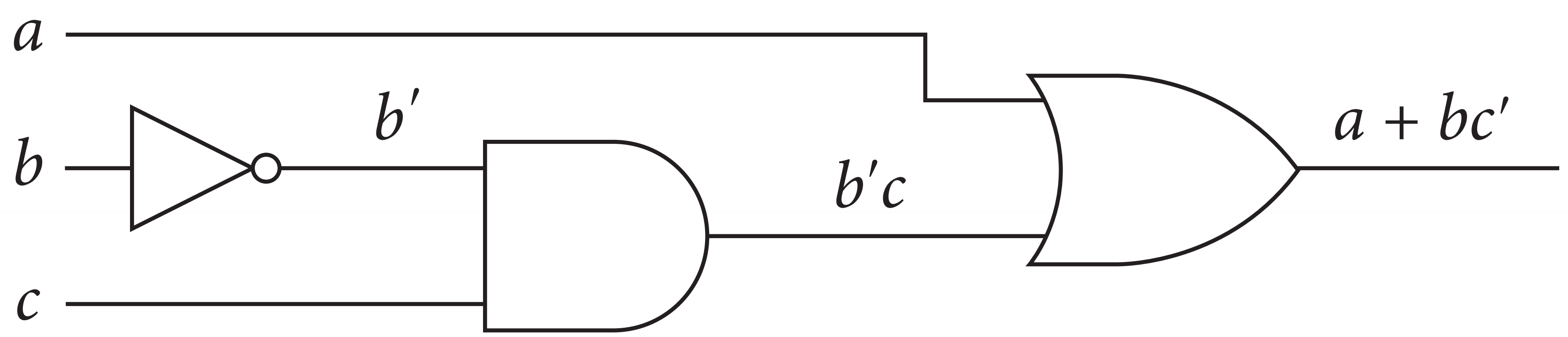
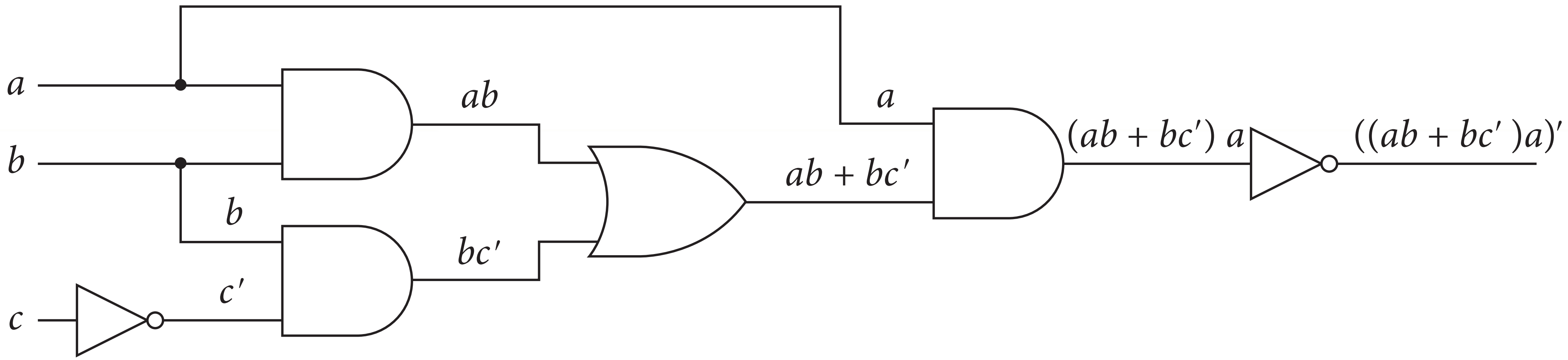


Figure 10.14



Truth tables and boolean expressions

- Given a truth table, write a boolean expression without parentheses as an OR of several AND terms
- Each AND term corresponds to a 1 in the truth table

Figure 10.17

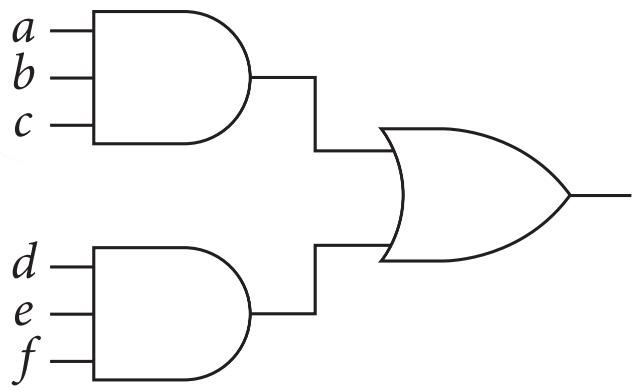
a	*b*	*c*	*x*
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Two-level circuits

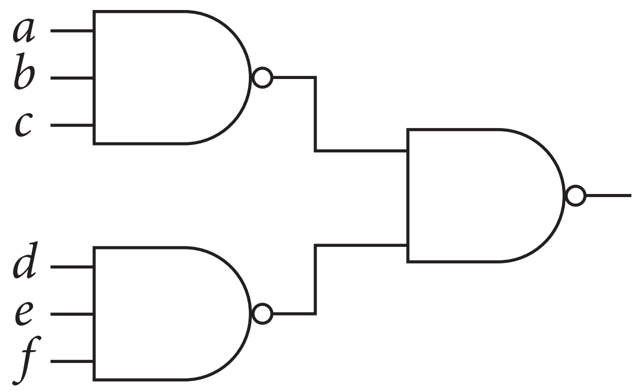
- The *gate delay* is the time it takes for the output of a gate to respond to a change in its input
- Any combinational circuit can be transformed into an AND-OR circuit or an OR-AND circuit with at most two gate delays (not counting the gate delay of any inverters)

Figure 10.22

$$abc + def = [(abc)'(def)']'$$



(a) An AND-OR circuit.



(b) The equivalent NAND-NAND circuit.

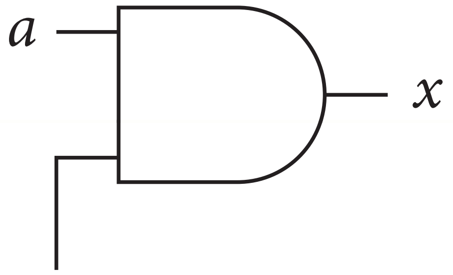
Canonical expressions

- A *minterm* is a term in an AND-OR expression in which all input variables occur exactly once
- A *canonical expression* is an OR of minterms in which no two identical minterms appear
- A canonical expression is directly related to a truth table because each minterm in the expression represents a 1 in the truth table

Enable lines

- An enable line to a combinational device turns the device on or off
  - If enable = 0 the output is 0 regardless of any other inputs
  - If enable = 1 the device performs its function with the output depending on the other inputs

Figure 10.41



(a) Logic diagram of enable gate.

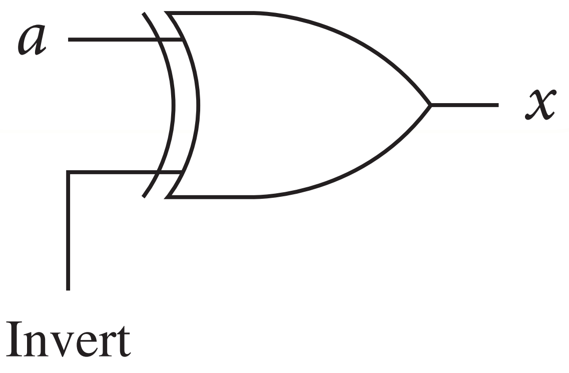
Enable = 1	
a	***x***
0	0
1	1

(b) Truth table with the device turned on.

Enable = 0	
a	***x***
0	0
1	0

(c) Truth table with the device turned off.

Figure 10.42



(a) Logic diagram of the selective inverter.

Invert = 1	
a	***x***
0	1
1	0

(b) Truth table with the inverter turned on.

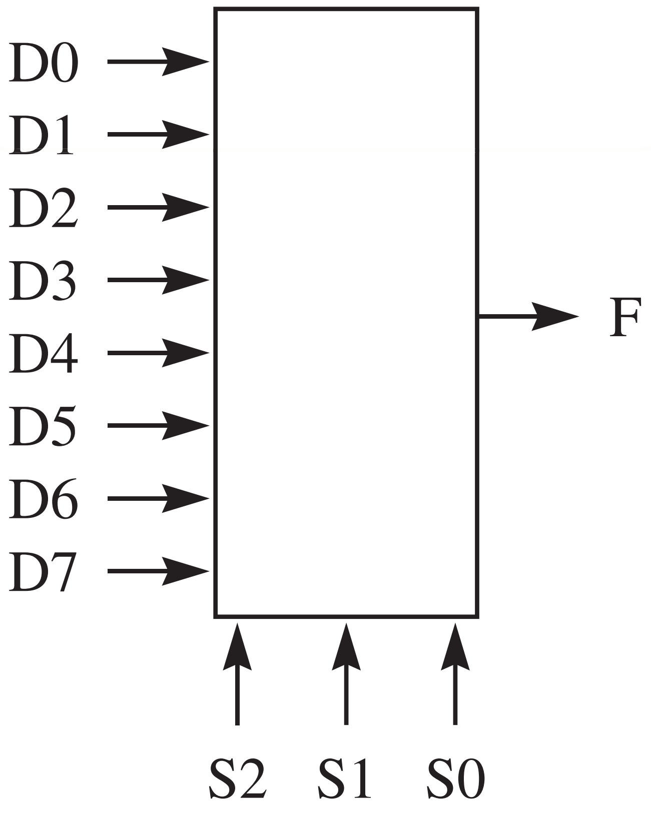
Invert = 0	
a	***x***
0	0
1	1

(c) Truth table with the inverter turned off.

Multiplexer

- A multiplexer selects one of several data inputs to be routed to a single data output
- Control lines determine the particular data input to be passed through

The eight-input multiplexer

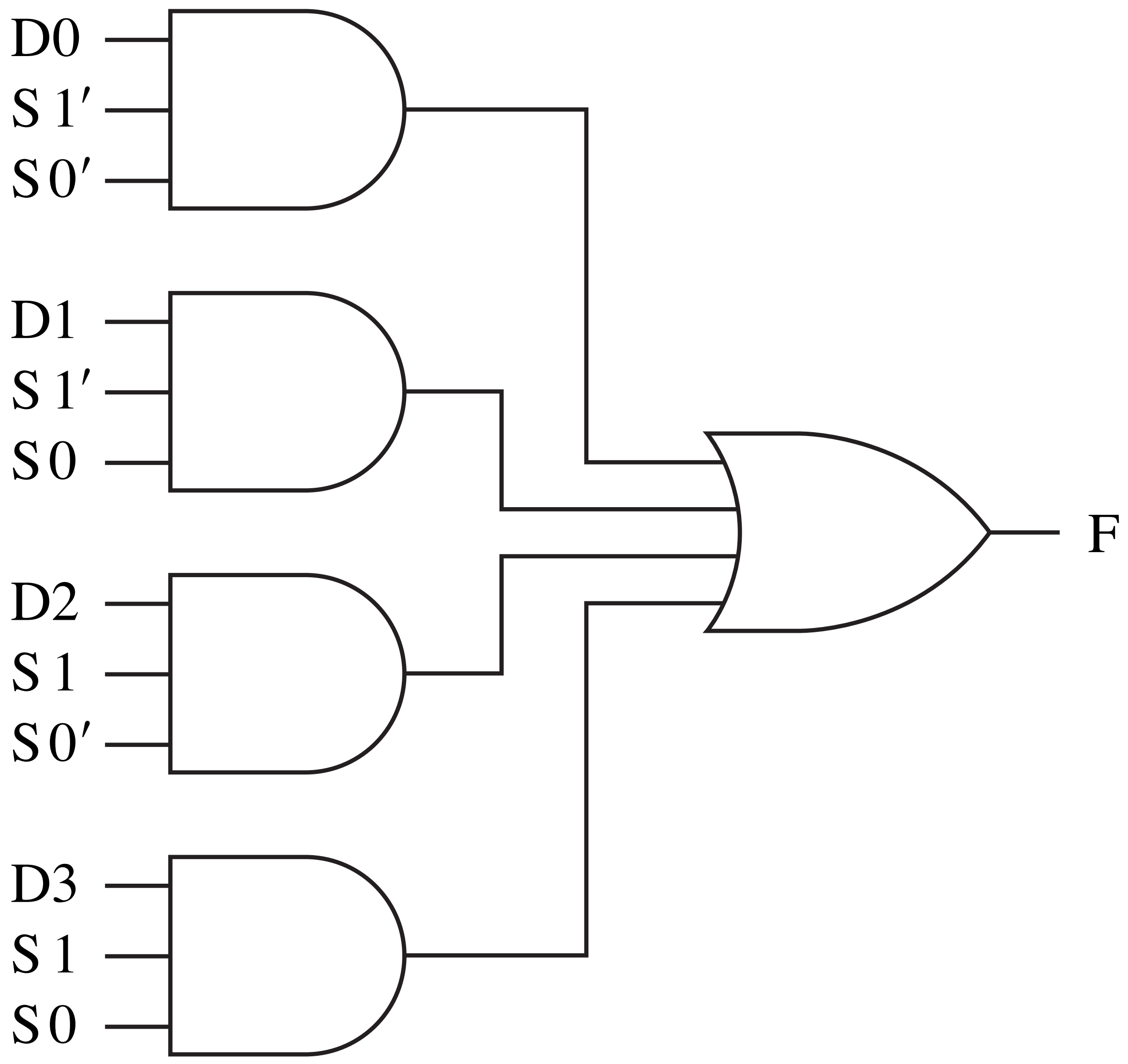


(a) Block diagram.

S2	S1	S0	F
0	0	0	D0
0	0	1	D1
0	1	0	D2
0	1	1	D3
1	0	0	D4
1	0	1	D5
1	1	0	D6
1	1	1	D7

(b) Truth table.

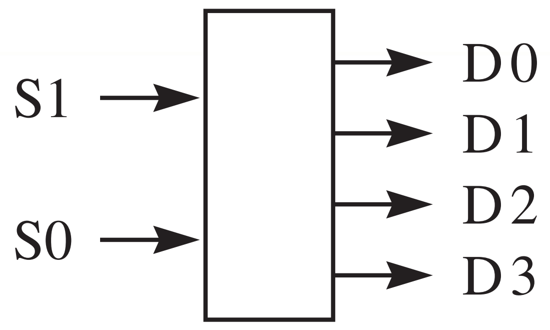
Figure 10.44



Binary decoder

- A decoder takes a binary number as input and sets one of the data output lines to 1 and the rest to 0
- The data line that is set to 1 depends on the value of the binary number that is input

The 2 x 4 binary decoder

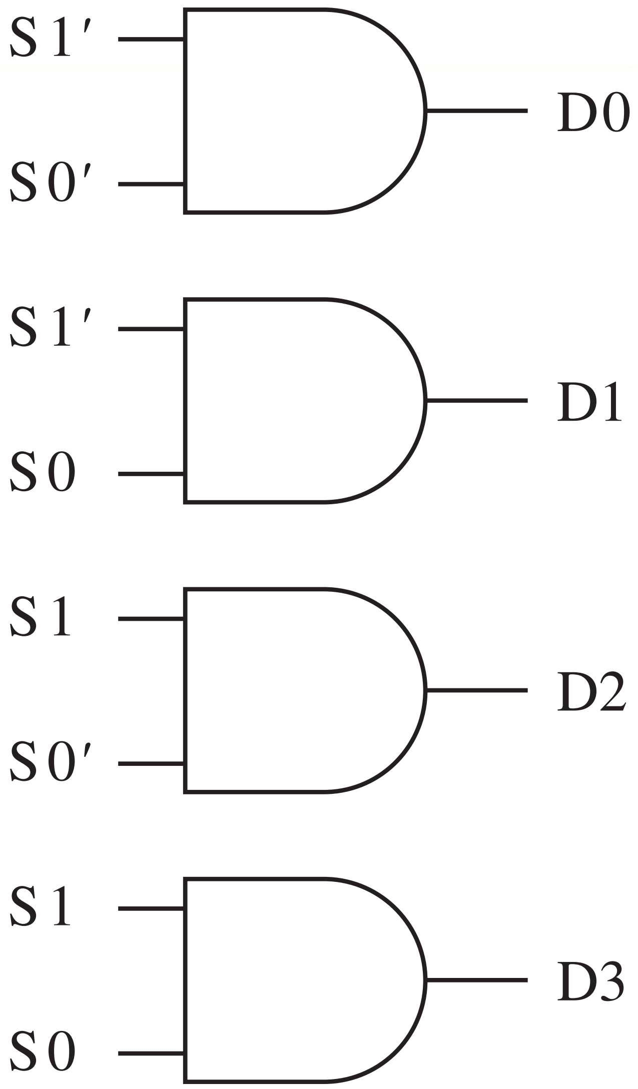


(a) Block diagram.

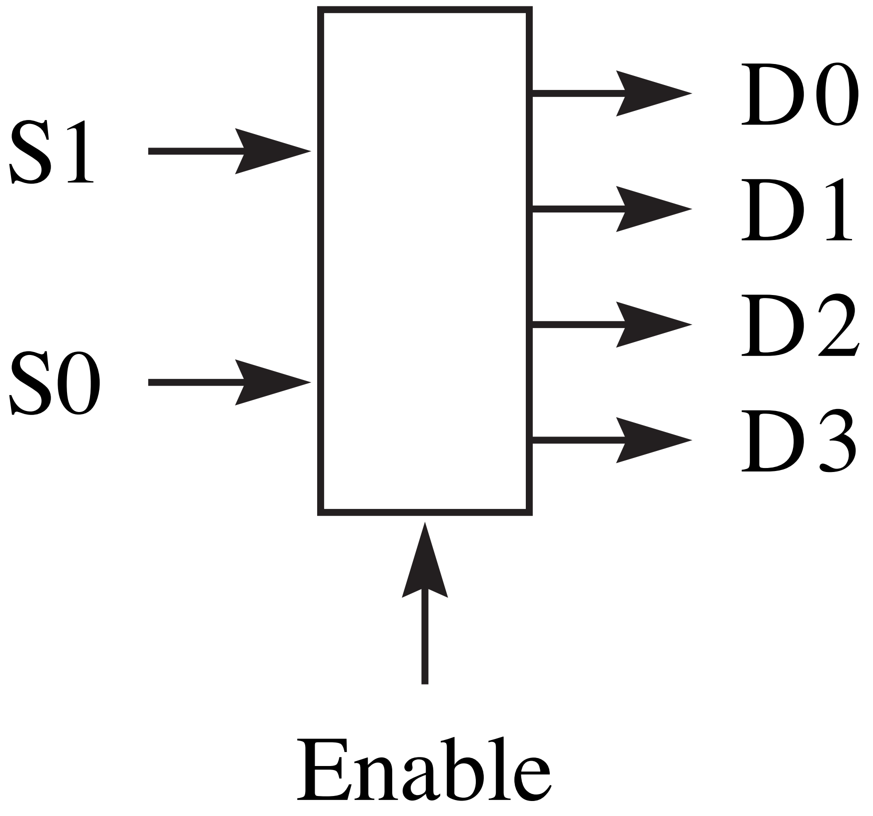
S1	S0	D0	D1	D2	D3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

(b) Truth table.

Figure 10.46



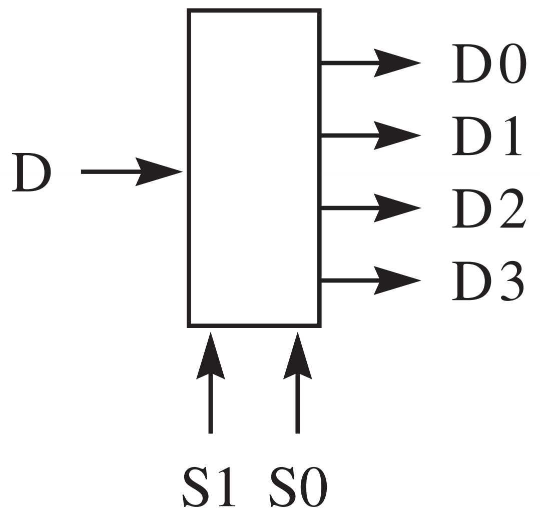
A 2 x 4 binary decoder with enable



Demultiplexer

- A demultiplexer routes a single input value to one of several output lines
- Control lines determine the data output line to which the input gets routed

The four-output demultiplexer



(a) Block diagram.

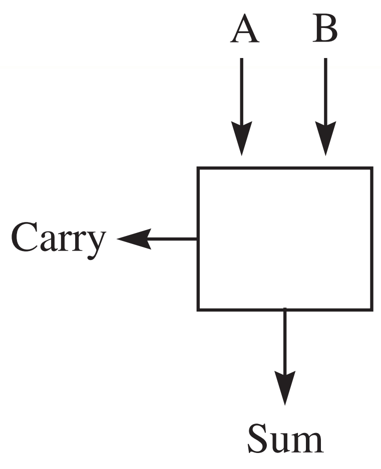
S1	S0	D0	D1	D2	D3
0	0	D	0	0	0
0	1	0	D	0	0
1	0	0	0	D	0
1	1	0	0	0	D

(b) Truth table.

Half adder

- The half adder adds the right-most two bits of a binary number
- Inputs: The two bits
- Outputs: The sum bit and the carry bit

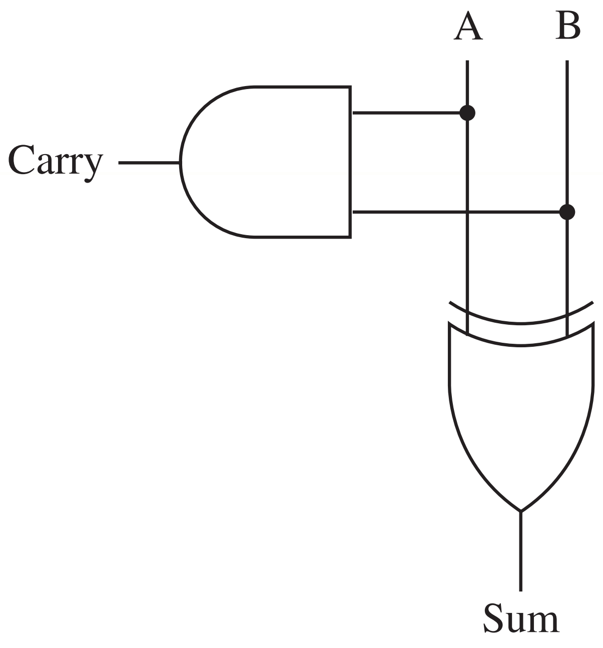
The half adder



(a) Block diagram.

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

(b) Truth table.

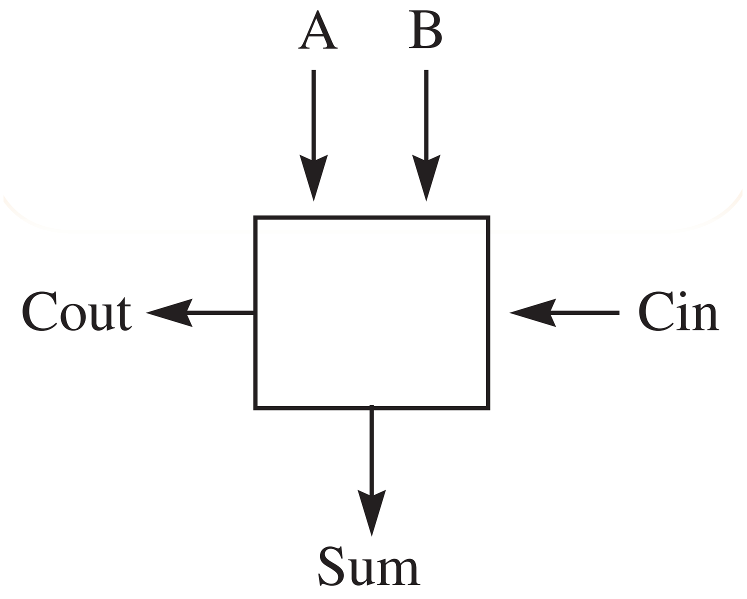


(c) Implementation.

Full adder

- The full adder adds one column of a binary number
- Inputs: The two bits for that column and the carry bit from the previous column
- Outputs: The sum bit and the carry bit for the next column

The full adder

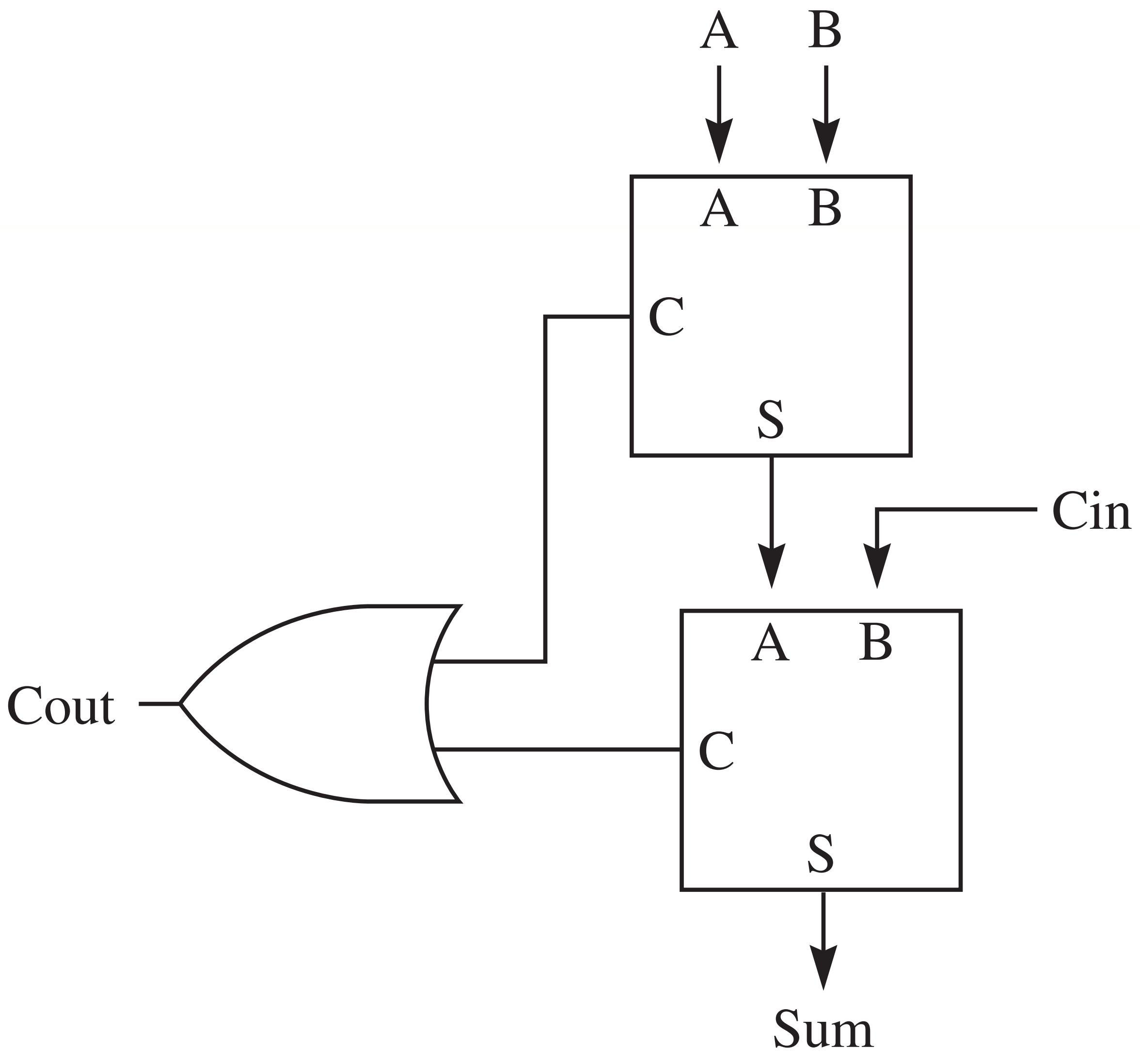


(a) Block diagram.

A	B	Cin	Sum	Cout
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

(b) Truth table.

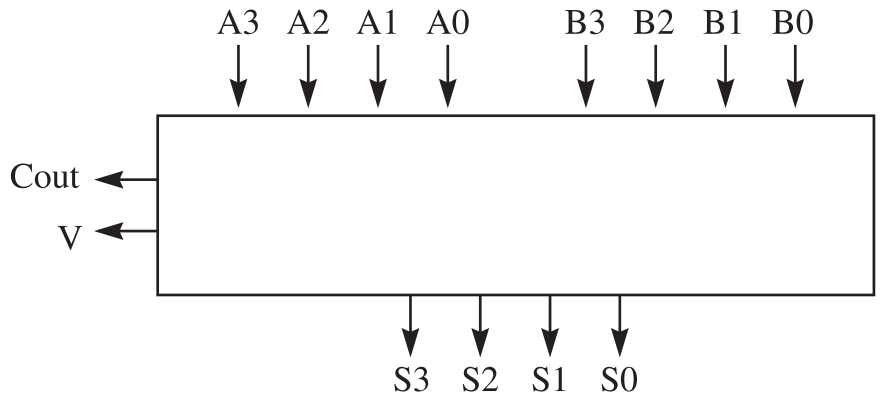
Figure 10.51



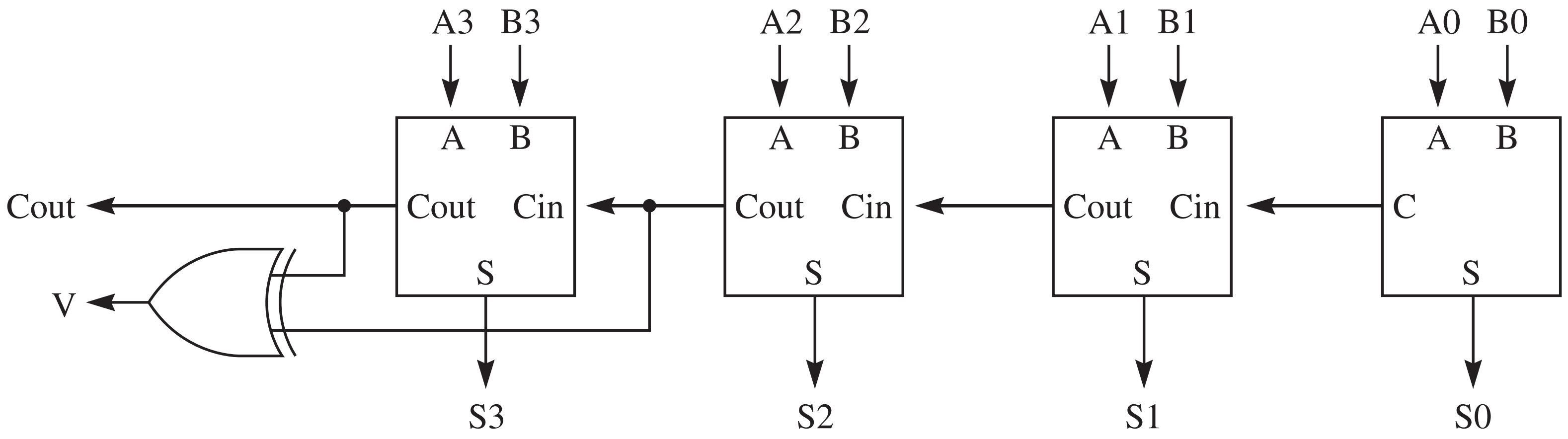
Ripple-carry adder

- The ripple-carry adder adds two *n*-bit binary numbers
- Inputs: The two *n*-bit binary numbers to be added
- Outputs: The *n*-bit sum, the C bit for the carry out, and the V bit for signed integer overflow

Figure 10.52



(a) Block diagram.



(b) Implementation.

Computing the V bit

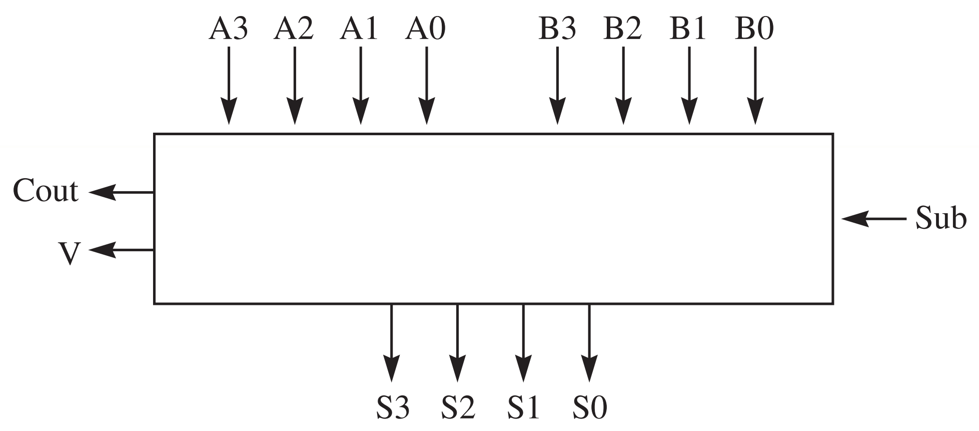
- You can only get an overflow in one of two cases
  - A and B are both positive, and the result is negative
  - A and B are both negative, and the result is positive

Adder/subtracter

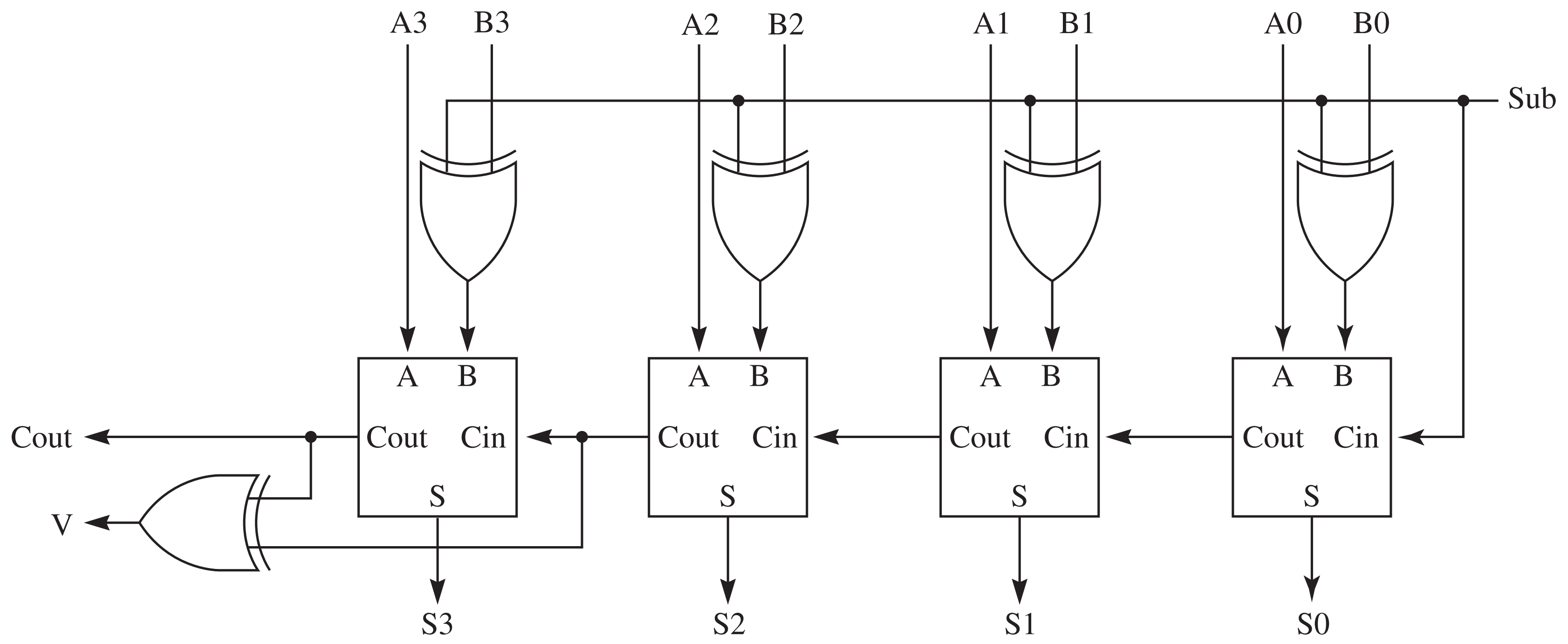
- Based on the relation

  NEG x = 1 + NOT x

- XOR gates act as selective inverters
- A – B = A + (–B)



(a) Block diagram.



(b) Implementation.

Arithmetic Logic Unit (ALU)

- Performs 16 different functions
- Inputs: Two n-bit binary numbers, four control lines that determine which function will be executed, and one carry input line
- Outputs: The n-bit result, the NZVC bits

Figure 10.54

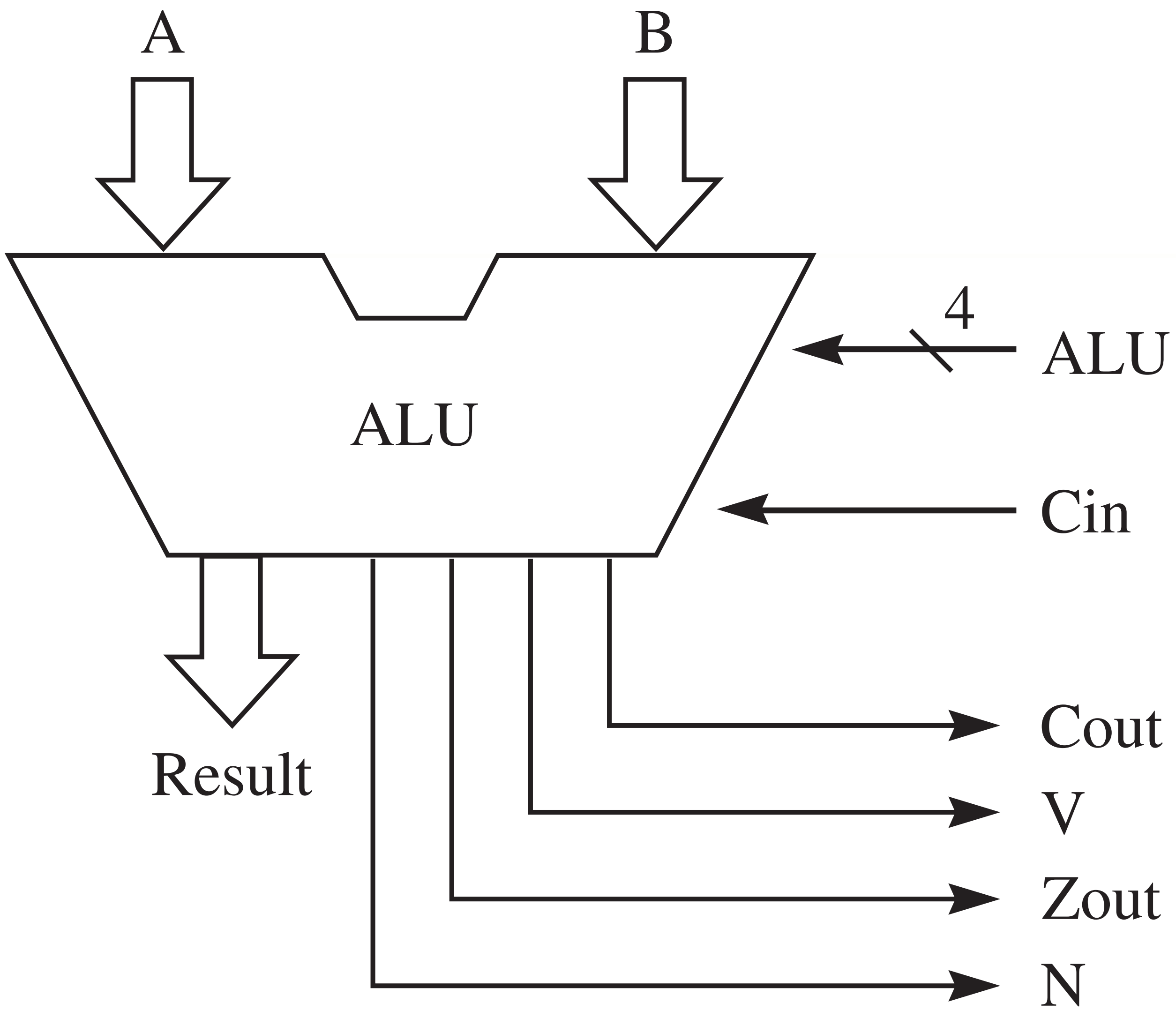
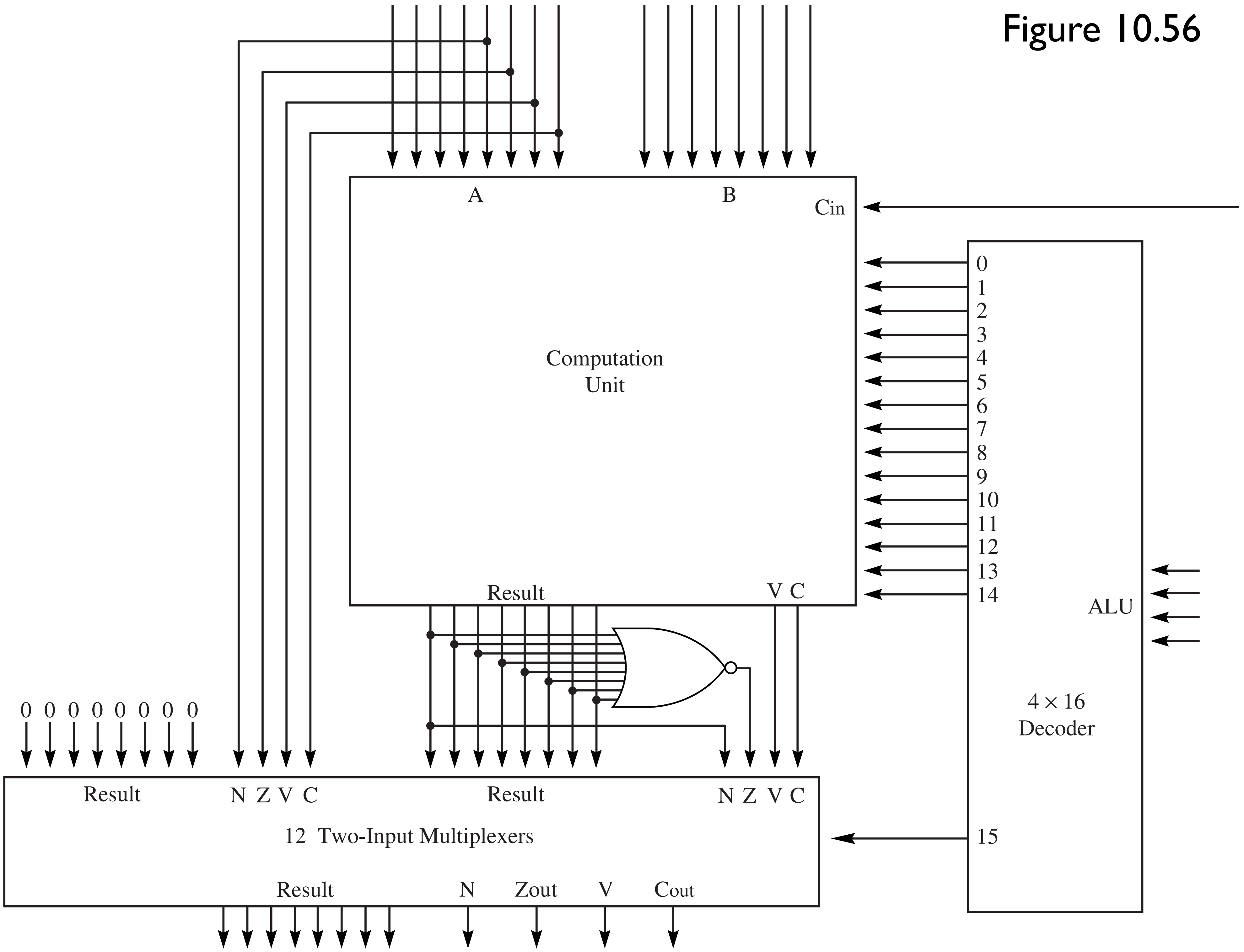


Figure 10.55

ALU Control			Status Bits			
(bin)	**(dec)**	**Result**	**N**	**Zout**	**V**	**Cout**
0000	0	A	N	Z	0	0
0001	1	A plus B	N	Z	V	C
0010	2	A plus B plus Cin	N	Z	V	C
0011	3	A plus $\overline{B}$ plus 1	N	Z	V	C
0100	4	A plus $\overline{B}$ plus Cin	N	Z	V	C
0101	5	$A \cdot B$	N	Z	0	0
0110	6	$\overline{A \cdot B}$	N	Z	0	0
0111	7	$A + B$	N	Z	0	0
1000	8	$\overline{A + B}$	N	Z	0	0
1001	9	$A \oplus B$	N	Z	0	0
1010	10	$\overline{A}$	N	Z	0	0
1011	11	ASL A	N	Z	V	C
1100	12	ROL A	N	Z	V	C
1101	13	ASR A	N	Z	0	C
1110	14	ROR A	N	Z	0	C
1111	15	0	A<4>	A<5>	A<6>	A<7>

Figure 10.56

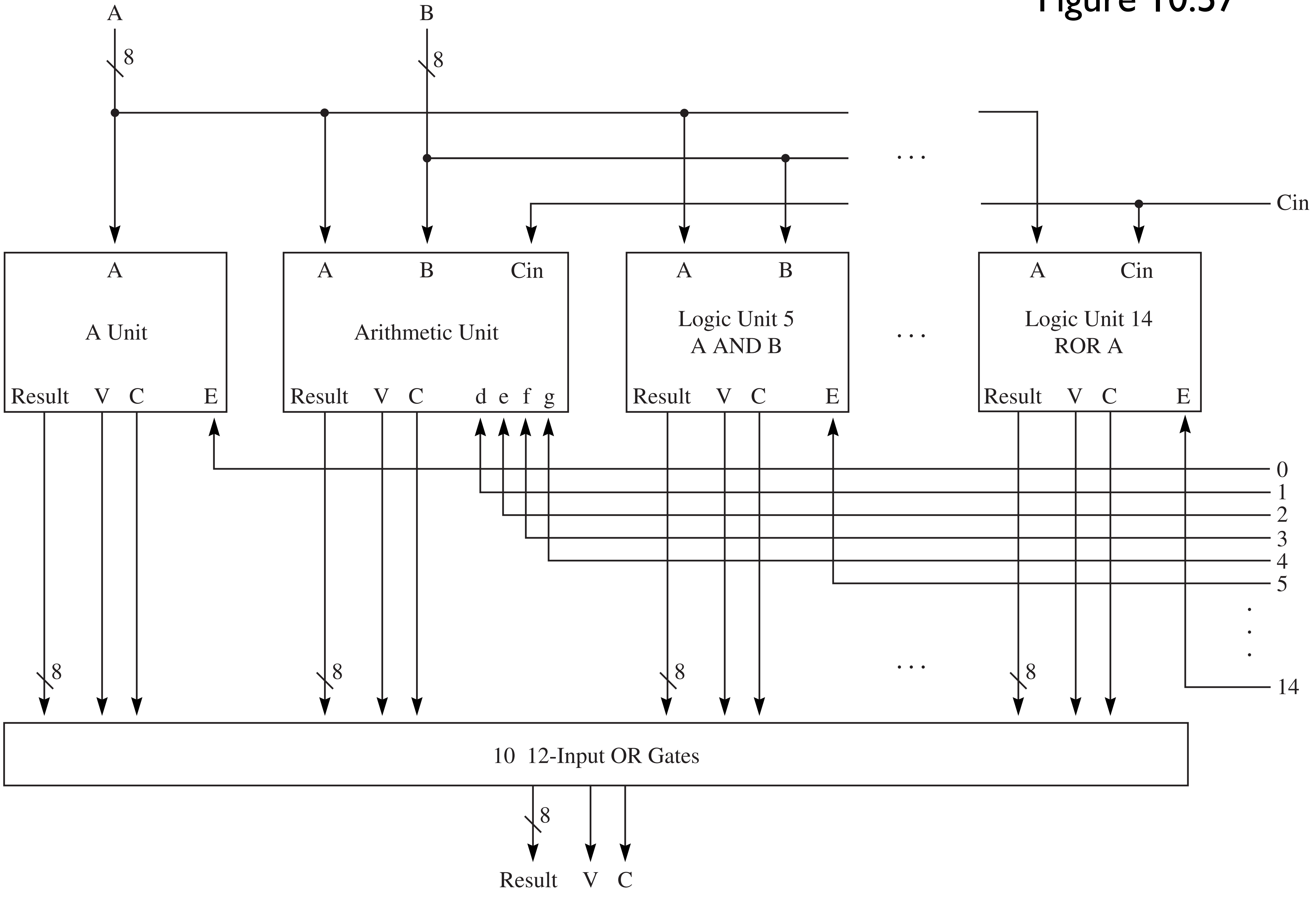


A
B
Cin
Computation Unit
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
Result
V C
ALU
4 × 16 Decoder
0 0 0 0 0 0 0 0
Result
N Z V C
Result
N Z V C
12 Two-Input Multiplexers
15
Result
N
Zout
V
Cout

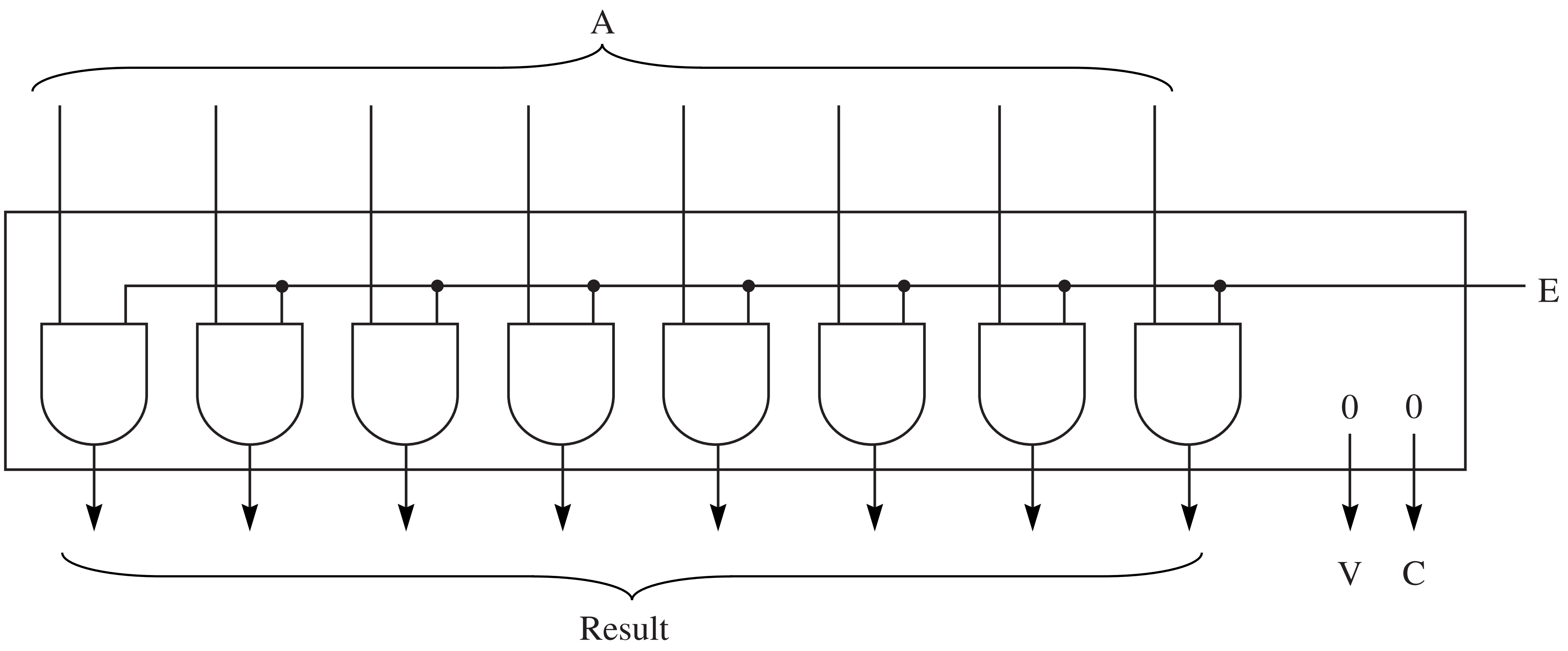
The multiplexer of Figure 10.56

- If line 15 is 1, Result and NZVC from the *left* are routed to the output
- If line 15 is 0, Result and NZVC from the *right* are routed to the output

Figure 10.57


A
B
8
8
. . .
Cin
A
A Unit
Result V C E
A B Cin
Arithmetic Unit
Result V C d e f g
A B
Logic Unit 5
A AND B
Result V C E
. . .
A Cin
Logic Unit 14
ROR A
Result V C E
0
1
2
3
4
5
14
8
8
8
8
10 12-Input OR Gates
8
Result V C

Implementation of the A unit of Figure 10.57



Implementation of the arithmetic unit of Figure 10.57

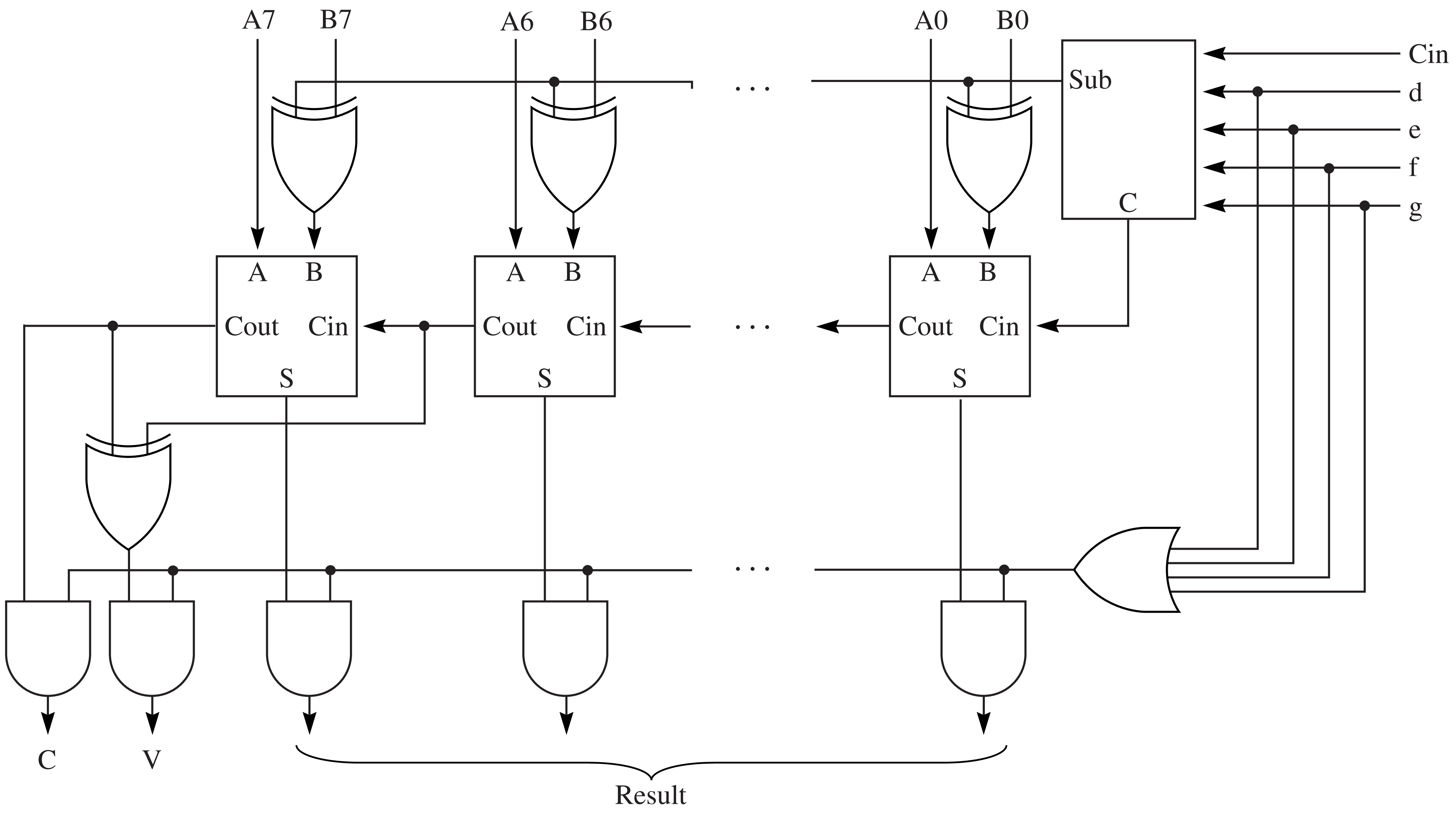
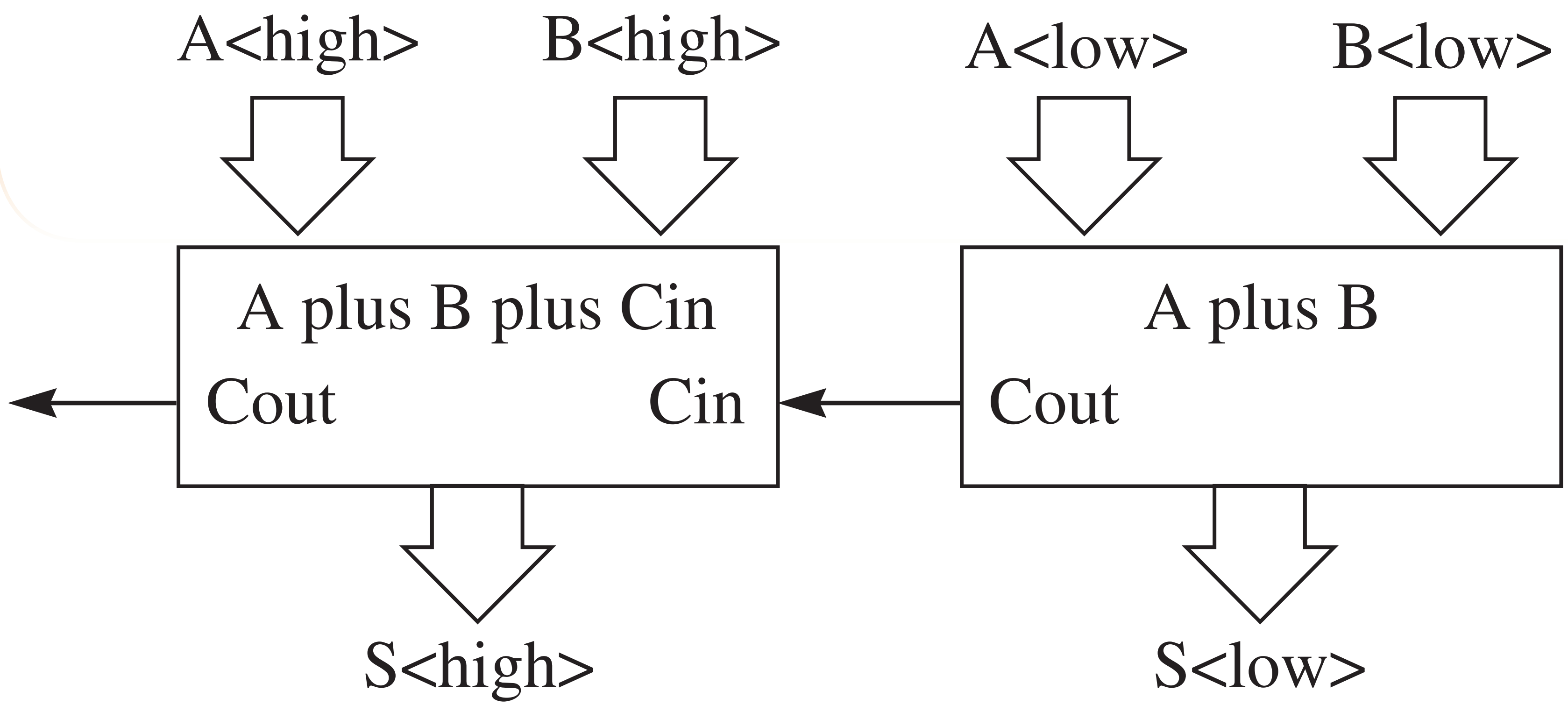
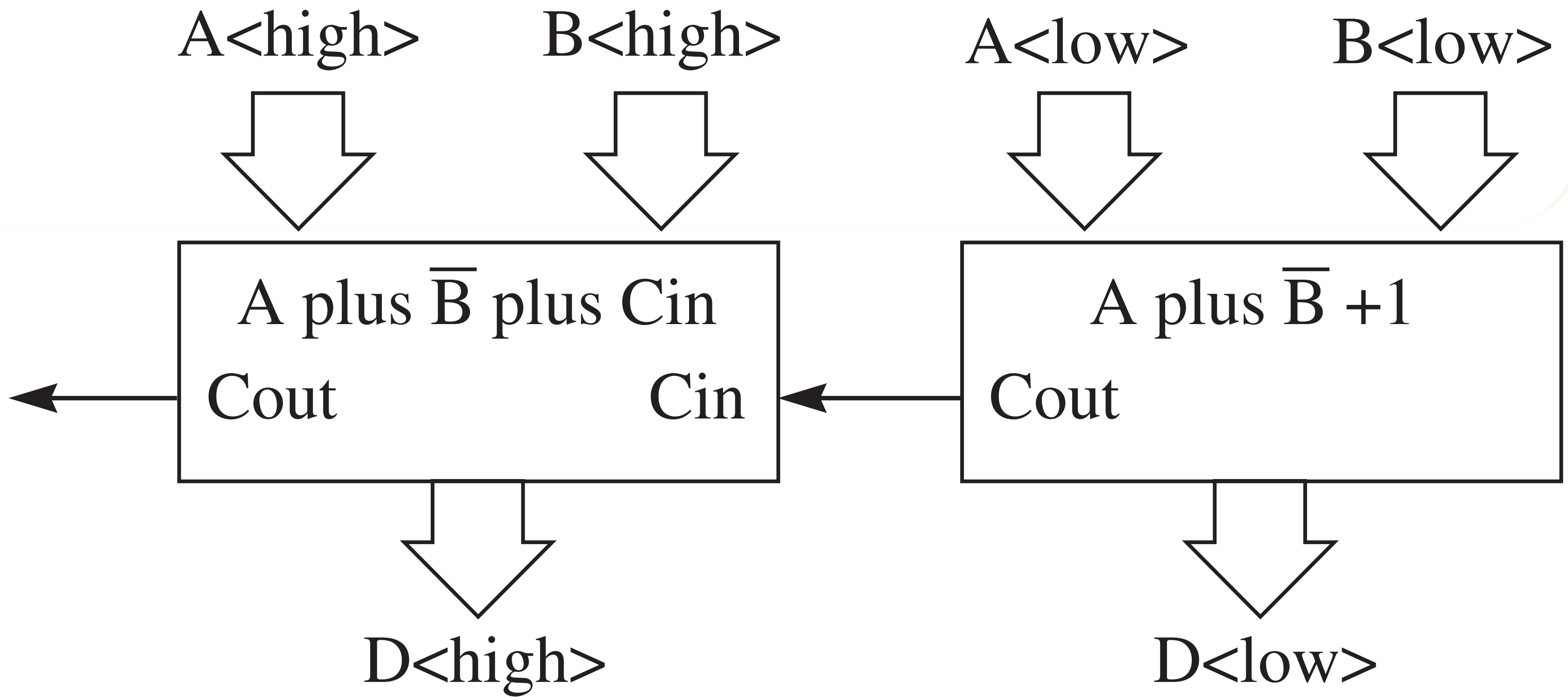


Figure 10.60



(a) 16-bit addition.

Figure 10.60 (continued)

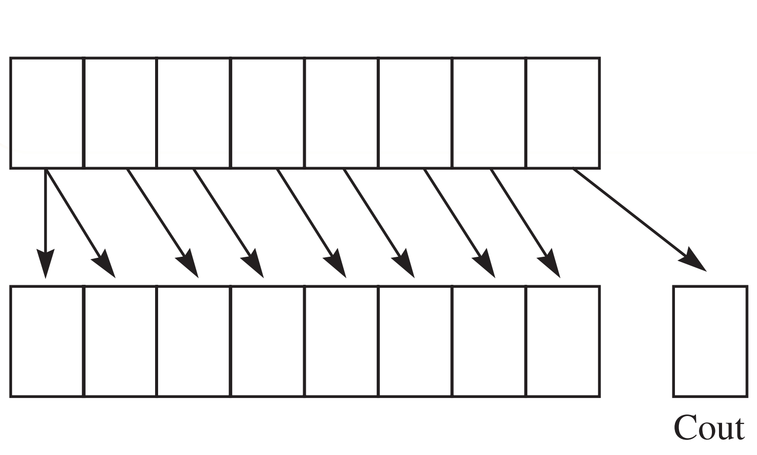


(b) 16-bit subtraction.

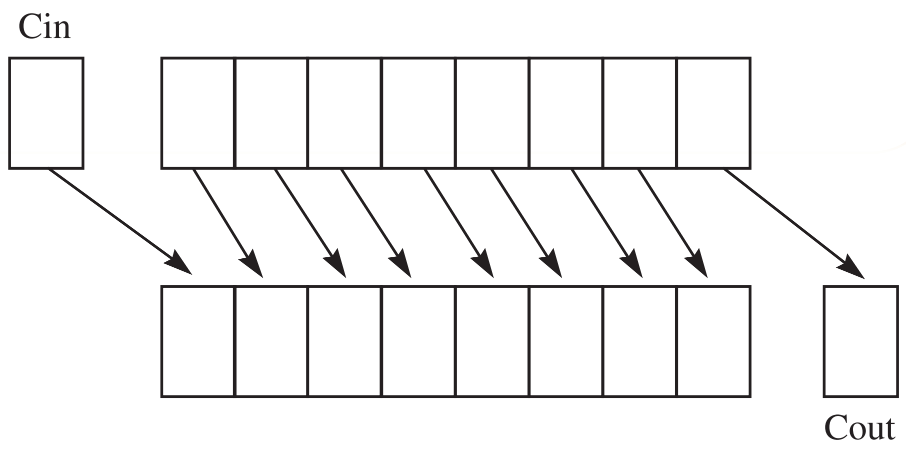
Figure 10.61

Function	d	e	f	g	Sub	C
A plus B	1	0	0	0	0	0
A plus B plus Cin	0	1	0	0	0	Cin
A plus $\overline{B}$ plus 1	0	0	1	0	1	1
A plus $\overline{B}$ plus Cin	0	0	0	1	1	Cin

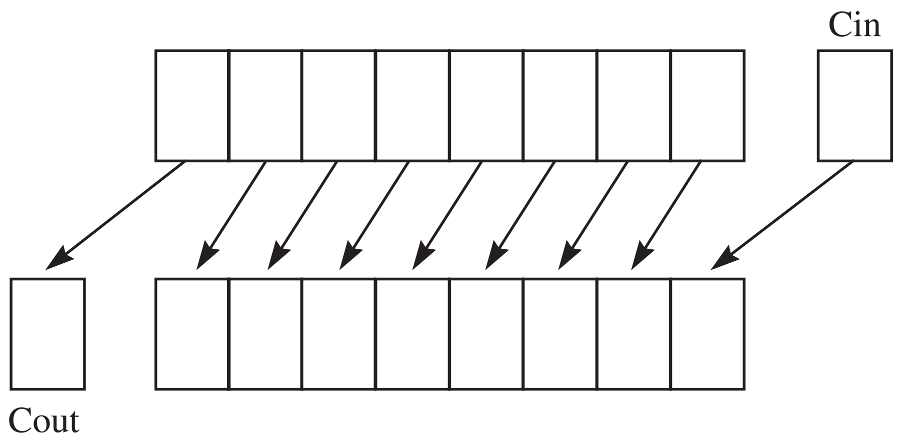
Figure 10.62



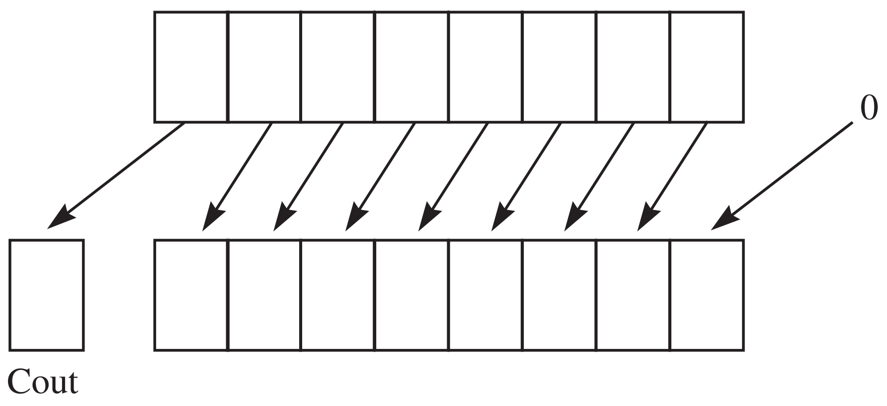
(a) Arithmetic shift right (ASR).



(b) Rotate right (ROR).

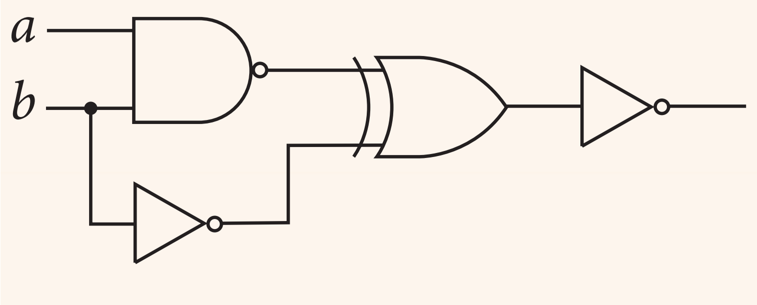


(c) Rotate left (ROL).

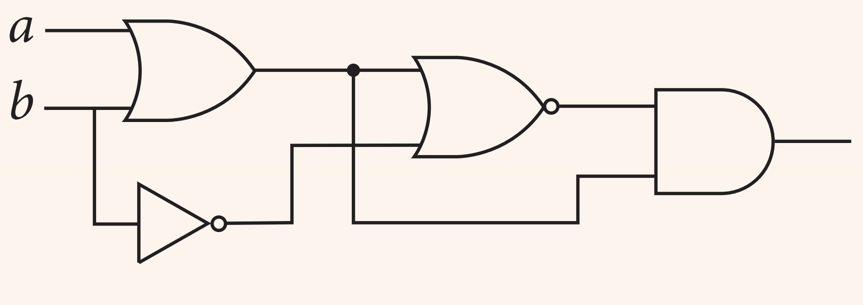


(d) Arithmetic shift left (ASL).

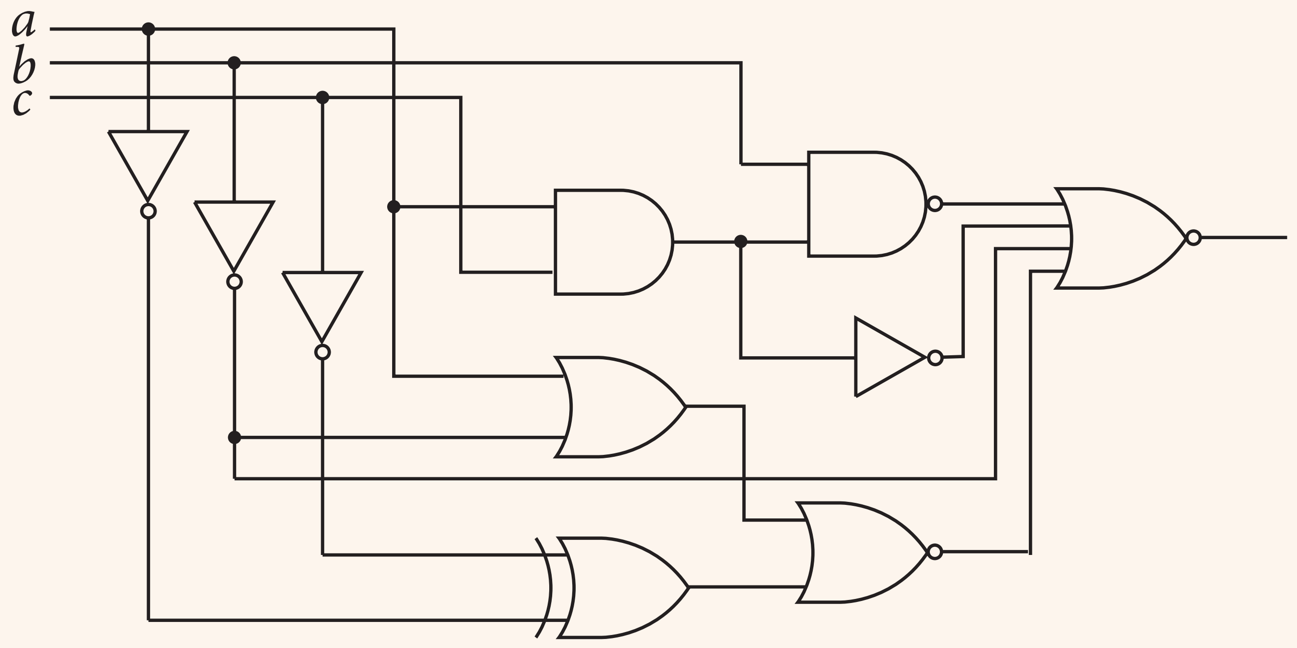
For Exercise 18



(a)

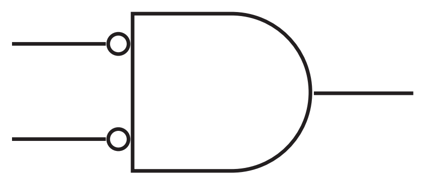


(b)

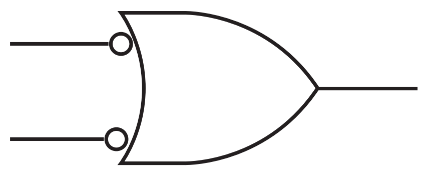


(c)

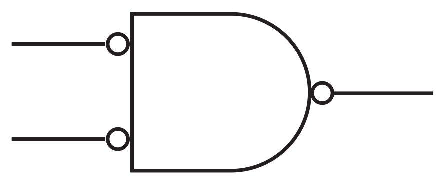
For Exercise 26



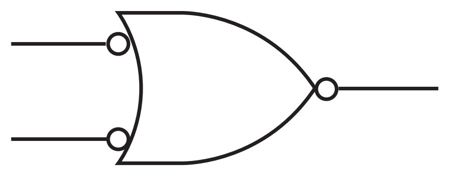
(a)



(b)

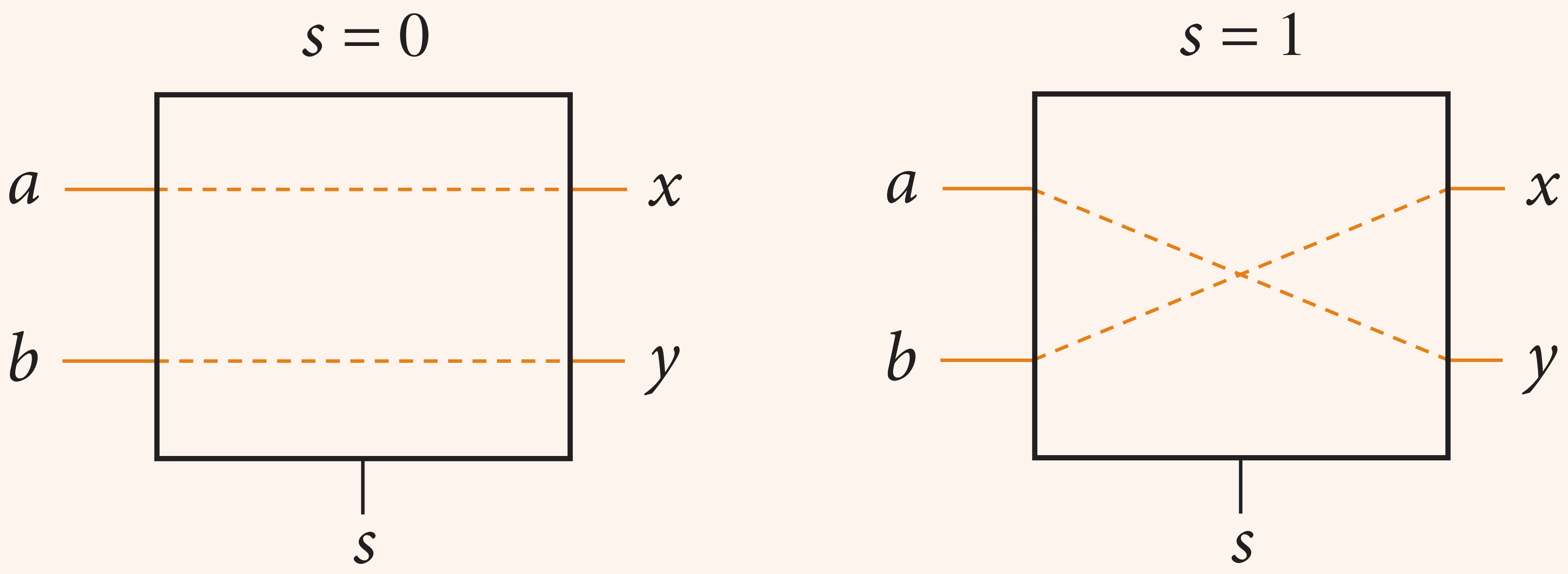


(c)

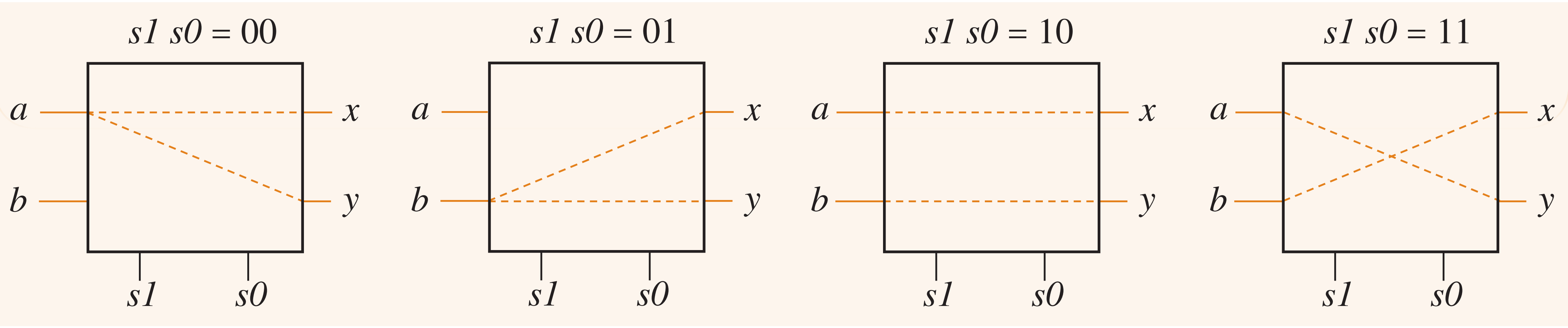


(d)

For Exercise 51



For Exercise 52


s1 s0 = 00
s1 s0 = 01
s1 s0 = 10
s1 s0 = 11
a
b
x
y
s1
s0