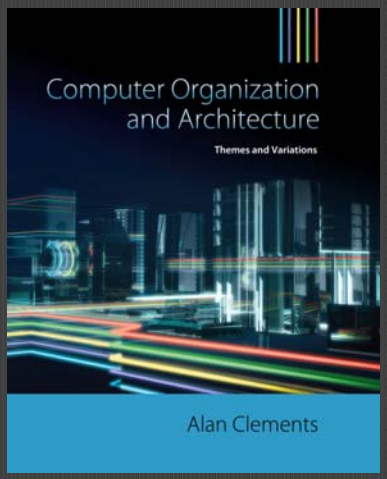
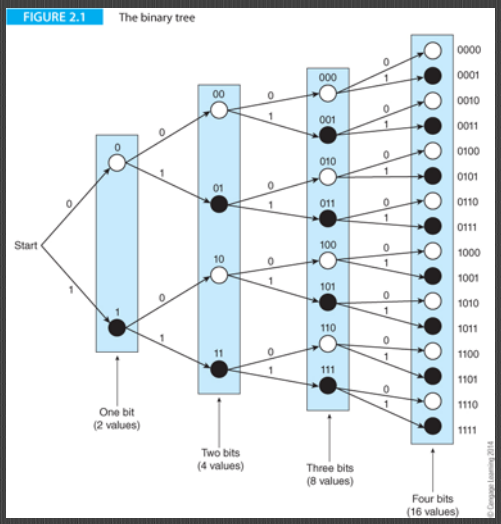
CHAPTER 2

Computer Arithmetic and Digital Logic



1



2

TABLE 2.1 The ASCII Code

		0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
0	0000	NULL	DCL	SP	0	@	P	`	p
1	0001	SOH	DC1	!	1	A	Q	a	q
2	0010	STX	DC2	"	2	B	R	b	r
3	0011	ETX	DC3	#	3	C	S	c	s
4	0100	EOT	DC4	$	4	D	T	d	t
5	0101	ENQ	NAK	%	5	E	U	e	u
6	0110	ACK	SYN	&	6	F	V	f	v
7	0111	BEL	ETB	'	7	G	W	g	w
8	1000	BS	CAN	(	8	H	X	h	x
9	1001	HT	EM	)	9	I	Y	i	y
A	1010	LF	SUB	*	:	J	Z	j	z
B	1011	VT	ESC	+	;	K	[	k	]
C	1100	FF	FS	,	<	L	\	l	|
D	1101	CR	GS	–	=	M	]	m	}
E	1110	SO	RS	.	>	N	^	n	~
F	1111	SI	US	/	?	O	_	o	DEL

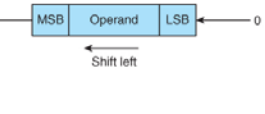
3

TABLE 2.2 The Digits Employed by Four Number Bases

System	Base	Set of Digits
Decimal	$b = 10$	$a = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
Binary	$b = 2$	$a = \{0, 1\}$
Octal	$b = 8$	$a = \{0, 1, 2, 3, 4, 5, 6, 7\}$
Hexadecimal	$b = 16$	$a = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F,\}$

4

FIGURE 2.2 Arithmetic shift operations



(a) Arithmetic shift left:

A zero enters the least-significant bit position and the most-significant bit is copied into the carry flag. For example: 11000101 becomes 10001010 after shifting one place left.

(b) Arithmetic shift right:

A copy of the most-significant bit enters the most-significant bit position. All other bits are shifted one place right. The least-significant bit is copied into the carry flag.

For example; 00100101 becomes 00010010 after shifting one place right, and 11100101 becomes 11110010 after shifting one place right.

5

FIGURE 2.3 An algorithm for multiplication

Step a.	Set a counter to *n*.
Step b.	Clear the 2*n*-bit partial product register.
Step c.	Examine the rightmost bit of the multiplier (initially the least-significant bit). This bit is underlined in Table 2.3. If it is one, add the multiplicand to the *n* most-significant bits of the partial product.
Step d.	Shift the partial product one place to the right.
Step e.	Shift the multiplier one place to the right (the rightmost bit is, of course, lost).
Step f.	Decrement the counter and repeat from step c until the count is 0 after *n* cycles. The product is in the partial product register.

6

TABLE 2.3 Mechanizing Unsigned Multiplication from Figure 2.3

		Multiplier = 1101_2	Multiplicand = 1010_2	
Cycle	Step	Counter	Multiplier	Partial Product
	a and b	4	1101	00000000
1	c	4	1101	10100000
1	d and e	4	0110	01010000
1	f	3	0110	01010000
2	c	3	0110	01010000
2	d and e	3	0011	00101000
2	f	2	0011	00101000
3	c	2	0011	11001000
3	d and e	2	0001	01100100
3	f	1	0001	01100100
4	c	1	0001	10000010
4	d and e	1	0000	10000010

TABLE 2.4 Demonstration of Booth's Algorithm

Step	Multiplier Bits	Partial Product
		0000000000
Subtract multiplicand	10011**0**	1000100000
Shift partial product right		1100010000
Do nothing	1001**1**	1100010000
Shift partial product right		1110001000
Add multiplicand	100**1**1	10101101000
Shift partial product right		0010110100
Do nothing	1**0**011	0010110100
Shift partial product right		0001011010
Subtract multiplicand	**1**0011	1001111010
Shift partial product right		1100111101

FIGURE 2.4 The flowchart for restoring division

TABLE 2.5 Example of Restoring Division for 1001|0100111

Step	Description	Partial Dividend	Divisor	Quotient
		01100111	00001001	00000000
1	Align	01100111	01001000	00000000
2	Subtract divisor from partial dividend	00011111	01001000	00000000
4	Result positive—shift in 1 quotient	00011111	01001000	00000001
5	Test for end			
6	Shift divisor one place right	00011111	00100100	00000001
2	Subtract divisor from partial dividend	−00000101	00100100	00000001
3	Restore divisor, shift in 0 in quotient	00011111	00100100	00000010
5	Test for end			
6	Shift divisor one place right	00011111	00010010	00000010
2	Subtract divisor from partial dividend	00001101	00010010	00000010
4	Result positive—shift in 1 quotient	00001101	00010010	00000101
5	Test for end			
6	Shift divisor one place right	00001101	00001001	00000101
2	Subtract divisor from partial dividend	00000100	00001001	00000101
4	Result positive—shift in 1 quotient	00000100	00001001	00001011
5	Test for end			

FIGURE 2.5 Flowchart for non-restoring division

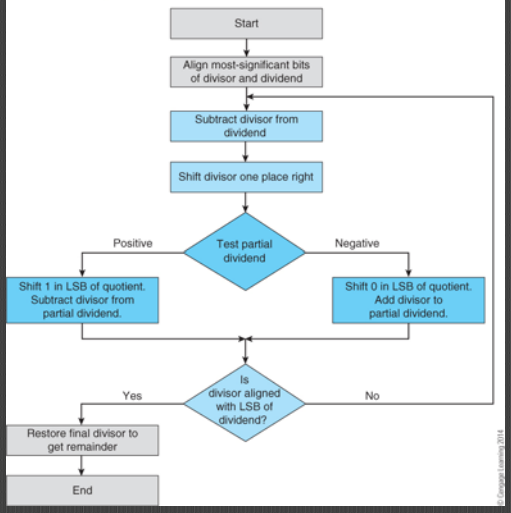


TABLE 2.6 An Example of Non-Restoring Division for 1001|01100111

Step	Description	Partial Dividend	Divisor	Quotient
		01100111	00001001	00000000
1	Align divisor	01100111	01001000	00000000
2	Subtract divisor from partial dividend	00011111	01001000	00000000
3	Shift divisor right	00011111	00100100	00000000
4	Test partial dividend—enter 1 in quotient and subtract divisor from partial dividend	−00000101	00100100	00000001
6	Test for end of process	−00000101	00100100	00000001
3	Shift divisor right	−00000101	00010010	00000001
5	Test partial dividend—enter 0 in quotient and add divisor to partial dividend	00001101	00010010	00000010
6	Test for end of process	00001101	00010010	00000010
3	Shift divisor right	00001101	00001001	00000010
4	Test partial dividend — enter 1 in quotient and subtract divisor from partial dividend	00000100	00001001	00000101
6	Test for end of process	00000100	00001001	00000101
3	Shift divisor right	00000100	00000100.1	00000101
4	Test partial dividend—enter 1 in quotient and subtract divisor from partial dividend	−00000000.1	00000100.1	00001011
6	Test for end of process	−00000000.1	00000100.1	00001011
7	Restore last divisor	00000100	00000100.1	00001011

FIGURE 2.6 Representing the value zero in floating-point arithmetic

FIGURE 2.7 Structure of a 32-bit IEEE floating-point number

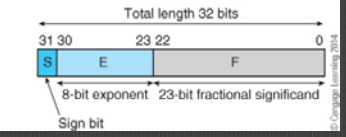


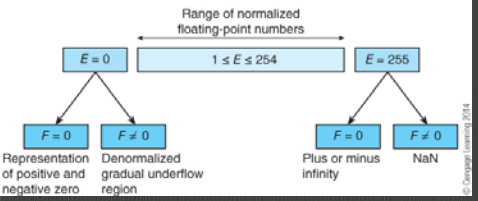
TABLE 2.7 IEEE Floating-Point Formats

	Single Precision	Double Precision (Single Extended)	Quad Precision (Double Extended)
Field width in bits			
S = sign	1	1	1
E = exponent	8	11	15
L = leading bit	1 (not stored)	1 (not stored)	1 (stored)
F = fraction	23	52	111
Total width	32	64	128
Exponent			
Maximum E	255	2047	32,767
Minimum E	0	0	0
Bias	127	1023	16,383
E_{max}	127	1023	16,383
E_{min}	−126	−1022	−16,382

S = sign bit (0 for a negative number, 1 for a positive number)
L = leading bit (always 1 in a normalized, non-zero significand)
F = fractional part of the significand
The range of exponents is from the minimum $E + 1$ to the maximum $E - 1$
The number is represented by $-1^S \times 2^{E - \text{exponent}} \times L.F$
Zero is represented by the minimum exponent, $L = 0$, and $F = 0$, for all three formats.
The maximum exponent, $E_{max} + 1$ represents signed infinity for all three formats.

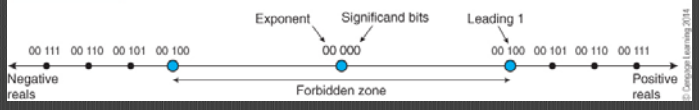
15

FIGURE 2.8 IEEE floating-point number space for a single-precision number



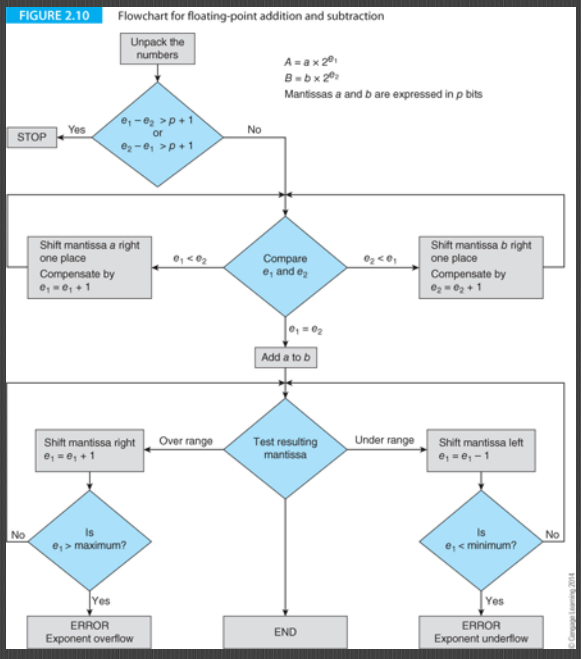
16

FIGURE 2.9 Floating-point numbers near zero

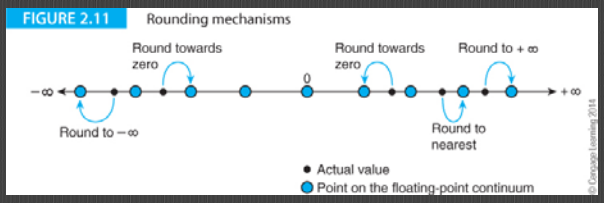


17

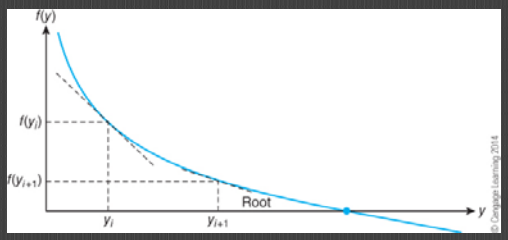
FIGURE 2.10 Flowchart for floating-point addition and subtraction



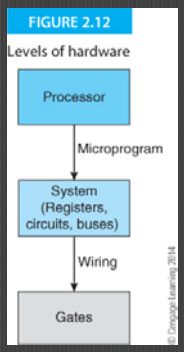
18



19



20



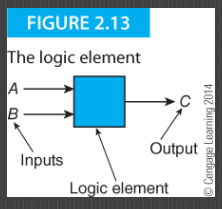
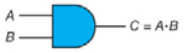


FIGURE 2.14 The symbol for an AND gate



(a) Two-input AND gate (b) Three-input AND gate

TABLE 2.8 Truth Table for the AND Gate

B	A	C = A·B
0	0	0
0	1	0
1	0	0
1	1	1

(a) Two-input AND gate

C	B	A	D = A·B·C
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

(b) Three-input AND gate

TABLE 2.9 Truth Table for the OR Gate

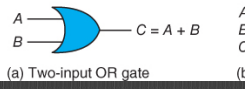
B	A	C = A + B
0	0	0
0	1	1
1	0	1
1	1	1

(a) Two-input OR gate

C	B	A	D = A + B + C
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

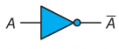
(b) Three-input OR gate

FIGURE 2.15 The symbol for an OR gate



(a) Two-input OR gate (b) Three-input OR gate

FIGURE 2.16 The symbol and truth table for an inverter



(a) Symbol for inverter

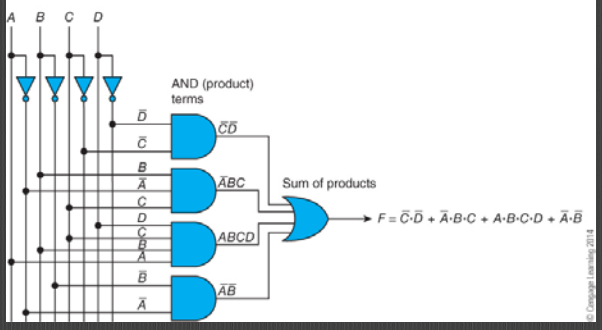
A	$\bar{A}$
0	1
1	0

(b) Truth table of inverter

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

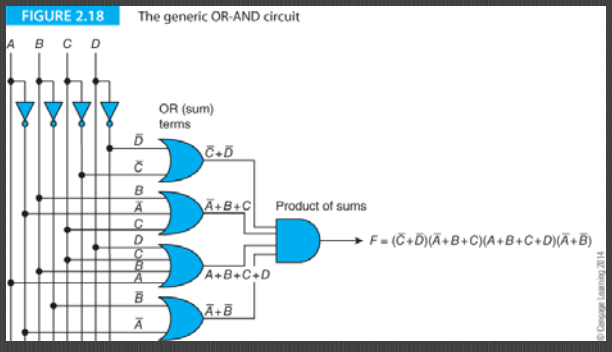
AND		OR	
Constant	Variable	Constant	Variable
$0 \cdot 0 = 0$	$A \cdot 0 = 0$	$0 + 0 = 0$	$A + 0 = A$
$0 \cdot 1 = 0$	$A \cdot 1 = A$	$0 + 1 = 1$	$A + 1 = 1$
$1 \cdot 0 = 0$	$A \cdot \bar{A} = 0$	$1 + 0 = 1$	$A + \bar{A} = 1$
$1 \cdot 1 = 1$	$A \cdot A = A$	$1 + 1 = 1$	$A + A = A$

FIGURE 2.17 The generic AND-OR circuit



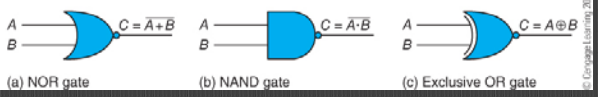
29

FIGURE 2.18 The generic OR-AND circuit



30

FIGURE 2.19 Three derived gates



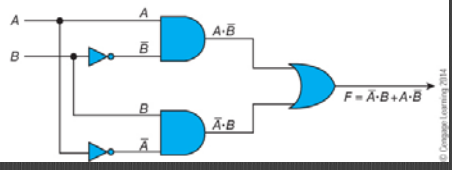
(a) NOR gate (b) NAND gate (c) Exclusive OR gate

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

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

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



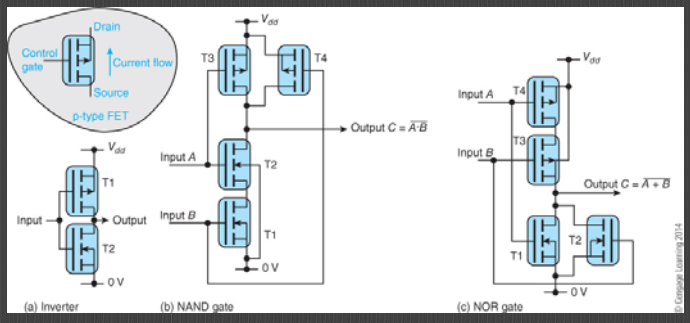


FIGURE 2.21 A circuit with four gates

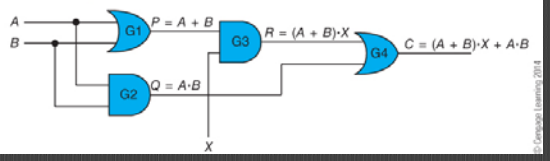


TABLE 2.12 Truth Table for Figure 2.21

Inputs			Intermediate Values			Output
X	A	B	P = A + B	Q = A·B	R = (A + B)·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

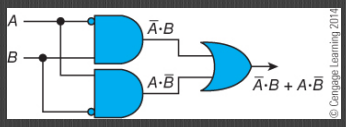


FIGURE 2.22 The two-bit adder (the half adder)

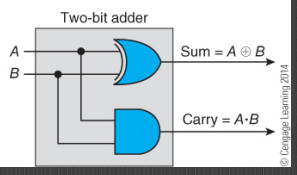
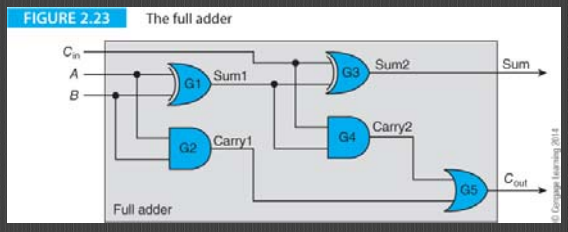


TABLE 2.13
Truth Table of a Half Adder

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

FIGURE 2.23 The full adder



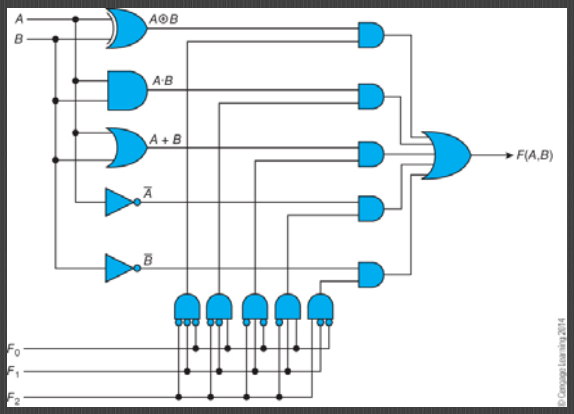


TABLE 2.14 Truth Table for the Circuit of Figure 2.24

Inputs			Full Adder		Full Adder Circuit				
C_{in}	A	B	C_{out}	Sum	Carry1	Sum1	Sum2	Carry2	C_{out}
0	0	0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	1	0	0
0	1	0	0	1	0	1	1	0	0
0	1	1	1	0	1	0	0	0	1
1	0	0	0	1	0	0	1	0	0
1	0	1	1	0	0	1	0	1	1
1	1	0	1	0	0	1	0	1	1
1	1	1	1	1	1	0	1	0	1

FIGURE 2.24 Alternative full adder circuit

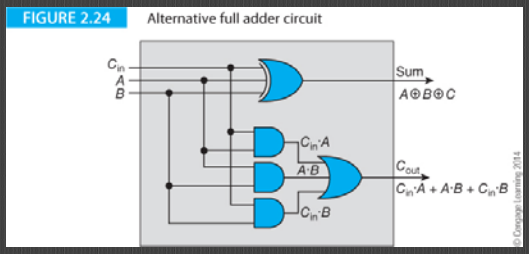


FIGURE 2.25 The parallel adder

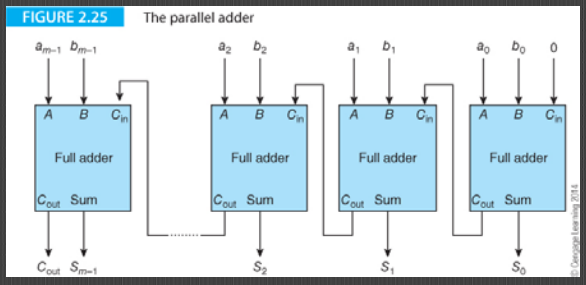


FIGURE 2.26 The programmable inverter

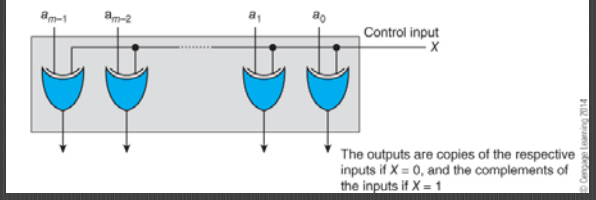


FIGURE 2.27 The adder/subtractor

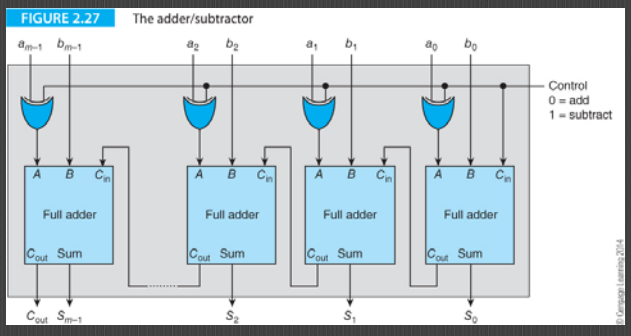


FIGURE 2.28 Application of a decoder

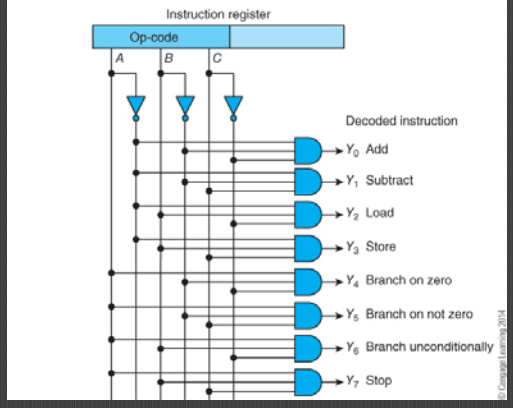
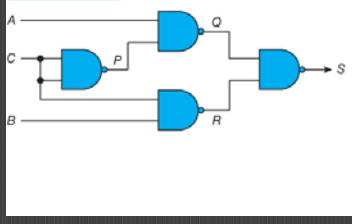


TABLE 2.15 The Decoder

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 2.29 The two-input multiplexer and its truth table



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

FIGURE 2.30 Alternative representation of the two-input multiplexer

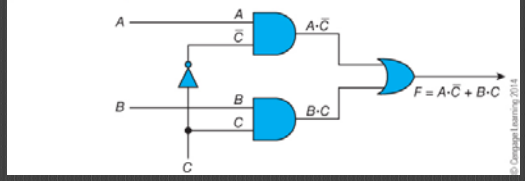


FIGURE 2.31 The majority logic circuit

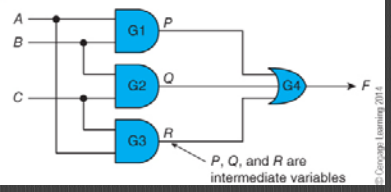


TABLE 2.16 Truth Table for the Majority Logic Circuit of Figure 2.31

Inputs			Intermediate Values			Output
A	B	C	P = A·B	Q = B·C	R = A·C	F = P + Q + R
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	1	0	1
1	0	0	0	0	0	0
1	0	1	0	0	1	1
1	1	0	1	0	0	1
1	1	1	1	1	1	1

FIGURE 2.32 The 3-of-4 detector

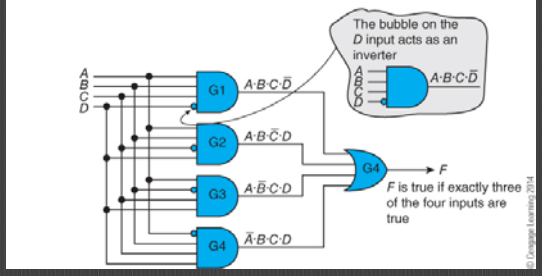
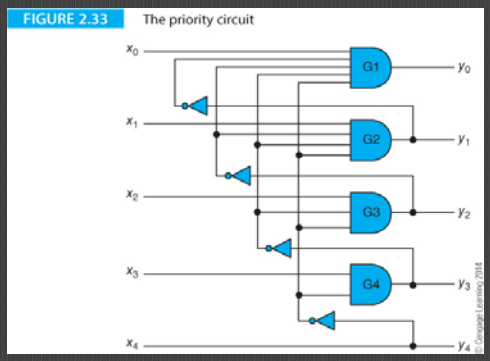


FIGURE 2.33 The priority circuit



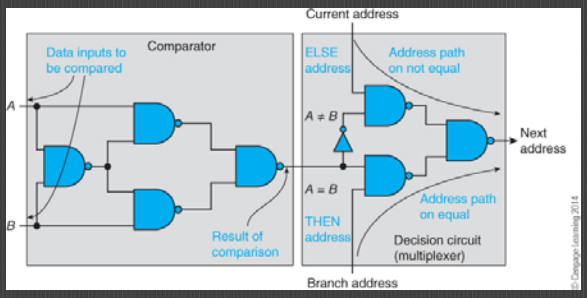
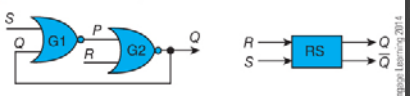


FIGURE 2.34 Feedback in a logic circuit



(a) Circuit of RS latch (b) Symbol of RS latch

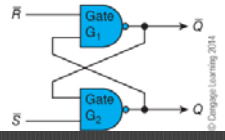
TABLE 2.17 Truth Table for the Circuit in Figure 2.34

Inputs			Output
R	*S*	*Q*	Q^+
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	?
1	1	1	?

TABLE 2.18 An Alternative Truth Table for the RS Latch

Inputs		Output	Description
R	*S*	Q^+	
0	0	*Q*	No change
0	1	1	Set output to 1
1	0	0	Reset output to 0
1	1	X	Forbidden[17]

FIGURE 2.35 RS latch constructed from two cross-coupled NAND gates



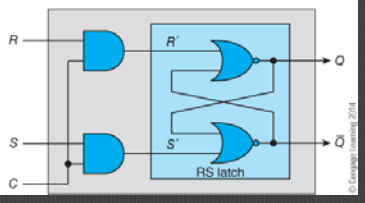
59

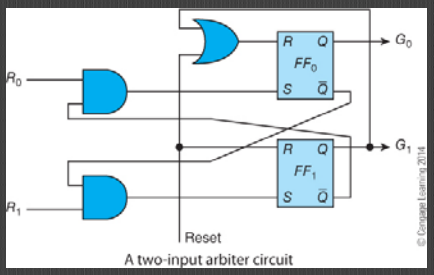
TABLE 2.19 Truth Table for an RS Latch Constructed from NAND Gates

Inputs		Output	Comment
R	S	Q^+	
0	0	X	Forbidden
0	1	1	Reset output to 0
1	0	0	Set output to 1
1	1	Q	No change

60

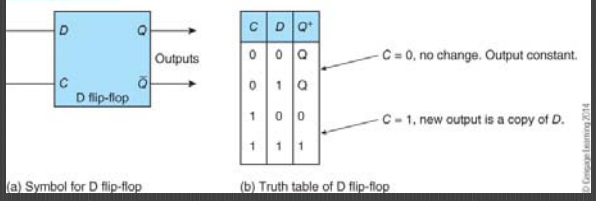
FIGURE 2.36 The clocked RS flip-flop





A two-input arbiter circuit

FIGURE 2.37 The D flip-flop



(a) Symbol for D flip-flop (b) Truth table of D flip-flop

FIGURE 2.38 Circuit of a D flip-flop

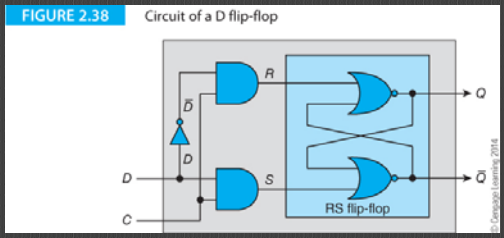
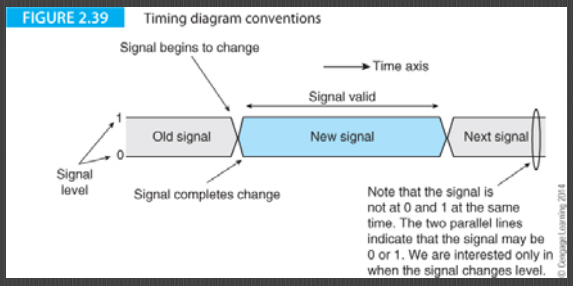
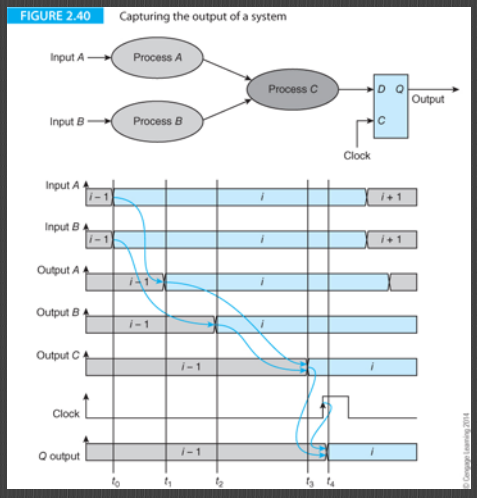


FIGURE 2.39 Timing diagram conventions



65

FIGURE 2.40 Capturing the output of a system



66

FIGURE 2.41 Pipelining using flip-flops

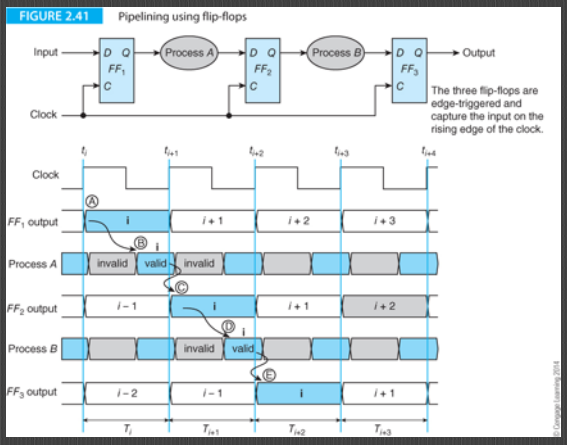
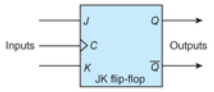


FIGURE 2.42 The JK flip-flop



J	K	Q	Q+
0	0	0	Q
0	0	1	Q
0	1	0	Q
0	1	1	Q
1	0	0	Q
1	0	1	0
1	1	0	1
1	1	1	0

J = K = 1 toggles output.

FIGURE 2.43 Construction of a basic JK flip-flop

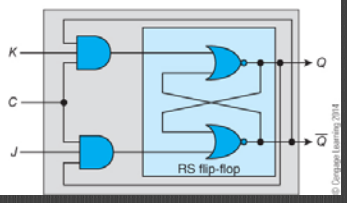


FIGURE 2.44 The register

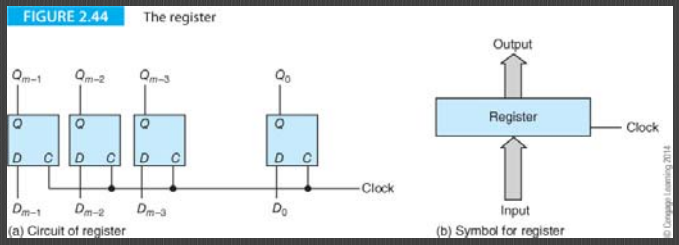


FIGURE 2.45 The shift register

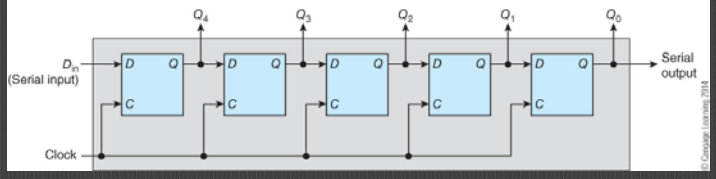


FIGURE 2.46 Timing diagram of a shift register

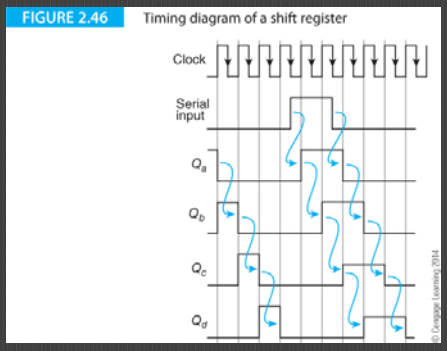


FIGURE 2.47 The left-shift register

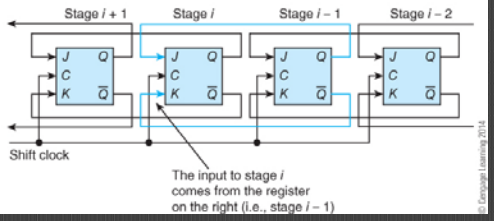


TABLE 2.20 The Effect of Logical, Arithmetic, and Circular Shifts

Shift Type	Shift Left	Shift Right
Original bit pattern before shift	11010111	11010111
Logical shift	10101110	01101011
Arithmetic shift	10101110	11101011
Circular shift	10101111	11101011

FIGURE 2.48 Shift register with a parallel load capability

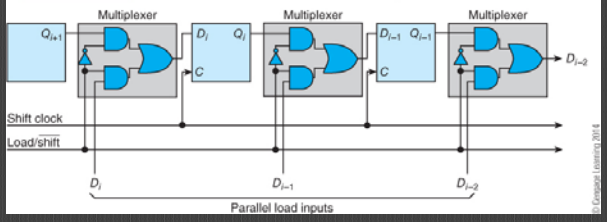


FIGURE 2.49 The binary counter

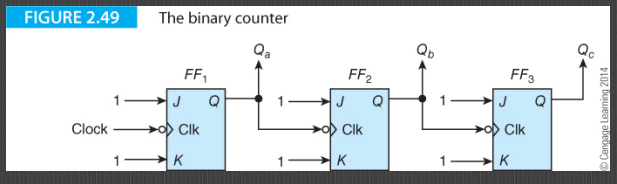


FIGURE 2.50 Timing diagram for the counter of Figure 2.49

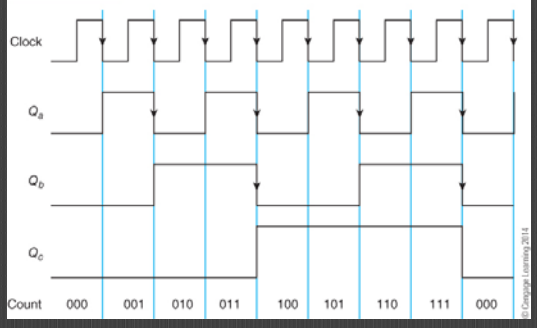
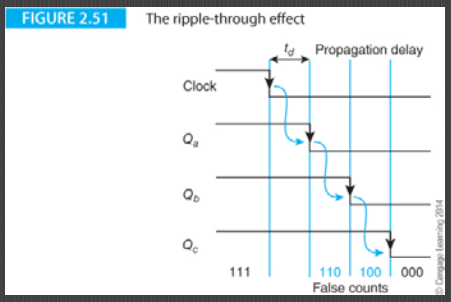
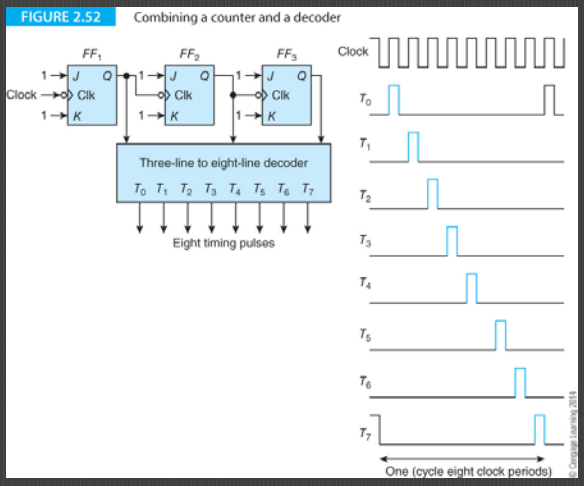


FIGURE 2.51 The ripple-through effect





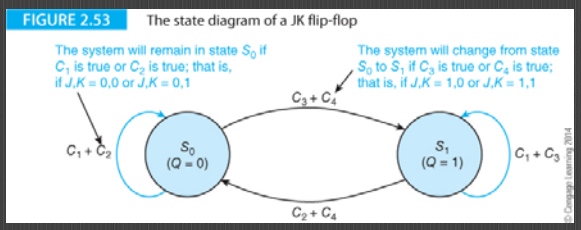


TABLE 2.21

Relationship Between JK Inputs and Conditions C_1 to C_4

J	K	Condition
0	0	C_1
0	1	C_2
1	0	C_3
1	1	C_4

TABLE 2.22

Excitation Table of a JK Flip-Flop

Inputs		Transition
J	K	$Q \rightarrow Q^+$
0	d	$0 \rightarrow 0$
1	d	$0 \rightarrow 1$
d	1	$1 \rightarrow 0$
d	0	$1 \rightarrow 1$

TABLE 2.23 Truth Table for a Synchronous Counter

	Output			Next State			J, K Inputs Required to Force Transition					
Count	Q_c	Q_b	Q_a	Q_c	Q_b	Q_a	J_c	K_c	J_b	K_b	J_a	K_a
0	0	0	0	0	1	0	0	d	1	d	0	d
2	0	1	0	1	0	0	1	d	d	1	0	d
4	1	0	0	1	1	0	d	0	1	d	0	d
6	1	1	0	1	1	1	d	0	d	0	1	d
7	1	1	1	1	0	1	d	0	d	1	d	0
5	1	0	1	0	1	1	d	1	1	d	d	0
3	0	1	1	0	0	0	0	d	d	1	d	1
0	0	0	0	0	1	0	0	d	1	d	0	d

83

TABLE 2.24 Truth Tables for the J, K Inputs for the System of Table 2.23

Q_c	Q_b	Q_a	J_c	Q_c	Q_b	Q_a	K_c	Q_c	Q_b	Q_a	J_b	Q_c	Q_b	Q_a	K_b	Q_c	Q_b	Q_a	J_a	Q_c	Q_b	Q_a	K_a
0	0	0	0	0	0	0	d	0	0	0	1	0	0	0	d	0	0	0	0	0	0	0	d
0	0	1	x	0	0	1	x	0	0	1	x	0	0	1	x	0	0	1	x	0	0	1	x
0	1	0	1	0	1	0	d	0	1	0	d	0	1	0	1	0	1	0	0	0	1	0	d
0	1	1	0	0	1	1	d	0	1	1	d	0	1	1	1	0	1	1	d	0	1	1	1
1	0	0	d	1	0	0	0	1	0	0	1	1	0	0	d	1	0	0	0	1	0	0	d
1	0	1	d	1	0	1	1	1	0	1	1	1	0	1	d	1	0	1	d	1	0	1	0
1	1	0	d	1	1	0	0	1	1	0	d	1	1	0	0	1	1	0	1	1	1	0	d
1	1	1	d	1	1	1	0	1	1	1	d	1	1	1	1	1	1	1	d	1	1	1	0

84

FIGURE 2.54 Karnaugh maps for Table 2.23

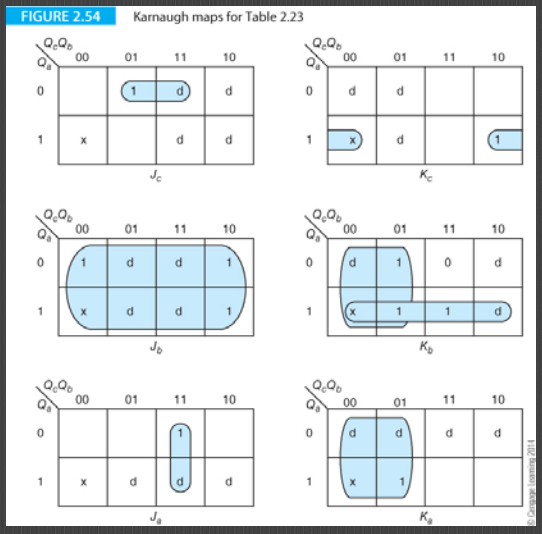


FIGURE 2.55 Circuit of the synchronous counter of Table 2.23

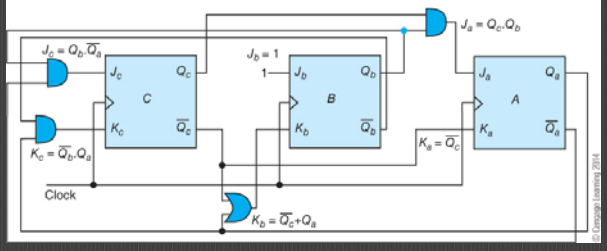


FIGURE 2.56 The tristate gate (tristate buffer)

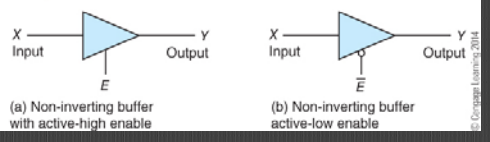


FIGURE 2.57 Registers and buses

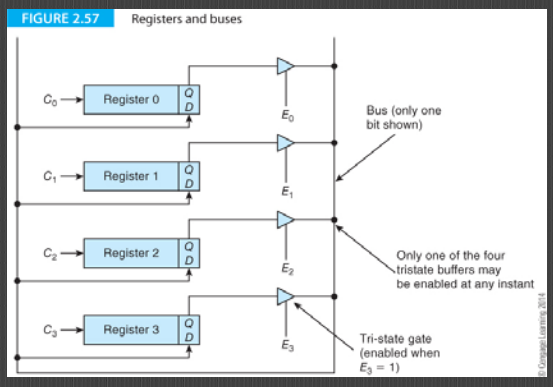


FIGURE 2.58 Controlling the bus

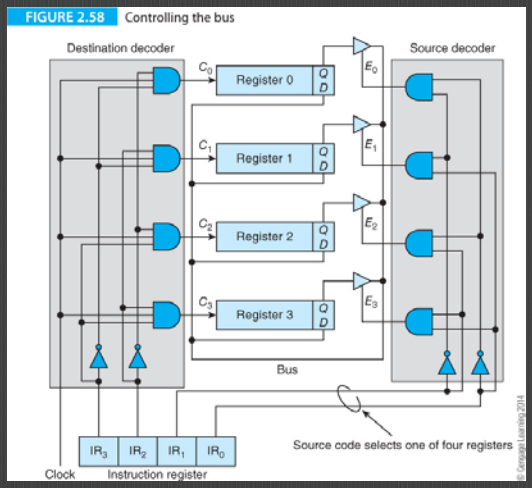


FIGURE 2.59 The registers, buses, and ALU

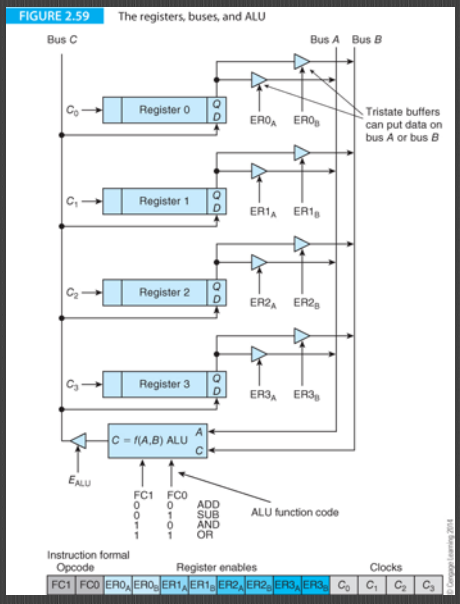


TABLE 2.25 Microinstructions

Operation	Bus Driver Enables									Register Clocks				ALU Control	
	$ER0_A$	$ER0_B$	$ER1_A$	$ER1_B$	$ER2_A$	$ER2_B$	$ER3_A$	$ER3_B$	E_{ALU}	C_0	C_1	C_2	C_3	FC1	FC2
ADD **R2**,R1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	0
SUB **R2**,R0	1	0	0	0	0	1	0	0	1	0	0	1	0	0	1
MOVE **R1**,R0	1	1	0	0	0	0	0	0	1	0	1	0	0	1	1

FIGURE P2.34

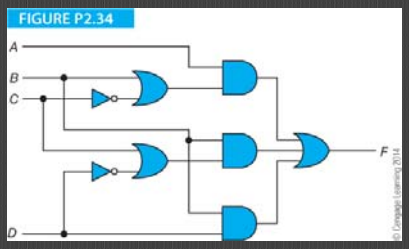


FIGURE P2.38

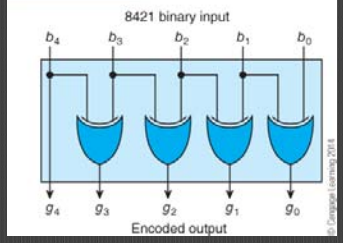


TABLE P2.39

Condition	X	Y	Z
$A > B$	1	0	0
$A < B$	0	1	0
$A = B$	0	0	1

FIGURE P2.40

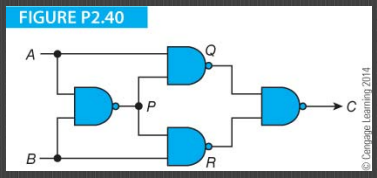


FIGURE P2.41

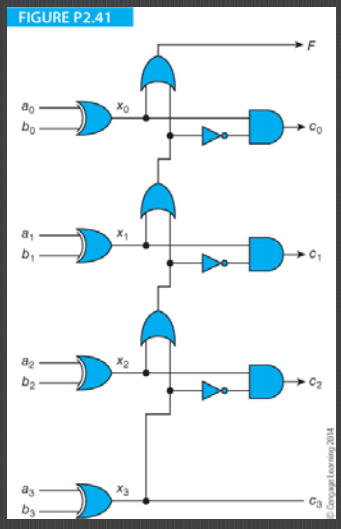


FIGURE P2.49

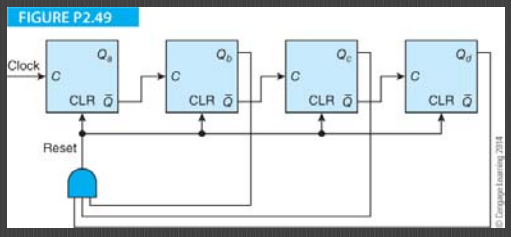


FIGURE P2.50

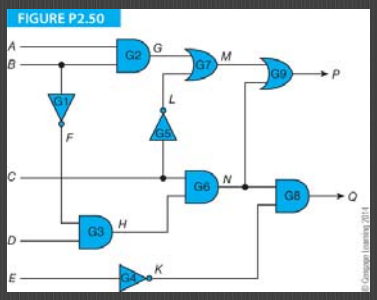


FIGURE P2.51

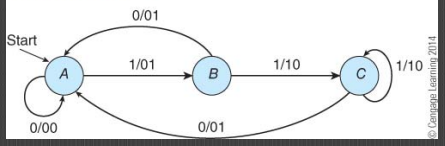


FIGURE P2.52