ESE DLD

Digital Logic design (National University of Technology)

Notes:

Conversions of numbers:

Octal to Decimal:

$(127:4)_8$ to decimal equivalent

$1 \times 8^2 + 2 \times 8^1 + 7 \times 8^0 + 4 \times 8^{-1} = (87.5)_{10}$

Hexadecimal to Decimal:

10 = A

11 = B

12 = C

13 = D

14 = E

15 = F

$(B65F)_{16}$ to decimal equivalent

$11 \times 16^3 + 6 \times 16^2 + 5 \times 16^1 + 15 \times 16^0 = (46687)_{10}$

Decimal to Binary:

Convert decimal 41 to binary

Divisor:	Number:	Remainders:
2	41	
2	20	1
2	10	0
2	5	0
2	2	1
2	1	0
	0	1

$(41)_{10} = (101001)_2$

Decimal to Octal:

Convert decimal 33 to octal

Divisor:	Number:	Remainders:
8	33	
8	4	1
8	0	4

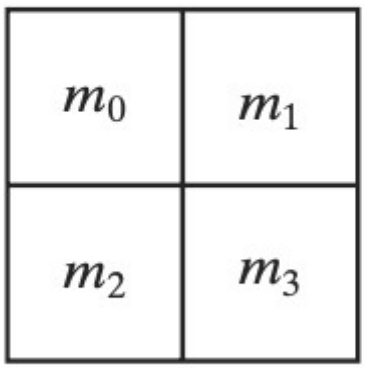
$(33)_{10} = (41)_8$

Binary to -:

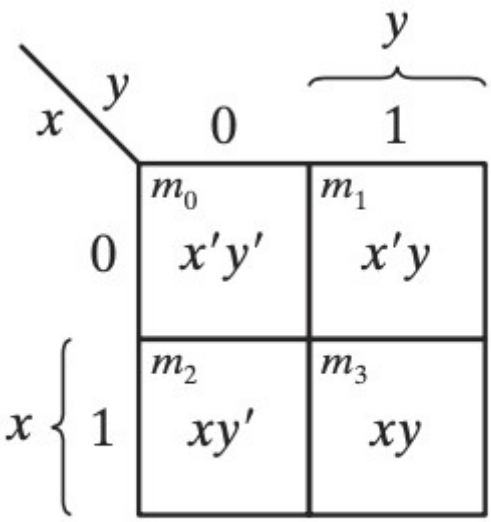
Converting is similar to any number to decimal but with a base multiplier of 2.

K Maps:

Two variable:



(a)

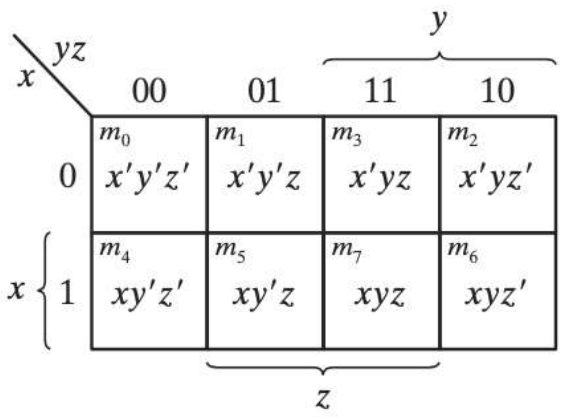


(b)

Three variable:

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6

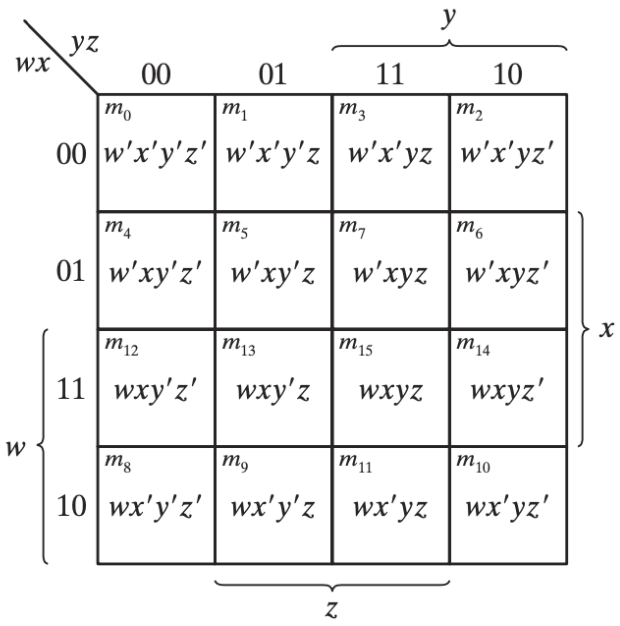
(a)



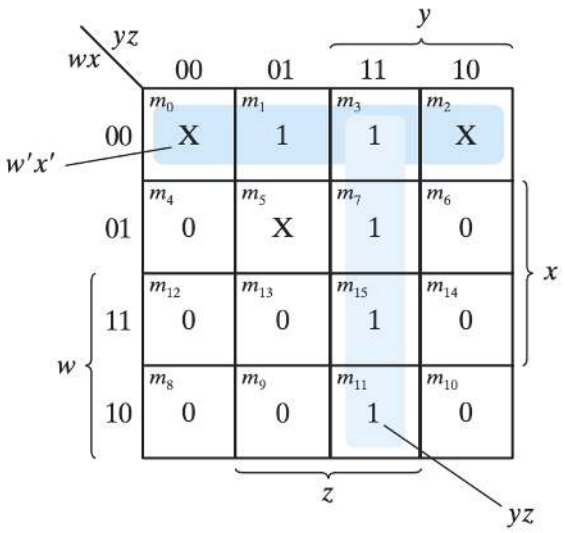
(b)

Four variable:

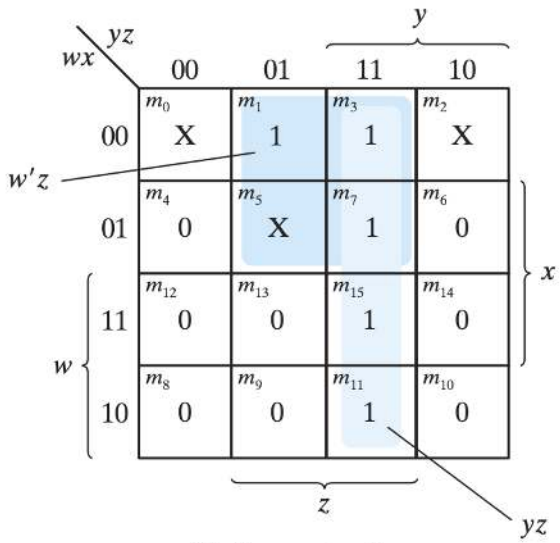
m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6
m_{12}	m_{13}	m_{15}	m_{14}
m_8	m_9	m_{11}	m_{10}



Don’t care:

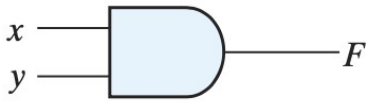
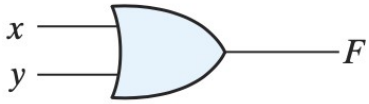
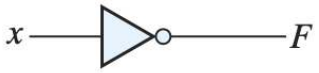


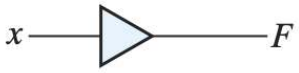
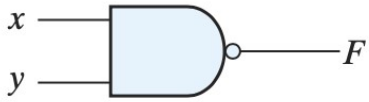
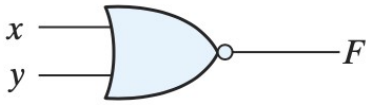
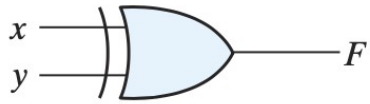
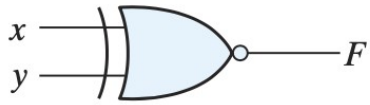
(a) $F = yz + w'x'$



(b) $F = yz + w'z$

Logic gates:

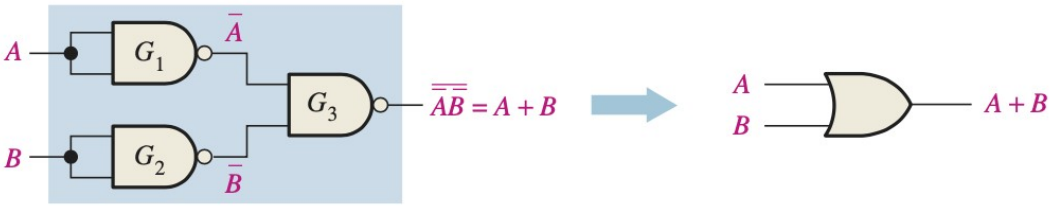
Name	Graphic symbol	Algebraic function	Truth table
AND		$F = x \cdot y$	x y | F 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1
OR		$F = x + y$	x y | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1
Inverter		$F = x'$	x | F 0 | 1 1 | 0

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

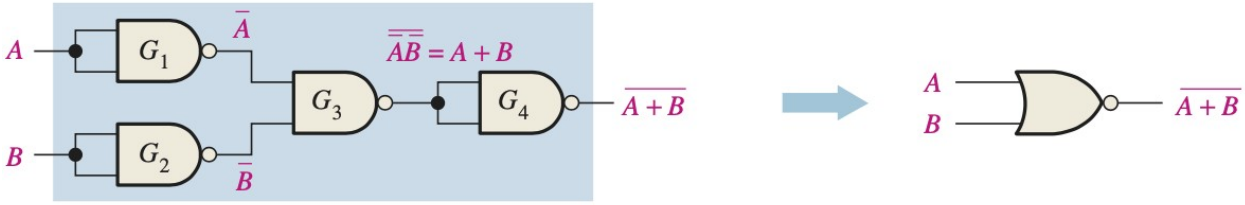
The NAND gate, as a universal logic element:

(a) One NAND gate used as an inverter

(b) Two NAND gates used as an AND gate

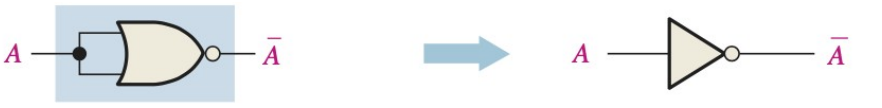


(c) Three NAND gates used as an OR gate



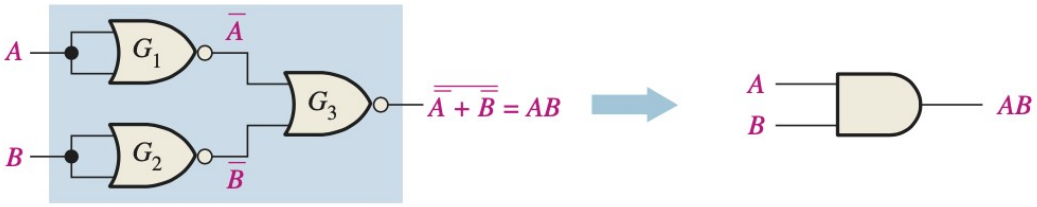
(d) Four NAND gates used as a NOR gate

The NOR gate, as a universal logic element:

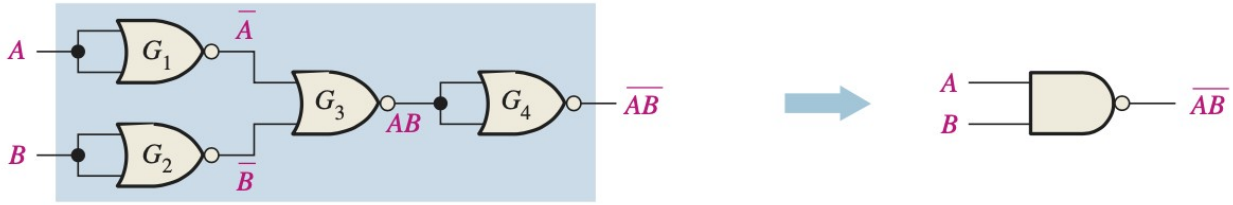


(a) One NOR gate used as an inverter

(b) Two NOR gates used as an OR gate



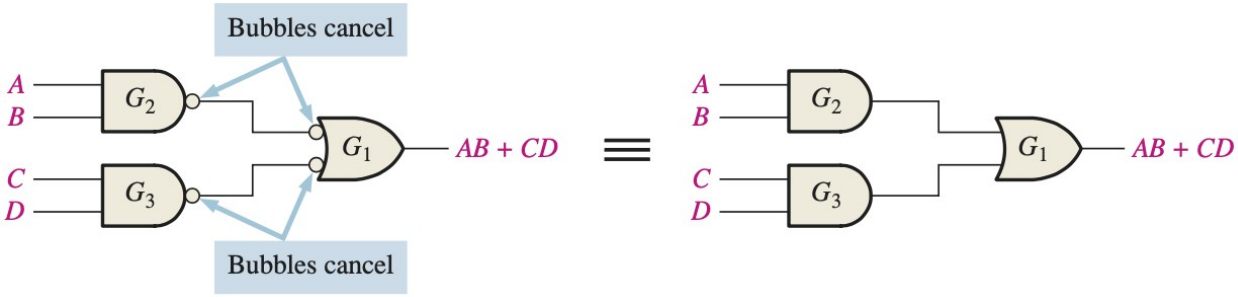
(c) Three NOR gates used as an AND gate

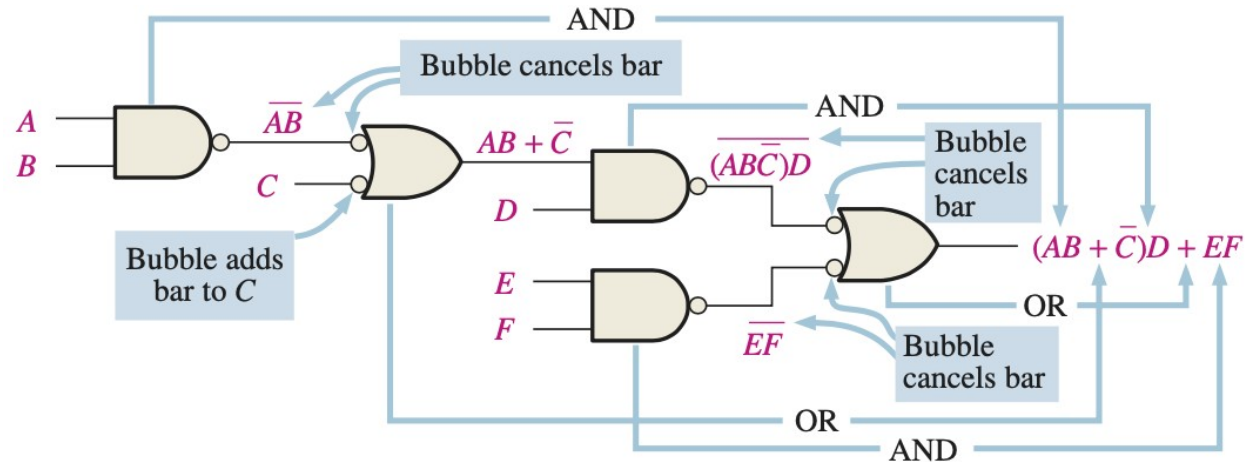


(d) Four NOR gates used as a NAND gate

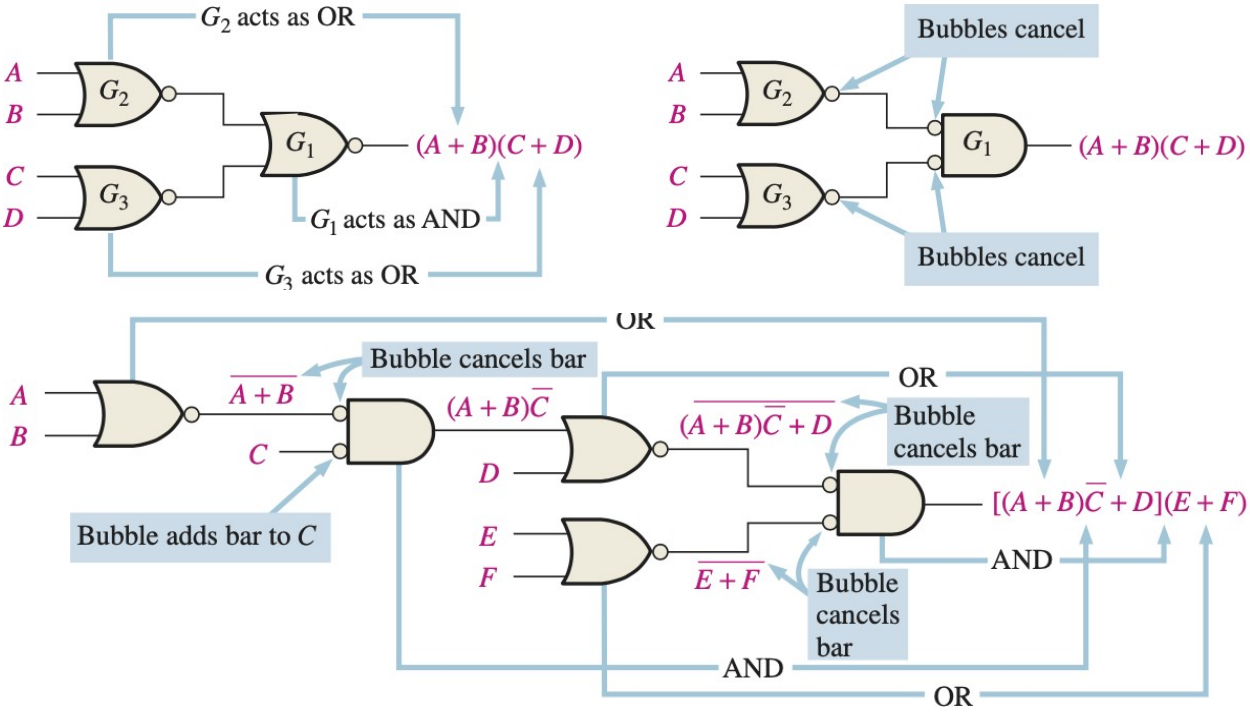
Combinational logic:

NAND logic:

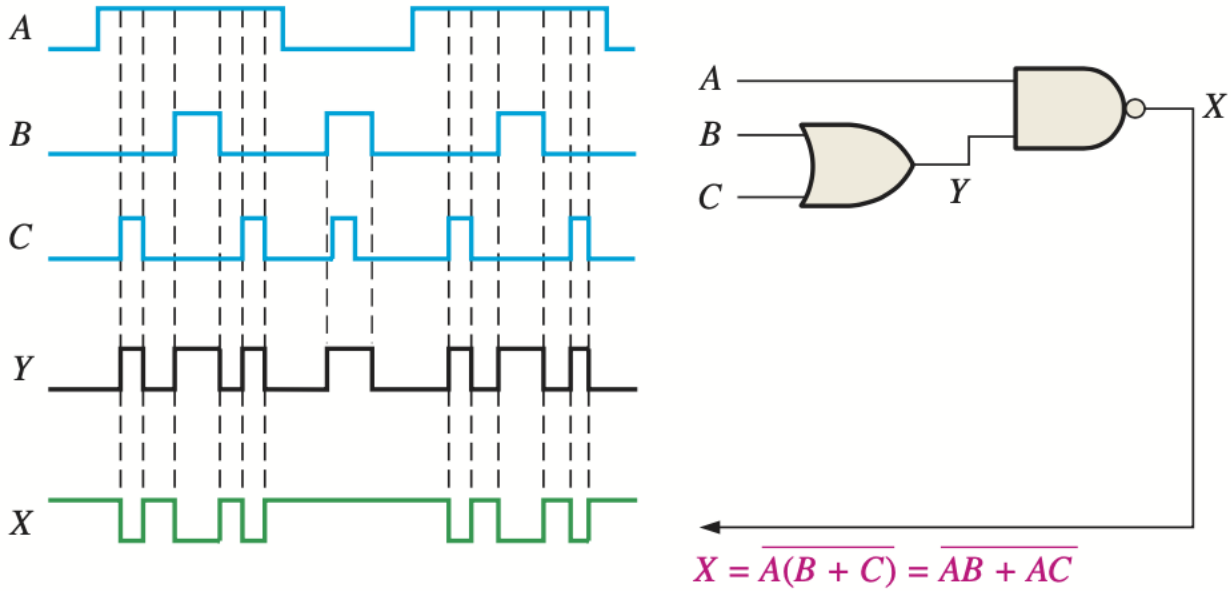




NOR logic:

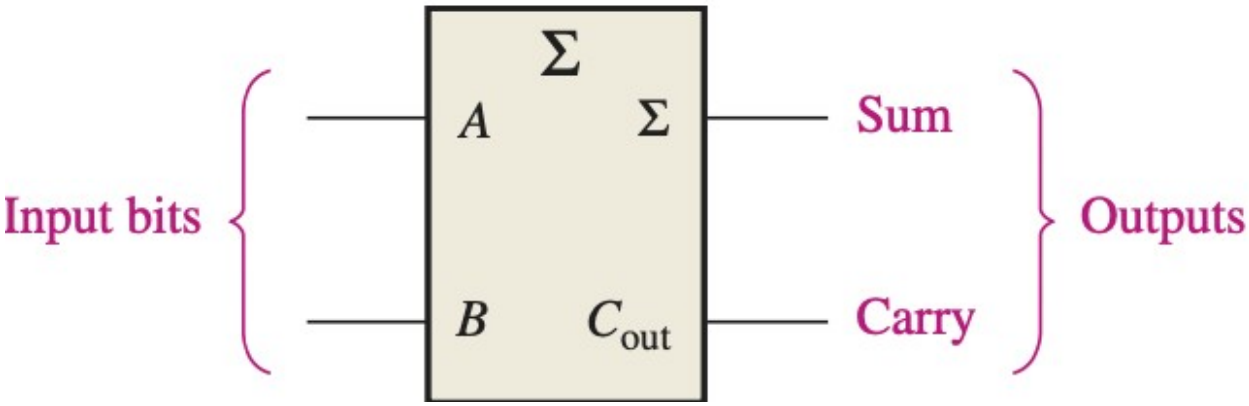


Pulse waveform operation:



Functions of combinational logic:

Half adder:



Half-adder truth table.

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

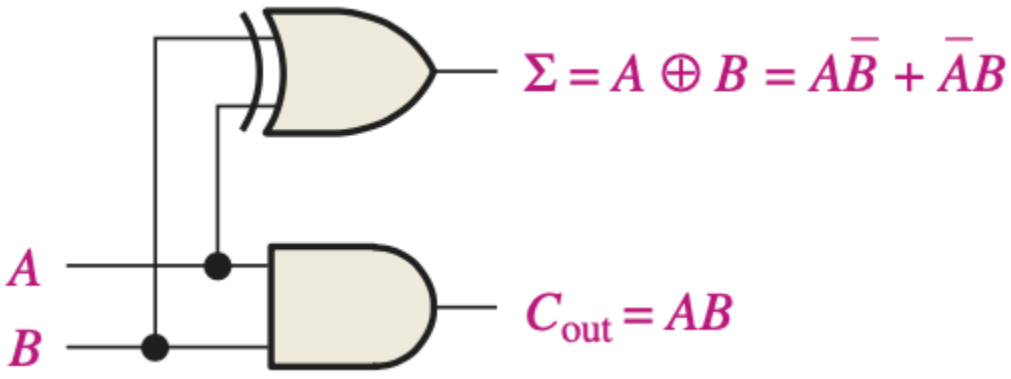
Σ = sum

C_{out} = output carry

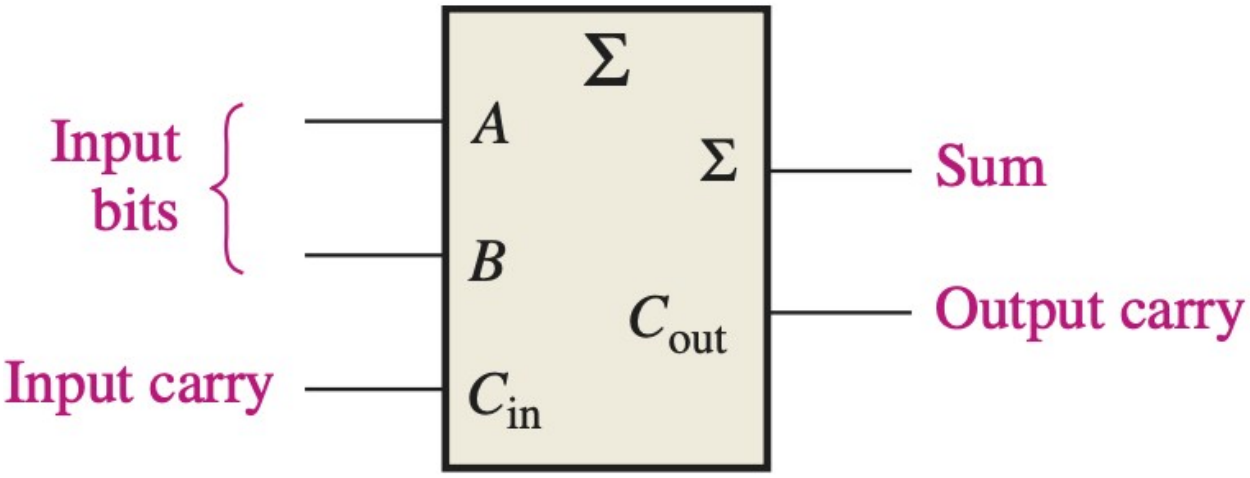
A and B = input variables (operands)

$$C_{out} = AB$$

$$\Sigma = A \oplus B$$



Full adder:



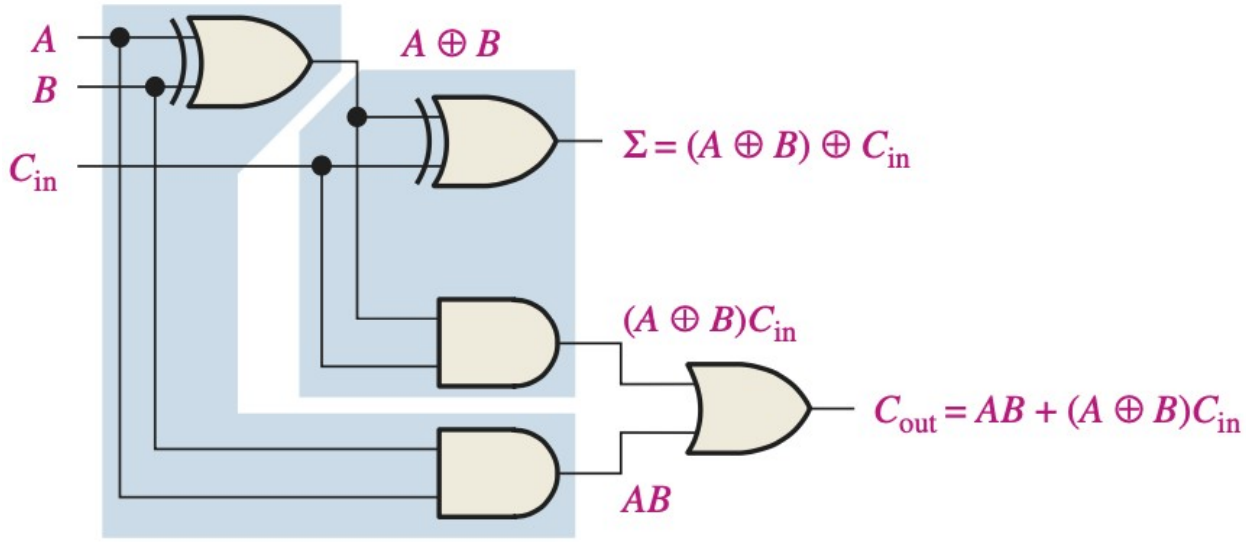
Full-adder truth table.

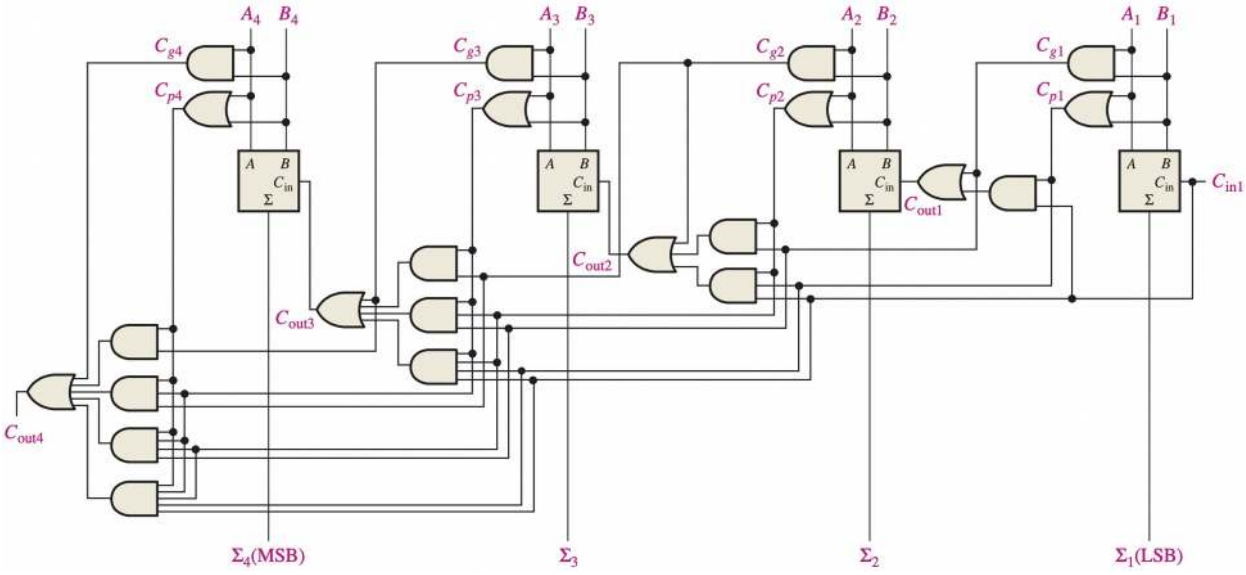
A	B	C_{in}	C_{out}	Σ
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

C_{in} = input carry, sometimes designated as CI
C_{out} = output carry, sometimes designated as CO
Σ = sum
A and B = input variables (operands)

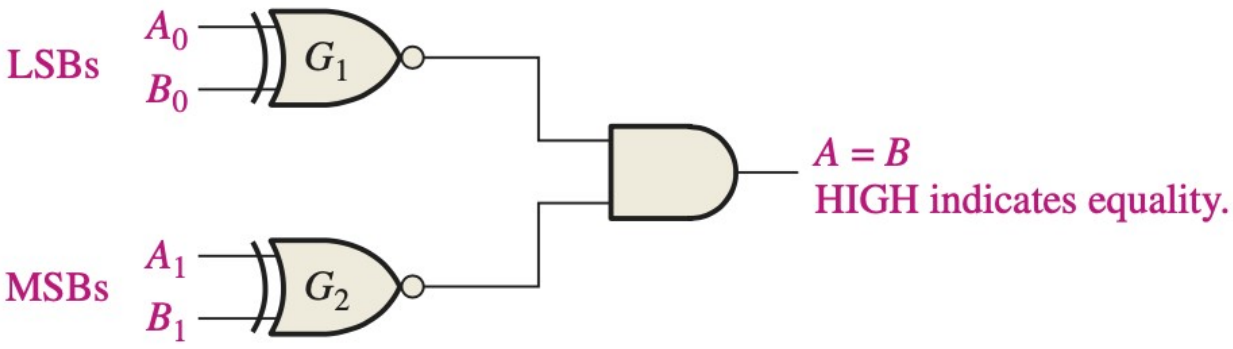
$$\Sigma = (A \oplus B) \oplus C_{\text{in}}$$

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





Comparators:



General format: Binary number $A \rightarrow A_1A_0$
Binary number $B \rightarrow B_1B_0$

Determine the $A = B$, $A > B$, and $A < B$ outputs for the input numbers shown on the comparator in Figure 6–22.

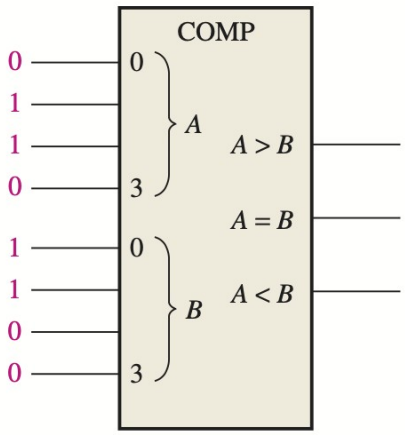
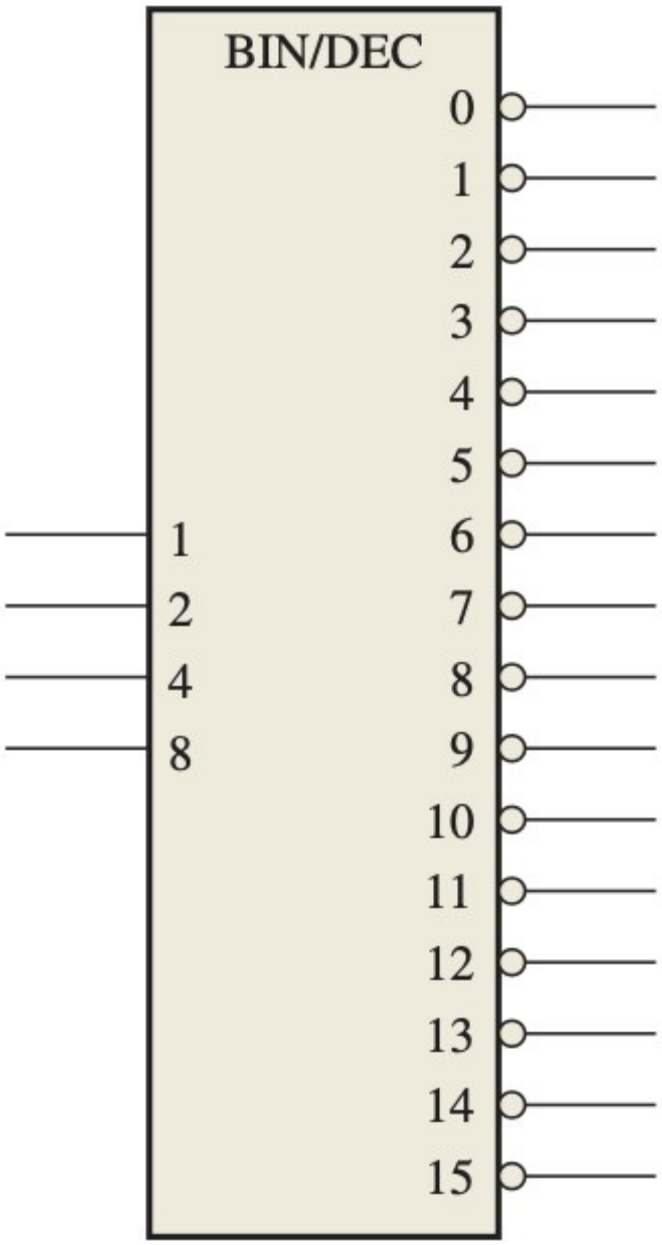


FIGURE 6–22

Decoders:

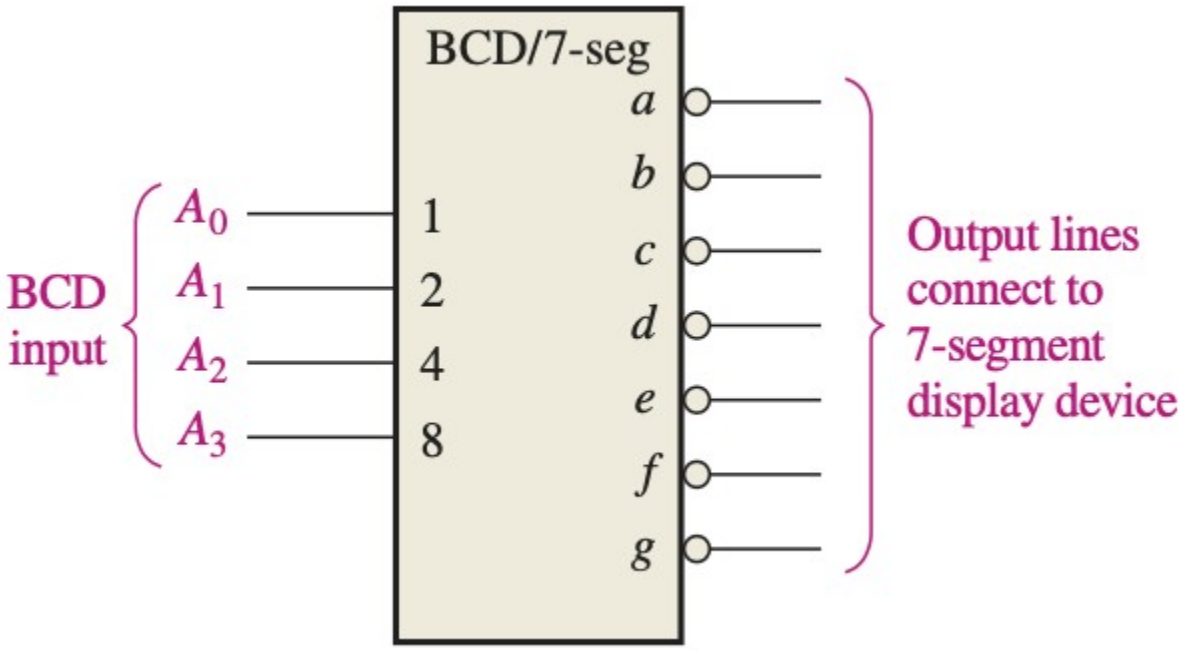
Decoding functions and truth table for a 4-line-to-16-line (1-of-16) decoder with active-LOW outputs.

Decimal Digit	Binary Inputs				Decoding Function	Outputs														
	A_3	A_2	A_1	A_0		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	$\overline{A}_3\overline{A}_2\overline{A}_1\overline{A}_0$	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	0	0	1	$\overline{A}_3\overline{A}_2\overline{A}_1A_0$	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	0	0	1	0	$\overline{A}_3\overline{A}_2A_1\overline{A}_0$	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
3	0	0	1	1	$\overline{A}_3\overline{A}_2A_1A_0$	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
4	0	1	0	0	$\overline{A}_3A_2\overline{A}_1\overline{A}_0$	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
5	0	1	0	1	$\overline{A}_3A_2\overline{A}_1A_0$	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
6	0	1	1	0	$\overline{A}_3A_2A_1\overline{A}_0$	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
7	0	1	1	1	$\overline{A}_3A_2A_1A_0$	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
8	1	0	0	0	$A_3\overline{A}_2\overline{A}_1\overline{A}_0$	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
9	1	0	0	1	$A_3\overline{A}_2\overline{A}_1A_0$	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
10	1	0	1	0	$A_3\overline{A}_2A_1\overline{A}_0$	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
11	1	0	1	1	$A_3\overline{A}_2A_1A_0$	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
12	1	1	0	0	$A_3A_2\overline{A}_1\overline{A}_0$	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
13	1	1	0	1	$A_3A_2\overline{A}_1A_0$	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
14	1	1	1	0	$A_3A_2A_1\overline{A}_0$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
15	1	1	1	1	$A_3A_2A_1A_0$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

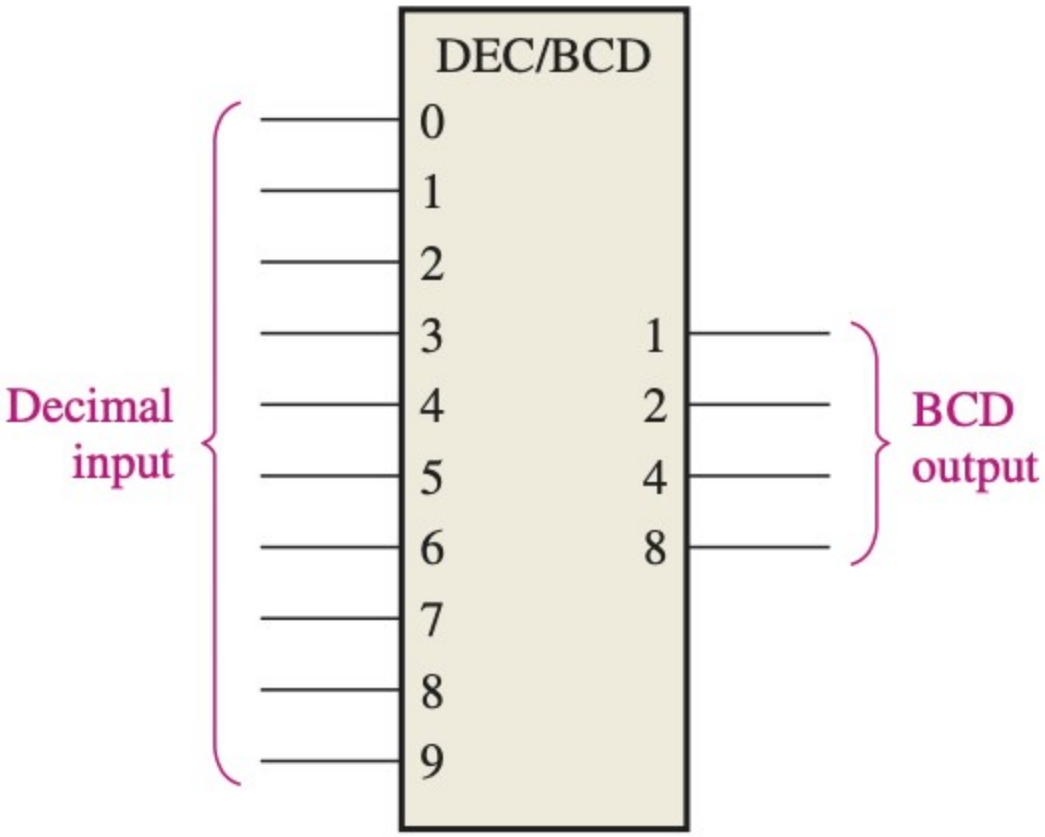


BCD decoding functions.

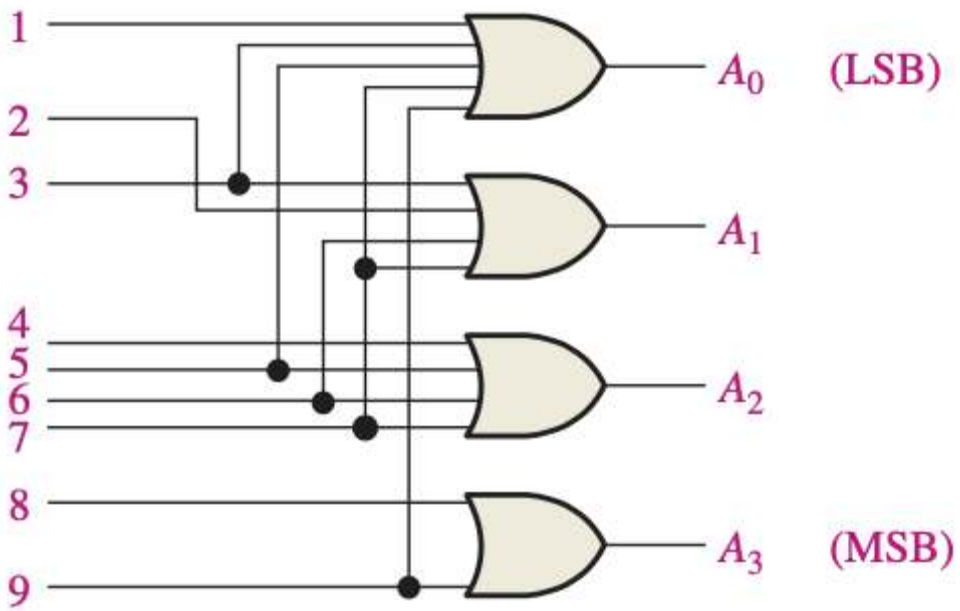
Decimal Digit	BCD Code A_3	A_2	A_1	A_0	Decoding Function
0	0	0	0	0	$\bar{A}_3\bar{A}_2\bar{A}_1\bar{A}_0$
1	0	0	0	1	$\bar{A}_3\bar{A}_2\bar{A}_1A_0$
2	0	0	1	0	$\bar{A}_3\bar{A}_2A_1\bar{A}_0$
3	0	0	1	1	$\bar{A}_3\bar{A}_2A_1A_0$
4	0	1	0	0	$\bar{A}_3A_2\bar{A}_1\bar{A}_0$
5	0	1	0	1	$\bar{A}_3A_2\bar{A}_1A_0$
6	0	1	1	0	$\bar{A}_3A_2A_1\bar{A}_0$
7	0	1	1	1	$\bar{A}_3A_2A_1A_0$
8	1	0	0	0	$A_3\bar{A}_2\bar{A}_1\bar{A}_0$
9	1	0	0	1	$A_3\bar{A}_2\bar{A}_1A_0$



Encoders:

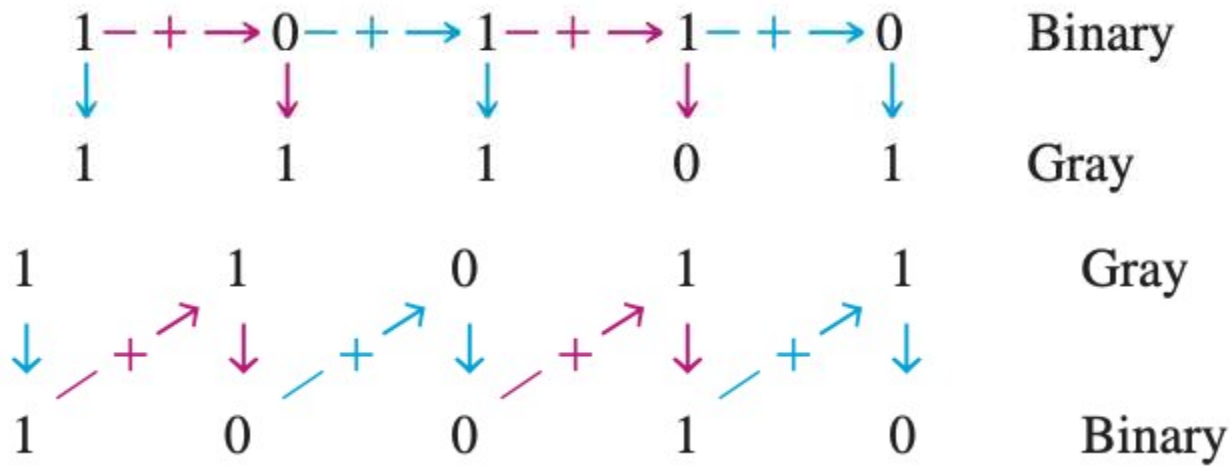


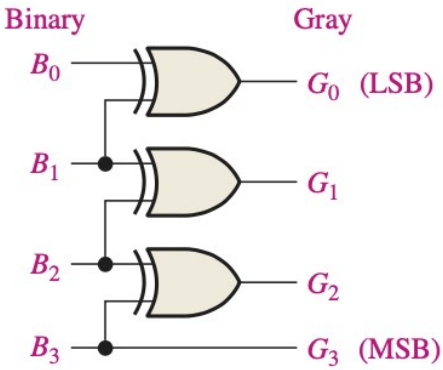
Decimal Digit	BCD Code A_3	A_2	A_1	A_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

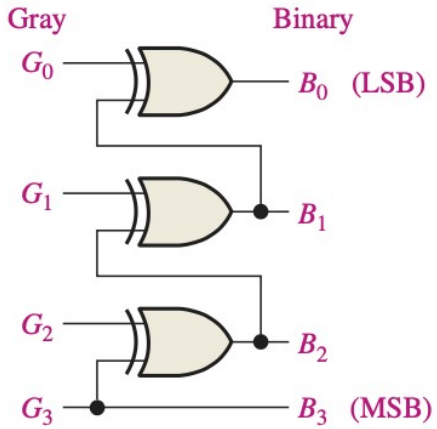


Code converters:

Gray code:







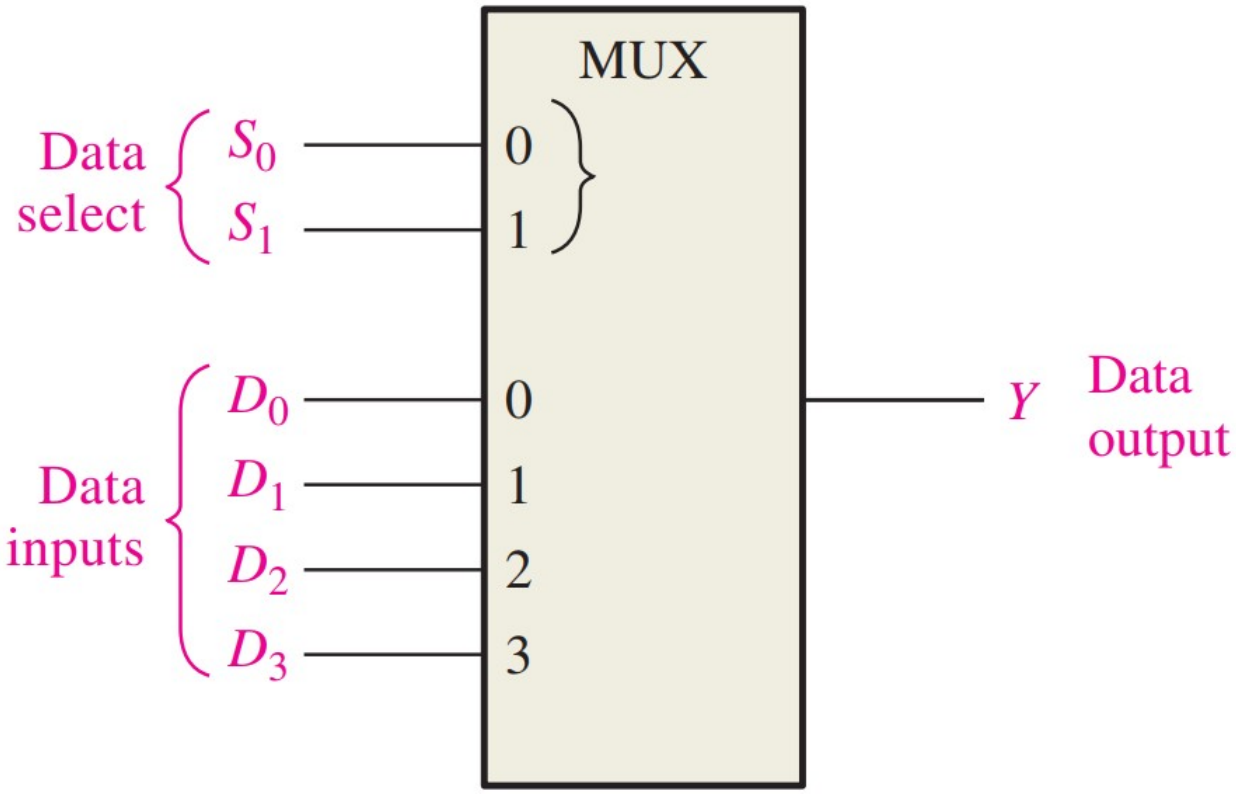
BCD to Binary:

Binary representations of BCD bit weights.

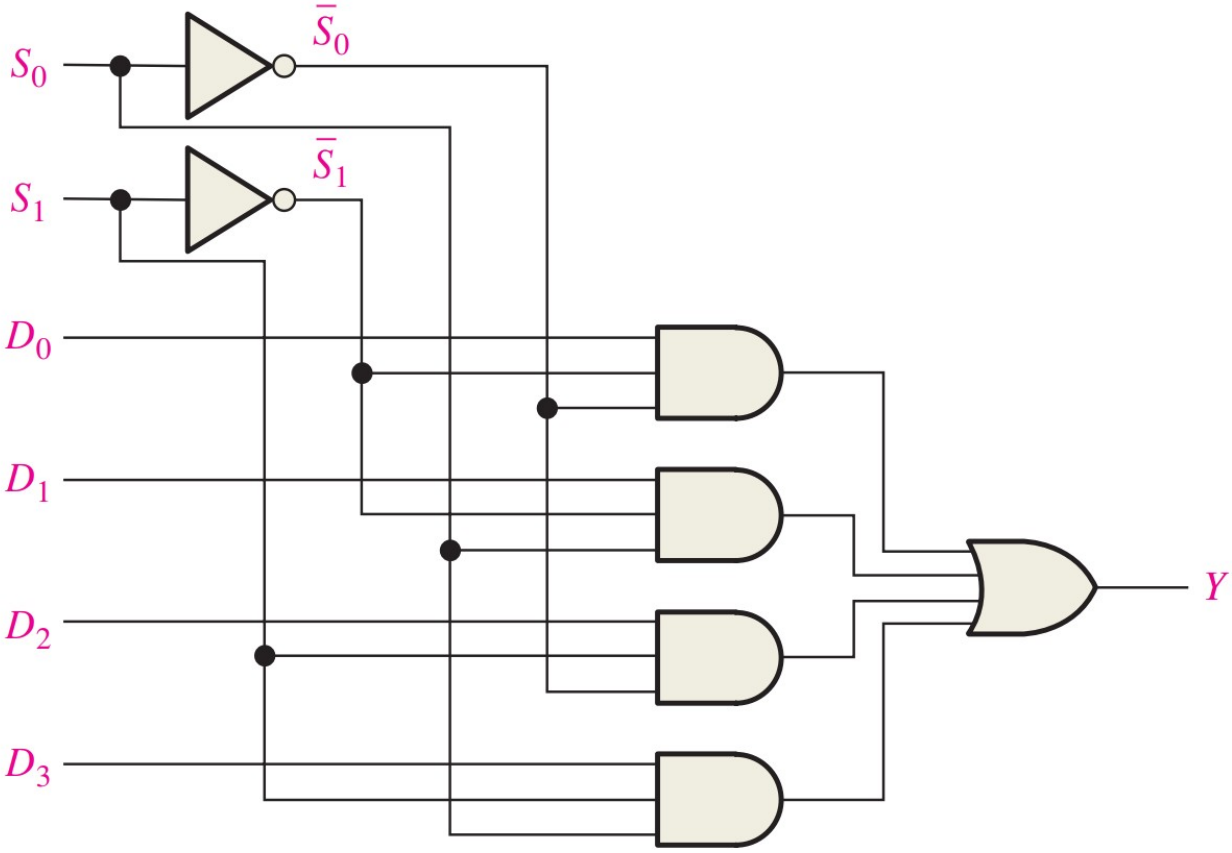
BCD Bit	BCD Weight	(MSB) 64	32	16	8	4	2	(LSB) 1
		Binary Representation						
A_0	1	0	0	0	0	0	0	1
A_1	2	0	0	0	0	0	1	0
A_2	4	0	0	0	0	1	0	0
A_3	8	0	0	0	1	0	0	0
B_0	10	0	0	0	1	0	1	0
B_1	20	0	0	1	0	1	0	0
B_2	40	0	1	0	1	0	0	0
B_3	80	1	0	1	0	0	0	0

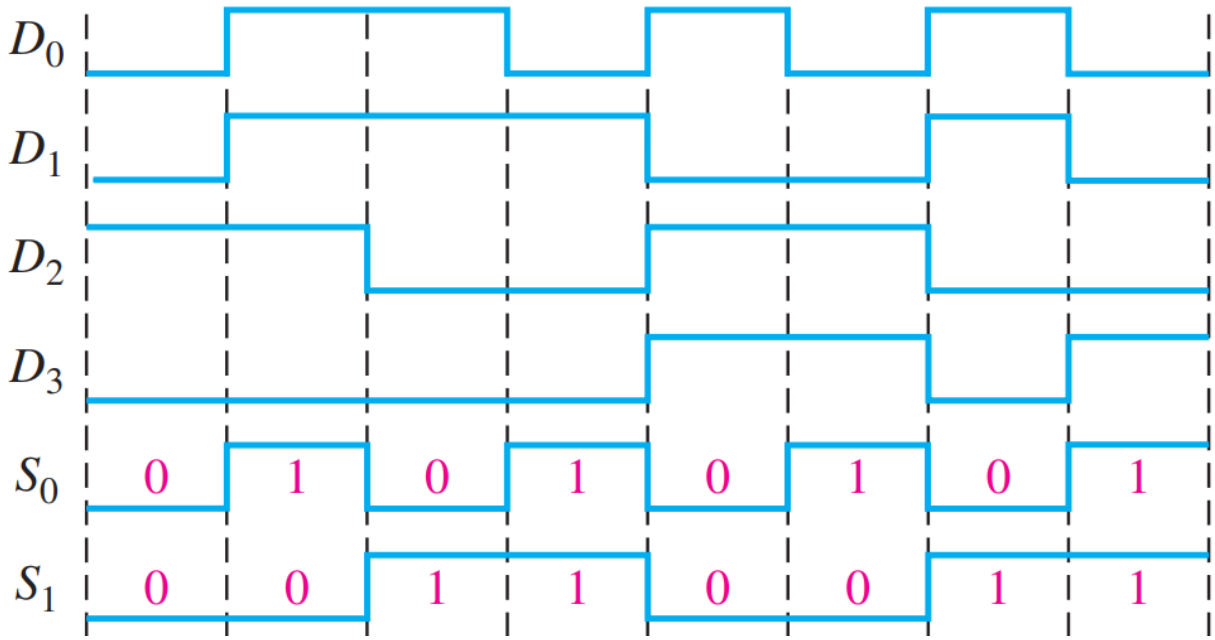
80	40	20	10	8	4	2	1	
0	0	1	0	0	1	1	1	
							0000001	1
							0000010	2
							0000100	4
						+	0010100	20
							0011011	Binary number for decimal 27

Multiplexers:

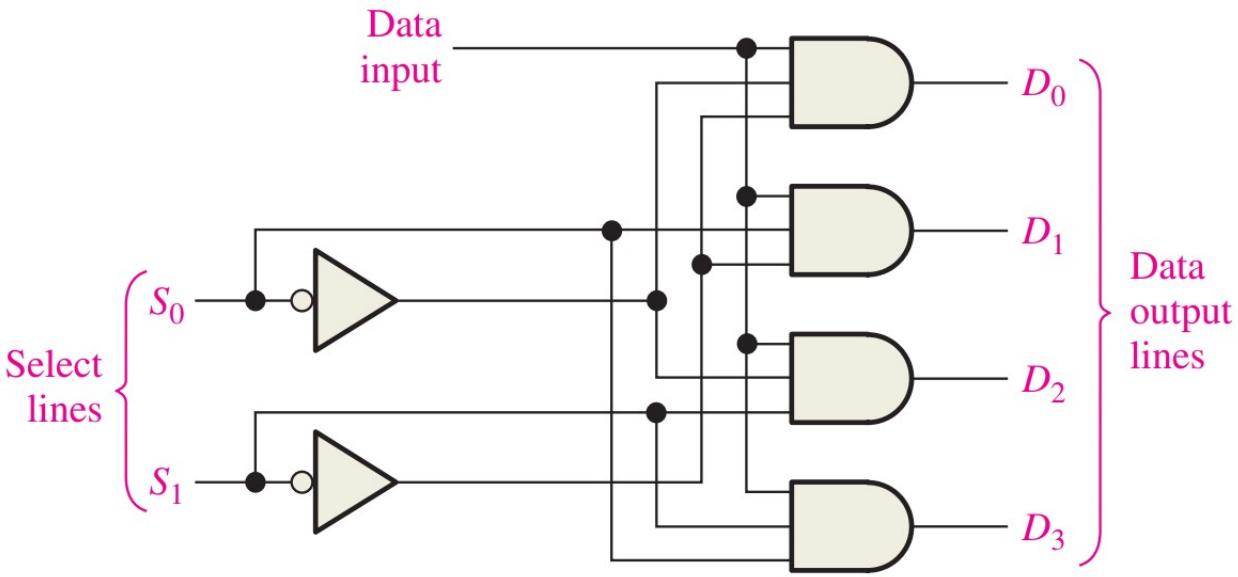


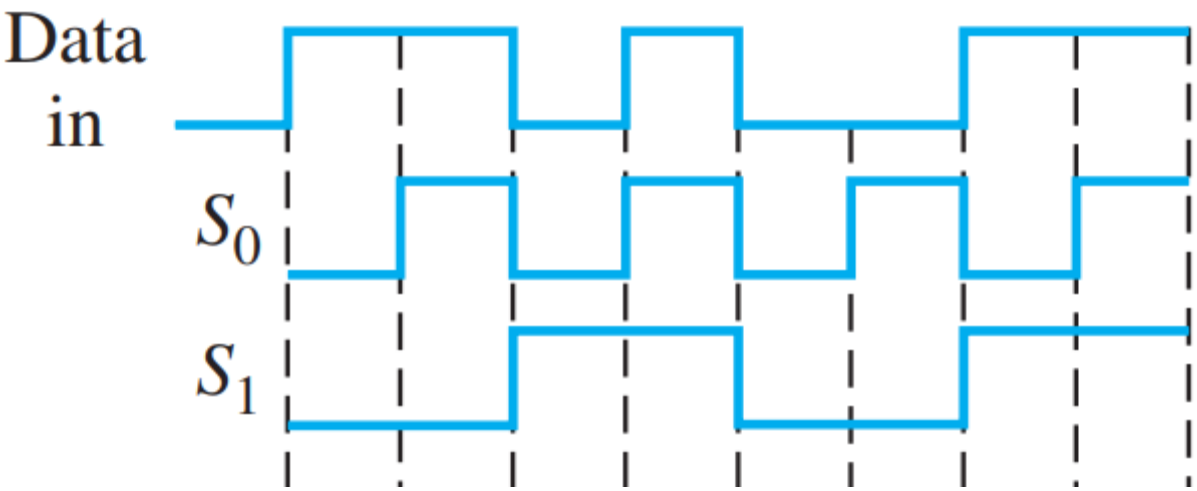
Data-Select Inputs		
S_1	S_0	Input Selected
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

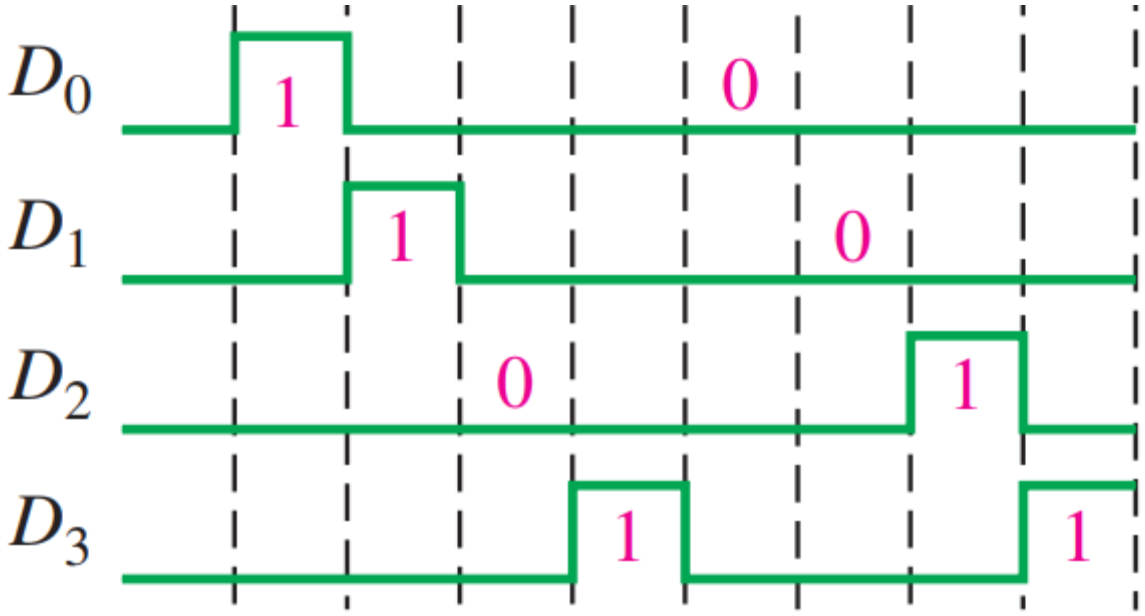




Demultiplexer:

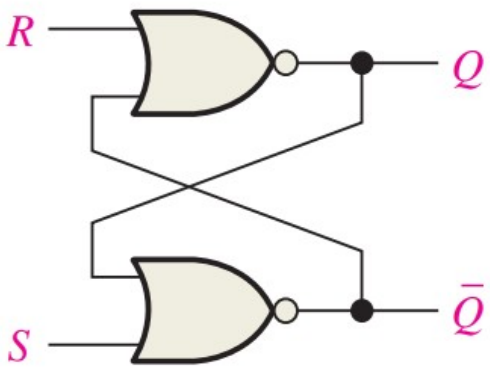


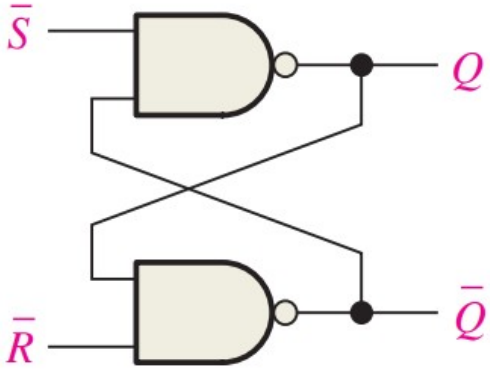


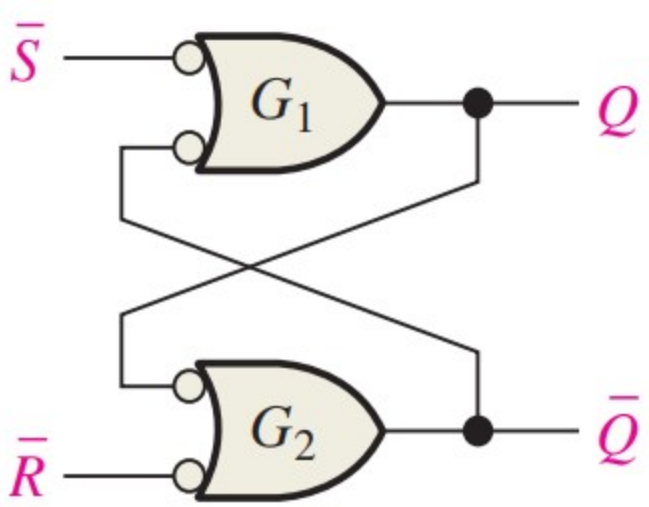


Latches:

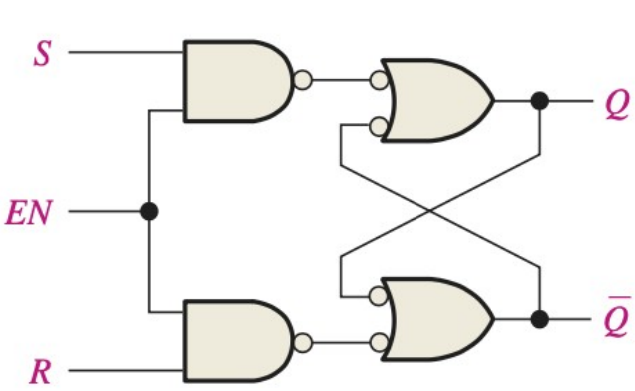
S-R Latch:

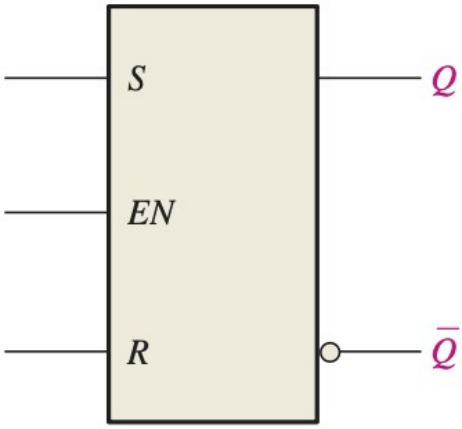




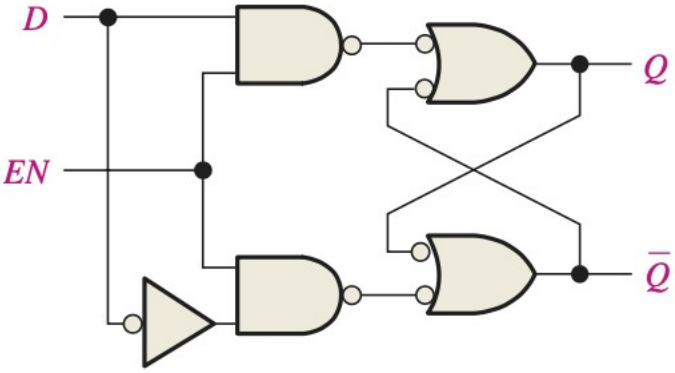


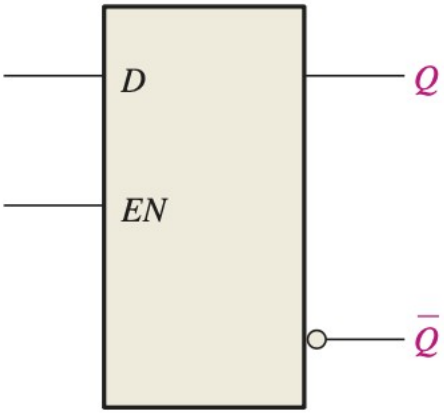
Gated S-R Latch:



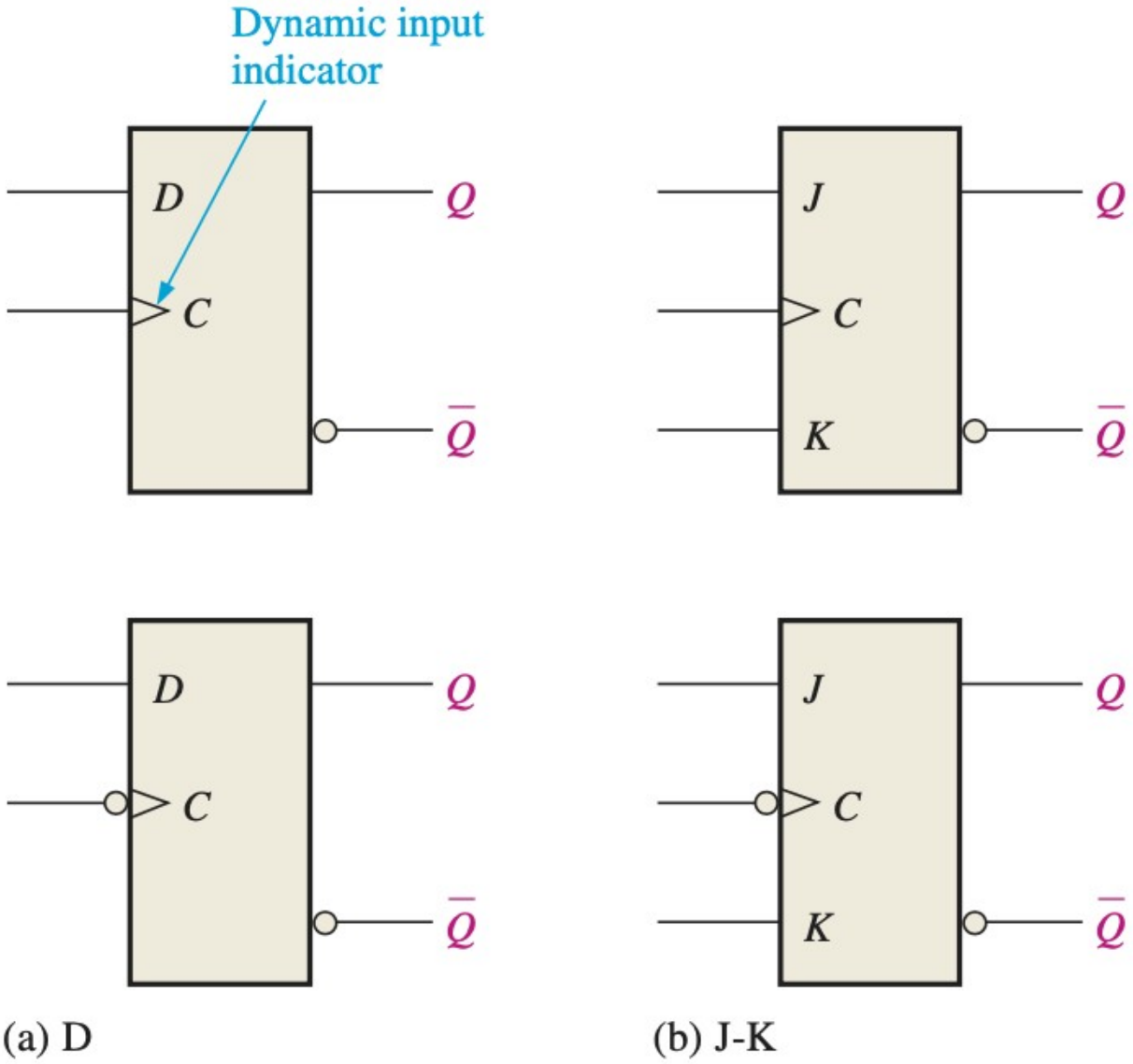


Gated D-Latch:

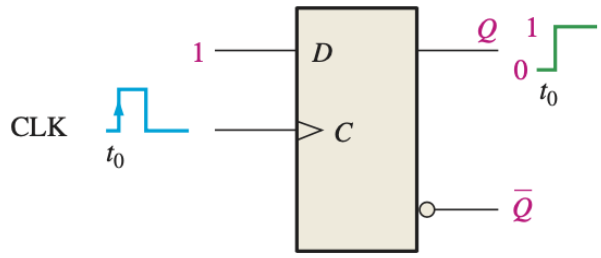




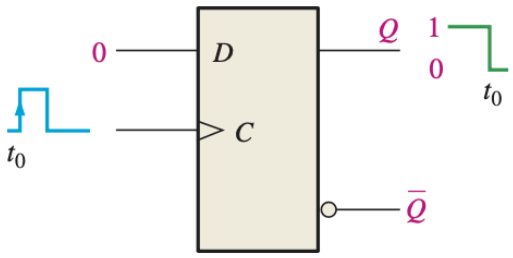
Flip flops:



D-Flip flops:

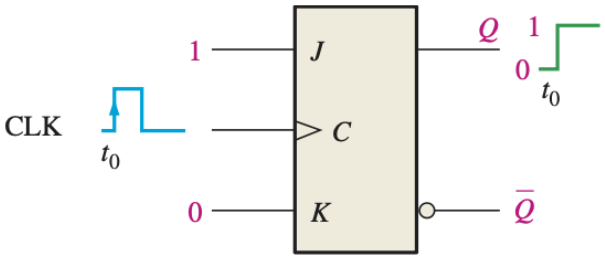


(a) $D = 1$ flip-flop SETS on positive clock edge. (If already SET, it remains SET.)

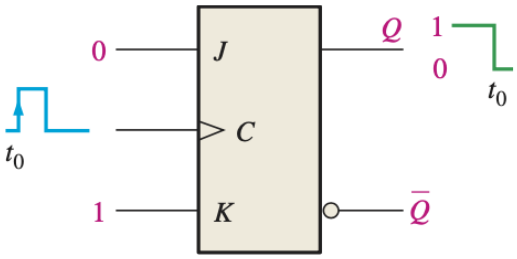


(b) $D = 0$ flip-flop RESETS on positive clock edge. (If already RESET, it remains RESET.)

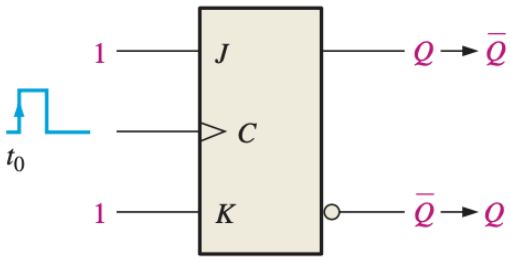
J-K Flip flops:



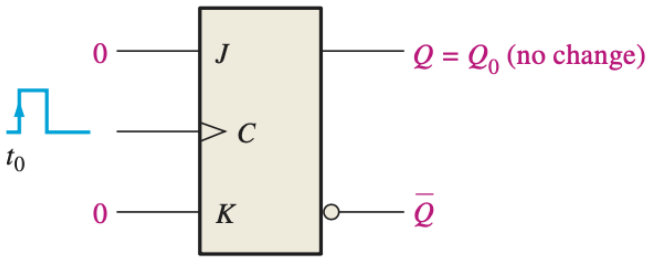
(a) $J = 1$, $K = 0$ flip-flop SETS on positive clock edge. (If already SET, it remains SET.)



(b) $J = 0$, $K = 1$ flip-flop RESETS on positive clock edge. (If already RESET, it remains RESET.)

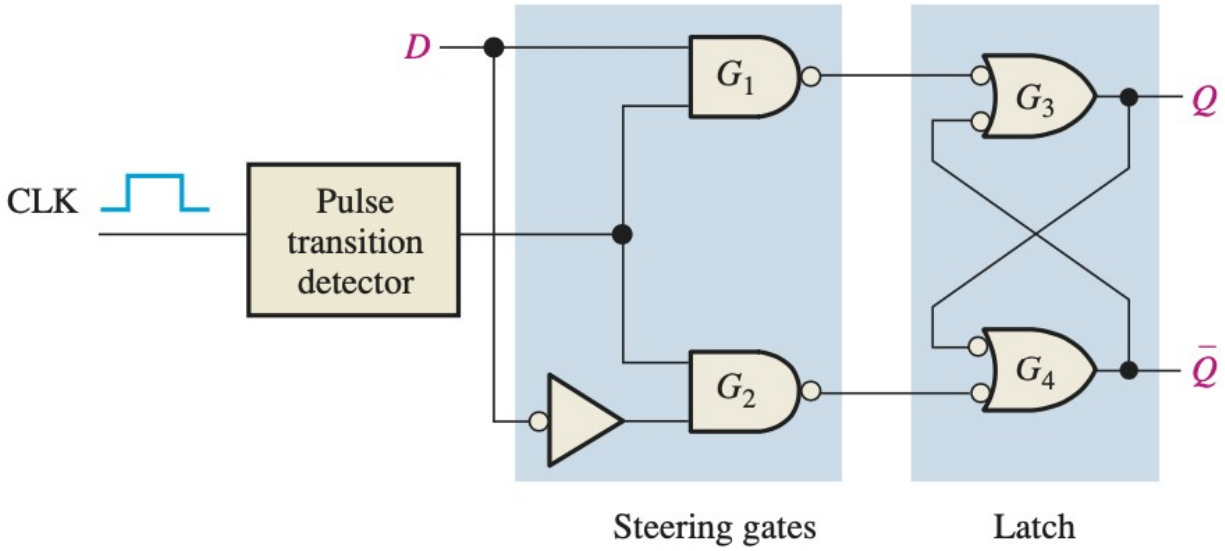


(c) $J = 1$, $K = 1$ flip-flop changes state (toggle).

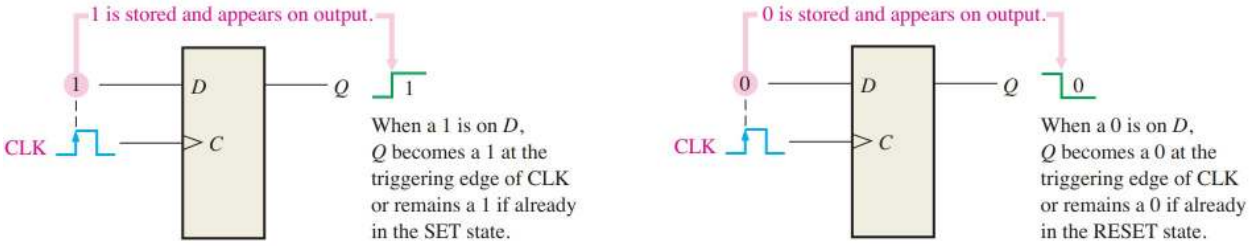


(d) $J = 0$, $K = 0$ flip-flop does not change. (If SET, it remains SET; if RESET, it remains RESET.)

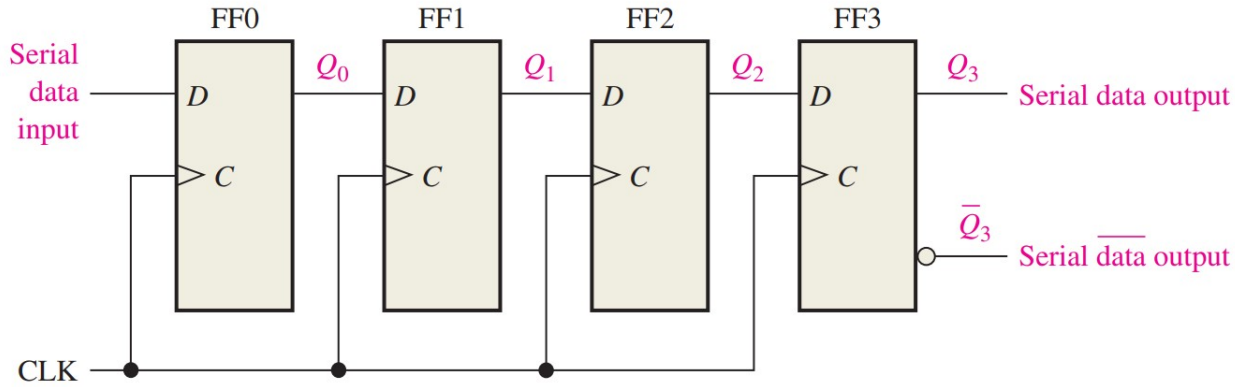
Edge triggered operation:



Shift registers:



Serial in/Serial out:



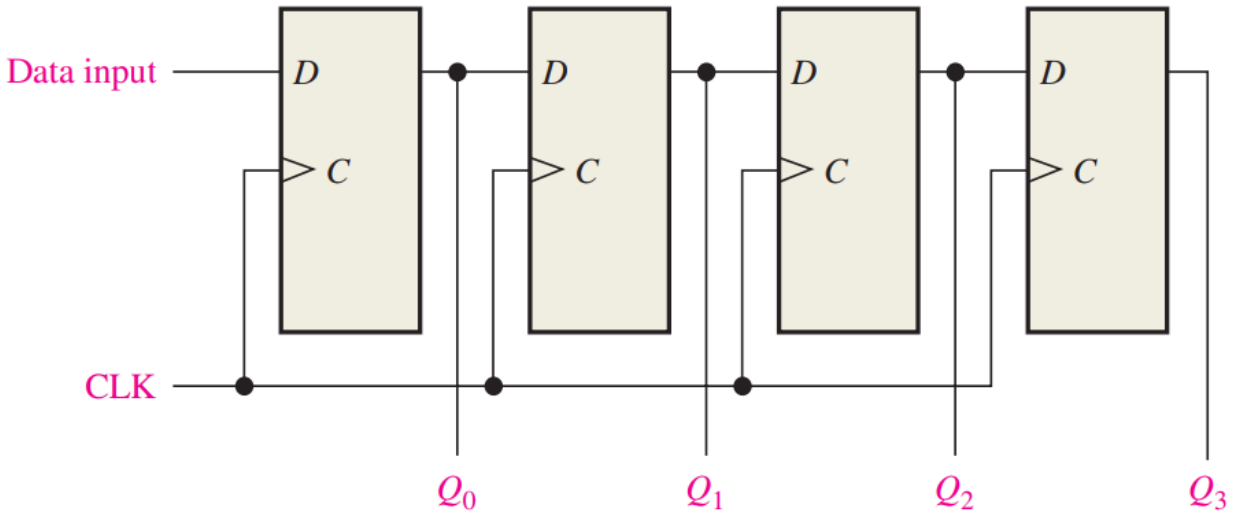
Shifting a 4-bit code into the shift register in Figure 8–3. Data bits are indicated by a beige screen.

CLK	FF0 (Q_0)	FF1 (Q_1)	FF2 (Q_2)	FF3 (Q_3)
Initial	0	0	0	0
1	**0**	0	0	0
2	**1**	**0**	0	0
3	**0**	**1**	**0**	0
4	**1**	**0**	**1**	**0**

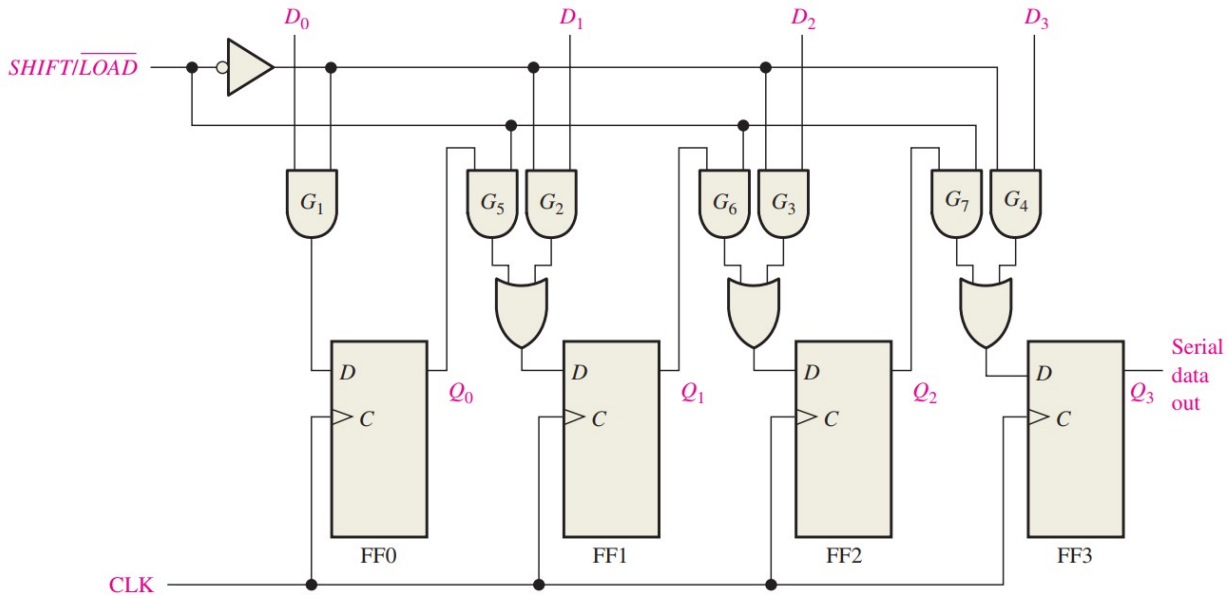
Shifting a 4-bit code out of the shift register in Figure 8–3. Data bits are indicated by a beige screen.

CLK	FF0 (Q_0)	FF1 (Q_1)	FF2 (Q_2)	FF3 (Q_3)
Initial	**1**	**0**	**1**	**0**
5	0	**1**	**0**	**1**
6	0	0	**1**	**0**
7	0	0	0	**1**
8	0	0	0	0

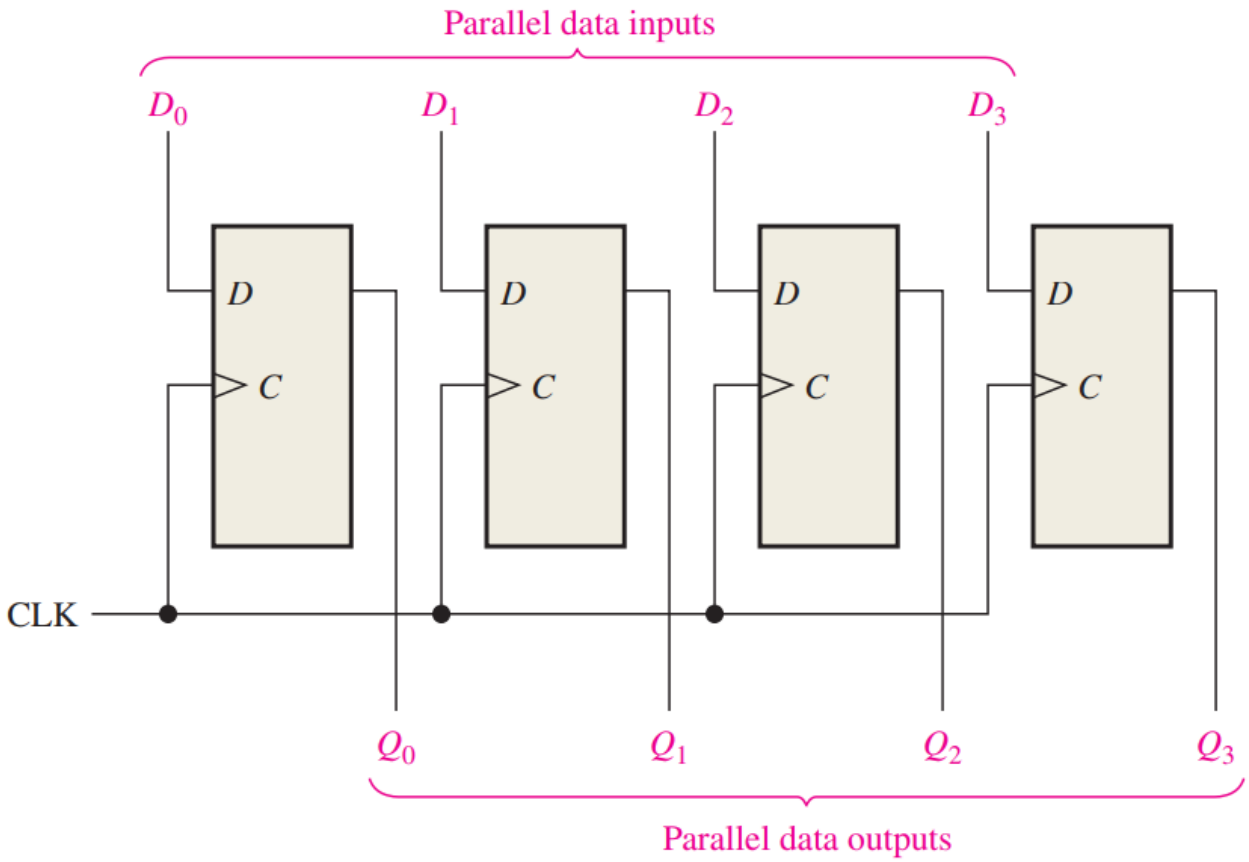
Serial in/Parallel out:



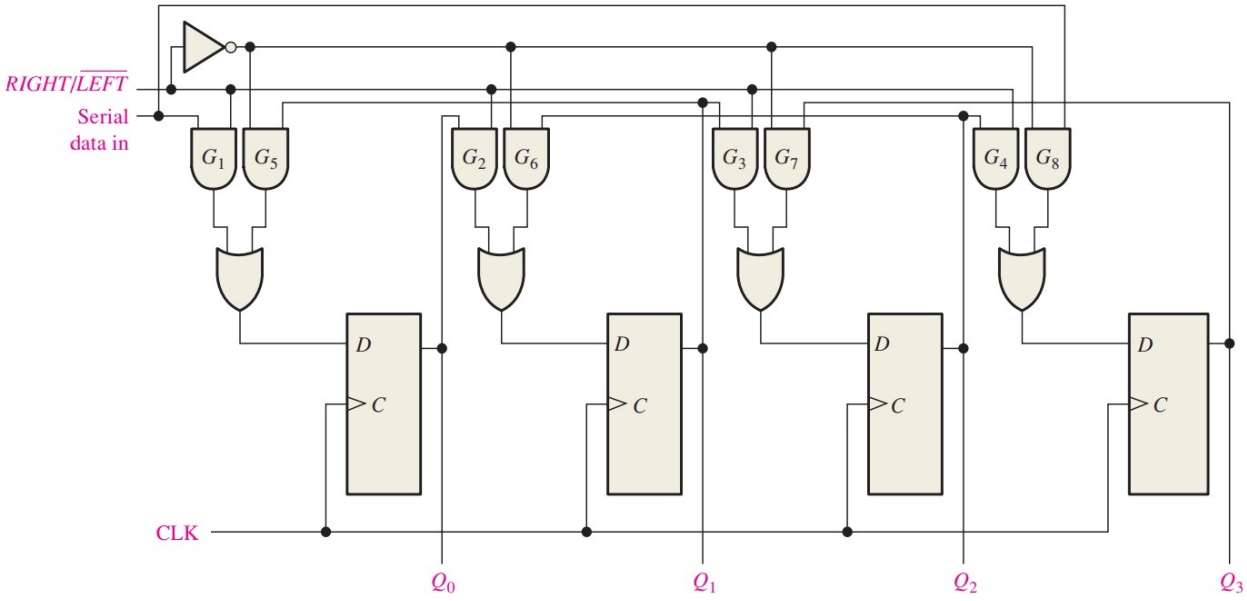
Parallel in/Serial out:



Parallel in/Parallel out:



Bi-directional shift registers:

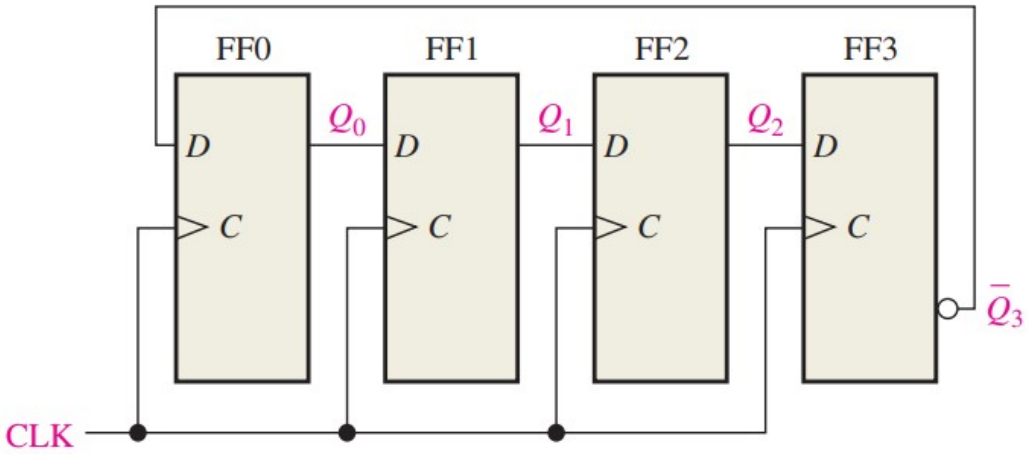


Shift register counters:

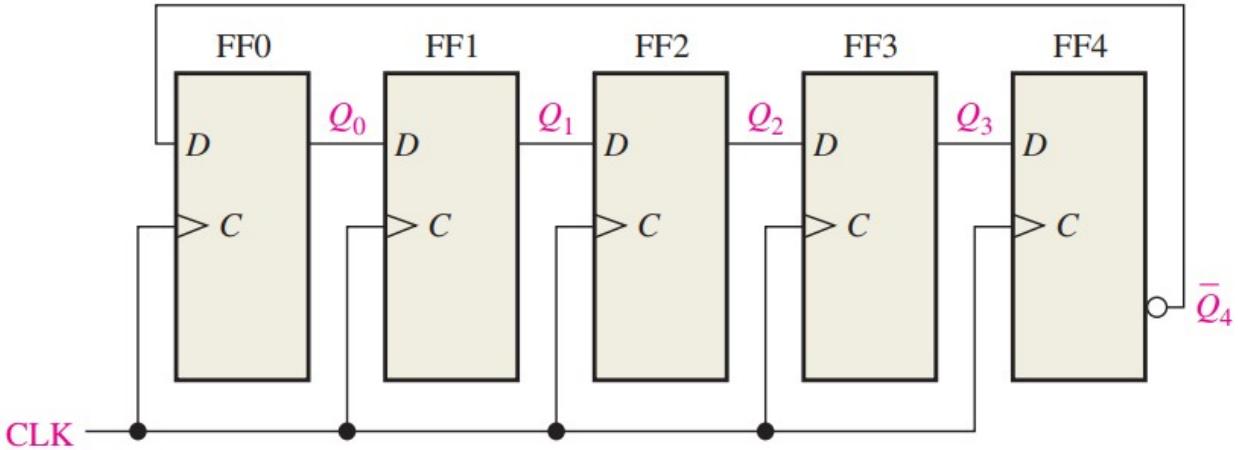
The Johnson counter:

Five-bit Johnson sequence.

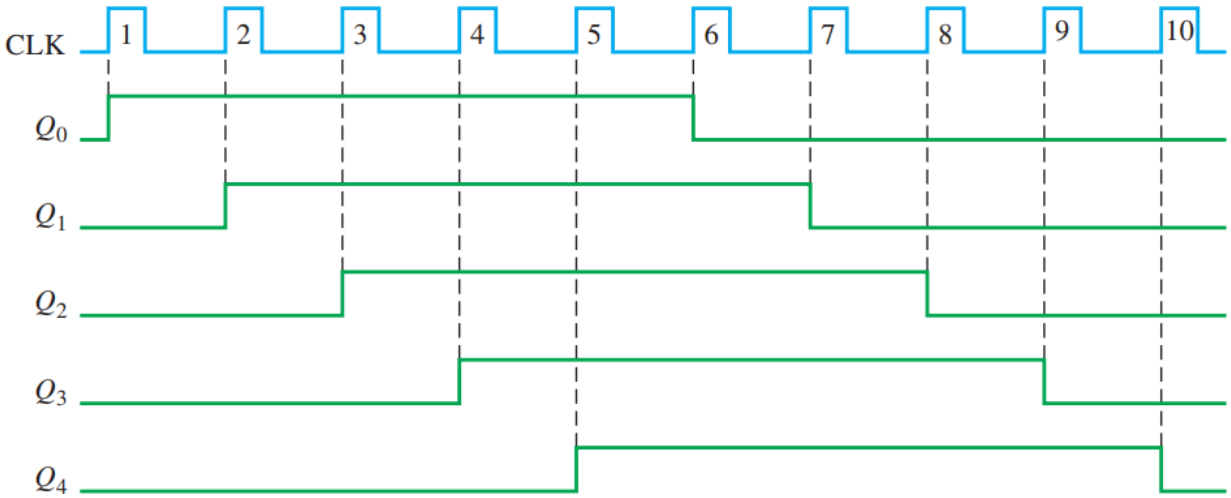
Clock Pulse	Q_0	Q_1	Q_2	Q_3	Q_4
0	0	0	0	0	0
1	1	0	0	0	0
2	1	1	0	0	0
3	1	1	1	0	0
4	1	1	1	1	0
5	1	1	1	1	1
6	0	1	1	1	1
7	0	0	1	1	1
8	0	0	0	1	1
9	0	0	0	0	1



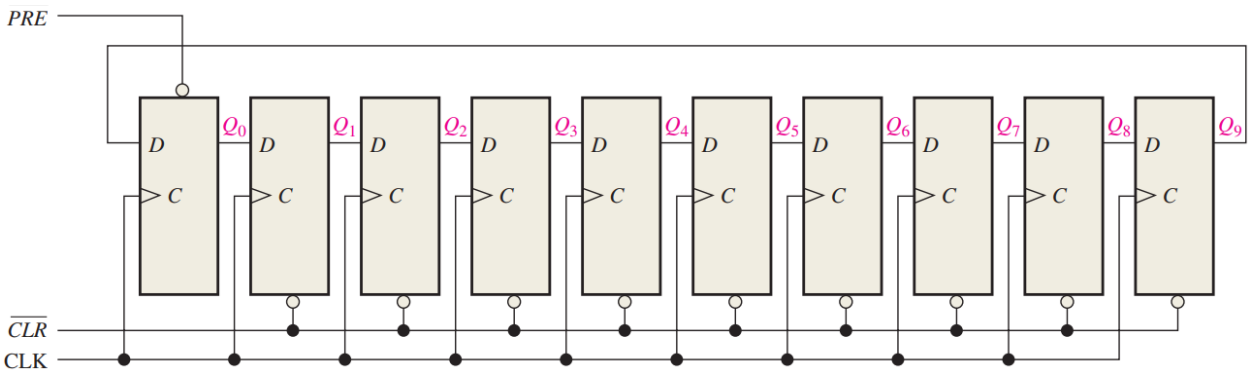
(a) Four-bit Johnson counter



(b) Five-bit Johnson counter



The ring counter:



Ten-bit ring counter sequence.

Clock Pulse	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	Q_9
0	1	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0
2	0	0	1	0	0	0	0	0	0	0
3	0	0	0	1	0	0	0	0	0	0
4	0	0	0	0	1	0	0	0	0	0
5	0	0	0	0	0	1	0	0	0	0
6	0	0	0	0	0	0	1	0	0	0
7	0	0	0	0	0	0	0	1	0	0
8	0	0	0	0	0	0	0	0	1	0
9	0	0	0	0	0	0	0	0	0	1