Digital Electronics

Dr Binu P Chacko

Associate Professor

Prajyoti Niketan College, Pudukad

Number System

- Decimal numbers (base 10): 0 to 9
- Combination of digits with different magnitude (weight) in each position
- $22 = 2 \times 10^1 + 2 \times 10^0 = 20 + 2$

? 1370, 10.658

- Binary numbers (base 2): 0, 1
- With *n* bits we can count from 0 to $2^n - 1$

Cont…

Binary to Decimal conversion

- Convert 1101_2 to decimal
- $1101_2 = 1x2^3+1x2^2+0x2^1+1x2^0$
  $= 8 + 4 + 0 + 1 = 13_{10}$
- Convert 10.101_2 to decimal
- $10.101_2 = 1x2^1+0x2^0+1x2^{-1}+0x2^{-2}+1x2^{-3}$
  $= 2 + 0 +0.5 + 0 + 0.125$
  $= 2.625_{10}$

? Convert to Decimal: 11011_2, 1011.110_2

Decimal Number	Binary Number
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

Decimal to Binary Conversion

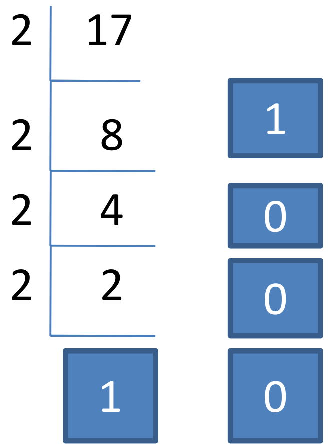
Sum of weights

- $17_{10} = 2^4 + 2^0 = 10001_2$
- $0.75 = 0.5 + 0.25 = 2^{-1} + 2^{-2} = 0.11$

Repeated multiplication by 2

- $0.75 \times 2 = 1.5$
- $0.5 \times 2 = 1.0$
- $0.75_{10} = 0.11_2$

Repeated division by 2



Octal Numbers

- 0, 1, 2, 3,4, 5, 6, 7

Octal to Decimal conversion

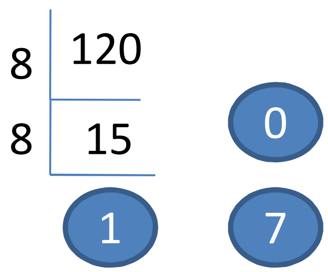
- $170_8 = 1 \times 8^2 + 7 \times 8^1 + 0 \times 8^0$

  $= 64 + 56 + 0 = 120_{10}$

Decimal to Octal conversion

- $120_{10} = 170_8$
- $0.41_8 = 4 \times 8^{-1} + 1 \times 8^{-2}$

  $= 4 \times 0.125 + 1 \times 0.015625 = 0.515625_{10}$



Octal to Binary conversion

- $53_8 = 101011_2$; $36_8 = 11110_2$

Binary to Octal conversion

- $11110_2 = 36_8$
- $10111001.1011_2 = 271.54_8$

Hexadecimal Numbers

- 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Binary to Hexadecimal conversion

- $1101101101_2 = 36D_{16}$
- $110111001.101_2 = 1B9.A_{16}$

Hexadecimal to Binary conversion

- $903_{16} = 100100000011_2$
- $AC.6_{16} = 10101100.011_2$

Hexadecimal to Decimal conversion

- $1A_{16} = 11010_2 = 26_{10}$
- $1A_{16} = 1 \times 16^1 + A \times 16^0$
  $= 16 + 10 \times 1 = 26_{10}$

Decimal to Hexadecimal conversion

- $500_{10} = 1F4_{16}$

16 | 500
16 | 31 → 4
1 → F

Digital Codes

- Combination of bits that represents numbers, letters, or symbols

8421 BCD code

- 38 -> 00111000
- 65.4 -> 01100101.0100
- 00010010 -> 12
- 10010111.1000 -> 97.8
- BCD addition

Decimal	8421 BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

```
  0100 0111        47
  0011 0010        32
  ---------        --
  0111 1001        79
```

```
       0110 0011        63
       0101 0111        57
       ---------       ---
       1011 1010       120
       0110 0110
       ---------
  0001 0010 0000
```

Gray Code

- Unweighted code

Decimal	Binary	Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101

Single bit change

Binary	Gray
10110	1
10110	11
10110	111
10110	1110
10110	11101

Gray to Binary	
11101	Gray
1	
10	
101	
1011	
10110	Binary

Excess-3 Code

Decimal	BCD	Excess-3
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

- Unweighted code obtained by adding 3 to each decimal digit and then converting the result to 4-bit binary

  25
  33
  ———
  58

0101 1000

- Self-complementing property: 1's complement of an Excess-3 number is the Excess-3 code for the 9's complement of the corresponding decimal number
- 2 $\xrightarrow{\text{Excess-3}}$ 0101 $\xrightarrow{\text{1's complement}}$ 1010 $\xleftarrow{\text{Excess-3}}$ 7 $\xrightarrow{\text{9's complement}}$ 2

Alphanumeric Codes

- ASCII: 7-bit code
- A (1000001), a (1100001), decimal digits (011 BCD), symbols, instructions
- EBCDIC: 8-bit code
- A (11000001), a (10000001), decimal digits (1111 BCD), symbols, commands

Boolean Algebra

Boolean Addition	Boolean Multiplication
0 + 0 = 0	0 . 0 = 0
0 + 1 = 1	0 . 1 = 0
1 + 0 = 1	1 . 0 = 0
1 + 1 = 1	1 . 1 = 1

Laws of Boolean Algebra

- Commutative law: A + B = B + A [1 + 0 = 0 + 1]
- AB = BA
- Associative law: A + (B + C) = (A + B) + C
- A(BC) = (AB)C [1(0.1) = 1(0) = 0] = [(1.0)1 = (0)1 = 0]
- Distributive law: A(B + C) = AB + AC

[1(0 + 1) = 1(1) = 1] = [1.0 + 1.1 = 0 + 1 = 1]

Boolean Rules

- $A + 0 = A$
- $A + 1 = 1$
- $A \cdot 0 = 0$
- $A \cdot 1 = A$
- $A + A = A$
- $A + \overline{A} = 1$
- $A \cdot A = A$
- $A \cdot \overline{A} = 0$
- $\overline{\overline{A}} = A$
- $A + AB = A$

$$
\begin{aligned}
A + AB &= A(1 + B) \\
&= A \cdot 1 \\
&= A
\end{aligned}
$$

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

$$
\begin{aligned}
A + \overline{A}B &= (A + AB) + \overline{A}B \\
&= (AA + AB) + \overline{A}B \\
&= AA + AB + A\overline{A} + \overline{A}B \\
&= (A + \overline{A})(A + B) \\
&= 1 \cdot (A + B) \\
&= A + B
\end{aligned}
$$

$(A + B)(A + C) = A + BC$

$$
\begin{aligned}
(A + B)(A + C) &= AA + AC + AB + BC \\
&= A + AC + AB + BC \\
&= A(1 + C) + AB + BC \\
&= A \cdot 1 + AB + BC \\
&= A + AB + BC \\
&= A(1 + B) + BC \\
&= A \cdot 1 + BC \\
&= A + BC
\end{aligned}
$$

Demorgan's Theorems

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

- Complement of a product is equal to the sum of the complements

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

- Complement of a sum is equal to the product of the complements

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

$$\overline{\overline{\left(\overline{A} + B\right)} + \overline{CD}} = \overline{\overline{\left(\overline{A} + B\right)}}.\overline{\overline{CD}}$$

$$= \left(\overline{A} + B\right)CD$$

Duality Principle

- If we have postulates or theorems of Boolean Algebra for one type of operation then that operation can be converted into another type of operation (i.e., **AND can be converted to OR and vice-versa**) just by interchanging **'0 with 1'**, **'1 with 0'**, **'(+) sign with (.) sign'** and **'(.) sign with (+) sign'**

Expression	Dual
0.1 = 0	0 + 1 = 1
A.0 = 0	A + 1 = 1
(A + B)' = A' . B'	(AB)' = A' + B'
A.A' = 0	A + A' = 1

Boolean Functions

- $F = xy + xy'z + x'yz$
- $F = 1$, if $x = 1$ & $y = 1$; OR $x = 1$, $y = 0$, $z = 1$; OR $x = 0$, $y = 1$, $z = 1$
- Otherwise, $F = 0$

Canonical Form

- Boolean functions expressed as sum of minterms (products) or product of maxterms (sums)
- Minterm: functions which result in 1
- Maxterm: functions which result in 0

X	Y	Z	Minterm=1	Maxterm=0
0	0	0	X'.Y'.Z'	X+Y+Z
0	0	1	X'.Y'.Z	X+Y+Z'
0	1	0	X'.Y.Z'	X+Y'+Z
0	1	1	X'.Y.Z	X+Y'+Z'
1	0	0	X.Y'.Z'	X'+Y+Z
1	0	1	X.Y'.Z	X'+Y+Z'
1	1	0	X.Y.Z'	X'+Y'+Z
1	1	1	X.Y.Z	X'+Y'+Z'

SOP & POS

- Implement the following with logic gates
- X = AB + CD + EF
- X = (A + B)(C + D)(E + F)

$$X = \overline{A}BC + A\overline{B}C + ABC + \overline{A}B\overline{C}$$

- Simplify the following using Boolean algebra
- AB + A(B + C) + B(B + C)

$$[A\overline{B}(C + BD) + \overline{A}\,\overline{B}]C$$

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

$$(\overline{A} + B)C + ABC$$

$$A\overline{B}C(BD + CDE) + A\overline{C}$$

Karnaugh Map

- Minimise the following to minimum SOP:

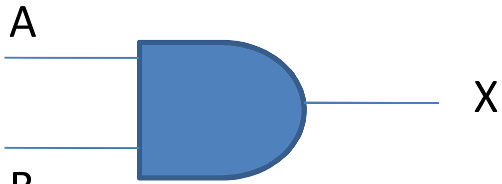
$$X = A\bar{B}C + \bar{A}BC + \bar{A}\bar{B}C + \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C}$$

$$X = \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}B\bar{C}\bar{D} + AB\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}D + \bar{A}\bar{B}CD + A\bar{B}CD + \bar{A}\bar{B}C\bar{D} + \bar{A}BC\bar{D} + ABC\bar{D} + A\bar{B}C\bar{D}$$

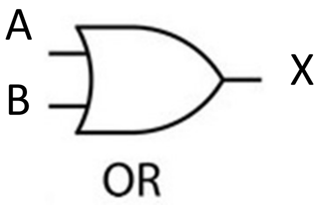
$$X = \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}C\bar{D} + A\bar{B}\bar{C}\bar{D} + \bar{A}CD + A\bar{B}C\bar{D}$$

Logic Gates

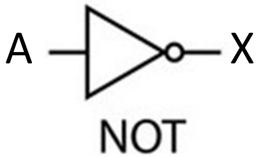
- AND



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

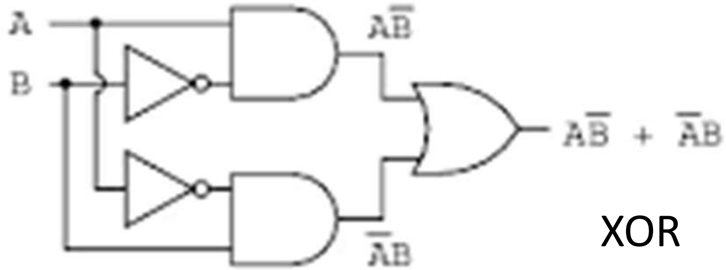


A	B	X
0	0	0
0	1	1
1	0	1
1	1	1



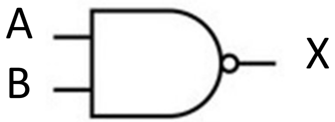
A	X
0	1
1	0

Cont…



XOR

- Universal gates



NAND

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

NOR

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

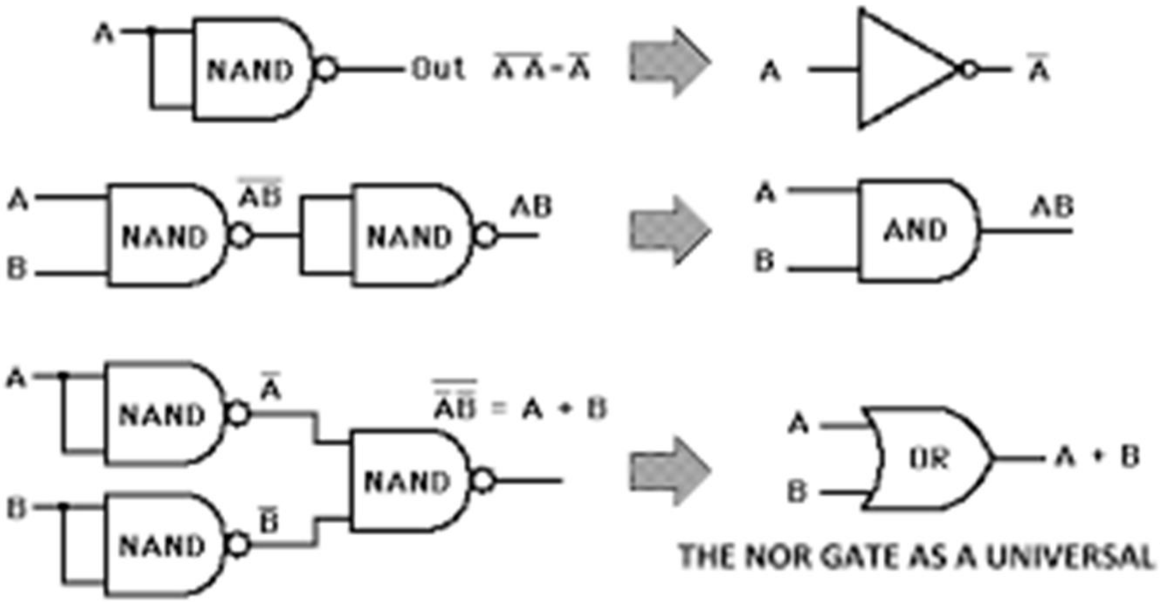
XOR

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

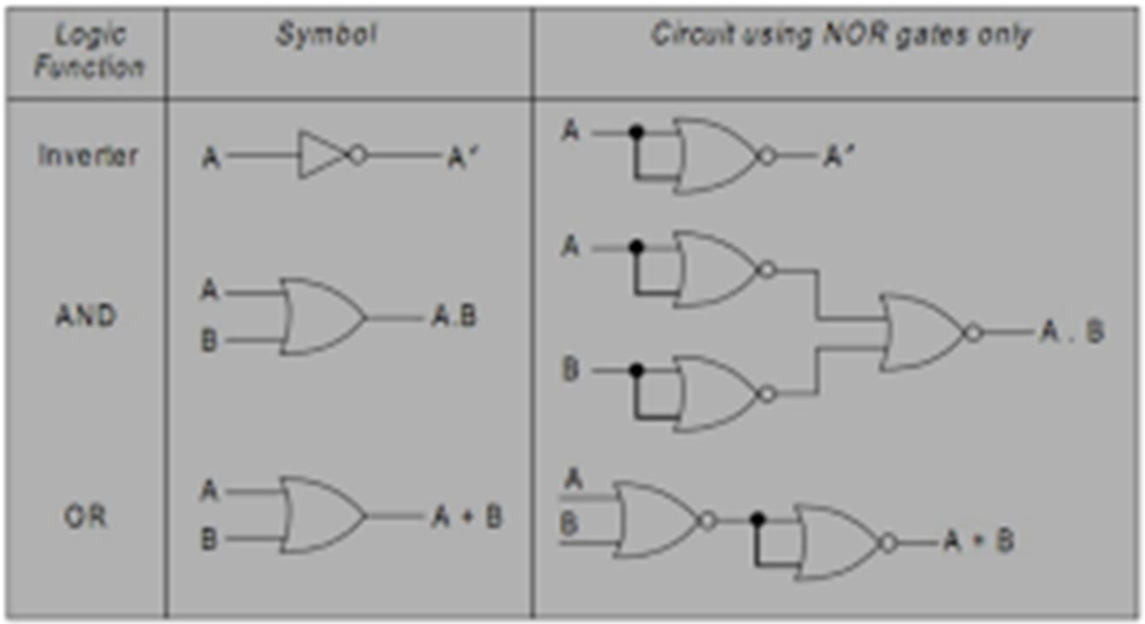
XNOR

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

Universal Properties

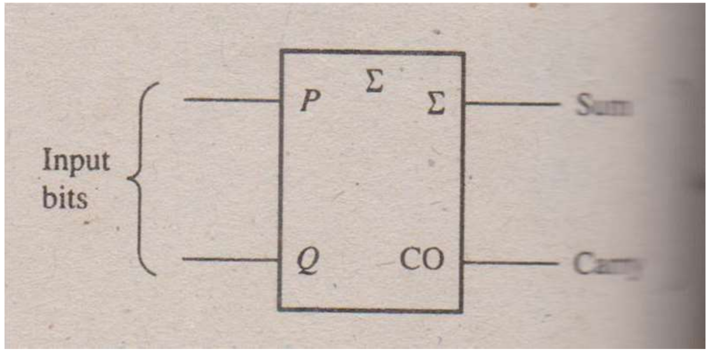


THE NOR GATE AS A UNIVERSAL GATE:

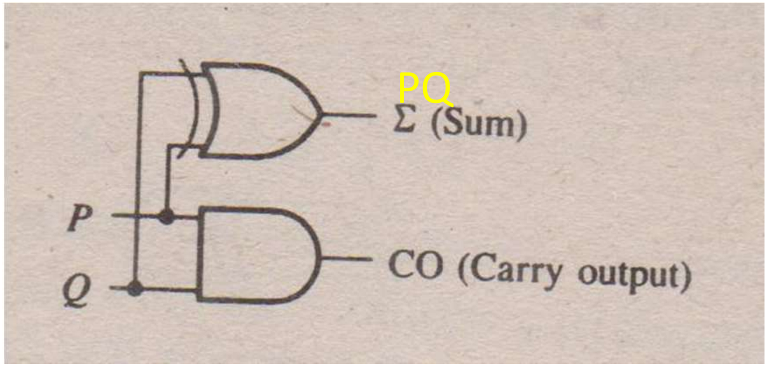


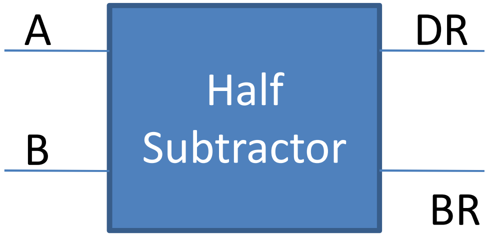
=P⊕ Q

Half Adder/Subtractor

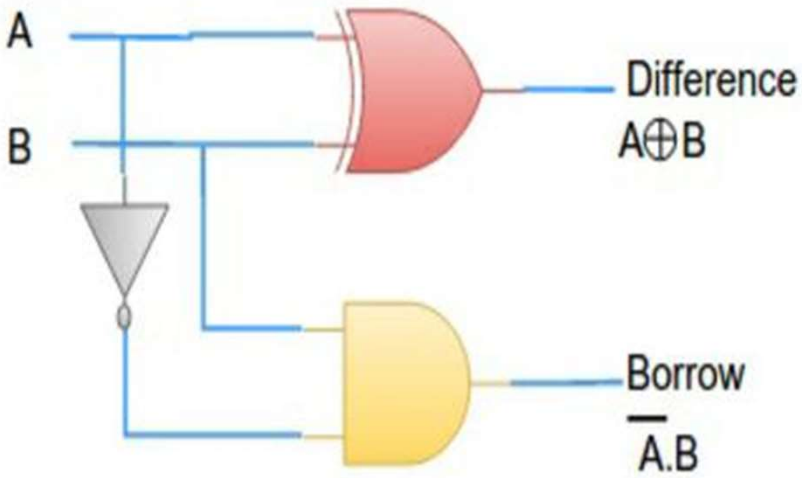


P	Q	CO	Σ
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

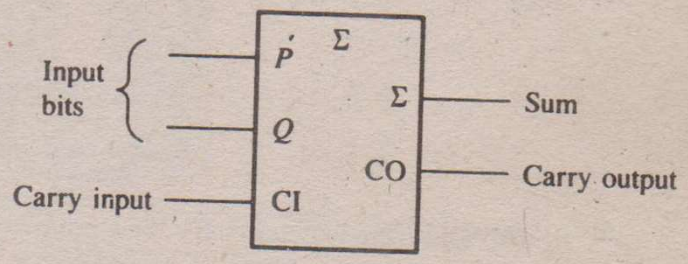




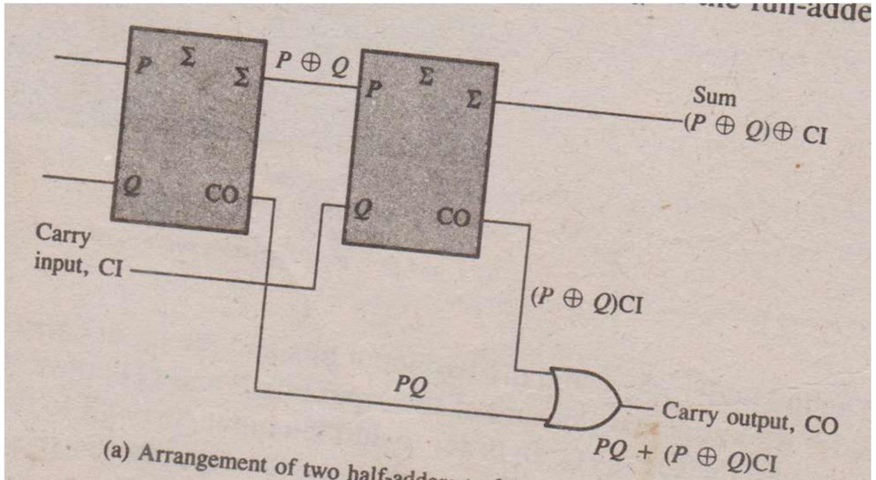
A	B	BR	DR
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0



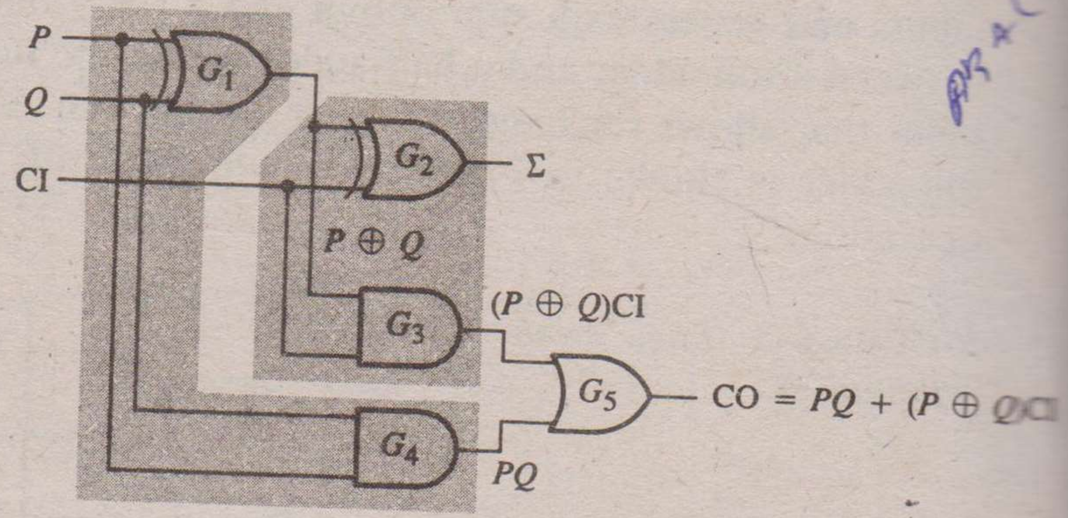
Full Adder



P	Q	CI	CO	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



(a) Arrangement of two half-adders to form a full add...

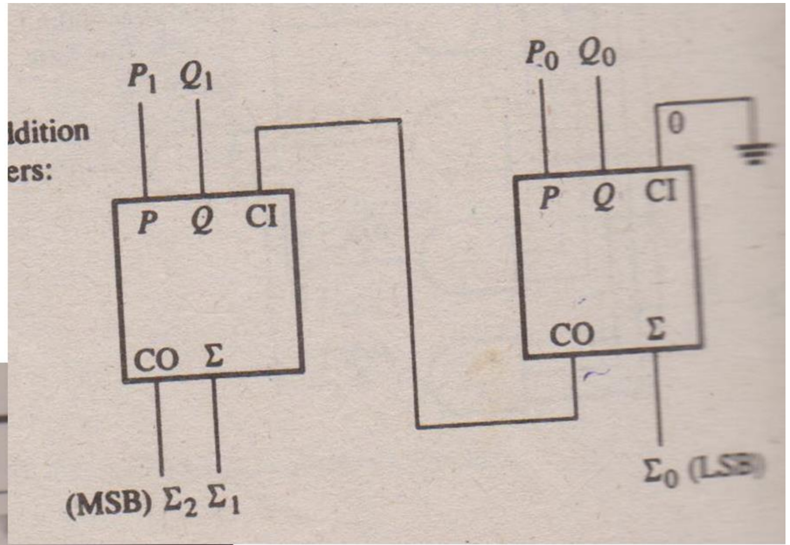


Parallel Binary Adder

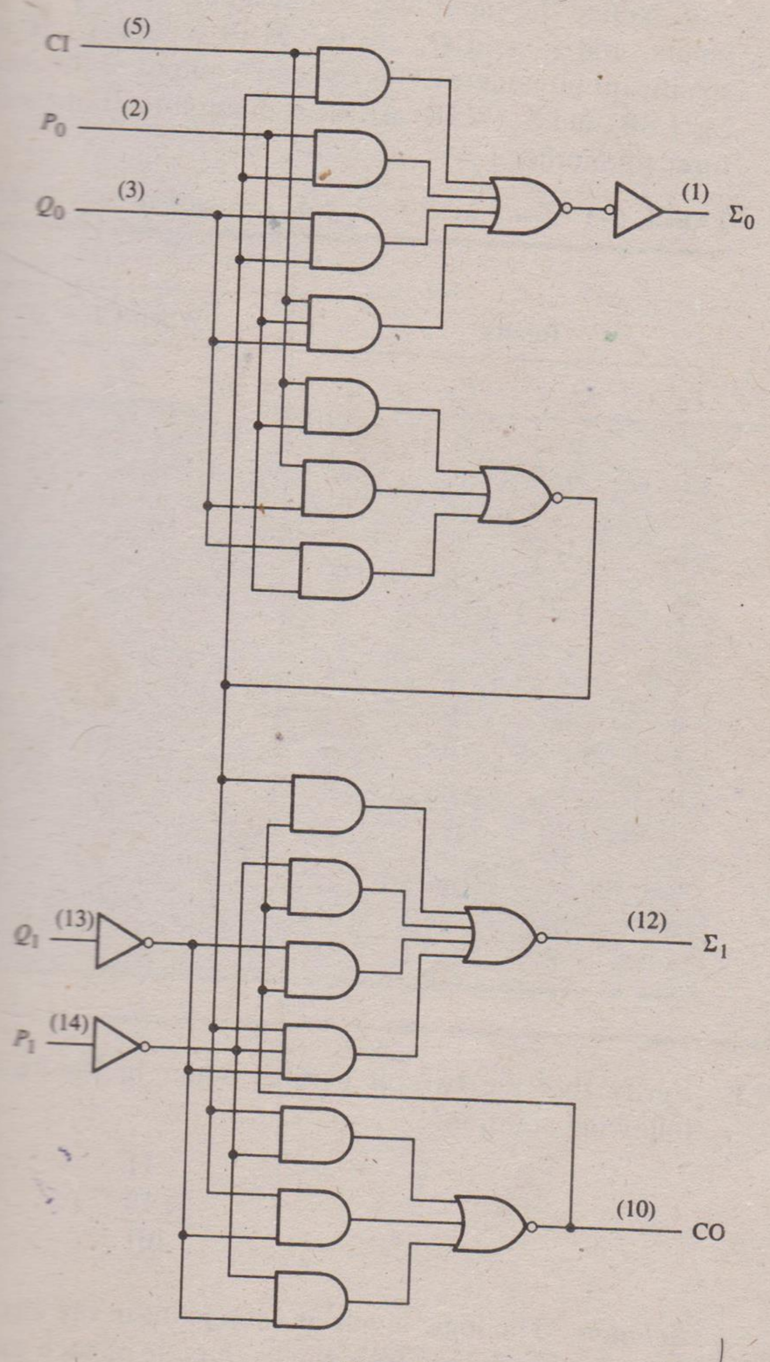
turer to another.)

TABLE 6-3 *Truth table for a 7482 two-bit binary full-adder.*

Inputs				Outputs: When CI = 0		
P_0	Q_0	P_1	Q_1	Σ_0	Σ_1	CO
0	0	0	0	0	0	0
1	0	0	0	1	0	0
0	1	0	0	1	0	0
1	1	0	0	0	1	0
0	0	1	0	0	1	0
1	0	1	0	1	1	0
0	1	1	0	1	1	0
1	1	1	0	0	0	1
0	0	0	1	0	1	0
1	0	0	1	1	1	0
0	1	0	1	1	1	0
1	1	0	1	0	0	1
0	0	1	1	0	0	1
1	0	1	1	1	0	1
0	1	1	1	1	0	1
1	1	1	1	0	1	1

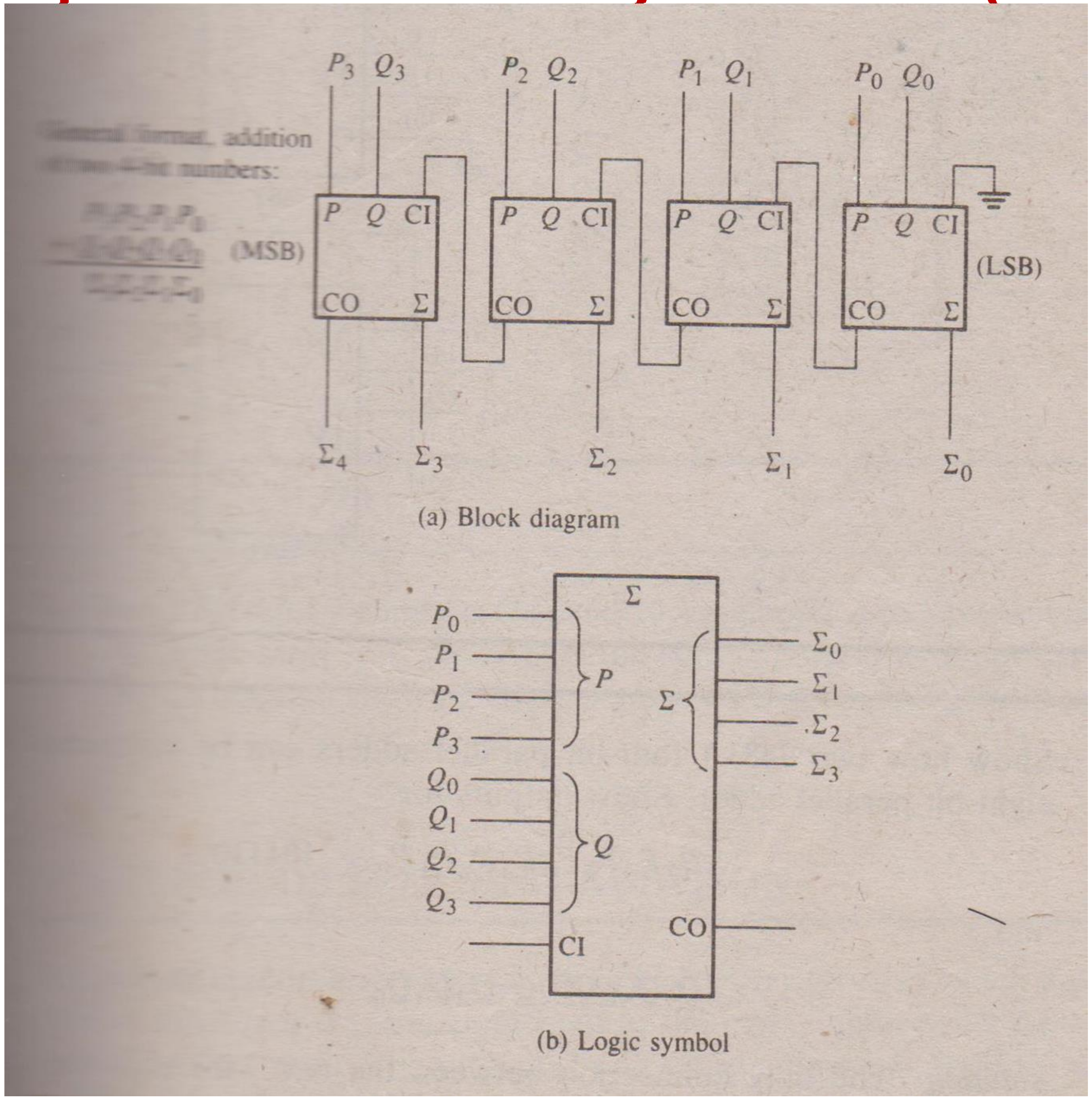


Block diagram of
2-bit binary full adder



(b) Logic diagram

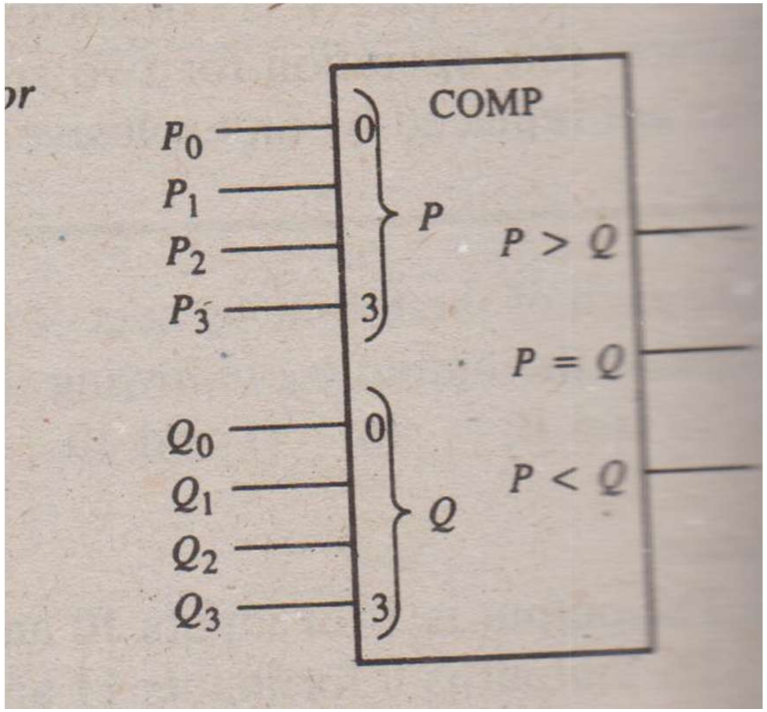
4-bit parallel binary adder (7483A)

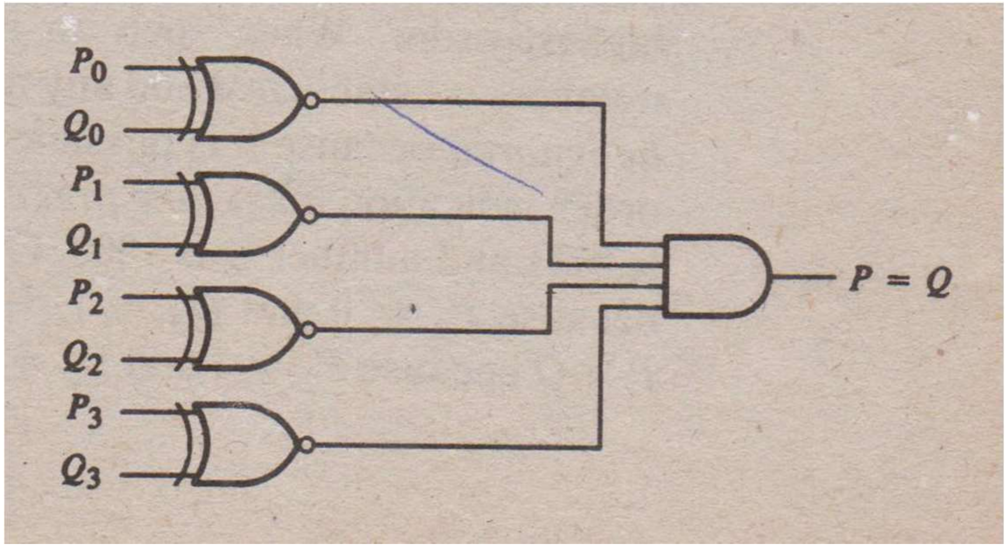


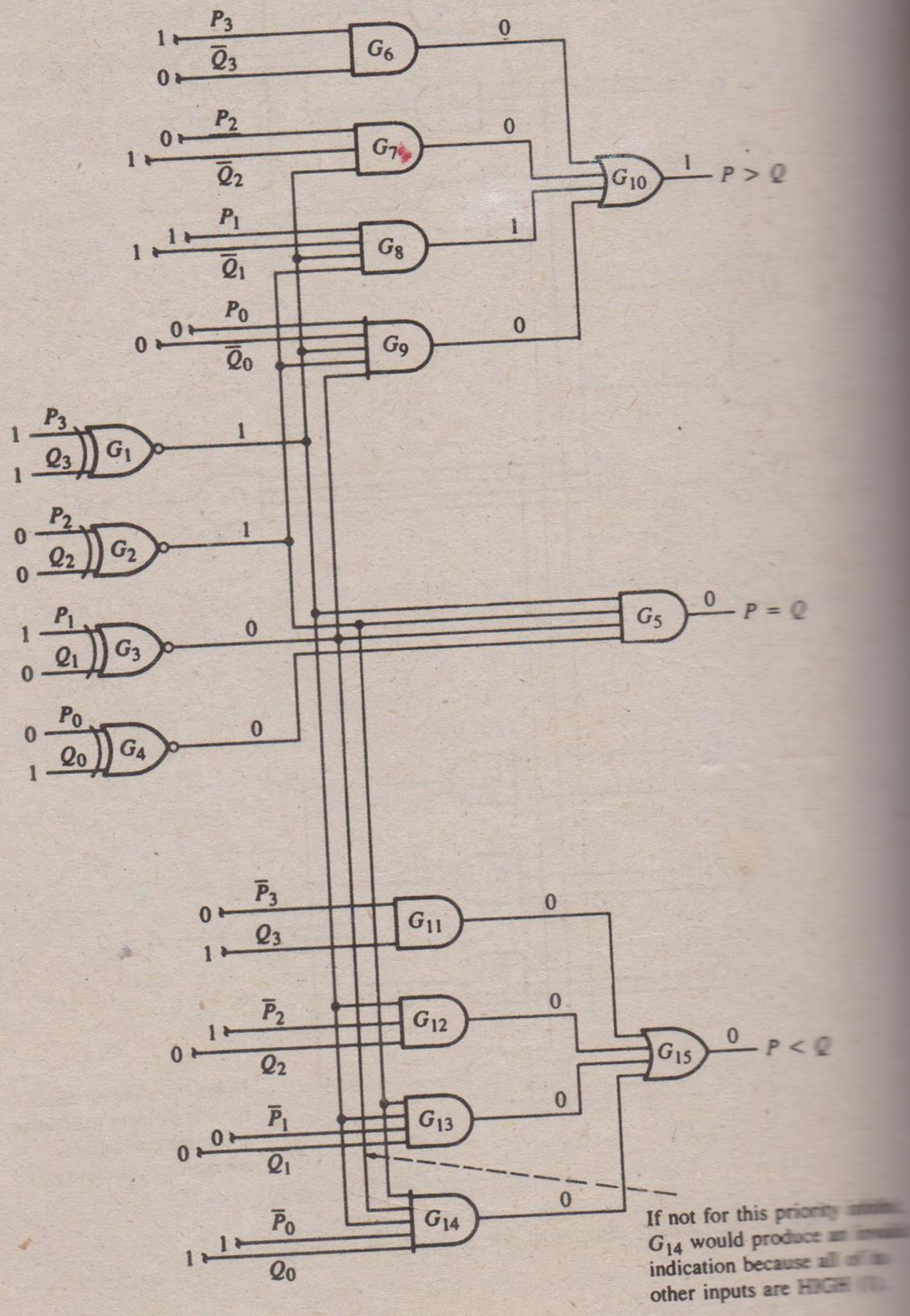
(a) Block diagram

(b) Logic symbol

Comparator


COMP
P
Q
P > Q
P = Q
P < Q
P0
P1
P2
P3
Q0
Q1
Q2
Q3
0
3
0
3


P0
Q0
P1
Q1
P2
Q2
P3
Q3
P = Q



G1
G2
G3
G4
G5
G6
G7
G8
G9
G10
G11
G12
G13
G14
G15
P > Q
P = Q
P < Q
If not for this priority
G14 would produce an
indication because all
other inputs are

Decoder

- To detect the presence of a specified combination of bits

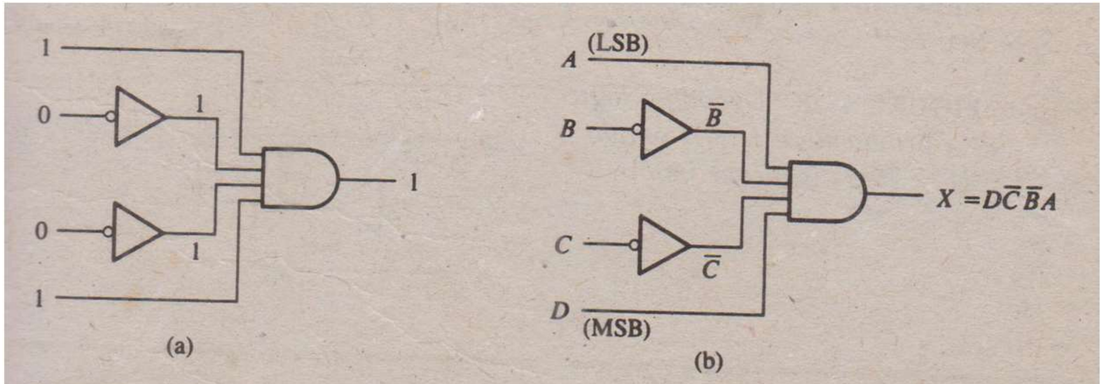
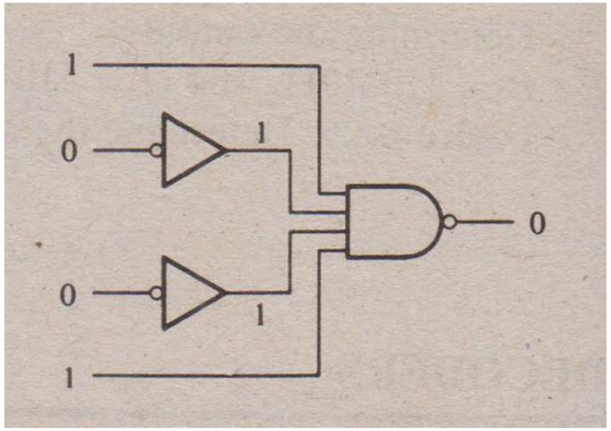


FIGURE 6–26 *Decoding logic for 1001 with an active-HIGH output.*



4-bit Binary Decoder

- 4-line-to-16-line decoder (74154)
- 16 decoding gates are required

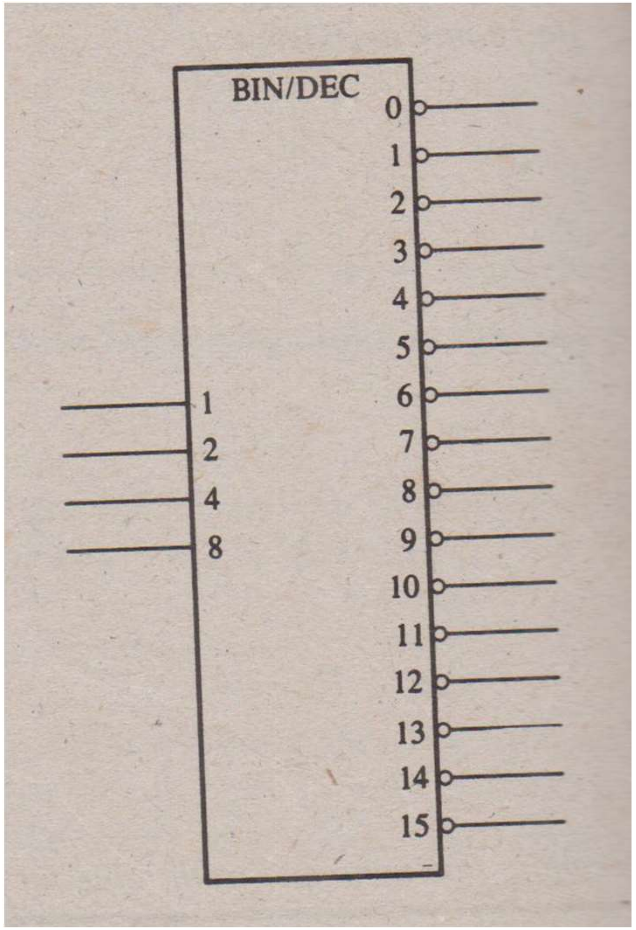
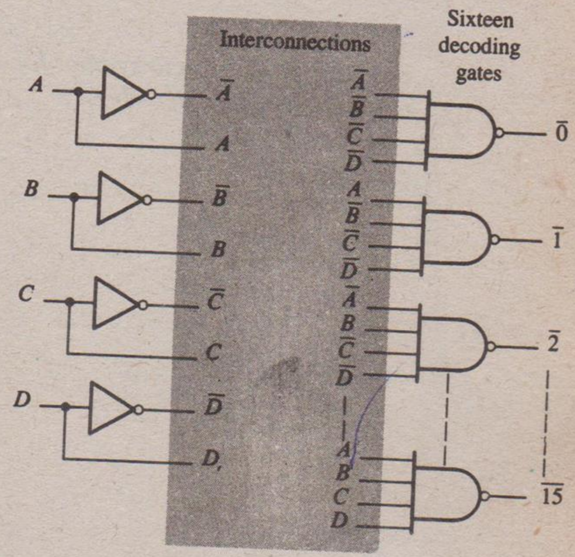


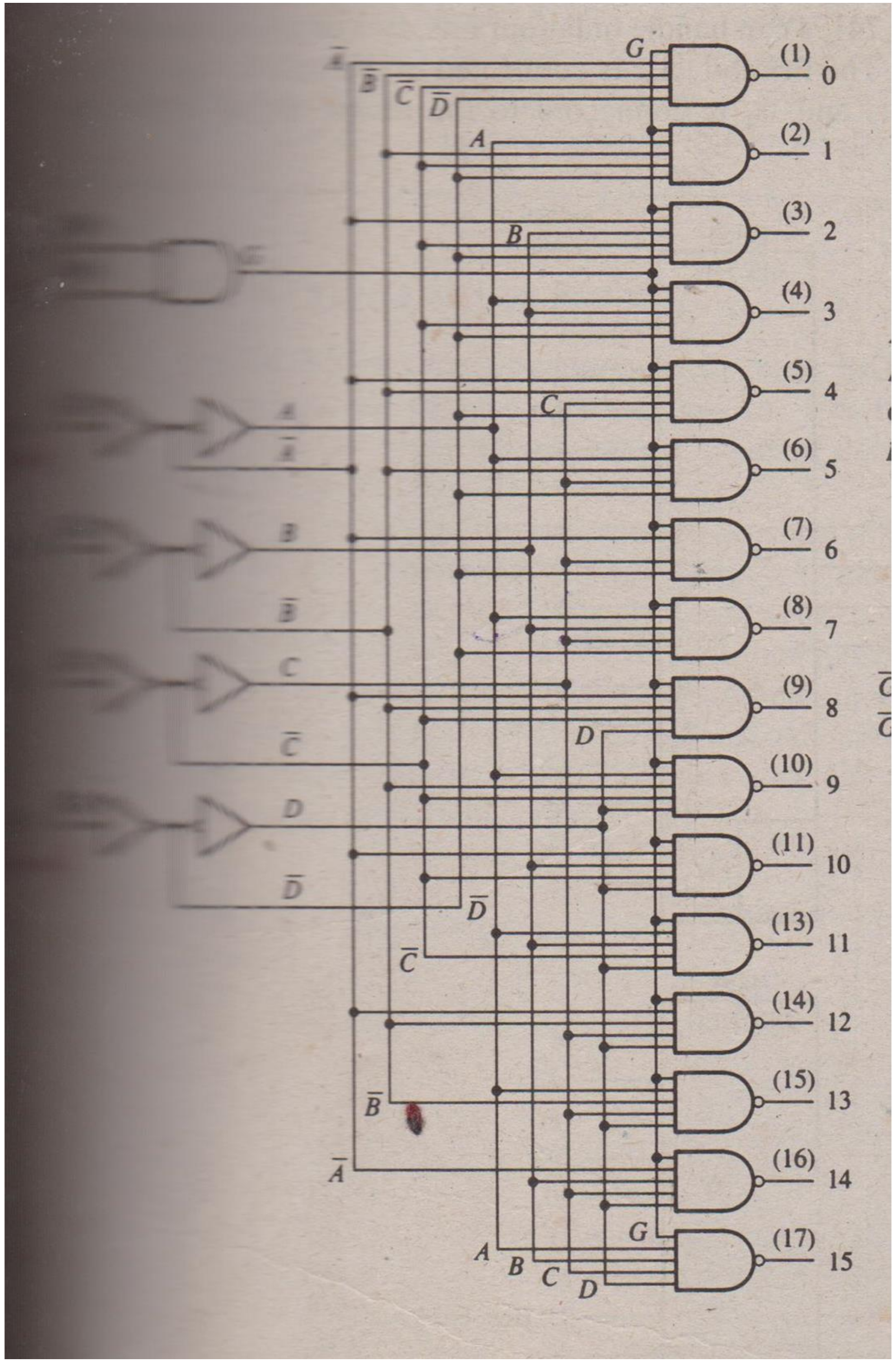
FIGURE 6–29 *Logic for a 4-line-to-16-line decoder.*



[illegible] *Decoding functions and truth table for a 4-line-to-16-line decoder.*

Binary Inputs				Logic Function	Outputs															
D	C	B	A		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
[illegible]	0	0	0	$\overline{D}\overline{C}\overline{B}\overline{A}$	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
[illegible]	0	0	1	$\overline{D}\overline{C}\overline{B}A$	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
[illegible]	0	1	0	$\overline{D}\overline{C}B\overline{A}$	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
[illegible]	0	1	1	$\overline{D}\overline{C}BA$	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
[illegible]	1	0	0	$\overline{D}C\overline{B}\overline{A}$	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
[illegible]	1	0	1	$\overline{D}C\overline{B}A$	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
[illegible]	1	1	0	$\overline{D}CB\overline{A}$	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
[illegible]	1	1	1	$\overline{D}CBA$	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
[illegible]	0	0	0	$D\overline{C}\overline{B}\overline{A}$	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
[illegible]	0	0	1	$D\overline{C}\overline{B}A$	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
[illegible]	0	1	0	$D\overline{C}B\overline{A}$	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
[illegible]	0	1	1	$D\overline{C}BA$	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
[illegible]	1	0	0	$DC\overline{B}\overline{A}$	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
[illegible]	1	0	1	$DC\overline{B}A$	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
[illegible]	1	1	0	$DCB\overline{A}$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
[illegible]	1	1	1	$DCBA$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

74154



(1) 0
(2) 1
(3) 2
(4) 3
(5) 4
(6) 5
(7) 6
(8) 7
(9) 8
(10) 9
(11) 10
(13) 11
(14) 12
(15) 13
(16) 14
(17) 15
G
A
B
C
D

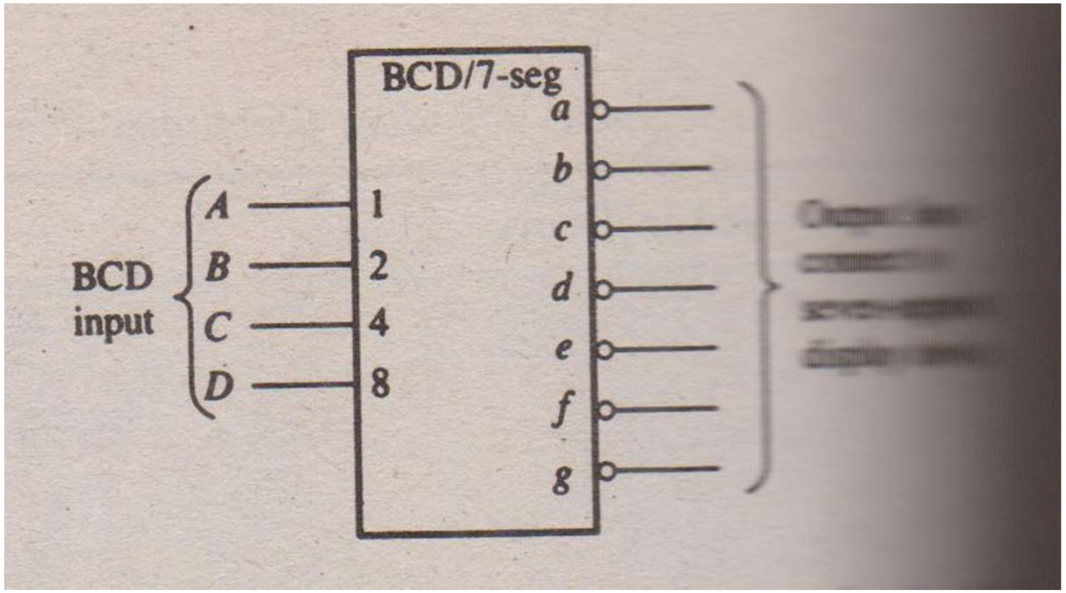
BCD-to-Decimal Decoder

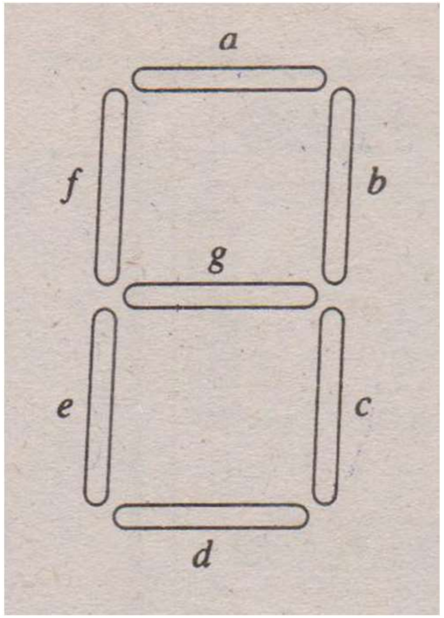
- 4-line-to-10-line decoder (7442A) converts each BCD code into one of 10 possible digit indications

Decimal Digit	BCD Code				Logic Function
	D	C	B	A	
0	0	0	0	0	$\bar{D}\bar{C}\bar{B}\bar{A}$
1	0	0	0	1	$\bar{D}\bar{C}\bar{B}A$
2	0	0	1	0	$\bar{D}\bar{C}B\bar{A}$
3	0	0	1	1	$\bar{D}\bar{C}BA$
4	0	1	0	0	$\bar{D}C\bar{B}\bar{A}$
5	0	1	0	1	$\bar{D}C\bar{B}A$
6	0	1	1	0	$\bar{D}CB\bar{A}$
7	0	1	1	1	$\bar{D}CBA$
8	1	0	0	0	$D\bar{C}\bar{B}\bar{A}$
9	1	0	0	1	$D\bar{C}\bar{B}A$

BCD-to-Seven Segment Decoder

- Accepts BCD code as inputs and produce decimal readout on seven segment display devices

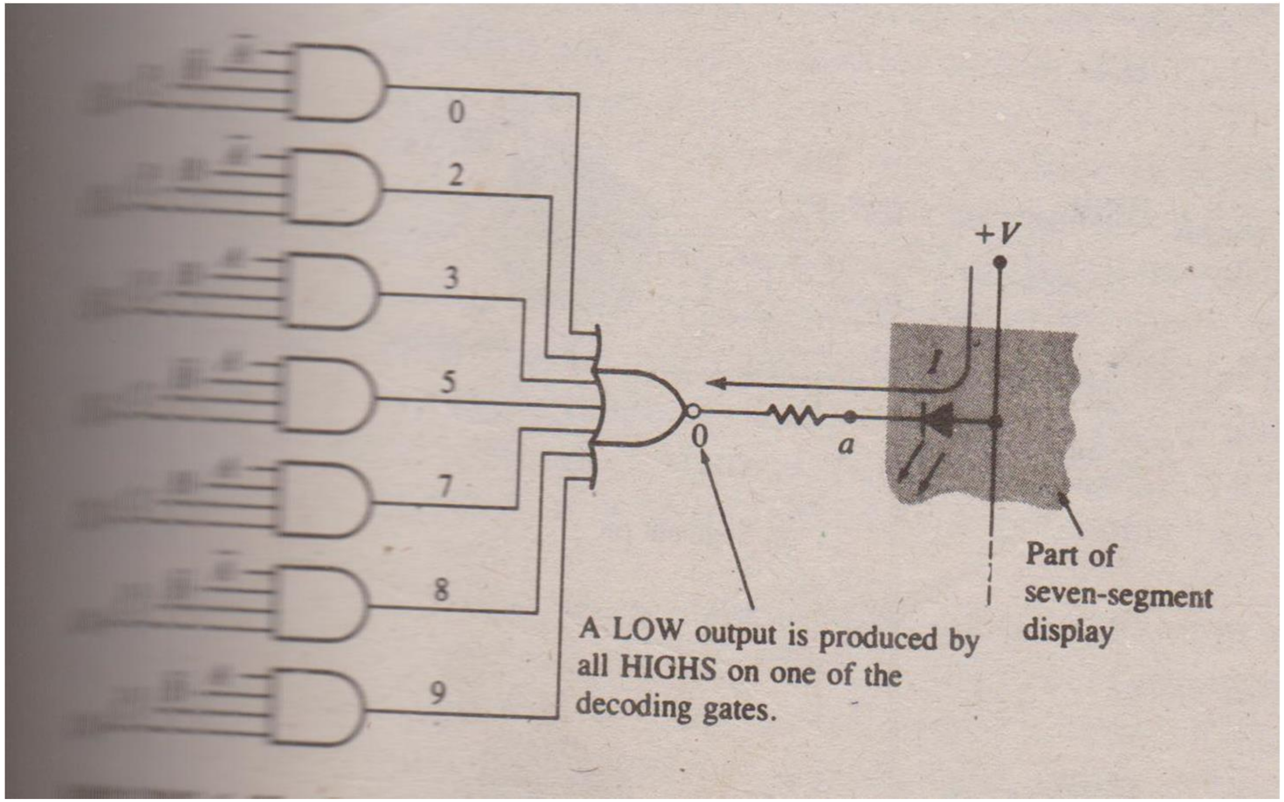




Digit	Segments Activated
0	*a, b, c, d, e, f*
1	*b, c*
2	*a, b, g, e, d*
3	*a, b, c, d, g*
4	*b, c, f, g*
5	*a, c, d, f, g*
6	*c, d, e, f, g*
7	*a, b, c*
8	*a, b, c, d, e, f, g*
9	*a, b, c, f, g*

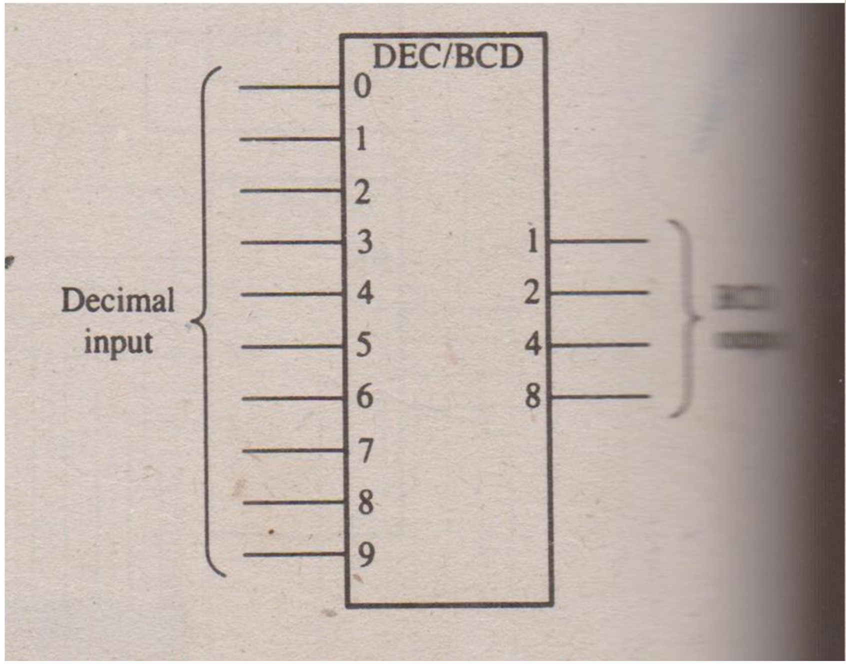
...segment decoding functions.

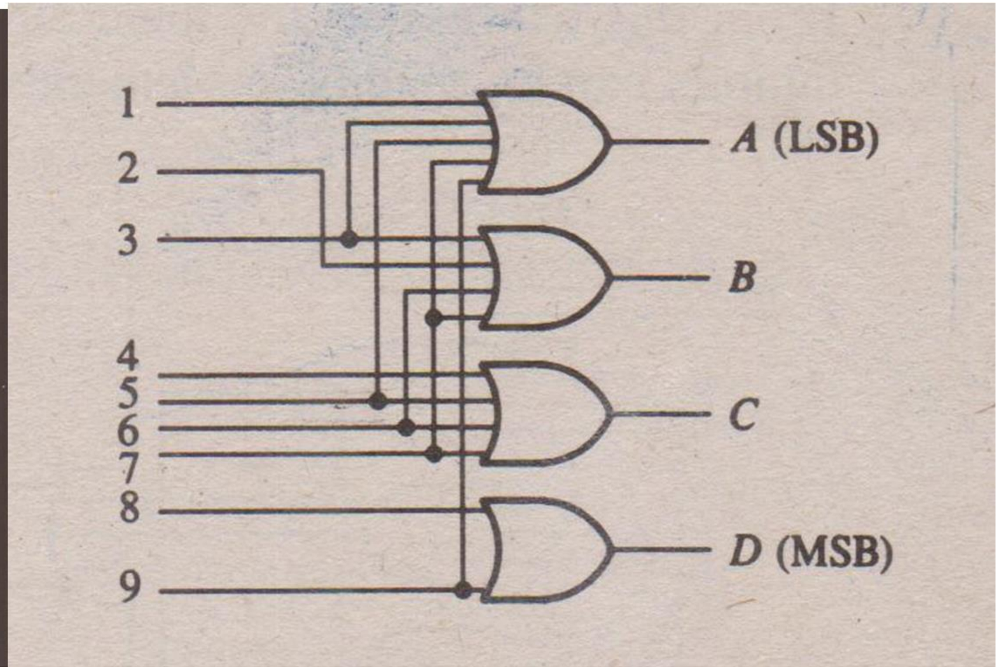
[illegible]	Used in Digits	Logic Function
[illegible]	0, 2, 3, 5, 7, 8, 9	$\bar{D}\bar{C}\bar{B}\bar{A} + \bar{D}\bar{C}B\bar{A} + \bar{D}\bar{C}BA + \bar{D}C\bar{B}A + \bar{D}CBA + D\bar{C}\bar{B}\bar{A} + D\bar{C}\bar{B}A$
[illegible]	0, 1, 2, 3, 4, 7, 8, 9	$\bar{D}\bar{C}\bar{B}\bar{A} + \bar{D}\bar{C}\bar{B}A + \bar{D}\bar{C}B\bar{A} + \bar{D}\bar{C}BA + \bar{D}C\bar{B}\bar{A} + \bar{D}CBA + D\bar{C}\bar{B}\bar{A} + D\bar{C}\bar{B}A$
[illegible]	0, 1, 3, 4, 5, 6, 7, 8, 9	$\bar{D}\bar{C}\bar{B}\bar{A} + \bar{D}\bar{C}\bar{B}A + \bar{D}\bar{C}BA + \bar{D}C\bar{B}\bar{A} + \bar{D}C\bar{B}A + \bar{D}CB\bar{A} + \bar{D}CBA + D\bar{C}\bar{B}\bar{A} + D\bar{C}\bar{B}A$
[illegible]	0, 2, 3, 5, 6, 8	$\bar{D}\bar{C}\bar{B}\bar{A} + \bar{D}\bar{C}B\bar{A} + \bar{D}\bar{C}BA + \bar{D}C\bar{B}A + \bar{D}CB\bar{A} + D\bar{C}\bar{B}\bar{A}$
[illegible]	0, 2, 6, 8	$\bar{D}\bar{C}\bar{B}\bar{A} + \bar{D}\bar{C}B\bar{A} + \bar{D}CB\bar{A} + D\bar{C}\bar{B}\bar{A}$
[illegible]	0, 4, 5, 6, 8, 9	$\bar{D}\bar{C}\bar{B}\bar{A} + \bar{D}C\bar{B}\bar{A} + \bar{D}C\bar{B}A + \bar{D}CB\bar{A} + D\bar{C}\bar{B}\bar{A} + D\bar{C}\bar{B}A$
[illegible]	2, 3, 4, 5, 6, 8, 9	$\bar{D}\bar{C}B\bar{A} + \bar{D}\bar{C}BA + \bar{D}C\bar{B}\bar{A} + \bar{D}C\bar{B}A + \bar{D}CB\bar{A} + D\bar{C}\bar{B}\bar{A} + D\bar{C}\bar{B}A$


0
2
3
5
7
8
9
+V
I
0
a
A LOW output is produced by all HIGHS on one of the decoding gates.
Part of seven-segment display

Encoder

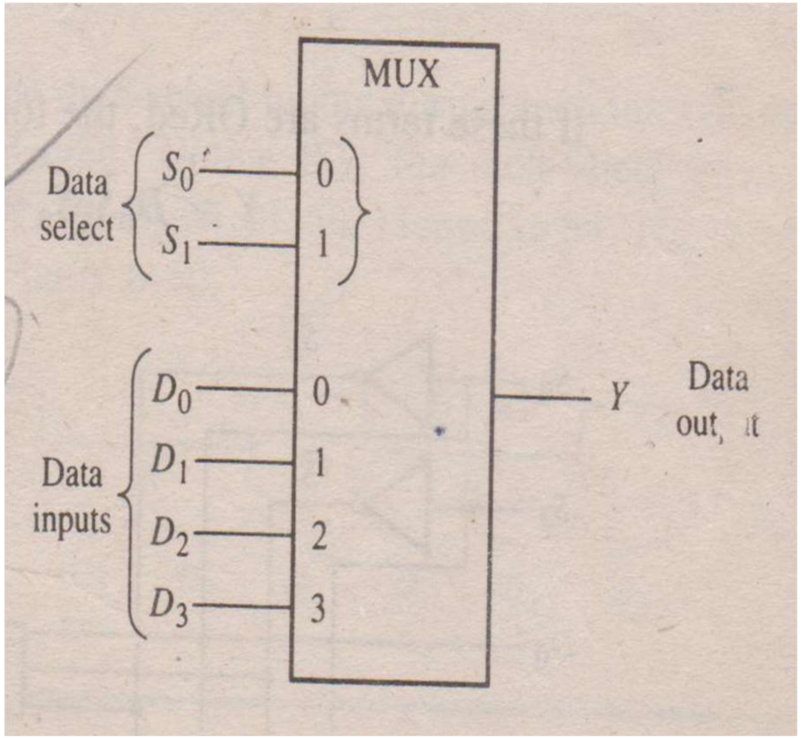
- The process of converting symbols/numbers to a coded format is called encoding
- Decimal-to-BCD encoder accepts a decimal digit as input and converts it to the BCD code



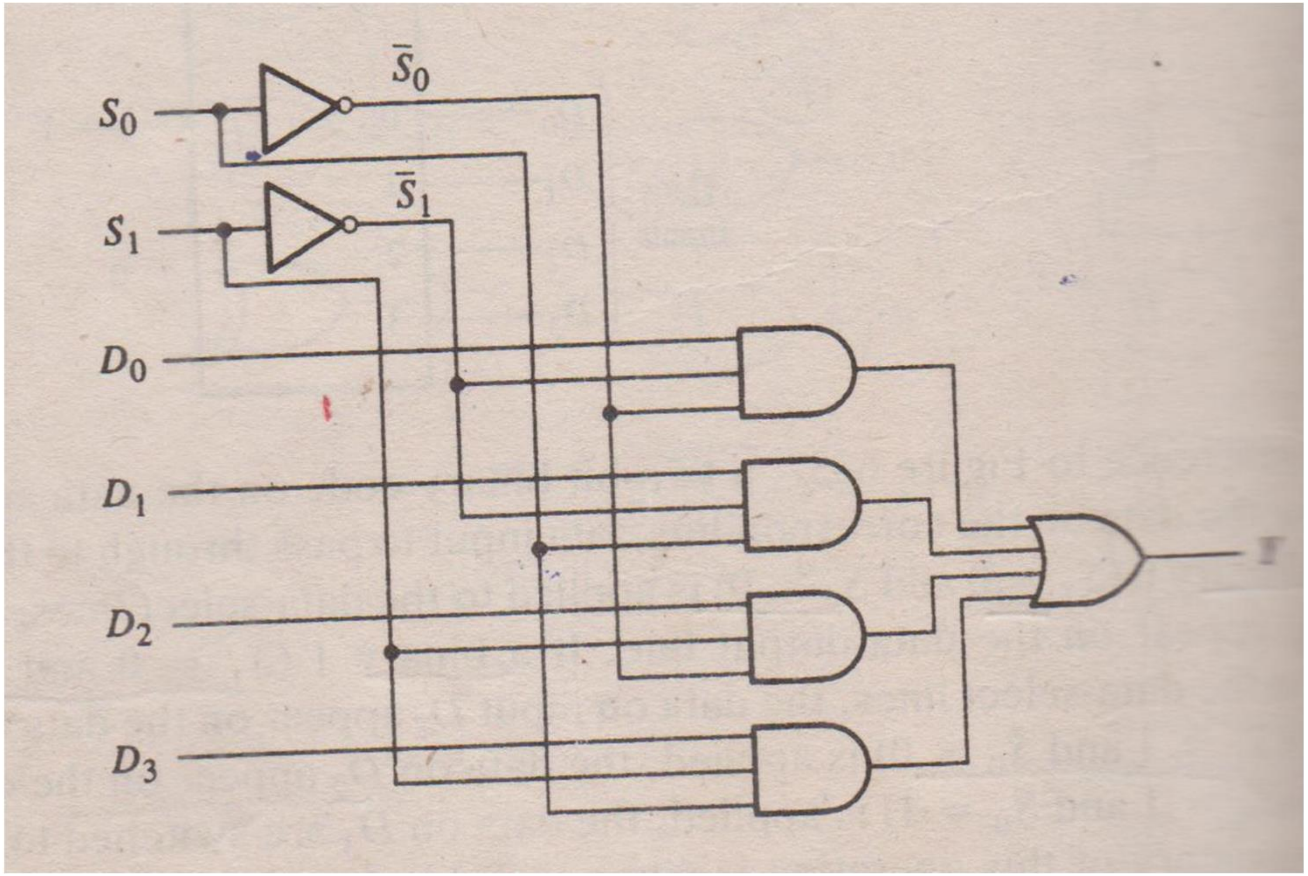


Multiplexer (Data Selector)

- Allows digital information from several sources to be routed onto a single line for transmission to a common destination

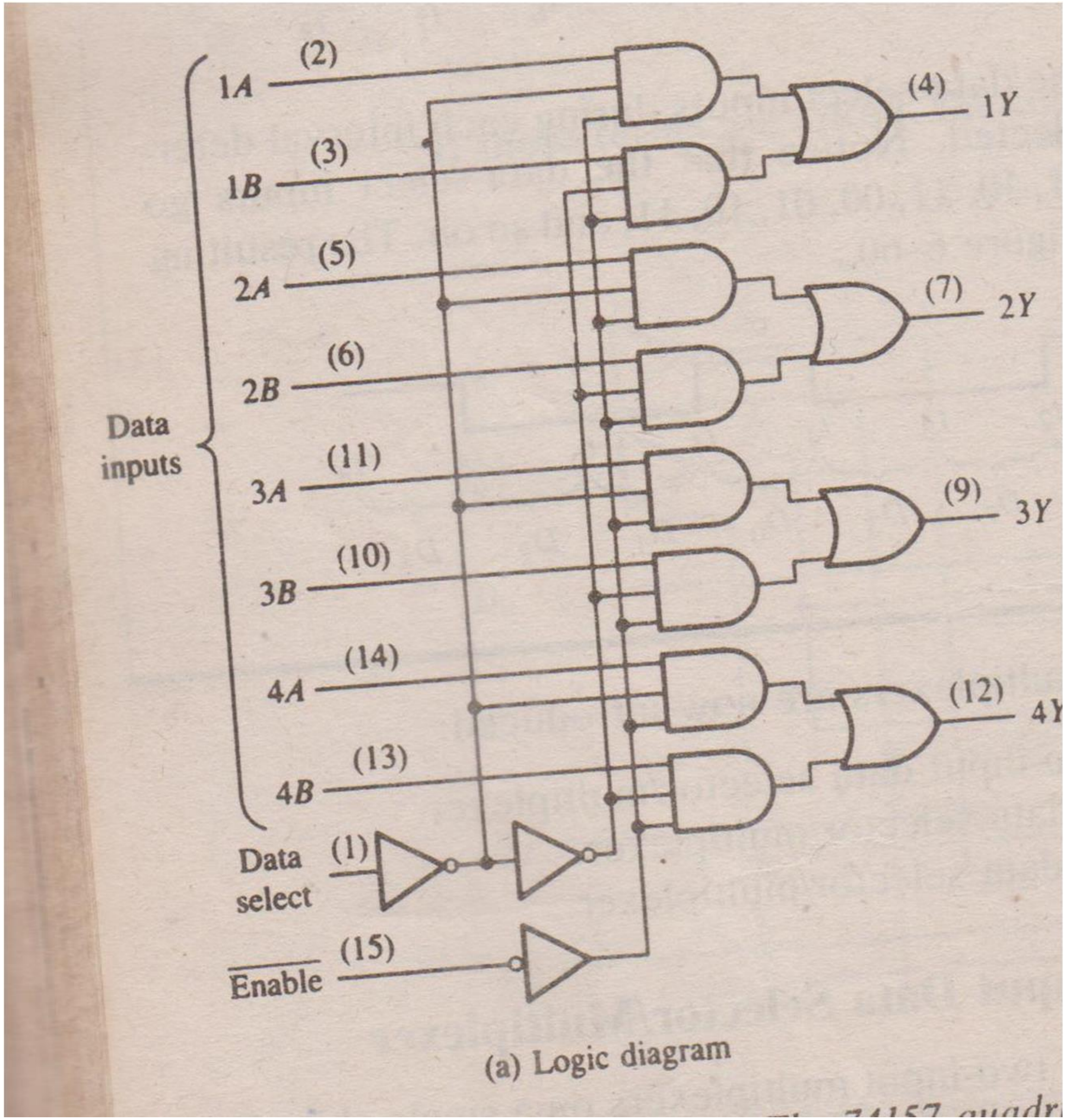


Data-Select Inputs	
S_1	S_0
0	0
0	1
1	0
1	1



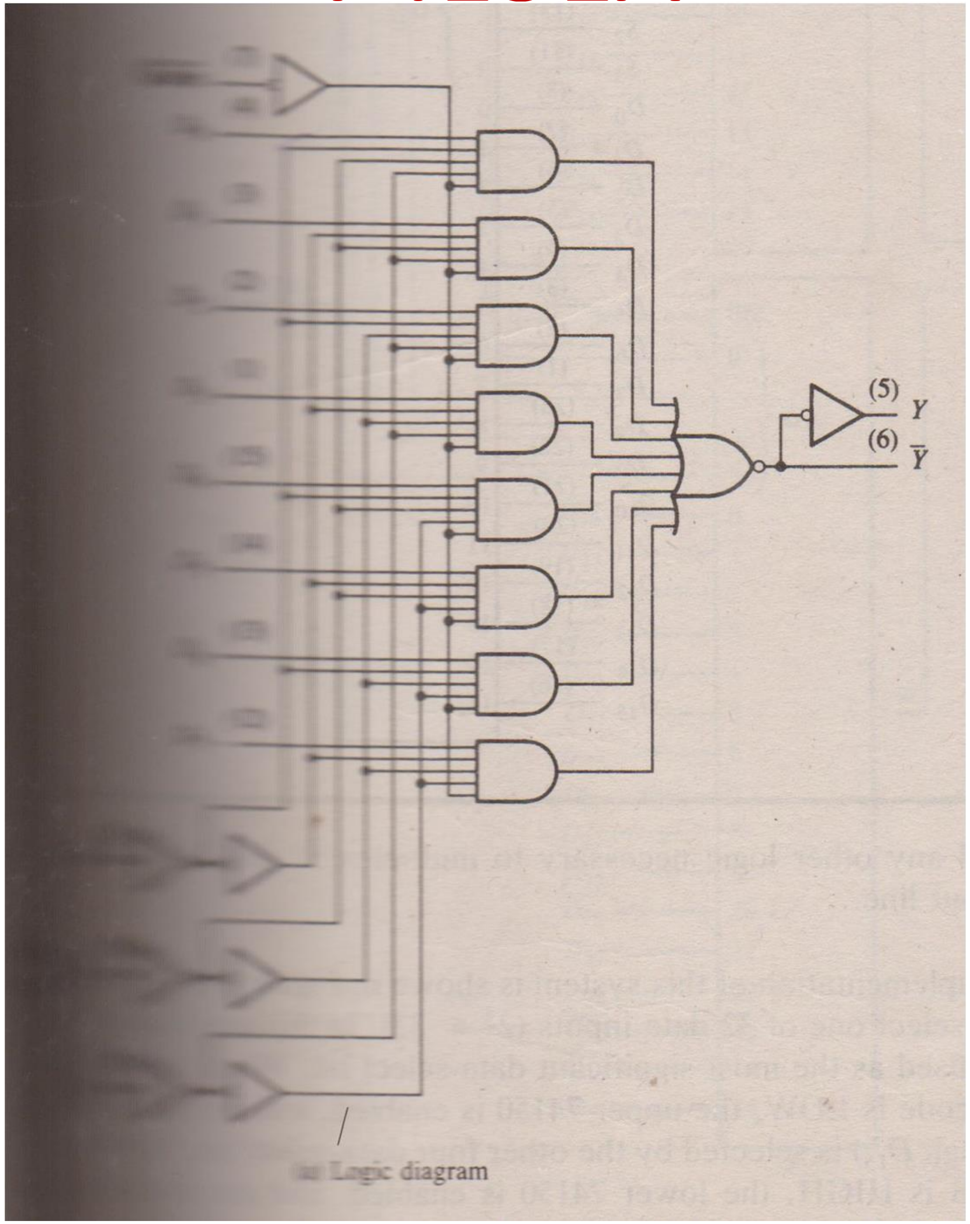
$$Y = D_0\overline{S_1}\,\overline{S_0} + D_1\overline{S_1}S_0 + D_2 S_1\overline{S_0} + D_3 S_1 S_0$$

74157



(a) Logic diagram

74151A

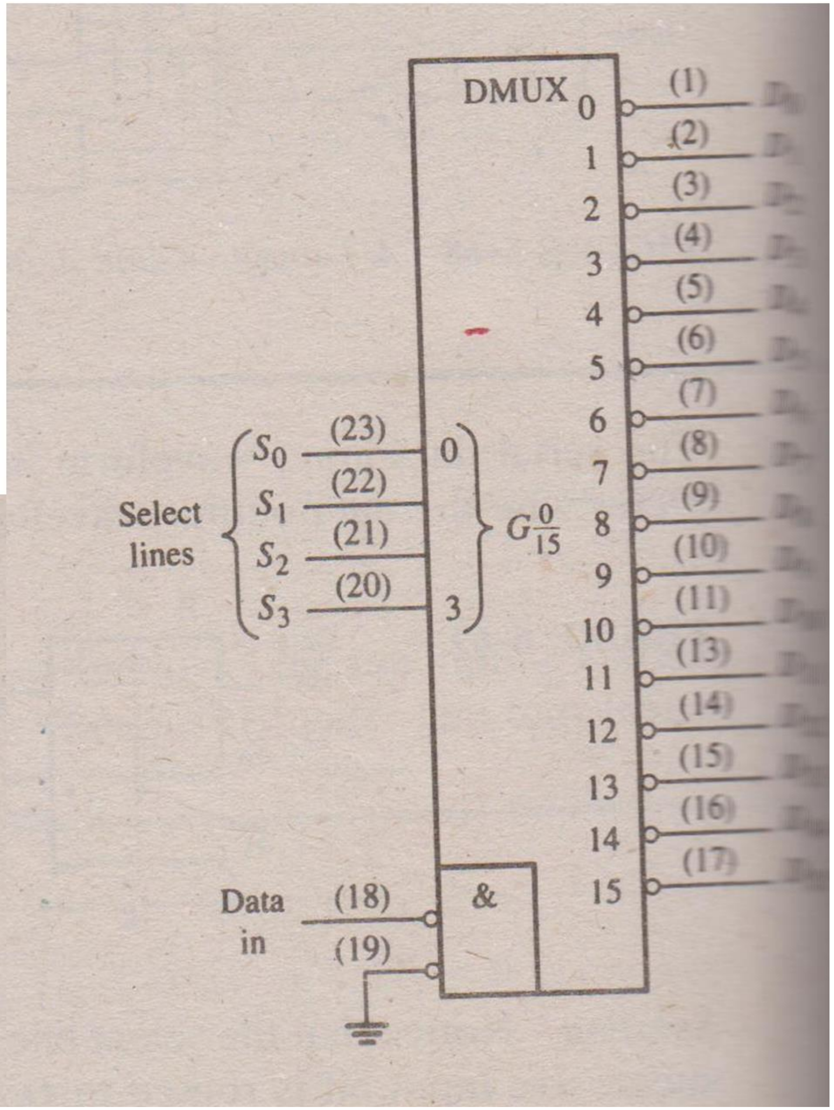


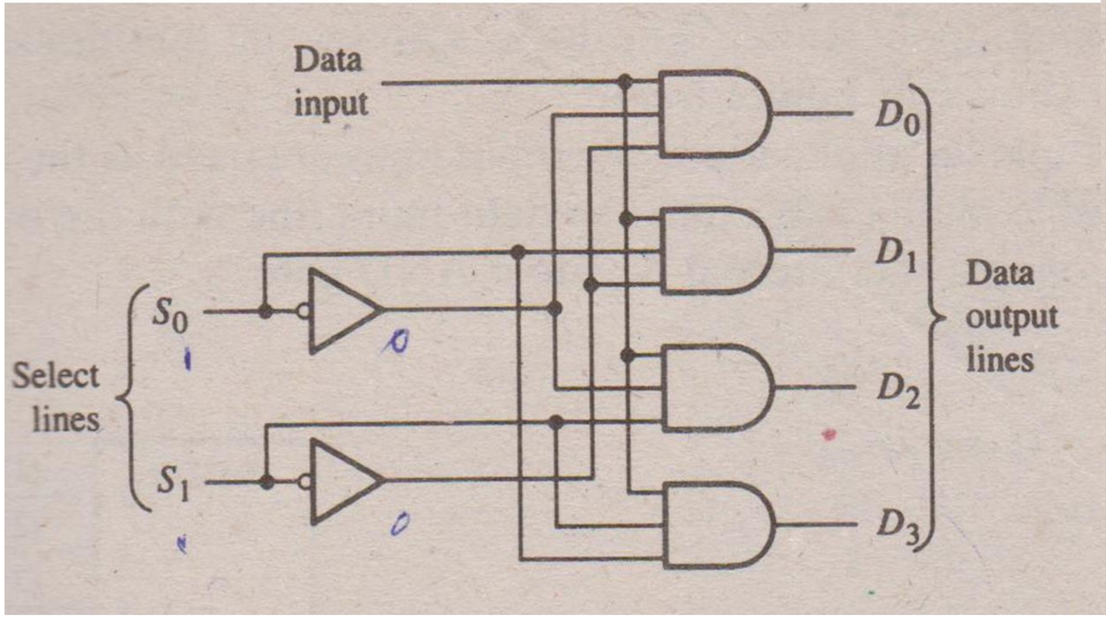
Logic diagram

Demultiplexer

- It takes data from one line and distributes them to a given number of output lines, but the select lines activate only one among them

74154 used as a DMUX

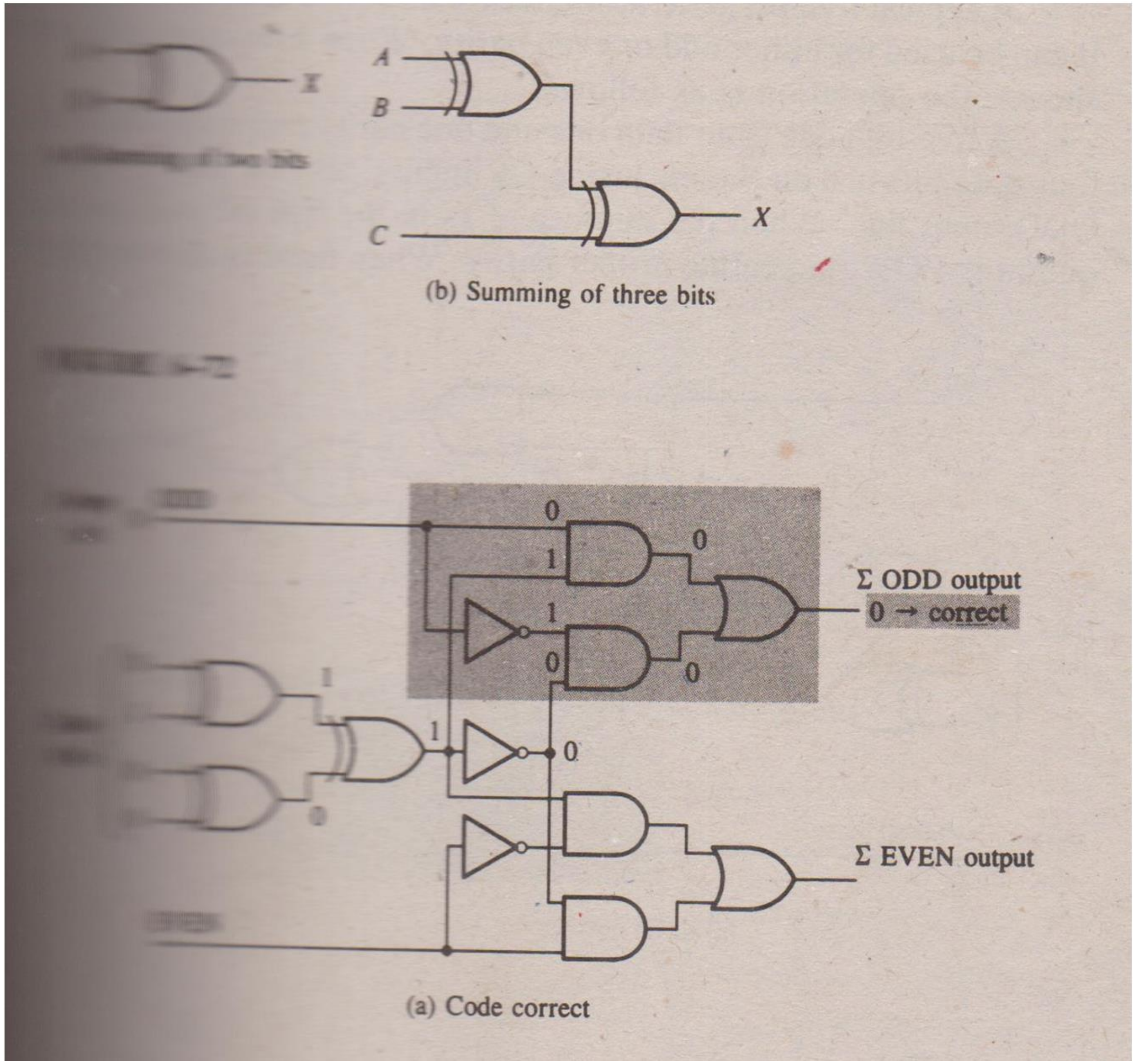




Parity Generator

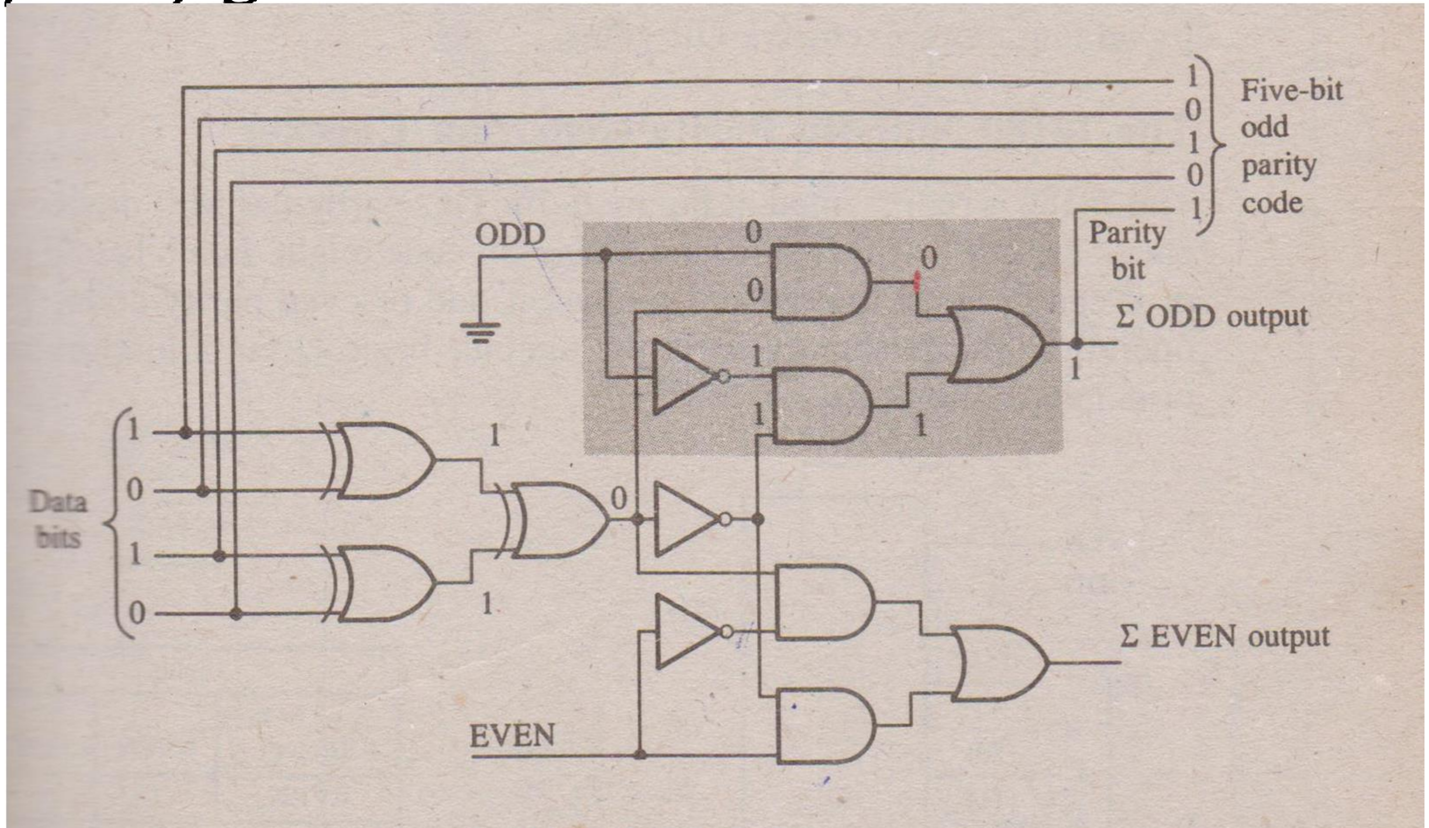
- Errors may occur as digital codes are being transferred from one point to another. These errors may be in the form of a change from 1 to 0 or 0 to 1 due to component mulfunctions or electrical noise
- A parity bit is used to detect a bit error. It is attached to the beginning or end of a group of information bits in order to make the total number of 1s always even (even parity) or always odd (odd parity). A given system operates in one of these parities

- Parity logic: sum of an even number of 1s is always 0, and the sum of an odd number of 1s is always 1
- When the number of 1s in the code is odd (even for even parity), ∑ODD (∑EVEN for even parity) output is LOW, indicating proper parity. Otherwise indicating incorrect parity



Parity Detection

- ODD/EVEN line is grounded for odd/even parity generation



(a) Odd parity generation

Hamming Code

- Identifies an error bit
- $2^p \geq m + p + 1$

p – parity bits, m – information bits

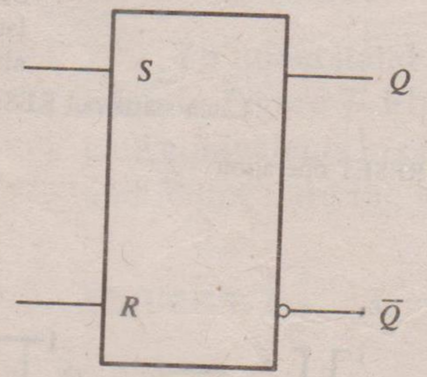
- Determine single error-correcting code for 1001

Bit designation	P_1	P_2	M_1	P_3	M_2	M_3	M_4
Bit position	1	2	3	4	5	6	7
Bit position number	001	010	011	100	101	110	111
Information bits			1		0	0	1
Parity bits	0	0		1			

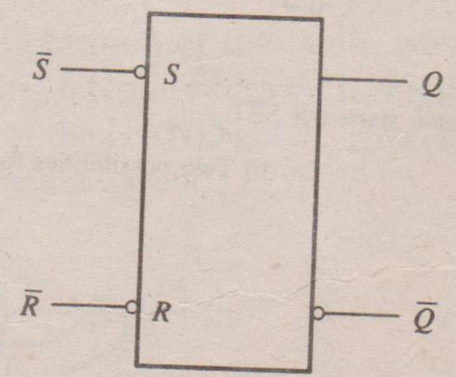
0011001

Multivibrators

- Latch is a bistable device
- Feedback is the characteristic of all multivibrators
- Set-Reset Latch: Assume that both inputs and Q output are HIGH at the beginning
- When the Q output is HIGH, latch is in SET state, otherwise in RESET state.
- It will remain in SET state indefinitely until a LOW is temporarily applied to the $\overline{R}$ input
- It will remain in RESET state indefinitely until a LOW is applied to the $\overline{S}$ input
- LOWs on both inputs are not allowed



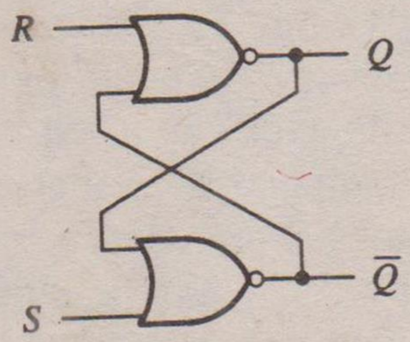
(a) Active-HIGH input S-R latch



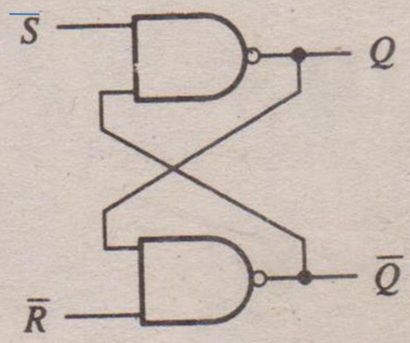
(b) Active-LOW input S-R latch

Inputs		Outputs		
$\bar{S}$	$\bar{R}$	Q	$\bar{Q}$	Comments
1	1	NC	NC	No change. Latch remains in present state.
0	1	1	0	Latch SETS.
1	0	0	1	Latch RESETS.
0	0	1	1	Invalid condition.

SR Latch

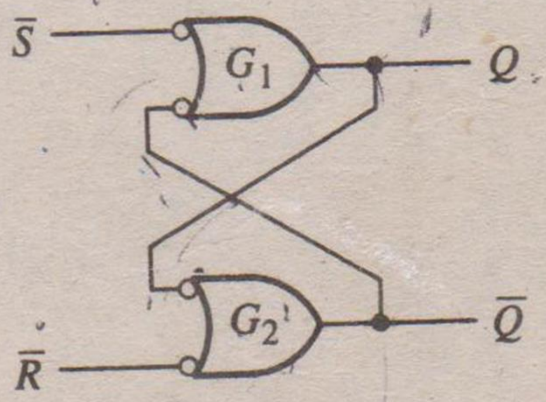


(a) Active-HIGH input S-R latch



(b) Active-LOW input S-R latch

Using negative-OR gate



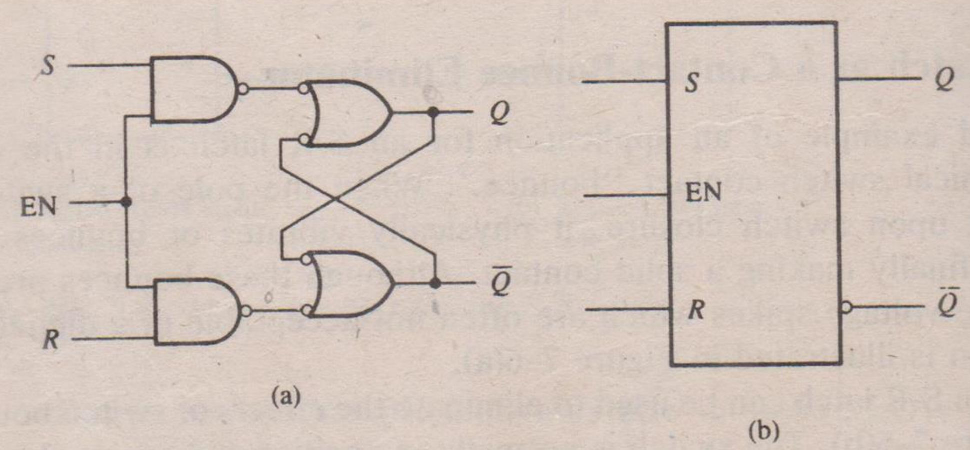


FIGURE 7–7 *A gated S-R latch.*

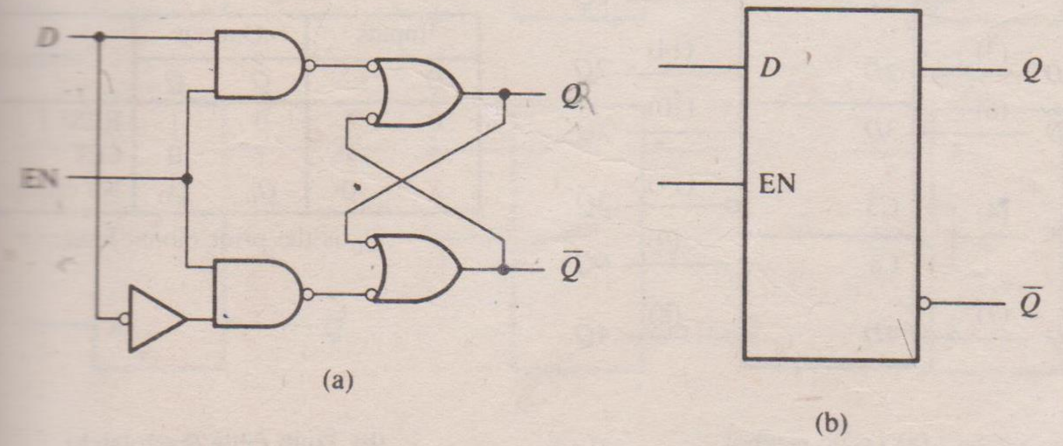
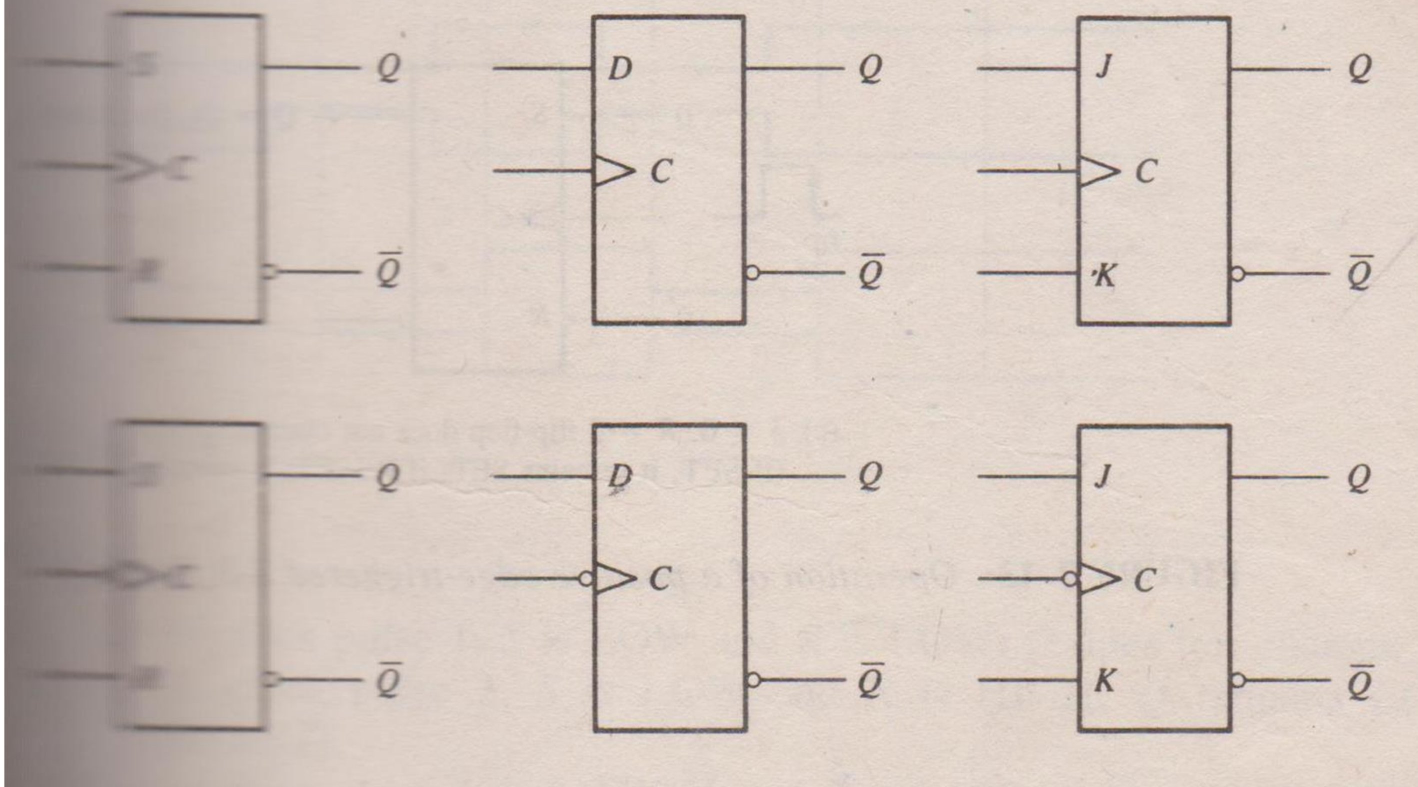


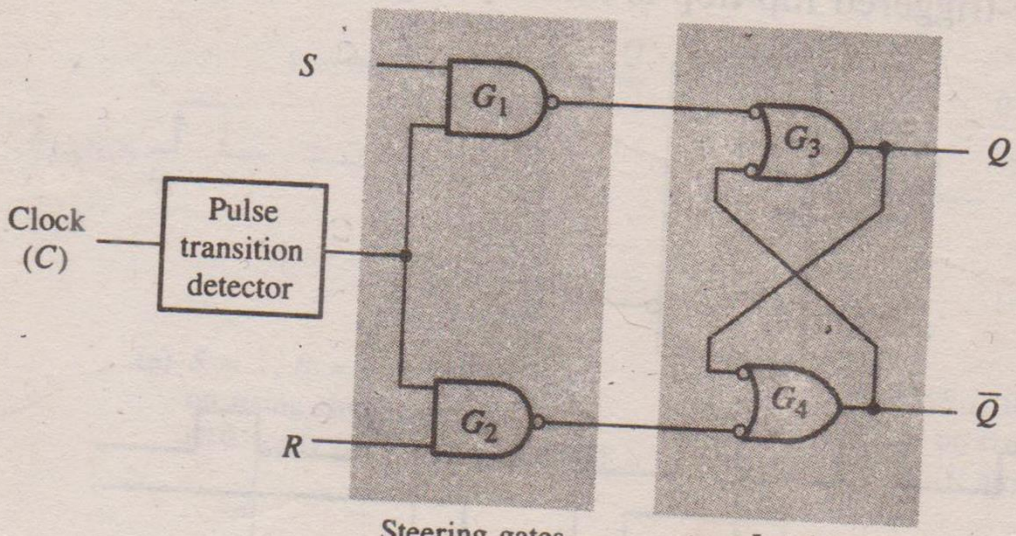
FIGURE 7–9 *A gated D latch.*

Edge Triggered Flip Flops

- Flip flops are synchronous bistable devices
- Synchronous – output state changes only at a specified point on a triggering input called clock. i.e, changes in the output occur in synchronous with the clock
- Edge-triggered – flip flop changes the state either at the positive edge or at the negative edge of the clock

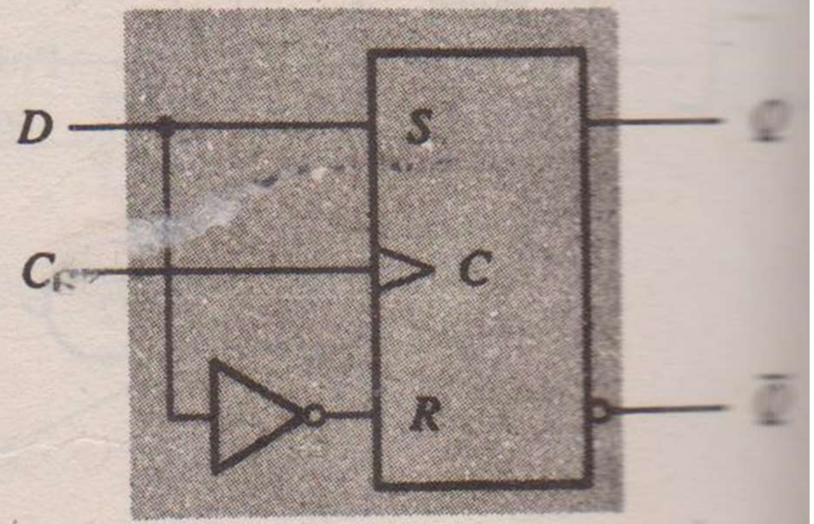


(a) S-R (b) D (c) J-K

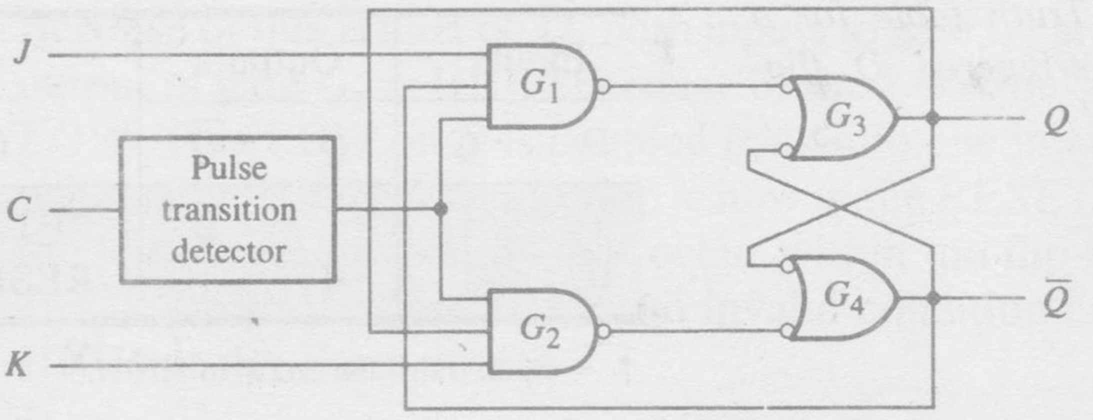


(a) A simplified logic diagram for a positive edge-triggered S-R flip-flop

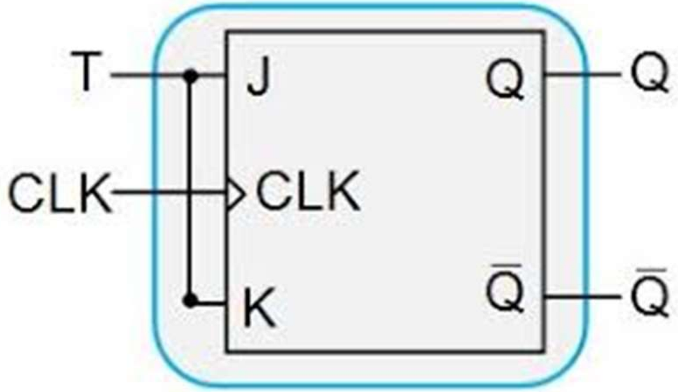
Inputs			Outputs		Comments
S	R	C	Q	$\overline{Q}$	
0	0	X	Q_0	$\overline{Q}_0$	No change
0	1	↑	0	1	RESET
1	0	↑	1	0	SET
1	1	↑	?	?	Invalid

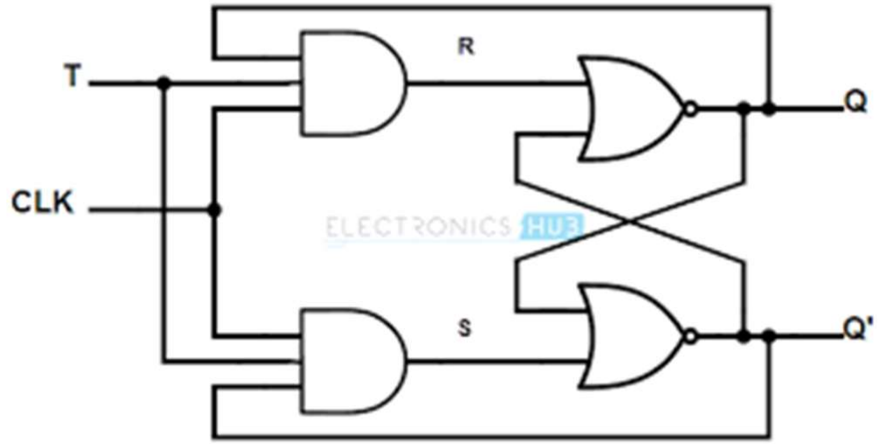


Inputs		Outputs		Comments
D	C	Q	$\overline{Q}$	
1	↑	1	0	SET (stores a 1)
0	↑	0	1	RESET (stores a 0)



Inputs			Outputs		Comments
J	K	C	Q	$\overline{Q}$	
0	0	↑	Q_0	$\overline{Q}_0$	No change
0	1	↑	0	1	RESET
1	0	↑	1	0	SET
1	1	↑	$\overline{Q}_0$	Q_0	Toggle

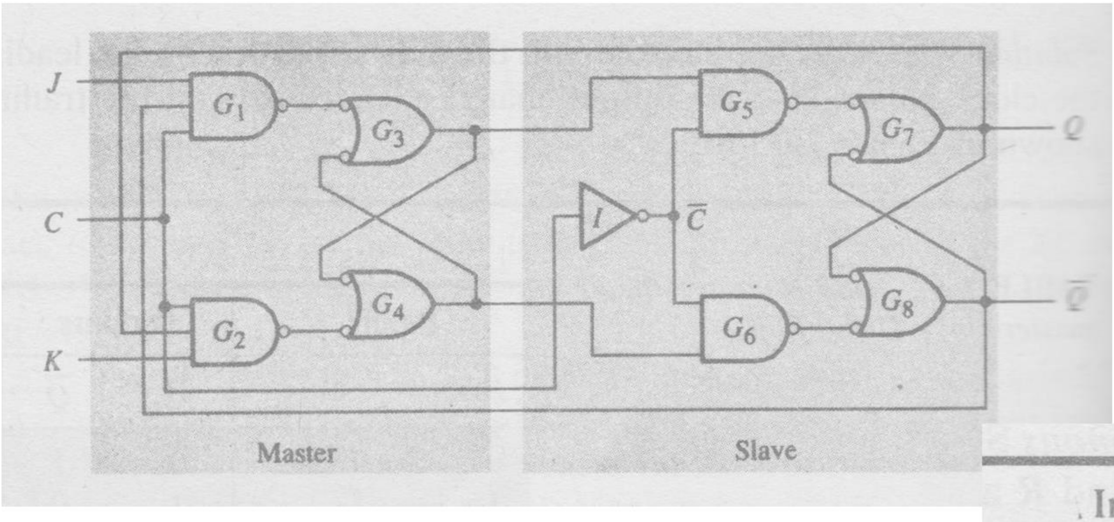


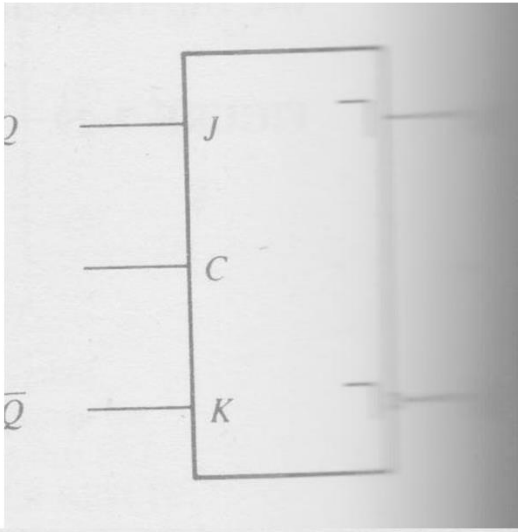


T	Q_n	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

Pulse-triggered (Master-slave) flip flop

- Data are entered into the flip flop on the leading edge of the clock pulse, but the output is postponed until the trailing edge of the clock pulse





Inputs			Outputs		Comments
J	K	C	Q	$\overline{Q}$	
0	0		Q_0	$\overline{Q}_0$	No change
0	1		0	1	RESET
1	0		1	0	SET
1	1		$\overline{Q}_0$	Q_0	Toggle

Cont…

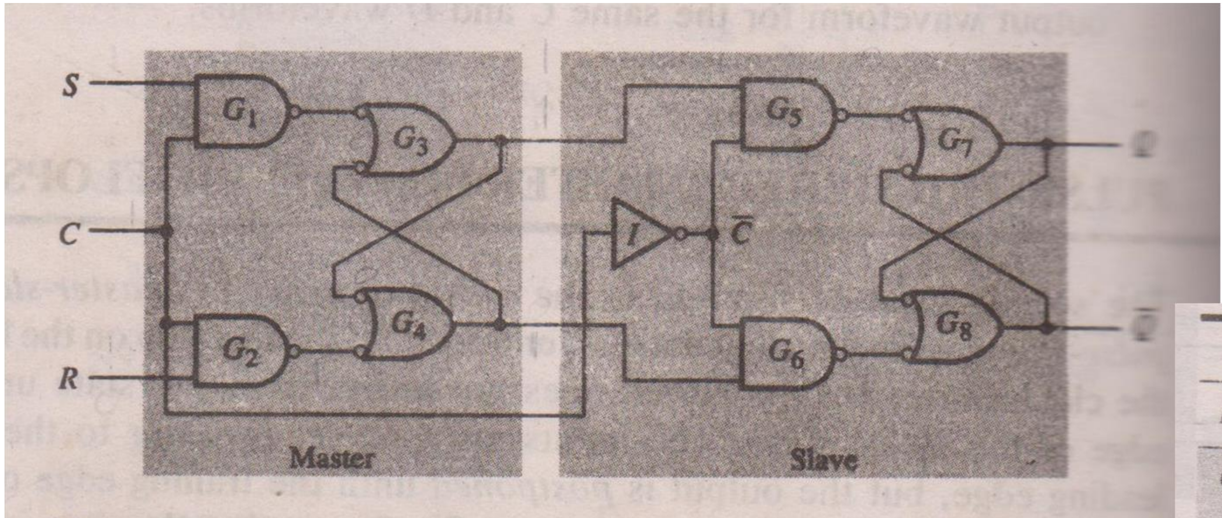


FIGURE 7–31 *Logic diagram for a basic master-slave S-R flip-flop.*

Truth table of MS-D-FF

Inputs		Outputs		
D	C	Q	$\overline{Q}$	Comments
0	⎍	0	1	RESET
1	⎍	1	0	SET

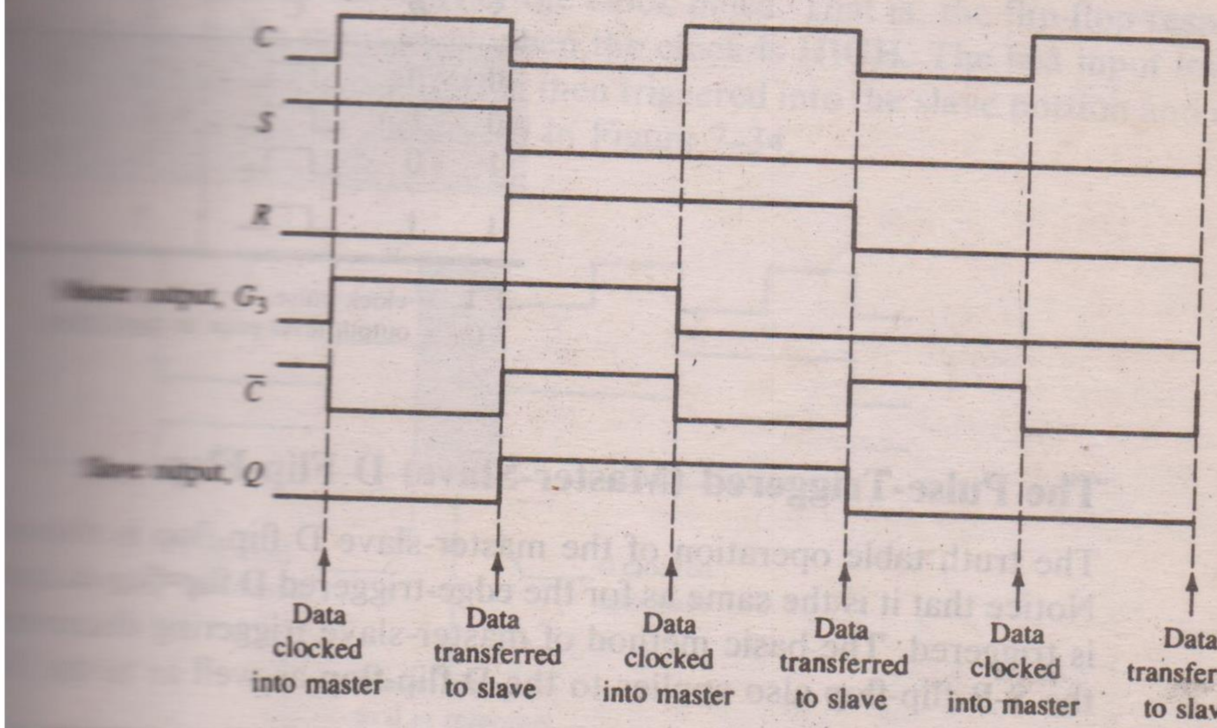
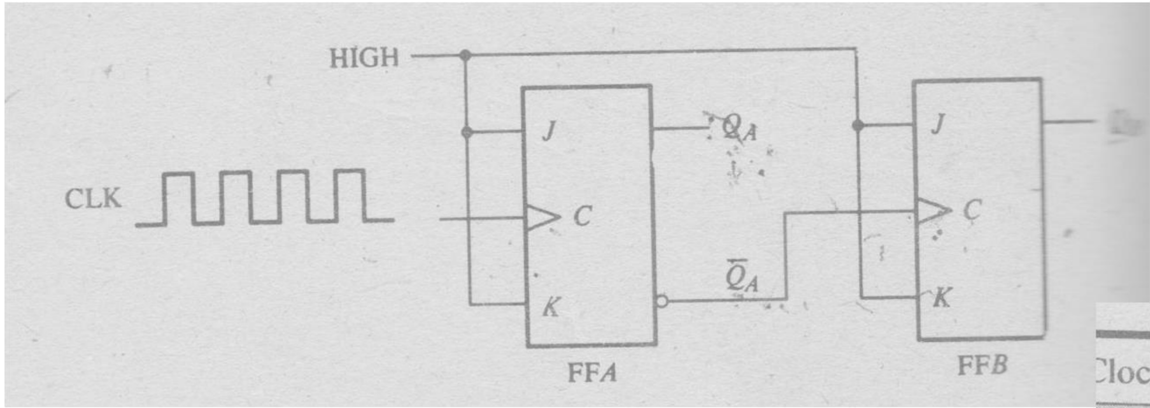


FIGURE 7–32 *Timing diagram for master-slave flip-flop in Figure 7–31, showing SET, RESET, and no-change conditions.*

Counters

- Asynchronous – events that don't occur at the same time
- Asynchronous counter (ripple counter) – flip flops within the counter aren't made to change states at exactly the same time. Clock pulses aren't directly connected to each flip flop in the counter
- The maximum number of possible states (maximum modulus) of a counter is 2^n, where n is the number of flip flops in the counter



Clock Pulse	Q_B	Q_A
0	0	0
1	0	1
2	1	0
3	1	1

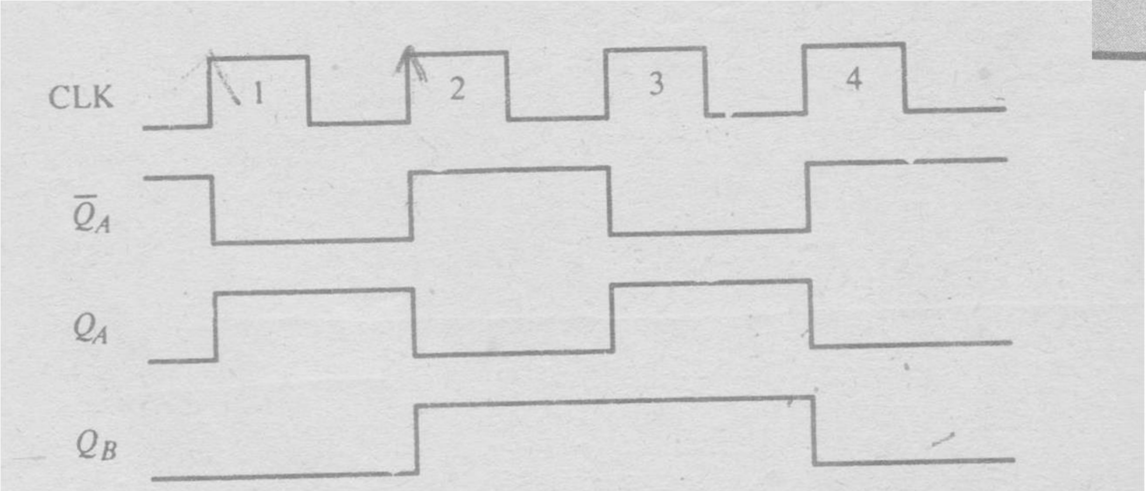
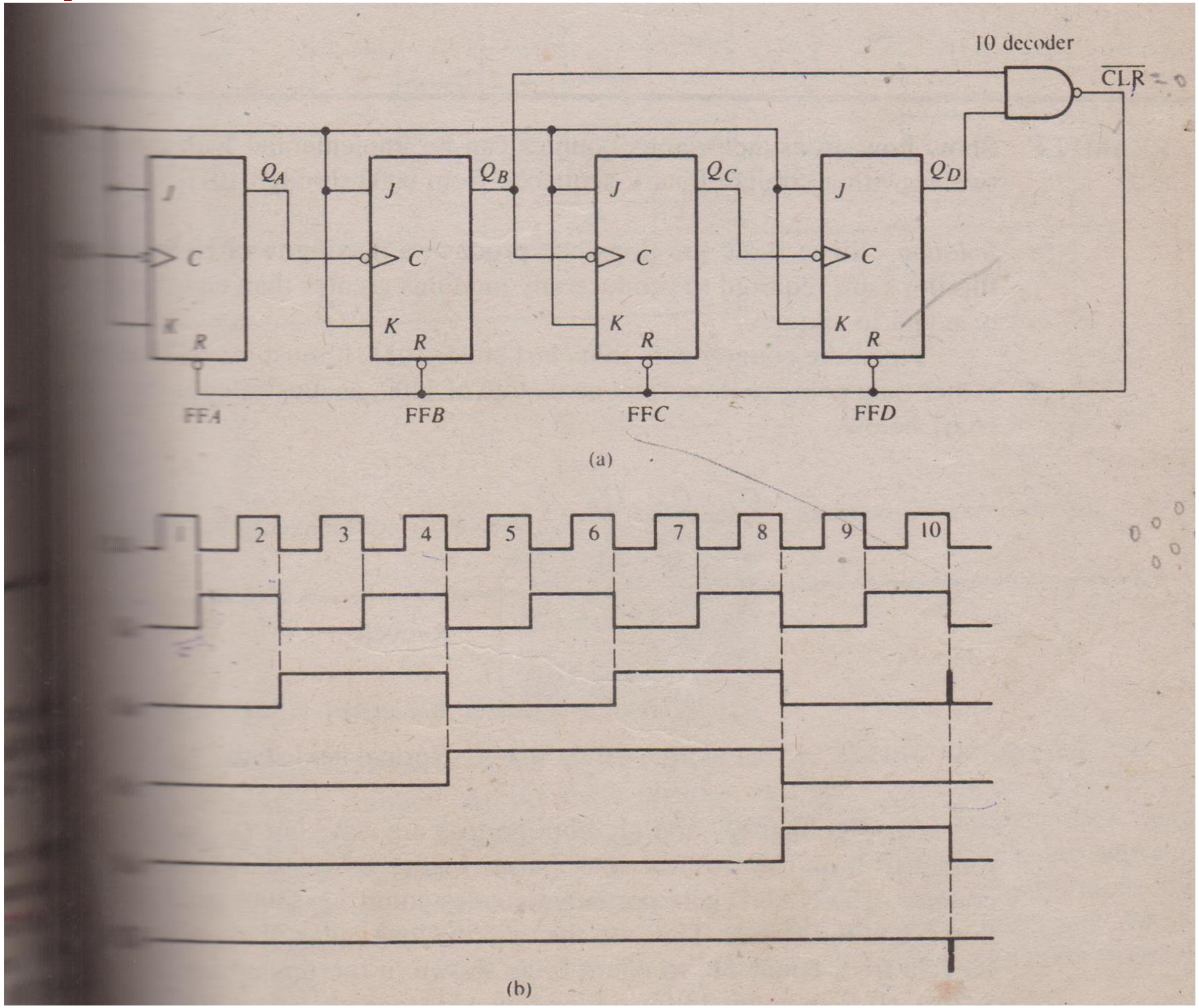


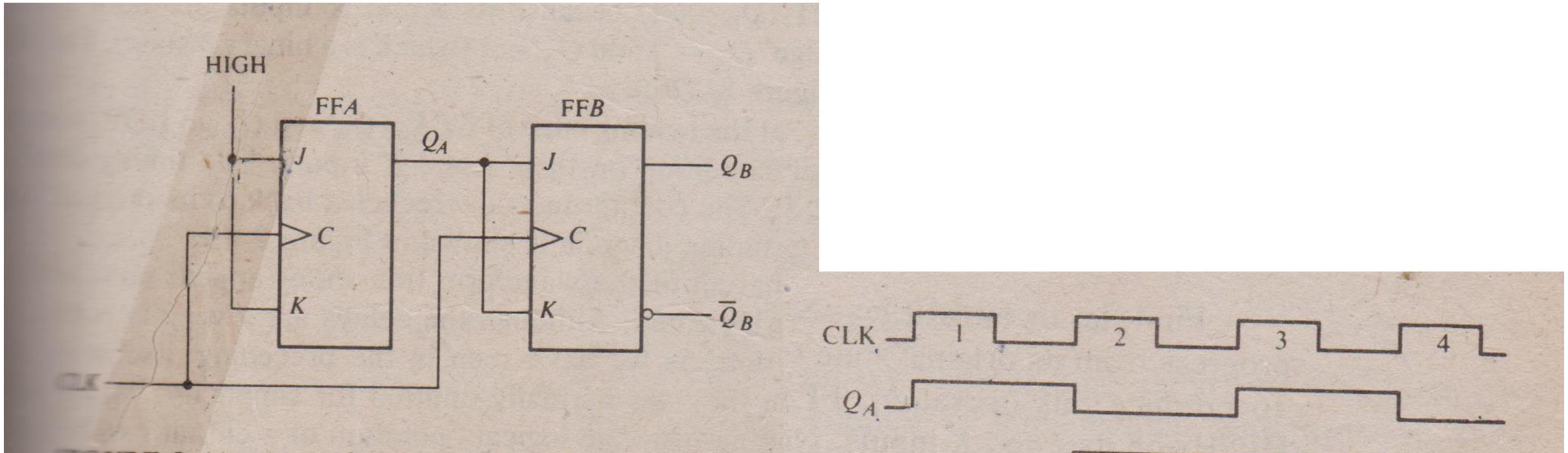
FIGURE 8–2 *Timing diagram for the counter of Figure 8–1.*

Asynchronous Decade Counter



Synchronous Counter

- The counter is clocked such that each flip flop in the counter is triggered at the same time



GURE 8-11 *A two-bit synchronous binary counter.*

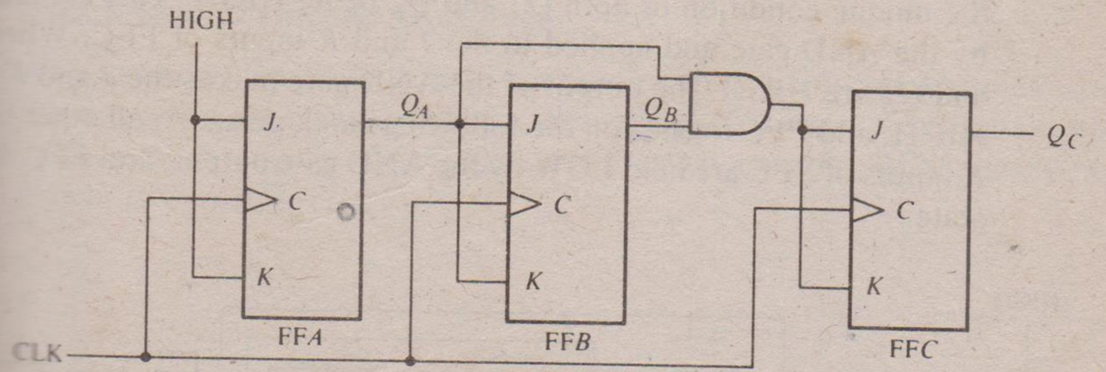


FIGURE 8–14 *A three-bit synchronous binary counter.*

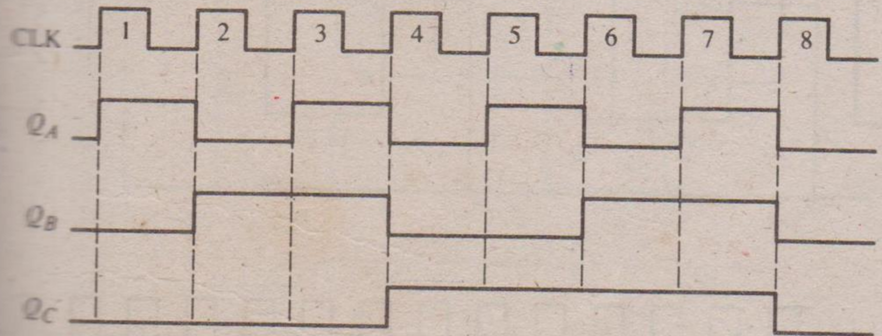


FIGURE 8–15 *Timing diagram for the counter of Figure 8–14.*

TABLE 8–3 *State sequence for a three-stage binary counter.*

Clock Pulse	Q_C	Q_B	Q_A
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

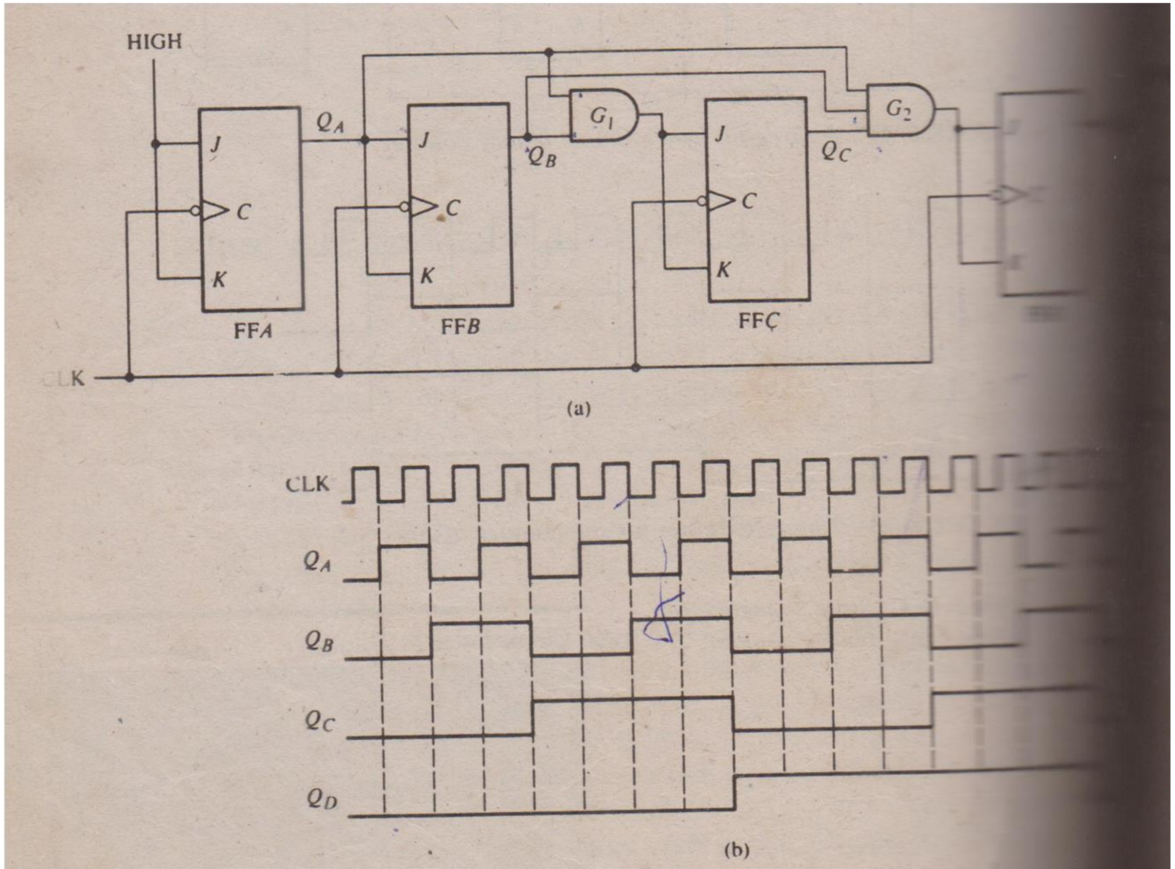


FIGURE 8–16 *A four-bit synchronous binary counter and timing diagram.*

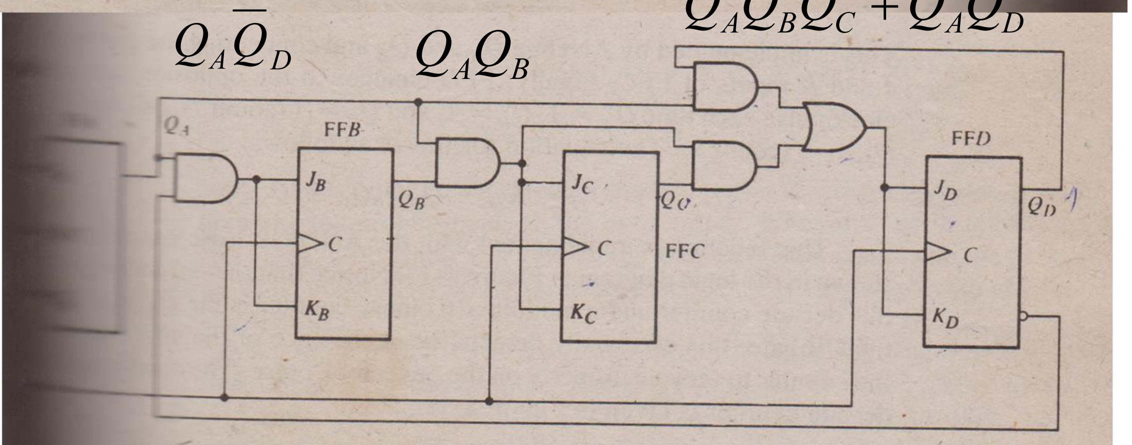


FIGURE 8–17 *A synchronous BCD decade counter.*

UP/DOWN (bidirectional) Synchronous Counter

$$J_B = K_B = Q_A.UP + \overline{Q}_A.DOWN$$

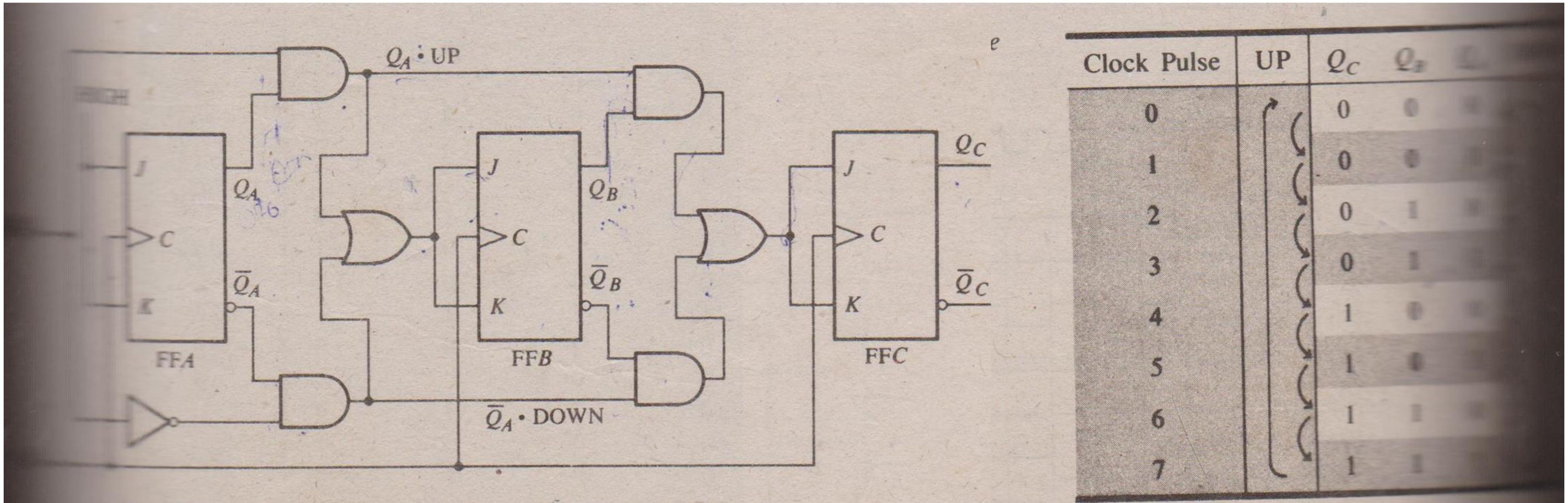
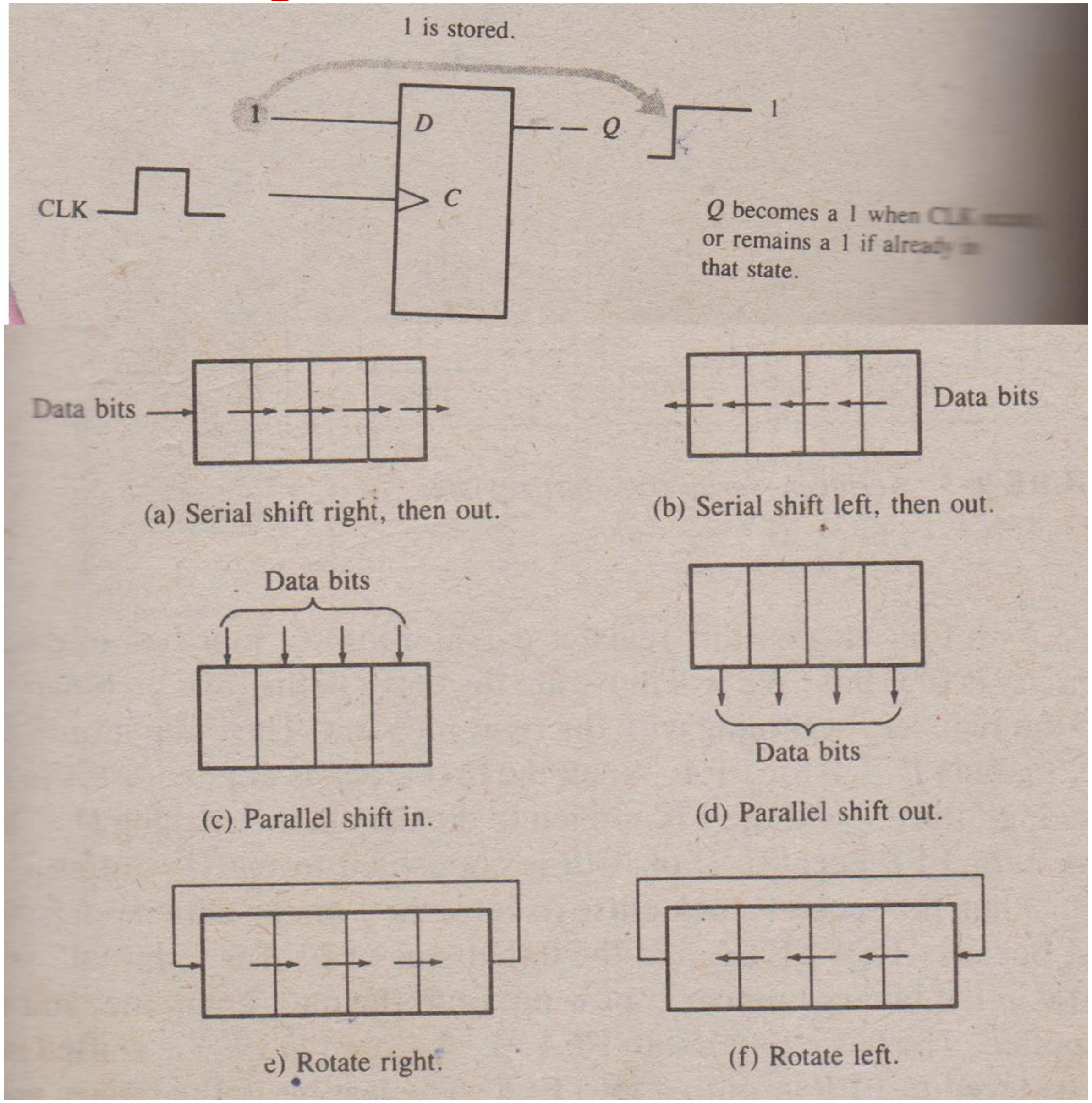


FIGURE 8-23 *A basic three-bit up/down synchronous counter.*

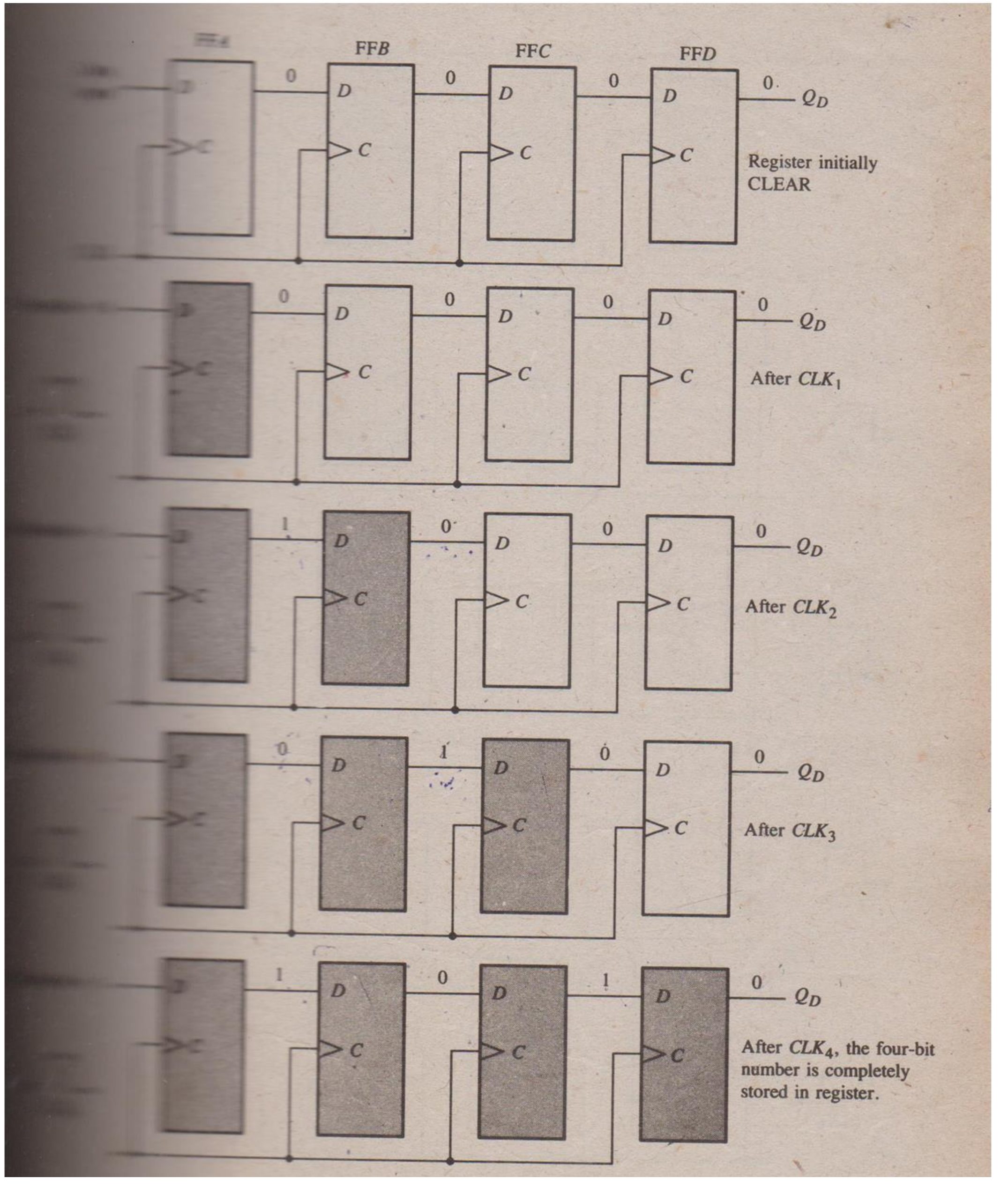
$$J_C = K_C = Q_A.Q_B.UP + \overline{Q}_A.\overline{Q}_B.DOWN$$

Shift Register

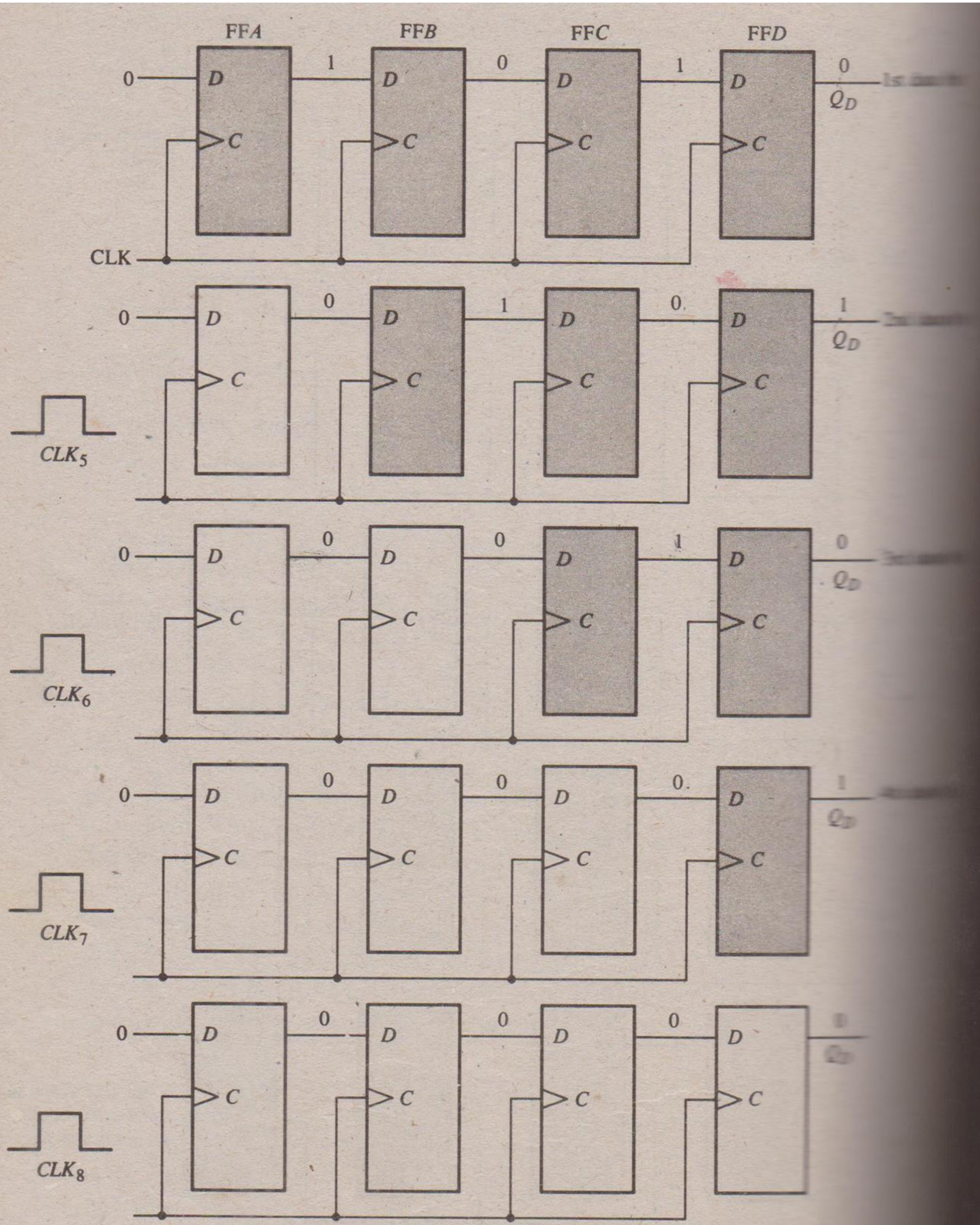
- Implemented with flip flops
- Used for storing (temporarily) and shifting data
- Storage capacity depends on the number of stages in it



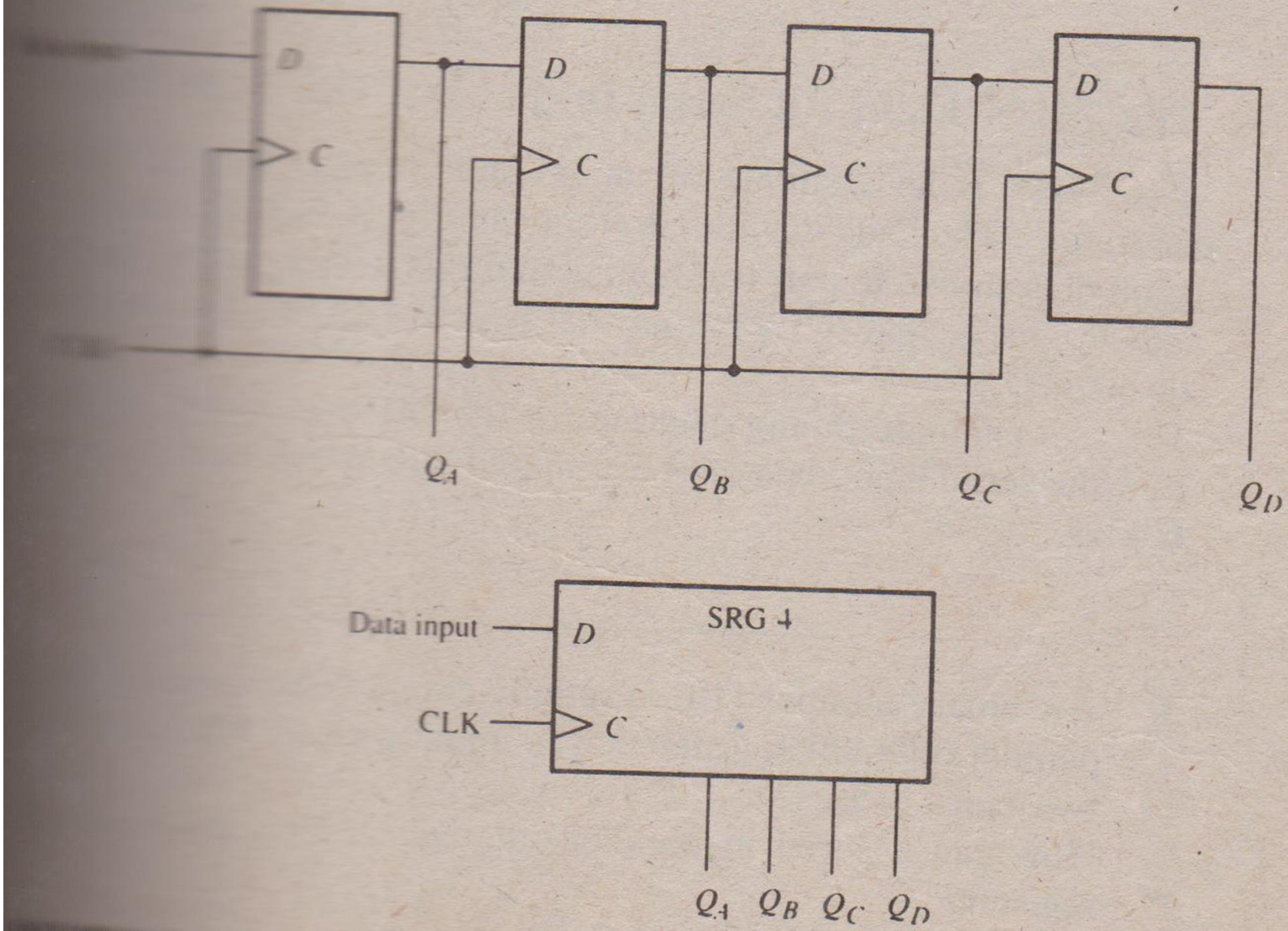
SISO



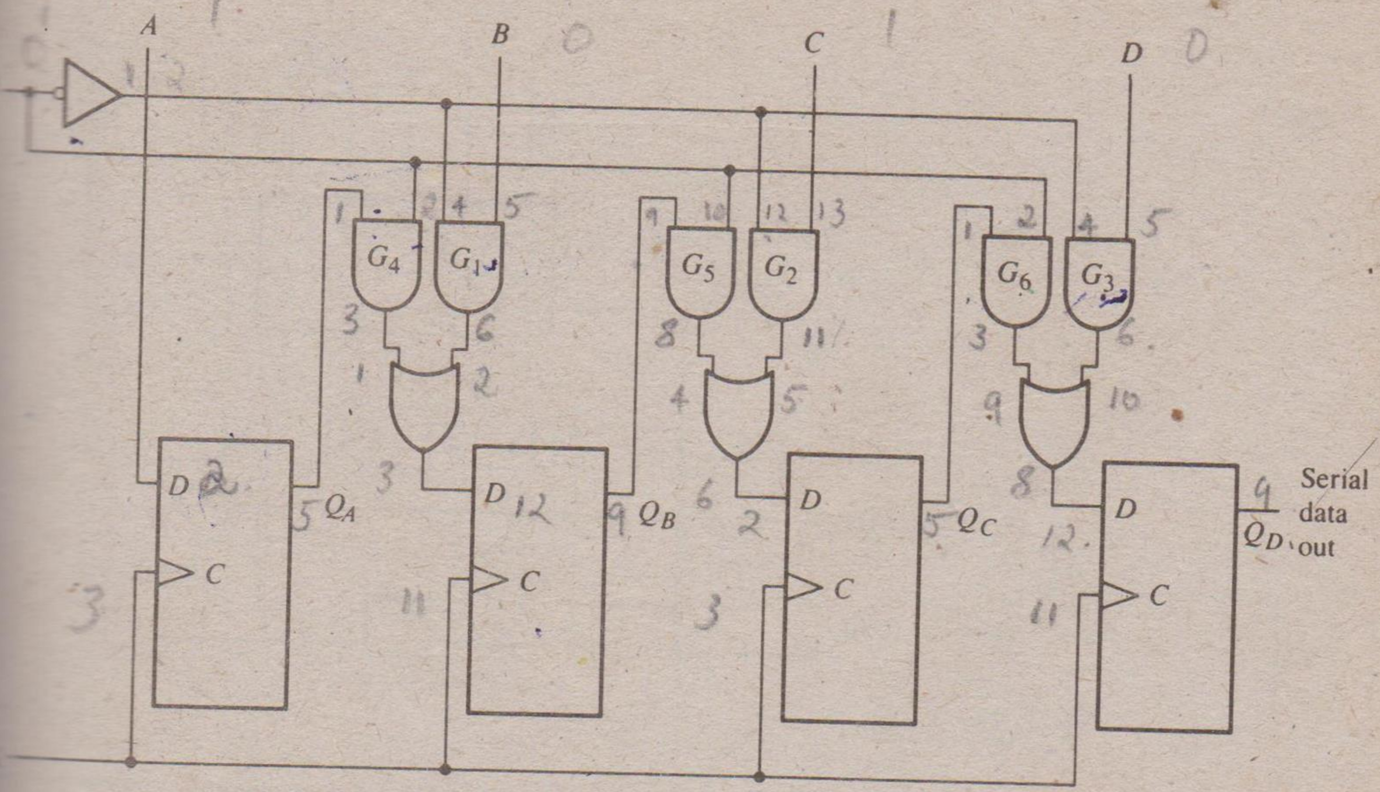
FFB
FFC
FFD
Register initially CLEAR
After CLK_1
After CLK_2
After CLK_3
After CLK_4, the four-bit number is completely stored in register.



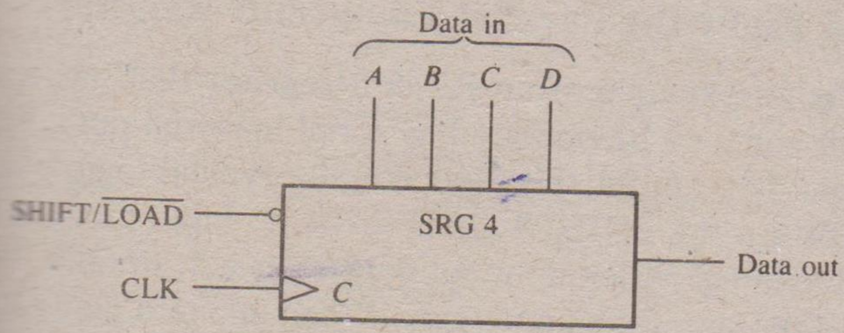
FFA
FFB
FFC
FFD
D
C
Q_D
CLK
CLK_5
CLK_6
CLK_7
CLK_8



A serial in–parallel out shift register.



(a) Logic diagram



(b) Logic symbol

FIGURE 9–12 *A four-bit parallel in–serial out shift register.*

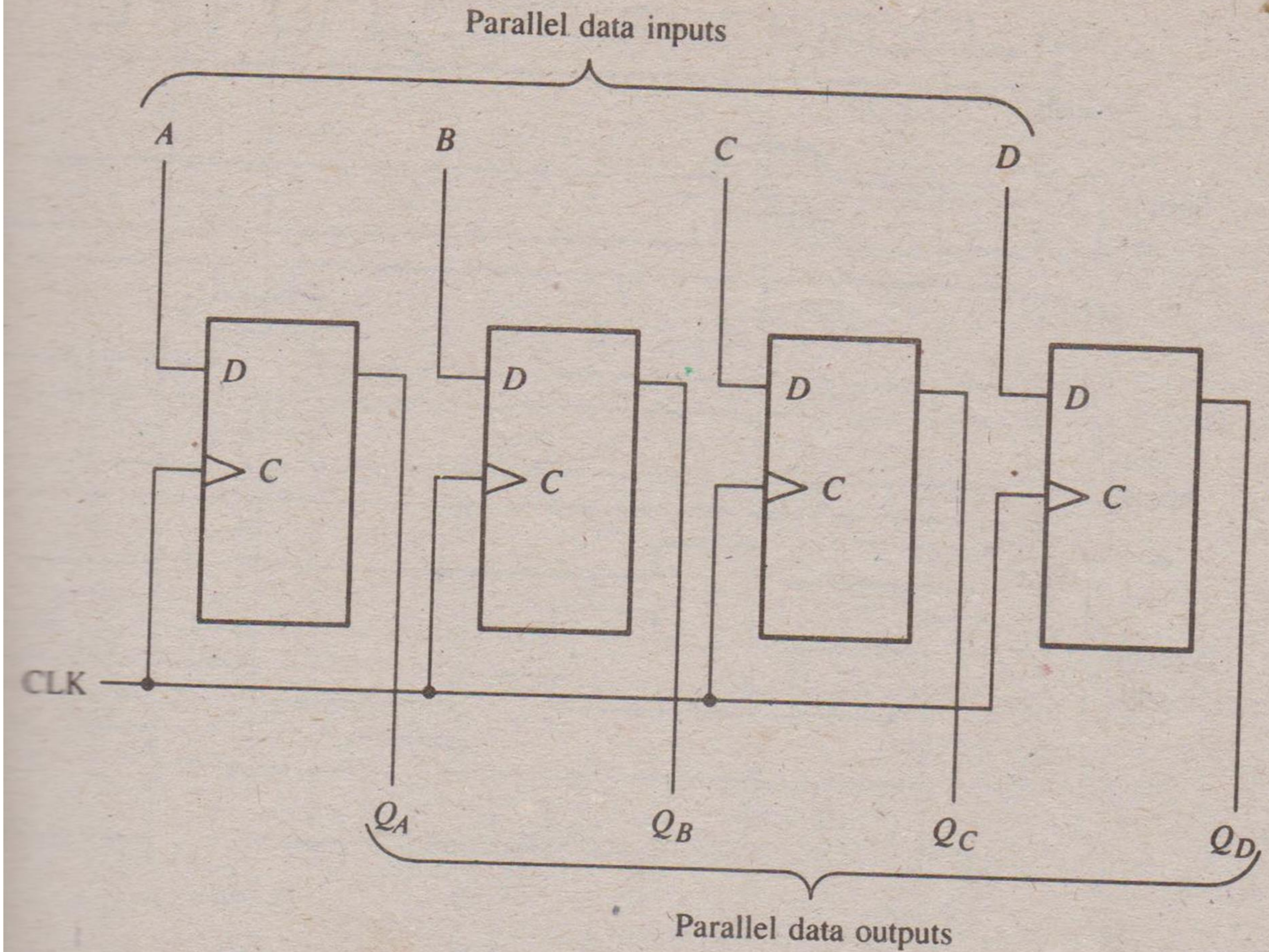
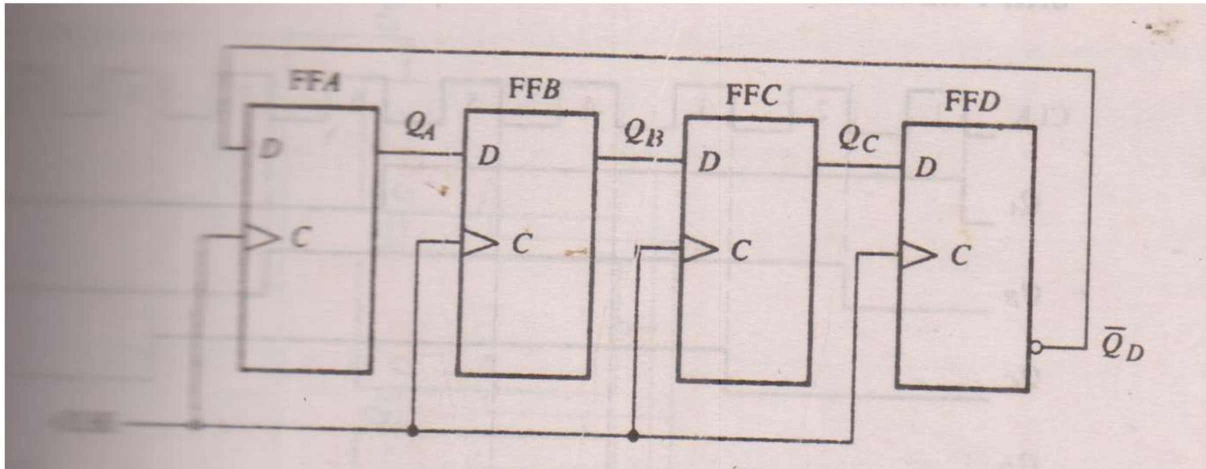


FIGURE 9–16 *A parallel in–parallel out register.*

Johnson Counter



(a) Four-bit Johnson counter

- An n-bit sequence has a total of 2n states

Clock pulse	Q_A	Q_B	Q_C	Q_D
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	0	0	0	1

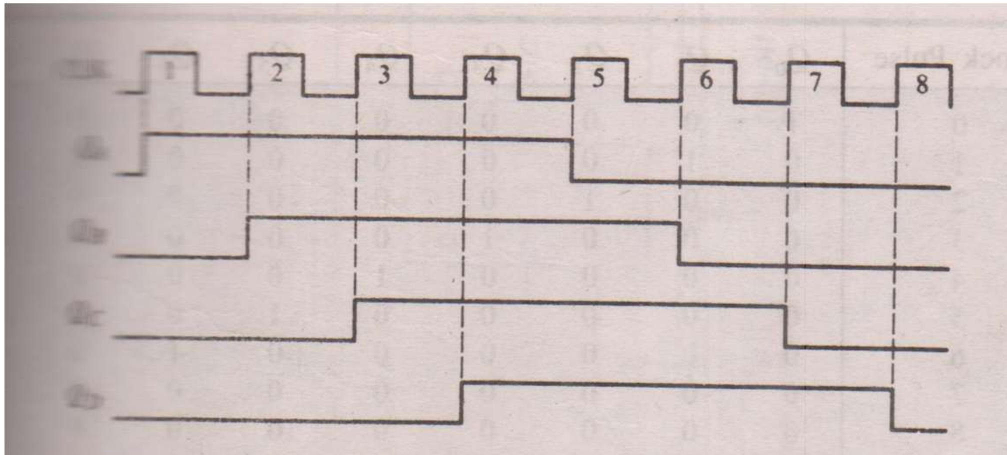


FIGURE 9-24 *Timing sequence for a four-bit Johnson counter.*

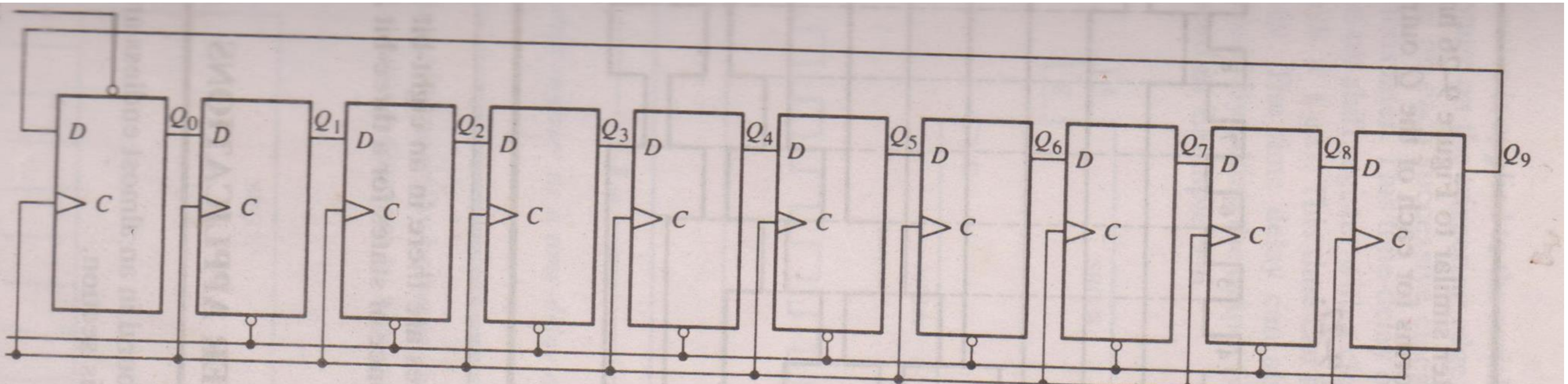


FIGURE 9–26 *A ten-bit ring counter.*

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