SUBJECT: AIDS/IIOT 205, Digital Logic Design

Date: 2022-2023

Analog and Digital

Analog

- A analog system processes analog signals.
- Example: voice, temperature captured by analog sensor, wind flow, etc.

Digital

- A digital system processes digital signals.
- Examples: computer, cellphone, DVD, digital camera, etc.

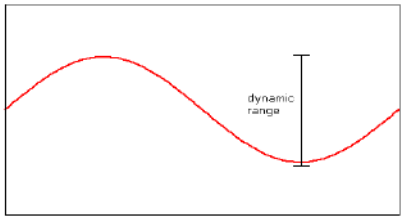
ANALOG GOES DIGITAL

- Photography
- Video
- Automobile applications
- Telephony/Telecomunications
- Traffic lights
- Special effects

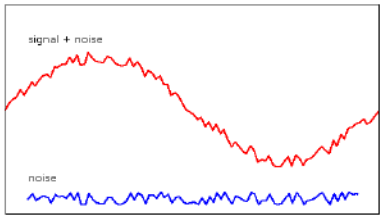
Application of digital technology

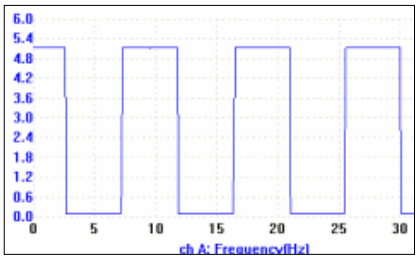
- Television
- Communication systems
- Radar
- Navigation and guidance system
- Military system
- Medical instrument
- Industrial process control etc.

Example of using digital over analog: Telecommunications

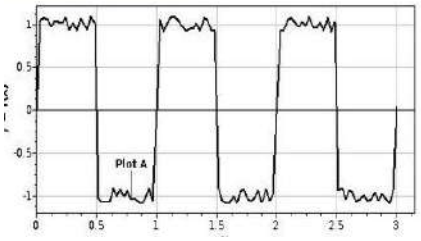


Noisy channel





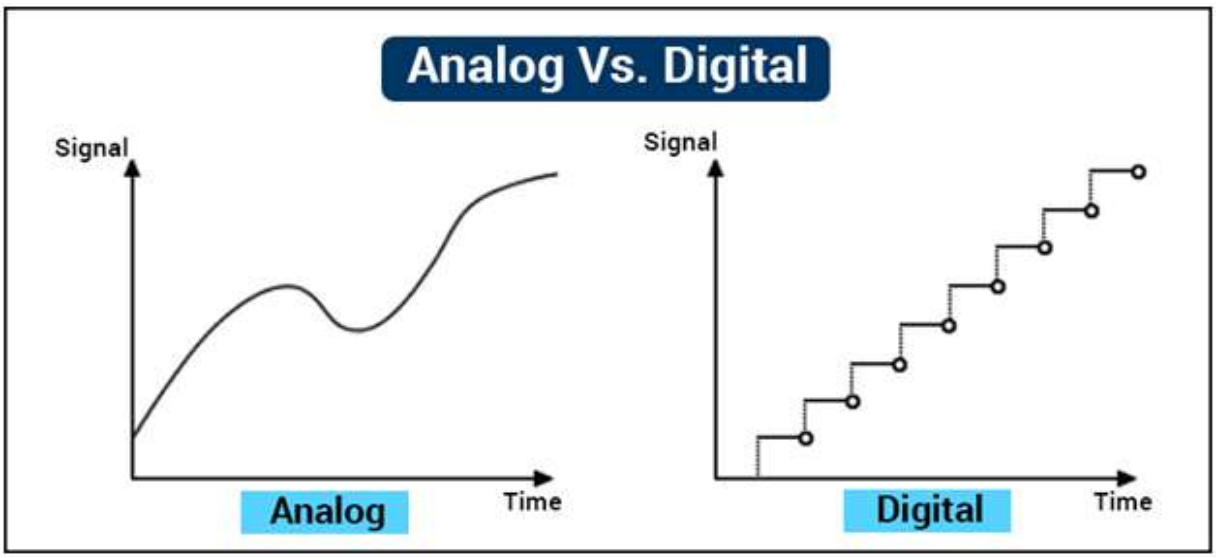
Noisy channel



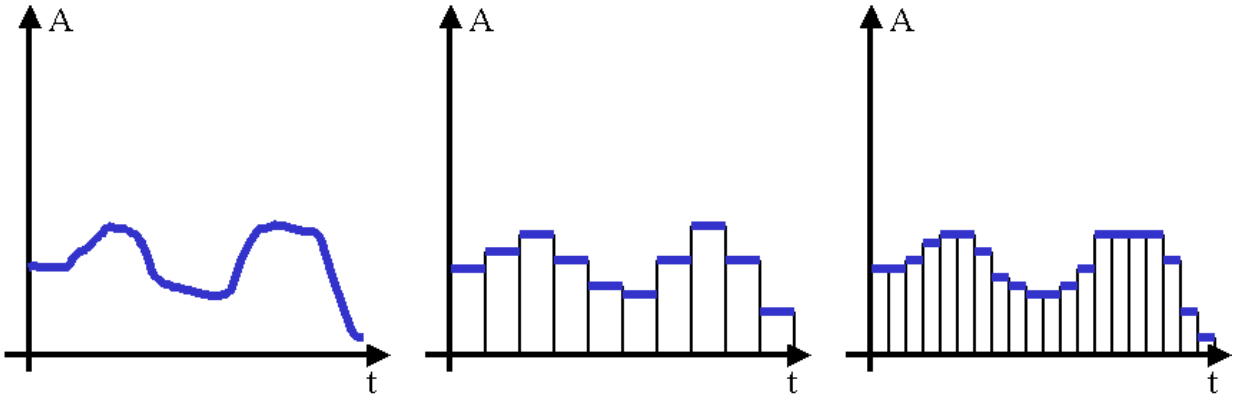
- Digital systems are less sensitive to noise
- As long as 0 is distinguishable from 1

Introduction

Difference Between Analog And Digital Signal



Difference Between Analog And Digital Signal

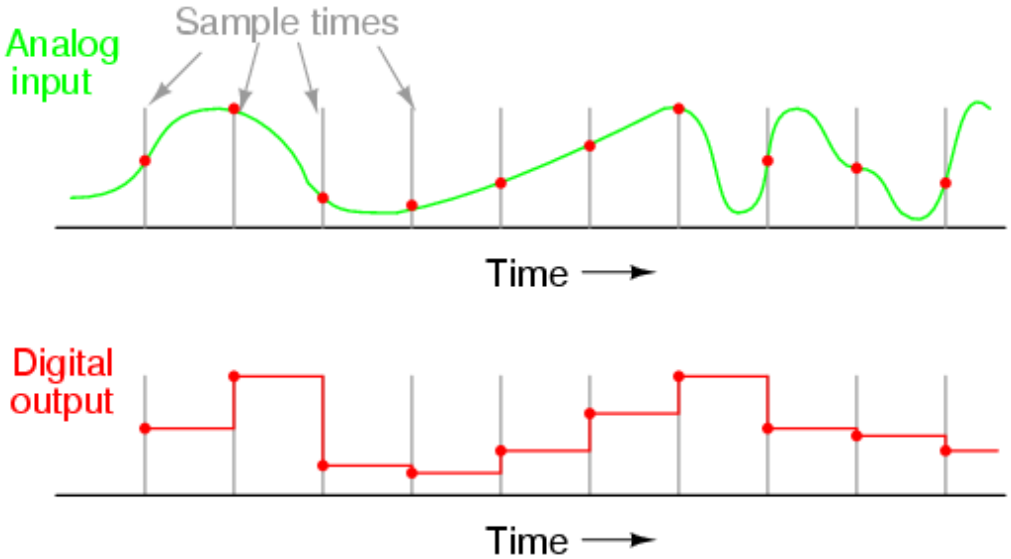


Analog signal – continuously varying

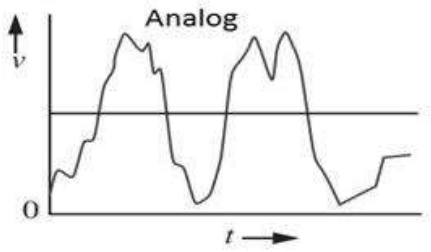
Digital signal – large time divisions

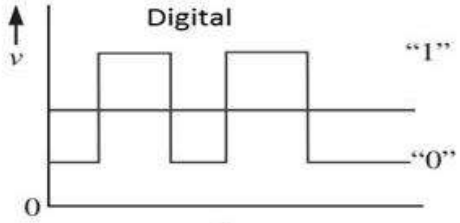
Digital signal – small time divisions

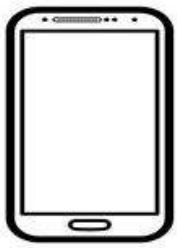
Difference Between Analog And Digital Signal



Analog vs Digital







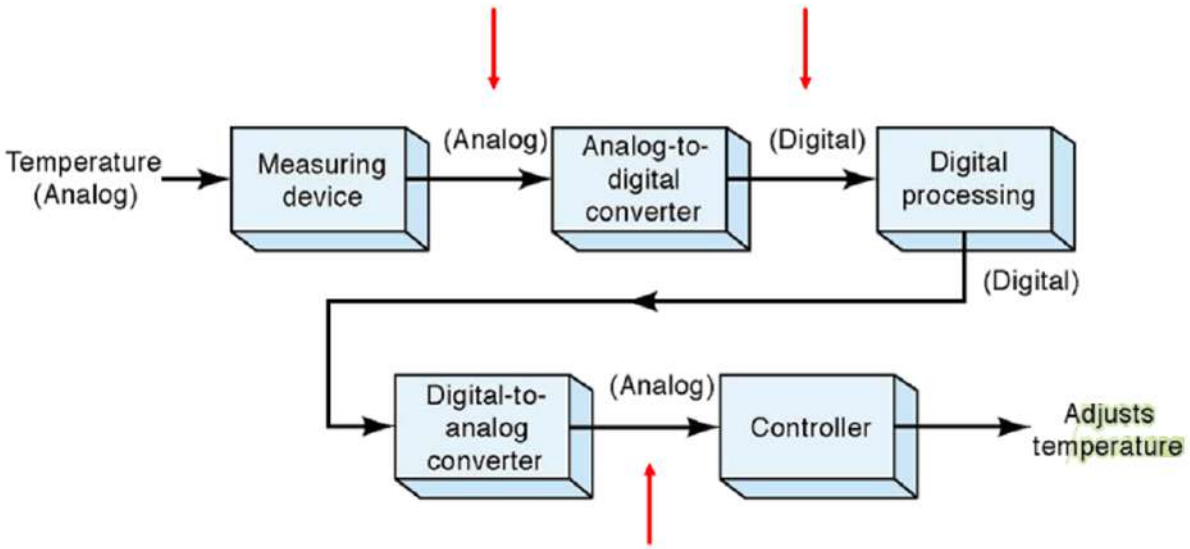
Advantages of Digital Systems over Analog Systems

- More reliable than analog systems due to better immunity to noise & better accuracy.
- Ease of design: No special math skills needed to visualize the behavior of small digital (logic) circuits.
- Programmability.
- Speed: A digital logic element can produce an output in less than 10 nanoseconds (10^{-8} seconds).
- Economy: Due to the integration of millions of digital logic elements on a single miniature chip forming low cost integrated circuit (ICs).

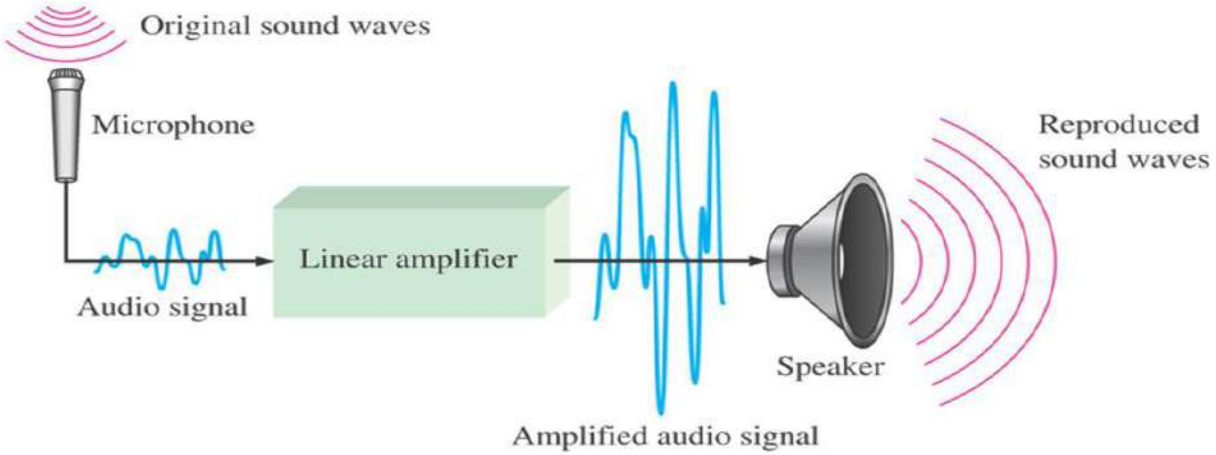
Disadvantages of Digital Systems

The world in which we live is analog, and signals from this world such as light, temperature, sound, electrical conductivity, electric and magnetic fields, and phenomena such as the flow of time, are for most practical purposes continuous and thus analog quantities rather than discrete digital ones.

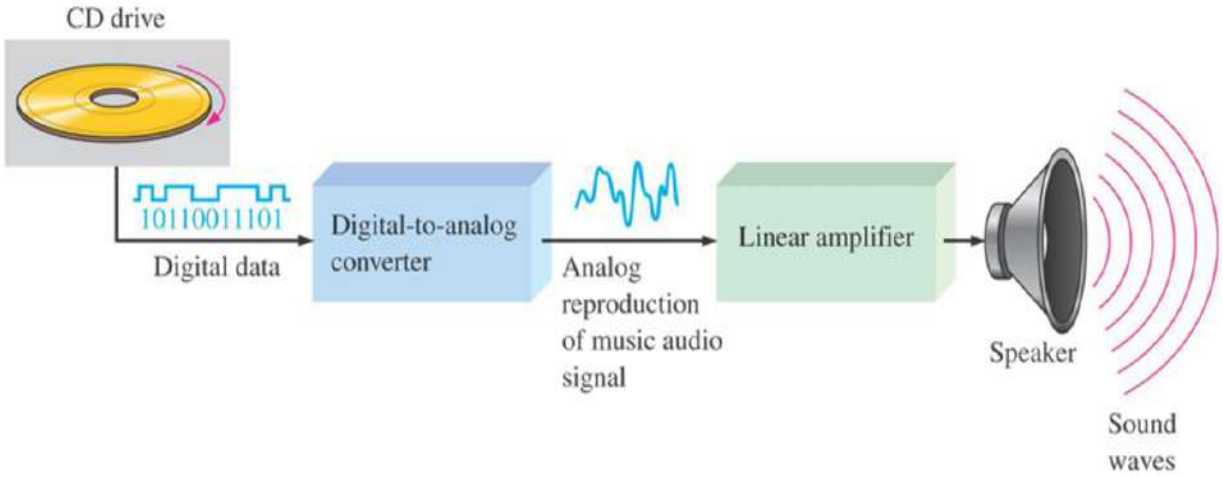
A system using Digital and Analog Methods



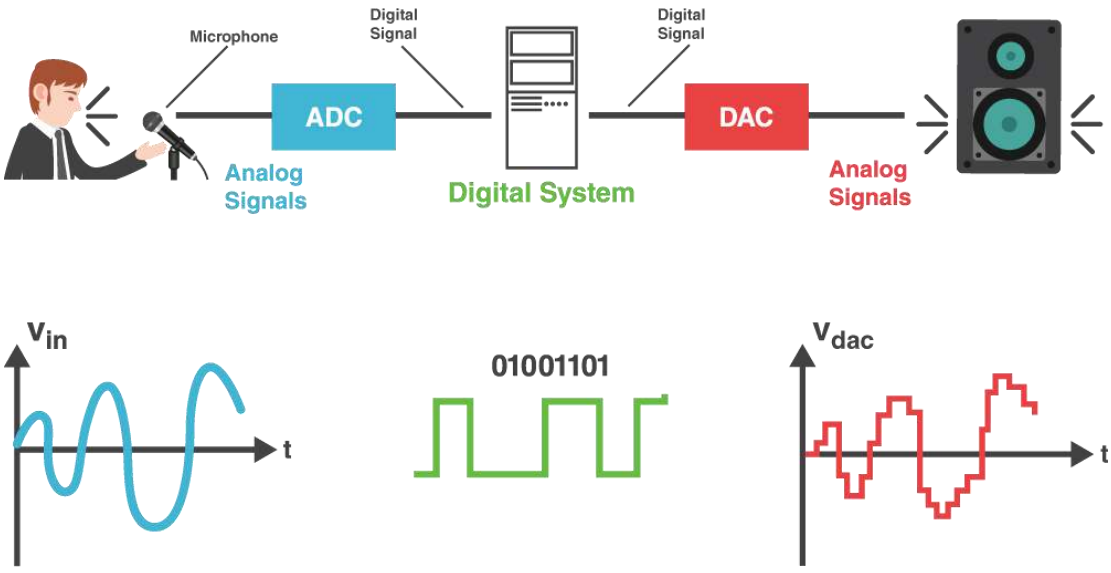
An Analog Electronic System



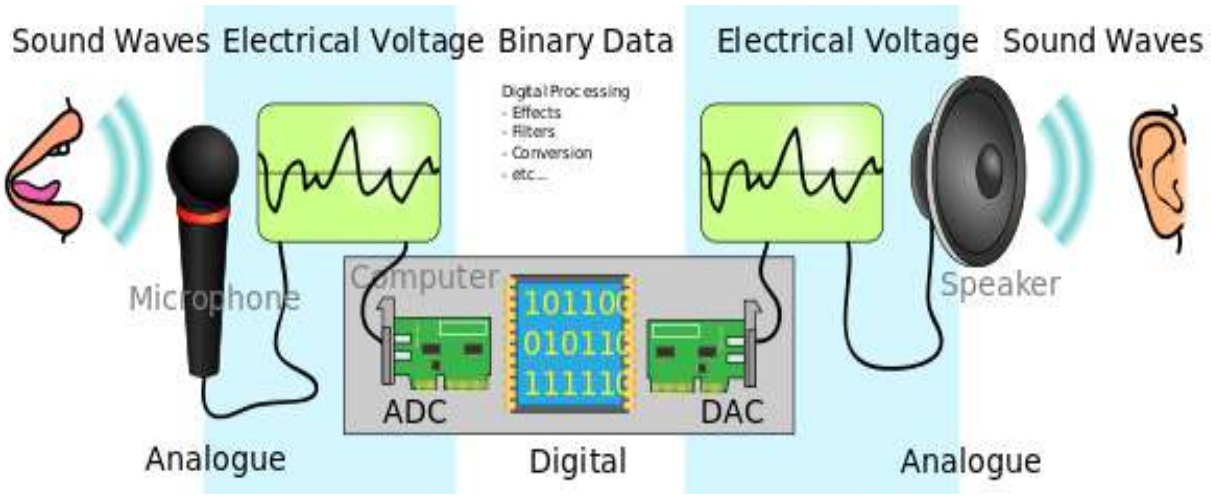
A System using both Digital And Analog Signals



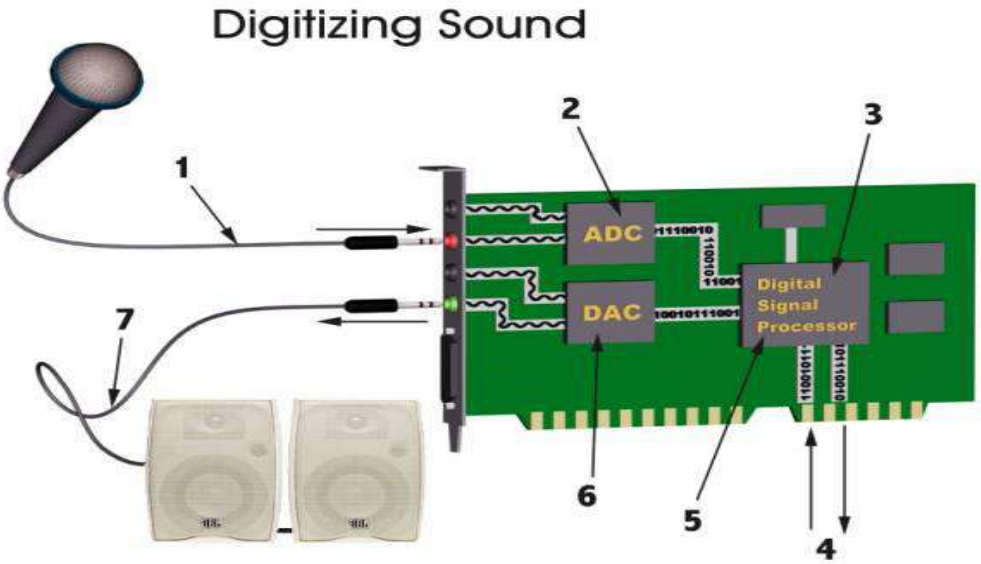
A system using Digital and Analog Methods



A system using Digital and Analog Methods



A system using Digital and Analog Methods



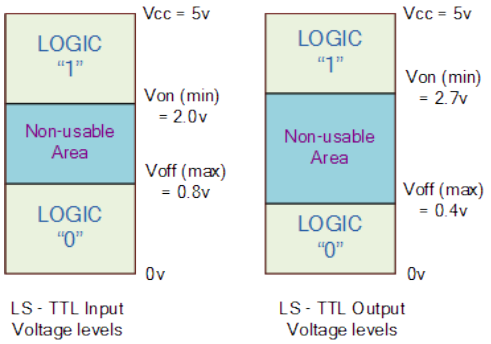
Number Systems

Digital is represented by binary numerical system instead of decimal numerical system:

- Decimal System: **0,1,2,3,4,5,6,7,8,9**
- Binary System: **0,1**
- Octal System: **0,1,2,3,4,5,6,7**
- Hexadecimal System: **0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F**

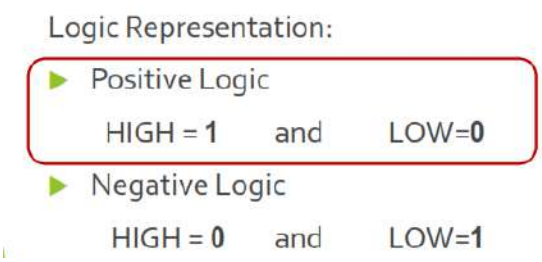
What is a Logic Level?

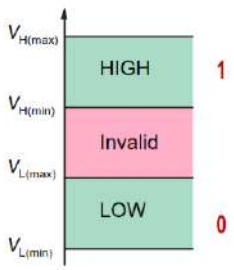
A logic level is a specific voltage or a state in which a signal can exist. We often refer to the two states in a digital circuit to be ON or OFF. Represented in binary, an ON translates to a binary 1, and an OFF translates to a binary 0.

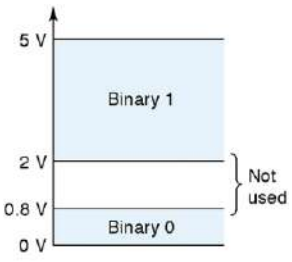


Logic 0 or Logic 1

Digital electronics rely on binary logic to store, process, and transmit data or information. Binary Logic refers to one of two states -- ON or OFF. This is commonly translated as a binary 1 or binary 0. A binary 1 is also referred to as a HIGH signal and a binary 0 is referred to as a LOW signal.

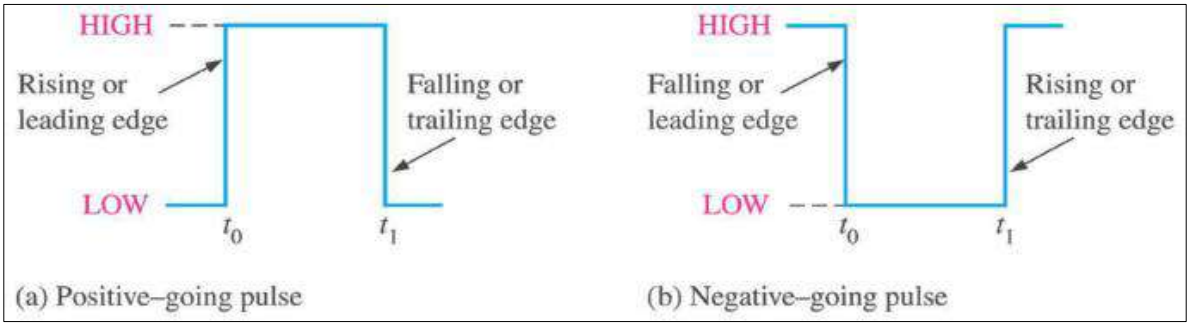






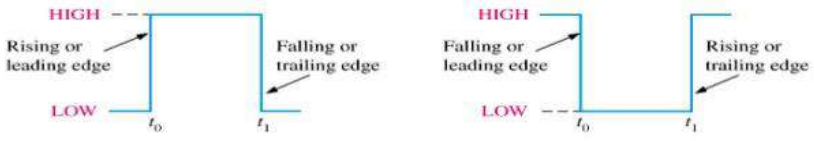
Digital Waveform (Ideal Pulse)

- Digital waveforms change between the LOW and HIGH levels.
- A positive going pulse is one that goes from a normally LOW logic level to a HIGH level and then back again.
- Digital waveforms are made up of a series of pulses.
- Ideal Pulse : the rising and falling edges are assumed to change in zero time (instantaneously).



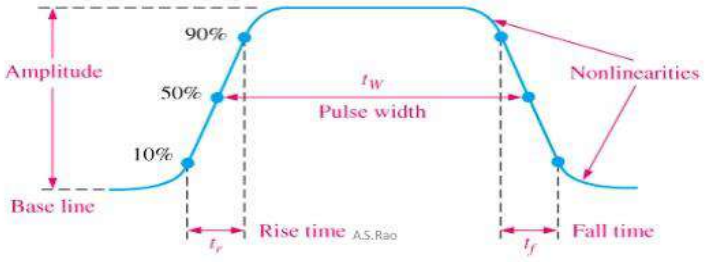
Ideal Digital Pulse

Ideal pulse shapes.

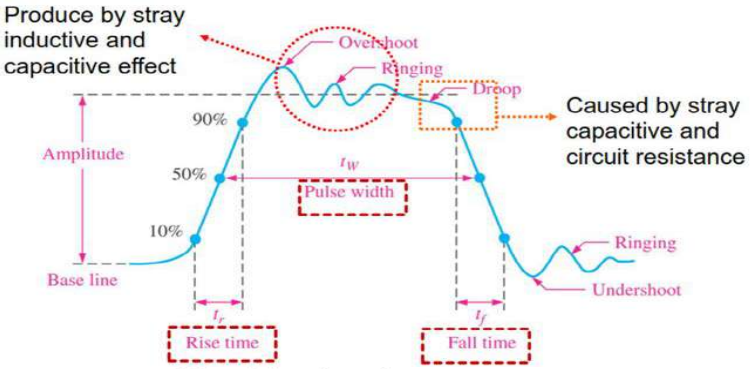


Non-ideal Digital Pulse

Non ideal pulse shape.



Digital Waveform (Nonideal Pulse)



- Important items:
  - Rise time
  - Fall time
  - Amplitude
  - Pulse width

- Non-ideal pulse
  - Real applications exhibit this characteristic
  - Overshoot and ringing – produced by stray inductive and capacitive effects
  - Droop – caused by stray capacitive and circuit resistance (forming an RC circuit)

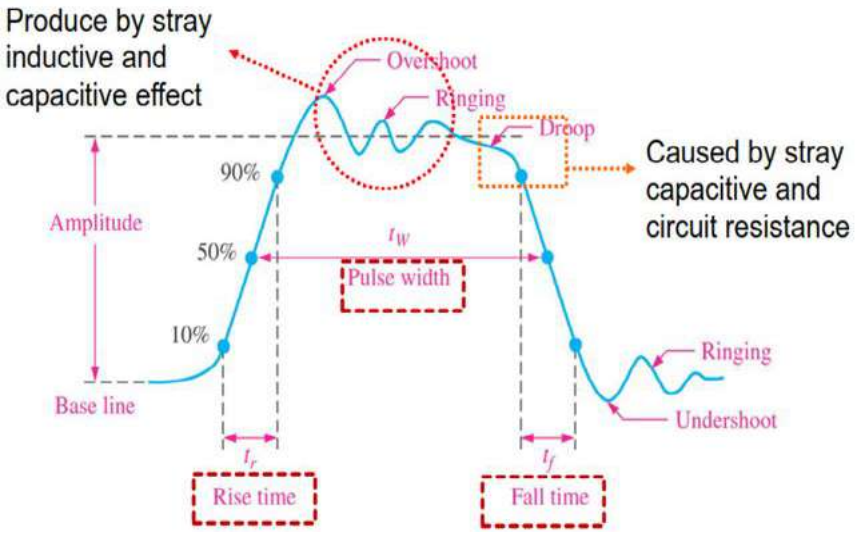
Digital Waveform (Nonideal Pulse)

Rise time, tr - time required to go from LOW to HIGH

Fall time, tf - time required to go from HIGH to LOW

Amplitude – height measured between HIGH and LOW (or vice versa). Measurement for rise and fall time usually made within 10% to 90% of pulse amplitude

Pulse width, tw -- duration of pulse, measured at 50% points on the rising and falling edges



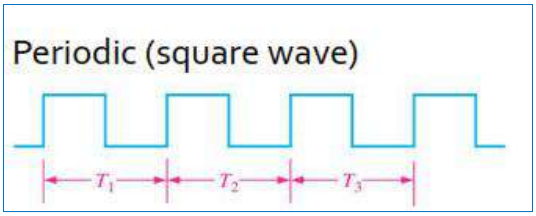
Waveform Characteristics

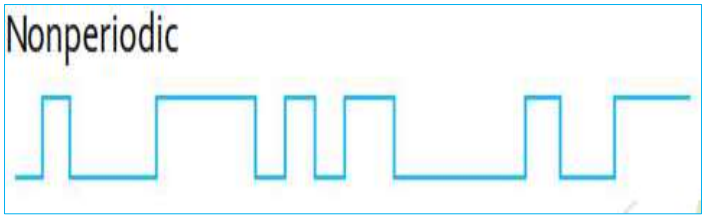
Series of pulses can be found in digital systems – called as pulse trains

• This can be further classified as periodic and nonperiodic

– Periodic – repeating the same waveform at a fixed interval, called period (T)

– Nonperiodic – opposite to periodic, where the waveform does not repeat itself at a fixed interval





Important Terms

– **Period** – total of time that a waveform repeats itself

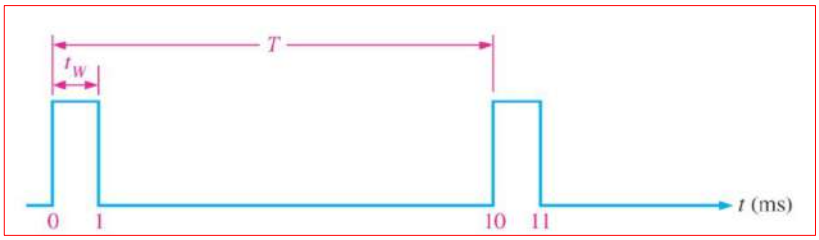
– **Frequency** – rate of how many times a waveform repeats itself

– **Duty cycle** – ratio of the pulse width to the period in percentage

Relationships between period, frequency and duty cycle

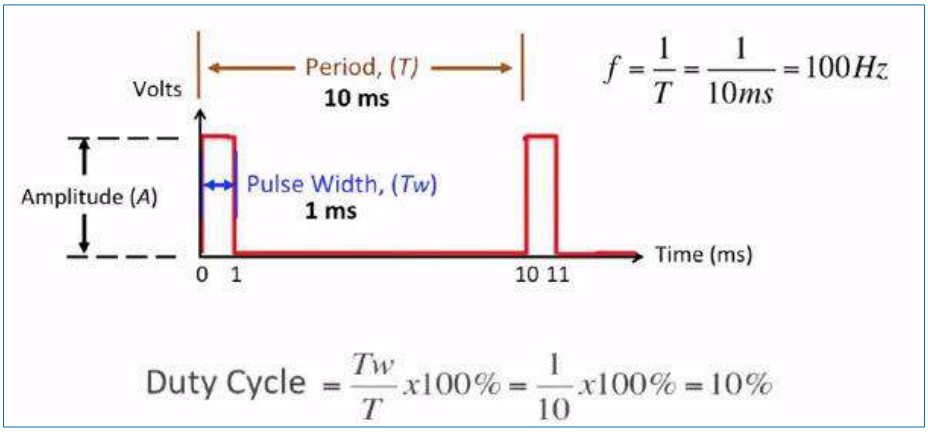
$$f = \frac{1}{T} \qquad T = \frac{1}{f}$$

$$\text{Duty cycle} = \left(\frac{t_w}{T}\right)100\%$$



Numerical

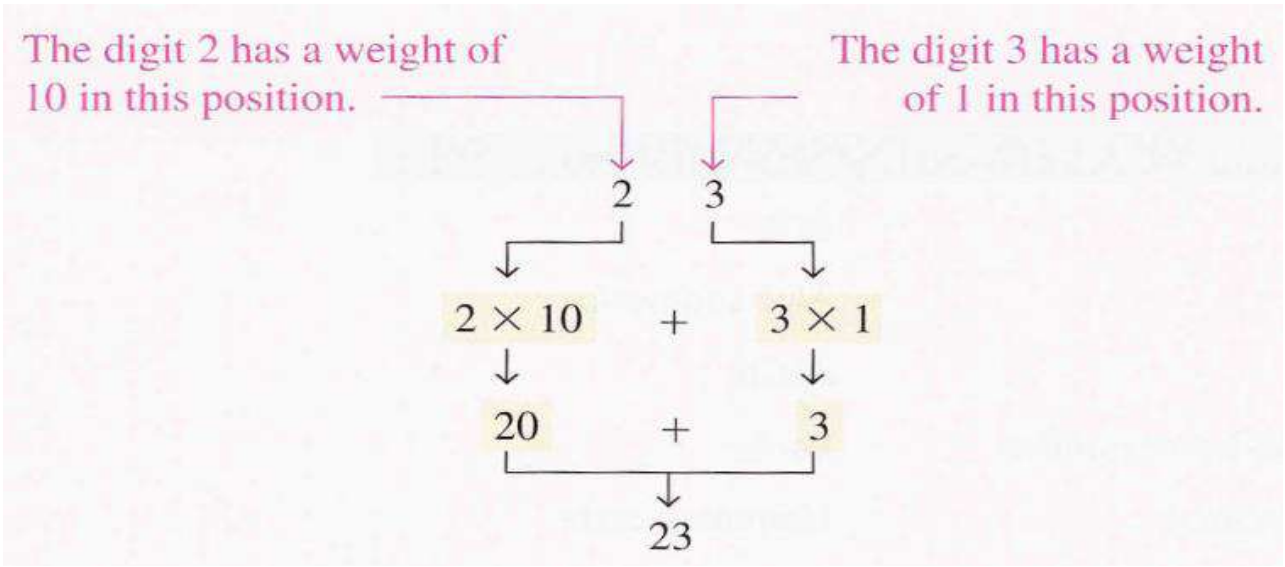
Example: Determine the following:
a) Period
b) Frequency
c) Duty Cycle



Number Systems

Decimal Numbers	Binary Numbers	Octal Numbers	Hexa-decimal Numbers
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	-	8
9	1001	-	9
10	1010	-	A
11	1011	-	B
12	1100	-	C
13	1101	-	D
14	1110	-	E
15	1111	-	F

Decimal Number System



Decimal Number System

The position of each digit in a decimal number indicates the magnitude of the quantity represented and can be assigned a **weight.** The weights for whole numbers are positive powers of ten that increase from right to left, beginning with $10^0 = 1$.

$$\ldots 10^5\ 10^4\ 10^3\ 10^2\ 10^1\ 10^0$$

For fractional numbers, the weights are negative powers of ten that decrease from left to right beginning with 10^{-1}.

$$10^2\ 10^1\ 10^0.10^{-1}\ 10^{-2}\ 10^{-3}\ldots$$

Decimal point

Decimal Number System

As an illustration, in the case of the decimal number **3586.265**,

the integer part (i.e. 3586) can be expressed as

3586 = $6\times10^{0} + 8\times10^{1} + 5\times10^{2} + 3\times10^{3} = 6+80+500+3000 = 3586$

and the fractional part can be expressed as

265 = $2\times10^{-1} + 6\times10^{-2} + 5\times10^{-3} = 0.2+0.06+0.005 = 0.265$

Question

Express the decimal number 47 as a sum of the values of each digit.

Solution

The digit 4 has a weight of 10, which is 10^1, as indicated by its position. The digit 7 has a weight of 1, which is 10^0, as indicated by its position.

$$47 = (4 \times 10^1) + (7 \times 10^0)$$
$$= (4 \times 10) + (7 \times 1) = \mathbf{40 + 7}$$

Decimal Number System

Express the decimal number 568.23 as a sum of the values of each digit.

Solution The whole number digit 5 has a weight of 100, which is 10^2, the digit 6 has a weight of 10, which is 10^1, the digit 8 has a weight of 1, which is 10^0, the fractional digit 2 has a weight of 0.1, which is 10^{-1}, and the fractional digit 3 has a weight of 0.01, which is 10^{-2}.

$$\begin{aligned} 568.23 &= (5 \times 10^2) + (6 \times 10^1) + (8 \times 10^0) + (2 \times 10^{-1}) + (3 \times 10^{-2}) \\ &= (5 \times 100) + (6 \times 10) + (8 \times 1) + (2 \times 0.1) + (3 \times 0.01) \\ &= \mathbf{500} + \mathbf{60} + \mathbf{8} + \mathbf{0.2} + \mathbf{0.03} \end{aligned}$$

Binary Number System

The binary number system is another way to represent quantities. It is less complicated than the decimal system because it has only two digits. The decimal system with its ten digits is a base-ten system; the binary system with its two digits is a base-two system. The two binary digits (bits) are 1 and 0. The position of a 1 or 0 in a binary number indicates its weight, or value within the number, just as the position of a decimal digit determines the value of that digit. The weights in a binary number are based on powers of two.

Binary Number System

Counting in Binary

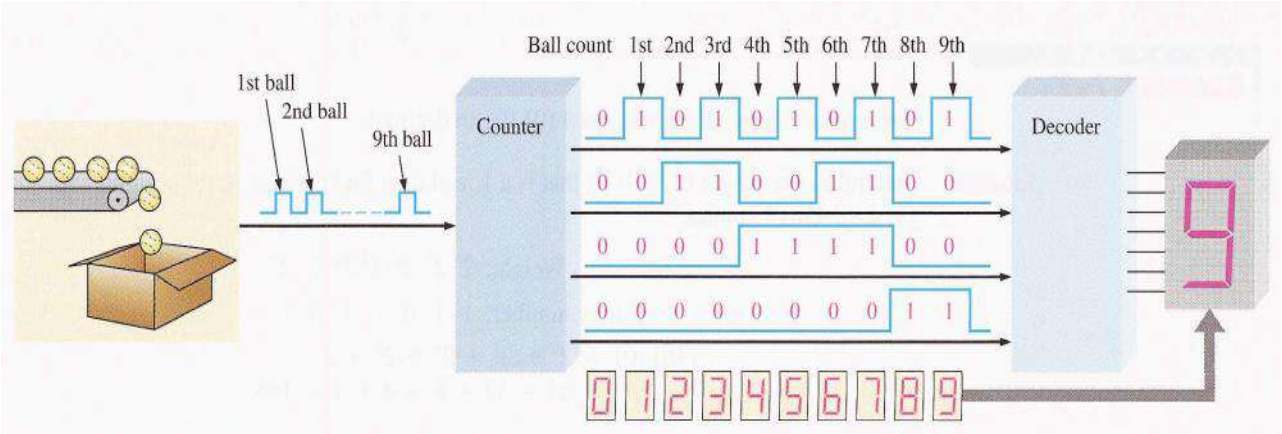
To learn to count in the binary system, first look at how you count in the decimal system. You start at zero and count up to nine before you run out of digits. You then start another digit position (to the left) and continue counting 10 through 99. At this point you have exhausted all two-digit combinations, so a third digit position is needed to count from 100 through 999.

A comparable situation occurs when you count in binary, except that you have only two digits, called *bits*. Begin counting: 0, 1. At this point you have used both digits, so include another digit position and continue: 10, 11. You have now exhausted all combinations of two digits, so a third position is required. With three digit positions you can continue to count: 100, 101, 110, and 111. Now you need a fourth digit position to continue, and so on.

Binary Number System

DECIMAL NUMBER	BINARY NUMBER			
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
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Application of Binary Counting



Binary Weights

POSITIVE POWERS OF TWO (WHOLE NUMBERS)									NEGATIVE POWERS OF TWO (FRACTIONAL NUMBER)					
2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}
256	128	64	32	16	8	4	2	1	1/2	1/4	1/8	1/16	1/32	1/64
									0.5	0.25	0.125	0.0625	0.03125	0.015625

Question

Convert the binary whole number 1101101 to decimal.

Solution Determine the weight of each bit that is a 1, and then find the sum of the weights to get the decimal number.

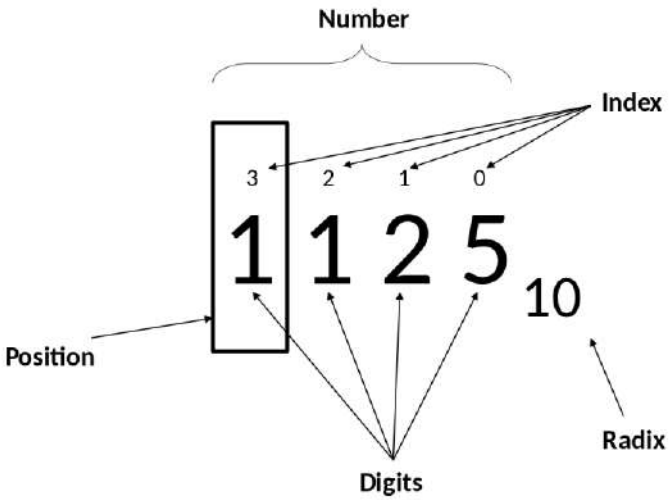
Weight: $2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

Binary number: 1 1 0 1 1 0 1

$1101101 = 2^6 + 2^5 + 2^3 + 2^2 + 2^0$

$= 64 + 32 + 8 + 4 + 1 = \mathbf{109}$

Base (Radix)



Decimal Number System

In general, the value of any mixed decimal number

$$d_n\, d_{n-1}\, d_{n-2} \ldots d_1\, d_0 \,.\, d_{-1}\, d_{-2}\, d_{-3} \ldots d_{-k}$$

is given by

$$(d_n \times 10^n) + (d_{n-1} \times 10^{n-1}) + \ldots + (d_1 \times 10^1) + (d_0 \times 10^0) + (d_{-1} \times 10^{-1}) + (d_{-2} \times 10^{-2}) + \ldots$$

Consider the decimal number 9256.26 using digits 2, 5, 6, 9. This is a mixed number. Hence,

$$9256.26 = 9 \times 1000 + 2 \times 100 + 5 \times 10 + 6 \times 1 + 2 \times (1/10) + 6 \times (1/100)$$
$$= 9 \times 10^3 + 2 \times 10^2 + 5 \times 10^1 + 6 \times 10^0 + 2 \times 10^{-1} + 6 \times 10^{-2}$$

Consider another number 6592.69, using the same digits 2, 5, 6, 9. Here,

$$6592.69 = 6 \times 10^3 + 5 \times 10^2 + 9 \times 10^1 + 2 \times 10^0 + 6 \times 10^{-1} + 9 \times 10^{-2}$$

Note the difference in position values of the same digits, when placed in different positions.

9's and 10's Complements

2.1.1 9's and 10's Complements

Subtraction of decimal numbers can be accomplished by the 9's and 10's complement methods similar to the 1's and 2's complement methods of binary. The 9's complement of a decimal number is obtained by subtracting each digit of that decimal number from 9. The 10's complement of a decimal number is obtained by adding a 1 to its 9's complement.

EXAMPLE 2.1 Find the 9's complement of the following decimal numbers.
(a) 3465 (b) 782.54 (c) 4526.075

Solution

(a)
```
  9999
- 3465
------
  6534   (9's complement of 3465)
```

(b)
```
  999.99
- 782.54
--------
  217.45   (9's complement of 782.54)
```

(c)
```
  9999.999
- 4526.075
---------
  5473.924   (9's complement of 4526.075)
```

9's and 10's Complements

EXAMPLE 2.2 Find the 10's complement of the following decimal numbers.
(a) 4069 (b) 1056.074

Solution

(a)
```
  9 9 9 9
- 4 0 6 9
---------
  5 9 3 0   (9's complement of 4069)
+       1     Add 1
---------
  5 9 3 1   (10's complement of 4069)
```

(b)
```
  9 9 9 9 . 9 9 9
- 1 0 5 6 . 0 7 4
-----------------
  8 9 4 3 . 9 2 5   (9's complement of 1056.074)
+               1     Add 1
-----------------
  8 9 4 3 . 9 2 6   (10's complement of 1056.074)
```

9's Complement Method of Subtraction

To perform decimal subtraction using the 9's complement method, obtain the 9's complement of the subtrahend and add it to the minuend. Call this the intermediate result.

- **If there is carry,** it indicates that the answer is positive. Add the carry to the LSD of this result to get the answer. This is called end around carry.

- **If there is no carry,** it indicates that the answer is negative and the intermediate result is its 9's complement. Take the 9's complement of this result and place a negative sign in front to get the answer.

EXAMPLE 2.3 Subtract the following numbers using the 9's complement method.
(a) 745.81 – 436.62 (b) 436.62 – 745.81

Solution

(a)

```
  7 4 5 . 8 1                  7 4 5 . 8 1
- 4 3 6 . 6 2      ⇒        + 5 6 3 . 3 7   (9's complement of 436.62)
-------------              -------------
  3 0 9 . 1 9              ❶ 3 0 9 . 1 8   (Intermediate result)
                             ↳       + 1   (End around carry)
                           -------------
                             3 0 9 . 1 9   (Answer)
```

The carry indicates that the answer is positive. So the answer is +309.19.

(b)

```
  4 3 6 . 6 2                  4 3 6 . 6 2
- 7 4 5 . 8 1      ⇒        + 2 5 4 . 1 8   (9's complement of 745.81)
-------------              -------------
- 3 0 9 . 1 9                6 9 0 . 8 0   (Intermediate result with no carry)
```

There is no carry indicating that the answer is negative. So, take the 9's complement of the intermediate result and put a minus sign. The 9's complement of 690.80 is 309.19. Therefore, the answer is –309.19.

10's Complement Method of Subtraction

To perform decimal subtraction using the 10's complement method, obtain the 10's complement of the subtrahend and add it to the minuend.

- **If there is a carry**, ignore it. The presence of the carry indicates that the answer is positive; the result obtained is itself the answer.

- **If there is no carry**, it indicates that the answer is negative and the result obtained is its 10's complement. Obtain the 10's complement of the result and place a negative sign in front to get the answer.

EXAMPLE 2.4 Subtract the following numbers using the 10's complement method.
(a) 2928.54 – 416.73 (b) 416.73 – 2928.54

Solution

(a)
$$\begin{array}{r} 2928.54 \\ -0416.73 \\ \hline 2511.81 \end{array} \quad \Rightarrow \quad \begin{array}{r} 2928.54 \\ +9583.27 \\ \hline \textbf{1}\,2511.81 \end{array}$$
(10's complement of 416.73)
(Ignore the carry)

There is a carry indicating that the answer is positive. Ignore the carry.
The answer is 2511.81.

(b)
$$\begin{array}{r} 0416.73 \\ -2928.54 \\ \hline -2511.81 \end{array} \quad \Rightarrow \quad \begin{array}{r} 0416.73 \\ +7071.46 \\ \hline 7488.19 \end{array}$$
(10's complement of 2928.54)
(No carry)

There is no carry indicating that the answer is negative. So, take the 10's complement of the intermediate result and put a minus sign. The 10's complement of 7488.19 is 2511.81. Therefore, the answer is –2511.81.

Counting in binary

Decimal number	Binary number	Decimal number	Binary number
0	0	20	10100
1	1	21	10101
2	10	22	10110
3	11	23	10111
4	100	24	11000
5	101	25	11001
6	110	26	11010
7	111	27	11011
8	1000	28	11100
9	1001	29	11101
10	1010	30	11110
11	1011	31	11111
12	1100	32	100000
13	1101	33	100001
14	1110	34	100010
15	1111	35	100011
16	10000	36	100100
17	10001	37	100101
18	10010	38	100110
19	10011	39	100111

Binary to Decimal Conversion

Binary numbers may be converted to their decimal equivalents by the positional weights method.

In this method, each binary digit of the number is multiplied by its position weight and the product terms are added to obtain the decimal number.

EXAMPLE 2.5 Convert 10101_2 to decimal.

Solution

(Positional weights) $2^4\ 2^3\ 2^2\ 2^1\ 2^0$

(Binary number) $1\ 0\ 1\ 0\ 1 = (1 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0)$

$= 16 + 0 + 4 + 0 + 1$

$= 21_{10}$

Binary to Decimal Conversion

EXAMPLE 2.6 Convert 11011.101_2 to decimal.

Solution

$2^4\ 2^3\ 2^2\ 2^1\ 2^0\ 2^{-1}\ 2^{-2}\ 2^{-3}$

$1\ 1\ 0\ 1\ 1 \cdot 1\ 0\ 1 = (1 \times 2^4) + (1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0)$

$+ (1 \times 2^{-1}) + (0 \times 2^{-2}) + (1 \times 2^{-3})$

$= 16 + 8 + 0 + 2 + 1 + 0.5 + 0 + 0.125$

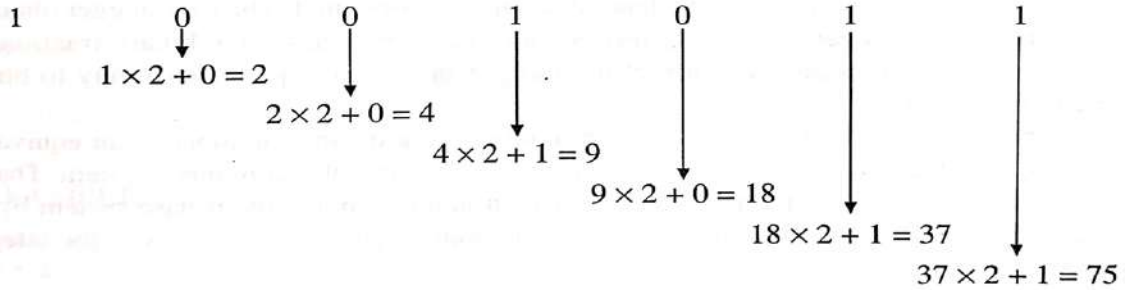
$= 27.625_{10}$

Binary to Decimal Conversion

EXAMPLE 2.7 Convert 1001011_2 to decimal.

Solution

An integer binary number can also be converted to an integer decimal number as follows.
Starting with the extreme left bit, i.e. MSB, multiply this bit by 2 and add the product to the next bit to the right.
Multiply the result obtained in the previous step by 2 and add the product to the next bit to the right.
Continue this process as shown below till all the bits in the number are exhausted.



Decimal to Binary Conversion

EXAMPLE 2.9 Convert 52_{10} to binary using the double-dabble method.

Solution

We divide the given decimal number successively by 2 and read the remainders upwards to get the equivalent binary number.

(double-dabble)

Successive division		*Remainder*
2	52	0
2	26	0
2	13	1
2	6	0
2	3	1
2	1	1
	0	

Reading the remainders from bottom to top, the result is $52_{10} = 110100_2$.

Decimal to Binary Conversion

EXAMPLE 2.10 Convert 0.75_{10} to binary using the double-dabble method.

Solution

Multiply the given fraction by 2. Keep the integer in the product as it is and multiply the new fraction in the product by 2. Continue this process and read the integers in the products from top to bottom.

Given fraction		0.75
Multiply 0.75 by 2	|	1.50
Multiply 0.50 by 2	↓	1.00

Reading the integers from top to bottom, $0.75_{10} = 0.11_2$.

EXAMPLE 2.11 Convert 105.15_{10} to binary.

Solution

Conversion of integer 105

Successive division		*Remainder*
2 | 105		
2 | 52		1
2 | 26		0
2 | 13		0
2 | 6		1
2 | 3		0
2 | 1		1
0		1

Reading the remainders from bottom to top, $105_{10} = 1101001_2$.

Conversion of fraction 0.15_{10}

Given fraction		0.15
Multiply 0.15 by 2	|	0.30
Multiply 0.30 by 2	|	0.60
Multiply 0.60 by 2	|	1.20
Multiply 0.20 by 2	↓	0.40
Multiply 0.40 by 2		0.80
Multiply 0.80 by 2		1.60

Reading the integers from top to bottom, $0.15_{10} = 0.001001_2$.

Therefore, the final result is, $105.15_{10} = 1101001.001001_2$.

Binary Arithmetic's

1. **Binary Addition**

The rules for binary addition are the following:

0 + 0 = 0;

0 + 1 = 1;

1 + 0 = 1 ;

1 + 1 = 10, i.e. 0 with a carry of 1

Binary Addition

EXAMPLE 2.12 Add the binary numbers 1101.101 and 111.011.

Solution

	8	4	2	1		2^{-1}	2^{-2}	2^{-3}	(Column numbers)
	1	1	0	1	·	1	0	1	
+		1	1	1	·	0	1	1	
	1	0	1	0	1	·	0	0	0

Binary Arithmetic's

2. Binary Subtraction

The rules for binary subtraction are the following:

0 - 0 = 0;

1 - 1 = 0 ;

1 - 0 = 1 ;

0 - 1 = 1, with a borrow of 1.

Binary Subtraction

EXAMPLE 2.13 Subtract 111.111_2 from 1010.01_2.

Solution

8	4	2	1		2^{-1}	2^{-2}	2^{-3}	(Column numbers)
1	0	1	0	.	0	1	0	
		1	1	.	1	1	1	
0	0	1	0	.	0	1	1	

Binary Arithmetic's

3. Binary Multiplication

The rules for binary multiplication are the following:

0 x 0 = 0;

1 x 1 = 1;

1 x 0 = 0;

0 x 1 = 0

Binary Multiplication

EXAMPLE 2.14 Multiply 1101_2 by 110_2.

Solution

The LSB of the multiplier is a 0. So, the first partial product is a 0. The next two bits of the multiplier are 1s. So, the next two partial products are equal to the multiplicand itself. The sum of the partial products gives the answer.

```
    1 1 0 1
  × 1 1 0
  ---------
    0 0 0 0
  1 1 0 1
1 1 0 1
---------
1 0 0 1 1 1 0
```

Binary Multiplication

EXAMPLE 2.15 Multiply 1011.101_2 by 101.01_2.

Solution

```
      1 0 1 1 . 1 0 1
    × 1 0 1 . 0 1
-----------------------
      1 0 1 1 1 0 1
    0 0 0 0 0 0 0
  1 0 1 1 1 0 1
0 0 0 0 0 0 0
1 0 1 1 1 0 1
-----------------------
1 1 1 1 0 1.0 0 0 0 1
```

Binary Arithmetic's

4. Binary Division

EXAMPLE 2.17 Divide 101101_2 by 110_2.

```
1 1 0 ) 1 0 1 1 0 1 ( 1 1 1 . 1
        1 1 0
      -------
        1 0 1 0
          1 1 0
        -------
          1 0 0 1
            1 1 0
          -------
              1 1 0
              1 1 0
          -------
                0 0 0
```

Therefore, $101101 \div 110 = 111.1$

Binary Division

EXAMPLE 2.18 Divide 110101.11_2 by 101_2.

Solution

```
      1 0 1 ) 1 1 0 1 0 1 . 1 1 ( 1 0 1 0 . 1 1
              1 0 1
              -----
              0 1 1 0
                1 0 1
                -----
                  1 1 1
                  1 0 1
                  -----
                    1 0 1
                    1 0 1
                    -----
                      0 0 0
```

Therefore, $110101.11 \div 101 = 1010.11$

1's Complement

Method of obtaining the 1's complement of a number:

- The 1's complement of a number is obtained by simply complementing each bit of the number, that is, by changing all the 0's to 1's and all the 1's to 0's.
- We can also say that the 1's complement of a number is obtained by subtracting each bit of the number from 1. This complemented value represents the negative of the original number.
- This system is very easy to implement in hardware by simply feeding all bits through inverters.
- One of the difficulties of using 1's complements its representation of zero. Both 00000000 and its 1's complement 11111111 represent zero (The 00000000 is called positive zero and the 11111111 is called negative zero)

1's Complement

EXAMPLE 2.30 Represent – 99 and – 77.25 in 8-bit 1's complement form.

Solution

We first write the positive representation of the given number in binary form and then complement each of its bits to represent the negative of the number.

(a) $+99 = 01100011$

$-99 = 10011100$ (In 1's complement form)

(b) $+77.25 = 01001101.0100$

$-77.25 = 10110010.1011$ (In 1's complement form)

2's Complement

Method of obtaining the 2's complement of a number:

- The 2's complement of a number is obtained by simply obtaining the 1's complement of the number and then adding a 1 to the LSB.
- Starting at the LSB, copying down each bit up to and including the first 1 bit encountered, and complementing the remaining bits.

2's Complement

EXAMPLE 2.21 Express –45 in 8-bit 2's complement form.

Solution

+45 in 8-bit form is 00101101.

First method

Obtain the 1's complement of 00101101 and then add 1.

Positive expression of the given number	0 0 1 0 1 1 0 1
1's complement of it	1 1 0 1 0 0 1 0
Add 1	+ 1
Thus, the 2's complement form of – 45 is	1 1 0 1 0 0 1 1

Second method

Copy down the bits starting from LSB up to and including the first 1, and then complement the remaining bits.

Original number	0 0 1 0 1 1 0 1
Copy up to the first 1 bit	1
Complement the remaining bits	1 1 0 1 0 0 1
Thus, the 2's complement form of – 45 is	1 1 0 1 0 0 1 1

1's Complement Arithmetic Rule

In l 's complement subtraction, add the l 's complement of the subtrahend to the minuend.

- If there is a carry out, bring the carry around and add it to the LSB. This is called the end around carry. Look at the sign bit (MSB). If this is a 0, the result is positive and is in true binary. If the MSB is a 1 (whether there is a carry or no carry at all), the result is negative and is in its 1 s complement form. Take its l 's complement to get the magnitude in binary.

1's Complement Arithmetic

EXAMPLE 2.31 Subtract 14 from 25 using the 8-bit 1's complement arithmetic.

Solution

```
  25          00011001
 -14    ⇒    +11110001      (In 1's complement form)
 +11         (1)00010 10
                    +1      (End around carry)
             00010 11
```

The MSB is a 0. So, the result is positive and is in pure binary. Therefore, the result is, $00001011 = +11_{10}$.

1's Complement Arithmetic

EXAMPLE 2.32 Add – 25 to + 14 using the 8-bit 1's complement method.

Solution

```
 +14          00001110.
 -25    ⇒    +11100110      (In 1's complement form)
 -11          11110100      (No carry)
```

There is no carry. The MSB is a 1. So, the result is negative and is in its 1's complement form. The 1's complement of 11110100 is 00001011. The result is, therefore, -11_{10}.

2's Complement Arithmetic Rule

In the 2's complement subtraction, add the 2's complement of the subtrahend to the minuend.

- If there is a carry out, ignore it. Look at the sign bit, i.e. MSB of the sum term. If the MSB is a 0, the result is positive and is in true binary form. If the MSB is a 1 (whether there is a carry or no carry at all) the result is negative and is in its 2's complement form. Take its 2 s complement to find its magnitude in binary.

2's Complement Arithmetic

EXAMPLE 2.23 Subtract 14 from 46 using the 8-bit 2's complement arithmetic.

Solution

+14	=	0 0 0 0 1 1 1 0	
−14	=	1 1 1 1 0 0 1 0	(In 2's complement form)
+46		0 0 1 0 1 1 1 0	
−14	⇒	+ 1 1 1 1 0 0 1 0	(2's complement form of – 14)
+32		❶ 0 0 1 0 0 0 0 0	(Ignore the carry)

There is a carry, ignore it. The MSB is 0. So, the result is positive and is in normal binary form. Therefore, the result is + 00100000 = + 32.

2's Complement Arithmetic

EXAMPLE 2.24 Add – 75 to + 26 using the 8-bit 2's complement arithmetic.

Solution

+75	=	01001011	
–75	=	10110101	(In 2's complement form)
+26		00011010	
–75	⇒	+10110101	(2's complement form of – 75)
–49		11001111	(No carry)

There is no carry, the MSB is a 1. So, the result is negative and is in 2's complement form. The magnitude is 2's complement of 11001111, that is, 00110001 = 49. Therefore, the result is – 49.

Octal to Binary Conversion

- To convert a given octal number to a binary, just replace each octal digit by its 3-bit binary equivalent.

EXAMPLE 2.42 Convert 367.52_8 to binary.

Solution

Given octal number is	3	6	7	.	5	2
Convert each octal digit to binary	011	110	111	.	101	010
The result is	$011110111 . 101010_2$					

Binary to Octal Conversion

- To convert a binary number to an octal number, starting from the binary point make groups of 3 bits each, on either side of the binary point, and replace each 3-bit binary group by the equivalent octal digit.

EXAMPLE 2.43 Convert 110101.101010_2 to octal.

Solution

Groups of three bits are	110	101	.	101	010	
Convert each group to octal	6	5	.	5	2	
The result is			65.52_8			

EXAMPLE 2.44 Convert 10101111001.0111_2 to octal.

Solution

Groups of three bits are		10	101	111	001	. 011	1
	=	010	101	111	001	. 011	100
Convert each group into octal		2	5	7	1	. 3	4
The result is					2571.34_8		

Octal to Decimal Conversion

To convert an octal number to a decimal number, multiply each digit in the octal number by the weight of its position and add all the product terms.

The decimal value of the octal number $d_n d_{n-1} d_{n-2} \ldots d_1 d_0 \cdot d_{-1} d_{-2} \ldots d_{-k}$ is

$$(d_n \times 8^n) + (d_{n-1} \times 8^{n-1}) + \ldots + (d_1 \times 8^1) + (d_0 \times 8^0) + (d_{-1} \times 8^{-1}) + (d_{-2} \times 8^{-2}) + \ldots$$

EXAMPLE 2.45 Convert 4057.06_8 to decimal.

Solution

$$\begin{aligned} 4057.06_8 &= 4 \times 8^3 + 0 \times 8^2 + 5 \times 8^1 + 7 \times 8^0 + 0 \times 8^{-1} + 6 \times 8^{-2} \\ &= 2048 + 0 + 40 + 7 + 0 + 0.0937 \\ &= 2095.0937_{10} \end{aligned}$$

Decimal to Octal Conversion

EXAMPLE 2.46 Convert 378.93_{10} to octal.

Solution

Conversion of 378_{10} *to octal*

Successive division		*Remainders*
8	378	
8	47	↑ 2
8	5	| 7
	0	| 5

Read the remainders from bottom to top. Therefore, $378_{10} = 572_8$.

Conversion of 0.93_{10} *to octal*

0.93×8	| 7.44
0.44×8	| 3.52
0.52×8	↓ 4.16
0.16×8	1.28

Read the integers to the left of the octal point downwards. Therefore, $0.93_{10} = 0.7341_8$. Hence $378.93_{10} = 572.7341_8$.

Decimal to Binary via Octal Conversion

EXAMPLE 2.47 Convert 5497_{10} to binary.

Solution

Since the given decimal number is large, we first convert this number to octal and then convert the octal number to binary.

Successive division		*Remainders*
8	5497	
8	687	↑ .1
8	85	| 7
8	10	| 5
8	1	| 2
	0	1

Therefore, $5497_{10} = 12571_8 = 001010101111001_2$.

Binary to Decimal via Octal Conversion

EXAMPLE 2.48 Convert 101111010001_2 to decimal.

Solution

Since the given binary number is large, we first convert this number to octal and then convert the octal number to decimal.

$101111010001_2 = 5721_8 = 5 \times 8^3 + 7 \times 8^2 + 2 \times 8^1 + 1 \times 8^0 = 2560 + 448 + 16 + 1 = 3025_{10}$.

Octal Arithmetic (Addition)

EXAMPLE 2.49 Add $(27.5)_8$ and $(74.4)_8$.

Solution

$$\begin{array}{rcr} 27.5_8 & = & 010\ \ 111 \cdot 101_2 \\ +74.4_8 & = & +111\ \ 100 \cdot 100_2 \\ \hline 124.1_8 & = & 1010\ \ 100 \cdot 001 \end{array}$$

Octal Arithmetic (Subtraction)

EXAMPLE 2.50 Subtract (a) 45_8 from 66_8 and (b) 73_8 from 25_8.

Solution

Using 8-bit representation and 2's complement method

(a)	66_8	=	0 0	1 1 0	1 1 0_2	
	-45_8	=	+ 1 1	0 1 1	0 1 1_2	(2's complement of -45_8)
	21_8		❶ 0 0	0 1 0	0 0 1_2	(Ignore the carry. Answer is positive)
(b)	25_8	=	0 0	0 1 0	1 0 1_2	
	-73_8	=	+ 1 1	0 0 0	1 0 1_2	(2's complement of -73_8)
	-46_8		1 1	0 1 1	0 1 0_2	(Answer is negative)

2's complement of $1\,1\,0\,1\,1\,0\,1\,0_2 = -0\,0\,1\,0\,0\,1\,1\,0 = -48_8$

Binary to Hexadecimal Conversion

EXAMPLE 2.51 Convert 1011011011_2 to hexadecimal.

Solution

Make groups of 4 bits, and replace each 4-bit group by a hex digit.

Given binary number is	1011011011
Groups of four bits are	0010 1101 1011
Convert each group to hex	2 D B
The result is	$2DB_{16}$

Binary to Hexadecimal Conversion

EXAMPLE 2.52 Convert 01011111011.011111_2 to hexadecimal.

Solution

Given binary number is					01011111011.011111					
Groups of four bits are	0010		1111		1011	.	0111		1100	
Convert each group to hex	2		F		B	.	7		C	
The result is					$2FB.7C_{16}$					

Hexadecimal to Binary Conversion

To convert a hexadecimal number to binary, replace each hex digit by its 4-bit binary group.

EXAMPLE 2.53 Convert $4BAC_{16}$ to binary.

Solution

Given hex number is	4	B	A	C
Convert each hex digit to 4-bit binary	0100	1011	1010	1100
The result is	0100101110101100_2			

EXAMPLE 2.54 Convert $3A9E.B0D_{16}$ to binary.

Solution

Given hex number is	3		A		9		E	.	B	0	D
Convert each hex digit to 4-bit binary	0011		1010		1001		1110	.	1011	0000	1101
The result is	$0011101010011110.101100001101_2$										

Hexadecimal to Decimal Conversion

EXAMPLE 2.55 Convert $5C7_{16}$ to decimal.

Solution

Multiply each digit of 5C7 by its position weight and add the product terms.

$$\begin{aligned} 5C7_{16} &= (5 \times 16^2) + (12 \times 16^1) + (7 \times 16^0) \\ &= 1280 + 192 + 7 \\ &= 1479_{10} \end{aligned}$$

Hexadecimal to Decimal Conversion

EXAMPLE 2.56 Convert $A0F9.0EB_{16}$ to decimal.

Solution

$$\begin{aligned} A0F9.0EB_{16} &= (10 \times 16^3) + (0 \times 16^2) + (15 \times 16^1) \\ &\quad + (9 \times 16^0) + (0 \times 16^{-1}) + (14 \times 16^{-2}) + (11 \times 16^{-3}) \\ &= 40960 + 0 + 240 + 9 + 0 + 0.0546 + 0.0026 \\ &= 41209.0572_{10} \end{aligned}$$

Decimal to Hexadecimal Conversion

EXAMPLE 2.57 Convert 2598.675_{10} to hex.

Solution

The given decimal number is a mixed number. Convert the integer and the fraction parts separately to hex.

Conversion of 2598_{10}

Successive division		*Remainder*	
		Decimal	*Hex*
16	2598		
16	162	6	↑ 6
16	10	2	| 2
	0	10	| A

Reading the remainders upwards, $2598_{10} = A26_{16}$.

Decimal to Hexadecimal Conversion

Conversion of 0.675_{10}

Given fraction is 0.675

0.675×16	| 10.8
0.800×16	| 12.8
0.800×16	| 12.8
0.800×16	↓ 12.8

Reading the integers to the left of hexadecimal point downwards, $0.675_{10} = 0.ACCC_{16}$.

Therefore, $2598.675_{10} = A26.ACCC_{16}$.

Octal to Hexadecimal Conversion

To convert an octal number to hexadecimal, the simplest way is to first convert the given octal number to binary and then the binary number to hexadecimal.

EXAMPLE 2.60 Convert 756.603_8 to hex.

Solution

Given octal number is	7	5	6	.	6	0	3
Convert each octal digit to binary	111	101	110	.	110	000	011
Groups of four bits are	0001	1110	1110	.	1100	0001	1000
Convert each four-bit group to hex	1	E	E	.	C	1	8
The result is	$1EE.C18_{16}$						

Hexadecimal to Octal Conversion

To convert a hexadecimal number to octal, the simplest way is to first convert the given hexadecimal number to binary and then the binary number to octal.

EXAMPLE 2.61 Convert $B9F.AE_{16}$ to octal.

Solution

Given hex number is		B	9	F	.	A	E	
Convert each hex digit to binary		1011	1001	1111	.	1010	1110	
Groups of three bits are	101	110	011	111	.	101	011	100
Convert each three-bit group to octal	5	6	3	7	.	5	3	4
The result is	5637.534							

Hexadecimal Arithmetic

EXAMPLE 2.62 (a) Add $6E_{16}$ and $C5_{16}$, (b) subtract $7B_{16}$ from $C4_{16}$ and (c) subtract $5D_{16}$ from $3A_{16}$.

Solution

(a)

$$\begin{array}{rcl} 6E_{16} & = & 0\,1\,1\,0 \quad 1\,1\,1\,0_2 \\ +\ C5_{16} & = & +1\,1\,0\,0 \quad 0\,1\,0\,1_2 \\ \hline 133_{16} & & 1\,0\,0\,1\,1 \quad 0\,0\,1\,1_2 \end{array}$$

(b)

$$\begin{array}{rcll} C4_{16} & = & 1\,1\,0\,0 \quad 0\,1\,0\,0_2 & \\ -\ 7B_{16} & = & +1\,0\,0\,0 \quad 0\,1\,0\,1_2 & (2\text{'s complement form of } -7B_{16}) \\ \hline 49_{16} & = & \textcircled{1}\,0\,1\,0\,0 \quad 1\,0\,0\,1_2 & \text{(Ignore the carry. Answer is positive)} \end{array}$$

(c)

$$\begin{array}{rcll} 3A_{16} & = & 0\,0\,1\,1 \quad 1\,0\,1\,0_2 & \\ -5D_{16} & = & +1\,0\,1\,0 \quad 0\,0\,1\,1_2 & (2\text{'s complement of } -5D_{16}) \\ \hline -23_{16} & & 1\,1\,0\,1 \quad 1\,1\,0\,1_2 & \text{(No carry. Answer is negative)} \end{array}$$

2's complement of $1\,1\,0\,1\,1\,1\,0\,1_2 = -0\,0\,1\,0\,0\,0\,1\,1 = -23_{16}$

Convert given radix to decimal and then to binary

EXAMPLE 2.63 Convert the following numbers with the given radix to decimal and then to binary.

(a) 4433_5 (b) 1199_{12} (c) 5654_7 (d) 1221_3

Solution

To convert the given number in any number system into decimal use sum of weights method. The obtained decimal number can be converted into binary by sum of weights method or by successive division by 2 method.

(a) $4433_5 = 4 \times 5^3 + 4 \times 5^2 + 3 \times 5^1 + 3 \times 5^0$

$= 4 \times 125 + 4 \times 25 + 3 \times 5 + 3 \times 1 = 618_{10}$

$618_{10} = 512 + 0 + 0 + 64 + 32 + 0 + 8 + 0 + 2 + 0 = 1\,0\,01\,1\,0\,1\,0\,1\,0_2$

(b) $1199_{12} = 1 \times 12^3 + 1 \times 12^2 + 9 \times 12^1 + 9 \times 12^0$

$= 1 \times 1728 + 1 \times 144 + 9 \times 12 + 9 \times 1 = 1989_{10}$

$1989_{10} = 1024 + 512 + 256 + 128 + 64 + 0 + 0 + 0 + 4 + 0 + 1$

$= 1\,1\,1\,1\,1\,0\,0\,0\,1\,0\,1_2$

Convert given radix to decimal and then to binary

(c) $5654_7 = 5 \times 7^3 + 6 \times 7^2 + 5 \times 7^1 + 4 \times 7^0$

$= 5 \times 343 + 6 \times 49 + 5 \times 7 + 4 \times 1 = 2048_{10}$

$2048_{10} = 2048 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0$

$= 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0_2$

(d) $1221_3 = 1 \times 3^3 + 2 \times 3^2 + 2 \times 3^1 + 1 \times 3^0$

$= 1 \times 27 + 2 \times 9 + 2 \times 3 + 1 \times 1 = 52_{10}$

$52_{10} = 32 + 16 + 0 + 4 + 0 + 0 = 1\ 1\ 0\ 1\ 0\ 0_2$

Counting in Another Number System

EXAMPLE 2.64 List the first 20 numbers in base 3, base 5, base 7 and base 12 systems.

Solution

The first 20 numbers in different bases are given in Table 2.5.

Table 2.5 First 20 numbers in different bases

Decimal	Base 3	Base 5	Base 7	Base 12	Decimal	Base 3	Base 5	Base 7	Base 12
0	0	0	0	0	10	101	20	13	A
1	1	1	1	1	11	102	21	14	B
2	2	2	2	2	12	110	22	15	10
3	10	3	3	3	13	111	23	16	11
4	11	4	4	4	14	112	24	20	12
5	12	10	5	5	15	120	30	21	13
6	20	11	6	6	16	121	31	22	14
7	21	12	10	7	17	122	32	23	15
8	22	13	11	8	18	200	33	24	16
9	100	14	12	9	19	201	34	25	17

Finding the Radix/Base of a given number

EXAMPLE 2.66 Given that $16_{10} = 100_b$, find the value of b.

Solution

Given $16_{10} = 100_b$.

Convert 100_b to decimal.

Therefore, $16 = 1 \times b^2 + 0 \times b^1 + 0 \times b^0 = b^2$

or $b = 4$.

Finding the Radix/Base of a given number

EXAMPLE 2.67 Given that $292_{10} = 1204$ in some number system, find the base of that system.

Solution

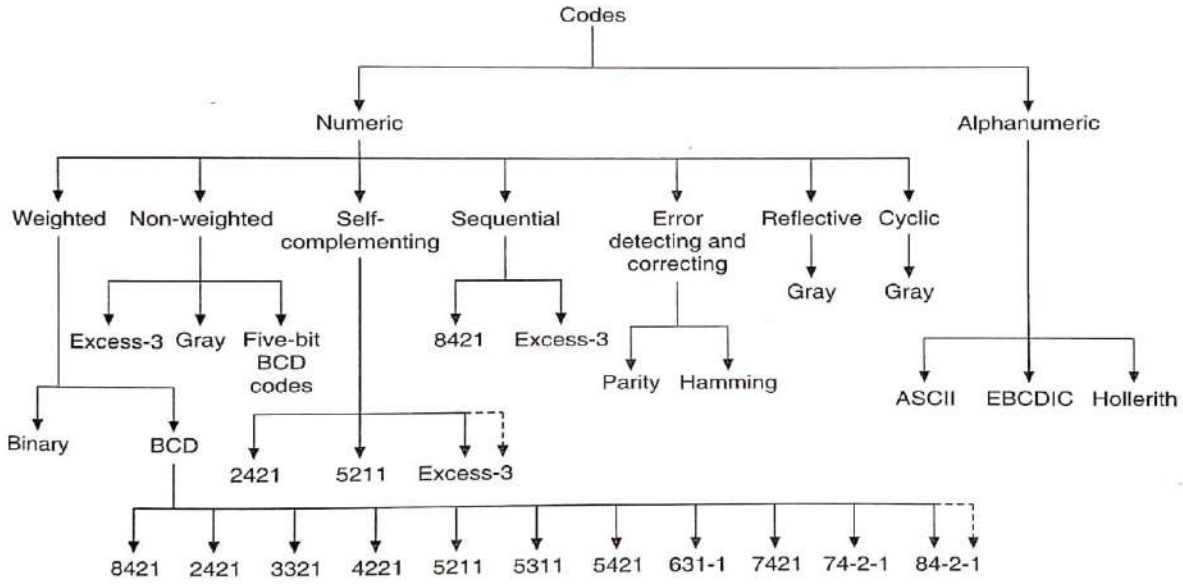
Let the base be b. Then

$$292_{10} = 1204_b = 1 \times b^3 + 2 \times b^2 + 0 \times b^1 + 4 \times b^0$$

or $$292_{10} = b^3 + 2b^2 + 4$$

Since 4 is the largest digit in the given number, $b \geq 5$. By trial and error, $b = 6$.

CLASSIFICATION OF BINARY CODES



CLASSIFICATION OF BINARY CODES

1. Numeric and Alphanumeric Codes
2. Weighted and Non-weighted Codes
3. Positively-weighted and Negatively-weighted Codes
4. Error Detecting and Error Correcting Codes
5. Sequential Codes
6. Self-complementing Codes
7. Cyclic Codes
8. Reflective Codes
9. Straight Binary Code

Binary coded decimal codes

Decimal digit	8 4 2 1	2 4 2 1	5 2 1 1	5 4 2 1	6 4 2 –3	8 4 –2 –1	XS-3
0	0000	0000	0000	0000	0000	0000	0011
1	0001	0001	0001	0001	0101	0111	0100
2	0010	0010	0011	0010	0010	0110	0101
3	0011	0011	0101	0011	1001	0101	0110
4	0100	0100	0111	0100	0100	0100	0111
5	0101	1011	1000	1000	1011	1011	1000
6	0110	1100	1010	1001	0110	1010	1001
7	0111	1101	1100	1010	1101	1001	1010
8	1000	1110	1110	1011	1010	1000	1011
9	1001	1111	1111	1100	1111	1111	1100
Unused bit combinations	1010	0101	0010	0101	0001	0001	0000
	1011	0110	0100	0110	0011	0010	0001
	1100	0111	0110	0111	0111	0011	0010
	1101	1000	1001	1101	1000	1100	1101
	1110	1001	1011	1110	1100	1101	1110
	1111	1010	1101	1111	1110	1110	1111

THE 8421 BCD CODE (NATURAL BCD CODE)

- The 8421 BCD code is so widely used that it is a common practice to refer to it simply as BCD code. In this code, each decimal digit, 0 through 9, is coded by a 4-bit binary number. It is also called the natural binary code because of the 8,4, 2 and 1 weights attached to it. It is a weighted code and is also sequential.
- For example, the decimal number 14 can be represented as 1110 in pure binary but as 0001 0100 in 8421 code.

THE 8421 BCD CODE (NATURAL BCD CODE)

- Another disadvantage of the BCD code is that, arithmetic operations are more complex than they are in pure binary. There are six illegal combinations 1010, 1011, 1100, 1101, 1110 and 1111 in this code, i.e. they are not part of the 8421 BCD code system.

BCD Addition

- If there is no carry and the sum term is not an illegal code, no correction is needed.
- If there is a carry out of one group to the next group, or if the sum term is an illegal code, then 6 (0110) is added to the sum term of that group and the resulting carry is added to the next group. (This is done to skip the six illegal states).

EXAMPLE 3.1 Perform the following decimal additions in the 8421 code.
(a) 25 + 13 (b) 679.6 + 536.8

Solution

(a)

	25		0010 0101	(25 in BCD)
	+13	⇒	+0001 0011	(13 in BCD)
	38		0011 1000	(No carry, no illegal code. So, this is the correct sum.)

BCD Addition

(b)

679.6	⇒	0110	0111	1001	.0110	(679.6 in BCD)
+ 536.8		+0101	0011	0110	.1000	(536.8 in BCD)
1216.4		1011	1010	1111	.1110	(All are illegal codes)
		+0110	+0110	+0110	+.0110	(Add 0110 to each)
		❶0001	❶0000	❶0101	❶.0100	(Propagate carry)
		+1 ↵	+1 ↵	+1 ↵	+1 ↵	
		0001	0010	0001	0110	.0100	(Corrected sum = 1216.4)
		1	2	1	6	. 4	

BCD Subtraction

- If there is no borrow from the next higher group then no correction is required.
- If there is a borrow from the next group, then 6 (0110) is subtracted from the difference term of this group. (This is done to skip the six illegal states.)

EXAMPLE 3.2 Perform the following decimal subtractions in the 8421 BCD code.
(a) 38 – 15 (b) 206.7 – 147.8

Solution

(a)

38	⇒	0011	1000	(38 in BCD)
–15		–0001	0101	(15 in BCD)
23		0010	0011	(No borrow. So, this is the correct difference.)

(b)

206.7	⇒	0010	0000	0110	.0111	(206.7 in BCD)
–147.8		–0001	0100	0111	.1000	(147.8 in BCD)
58.9		0000	1011	1110	.1111	(Borrows are present, subtract 0110)
			–0110	–0110	–.0110	
			0101	1000	.1001	(Corrected difference = 58.9_{10})

BCD Subtraction Using 9's Complement Methods

EXAMPLE 3.3 Perform the following decimal subtractions in BCD by the 9's complement method.

(a) 305.5 – 168.8 (b) 679.6 – 885.9

Solution

(a)

305.5	⇒	305.5		
−168.8		+831.1		(9's complement of 168.8)
136.7		❶136.6		
		↳ + 1		(End around carry)
		136.7		(Corrected difference)

BCD Subtraction Using 9's Complement Methods

305.5_{10}	⇒	0011	0000 0101	.0101	(305.5 in BCD)
$+831.1_{10}$		+1000	0011 0001	.0001	(9's complement of 168.8 in BCD)
		+1011	0011 0110	.0110	(1011 is an illegal code, add 0110)
		+0110			
		❶0001	0011 0110	.0110	
		↳		+1	(End around carry)
		0001	0011 0110	.0111	(Corrected difference = 136.7)

BCD Subtraction Using 10's Complement Methods

EXAMPLE 3.4 Perform the following subtractions in 8421 code using the 10's complement method.

(a) 342.7 – 108.9 (b) 206.4 – 507.6

Solution

(a)

```
  342.7          342.7
 -108.9    ⇒    +891.1    (10's complement of 108.9)
-------        --------
  233.8        ❶233.8     (Ignore carry)
```

```
342.7
+891.1  ⇒   0011   0100   0010   .0111   (342.7 in BCD)
           +1000   1001   0001   .0001   (10's complement of 108.9 in BCD)
           ------------------------------
            1011   1101   0011   .1000   (1011 and 1101 are illegal codes,
                                           add 0110 to each.)
           +0110  +0110
           ------------------------------
           ❶0001  ❶0011   0011   .1000   (Propagate carry)
        +1 ↵   +1 ↵
           ------------------------------
         ❶ 0010   0011   0011   .1000   (Ignore carry)
           ------------------------------
           0010   0011   0011   .1000   (Corrected difference = 233.8)
```

BCD Subtraction Using 10's Complement Methods

(b)

```
  206.4          206.4
 -507.6    ⇒    +492.4    (10's complement of 507.6)
-------        --------
 -301.2         698.8     (No carry)
```

No carry. So, the answer is negative and is in 10's complement form. The 10's complement of 698.8 is – 301.2.

```
206.4
+492.4  ⇒   0010 0000 0110 .0100   (206.4 in BCD)
           +0100 1001 0010 .0100   (10's complement of 507.6 in BCD)
           ------------------------
            0110 1001 1000 .1000   (No illegal codes, no carry)
```

As there is no carry, the result is negative and is in its 10's complement form. The 10's complement of 698.8 is 301.2. So, the corrected difference is –301.2.

THE EXCESS THREE (XS-3) CODE

- The Excess-3 code, also called XS-3, is a non-weighted BCD code. This code derives its name from the fact that each binary code word is the corresponding 8421 code word plus 0011(3).
- It is a sequential code and, therefore, can be used for arithmetic operations. It is a self-complementing code. Therefore, subtraction by the method of complement addition is more direct in XS-3 code than that in 8421 code.
- The XS-3 code has six invalid states 0000, 0001, 0010, 1101, 1110 and 1111.
- The XS-3 code has some very interesting properties when used in addition and subtraction.

XS-3 Addition

- If there is no carry out from the addition of any of the 4-bit groups, subtract 0011 from the sum term of those groups (because when two decimal digits are added in XS-3 and there is no carry, the result is in XS-6).
- If there is a carry out, add 0011 to the sum term of those groups (because when there is a carry, the invalid states are skipped and the result is in normal binary).

XS-3 Addition

EXAMPLE 3.5 Perform the following additions in XS-3 code.
(a) 37 + 28 (b) 247.6 + 359.4

Solution

(a)

37	0110	1010	(37 in XS-3)
+28	+0101	1011	(28 in XS-3)
65	1011	❶0101	(Carry generated)
	+1 ↷		(Propagate carry)
	1100	0101	(Add 0011 to correct 0101 and
	−0011	+0011	subtract 0011 to correct 1100)
	1001	1000	(Corrected sum in XS-3 = 65_{10})

XS-3 Addition

(b)

247.6	0101	0111	1010	.1001	(247.6 in XS-3)
+359.4	+0110	1000	1100	.0111	(359.4 in XS-3)
607.0	1011	1111	❶0110	❶.0000	(Carry generated)
		+1 ↷	+1 ↷		(Propagate carry)
	1011	❶0000	0111	.0000	
	+1 ↷				
	1100	0000	0111	.0000	(Add 0011 to 0000, 0111, 0000
	−0011	+0011	+0011	+.0011	and subtract 0011 from 1100)
	1001	0011	1010	.0011	(Corrected sum in XS-3 = 607.0_{10})

XS-3 Subtraction

- If there is no borrow from the next 4-bit group, add 0011 to the difference term of such groups (because when decimal digits are subtracted in XS-3 and there is no borrow, the result is in normal binary).
- If there is a borrow, subtract 0011 from the difference term (because taking a borrow is equivalent to adding six invalid states, so the result is in XS-6).

XS-3 Subtraction

EXAMPLE 3.6 Perform the following subtractions in XS-3 code.
(a) 267 – 175 (b) 57.6 – 27.8

Solution

(a)

267	0101	1001	1010	(267 in XS-3)
–175	–0100	1010	1000	(175 in XS-3)
092	0000	1111	0010	(Correct 0010 and 0000 by adding 0011and
	+0011	–0011	+0011	correct 1111 by subtracting 0011)
	0011	1100	0101	(Corrected difference in XS-3 = 92_{10})

XS-3 Subtraction

(b)	57.6	⇒	1000	1010	.1001	(57.6 in XS-3)
	−27.8		−0101	1010	.1011	(27.8 in XS-3)
	29.8		0010	1111	.1110	(Correct 0010 by adding 0011 and correct
			+0011	−0011	−.0011	1110 and 1111 by subtracting 0011)
			0101	1110	.1011	(Corrected difference in XS-3 = 29.8_{10})

XS-3 Subtraction Using 9's Complement Methods

EXAMPLE 3.7 Perform the following subtractions in XS-3 code using the 9's complement method.
(a) 687 − 348 (b) 246 − 592

Solution

(a) The subtrahend (348) in XS-3 code and its complements are:

9's complement of 348 = 651

687	⇒	687	
−348		+651	(9's complement of 348)
339		❶338	
		↳ +1	(End around carry)
		339	(Corrected difference in decimal)

	1001	1011	1010	(687 in XS-3)
	+1001	1000	0100	(1's complement of 348 in XS-3)
	❶0010	❶0011	1110	(Carry generated)
+1 ↲	+1 ↲			(Propagate carry)
❶	0011	0011	1110	
↳			+1	(End around carry)
	0011	0011	1111	(Correct 1111 by subtracting 0011 and
	+0011	+0011	−0011	correct both groups of 0011 by adding 0011)
	0110	0110	1100	(Corrected difference in XS-3 = 339_{10})

XS-3 Subtraction Using 10's Complement Methods

EXAMPLE 3.8 Perform the following subtractions in XS-3 code using the 10's complement method.

(a) 597 – 239 (b) 354 – 672

Solution

(a) 10's complement of 239 = 761

XS-3 code of 239 = 0101 0110 1100

2's complement of 239 in XS-3 code = 1010 1001 0100

XS-3 code of 597 = 1000 1100 1010

597	⇒	597	
–239		+761	(10's complement of 239)
358		❶358	(Ignore carry)
		358	(Corrected difference in decimal)

XS-3 Subtraction Using 10's Complement Methods

	1000	1100	1010	(597 in XS-3)
	+1010	1001	0100	(2's complement of 239 in XS-3)
	❶0010	❶0101	1110	(Propagate carry)
+1	↵ +1	↵		
1	0011	0101	1110	(Ignore carry)
	+0011	+0011	–0011	(Correct 1110 by subtracting 0011 and
	0110	1000	1011	correct 0101 and 0011 by adding 0011)
				(Corrected difference in XS-3 code = 358)

THE GRAY CODE (REFLECTIVE-CODE)

- The Gray code is a non-weighted code, and is not suitable for arithmetic operations.
- It is not a BCD code.
- It is a cyclic code because successive code words in this code differ in one bit position only, i.e. it is a unit distance code.
- It is the most popular of the unit distance codes.
- It is also a reflective code, i.e. it is both reflective and unit distance.

Reflection of Gray codes

Gray Code				Decimal	4-bit binary
1-bit	2-bit	3-bit	4-bit		
0	00	000	0000	0	0000
1	01	001	0001	1	0001
	11	011	0011	2	0010
	10	010	0010	3	0011
		110	0110	4	0100
		111	0111	5	0101
		101	0101	6	0110
		100	0100	7	0111
			1100	8	1000
			1101	9	1001
			1111	10	1010
			1110	11	1011
			1010	12	1100
			1011	13	1101
			1001	14	1110
			1000	15	1111

Doubts

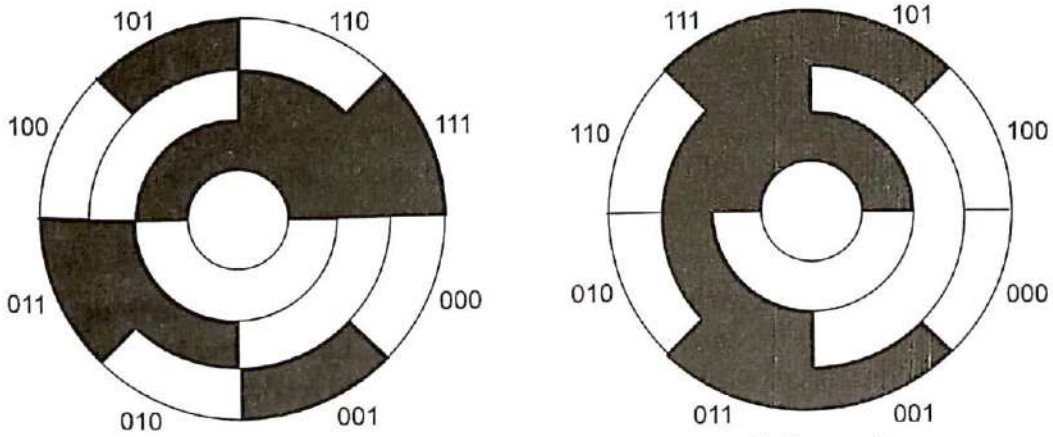


Figure 3.2 Position indicating system.

Binary-to-Gray Conversion

$G_n = B_n$	$G_{n-1} = B_n \oplus B_{n-1}$	$G_{n-2} = B_{n-1} \oplus B_{n-2} \cdots$	$G_1 = B_2 \oplus B_1$

EXAMPLE 3.10 Convert the binary 1001 to the Gray code.

Solution

(a)

Binary	1	⊕	0	⊕	0	⊕	1
	↓		↓		↓		↓
Gray	1		1		0		1

(b)

Binary	1 0 0 1
Shifted binary	1 0 0 ❶
Gray	1 1 0 1

Binary-to-Gray Conversion

$G_n = B_n$	$G_{n-1} = B_n \oplus B_{n-1}$	$G_{n-2} = B_{n-1} \oplus B_{n-2} \cdots$	$G_1 = B_2 \oplus B_1$

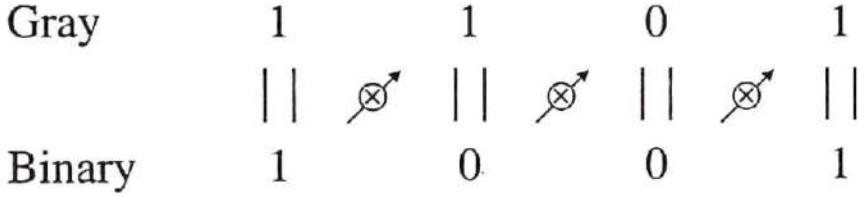
Gray-to-Binary Conversion

$B_n = G_n$	$B_{n-1} = B_n \oplus G_{n-1}$	$B_{n-2} = B_{n-1} \oplus G_{n-2} \cdots$	$B_1 = B_2 \oplus G_1$

EXAMPLE 3.11 Convert the Gray code 1101 to binary.

Solution

The conversion is done as shown below:



Gray-to-Binary Conversion

$B_n = G_n$	$B_{n-1} = B_n \oplus G_{n-1}$	$B_{n-2} = B_{n-1} \oplus G_{n-2} \cdots$	$B_1 = B_2 \oplus G_1$

Numerical

EXAMPLE 3.12

(a) Convert the following into the Gray number.

(i) $3\,A\,7_{16}$ (ii) $5\,2\,7_8$ (iii) $6\,5\,2_{10}$

Solution

(a) To convert the given number in any system into Gray number first convert it into binary and then convert that binary number into the Gray code using the normal procedure.

(i) $3\,A\,7_{16} = 0\,0\,1\,1, 1\,0\,1\,0, 0\,1\,1\,1_2 = 1\,0\,0\,1\,1\,1\,0\,1\,0\,0$ (Gray)

(ii) $527_8 = 1\,0\,1, 0\,1\,1, 0\,1\,1_2 = 1\,1\,1\,1\,1\,0\,1\,1\,0$ (Gray)

(iii) $652_{10} = 1\,0\,1\,0\,0\,0\,1\,1\,0\,0_2 = 1\,1\,1\,1\,0\,0\,1\,0\,1\,0$ (Gray)

Numerical

EXAMPLE 3.12

(b) Convert the Gray number 1 0 1 1 0 0 1 0 into
(i) hex (ii) octal (iii) decimal

Solution

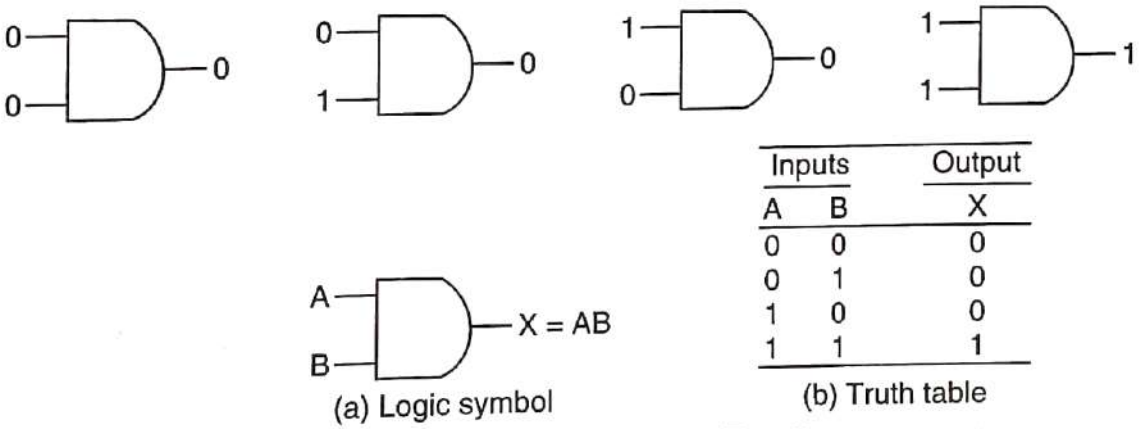
(b) To convert the given Gray number into any other number system, first convert the given Gray number into a binary number and then convert that binary number into the required number system.

$$1\,0\,1\,1\,0\;\;0\,1\,0\;\;(\text{Gray}) = 1\,1\,0\,1\,1\,1\,0\,0_2 = DC_{16} = 334_8 = 220_{10}$$

Logic Gate and Logic Design

- Logic gates are the fundamental building blocks of digital systems. The name logic gate is derived from the ability of such a device to make decisions.
- The interconnection of gates to perform a variety of logical operations is called logic design.
- A logic in which the voltage levels represent logic 1 and logic 0, Level logic may be positive logic or negative logic.
- In transistor-transistor logic (TTL, the most widely used logic family), the voltage levels are + 5 V and 0 V.

THE AND GATE



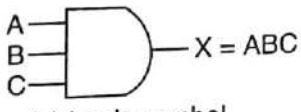
Inputs		Output
A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

(a) Logic symbol (b) Truth table

Figure 4.1 A two-input AND gate.

THE AND GATE

The logic symbol and the truth table of a three-input AND gate are shown in Figure 4.2.



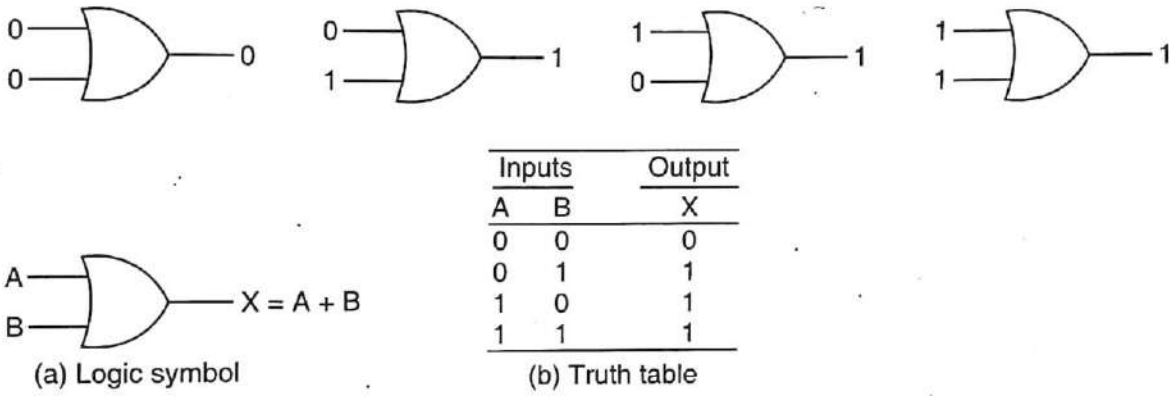
(a) Logic symbol

Inputs			Output
A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

(b) Truth table

Figure 4.2 A three-input AND gate.

THE OR GATE



Inputs		Output
A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

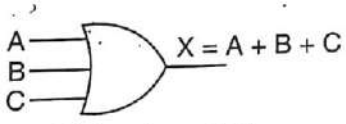
(a) Logic symbol

(b) Truth table

Figure 4.4 A two-input OR gate.

THE OR GATE

The logic symbol and the truth table of a three-input OR gate are shown in Figure 4.5.



(a) Logic symbol

Inputs			Output
A	B	C	X
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) Truth table

Figure 4.5 A three-input OR gate.

THE NOT GATE (INVERTER)

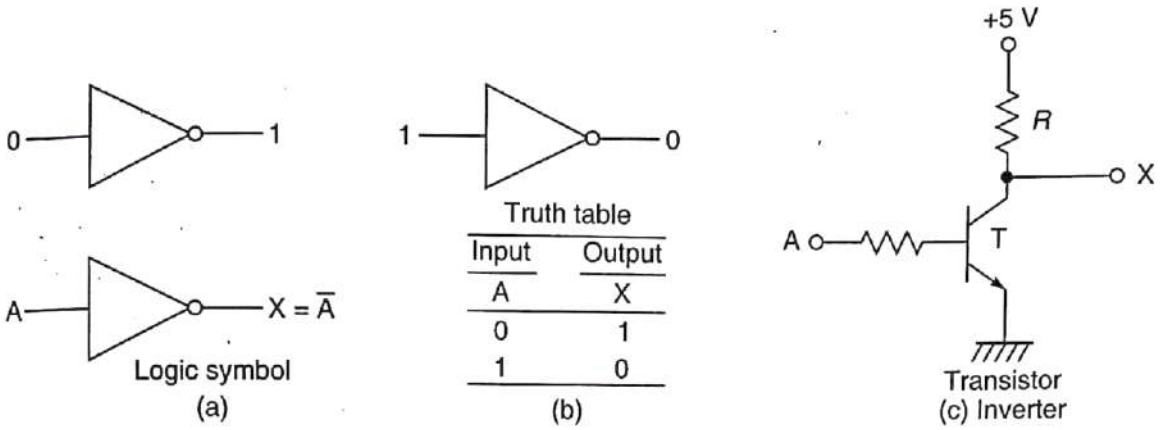


Figure 4.7 The inverter.

The NAND Gate

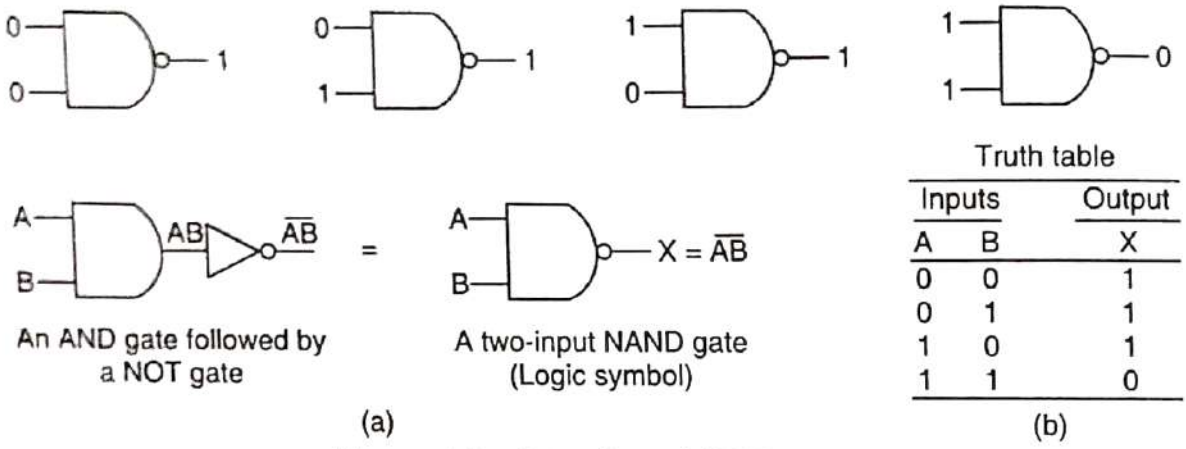
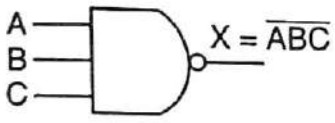


Figure 4.8 A two-input NAND gate.

The NAND Gate



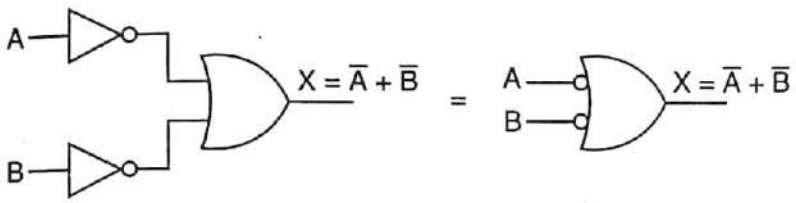
(a) Logic symbol

Inputs			Output
A	B	C	X
0	0	0	1
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	0

(b) Truth table

Figure 4.9 A three-input NAND gate.

The NAND Gate = Bubbled OR gate



(a)

Inputs		Inverted inputs		Output
A	B	$\overline{A}$	$\overline{B}$	X
0	0	1	1	1
0	1	1	0	1
1	0	0	1	1
1	1	0	0	0

(b) Truth table

Figure 4.10 Bubbled OR gate.

Bubbled NAND gate

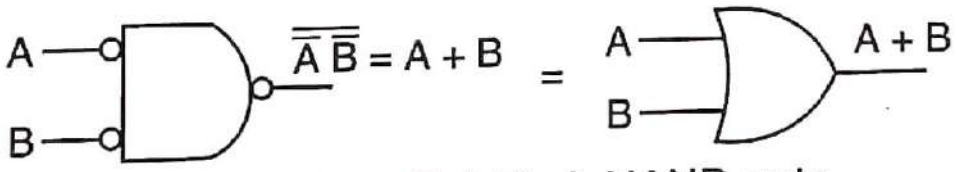
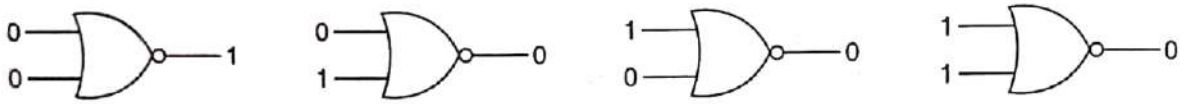
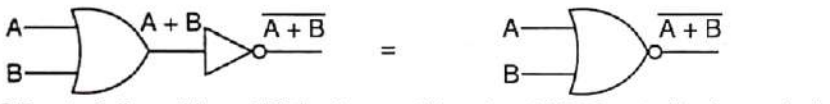


Figure 4.12 Bubbled NAND gate.

The NOR Gate





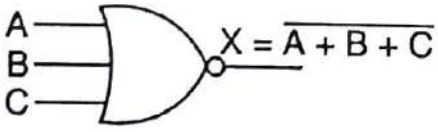
An OR gate followed by a NOT gate (a) A two-input NOR gate (logic symbol)

Inputs		Output
A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

(b) Truth table

Figure 4.14 A two-input NOR gate.

The NOR Gate



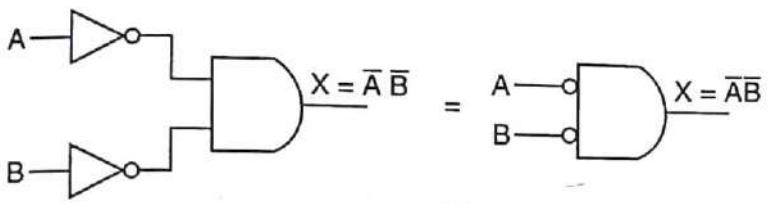
(a) Logic symbol

Inputs			Output
A	B	C	X
0	0	0	1
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	0

(b) Truth table

Figure 4.15 A three-input NOR gate.

The NOR Gate = Bubbled AND gate



(a)

Inputs		Inverted inputs		Output
A	B	$\overline{A}$	$\overline{B}$	X
0	0	1	1	1
0	1	1	0	0
1	0	0	1	0
1	1	0	0	0

(b) Truth table

Figure 4.16 Bubbled AND gate.

Bubbled NOR gate

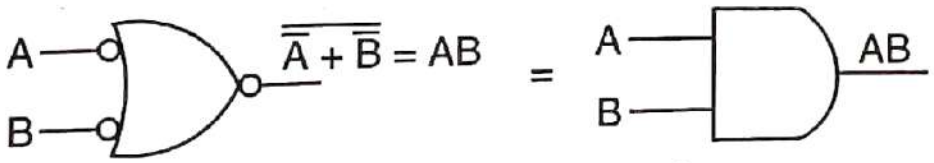
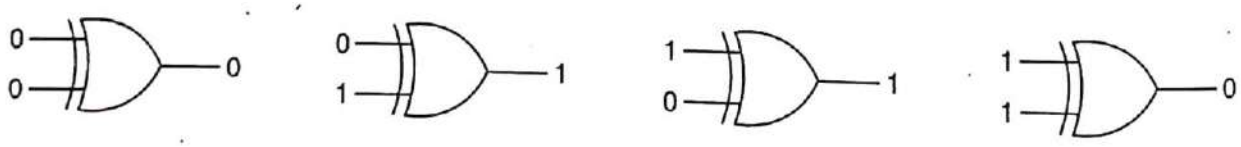
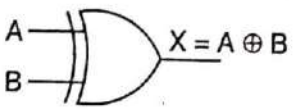


Figure 4.18 Bubbled NOR gate.

THE EXCLUSIVE-OR (X-OR) GATE



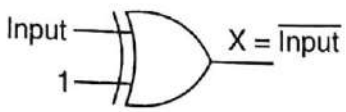


Logic symbol

(a)

Inputs		Output
A	B	X = A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

(b) Truth table



(c) X-OR gate as an inverter

Figure 4.20 Exclusive-OR gate.

PROPERTIES OF EXCLUSIVE-OR

1. $A \oplus 1 = \overline{A}$

Proof: $A \oplus 1 = A \cdot \overline{1} + \overline{A} \cdot 1 = A \cdot 0 + \overline{A} = \overline{A}$

2. $A \oplus 0 = A$

Proof: $A \oplus 0 = A \cdot \overline{0} + \overline{A} \cdot 0 = A \cdot 1 + 0 = A$

3. $A \oplus A = 0$

Proof: $A \oplus A = A \cdot \overline{A} + \overline{A} \cdot A = 0 + 0 = 0$

4. $A \oplus \overline{A} = 1$

Proof: $A \oplus \overline{A} = A \cdot \overline{\overline{A}} + \overline{A} \cdot \overline{A} = A + \overline{A} = 1$

PROPERTIES OF EXCLUSIVE-OR

5. $AB \oplus AC = A(B \oplus C)$

Proof: $AB \oplus AC = AB \cdot \overline{AC} + \overline{AB} \cdot AC = AB(\overline{A} + \overline{C}) + (\overline{A} + \overline{B})\,AC$

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

6. If $A \oplus B = C$, then $A \oplus C = B$, $B \oplus C = A$, $A \oplus B \oplus C = 0$

Proof: $A \oplus B \oplus C = (A \oplus B) \oplus (A \oplus B) = (A \oplus B) \cdot \overline{(A \oplus B)} + \overline{(A \oplus B)} \cdot (A \oplus B) = 0 + 0 = 0$

Proof: $A \oplus C = A \oplus (A \oplus B) = A \cdot \overline{(A \oplus B)} + \overline{A} \cdot (A \oplus B) = A(AB + \overline{A}\,\overline{B}) + \overline{A}(A\overline{B} + \overline{A}B)$

$= AB + \overline{A}B = B\,(A + \overline{A}) = B \cdot 1 = B$

Proof: $B \oplus C = B \oplus (A \oplus B) = B \cdot \overline{A \oplus B} + \overline{B}\,(A \oplus B) = B \cdot (AB + \overline{A}\,\overline{B}) + \overline{B}\,(A\overline{B} + \overline{A}B)$

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

THE EXCLUSIVE-NOR (X-NOR) GATE

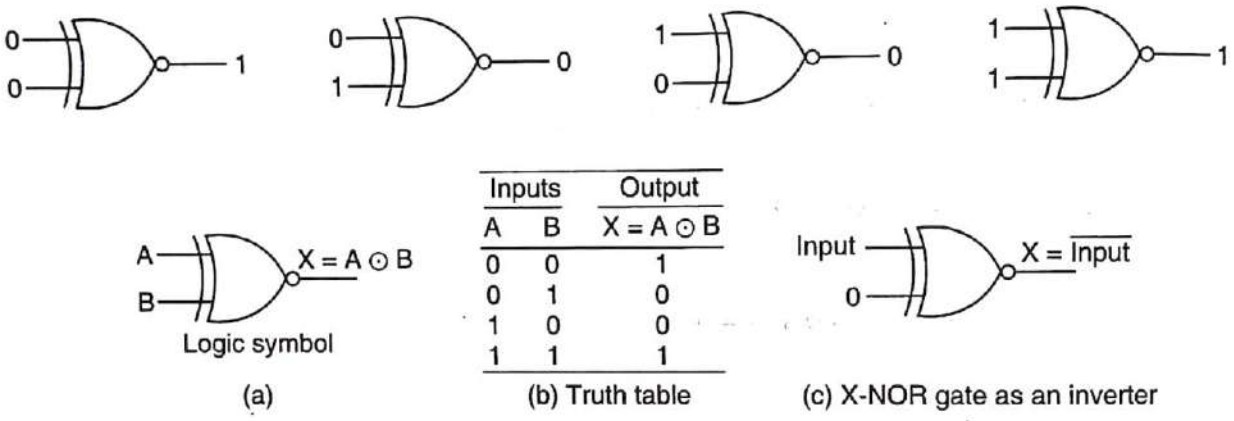


Figure 4.21 Exclusive-NOR gate.

Some Properties of X-NOR Gate

$$X = A \odot B = AB + \bar{A}\bar{B} = \overline{A \oplus B} = \overline{A\bar{B} + \bar{A}B}$$

$$A \odot B = \overline{A \oplus B}$$

$$A \odot B \odot C \neq \overline{A \oplus B \oplus C}$$

Numerical

EXAMPLE 4.1 Find the logical equivalent of the following expressions.

(a) $A \oplus 0$ (b) $A \oplus 1$ (c) $A \odot 0$ (d) $A \odot 1$

(e) $1 \oplus \overline{A}$ (f) $0 \oplus \overline{A}$

Solution

(a) $A \oplus 0 = A \cdot \overline{0} + \overline{A} \cdot 0 = A \cdot 1 + \overline{A} \cdot 0 = A + 0 = A$

(b) $A \oplus 1 = A \cdot \overline{1} + \overline{A} \cdot 1 = A \cdot 0 + \overline{A} \cdot 1 = 0 + \overline{A} = \overline{A}$

(c) $A \odot 0 = \overline{A} \cdot \overline{0} + A \cdot 0 = \overline{A} \cdot 1 + A \cdot 0 = \overline{A} + 0 = \overline{A}$

(d) $A \odot 1 = \overline{A} \cdot \overline{1} + A \cdot 1 = \overline{A} \cdot 0 + A \cdot 1 = 0 + A = A$

(e) $1 \oplus \overline{A} = 1 \cdot \overline{\overline{A}} + \overline{1} \cdot \overline{A} = 1 \cdot A + 0 \cdot \overline{A} = A + 0 = A$

(f) $0 \oplus \overline{A} = 0 \cdot \overline{\overline{A}} + \overline{0} \cdot \overline{A} = 0 \cdot A + 1 \cdot \overline{A} = 0 + \overline{A} = \overline{A}$

EXAMPLE 4.7 Determine which of the following expressions are equivalent to $A \oplus B$ and which to $A \odot B$?

(a) $\overline{A} \oplus B$ (b) $\overline{A} \odot B$ (c) $\overline{A} \oplus \overline{B}$ (d) $\overline{A} \odot \overline{B}$

(e) $A \oplus \overline{B}$ (f) $A \odot \overline{B}$

Solution

(a) $\overline{A} \oplus B = \overline{A}\overline{B} + \overline{\overline{A}}B = \overline{A}\overline{B} + AB = A \odot B$

(b) $\overline{A} \odot B = \overline{\overline{A}}\overline{B} + \overline{A}B = A\overline{B} + \overline{A}B = A \oplus B$

(c) $\overline{A} \oplus \overline{B} = \overline{\overline{A}}\overline{B} + \overline{A}\overline{\overline{B}} = A\overline{B} + \overline{A}B = A \oplus B$

(d) $\overline{A} \odot \overline{B} = \overline{\overline{A}}\overline{\overline{B}} + \overline{A}\overline{B} = AB + \overline{A}\overline{B} = A \odot B$

(e) $A \oplus \overline{B} = A\overline{\overline{B}} + \overline{A}\overline{B} = AB + \overline{A}\overline{B} = A \odot B$

(f) $A \odot \overline{B} = A\overline{B} + \overline{A}\overline{\overline{B}} = A\overline{B} + \overline{A}B = A \oplus B$

Enable and Inhibit Circuits

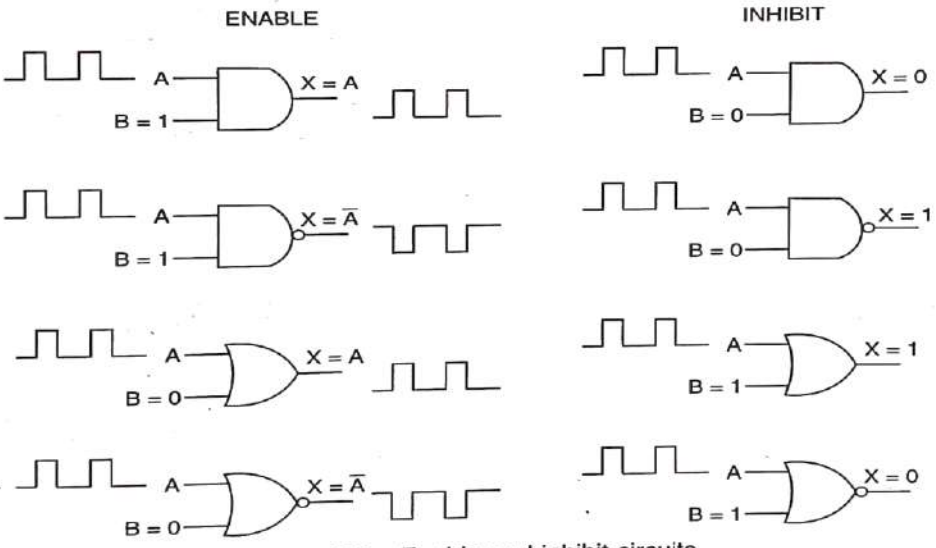


Figure 4.22 Enable and inhibit circuits.

EXAMPLE 4.2 Show that $A \oplus B = A\bar{B} + \bar{A}B$ and construct the corresponding logic diagrams.

Solution

The truth tables constructed below show that $A \oplus B = A\bar{B} + \bar{A}B$. The corresponding logic diagrams are also shown in Figure 4.23.

A	B	A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

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

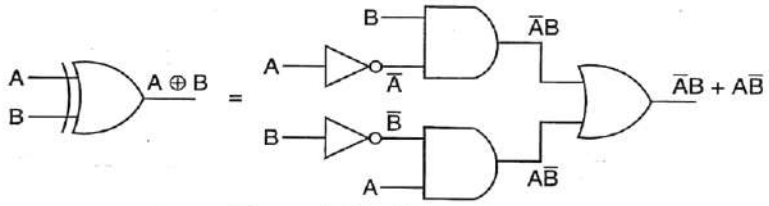


Figure 4.23 Example 4.2.

EXAMPLE 4.3 Show that $A \odot B = AB + \overline{A}\,\overline{B} = \overline{A \oplus B} = \overline{A\overline{B} + \overline{A}B}$. Also, construct the corresponding logic diagrams.

Solution

The truth tables constructed below show that $A \odot B = AB + \overline{A}\,\overline{B} = \overline{A \oplus B} = \overline{A\overline{B} + \overline{A}B}$. The corresponding logic diagrams are also shown in Figure 4.24.

A	B	$A \odot B$
0	0	1
0	1	0
1	0	0
1	1	1

A	B	$\overline{A}$	$\overline{B}$	AB	$\overline{A}\,\overline{B}$	$AB + \overline{A}\,\overline{B}$	$\overline{A\overline{B} + \overline{A}B}$
0	0	1	1	0	1	1	1
0	1	1	0	0	0	0	0
1	0	0	1	0	0	0	0
1	1	0	0	1	0	1	1

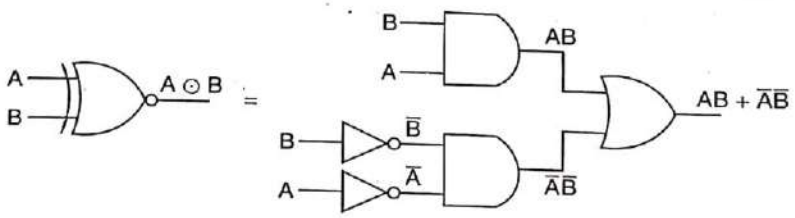


Figure 4.24 Example 4.3.

EXAMPLE 4.4 Show that $(A + B)\ \overline{AB}$ is equivalent to $A \oplus B$. Also, construct the corresponding logic diagrams.

Solution

The truth tables constructed below show that $(A + B)\ \overline{AB} = A \oplus B$. The logic diagrams are also shown in Figure 4.25.

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

A	B	A + B	AB	$\overline{AB}$	$(A + B)(\overline{AB})$
0	0	0	0	1	0
0	1	1	0	1	1
1	0	1	0	1	1
1	1	1	1	0	0

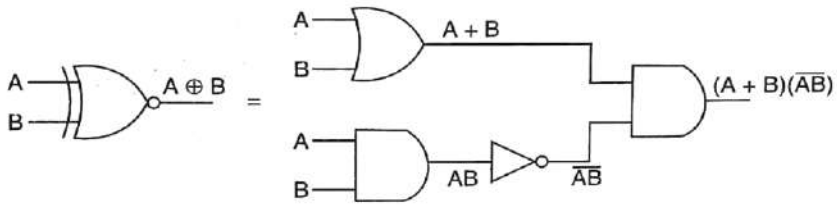


Figure 4.25 Example 4.4.

EXAMPLE 4.6 Show that $AB + (\overline{A + B})$ is equivalent to $A \odot B$. Also construct the corresponding logic diagrams.

Solution

The truth tables constructed below show that $AB + (\overline{A + B}) = A \odot B$. The corresponding logic diagrams are also shown in Figure 4.27.

A	B	$A \odot B$
0	0	1
0	1	0
1	0	0
1	1	1

A	B	$A + B$	AB	$\overline{A + B}$	$AB + \overline{A + B}$
0	0	0	0	1	1
0	1	1	0	0	0
1	0	1	0	0	0
1	1	1	1	0	1

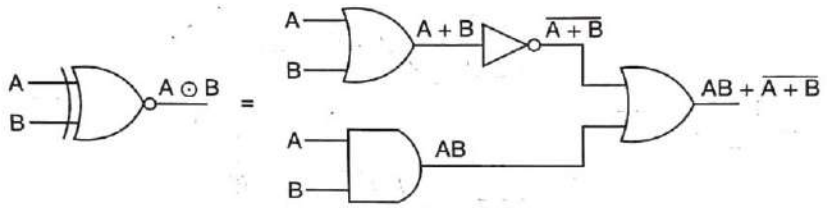


Figure 4.27 Example 4.6.

EXAMPLE 4.8 For a two-input AND gate, determine its output waveform in relation to input waveforms of Figure 4.28a.

Solution

The output waveform X is shown in Figure 4.28b.

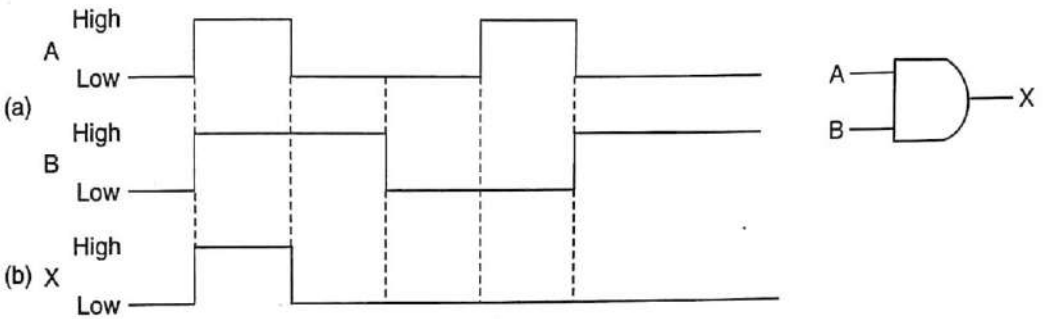


Figure 4.28 Example 4.8.

EXAMPLE 4.9 If the three waveforms A, B, and C shown in Figure 4.29a are applied to a three-input AND gate, determine the resulting output waveform.

Solution

The output waveform X is shown in Figure 4.29b.

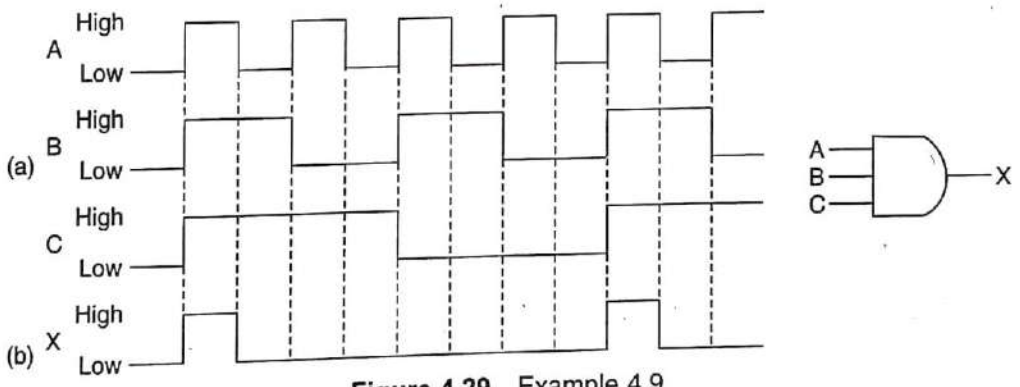


Figure 4.29 Example 4.9.

EXAMPLE 4.10 For a two-input OR gate, determine its output waveform in relation to the inputs A and B shown in Figure 4.30a.

Solution

The output waveform is shown in Figure 4.30b.

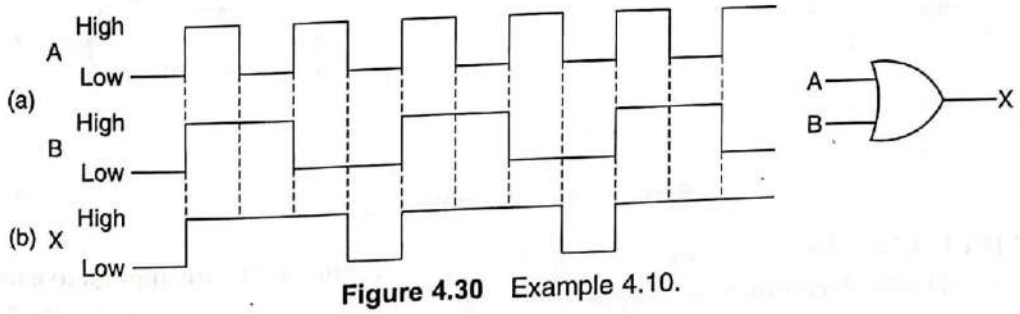
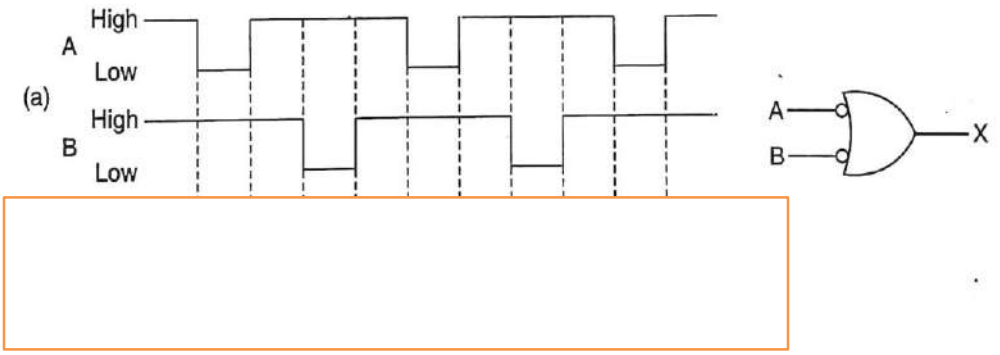


Figure 4.30 Example 4.10.

EXAMPLE 4.15 For the two-input NAND gate operating as a negative OR gate, determine the output waveform when the input waveforms A and B are as shown in Figure 4.35a.

Solution

The output of an active-LOW OR gate is HIGH, if either A is LOW or B is LOW or both A and B are LOW. The output waveform is shown in Figure 4.35b.



EXAMPLE 4.15 For the two-input NAND gate operating as a negative OR gate, determine the output waveform when the input waveforms A and B are as shown in Figure 4.35a.

Solution

The output of an active-LOW OR gate is HIGH, if either A is LOW or B is LOW or both A and B are LOW. The output waveform is shown in Figure 4.35b.

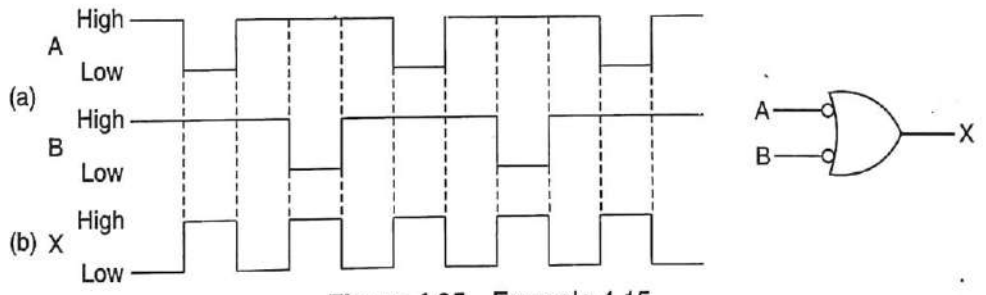
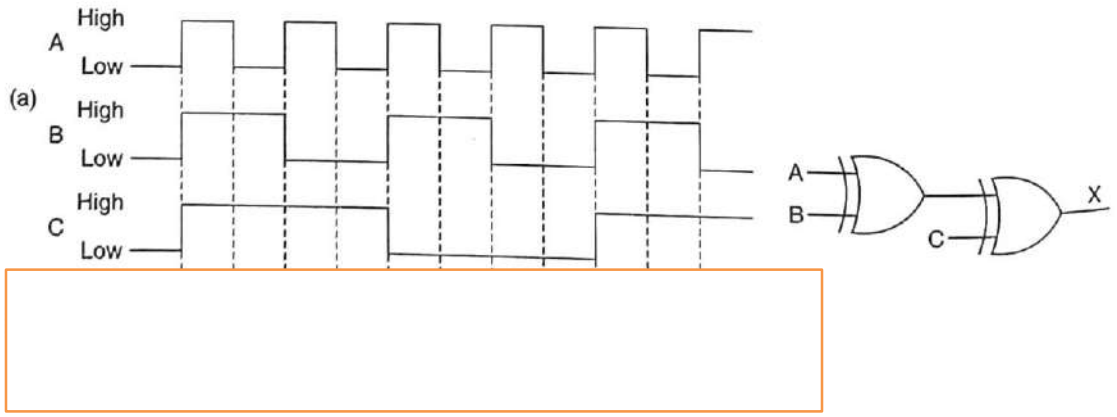


Figure 4.35 Example 4.15.

EXAMPLE 4.19 If the three waveforms A, B and C, shown in Figure 4.39a, are to be X-ORed, determine the output waveform.

Solution

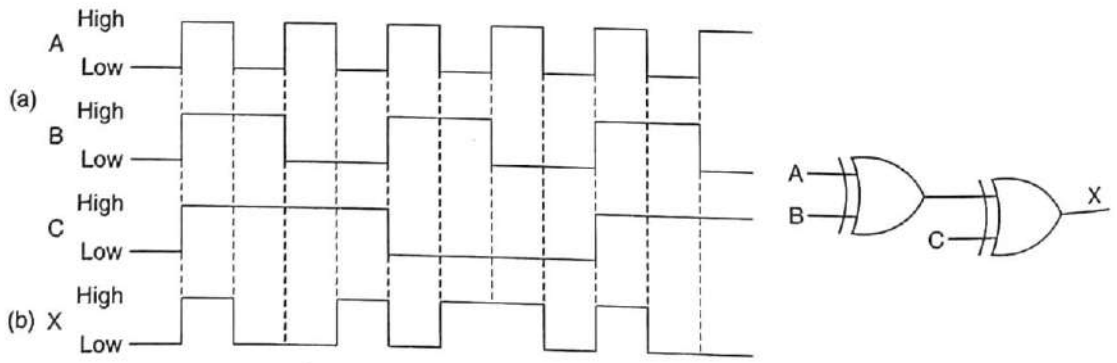
The X-OR output of a number of variables is HIGH, only when an odd number of input variables are HIGH. The output waveform is shown in Figure 4.39b.



EXAMPLE 4.19 If the three waveforms A, B and C, shown in Figure 4.39a, are to be X-ORed, determine the output waveform.

Solution

The X-OR output of a number of variables is HIGH, only when an odd number of input variables are HIGH. The output waveform is shown in Figure 4.39b.



AXIOMS OF BOOLEAN ALGEBRA

AND operation	OR operation	NOT operation
Axiom 1: $0 \cdot 0 = 0$	Axiom 5: $0 + 0 = 0$	Axiom 9: $\overline{1} = 0$
Axiom 2: $0 \cdot 1 = 0$	Axiom 6: $0 + 1 = 1$	Axiom 10: $\overline{0} = 1$
Axiom 3: $1 \cdot 0 = 0$	Axiom 7: $1 + 0 = 1$	
Axiom 4: $1 \cdot 1 = 1$	Axiom 8: $1 + 1 = 1$	

LAWS OF BOOLEAN ALGEBRA

1. Complementation Laws

Law 1:		$\overline{0} = 1$		
Law 2:		$\overline{1} = 0$		
Law 3:	If	$A = 0$,	then	$\overline{A} = 1$
Law 4:	If	$A = 1$,	then	$\overline{A} = 0$
Law 5:		$\overline{\overline{A}} = A$	(double complementation law)	

Notice that the double complementation does not change the function.

LAWS OF BOOLEAN ALGEBRA

2. AND Laws

The four AND laws are as follows:

Law 1:	$A \cdot 0 = 0$ (Null law)
Law 2:	$A \cdot 1 = A$ (Identity law)
Law 3:	$A \cdot A = A$
Law 4:	$A \cdot \overline{A} = 0$

LAWS OF BOOLEAN ALGEBRA

3. OR Laws

The four OR laws are as follows:

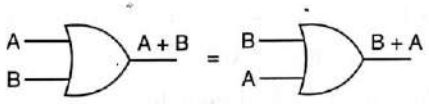
Law 1:	$A + 0 = A$ (Null law)
Law 2:	$A + 1 = 1$ (Identity law)
Law 3:	$A + A = A$
Law 4:	$A + \overline{A} = 1$

LAWS OF BOOLEAN ALGEBRA

4. Commutative Law1

Commutative laws allow change in position of AND or OR variables. There are two commutative laws.

Law 1: $A + B = B + A$



A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	1

=

B	A	B + A
0	0	0
0	1	1
1	0	1
1	1	1

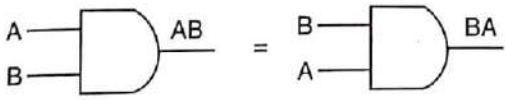
This law can be extended to any number of variables. For example,

$$A + B + C = B + C + A = C + A + B = B + A + C$$

LAWS OF BOOLEAN ALGEBRA

4. Commutative Law2

Law 2: $A \cdot B = B \cdot A$



A	B	A · B
0	0	0
0	1	0
1	0	0
1	1	1

=

B	A	B · A
0	0	0
0	1	0
1	0	0
1	1	1

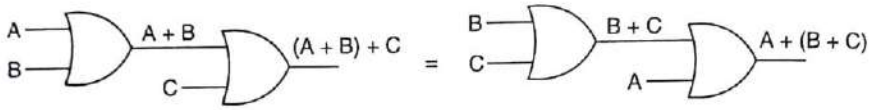
This law can be extended to any number of variables. For example,

$$A \cdot B \cdot C = B \cdot C \cdot A = C \cdot A \cdot B = B \cdot A \cdot C$$

LAWS OF BOOLEAN ALGEBRA

5. Associative Law1

Law 1: $(A + B) + C = A + (B + C)$



A	B	C	A + B	(A + B) + C
0	0	0	0	0
0	0	1	0	1
0	1	0	1	1
0	1	1	1	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

=

A	B	C	B + C	A + (B + C)
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	1
1	0	0	0	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

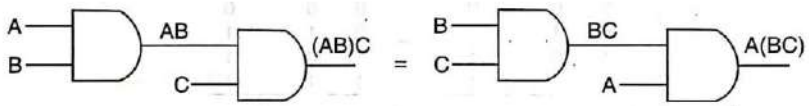
This law can be extended to any number of variables. For example,

$A + (B + C + D) = (A + B + C) + D = (A + B) + (C + D).$

LAWS OF BOOLEAN ALGEBRA

5. Associative Law2

Law 2: $(A \cdot B)C = A(B \cdot C)$.



A	B	C	AB	(AB)C
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	0	0
1	0	1	0	0
1	1	0	1	0
1	1	1	1	1

=

A	B	C	BC	A(BC)
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	0
1	0	0	0	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

This law can be extended to any number of variables. For example,

$A(BCD) = (ABC)D = (AB)(CD)$

LAWS OF BOOLEAN ALGEBRA

6. Distributive Laws

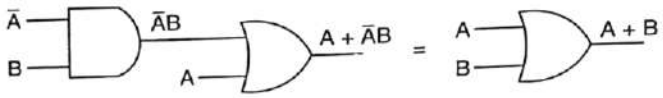
Law 1: $A(B + C) = AB + AC$

Law 2: $A + BC = (A + B)(A + C)$

LAWS OF BOOLEAN ALGEBRA

7. Redundant Literal Rule (RLR)

Law 1: $A + \bar{A}B = A + B$



A	B	ĀB	A + ĀB
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	1

=

A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	1

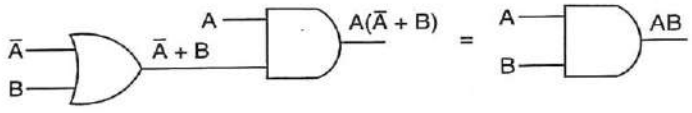
This law can be proved algebraically as shown below.

$$A + \bar{A}B = (A + \bar{A})(A + B)$$
$$= 1 \cdot (A + B)$$
$$= A + B$$

LAWS OF BOOLEAN ALGEBRA

7. Redundant Literal Rule (RLR)

Law 2: $A(\bar{A} + B) = AB$



A	B	Ā + B	A(Ā + B)
0	0	1	0
0	1	1	0
1	0	0	0
1	1	1	1

=

A	B	AB
0	0	0
0	1	0
1	0	0
1	1	1

This law can be proved algebraically as shown below.

$$A(\bar{A} + B) = A\bar{A} + AB$$
$$= 0 + AB$$
$$= AB$$

Complement of a term appearing in another term is redundant.

LAWS OF BOOLEAN ALGEBRA

8. Idempotence Laws

Law 1: $A \cdot A = A$

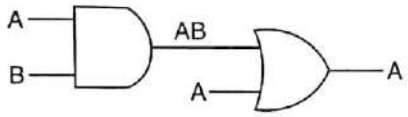
Law 2: $A + A = A$

LAWS OF BOOLEAN ALGEBRA

9. Absorption Laws

Law 1: $A + A \cdot B = A$

This law states that ORing of a variable (A) with the AND of that variable (A) and another variable (B) is equal to that variable itself (A).



A	B	AB	A + AB
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

Algebraically, we have

$$A + A \cdot B = A(1 + B) = A \cdot 1 = A$$

Therefore,

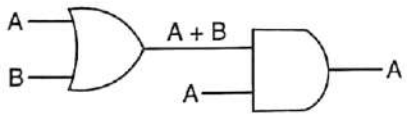
$$A + A \cdot \text{Any term} = A$$

LAWS OF BOOLEAN ALGEBRA

9. Absorption Laws

Law 2: $A(A + B) = A$

This law states that ANDing of a variable (A) with the OR of that variable (A) and another variable (B) is equal to that variable itself (A).



A	B	A + B	A(A + B)
0	0	0	0
0	1	1	0
1	0	1	1
1	1	1	1

Algebraically, we have

$$A(A + B) = A \cdot A + A \cdot B = A + AB = A(1 + B) = A \cdot 1 = A$$

Therefore,

$$A(A + \text{Any term}) = A$$

LAWS OF BOOLEAN ALGEBRA

10. Consensus Theorem (Included Factor Theorem)

Theorem 1: $AB + \bar{A}C + BC = AB + \bar{A}C$

Proof:

$$\begin{aligned} LHS &= AB + \bar{A}C + BC \\ &= AB + \bar{A}C + BC(A + \bar{A}) \\ &= AB + \bar{A}C + BCA + BC\bar{A} \\ &= AB(1 + C) + \bar{A}C(1 + B) \\ &= AB(1) + \bar{A}C(1) \\ &= AB + \bar{A}C \\ &= RHS \end{aligned}$$

This theorem can be extended to any number of variables. For example,

$$AB + \bar{A}C + BCD = AB + \bar{A}C$$

$$LHS = AB + \bar{A}C + BCD = AB + \bar{A}C + BC + BCD = AB + \bar{A}C + BC = AB + \bar{A}C = RHS$$

LAWS OF BOOLEAN ALGEBRA

10. Consensus Theorem (Included Factor Theorem)

Theorem 2: $(A + B)(\bar{A} + C)(B + C) = (A + B)(\bar{A} + C)$

Proof:

$$\begin{aligned} LHS &= (A + B)(\bar{A} + C)(B + C) = (A\bar{A} + AC + B\bar{A} + BC)(B + C) \\ &= (AC + BC + \bar{A}B)(B + C) \\ &= ABC + BC + \bar{A}B + AC + BC + \bar{A}BC = AC + BC + \bar{A}B \end{aligned}$$

$$\begin{aligned} RHS &= (A + B)(\bar{A} + C) \\ &= A\bar{A} + AC + BC + \bar{A}B \\ &= AC + BC + \bar{A}B = LHS \end{aligned}$$

LAWS OF BOOLEAN ALGEBRA

11. Transposition Theorem

Theorem: $AB + \bar{A}C = (A + C)(\bar{A} + B)$

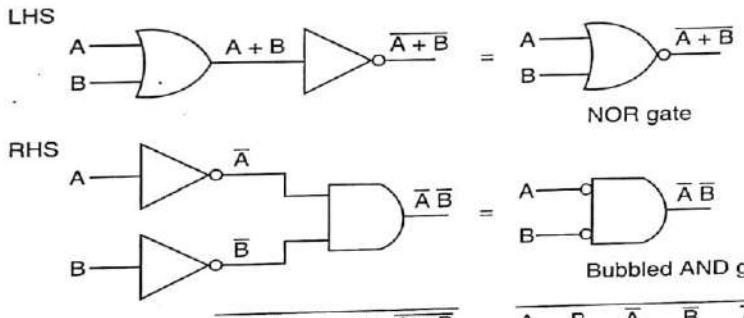
Proof:

$$\begin{aligned} RHS &= (A + C)(\bar{A} + B) \\ &= A\bar{A} + C\bar{A} + AB + CB \\ &= 0 + \bar{A}C + AB + BC \\ &= \bar{A}C + AB + BC(A + \bar{A}) \\ &= AB + ABC + \bar{A}C + \bar{A}BC \\ &= AB + \bar{A}C \\ &= LHS \end{aligned}$$

LAWS OF BOOLEAN ALGEBRA

12. De Morgan's Theorem

Law 1: $\overline{A + B} = \bar{A}\,\bar{B}$



A	B	A + B	$\overline{A+B}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

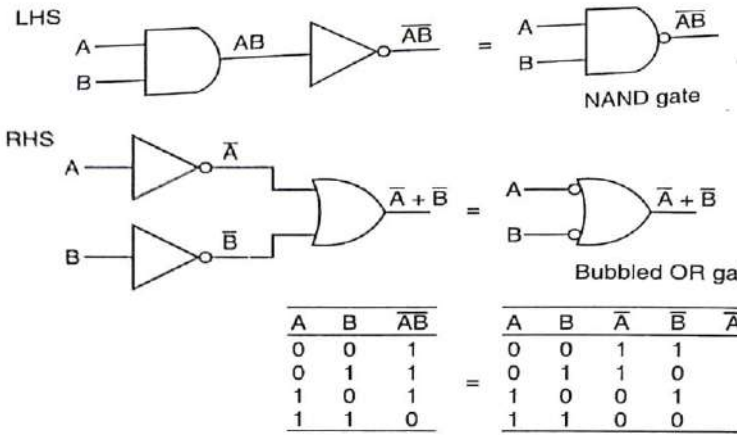
=

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

LAWS OF BOOLEAN ALGEBRA

12. De Morgan's Theorem

Law 2: $\overline{AB} = \overline{A} + \overline{B}$



A	B	$\overline{AB}$
0	0	1
0	1	1
1	0	1
1	1	0

=

A	B	$\overline{A}$	$\overline{B}$	$\overline{A} + \overline{B}$
0	0	1	1	1
0	1	1	0	1
1	0	0	1	1
1	1	0	0	0

Questions

EXAMPLE 5.3 Reduce the expression $f = \overline{\overline{AB} + \overline{A} + AB}$.

Questions

EXAMPLE 5.2 Apply Demorgan's theorem to the expression

$$f = \overline{\overline{AB}(CD + \overline{E}F)(\overline{AB} + \overline{CD})}.$$

BOOLEAN FUNCTIONS AND THEIR REPRESENTATION

1. **Sum-of-products (SOP) form** $f(A, B, C) = \overline{A}B + \overline{B}C.$

2. **Product-of-sums (POS) form** $f(A, B, C) = (\overline{A} + \overline{B})(B + C)$

3. **Truth table form**

A	B	C	f(A, B, C)
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

BOOLEAN FUNCTIONS AND THEIR REPRESENTATION

4. Standard sum-of-products form

In this form, the function is the sum of a number of product terms where each product term contains all the variables of the function either in complemented or uncomplemented form.

$$f(A, B, C) = \bar{A}B + \bar{B}C = \bar{A}B\,(C + \bar{C}) + \bar{B}C(A + \bar{A})$$
$$= \bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}C$$

A product term which contains all the variables of the function either in complemented or uncomplemented form is called a **minterm.** $m_0 = \bar{A}\bar{B}\bar{C}$, $m_1 = \bar{A}\bar{B}C$, $m_2 = \bar{A}B\bar{C}$, $m_3 = \bar{A}BC$, $m_4 = A\bar{B}\bar{C}$, $m_5 = A\bar{B}C$, $m_6 = AB\bar{C}$, and $m_7 = ABC$.

$$f(A, B, C) = m_1 + m_2 + m_3 + m_5$$
$$f(A, B, C) = \Sigma\, m\,(1, 2, 3, 5)$$

BOOLEAN FUNCTIONS AND THEIR REPRESENTATION

5. Standard product-of-sums form

This is derived by considering the combinations for which f = 0. Each term is a sum of all the variables.

$$f(A, B, C) = (\bar{A} + \bar{B} + C\bar{C})(A + B + C\bar{C}) = (\bar{A} + \bar{B} + C)(\bar{A} + \bar{B} + \bar{C})(A + B + C)(A + B + \bar{C})$$

A sum term which contains each of the n variables in either complemented or uncomplemented form is called a **maxterm.**

Maxterms are often represented as $M_0, M_1, M_2, \ldots$, where the suffixes denote their decimal code. Thus the CCF of f may be written as

$$f(A, B, C) = M_0 \cdot M_4 \cdot M_6 \cdot M_7 \quad \text{or simply as}$$
$$f(A, B, C) = \Pi M(0, 4, 6, 7)$$

BOOLEAN FUNCTIONS AND THEIR REPRESENTATION

6. Karnaugh Map

In this representation we put the truth table in a compact form by labelling the rows and columns of a map. This is extremely useful and extensively used in the minimization of functions of 3, 4, 5 or 6 variables.

A \ B	0	1
0	0	1
1	2	3

(a) 2-variable K-map

A \ BC	00	01	11	10
0	0	1	3	2
1	4	5	7	6

(b) 3-variable K-map

AB \ CD	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

(c) 4-variable K-map

0 A

BC \ DE	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

1

BC \ DE	00	01	11	10
00	16	17	19	18
01	20	21	23	22
11	28	29	31	30
10	24	25	27	26

(d) 5-variable K-map

BOOLEAN FUNCTIONS AND THEIR REPRESENTATION

6. Karnaugh Map

0 B 1

A \ CD \ EF	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

A \ CD \ EF	00	01	11	10
00	16	17	19	18
01	20	21	23	22
11	28	29	31	30
10	24	25	27	26

1

A \ CD \ EF	00	01	11	10
00	32	33	35	34
01	36	37	39	38
11	44	45	47	46
10	40	41	43	42

A \ CD \ EF	00	01	11	10
00	48	49	51	50
01	52	53	55	54
11	60	61	63	62
10	56	57	59	58

(e) 6-variable K-map

BOOLEAN FUNCTIONS AND THEIR REPRESENTATION

The possible minterms and maxterms of a 2-variable function f(A, B) are:

$m_0 = \bar{A}\bar{B}$, $m_1 = \bar{A}B$, $m_2 = A\bar{B}$, $m_3 = AB$.

$M_0 = (A + B)$, $M_1 = A + \bar{B}$, $M_2 = \bar{A} + B$, $M_3 = \bar{A} + \bar{B}$.

The possible minterms and maxterms of a 3-variable function f(A,B,C) are:

$m_0 = \bar{A}\bar{B}\bar{C}$, $m_1 = \bar{A}\bar{B}C$, $m_2 = \bar{A}B\bar{C}$, $m_3 = \bar{A}BC$,

$m_4 = A\bar{B}\bar{C}$, $m_5 = A\bar{B}C$, $m_6 = AB\bar{C}$, $m_7 = ABC$.

$M_0 = A + B + C$, $M_1 = A + B + \bar{C}$, $M_2 = A + \bar{B} + C$, $M_3 = A + \bar{B} + \bar{C}$,

$M_4 = \bar{A} + B + C$, $M_5 = \bar{A} + B + \bar{C}$, $M_6 = \bar{A} + \bar{B} + C$, $M_7 = \bar{A} + \bar{B} + \bar{C}$.

BOOLEAN FUNCTIONS AND THEIR REPRESENTATION

The possible minterms and maxterms of a 4-variable function f(A,B,C,D) are:

$m_0 = \bar{A}\bar{B}\bar{C}\bar{D}$, $m_1 = \bar{A}\bar{B}\bar{C}D$, $m_2 = \bar{A}\bar{B}C\bar{D}$, ... $m_{14} = ABC\bar{D}$,

$m_{15} = ABCD$.

$M_0 = A + B + C + D$, $M_1 = A + B + C + \bar{D}$, $M_2 = A + B + \bar{C} + D$, ...

$M_{14} = \bar{A} + \bar{B} + \bar{C} + D$, $M_{15} = \bar{A} + \bar{B} + \bar{C} + \bar{D}$.

The possible minterms and maxterms of a 5-variable function f(A, B, C, D, E) are:

$m_0 = \bar{A}\bar{B}\bar{C}\bar{D}\bar{E}$, $m_1 = \bar{A}\bar{B}\bar{C}\bar{D}E$, $m_2 = \bar{A}\bar{B}\bar{C}D\bar{E}$, ... $m_{30} = ABCD\bar{E}$,

$m_{31} = ABCDE$.

$M_0 = A + B + C + D + E$, $M_1 = A + B + C + D + \bar{E}$,

$M_2 = A + B + C + \bar{D} + E$, ... $M_{30} = \bar{A} + \bar{B} + \bar{C} + \bar{D} + E$,

$M_{31} = \bar{A} + \bar{B} + \bar{C} + \bar{D} + \bar{E}$.

Expansion of a Boolean Expression in SOP Form to the Standard SOP Form

EXAMPLE 5.15 Expand $\bar{A} + \bar{B}$ to minterms and maxterms.

Method 1

$$\begin{aligned}
\bar{A} + \bar{B} &= \bar{A}(B + \bar{B}) + \bar{B}(A + \bar{A}) \\
&= \bar{A}B + \bar{A}\bar{B} + \bar{B}A + \bar{B}\bar{A} \\
&= \bar{A}B + \bar{A}\bar{B} + A\bar{B} + \bar{A}\bar{B} \\
&= \bar{A}B + \bar{A}\bar{B} + A\bar{B} \\
&= 01 + 00 + 10 \\
&= m_1 + m_0 + m_2 \\
&= \Sigma\, m(0, 1, 2)
\end{aligned}$$

Expansion of a Boolean Expression in SOP Form to the Standard SOP Form

EXAMPLE 5.15 Expand $\bar{A} + \bar{B}$ to minterms and maxterms.

Method 2

$$\begin{aligned}
\bar{A} + \bar{B} &= \bar{A} \cdot X + X \cdot \bar{B} \\
&= 0X + X0 \\
&= 00 + 01 + 00 + 10 \\
&= 00 + 01 + 10 \\
&= \Sigma\, m(0, 1, 2)
\end{aligned}$$

Conversion between Canonical Forms

$$f(A, B, C) = \Sigma m(0, 2, 4, 6, 7)$$

This has a complement that can be expressed as

$$\overline{f(A, B, C)} = \Sigma m(1, 3, 5) = m_1 + m_3 + m_5$$

Now if we take the complement of $\overline{f}$ by De Morgan's theorem we obtain f in a different form.

$$f = \overline{(m_1 + m_3 + m_5)} = \overline{m_1} \cdot \overline{m_3} \cdot \overline{m_5} = M_1 M_3 M_5 = \Pi M(1, 3, 5)$$

Expansion of a Boolean Expression in POS Form to the Standard POS Form

EXAMPLE 5.18 Expand $A(\overline{A} + B)(\overline{A} + B + \overline{C})$ to maxterms and minterms.

$$\begin{aligned} A &= A + B\overline{B} + C\overline{C} = (A + B)(A + \overline{B}) + C\overline{C} \\ &= (A + B + C\overline{C})(A + \overline{B} + C\overline{C}) \\ &= (A + B + C)(A + B + \overline{C})(A + \overline{B} + C)(A + \overline{B} + \overline{C}) \\ \overline{A} + B &= \overline{A} + B + C\overline{C} = (\overline{A} + B + C)(\overline{A} + B + \overline{C}) \end{aligned}$$

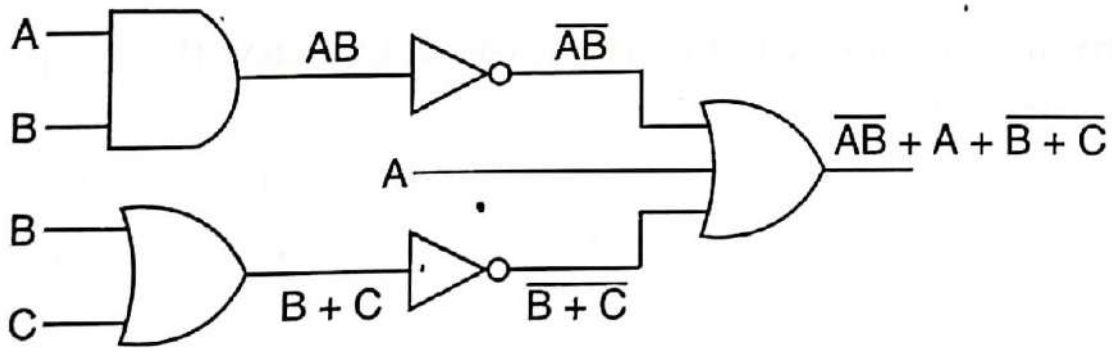
Therefore,

$$A(\overline{A} + B)(\overline{A} + B + \overline{C})$$
$$= (A + B + C)(A + B + \overline{C})(A + \overline{B} + C)\ (A + \overline{B} + \overline{C})(\overline{A} + B + C)(\overline{A} + B + \overline{C})$$
$$= (000)(001)(010)(011)(100)(101)$$
$$= M_0 \cdot M_1 \cdot M_2 \cdot M_3 \cdot M_4 \cdot M_5$$
$$= \Pi\, M(0, 1, 2, 3, 4, 5)$$

The maxterms M_6 and M_7 are missing in the POS form. So, the SOP form will contain the minterms 6 and 7. Therefore, the given expression in the SOP form is $\Sigma\, m(6, 7)$.

Converting Boolean Expressions to Logic

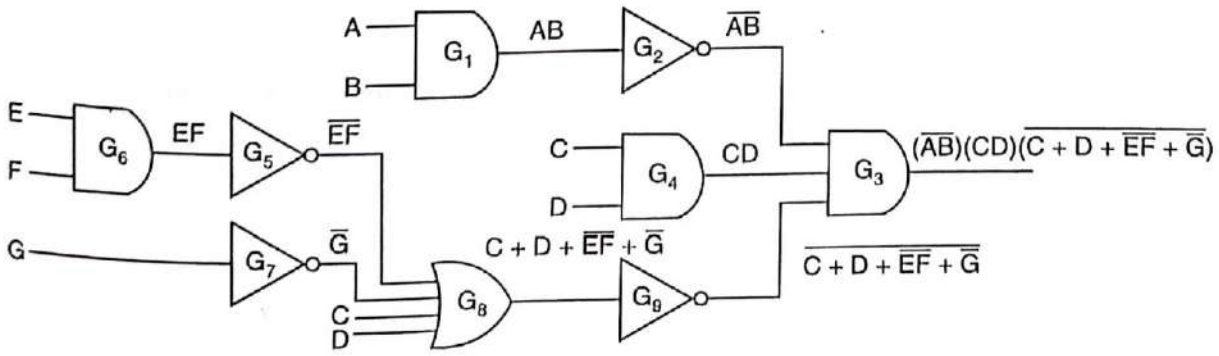
The easiest way to convert a Boolean expression to a logic circuit is to start with the output and work towards the input. Assume that the expression $\overline{AB} + A + \overline{B+C}$ is to be realized using AOI logic. Start with the final expression $\overline{AB} + A + \overline{B+C}$.



Converting Boolean Expressions to Logic

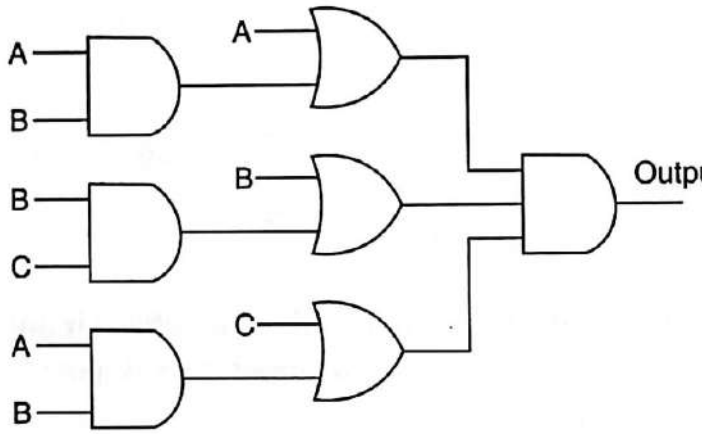
The easiest way to convert a Boolean expression to a logic circuit is to start with the output and work towards the input. Assume that the expression $\overline{AB} + A + \overline{B+C}$ is to be realized using AOI logic. Start with the final expression $\overline{AB} + A + \overline{B+C}$.

Converting Logic to Boolean Expressions



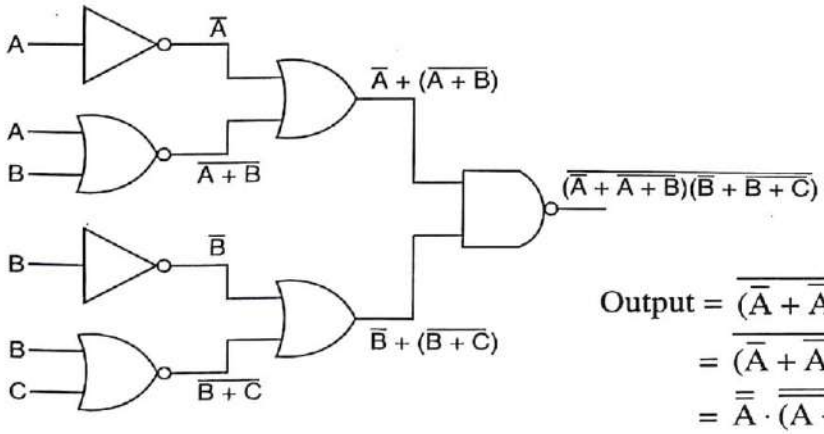
Converting Logic to Boolean Expressions

EXAMPLE 5.22 Write the Boolean expression for the logic diagram given below and simplify it as much as possible and draw the logic diagram that implements the simplified expression.



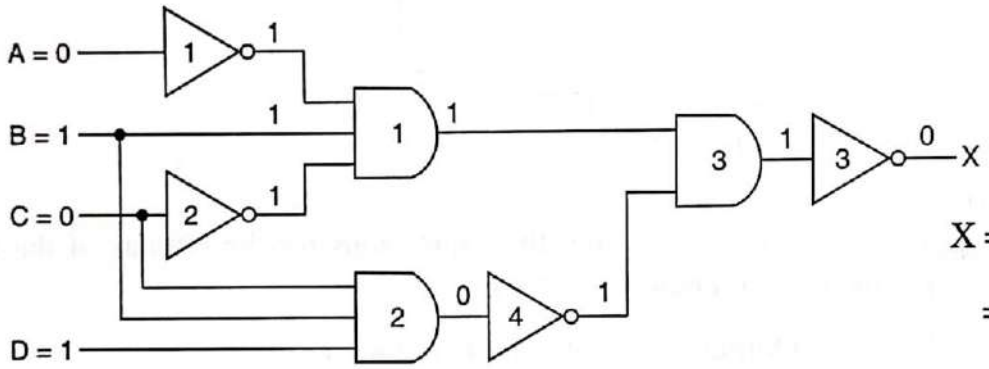
$$
\begin{aligned}
\text{Output} &= (A + AB)(B + BC)(C + AB) \\
&= A(1 + B)B(1 + C)(C + AB) \\
&= AB(C + AB) \\
&= ABC + AB \\
&= AB(1 + C) \\
&= AB
\end{aligned}
$$

Converting Logic to Boolean Expressions



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

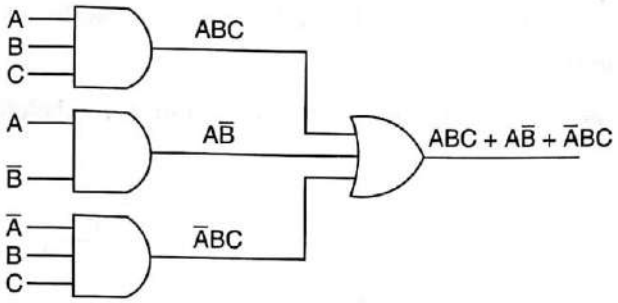
Determination Of Output Level From The Diagram



$$X = \overline{(\bar{A}B\bar{C})(\overline{BCD})}$$
$$= \overline{(\bar{0}\cdot 1\cdot \bar{0})(\overline{1\cdot 0\cdot 1})}$$
$$= \overline{1\cdot \bar{0}}$$
$$= 0$$

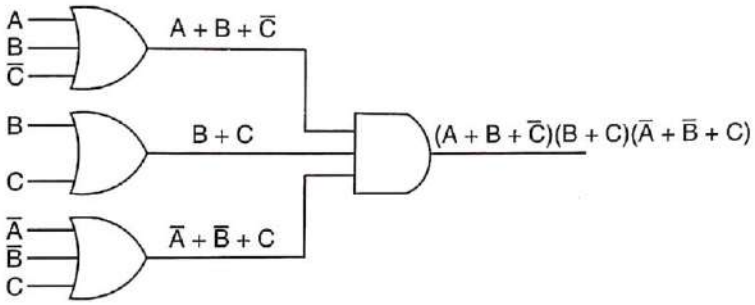
AND/OR/INVERT LOGIC

Boolean expressions, the variable and its complement are assumed to be available. For example, the SOP expression $ABC + A\bar{B} + \bar{A}BC$ can be implemented in A/O logic as shown below.



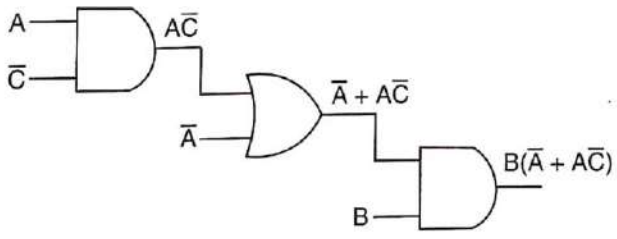
AND/OR/INVERT LOGIC

The POS expression $(A + B + \overline{C})(B + C)(\overline{A} + \overline{B} + C)$ can be implemented using OR and AND gates as shown below.



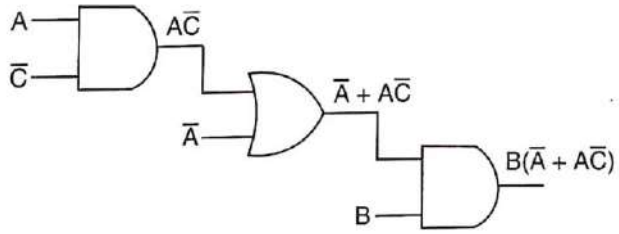
AND/OR/INVERT LOGIC

The expression $AB\overline{C} + \overline{A}B$ $[= B(\overline{A} + A\overline{C})]$ can be implemented in hybrid form as shown below.



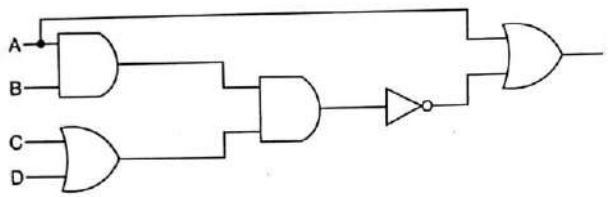
AND/OR/INVERT LOGIC

The expression $AB\overline{C} + \overline{A}B$ [$= B(\overline{A} + A\overline{C})$] can be implemented in hybrid form as shown below.

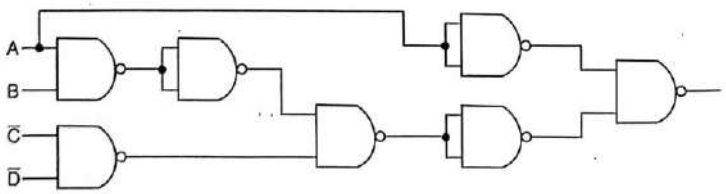


Converting AOI Logic to NAND/NOR Logic

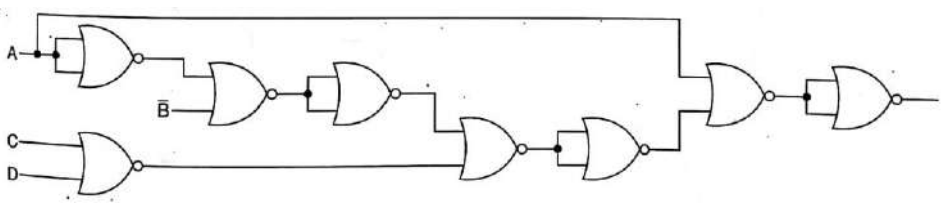
AOI Logic



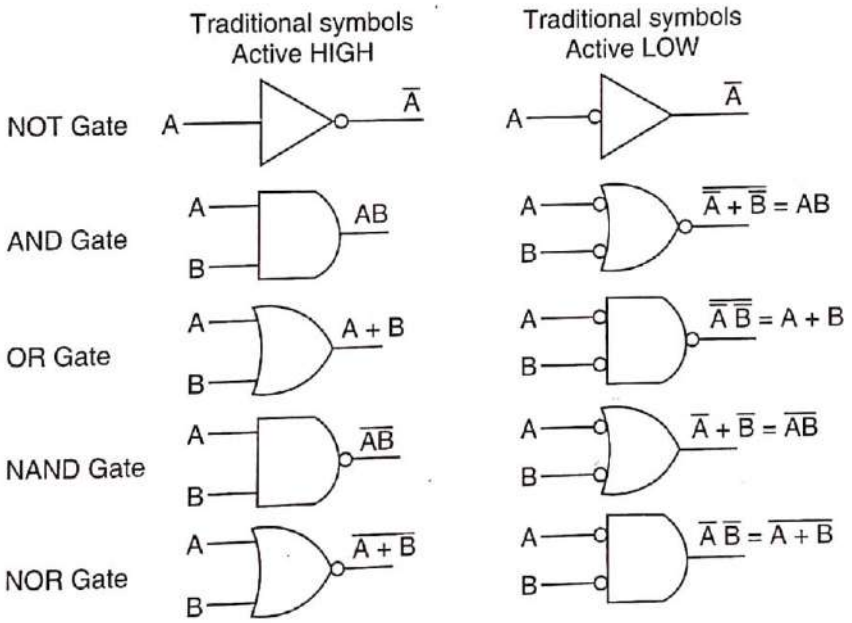
NAND Logic



NOR Logic



Things to Remember



K-Map

- The Karnaugh map (K-map) method, is a systematic method of simplifying the Boolean expression. The K-map is a chart or a graph, composed of an arrangement of adjacent cells, each representing a particular combination of variables in sum or product form.

- Although a K-map can be used for problems involving any number of variables, it becomes tedious for problems involving five or more variables.

- Usually it is limited to six variables.

2-VARIABLE K-MAP

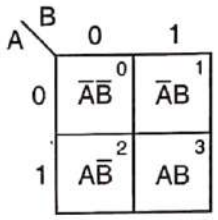
- A two-variable expression can have $2^2 = 4$ possible combinations of the input variables A and B.

Consider the expression $f = \bar{A}\bar{B} + A\bar{B} + AB$. It can be expressed using minterms as

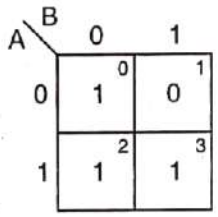
$$F = m_0 + m_2 + m_3 = \Sigma\, m\,(0, 2, 3)$$

Minterm	Inputs A	B	Output f
0	0	0	1
1	0	1	0
2	1	0	1
3	1	1	1

1. Mapping of SOP Expressions

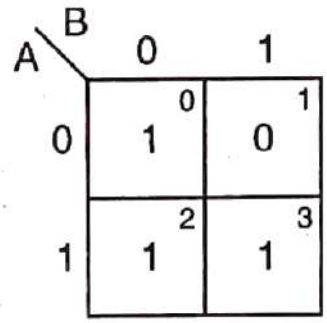


The minterms of a two-variable K-map.

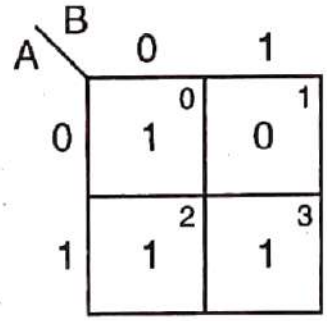


K-map of Σ m(0, 2, 3).

2. Minimization of SOP Expressions



3. Mapping of POS Expressions



EXAMPLE 6.2 Reduce the expression $f = \bar{A}\bar{B} + \bar{A}B + AB$ using mapping.

1. Mapping of SOP Expressions

2. Minimization of SOP Expressions

1. Mapping of POS Expressions

2. Minimization of POS Expressions

EXAMPLE 6.3 Plot the expression $f = (A + B)(\bar{A} + B)(\bar{A} + \bar{B})$ on the K-map.

3-VARIABLE K-MAP

- A 3-variable expression can have $2^3 = 8$ possible combinations of the input variables A, B and C.

A \ BC	00	01	11	10
0	$\overline{A}\overline{B}\overline{C}$ (m_0) 0	$\overline{A}\overline{B}C$ (m_1) 1	$\overline{A}BC$ (m_3) 3	$\overline{A}B\overline{C}$ (m_2) 2
1	$A\overline{B}\overline{C}$ (m_4) 4	$A\overline{B}C$ (m_5) 5	ABC (m_7) 7	$AB\overline{C}$ (m_6) 6

(a) Minterms

A \ BC	00	01	11	10
0	$A+B+C$ (M_0) 0	$A+B+\overline{C}$ (M_1) 1	$A+\overline{B}+\overline{C}$ (M_3) 3	$A+\overline{B}+C$ (M_2) 2
1	$\overline{A}+B+C$ (M_4) 4	$\overline{A}+B+\overline{C}$ (M_5) 5	$\overline{A}+\overline{B}+\overline{C}$ (M_7) 7	$\overline{A}+\overline{B}+C$ (M_6) 6

(b) Maxterms

Figure 6.10 The three-variable K-map.

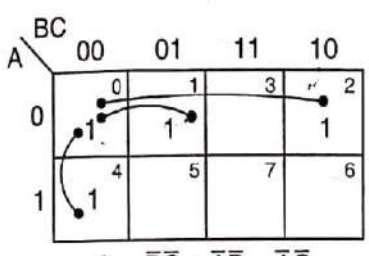
3-VARIABLE K-MAP in SOP form

Map the expression $f = \overline{A}\overline{B}C + A\overline{B}C + \overline{A}B\overline{C} + AB\overline{C} + ABC$.

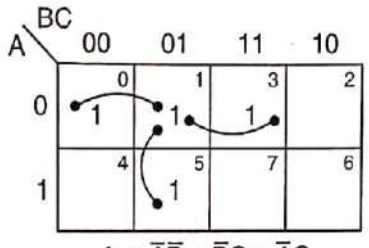
3-VARIABLE K-MAP in POS form

Map the expression $f = (A + B + C)(\bar{A} + B + \bar{C})(\bar{A} + \bar{B} + \bar{C})(A + \bar{B} + \bar{C})(\bar{A} + \bar{B} + C)$.

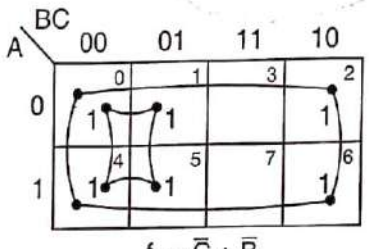
Rules for Pairing of Adjacent Cells in SOP form



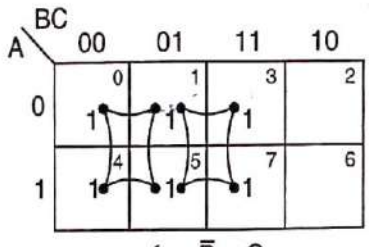
$f_1 = \bar{B}\bar{C} + \bar{A}\bar{B} + \bar{A}\bar{C}$



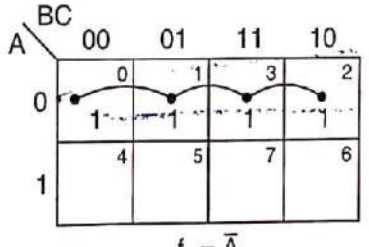
$f_2 = \bar{A}\bar{B} + \bar{B}C + \bar{A}C$



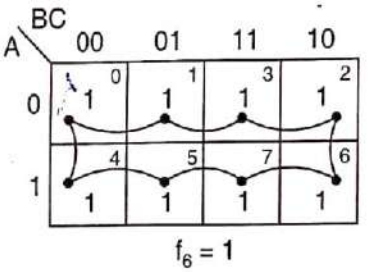
$f_3 = \bar{C} + \bar{B}$



$f_4 = \bar{B} + C$

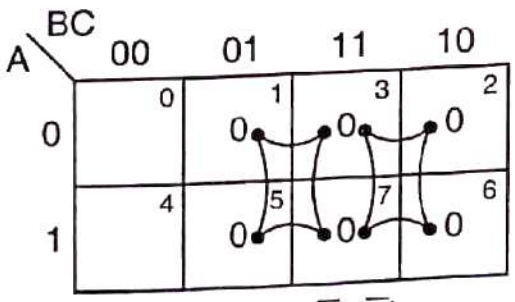


$f_5 = \bar{A}$

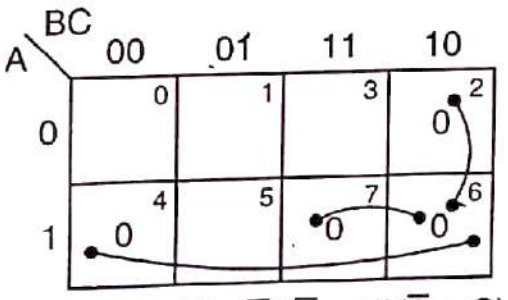


$f_6 = 1$

Rules for Pairing of Adjacent Cells in POS form



(a) $f_1 = (\overline{C})(\overline{B})$



(b) $f_2 = (\overline{A} + \overline{B})(\overline{B} + C)(\overline{A} + C)$

Questions

Reduce the expression $f = \Sigma m\,(0, 2, 3, 4, 5, 6)$ using mapping and implement it in AOI logic as well as in NAND logic.

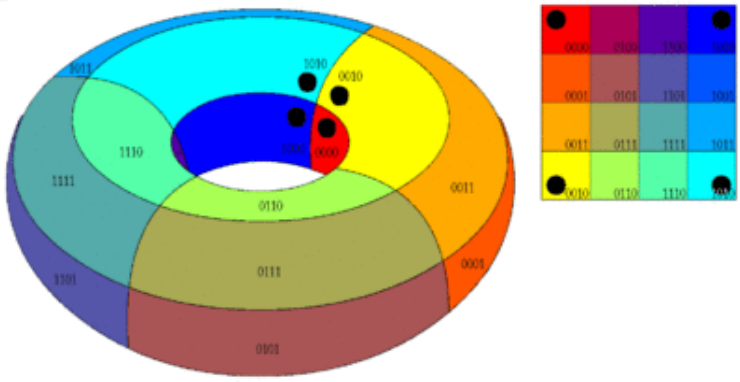
Questions

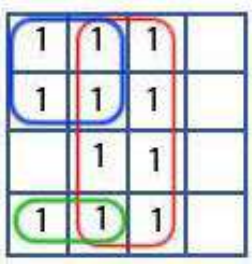
Reduce the expression $f = \Pi M (0, 1, 2, 3, 4, 7)$ using mapping and implement it in AOI logic as well as in NOR logic.

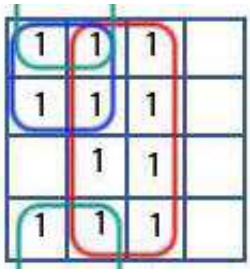
Questions

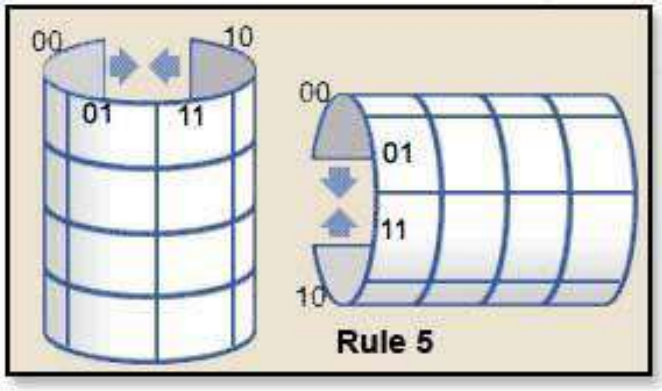
Obtain the real minimal expression for $f = \Sigma m (1, 2, 4, 6, 7)$ and implement it using universal gates.

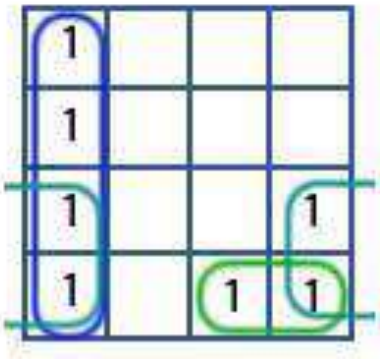
4-VARIABLE K-MAP

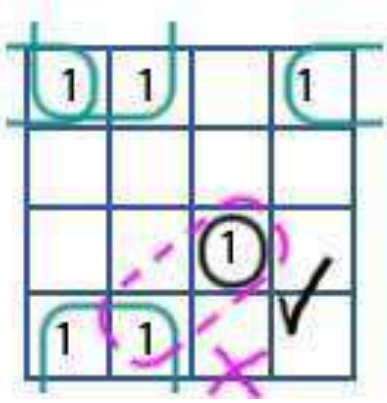










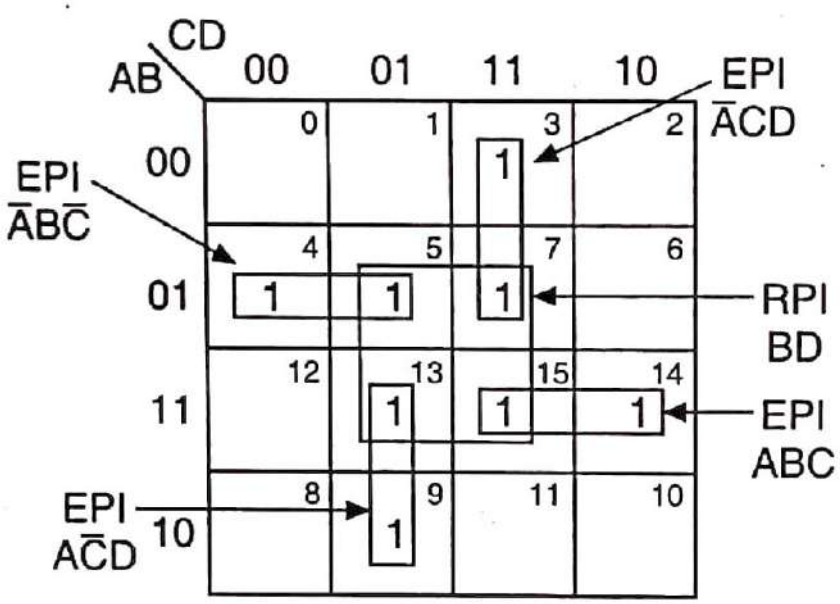


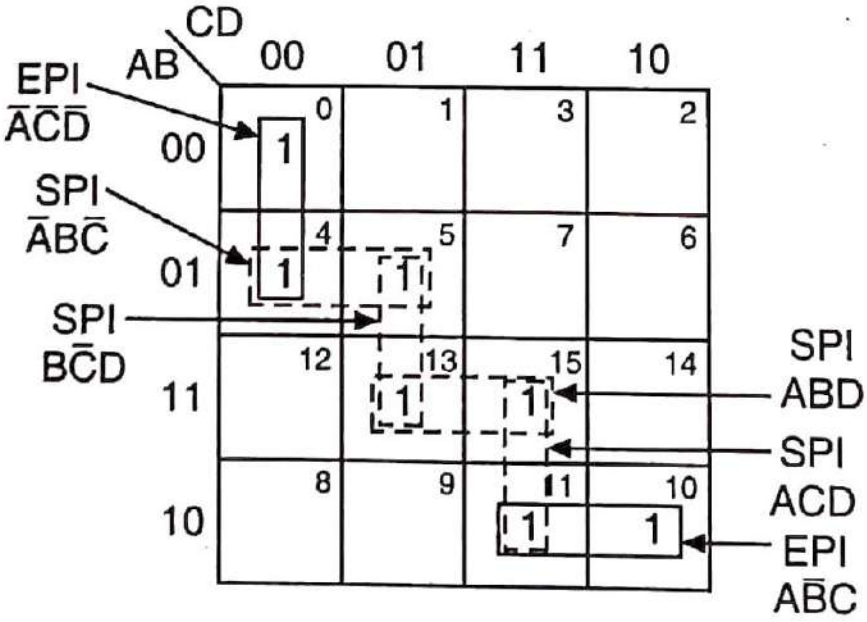
4-VARIABLE K-MAP

Reduce using mapping the expression $f = \Sigma\, m(2, 3, 6, 7, 8, 10, 11, 13, 14)$.

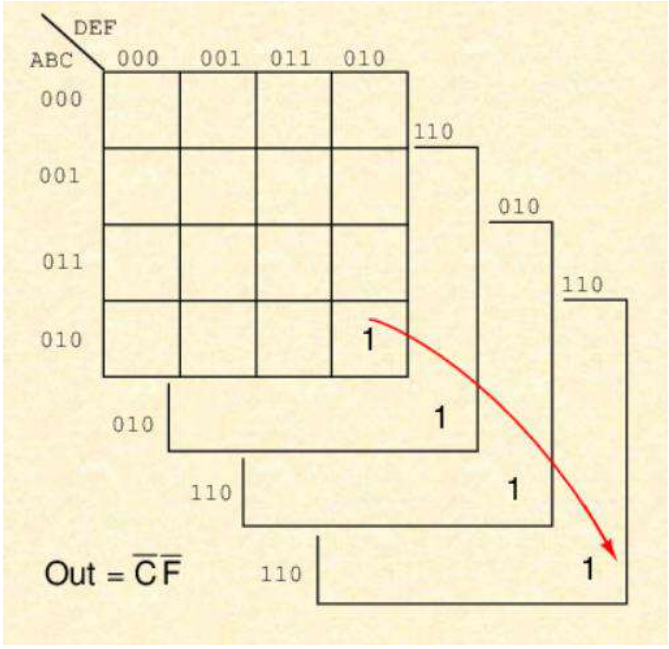
EXAMPLE 6.12 Reduce using mapping the expression $f = \Sigma\, m(0, 1, 2, 3, 5, 7, 8, 9, 10, 12, 13)$ and implement the real minimal expression in universal logic.

Prime Implicants, Essential Prime Implicants, Redundant Prime Implicants and Selective Prime Implicants

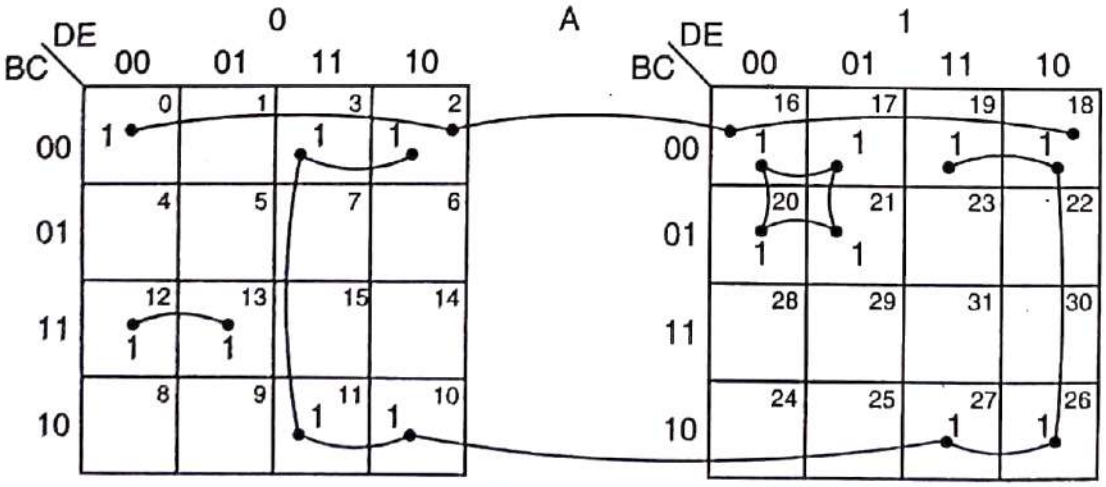




5-VARIABLE K-MAP



EXAMPLE 6.16 Reduce the following expression in SOP and POS forms using mapping:
$f = \Sigma\, m(0, 2, 3, 10, 11, 12, 13, 16, 17, 18, 19, 20, 21, 26, 27)$



$f = \bar{A}BC\bar{D} + \bar{B}\bar{C}\bar{E} + A\bar{B}\bar{D} + \bar{C}D$

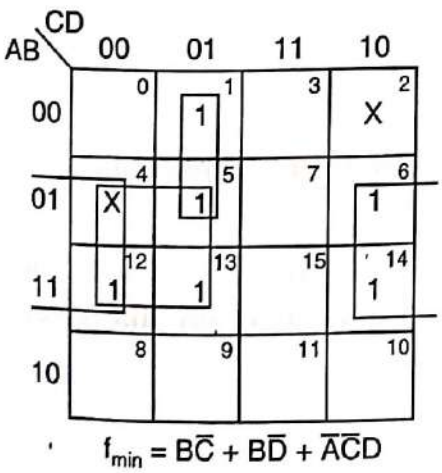
DON'T CARE COMBINATIONS

EXAMPLE 6.20 Reduce the expression $f = \Sigma\, m(1, 5, 6, 12, 13, 14) + d(2, 4)$ and implement the real minimal expression in universal logic.

The given expression written in the POS form is $f = \Pi\, M(0, 3, 7, 8, 9, 10, 11, 15).\ \Pi\, d(2, 4)$.

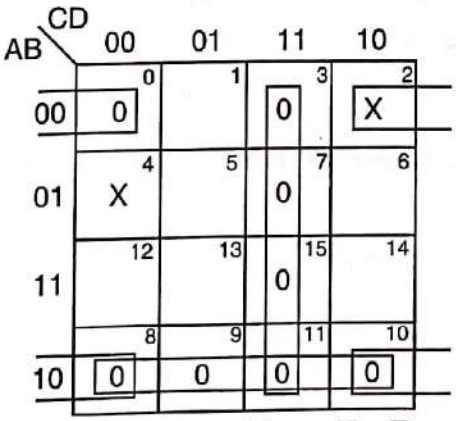
DON'T CARE COMBINATIONS

The given expression written in the POS form is $f = \Pi M(0, 3, 7, 8, 9, 10, 11, 15). \Pi d(2, 4)$.



$f_{min} = B\overline{C} + B\overline{D} + \overline{A}\overline{C}D$

(a) SOP K-map



$f_{min} = (B + D)(\overline{A} + B)(\overline{C} + \overline{D})$

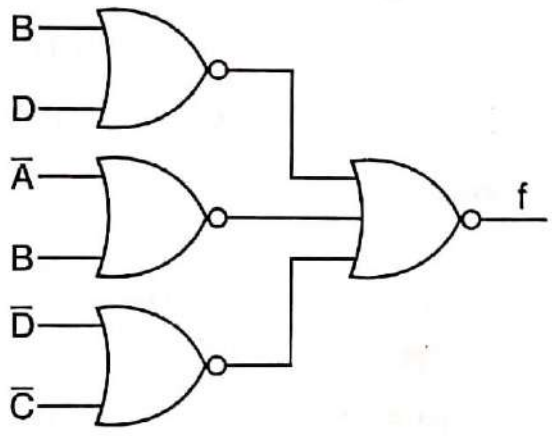
(b) POS K-map

SOP minimal is

$$f_{min} = B\overline{C} + B\overline{D} + \overline{A}\,\overline{C}D$$

POS minimal is

$$f_{min} = (B + D)(\overline{A} + B)(\overline{C} + \overline{D}) = \overline{\overline{(B + D)} + \overline{(\overline{A} + B)} + \overline{(\overline{C} + \overline{D})}}$$



(c) NOR logic

Mapping when the function is not expressed in MINTERMS or (MAXTERMS)

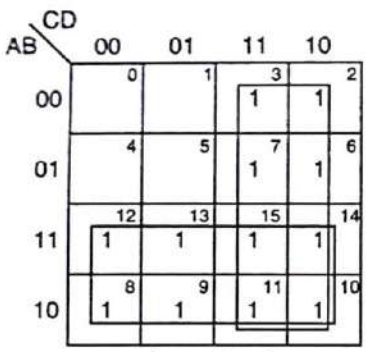
EXAMPLE 6.26 Make a K-map of the following expression and obtain the minimal SOP and POS forms.

$$f = AB + A\overline{C} + C + AD + A\overline{B}C + ABC$$

By expansion on the map we see that $f = (2, 3, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15)$. So the POS expression is $f = (0, 1, 4, 5)$.

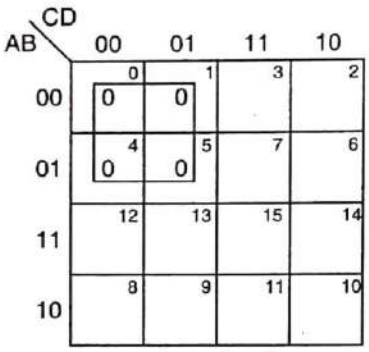
SOP minimal is $f_{min} = A + C$

POS minimal is $f_{min} = A + C$



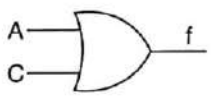
$f_{min} = A + C$

(a) SOP K-map



$f_{min} = A + C$

(b) POS K-map



(c) Logic diagram

LIMITATIONS OF KARNAUGH MAPS

- The K-map method of simplification is convenient as long as the number of variables does not exceed six.
- As the number of variables increases it becomes difficult to make judgments about which combinations form the minimal expression. It is almost impossible to work with problems of 7 or more variables using K-maps.
- Another important point is that the K-map simplification is a manual technique and simplification process is heavily dependent on the human abilities. It cannot be programmed.
- To meet this need, W.V. Quine and E.J. McCluskey developed an exact tabular method to simplify the Boolean expressions. This method is called the Quine-McCluskey method or tabular method.

COMBINATIONAL LOGIC DESIGN

- Logic circuits for digital systems may be combinational or sequential.

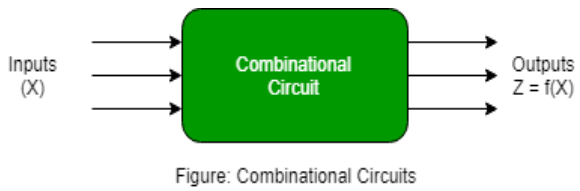


Figure: Combinational Circuits

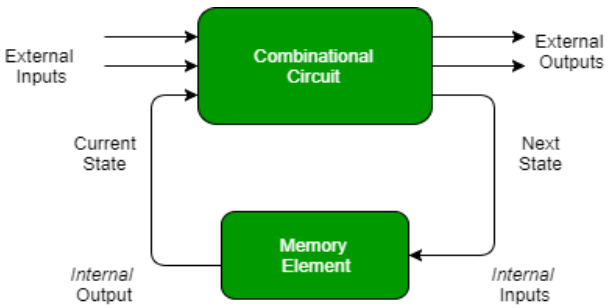


Figure: Sequential Circuit

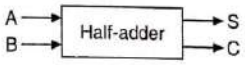
- The output of a combinational circuit depends on its present inputs only.
- Combinational circuits perform a specific information processing operation fully specified logically by a set of Boolean functions.
- A combinational circuit consists of input variables, logic gates, and output variables.

Adders

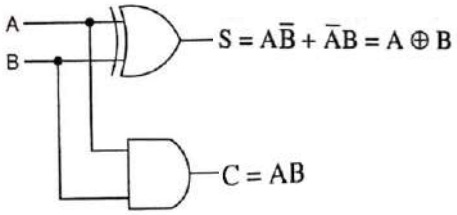
Half-Adder

Inputs		Outputs	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

(a) Truth table



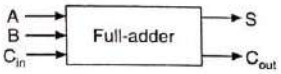
(b) Block diagram



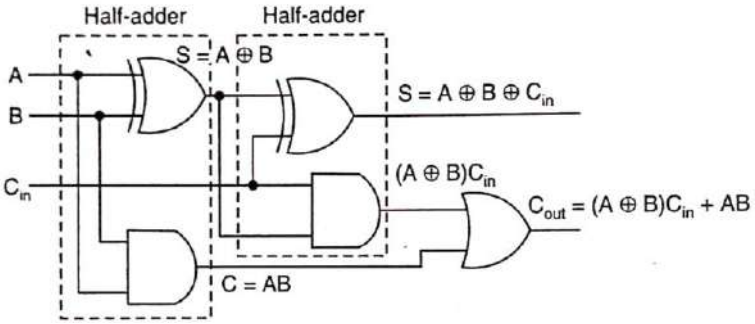
Full-Adder

Inputs			Sum	Carry
A	B	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

(a) Truth table



(b) Block diagram



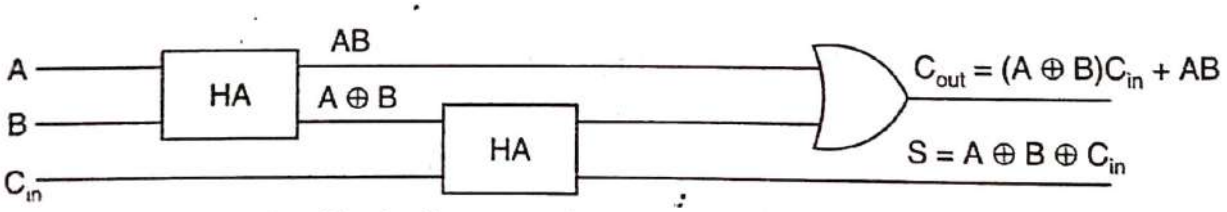


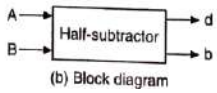
Figure 7.8 Block diagram of a full-adder using two half-adders.

Subtractors

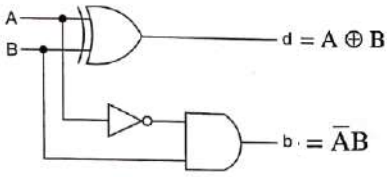
Half-Subtractor

Inputs		Outputs	
A	B	d	b
0	0	0	0
1	0	1	0
1	1	0	0
0	1	1	1

(a) Truth table



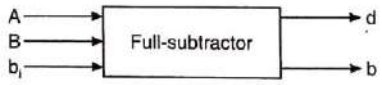
(b) Block diagram



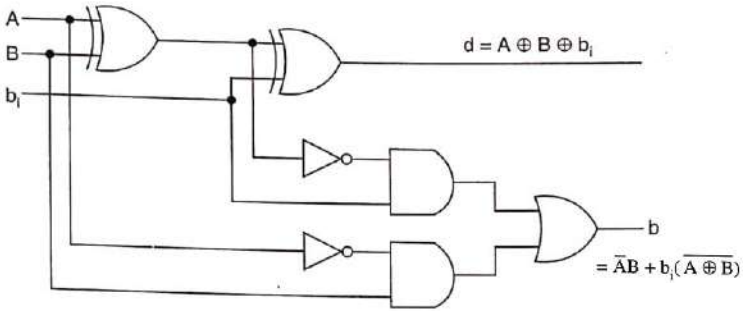
Full-Subtractor

Inputs			Difference	Borrow
A	B	b_i	d	b
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

(a) Truth table

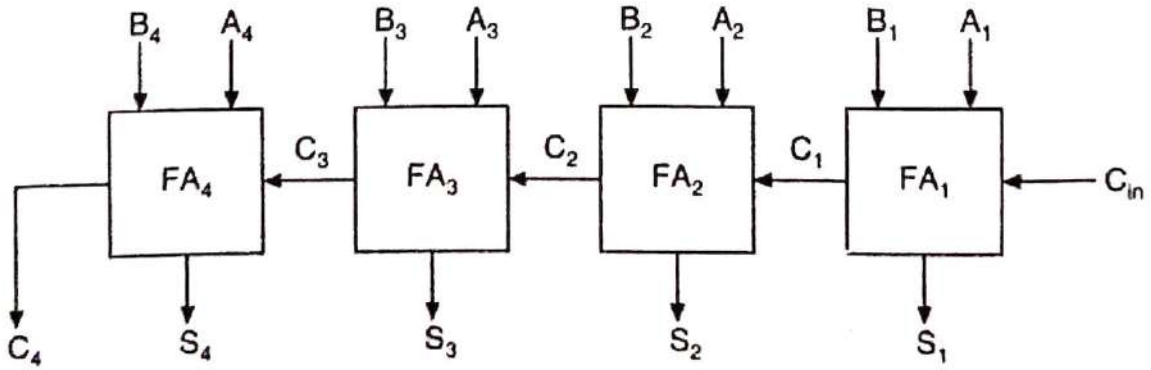


(b) Block diagram



BINARY PARALLEL ADDER/ The Ripple Carry Adder

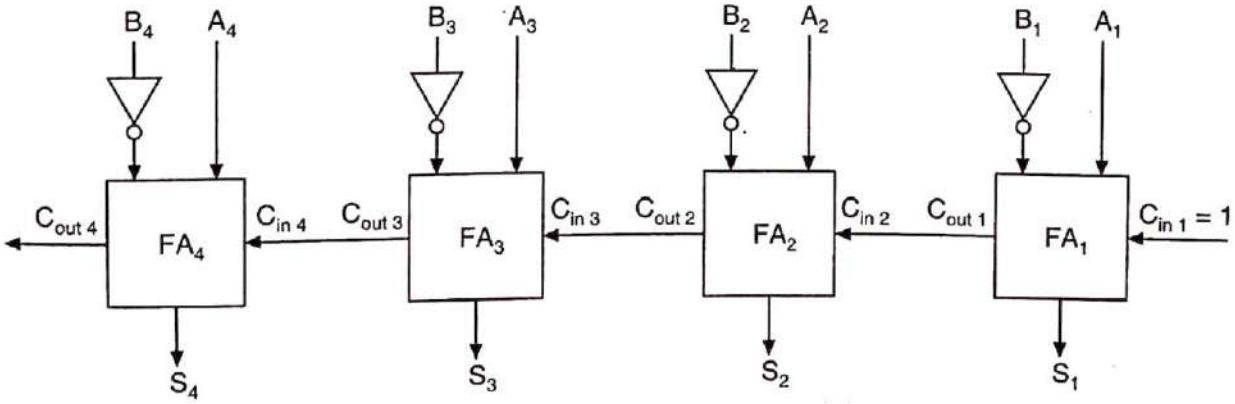
A binary parallel adder is a digital circuit that adds two binary numbers in parallel form and produces the arithmetic sum of those numbers in parallel form. It consists of full adders connected in a chain, with the output carry from each full-adder connected to the input carry of the next full adder in the chain.



4-BIT PARALLEL SUBTRACTOR

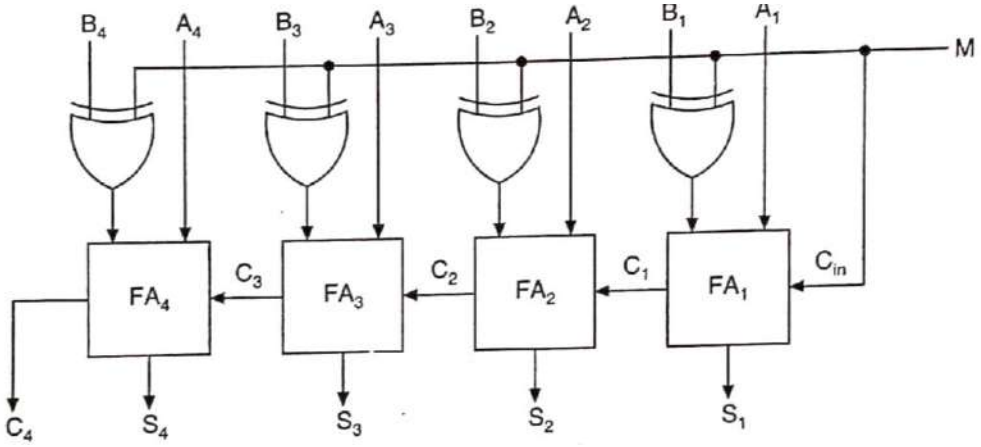
The subtraction of binary numbers can be carried out most conveniently by means of complements.

Remember that the subtraction A - B can be done by taking the 2's complement of B and adding it to A-



BINARY ADDER-SUBTRACTOR

Here the addition and subtraction operations are combined into one circuit with one common binary adder. This is done by including an X-OR gate with each full-adder. The mode input M controls the operation. When M = 0, the circuit is an adder, and when M = 1, the circuit becomes a subtractor.

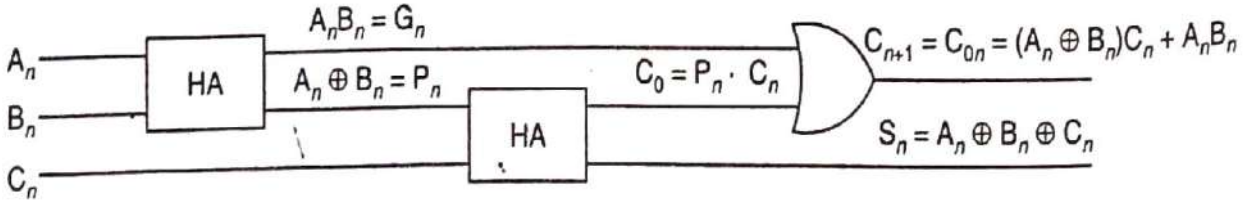


THE LOOK-AHEAD-CARRY ADDER

The look-ahead-carry adder speeds up the process by eliminating this ripple carry delay.

It examines all the input bits simultaneously and also generates the carry-in bits for all the stages simultaneously.

The method of speeding up the addition process is based on the two additional functions of the full-adder, called carry generate and carry propagate functions.



For the final sum and carry outputs of the *n*th stage, we get the following Boolean expressions.

$$S_n = P_n \oplus C_n \text{ where } P_n = A_n \oplus B_n$$

$$C_{on} = C_{n+1} = G_n + P_nC_n \text{ where } G_n = A_n \cdot B_n$$

Observe the recursive nature of the expression for the output carry at the nth stage which becomes the input carry for the (n + l)th stage.

By successive substitution, it is possible to express the output carry of a higher significant stage in terms of the applied input variables A, B and the carry-in to the LSB adder.

The carry-in to each stage is the carry-out of the previous stage.

Based on these, the expressions for the carry-outs of various full adders are as follows:

$$C_1 = G_0 + P_0 \cdot C_0$$

$$C_2 = G_1 + P_1 \cdot C_1 = G_1 + P_1 \cdot G_0 + P_1 \cdot P_0 \cdot C_0$$

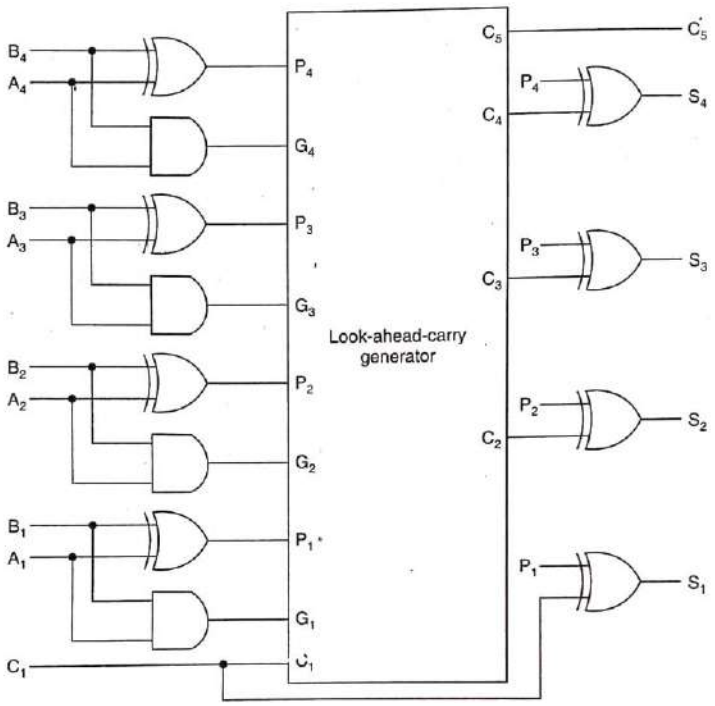
$$C_3 = G_2 + P_2 \cdot C_2 = G_2 + P_2 \cdot G_1 + P_2 \cdot P_1 \cdot G_0 + P_2 \cdot P_1 \cdot P_0 \cdot C_0$$

$$C_4 = G_3 + P_3 \cdot C_3 = G_3 + P_3 \cdot G_2 + P_3 \cdot P_2 \cdot G_1 + P_3 \cdot P_2 \cdot P_1 \cdot G_0 + P_3 \cdot P_2 \cdot P_1 \cdot P_0 \cdot C_0$$

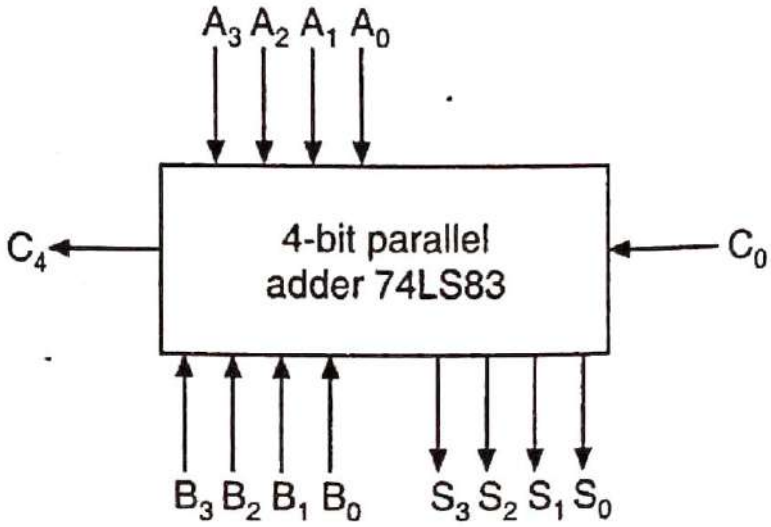
The general expression for n stages designated as 0 through $(n-1)$ would be

$$C_n = G_{n-1} + P_{n-1} \cdot C_{n-1} = G_{n-1} + P_{n-1} \cdot G_{n-2} + P_{n-1} \cdot P_{n-2} \cdot G_{n-3} + \ldots + P_{n-1} \cdot \ldots P_0 \cdot C_0$$

4-bit look-ahead-carry adder



IC PARALLEL ADDERS/ 74LS83



Logic symbol of 74LS83

CODE CONVERTERS

- **Design of a 4-bit Binary-to-Gray Code Converter**

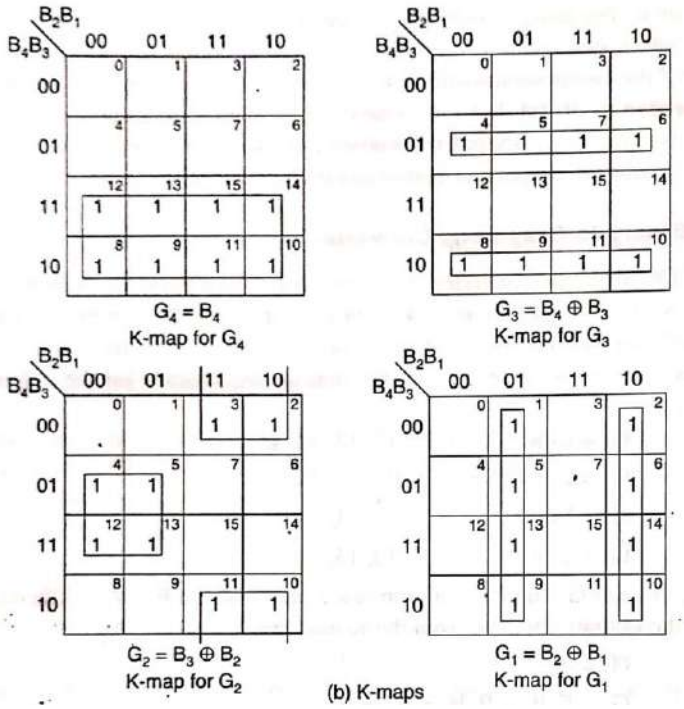
- **Design of a 4-bit Gray-to-Binary Code Converter**

Design of a 4-bit Binary-to-Gray Code Converter

4-bit binary				4-bit Gray			
B_4	B_3	B_2	B_1	G_4	G_3	G_2	G_1
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

$$G_4 = B_4$$
$$G_3 = \bar{B}_4 B_3 + B_4 \bar{B}_3 = B_4 \oplus B_3$$
$$G_2 = \bar{B}_3 B_2 + B_3 \bar{B}_2 = B_3 \oplus B_2$$
$$G_1 = \bar{B}_2 B_1 + B_2 \bar{B}_1 = B_2 \oplus B_1$$

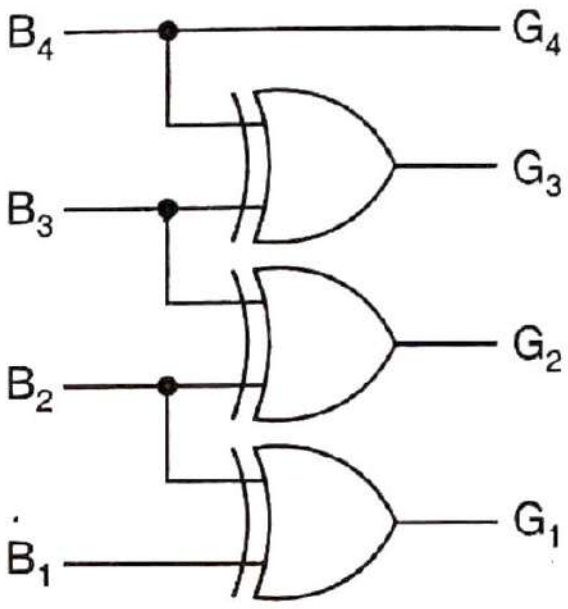
$$G_4 = \Sigma m(8, 9, 10, 11, 12, 13, 14, 15)$$
$$G_3 = \Sigma m(4, 5, 6, 7, 8, 9, 10, 11)$$
$$G_2 = \Sigma m(2, 3, 4, 5, 10, 11, 12, 13)$$
$$G_1 = \Sigma m(1, 2, 5, 6, 9, 10, 13, 14)$$



(b) K-maps

$$G_4 = B_4$$
$$G_3 = \bar{B}_4 B_3 + B_4 \bar{B}_3 = B_4 \oplus B_3$$
$$G_2 = \bar{B}_3 B_2 + B_3 \bar{B}_2 = B_3 \oplus B_2$$
$$G_1 = \bar{B}_2 B_1 + B_2 \bar{B}_1 = B_2 \oplus B_1$$

$$G_4 = B_4$$
$$G_3 = \overline{B}_4 B_3 + B_4 \overline{B}_3 = B_4 \oplus B_3$$
$$G_2 = \overline{B}_3 B_2 + B_3 \overline{B}_2 = B_3 \oplus B_2$$
$$G_1 = \overline{B}_2 B_1 + B_2 \overline{B}_1 = B_2 \oplus B_1$$



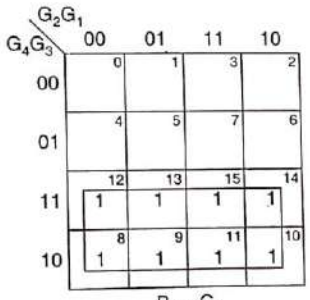
$$G_4 = B_4$$
$$G_3 = \overline{B}_4 B_3 + B_4 \overline{B}_3 = B_4 \oplus B_3$$
$$G_2 = \overline{B}_3 B_2 + B_3 \overline{B}_2 = B_3 \oplus B_2$$
$$G_1 = \overline{B}_2 B_1 + B_2 \overline{B}_1 = B_2 \oplus B_1$$

Design of a 4-bit Gray-to-Binary Code Converter

4-bit Gray				4-bit binary			
G_4	G_3	G_2	G_1	B_4	B_3	B_2	B_1
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	1	0	0	1	0
0	0	1	0	0	0	1	1
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
0	1	0	1	0	1	1	0
0	1	0	0	0	1	1	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	1	1	0	1	0
1	1	1	0	1	0	1	1
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	0	0	1	1	1	1	0
1	0	0	0	1	1	1	1

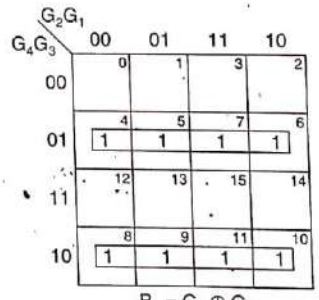
$B_4 = \Sigma m(8, 9, 10, 11, 12, 13, 14, 15)$
$B_3 = \Sigma m(4, 5, 6, 7, 8, 9, 10, 11)$
$B_2 = \Sigma m(2, 3, 4, 5, 8, 9, 14, 15)$
$B_1 = \Sigma m(1, 2, 4, 7, 8, 11, 13, 14)$

$B_4 = \Sigma m(8, 9, 10, 11, 12, 13, 14, 15)$
$B_3 = \Sigma m(4, 5, 6, 7, 8, 9, 10, 11)$
$B_2 = \Sigma m(2, 3, 4, 5, 8, 9, 14, 15)$
$B_1 = \Sigma m(1, 2, 4, 7, 8, 11, 13, 14)$



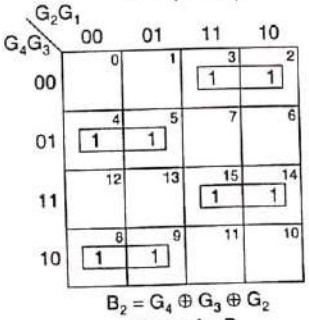
$B_4 = G_4$

K-map for B_4



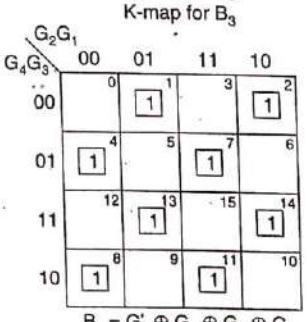
$B_3 = G_4 \oplus G_3$

K-map for B_3



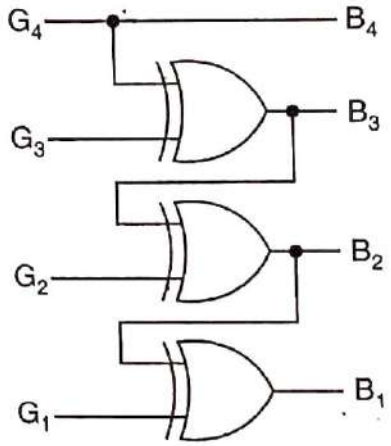
$B_2 = G_4 \oplus G_3 \oplus G_2$

K-map for B_2



$B_1 = G_4' \oplus G_3 \oplus G_2 \oplus G_1$

K-map for B_1



Design of a 4-bit BCD-to-XS-3 Code Converter

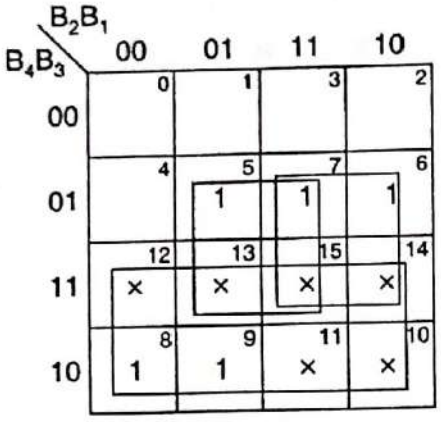
8421 code				XS-3 code			
B_4	B_3	B_2	B_1	X_4	X_3	X_2	X_1
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

$X_4 = \Sigma\, m(5, 6, 7, 8, 9) + d(10, 11, 12, 13, 14, 15)$
$X_3 = \Sigma\, m(1, 2, 3, 4, 9) + d(10, 11, 12, 13, 14, 15)$
$X_2 = \Sigma\, m(0, 3, 4, 7, 8) + d(10, 11, 12, 13, 14, 15)$
$X_1 = \Sigma\, m(0, 2, 4, 6, 8) + d(10, 11, 12, 13, 14, 15)$

The minimal expressions are

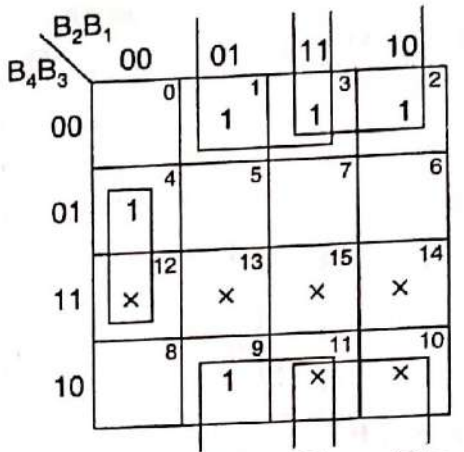
$X_4 = B_4 + B_3B_2 + B_3B_1$
$X_3 = B_3\overline{B}_2\overline{B}_1 + \overline{B}_3B_1 + \overline{B}_3B_2$
$X_2 = \overline{B}_2\overline{B}_1 + B_2B_1$
$X_1 = \overline{B}_1$

$X_4 = \Sigma\, m(5, 6, 7, 8, 9) + d(10, 11, 12, 13, 14, 15)$



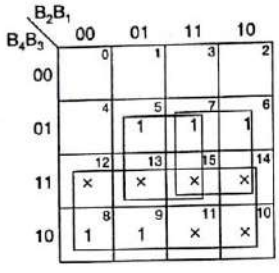
$X_4 = B_4 + B_3B_2 + B_3B_1$
K-map for X_4

$$X_3 = \Sigma\, m(1, 2, 3, 4, 9) + d(10, 11, 12, 13, 14, 15)$$



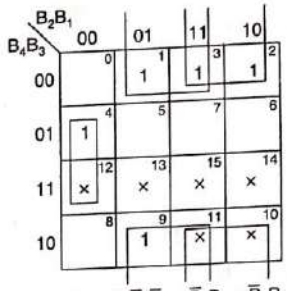
$X_3 = B_3\overline{B}_2\overline{B}_1 + \overline{B}_3B_1 + \overline{B}_3B_2$

K-map for X_3



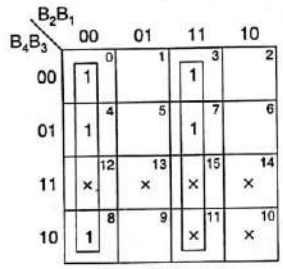
$X_4 = B_4 + B_3B_2 + B_3B_1$

K-map for X_4



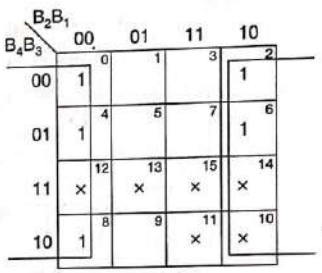
$X_3 = B_3\overline{B}_2\overline{B}_1 + \overline{B}_3B_1 + \overline{B}_3B_2$

K-map for X_3



$X_2 = \overline{B}_2\overline{B}_1 + B_2B_1$

K-map for X_2



$X_1 = \overline{B}_1$

K-map for X_1

(c) K-maps

COMPARATORS

A comparator is a logic circuit used to compare the magnitudes of two binary numbers. Depending on the design, it may either simply provide an output that is active (goes HIGH for example) when the two numbers are equal, or additionally provide outputs that signify which of the numbers is greater when equality does not hold.

The X-NOR gate (coincidence gate) is a basic comparator, because its output is a 1 only if its two input bits are equal, i.e. the output is a 1 if and only if the input bits coincide.

COMPARATORS

Two binary numbers are equal, if and only if all their corresponding bits coincide.

For example, two 4-bit binary numbers, A3A2A1A0 and B3B2B1B0 are equal, if and only if, A3 = B3, A2 = B2, A1 = B1 and A0 = B0.

$$\text{EQUALITY} = (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1)(A_0 \odot B_0)$$

1 -bit Magnitude Comparator

The logic for a 1-bit magnitude comparator: Let the 1-bit numbers be $A = A_0$ and $B = B_0$.
If $A_0 = 1$ and $B_0 = 0$, then $A > B$.
Therefore,

$$A > B: G = A_0\overline{B}_0$$

If $A_0 = 0$ and $B_0 = 1$, then $A < B$.
Therefore,

$$A < B: L = \overline{A}_0 B_0$$

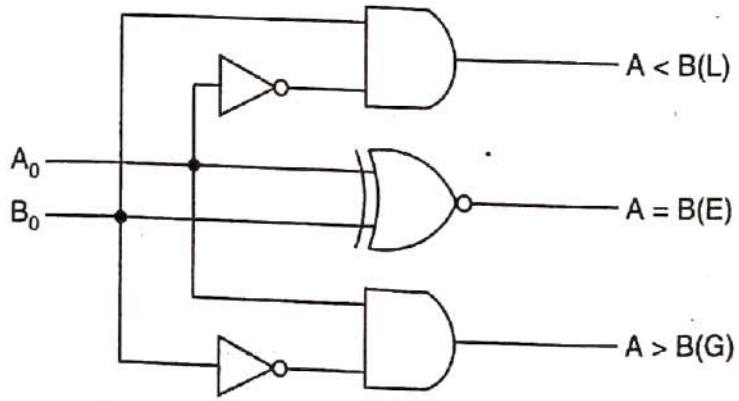
If A_0 and B_0 coincide, i.e. $A_0 = B_0 = 0$ or if $A_0 = B_0 = 1$, then $A = B$.
Therefore,

$$A = B : E = A_0 \odot B_0$$

1 -bit Magnitude Comparator

A_0	B_0	L	E	G
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

(a) Truth table



(b) Logic diagram

Figure 7.56 1-bit comparator.

2-bit Magnitude Comparator

The logic for a 2-bit magnitude comparator: Let the two 2-bit numbers be $A = A_1A_0$ and $B = B_1B_0$.

1. If $A_1 = 1$ and $B_1 = 0$, then $A > B$ or
2. If A_1 and B_1 coincide and $A_0 = 1$ and $B_0 = 0$, then $A > B$. So the logic expression for $A > B$ is

$$A > B : G = A_1\bar{B}_1 + (A_1 \odot B_1)A_0\bar{B}_0$$

1. If $A_1 = 0$ and $B_1 = 1$, then $A < B$ or
2. If A_1 and B_1 coincide and $A_0 = 0$ and $B_0 = 1$, then $A < B$. So the expression for $A < B$ is

$$A < B : L = \bar{A}_1B_1 + (A_1 \odot B_1)\bar{A}_0B_0$$

If A_1 and B_1 coincide and if A_0 and B_0 coincide then $A = B$. So the expression for $A = B$ is

$$A = B : E = (A_1 \odot B_1)(A_0 \odot B_0)$$

2-bit Magnitude Comparator

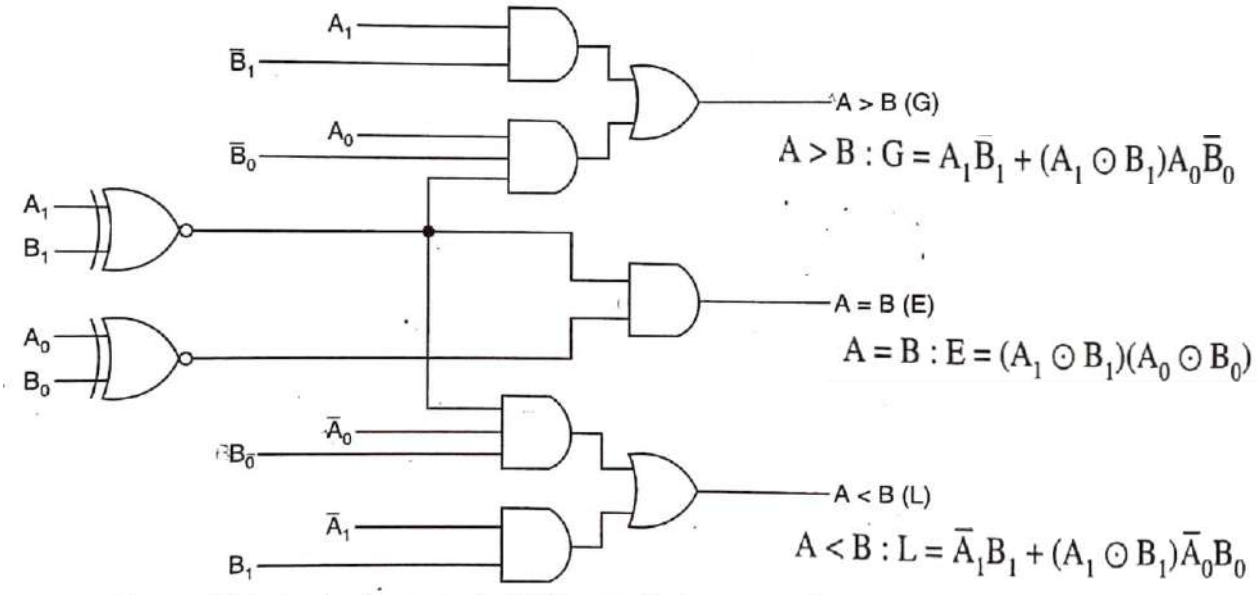
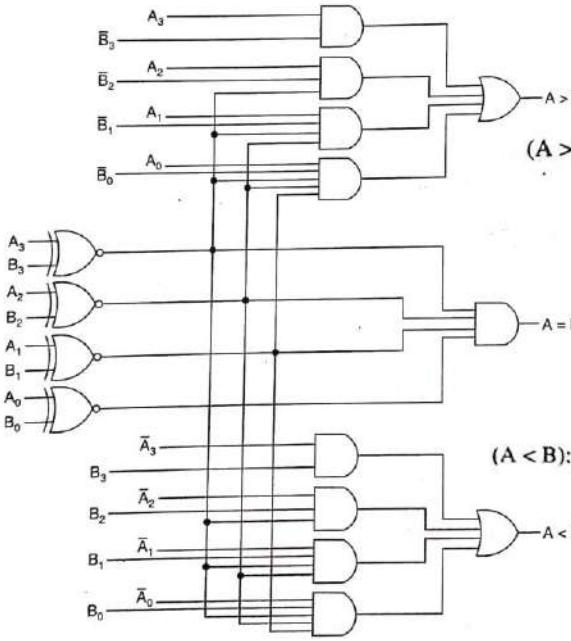


Figure 7.57 Logic diagram of a 2-bit magnitude comparator.

4-bit Magnitude Comparator



$$(A > B) : G = A_3\bar{B}_3 + (A_3 \odot B_3)A_2\bar{B}_2 + (A_3 \odot B_3)(A_2 \odot B_2)A_1\bar{B}_1 + (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1)A_0\bar{B}_0$$

$$(A = B): E = (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1)(A_0 \odot B_0)$$

$$(A < B): L = \bar{A}_3B_3 + (A_3 \odot B_3)\bar{A}_2B_2 + (A_3 \odot B_3)(A_2 \odot B_2)\bar{A}_1B_1 + (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1)\bar{A}_0B_0$$

DECODERS

A decoder is a logic circuit that converts an 2V-bit binary input code into M output lines such that only one output line is activated for each one of the possible combinations of inputs.

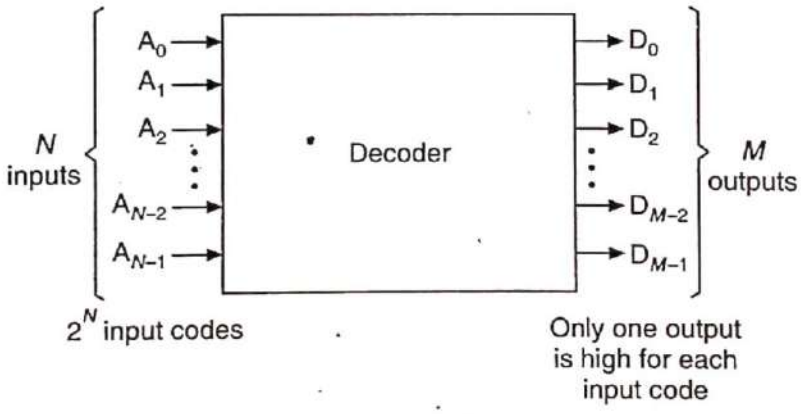


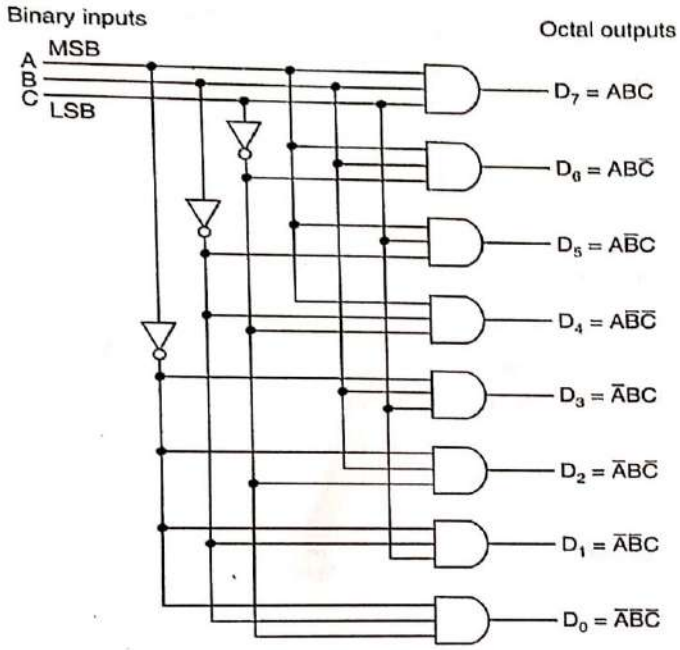
Figure 7.68 General block diagram of a decoder.

3-Line-to-8-Line Decoder

Inputs			Outputs							
A	B	C	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_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

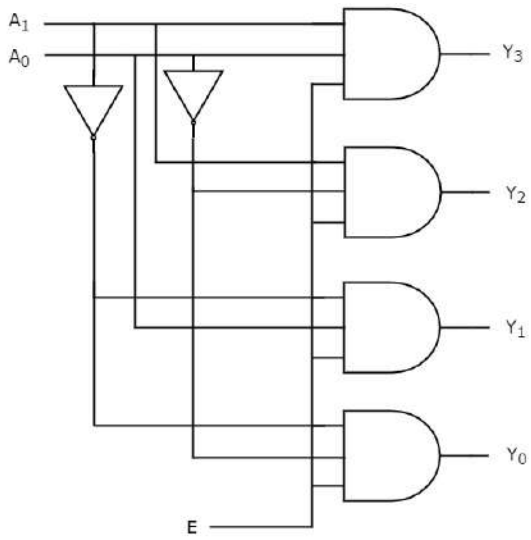
3-Line-to-8-Line Decoder

A \ BC	00	01	11	10
0	$\bar{A}\bar{B}\bar{C}$ (m_0) 0	$\bar{A}\bar{B}C$ (m_1) 1	$\bar{A}BC$ (m_3) 3	$\bar{A}B\bar{C}$ (m_2) 2
1	$A\bar{B}\bar{C}$ (m_4) 4	$A\bar{B}C$ (m_5) 5	ABC (m_7) 7	$AB\bar{C}$ (m_6) 6



2-Line-to-4-Line Decoder

2-Line-to-4-Line Decoder



Combinational Logic Implementation using Decoders

A decoder provides 2^n minterms of *n* input variables. Since any Boolean function can be expressed in sum of minterms, one can use a decoder to generate the minterms and an external OR gate to form the logic sum. In this way any combinational circuit with *n* inputs and *m* outputs can be implemented with an n-to-2^n decoder and *m* OR gates.

$$S = \bar{A}\bar{B}C_{in} + \bar{A}B\bar{C}_{in} + A\bar{B}\bar{C}_{in} + ABC_{in} = A \oplus B \oplus C_{in} = \Sigma\, m(1, 2, 4, 7)$$

$$C_{out} = \bar{A}BC_{in} + A\bar{B}C_{in} + AB\bar{C}_{in} + ABC_{in} = AB + (A \oplus B)C_{in} = \Sigma\, m(3, 5, 6, 7)$$

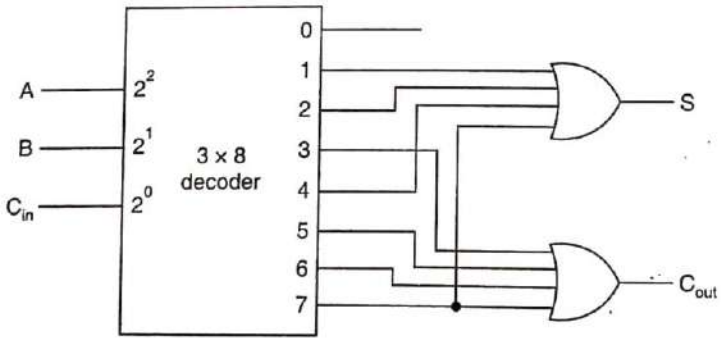


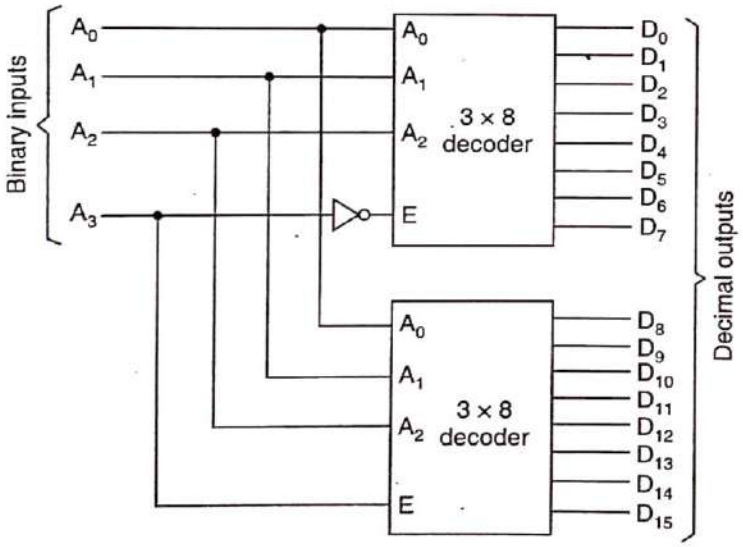
Figure 7.71 Logic diagram of a full adder using a decoder.

4-to-16 Decoder from Two 3-to-8 Decoders

	Binary inputs				Decimal output (active low)
	A_3	A_2	A_1	A_0	
Upper decoder	0	0	0	0	D_0
	0	0	0	1	D_1
	0	0	1	0	D_2
	0	0	1	1	D_3
	0	1	0	0	D_4
	0	1	0	1	D_5
	0	1	1	0	D_6
	0	1	1	1	D_7
Lower decoder	1	0	0	0	D_8
	1	0	0	1	D_9
	1	0	1	0	D_{10}
	1	0	1	1	D_{11}
	1	1	0	0	D_{12}
	1	1	0	1	D_{13}
	1	1	1	0	D_{14}
	1	1	1	1	D_{15}

(b) Function table

4-to-16 Decoder from Two 3-to-8 Decoders



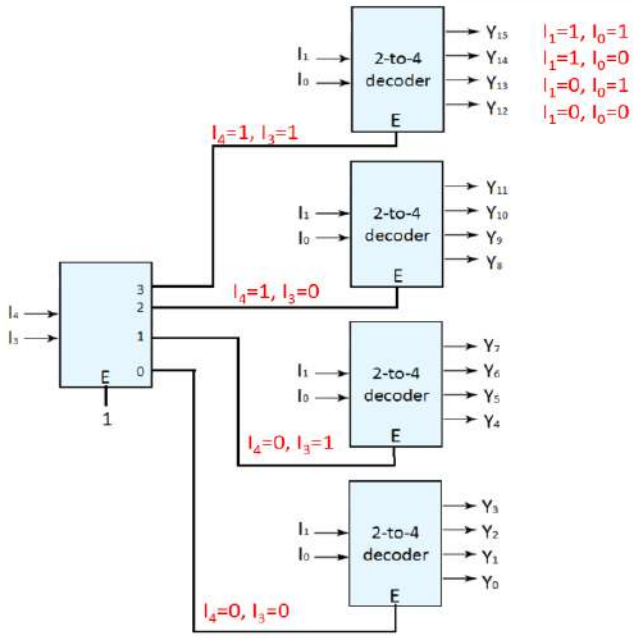
(a) Logic diagram

	Binary inputs A_3	A_2	A_1	A_0	Decimal output (active low)
Upper decoder	0	0	0	0	D_0
	0	0	0	1	D_1
	0	0	1	0	D_2
	0	0	1	1	D_3
	0	1	0	0	D_4
	0	1	0	1	D_5
	0	1	1	0	D_6
	0	1	1	1	D_7
Lower decoder	1	0	0	0	D_8
	1	0	0	1	D_9
	1	0	1	0	D_{10}
	1	0	1	1	D_{11}
	1	1	0	0	D_{12}
	1	1	0	1	D_{13}
	1	1	1	0	D_{14}
	1	1	1	1	D_{15}

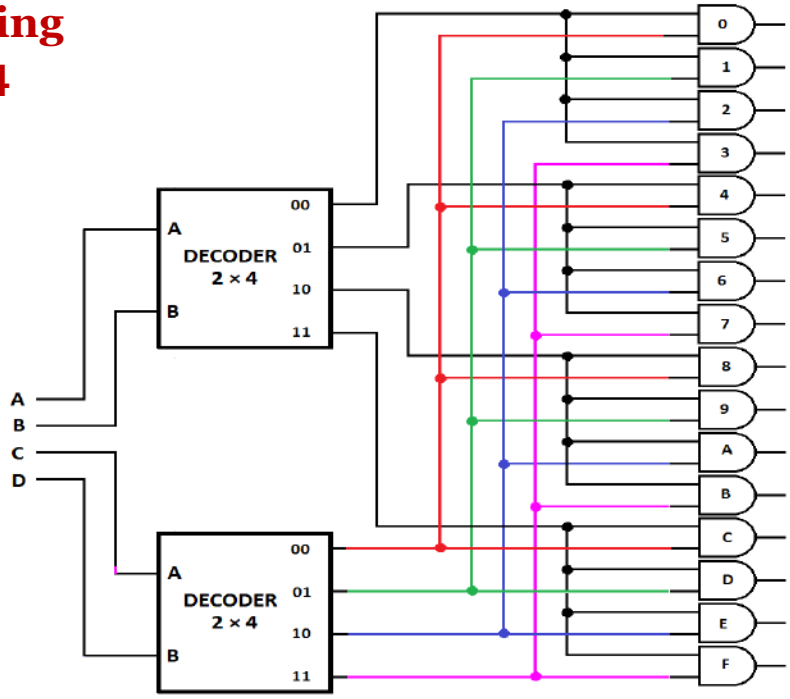
(b) Function table

Figure 7.72 Connecting two 74138 3-to-8 decoders to obtain a 4-to-16 decoder.

Build a 4-to-16 decoder using 2-to-4 decoders

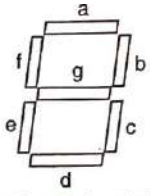


4x16 decoder using ONLY two 2x4 decoders

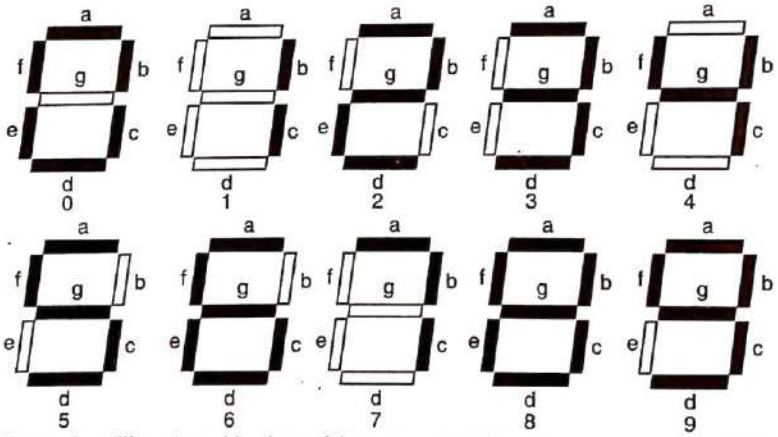


Decoder Applications

BCD-to-Seven Segment Decoders



(a) Letters used to designate the segments

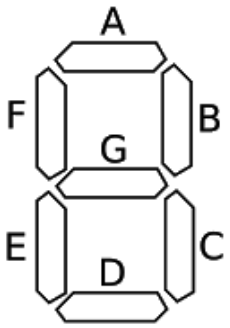


(b) By causing different combinations of the segments to illuminate (shown with solid black), the numerals 0–9 can be displayed.

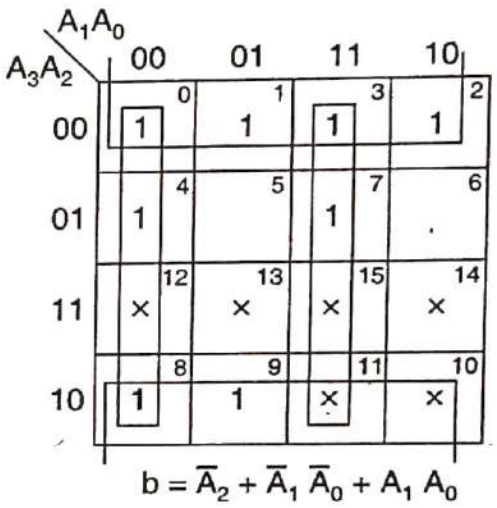
BCD-to-Seven Segment Decoders

Decimal digit	8-4-2-1 BCD				Seven segment code						
	A_3	A_2	A_1	A_0	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1

(b) Function table



Decimal digit	b
0	1
1	1
2	1
3	1
4	1
5	0
6	0
7	1
8	1
9	1



$$b = \Sigma\, m(0, 1, 2, 3, 4, 7, 8, 9) + \Sigma\, m(10, 11, 12, 13, 14, 15)$$

a = ∑m (0, 2, 3, 5, 7, 8, 9)

b = ∑m (0, 1, 2, 3, 4, 7, 8, 9)

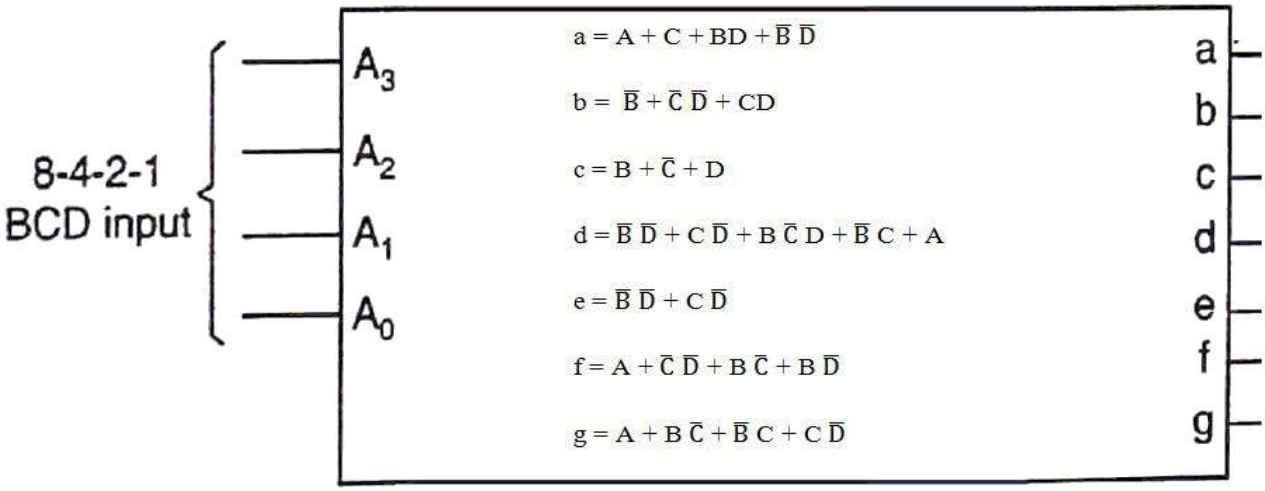
c = ∑m (0, 1, 3, 4, 5, 6, 7, 8, 9)

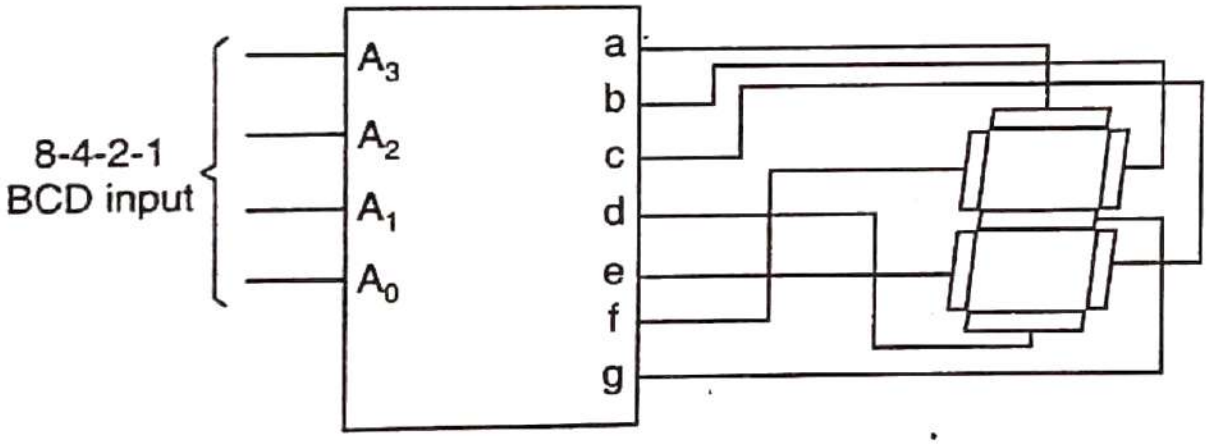
d = ∑m (0, 2, 3, 5, 6, 8)

e = ∑m (0, 2, 6, 8)

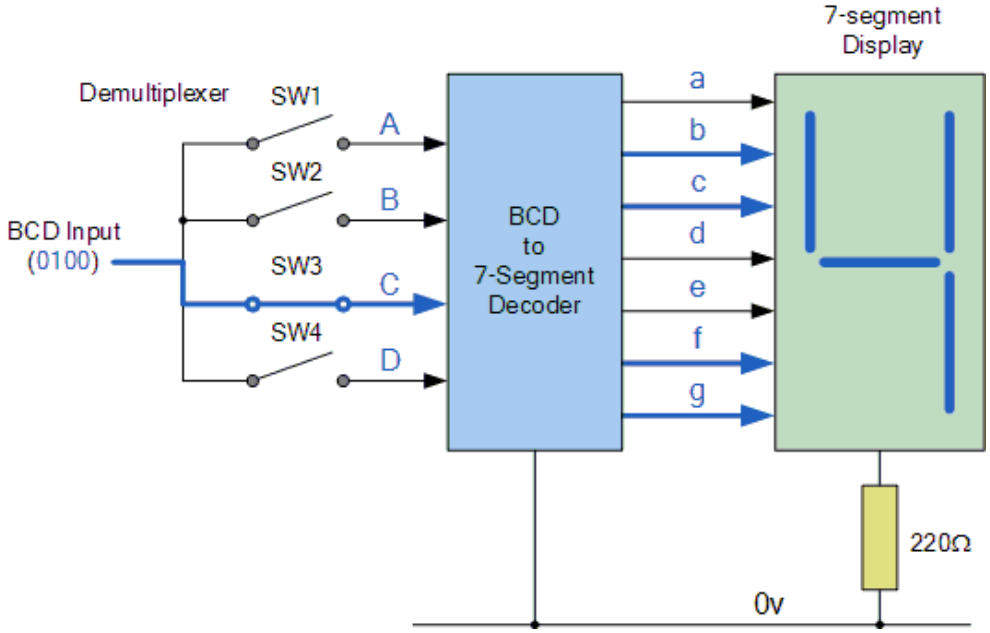
f = ∑m (0, 4, 5, 6, 8, 9)

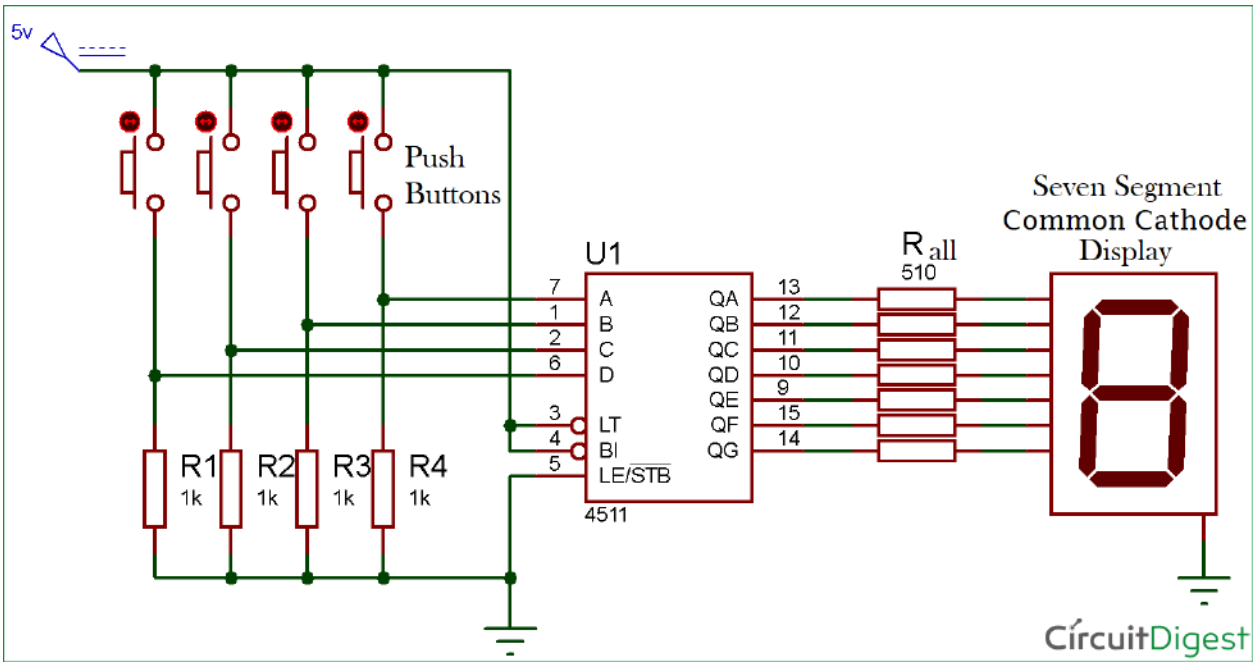
g = ∑m (2, 3, 4, 5, 6, 8, 9)

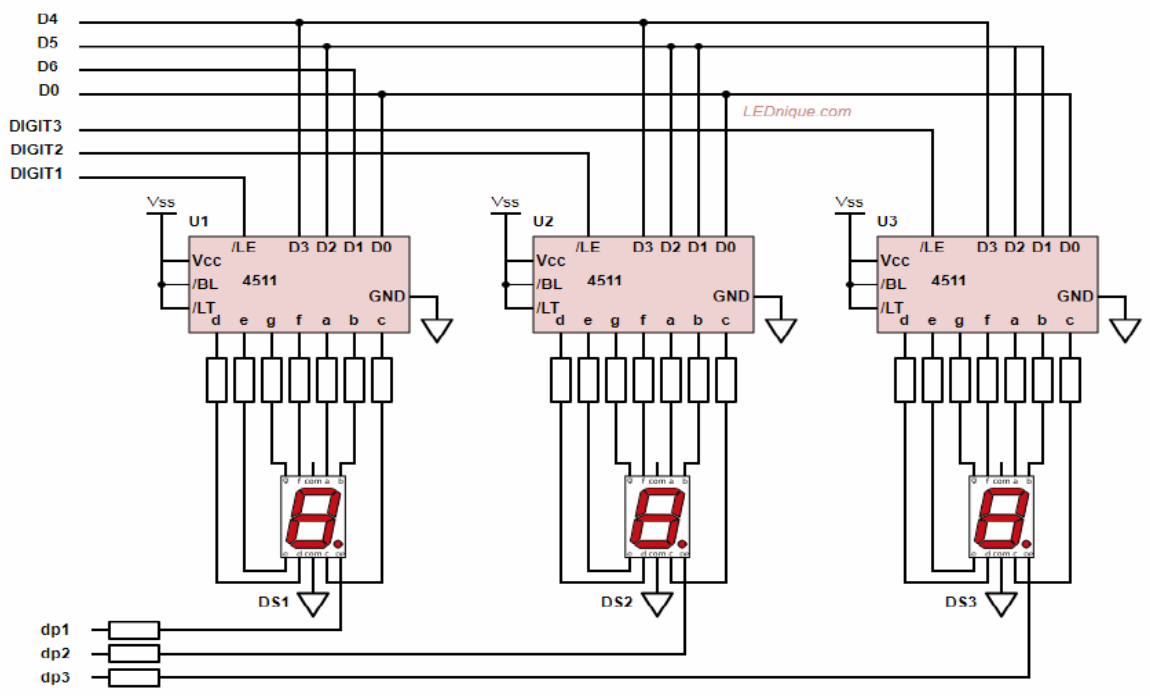




(a) Logic circuit


7-segment
Display
Demultiplexer
SW1
SW2
SW3
SW4
A
B
C
D
BCD Input
(0100)
BCD
to
7-Segment
Decoder
a
b
c
d
e
f
g
220Ω
0v


5v
Push
Buttons
R1
1k
R2
1k
R3
1k
R4
1k
U1
A
B
C
D
LT
BI
LE/STB
4511
QA
QB
QC
QD
QE
QF
QG
7
1
2
6
3
4
5
13
12
11
10
9
15
14
R all
510
Seven Segment
Common Cathode
Display
CircuitDigest



ENCODERS

An encoder is a device whose inputs are decimal digits and/or alphabetic characters and whose outputs are the coded representation of those inputs, i.e. an encoder is a device which converts familiar numbers or symbols into coded format. In other words, an encoder may be said to be a combinational logic circuit that performs the 'reverse' operation of the decoder.

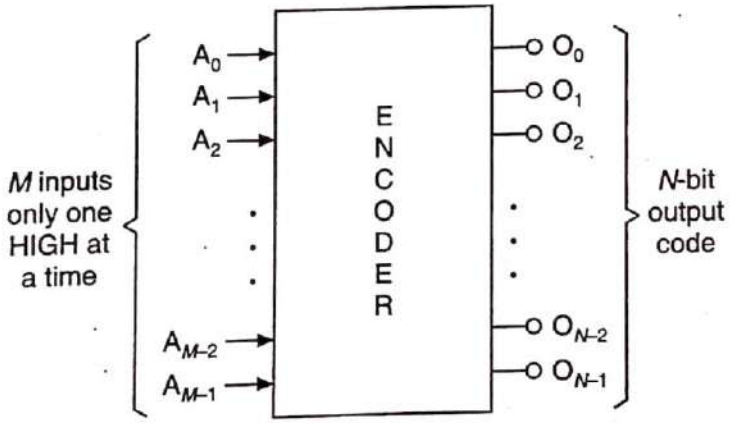


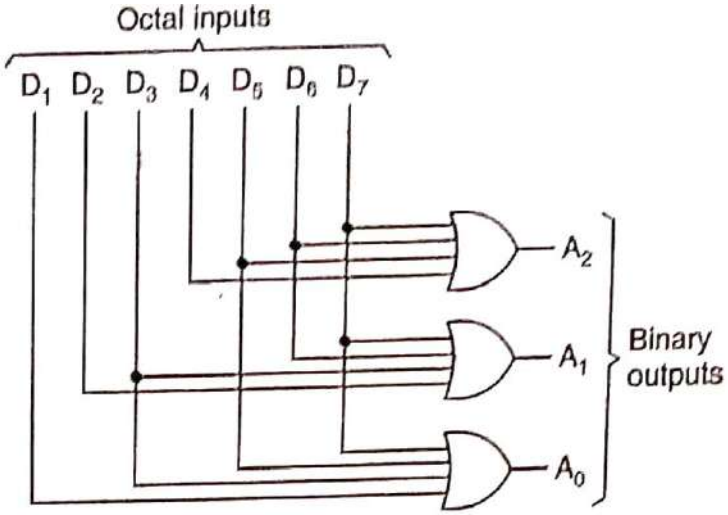
Figure 7.61 Block diagram of encoder.

Octal-to-Binary Encoder

Octal-to-Binary Encoder

Octal digits		Binary		
		A_2	A_1	A_0
D_0	0	0	0	0
D_1	1	0	0	1
D_2	2	0	1	0
D_3	3	0	1	1
D_4	4	1	0	0
D_5	5	1	0	1
D_6	6	1	1	0
D_7	7	1	1	1

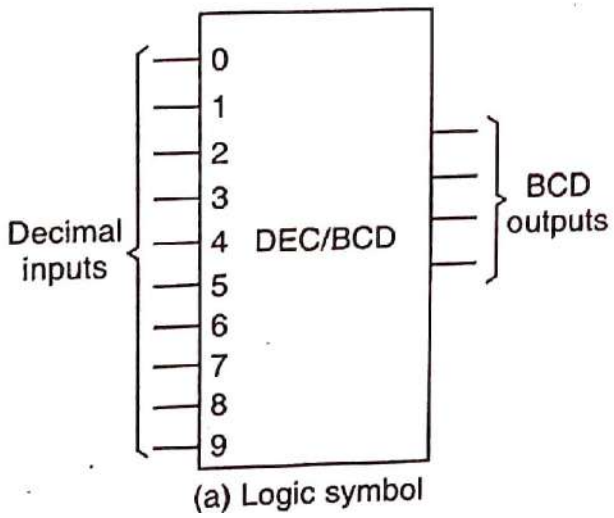
(a) Truth table



(b) Logic diagram

Figure 7.62 Octal-to-binary encoder.

Decimal-to-BCD Encoder

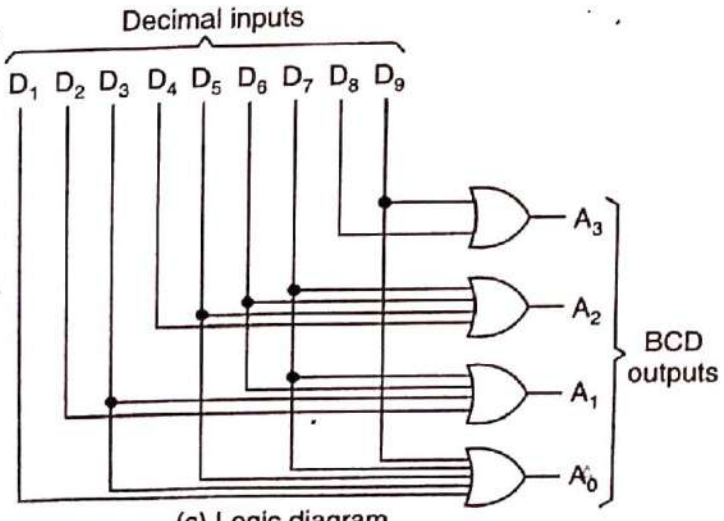


(a) Logic symbol

Decimal-to-BCD Encoder

Decimal inputs		Binary			
		A_3	A_2	A_1	A_0
D_0	0	0	0	0	0
D_1	1	0	0	0	1
D_2	2	0	0	1	0
D_3	3	0	0	1	1
D_4	4	0	1	0	0
D_5	5	0	1	0	1
D_6	6	0	1	1	0
D_7	7	0	1	1	1
D_8	8	1	0	0	0
D_9	9	1	0	0	1

(b) Truth table

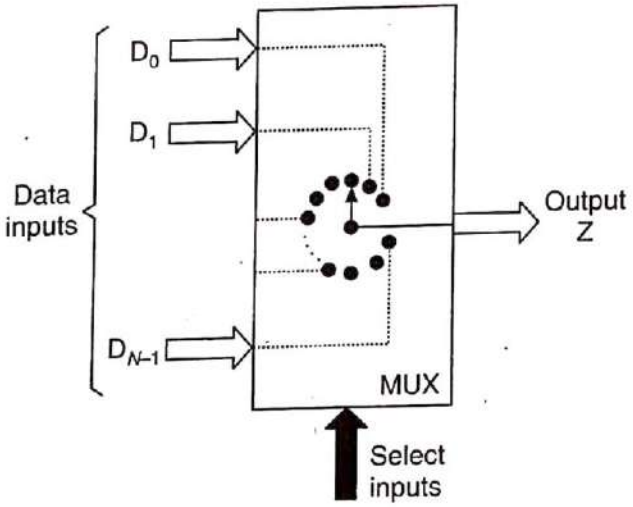


(c) Logic diagram

Figure 7.63 Decimal-to-BCD encoder.

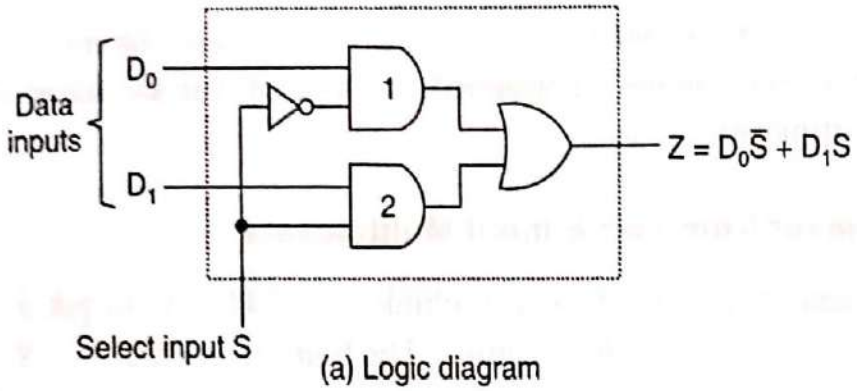
MULTIPLEXERS (DATA SELECTORS)

A multiplexer (MUX) or data selector is a logic circuit that accepts several data inputs and allows only one of them at a time to get through to the output. The routing of the desired data input to the output is controlled by SELECT inputs.



Basic 2-lnput Multiplexer (2x1 MUX)

When S = 0, AND gate 1 is enabled and AND gate 2 is disabled. So, $Z = D_0$.
When S = 1, AND gate 1 is disabled and AND gate 2 is enabled. So, $Z = D_1$.

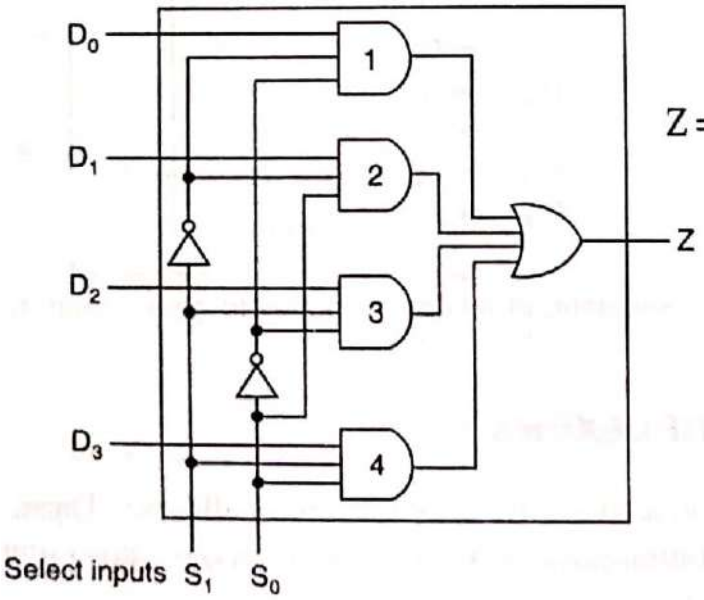


(a) Logic diagram

S	Output (Z)
0	D_0
1	D_1

(b) Function table

The 4-lnput Multiplexer (4x1 MUX)



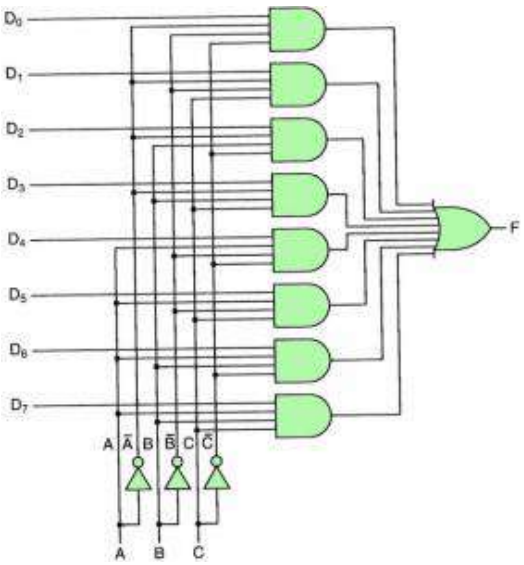
(a) Logic diagram

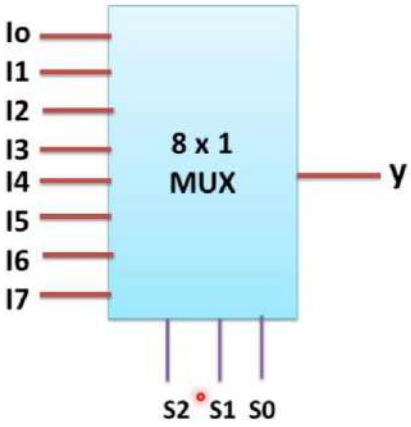
$$Z = \bar{S}_1 \bar{S}_0 D_0 + \bar{S}_1 S_0 D_1 + S_1 \bar{S}_0 D_2 + S_1 S_0 D_3$$

Select inputs		Output
S_1	S_0	Z
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

(b) Function table

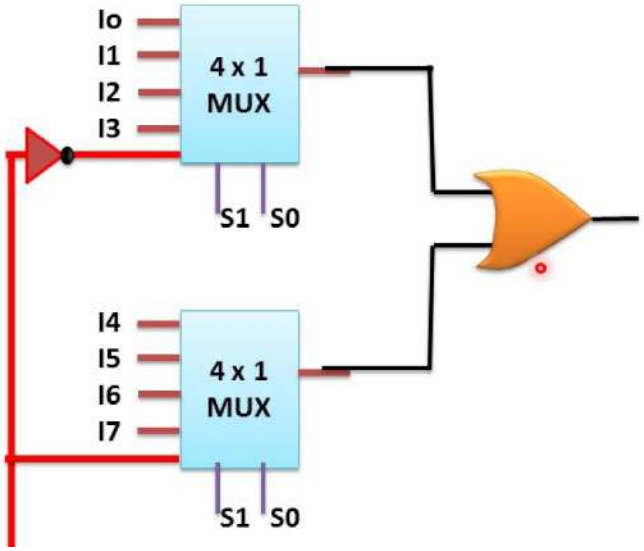
The 8-lnput Multiplexer (8x1 MUX)





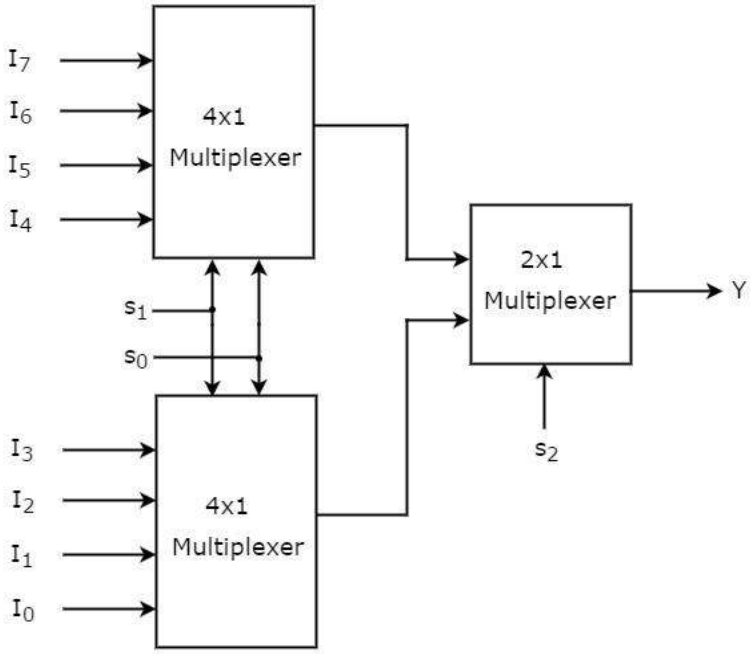
S2	S1	S0	Y
0	0	0	Io
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

8x1 MUX using two 4x1 MUX



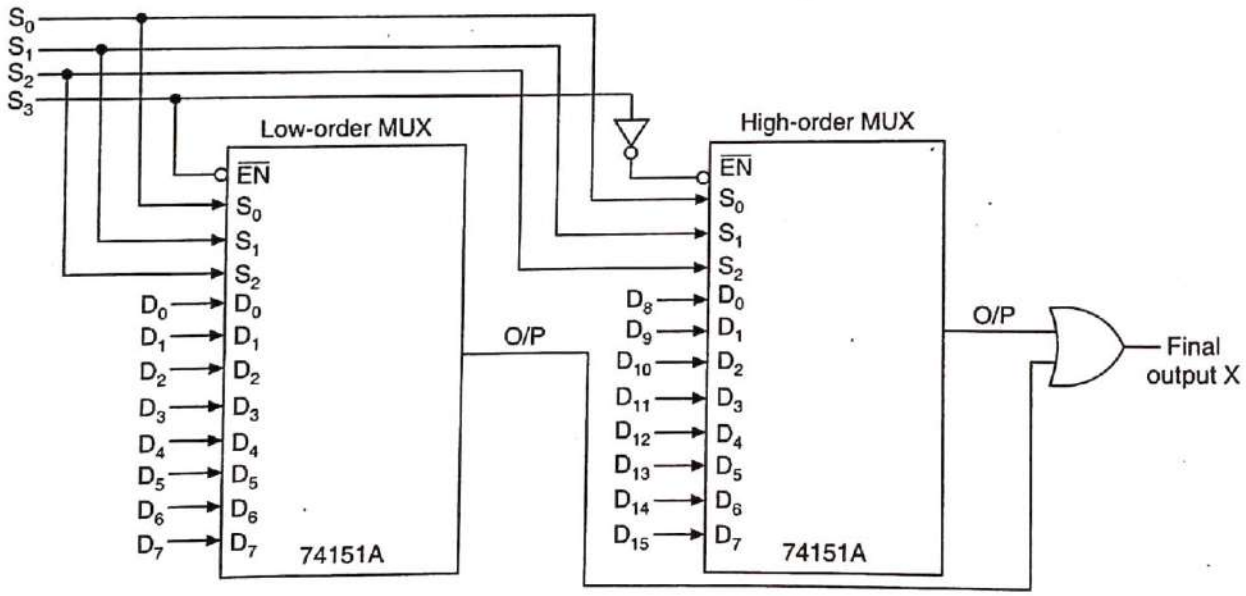
S2	S1	S0	Y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7

8x1 MUX using two 4x1 MUX and one 2x1 MUX



The 16-lnput Multiplexer from Two 8-lnput Multiplexers

The 16-lnput Multiplexer from Two 8-lnput Multiplexers



APPLICATIONS OF MULTIPLEXERS

Logic Function Generator

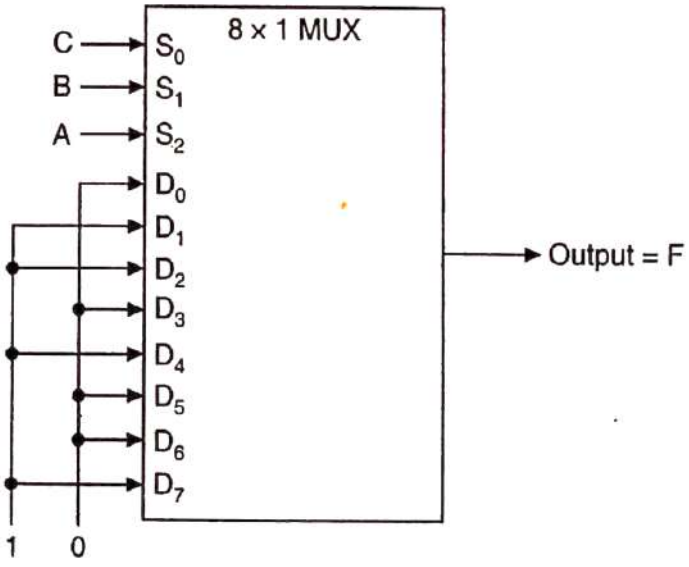
EXAMPLE 7.11 Use a multiplexer to implement the logic function $F = A \oplus B \oplus C$.

S_2	S_1	S_0	$F = A \oplus B \oplus C$
A	B	C	
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Logic Function Generator

EXAMPLE 7.11 Use a multiplexer to implement the logic function $F = A \oplus B \oplus C$.

S_2	S_1	S_0	$F = A \oplus B \oplus C$
A	B	C	
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1



EXAMPLE 7.12 Implement the following function using 8-to-1 MUX

$$F(x, y, z) = \Sigma\, m(0, 2, 3, 5)$$

S_2	S_1	S_0	
x	y	z	F
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

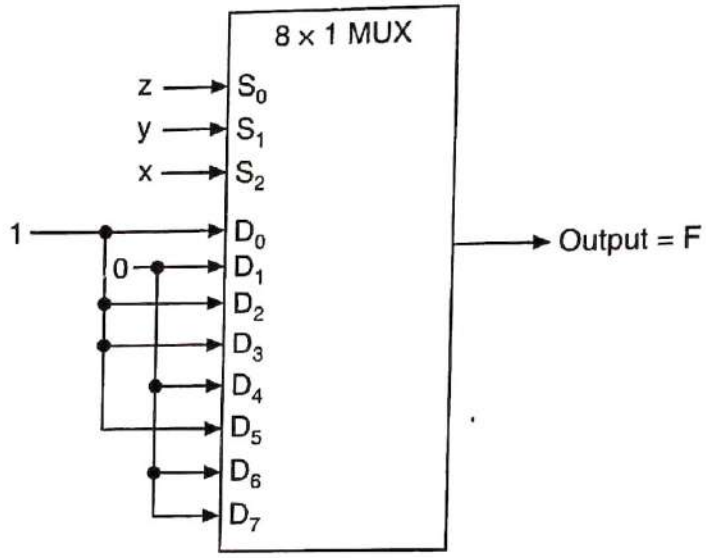
(a) Truth table

EXAMPLE 7.12 Implement the following function using 8-to-1 MUX

$$F(x, y, z) = \Sigma\, m(0, 2, 3, 5)$$

S_2	S_1	S_0	
x	y	z	F
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

(a) Truth table

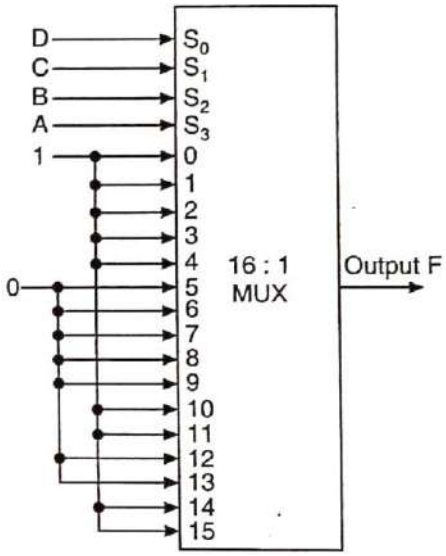


(b) Logic diagram

EXAMPLE 7.13 Use a multiplexer having three data select inputs to implement the logic for the function given below. Also realize the same using a 16:1 MUX.

$$F = \Sigma\, m(0, 1, 2, 3, 4, 10, 11, 14, 15).$$

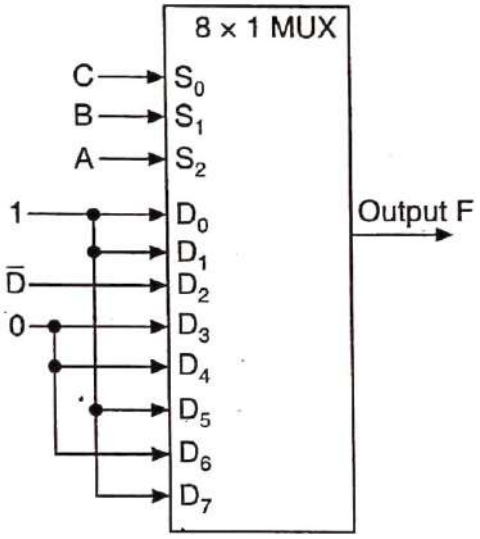
S_2 A	S_1 B	S_0 C	D	F	
0	0	0	0	1	F = 1
0	0	0	1	1	
0	0	1	0	1	F = 1
0	0	1	1	1	
0	1	0	0	1	$F = \overline{D}$
0	1	0	1	0	
0	1	1	0	0	F = 0
0	1	1	1	0	
1	0	0	0	0	F = 0
1	0	0	1	0	
1	0	1	0	1	F = 1
1	0	1	1	1	
1	1	0	0	0	F = 0
1	1	0	1	0	
1	1	1	0	1	F = 1
1	1	1	1	1	



EXAMPLE 7.13 Use a multiplexer having three data select inputs to implement the logic for the function given below. Also realize the same using a 16:1 MUX.

$$F = \Sigma\, m(0, 1, 2, 3, 4, 10, 11, 14, 15).$$

S_2 A	S_1 B	S_0 C	D	F	
0	0	0	0	1	F = 1
0	0	0	1	1	
0	0	1	0	1	F = 1
0	0	1	1	1	
0	1	0	0	1	$F = \overline{D}$
0	1	0	1	0	
0	1	1	0	0	F = 0
0	1	1	1	0	
1	0	0	0	0	F = 0
1	0	0	1	0	
1	0	1	0	1	F = 1
1	0	1	1	1	
1	1	0	0	0	F = 0
1	1	0	1	0	
1	1	1	0	1	F = 1
1	1	1	1	1	



- In general, a multiplexer with n-data select inputs can implement any function of n + 1 variables.

- The key to this design is to use the first n variables of the function as the select inputs and to use the least significant input variable and its complement to drive some of the data inputs.

EXAMPLE 7.14 Use a 4 × 1 MUX to implement the logic function

$$F(A, B; C) = \Sigma\, m(1, 2, 4, 7).$$

S_1 A	S_0 B	C	F	
0	0	0	0	F = C
0	0	1	1	
0	1	0	1	$F = \overline{C}$
0	1	1	0	
1	0	0	0	F = 0
1	0	1	0	
1	1	0	1	F = 1
1	1	1	1	

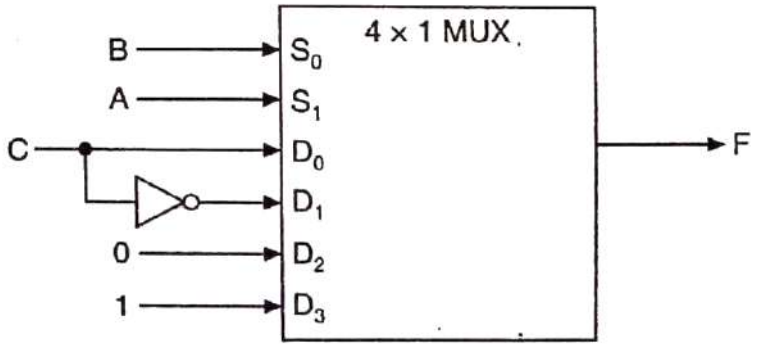
(a) Truth table

EXAMPLE 7.14 Use a 4 × 1 MUX to implement the logic function

$$F(A, B, C) = \Sigma\, m(1, 2, 4, 7).$$

S_1 A	S_0 B	C	F	
0	0	0	0	F = C
0	0	1	1	
0	1	0	1	F = $\overline{C}$
0	1	1	0	
1	0	0	0	F = 0
1	0	1	0	
1	1	0	1	F = 1
1	1	1	1	

(a) Truth table



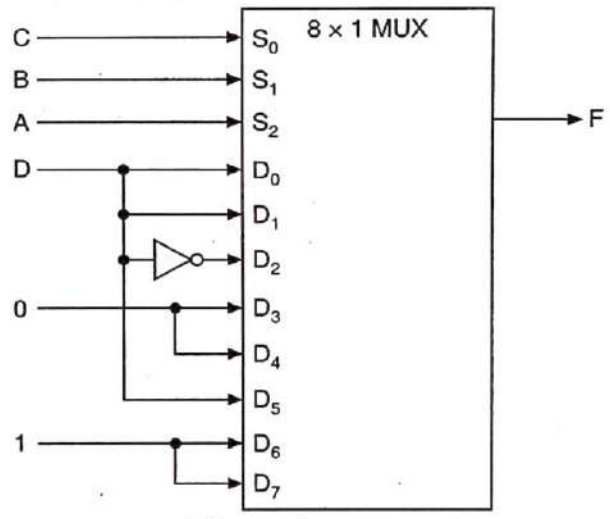
(b) Multiplexer implementation

EXAMPLE 7.17 Implement the following logic function using an 8 × 1 MUX:

$$F(A, B, C, D) = \Sigma\, m(1, 3, 4, 11, 12, 13, 14, 15)$$

S_2 A	S_1 B	S_0 C	D	F	
0	0	0	0	0	F = D
0	0	0	1	1	
0	0	1	0	0	F = D
0	0	1	1	1	
0	1	0	0	1	F = $\overline{D}$
0	1	0	1	0	
0	1	1	0	0	F = 0
0	1	1	1	0	
1	0	0	0	0	F = 0
1	0	0	1	0	
1	0	1	0	0	F = D
1	0	1	1	1	
1	1	0	0	1	F = 1
1	1	0	1	1	
1	1	1	0	1	F = 1
1	1	1	1	1	

(a) Truth table



(b) Logic diagram

EXAMPLE 7.19 Implement the following Boolean function using an 8:1 multiplexer considering D as the input and A, B, C as the selection lines:

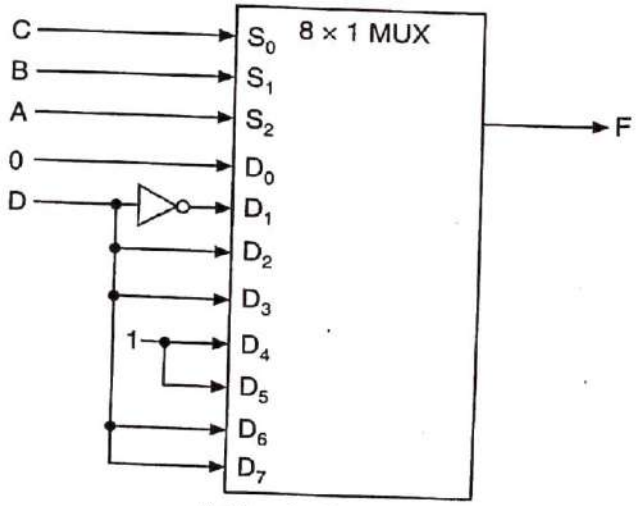
$$F(A, B, C, D) = A\overline{B} + BD + \overline{B}C\overline{D}$$

$$F(A, B, C, D) = A\overline{B} + BD + \overline{B}C\overline{D} = 10XX + X1X1 + X010$$
$$= 1000 + 1001 + 1010 + 1011 + 0101 + 0111 + 1101 + 1111 + 0010 + 1010$$
$$= \Sigma\, m(8, 9, 10, 11, 5, 7, 13, 15, 2, 10) = \Sigma\, m\,(2, 5, 7, 8, 9, 10, 11, 13, 15)$$

$$F(A, B, C, D) = \Sigma\, m\,(2, 5, 7, 8, 9, 10, 11, 13, 15)$$

S_2	S_1	S_0			
A	B	C	D	F	
0	0	0	0	0	F = 0
0	0	0	1	0	
0	0	1	0	1	F = $\overline{D}$
0	0	1	1	0	
0	1	0	0	0	F = D
0	1	0	1	1	
0	1	1	0	0	F = D
0	1	1	1	1	
1	0	0	0	1	F = 1
1	0	0	1	1	
1	0	1	0	1	F = 1
1	0	1	1	1	
1	1	0	0	0	F = D
1	1	0	1	1	
1	1	1	0	0	F = D
1	1	1	1	1	

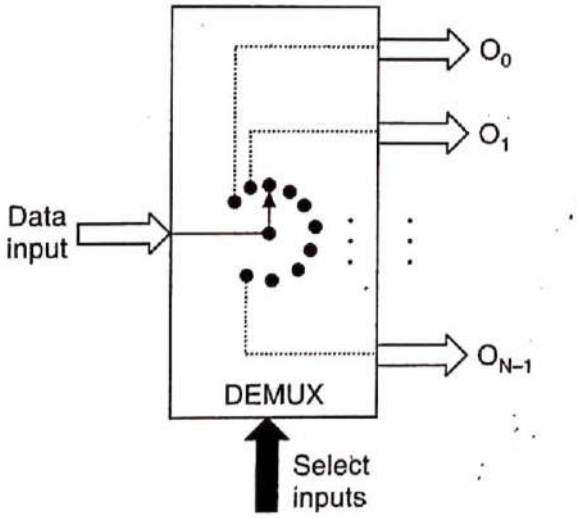
(a) Truth table



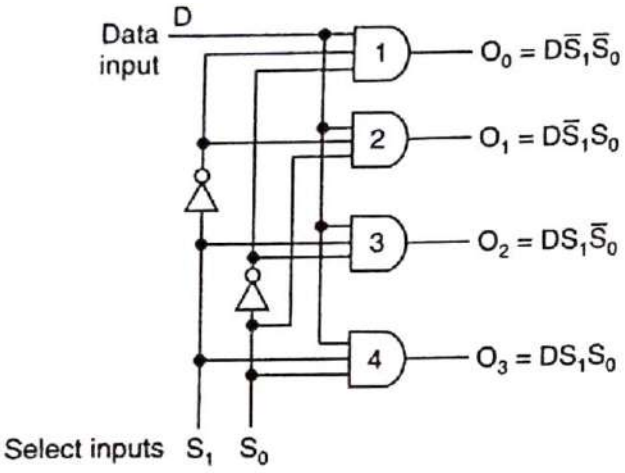
(b) Logic diagram

DEMULTIPLEXERS (DATA DISTRIBUTORS)

A demultiplexer performs the reverse operation; it takes a single input and distributes it over several outputs. So a demultiplexer can be thought of as a 'distributor', since it transmits the same data to different destinations. Thus, whereas a multiplexer is an N-to-l device, a demultiplexer is a 1-to-N device.



1-Line to 4-Line Demultiplexer

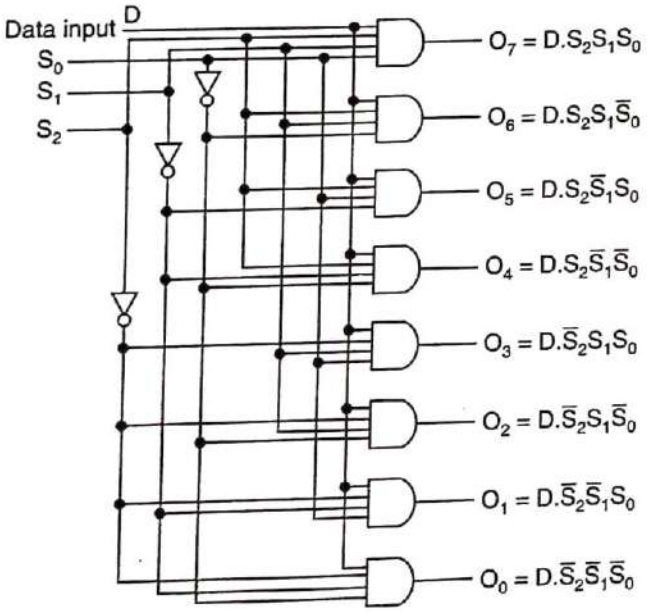


(a) Logic diagram

Select code		Outputs			
S_1	S_0	O_3	O_2	O_1	O_0
0	0	0	0	0	D
0	1	0	0	D	0
1	0	0	D	0	0
1	1	D	0	0	0

(b) Truth table

1-Line to 8-Line Demultiplexer



(a) Logic diagram

Select code			Outputs							
S_2	S_1	S_0	O_7	O_6	O_5	O_4	O_3	O_2	O_1	O_0
0	0	0	0	0	0	0	0	0	0	D
0	0	1	0	0	0	0	0	0	D	0
0	1	0	0	0	0	0	0	D	0	0
0	1	1	0	0	0	0	D	0	0	0
1	0	0	0	0	0	D	0	0	0	0
1	0	1	0	0	D	0	0	0	0	0
1	1	0	0	D	0	0	0	0	0	0
1	1	1	D	0	0	0	0	0	0	0

(b) Truth table

Flip-Flops

Basically, switching circuits may be combinational switching circuits or sequential switching circuits.

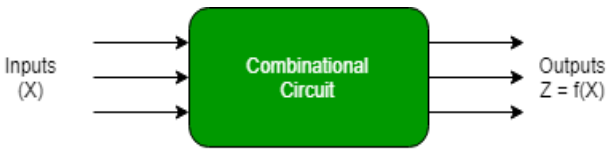


Figure: Combinational Circuits

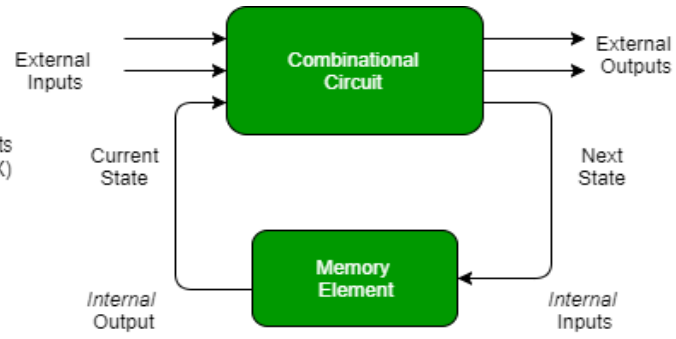


Figure: Sequential Circuit

Combinational circuits	Sequential circuits
In combinational circuits, the output variables at any instant of time are dependent only on the present input variables.	In sequential circuits, the output variables at any instant of time are dependent not only on the present input variables, but also on the present state, i.e. on the past history of the system.
Memory unit is not required in combinational circuits.	Memory unit is required to store the past history of the input variables in sequential circuits.
Combinational circuits are faster because the delay between the input and the output is due to propagation delay of gates only	Sequential circuits are slower than combinational circuits.
Combinational circuits are easy to design.	Sequential circuits are comparatively harder to design.

Key	**Synchronous Sequential Circuits**	**Asynchronous Sequential Circuits**
Definition	Synchronous sequential circuits are digital sequential circuits in which the feedback to the input for next output generation is governed by clock signals.	On other hand Asynchronous sequential circuits are digital sequential circuits in which the feedback to the input for next output generation is not governed by clock signals.
Memory Unit	In Synchronous sequential circuits, the memory unit which is being get used for governance is clocked flip flop.	On other hand unclocked flip flop or time delay is used as memory element in case of Asynchronous sequential circuits.
State	The states of Synchronous sequential circuits are always predictable and thus reliable.	On other hand there are chances for the Asynchronous circuits to enter into a wrong state because of the time difference between the arrivals of inputs. This is called as race condition.

Key	Synchronous Sequential Circuits	Asynchronous Sequential Circuits
Complexity	It is easy to design Synchronous sequential circuits	However on other hand the presence of feedback among logic gates causes instability issues making the design of Asynchronous sequential circuits difficult.
Performance	Due to the propagation delay of clock signal in reaching all elements of the circuit the Synchronous sequential circuits are slower in its operation speed	Since there is no clock signal delay, these are fast compared to the Synchronous Sequential Circuits
Example	Synchronous circuits are used in counters, shift registers, memory units.	On other hand Asynchronous circuits are used in low power and high speed operations such as simple microprocessors, digital signal processing units and in communication systems for email applications, internet access and networking.

Asynchronous Sequential Circuit

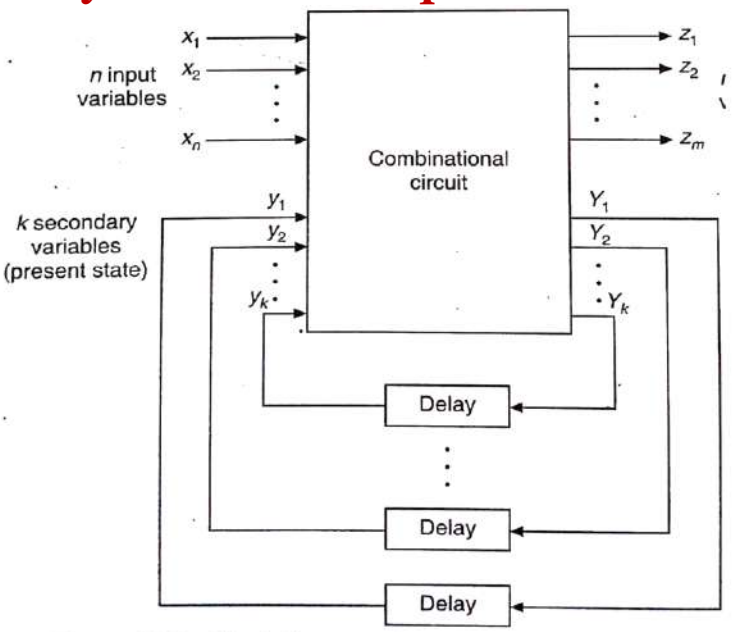


Figure 10.2 Block diagram of an asynchronous sequential

LATCHES AND FLIP-FLOPS

- The most important memory element is the flip-flop, which is made up of an assembly of logic gates.
- Even though a logic gate by itself has no storage capability, several logic gates can be connected together in ways that permit information to be stored).
- There are several different gate arrangements that are used to construct flip-flops in a wide variety of ways.
- Flip-flops are the basic building blocks of most sequential circuits.
- A flip-flop (FF), known more formally as a bistable multivibrator, has two stable states. It can remain in either of the states indefinitely. Its state can be changed by applying the proper triggering signal.
- It is also called a binary or one-bit memory.

Flip-Flops

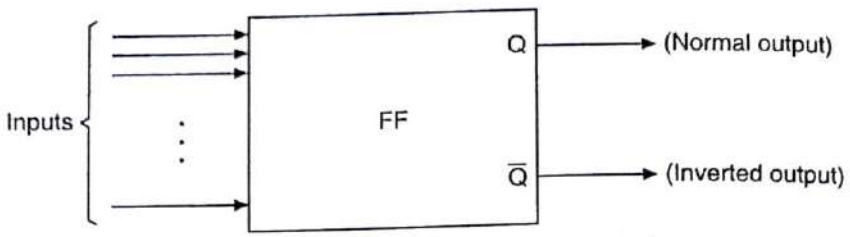
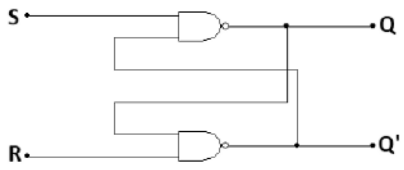
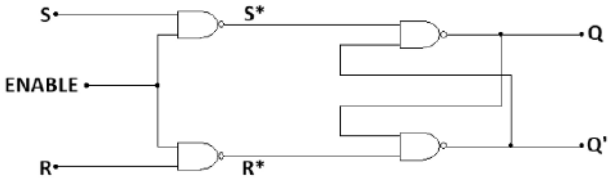


Figure 10.3 General flip-flop symbol.

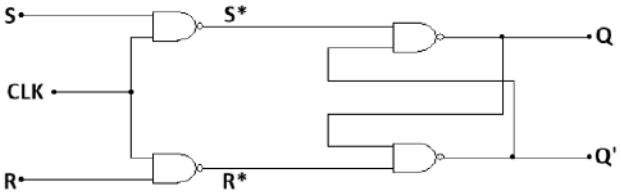
- The input signals which command the flip-flop to change state are called excitations.
- These inputs are used to cause the flip-flop to switch back and forth (i.e. 'flip-flop') between its possible output states.
- A flip-flop input has to be pulsed momentarily to cause a change in the flip-flop output, and the output will remain in that new state even after the input pulse has been removed. This is the flip-flop's memory characteristic.



Latches



Gated Latch



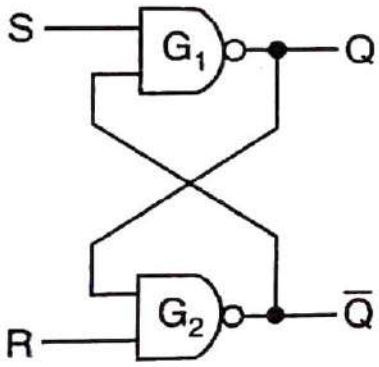
Flip Flop

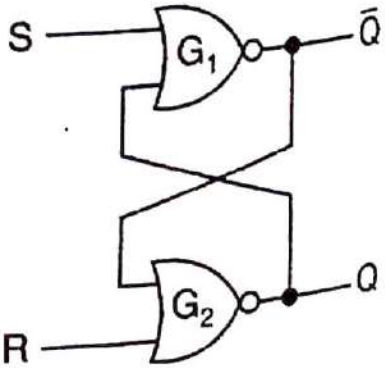
Flip-flops	Latches
Bistable device	Bistable device
Clock	No clock
.....	.....
.....	.....

The S-R Latch

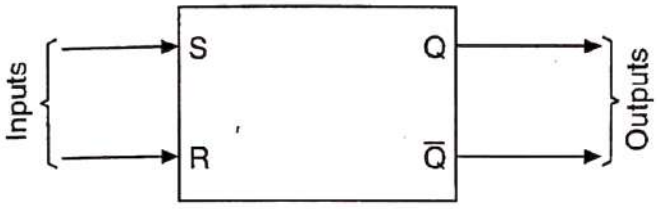
- The simplest type of flip-flop is called an S-R latch. It has two outputs labelled Q and Q and two inputs labelled S and R.

- It can be constructed using either two cross-coupled NAND gates or two-cross coupled NOR gates.

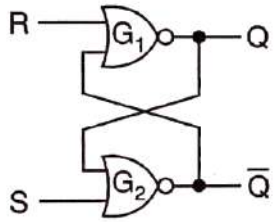




The NOR gate S-R latch (active-high S-R latch)



(a) Logic symbol



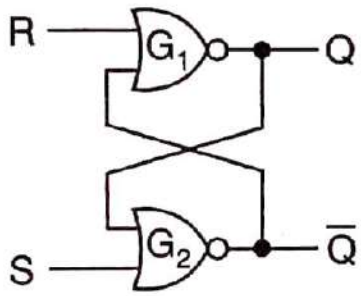
(b) Logic diagram

S	R	Q_n	Q_{n+1}	State
0	0	0	0	No Change (NC)
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0	1	Set
1	0	1	1	
1	1	0	×	Indeterminate (invalid)
1	1	1	×	

(c) Truth table

Figure 10.4 Active-HIGH S-R latch.

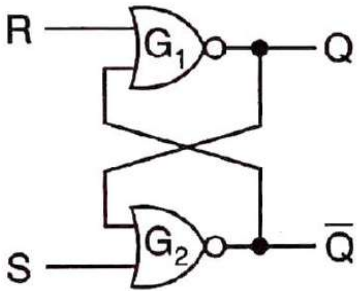
The NOR gate S-R latch (active-high S-R latch)



S	R	Q_n	Q_{n+1}	State
0	0	0	0	No Change (NC)
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0	1	Set
1	0	1	1	
1	1	0	×	Indeterminate (invalid)
1	1	1	×	

A	B	Output
0	0	1
0	1	0
1	0	0
1	1	0

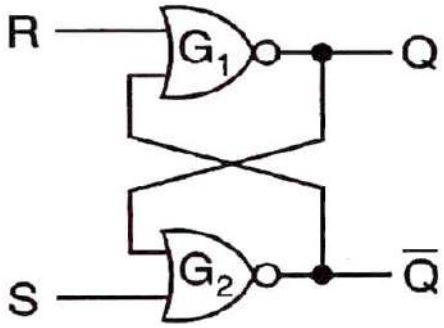
The S-R Latch using NOR Gates



1. **SET = 0, RESET = 0:** This is the normal resting state of the NOR latch and it has no effect on the output state. Q and $\overline{Q}$ will remain in whatever state they were prior to the occurrence of this input condition.

S	R	Q_n	Q_{n+1}	State
0	0	0	0	No Change (NC)
0	0	1	1	

The S-R Latch using NOR Gates

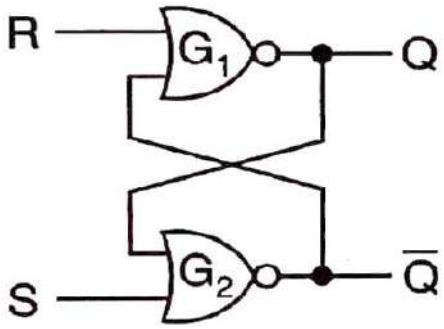


SET = 0, RESET = 0: This is the normal resting state of the NOR latch and it has no effect on the output state. Q and $\overline{Q}$ will remain in whatever state they were prior to the occurrence of this input condition.

SET = 1, RESET = 0: This will always set Q = 1, where it will remain even after SET returns to 0.

S	R	Q_n	Q_{n+1}	State
0	0	0	0	No Change (NC)
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	

The S-R Latch using NOR Gates



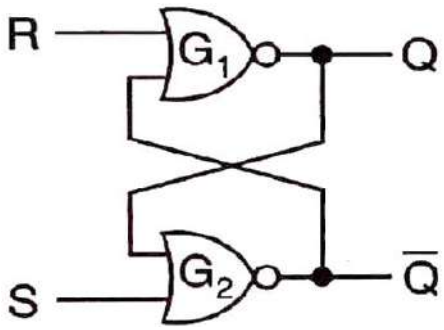
S	R	Q_n	Q_{n+1}	State
0	0	0	0	No Change (NC)
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0	1	Set
1	0	1	1	

SET = 0, RESET = 0: This is the normal resting state of the NOR latch and it has no effect on the output state. Q and $\overline{Q}$ will remain in whatever state they were prior to the occurrence of this input condition.

SET = 1, RESET = 0: This will always set Q = 1, where it will remain even after SET returns to 0.

SET = 0, RESET = 1: This will always reset Q = 0, where it will remain even after RESET returns to 0.

The S-R Latch using NOR Gates



S	R	Q_n	Q_{n+1}	State
0	0	0	0	No Change (NC)
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0	1	Set
1	0	1	1	
1	1	0	×	Indeterminate (invalid)
1	1	1	×	

SET = 0, RESET = 0: This is the normal resting state of the NOR latch and it has no effect on the output state. Q and $\overline{Q}$ will remain in whatever state they were prior to the occurrence of this input condition.

SET = 1, RESET = 0: This will always set Q = 1, where it will remain even after SET returns to 0.

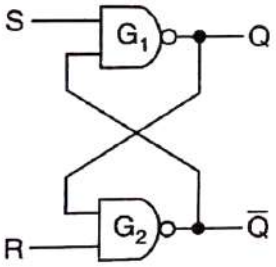
SET = 0, RESET = 1: This will always reset Q = 0, where it will remain even after RESET returns to 0.

SET = 1, RESET = 1: This condition tries to SET and RESET the latch at the same time, and it produces $Q = \overline{Q} = 0$. If the inputs are returned to zero simultaneously, the resulting output state is erratic and unpredictable. This input condition should not be used. It is forbidden.

The NAND gate S-R latch (active-low S-R latch)

The operation of this latch is the reverse of the operation of the NOR gate latch discussed earlier. That is why it is called an active-LOW S-R latch. If the 0's are replaced by 1's and 1's by 0's

Input	Input	Output
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

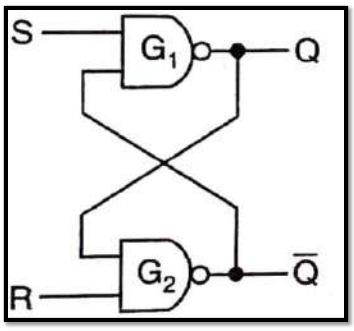


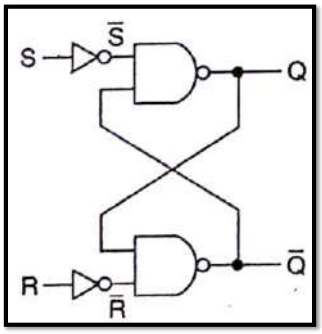
(a) Using NAND gates

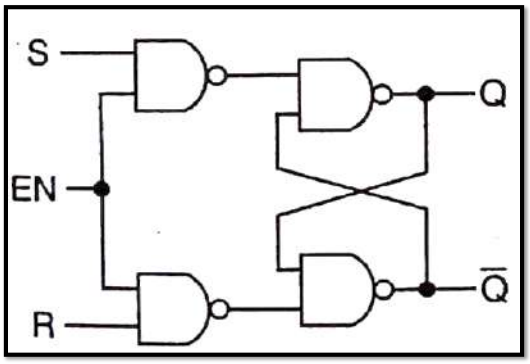
S	R	Q_n	Q_{n+1}	State
0	0	0	×	Indeterminate (invalid)
0	0	1	×	
0	1	0	1	Set
0	1	1	1	
1	0	0	0	Reset
1	0	1	0	
1	1	0	0	No Change (NC)
1	1	1	1	

(b) Truth table

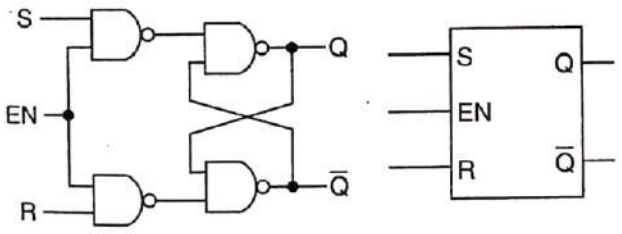
Figure 10.5 An active-LOW S-R latch.







The gated S-R latch



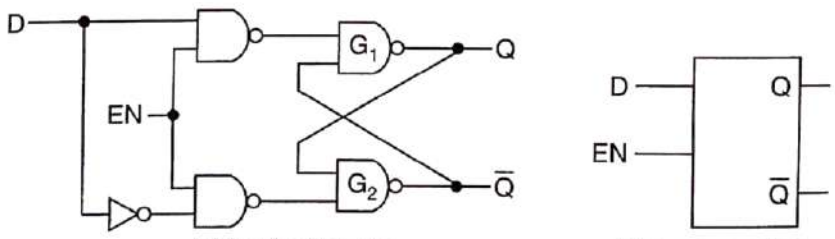
(a) Logic diagram (b) Logic symbol

EN	S	R	Q_n	Q_{n+1}	State
1	0	0	0	0	No change (NC)
1	0	0	1	1	
1	0	1	0	0	Reset
1	0	1	1	0	
1	1	0	0	1	Set
1	1	0	1	1	
1	1	1	0	×	Indeterminate (invalid)
1	1	1	1	×	
0	×	×	0	0	No Change (NC)
0	×	×	1	1	

(c) Truth table

Figure 10.7 A gated S-R latch.

The gated D-latch



(a) Logic diagram (b) Logic symbol

EN	D	Q_n	Q_{n+1}	State
1	0	0	0	Reset
1	0	1	0	
1	1	0	1	Set
1	1	1	1	
0	×	0	0	No Change (NC)
0	×	1	1	

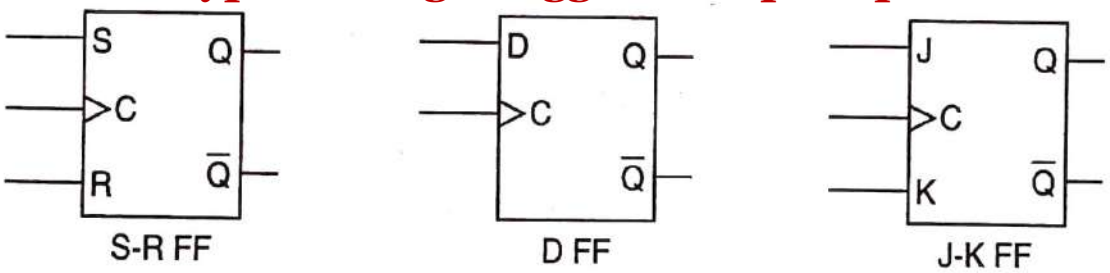
(c) Truth table

Figure 10.9 A gated D-latch.

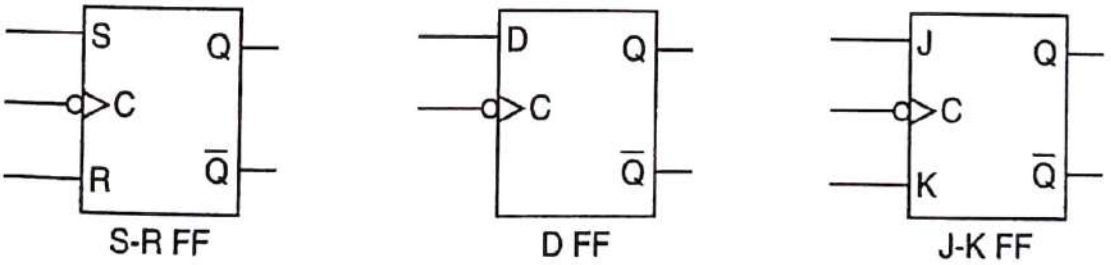
Flip-Flops

1. The edge-triggered **S-R flip-flop**
2. The edge-triggered **D flip-flop**
3. The edge-triggered **J-K flip-flop**
4. The edge-triggered **T flip-flop**

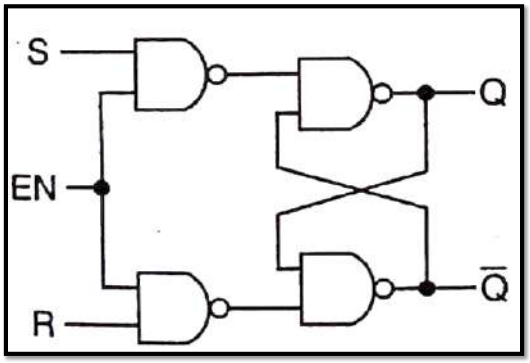
Types of Edge-triggered Flip-Flops



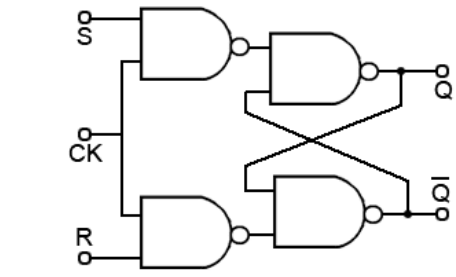
(a) Logic symbols of positive edge-triggered FFs

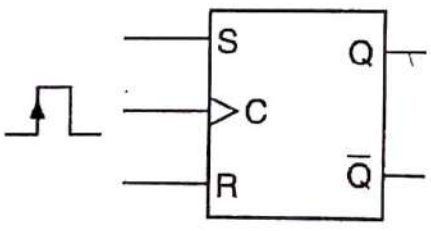


(b) Logic symbols of negative edge-triggered FFs



The edge-triggered S-R flip-flop





(a) Logic symbol

C	S	R	Q_n	Q_{n+1}	State
↑	0	0	0	0	No Change (NC)
↑	0	0	1	1	
↑	0	1	0	0	Reset
↑	0	1	1	0	
↑	1	0	0	1	Set
↑	1	0	1	1	
↑	1	1	0	×	Indeterminate
↑	1	1	1	×	
0	×	×	0	0	No Change (NC)
0	×	×	1	1	

The edge-triggered S-R flip-flop

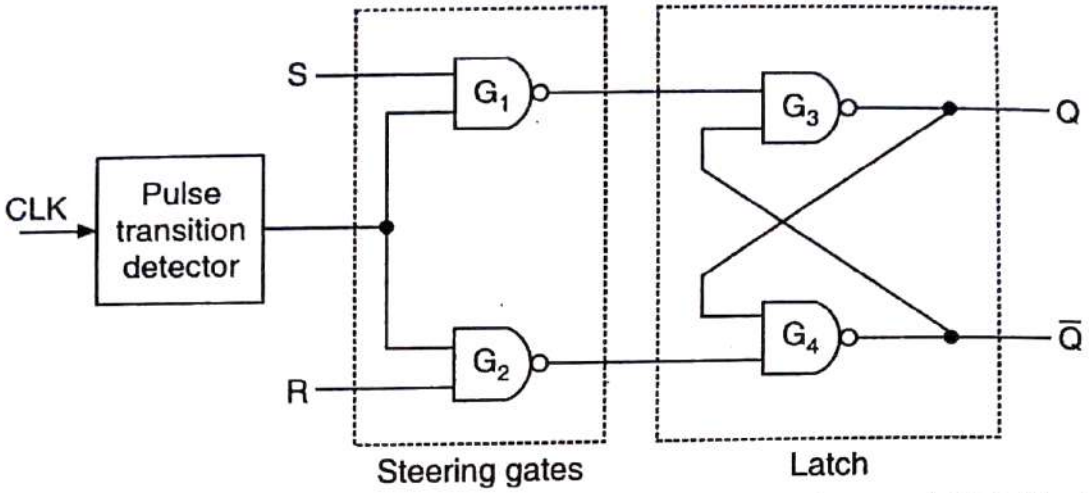


Figure 10.14 Simplified circuit diagram of the edge-triggered S-R flip-flop.

EXAMPLE 10.2 The waveforms shown in Figure 10.13a are applied to the positive edge-triggered S-R flip-flop shown in Figure 10.13b. Sketch the output waveforms.

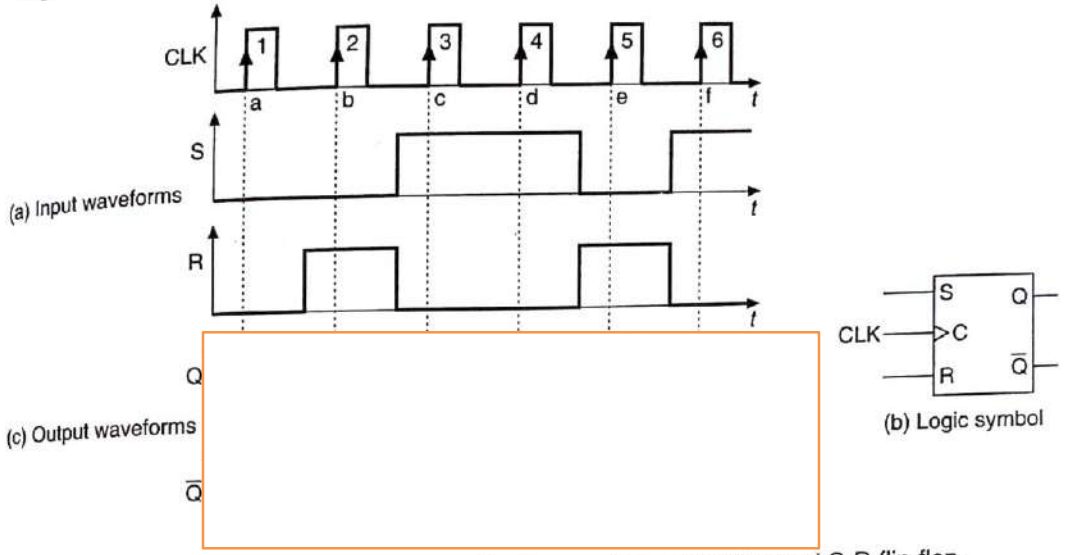


Figure 10.13 Example 10.2: Waveforms—positive edge-triggered S-R flip-flop.

EXAMPLE 10.2 The waveforms shown in Figure 10.13a are applied to the positive edge-triggered S-R flip-flop shown in Figure 10.13b. Sketch the output waveforms.

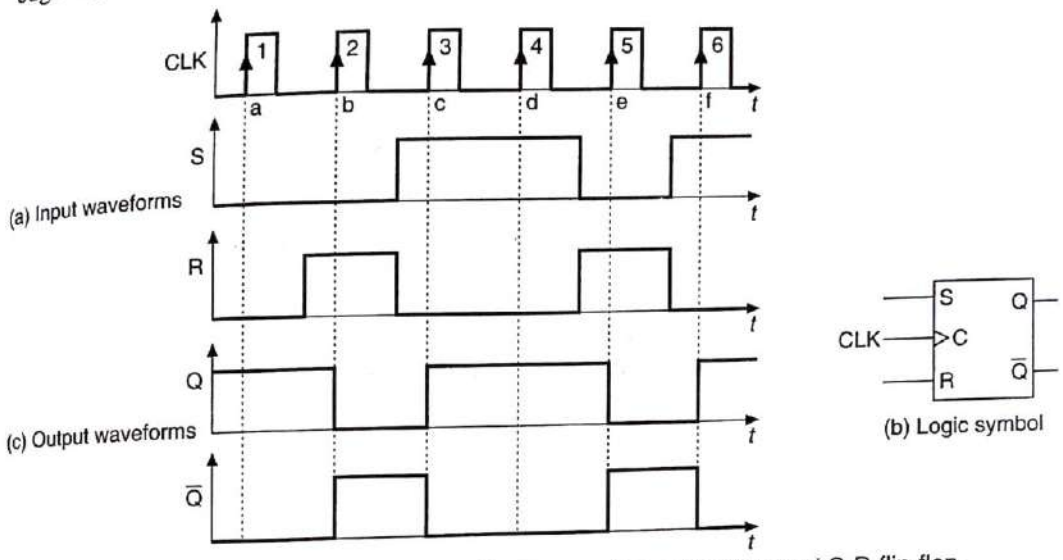
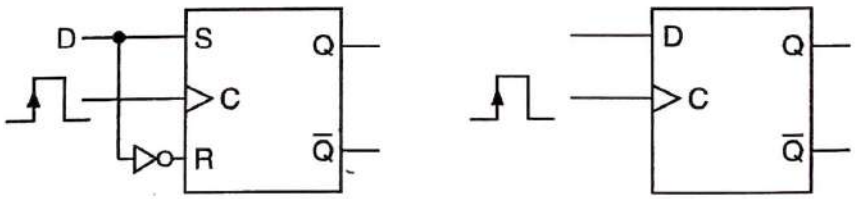


Figure 10.13 Example 10.2: Waveforms—positive edge-triggered S-R flip-flop.

The edge-triggered D flip-flop



(a) D flip-flop from the S-R flip-flop (b) Logic symbol

C	D	Q_n	Q_{n+1}	State
↑	0	0	0	Reset
↑	0	1	0	
↑	1	0	1	Set
↑	1	1	1	
0	×	0	0	No Change (NC)
0	×	1	1	

(c) Truth table

Figure 10.15 The positive edge-triggered D flip-flop.

The edge-triggered D flip-flop

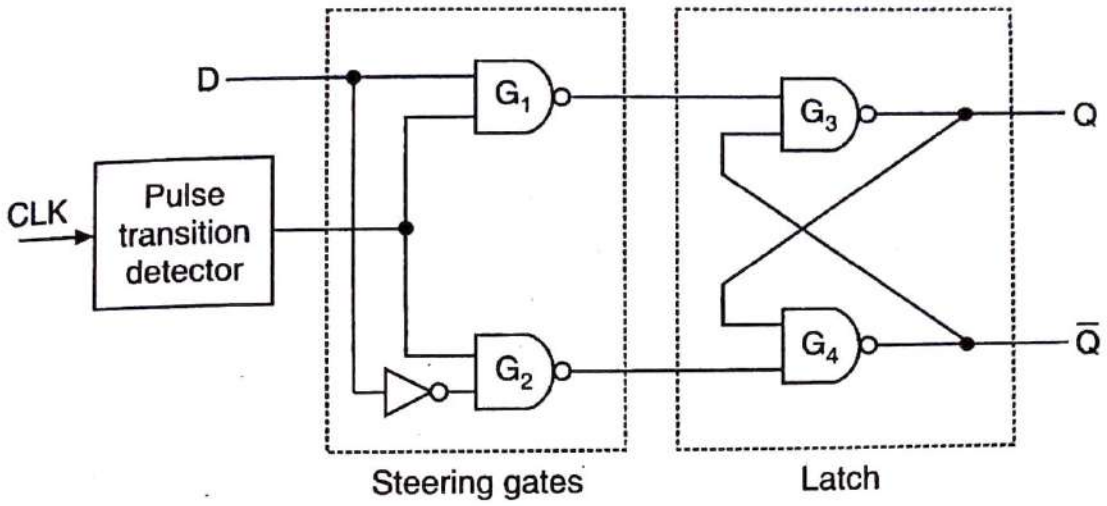
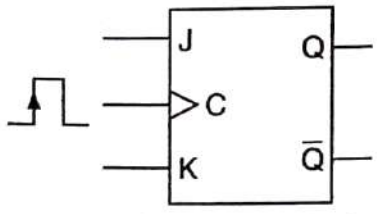


Figure 10.16 Simplified circuit diagram of the edge-triggered D flip-flop.

The edge-triggered J-K flip-flop



(a) Logic symbol

C	J	K	Q_n	Q_{n+1}	State
↑	0	0	0	0	No Change (NC)
↑	0	0	1	1	
↑	0	1	0	0	Reset
↑	0	1	1	0	
↑	1	0	0	1	Set
↑	1	0	1	1	
↑	1	1	0	1	Toggle
↑	1	1	1	0	
0	×	×	0	0	No Change (NC)
0	×	×	1	1	

(b) Truth table

Figure 10.17 Positive edge-triggered J-K flip-flop.

The edge-triggered J-K flip-flop

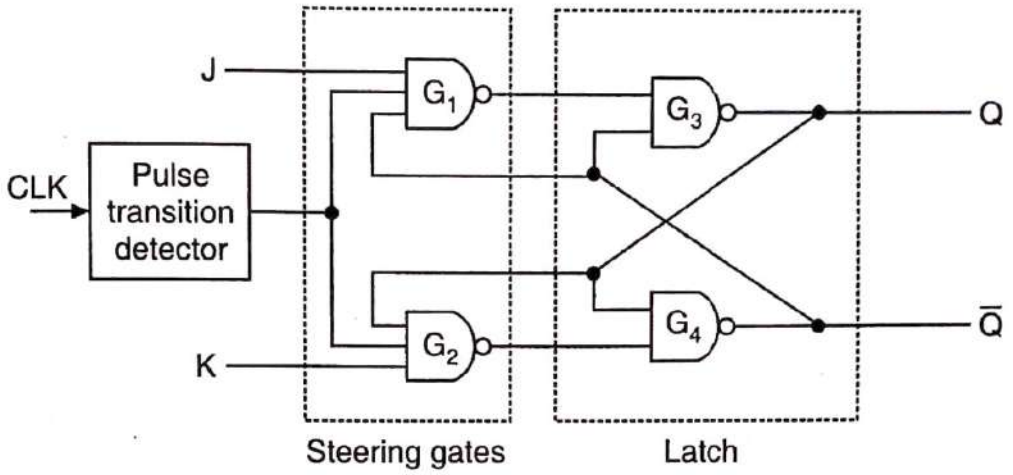
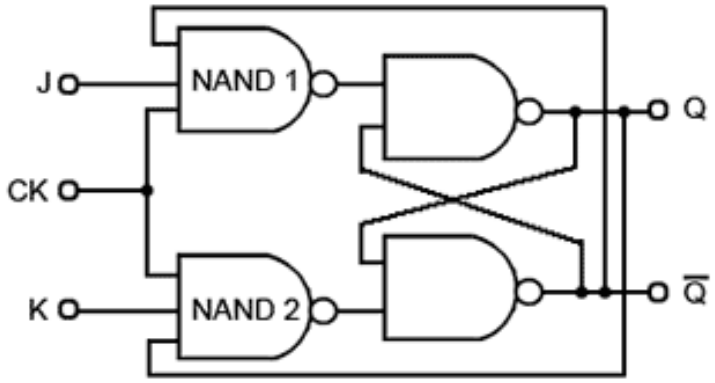


Figure 10.19 Simplified circuit diagram of the edge-triggered J-K flip-flop.

The edge-triggered J-K flip-flop



The edge-triggered J-K flip-flop

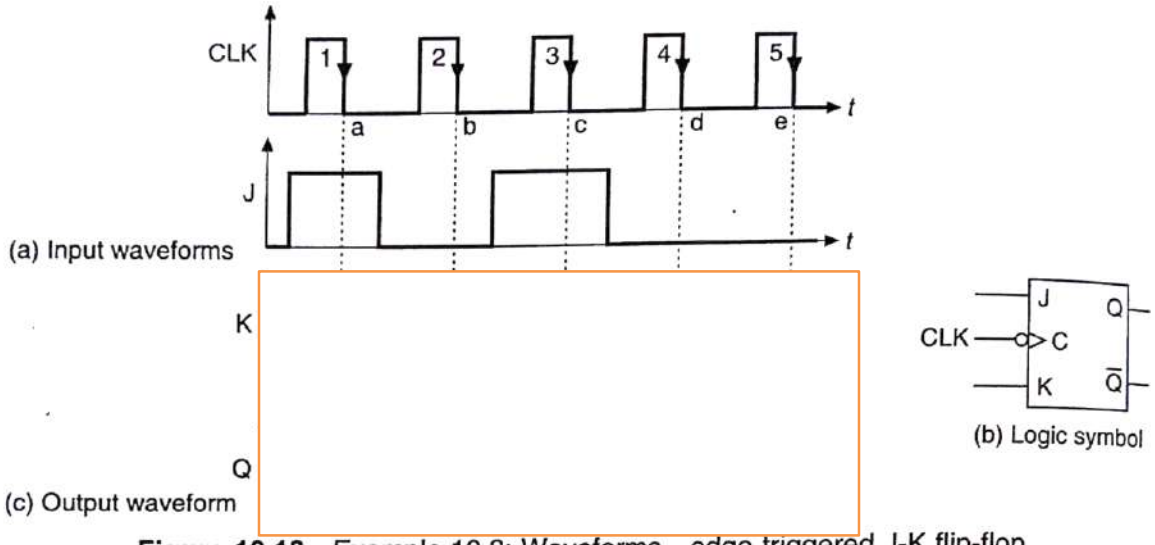


Figure 10.18 Example 10.3: Waveforms—edge-triggered J-K flip-flop.

The edge-triggered J-K flip-flop

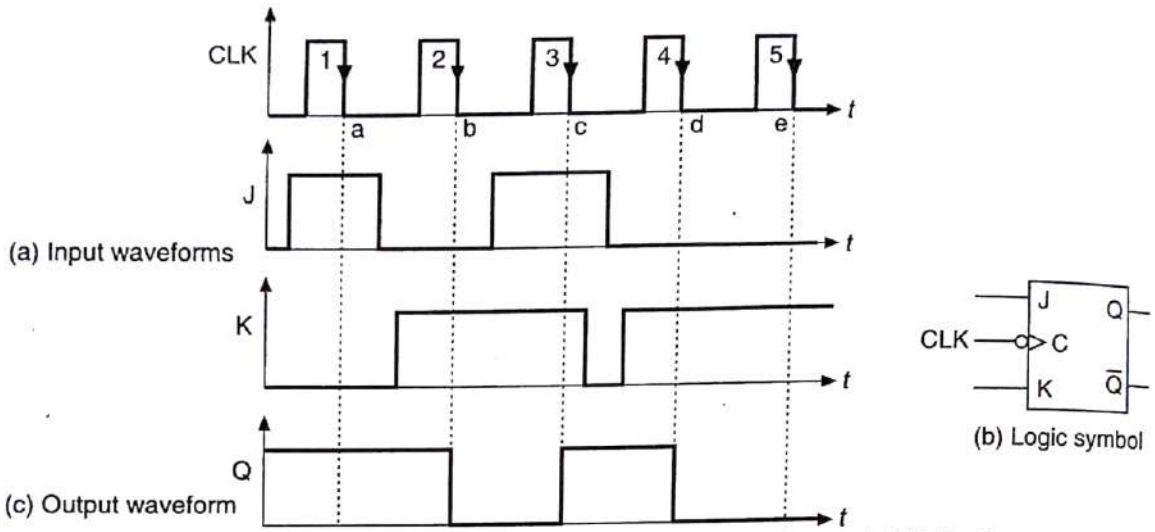
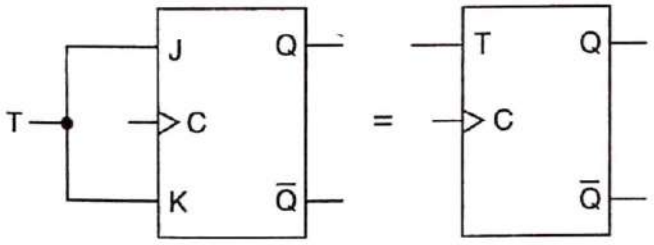


Figure 10.18 Example 10.3: Waveforms—edge-triggered J-K flip-flop.

The edge-triggered T flip-flop



(a) T flip-flop from JK flip-flop (b) Logic symbol

C	T	Q_n	Q_{n+1}	State
↑	0	0	0	No Change (NC)
↑	0	1	1	
↑	1	0	1	Toggle
↑	1	1	0	
0	×	0	0	No Change (NC)
0	×	1	1	

(c) Truth table

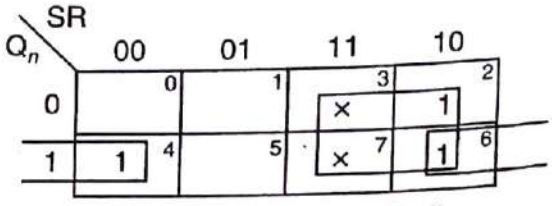
Figure 10.20 Edge-triggered T flip-flop.

Triggering and Characteristic Equations of Flip-Flops

SR Flip Flop

Present state	Inputs		Next state
Q_n	S	R	Q_{n+1}
0	0	0	0
0	0	1	0
0	1	0	1
1	0	0	1
1	0	1	0
1	1	0	1

Excitation requirements of SR flip-flop



K-map for Q_{n+1} of SR flip-flop

The characteristic equation of SR flip-flop

$Q_{n+1} = S + Q_n\overline{R}$

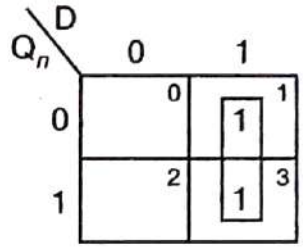
Figure 10.23 Excitation requirements and K-map of an S-R flip-flop.

Triggering and Characteristic Equations of Flip-Flops

D Flip Flop

Present state Q_n	Input D	Next state Q_{n+1}

Excitation requirements of D flip-flop



K-map for Q_{n+1} of D flip-flop

The characteristic equation of D flip-flop is $Q_{n+1} = D$

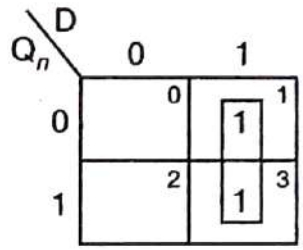
Figure 10.25 Excitation requirements and K-map of a D flip-flop.

Triggering and Characteristic Equations of Flip-Flops

D Flip Flop

Present state Q_n	Input D	Next state Q_{n+1}
0	0	0
0	1	1
1	0	0
1	1	1

Excitation requirements of D flip-flop



K-map for Q_{n+1} of D flip-flop

The characteristic equation of D flip-flop is $Q_{n+1} = D$

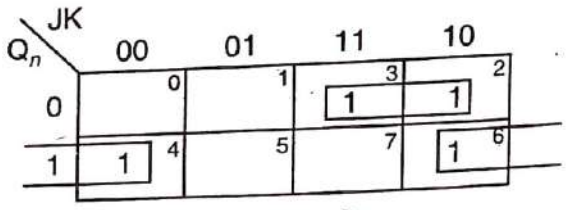
Figure 10.25 Excitation requirements and K-map of a D flip-flop.

Triggering and Characteristic Equations of Flip-Flops

JK Flip Flop

Present state	Inputs		Next state
Q_n	J	K	Q_{n+1}

Excitation requirements of JK flip-flop



K-map for Q_{n+1}

The characteristic equation of a JK flip-flop is

$$Q_{n+1} = \overline{Q}_n J + Q_n \overline{K}$$

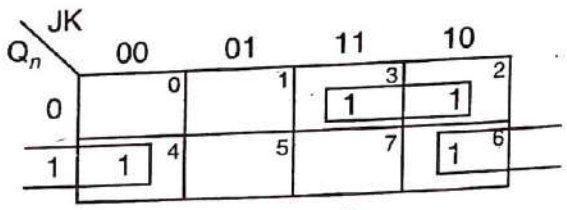
Figure 10.22 Excitation requirements and K-map of a JK flip-flop.

Triggering and Characteristic Equations of Flip-Flops

JK Flip Flop

Present state	Inputs		Next state
Q_n	J	K	Q_{n+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Excitation requirements of JK flip-flop



K-map for Q_{n+1}

The characteristic equation of a JK flip-flop is

$$Q_{n+1} = \overline{Q}_n J + Q_n \overline{K}$$

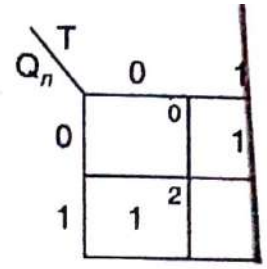
Figure 10.22 Excitation requirements and K-map of a JK flip-flop.

Triggering and Characteristic Equations of Flip-Flops

T Flip Flop

Present state Q_n	Input T	Next state Q_{n+1}

Excitation requirements of T flip-flop



K-map for Q_{n+1} of T flip-flop

The characteristic equation of T flip-flop is

$$Q_{n+1} = \overline{Q}_n T + Q_n \overline{T}$$

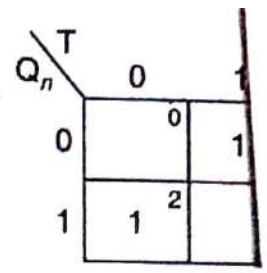
Figure 10.24 Excitation requirements and K-map of a T flip-flop.

Triggering and Characteristic Equations of Flip-Flops

T Flip Flop

Present state Q_n	Input T	Next state Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

Excitation requirements of T flip-flop



K-map for Q_{n+1} of T flip-flop

The characteristic equation of T flip-flop is

$$Q_{n+1} = \overline{Q}_n T + Q_n \overline{T}$$

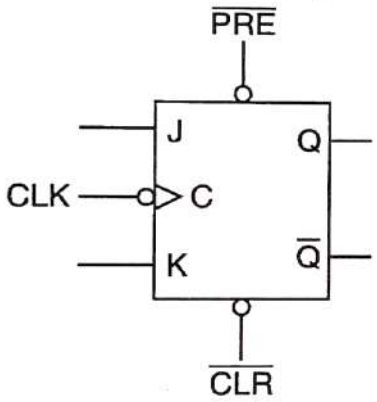
Figure 10.24 Excitation requirements and K-map of a T flip-flop.

The characteristic equations of flip-flops

Table 10.3 Characteristic equations of flip-flops

Flip-flop	Characteristic equation
D	$Q_{n+1} = D$
J-K	$Q_{n+1} = J\overline{Q}_n + \overline{K}Q_n$
T	$Q_{n+1} = T\overline{Q}_n + \overline{T}Q_n$
S-R	$Q_{n+1} = S + \overline{R}Q_n$

ASYNCHRONOUS INPUTS of flip-flop



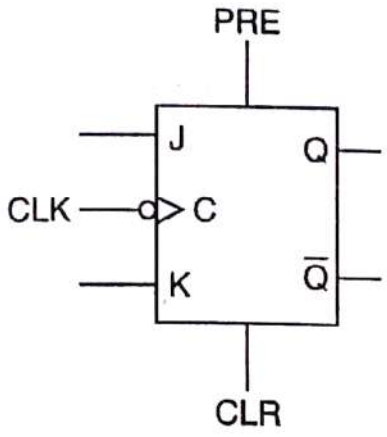
(a) Logic symbol

DC SET ($\overline{PRE}$)	DC RESET ($\overline{CLR}$)	FF response
0	0	Not used
1	0	Q = 0
0	1	Q = 1
1	1	Clocked operation

(b) Truth table

Figure 10.26 J-K flip-flop with active-LOW PRESET and CLEAR inputs.

ASYNCHRONOUS INPUTS of flip-flop



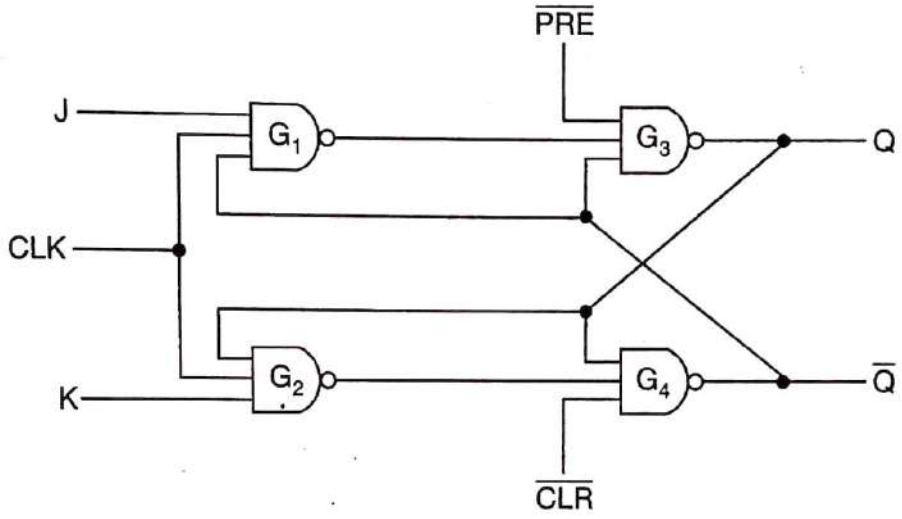
(a) Logic symbol

DC SET (PRE)	DC RESET (CLR)	FF response
0	0	Clocked operation
0	1	Q = 0
1	0	Q = 1
1	1	Not used

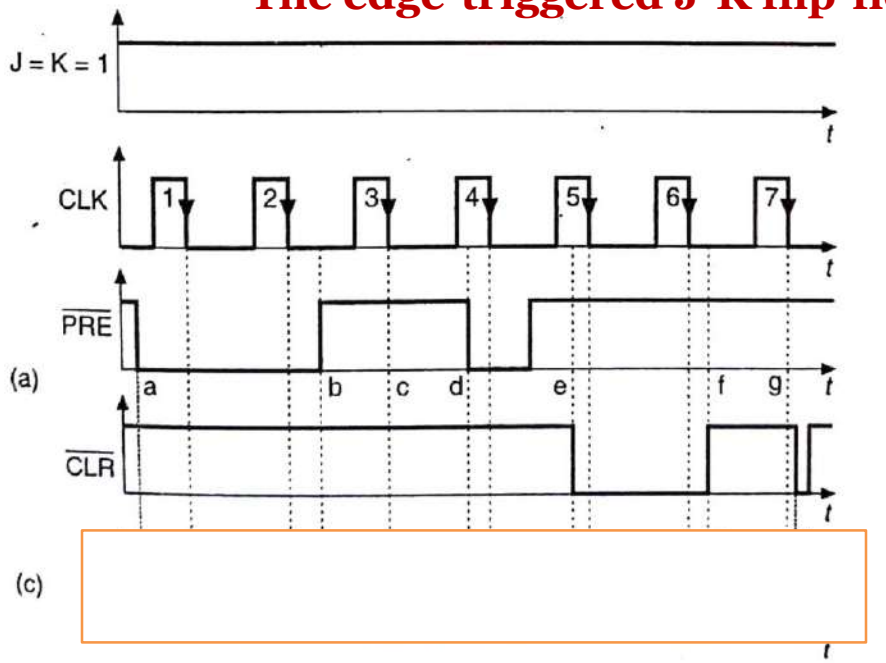
(b) Truth table

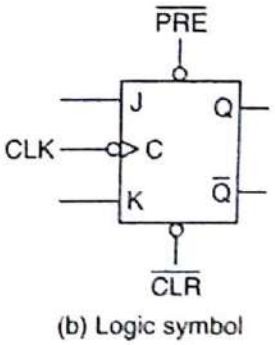
Figure 10.27 J-K flip-flop with active-HIGH PRESET and CLEAR inputs.

J-K flip-flop with active-LOW PRESET and CLEAR



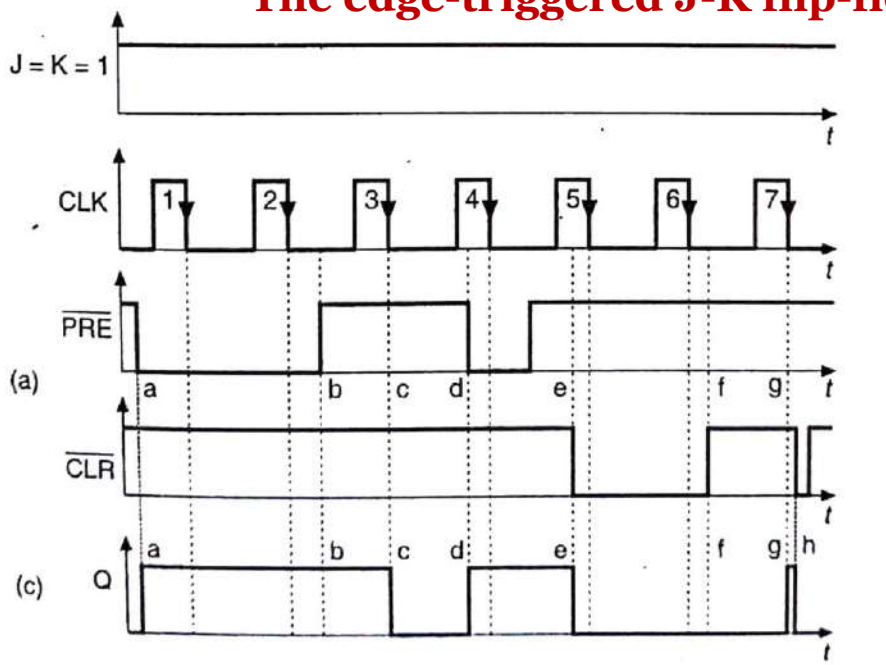
The edge-triggered J-K flip-flop

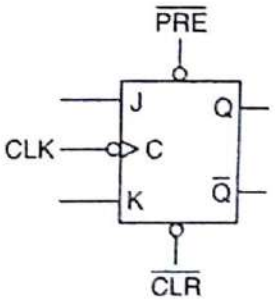




(b) Logic symbol

The edge-triggered J-K flip-flop





(b) Logic symbol

FLIP-FLOP OPERATING CHARACTERISTICS

1. Propagation delay time
2. Set-up time
3. Hold time
4. Maximum clock frequency
5. Pulse widths
6. Clock transition times
7. Power dissipation

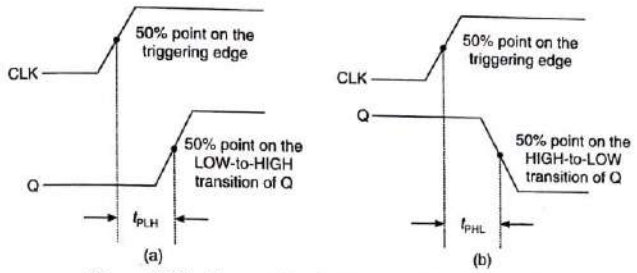


Figure 10.30 Propagation delays t_{PLH} and t_{PHL} w.r.t. CLK.

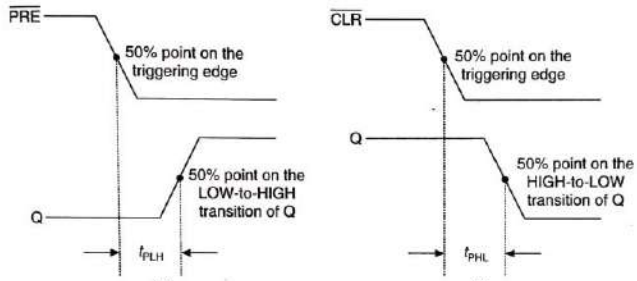


Figure 10.31 Propagation delays t_{PLH} and t_{PHL} w.r.t. PRESET and CLEAR.

FLIP-FLOP OPERATING CHARACTERISTICS

1. Propagation delay time
2. Set-up time
3. Hold time
4. Maximum clock frequency
5. Pulse widths
6. Clock transition times
7. Power dissipation

$$P = V_{CC} \cdot I_{CC}$$

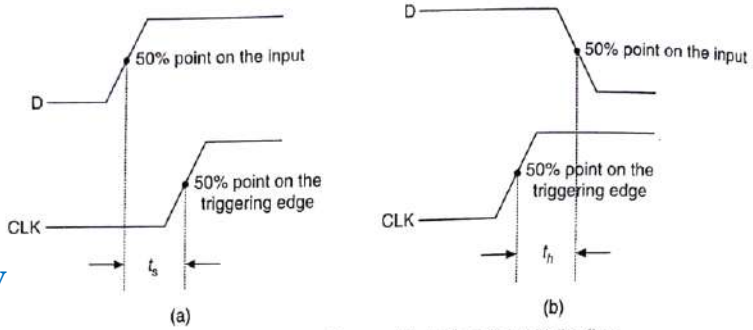


Figure 10.32 Set-up time and hold time for a D flip-flop.

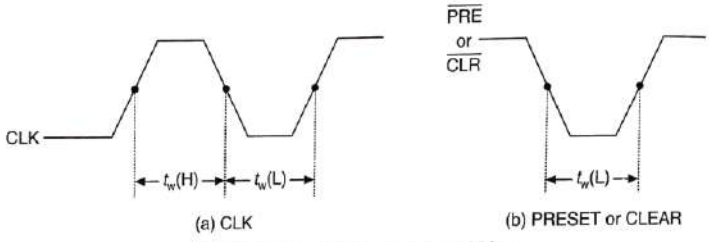


Figure 10.33 Minimum pulse widths.

CLOCK SKEW AND TIME RACE

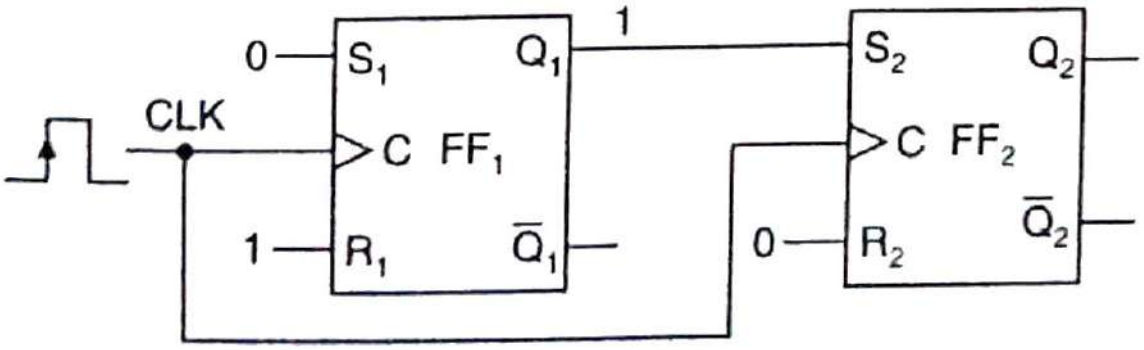


Figure 10.34 Illustration of timing problem.

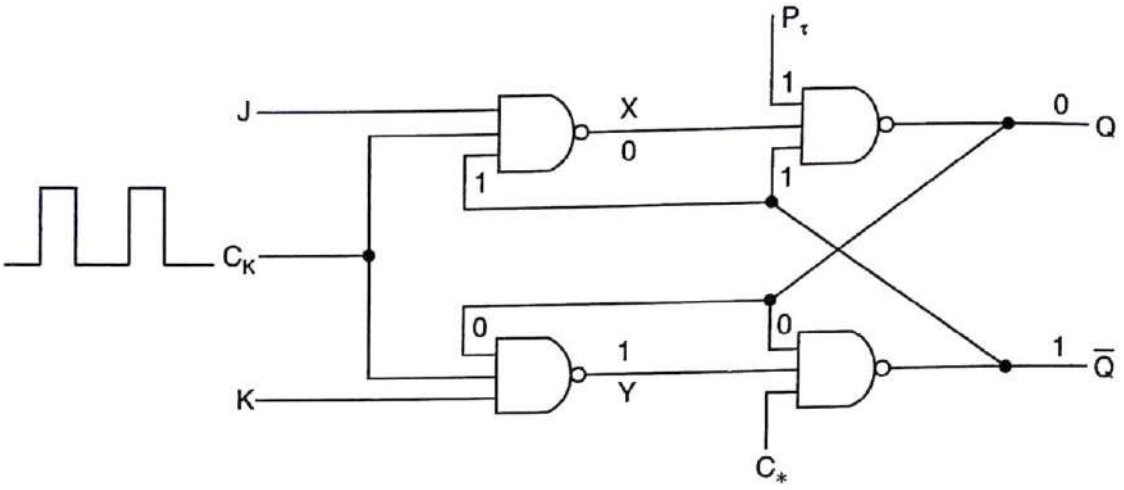
RACE AROUND CONDITION

In a J-K flip-flop when excitations J = K = 1. If the width of the clock pulse t_p is too long, the state of the flip-flop will keep on changing from 0 to 1, 1 to 0,0 to 1 and so on, and at the end of the clock pulse, its state will be uncertain. This phenomenon is called the *race around condition*.

The outputs Q and Q will change on their own if the clock pulse width t_p is too long compared with the propagation delay τ of each NAND gate.

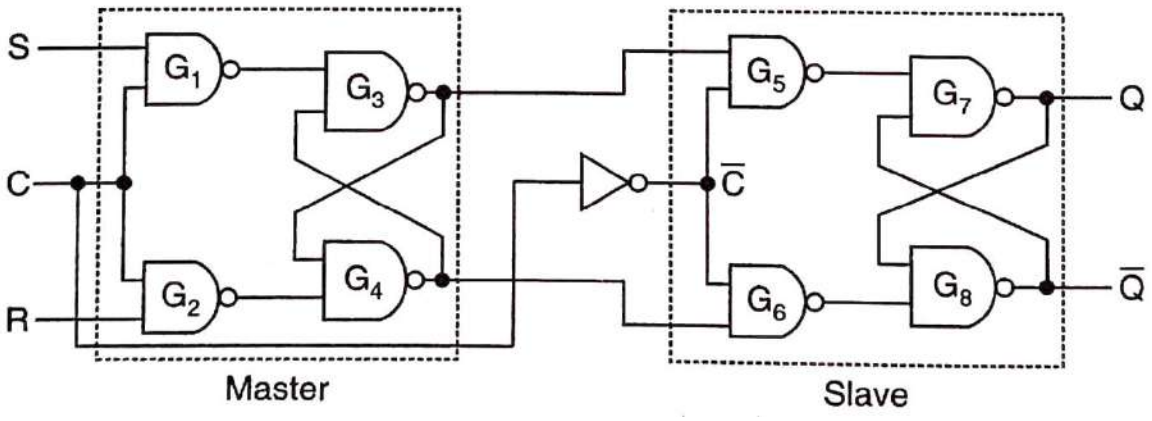
$$t_p >> \tau$$

FLIP-FLOP OPERATING CHARACTERISTICS

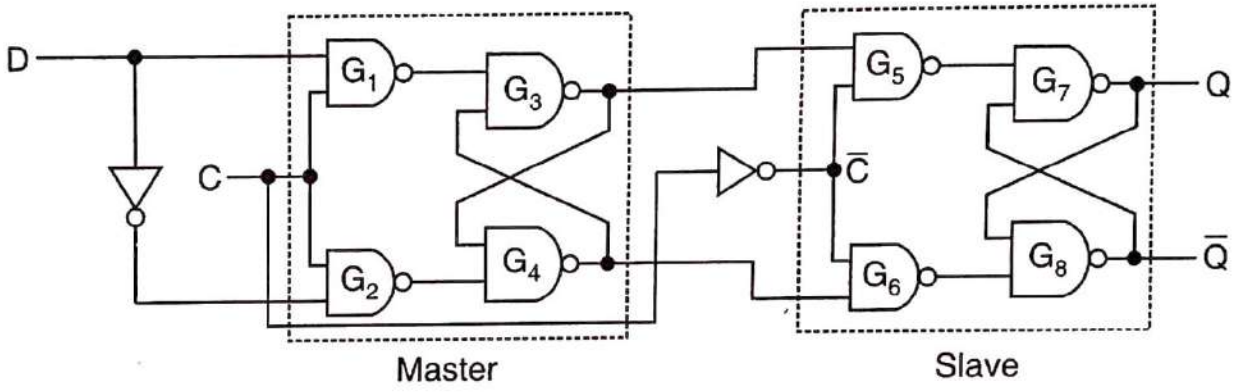


MASTER-SLAVE (PULSE-TRIGGERED) FLIP-FLOPS

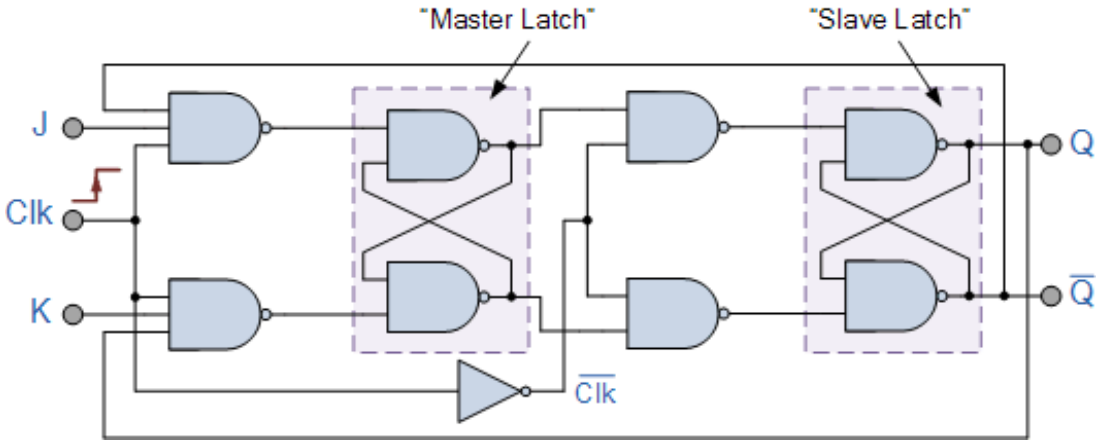
Before the development of edge-triggered flip-flops, the timing problems were often handled by a class of flip-flops called the **master-slave flip-flops.**



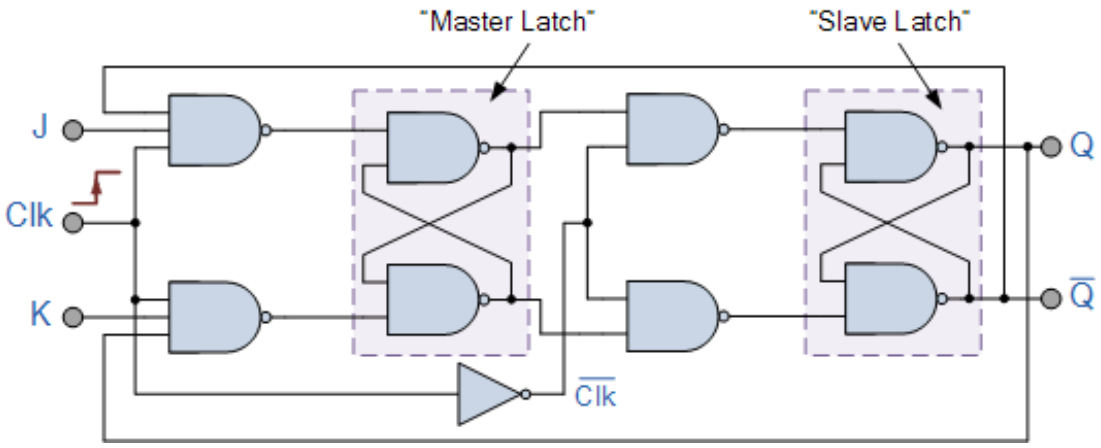
The Master-Slave (Pulse-Triggered) D Flip-Flop



The Master-Slave (Pulse-Triggered) JK Flip-Flop



The Master-Slave (Pulse-Triggered) T Flip-Flop



FLIP-FLOP EXCITATION TABLES

SR Flip Flop

Present state	Inputs		Next state
Q_n	S	R	Q_{n+1}
0	0	0	0
0	0	1	0
0	1	0	1
1	0	0	1
1	0	1	0
1	1	0	1

Table 10.5b S-R excitation table

PS	NS	Required inputs	
Q_n	Q_{n+1}	S	R
0	0	0	×
0	1	1	0
1	0	0	1
1	1	×	0

FLIP-FLOP EXCITATION TABLES

D Flip Flop

Present state	Input	Next state
Q_n	D	Q_{n+1}
0	0	0
0	1	1
1	0	0
1	1	1

Table 10.7b D excitation table

PS	NS	Required input
Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

FLIP-FLOP EXCITATION TABLES

JK Flip Flop

Present state	Inputs		Next state
Q_n	J	K	Q_{n+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Table 10.6b J-K excitation table

PS	NS	Required inputs	
Q_n	Q_{n+1}	J	K
0	0	0	×
0	1	1	×
1	0	×	1
1	1	×	0

FLIP-FLOP EXCITATION TABLES

T Flip Flop

Present state Q_n	Input T	Next state Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

Table 10.8b T excitation table

PS Q_n	NS Q_{n+1}	Required input T
0	0	0
0	1	1
1	0	1
1	1	0

CONVERSION OF FLIP-FLOPS

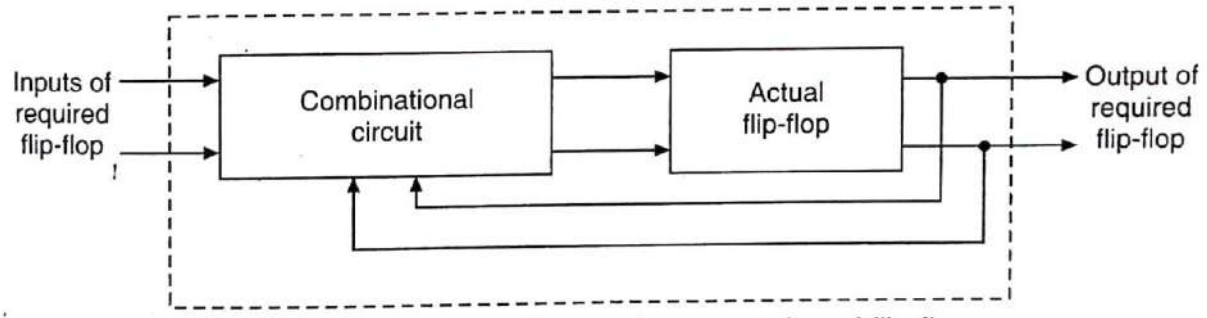


Figure 10.43 Block diagram for conversion of flip-flop.

S-R flip-flop to J-K flip-flop

Table 10.5b S-R excitation table

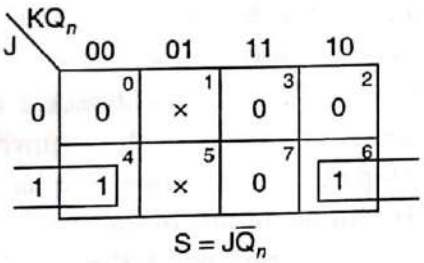
PS	NS	Required inputs	
Q_n	Q_{n+1}	S	R
0	0	0	×
0	1	1	0
1	0	0	1
1	1	×	0

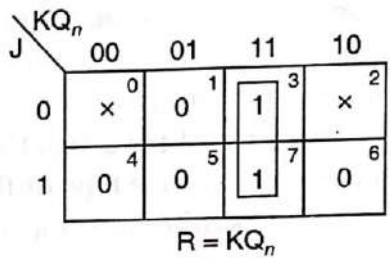
Present state	Inputs		Next state
Q_n	J	K	Q_{n+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

External Inputs		Present State	Next State	Flip-flop Inputs	
J	K	Q_n	Q_{n+1}	S	R
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

(a) Conversion table

S-R flip-flop to J-K flip-flop

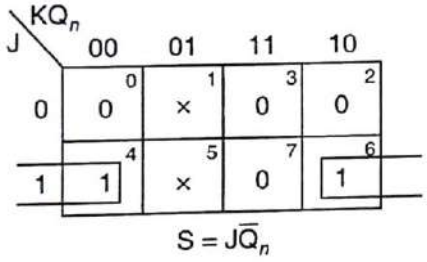


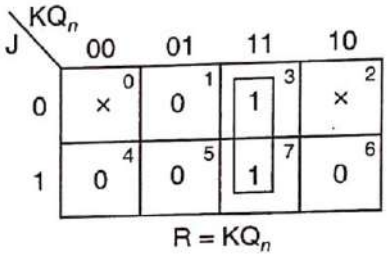


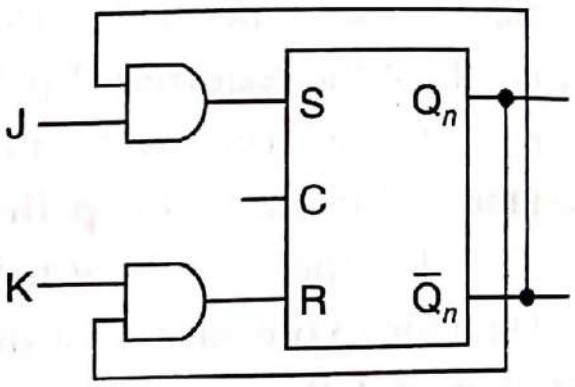
External Inputs		Present State	Next State	Flip-flop Inputs	
J	K	Q_n	Q_{n+1}	S	R
0	0	0	0	0	×
0	0	1	1	×	0
0	1	0	0	0	×
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	1	×	0
1	1	0	1	1	0
1	1	1	0	0	1

(a) Conversion table

S-R flip-flop to J-K flip-flop







(c) Logic diagram

J-K flip-flop to S-R flip-flop

Present state	Inputs		Next state
Q_n	S	R	Q_{n+1}
0	0	0	0
0	0	1	0
0	1	0	1
1	0	0	1
1	0	1	0
1	1	0	1

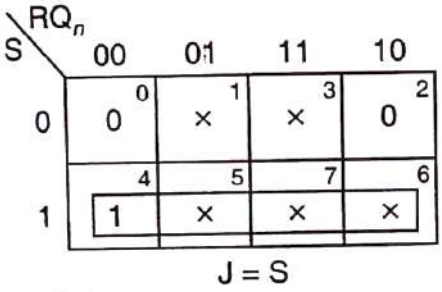
Table 10.6b J-K excitation table

PS	NS	Required inputs	
Q_n	Q_{n+1}	J	K
0	0	0	×
0	1	1	×
1	0	×	1
1	1	×	0

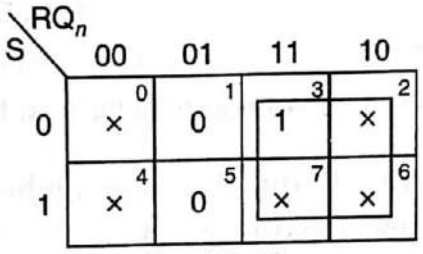
External Inputs		Present State	Next State	Flip-flop Inputs	
S	R	Q_n	Q_{n+1}	J	K
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			

(a) Conversion table

J-K flip-flop to S-R flip-flop



J = S

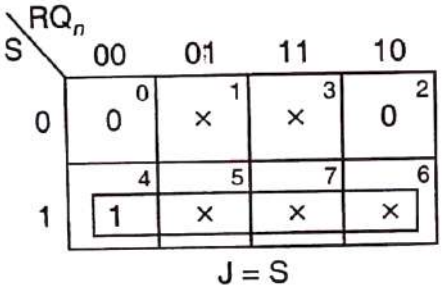


K = R

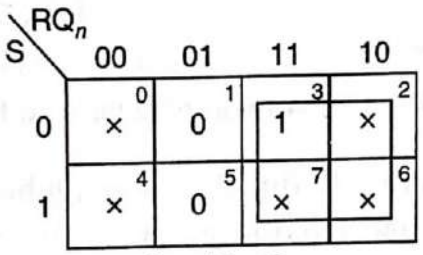
External Inputs		Present State	Next State	Flip-flop Inputs	
S	R	Q_n	Q_{n+1}	J	K
0	0	0	0	0	×
0	0	1	1	×	0
0	1	0	0	0	×
0	1	1	0	×	1
1	0	0	1	1	×
1	0	1	1	×	0

(a) Conversion table

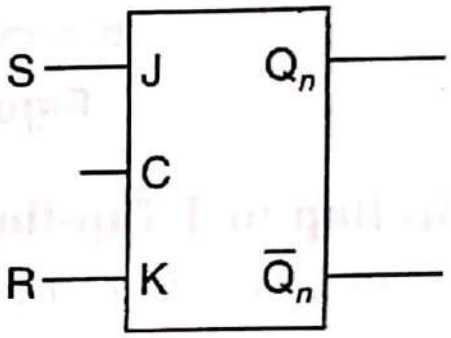
J-K flip-flop to S-R flip-flop



J = S



K = R



D flip-flop to S-R flip-flop

Present state	Inputs		Next state
Q_n	S	R	Q_{n+1}
0	0	0	0
0	0	1	0
0	1	0	1
1	0	0	1
1	0	1	0
1	1	0	1

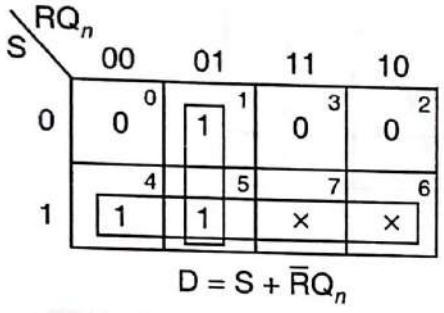
Table 10.7b D excitation table

PS	NS	Required input
Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

External Inputs		Present State	Next State	Flip-flop Input
S	R	Q_n	Q_{n+1}	D
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		

(a) Conversion table

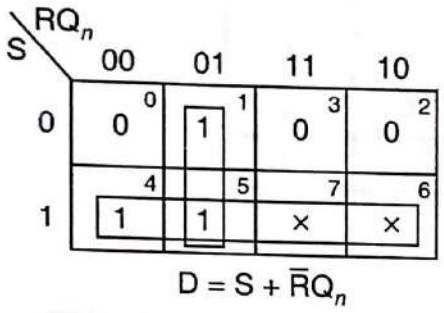
D flip-flop to S-R flip-flop

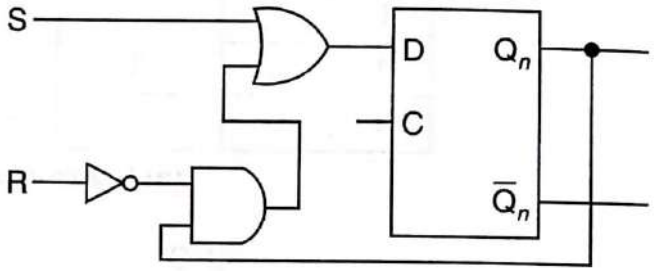


External Inputs		Present State	Next State	Flip-flop Input
S	R	Q_n	Q_{n+1}	D
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	0
1	0	0	1	1
1	0	1	1	1

(a) Conversion table

D flip-flop to S-R flip-flop





APPLICATIONS OF FLIP-FLOPS

1. Registers: Registers are the devices which are meant to store the data. As known, each flip-flop can store a single-bit of information. This means that by cascading n flip-flops, one can store n bits of information. Such an arrangement is called an n-bit register.

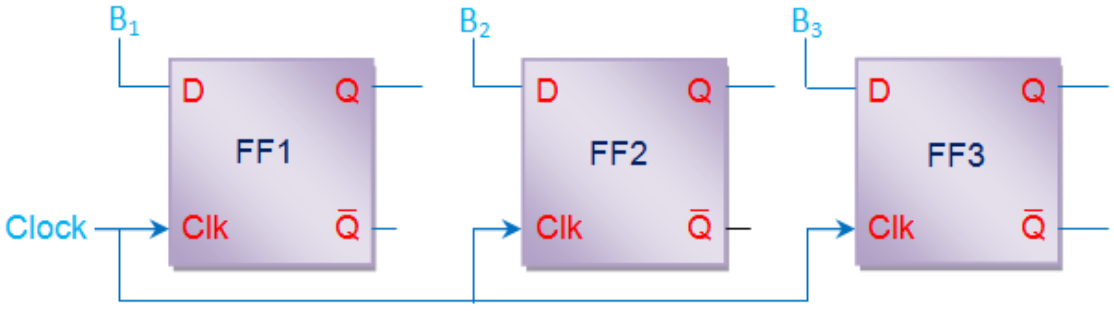


Figure 1 3-bit register formed by cascading three D flip-flops

APPLICATIONS OF FLIP-FLOPS

2. Counters: Counters are the digital circuits which are used to count the number of events. These are nothing but a series of flip-flops (JK or D or T) arranged in a definite manner.

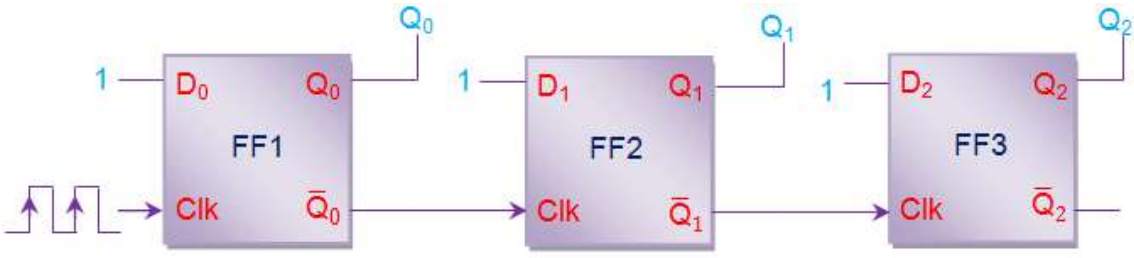
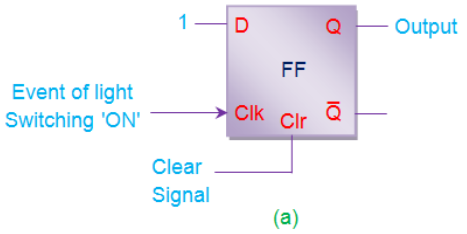


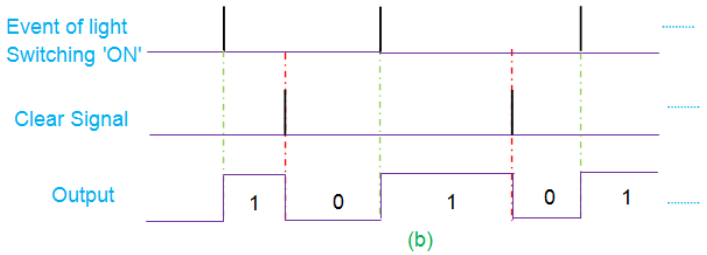
Figure 3 3-bit asynchronous positive edge triggered up-counter

APPLICATIONS OF FLIP-FLOPS

3. Event Detectors: Event detectors are the circuits which aid in determining the occurrence of a particular event.



(a)



(b)

Figure 4 Event detector (a) Circuit (b) Timing diagram

APPLICATIONS OF FLIP-FLOPS

4. Frequency Divider:

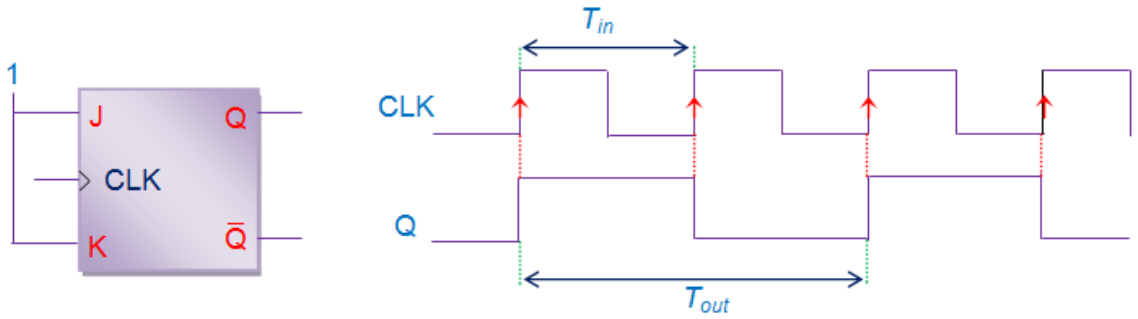


Figure 6 JK flip-flop as a frequency divider

APPLICATIONS OF FLIP-FLOPS

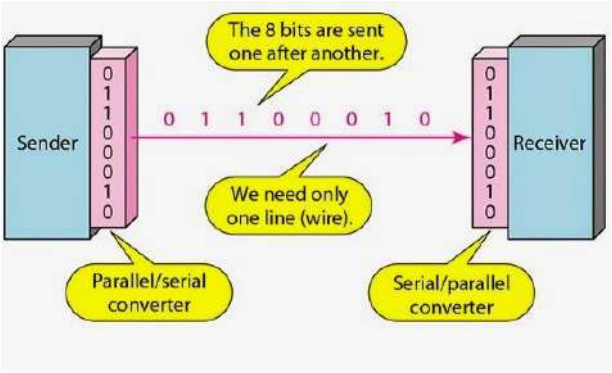
5. Data Storage

6. Serial to Parallel Data Conversion

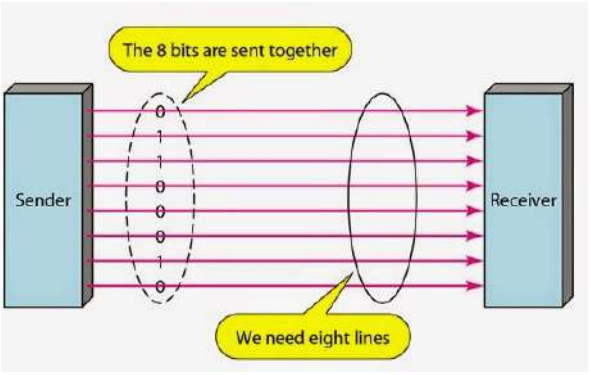
7. ADC and DAC

SHIFT REGISTERS

Data may be available in parallel form or in serial form.



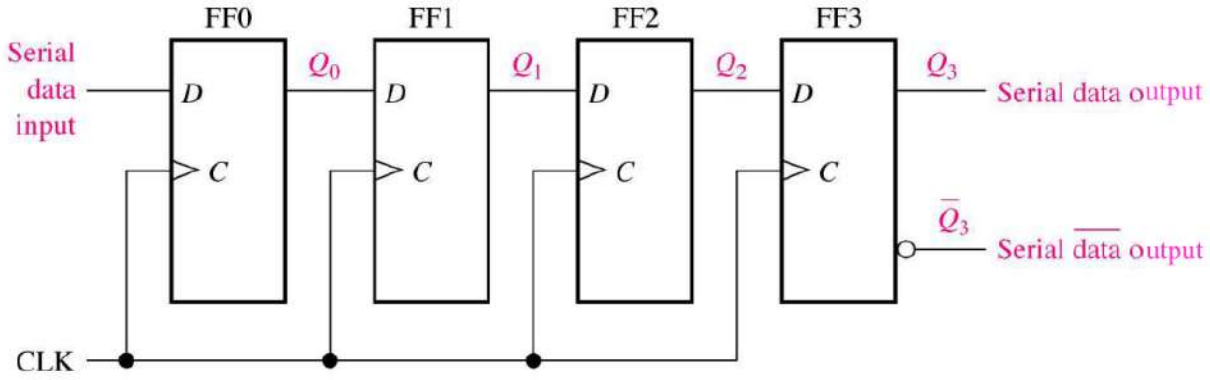
serial communication



parallel communication

SHIFT REGISTERS

As a flip-flop (FF) can store only one bit of data, a 0 or a 1, it is referred to as a single-bit register. When more bits of data are to be stored, a number of FFs are used. A register is a set of FFs used to store binary data.



BUFFER REGISTER

Some registers do nothing more than storing a binary word. The buffer register is the simplest of registers. It simply stores the binary word. The buffer may be a controlled buffer. Most of the buffer registers use D flip-flops.

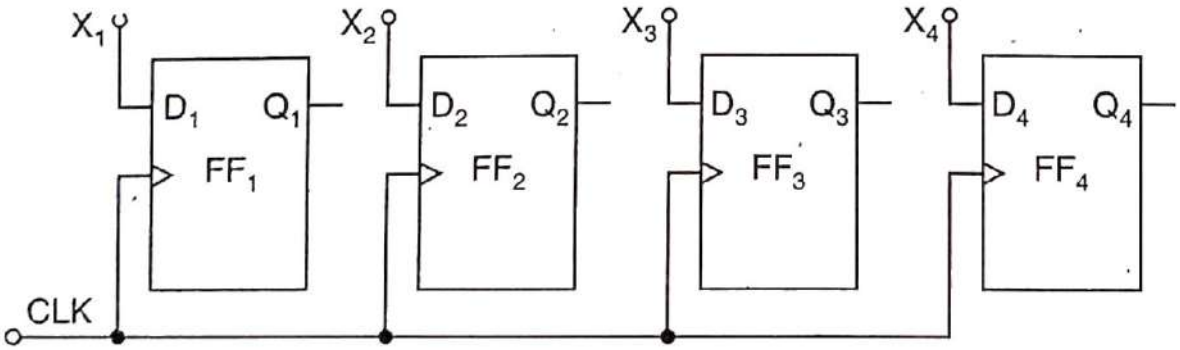
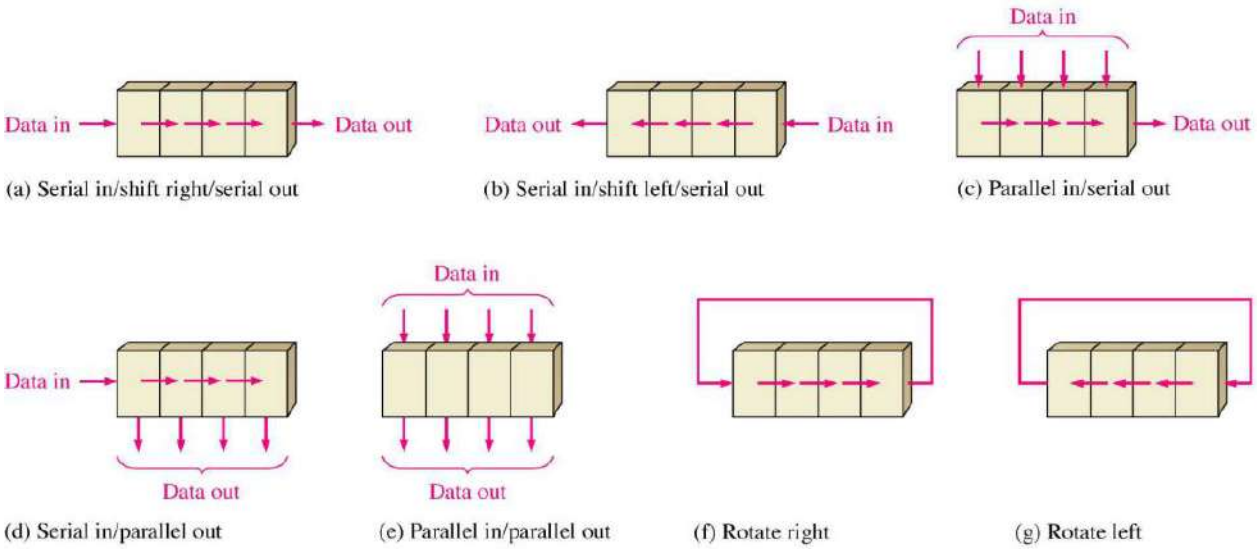
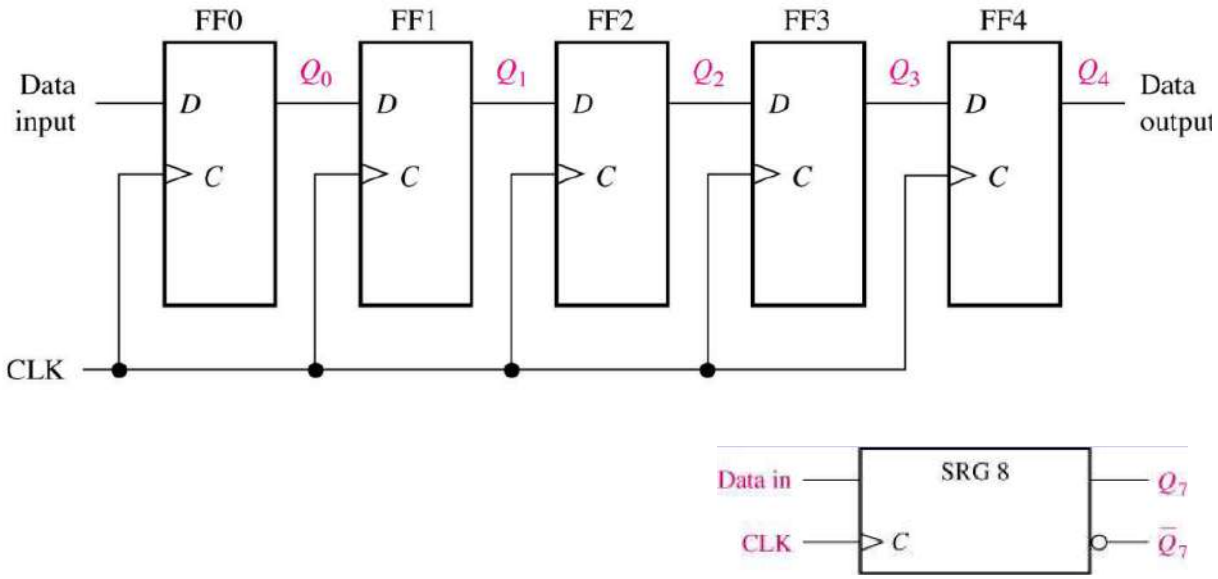


Figure 11.1 Logic diagram of a 4-bit buffer register.

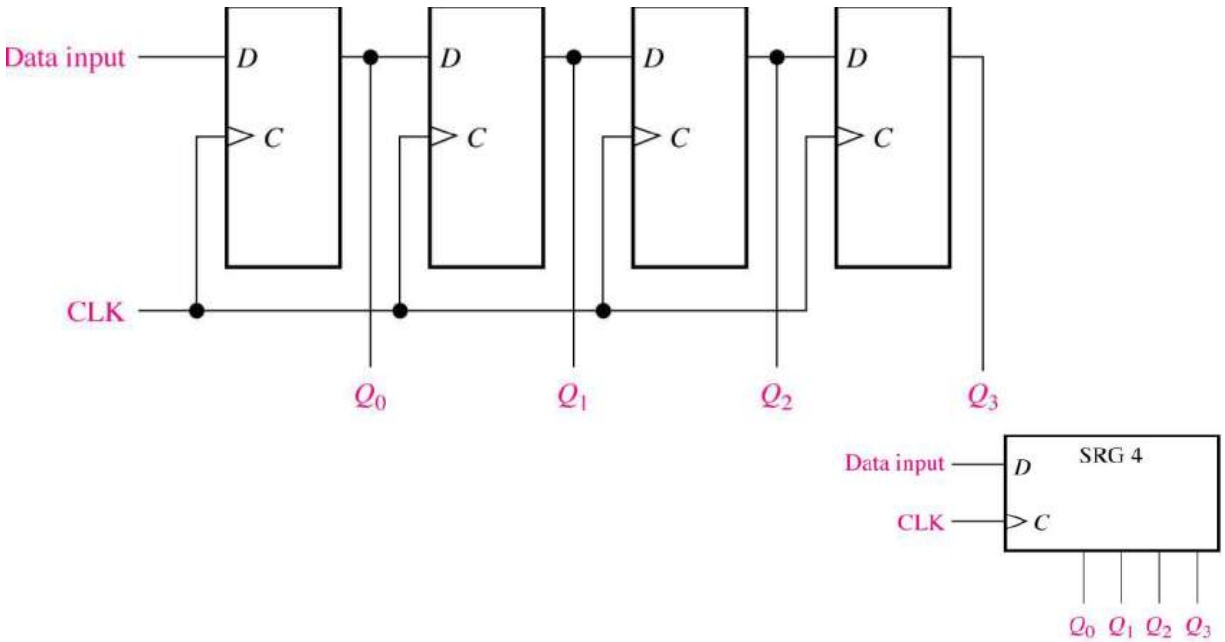
Types of Shift Registers



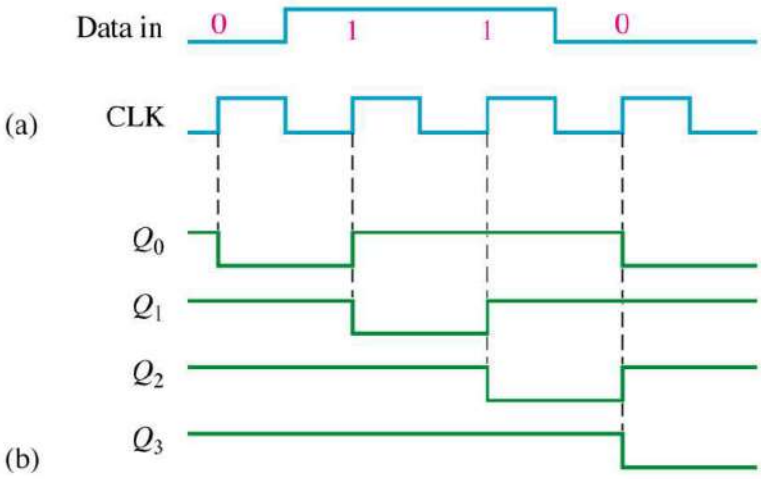
SERIAL-IN, SERIAL-OUT, SHIFT REGISTER

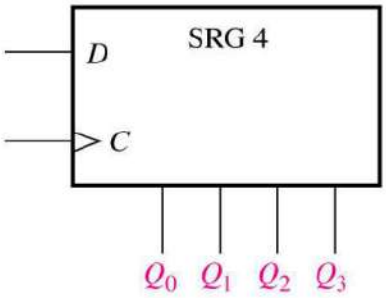


SERIAL-IN, PARALLEL-OUT, SHIFT REGISTER

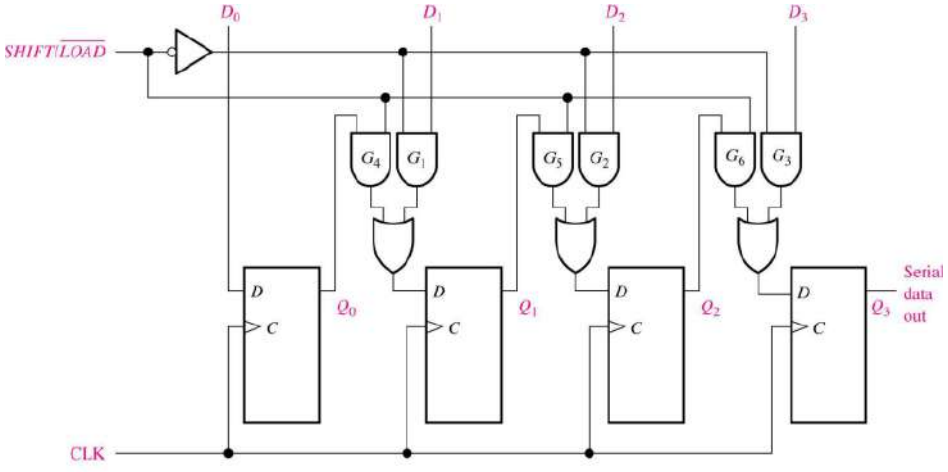


Example

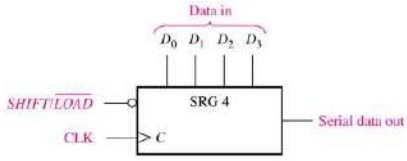




PARALLEL-IN, SERIAL-OUT, SHIFT REGISTER

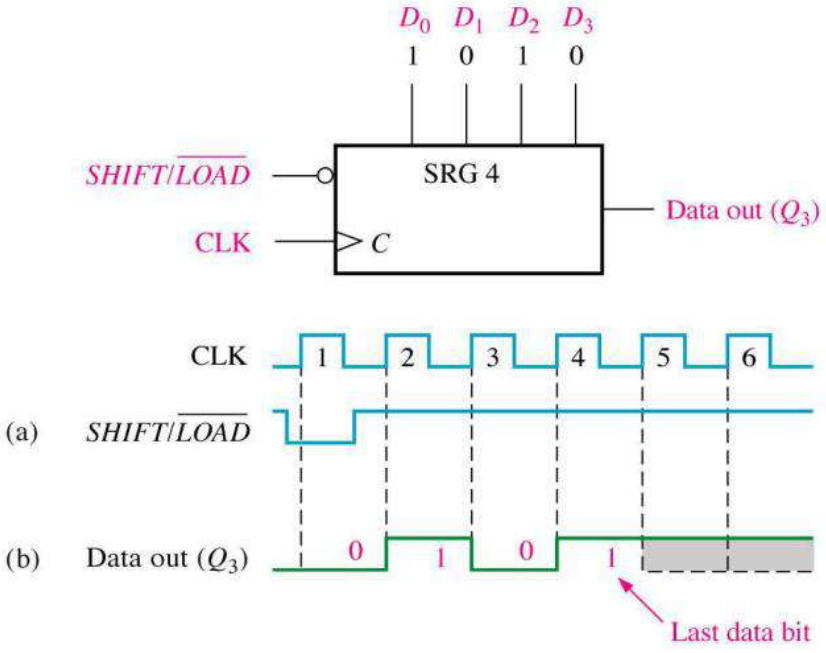


(a) Logic diagram

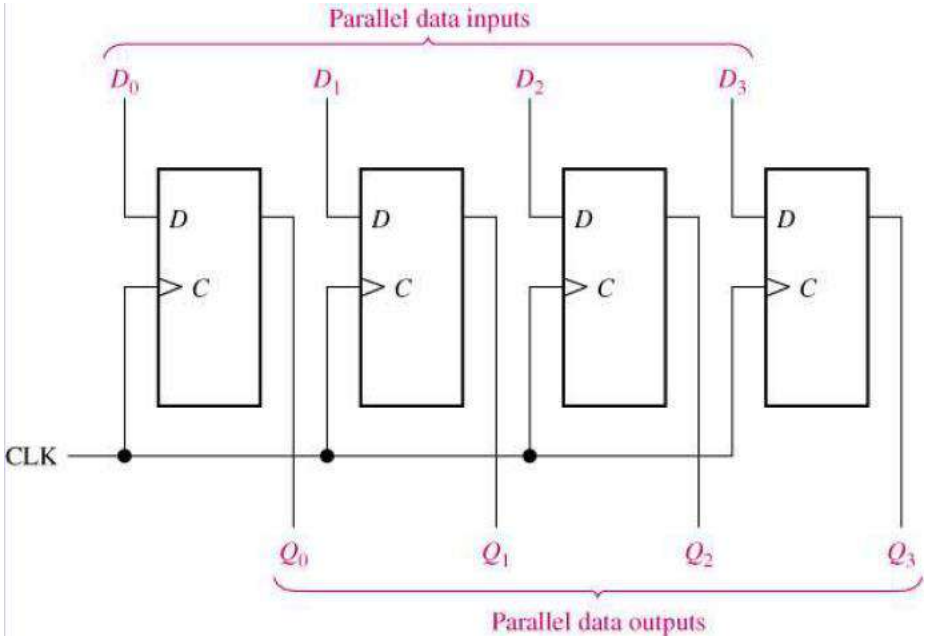


(b) Logic symbol

Example



PARALLEL-IN, PARALLEL-OUT, SHIFT REGISTER



APPLICATIONS OF SHIFT REGISTERS

1. Time delays
2. Serial/Parallel data conversion
3. Ring counters
4. Universal asynchronous receiver transmitter (UART)

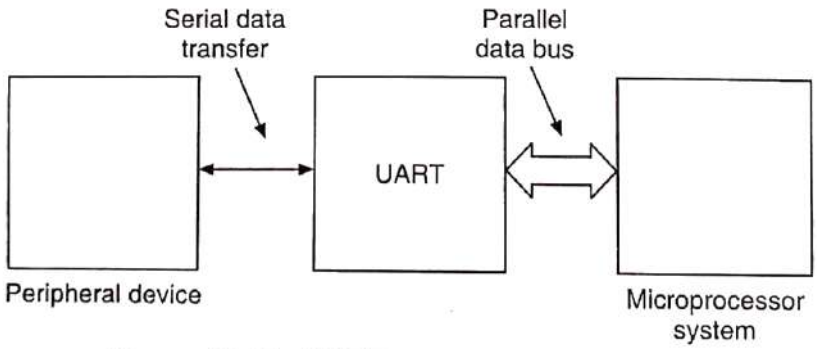


Figure 11.13 UART as an interfacing device.

COUNTER

A digital counter is a set of flip-flops (FFs) whose states change in response to pulses applied at the input to the counter.

The FFs are interconnected such that their combined state at any time is the binary equivalent of the total number of pulses that have occurred up to that time.

Thus, as its name implies, a counter is used to count pulses.

Counters may be asynchronous counters or synchronous counters. Asynchronous counters are also called ripple counters.

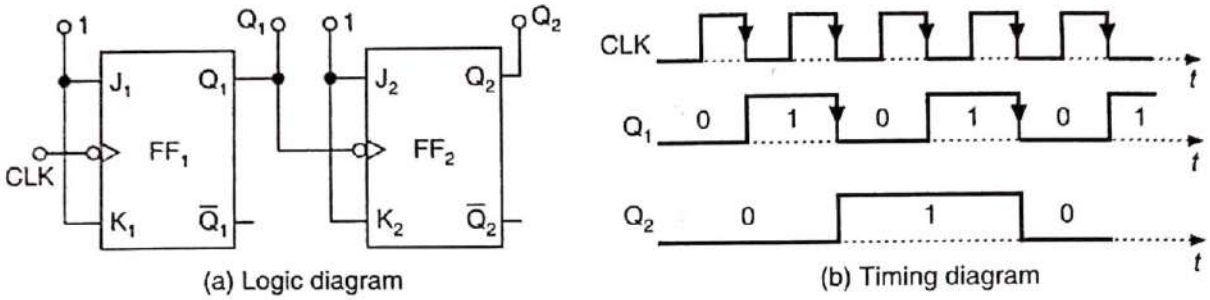
Types of Counter

Table 12.1 Synchronous versus asynchronous counters

Asynchronous counters	Synchronous counters
1. In this type of counter FFs are connected in such a way that the output of first FF drives the clock for the second FF, the output of the second the clock of the third and so on.	1. In this type of counter there is no connection between the output of first FF and clock input of next FF and so on.
2. All the FFs are not clocked simultaneously.	2. All the FFs are clocked simultaneously.
3. Design and implementation is very simple even for more number of states.	3. Design and implementation becomes tedious and complex as the number of states increases.
4. Main drawback of these counters is their low speed as the clock is propagated through a number of FFs before it reaches the last FF.	4. Since clock is applied to all the FFs simultaneously the total propagation delay is equal to the propagation delay of only one FF. Hence they are faster.

ASYNCHRONOUS COUNTERS

Two-bit Ripple Up-counter Using Negative Edge-triggered FF

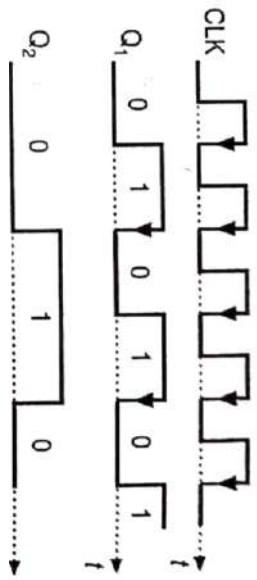


(a) Logic diagram (b) Timing diagram

Figure 12.1 Asynchronous 2-bit up-counter using negative edge-triggered flip-flops.

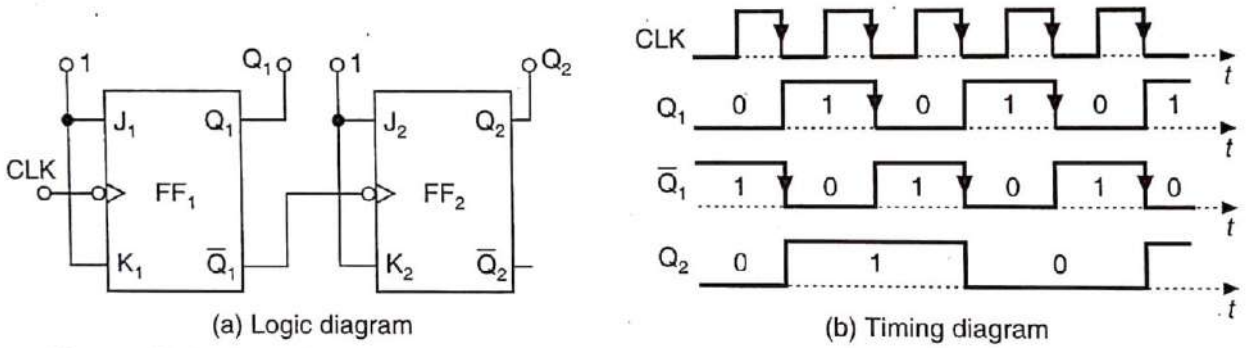
ASYNCHRONOUS COUNTERS

Two-bit Ripple Up-counter Using Negative Edge-triggered FF



ASYNCHRONOUS COUNTERS

Two-bit Ripple Down-counter Using Negative Edge-triggered FF

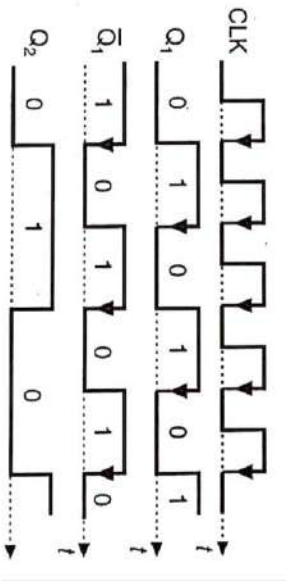


(a) Logic diagram (b) Timing diagram

Figure 12.2 Asynchronous 2-bit down-counter using negative edge-triggered flip-flops.

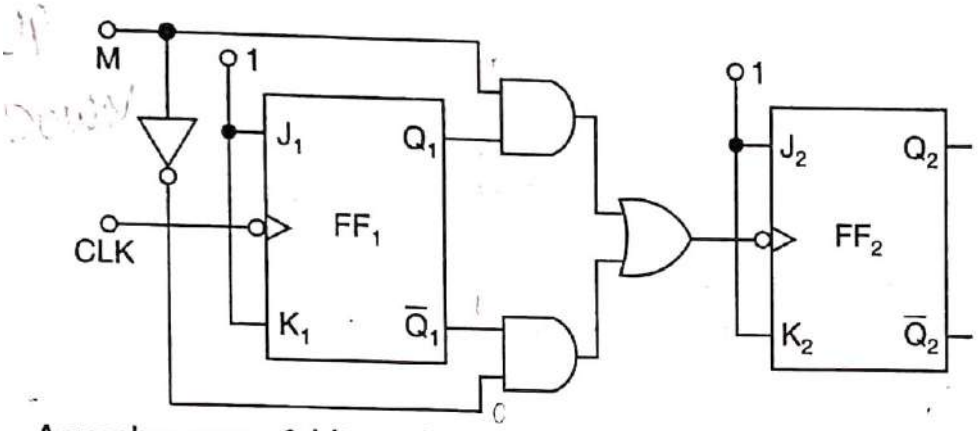
ASYNCHRONOUS COUNTERS

Two-bit Ripple Down-counter Using Negative Edge-triggered FF



Two-bit Ripple Up-down Counter Using Negative Edge-triggered Flip-Flops

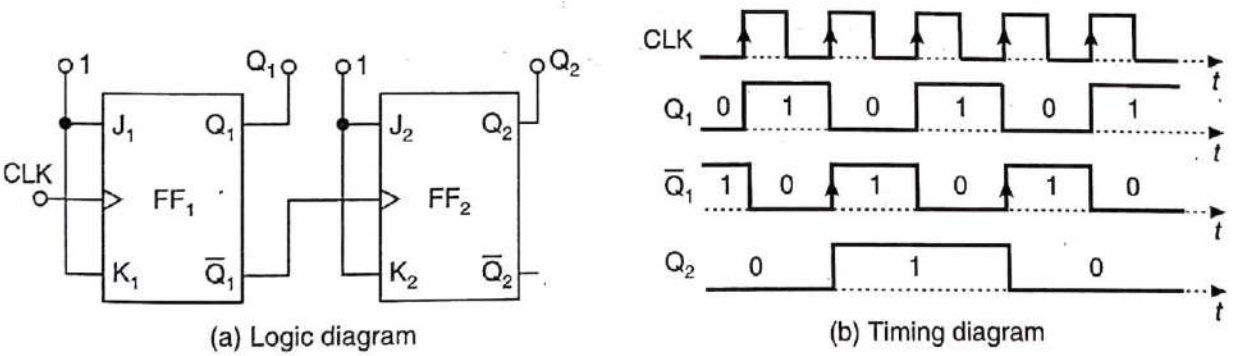
Clock signal to $FF_2 = (Q_1 \cdot Up) + (\bar{Q}_1 \cdot Down) = Q_1M + \bar{Q}_1\bar{M}$



Asynchronous 2-bit up-down counter using negative edge-triggered flip-flops.

ASYNCHRONOUS COUNTERS

Two-bit Ripple Up-counter Using Positive Edge-triggered FF

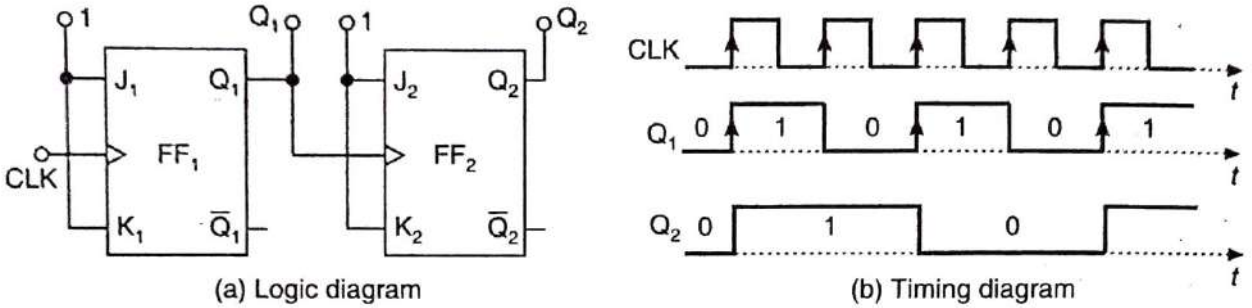


(a) Logic diagram (b) Timing diagram

Figure 12.4 Asynchronous 2-bit up-counter using positive-edge triggered J-K flip-flops.

ASYNCHRONOUS COUNTERS

Two-bit Ripple Down-counter Using Positive Edge-triggered FF



(a) Logic diagram (b) Timing diagram

Figure 12.5 Asynchronous 2-bit down-counter using positive edge-triggered J-K flip-flops.

Two-bit Ripple Up/Down Counter Using Positive Edge-triggered Flip-Flops

Clock signal to $FF_2 = (\overline{Q}_1 \cdot \text{Up}) + (Q_1 \cdot \text{Down}) = \overline{Q}_1 M + Q_1 \overline{M}$

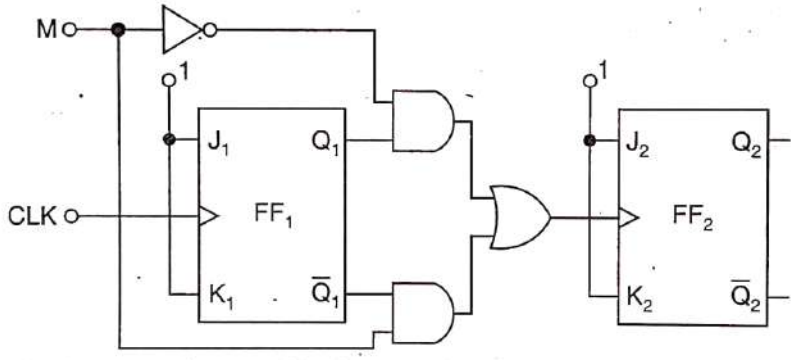
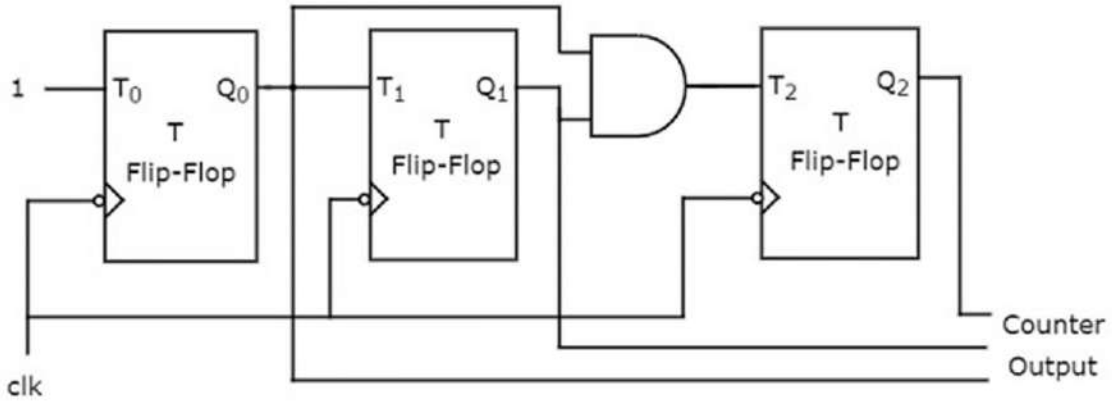


Figure 12.6 Logic diagram of a two-bit ripple up/down counter using positive edge-triggered flip-flops.

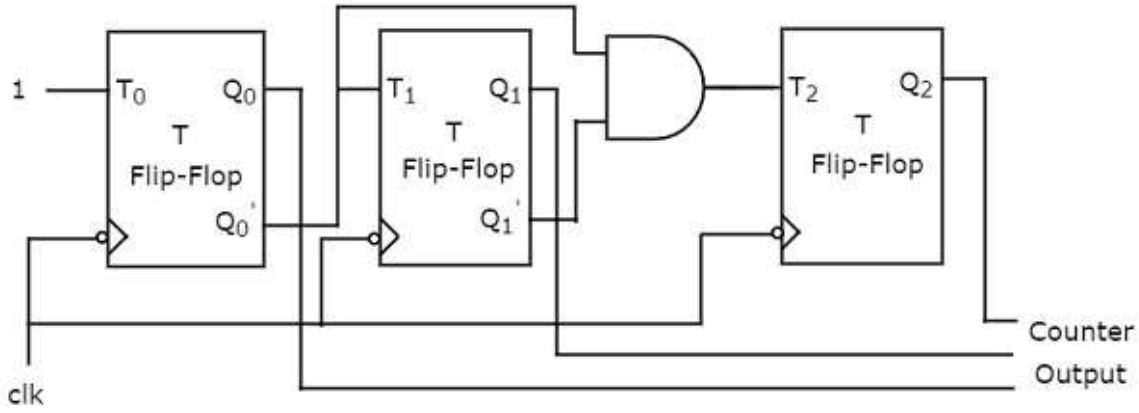
SYNCHRONOUS COUNTERS

3-bit Synchronous Up-counter Using Negative Edge-triggered FF



SYNCHRONOUS COUNTERS

3-bit Synchronous Down-counter Using Negative Edge-triggered FF



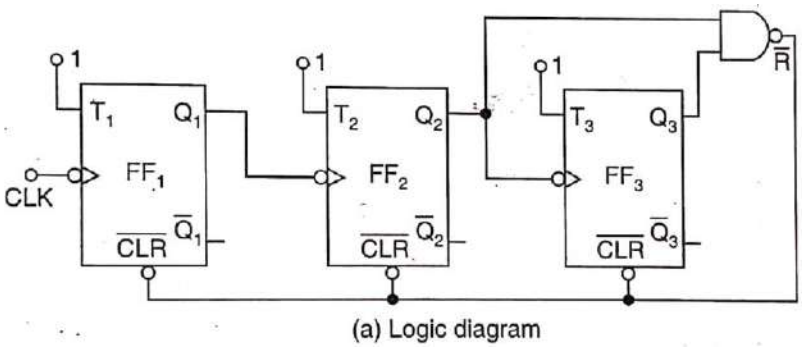
DESIGN OF ASYNCHRONOUS COUNTERS

Design of a Mod-6 Asynchronous Counter Using T FFs

A mod-6 counter has six stable states 000, 001, 010, 011, 100, and 101. When the sixth clock pulse is applied, the counter temporarily goes to 110 state, but immediately resets to 000 because of the feedback provided.

It is a 'divide-by-6 counter', in the sense that it divides the input clock k frequency by 6. It requires three FFs, because the smallest value of n satisfying the condition N < 2^n is n = 3; three FFs can have eight possible states, out of which only six are utilized and the remaining two states 110 and 111, are invalid.

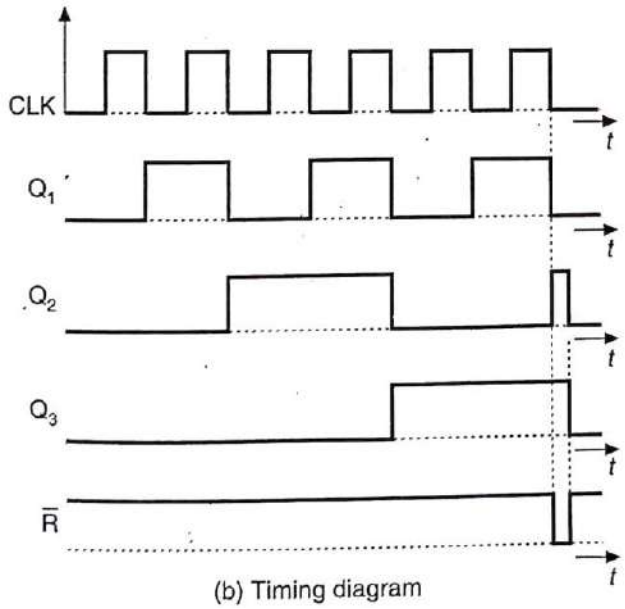
Design of a Mod-6 Asynchronous Counter Using T-FFs



(a) Logic diagram

After pulses	State Q_3	Q_2	Q_1	R
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	0
5	1	0	1	0
6	1	1	0	1
	↓	↓	↓	
	0	0	0	0
7	0	0	1	0

Design of a Mod-6 Asynchronous Counter Using T-FFs



(b) Timing diagram

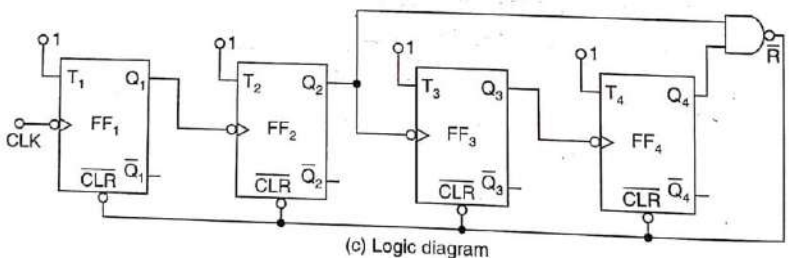
After pulses	State Q_3	Q_2	Q_1	R
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	0
5	1	0	1	0
6	1	1	0	1
	↓	↓	↓	
	0	0	0	0
7	0	0	1	0

Design of a Mod-10 Asynchronous Counter Using T-FFs

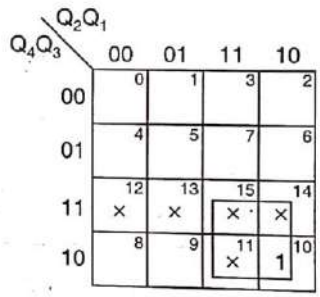
After pulses	Count			
	Q_4	Q_3	Q_2	Q_1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	0	0	0	0

(a) Count table

Design of a Mod-10 Asynchronous Counter Using T-FFs



(c) Logic diagram



(b) K-Map

After pulses	Count			
	Q_4	Q_3	Q_2	Q_1
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
10	0	0	0	0

(a) Count table

Example

EXAMPLE 12.1 Implement a 3-bit ripple counter using D FFs.

Solution

For ripple counters, the FFs used must be in toggle mode. The D FFs may be used in toggle mode by connecting the $\overline{Q}$ of each FF to its D terminal. The 3-bit ripple counter using D FFs is shown in Figure 12.11.

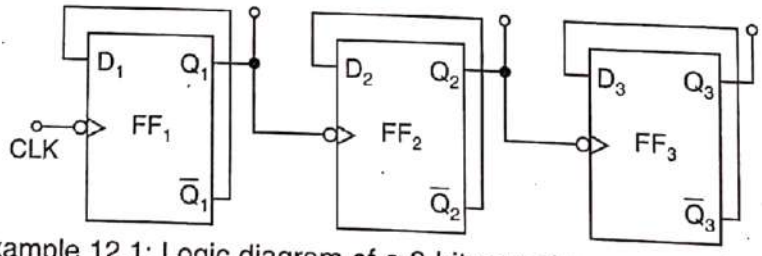
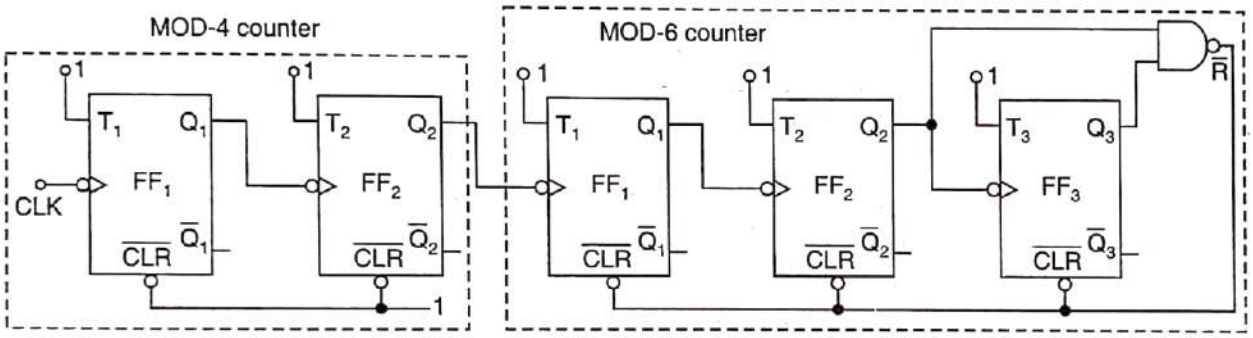
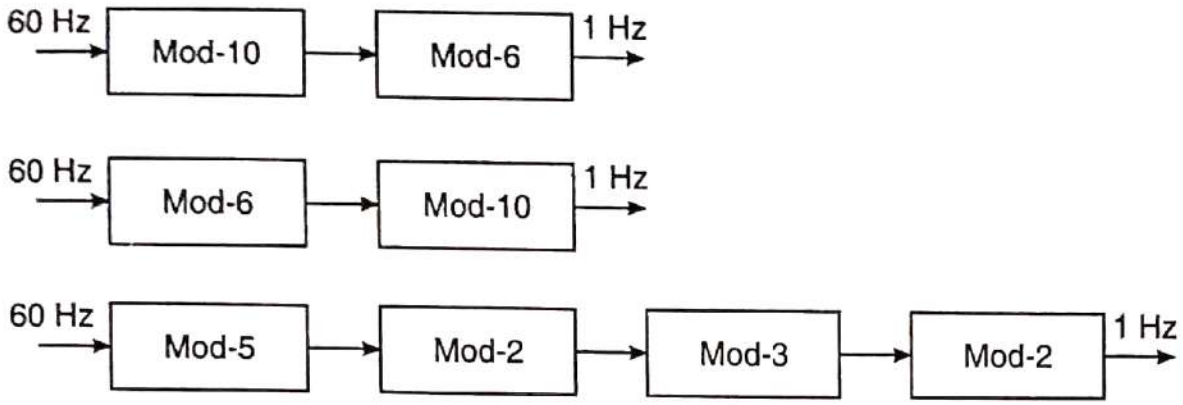


Figure 12.11 Example 12.1: Logic diagram of a 3-bit asynchronous counter using D flip-flops.

Cascading of Ripple Counters



Ripple Counters as Frequency Divider



Ripple Counters as Frequency Divider

EXAMPLE 12.2 A binary ripple counter is required to count up to $16{,}383_{10}$. How many FFs are required? If the clock frequency is 8.192 MHz, what is the frequency at the output of the MSB?

Solution

The number of FFs n is to be selected such that the number of states $N \leq 2^n$. With n FFs, the largest count possible is $2^n - 1$. Therefore,

$$2^n - 1 = 16{,}383$$

or

$$n = \log_2 16{,}384 = 14$$

So, the number of FFs required is 14.

Frequency at the output of last stage is

$$f_{14} = \frac{f_C}{2^{14}} = \frac{8.192\ \text{MHz}}{16{,}384} = 500\ \text{Hz}$$

Design of Synchronous Counters

Step 1. Number of flip-flops

Step 2. State diagram

Step 3. Choice of flip flops and excitation table

Step 4. Minimal expressions for excitations

Step 5. Logic diagram

Excitation tables

PS	NS	Required inputs	
Q_n	Q_{n+1}	S	R
0	0	0	×
0	1	1	0
1	0	0	1
1	1	×	0

(a) S-R FF excitation table

PS	NS	Required inputs	
Q_n	Q_{n+1}	J	K
0	0	0	×
0	1	1	×
1	0	×	1
1	1	×	0

(b) J-K FF excitation table

PS	NS	Required input
Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

(c) D FF excitation table

PS	NS	Required input
Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

(d) T FF excitation table

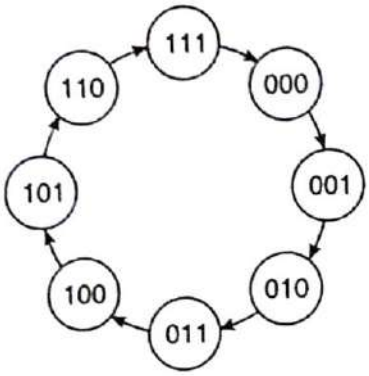
Design of a Synchronous 3-bit Up Counter Using J-K FFs

Step 1. Determine the number of flip-flops required

Step 2. Draw the state diagram

Design of a Synchronous 3-bit Up Counter Using J-K FFs

Step 3. Select the type of flip-flops and draw the excitation table:



(a) State diagram

PS			NS			Required excitations					
Q_3	Q_2	Q_1	Q_3	Q_2	Q_1	J_3	K_3	J_2	K_2	J_1	K_1
0	0	0	0	0	1	0	×	0	×	1	×
0	0	1	0	1	0	0	×	1	×	×	1
0	1	0	0	1	1	0	×	×	0	1	×
0	1	1	1	0	0	1	×	×	1	×	1
1	0	0	1	0	1	×	0	0	×	1	×
1	0	1	1	1	0	×	0	1	×	×	1
1	1	0	1	1	1	×	0	×	0	1	×
1	1	1	0	0	0	×	1	×	1	×	1

(b) Excitation table

Design of a Synchronous 3-bit Up Counter Using J-K FFs

Step 4. Obtain the minimal expressions

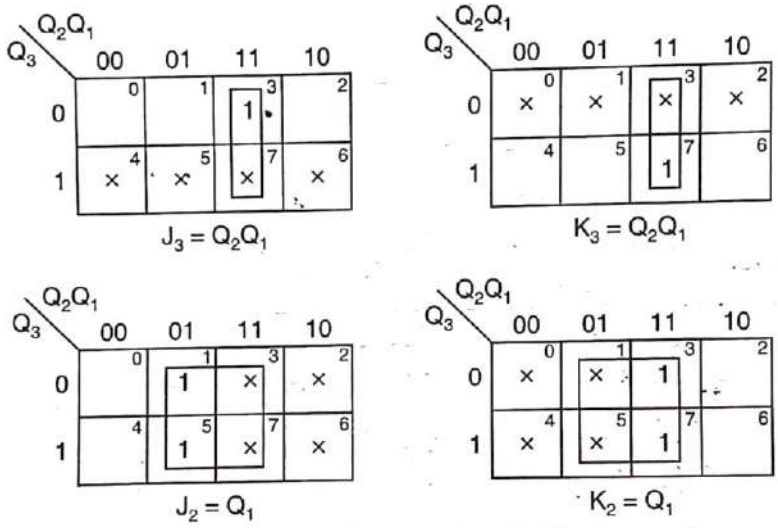


Figure 12.17 Karnaugh maps for a 3-bit up-counter.

Design of a Synchronous 3-bit Up Counter Using J-K FFs

Step 5. Logic diagram

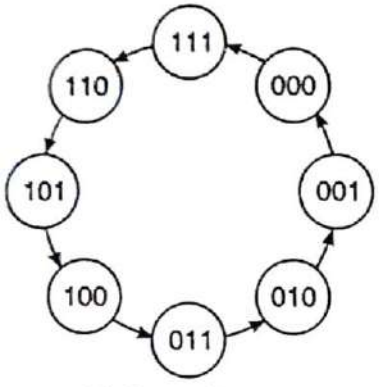
Design of a Sync. 3-bit Down Counter Using J-K FFs

Step 1. Determine the number of flip-flops required

Step 2. Draw the state diagram

Design of a Sync. 3-bit Down Counter Using J-K FFs

Step 3. Select the type of flip-flops and draw the excitation table:



(a) State diagram

PS			NS			Required excitations					
Q_3	Q_2	Q_1	Q_3	Q_2	Q_1	J_3	K_3	J_2	K_2	J_1	K_1
0	0	0	1	1	1	1	×	1	×	1	×
1	1	1	1	1	0	×	0	×	0	×	1
1	1	0	1	0	1	×	0	×	1	1	×
1	0	1	1	0	0	×	0	0	×	×	1
1	0	0	0	1	1	×	1	1	×	1	×
0	1	1	0	1	0	0	×	×	0	×	1
0	1	0	0	0	1	0	×	×	1	1	×
0	0	1	0	0	0	0	×	0	×	×	1

(b) Excitation table

Design of a Sync. 3-bit Down Counter Using J-K FFs

Step 4. Obtain the minimal expressions

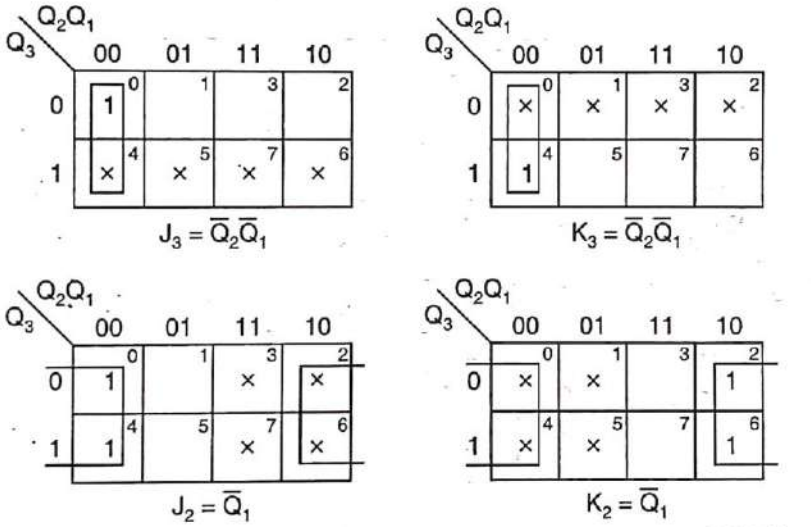
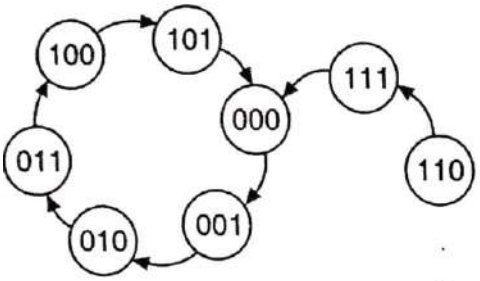


Figure 12.19 K-maps for a three-bit down counter using J-K FFs.

Design of a Sync. 3-bit Down Counter Using J-K FFs

Step 5. Logic diagram

Design of a Synchronous Mod-6 Counter Using J-K FFs



(a) State diagram

PS			NS			Required excitations					
Q_3	Q_2	Q_1	Q_3	Q_2	Q_1	J_3	K_3	J_2	K_2	J_1	K_1
0	0	0	0	0	1	0	×	0	×	1	×
0	0	1	0	1	0	0	×	1	×	×	1
0	1	0	0	1	1	0	×	×	0	1	×
0	1	1	1	0	0	1	×	×	1	×	1
1	0	0	1	0	1	×	0	0	×	1	×
1	0	1	0	0	0	×	1	0	×	×	1

(b) Excitation table

Figure 12.36 Synchronous mod-6 counter using J-K flip-flops.

Design of a Synchronous Mod-6 Counter Using J-K FFs

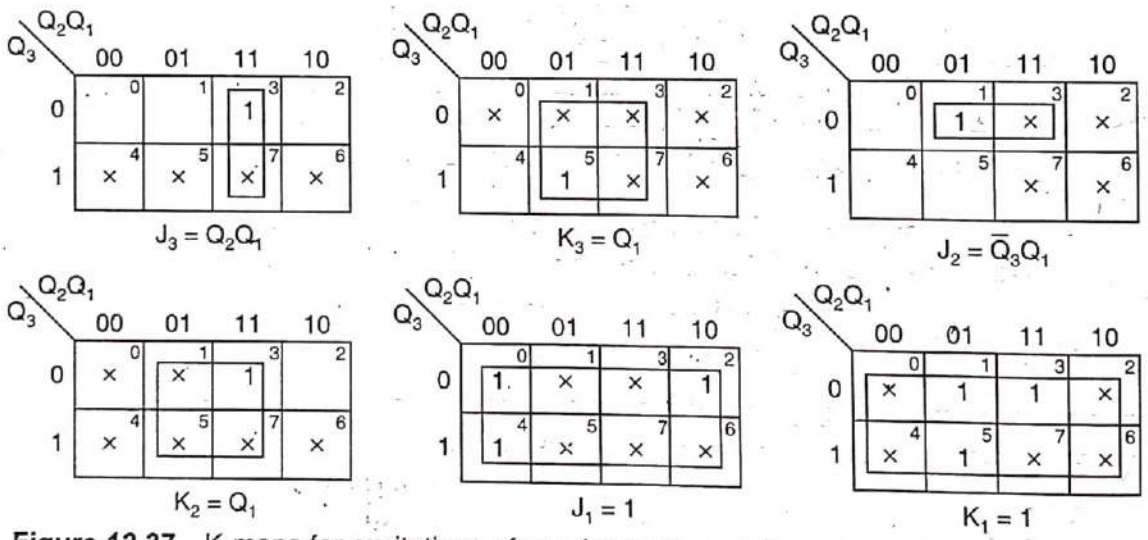
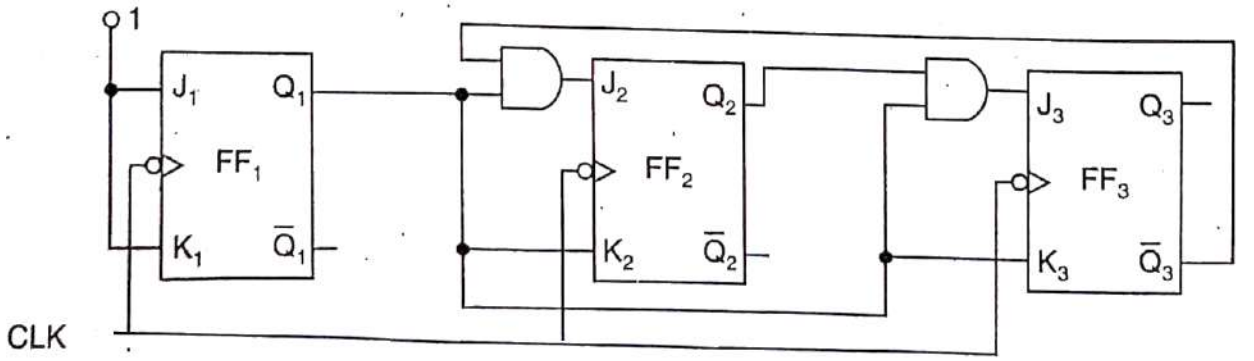
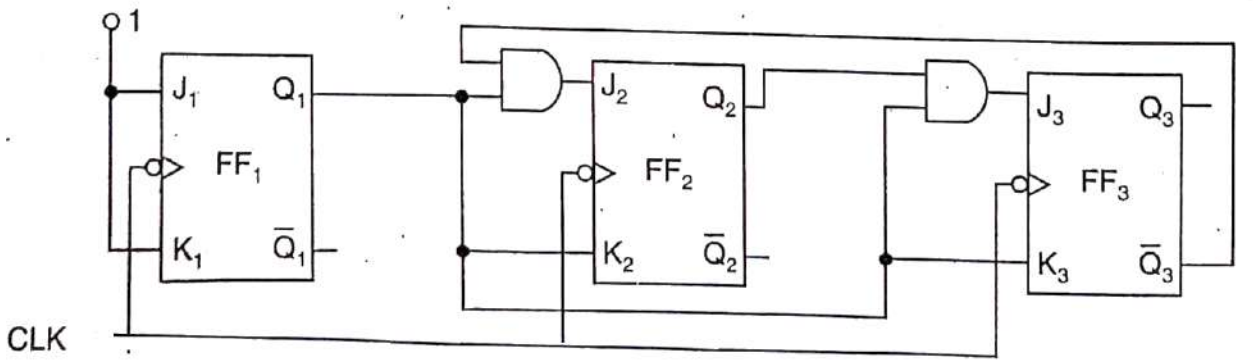


Figure 12.37 K-maps for excitations of synchronous mod-6 counter using J-K flip-flops.

Design of a Synchronous Mod-6 Counter Using J-K FFs



Design of a Synchronous Mod-6 Counter Using J-K FFs



Design of a Synchronous Mod-6 Counter Using J-K FFs

Checking for Self Starting and Lock-Out

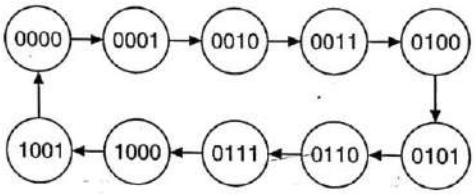
Table 12.3 Check for lock-out

PS			Present inputs						NS		
Q_3	Q_2	Q_1	J_3	K_3	J_2	K_2	J_1	K_1	Q_3	Q_2	Q_1
1	1	0	0	0	0	0	1	1	1	1	1
1	1	1	1	1	0	1	1	1	0	0	0

Design of a Synchronous BCD Counter Using J-K FFs

PS				NS				Required excitations							
Q_4	Q_3	Q_2	Q_1	Q_4	Q_3	Q_2	Q_1	J_4	K_4	J_3	K_3	J_2	K_2	J_1	K_1

Design of a Synchronous BCD Counter Using J-K FFs



(a) State diagram

PS				NS				Required excitations							
Q_4	Q_3	Q_2	Q_1	Q_4	Q_3	Q_2	Q_1	J_4	K_4	J_3	K_3	J_2	K_2	J_1	K_1
0	0	0	0	0	0	0	1	0	×	0	×	0	×	1	×
0	0	0	1	0	0	1	0	0	×	0	×	1	×	×	1
0	0	1	0	0	0	1	1	0	×	0	×	×	0	1	×
0	0	1	1	0	1	0	0	0	×	1	×	×	1	×	1
0	1	0	0	0	1	0	1	0	×	×	0	0	×	1	×
0	1	0	1	0	1	1	0	0	×	×	0	1	×	×	1
0	1	1	0	0	1	1	1	0	×	×	0	×	0	1	×
0	1	1	1	1	0	0	0	1	×	×	1	×	1	×	1
1	0	0	0	1	0	0	1	×	0	0	×	0	×	1	×
1	0	0	1	0	0	0	0	×	1	0	×	0	×	×	1

(b) Excitation table

Design of a Synchronous BCD Counter Using J-K FFs

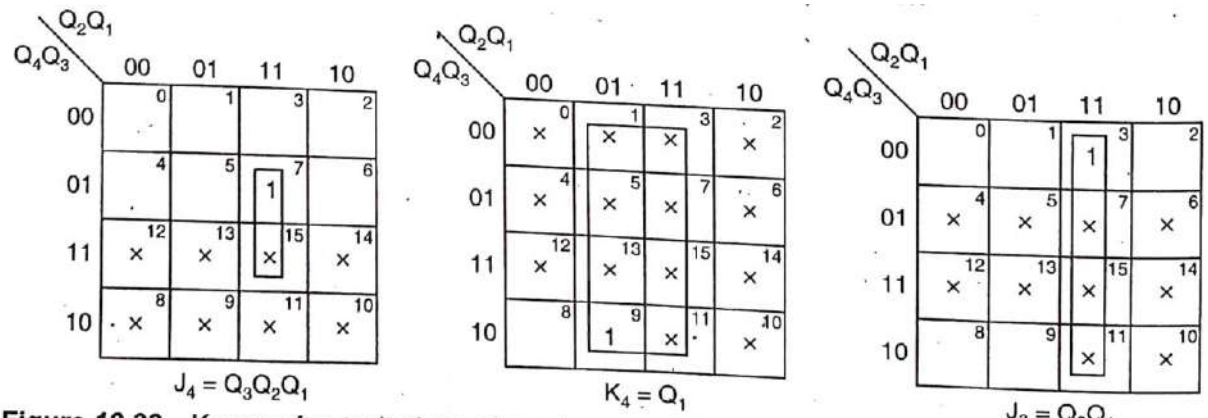


Figure 12.33 K-maps for excitations of synchronous BCD counter using J-K flip-flops *(Contd.)...*

Design of a Synchronous BCD Counter Using J-K FFs

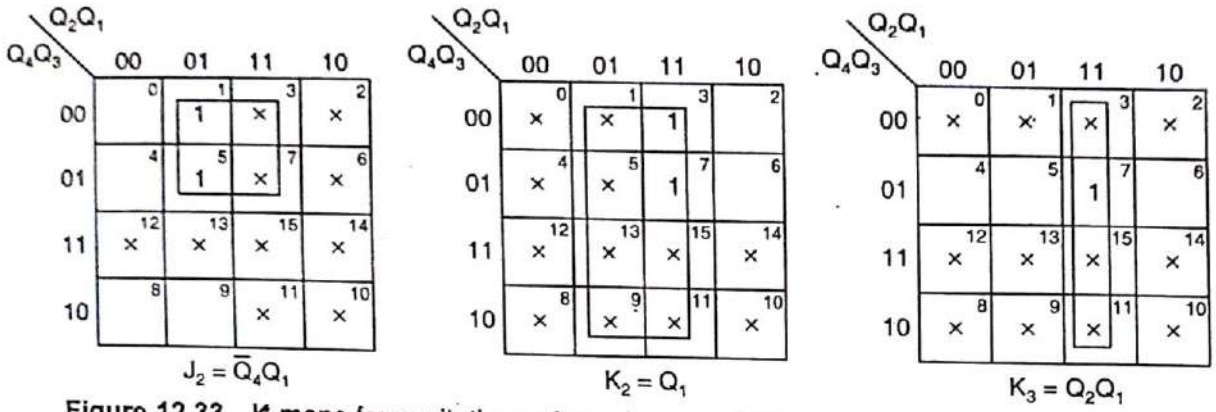


Figure 12.33 K-maps for excitations of synchronous BCD counter using J-K flip-flops.

Design of a Synchronous BCD Counter Using J-K FFs

Design of a Synchronous BCD Counter Using J-K FFs

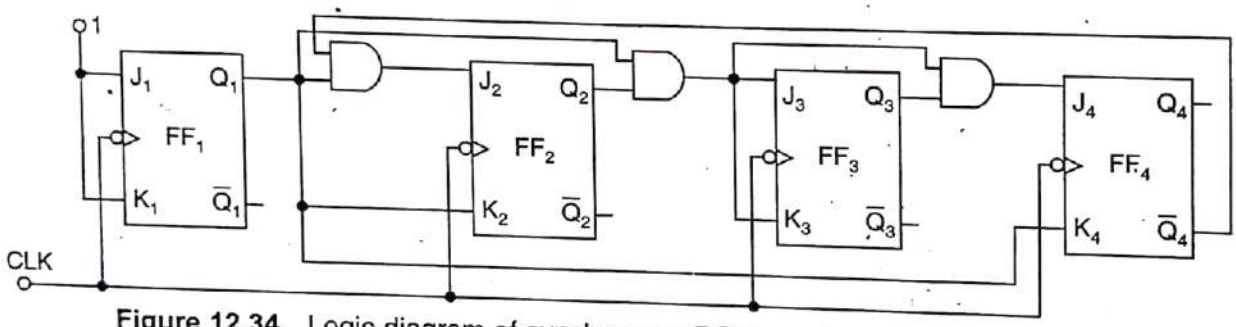
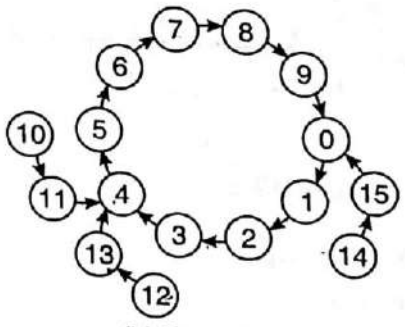


Figure 12.34 Logic diagram of synchronous BCD counter using J-K flip-flops.

Checking for lock out of Synchronous BCD counter using J-K flip-flops.

PS				Present inputs							NS			
Q_4	Q_3	Q_2	Q_1	J_4	K_4	J_3	K_3	J_2	K_2	J_1	K_1	Q_4	Q_3	Q_2	Q_1
1	0	1	0	0	0	0	0	0	0	1	1	1	0	1	1
1	0	1	1	0	1	1	1	0	1	1	1	0	1	0	0
1	1	0	0	0	0	0	0	0	0	1	1	1	1	0	1
1	1	0	1	0	1	0	0	0	1	1	1	0	1	0	0
1	1	1	0	0	0	0	0	0	0	1	1	1	1	1	1
1	1	1	1	1	1	1	1	0	1	1	1	0	0	0	0

(a) Table to check for lock-out



(b) State diagram

Figure 12.35 Checking for lock out of Synchronous BCD counter using J-K flip-flops.

ANALOG-TO-DIGITAL CONVERTERS (ADC) AND DIGITAL-TO-ANALOG CONVERTERS (DAC)

An analog quantity is one that can take on any value over a continuous range of values. It represents an exact value. Most physical variables are analog in nature. Temperature, pressure, light and sound intensity, position, rotation, speed, etc. are some examples of analog quantities.

A digital quantity takes on only discrete values. The value is expressed in a digital code such as a binary or BCD number.

Interfacing a digital computer to the analog world

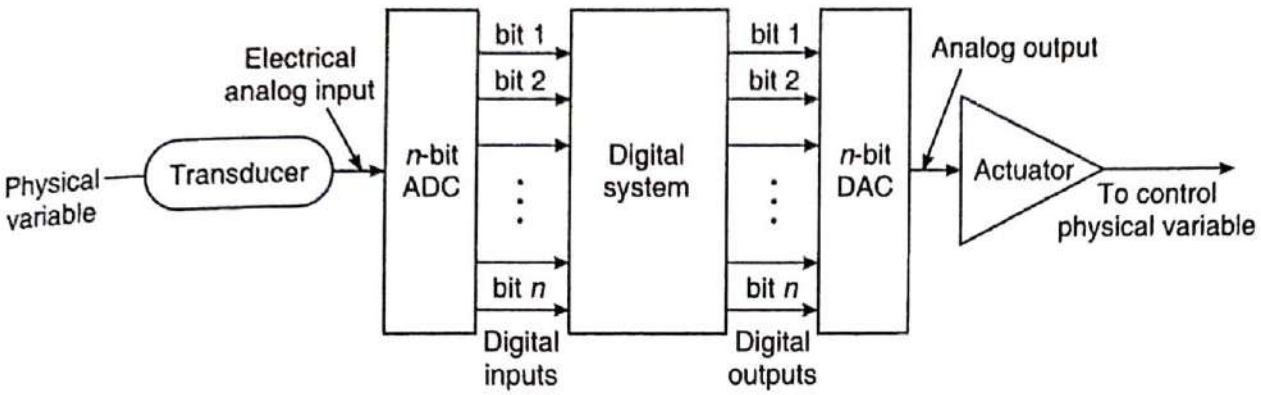


Figure 17.1 Interfacing a digital computer to the analog world.

DIGITAL-TO-ANALOG (D/A) CONVERSION

Basically, D/A conversion is the process of converting a value represented in digital code, such as straight binary or BCD, into a voltage or current which is proportional to the digital value.

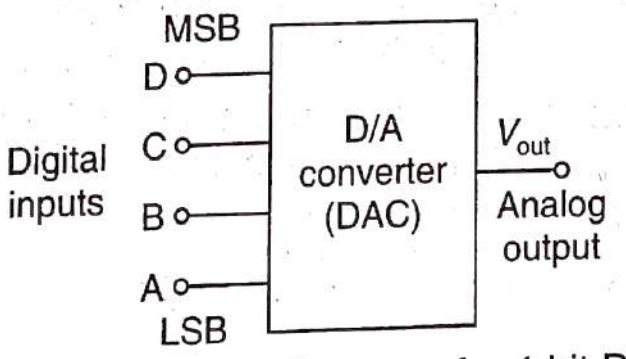


Figure 17.2 Block diagram of a 4-bit DAC.

DIGITAL-TO-ANALOG (D/A) CONVERSION

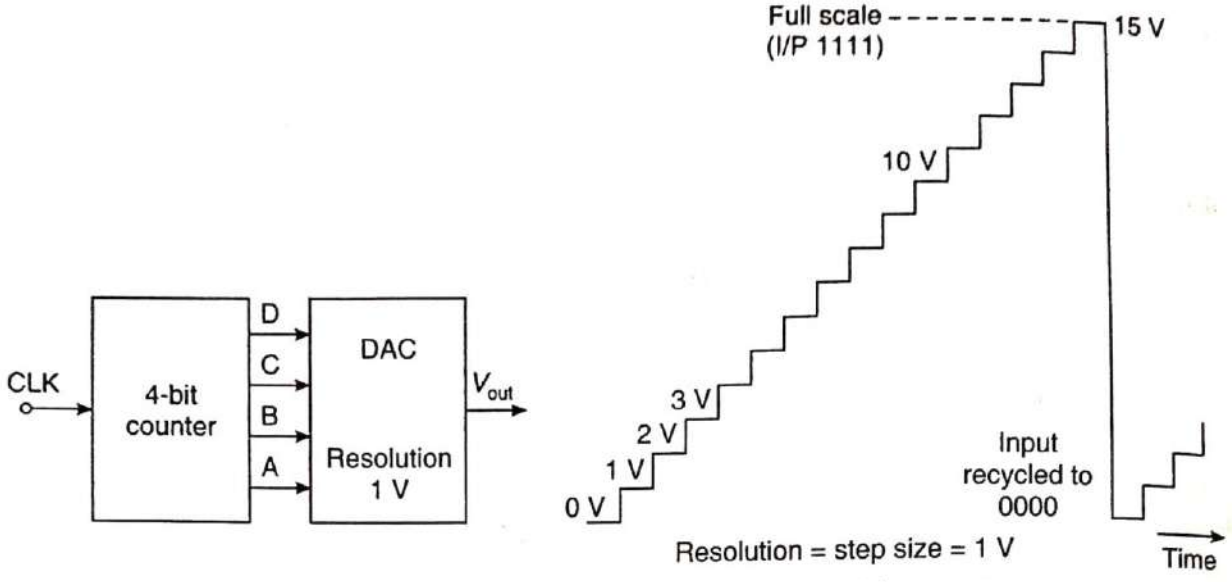


Figure 17.3 Output waveform of a DAC fed by a binary counter.

Parameters of DAC

1. Resolution (step size): The resolution of a DAC is defined as the smallest change that can occur in an analog output as a result of a change in the digital input.

$$\% \text{ resolution} = \frac{\text{step size}}{\text{full scale}} \times 100\%$$

Since, full-scale = number of steps × step size, resolution can be expressed as

$$\% \text{ resolution} = \frac{1}{\text{total number of steps}} \times 100\%$$

In general, for an N-bit DAC, the number of different levels will be 2N and the number of steps will be 2N - 1. The greater the number of bits, the greater will be the number of steps and the smaller will be the step size, and therefore, the finer will be the resolution. Of course, the cost of the DAC increases with the number of input bits.

Parameters of DAC

2. Accuracy: The accuracy of a DAC is usually specified in terms of its full-scale error and linearity error, which are normally expressed as a percentage of the converter's full-scale output.

3. Settling Time : The operating speed of a DAC is usually specified by giving its settling time.

4. Offset Voltage: Ideally, the output of a DAC should be zero when the binary input is zero. In practice however, there is a very small output voltage under this situation called the offset voltage.

5. Monotonicity: A DAC is said to be monotonic if its output increases as the binary input is incremented from one value to the next.

Question

EXAMPLE 17.1 Determine the resolution of (a) a 6-bit DAC and that of (b) a 12-bit DAC in terms of percentage.

Solution

(a) For the 6-bit DAC,

$$\% \text{ resolution} = \frac{1}{2^N - 1} \times 100 = \frac{1}{2^6 - 1} \times 100 = \frac{1}{63} \times 100 = 1.587\%$$

(b) For the 12-bit DAC,

$$\% \text{ resolution} = \frac{1}{2^N - 1} \times 100 = \frac{1}{2^{12} - 1} \times 100 = \frac{1}{4095} \times 100 = 0.0244\%$$

Question

EXAMPLE 17.2 A 6-bit DAC has a step size of 50 mV. Determine the full-scale output voltage and the percentage resolution.

Solution

With 6 bits, there will be $2^6 - 1 = 63$ steps of size 50 mV each.

The full-scale output will, therefore, be

$$63 \times 50 \text{ mV} = 3.15 \text{ V}$$

$$\% \text{ resolution} = \frac{50 \text{ mV}}{3.15 \text{ V}} \times 100 = \frac{1}{63} \times 100 = 1.587\%$$

Question

EXAMPLE 17.3 An 8-bit DAC produces $V_{out} = 0.05$ V for a digital input of 00000001. Find the full-scale output. What is the resolution? What is V_{out} for an input of 00101010?

Solution

$$\text{Full-scale output} = \text{Step size} \times \text{Number of steps}$$
$$= 0.05(2^8 - 1) = 0.05 \times 255 = 12.75 \text{ V}$$
$$\% \text{ resolution} = \frac{1}{2^N - 1} \times 100 = \frac{1}{255} \times 100 = 0.392\%$$

V_{out} for an input of 00101010 $= 42 \times 0.05 = 2.10$ V

Question

EXAMPLE 17.6 A certain 12-bit BCD DAC has a full-scale output of 19.98 V. Determine (a) the percentage resolution and (b) the converter's step size.

Solution

(a) 12 BCD bits correspond to three decimal digits, i.e. decimal numbers from 000 to 999. Therefore, the output of the DAC has 999 possible steps from 0 V to 19.98 V. Thus, we have

$$\% \text{ resolution} = \frac{1}{\text{number of steps}} \times 100\% = \frac{1}{999} \times 100\% \approx 0.1\%$$

(b)
$$\text{Step size} = \frac{\text{full-scale output}}{\text{number of steps}} = \frac{19.98 \text{ V}}{999} = 0.02 \text{ V}$$

Types of DAC

1. THE R-2R LADDER TYPE DAC

2. THE WEIGHTED-RESISTOR TYPE DAC

1. THE R-2R LADDER TYPE DAC

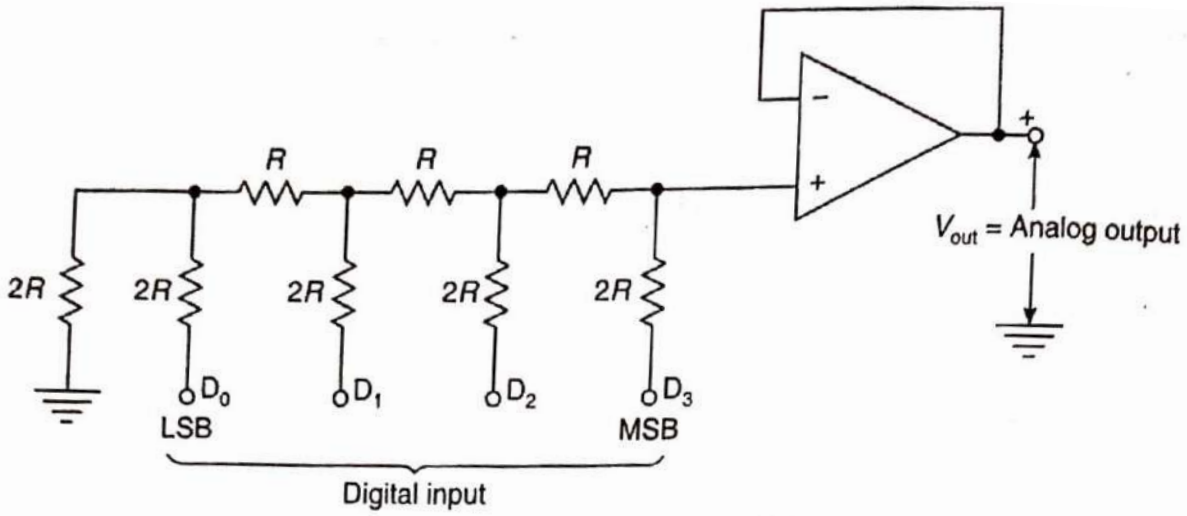


Figure 17.5 *R-2R* ladder type DAC.

Case 1: When the input is 1000

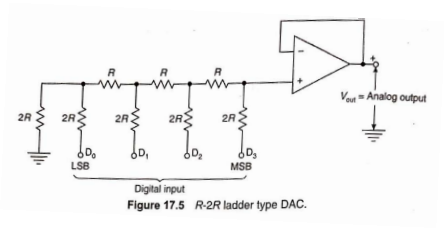


Figure 17.5 *R-2R* ladder type DAC.

Case 1: When the input is 1000

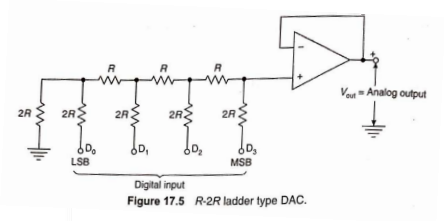
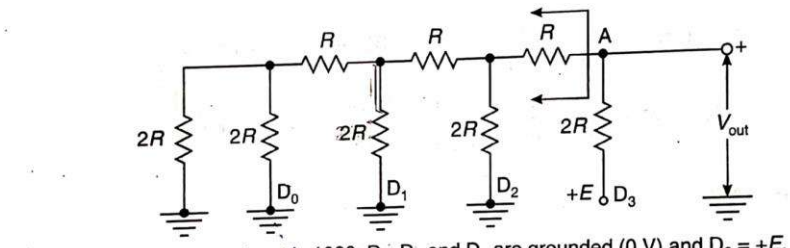
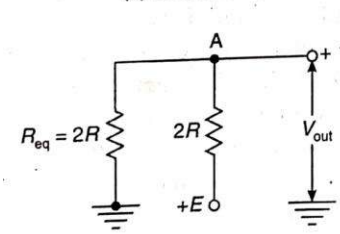


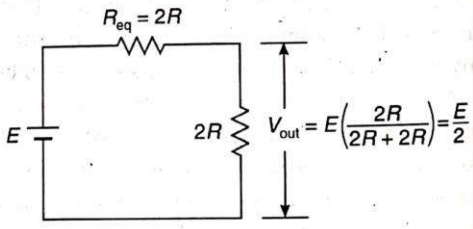
Figure 17.5 *R-2R* ladder type DAC.



(a) When the input is 1000; D_0, D_1 and D_2 are grounded (0 V) and $D_3 = +E$.



(b) The circuit equivalent to (a) when the circuit to the left of A is replaced by its equivalent resistance R_{eq}.



(c) Calculation of V_{out} using the voltage divider rule.

Figure 17.6 Calculation of V_{out} when the input is 1000.

Case 2: When the input is 0100

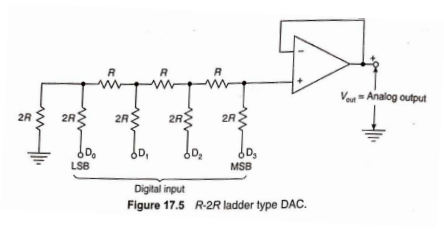


Figure 17.5 *R-2R* ladder type DAC.

Case 2: When the input is 0100

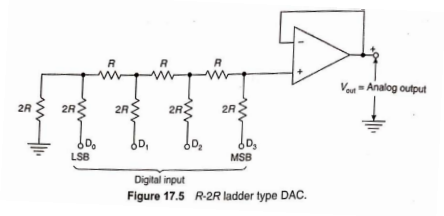
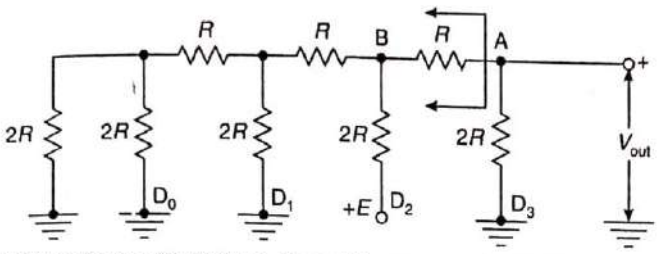
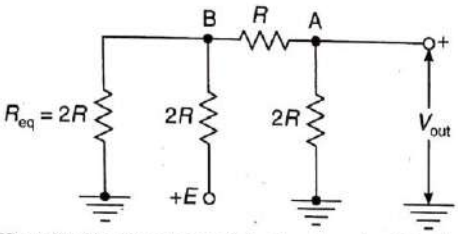


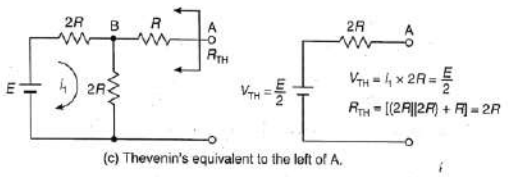
Figure 17.5 *R-2R* ladder type DAC.



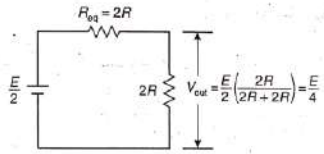
(a) When the input is 0100; D_0, D_1 and D_2 are grounded (0 V) and $D_2 = +E$.



(b) The circuit equivalent to (a) when the circuit to the left of B is replaced by its equivalent resistance R_{eq}.



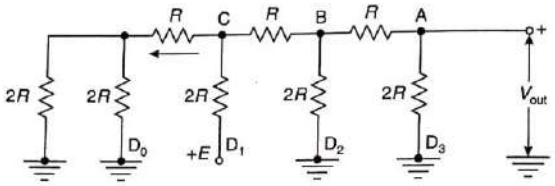
(c) Thevenin's equivalent to the left of A.



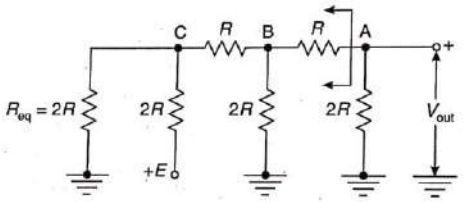
(d) Calculation of V_{out} using the voltage divider rule.

Figure 17.7 Calculation of V_{out} when the input is 0100.

Case 3: When the input is 0010



(a) When the input is 0010; D_0, D_2 and D_3 are grounded (0 V) and $D_1 = +E$.



(b) The circuit equivalent to (a) when the circuit to the left of C is replaced by $R_{eq} = 2R$.

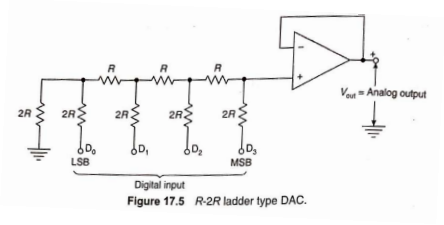
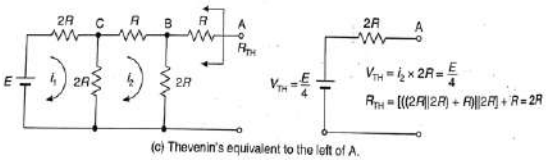
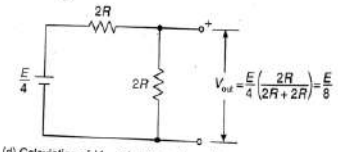


Figure 17.5 *R-2R* ladder type DAC.



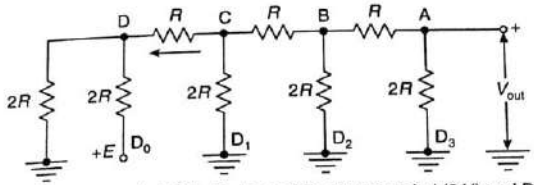
(c) Thevenin's equivalent to the left of A.



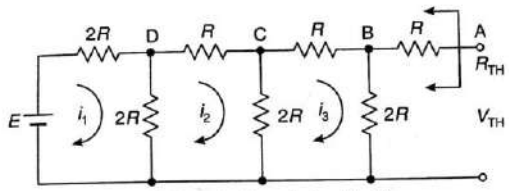
(d) Calculation of V_{out} using the voltage divider rule.

Figure 17.8 Calculation of V_{out} when the input is 0010.

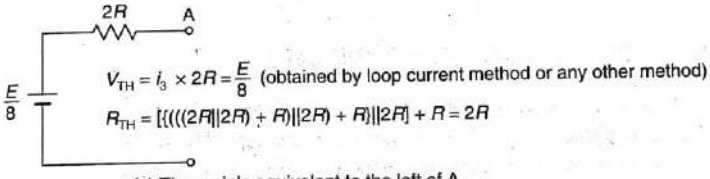
Case 4: When the input is 0001



(a) When the input is 0001; D_1, D_2 and D_3 are grounded (0 V) and $D_0 = +E$.



(b) The circuit equivalent to (a).



(c) Thevenin's equivalent to the left of A.

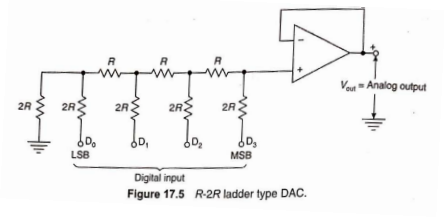
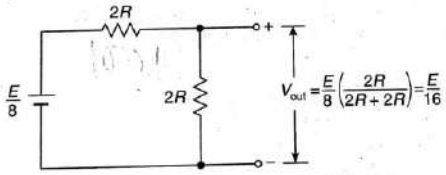


Figure 17.5 *R-2R* ladder type DAC.



(d) Calculation of V_{out} using the voltage divider rule.

Figure 17.9 Calculation of V_{out} when the input is 0001.

Question

Q. What are the output voltages caused by logic 1 in each bit position in an 8-bit ladder if the input level for 0 is 0V and that for 1 is +10V?

Q. What is the resolution of a 9-bit DAC, which uses a ladder network? What is this resolution expressed as a percentage? If the full-scale output voltage of this converter is + 5 V, what is the resolution in volts?

2. THE WEIGHTED-RESISTOR TYPE DAC

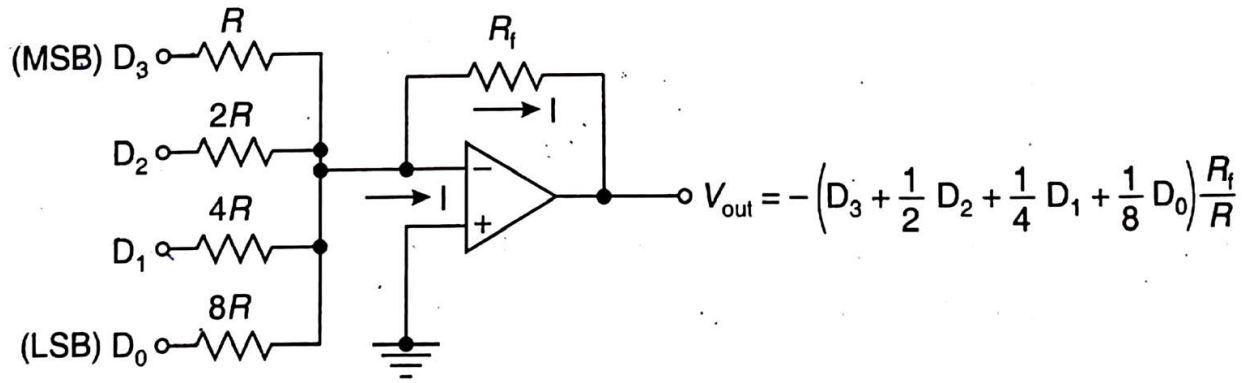
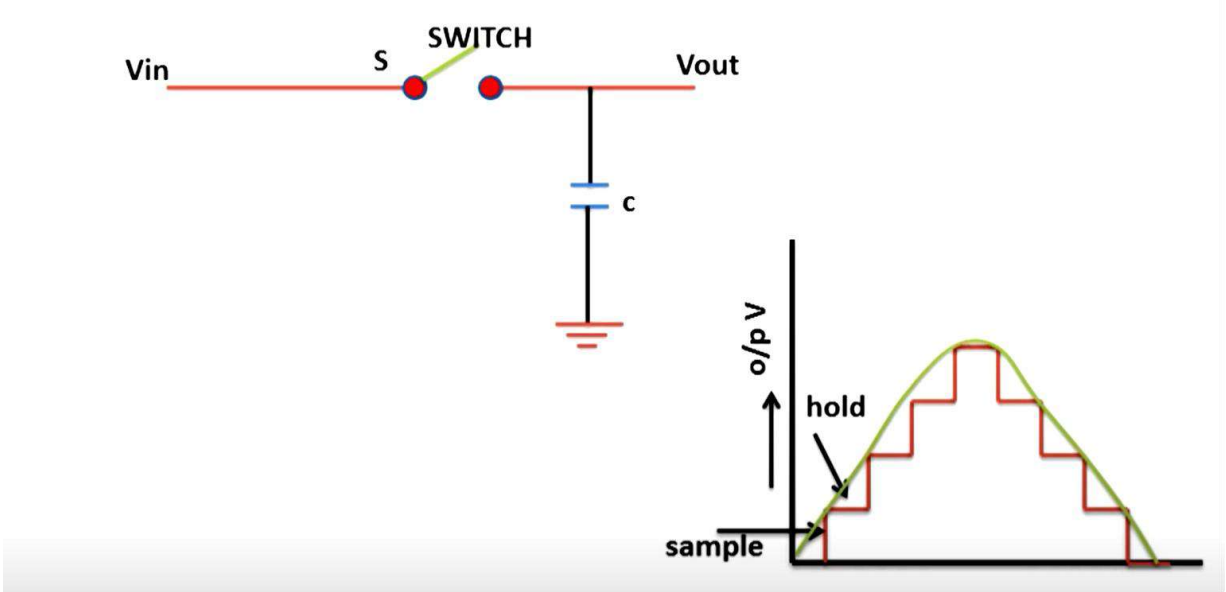


Figure 17.10 Weighted-resistor type DAC.

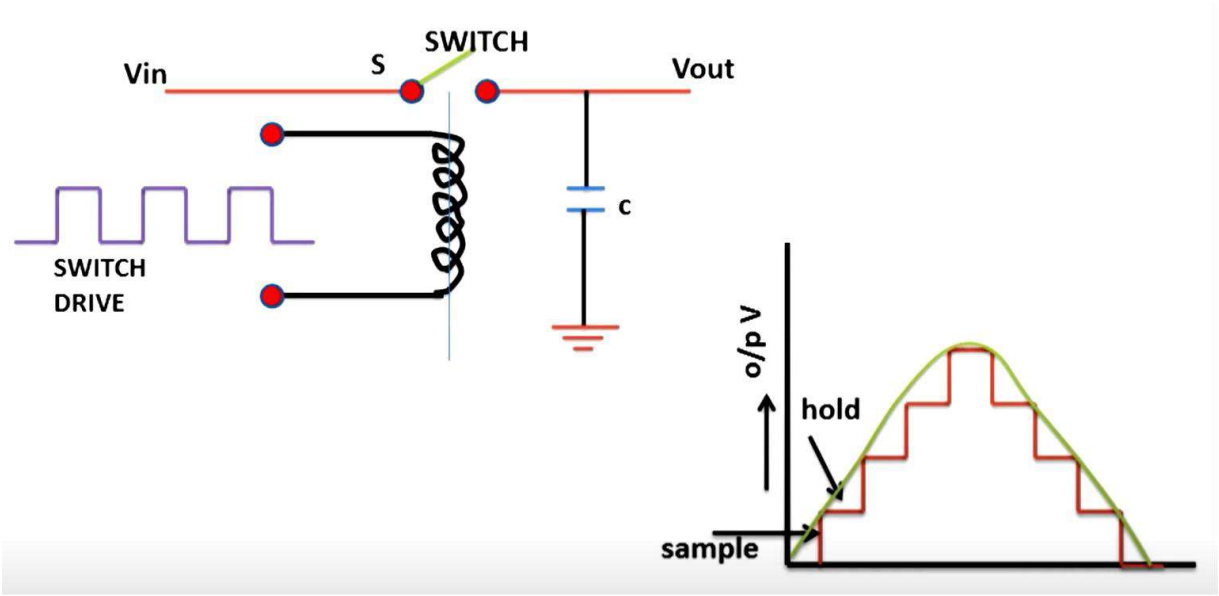
ANALOG-TO-DIGITAL (A/D) CONVERSION

1. THE COUNTER-TYPE A/D CONVERTER
2. THE TRACKING-TYPE A/D CONVERTER
3. THE DUAL-SLOPE TYPE A/D CONVERTER
4. THE SUCCESSIVE-APPROXIMATION TYPE ADC

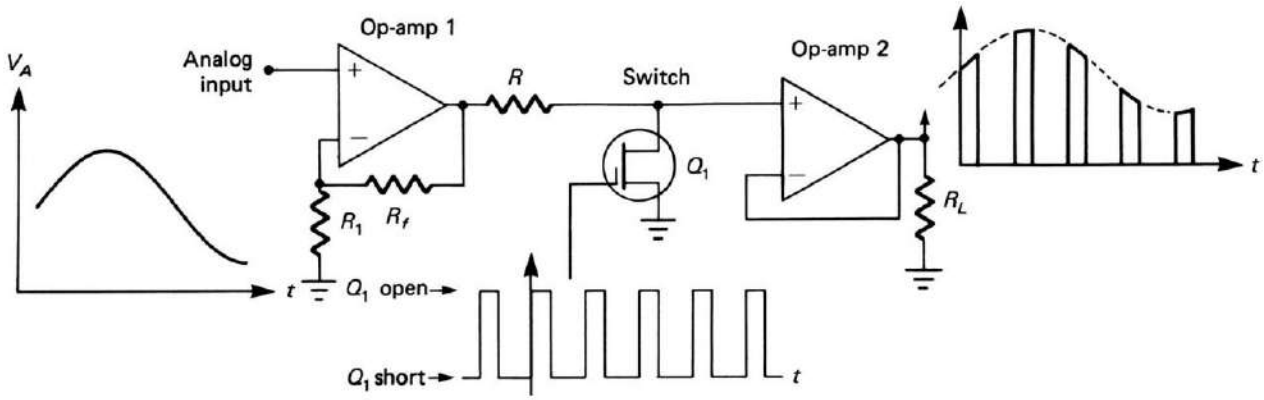
Sample and Hold Circuit



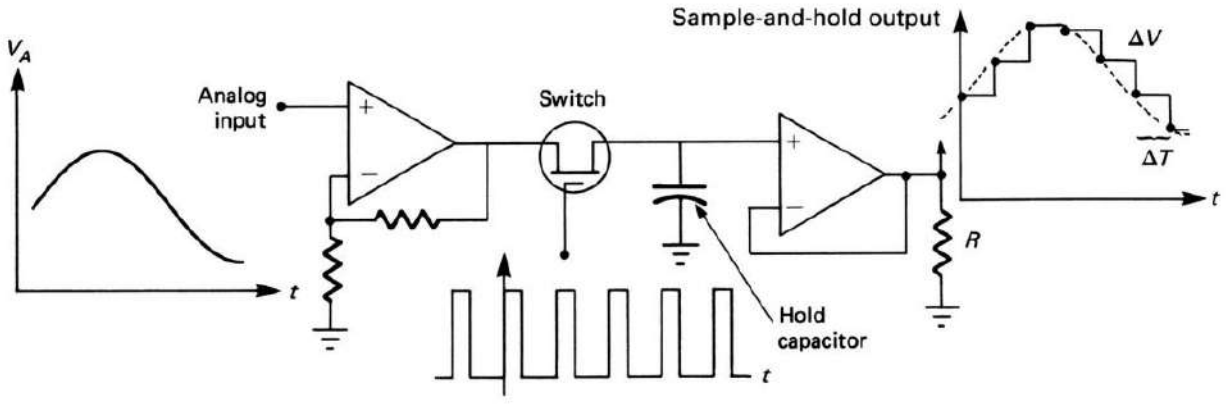
Sample and Hold Circuit



Natural Sampling



Flat Top Sampling



1. THE COUNTER-TYPE A/D CONVERTER

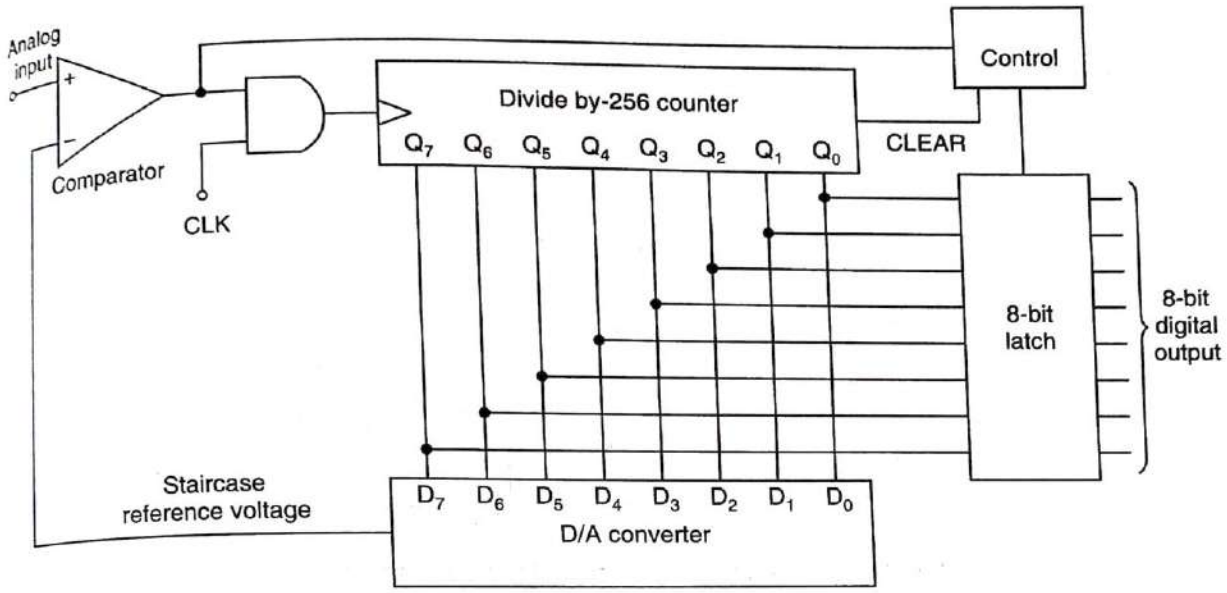


Figure 17.15 Logic diagram of the counter-type ADC.

1. THE COUNTER-TYPE A/D CONVERTER

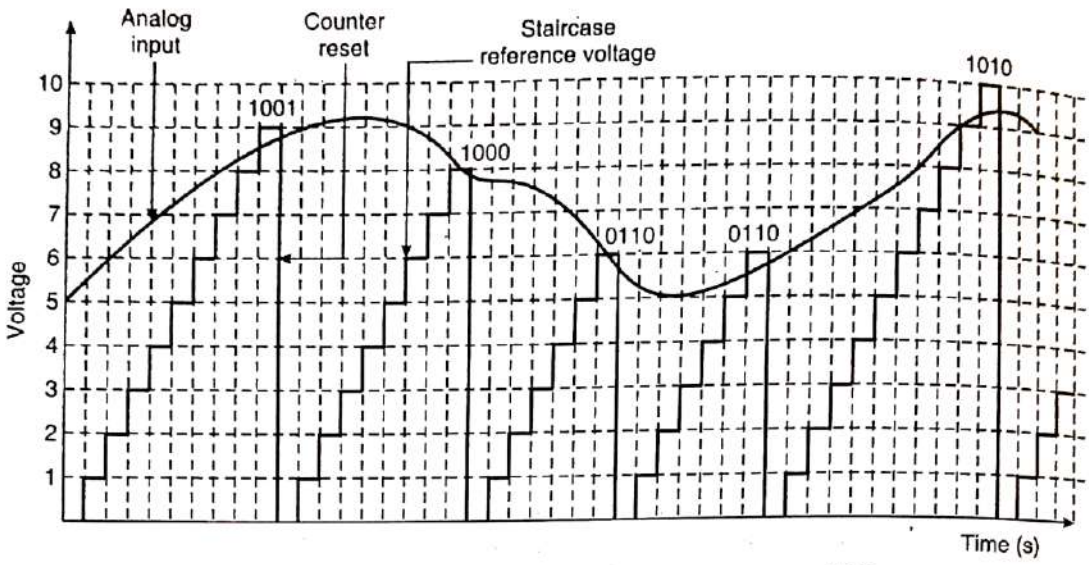


Figure 17.16 Output waveform of the counter-type ADC.

2. THE TRACKING-TYPE A/D CONVERTER

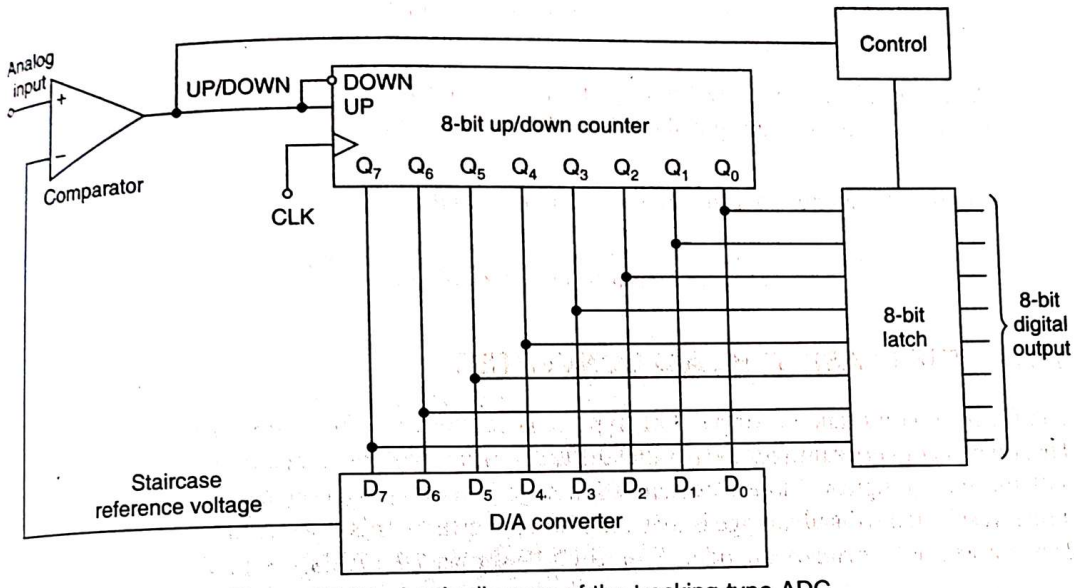


Figure 17.17 Logic diagram of the tracking-type ADC.

2. THE TRACKING-TYPE A/D CONVERTER

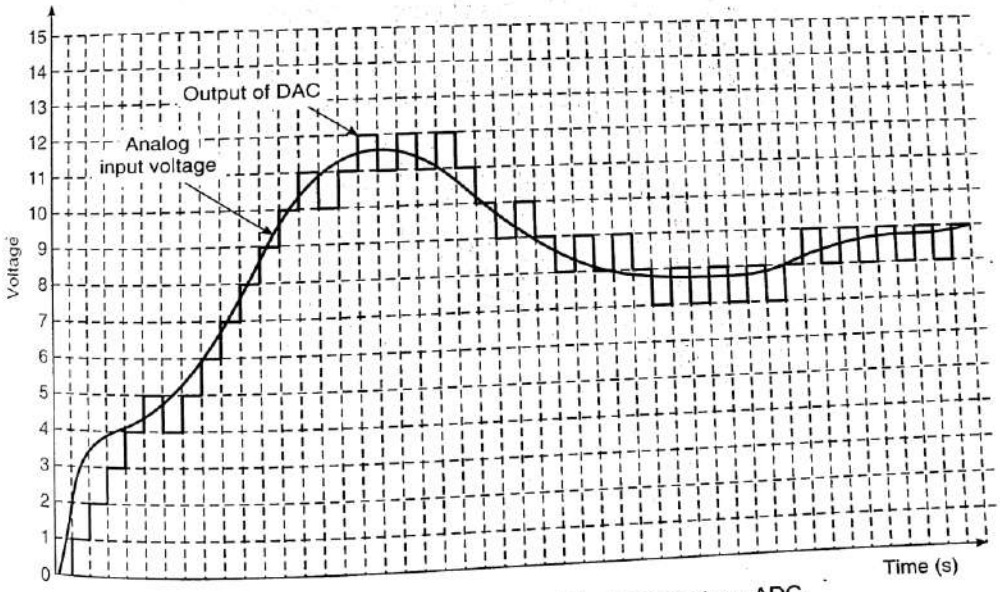


Figure 17.18 Output waveform of the tracking-type ADC.

3. THE DUAL-SLOPE TYPE A/D CONVERTER

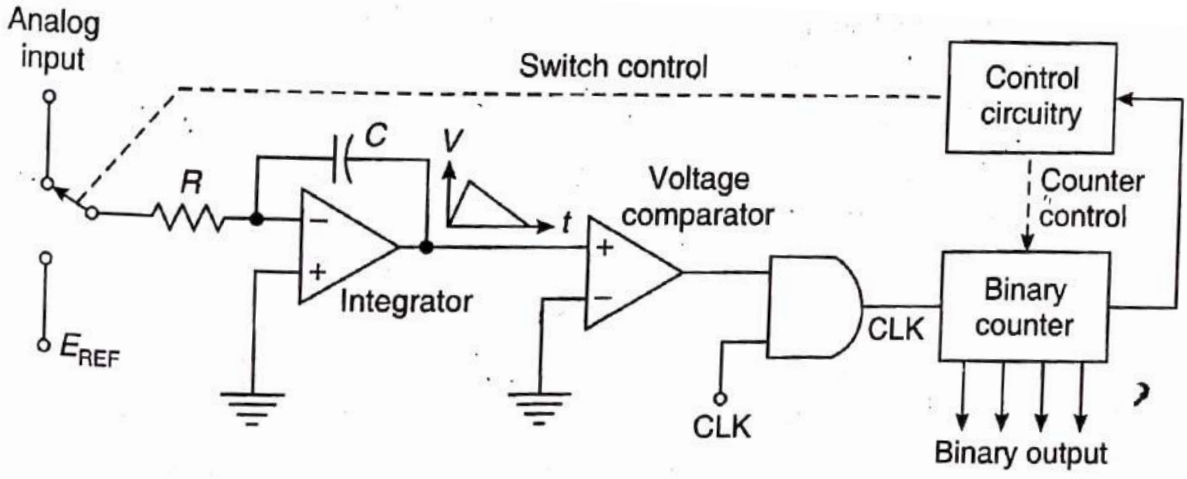


Figure 17.23 The dual-slope ADC.

4. THE SUCCESSIVE-APPROXIMATION TYPE ADC

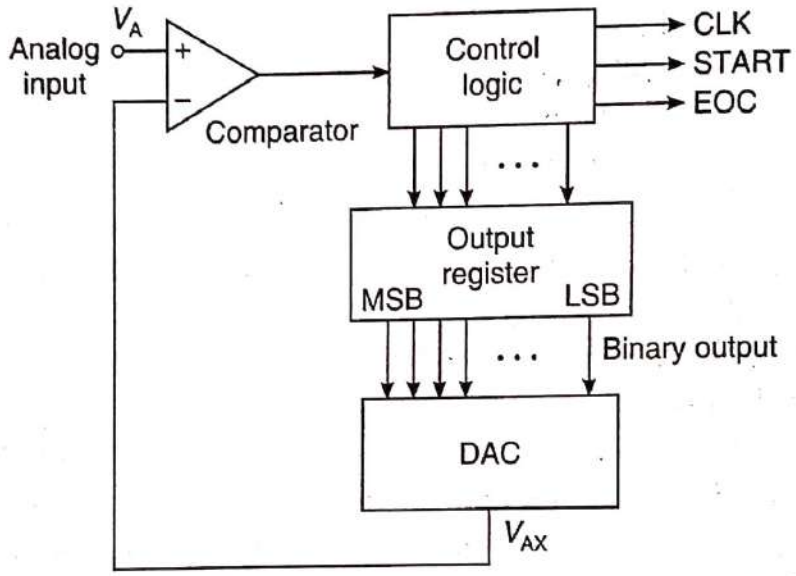


Figure 17.24 The successive-approximation type ADC.

MEMORIES

Basic Types of Memory

- Memory Cell
- Capacity
- Address
- Read and write operation
- Volatile and non-volatile memory
- Random Access Memory (RAM)
- Read Only Memory (ROM)

- **Overview of Memory**
- **Random Access Memory (RAM)**
- **Read Only Memory (ROM)**
- **Programmable ROM (PROM)**
- **Nonvolatile Read/Write Memory**
- **Memory Packaging**
- **Computer Bulk Storage Devices**

OVERVIEW OF MEMORY

Three Important Characteristics of Semiconductor Memory:

- *Density*
  - Amount of data that the memory can store
- *(Non-) Volatility*
  - Data storage capability if power is disconnected
- *Read/write capability*
  - Capability to update memory

OVERVIEW OF MEMORY (Cont'd.)

Categories of Semiconductor Storage Cells:

- SRAM (static random-access memory)
- DRAM (dynamic random-access memory)
- ROM (read-only memory)
- EPROM (electrically programmable ROM)
- EEPROM (electrically erasable PROM)
- Flash Memory

See next slide for characteristics of each category of memory.

RANDOM-ACCESS MEMORY (RAM)

Characteristics of RAM:

- Data can be "written" to RAM
- Stored data can be "read" at any time
- *Volatile* - cannot be used for permanent memory
- Access to any memory location (address) at any moment

Types of RAM:

- SRAM (static RAM) - stores data in flip-flop-like cells. Holds 0 or 1 as long as IC has power (volatile).
- DRAM (dynamic RAM) - memory cells need refreshing many times per second. Also volatile.

READ-ONLY MEMORY (ROM)

Characteristics of ROM:

- *Non-volatile* - memory is not lost when power is turned off
- Data is stored permanently
- Data stored in ROM can be "read" at any time
- ROM cannot be reprogrammed
- High density

PROGRAMMABLE READ-ONLY MEMORY (PROM)

Data can be programmed or "burned" into a PROM

Types of PROM:

- Mask-programmable ROM (usually simply called ROM)
- Field-programmable ROM (PROM)
- Erasable programmable ROM (EPROM)
- Electrically erasable PROM (EEPROM or E^2PROM)
- Flash EEPROM

NON-VOLATILE READ/WRITE MEMORY

- SRAM with battery backup
  - Typically a long-life lithium battery
- NVSRAM (non-volatile static RAM)
  - Better access speed and overall life than SRAM with battery backup
- Flash Memory
  - Nonvolatile
  - In-system rewritable (read/write)
  - Highly reliable
  - Low power consumption
  - High density

MEMORY PACKAGING

Common Methods of Packaging Semiconductor Memory:

- DIP (dual in-line package)
- SIP (single in-line package)
- ZIP (zig-zag in-line package)
- SIMM (single in-line memory module)
- Memory cards

COMPUTER BULK STORAGE DEVICES

Primary storage - computer's internal storage
Secondary storage - external storage

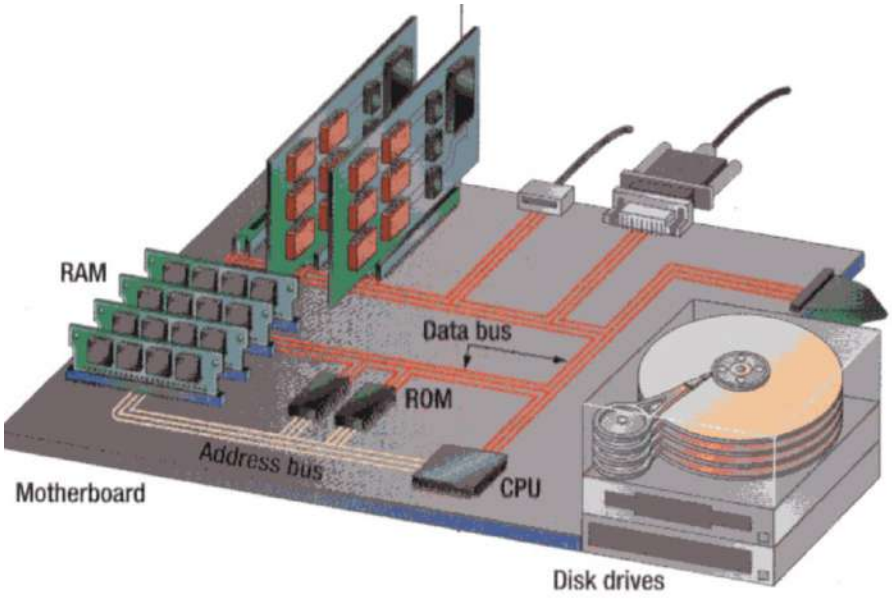
Types of secondary storage devices:

- Mechanical Devices
  - Punched paper card
  - Punched or perforated paper tape
- Magnetic Devices
  - Magnetic tape (sequential-access device)
  - Magnetic drum
  - Hard disk
  - Floppy disk

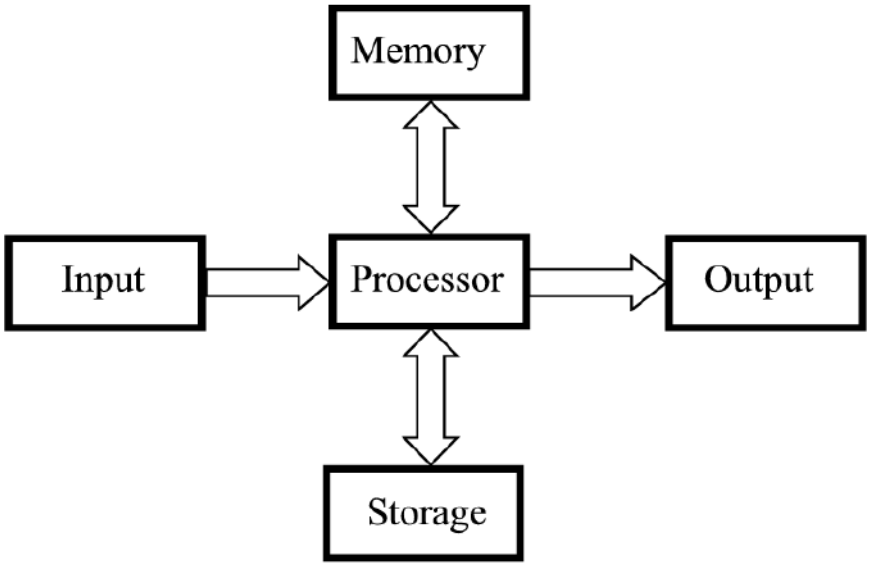
COMPUTER STORAGE DEVICES

Types of Secondary Storage Devices (cont'd.):

- Optical Devices
  - CD-ROM (Read-only)
  - WORM (write-once, read-many)
  - Read/write
- Semiconductor Devices
  - Flash EEPROM semiconductor


RAM
Data bus
ROM
Address bus
CPU
Motherboard
Disk drives

Memory


Memory
Input
Processor
Output
Storage

Characteristics of Storage Devices

- Speed
- Volatility
- Access method
- Portability
- Cost and capacity

Basic Units Of Measurement

- **Bit**  on OR  off

  Bi*nary digi*t

  Smallest unit of measurement

  Two possible values 0 1

- **Byte** 

- **8 bits**

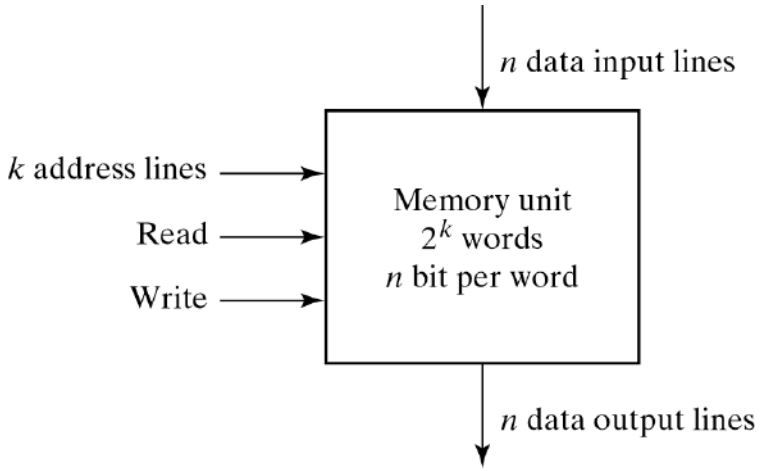
Small Units Of Measurement (Memory, Storage)

- Note: powers of two are used because computer memory and storage are based on the basic unit (bit).
- Kilobyte (KB) – a thousand bytes (1,024 = 2^{10})
- Megabyte (MB) - a million (1,048,576 = 2^{20})

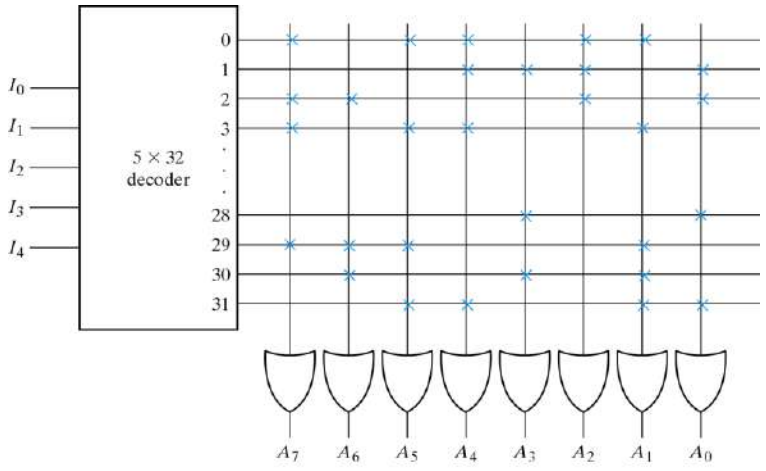
Large Units Of Measurement (Memory, Storage)

- Gigabyte (GB) – a billion (1,073,741,824 = 2^{30})
  - ~ A complete set of encyclopedias requires about 700 MB of storage
  - ~ 30 minutes of video (1/4 of the information stored on a typical DVD)
- Terabyte (TB) – a trillion (1,099,511,627,776 = 2^{40})
  - ~ 20 million four-drawer filing cabinets full of text
  - ~ 200 DVD's of information

RAM



Structure of 32x8 ROM



CONTENTS

- Introduction
- RAM,ROM,PROM,EPROM
- Auxiliary Storage Devices- Magnetic Tape, Hard Disk, Floppy Disk
- Optical Disks: CD-R Drive,CD-RW disks,DVD,Blue ray Discs.

Storage Vs. Memory

Storage (e.g., Hard disk)

- The information is retained longer (non-volatile)
- Slower
- Cheaper

Memory (e.g., RAM)

- Keep the information for a shorter period of time (usually volatile)
- Faster
- More expensive

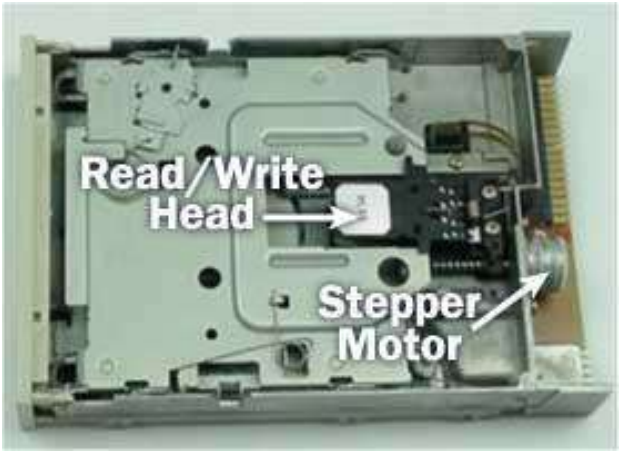
Categories Of Storage

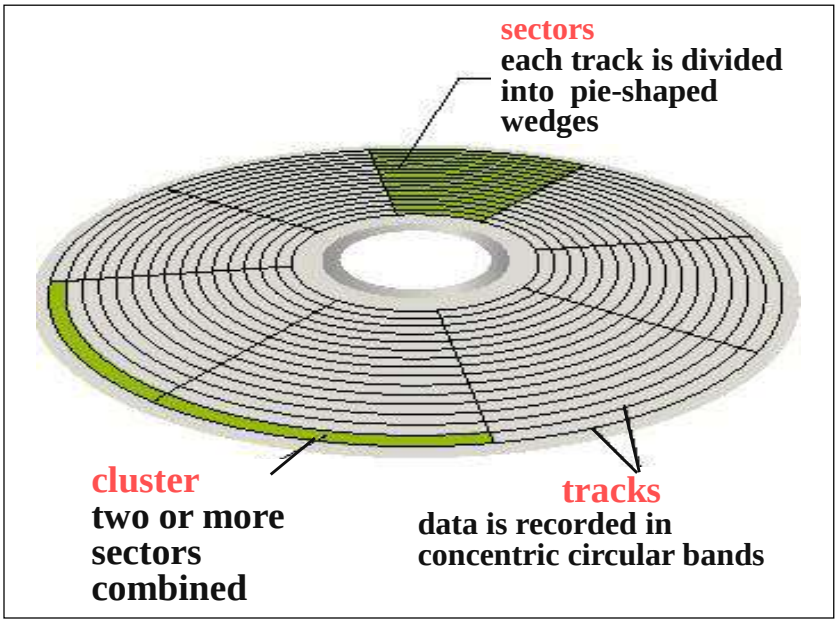
- Magnetic
  - Floppy disks
  - Zip disks
  - Hard drives
- Optical
  - CD-ROM
  - DVD
- Solid state storage devices
  - USB Key (a very common form of solid state storage)

Magnetic Storage

- Exploits duality of magnetism and electricity
  - Converts electrical signals into magnetic charges
  - Captures magnetic charge on a storage medium
  - Later regenerates electrical current from stored magnetic charge
- Polarity of magnetic charge represents bit values zero and one

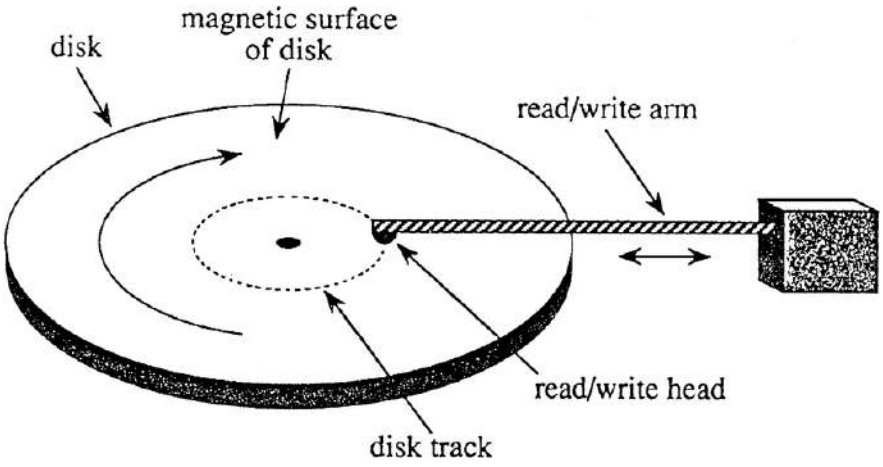
1. Magnetic Drives





500

Magnetic Disk



- Flat, circular platter with metallic coating that is rotated beneath read/write heads
- Random access device; read/write head can be moved to any location on the platter
- Hard disks and floppy disks
- Cost performance leader for general-purpose on-line secondary storage

1. Magnetic Drives: Storage Capacities

- Floppy disks
  - ~ 1 MB
- Hard drives
  - ~80 – 500 GB (TB is possible but very rare)

Floppy Disks

A floppy disk is a portable, inexpensive storage medium that consists of a thin, circular, flexible plastic disk with a magnetic coating enclosed in a square-shaped plastic shell.

Structure Of Floppy Disks

- Initially Floppy disks were 8-inches wide, they then shrank to 5.25 inches, and today the most widely used folly disks are 3.5 inches wide and can typically store 1.44 megabytes of data.
- A folly disk is a magnetic disk, which means that it used magnetic patterns to store data.
- Data in floppy disks can be read from and written to.
- **Formatting** is the process of preparing a disk for reading and writing.
- A track is a narrow recording band that forms a full circle on the surface of the disk.

Hard Disks

- Another form of auxiliary storage is a hard disk. A hard disk consists of one or more rigid metal plates coated with a metal oxide material that allows data to be magnetically recorded on the surface of the platters.
- The hard disk platters spin at a high rate of speed, typically 5400 to 7200 revolutions per minute (RPM).
- Storage capacities of hard disks for personal computers range from 10 GB to 120 GB (one billion bytes are called a gigabyte).

Optical Mass Storage Devices

- Store bit values as variations in light reflection
- Higher areal density & longer data life than magnetic storage
- Standardized and relatively inexpensive
- Uses: read-only storage with low performance requirements, applications with high capacity requirements & where portability in a standardized format is needed

2. Optical Drives

•CD's (Compact Disk)

~ 700 MB storage

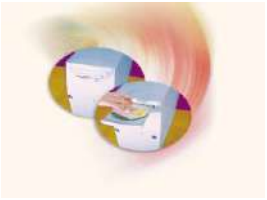
–CD-ROM (read only)

–CD-R: (**r**ecord) to a CD

–CD-RW: can write and erase CD to reuse it (**r**e-**w**ritable)

•DVD(Digital Video Disk)

Compact Discs (CD)

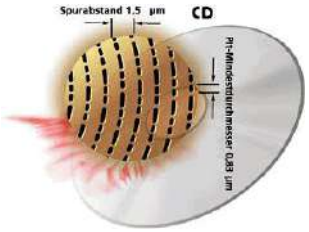


- A compact disk (CD), also called an optical disc, is a flat round, portable storage medium that is usually 4.75 inch in diameter.
- A CD-ROM (read only memory), is a compact disc that used the same laser technology as audio CDs for recording music. In addition it can contain other types of data such as text, graphics, and video.
- The capacity of a CD-ROM is 650 MB of data.

DVD (Digital Video Disk)

DVD-ROM

- Over 4 GB storage (varies with format)
- DVD- ROM (read only)
- Many recordable formats (e.g., DVD-R, DVD-RW; ..)
- Are more highly compact than a CD.
- Special laser is needed to read them



Blu-ray Technology

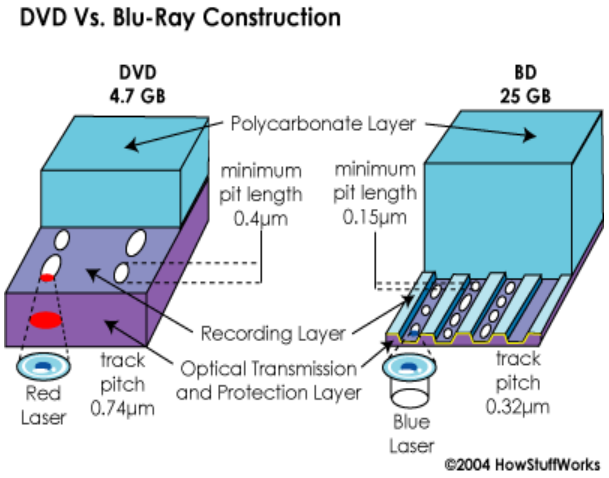
- Name

Derived from the blue-violet laser used to read and write data.

- Developed by the Blu-ray Disc Association with more than 180 members.
  - Dell
  - Sony
  - LG

Blu-ray Technology Cont.

- Data capacity
  - Because Blu-ray uses a blue laser(405 nanometers) instead of a red laser(650 nanometers) this allows the data tracks on the disc to be very compact.
  - This allows for more than twice as small pits as on a DVD.

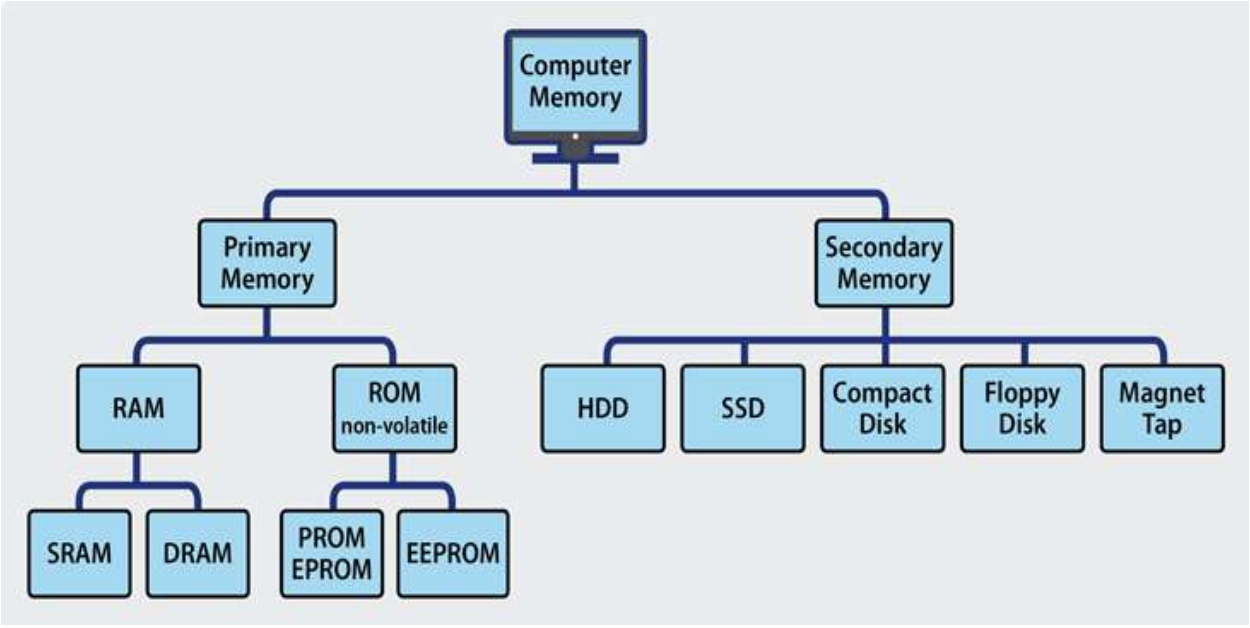


Blu-ray Technology Cont.

Formats

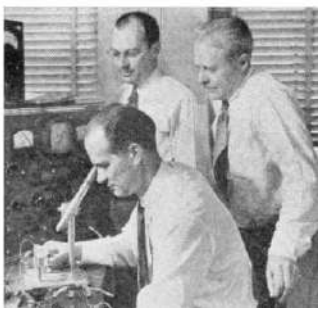
- BD-ROM (read-only) - for pre-recorded content
- BD-R (recordable) - for PC data storage
- BD-RW (rewritable) - for PC data storage
- BD-RE (rewritable) - for HDTV recording

Summary



A bit of history

- The first transistors were fabricated in 1947 at Bell Laboratories (Bell Labs) by Brattain with Bardeen providing the theoretical background and Shockley managed the activity.
  - The trio received a Nobel Prize in Physics for their work in 1956.
    - The transistor was called a point-contact transistor and was a type of bipolar junction transistor (BJT).

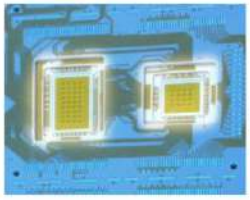


Integrated Circuits

- Integrated circuits (ICs) are chips, pieces of semiconductor material, that contain all of the transistors, resistors, and capacitors necessary to create a digital circuit or system.
  - The first ICs were fabricated using Ge BJTs in 1958.
    - Jack Kirby of Texas Instruments, Nobel Prize in 2000
    - Robert Noyes of Fairchild Semiconductors fabricated the first Si ICs in 1959.

Integration Levels

– SSI Small scale integration	12 gates/chip
– MSI Medium scale integration	100 gates/chip
– LSI Large scale integration	1K gates/chip
– VLSI Very large scale integration	10K gates/chip
– ULSI Ultra large scale integration	100K gates/chip



Logic Families

- Logic families are sets of chips that may implement different logical functions, but use the same type of transistors and voltage levels for logical levels and for the power supplies.
- These families vary by speed, power consumption, cost, voltage & current levels

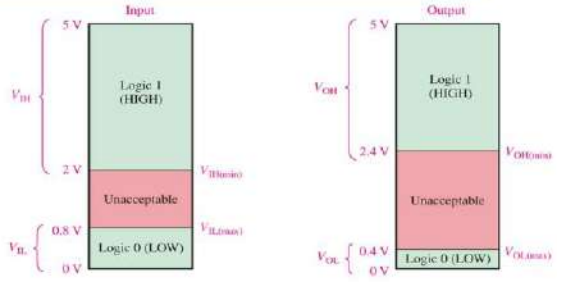
- IC digital logic families
  - DL (Diode- logic)
  - DTL (Diode-transistor logic)
  - RTL (Resistor-transistor logic)
  - TTL (Transistor -transistor logic)
  - ECL (Emitter-coupled logic)
  - MOS (Metal-oxide semiconductor)
  - CMOS (Complementary Metal-oxide semiconductor)

Digital IC Terminology

- Voltage Parameters:
  - V_{IH}(min): high-level input voltage, the minimum voltage level required for a logic 1 at an *input*.
  - V_{IL}(max): low-level input voltage
  - V_{OH}(min): high-level output voltage
  - V_{OL}(max): low-level output voltage

- For proper operation the input voltage levels to a logic must be kept outside the indeterminate range.
- Lower than V_{IL}(max) and higher than V_{IH}(min).

For TTL

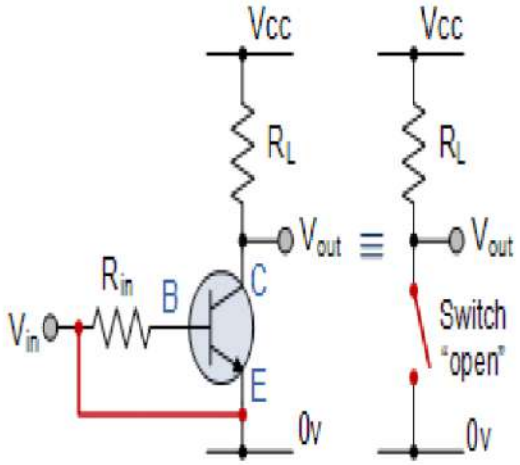


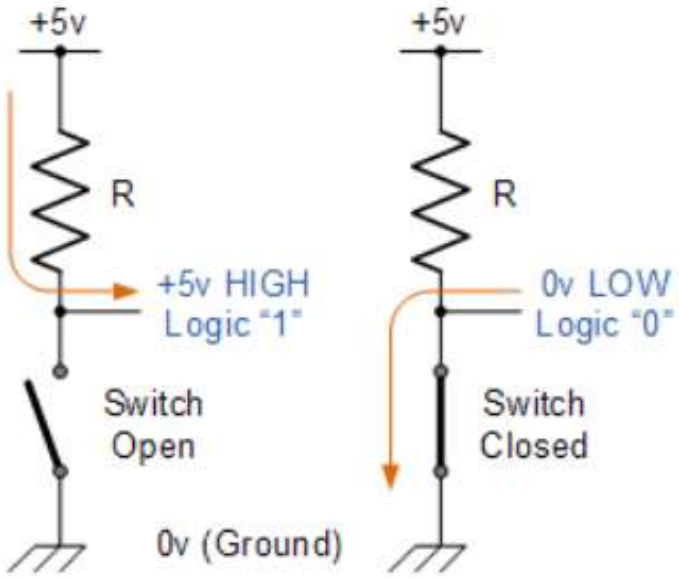
Transistor as a switch

- A circuit that can turn on/off current in electrical circuit is referred to a switching circuit and transistor can be employed as an electronic switch.

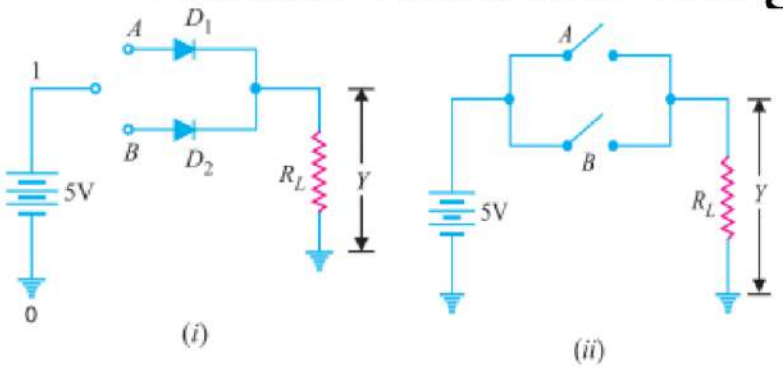
- Cut off region - OFF State
  Both junctions are reverse biased,
  Ic = 0 and V(BE) < 0.7 v

- Saturation region - ON State
  Ic = maximum and V(BE)>0.7 v



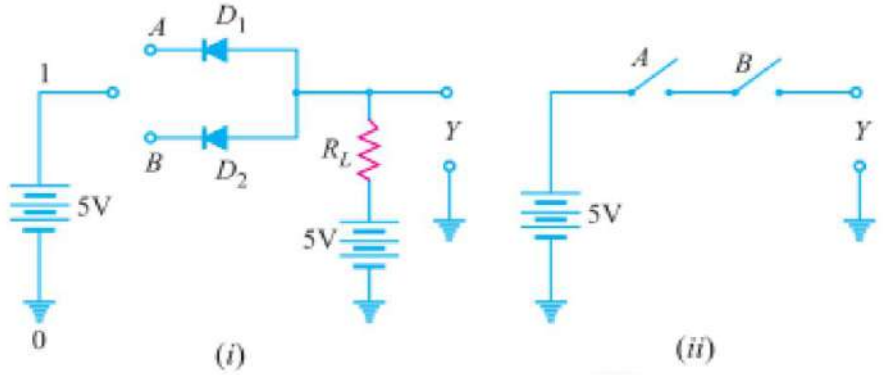


Diode-Resistor OR gate



A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

(iii)

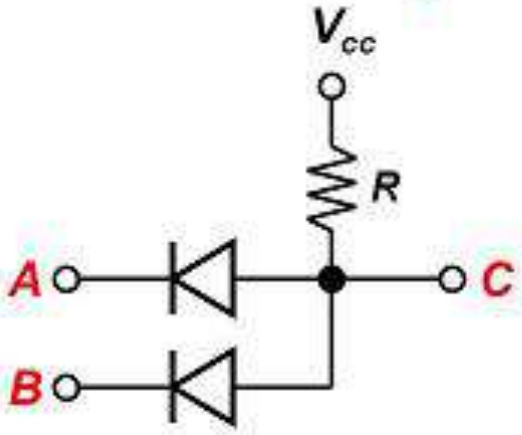


A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

(iii)

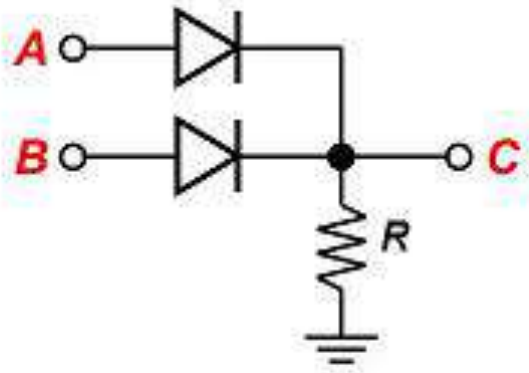
AND gate

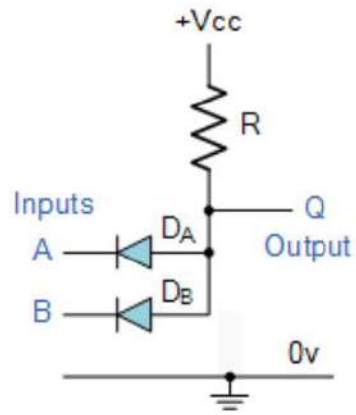
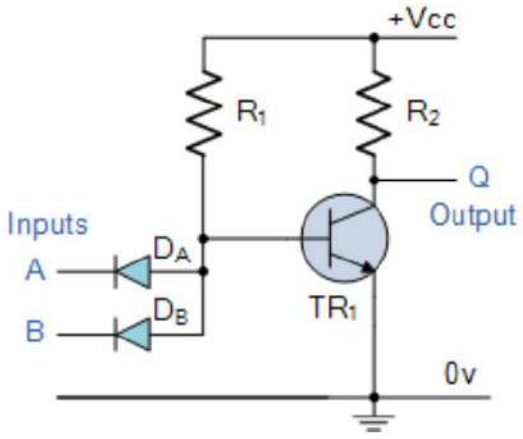
output voltage is high only if both A and B are high

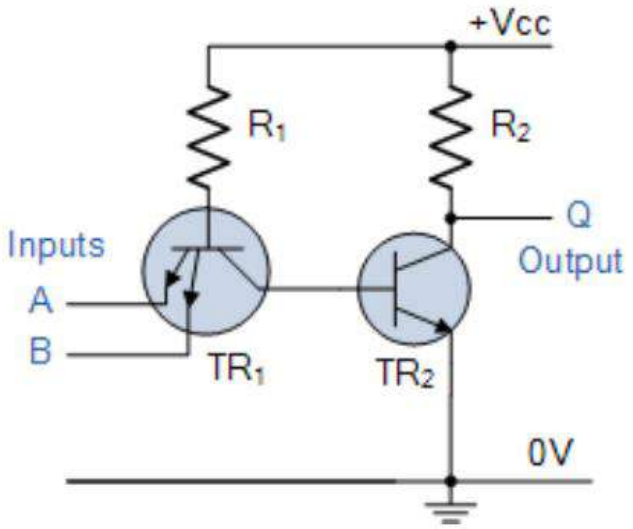


OR gate

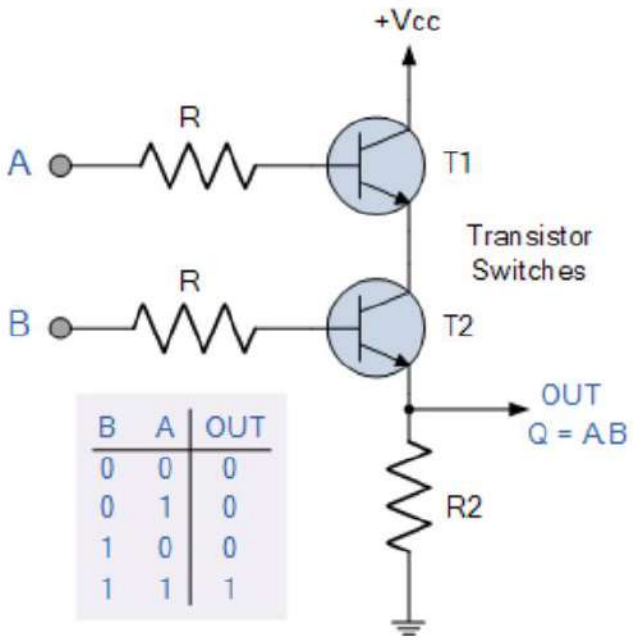
output voltage is high if either (or both) A and B are high



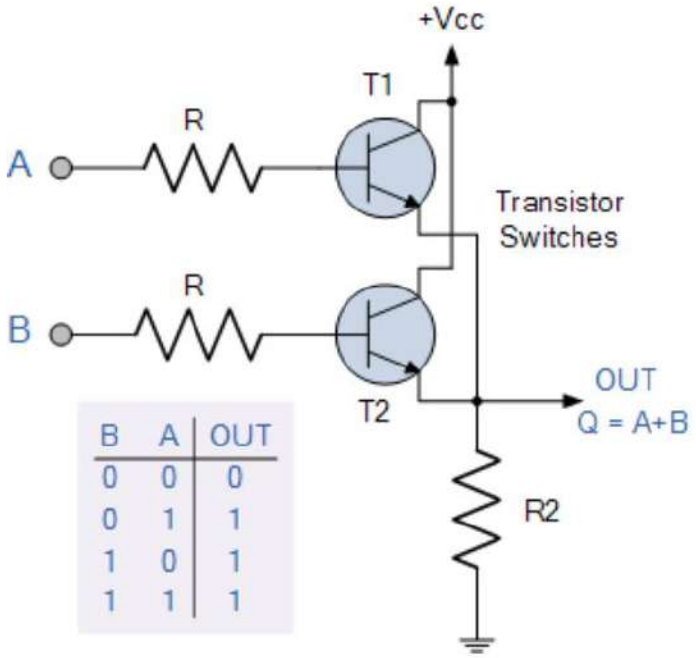
Diode-Resistor Circuit	Diode-Transistor circuit
	
2-input AND Gate	2-input NAND Gate



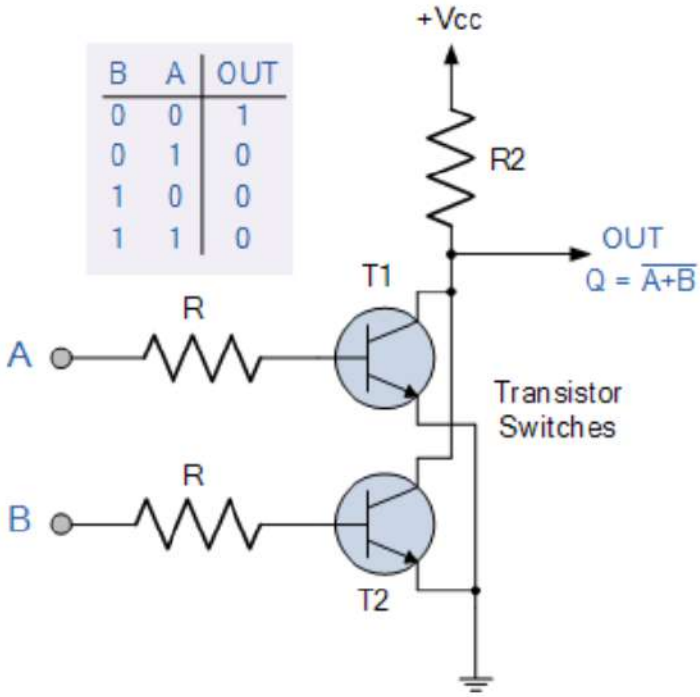
2-input NAND Gate



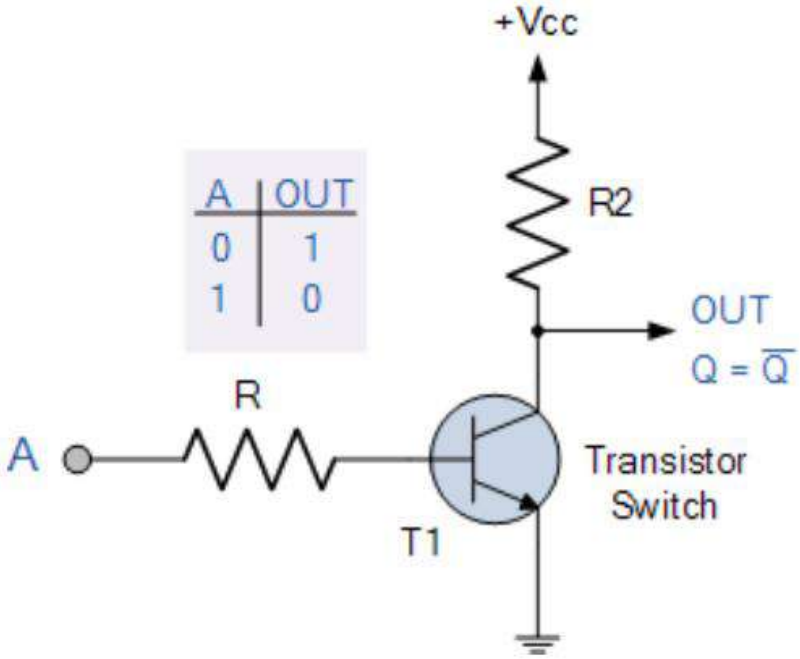
B	A	OUT
0	0	0
0	1	0
1	0	0
1	1	1

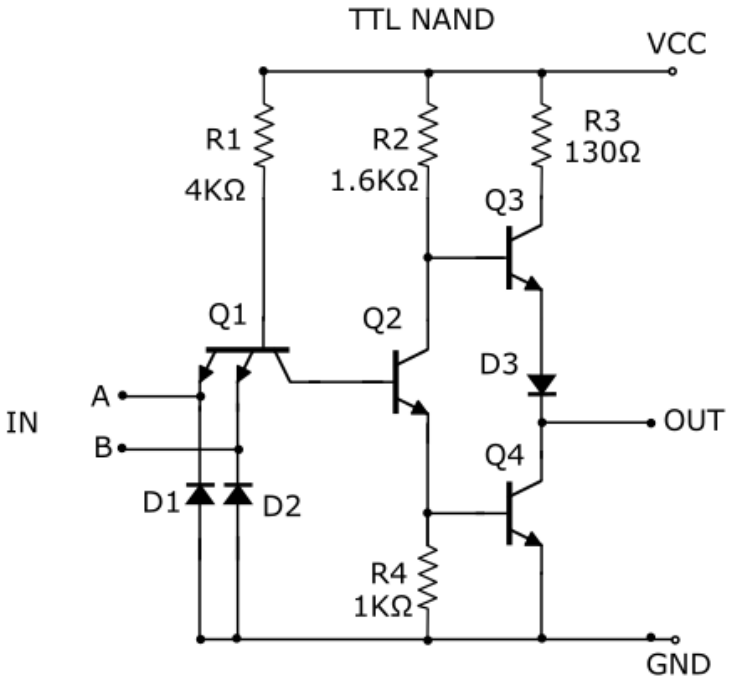


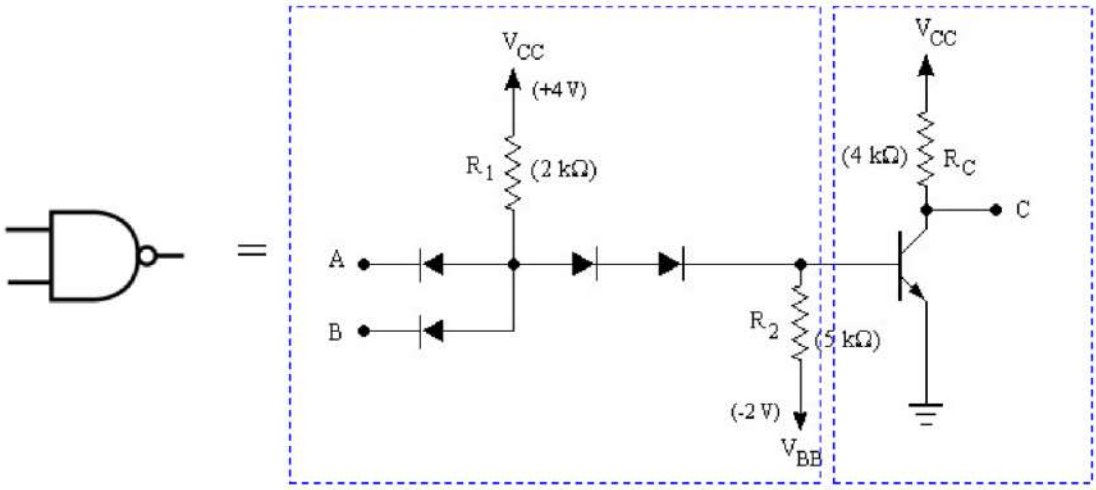
B	A	OUT
0	0	0
0	1	1
1	0	1
1	1	1



B	A	OUT
0	0	1
0	1	0
1	0	0
1	1	0


+Vcc
A | OUT
0 | 1
1 | 0
R2
OUT
Q = $\overline{Q}$
R
A
Transistor
Switch
T1

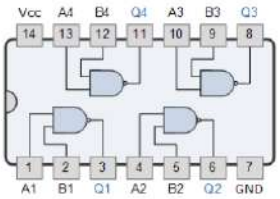

TTL NAND
VCC
R1
4KΩ
R2
1.6KΩ
R3
130Ω
Q3
Q1
Q2
D3
A
IN
B
OUT
Q4
D1
D2
R4
1KΩ
GND



DL AND gate

Saturating inverter

Common ICs



7400 Quad 2-input Logic NAND Gate

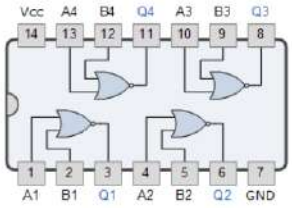
Also:

74LS00 Quad 2-input

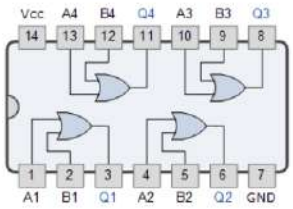
74LS10 Triple 3-input

74LS20 Dual 4-input

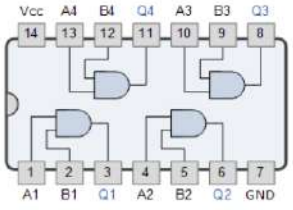
74LS30 Single 8-input



7402 Quad 2-input NOR Gate



7432 Quad 2-input Logic OR Gate



7408 Quad 2-input AND Gate

Question

Q. What are the output voltages caused by logic 1 in each bit position in an 8-bit ladder if the input level for 0 is 0V and that for 1 is +10V?

Q. What is the resolution of a 9-bit DAC, which uses a ladder network? What is this resolution expressed as a percentage? If the full-scale output voltage of this converter is + 5 V, what is the resolution in volts?