Course Code	Course Title	Hours per week L-T-P	Credit C
CSE181303	Digital Systems	3-0-2	4

MODULE 1: Fundamentals of Digital Systems and logic families

Digital signals, digital circuits, AND, OR, NOT, NAND, NOR and Exclusive-OR operations, Boolean algebra, examples of IC gates, number systems-binary, signed binary, octal hexadecimal number, binary arithmetic, one's and two's complements arithmetic, codes, error detecting and correcting codes, characteristics of digital ICs, digital logic families, TTL, Schottky TTL and CMOS logic, interfacing CMOS and TTL, Tri-state logic.

Number Systems

Number system defines a set of values used to represent a quantity.

Name	Base or Radix (r)	Digits (0 to $r-1$)
Binary	2	0, 1
Octal	8	0, 1, 2, 3, 4, 5, 6, 7
Decimal	10	0, 1, 2, ..., 8, 9
Duodecimal	12	0, 1, 2, ..., 7, 8, 9, A, B
Hexadecimal	16	0, 1, 2, ..., 8, 9, A, B, ..., E, F

10 = A 13 = D
11 = B 14 = E
12 = C 15 = F

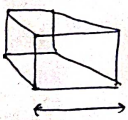
Number System & Codes

- Weighted
  - ex: Decimal, Binary, Octal etc
- Unweighted
  - ex: Gray Code, Excess - 3 Code

$$\underset{\downarrow 10^3}{7}\ \underset{\downarrow 10^2}{3}\ \underset{\downarrow 10^1}{9}\ \underset{\downarrow 10^0}{2} = 7000 + 300 + 90 + 2 = 7\times10^3 + 3\times10^2 + 9\times10^1 + 2\times10^0$$

Weights of each position

So, decimal number system is weighted number system



Any Quantity

Let a quantity is expressed in number system 1 with base r_1 and let N_1 (number of digits) are used to express it.

Let the same quantity is expressed in number system 2 with base r_2 and N_2 digits are used.

If, $r_1 < r_2$ then $N_1 \geq N_2$

For example, $(7392)_{10} = (1110011100000)_2$ ⇒ Base decrease, No. of digit ↑↑

$(7392)_{10} = (1CE0)_6$ ⇒ Base ↓↓, No. of digit is same

- Number System Conversions :-

(i) Decimal to any other

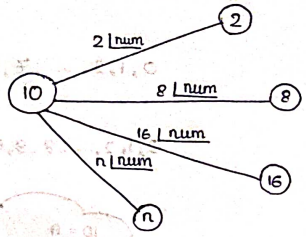
(ii) Any other to decimal

(iii) Octal to binary and vice versa

(iv) Hexa to binary and vice versa

(v) Octal to Hexa and vice versa

(i) Decimal to any other



$24_{10} \rightarrow$

$2 \mid 24$
$2 \mid 12 - 0$
$2 \mid 6 - 0$
$2 \mid 3 - 0$
$1 - 1$

$\rightarrow (11000)_2$

Octal $\hookrightarrow$

$8 \mid 24$
$3 - 0$

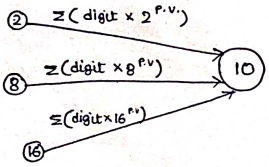
$\rightarrow (30)_8$

Hexadecimal $\hookrightarrow$

$16 \mid 24$
$1 - 8$

$\rightarrow (18)_{16}$

(ii) Any other number system into decimal



Place value (P.V) → a b c . d e

2 1 0 -1 -2 ← place values

↓

Important

Q. Convert $(24.25)_{10}$ to binary.

$2 \mid 24$
$2 \mid 12 - 0$
$2 \mid 6 - 0$
$2 \mid 3 - 0$
$1 - 1$

$0.25 \times 2 = 0.5 \rightarrow 0$

$0.5 \times 2 = 1.0 \rightarrow 1$

$0.0 \times 2 = 0.0$

$$24.25_{10} = 11000.01_2$$

(iii) Octal to binary and binary to octal conversion :

→ There are eight different digits in octal system ; 7+8 ; 0 to 7

→ Each one of them can be represented by a equivalent 3-bit number.

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

• Octal to binary :

(i) $(37 \cdot 45)_8 = (011111 \cdot 100101)_2 = (11111 \cdot 100101)_2$

011 111 100 101

(ii) $(22 \cdot 07)_8 = (10010 \cdot 000111)_2$

010 010 000 111

• Binary to octal :

(i) $(010110 \cdot 110)_2 = (26 \cdot 6)_8$

(ii) $(001001 \cdot 100)_2 = (11 \cdot 4)_8$

(iv) Hexadecimal to binary and binary to hexadecimal conversion :

→ In hexadecimal number system we have 16 different digits

→ Each one of them can be represented by a equivalent 4-bit binary number.

0 - 0000
1 - 0001
2 - 0010
3 - 0011
4 - 0100
5 - 0101
6 - 0110
7 - 0111
8 - 1000
9 - 1001
A - 1010
B - 1011
C - 1100
D - 1101
E - 1110
F - 1111

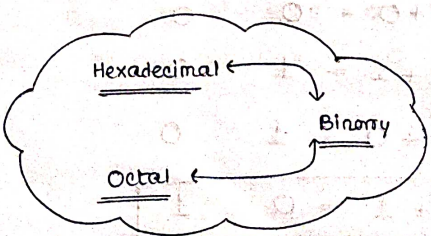
• Hex to binary :

(i) $(259A)_{16} \longrightarrow (0010010110011010)_2$

(ii) $(CAFE \cdot 3D)_{16} \longrightarrow (1100101011111110 \cdot 0011 1101)_2$

• Binary to hexadecimal :

(i) $(000100001001 \cdot 1100)_2 \longrightarrow (189 \cdot C)_{16}$



PRACTICE a few conversions yourself.
Try to solve the * of PYQs

- Binary Arithmetic :

⓪	①	
5	4	₹
+	2	5
5	₹	2

⟶

(III)	(II)	(I)
5×10^2	$+\ 4\times10^1$	$+\ ₹\times10^0$
	2×10^1	$+\ 5\times10^0$
5	₹	2

5×10^2 $\quad$ $₹\times10^1$

$12 > 9$ $\quad$ 0 to 9

12×10^0

$12 = 10 + 2$ → radix

$= (10+2)\times10^0$

$= 10\times10^0 + 2\times10^0$

$= \underbrace{1\times10^1}_{\text{Carry}} + 2\times10^0$

- Binary addition :

ⓘ	2^3	2^2	2^1	2^0
	1	1	1	0
+	1	0	1	
	1	0	1	1

⟶

1×2^2	$+\ 1\times2^1$	$+\ 0\times2^0$	
$+\ 1\times2^2$	$+\ 0\times2^1$	$+\ 1\times2^0$	
1×2^3	0×2^2	1×2^1	1×2^0

2×2^2

$2 > 1$ $\quad$ 0 to 1

$2 = 2 + 0$ → Radix

$\therefore (2+0)\times2^2$

$= 2\times2^2 + 0\times2^2$

$= \underbrace{1\times2^3}_{\text{Carry}} + 0\times2^2$

★ Summary of binary addition:

	Sum	Carry
$0+0$	0	0
$0+1$	1	0
$1+0$	1	0
$1+1$	0	1

- Binary subtraction :

(i) 1 1 0 1 1 → $1\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0$

borrow: $1\times2^3 = 2\times2^2$

− 1 0 1 1 0 → $1\times2^4 + 0\times2^3 + 1\times2^2 + 1\times2^1 + 0\times2^0$

0 0 1 0 1 ← $0\times2^4 + 0\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0$

Recheck

1 1 0 1 1 (×16 ×8 ×4 ×2 ×1) = 16 + 8 + 0 + 2 + 1 = (27)

− 1 0 1 1 0 = 16 + 0 + 4 + 2 + 0 = (22)

0 0 1 0 1 = 4 + 0 + 1 = (5)

27 − 22 = 5

Verified !! Correct

(ii) 1 1 1 0 → 14

− 1 1 1 → 7

1 1 1 → 7

- Binary multiplication :

In Summation

	Sum	Carry
0 + 0 = 0	0	
0 + 1 = 1	0	
1 + 0 = 1	0	
1 + 1 = 0	1	

In multiplication

multiplication

0 × 0 = 0

0 × 1 = 0

1 × 0 = 0

1 × 1 = 1

(i) 1 0 1 0 → 10

× 1 0 1 → 5

1 1 0 0 1 0 → 50

$1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0$

× $1\times2^2 + 0\times2^1 + 1\times2^0$

$1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0$

$0\times2^4 + 0\times2^3 + 0\times2^2 + 0\times2^1$

$1\times2^5 + 0\times2^4 + 1\times2^3 + 0\times2^2$

$1\times2^5 + 1\times2^4 + 0\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0$

1 1 0 0 1 0

(ii)
1 0 1 0
× 1 1
1 0 1 0
1 0 1 0 ×
1 1 1 1 0

- Division :

$$8 \overline{)128} = 16$$

$$-0$$
$$12$$
$$-8$$
$$48$$
$$-48$$
$$0$$

(ii)

$$110 \overline{)101010} = 111$$

$$-0$$
$$10$$
$$-0$$
$$101$$
$$-0$$
$$1010$$
$$-110$$
$$01001$$
$$-110$$
$$00110$$
$$-110$$
$$0$$

$$\frac{101010}{110} = 111$$

(iii) $1001 \div 11$

$$11 \overline{)1001} = 0011$$

$$-0$$
$$10$$
$$-0$$
$$100$$
$$-11$$
$$01\,01$$
$$-11$$
$$0$$

$$\frac{1001}{11} = 11$$

(iii) $111000 \div 111$

$$111 \overline{)111000} = 001000$$

$$-0$$
$$11$$
$$-0$$
$$111$$
$$-111$$
$$0$$
$$-0$$
$$0$$
$$-0$$
$$0$$
$$-0$$
$$0$$

$$\frac{111000}{111} = 1000$$

Complement

r's complement
[Radix complement]

(r-1)'s complement
[Diminished Radix complement]

Ex:- Complement of 7_{10} ⟶ N = 7

$10 - 7 = 3$

n = 1 ⟶ No. of digits

Complement of 9_{10}

$10 - 9 = 1$

r = 10 ⟶ Radix

$r^n - N$ ⟶ r's complement

Complement of 6_{10}

$10 - 6 = 4$

(i) N = 5690. Determine 10's complement.

r = 10, N = 5690, n = 4

$= 10^4 - 5690$

$= 10000 - 5690$

$= 4310$ Ans

(ii) N = 1101. Find 2's complement

r = 2, N = 1101, n = 4

$= 2^4 - 1101$

$= 16_{10} - 1101$

$= 10000 - 1101$

$= 11$ Ans

$$\begin{array}{r} 1\,0\,0\,0\,0 \\ -\ 1\,1\,0\,1 \\ \hline 0\,0\,1\,1 \end{array}$$

- $(r-1)$'s complement

r's complement ⟵ $(r-1)$'s complement

$r = 10$	10's complement	9's complement
$r = 2$	2's complement	1's complement
$r = 8$	8's complement	7's complement
$r = 16$	16's complement	15's complement

r's complement $= r^n - N$

$(r-1)$'s complement $= r^n - N - 1$

$(r-1)$'s complement = r's complement $- 1$

$\Rightarrow$ r's complement = $(r-1)$'s complement $+ 1$

Borrow involved — Borrowing not involved → so it is easier to calculate

Ex: 7's complement of 5674_8

Skip
- $= 8^4 - 5674 - 1$
- $= (4096)_{10} - 5674 - 1$
- $= 10000_8 - 5674 - 1$

$= $ ~~9999 - 5674~~ 7777 - 5674

$= 2103$ Ans

$r = 8$
$N = 5674$
$n = 4$

10000
1
7777

8 | 4096
8 | 512 - 0
8 | 64 - 0
8 | 8 - 0
1 - 0

Ex: 1's complement of 1101.

Skip
- $= 2^4 - 1101 - 1$
- $= 16_{10} - 1101 - 1$
- $= 10000 - 1101 - 1$
- $= 1111 - 1101$
- $= 10$ Ans ⟶ 1's complement

$r = 2$
$N = 1101$
$n = 4$

10000
1
1111

2's complement of 1101 = 1's complement $+ 1 = 11$ Ans

- 1's and 2's complement :

(i) Obtain 1's complement of 1010

= 1111 − 1010

= 101 Ans

1 0 1 0 (Given No.)

↓ ↓ ↓ ↓

0 1 0 1 (1's complement)

$(r-1)$'s complement = $r^4 - N - 1$

$= (r^4 - 1) - N$

If $r = 10 \longrightarrow r^4 - 1 \longrightarrow 9999$

" $r = 8 \longrightarrow r^4 - 1 \longrightarrow 7777$

" $r = 16 \longrightarrow r^4 - 1 \longrightarrow FFFF$

" $r = 2 \longrightarrow r^4 - 1 \longrightarrow 1111$

- Shortcut for 2's complement :

Step 1 : Write down the given number

Step 2 : Starting from LSB, copy all the zeroes till the first 1.

Step 3 : Copy the first 1

Step 4 : Complement all the remaining bits

$(10111000)_2$

1 0 1 1 1 0 0 0 ← LSB

1's complement : 0 1 0 0 0 1 1 1

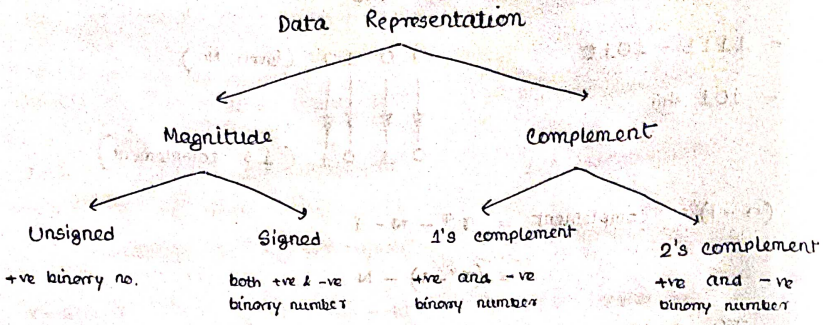
\+ 1

2's complement : 0 1 0 0 1 0 0 0

2's complement :

0 1 0 0 1 0 0 0

- Data representation using signed magnitude :



→ In all four representation, positive number is represented in same way.

Imp. Note

- Unsigned magnitude :

$+6 = 110$

$-6 =$ can't represent

- Signed magnitude :

$+6 = 0\,110$ (0 : Sign bit; 110 : magnitude)

$-6 = 1\,110$ (1 : Sign bit; 110 : magnitude)

Sign bit = 0 ⇒ Number is +ve

" " = 1 ⇒ " " -ve

$+13 = 01101$

$-13 = 11101$

Range : $-(2^{n-1}-1)$ to $+(2^{n-1}-1)$ | n = no. of variable

For $n = 4$,

range $= -(2^3-1)$ to $+(2^3-1)$

$= -7$ to $+7$

- Data representation using 1's complement :

$+6 = 0\ 1\ 1\ 0$

$-6 = 1\ 0\ 0\ 1$

$+9 = 0\ 1\ 0\ 0\ 1$

$-9 = 1\ 0\ 1\ 1\ 0$

$+0 = 0\ 0\ 0\ 0$ (positive zero)

$-0 = 1\ 1\ 1\ 1$ (negative ")

Range : $-(2^{n-1}-1)$ to $+(2^{n-1}-1)$

For 4 variables, $n = 4$

$-(2^3-1)$ to $+(2^3-1) = -7$ to $+7$

- Data representation using 2's complement :

$+4 = 0\ 1\ 0\ 0$

1's comp. $\rightarrow 1\ 0\ 1\ 1$

2's comp. $\rightarrow$ 

$1\ 1\ 0\ 0 = -4$

$+6 = 0\ 1\ 1\ 0$

1's comp. $\rightarrow 1\ 0\ 0\ 1$

2's " $\rightarrow 1\ 0\ 1\ 0 = -6$

$\rightarrow$ Only one zero = 0000

Range : -2^{n-1} to $+2^{n-1}-1$

$n=4$

-2^3 to $+2^3-1$

$\Rightarrow -8$ to $+7$

Ex: 2's complement $\rightarrow$ 1011, find the positive number.

$1\ 0\ 1\ 1 \rightarrow$ 2's comp $\rightarrow -5$

$0\ 1\ 0\ 0 \rightarrow$ 1's "

$\quad\quad +\ \ 1$

$0\ 1\ 0\ 1 \rightarrow$ 2's comp $\rightarrow +5$

∴ The positive no. is +5

- MSB indicates sign

1 $\rightarrow$ -ve number.

3. Starting at the LSB, copying down each bit up to and including the first 1 bit encountered, and complementing the remaining bits.

EXAMPLE 2.23 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

Subtract the given number N from 2^n

2^n =	1 0 0 0 0 0 0 0 0
Subtract 45 = –	0 0 1 0 1 1 0 1
Thus, the 2's complement form of – 45 is	1 1 0 1 0 0 1 1

Third 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

EXAMPLE 2.24 Express – 73.75 in 12-bit 2's complement form.

Solution

+ 73.75 = N = 01001001.1100

First method

Positive expression of the given number	0 1 0 0 1 0 0 1 . 1 1 0 0
1's complement of it	1 0 1 1 0 1 1 0 . 0 0 1 1
Add 1	+ 1
Thus, the 2's complement of – 73.75 is	1 0 1 1 0 1 1 0 . 0 1 0 0

Second method

2^8 =	1 0 0 0 0 0 0 0 0 . 0 0 0 0
= –	0 1 0 0 1 0 0 1 . 1 1 0 0
Subtract 73.75	
Thus, the 2's complement of – 73.75 is	1 0 1 1 0 1 1 0 . 0 1 0 0

Third method

Original number	0 1 0 0 1 0 0 1 . 1 1 0 0
Copy up to the first 1 bit	1 0 0
Complement the remaining bits	1 0 1 1 0 1 1 0 . 0
Thus, – 73.75 in 2's complement form is	1 0 1 1 0 1 1 0 . 0 1 0 0

Two's Complement Arithmetic

The 2's complement system is used to represent negative numbers using modulus arithmetic. The word length of a computer is fixed. That means, if a 4-bit number is added to another 4-bit number, the result will be only of 4 bits. Carry, if any, from the fourth bit will overflow. This is called the *modulus arithmetic*. For example: 1100 + 1111 = 1011.

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.

EXAMPLE 2.25 Subtract 14 from 46 using the 8-bit 2's complement arithmetic.

Solution

+ 1 4	=	0 0 0 0 1 1 1 0
– 1 4	=	1 1 1 1 0 0 1 0 (In 2's complement form)
+ 4 6		0 0 1 0 1 1 1 0
– 1 4	⇒	+ 1 1 1 1 0 0 1 0 (2's complement form of – 14)
+ 3 2		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

EXAMPLE 2.26 Add – 75 to + 26 using the 8-bit 2's complement arithmetic.

Solution

+ 7 5	=	0 1 0 0 1 0 1 1
– 7 5	=	1 0 1 1 0 1 0 1 (In 2's complement form)
+ 2 6		0 0 0 1 1 0 1 0
– 7 5	⇒	+ 1 0 1 1 0 1 0 1 (2's complement form of – 75)
– 4 9		1 1 0 0 1 1 1 1 (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.

EXAMPLE 2.27 Add – 45.75 to + 87.5 using the 12-bit 2's complement arithmetic.

Solution

```
 +87.5          0 1 0 1 0 1 1 1 . 1 0 0 0
 -45.75  ⇒    + 1 1 0 1 0 0 1 0 . 0 1 0 0   (– 45.75 in 2's complement form)
 +41.75       ❶ 0 0 1 0 1 0 0 1 . 1 1 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 + 41.75

EXAMPLE 2.28 Add 27.125 to – 79.625 using the 12-bit 2's complement arithmetic.

Solution

```
 +27.125          0 0 0 1 1 0 1 1 . 0 0 1 0
 -79.625  ⇒     + 1 0 1 1 0 0 0 0 . 0 1 1 0   (– 79.625 in 2's
                                                complement form)
 -52.500          1 1 0 0 1 0 1 1 . 1 0 0 0   (No carry)
```

There is no carry, indicating that the result is negative and is in its 2's complement form. The 2's complement of 11001011.1000 is 00110100.1000. Therefore, the result is – 52.5.

EXAMPLE 2.29 Add – 31.5 to – 93.125 using the 12-bit 2's complement arithmetic.

Solution

```
 -93.125          1 0 1 0 0 0 1 0 . 1 1 1 0   (– 93.125 in 2's complement form)
 -31.500  ⇒     + 1 1 1 1 0 0 0 0 . 1 0 0 0   (– 31.5 in 2's complement form)
-124.625        ❶ 1 0 0 0 0 0 1 1 . 0 1 1 0   (Ignore the carry)
```

There is a carry, ignore it. The MSB is a 1; so, the result is negative and is in its 2's complement form. The 2's complement of 1000 0011.0110 is 0111 1100.1010. Therefore, the result is – 124.625.

EXAMPLE 2.30 Add 47.25 to 55.75 using the 2's complement method.

Solution

```
 47.25          0 0 1 0 1 1 1 1 . 0 1 0 0
 55.75   ⇒    + 0 0 1 1 0 1 1 1 . 1 1 0 0
103.00          0 1 1 0 0 1 1 1 . 0 0 0 0   (No carry)
```

There is no carry, and the MSB is 0. Therefore, the result is positive and is in its true binary form. Hence, it is equal to + 103.0.

EXAMPLE 2.31 Add + 40.75 to – 40.75 using the 12-bit 2's complement arithmetic.

Solution

```
 +40.75          0 0 1 0 1 0 0 0 . 1 1 0 0
 -40.75  ⇒     + 1 1 0 1 0 1 1 1 . 0 1 0 0   (– 40.75 in 2's complement form)
  00.00        ❶ 0 0 0 0 0 0 0 0 . 0 0 0 0   (Ignore the carry)
```

There is a carry, ignore it. The result is 0.

One's Complement Arithmetic

The 1's complement of a number is obtained by simply complementing each bit of the number, that is, by changing all the 0s to 1s and all the 1s to 0s. 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 complement is 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.*

EXAMPLE 2.32 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 = 0 1 1 0 0 0 1 1
-99 = 1 0 0 1 1 1 0 0     (In 1's complement form)
```

(b)
```
+77.25 = 0 1 0 0 1 1 0 1 . 0 1 0 0
-77.25 = 1 0 1 1 0 0 1 0 . 1 0 1 1   (In 1's complement form)
```

In 1's complement subtraction, add the 1'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 1's complement to get the magnitude in binary.

EXAMPLE 2.33 Subtract 14 from 25 using the 8-bit 1's complement arithmetic.

Solution

```
 25          0 0 0 1 1 0 0 1
 -14   ⇒   + 1 1 1 1 0 0 0 1   (In 1's complement form)
 +11       ❶ 0 0 0 0 1 0 1 0
                         + 1   (Add the end around carry)
           0 0 0 0 1 0 1 1     = + 11₁₀
```

EXAMPLE 2.34 Add – 25 to + 14 using the 8-bit 1's complement method.

Solution

```
 +14          0 0 0 0 1 1 1 0
 -25   ⇒    + 1 1 1 0 0 1 1 0   (In 1's complement form)
 -11          1 1 1 1 0 1 0 0   (No carry)
```

There is no carry and 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}.

EXAMPLE 2.35 Add − 25 to − 14 using the 8-bit 1's complement method.

Solution

```
 - 2 5          1 1 1 0 0 1 1 0     (In 1's complement form)
 - 1 4   ⇒    + 1 1 1 1 0 0 0 1     (In 1's complement form)
 -----        -----------------
 - 3 9        0 1 1 0 1 0 1 1 1
                            + 1     (Add the end around carry)
              -----------------
                1 1 0 1 1 0 0 0
```

The MSB is a 1. So, the result is negative and is in its 1's complement form. The 1's complement of 11011000 is 00100111. Therefore, the result is − 39.

EXAMPLE 2.36 Add + 25 to + 14 using the 8-bit 1's complement arithmetic.

Solution

```
 + 2 5          0 0 0 1 1 0 0 1
 + 1 4   ⇒    + 0 0 0 0 1 1 1 0
 -----        -----------------
 + 3 9          0 0 1 0 0 1 1 1
```

There is no carry. The MSB is a 0. So, the result is positive and is in pure binary. Therefore, the result is, 00100111 = + 39.

EXAMPLE 2.37 Add + 25 to − 25 using the 8-bit 1's complement method.

Solution

```
 + 2 5          0 0 0 1 1 0 0 1
 - 2 5   ⇒    + 1 1 1 0 0 1 1 0     (In 1's complement form)
 -----        -----------------
   0 0          1 1 1 1 1 1 1 1
```

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 11111111 is 00000000. Therefore, the result is − 0.

EXAMPLE 2.38 Subtract 27.50 from 68.75 using the 12 bit 1's complement arithmetic.

Solution

```
 + 6 8 . 7 5          0 1 0 0 0 1 0 0 . 1 1 0 0
 - 2 7 . 5 0   ⇒    + 1 1 1 0 0 1 0 0 . 0 1 1 1     (In 1's complement form)
 -----------        -------------------------
 + 4 1 . 2 5        0 0 0 1 0 1 0 0 1 . 0 0 1 1
                                              + 1   (Add the end around carry)
                    -------------------------
                      0 0 1 0 1 0 0 1 . 0 1 0 0
```

The MSB is a 0. So, the result is positive and is in its normal binary form. Therefore, the result is + 41.25.

1's complement, 2's complement arithmetic is a bit confusing topic.

Do practice the above examples yourself

EXAMPLE 2.39 Add − 89.75 to + 43.25 using the 12-bit 1's complement method.

Solution

```
 + 4 3 . 2 5          0 0 1 0 1 0 1 1 . 0 1 0 0
 - 8 9 . 7 5   ⇒    + 1 0 1 0 0 1 1 0 . 0 0 1 1     (In 1's complement form)
 -----------        -------------------------
 - 4 6 . 5 0          1 1 0 1 0 0 0 1 . 0 1 1 1
```

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 11010001.0111 is 00101110.1000. Therefore, the result is − 46.50.

Double Precision Numbers

I[illegible] omputer the word length is fixed. In a 16-bit computer, that is, in a computer [illegible] ngth, only numbers from $+ 2^{16-1} - 1$ (+ 32,767) to $- 2^{16-1}$ (− 3[illegible] be e[illegible] ch register. If numbers greater than this are to be expressed [illegible] locations [illegible] ed, that is, each such number has to be stored in tw[illegible] his is called *dou*[illegible] Leaving the MSB which is the sign bit, t[illegible] a 31-bit number length w[illegible] registers. If still larger numbers are [illegible] ssed, three registers are used to [illegible] mber. This is called *triple pr*[illegible]

Floating Point Number[illegible]

In the decimal system, very large an[illegible] numbe[illegible] ssed in scientific notation, by stating a number (mantissa) and a[illegible] ples are 6.53×10^{-27} and 1.58×10^{21}. Binary numbers can also be ex[illegible] e notation by stating a number and an exponent of 2. However, the format [illegible] ter may be as shown below.

Mantiss[illegible]

011000[illegible] 10010[illegible]

In this machine, the 16-bit wo[illegible] of two parts, a [illegible] tissa and a 6-bit exponent. The mantissa is in [illegible] ment form. So, the MSB [illegible] ught of as the sign bit. The binary point [illegible] to be to the right of this sig[illegible] bits of the exponent could represen[illegible] gh 63. However, to express nega[illegible] ents, the number 32_{10} (100000[illegible] added to the desired exponent.

The actual expo[illegible] e number, therefore, is equal to the exponen[illegible] the 6 bits minus 32. [illegible] common system used in floating point formats. It [illegible] excess-32 notati[illegible] ing to these definitions, the floating-point number shown [illegible]

The m[illegible] tion = + 0 . 1 1 0 0 0 0 0 0 0

The [illegible] portion = 1 0 0 1 0 1

Th[illegible] xponent = 1 0 0 1 0 1 − 1 0 0 0 0 0 = 0 0 0 1 0 1

T[illegible] number $= N = + 0.1100_2 \times 2^5 = 11000_2 = 24_{10}$

[illegible] are many formats of floating point numbers, each computer having its own. So[illegible] two words for the mantissa, and one for the exponent; others use one and half

PRACTICE :

Q.1 Convert binary to decimal : ~~10011~~ $11011 \cdot 101_2$

Soln We have

$$11011 \cdot 101 = (1\times 2^4) + (1\times 2^3) + (0\times 2^2) + (1\times 2^1) + (1\times 2^0) + (1\times 2^{-1}) + (0\times 2^{-2}) + (1\times 2^{-3})$$

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

$$= 26.625_{10}$$

Q.2 Convert 163.875_{10} to binary.

Soln Conversion of integer 163 -

$$\begin{array}{r|l} 2 & 163 \\ 2 & 81 - 1 \\ 2 & 40 - 1 \\ 2 & 20 - 0 \\ 2 & 10 - 0 \\ 2 & 5 - 0 \\ 2 & 2 - 1 \\ & 1 - 0 \end{array}$$

$\therefore 163_{10} = 10100011_2$

Conversion of fraction part

$0.875 \times 2 = 1.75 \rightarrow 1$

$0.75 \times 2 = 1.5 \rightarrow 1$

$0.5 \times 2 = 1.0 \rightarrow 1$

$\therefore 0.875_{10} = 111_2$

Hence, $163.875_{10} = 10100011.111_2$

Q.3 Add the binary numbers 1101.101 and 111.011

Soln

$$\begin{array}{r} \overset{1}{} \quad \overset{1}{} \\ 1101.101 \\ 111.011 \\ \hline 10101.000 \end{array}$$

1+1+1 = 1 and carry 1

Hence, required result is 10101.000_2

Q.4 Subtract 111.111_2 from 1010.010_2

Soln

$$\begin{array}{r} 1010.010 \\ - \; 111.111 \\ \hline 0010.011 \end{array}$$

Hence, the result is 0010.011_2

Q.5 Multiply 1011.101_2 by 101.01_2.

$4 \times (1 \times 2^2) = 4 \times 2^2$
$= 1 \times 2^4$

Soln//

```
            1011.101
          × 101.010
      ----------------
          0000 000
        10111 01 ×
      000000 0 ×
     1011101  ×
    0000000 ×
   1011 101 ×
   ----------------
   111101.00000010
```

Hence, the result is 111101.00001

Q.6 Divide 110101.11_2 by 101_2

```
101 ) 110101.11 ( 1010.11
      101
      ----
       110
       101
       ----
        111
        101
        ----
         101
         101
         ----
         000
```

Q.7 Express −45 in 8-bit 2's complement form.

```
2 | 45
2 | 22 - 1
2 | 11 - 0
2 | 5 - 1
2 | 2 - 1
    1 - 0
```

Soln// +45 in 8 bit form is 00101101

1's complement of 00101101 = 11010010

Adding 1 + 1

11010011

Thus, 2's complement form of −45 is 11010011

Q.8 Express −73.75 in 12 bit 2's complement form.

Soln// +73.75 = 01001001.1100

1's complement = 10110110.0011

\+ 1

10110110.0100 → −73.75 in 2's complement form

IMPORTANT

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

Soln// $+46_{10} = 0\ 0\ 1\ 0\ 1\ 1\ 1\ 0_2$

$+14_{10} = 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0_2$

-14 in 1's complement form is 11110001

Adding 1 $\quad +\ \ 1$

$\quad 11110010$

$\therefore -14$ in 2's complement form is 11110010.

Adding +46 and −14, we get

$$\begin{array}{r} \overset{\text{carry}}{\overset{\curvearrowleft}{\textcircled{1}}} \\ 0\ 0\ 1\ 0\ 1\ 1\ 1\ 0 \\ +\ 1\ 1\ 1\ 1\ 0\ 0\ 1\ 0 \\ \hline 1\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0 \end{array}$$

There is a carry. Ignore it. MSB is 0. ∴ The result is positive and is in normal binary form.

∴ The result is $0\ 0\ 1\ 0\ 0\ 0\ 0\ 0 = +32$

Q.9 Add −75 to +26 using the 8 bit 2's complement arithmetic

Soln// $+75 = 0\ 1\ 0\ 0\ 1\ 0\ 1\ 1$

$+26 = 0\ 0\ 0\ 1\ 1\ 0\ 1\ 0$

2 | 26 → 13-0, 6-1, 3-0, 1-1

2 | 75 → 37-1, 18-1, 9-0, 4-1, 2-0, 1-0

-75 in 1's complement form = 10110100

Adding 1 $\quad = \quad +\ \ 1$

$\quad 10110101$ ⟶ −75 in 2's complement form

Adding −75 to +26 in 2's complement form,

$$\begin{array}{r} 1\ 0\ 1\ 1\ 0\ 1\ 0\ 1 \\ +\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 0 \\ \hline 1\ 1\ 0\ 0\ 1\ 1\ 1\ 1 \end{array}$$

No carry. MSB is 1. The result is negative and is in 2's complement form.

2's complement form : $1\ 1\ 0\ 0\ 1\ 1\ 1\ 1$

1's " " = $0\ 0\ 1\ 1\ 0\ 0\ 0\ 0$

Adding 1 = $\quad +\ \ 1$

$\quad 0\ 0\ 1\ 1\ 0\ 0\ 0\ 1$ ⟶ +49 ⟶ −49

Result

Subtract B from A →

Write A ~~in its 2's com~~ in binary

Write B in 2's complement form.

2's complement of B = − B

Add A and (− B)

2's COMPLEMENT ARITHMETIC

Case 1 : There is a carry. Ignore it.

If MSB is 0, the result is positive & in true form

If MSB is 1, the result is − negative, and in ~~2's true complement~~ 2's complement form.

Case 2 : No carry.

If MSB is 0, the result is positive & in ~~2's complement~~ true form

If MSB is 1, the result is negative & in 2's complement.

Q.10 Subtract 14 from 25 using the 8-bit 1's complement arithmetic

Soln

25 → 0 0 0 1 1 0 0 1

−14 → + 1 1 1 0 0 0 0 1 (in 1's complement form)

(1) 0 0 0 0 1 0 1 0

↳ + 1

0 0 0 0 1 0 1 1 = $+11_{10}$

MSB is 0. The result is positive & in true form

Q.11 Add −25 to −14 using the 8 bit 1's complement method

−25 → 1 1 1 0 0 1 1 0 (in 1's complement form)

−14 → + 1 1 1 1 0 0 0 1

(1) 1 1 0 1 0 1 1 1

↳ + 1

1 1 0 1 1 0 0 0

MSB is 1. The result is negative & in 1's complement form.

1's complement of 11011000 is 00100111 = +39.

∴ The result is −39

Q.12 Add −25 to +14 using 8 bit 1's complement method.

Soln

$+14 \rightarrow \quad 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0$

$-25 \rightarrow \quad +\ 1\ 1\ 1\ 0\ 0\ 1\ 1\ 0$ (in 1's complement form)

$1\ 1\ 1\ 1\ 0\ 1\ 0\ 0$

No carry. MSB = 1. ∴ result is negative & is in its 1's complement.

1's complement of 11110100 is 00001011 = 11

∴ The result is −11.

Q.13 Add +25 to +14 using 8 bit 1's complement method.

Soln

$+25 \rightarrow \quad 0\ 0\ 0\ 1\ 1\ 0\ 0\ 1$

$+14 \rightarrow \quad +\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0$

$0\ 0\ 1\ 0\ 0\ 1\ 1\ 1$

No carry. MSB = 0. ∴ result is positive & in its true form.

Q.14 Add −89.75 to +43.25 using 12 bit 1's complement method

Soln

$+43.25 \rightarrow \quad 0\ 0\ 1\ 0\ 1\ 0\ 1\ 1.\ 0\ 1\ 0\ 0$

$-89.75 \rightarrow \quad +\ 1\ 0\ 1\ 0\ 0\ 1\ 1\ 0.\ 0\ 0\ 1\ 1$ (1's complement form)

$1\ 1\ 0\ 1\ 0\ 0\ 0\ 1.\ 0\ 1\ 1\ 1$

No carry. MSB = 1. ∴ result is negative & in 1's complement form.

1's complement of 11010001.0111 is 00101110.1000 = 46.5

∴ The result is −46.5.

★ 1'S COMPLEMENT ARITHMETIC.

If carry is there → add it

If MSB = 0, result is positive & in true form

If MSB = 1, " " negative & in 1's complement form

If no carry, If MSB = 0, " " positive, & in true form

If MSB = 1, " " negative & " 1's complement form

V.V.I.

* Codes - Group of symbols

Classification of codes

(i) Weighted codes → ex: Binary, 8421, 2421 etc.

a_3 a_2 a_1 a_0 (weights 8, 4, 2, 1)

b_3 b_2 b_1 b_0 (weights 2, 4, 2, 1)

(ii) Non-weighted codes → ex: XS-3, Gray code etc.

(iii) Reflective codes → Self complementing code. Ex: 2421, XS-3

9 → complement of 0

8 → " " 1

7 → " " 2

Decimal	2421 code
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	1 0 1 1
6	1 1 0 0
7	1 1 0 1
8	1 1 1 0
9	1 1 1 1

(iv) Sequential code → Each succeeding code is 1 binary number greater than the preceding code.

Ex: 8421, XS-3 etc.

(v) Alphanumeric code → ex: ASCII code

(vi) Error detecting & correcting codes → Ex: Hamming code.

* Binary Coded Decimal (BCD) code :- (Also called 8-4-2-1 code)

→ In this code, each decimal digit is represented by a 4-bit binary number.

→ Positional weights are 8-4-2-1

r = 10

0 to 9

10 decimal digits

143 → Decimal no.
→ 3 digits

Decimal	BCD 8	4	2	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

BCD code for decimal digits

$2^4 = 16$

10 out of 16

Decimal nos not decimal digits

10	x	x	x	x
11				
12				
13				
14				
15				
16	x	x	x	x

Invalid / Don't comes.

- Conversion of decimal numbers to BCD :

(i) $(17)_{10}$ → 0 0 0 1 0 1 1 1

(ii) $(156)_{10}$ → 0 0 0 1 0 1 0 1 0 1 1 0

- BCD to decimal :

(i) 0001 0100 → 14_{10} (ii) 1001001 → 49_{10}

- Comparison between binary and BCD :

	Binary	BCD
$(10)_{10}$ →	1010	00010000
$(12)_{10}$ →	1100	00010010

* BCD is less efficient than binary as it consumes more bits

* BCD Addition :

① Sum ⩽ 9, Final Carry = 0 Answer is correct

② Sum ⩽ 9, " " = 1 " " incorrect (Add 6 i.e. 0110 to correct)

③ Sum > 9, " " = 0 " " (Add 0110)

Ex 1: $2_{10} + 6_{10}$ BCD addition

Sol'n

```
  0 0 1 0      Sum < 9
+ 0 1 1 0      F.C. = 0
-----------
  1 0 0 0   ← Correct Ans.
```

Ex 2: $3_{10} + 7_{10}$

```
  0 0 1 1      Sum > 9
  0 1 1 1      F.C. = 0
-----------
  1 0 1 0   ← Incorrect Ans
+ 0 1 1 0
-----------
0 0 0 1 0 0 0 0   ← Correct Ans
  ↓       ↓
  1       0
```

Ex 3: $8_{10} + 9_{10}$ BCD addition

Sol'n

```
  1 0 0 0      Sum < 9
+ 1 0 0 1      F.C. = 01
-----------
1 0 0 0 1   ← Incorrect Ans
+ 0 1 1 0
-----------
0 0 0 1 0 1 1 1   ← Correct Ans
  ↓   ↓
  1   7
```

Ex 4: $57_{10} + 26_{10}$

```
  0 1 0 1 0 1 1 1
+ 0 0 1 0 0 1 1 0
-----------------
  0 1 1 1 1 1 0 1
    7 < 9 → Case ①     13 > 9 → Case ②
F.C. = 0                 0 1 1 0
-----------------------------------
  1 0 0 0 0 0 1 1   = 83
    8               3
  Correct           Correct
```

* 2-4-2-1 Code → BCD code

Preferred set

Not preferred

Self complementing property of 2-4-2-1

Decimal Digit	Set 1 (×) 2 4 2 1	Set 2 2 4 2 1
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1
2	0 0 1 0	0 0 1 0
3	0 0 1 1	0 0 1 1
4	0 1 0 0	0 1 0 0
5	0 1 0 1	1 0 1 1
6	0 1 1 0	1 1 0 0
7	0 1 1 1	1 1 0 1
8	1 1 1 0	1 1 1 0
9	1 1 1 1	1 1 1 1

* Excess - 3 code :

Decimal ⟶ 8421 code (BCD code) $\xrightarrow{\text{Add } 0011 \ (3)}$ Excess - 3

5 ⟶ 0101 $\xrightarrow{+0011}$ 1000 (8)

XS-3 code or X-3 code

- XS - 3 code is unweighted code
- It is 4 bit code

Decimal	BCD	XS - 3
0	0 0 0 0	0 0 1 1
1	0 0 0 1	0 1 0 0
2	0 0 1 0	0 1 0 1
3	0 0 1 1	0 1 1 0
4	0 1 0 0	0 1 1 1
5	0 1 0 1	1 0 0 0
6	0 1 1 0	1 0 0 1
7	0 1 1 1	1 0 1 0
8	1 0 0 0	1 0 1 1
9	1 0 0 1	1 1 0 0

- XS - 3 code for decimal nos. :-

(i) 24 ⟶ 0 0 1 0 0100 (BCD) $\xrightarrow{+0011}$ 0101 0111 (2, 4)

\+ 0 0 1 1 0 0 1 1

0 1 0 1 0 1 1 1

→ XS - 3 code is the only unweighted code which is self-complementing

* Gray Code :

→ Also known as Reflected Binary code

→ Unweighted code 'Cyclic code'

→ Unit distance code and minimum error code.

→ Two successive values ~~difference~~ difference is only 1 bit.

→ Binary number is converted to Gray code to reduce switching operation.

Decimal	Binary	Gray Code
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1
2	0 0 1 0	0 0 1 1
3	0 0 1 1	0 0 1 0
4	0 1 0 0	0 1 1 0
5	0 1 0 1	0 1 1 1
6	0 1 1 0	0 1 0 1
7	0 1 1 1	0 1 0 0
8	1 0 0 0	1 1 0 0
9	1 0 0 1	1 1 0 1
10	1 0 1 0	1 1 1 1
11	1 0 1 1	1 1 1 0
12	1 1 0 0	1 0 1 0
13	1 1 0 1	1 0 1 1
14	1 1 1 0	1 0 0 1

Decimal	Binary	Gray Code
15	1 1 1 1	1 0 0 0
16	1 0 0 0 0	0 0 0 0

* Binary to Gray Code conversion :-

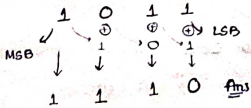
Step 1 → Record the MSB as it is.

Step 2 → Add the MSB to the next bit, record the sum and neglect the carry. (X-OR)

Step 3 → Repeat the process.

Ex 1: Convert 1011 to Gray code.

Sol'n

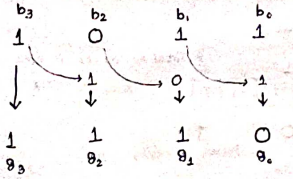


Ans

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

In Binary ↔ Gray Code, MSB remains same.

XOR → Odd 1's detector



$g_3 = b_3$

$g_2 = b_2 \oplus b_3$

$g_1 = b_1 \oplus b_2$

$g_0 = b_0 \oplus b_1$

Ex 2 : Convert 1110 to Gray code

Sol'n 1001 → Ans

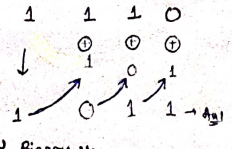
* Gray Code to binary conversion :-

Step 1 : Record the MSB as it is

Step 2 : Add MSB to the next bit of Gray code, record the sum & neglect the carry

Step 3 : Repeat the process by adding the previous resultant to next bit.

Ex 1:



X-OR

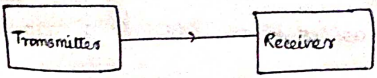
Ex 2: 1 0 0 1

Sol'n 1 1 1 0

* Parity :

→ It is concept to detect errors

→ A single bit error is detected by it.



4 bit signal

We send a bit stream and also an extra bit. This extra bit tells us about the total no. of 1's in the transmitted signal.

Two types of parity → i) Even ii) Odd

In even parity, there is overall even number of 1.

In odd " " " odd " " 1.

Even parity →

Original Signal				Parity bit	
0	1	0	0	1	} Total no. of 1s is even
1	1	0	0	0	} "

Odd parity →

0	1	0	0	0	} Total no. of 1s is odd
1	1	0	0	1	

0 1 0 0 1 ——→ 0 1 0 1 1

Noise

Transmitted

Even no. of 1 — Odd no. of 1 → ∴ Error detected

* Hamming Code – Error detection :

V.V.I.

→ Given by R W. Hamming

→ Easy to implement

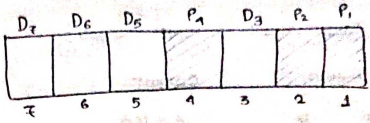
→ 7-bit hamming code is used commonly.

Data bits → 4

Parity bits → 3

→ 2^n where $n = 0, 1, 2, 3, \ldots$

$2^0 = 1$
$2^1 = 2$
$2^2 = 4$
$2^3 = 8$



Hamming Code

If we are working with even parity, and D_3 D_5 D_7 corresponds to bits 0, 1, 0 respectively.

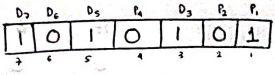
Then to make the total number of 1's even, as P_1 corresponds to bit 1.

$P_1 \rightarrow D_3\ D_5\ D_7$

$P_2 \rightarrow D_3\ D_6\ D_7$

$P_4 \rightarrow D_5\ D_6\ D_7$

• 1 0 1 1 →



For even parity, $P_1 = 1$ (1 1 1 — D_3 D_5 D_7)

$P_2 = 0$ (1 0 1 — D_3 D_6 D_7)

$P_4 = 0$ (1 0 1 — D_5 D_6 D_7)

1 0 1 0 1 0 1

Let, noise changes the transmitted bit stream to 1 1 1 0 1 0 1 in channel.

Now, receiver will check the parity bits.

$P_1 = 1$ (1 1 1) → No error — D_3 D_5 D_7 Correct

$P_2 = 0$ (1 1 1) → Error — D_6 Incorrect

$P_4 = 0$ (1 1 1) → Error — D_6 Incorrect

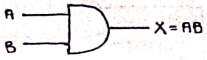
∴ The 6th bit is having error.

* Logic gates : (Study from mam's notes)

Basic gates → AND, OR and NOT

Universal gates → NAND, NOR

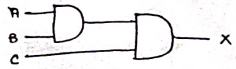
- **AND gate :**



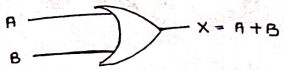
Truth table : [2-input AND gate]

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

3-input AND gate :



- **OR gate :**



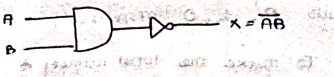
Inputs		Output
A	B	X = A + B
0	0	0
0	1	1
1	0	1
1	1	1

- **NOT gate :**

Truth table :

Inputs	Output
A	X = Ā
0	1
1	0

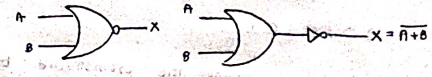
- **NAND gate :**



Truth table :

Inputs		Output
A	B	X = $\overline{AB}$
0	0	1
0	1	1
1	0	1
1	1	0

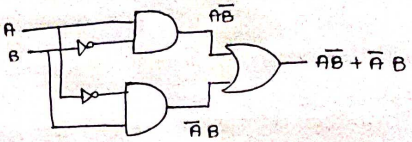
- **NOR gate :**



Inputs		Output
A	B	X = $\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

- **XOR gate :**

The exclusive OR gate gives high output when the inputs are not at equal logic level.



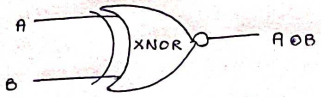
$A \oplus B = A\bar{B} + \bar{A}B$

Truth table:

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

- **XNOR gate :**

The exclusive NOR gate gives high output when the inputs are at equal logic level.



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

* Boolean Algebra :- (Boolean algebra is explained in Anand Kumar in better way)

* Introduction to Boolean Algebra :-

→ It is the set of rules used to simplify the given logic expression without changing its functionality.

→ It is used when number of variables are less.

Rules :

(i) Complement :

A complement = $\bar{A}$ or A' or (not A)

- $(A)' = A$

(ii) AND :

- $A \cdot A = A$
- $A \cdot 0 = 0$
- $A \cdot 1 = A$
- $A \cdot A' = 0$

(iii) OR :

- $A + A = A$
- $A + 0 = A$
- $A + 1 = 1$
- $A + A' = 1$

(iv) Distributive law :

- $A \cdot (B + C) = A \cdot B + A \cdot C$
- $A + (B \cdot C) = (A + B) \cdot (A + C)$

(v) Commutative law :

- $A \cdot B = B \cdot A$
- $A + B = B + A$

(vi) Associative law :

- $(A \cdot B) \cdot C = A \cdot (B \cdot C)$

(vii) De Morgan's law : - $\overline{A+B} = A' \cdot B'$ - $\overline{A \cdot B} = A' + B'$

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

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

Priority :
NOT > AND > OR

Study De Morgan's law in detail from somewhere else

* Boolean Algebra examples :

① $AB + AB'$

Sol^n $F = A \cdot B + A \cdot B'$

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

$= A \cdot 1$

$= A$

② $AB + AB'C + AB'C'$

Sol^n $F = A \cdot B + A \cdot B' \cdot C + A \cdot B' \cdot C'$

$= A \cdot [B + B'C + B' \cdot C']$

$= A \cdot [B + C + B'C']$

$= A \cdot [B + B'C' + C]$

$= A \cdot [B + C' + C]$

$= A \cdot [B + 1]$

$= A \cdot 1$

$= A$

③ $Y = A \cdot B + A' \cdot C + B \cdot C$

~~$= A \cdot B + (A' + B) \cdot C$~~

~~$= B(A + C) + A' \cdot C$~~

~~$= A \cdot B + (A$~~

$= A \cdot B + A' \cdot C + (A + A') B \cdot C$

$= A \cdot B + A' \cdot C + ABC + A'BC$

$= AB(1 + C) + A'C(1 + B)$

$= AB + A'C$

④ $F = (A + B + C)(A + B' + C)(A + B + C')$

$\Rightarrow$ let $X = A + B$

$F = (X + C)(A + B' + C)(X + C')$

$= [X + (C \cdot C')](A + B' + C)$

$= X \cdot (A + B' + C)$

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

$= A + (B \cdot (B' + C))$

$= A + (BB' + BC)$

$= A + (0 + BC)$

$= A + BC$

⑤ $G = (A + B)(A + B')(A' + B)(A' + B')$

$= [A + (BB')][A' + (BB')]$

$= (A + 0)(A' + 0)$

$= AA'$

$= 0$

* Redundancy Theorem :

(i) Three variables

(ii) Each variable is repeated twice

(iii) One variable is complimented

(iv) Take the complimented variable.

⊙ $Y = AB + A'C + BC$

All the above 3 conditions are followed.

The last term is redundant.

$\therefore Y = AB + A'C$

⊙ $F = AB + B\bar{C} + AC$

$= B\bar{C} + AC$ [By redundancy theorem]

⊙ $F = A\bar{B} + B\bar{C} + \bar{A}C$

$= A\bar{B} + BC$

⊙ $(A + B)(\bar{A} + C)(B + C)$

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

⊙ $G = (A + B) \cdot (\bar{B} + C) \cdot (A + C)$

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

⊙ $F = \bar{A}\bar{B} + A\bar{C} + \bar{B}\bar{C}$

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

When all the variables are complemented, then we check which variable is complemented once.

These are beginner level Boolean algebra questions.
Practice some more questions on
"Minimization of boolean expression"
from Anand Kumar's book.
In exam, one question of 4 marks will come from this topic

* Sum of product : (SOP form)

SOP form
POS form

	A	B	C	F
0 m_0	0	0	0	0
1 m_1	0	0	1	0
2 m_2	0	1	0	1
3 m_3	0	1	1	0
4 m_4	1	0	0	1
5 m_5	1	0	1	1
6 m_6	1	1	0	1
7 m_7	1	1	1	1

Total no. of combinations
$= 2 \times 2 \times 2$
$= 8$

$F(A,B,C) = m_2 + m_4 + m_5 + m_6 + m_7$
$F(A,B,C) = \Sigma m(2,4,5,6,7)$

SOP form

$F = \bar{A}\cdot B\cdot\bar{C} + A\cdot\bar{B}\cdot\bar{C} + A\cdot\bar{B}\cdot C + A\cdot B\cdot\bar{C} + A\cdot B\cdot C$
(m_2, m_4, m_5, m_6, m_7)

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

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

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

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

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

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

$F = A + B\bar{C}$ → Minimal SOP form

Standard or canonical SOP form (written directly from truth table)

minterm: $\bar{A}\cdot B\cdot C$, $A\bar{B}C$, AB, $A\bar{C}$ etc.

maxterm → $A+B+C$

minterm → m

Maxterm → M

These SOP, POS topics are not written in syllabus directly.
But these are very important for exam.

- Canonical or standard SOP form :

Each minterm is having all the variables in normal or complimented form

Ex : $F = \overline{A}B + AB + \overline{A}\overline{B}$

- Minimal SOP form :

Each minterm does not have all the variables in normal or complimented form.

Ex : $G = A + \overline{B}C$

Q. For the given truth table, minimize the SOP expression.

A	B	Y
0	0	0
0	1	1
1	0	0
1	1	1

Sol// $Y = \overline{A} \cdot B + AB$ ← Canonical/Standard SOP form

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

$= B \cdot 1$

$Y = B$ ← minimal SOP form

Q. Simplify the expression for $Y(A, B) = \Sigma m(0, 2, 3)$

Sol// $Y = m_0 + m_2 + m_3$

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

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

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

$= \overline{B} + AB$

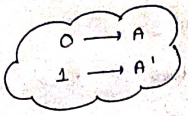
$= (\overline{B} + A)(\overline{B} + B)$

$Y = A + \overline{B}$ ⟶ minimal SOP form

* Product of sum form : (POS form)

A	B	C	Y (SOP / POS)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

POS form is used when the output is "0"



Canonical / standard POS form.

$Y = \underbrace{(A+B+C)}_{x} \cdot \underbrace{(A+B+\bar{C})}_{x} \cdot (A+\bar{B}+\bar{C})$ → Maxterms

$= [(A+B)+(C\cdot\bar{C})] \cdot (A+\bar{B}+\bar{C})$

$(x+c)(x+\bar{c}) = x+(c\bar{c})$

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

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

$= A+[(B\cdot\bar{B})+(B\cdot\bar{C})]$

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

$= (A+B)\cdot(A+\bar{C})$ → minimal POS form

Q. For the given truth table minimize the POS expression.

	A	B	Y
M_0	0	0	1
M_1	0	1	0
M_2	1	0	1
M_3	1	1	0

Soln

$Y = (A+\bar{B})(\bar{A}+\bar{B})$ → SPOS

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

$= \bar{B}+0$

$= \bar{B}$ → minimal form

* SOP & POS form Example :

A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

SOP form :

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

$= [\bar{A}\cdot\bar{C}(\bar{B}+B)] + [B\cdot C(\bar{A}+A)] + (A\cdot B\cdot\bar{C})$

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

$= (\bar{A}\cdot\bar{C}) + [B(C + A\bar{C})]$

$= (\bar{A}\cdot\bar{C}) + (BC + BA)$

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

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

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

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

$\boxed{Y = \bar{A}C + B}$

POS form :

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

$= (A+B+\bar{C})\cdot[(\bar{A}+B) + (C\cdot\bar{C})]$

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

$= B + [(A+\bar{C})\cdot\bar{A}]$

$= B + [A\cdot\bar{A} + \bar{A}\cdot\bar{C}]$

$= B + \bar{A}\cdot\bar{C}$ ⟶ *** This is not maxterm

$\boxed{Y = (\bar{A}+B)(B+\bar{C})}$ ⟶ Minimal POS form

* Minimal to Canonical form conversion :

- **SOP form :**

$Y = A + B'C$ → minimal SOP form

Step 1 : Count the no. of variables

Step 2 : Find out the missing variables in each minterm.

Step 3 : For each missing variable in a minterm, multiply it with 1 and expand that 1.

$Y = A + B'C$

(i) 3 variables → A, B, C

(ii) m_1 ⟶ B & C are missing

m_2 ⟶ A is missing

(iii) $Y = A \cdot 1 \cdot 1 + B'C \cdot 1$

$= A \cdot (B + B') \cdot (C + C') + B'C \cdot (A + A')$

$= (A \cdot B + A \cdot B') \cdot (C + C') + B'CA + B'CA'$

$Y = ABC + ABC' + AB'C + AB'C' + B'CA + B'CA'$

Redundant → remove

$\Rightarrow Y = ABC + ABC' + AB'C + AB'C' + AB'C$ Ans

IMPORTANT !!!

- **POS form :**

$F = (A + B + C')(A' + C)$

Step 1 : Same as that of in SOP form

Step 2 :

Step 3 : For each missing variable in a maxterm, OR it with 0 and expand that 0.

$F = (A + B + C')(A' + C)$

(i) 3 variables → A, B, C

(iii) $M_1 \rightarrow$ no variable missing

$M_2 \rightarrow$ B missing

(iii) $F = (A + B + C')(A' + C)$

$= (A + B + C')(A' + C + 0)$

$= (A + B + C')(A' + C + (B \cdot B'))$

$F = (A + B + C')(A' + C + B)(A' + C + B')$ Ans

LEFT OUT TOPICS....

I found these topics in book (haven't studied).

So I am adding them here. No need to study in that much detail.

Generally short notes comes from these topics

1.2 DIGITAL SIGNALS

As mentioned above, a digital signal has two discrete levels or values. Two different representations of digital signals are shown in Fig. 1.1. In each case there are two discrete levels. These levels can be represented using the terms LOW and HIGH. In Fig. 1.1*a*, lower of the two levels has been designated as LOW level and the higher as HIGH level. In contrast to this, in Fig. 1.1*b*, higher of the two levels has been designated as LOW level and the lower as HIGH level. Digital systems using the representation of signal shown in Fig. 1.1*a* are said to employ *positive logic system* and those using the other representation of the signal shown in Fig. 1.1*b* are said to employ *negative logic system.* The genesis of the term *logic* is given later. In each of the two signals we observe that the voltage corresponding to a given level is not fixed, rather voltages in a limited range are designated as a level. As long as the voltage belongs to a level it will be taken as that level and the exact value of the voltage is immaterial. For example, any voltage in the range of 3.5 to 5 V will be considered as HIGH level in the positive logic system and LOW level in the negative logic system. Similarly, any voltage in the range of 0 to 1 V will be considered as LOW level in the positive logic system and HIGH level in the negative logic system. The actual voltage ranges corresponding to LOW and HIGH level are not same for all types of circuits and are different for different logic families (see Chapter 4). Unless otherwise specified, we shall be dealing with positive logic system.

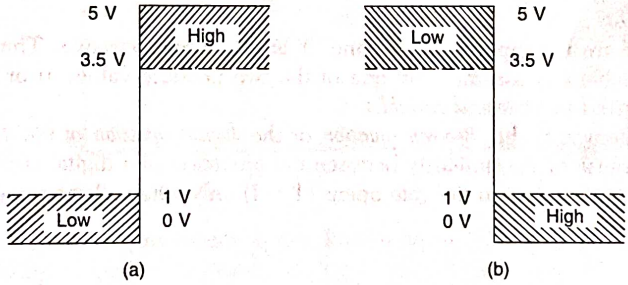


Fig. 1.1
Digital signal representation (a) Positive logic (b) Negative logic.

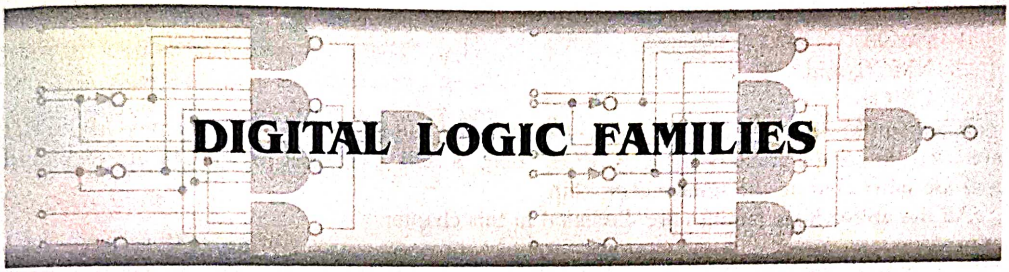
The above discussion brings out one of the main advantages of digital systems, viz. they are less susceptible to noise, fluctuations in the characteristics of components, etc.

The two discrete signal levels HIGH and LOW can also be represented by the *binary digits* 1 and 0 respectively. A *binary digit* (0 or 1) is referred to as a *bit.* Since a digital signal can have only one of the two possible levels 1 or 0, the *binary number system* (see Chapter 2) can be used for the analysis and design of digital systems. The two levels (or states) can also be designated as ON and OFF or TRUE and FALSE. George Boole introduced the concept of binary number system in the studies of the mathematical theory of LOGIC in the work entitled *An Investigation of the Laws of Thought* in 1854 and developed its algebra known as *Boolean algebra.* These logic concepts have been adapted for the design of digital hardware since 1938 when Claude Shannon-organised and systematized Boole's work in *Symbolic Analysis of Relay and Switching Circuits.*

1.3 BASIC DIGITAL CIRCUITS

In a digital system there are only a few basic operations performed, irrespective of the complexities of the system. These operations may be required to be performed a number of times in a large digital system like digital computer or a digital control system, etc. The basic operations are AND, OR, NOT, and FLIP-FLOP. The AND, OR, and NOT operations are discussed here and the FLIP-FLOP, which is a basic memory element used to store binary information (one bit is stored in one FLIP-FLOP), will be introduced in Chapter 7.

4



DIGITAL LOGIC FAMILIES

4.1 INTRODUCTION

The switching characteristics of semiconductor devices have been discussed in Chapter 3. Basically, there are two types of semiconductor devices: bipolar and unipolar. Based on these devices, digital integrated circuits have been made which are commercially available. Various digital functions are being fabricated in a variety of forms using bipolar and unipolar technologies. A group of compatible ICs with the same logic levels and supply voltages for performing various logic functions have been fabricated using a specific circuit configuration which is referred to as a *logic family.*

4.1.1 Bipolar Logic Families

The main elements of a bipolar IC are resistors, diodes (which are also capacitors) and transistors. Basically, there are two types of operations in bipolar ICs:

1. Saturated, and
2. Non-saturated.

In saturated logic, the transistors in the IC are driven to saturation, whereas in the case of non-saturated logic, the transistors are not driven into saturation.

The saturated bipolar logic families are:

1. Resistor–transistor logic (RTL),
2. Direct–coupled transistor logic (DCTL),
3. Integrated–injection logic (I^2L),
4. Diode–transistor logic (DTL),
5. High–threshold logic (HTL), and
6. Transistor-transistor logic (TTL).

The non-saturated bipolar logic families are:
1. Schottky TTL, and
2. Emitter-coupled logic (ECL).

4.1.2 Unipolar Logic Families

MOS devices are unipolar devices and only MOSFETs are employed in MOS logic circuits. The MOS logic families are:
1. PMOS,
2. NMOS, and
3. CMOS

While in PMOS only *p*-channel MOSFETs are used and in NMOS only *n*-channel MOSFETs are used, in complementary MOS (CMOS), both *p*- and *n*-channel MOSFETs are employed and are fabricated on the same silicon chip.

All the above logic families are discussed in this chapter.

4.2 CHARACTERISTICS OF DIGITAL ICs

With the widespread use of ICs in digital systems and with the development of various technologies for the fabrication of ICs, it has become necessary to be familiar with the characteristics of IC logic families and their relative advantages and disadvantages. Digital ICs are classified either according to the complexity of the circuit, as the relative number of individual basic gates (2-input NAND gates) it would require to build the circuit to accomplish the same logic function or the number of components fabricated on the chip. The classification of digital ICs is given in Table 4.1.

Table 4.1 Classification of digital ICs

IC Classification	*Equivalent individual basic gates*	*Number of components*
Small-scale integration (SSI)	Less than 12	Up to 99
Medium-scale integration (MSI)	12–99	100–999
Large-scale integration (LSI)	100–999	1,000–9,999
Very large-scale integration (VLSI)	Above 1,000	Above 10,000

The various characteristics of digital ICs used to compare their performances are:
1. Speed of operation,
2. Power dissipation,
3. Figure of merit,
4. Fan-out,
5. Current and voltage parameters,
6. Noise immunity,
7. Operating temperature range,
8. Power supply requirements, and
9. Flexibilities available.

4.2.1 Speed of Operation

The speed of a digital circuit is specified in terms of the propagation delay time. The input and output waveforms of a logic gate are shown in Fig. 4.1. The delay times are measured between the 50 per cent voltage levels of input and output waveforms. There are two delay times: t_{pHL}, when the output goes from the HIGH state to the LOW state and t_{pLH}, corresponding to the output making a transition from the LOW state to the HIGH state. The propagation delay time of the logic gate is taken as the average of these two delay times.

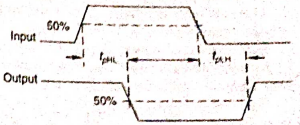


Fig. 4.1 Input and output voltage waveforms to define propagation delay times.

4.2.2 Power Dissipation

This is the amount of power dissipated in an IC. It is determined by the current, I_{CC}, that it draws from the V_{CC} supply, and is given by $V_{CC} \times I_{CC}$. I_{CC} is the average value of $I_{CC}(0)$ and $I_{CC}(1)$. This power is specified in milliwatts.

4.2.3 Figure of Merit

The figure of merit of a digital IC is defined as the product of speed and power. The speed is specified in terms of propagation delay time expressed in nanoseconds.

Figure of merit = propagation delay time (*ns*) × power (mW)

It is specified in pico joules (*ns* × mW = pJ)

A low value of speed-power product is desirable. In a digital circuit, if it is desired to have high speed, i.e. low propagation delay, then there is a corresponding increase in the power dissipation and vice-versa.

4.2.4 Fan-Out

This is the number of similar gates which can be driven by a gate. High fan-out is advantageous because it reduces the need for additional drivers to drive more gates.

4.2.5 Current and Voltage Parameters

The following currents and voltages are specified which are very useful in the design of digital systems.

High-level input voltage, V_{IH}: This is the minimum input voltage which is recognized by the gate as logic 1.

Low-level input voltage, V_{IL}: This is the maximum input voltage which is recognized by the gate as logic 0.

High-level output voltage, V_{OH}: This is the minimum voltage available at the output corresponding to logic 1.

Low-level output voltage, V_{OL}: This is the maximum voltage available at the output corresponding to logic 0.

High-level input current, I_{IH}: This is the minimum current which must be supplied by a driving source corresponding to 1 level voltage.

Low-level input current, I_{IL}: This is the minimum current which must be supplied by a driving source corresponding to 0 level voltage.

High-level output current, I_{OH}: This is the maximum current which the gate can sink in 1 level.

Low-level output current, I_{OL}: This is the maximum current which the gate can sink in 0 level.

High-level supply current, I_{CC} (1): This is the supply current when the output of the gate is at logic 1.

Low-level supply current, I_{CC} (0): This is the supply current when the output of the gate is at logic (0).

The current directions are illustrated in Fig. 4.2.

Fig. 4.2
A gate with current directions marked.

4.2.6 Noise Immunity

The input and output voltage levels defined above are shown in Fig. 4.3. Stray electric and magnetic fields may induce unwanted voltages, known as *noise*, on the connecting wires between logic circuits. This may cause the voltage at the input to a logic circuit to drop below V_{IH} or rise above V_{IL} and may produce undesired operation. The circuit's ability to tolerate noise signals is referred to as the *noise immunity*, a quantitative measure of which is called *noise margin*. Noise margins are illustrated in Fig. 4.3.

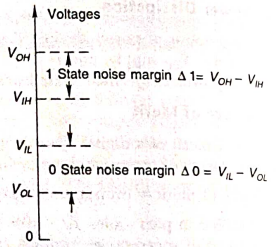


Fig. 4.3
Voltage levels and noise margins of ICs.

The noise margins defined above are referred to as *dc noise margins*. Strictly speaking, the noise is generally thought of as an a.c. signal with amplitude and pulse width. For high speed ICs, a pulse width of a few microseconds is extremely long in comparison to the propagation delay time of the circuit and therefore, may be treated as d.c. as far as the response of the logic circuit is concerned. As the noise pulse width decreases and approaches the propagation delay time of the circuit, the pulse duration is too short for the circuit to respond. Under this condition, a large pulse amplitude would be required to produce a change in the circuit output. This means that a logic circuit can effectively tolerate a large noise amplitude if the noise is of a very short duration. This is referred to as *ac noise margin* and is substantially greater than the dc noise margin. It is generally supplied by the manufacturers in the form of a curve between noise margin and noise pulse width.

4.2.7 Operating Temperature

The temperature range in which an IC functions properly must be known. The accepted temperature ranges are: 0 to + 70 °C for consumer and industrial applications and −55 °C to + 125 °C for military purposes.

4.2.8 Power Supply Requirements

The supply voltage(s) and the amount of power required by an IC are important characteristics required to choose the proper power supply.

4.2.9 Flexibilities Available

Various flexibilities are available in different IC logic families and these must be considered while selecting a logic family for a particular job. Some of the flexibilities available are:

1. *The breadth of the series:* Type of different logic functions available in the series.
2. *Popularity of the series:* The cost of manufacturing depends upon the number of ICs manufactured. When a large number of ICs of one type are manufactured, the cost per function will be very small and it will be easily available because of multiple sources.
3. *Wired-logic capability:* The outputs can be connected together to perform additional logic without any extra hardware.
4. *Availability of complement outputs:* This eliminates the need for additional inverters.
5. *Type of output:* Passive pull-up, active pull-up, open-collector/drain, and tristate. These will be explained in subsequent sections.

4.3 RESISTOR–TRANSISTOR LOGIC (RTL)

The resistor–transistor logic was the most popular form of logic in common use [illegible] ent of ICs. RTL circuits consist of resistors and transistors and wa[illegible]grated. Although RTL has become obsolete now, beca[illegible] and fo[illegible] it is proper to devote some attention to [illegible]me of the importa[illegible] for all types of gates, through thi[illegible]gate is a NOR gate as shown in Fig[illegible]ake of simplicity, a two-i[illegible]ving N similar gates is shown in the figure, [illegible]e extended to ac[illegible]arger number of inputs. The number of input termi[illegible] to as the

4.3.1 Logic Operation

Inputs representing the logic leve[illegible] A and B terminals. The voltage correspond-ing to LOW level should be lo[illegible]ve the corresponding transistor to cut-off. Similarly, the input voltage co[illegible]el should be high enough to drive the corresponding transistor to [illegible]

If both the inputs are [illegible]sistors T_1 a[illegible] and the output is HIGH. A HIGH level on any [illegible]rive the correspond[illegible]o saturation causing the output to go LOW [illegible](0) level output voltage is V_{CE}[illegible]tor (~ 0.2 V) and the HIGH (1) level [illegible]age depends on the number of gate[illegible] the output. This causes the ou[illegible]e to be variable and is a deciding factor fo[illegible] of the gate.

4.3.2 I[illegible]nsiderations

If all [illegible] to the gate are LOW, the output is HIGH and if the gate is [illegible] oth[illegible]. no load is connected, the output voltage will be slightly less than V_{CC} [illegible]op across the common collector resistor due to I_{CO} of T_1 and T_2).

Because of its speed limitations, DTL has become outdated and is completely replaced by another logic family referred to as transistor– transistor logic (TTL). The main cause for the speed limitation in DTL is the slow process of removal of stored base charge of the output transistor. For example, in the DTL gate of Fig. 4.12, when T goes from saturation to cut-off, the diodes D_1 and D_2 are nonconducting and hence, the base charge must leak-off through the resistor R_B, which is a relatively slow mechanism.

The DTL speed limitation is overcome by making the following modifications in the circuit of Fig. 4.12:

1. The input diodes D_A, D_B, and D_C are replaced by emitter-base junctions of multiple-emitter transistor (T_1), which is easily and economically fabricated in IC.
2. The collector–base junction of T_1 acts as the diode D_1.
3. The diode D_2 is replaced by emitter–base junction of another transistor (T_2).

The modified circuit is known as TTL and is shown in Fig. 4.16.

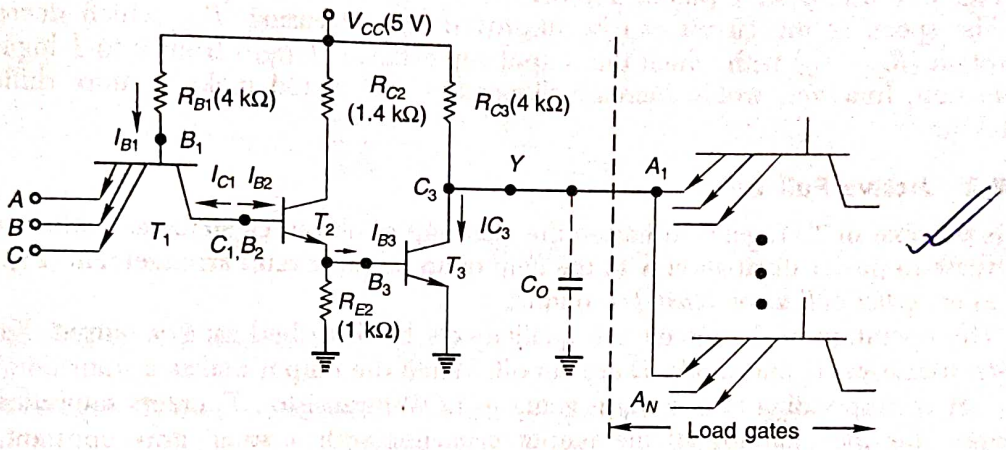


Fig. 4.16

A 3-input TTL NAND gate driving N similar gates.

4.8.1 Operation of TTL NAND Gate

The operation of the TTL gate of Fig. 4.16 is similar to the operation of the DTL gate of Fig. 4.12 as far as the steady-state operation is concerned, as is evident from conditions I and II discussed below. It is condition III that differentiates the operation of TTL from that of DTL and makes it the fastest of all saturating logic families.

For the operation discussed below, we assume that the load gates are not present and the voltages for logic 0 and 1 are $V_{CE,\,sat} \approx 0.2$ V and $V_{CC} = 5$ V respectively.

Condition I At least one input is LOW. The emitter–base junction of T_1 corresponding to the input in the LOW state is forward-biased making voltage at B_1, $V_{B1} = 0.2 + 0.7 = 0.9$ V. For base–collector junction of T_1 to be forward-biased, and for T_2 and T_3 to be conducting, V_{B1} is required to be at least $0.6 + 0.5 + 0.5 = 1.6$ V. Hence, T_2 and T_3 are OFF.

Since T_3 is OFF, therefore $Y = V(1) = V_{CC}$.

Condition II All inputs are HIGH. The emitter–base junctions of T_1 are reverse-biased. If we assume that T_2 and T_3 are ON, then $V_{B2} = V_{C1} = 0.8 + 0.8 = 1.6$ V. Since B_1 is connected to V_{CC} (5 V) through R_{B1}, the collector-base junction of T_1 is forward-biased. The transistor T_1 is operating in the active inverse mode, making I_{C1} flow in the reverse direction. This current flows into the base of T_2 driving T_2 and T_3 into saturation. Therefore, $Y = V(0) \approx 0.2$ V.

From conditions I and II, it appears that T_1 is acting as back-to-back diodes. The importance of T_1 will become clear from condition III.

Condition III Let the circuit be operating under condition II when one of the inputs suddenly goes to $V(0)$. The corresponding emitter–base junction of T_1 starts conducting and V_{B1} drops to 0.9 V. T_2 and T_3 will be turned off when the stored base charge is removed. Since $V_{C1} = V_{B2} = 1.6$ V, therefore the collector–base junction of T_1 is back-biased, making T_1 operate in the normal active region. This large collector current of T_1 is in a direction which helps in the removal of stored base charge in T_2 and T_3 and improves the speed of circuit.

The discussion in Sec. 4.6 regarding loading (fan-out) considerations, noise-margins, average power dissipation, propagation delays, and wired-AND connection, is equally valid for TTL gate of Fig. 4.16 with passive pull-up resistor.

The speed of the circuit can be improved by decreasing R_{C3} which decreases the time constant $(R_{C3} \cdot C_0)$ with which the output capacitance charges from 0 to 1 logic level. Such a reduction, however, would increase dissipation and would make it more difficult for T_3 to saturate.

4.8.2 Active Pull-up

It is possible in TTL gates to hasten the charging of output capacitance without corresponding increase in power dissipation with the help of an output circuit arrangement (Fig. 4.17) referred to as an *active pull-up* or *totem-pole* output.

The operation of the circuit can qualitatively be described as: For output Y to be in LOW state, transistor T_4 and diode D are cut-off. When the output makes a transition from LOW to HIGH corresponding to any input going to LOW, transistor T_4 enters saturation and supplies current for the charging of the output capacitor with a small time constant. This current decreases and eventually becomes zero under steady-state condition when $Y = V(1)$.

Diode D is used in the circuit to keep T_4 in cut-off when the output is at logic 0. Corresponding to this, T_2 and T_3 are in saturation, therefore,

$$V_{C2} = V_{B4} = V_{BE3,\,sat} + V_{CE2,\,sat} = 0.8 + 0.2 = 1.0 \text{ V} \qquad (4.8)$$

Since $V_0 = V_{CE3,\,sat} \approx 0.2$ V, the voltage across the base-emitter junction of T_4 and diode D equals $1.0 - 0.2 = 0.8$ V, which means T_4 and D are cut-off.

If one of the inputs drops to LOW logic level, T_2 and T_3 go to cut-off. The output voltage cannot change instantaneously (being the voltage across C_0) and because of T_2 going to cut-off, the voltage at the base of T_4 rises driving it to saturation.

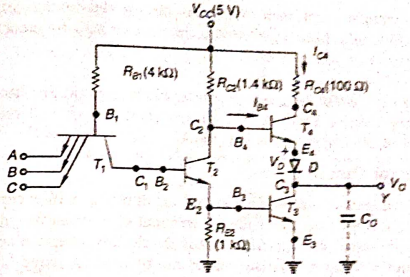


Fig. 4.17
A TTL gate with totem-pole output driver.

As soon as T_2 is cut-off,

$$V_{B4} = V_{BE4,\,sat} + V_D + V_0 = 0.8 + 0.7 + 0.2 = 1.7 \text{ V} \qquad (4.9)$$

Therefore,
$$I_{B4} = \frac{V_{CC} - V_{B4}}{R_{C2}} = \frac{5 - 1.7}{1.4} = 2.36 \text{ mA} \qquad (4.10)$$

and
$$I_{C4} = \frac{V_{CC} - V_{CE4,\,sat} - V_D - V_0}{R_{C4}} = \frac{5 - 0.2 - 0.7 - 0.2}{0.1} = 39 \text{ mA} \qquad (4.11)$$

Hence, T_4 is in saturation if h_{FE} exceeds $\frac{39}{2.36} = 16.5$.

The output voltage V_0 rises exponentially towards V_{CC} with the time constant $= (R_{C4} + R_{CS4} + R_f) C_0$, where R_{CS4} is the saturation resistance of T_4 and R_f is the forward resistance of the diode.

As V_0 increases, the base and collector currents of T_4 are decreased and eventually T_4 just comes out of conduction at steady-state. Therefore,

$$V(1) = V_{CC} - V_\gamma(T_4) - V_\gamma(\text{diode}) = 5 - 0.5 - 0.6 = 3.9 \text{ V}$$

Now, if the output is at $V(1)$ and all the inputs go to HIGH, T_2 goes ON. Consequently T_4 and D go OFF and T_3 conducts. The capacitor C_0 discharges through T_3 and as V_0 approaches $V(0)$, T_3 enters into saturation.

From the above discussion it is clear that the maximum current is drawn from the supply when the output makes a transition from $V(0)$ to $V(1)$ and equals $I_{C4} + I_{B4} = 39 + 2.4 = 41.4$ mA.

This current spike generates noise in the power supply distribution system and increase power dissipation in the gate, more so when it is operated at high frequencies.

4.8.3 Wired-AND

Wired-AND connection must not be used for totem-pole output circuits because of the current spike problem discussed above (Prob. 4.16). TTL circuits with open-collector outputs are available which can be used for wired-AND connections.

4.8.4 Open-Collector Output

A circuit with open-collector output is same as the circuit of Fig. 4.16 except for the collector-resistor R_{C3} of T_3 which is missing. The collector terminal C_3 is available outside the IC and the passive pull-up is to be connected externally. Naturally, the advantages of active pull-up are not available in this. Gates with open-collector output can be used for wired-AND operation (Prob. 4.18).

4.8.5 Unconnected Inputs

If any input of a TTL gate is left disconnected (open or floating) the corresponding E–B junction of T_1 will not be forward-biased. Hence, it acts exactly in the same way as if a logical 1 is applied to that input. Therefore, in TTL ICs, all unconnected inputs are treated as logical 1s. However, the unused inputs should either be connected to some used input(s) or returned to V_{CC} through a resistor.

4.8.6 Clamping Diodes

Clamping diodes are commonly used in all TTL gates to suppress the ringing caused from the fast voltage transitions found in TTL. These diodes shown in Fig. 4.18 clamp the negative undershoot at approximately – 0.7 V.

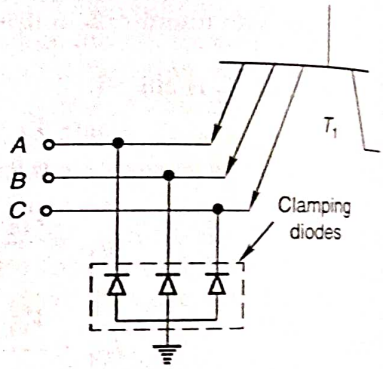


Fig. 4.18
A portion of a TTL gate showing the clamping diodes.

4.9 SCHOTTKY TTL

The speed limitation of TTL is mainly due to the turn-off time delays involved in transistors while making transitions from saturation to cut-off. This can be eliminated by replacing the transistors of TTL gate by Schottky transistors.

With this, the transistors are prevented from entering saturation and hence, there is saving in turn-off time. Schottky TTL gates have propagation delay time of the order of 2 ns which is very small in comparison with the propagation delay time of standard TTL which is of the order of 10 ns. It is a nonsaturating bipolar logic.

4.14 CMOS LOGIC

A complementary MOSFET (CMOS) is obtained by connecting a p-channel and an n-channel MOSFET in series, with drains tied together and the output is taken at the common drain. Input is applied at the common gate formed by connecting the two gates together (Fig. 3.33). In a CMOS, p-channel and n-channel enhancement MOS devices are fabricated on the same chip, which makes its fabrication more complicated and reduces the packing density. But because of negligibly small power consumption, CMOS is ideally suited for battery operated systems.

Its speed is limited by substrate capacitances. To reduce the effect of these substrate capacitances, the latest technology known as silicon on sapphire (SOS) is used in microprocessor fabrication which employs an insulating substrate (sapphire). CMOS has become the most popular in MSI and LSI areas and is the only possible logic for the fabrication of VLSI devices.

4.14.1 CMOS Inverter

The basic CMOS logic circuit is an inverter shown in Fig. 3.33. For this circuit the logic levels are 0 V (logic 0) and V_{CC} (logic 1). When $V_i = V_{CC}$, T_1 turns ON and T_2 turns OFF. Therefore $V_0 \approx 0$ V, and since the transistors are connected in series the current I_D is very small. On the other hand, when $V_i = 0$ V, T_1 turns OFF and T_2 turns ON giving an output voltage $V_0 \approx V_{CC}$ and I_D is again very small. In either logic state, T_1 or T_2 is OFF and the quiscent power dissipation which is the product of the OFF leakage current and V_{CC} is very low. More complex functions can be realized by combinations of inverters.

4.14.2 CMOS NAND and NOR Gates

A 2-input CMOS NAND gate is shown in Fig. 4.27 and NOR gate in Fig. 4.28. In the NAND gate, the NMOS drivers are connected in series, where as the PMOS loads are connected in parallel. On the other hand, the CMOS NOR gate is obtained by connecting the NMOS drivers in parallel and PMOS loads in series. The operation of NAND gate can be understood from Table 4.8. The operation of the NOR gate can be verified in the similar manner (Prob. 4.29).

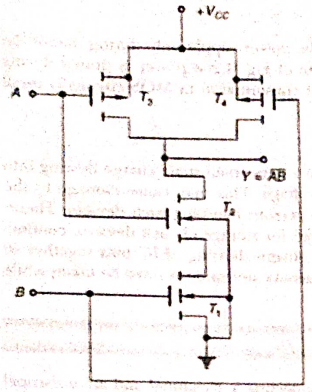


Fig. 4.27
A 2-input CMOS NAND gate.

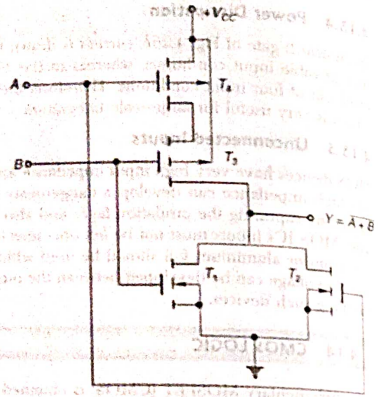


Fig. 4.28
A 2-input CMOS NOR gate.

Table 4.8 Operation of CMOS NAND gate

Inputs		State of MOS devices				Output
A	B	T_1	T_2	T_3	T_4	Y
0	0	OFF	OFF	ON	ON	V_{CC}
0	V_{CC}	ON	OFF	ON	OFF	V_{CC}
V_{CC}	0	OFF	ON	OFF	ON	V_{CC}
V_{CC}	V_{CC}	ON	ON	OFF	OFF	0

4.14.3 CMOS Transmission Gate

A CMOS transmission gate controlled by gate voltages C and $\bar{C}$ is shown in Fig. 4.29. Assume $C = 1$. If $A = V(1)$, then T_1 is OFF and T_2 conducts in the ohmic region because there is no voltage applied at the drain. Therefore, T_2 behaves as a small resistance connecting the output to the input and $B = A = V(1)$. Similarly, if $A = V(0)$, then T_2 is OFF and T_1 conducts, connecting the output to the input and $B = A = V(0)$. This means the signal is transmitted from A to B when $C = 1$.

In a similar manner, it can be shown that if $C = 0$, transmission is not possible.

In this gate the control C is binary, whereas the input at A may be either digital or analog [the instantaneous value must lie between $V(0)$ and $V(1)$].

4.14.4 Noise Margin

Noise margin of CMOS logic ICs is considerably higher than that of TTL ICs. CMOS devices

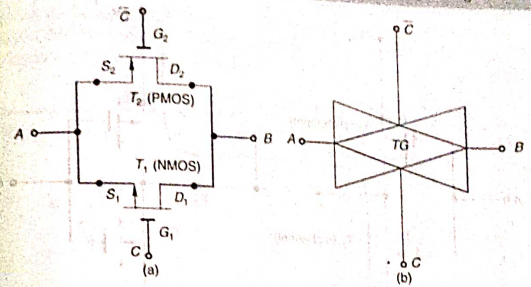


Fig. 4.29
(a) A CMOS transmission gate (b) Its symbol.

have wide supply voltage range and the noise margin increases with the supply voltage V_{CC}. Typically, it is 0.45 V_{CC}.

4.14.5 Unconnected Inputs

The unconnected CMOS ICs inputs behave in a way similar to MOS devices discussed in Section 4.13. Therefore, the unused inputs must be connected to either the supply voltage terminal or one of the used inputs provided that the fan-out of the signal source is not exceeded. This is highly unlikely for CMOS circuits because of their high fan-out.

Some CMOS ICs have Zener diodes connected at the inputs for protection against high input voltages.

4.14.6 Wired-Logic

Figure 4.30 shows two CMOS inverters with their outputs connected together. In this circuit,

(i) When $A = B = V(0)$
T_1 and T_1' are cut-off and $Y = V(1) = V_{CC}$

(ii) When $A = B = V(1)$
T_1 and T_1' are ON and $Y = V(0) = 0$

(iii) When $A = V(1)$ and $B = V(0)$
T_1 and T_2' are ON whereas
T_1' and T_2 are OFF

Therefore, a large current I will flow as shown in Fig. 4.30. This will make voltage at Y equal to $V_{CC}/2$ which is neither in the range of logic 0 nor in the range of logic 1. Therefore, the circuit will not operate properly. Also because of large current I, the transistors will be damaged.

Similarly, corresponding to $A = V(0)$ and $B = V(1)$ the operation will not be proper. Therefore, wired-logic must not be used for CMOS logic circuits.

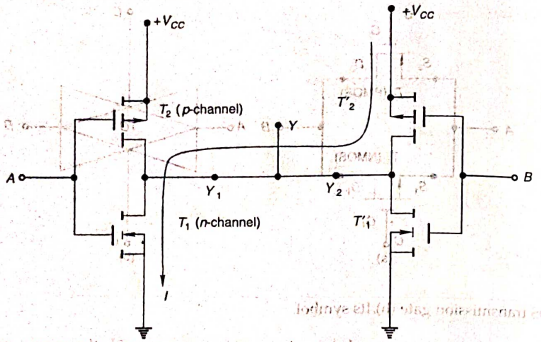


Fig. 4.30
CMOS inverters with outputs connected.

4.14.7 Open-Drain Outputs

CMOS gates with open-drain output are available which are useful for wired-AND operation. In this the drain terminal of the output transistor (*n*-channel) is available outside and the load resistor is to be connected externally since *p*-channel load does not exist.

4.14.8 54C00/74C00 CMOS Series

There are two commonly used CMOS series ICs. These are the 4000 series and 54C/74C series. 54C/74C CMOS series is pin-for-pin, function-for-function equivalent to the 54/74 TTL family and has, therefore, become very popular. The temperature range for 54C series is −55 °C to + 125 °C and for 74C series is −40 °C to 85 °C. It has a wide supply voltage range, 3 V to 15 V. A person can take full advantage of his knowledge of the 54/74 TTL series for the effective use of 54C/74C series.

There have been significant improvements in 54C/74C series. The 74HC/74HCT have higher speed and better current capabilities. 74HC is known as *high-speed* CMOS and 74HCT is known as *high-speed, TTL compatible* CMOS series. 74AC/74ACT are very fast and have very high current sinking capabilities. These are known as *advanced* CMOS and *advanced, TTL compatible* CMOS, respectively. The 74 HC/74HCT/74AC/74ACT series can be operated at supply voltages in the range of 2–6 volts.

The voltage and current parameters of various 74 CMOS series with 5 V supply voltage are given in Table 4.9. From the table, we observe that the output currents and voltages for 74HC/74HCT/74AC/74ACT are different when gates of these series are driving CMOS circuits and TTL circuits. 74 HCT and 74 ACT series are compatible with TTL series for input as well as output and therefore, can easily be used along with TTL ICs for optimum system design from the point of view of speed, power dissipation, noise margins, cost, etc.

The fan-out of 74 HC/74HCT series is 20, whreas for 74AC/74ACT series it is 50 while driving these CMOS series. The fan-out of these gates while driving various TTL series gates can be determined using the specifications of TTL (Table 4.3) and CMOS (Table 4.9).

Table 4.9 Specifications of CMOS IC families

Parameter	*Load*	*74C*	*74HC*	*74HCT*	*74AC*	*74ACT*	*Units*
V_{IH}		3.5	3.85	2.0	3.85	2.0	volts
V_{IL}		1.5	1.35	0.8	1.35	0.8	volts
V_{OH}	CMOS	4.5	4.4	4.4	4.4	4.4	volts
	TTL		3.84	3.84	3.76	3.76	volts
V_{OL}	CMOS	0.5	0.1	0.1	0.1	0.1	volt
	TTL		0.33	0.33	0.37	0.37	volt
I_{IH}		1	1	1	1	1	μA
I_{IL}		− 1	− 1	− 1	− 1	− 1	μA
I_{OH}	CMOS	− 0.1	− 0.02	− 0.02	− 0.05	− 0.05	mA
	TTL		− 4.0	− 4.0	− 24.0	− 24.0	mA
I_{OL}	CMOS	0.36	0.02	0.02	0.05	0.05	mA
	TTL		4.0	4.0	24.0	24.0	mA

4.15 INTERFACING CMOS AND TTL

To achieve optimum performance in a digital system, devices from more than one logic family can be used, taking advantages of the superior characteristics of each family for different parts of the system. For example, CMOS logic ICs can be used in those parts of the system where low power dissipation is required, while TTL can be used in those portions of the system which requires high speed of operation. Also, some functions may be easily available in TTL and others may be available in CMOS. Therefore, it is necessary to examine the interface between TTL and CMOS devices.

The 74C series of CMOS ICs can be operated for any supply voltage in the range of 3 V to 15 V, whereas the 74HC/74HCT/74AC/74ACT series have the supply voltage range of 2 V to 6 V. Since the supply voltage used for all 74 series TTL ICs is 5 V, therefore, it is necessary to operate CMOS devices at + 5 V, to make it compatible with TTL devices.

4.15.1 CMOS Driving TTL

Figure 4.31 shows a CMOS gate driving *N* TTL gates. For such an arrangement to operate properly the following conditions are required to be satisfied,

$$V_{OH}\text{ (CMOS)} \geq V_{IH}\text{ (TTL)} \tag{4.12}$$

$$V_{OL}\text{ (CMOS)} \leq V_{IL}\text{ (TTL)} \tag{4.13}$$

$$-I_{OH}\text{ (CMOS)} \geq NI_{IH}\text{ (TTL)} \tag{4.14}$$

$$I_{OL}\text{ (CMOS)} \geq +NI_{IL}\text{ (TTL)} \tag{4.15}$$

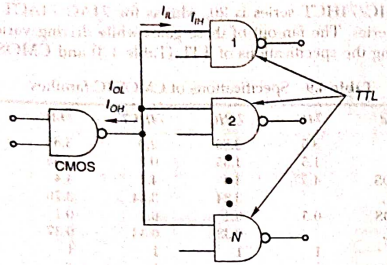


Fig. 4.31
A CMOS gate driving N TTL gates.

From the specifications given in Tables 4.3 and 4.9, we observe the following:

(i) The conditions of Eqs (4.12) and (4.13) are always satisfied. The noise margins when 74ACT is driving 74ALS gates are

$$\Delta 1 = 3.76 - 2.0 = 1.76 \text{ V}$$
$$\Delta 0 = 0.8 - 0.37 = 0.43 \text{ V}$$

(ii) The conditions of Eqs (4.14) and (4.15) are always satisfied for 74 HC/74 HCT/74 AC/74 ACT series. The value of N is different for different series. The value of N when 74 ACT is driving 74ALS gates is 240.

In case of 74 C series, the condition of Eq. (4.14) is satisfied for small values of N but the condition of Eq. (4.15) is not satisfied even for $N = 1$, except in case of 74L and 74ALS TTL series. This difficulty can be overcome by using CMOS buffers having an adequate available output current.

If 74C series gate is driving 74L series gates, the condition of Eq. (4.15) is satisfied for $N = 2$ and in case of 74ALS gates for $N = 3$.

4.15.2 TTL Driving CMOS

Figure 4.32 shows a TTL gate driving N CMOS gates. For such an arrangement to operate properly, the following conditions are required to be satisfied:

$$V_{OH}\text{ (TTL)} \geq V_{IH}\text{ (CMOS)} \quad (4.16)$$
$$V_{OL}\text{ (TTL)} \leq V_{IL}\text{ (CMOS)} \quad (4.17)$$
$$-I_{OH}\text{ (TTL)} \geq NI_{IH}\text{ (CMOS)} \quad (4.18)$$
$$I_{OL}\text{ (TTL)} \geq -NI_{IL}\text{ (CMOS)} \quad (4.19)$$

All the above conditions are always satisfied in case of 74 HCT and 74 ACT series for high values of N. This shows that these two CMOS series are TTL compatible. In the case of 74C/74HC/74AC series, the condition of Eq. (4.16) is not satisfied. A circuit modification used to

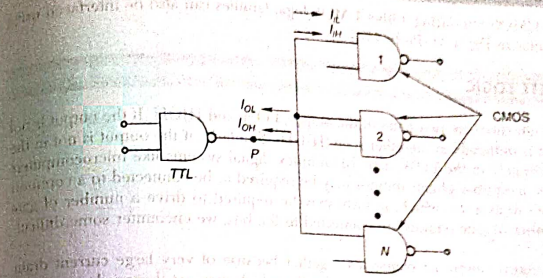


Fig. 4.32
A TTL gate driving N CMOS gates.

raise V_{OH} (TTL) above 3.5 V is obtained by connecting a resistance (≈ 2 K Ω) between points P and V_{CC} as shown in Fig. 4.33. This acts as a passive pull-up, which pulls up the voltage at P, by charging the capacitor C_0 present between P and the ground terminal, to a higher value ($= V_{CC}$) after the transistor T_4 of the TTL becomes non-conducting.

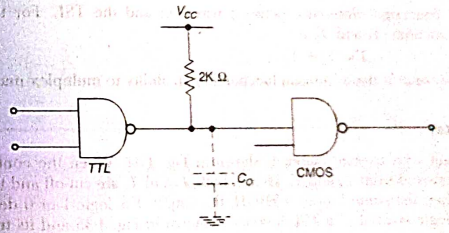


Fig. 4.33
Circuit to pull up the output voltage of TTL.

4.16 INTERFACING CMOS AND ECL

Using MC10H124 TTL-to-ECL translator and MC10H125 ECL-to-TTL translator ICs, it is possible to interface CMOS and ECL logic families. The input of MC10H124 translator is compatible to the output logic voltages of CMOS and therefore, this can be used for CMOs-to-ECL interfacing (Prob. 4.33). Similarly, the output of MC10H125 translator is compatible to the input logic voltages of CMOS (74 HCT and 74 ACT) families which makes it possible to be

used it for ECL-to-CMOS interfacing. Other CMOS logic families can also be interfaced using pull-up resistor similar to Fig. 4.33 (Prob. 4.34).

4.17 TRI-STATE LOGIC

In normal logic circuits there are two states of the output, LOW and HIGH. If the output is not in the LOW state, it is definitely in the other state (HIGH). Similarly, if the output is not in the HIGH state, it is definitely in the LOW state. In complex digital systems like microcomputers and microprocessors, a number of gate outputs may be required to be connected to a common line which is referred to as a *bus* which, in turn, may be required to drive a number of gate inputs. When a number of gate outputs are connected to the bus, we encounter some difficulties. These are:

1. Totem-pole outputs cannot be connected together because of very large current drain from the supply and consequent heating of the ICs which may get damaged.
2. Open-collector outputs can be connected together with a common collector-resistor connected externally. This causes the problems of loading and speed of operation.

To overcome these difficulties, special circuits have been developed in which there is one more state of the output, referred to as the *third state* or *high-impedance state*, in addition to the LOW and HIGH states. These circuits are known as *TRI-STATE, tri-state logic* (TSL) or *three-state logic*. TRI-STATE, is a registered trade mark of National Semiconductor Corporation of USA.

There is a basic functional difference between wired-OR and the TSL. For the wired-OR connection of two functions Y_1 and Y_2 is

$$Y = Y_1 + Y_2 \tag{4.20}$$

whereas for TSL, the result is not a Boolean function but an ability to multiplex many functions economically.

4.17.1 TSL Inverter

A TSL inverter circuit with tri-state output is shown in Fig. 4.34. When the control input is LOW, the drive is removed from T_3 and T_4. Hence, both T_3 and T_4 are cut-off and the output is in the third state. When the control input is HIGH, the output Y is logic 1 or 0 depending on the data input. The logic symbol of a TSL inverter is shown in Fig. 4.35 and its truth table is given in Table 4.10.

Table 4.10 Truth table of a TSL inverter

Data input	Control	Data output
0	0	HIGH – Z
1	0	HIGH – Z
0	1	0
1	1	1

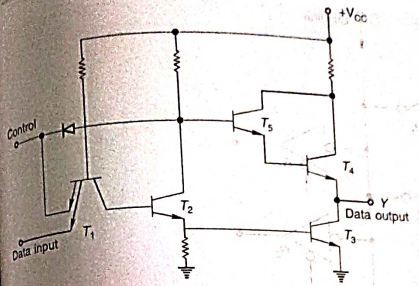


Fig. 4.34
A TSL inverter.

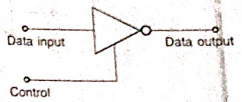


Fig. 4.35
Logic symbol of a TSL inverter.

The output and input current specifications of TSL family are given in Table 4.11.

Table 4.11 Current specifications of TSL family

Parameter	Control input	
	LOW (DISABLE)	*HIGH (ENABLE)*
I_{IH}	40 μA	40 μA
I_{IL}	– 1.6 mA	– 1.6 mA
I_{OH}	40 μA	– 5.2 mA
I_{OL}	– 40 μA	16 mA

Example 4.3 Consider the arrangement shown in Fig. 4.36. At any time one of the gates drives the bus line. Calculate the maximum possible value of N.

Solution
Let I_1 be driving the bus line. All other gates, I_2 through I_N must be tristated.
If the output of I_1 is in logic 1 state, it has to supply leakage current (40 μA) to each of the tri-stated gates and input current to G_1 and G_2 (40 μA). From Table 4.11, we have the maximum possible output current of TSL in logic 1 state as 5.2 mA. Therefore,

$$40(N-1) + 40(2) \le 5.2 \times 10^3$$

or

$$N \le 129$$

which means 129 TSL outputs can be connected to the bus line.

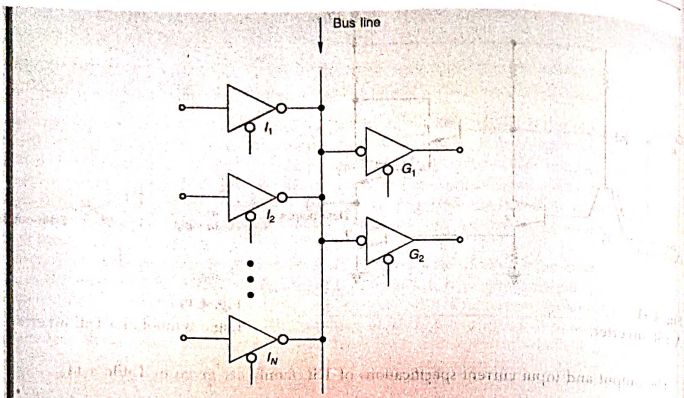


Fig. 4.36
N TSL gates driving a bus line.

4.18 SUMMARY

Essential features of all the major logic families have been discussed and the important conclusion are given below:

1. RTL and DTL families are no more used for new systems because of their low speed, high power dissipation, and low fan-out.
2. TTL is the most popular general purpose logic family. It is available in seven different series with a wide range of operating speed, power dissipation, and fan-out. There are a large number of functions in SSI and MSI available in TTL.

   TTL ICs are available with totem-pole output (which decreases speed-power product), open-collector output (which makes possible wired-AND connection and bus operation), and tri-state (TSL) outputs (which are ideally suited for bus operation).
3. HTL are best suited for an industrial environment where electrical noise level is high.
4. ECL is the fastest logic family. Its main disadvantages are low noise-margins and high power dissipation. For interfacing with other logic families, level-shifting networks are required.
5. I^2L is the only saturated bipolar logic suitable for LSI because of small silicon chip area required, and low power consumption. The supply voltage required is low hence it is highly suitable for battery operated systems.

   I^2L circuits can drive TTL circuits if a resistive load is connected to the output stage of I^2L with a higher supply voltage (5 V).
6. MOS devices occupy a very small fraction of silicon chip area in comparison to bipolar devices and require very small power. Therefore, MOS logic is the most popular logic for LSI. The main drawback of MOS logic is slow speed, which is being improved upon by improvements in the technology of MOS fabrication. HMOS, a variety of NMOS has speeds comparable to bipolar logic families.
7. CMOS has the lowest speed power product and requires very small power.
   It is the most popular logic family and has led to the VLSI chips.
8. Corresponding to TTL 54/74 series, 54C/74C, 54HC/74HC, 54HCT/74HCT, 54AC/74AC and 54ACT/74ACT series have been developed which are directly compatible with various 54/74TTL series and have the same numbering scheme and pin-outs.

   A comparison of various digital IC logic families is given in Table 4.12.

Glossary

Active pull-up A circuit with active devices used to pull up the output voltage of a logic circuit from LOW to HIGH in response to the appropriate inputs.

Bipolar logic Logic circuits using bipolar junction semiconductor devices.

Breadth of logic family The number of different types of gates and other functions available in an IC logic family.

Buffer A circuit or gate that can drive a substantially higher number of gates or other loads. Also known as Buffer driver.

Bus A group of conductors carrying a related set of signals.

CMOS (Complementary metal-oxide semiconductor) A MOS device that uses one *p*-channel and one *n*-channel device to make an inverter circuit.

Current sink logic A logic circuit in which the output sink current corresponding to logic 0 state is appreciably higher than the output source current corresponding to logic 1 state.

Current source logic A logic circuit in which the output source current corresponding to logic 1 state is appreciably higher than the output sink current corresponding to logic 0 state.

DCTL (Direct-coupled transistor logic) A form of bipolar logic that uses direct coupling.

Depletion mode MOSFET A MOS device in which channel width gets depleted when the voltage of proper polarity is applied at the gate.

DTL (Diode transistor logic) A form of bipolar logic circuit that uses diodes and bipolar junction transistors to realize a logic operation.

ECL (Emitter-coupled logic) A form of bipolar logic circuit that uses emitter-coupled configuration.

Enhancement mode MOSFET A MOS device in which the channel is formed only when a proper voltage is applied at the gate. The channel width enhances with the increased voltage at the gate.

Fan-in The number of inputs of a logic gate.