EE 8351 - Digital logic circuits

Unit-1 Number systems and digital logic families

Natural Numbers: Number 0 and any number obtained by repeatedly adding a count of 1 to 0

A value of each digit in a number can be determined using

* The digit
* position of the digit in the number
* Base of the number system

Number system and description

Decimal Number system - used in our day-to-day life

Radix or Base : 10

Decimal numbers : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

537.26
↑ MSB — Most Significant Bit
↑ Decimal Point
↖ LSB - least Significant Bit

Binary Number system - Each binary digit called as bits.

Base/Radix : 2

Use two digits 0 and 1 → Binary numbers

$(110010)_2$

Octal Number system

Base/Radix: 8

Octal numbers: 0, 1, 2, 3, 4, 5, 6, 7

$(12570)_8$ or $(735)_8$ → octal number

Hexa-decimal number system

Base/Radix: 16

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

Octal digital Value	Binary equivalent
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Relation ship between decimal, Binary, Octal and Hexa-decimal numbers

Decimal	Binary	Octal	Hexadecimal
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Conversion of Number Systems

① Binary to Decimal Number system

$(11011)_2 = (?)_{10}$

11011

$\rightarrow 1 \times 2^0 = 1$

$\rightarrow 1 \times 2^1 = 2$

$\rightarrow 0 \times 2^2 = 0$

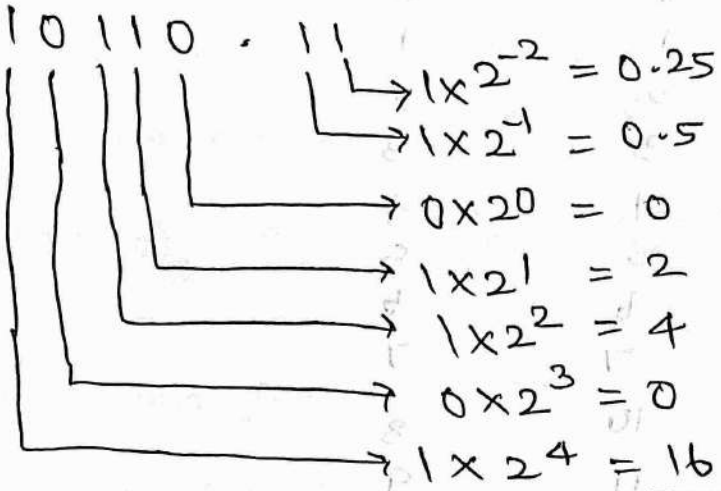
$\rightarrow 1 \times 2^3 = 8$

$\rightarrow 1 \times 2^4 = 16$

27

$(27)_{10}$

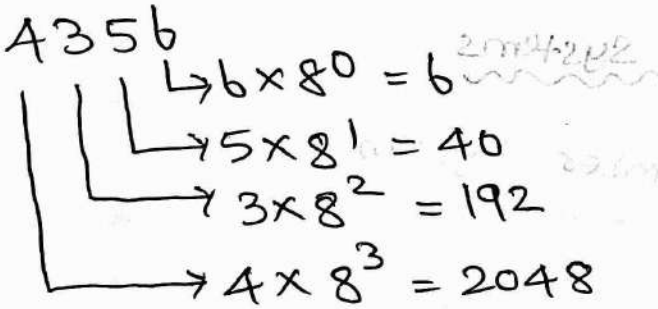
$(10110.11)_2 = (?)_{10}$



$(22.75)_{10}$

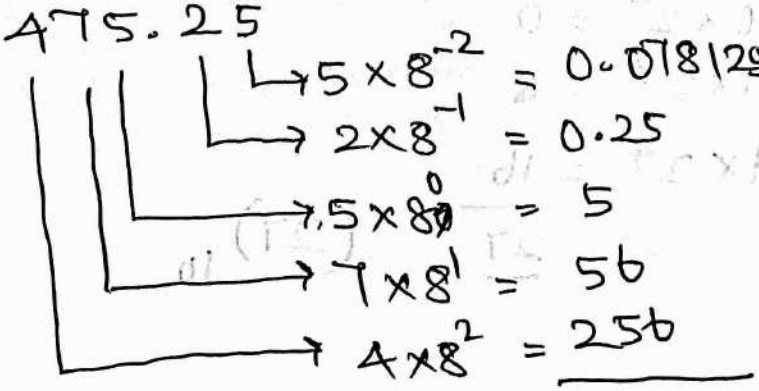
② octal to decimal Number system

$(4356)_8 = (?)_{10}$



$(2286)_{10}$

$(475.25)_8 = (?)_{10}$



$(5 \times 8^{-2} = 5/8^2)$

$(317.32813)_{10}$

③ Hexadecimal to decimal number system

$(ABC5)_{16} = (?)_{10}$

A B C 5

$5 \times 16^0 = 5$

$12 \times 16^1 = 192$

$11 \times 16^2 = 2816$

$10 \times 16^3 = 40960$

$(43973)_{10}$

$(9B2.1A)_{16} = (?)_{10}$

9 B 2 . 1 A

$10 \times 16^{-2} = 0.0390$

$1 \times 16^{-1} = 0.0625$

$2 \times 16^0 = 2$

$B(11) \times 16^1 = 176$

$9 \times 16^2 = 2304$

$(2482.1015)_{10}$

④ Decimal to Binary number system

$(985)_{10} = (?)_2$

2	985
2	492 – 1
2	246 – 0
2	123 – 0
2	61 – 1
2	30 – 1
2	15 – 0
2	7 – 1
2	3 – 1
	1 – 1

$(1111011001)_2$

$(37.812)_{10} = (?)_2$

2	37
2	18 – 1
2	9 – 0
2	4 – 1
2	2 – 0
	1 – 0

$0.812 \times 2 = 1.624 \quad 1$

$0.624 \times 2 = 1.248 \quad 1$

$0.248 \times 2 = 0.496 \quad 0$

$0.496 \times 2 = 0.992 \quad 0$

$(100101.1100)_2$

⑤ Decimal to Octal Number system

$(557)_{10} = (?)_8$

8	557
8	69 – 5
8	8 – 5
	1 – 0

$(1055)_8$

$(214.45)_{10} = (?)_8$

8	214
8	26 – 6
	3 – 2

$0.45 \times 8 = 3.6 \quad 3$

$0.6 \times 8 = 4.8 \quad 4$

$0.8 \times 8 = 6.4 \quad 6$

$0.4 \times 8 = 3.2 \quad 3$

$(326.3463)_8$

$(23.625)_{10} = (?)_8$

8	23
	2 – 7

$0.625 \times 8 = 5.0 \quad 5$

$(27.5)_8$

$(2225)_{10} = (?)_{16}$

16	2225	
16	139 – 1	↑
	8 – 11	

$(8B1)_{16}$

$(3509.64)_{10} = (?)_{16}$

16	3509	
16	219 – 5	↑
	13 – 11	

$(DB5)$

$0.64 \times 16 = 10.24 \quad 10 \ (A)$

$0.24 \times 16 = 3.84 \quad 3$

$0.84 \times 16 = 13.44 \quad 13 \ (D)$ ↓

$0.44 \times 16 = 7.04 \quad 7$

$(DB5.A3D7)_{16}$

⑦ Binary to Octal number system

From the rightmost end, each group of 3 bits is replaced by its decimal equivalent.

$(1111\ 01100)_2 = (?)_8$

$\underline{111}\ \underline{101}\ \underline{100} \quad (754)_8$

$7\ \ 5\ \ 4$

$(10011010.101)_2 = (?)_8$

$\underline{010}\ \underline{011}\ \underline{010} \cdot \underline{101} \quad (232.5)_8$

$2\ \ 3\ \ 2\ \ 5$

(8) Binary to Hexadecimal number system

From the rightmost end, each group of 4 bits is replaced by its equivalent hexadecimal.

$(1111\ 0110101)_2 = (?)_{16}$

0111	1011	0101	$(7B5)_{16}$
7	B	5	

$(11011.1)_2 = (?)_{16}$

0001	1011	.1000	$(1B.8)_{16}$
1	B	8	

(9) Octal to Binary Number system

$(55)_8 = (?)_2$

5 → 101

5 → 101

$(55)_8 = (101101)_2$

$(37.6)_8 = (?)_2$

3	7	.	6
↓	↓		↓
011	111		110

$(37.6)_8 = (011111.110)_2$

(10) Hexadecimal to Binary number system

Each digit in the given number is replaced by its 4-bit binary equivalent.

$(3FD)_{16} = (?)_2$

3 → 0011, F → 1111, D → 1101 $(001111111101)_2$

$(5A9.B4)_{16} = (?)_2$

5 → 0101, A → 1010, 9 → 1001, B → 1011, 4 → 0100

$(010110101001.10110100)_2$

⑪ Octal to Hexa-decimal number system

Step 1 : Convert the given octal number to binary

Step 2 : Convert the binary number obtained in step 1 to hexa-decimal

$(615)_8 = (?)_{16}$

615 → 101, 1, 110, 001 $= (110001101)_2$

$\underline{0001}\ \underline{1000}\ \underline{1101}$ $(18D)_{16}$

$\underline{1}\quad\underline{8}\quad\underline{D}$

$(54.34)_8 = (?)_{16}$

5 → 101, 4 → 100, 3 → 011, 4 → 100 $(101100.011100)_2$

$\underline{0010}\ \underline{1100}.\ \underline{0111}\ \underline{0000}$

$\underline{2}\quad\underline{C}\quad\underline{7}\quad\underline{0}$

$(2C.70)_{16}$

(12) Hexadecimal to octal number system

Step 1: Convert the given hexadecimal number to binary

Step 2: Convert the binary number obtained in Step 1 to octal

$(25B)_{16} = (?)_8$

2	5	B
↓	↓	↓
0010	0101	1011

$\underline{0010}\,\underline{0101}\,\underline{1011}$ $(1133)_8$

1 1 3 3

$(B2.A1)_{16} = (?)_8$

B	2	.	A	1
1011	0010		1010	0001

$\underline{101}\,\underline{100}\,\underline{10}\cdot\underline{101}\,\underline{000}\,\underline{010}$ $(262.502)_8$

2 6 2 5 0 2

Exercise problems

(1) Convert the hexadecimal number AB.CD into octal number. Ans: $(253.632)_8$

(2) Convert $(153.513)_{10}$ to octal. Ans: $(231.4065)_8$

(3) Convert 0.35 to equivalent hexadecimal number.
Ans: $(0.5999)_{16}$

(4) Convert $(101101.1101)_2$ to decimal form
Ans: $(45.8125)_{10}$

⑤ convert $(101101.1101)_2$ to hexadecimal form.

Ans: $(2D.D)_{16}$

⑥ convert $(143)_{10}$ into its binary equivalent

Ans: $(10001111)_2$

⑦ convert the following decimal number into hexadecimal number

(i) 115_{10} Ans: $(73)_{16}$

(ii) 235_{10} Ans: $(EB)_{16}$

⑧ convert $(43)_{10}$ into binary Ans: $(101011)_2$

⑨ convert $(100.625)_{10}$ to binary Ans: $(1100100.1010)_2$

⑩ convert the following into octal.

(i) $(144.25)_{10}$ Ans: $(220.20)_8$

(ii) $(232.52)_{10}$ Ans: $(350.412172)_8$

⑪ convert the following into hexadecimal

(i) $(342)_{10}$ Ans: $(15b)_{16}$

(ii) $(82.4)_{10}$ Ans: $(52.66)_{16}$

⑫ convert the following into decimal

(i) $(1110.101)_2$ Ans: $(14.625)_{10}$

(ii) $(11001)_2$ Ans: $(25)_{10}$

⑬ convert $(3714.27)_8$ into decimal. Ans: $(1996.359375)_{10}$

⑭ convert the following into decimal.

(i) $(64)_{16}$ Ans: $(100)_{10}$

(ii) $(CD47)_{16}$ Ans: $(52551)_{10}$

(iii) $(B9A.C)_{16}$ Ans: $(2970.75)_{10}$

(15) Convert the following octal to binary

(i) $(232)_8$ Ans: $(010011010)_2$

(ii) $(465.1)_8$ Ans: $(100110101.001)_2$

(16) Convert the $(47)_8$ to hexadecimal number

Ans: $(27)_{16}$

(17) Convert $(A2)_{16}$ to octal number

Ans: $(242)_8$

(18) Convert $(D9.DE)_{16}$ to octal number

Ans: $(331.674)_8$

Arithmetic Operations

* Done using Binary numbers.

Binary Addition

$0+0=0$

$0+1=1$

$1+0=1$

$1+1=10$

Binary Subtraction

$0-0=0$

$0-1=1$ with borrow as 1

$1-0=1$

$1-1=0$

$10-1=10$

Binary Multiplication

$0\times0=0$

$0\times1=0$

$1\times0=0$

$1\times1=1$

Binary division

$0\div0$ = Not allowed

$0\div1=0$

$1\div0$ = Not allowed

$1\div1=1$

Add the Binary numbers

(i) 111 and 101

```
 1 1 1  ← Carry
   1 1 1
   1 0 1
 -------
 1 1 0 0  = Sum
```

(7+5 = 12)

(ii) 11.011 and 101.0101

```
   11.011
 101.0101
 --------
 1000.1011  = Sum
```

(3.375 + 5.3125 = 8.6875)

Subtract the Binary numbers

(i) 1001_2 from 1101_2

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

(ii) 100.1_2 from 110.01_2

```
  110.01
  100.10
 -------
  0 0 1 - 1 1
```

Multiply the Binary numbers

(i) $1100_2 \times 1010_2$

```
          1 1 0 0
          1 0 1 0
         ---------
          0 0 0 0
        1 1 0 0 x
      0 0 0 0 x x
    1 1 0 0 x x x
  -----------------
  1 1 1 1 0 0 0
```

(ii) $1.01_2 \times 10.1_2$

```
      1.0 1
      10.1
     -----
      1 0 1
    0 0 0
  1 0 1
 ---------
  1 1.0 0 1
```

$= 11.001_2$

Divide the binary numbers

(i) $1100l_2 \div 10l_2$

$$
\begin{array}{r}
101 \\
101 \overline{)\,1100} \\
101 \\
\hline
0101 \\
101 \\
\hline
000
\end{array}
$$

Remainder $= 000_2$

Quotient $= 10l_2$

Representation of Binary numbers

Signed Binary number representation

* Binary number → represented with a seperate sign bit along with the magnitude

* In an 8 Bit number, MSB → Sign bit remaining 7 bit corresponds to magnitude

Number	Sign bit	Magnitude
+13	0	000 1101
-46	1	010 1110

1's complement representation

The 1's complement form of any binary number is obtained simply by changing each 0 to 1 and each 1 to 0.

11011 → 00100

1110011 → 0001100

2's Complement representation

* Obtained by adding 1 to its 1's complement

2's complement = 1's complement + 1

$11011 \rightarrow 00100 \rightarrow 00101$

Represent the following decimal number as

(a) Sign and magnitude binary numbers

(b) 1's complement (c) 2's complement.

(i) 378_{10}

(a) Sign and magnitude representation

$\underset{\text{Indicate sign}}{0}\ \underset{\text{Indicate magnitude}}{101111010}$

(b) 1's complement $0\ 010000101$

[No need to complement the sign bit]

(c) 2's complement $0\ 010000110$

(ii) -13.75_{10}

(a) Sign and magnitude representation

$\underset{\text{Indicate sign}}{1}\ \underset{\text{Indicate magnitude}}{1101.11}$

(b) 1's complement $1\ 1101.11$ is $1\ 0010.00$

(c) 2's complement 10010.01_2

1's complement subtraction

(a) Subtraction of smaller number from larger number

(i) convert the given numbers to binary

(ii) Determine the 1's complement of smaller number

(iii) Add 1's complement to larger number

(iv) Remove the carry and add it to result. This is called end-around carry.

Subtract 25_{10} from 45_{10} using 1's complement method.

$45_{10} = 101101_2$

$25_{10} = 011001_2$

1's complement $25_{10} = 100110_2$

Add 1's complement to larger number

$$\begin{array}{r} 101101 \\ 100110 \\ \hline 1\ 010011 \end{array}$$

↑ carry

Remove the carry $\quad 010011$

$$\begin{array}{r} +1 \\ \hline 010100_2 \end{array}$$

$45_{10} - 25_{10} = 20_{10} = 010100_2$

Subtract 69.25 from 86.75 using 1's complement

(Ans: $0010001.10_2 = 17.5_{10}$)

(b) Subtraction of larger number from smaller number

(i) Convert the given number to binary

(ii) Determine 1's complement of larger number

(iii) Add the 1's complement to the smaller number

(iv) Take 1's complement of answer. put one negative sign.

Subtract 36_{10} from 12_{10} using 1's complement method.

$36_{10} = 100100_2$

$12_{10} = 001100_2$

1's complement of larger number $36_{10} = 011011_2$

Add 1's complement to smaller number

$$\begin{array}{r} 011011 \\ 001100 \\ \hline 100111 \end{array}$$

Take 1's complement of answer. put one (-)ve sign

$100111 \longrightarrow 011000$

$-11000 = -24_{10}$.

Subtract 93.5_{10} from 55.25_{10} using 1's complement method.

Ans: $-0100110.01_2 = -38.25_{10}$

2's complement Subtraction

(a) Subtraction of smaller number from larger number

(i) Convert the given number to binary

(ii) Determine 2's complement of smaller number

(iii) Add this to larger number

(iv) discard the carry

Subtract 18_{10} from 30_{10} using 2's complement.

$30_{10} = 11110_2$

$18_{10} = 10010_2$

2's complement for 18_{10}

$$\begin{array}{r} 10010 \\ 1's = 01101 \\ +1 \\ \hline 01110_2 \end{array}$$

Add this to lasger number

$$\begin{array}{r} 01110 \\ 11110 \\ \hline 101100 \end{array}$$

↑ carry

discard the carry Ans: $(01100)_2$

$30_{10} - 18_{10} = 12_{10}$

Subtract 69.25_{10} from 86.75_{10} using 2's Complement

Ans: $(0010001.10)_2$

(b) Subtraction of larger number from smaller number

(i) Convert the given numbers to binary

(ii) determine the 2's complement of larger number

(iii) Add this to smaller number

(iv) Take 2's complement to the answer obtained in step 3 and put one negative sign.

Subtract 36_{10} from 12_{10} using 2's complement

$36_{10} = 100100_2$

$12_{10} = 001100_2$

2's complement of larger number 011100

Add this to smaller number

$$\begin{array}{r} 011100 \\ 001100 \\ \hline 101000 \end{array}$$

2's complement for the answer 010111

$$\begin{array}{r} +\ 1 \\ \hline 011000 \end{array}$$

Put one negative sign $(-011000)_2$

Subtract 93.5_{10} from 55.25_{10} using 2's complement

Ans: $-38.25_{10} = -0100110.01_2$

9's complement representation

9's complement of decimal number found by subtracting each digit in the number from 9.

Find 9's complement for the following number

(i) 146

$$\begin{array}{r} 999 \\ -\ 146 \\ \hline 853 \end{array}$$

(ii) 4397.152

$$\begin{array}{r} 9999.999 \\ 4397.152 \\ \hline 5602.847 \end{array}$$

9's Complement Subtraction

(a) Subtraction of smaller number from larger number

(i) Determine 9's complement of smaller number

(ii) Add this to larger number

(iii) Remove the carry and add it to result.

Perform the subtraction by using 9's complement

(a) 86 − 65

```
  99
(-)65
------
  34
```

(b) 109.34 − 86.91
(Ans: 22.43)

```
  86
+ 34
------
 120
 ↳ carry
  20
+  1
------
  21   = 86-65
```

(b) Subtraction of larger number from smaller number

(i) Determine 9's complement of larger number

(ii) Add this to smaller number

(iii) Take 9's of the answer obtained in Step 2. put one negative sign.

Perform the subtraction using 9's complement.

(a) 33 - 56

```
  99
(-) 56
  43
```

Add to smaller number

```
  43
+ 33
  76
```

9's complement & put one negative sign

```
  99
(-) 76
  23
```

33 - 56 = -23

(b) 193.5 - 236.7

(Ans: -43.2)

10's complement Representation

10's complement = 9's complement + 1

Find 10's complement for the following decimal number

(a) 739

```
  999
(-) 739
  260
  + 1
  261
```

(b) 4397.152

```
  9999.999
(-) 4397.152
  5602.847
      + 1
  5602.848
```

10's Complement Subtraction

(a) Subtract smaller number from larger number

(i) Determine 10's complement of smaller number

(ii) Add this to larger number

(iii) Discard the carry

Perform the subtraction using 10's complement

(a) 398 − 212

```
  999
- 212
-----
  787
+   1
-----
  788
```

Add this to larger number

```
  788
  398
-----
 1186
```

↳ Discard the carry.

398 − 212 = 186

(b) 236.54 − 112.44 (Ans: 124.10)

(b) Subtract larger number from smaller number

(i) Determine the 10's complement of larger number

(ii) Add this to smaller number

(iii) Take 10's complement of answer obtained in step 2. Put one negative sign.

Perform the following subtraction using 1's complement

(a) 138 - 853

```
  999
- 853
-----
  146
+   1
-----
  147
```

Add this to smaller number

```
  147
  138
-----
  285
```

Take 10's complement to answer and put (-)ve sign

```
  999
- 285
-----
  714
+   1
-----
  715
```

138 − 853 = −715

(b) 453.5 − 1085.63 (Ans: −632.13)

Binary Codes

* Digital data → represented, stored and transmitted as group of binary digit (bits).
* Group of bits called as Binary code.
* classified as
  - (a) weighted code
  - (b) non-weighted code
  - (c) sequential code
  - (d) Alphanumeric code
  - (e) Error detecting and correcting codes

Weighted codes

Each digit position of the number represent a specific weight.

BCD (8421) - Binary coded decimal

* 8-4-2-1 BCD ; Each decimal digit is represented by a 4 bit binary number.

* The weight associated with 4 bits are 8-4-2-1 from left to right.

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

Example:

$(185)_{10}$ = Decimal

1 8 5

↓ ↓ ↓

0001 1000 0101 = BCD

* BCD number needs more bits than its equivalent binary value.

* Advantage : Easy to convert between it and decimal

Convert the decimal number given below:

(a) 685_{10} $(0110\ 1000\ 0101)_{BCD}$

(b) 125.53_{10} $(0001\ 0010\ 0101.\ 0101\ 0011)_{BCD}$

BCD Addition

The rule for addition of two BCD number is given below:

(1) Add the two numbers using the rules for binary addition

(2) If a four bit number is equal to or less than $9\ (1001)_2$ then it is a valid BCD number

(3) If a four bit number is greater than $9\ (1001)_2$ it is a invalid BCD number. Add $6\ (0110)_2$ to the four bit sum in order to make a valid BCD number. If a carry results when 6 is added, add the carry to the next four bit group.

Add the following numbers using BCD addition

(a) $4 + 5$

$4_{10} = (0100)_{BCD}$

$5_{10} = (0101)_{BCD}$

$$\begin{array}{r} 0100 \\ 0101 \\ \hline 1001 \end{array}$$

$4_{10} + 5_{10} = 9_{10}$

(b) $4 + 8$

$4_{10} = (0100)_{BCD}$

$8_{10} = (1000)_{BCD}$

$$\begin{array}{r} 0100 \\ 1000 \\ \hline 1100 \\ +\ 0110 \\ \hline (0001\ 0010)_{BCD} \end{array} > 9_{10} = 12_{10}$$

(c) $184 + 576$

$184_{10} = (0001\ \ 1000\ \ 0100)_{BCD}$

$576_{10} = (0101\ \ 0111\ \ 0110)_{BCD}$

0111	10000	1010
	0110	0110
0111	① 0110	① 0000
7	6	0

$184 + 576 = 760_{10}$

(d) $24 + 18$

$24_{10} = 0010\ \ 0100$

$18_{10} = 0001\ \ 1000$

0110

$0100\ \ 1100$

$0100\ \ ①\ 0010$

$4\ \ \ \ \ 2$

$(42)_{10}$

(e) $22.57 + 191.63$

$22.57_{10} = 0010\ \ 0010\ .\ 0101\ \ 0111$

$191.63_{10} = 0001\ \ 1001\ \ 0001\ .\ 0110\ \ 0011$

$$
\begin{array}{r}
0000\ \ 0010\ \ 0010\ .\ 0101\ \ 0111 \\
+\ 0001\ \ 1001\ \ 0001\ .\ 0110\ \ 0011 \\
\hline
0010\ \ 1011\ \ 0100\ \ 1100\ \ 1010 \\
0110\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ 0110\ \ 0110 \\
\hline
0010\ \ ①\ 0001\ \ 0100\ \ ①\ 0010\ \ ①\ 0000
\end{array}
$$

$(214.20)_{10}$

(f) 8 + 9 Ans: $(0001\ 0111 = 17_{10})$

(g) 48 + 58 Ans: $(0001\ 0000\ 0110 = 106_{10})$

(h) 175 + 236 Ans: $(0101\ 0000\ 0001 = 501_{10})$

(i) 589 + 199 (Ans: $0111\ 1000\ 1000 = 788_{10}$)

BCD Subtraction

Subtraction of smaller number from larger number

1. Find the 1's complement of smaller number
2. Add it with larger number
3. Remove the carry and add it to the result.
4. Add 1010 if end around carry is 1
5. Ignore the intermediate carry.

Subtract 274 from 835 using BCD addition

$835_{10} = (1000\ 0011\ 0101)_{BCD}$

$274_{10} = (0010\ 0111\ 0100)_{BCD}$

(i) Find 1's complement of smaller number

1101 1000 1011

(ii) Add it with larger number

$$\begin{array}{r} 1000\ \ 0011\ \ 0101 \\ +\ 1101\ \ 1000\ \ 1011 \\ \hline (1)\ 0101\ \ 1100\ \ 0000 \end{array}$$

↑ carry

(iii) Remove the carry and add it to result

$$\begin{array}{r} 0101\ \ 1100\ \ 0000 \\ +\ \ \ \ \ \ \ \ \ \ \ \ 1 \\ \hline 0101\ \ 1100\ \ 0001 \end{array}$$

→ end around carry

(iv) Add 1010 if end-around carry is "1"

$$\begin{array}{llll} 0101 & 1100 & 0001 \\ (+)\ 0000 & 1010 & 0000 \\ \hline 0101 & (1)\ 0110 & 0001 \\ 5 & \quad 6 & 7 \end{array}$$

↑ Ignore the intermediate carry

$835_{10} - 274_{10} = 567_{10}$

Subtraction of larger number from smaller number

1. Find 1's complement of larger number
2. Add it with the smaller number
3. Remove the carry and add it to the result. Find 1's complement of it
4. Add 1010 if end-around carry is 1
5. Ignore the intermediate carry and provide a negative sign

Perform Subtraction using BCD Subtraction

(a) 429 - 476

$429_{10} = (0100\ 0010\ 1001)_{BCD}$

$476_{10} = (0100\ 0111\ 0110)_{BCD}$

(i) Find 1's complement of larger number

1011 1000 1001

(ii) add it with smaller number

$$\begin{array}{llll} & 0100 & 0010 & 1001 \\ & \text{carry} & & \text{1'} \\ (+) & 1011 & 1000 & 1001 \\ \hline (0) & 1111 & 1011 & 0010 \end{array}$$

↑ carry

(iii) Remove carry and add it to the result. Find 1's complement of it

0000 0100 1101

(iv) Add 1010 if end around carry is '1'

end-around carry

```
  0000  0100  1101
+ 0000  0000  1010
-------------------
  0000  0100 (1) 0111
```

Ignore intermediate carry.

$429_{10} - 476_{10} = -47_{10}$

Subtract 476.92 from 599.43 using BCD subtraction

Ans: $(0001\ 0010\ 0010.0101\ 0001)_{BCD} = 122.51_{10}$

Subtract $98.90_{10} - 99.22_{10}$ using BCD subtraction

Ans: $(-0000\ 0000.0011\ 0010)_{BCD} = -0.32_{10}$

Non-weighted codes

* Not assigned with any weight to each digit position.
* Excess-3, Gray codes are non-weighted codes.

Excess-3 code

* Derived from decimal number by adding 3 to each decimal digit and then converting to BCD.
* Also called as reflective code
* 0000, 0001, 0010, 1101. — Invalid Excess-3 codes.

Decimal	8421 BCD	Excess-3
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

Convert 643_{10} into Excess-3 code

6	4	3
+3	+3	+3
9	7	6
↓	↓	↓
1001	0111	0110

Excess-3 code for 643_{10} is 1001 0111 0110

Gray code

* Special case of unit-distance code
* In unit distance code, bit patterns for two consecutive numbers differ in only one bit position.
* also called as cyclic code.

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

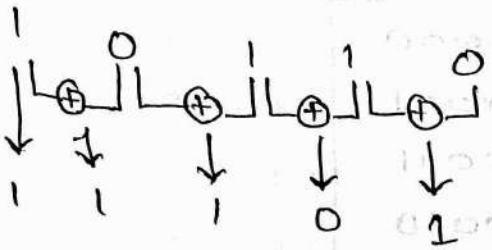
Conversion of a binary number to gray code

(i) The first bit of gray code is the same as the first bit of the binary number

(ii) second bit of the gray code equals the Exclusive OR of the first and second bit of the binary number.

(iii) third gay code bit equals the Exclusive-OR of the second and third bits of the binary number and so on.

Convert 10110_2 to gray code



11101 = Gray code

Convert the binary number 10101101_2 to gray code

(Ans: 11111011)

Conversion of gray code to Binary

(i) first binar bit (MSB) is the same as that of first gray code bit.

(ii) If the second gray code bit is 0, the second binary bit is the same as that of first binary. If the second gray bit is 1, the second binary bit is the complement of its First binary bit

(iii) step (ii) is repeated for each successive bit.

Convert the gray code to binary form

(i) 11010_G

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

10011_2

(ii) 1010111_G

$1 \to 1 \to 0 \to 0 \to 1 \to 0 \to 1_2$

Sequential Codes

In sequential codes each succeeding code is one binary number greater than its preceding codes.

Eg: 8421 and Excess-3 code

Alphanumeric codes ** consists of both numbers and alphabetic characters.

* most commonly used codes are

(a) ASCII code

American standard code for Information interchange (ASCII)

* used in most microcomputer by its manufacturers.
* 7-bit code → store as one byte with one bit unused.

Format: $A_6\ A_5\ A_4\ A_3\ A_2\ A_1\ A_0$
where each bit is as '0' or '1'

A → 100 0001
B → 100 0010

(b) EBCDIC code

Extended Binary Coded Decimal Information Code.

eight bit for each character
ninth bit for parity.

Error detecting and correct codes

* when digital information in binary form is transmitted from one s/m to another s/m error may occur.
* This refers $0 \rightarrow 1$ (or) $1 \rightarrow 0$.
* To maintain data integrity blw transmitter and receiver extra bit or more than one bit is added.

Parity Code

* most simple and commonly used error detecting method is parity check.
* classified into two types (a) odd parity method
  (b) Even parity method
* Even parity method → total number of 1's in the code group must be even number.
* odd-parity method → total number of 1's in the code group must be odd number.

message	Even parity	odd parity
1000001	01000001	11000001
1010100	11010100	01010100

* parity of each chacter is checked in the receiving end.
* This detects the presences of error in received message.
* Hamming code is used in detection and correction of error.

Hamming Code

* Error may be occur while transmitting the data from one end to the other end.
* 0→1 (or) 1→0, this is due to presence of noise
* Some extra bits must be added to the transmitting information which allow the detection and sometimes correction of error. This xtra bit called Redundant bits.

* Codes which allow only error detection are called error detecting codes.

* Codes which allow error detection and correction are called error detecting and correcting codes.

Number of redundancy bits

$$2^r \geq m + r + 1$$

- m → Number of information
- r → number of redundancy bits to be added to the information bit.

The minimum value of 'r' should satisfy the above equation be added to the information bit.

Step 1: Calculating the number of redundancy bit for the error detecting and correcting code

Step 2: Locating the redundancy bits.
Redundancy bits location 2^{r-1} position

Step 3: then calculate the value of 'r'.

Algorithm of Hamming code :

* Generalized form of code is $P_1 P_2 D_1 P_3 D_2 D_3 D_4 P_4 D_5 D_6 D_7 D_8 D_9 D_{10} D_{11} P_5 \ldots$ where P and D respectively represent parity and data bits.
* from above generalized code form that all bit positions that are powers of 2 (position 1, 2, 4, 8, 16 ...) as parity bits
* All other bit positions (3, 5, 6, 7, 9, 10, 11 ....) encode data.
* Each parity bit is alloted a group of bits from the data bits in the code word, and the value of the parity bit (0 or 1) is used to give it certain parity.
* Groups are formed by first checking N-1 bits and then alternately skipping and checking N bits following the parity bit. Here, N is the position of the parity bits: 1 for P_1
  2 for P_2
  4 for P_3
  8 for P_4 and so on.

* Hamming code is capable of correcting single-bit errors on messages of any length. Although the hamming code can detect two bit errors, it cannot give the error locations.

Example 1.57: Encode the binary word 1011 into seven bit hamming code.

Solution:

Step 1: Find the number of redundant bits by $2^r \geq m + r + 1$.

Here the number of information bit is m=4.

$$2^r \geq 4 + r + 1$$

Therefore the minimum value of 'r' that can satisfy the above equation is r=3. Therefore the hamming code contains 'm+r' bits (m+r=7)

Step 2: Locating the redundant bits

The position of redundant bits is given by 2^{r-1} where r=1,2,3. The redundant bits are in position r_1, r_2 and r_4

7	6	5	4	3	2	1
m_4	m_3	m_2	r_4	m_1	r_2	r_1
1	0	1		1		

Figure 1.8 Hamming code

Step 3: Calculating the value of 'r'

r_1 = XOR of bits (3,5,7) = $1 \oplus 1 \oplus 1 = 1$

r_2 = XOR of bits (3,6,7) = $1 \oplus 0 \oplus 1 = 0$

r_4 = XOR of bits (5,6,7) = $1 \oplus 0 \oplus 1 = 0$

Therefore the hamming code to be transmitted is 1010101

7	6	5	4	3	2	1
m_4	m_3	m_2	r_4	m_1	r_2	r_1
1	0	1	0	1	0	1

Figure 1.9 Hamming code

Example 1.58: Deduce the odd parity hamming code for the data : 1010. Introduce an error in the LSB of the hamming code and deduce the steps to detect the error.

Step 1: Find the number of redundant bits by $2^r \geq m + r + 1$.

Here the number of information bit is m=4.

$$2^r \geq 4 + r + 1$$

Therefore the minimum value of 'r' that can satisfy the above equation is r=3. Therefore the hamming code contains 'm+r' bits (m+r=7)

Step 2: Locating the redundant bits

The position of redundant bits is given by 2^{r-1} where r=1,2,3. The redundant bits are in position r_1, r_2 and r_4

7	6	5	4	3	2	1
m_4	m_3	m_2	r_4	m_1	r_2	r_1
1	0	1		0		

Figure 1.10 Hamming code

Step 3: Calculating the value of 'r'

r_1 = XNOR of bits (3,5,7) = $0 \odot 1 \odot 1 = 1$

r_2 = XNOR of bits (3,6,7) = $0 \odot 0 \odot 1 = 0$

r_4 = XNOR of bits (5,6,7) = $1 \odot 0 \odot 1 = 1$

Therefore the hamming code to be transmitted is 1011001

7	6	5	4	3	2	1
m_4	m_3	m_2	r_4	m_1	r_2	r_1
1	0	1	1	0	0	1

Figure 1.11 Hamming code

Step 4: *Introducing an error in the LSB bit of the hamming code, the received code is*

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

Figure 1.12 Hamming code

Step 5: Find the number of check bits.

$$2^c \geq l + 1$$

Here the number of bits in the hamming code is '7'

$$2^c \geq 7 + 1$$

The minimum value of 'c' that can satisfy the above equation is c=3. Therefore there are 3 check bits $C_3C_2C_1$

Step 6: Find the check bits

The check bits are given by the expression

$$C_1 = \text{XNOR of bits } (1,3,5,7)$$
$$C_2 = \text{XNOR of bits } (2,3,6,7)$$
$$C_3 = \text{XNOR of bits } (4,5,6,7)$$

Here

$$C_1 = 0 \odot 0 \odot 1 \odot 1 = 1$$
$$C_2 = 0 \odot 0 \odot 0 \odot 1 = 0$$
$$C_3 = 1 \odot 1 \odot 0 \odot 1 = 0$$

Hence the check bits $C_3C_2C_1 = 001$. Hence there is error in received code. Converting the check bits to decimal, we get

$$C_3C_2C_1 = 001 = 1_{10}$$

Therefore the error is in LSB bit and the corrected hamming code is

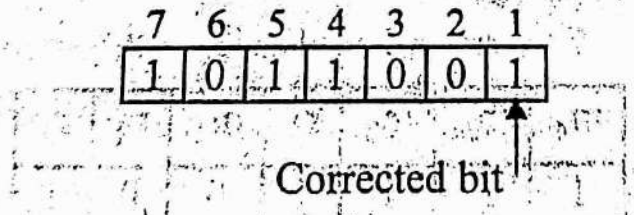


Figure 1.13 Error correction

Example 1.59: A 12-bit Hamming code received is 101110000110. Check for errors. What is the correct information transmitted.

Solution: The received code is

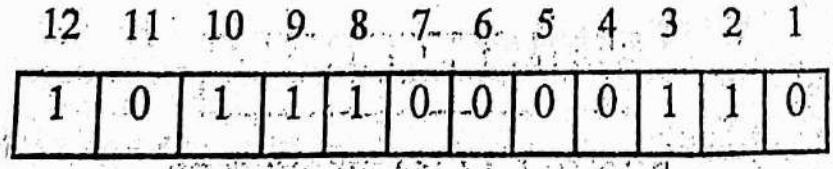


Figure 1.14 Received code

Step 1: Find the number of check bits.

$$2^c \geq l + 1$$

Here the number of bits in the hamming code is '12'

$$2^c \geq 12 + 1$$

The minimum value of 'c' that can satisfy the above equation is c = 4. Therefore there are 4 check bits $C_4C_3C_2C_1$

Step 2: Find the check bits

The check bits are given by the expression

C_1 = XOR of bits (1,3,5,7,9,11)

C_2 = XOR of bits (2,3,6,7,10,11)

C_3 = XOR of bits (4,5,6,7,12)

C_4 = XOR of bits (8,9,10,11,12)

Here

$$C_1 = 0 \oplus 1 \oplus 0 \oplus 0 \oplus 1 \oplus 0 = 0$$

$$C_2 = 1 \oplus 1 \oplus 0 \oplus 0 \oplus 1 \oplus 0 = 1$$

$$C_3 = 0 \oplus 0 \oplus 0 \oplus 0 \oplus 1 = 1$$

$$C_4 = 1 \oplus 1 \oplus 1 \oplus 0 \oplus 1 = 0$$

Hence the check bits $C_4C_3C_2C_1 = 0110$. Hence there is error in received code. Converting the check bits to decimal, we get

$$C_4C_3C_2C_1 = 0110 = 6_{10}$$

Therefore the corrected hamming code is

12	11	10	9	8	7	6	5	4	3	2	1
1	0	1	1	**1**	0	1	0	**0**	1	**1**	**0**

Corrected bit

Figure 1.15 Error correction

Step 3: Extracting the information bits

Discarding the redundant bits, the correct information bit that was transmitted was *10110101*

Example 1.60: Explain in detail the usage of Hamming codes for error detection and error correction with an example considering the data bits as **0101.**

Solution:

Step 1: Find the number of redundant bits by $2^r \geq m + r + 1$.

Here the number of information bit is m=4.

$$2^r \geq 4 + r + 1$$

Therefore the minimum value of 'r' that can satisfy the above equation is r=3. Therefore the hamming code contains 'm+r' bits (m+r=7)

Step 2: Locating the redundant bits

The position of redundant bits is given by 2^{r-1} where r=1,2,3. The redundant bits are in position r_1, r_2 and r_4

7	6	5	4	3	2	1
m_4	m_3	m_2	r_4	m_1	r_2	r_1
0	1	0		1		

Figure 1.16 Hamming code

Step 3: Calculating the value of 'r'

$$r_1 = \text{XOR of bits } (3,5,7) = 1 \oplus 0 \oplus 0 = 1$$

$$r_2 = \text{XOR of bits } (3,6,7) = 1 \oplus 1 \oplus 0 = 0$$

$$r_4 = \text{XOR of bits } (5,6,7) = 0 \oplus 1 \oplus 0 = 1$$

Therefore the hamming code to be transmitted is 0101101

7	6	5	4	3	2	1
m_4	m_3	m_2	r_4	m_1	r_2	r_1
0	1	0	1	1	0	1

Figure 1.17 Hamming code

Step 4: Introducing an error in the '3' of the hamming code, the received code is

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

Figure 1.18 Hamming code

Step 5: Find the number of check bits.

$$2^c \geq l+1$$

Here the number of bits in the hamming code is '7'

$$2^c \geq 7+1$$

The minimum value of 'c' that can satisfy the above equation is c=3. Therefore there are 3 check bits $C_3 C_2 C_1$

Step 6: Find the check bits

The check bits are given by the expression

C_1 = XOR of bits (1,3,5,7)

C_2 = XOR of bits (2,3,6,7)

C_3 = XOR of bits (4,5,6,7)

Here

$$C_1 = 1 \oplus 0 \oplus 0 \oplus 0 = 1$$

$$C_2 = 0 \oplus 0 \oplus 1 \oplus 0 = 1$$

$$C_3 = 1 \oplus 0 \oplus 1 \oplus 0 = 0$$

Hence the check bits $C_3 C_2 C_1 = 011$. Hence there is error in received code.

Converting the check bits to decimal, we get

$$C_3 C_2 C_1 = 011 = 3_{10}$$

Therefore the corrected hamming code is

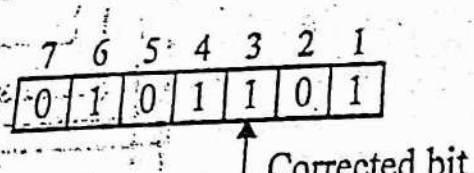


Figure 1.19 Error correction

Example 1.61: Design a odd parity hamming code generator and detector for 4-bit data and explain their logic.

Solution:

Let the 4-bit data bits be $m_4 m_3 m_2 m_1$. The number of redundant bits 'r' can be calculated using the relation,

$$2^r \geq m+r+1.$$

Here the number of information bit is m=4.

$$2^r \geq 4 + r + 1$$

Therefore the minimum value of 'r' that can satisfy the above equation is r=3. Therefore the hamming code contains 'm+r' bits (m+r=7). The position of redundant bits is given by 2^{r-1} where r=1,2,3. The redundant bits are in position r_1, r_2 and r_4.

7	6	5	4	3	2	1
m_4	m_3	m_2	r_4	m_1	r_2	r_1

Figure 1.20 Hamming code

The redundant bits for the odd parity hamming code can be calculated as

$$r_1 = \text{XNOR of bits } (3,5,7) = m_1 \odot m_2 \odot m_4$$

$$r_2 = \text{XNOR of bits } (3,6,7) = m_1 \odot m_3 \odot m_4$$

$$r_4 = \text{XNOR of bits } (5,6,7) = m_2 \odot m_3 \odot m_4$$

The logic diagram of hamming code generator can be drawn as shown in figure 1.21

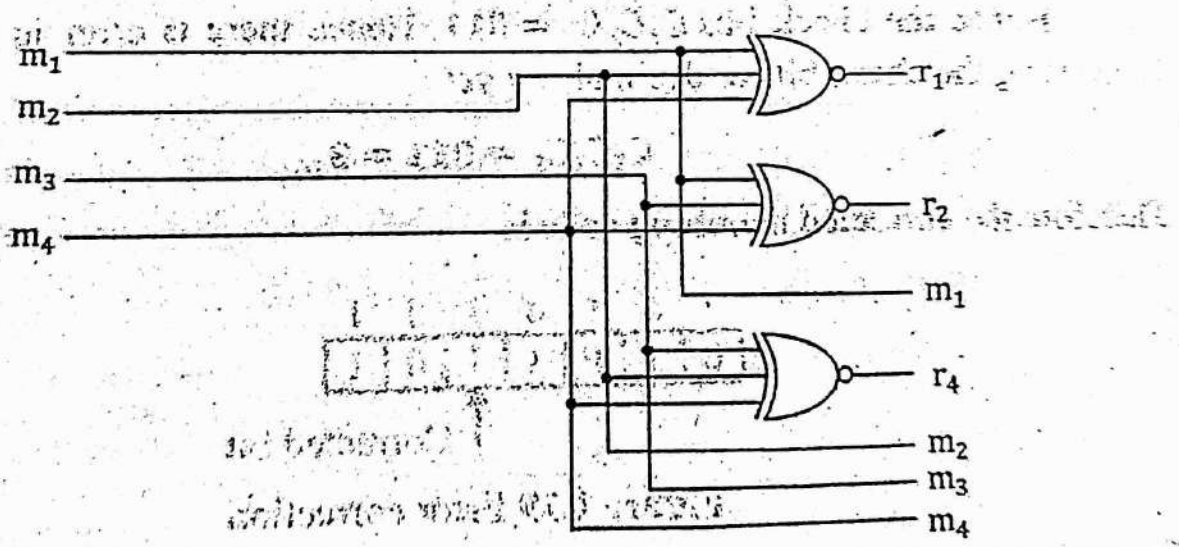


Figure 1.21 Hamming code generator

The hamming code is transmitted by the transmitter. At the receiver the check bits are estimated. The number of check bits can be calculated by using the relation,

$$2^c \geq l + 1$$

Where 'l' is the number of bits in hamming code. Here $l = 7$

Therefore $2^c \geq 7 + 1$

The minimum value of 'c' that can satisfy the above equation is c=3. Therefore there are 3 check bits $C_3C_2C_1$

The check bits are given by the expression

$$C_1 = \text{XNOR of bits } (1,3,5,7) = r_1 \odot m_1 \odot m_2 \odot m_4$$

$$C_2 = \text{XNOR of bits } (2,3,6,7) = r_2 \odot m_1 \odot m_3 \odot m_4$$

$$C_3 = \text{XNOR of bits } (4,5,6,7) = r_4 \odot m_2 \odot m_3 \odot m_4$$

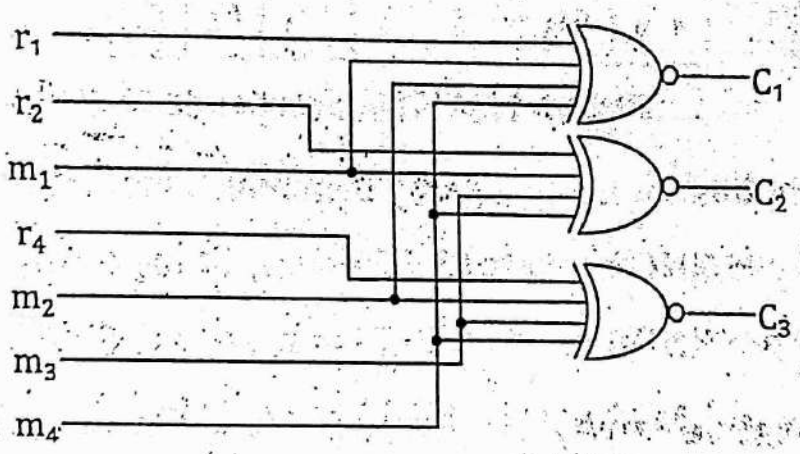


Figure 1.22 Hamming code detector

Logic Gates

* Logic gates are basic elements that form a digital system.
* logic gate ⇒ A digital circuit with one or more input but only one output.
* operation of logic gate can be easily understood with a help of a truth table.

The logic gates are classified in to three types:

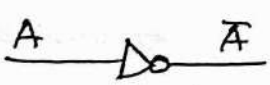
(a) Basic gate → NOT Gate, OR Gate, AND Gate

(b) Universal gate → NAND Gate, NOR Gate

(c) Other gates → Exclusive OR Gate, Exclusive NOR Gate

NOT Gate

* provide output which is complement of the input.



$Y = \bar{A}$

logic symbol

Input A	output Y
0	1
1	0

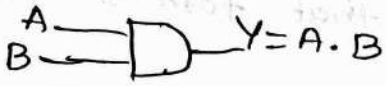
truth table

Boolean Expression

$Y = \bar{A}$

* when a low level is applied to an inverter, a high level will appear on its output
* when a high level is applied to an inverter, a low level will appear on its output

AND Gate



logic symbol

Boolean Expression

$Y = A \cdot B$

Input		output
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

Truth Table

logical multiplication

The output of AND gate is high only when all the inputs are high. If any one of the input is low, the output is low.

OR Gate

$Y = A + B$

Input		output
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

output of OR gate is high, if any one of the input is high.

NAND Gate

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

Input		output
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

* low output occur when all inputs are high.
* high output occur when any one of the input high

NOR gate

$Y = \overline{A + B}$

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

* low output occur when any one of the input is high.
* all inputs are low, output is high.

Exclusive OR gate (EX-OR / XOR)

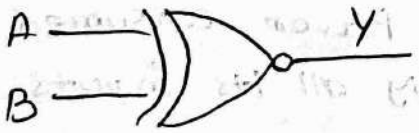
$Y = A \oplus B$

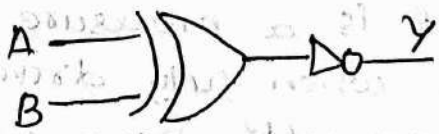
$Y = \overline{A}B + A\overline{B}$

Input		output
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

* output is high if input contain odd number of 1's.
* output is low if input contain even number of 1's

Exclusive NOR gate (EX-NOR / XNOR)





$Y = A \odot B$

Input		output
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

* output of EX-NOR gate is high if the input contain even number of 1's (or) all the inputs are '0'.
* output is low if the input contain odd number of 1's.

Digital logic families

Bipolar logic
(i) Resistor Transistor Logic
(ii) Diode Transistor Logic
(iii) Transistor Transistor logic
(iv) Emitter Coupled logic

Mos logic families
(1) P-channel MOSFET (PMOS)
(2) N-channel MOSFET (NMOS)
(3) Complementar MOSFET (CMOS)

Characteristec of digital logic families

Fan-out: It is a maximum number of similar logic gate that a logic gate can drive without any reducten in voltage levels.

Fan-In: It is the maximum number of inputs connectd to the gates without any reducten in the voltage level.

Noise-margin: Refers to the unwanted signal that affects the performance of the digital logic circuits. Noise margin is the difference between the operating input logic voltage level and threshold voltage.

Power dissipatem: It is a measure of Power consumed by the logic gate when fully driven by all its inputs. unit: milliwatts, microwatts, nano watts.

Positive logic
high - logic 1
low - logic 0

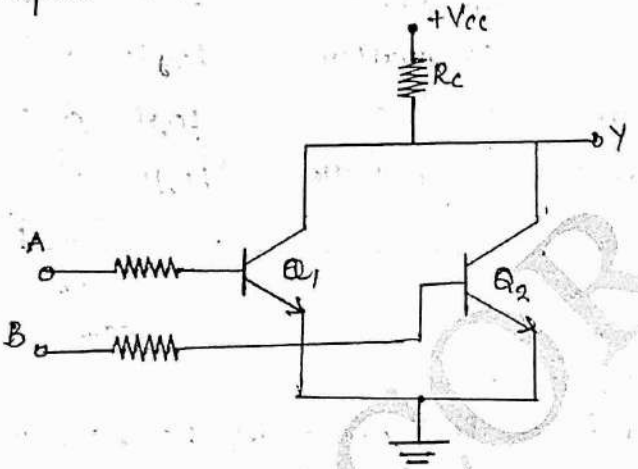
Negative logic
high - logic 0
low - logic 1

1(i). Explain in detail about Resistor transistor logic.

RESISTOR TRANSISTOR LOGIC (RTL)

* RTL circuit consist of resistors and transistors of both input and output stage circuits in a NOR logic gate.
* The emitters of both the transistors are connected to a common ground and collectors of both transistors are tied through a common collector resistor R_c to the supply voltage V_{cc}.
* The resistor R_c is commonly known as passive pull up resistor.

2 input RTL NOR Gate circuit diagram.



Circuit Operation.

* RTL gate input voltage corresponding to low level is required to be low enough for the corresponding transistor to be cut-off.
* When the input voltage corresponding to high level should be high enough to drive the corresponding transistor to saturation
* When both the inputs are low, transistor Q_1 and Q_2 are cut-off and the output is high.
* When the both inputs are high, transistor Q_1 and Q_2 are saturation and the output is low.
* The saturation voltage, $V_{CE(sat)}$ for transistor is approximately 0.2 V. So RTL gates low level output voltage is 0.2.

* A high level output voltage is depends on the number of gates connected to the output.

* The number of gates connected to the output increases output voltage decreases. This is deciding factor for the fanout of the gates.

* The number of gates connected to the output also affects the propagation delay time.

V_A	V_B	Transistor Q_1	Transistor Q_2	V_y
Logic 0	Logic 0	cut-off	cut-off	Logic 1
Logic 0	Logic 1	cut-off	saturation	Logic 0
Logic 1	Logic 0	saturation	cut-off	Logic 0
Logic 1	Logic 1	saturation	saturation	Logic 0

Characteristics of RTL family

* The speed of operation is low
  (The propagation delay is of the order of 500 ns, it can't operate at speeds above 4 MHz.)
* Fanout is 4 or 5 with a switching delay of 50ns, and fanin is 4.
* Poor noise immunity
* Elimination of base resistors in RTL will reduce the power dissipation.
* Sensitive to temperature
* The noise margin from zero to the threshold voltage is about 0.5V, and from one to the threshold voltage is only 0.2V.

Disadvantage

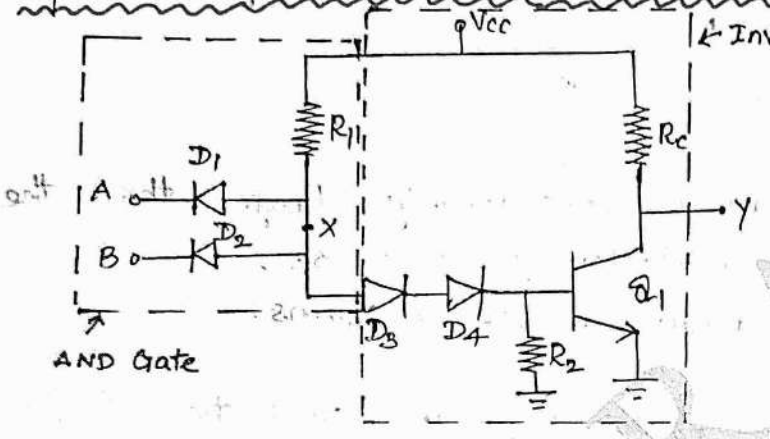
* Low speed
* poor noise immunity

1(i) Explain in detail about diode transistor logic.

DIODE TRANSISTOR LOGIC (DTL)

* The diode transistor logic is some what more complex than RTL. but it have greater fan-out and improved noise margins.

* The circuit consist of diodes and transistors of a two inputs diode transistor logic NAND gate.



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

Circuit Operation.

* When both inputs are low, diode D_1 and D_2 conduct resulting 0·7 volts at point x. This 0·7 voltage at point is not sufficient to drive transistor Q_1.

∴ Q_1 is cut off giving output voltage $V_0 = V_{cc}$ so logic 1.

* When both inputs are high. D_1 and D_2 are reversed biased. This causes the base current of transistor Q_1 to flow through R_1, D_3, D_4 and the base of the transistor Q_1

∴ The Transistor Q_1 is saturation giving output voltage

$V_{CE(sat)} = 0·2$ = Logic 0

* The Diode D_3, D_4. We need increased voltage level to drive transistor in saturation. This improve the noise margin for DTL gate

* when any one input is high (or) Low. The transistor Q_1 is cut-off giving output voltage $V_0 = V_{cc}$. and Logic 1.

Inputs		Diodes		Transistor	output
A	B	D_1	D_2	T	Y
Logic 0	Logic 0	Forward biased	Forward biased	Cut-off	Logic 1
Logic 0	Logic 1	Forward biased	Reverse biased	Cut-off	Logic 1
Logic 1	Logic 0	Reverse biased	Forward biased	Cut-off	Logic 1
Logic 1	Logic 1	Reverse biased	Reverse biased	Saturation	Logic 0

DTL family characteristics

Propagation delay

* The turn off delay is considerably larger than the turn on delay, often by a factor of 2 or 3.
* The propagation delay of DTL is 25ns.

Fan out

* A fan-out as high as 8 is possible with the DTL family because of the high input impedance of the subsequent gates in the logic 1 state.

Fan in

* It has a fan in of 8.

Noise immunity

* The noise magin is high due to the additional diode D_4 connected in series with D_1.

Advantages

* Fanout is high
* Power dissipation is 8-12 mW
* Noise immunity is good

Disadvantages

* More elements are required
* Propagation delay is more
* Speed of operation is less

Encode the binary word 1011 into seven bit even parity Hamming code. [APRIL/MAY 2015]

Soln:

step 1 : Find the number of parity bits required. Let P=3,

$$2^P = 2^3 = 8$$

$$x + P + 1 = 4 + 3 + 1 = 8$$

$2^P \geq x + P + 1$

$8 \geq 7$

Three parity bits are sufficient.

∴ Total code bits = 4+3 = 7

step 2 : construct a bit location table

Bit designation	D_7	D_6	D_5	P_4	D_3	P_2	P_1
Bit location	7	6	5	4	3	2	1
Binary location number	111	110	101	100	011	010	001
Information bits	1	0	1		1		
				0		0	1

step 3 : Determine the parity bits

For P_1 : Bit locations 3, 5 and 7 have three 1s and therefore to have an even parity P_1 must be 1.

For P_2 : Bit locations 3, 6 and 7 have two 1's and therefore to have an even parity P_2 must be 0.

For P_4 : Bit locations 5, 6 and 7 two 1's and therefore to have an even parity P_4 must be 0.

step 4 : Enter the parity bits into the table to form a seven bit Hamming code = 1010101

'The Hamming distance between two code words is defined as the number of bits changed from one code word to another'.

Given that a frame with bit sequence 11010111011 is transmitted, it has been received as 11010 11010. Determine the method of detecting the error using any one error detecting code. [NOV/DEC 2014]

Soln:

Step 1: Construct the bit location table.

Bit designation	D_{10}	D_9	P_8	D_7	D_6	D_5	P_4	D_3	P_2	P_1
Bit location	10	9	8	7	6	5	4	3	2	1
Binary location number	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
Received code	1	1	0	1	0	1	1	0	1	0

Step 2: Check for parity bits

For P_1: P_1 check locations 1, 3, 5, 7 and 9

There are three 1's in the group.

∴ Parity check for ODD parity is correct → 0

For P_2: P_2 check locations 2, 3, 6, 7 and 10

There are three 1's in the group

∴ Parity check for odd parity is correct → 0

For P_4: P_4 check location 4, 5, 6 and 7

There are three 1's in the group

∴ Parity check for ODD parity is correct → 0

For P_8: P_8 check location 8, 9 and 10

There are two 1's in the group

∴ Parity check for ODD parity is wrong → 1

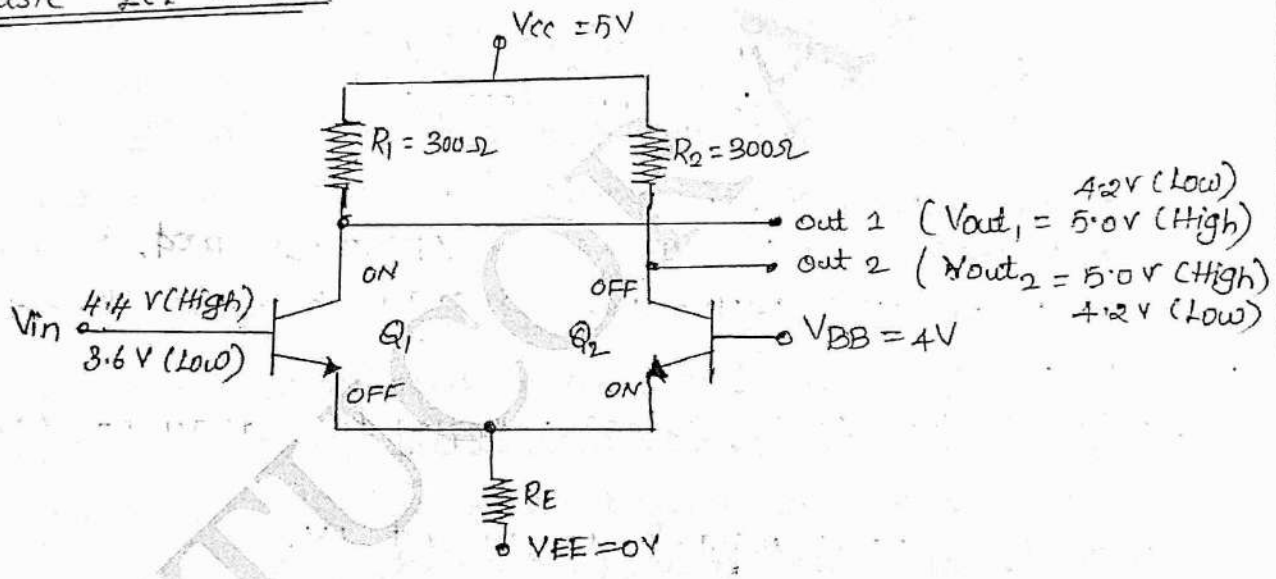
The resultant word is 0001, this says that the bit in the number 1 location is in error. It is 0 and should be a '1'. Therefore, the correct code is 11010 11011, which agrees with the transmitted code.

2: ^ ^ ^ ^ about Emitter coupled logic.

Emitter Coupled Logic (ECL) [NOV/DEC. 2012] [MAY/JUNE 2013], [MAY 2014] [APRIL/MAY 2015]

* TTL family uses transistors operating in the saturation mode. as a result their switching speed is limited by the storage delay time associated with a transistor is driven into saturation
* Another logic family has been developed that prevents transistor saturation. there by increasing overall switching speed by using radically different structure is called "current mode logic (CML). This family is also known as "Emitter coupled logic".
* ECL does not produce a large voltage swing between the low and high levels. It internally switches current between two possible path depending on the output state.

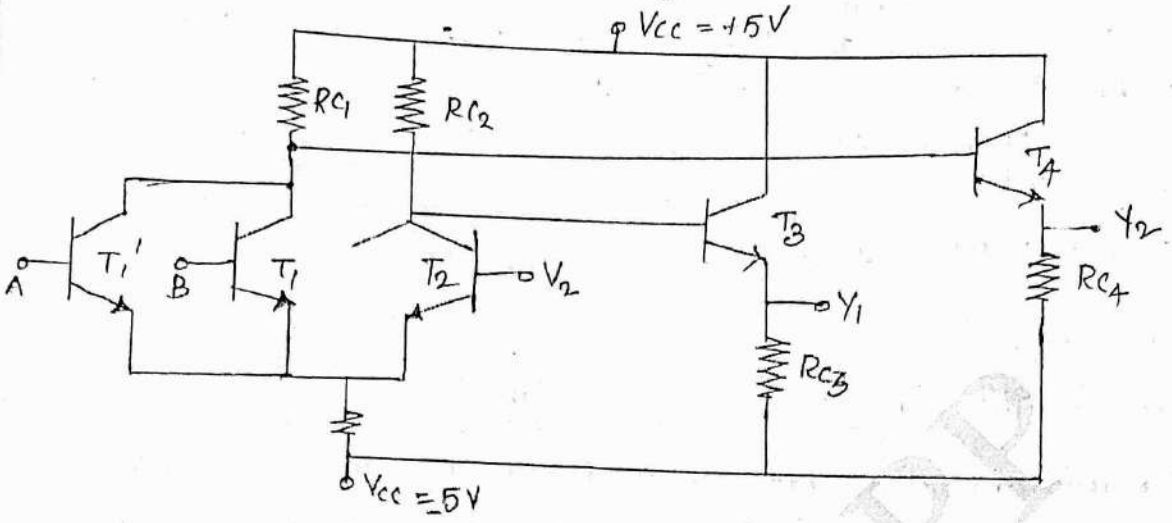
Basic ECL circuit



Circuit operation

* When input voltage Vin is high (4·4V), transistor Q_1 is ON, but not saturated and transistor Q_2 is OFF. V_{out2} is pulled to 5·0 V (High) through R_2 and drop across R_1 is 0·8V. So that V_{out1} is 4·2V (low)
* When input voltage Vin is low (3·6V), transistor Q_2 is ON, but not saturated and transistor Q_1 is OFF. Thus V_{out1} is pulled to 5·0 V (High) through R_1 and drop across R_2 is 0·8V so that V_{out2} is 4·2V (Low).

Emitter coupled Logic OR/NOR gate



* The circuit consist of emitter followers are used at the output of difference amplifier to shift the DC level.

* The circuit has two outputs Y_1 and Y_2, which are complementary. Y_1 corresponds to OR logic and Y_2 corresponds to NOR Logic.

Circuit Operation.

* When both inputs are Logic '0'. T_1 and T_1' operate in cut-off and T_2 operates in active region, voltage V_{O1} is high T_3 is ON and the output at Y_2 is logic '1', voltage V_{O2} is low. T_4 operates in cut off & output Y_1 is logic '0'.

* When any one of the input is logic '1', the transistors T_1 and T_1' are operated in active region and T_2 operates in cut-off, voltage V_{O1} is low, T_3 operates in cutoff & Y_2 is logic 0, voltage V_{O2} is high, T_4 operates in active region and Y_1 is logic '1'.

* When both inputs are logic 1 state T_1 and T_1' operate in active region and T_2 operates in cutoff, voltage V_{O1} is low, T_3 operates in cut-off and Y_2 is logic '0', voltage V_{O2} is high, T_4 operates in active region and Y_1 is Logic 1.

Inputs		Transistors			Transistors		Output	
A	B	T_1	T_1'	T_2	T_3	T_4	Y_1	Y_2
Logic 0	Logic 0	Cut off	Cut off	Active	Active	Cut off	Logic 0	Logic 1
Logic 0	Logic 1	cut off	Active	Cut off	cut off	Active	Logic 1	Logic 0
Logic 1	Logic 0	Active	Active	cut-off	cut off	Active	Logic 1	Logic 0
Logic 1	Logic 1	Active	Active	cut off	Cut off	Active	Logic 1	Logic 0

ECL characteristics.

* It is the fastest of Logic families.
* Transistors are not allowed to go into complete saturation, and thus eliminating storage delays
* To prevent transistors from going into complete saturation, logic levels are kept close to each other.
* Noise margin is reduced and it is difficult to achieve good noise immunity
* Another disadvantage of this approach is that power consumption is more because transistors are not completely saturated.
* Switching transients are less because power supply current is more suitable stable than in TTL and cmos circuits.

Advantages:

* Fastest Logic family
* Good noise immunity

Disadvantage:

* Power consumption is more

4(ii). Draw & Explain the CMOS NAND & CMOR NOR gates?

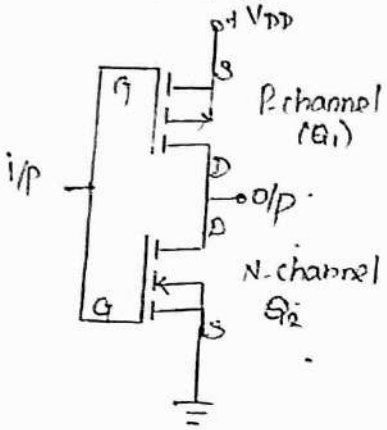
CMOS LOGIC

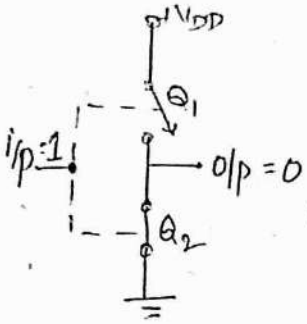
[MAY/JUNE 2013] [NOV/DEC 2015] [NOV/DEC 2014]

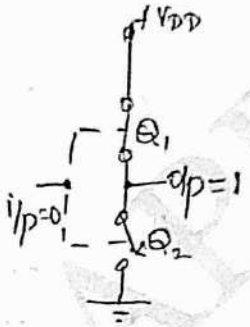
* Complementary Metal Oxide Semiconductor (CMOS) circuits contain both NMOS and PMOS devices to speed the switching of capacitive loads.

* It consumes low power and can be operated at high voltages, resulting in improved noise immunity.

CMOS Inverter







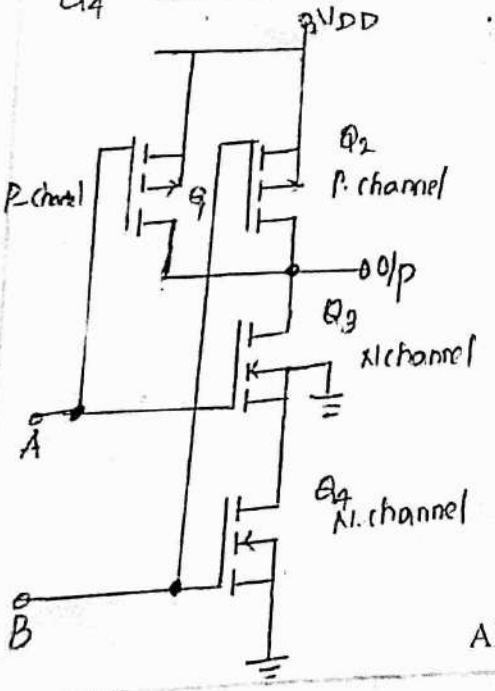
A	Q_1	Q_2	o/p
0	ON	OFF	1
1	OFF	ON	0

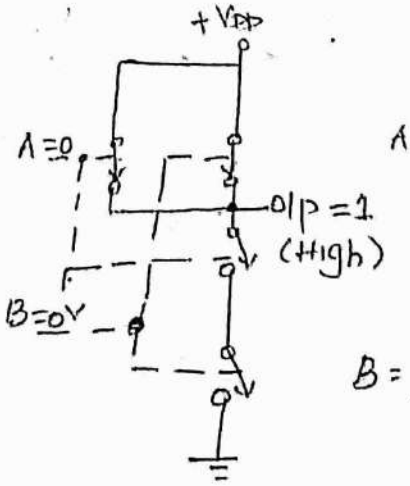
Circuit operation

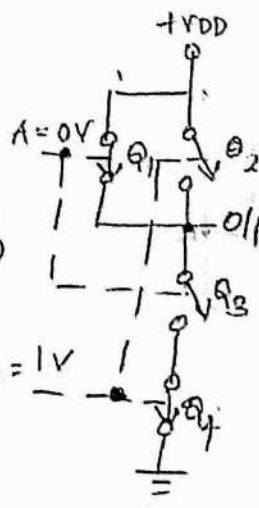
* When the input is low, Q_1 is ON and Q_2 is OFF, output is high.
* When the input is high, Q_1 is OFF and Q_2 is ON, output is low.

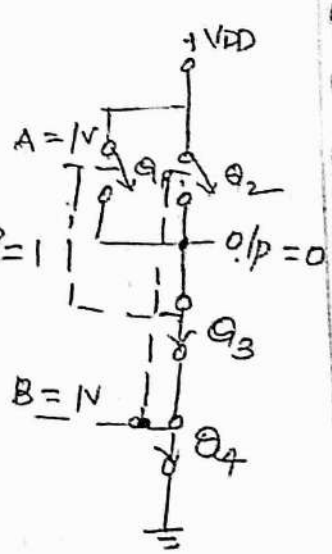
CMOS NAND GATE

* CMOS 2-input NAND gate consist of two P-channel MOSFETs, Q_1 and Q_2, connected in Parallel and two N-channel MOSFET Q_3 and Q_4 connected in Series.









Circuit operation of 2 input CMOS NAND Gate.

* When the inputs are low, Q_1 and Q_2 are ON, Q_3 and Q_4 are OFF, and the output is high (V_{DD})

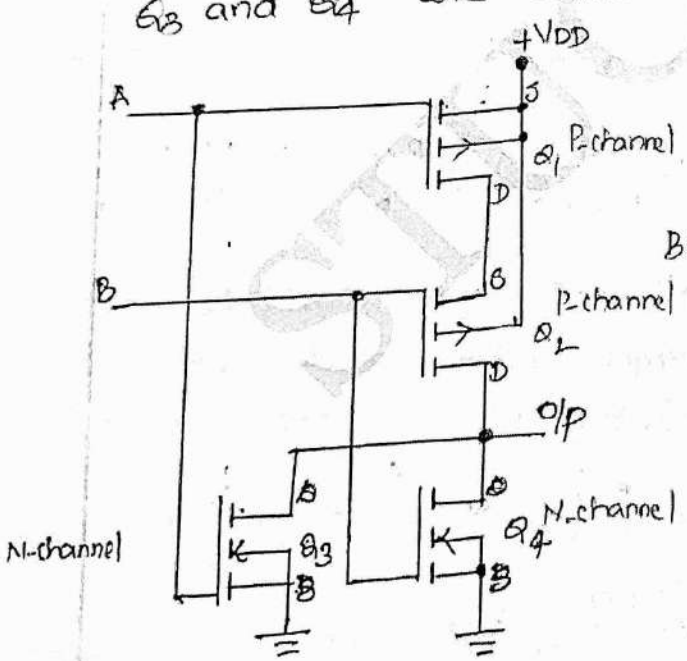
* When any one of the inputs is low (0V or $-V_e$), then the corresponding MOSFET Q_1 or Q_2 is ON, Q_3 or Q_4 is ON and the output is high.

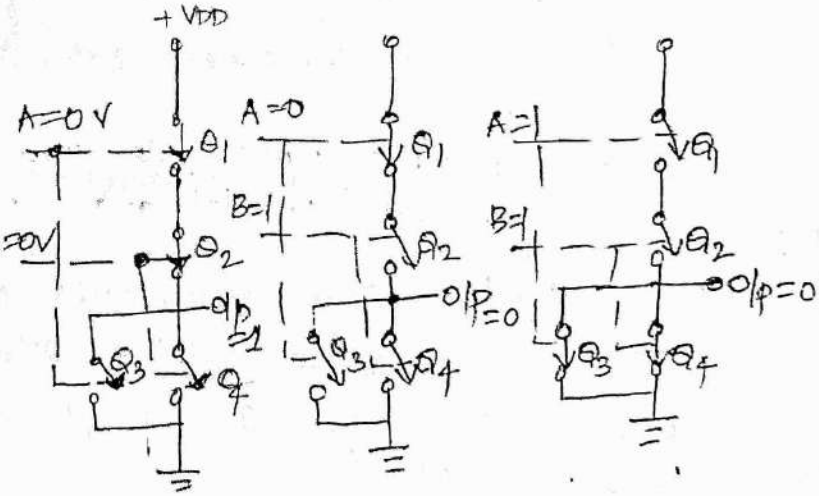
* When the inputs are high, Q_1 and Q_2 are OFF, Q_3 and Q_4 are ON and the output is low.

A	B	Q_1	Q_2	Q_3	Q_4	V_o (output)
0	0	ON	ON	OFF	OFF	1
0	1	ON	OFF	OFF	ON	1
1	0	OFF	ON	ON	OFF	1
1	1	OFF	OFF	ON	ON	0

CMOS NOR GATE

* CMOS 2 input NOR gate consist of two P-channel MOSFET Q_1 and Q_2 are connected in series and N-channel MOSFETs Q_3 and Q_4 are connected in parallel.





Advantages:

* Consumes less power
* can be operate high voltages
* Improved noise immunity
* Fan out more
* Better noise margin

Disadvantages

* Switching speed low
* Greater propagation delay

Circuit operation of 2 input CMOS NOR Gate.

* When both inputs are low, Q_1 and Q_2 are ON, Q_3 and Q_4 are OFF and the output is high (V_{DD}).

* When any one of the inputs is low (0V or -Ve), then the corresponding MOSFET Q_1 or Q_2 is ON, Q_3 or Q_4 is ON and the output is low.

* When inputs are high, Q_1 and Q_2 are OFF, Q_3 and Q_4 are ON, and the output is low.

A	B	Q_1	Q_2	Q_3	Q_4	V_o
0	0	ON	ON	OFF	OFF	1
0	1	ON	OFF	OFF	ON	0
1	0	OFF	ON	ON	OFF	1
1	1	OFF	OFF	ON	ON	0

Characteristics OF mos Logic.

* MOS Logic families are slower operating speed, it require much less power, have better noise margin & higher fanout.

Operating speed

* A typical Nmos NAND gate has a propagation delay of 50ns.

Noise margin

* The Nmos noise margins are around 1V.

Fan-out

* The fan-out capabilities of mos logic would be virtually unlimited because of the extremely high input resistance at each MOSFET input. It easily operate at a fan-out value of 50

Power dissipation

* The power dissipation of a cmos Ic is very low as long as it is in a dc condition.

* Unfortunately power dissipation of Cmos Ic increases in proportion to the frequency at which the circuits are switching states.

Propagation delay:

* the propagation delay in cmos is the sum of delay due to internal capacitance & APP due to the load capacitance

5. Design a TTL logic circuit for a 3 input NAND gate

TRANSISTOR TRANSISTOR LOGIC (TTL) [APRIL/MAY 2015], [NOV/DEC 2014]

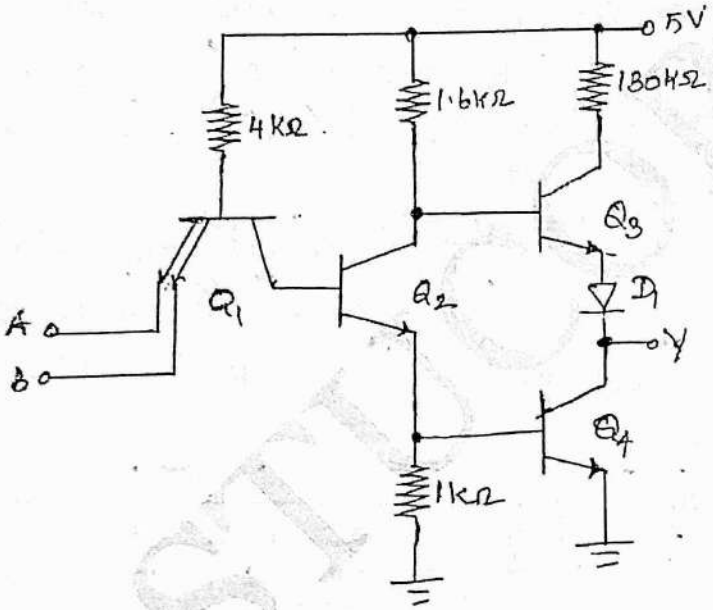
* Transistor transistor logic, TTL is named for its dependance on transistors alone to perform basic logic operation.

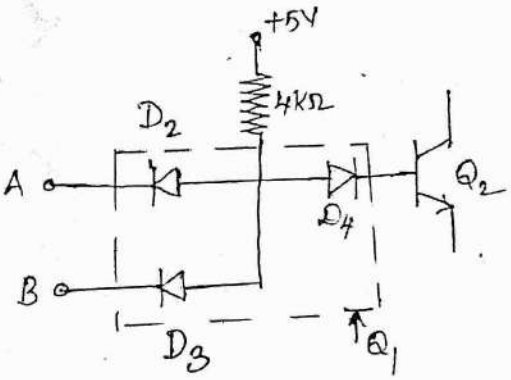
* The first version, which is now known as standard TTL 2-input TTL NAND Gates.

* The 2 input TTL NAND gate input structure consists of multiple emitter transistor and output structure consist of totem pole output.

* The transistor Q_1 having two emitters, one for each input to the gate.

* Diodes D_2 and D_3 represent the two emitter base junction of Q_1 and D_4 is the collector-base junction of Q_1.





Input		output
A	B	Y
Logic 0	Logic 0	Logic 1
Logic 0	Logic 1	Logic 1
Logic 1	Logic 0	Logic 1
Logic 1	Logic 1	Logic 0

Operation

* When both input voltage A and B are low. The transistor Q_1 is ON and the output voltage of Q_1 is almost zero. Therefore Q_2 is cutoff.

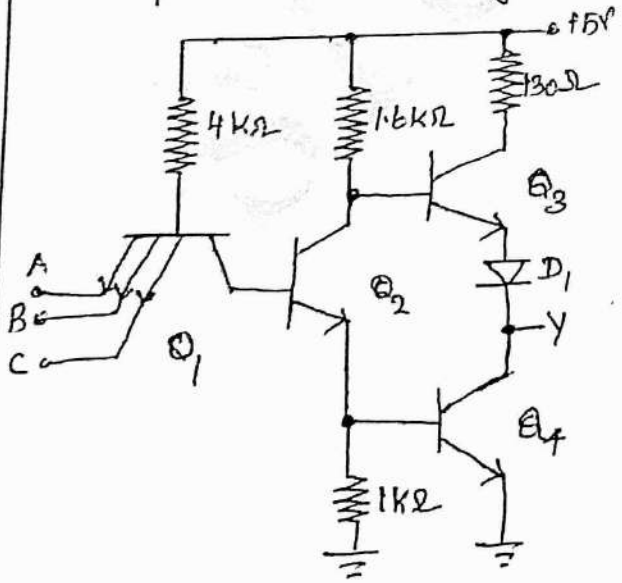
* When Q_2 is open (or) cutoff the output voltage of Q_2 is high. So the transistor Q_3 base is pulled high. Since Q_3 act as an emitter follower, the output Y is pulled up to a high voltage.

* When both input voltages are low (high). The transistor Q_1 is OFF (or) cutoff and output voltage of Q_1 is high. Therefore Q_2 is saturated. (or) ON.

* When the transistor Q_2 is ON the output voltage of Q_2 is low (or) almost zero. So the transistor Q_3 base is pulled down to low value. The transistor Q_4 is conducted. the output voltage is low value.

* When any one of the input voltage are low (or) high the output voltage of Q_1 is zero. So Q_2 is open. since. output voltage Y is pulled up to high value.

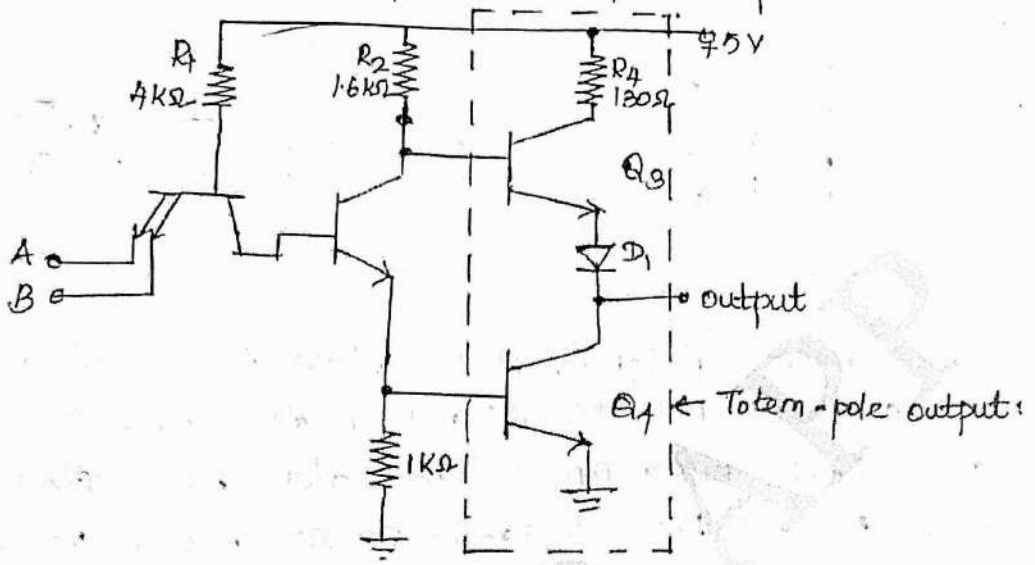
3-input TTL NAND gates.



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

TOTEM POLE OUTPUT

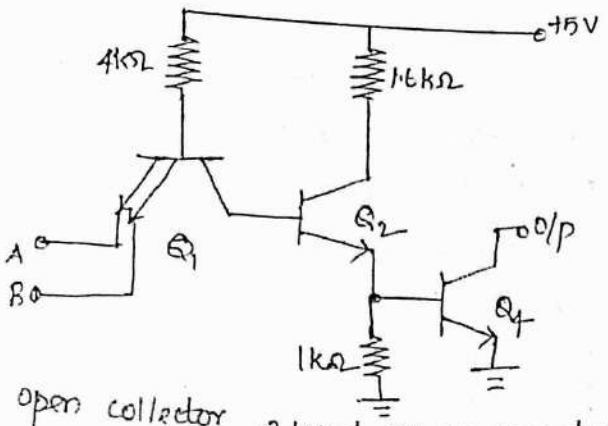
* Transistor Q_3 and Q_4 form a totem-pole. such a configuration is known as active pull-up or totem pole output.



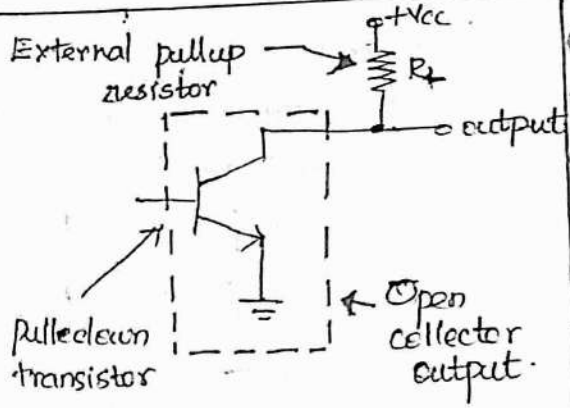
* Totem-pole transistors produce a low output impedance.
* Either Q_3 acts as an emitter follower (high output) or Q_4 is saturated (low output).
* When Q_3 is conducting, the output impedance is approximately 70Ω.
* When Q_4 is saturated, the output impedance is only 12Ω the output impedance value is low. this means the output voltage can change quickly from one state to the other because any stray output capacitance is rapidly charged or discharged through the low output impedance.
* The propagation delay is low in totem pole TTL logic.

Open-Collector output.

* TTL devices provide another type of output called open collector output.
* The output of two different gates with open collector output can be tied together. This is known as wired Logic.
* A 2 input NAND gate with an open collector output eliminates the pull up transistor Q_3, D_1 and R_4.
* The output is taken from the open collector terminal of transistor Q_4.



Open collector 2 input TTL NAND gate



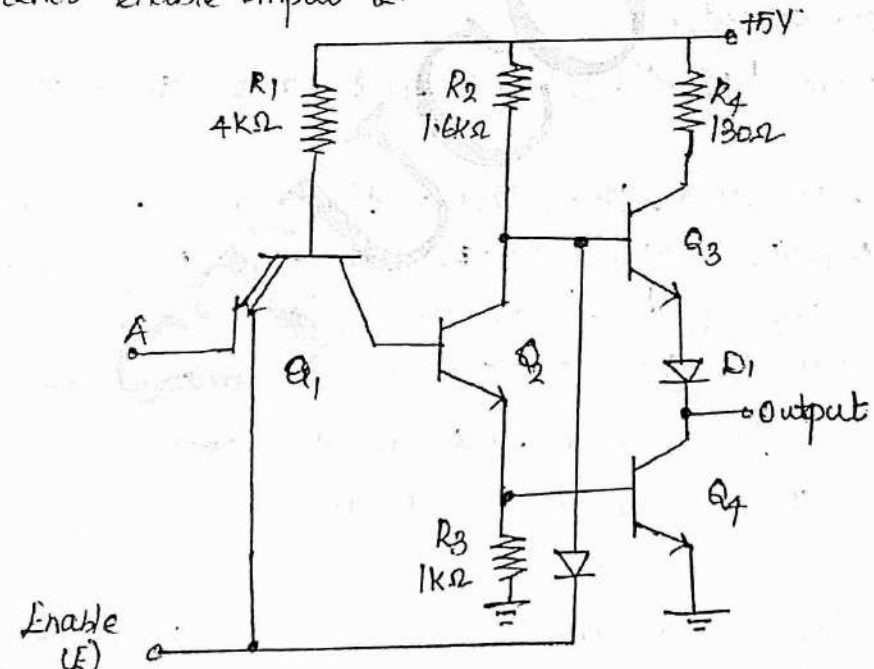
Open collector output with pullup resistor.

*The collector of Q4 is open, a gate like this will not work properly until connect an external pullup resistor.

*when Q4 is ON, the output is low and when Q4 is OFF output is tied to Vcc through an external pullup resistor.

Tri state TTL inverter.

*The tristate TTL inverter has two inputs - normal input A and enable input E.



*It utilizes the high speed operation of the totem-pole arrangement while permitting outputs to be connected together. It is called tristate TTL because it allows three possible output stage High, Low and high impedance.

* When the transistor Q_3 is ON when output is high and transistor Q_4 is ON output voltage of transistor Q_4 is low.

* In high impedance state both transistor Q_3 and Q_4 in the totem pole arrangement are turned off.

* The result of output is open or floating. neither low or high

* 'A' is the normal logic input whereas E is an ENABLE input.

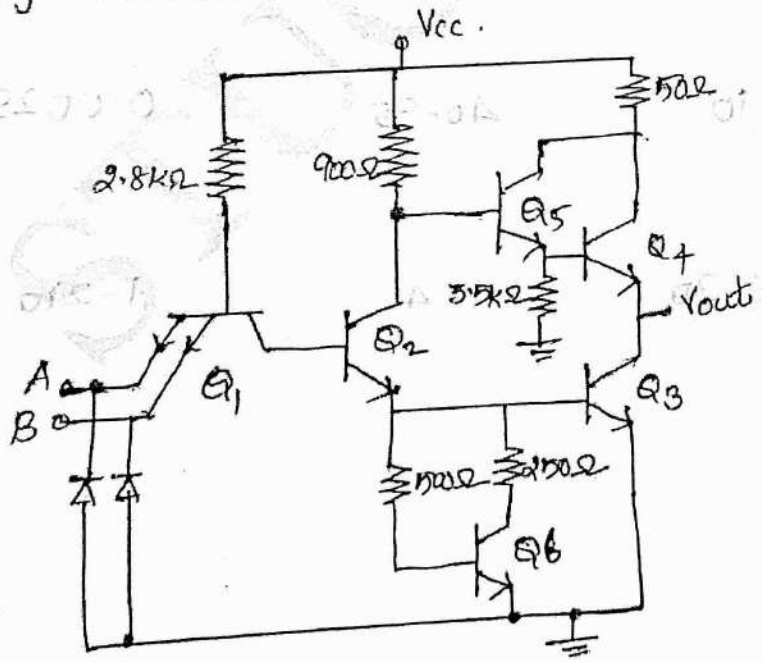
* When ENABLE input is high the circuit works as a normal inverter.

* When E is high the state of transistor Q_1 depends on logic input A

* When ENABLE input is low, regardless of the state of logic input A, the base emitter junction of Q_1 is forward biased and result it turn ON.

* When ENABLE input is low, both transistor Q_3, Q_4 is OFF and output is at high impedance state.

Schottky TTL Gate:



Advantages of TTL:

* High speed
* Propagation delay 10ns
* Moderate power dissipation.
* Low cost.
* Moderate packaging density

Disadvantages of TTL:

* Higher power dissipation than CMOS.
* Lower noise immunity than CMOS
* Less fanout than CMOS.

Comparison of TTL, ECL, CMOS

Specification	TTL	ECL	CMOS
Components	Transistor Passive Elements	Transistor passive elements	MOSFETs
Basic Gate	NAND	OR/NOR	NAND/NOR
Noise immunity	Strong	Good	Very strong
Fan-out	10	25	more than 25
Noise margin	Moderate	Low	high
clock rate	35	>60	10
Power/gate (mWatt)	10	40-55	0.0025
t_{PD} (ns)	1.5-30	1-4	1-210