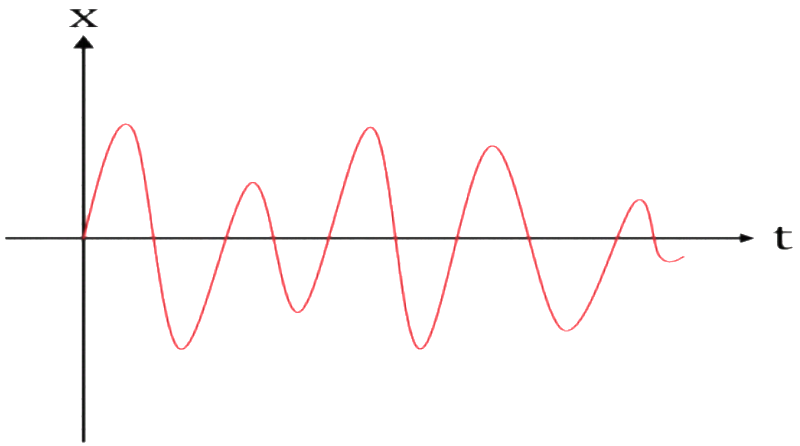
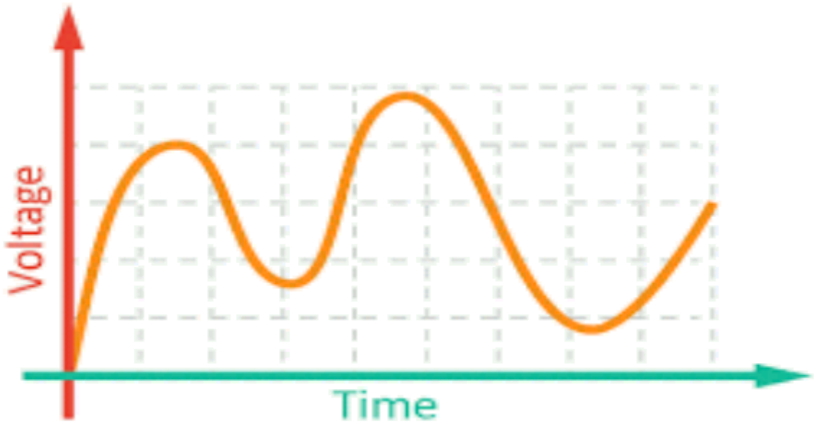
Chapter-1: Fundamentals of Digital Systems and logic families

1.1 Digital signals

- **Signal:** A physical quantity, which contain some information and which is function of one or more variables.
- **Type of Signal:**
  1. Analog Signals
  2. Digital Signals
- **Analog Signals:** Signal having continuous values and infinite number of different values.
  Examples: Things observed in nature are analog like Temperature, Pressure, Distance, Sound and Current.





§ **Digital Signals:** Signal which has only a finite number of distinct values.

§ **Type of Digital Signals:**

Type of Digital Signal	Number of Distinct values
Binary	2
Octal	8
Hexadecimal	16

§ **Source of Digital Signals:** Signals obtained directly from computers, Output of A to D converter.

§ **Advantages of Digital Signals:**

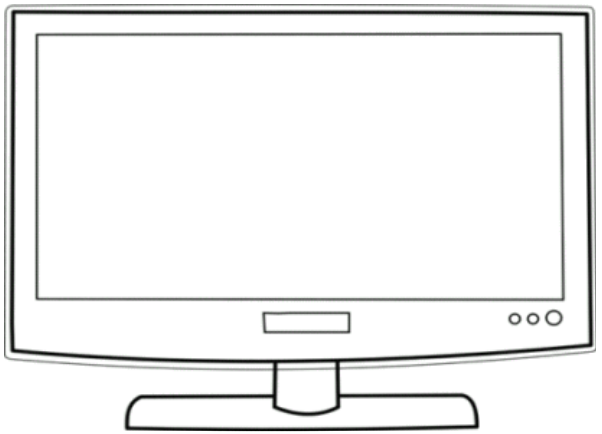
- § Processed and transmitted more efficiently.
- § Possible to stored data.
- § Less noise effected.

1.2 Digital Circuits/Systems

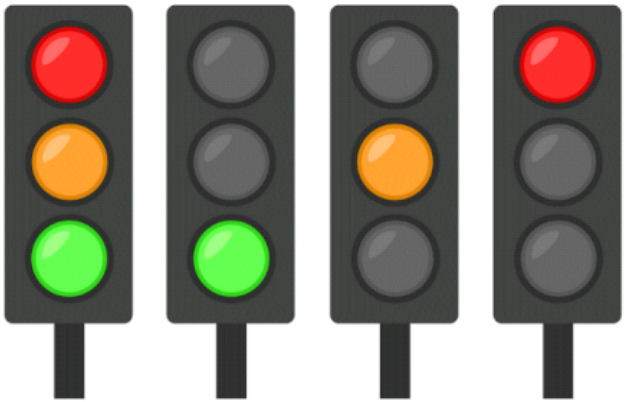
Digital circuits have input signal and output signal both are digitals.

Examples:

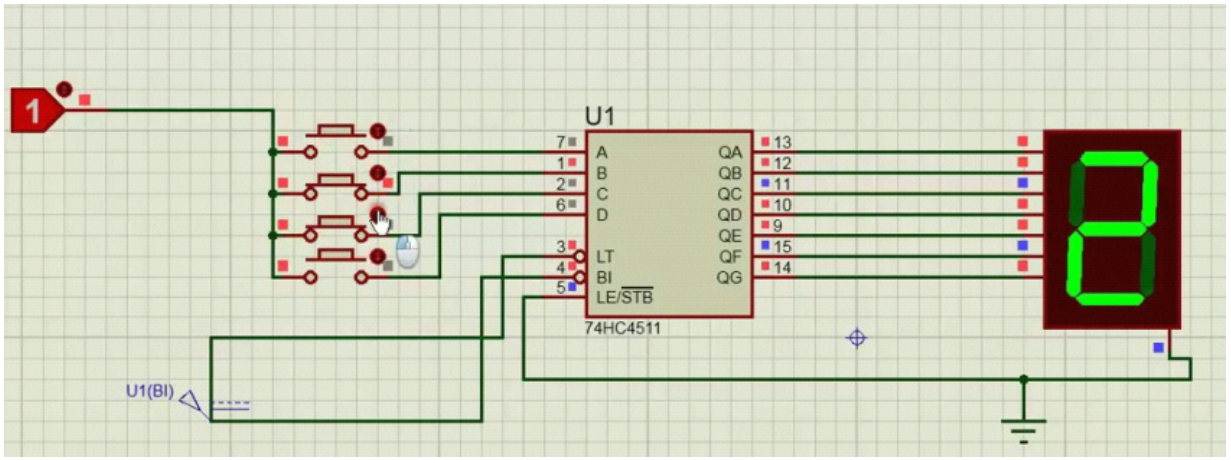
Digital Computer



Digital Television



Traffic control system



BCD to Seven Segment convertor

1.3 Number Systems

§ **Number System:** Sets of values used to represent quantity like number of students attending classes, Grades archived by students in tests.

§ **Various Number Systems:**

Name of Number system	Base (radix-r)	Number of Digits (r)	Largest value of Digit (r-1)	Weight value (r)	Range (0 to r-1)
Binary	2	2	1	2	0,1
Octal	8	8	7	8	0-7
Decimal	10	10	9	10	0-9
Hexadecimal	16	16	15	16	0-9, A-F

Decimal Number System

- Base (also called radix) = 10
  - 10 digits { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }
- Digit Position
  - Integer & fraction
- Digit Weight
  - Weight = (*Base*) ^{Position}
- Magnitude
  - Sum of "*Digit* x *Weight*"
- Formal Notation

2	1	0		-1	-2
5	1	2	.	7	4
100	10	1		0.1	0.01
500	10	2		0.7	0.04

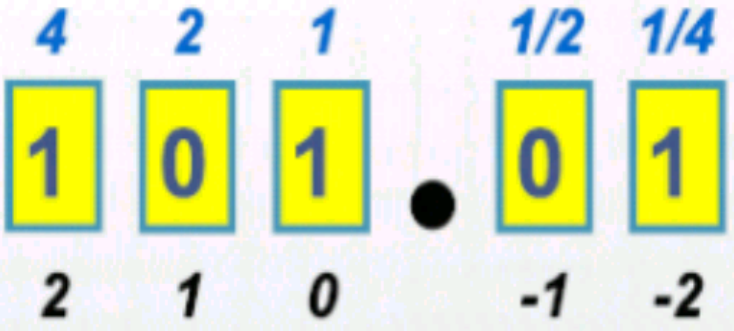
$$d_2*B^2+d_1*B^1+d_0*B^0+d_{-1}*B^{-1}+d_{-2}*B^{-2}$$

$$(512.74)_{10}$$

Binary Number System

- Base = 2
  - 2 digits { 0, 1 }, called *b*inary dig*its* or "*bits*"
- Weights
  - Weight = (*Base*) ^{Position}
- Magnitude
  - Sum of "*Bit* x *Weight*"
- Formal Notation
- Groups of bits 4 bits = *Nibble*

  8 bits = *Byte*



$1*2^2+0*2^1+1*2^0+0*2^{-1}+1*2^{-2}$

$=(5.25)_{10}$

$(101.01)_2$

1 0 1 1

1 1 0 0 0 1 0 1

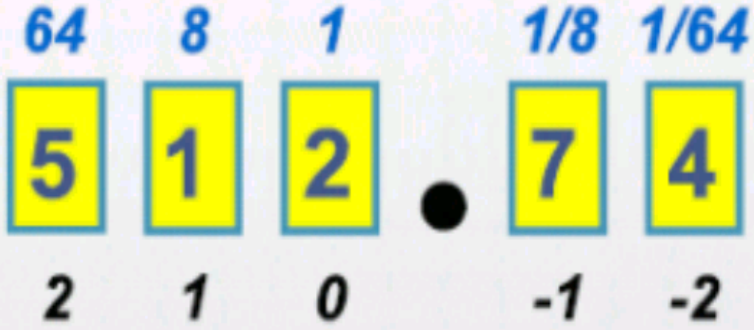
The Power of 2

n	2^n
0	$2^0=1$
1	$2^1=2$
2	$2^2=4$
3	$2^3=8$
4	$2^4=16$
5	$2^5=32$
6	$2^6=64$
7	$2^7=128$

n	2^n	
8	$2^8=256$	
9	$2^9=512$	
10	$2^{10}=1024$	Kilo
11	$2^{11}=2048$	
12	$2^{12}=4096$	
20	$2^{20}=1M$	Mega
30	$2^{30}=1G$	Giga
40	$2^{40}=1T$	Tera

Octal Number System

- Base = 8
  - 8 digits { 0, 1, 2, 3, 4, 5, 6, 7 }
- Weights
  - Weight = (*Base*) ^{Position}
- Magnitude
  - Sum of "*Digit* x *Weight*"
- Formal Notation



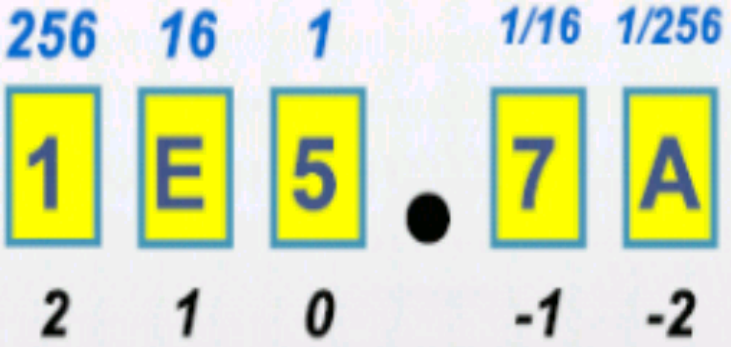
$5*8^{2}+1*8^{1}+2*8^{0}+7*8^{-1}+4*8^{-2}$

$=(330.9375)_{10}$

$(512.74)_{8}$

Hexadecimal Number System

- Base = 16
  - 16 digits { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F }
- Weights
  - Weight = $(Base)^{Position}$
- Magnitude
  - Sum of "*Digit* x *Weight*"
- Formal Notation



$$1*16^{2}+14*16^{1}+5*16^{0}+7*16^{-1}+10*16^{-2}$$

$$=(485.4765625)_{10}$$

$$(1E5.7A)_{16}$$

Quiz Time:

1. Represent the hexadecimal number $(6DE)_{16}$

2. Represent the number $(423.6)_7$

3. Count from 0 to 8 in radix 6 system.

1.3.1 Number System Conversion

§ Conversion from any radix to Decimal:

1. Note down given number.
2. Write the weight of different positions.
3. Multiply each digit with corresponding weight to obtain product numbers.
4. Add all product numbers.

Example: Convert $(3124)_5$ to Decimal

$$3124_5 \rightarrow = 3\times5^3 + 1\times5^2 + 2\times5^1 + 4\times5^0$$

$$= 3\times125 + 1\times25 + 2\times5 + 4\times1$$

$$= 375 + 25 + 10 + 4$$

$$= 414_{10}$$

§ Conversion from Decimal to other radix:

1. Separate integer and fractional parts.
2. For Integer part:
   1) Divide the integer by the base until there is nothing to left.
   2) Keeping track of remainders from each step.
   3) List the remainder values in reverse order to find the equivalent.
1. For Fractional part:
   1) Multiply the fractional part by the radix(r).
   2) Record the carry generated in this multiplication as MSD.
   3) Multiply only the fractional part of product in step-2 by the radix.
   4) Repeat step-2 and 3 up to end. Last carry will represent the LSD.
1. Combine result of Integer part and Fractional part.

Example: Convert the following number in octal

$(175)_{10}$

	Quotient	Remainder	Coefficient
175 / 8 =	21	7	$a_0 = 7$
21 / 8 =	2	5	$a_1 = 5$
2 / 8 =	0	2	$a_2 = 2$

Answer: $(175)_{10} = (a_2\, a_1\, a_0)_8 = (257)_8$

$(0.3125)_{10}$

	Integer	Fraction	Coefficient
0.3125 * 8 =	2	5	$a_{-1} = 2$
0.5 * 8 =	4	0	$a_{-2} = 4$

Answer: $(0.3125)_{10} = (0.a_{-1}\, a_{-2}\, a_{-3})_8 = (0.24)_8$

§ Conversion from Binary to Octal:

1. Group the binary bits into groups of 3 starting from LSB.
2. Convert each group into its equivalent octal.

Example: Convert the following number into octal:
1001011010111.1011111_2

$$\underline{010}\ \underline{010}\ \underline{110}\ \underline{101}\ \underline{111}\ .\ \underline{101}\ \underline{111}\ \underline{100}_2$$
$$\Downarrow\quad \Downarrow\quad \Downarrow\quad \Downarrow\quad \Downarrow\quad \Downarrow\quad \Downarrow\quad \Downarrow$$
$$2\quad 2\quad 6\quad 5\quad 7\ .\ 5\quad 7\quad 4_8$$

Answer = 22657.574_8

§ Conversion from Binary to Hex:

3. Break the binary number in to 4-bit sections from LSB to MSB
4. Convert each 4-bit binary number into hex equivalent.

Example:

$(1101011.00101)_2$

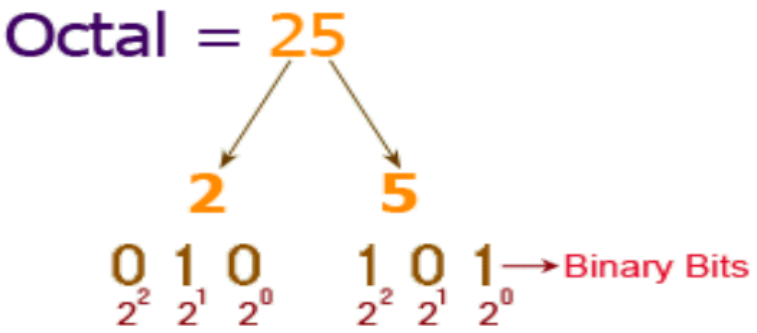
1101011.00101

01101011.00101000

§ Conversion from Octal to Binary:

1. Convert each octal digit into its equivalent 3-bit binary number.

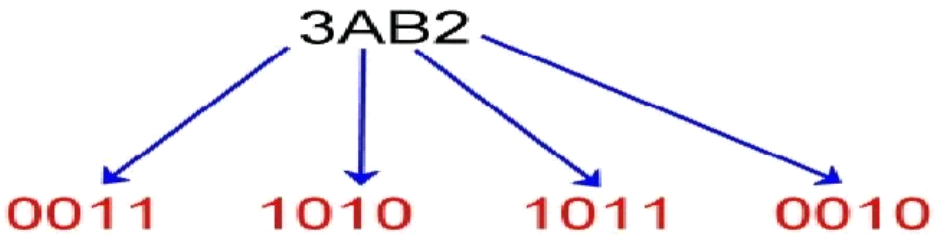
Example: Convert $(25)_8$ to Binary.



§ Conversion from Hex to Binary:

2. Convert each hex digit to its 4-bit binary equivalent.
3. Combine the 4-bit sections by removing the spaces.

Example: Convert $(3AB2)_{16}$ to Binary.

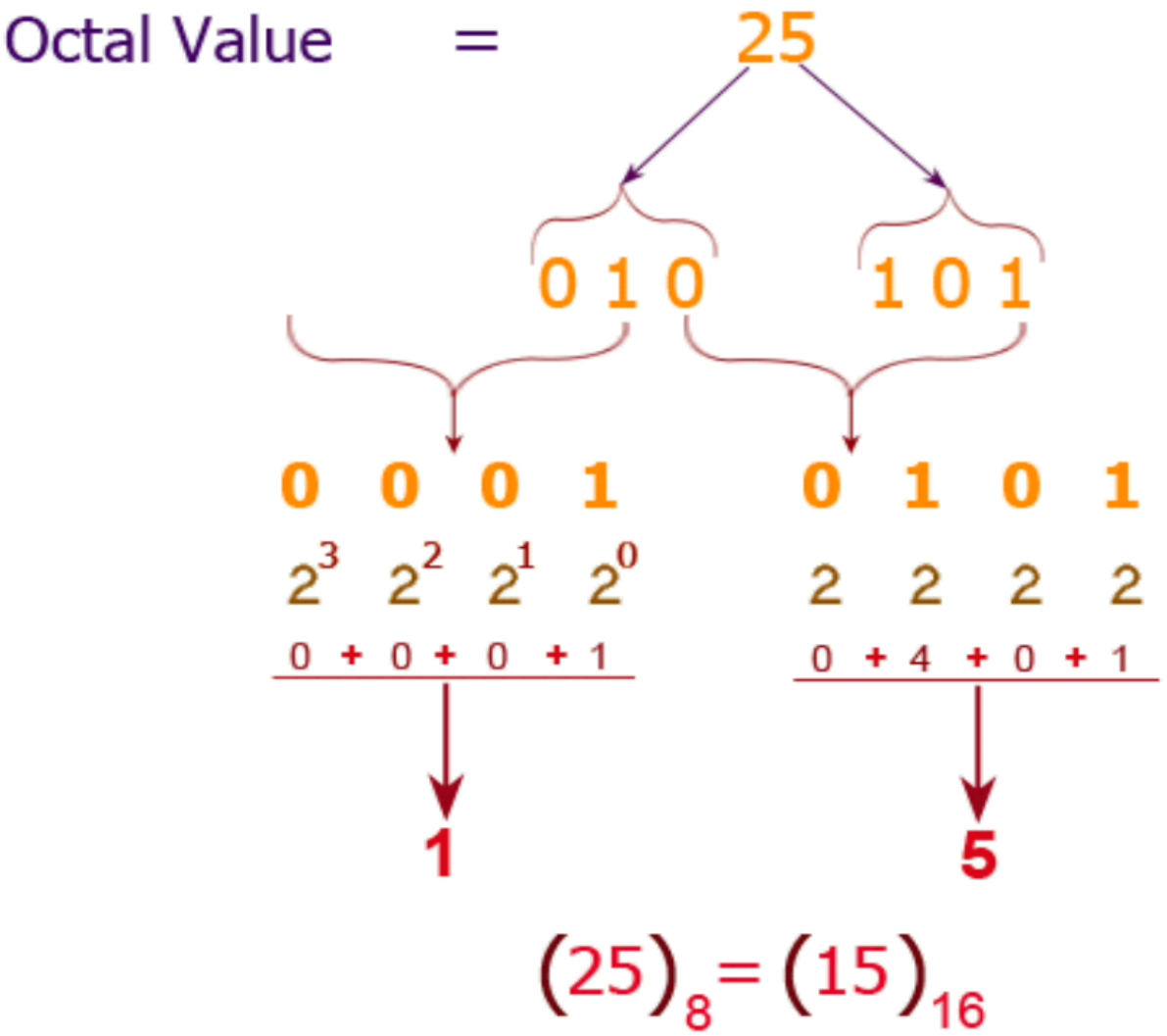


$3AB2_{16} = 11101010110010_2$

§ Conversion from Octal to Hex:

1. Convert the given octal number into equivalent binary.
2. Then convert this binary number into hex.

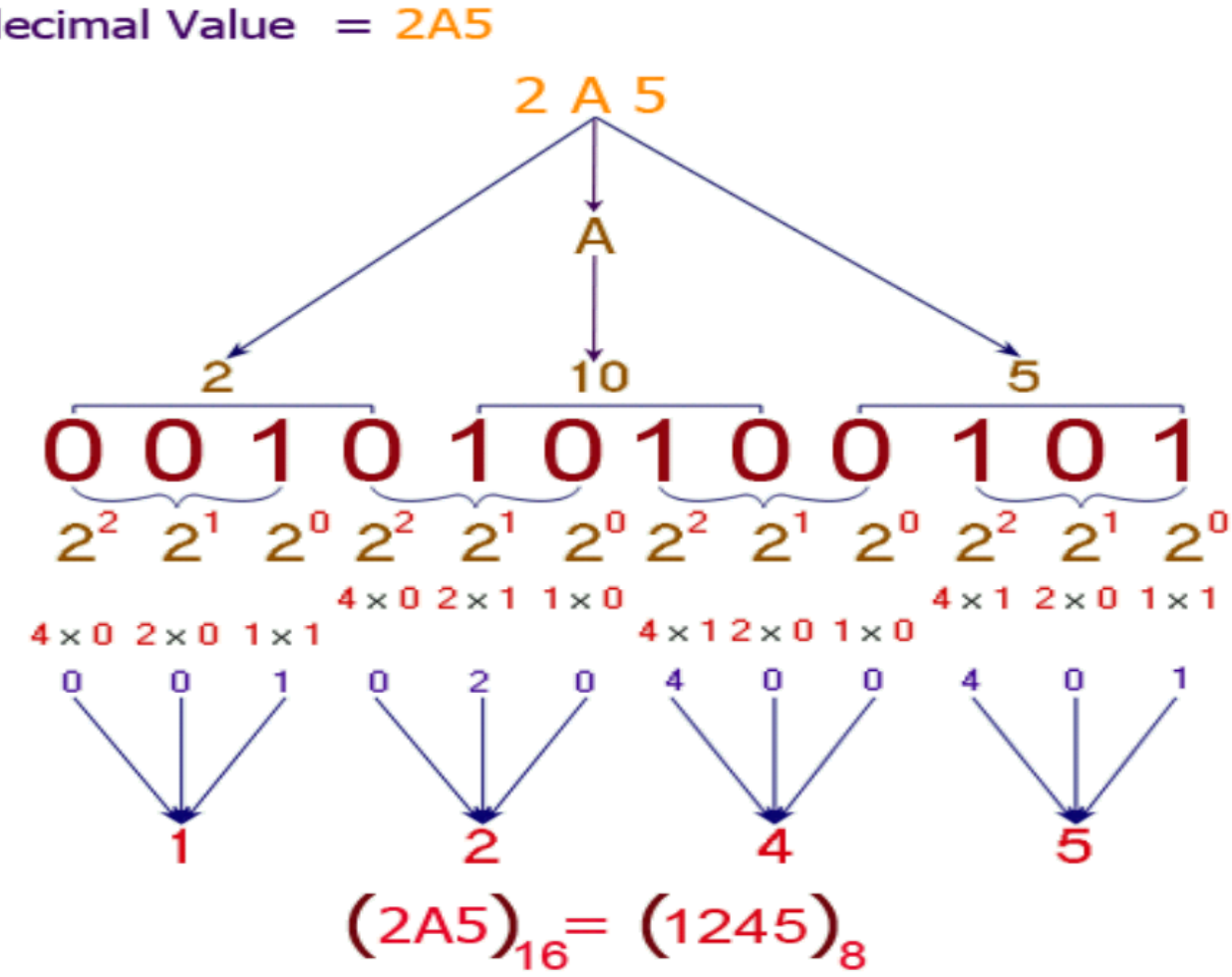
Example: Convert $(25)_8$ to Hex.



§ Conversion from hex to Octal:

1. Represent each hex digit by a 4-bit binary number.
2. Combine these 4-bit binary sections by removing the spaces.
3. Now group these binary bits into groups of 3 bits, starting from the LSB side.
4. Then convert each of this 3 bit group into an octal digit.

Example:



Binary Addition

Binary Addition

A + B	SUM	CARRY
0 + 0	0	0
0 + 1	1	0
1 + 0	1	0
1 + 1	0	1

EXAMPLE

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

$$\begin{array}{rr} & 1\ 1\ 1\quad 1\quad 1 \\ (85)_{10} & 01010101 \\ +(181)_{10} & +\ 10110101 \\ \hline (266)_{10} & 100001010 \end{array}$$

Binary Subtraction

A	B	DIFFERENCE	BORROW
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

i) 101 from 1001

Solution:

101 from 1001

1 Borrow

1 0 0 1

1 0 1

1 0 0

1.4 Binary arithmetic

§ **Binary Subtraction:** Its creates Difference(D) and Borrow(B).

A	B	DIFFERENCE	BORROW
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Example: Subtract binary number A= $(0011010)_2$ and B= $(0001100)_2$.

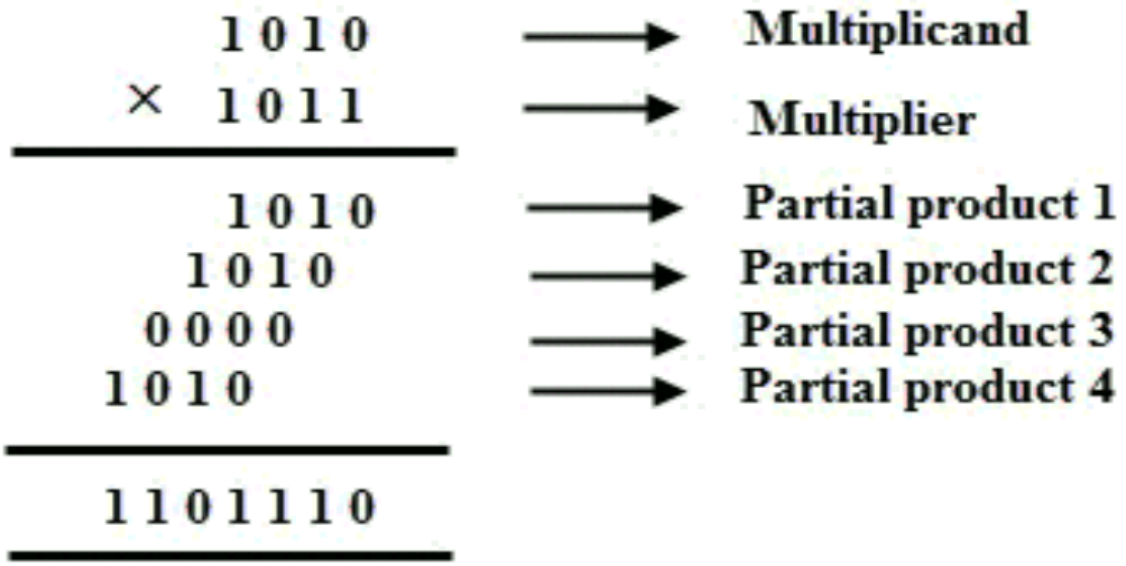
$$
\begin{array}{lrl}
 & & 1\ \ 1 \\
(26)_{10} & & 0011010 \\
-(12)_{10} & - & 0001100 \\
\hline
(14)_{10} & & 0001110
\end{array}
$$

1.4 Binary arithmetic

§ **Binary Multiplication:** It is exactly same as decimal multiplication.

Input A	Input B	Multiply (M) AxB
0	0	0
0	1	0
1	0	0
1	1	1

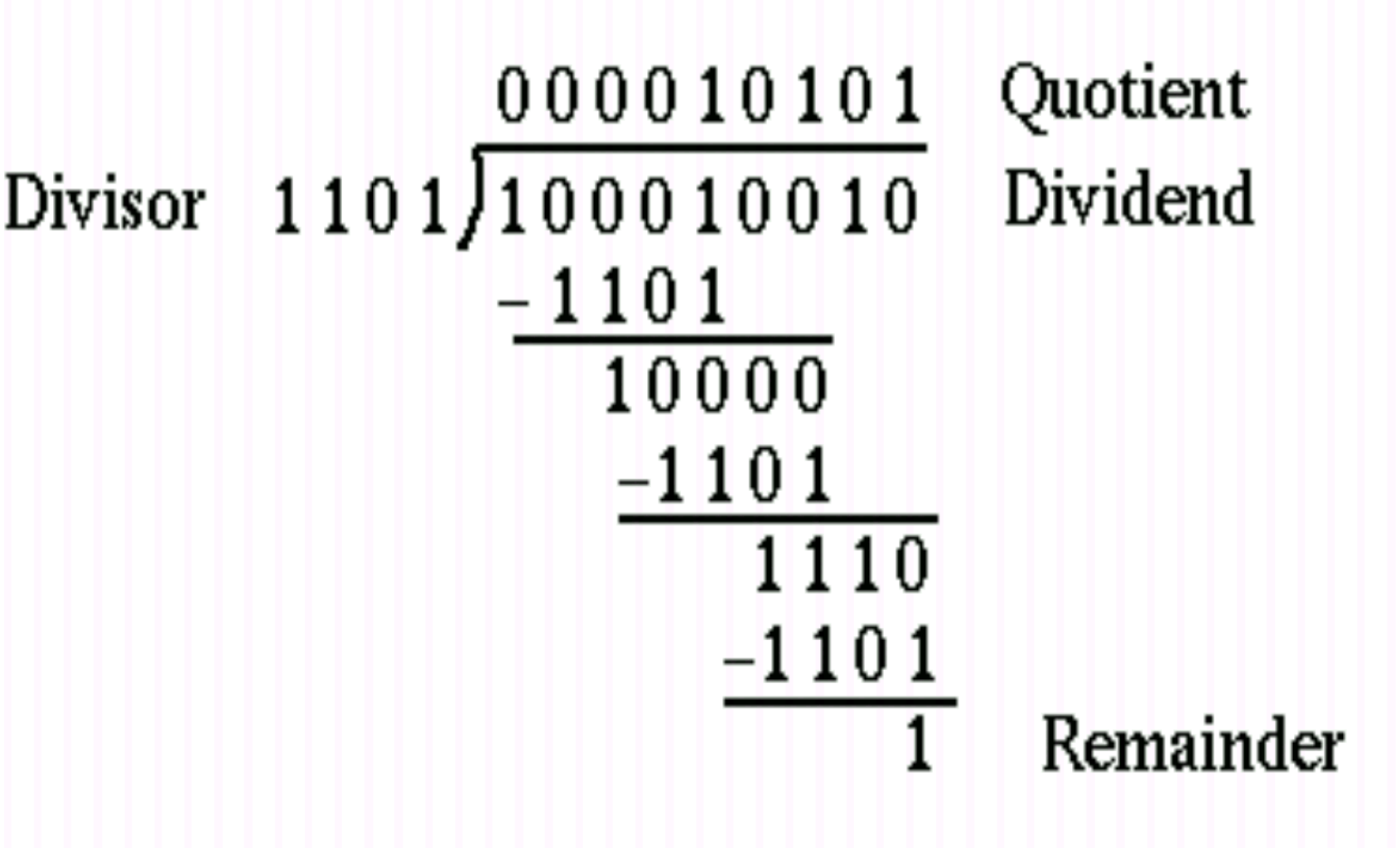
Example: Subtract binary number A= $(1010)_2$ and B= $(1011)_2$.



1.4 Binary arithmetic

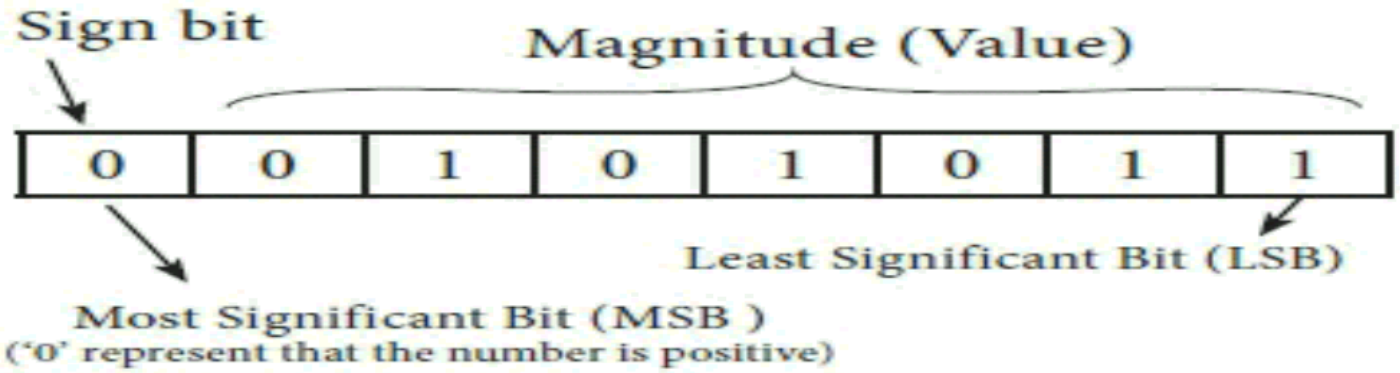
§ **Binary Division:** It is exactly same as decimal division.

Example: Divide binary number A= $(100010010)_2$ to B= $(1101)_2$.



1.5 Sign Binary Number

§ **8 bit sign Binary Number:** MSB of binary number is used to represent the sign and remaining bits are used for magnitude.



§ **Range of n-bit sign binary number:**

$$-2^{(n-1)} + 1 \quad \text{to} \quad 2^{(n-1)} - 1$$

1.6 Complements

§ Type of Complements

1. Radix complements (r's)
2. Diminished radix complement (r-1)

§ Diminished radix complement (r-1):

Given a number N in base r having n digits, the (r–1)'s complement of N is defined as: $(r^n - 1) - N$

Example:1

1. 7-digit binary numbers 1's complement is: $(2^7 - 1) - N = 1111111 - N$
2. 1's complement of 1011000 is: 1111111–1011000 = 0100111

Example:2

3. 6-digit decimal numbers 9's complement is: $(10^6 - 1) - N = 999999 - N$
4. 9's complement of 546700 is 999999–546700 = 453299

§ 1's Complement: All '0's become '1's and All '1's become '0's.

Example:3

5. 1's Complement of $(10110000)_2$ is $(01001111)_2$

r's Complement

Ex:- Comp of $(7)_{10}$

Ans = 3

10 - 7 = 3

10 - 9 = 1

10 - 6 = 4

N = 7

n = 1

r = 10

$r^n - N$ r's complement

1) N = 5690

determine 10's comp

r = 10

N = 5690

n = 4

$= 10^4 - 5690$

(r-1)'s Complement

"r's comp. ← "(r-1)'s comp

r = 10	10's comp	9's comp
r = 2	2's	1's
r = 8	8's	7's
r = 16	16's	F's

r's comp : $r^n - N$

(r-1)'s comp : $r^n - N - 1$

(r-1)'s comp : r's comp - 1

(r-1)'s comp + 1 : r's comp

1's & 2's Complement

r-1

1. Obtain 1's complement of $(1010)_2$

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

Ans

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

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

if r = 10
9999

if r = 8
7777

if r = 16
ffff

if r = 2
1111

2. Obtain 2's Complement of $(10111010)_2$

2) N = 1101 - find 2's comp

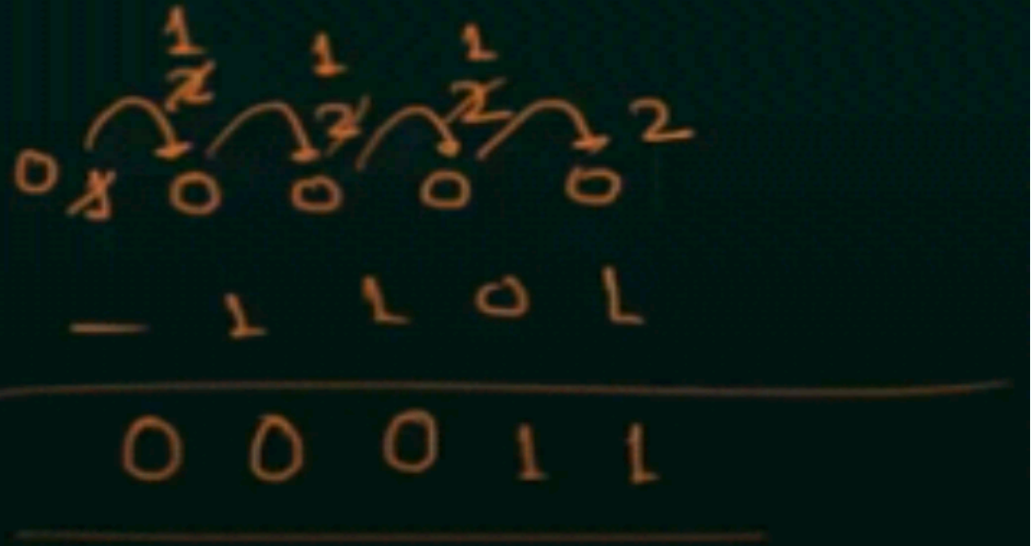
r = 2 n = 4

N = 1101 $\rightarrow (13)_{10}$

$= 2^4 - 1101$

$= (16)_{10} - 1101$

$= 10000 - 1101$



Radix Complement (r's complement):

§ The r's complement of an n-digit number N in base r is defined as:

$(r^n - N)$ for $N \neq 0$ and as 0 for $N = 0$

§ The r's complement is obtained by adding 1 to the (r - 1) 's complement,

Example:1 Base: 2

The 2's complement of 1101100 is 0010100.

Example:2 Base: 10

The 10's complement of 012398 is 987602.

§ 2's Complement (Radix Complement):

1.Take 1's complement then add 1.

2.Toggle all bits to the left of the first '1' from the right.

Example:3

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

1.7 Subtraction using Complements

The subtraction of two n-digit unsigned numbers M – N in base r can be done as follows:

1. Add the minuend M to the r's complement of the subtrahend N. Mathematically, $M + (r^n - N) = M - N + r^n$.
2. If $M \geq N$, the sum will produce and end carry r^n, which can be discarded; what is left is the result $M - N$.
3. If $M < N$, the sum does not produce an end carry and is equal to $r^n - (N - M)$, which is the r's complement of $(N - M)$. To obtain the answer in a familiar form, take the r's complement of the sum and place a negative sign in front.

Example:1

Using 10's complement, subtract 72532 – 3250.

	$M =$	72532
10's complement of	$N =$	+96750
	Sum =	169282
Discard end carry 10^5 =		−100000
	Answer =	69282

Example:2

Using 10's complement, subtract 3250 – 72532.

	M =	03250
10's complement of	N =	+27468
	Sum =	30718

→ There is no end carry.

→ Therefore, the answer is – (10's complement of 30718) = – 69282.

Example:3

Given the two binary numbers $X = 1010100$ and $Y = 1000011$, perform the subtraction (a) $X - Y$; and (b) $Y - X$, by using 2's complement.

(a)		
	X =	1010100
	2's complement of Y =	+0111101
	Sum =	10010001
	Discard end carry 2^7 =	−10000000
	Answer. X − Y =	0010001

(b)		
	Y =	1000011
	2's complement of X =	+ 0101100
	Sum =	1101111

There is no end carry. Therefore, the answer is Y − X = − (2's complement of 1101111) = − 0010001.

Subtraction of unsigned numbers can also be done by means of the (r - 1)'s complement. Remember that the (r - 1) 's complement is one less then the r's

(

but this time using 1's complement.

(a) $X - Y = 1010100 - 1000011$

$X =$	1010100
1's complement of $Y =$	± 0111100
Sum =	10010000
End-around carry =	+ 1
Answer. $X - Y =$	0010001

(b) $Y - X = 1000011 - 1010100$

$Y =$	1000011
1's complement of $X =$	+0101011
Sum =	1101110

There is no end carry, Therefore, the answer is Y – X = – (1's complement of 1101110) = – 0010001.

1.8 Binary codes

BCD code (Binary coded Decimal):

- A number with k decimal digits will require 4k bits in BCD.
- Decimal 396 is represented in BCD with 12bits as 0011 1001 0110, with each group of 4 bits representing one decimal digit.
- A decimal number in BCD is the same as its equivalent binary number only when the number is between 0 and 9.
- The binary combinations 1010 through 1111 are not used and have no meaning in BCD.

Binary-Coded Decimal (BCD)

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

Example:1

Consider decimal 185 and its corresponding value in BCD and binary:

$$(185)_{10} = (0001\ 1000\ 0101)_{BCD} = (10111001)_2$$

BCD Addition: Add 6 to convert invalid BCD to valid BCD.

4	0100	4	0100	8	1000
+5	+0101	+8	+1000	+9	+1001
9	1001	12	1100	17	10001
			+0110		+0110
			10010		10111

Example:2

Consider the addition of 184 + 576 = 760 in BCD:

BCD	1	1		
	0001	1000	0100	184
	+0101	0111	0110	+576
Binary sum	0111	10000	1010	
Add 6	___	0110	0110	___
BCD sum	0111	0110	0000	760

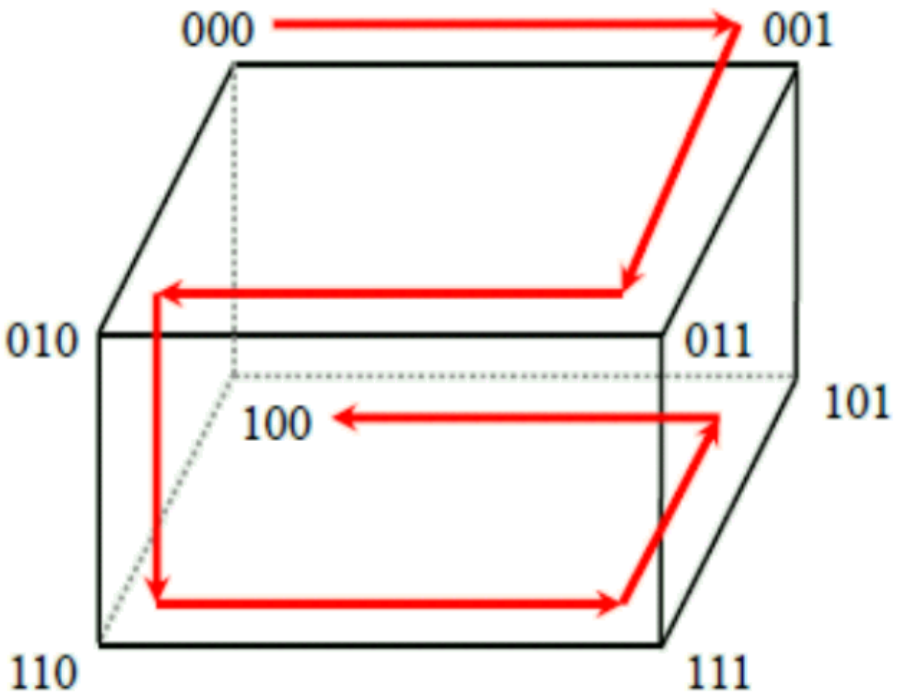
Decimal Codes:

Four Different Binary Codes for the Decimal Digits

Decimal Digit	BCD 8421	2421	Excess-3	8, 4, −2, −1
0	0000	0000	0011	0000
1	0001	0001	0100	0111
2	0010	0010	0101	0110
3	0011	0011	0110	0101
4	0100	0100	0111	0100
5	0101	1011	1000	1011
6	0110	1100	1001	1010
7	0111	1101	1010	1001
8	1000	1110	1011	1000
9	1001	1111	1100	1111
	1010	0101	0000	0001
Unused	1011	0110	0001	0010
bit	1100	0111	0010	0011
combi-	1101	1000	1101	1100
nations	1110	1001	1110	1101
	1111	1010	1111	1110

Gray Code:

- The advantage is that only bit in the code group changes in going from one number to the next.



Gray Code

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

American Standard Code for Information Interchange (ASCII)Code:

- It uses 7-bits to represent, 94 Graphic printing characters and 34 Non-printing characters
- Some non-printing characters are used for text format (e.g. BS =Backspace, CR = carriage return).
- Other non-printing characters are used for record marking and flow control (e.g. STX and ETX start and end text areas).
- ASCII has some interesting properties:
  1. Digits 0 to 9 span Hexadecimal values 3016 to 3916.
  2. Upper case A-Z span 4116 tc 5A16.
  3. Lower case a-z span 6116 to 7A16.

American Standard Code for Information Interchange (ASCII)

	$b_7b_6b_5$							
$b_4b_3b_2b_1$	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	-	o	DEL

Error-Detecting Code:

- To detect errors in data communication and processing, an eighth bit is sometimes added to the ASCII character to indicate its parity.
- A parity bit is an extra bit included with a message to make the total number of 1's either even or odd.
- **Example:** Consider the following two characters and their even and odd

	With even parity	**With odd parity**
ASCII A = 1000001	01000001	11000001
ASCII T = 1010100	11010100	01010100

Error-Correcting Code:

- Redundancy (e.g. extra information), in the form of extra bits, can be incorporated into binary code words to detect and correct errors.
- A simple form of redundancy is parity, an extra bit appended onto the code word to make the number of 1's odd or even. Parity can detect all single bit errors and some multiple-bit errors.
- A code word has even parity if the number of 1's in the code word is even.

§ A code word has odd parity if the number of 1's in the code word is odd.

Example:

Message A: 100010011 (even parity)

Message B: 100010010 (odd parity)

1.9 Binary logic

§ Binary logic consists of binary variables and a set of logical operations.

§ The variables are designated by letters of the alphabet, such as *A*, *B*, *C*, *x*, *y*, *z*, etc, with each variable having two and only two distinct possible values: 1 and 0.

§ Three basic logical operations: AND, OR, and NOT.

1. AND: This operation is represented by a dot or by the absence of an operator. For example, $x \cdot y = z$ or $xy = z$ is read "x AND y is equal to z," The logical operation AND is interpreted to mean that $z = 1$ if only $x = 1$ and $y = 1$; otherwise $z = 0$. (Remember that x, y, and z are binary variables and can be equal either to 1 or 0, and nothing else.)
2. OR: This operation is represented by a plus sign. For example, $x + y = z$ is read "x OR y is equal to z," meaning that $z = 1$ if $x = 1$ or $y = 1$ or if both $x = 1$ and $y = 1$. If both $x = 0$ and $y = 0$, then $z = 0$.
3. NOT: This operation is represented by a prime (sometimes by an overbar). For example, x' = z (or $\bar{x} = z$) is read "not x is equal to z," meaning that z is what z is not. In other words, if $x = 1$, then $z = 0$, but if $x = 0$, then $z = 1$, The NOT operation is also referred to as the complement operation, since it changes a 1 to 0 and a 0 to 1.

AND

x	y	z
0	0	0
0	1	0
1	0	0
1	1	1

$z = x \cdot y = x\,y$

OR

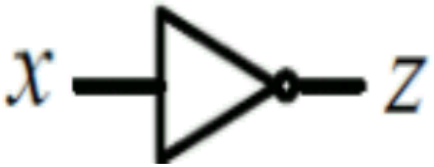
x	y	z
0	0	0
0	1	1
1	0	1
1	1	1

$z = x + y$

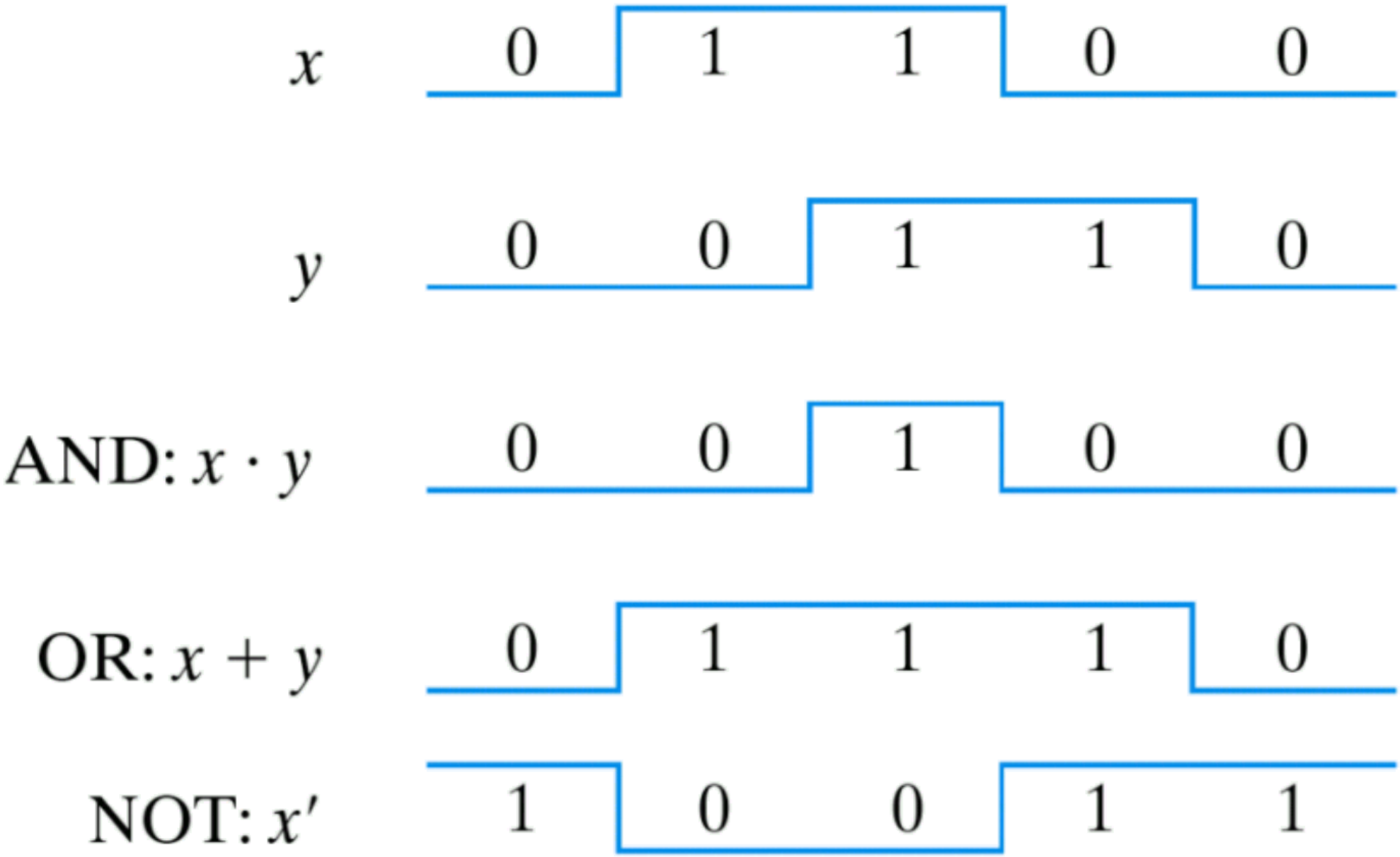
NOT

x	z
0	1
1	0

$z = \bar{x} = x'$



Input-Output signals for gates:



NAND

2 Input NAND gate		
A	B	$\overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0

NOR

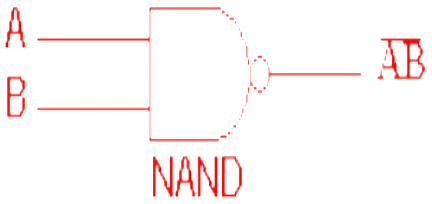
2 Input NOR gate		
A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

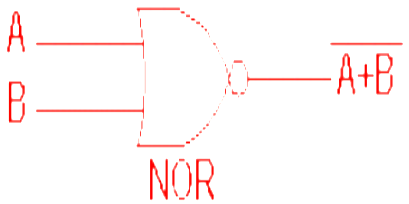
EXOR

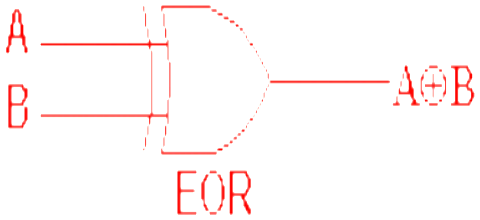
2 Input EXOR gate		
A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

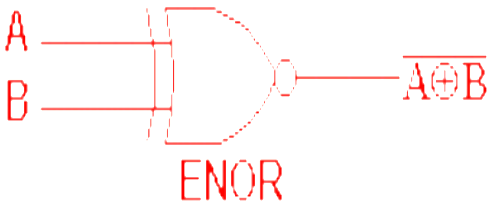
EXNOR

2 Input EXNOR gate		
A	B	$\overline{A \oplus B}$
0	0	1
0	1	0
1	0	0
1	1	1

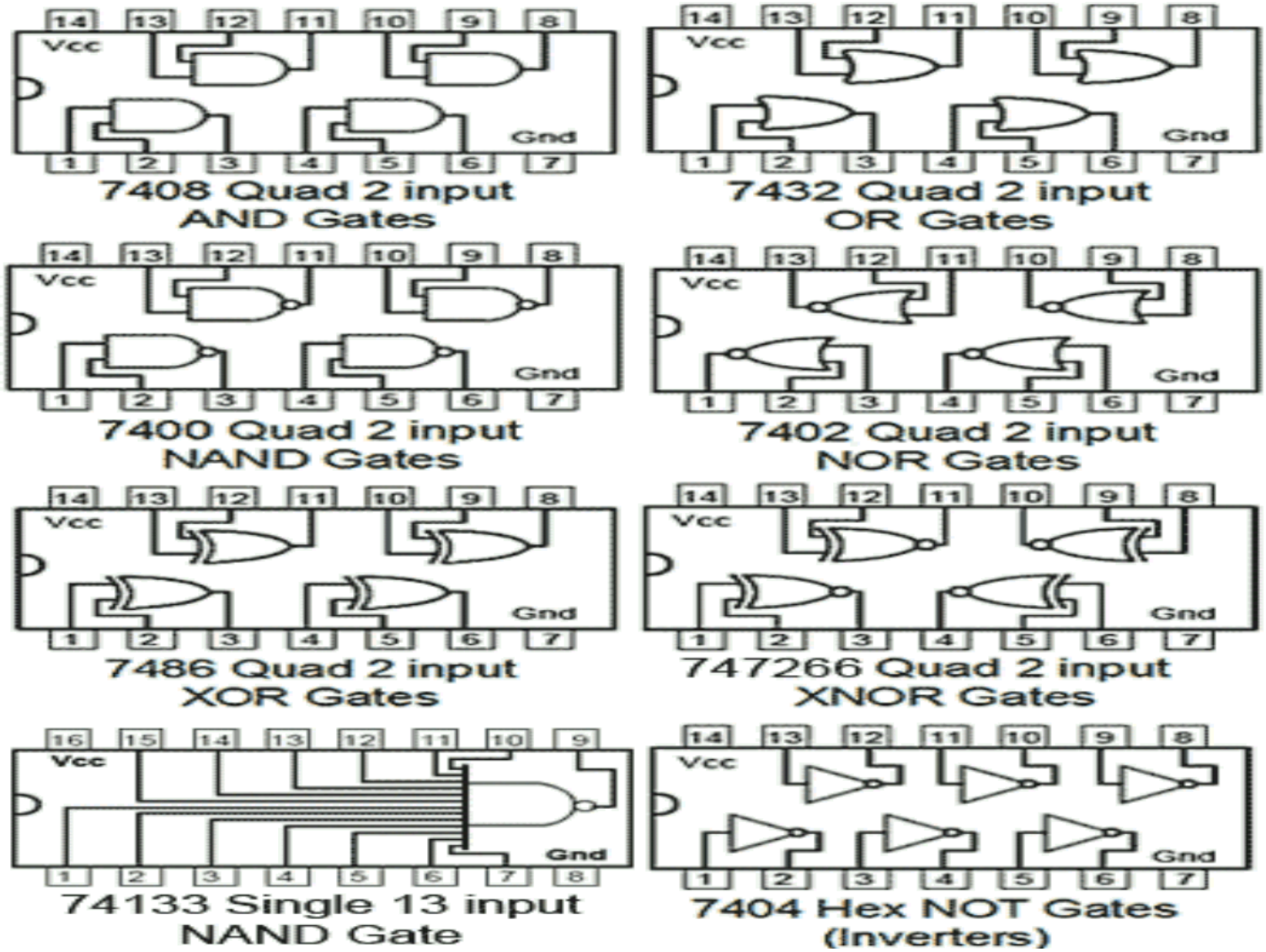








1.10 Logic gates IC:

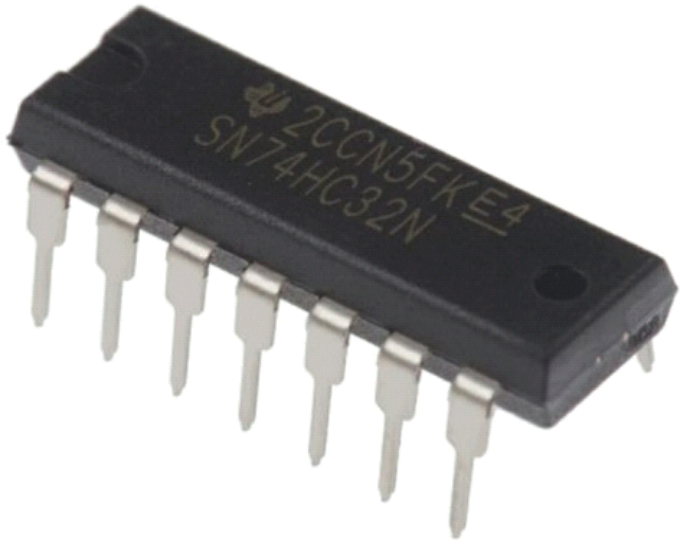


Two Input logic gate IC 74xx series

1.11 Digital Logic families

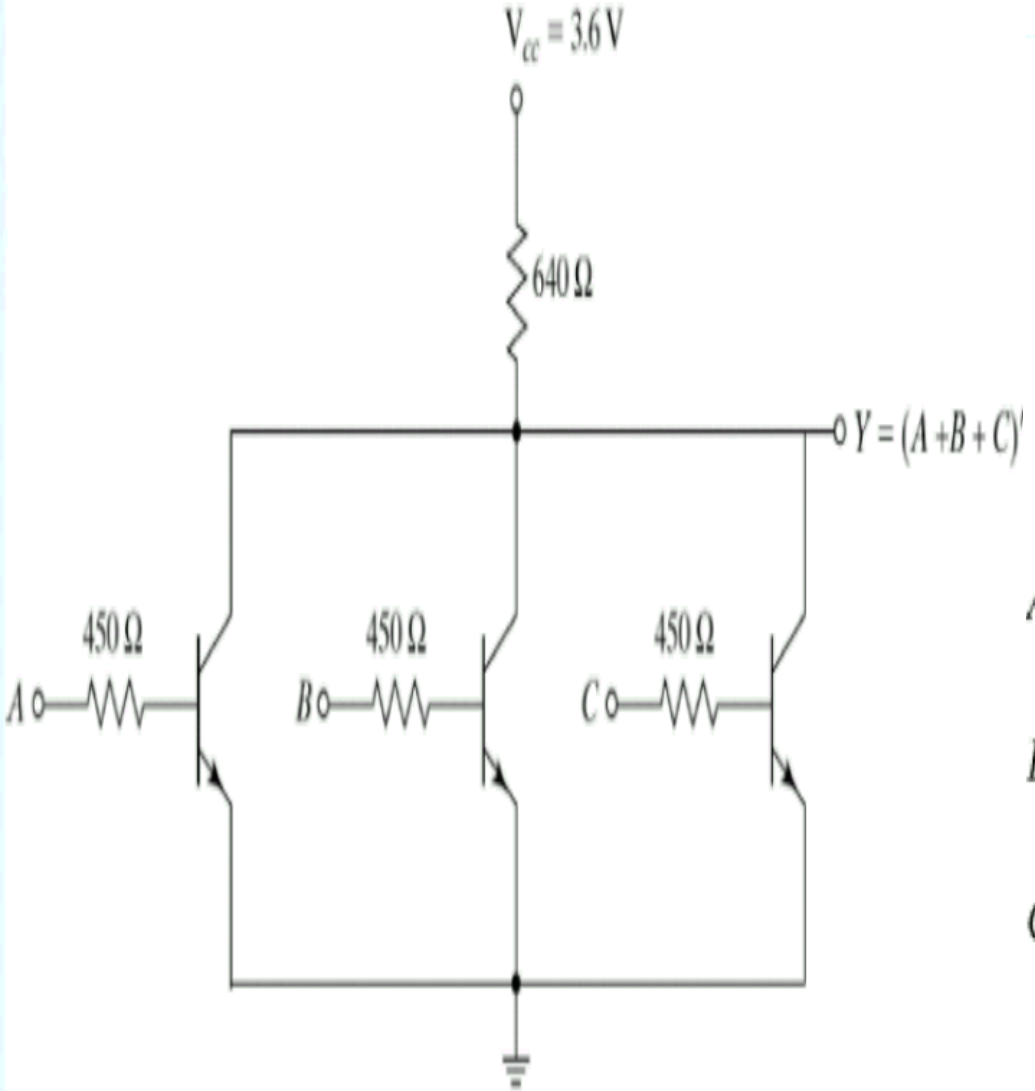
§ **Type of Digital Logic Families:**

1. RTL (Resistor-transistor logic)
2. DTL (Diode-transistor logic)
3. TTL (Transistor -transistor logic)
4. ECL (Emitter-coupled logic)
5. MOS (Metal-oxide semiconductor)
6. CMOS (Complementary Metal-oxide semicon
7. **Characteristic of Logic Families:**

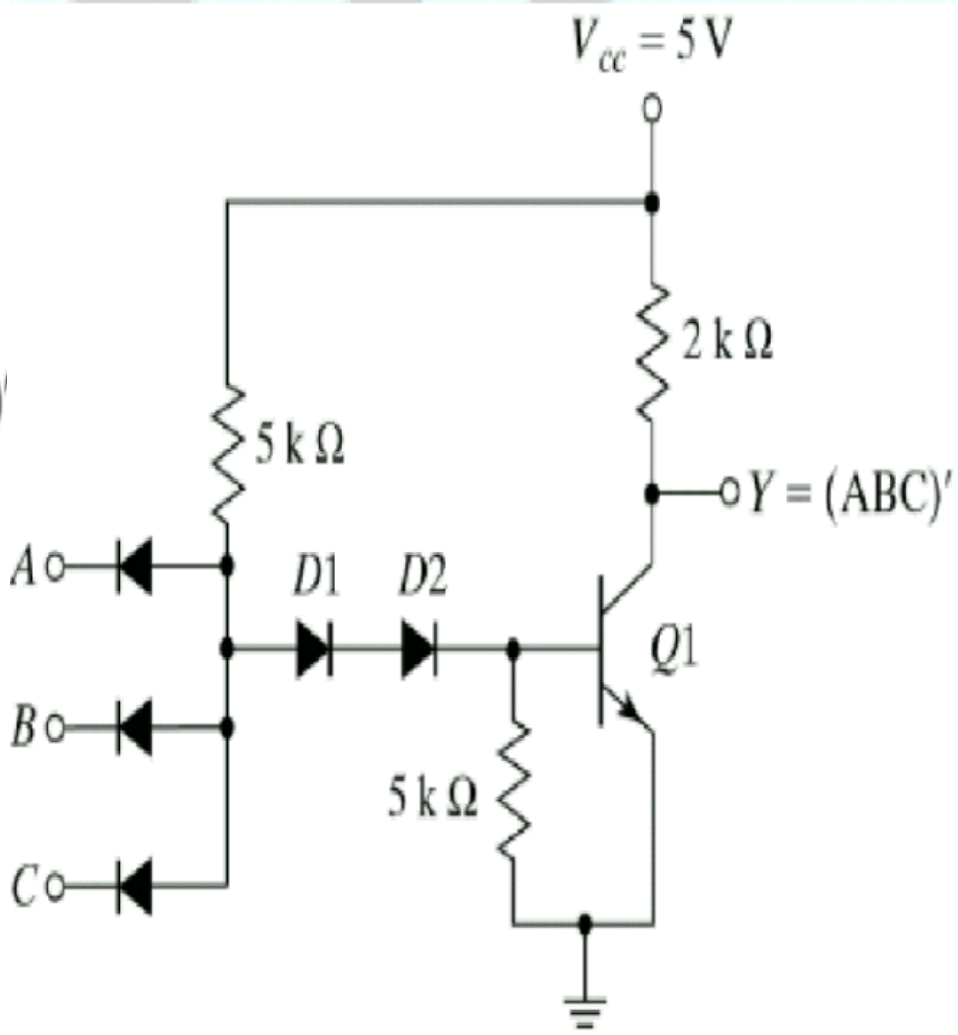


1. Logic Levels
2. Noise margin
3. Propagation delay
4. Fan-out
5. Power Dissipation
6. Speed-Power Product

RTL-NOR

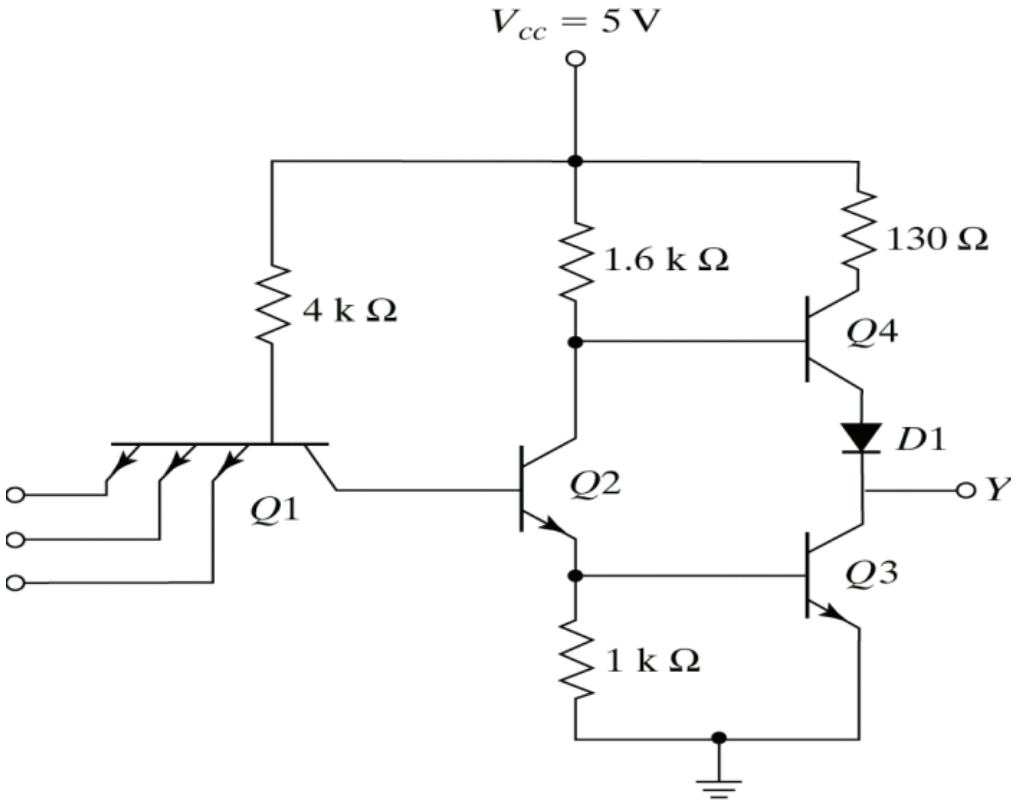


DTL-NAND

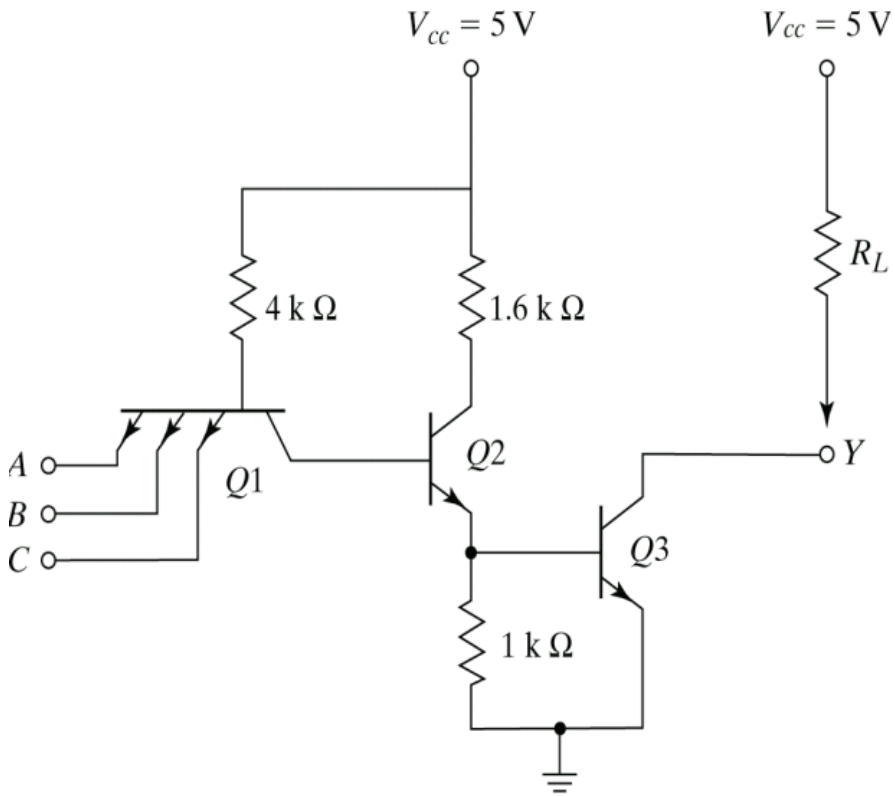


Transistor-Transistor Logic (TTL):

§ Three different types of output configurations
  1. Totem-pole output
  2. Open-collector output
  3. Three-state (or tri state) output

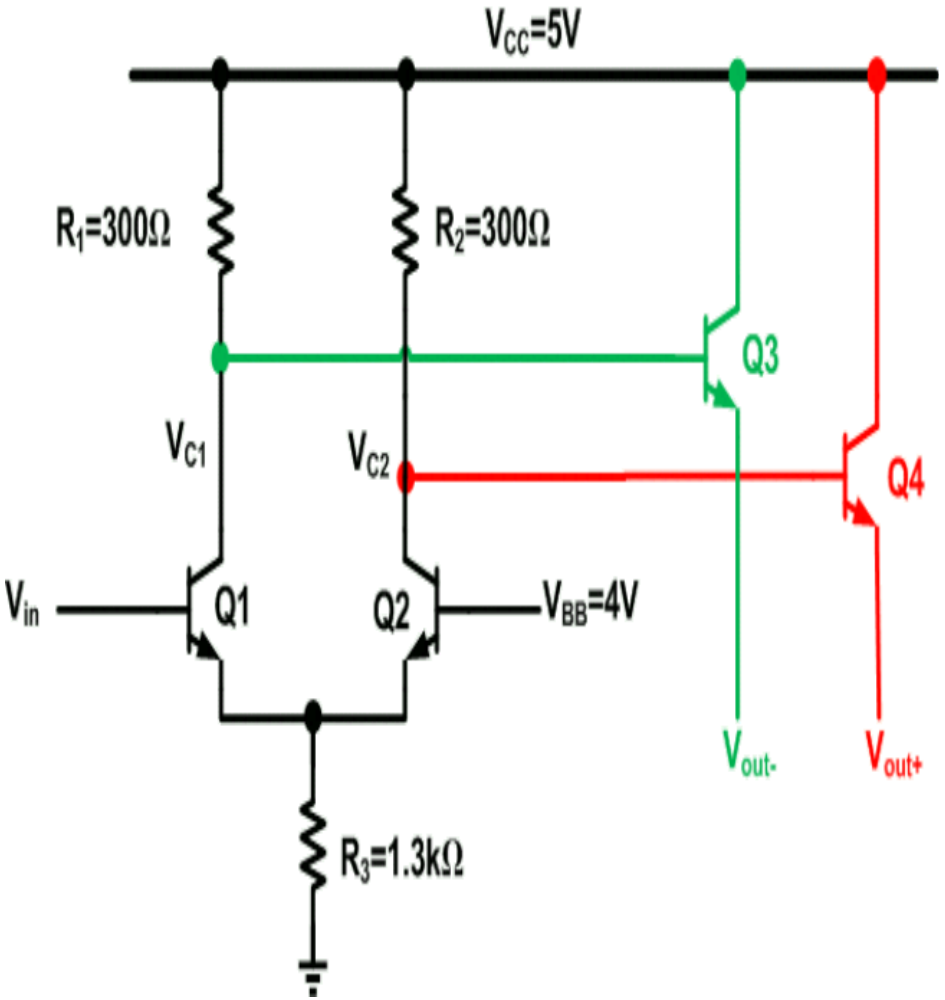


TTL NAND Gate With Totem-pole Output NAND Gate



Open collector TTL

EMITTER-COUPLED LOGIC (ECL)



CMOS