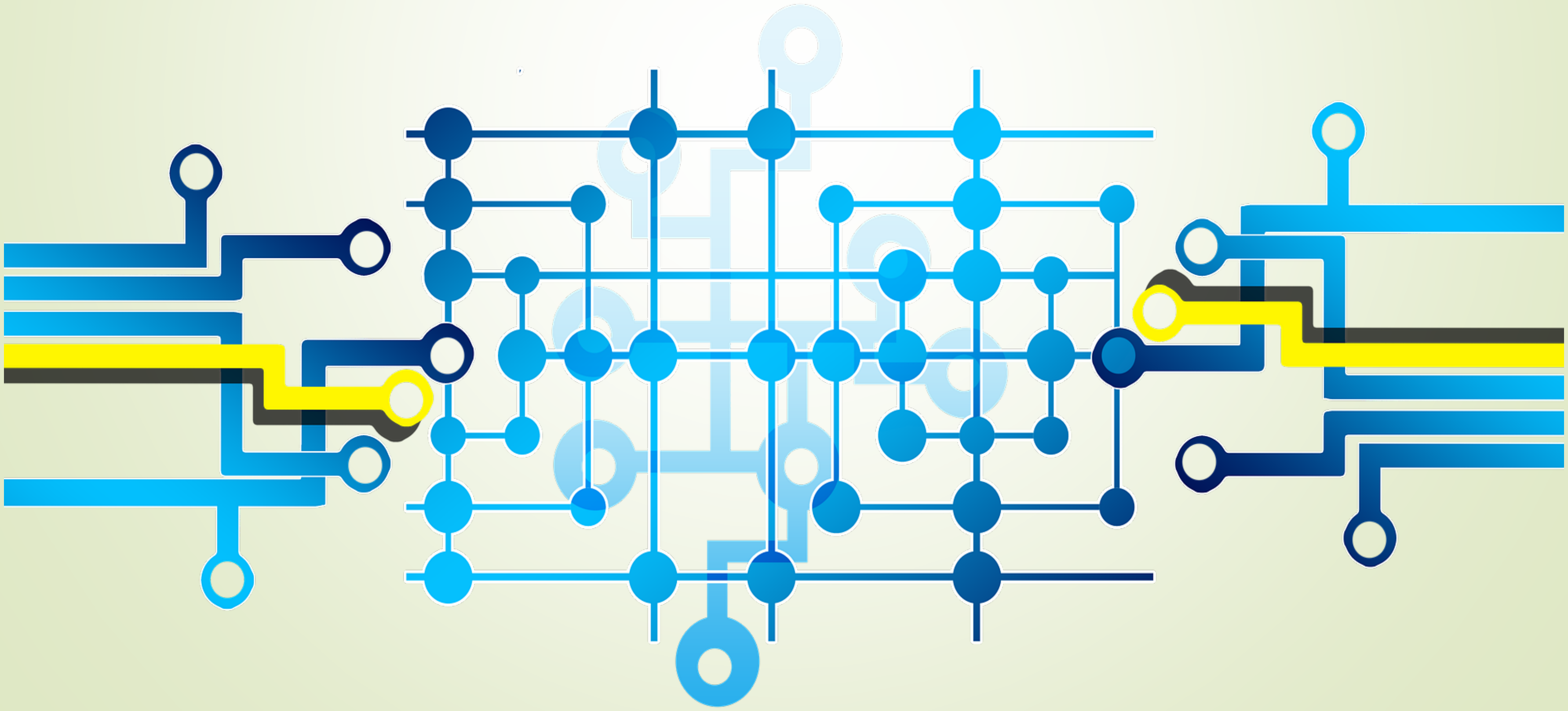
Digital Electronics (203105201)

Saurabh Srivastava, Assistant Professor
Mechatronics Engineering



CHAPTER-1

Fundamentals of Digital Systems and Logic Families

Digital signals, digital circuits, **Number Systems::**binary, signed binary,
octal hexadecimal number, binary arithmetic, one's and two's complements
arithmetic, codes, BCD arithmetic, error detecting and correcting codes,
AND, OR, NOT, NAND, NOR, and Exclusive-OR operations, examples of IC gates,
characteristics of digital ICs,
Digital Logic families:: TTL and CMOS logic, interfacing CMOS and TTL

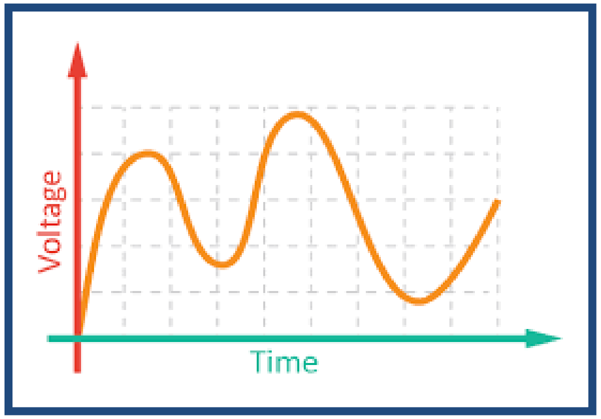
Fundamentals of Digital Systems and Logic Families

Signal: A physical quantity, which contain some information and which is function of one or more variables.

Type of Signal:

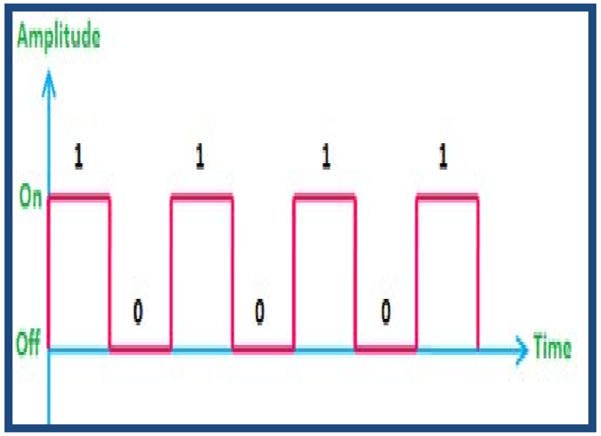
1. Analog Signals
2. Digital Signals

Is there anything between Analog and Digital Signal?



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.



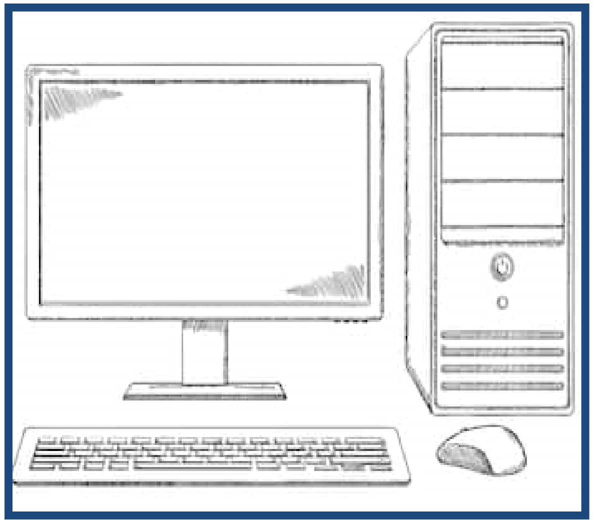
Digital Signals: Signal which has only a finite number of distinct values.

Examples: Signals obtained directly from computers, Output of A to D converter.

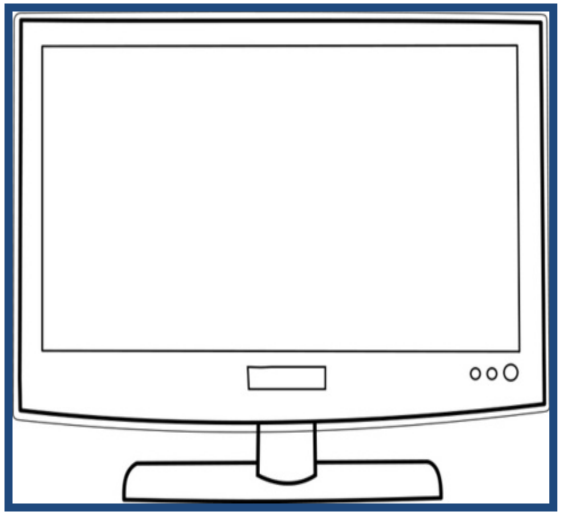
Image source : Google

Fundamentals of Digital Systems and Logic Families

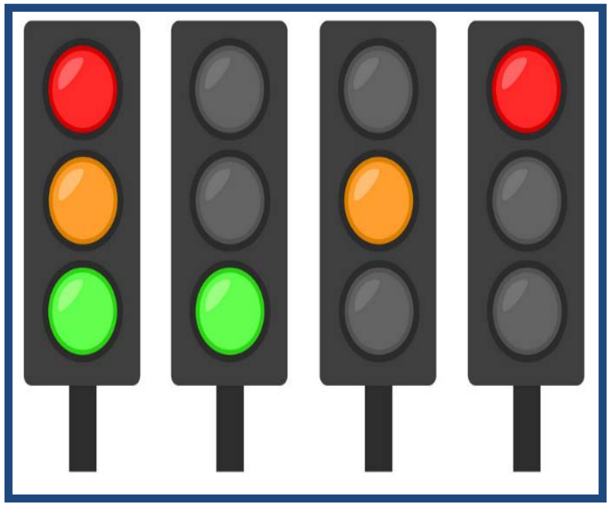
Digital circuits have input signal and output signal both are digitals.



Digital Computer



Digital Television



Traffic Control System

Image source : Google

Fundamentals of Digital Systems and Logic Families

Sets of values used to represent quantity like number of students attending classes, Grades archived by students in tests, etc.

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

Fundamentals of Digital Systems and Logic Families

Decimal Number System: 12345

- **Radix, r = 10**
- **Digits : 0,1,2,3,4,5,6,7,8,9 (10 digits)**
- **12345.678 = $1 \times 10^4 + 2 \times 10^3 + 3 \times 10^2 + 4 \times 10^1 + 5 \times 10^0 + 6 \times 10^{-1} + 7 \times 10^{-2} + 8 \times 10^{-3}$**

Binary Number System: 10011.11

- **Radix, r = 2**
- **Digits : 0,1 (2 digits)**
- **10011.11= $1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2}$ = 16+2+1+0.5+0.25 = $(19.75)_{10}$**

Fundamentals of Digital Systems and Logic Families

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.

Fundamentals of Digital Systems and Logic Families

Conversion from Decimal to other radix:

Separate integer and fractional parts.

For Integer part:

1. Divide the integer by the base repeatedly until there is nothing to divide.
2. Keeping track of remainders from each step.
3. List the remainder values in reverse order to find the equivalent.

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.
4. Repeat step-2 and 3 up to end. Last carry will represent the LSD.

Combine result of Integer part and Fractional part.

Fundamentals of Digital Systems and Logic Families

Conversion from Decimal to other radix: $(19.75)_{10}$ to binary

Separate integer and fractional parts: 19 & 0.75

For Integer part:

1. Divide the integer by the base (r=2) repeatedly until there is nothing divide
2. Keeping track of remainders from each step.
3. List the remainder values in reverse order to find the equivalent.

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.
4. Repeat step-2 and 3 up to end. Last carry will represent the LSD.

Combine result of Integer part and Fractional part.

19=2x9 +1	1
9=2x4 +1	1
4=2x2 +0	0
2=2x1 +0	0
1=2x0+ 1	1

0.75x2=1.5	0.5x2= 1
1 (MSD)	1 (LSD)

$$(19.75)_{10} = (10011.11)_2$$

Fundamentals of Digital Systems and Logic Families

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.

Conversion from Binary to Hex:

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

Conversion from Octal to Binary:

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

Conversion from Hex to Binary:

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

Fundamentals of Digital Systems and Logic Families

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.

$$(19.75)_{10} = (10011.11)_2 = \left(\underbrace{010}\,\underbrace{011}\,.\,\underbrace{110}\right)_2 = (23.6)_8$$

$$(10100.101)_2 = (24.5)_8$$

Fundamentals of Digital Systems and Logic Families

Conversion from Octal to Hex:

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

$$(\mathbf{15.3})_{\mathbf{8}} = (1101.011)_{\mathbf{2}} = (\mathrm{D}.6)_{\mathbf{16}}$$

Fundamentals of Digital Systems and Logic Families

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.

$$(\mathbf{3D.6})_{\mathbf{16}} = (0011\ 1101.0110)_{\mathbf{2}} = \left(\underbrace{111}\ \ \underbrace{101}\ .\ \underbrace{011}\right)_2 = \left(\underbrace{111}_{7}\ \underbrace{101}_{5}\ \underbrace{.011}_{.3}\right)_2 = (75.3)_8$$

Fundamentals of Digital Systems and Logic Families

Question [GATE (CS) 2014 SET-2]

Consider the equation *$(43)_x = (y3)_8$* , where x and y are unknowns. The number of possible solutions is _____

Sol:

$$(43)_x = 4 \times x^1 + 3 \times x^0 = 4 \times x + 3,$$

$$(y3)_8 = y \times 8^1 + 3 \times 8^0 = 8 \times y + 3,$$

$$\Rightarrow 4x = 8y,$$

$$\Rightarrow x = 2y,$$

Also, $4 < x, and, \ y < 8$

y	x
0	0
1	2
2	4
3	6
4	8
5	10
6	12
7	14
8	16

Fundamentals of Digital Systems and Logic Families

Question [GATE (CS) 2017 SET-2]

The representation of the value of a 16-bit unsigned integer X in a hexadecimal number system is BCA9. The representation of the value of X in octal number system is:

Sol: $(\boldsymbol{BCA9})_{16} = \left(\underbrace{\mathbf{1011}}_{B}\ \underbrace{\mathbf{1100}}_{C}\ \underbrace{\mathbf{1010}}_{A}\ \underbrace{\mathbf{1001}}_{9}\right)_2 = \left(\underbrace{\mathbf{001}}_{1}\ \underbrace{\mathbf{011}}_{3}\ \underbrace{\mathbf{110}}_{6}\ \underbrace{\mathbf{010}}_{2}\ \underbrace{\mathbf{101}}_{5}\ \underbrace{\mathbf{001}}_{1}\right)_2 = (\mathbf{136251})_8$

Question [GATE (CS) 2023]

A Particular number is written as 132 in radix-4 representation. The same number in radix-5 representation is:

Sol:

$$(132)_4 = 1 \times 4^2 + 3 \times 4^1 + 2 \times 4^0 = (16 + 12 + 2)_{10} = (30)_{10}$$

$$(30)_{10} = a \times 5^2 + b \times 5^1 + c \times 5^0 = 25a + 5b + c$$

Now, $(30)_{10} = 25 + 5 + 0 \equiv 25a + 5b + c \implies a = 1, b = 1, c = 0$

$$(132)_4 = (30)_{10} = \mathbf{1} \times 5^2 + \mathbf{1} \times 5^1 + \mathbf{0} \times 5^0 = (110)_5$$

Fundamentals of Digital Systems and Logic Families

Question [GATE (CS) 2021]
If x and y are two decimal digits and $(0.1101)_2 = (0.8xy5)_{10}$, the decimal value of $x + y$ is:
Ans: $x + y = 3$

Question [GATE (CS) 2021]
Let the representation of a number in base 3 be 210. What is the hexadecimal representation of the number?
Ans: $(15)_{16}$

Question [ISRO 2018]
Given $\sqrt{224_r} = 13_r$, the value of radix-r is:
Ans: $r = 5.$

Question [ISRO 2017]
1217_8 is equivalent to (in hexadecimal):
Ans: $(28F)_{16}$

Question [ISRO 2016]
If $(12A7C)_{16} = X_8$, then the value of X is
Ans: $X = 225174$

Question [GATE 2014]
The base or radix of the number system such that the following equation holds is ______

$$\frac{312}{20} = 13.1$$

Ans: $r = 5.$

Question [ISRO 2014]
The binary equivalent of decimal number 42.75 is:

Ans: $(101010.11)_2$

Fundamentals of Digital Systems and Logic Families

Binary arithmetic is an essential part of all digital computers and many other digital systems.

Binary Addition: Binary number are added, its creates sum(S) and carry(C).

0110 + 0011 = 01001

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

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

0110 - 0011 = 00011

A-B	DIFFEREN CE	BORROW
0-0	0	0
0-1	1	1
1-0	1	0
1-1	0	0

Fundamentals of Digital Systems and Logic Families

Binary Multiplication: It is exactly same as decimal multiplication.

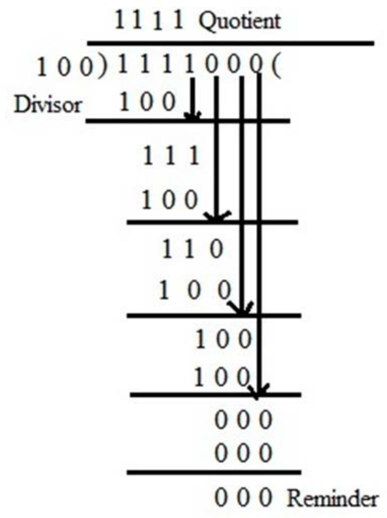
0110 x 0011 = 10010

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

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

Binary Division: It is exactly same as decimal division. It is called as the long division proced

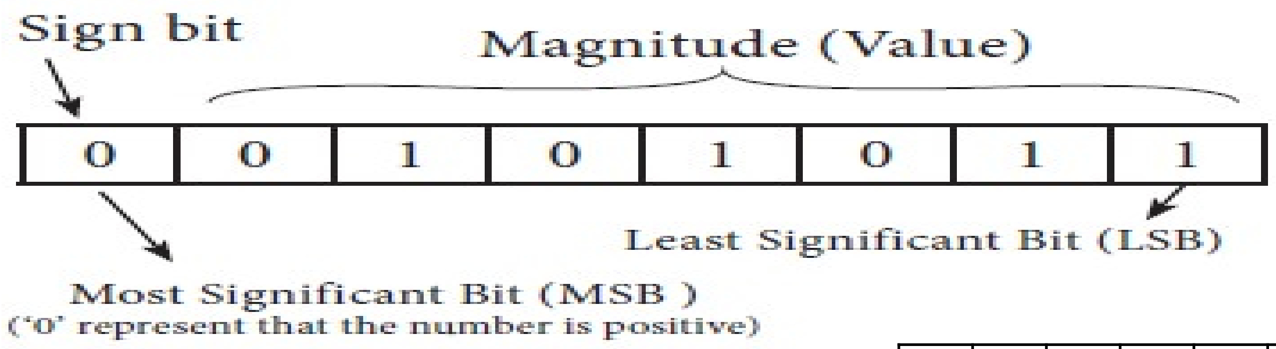
0110 ÷ 0011 = 10



Fundamentals of Digital Systems and Logic Families

Signed binary is very similar to binary, only that it includes negative numbers as well

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


Sign bit
Magnitude (Value)
0 0 1 0 1 0 1 1
Least Significant Bit (LSB)
Most Significant Bit (MSB)
('0' represent that the number is positive)

Range of n-bit sign binary number:

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

0	0	0	0	0	0	0
0	0	0	0	0	0	1
.	.	.	.	.	.	.
1	1	1	1	1	1	1

Image source : Google

Fundamentals of Digital Systems and Logic Families

Consider the unsigned 8-bit fixed point binary number representation, below,

$$b_7 b_6 b_5 b_4 b_3 \,.\, b_2\, b_1 b_0$$

where the position of the binary point is between b_3 and b_2 . Assume b_7 is the most significant bit. Some of the decimal numbers listed below cannot be represented exactly in the above representation:

i. 31.500
ii. 0.875
iii. 12.100
iv. 3.001

Which one of the following statements is true?

a) None of I, ii, iii, iv can be exactly represented
b) Only ii cannot be exactly represented
c) Only iii and iv cannot be exactly represented
d) Only i and ii cannot be exactly represented

Image source : Google

Fundamentals of Digital Systems and Logic Families

Complements are used in digital circuits because it is faster to subtract by adding complements than by performing true subtraction

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's complement of 1011000 is: $(2^7\text{-}1\text{-}N)$= 1111111–1011000 = 0100111
2. 9's complement of 546700 is: $(10^6\text{-}1\text{-}N)$=999999–546700 = 453299

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

Example:

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

Fundamentals of Digital Systems and Logic Families

Radix Complement (r’s complement):

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

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

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

Example:

1. The 2's complement of 1101100 is 0010100.
2. The 10's complement of 012398 is 987602.

2’s Complement (Radix Complement): Take 1’s complement then add 1 or Toggle all bits to the left of the first ‘1’ from the right.

Fundamentals of Digital Systems and Logic Families

Radix Complement (r’s complement):

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

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

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

Example:

1. The 2's complement of 1101100 is 0010100.
2. The 10's complement of 012398 is 987602.

2’s Complement (Radix Complement): Take 1’s complement then add 1 or Toggle all bits to the left of the first ‘1’ from the right.

Fundamentals of Digital Systems and Logic Families

The subtraction of two n-digit unsigned numbers, i.e., 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.

Fundamentals of Digital Systems and Logic Families

Consider two binary numbers:

M=10101 (21) and N=00111 (7)

We take 2's complement of subtrahend 00111, which is 11001.

Now, sum them. So, 10101 + 11001 = 1 01110

In the above result, we get the carry bit 1.

So we discard this carry bit and the remaining is the final result and a positive number.

M=00111 (7) and N=10101 (21)

We take 2's complement of subtrahend 10101, which is 01011.

Now, sum them. So, 00111 + 01011 = 10010

Now, 2's complement of sum 10010 = 01101+1 = 01110 (14)

1 (-ve sign) 1110 (magnitude, 14)

Fundamentals of Digital Systems and Logic Families

In the coding, when numbers or letters are represented by a specific group of symbols, it is said to be that number or letter is being encoded.

The group of symbols is called as **code.**

The digital data is represented, stored and transmitted as group of bits. This group of bits is also called as **binary code.**

Binary codes can be classified into two types.

- Weighted codes
- Unweighted codes

If the code has **positional weights**, then it is said to be **weighted code**. Otherwise, it is an unweighted code.

Weighted codes can be further classified as **positively weighted codes** and **negatively weighted codes.**

Fundamentals of Digital Systems and Logic Families

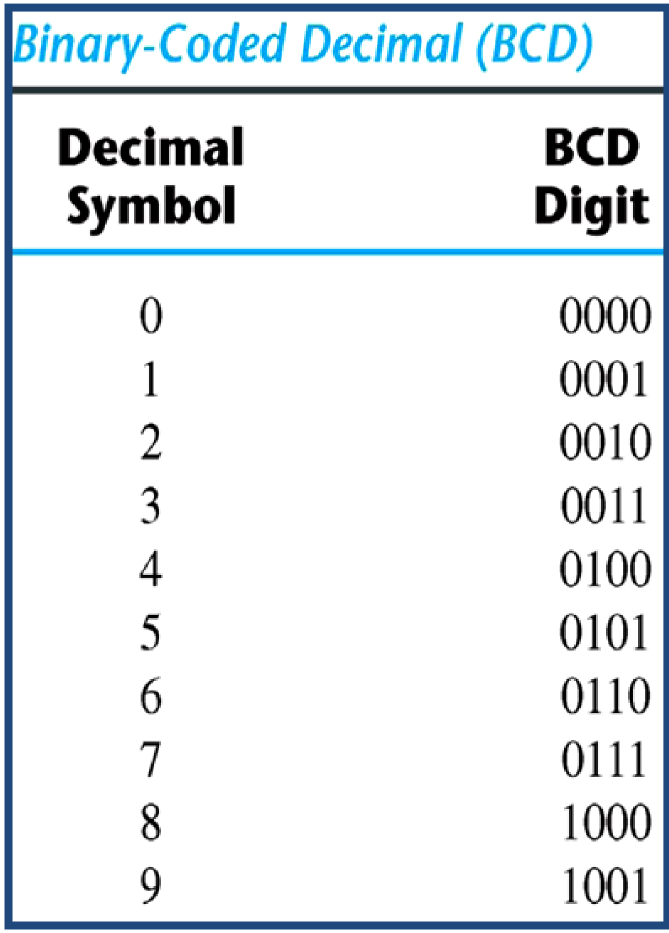
BCD code (Binary coded Decimal):

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.



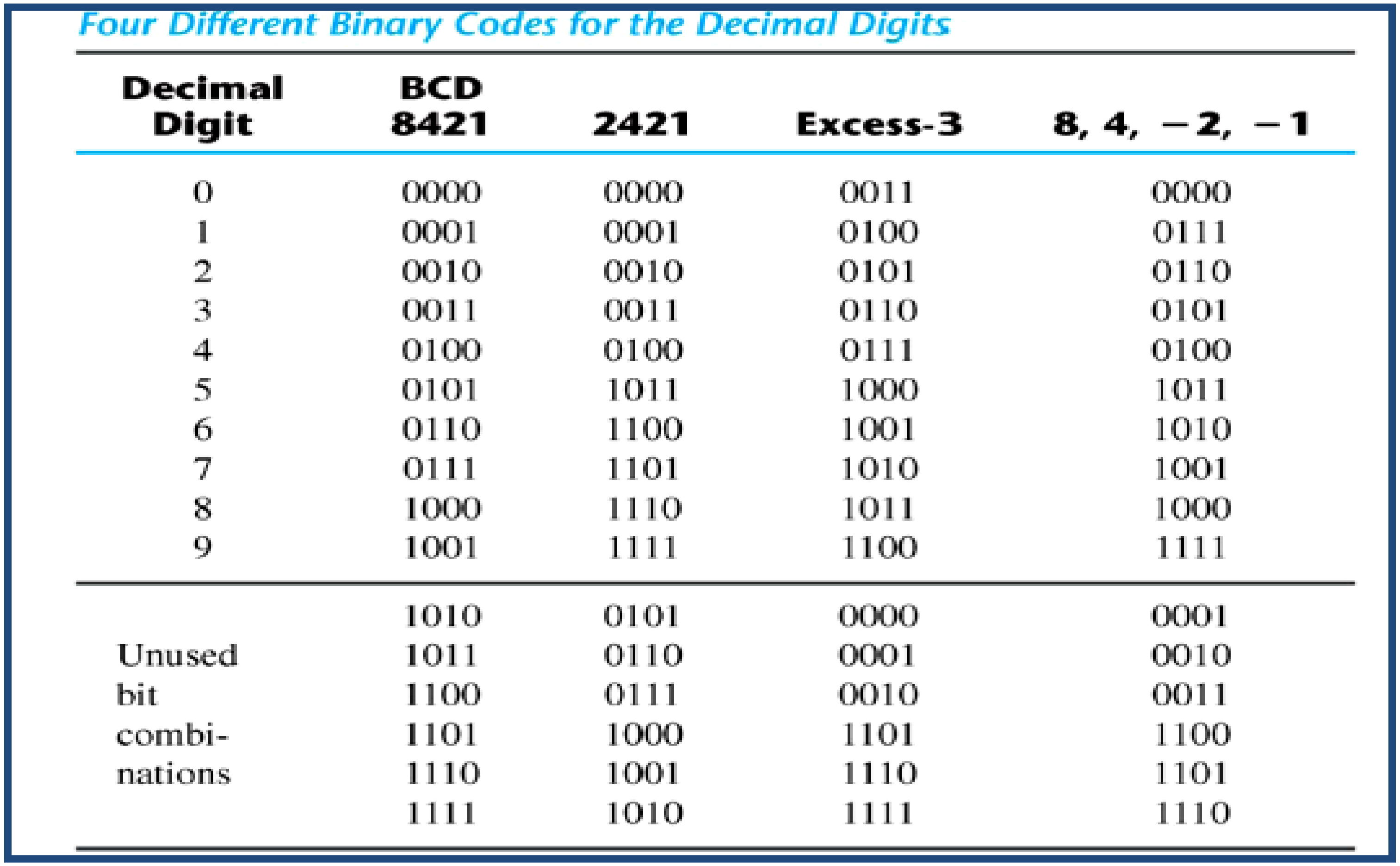
Binary-Coded Decimal (BCD)

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

Image source : Google

Fundamentals of Digital Systems and Logic Families

Decimal Codes:



Four Different Binary Codes for the Decimal Digits

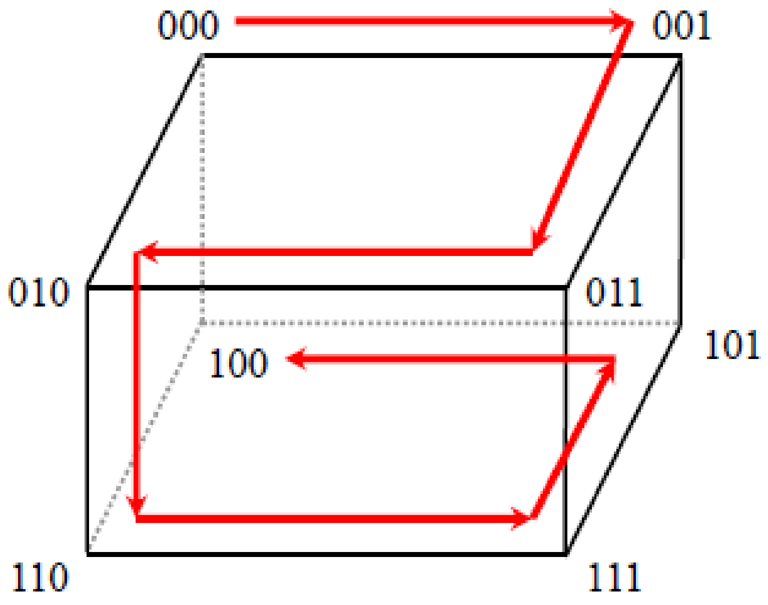
Decimal Digit	BCD 8421	2421	Excess-3	8, 4, −2, −1
0	0000	0000	0011	0000
1	0001	0001	0100	0111
2	0010	0010	0101	0110
3	0011	0011	0110	0101
4	0100	0100	0111	0100
5	0101	1011	1000	1011
6	0110	1100	1001	1010
7	0111	1101	1010	1001
8	1000	1110	1011	1000
9	1001	1111	1100	1111
	1010	0101	0000	0001
Unused	1011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

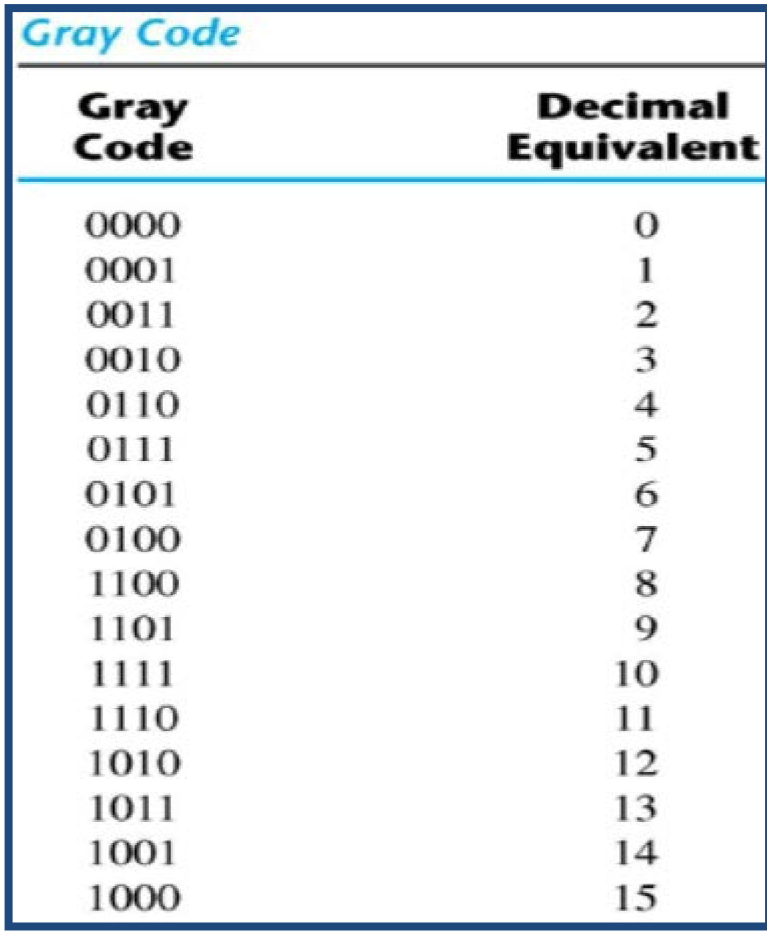
Image source : Google

Fundamentals of Digital Systems and Logic Families

Gray Code:

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





Gray Code

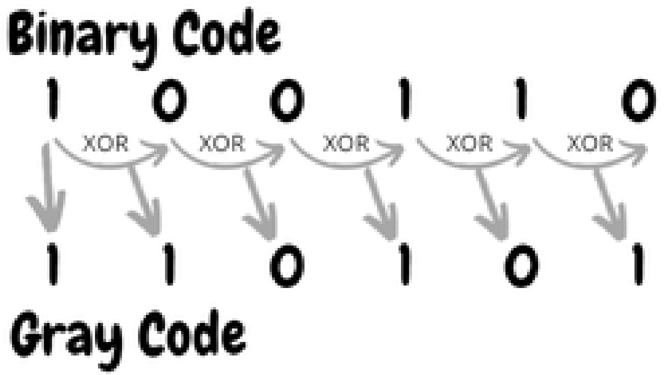
Gray Code	Decimal Equivalent
0000	0
0001	1
0011	2
0010	3
0110	4
0111	5
0101	6
0100	7
1100	8
1101	9
1111	10
1110	11
1010	12
1011	13
1001	14
1000	15

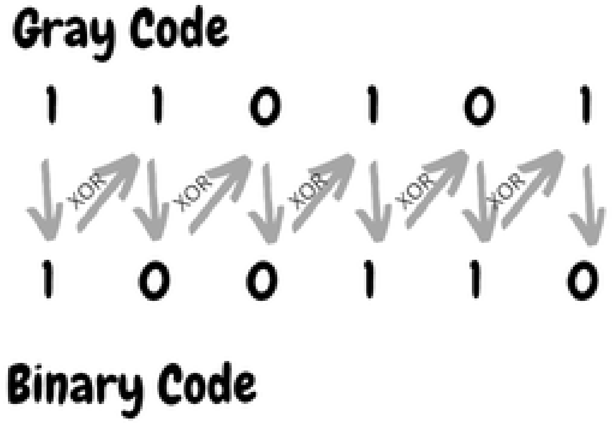
Image source : Google

Fundamentals of Digital Systems and Logic Families

Binary to gray conversion:

1. The Most Significant Bit (MSB) of the gray code is always equal to the MSB of the given binary code.
2. Other bits of the output gray code can be obtained by XORing binary code bit at that index and the previous index.





Gray to binary conversion :

1.The Most Significant Bit (MSB) of the binary code is always equal to the MSB of the given gray code.
2.Other bits of the output binary code can be obtained by checking the gray code bit at that index. If the current gray code bit is 0, then copy the previous binary code bit, else copy the invert of the previous binary code bit.

Image source : Google

Fundamentals of Digital Systems and Logic Families

American Standard Code for Information Interchange (ASCII) Code:

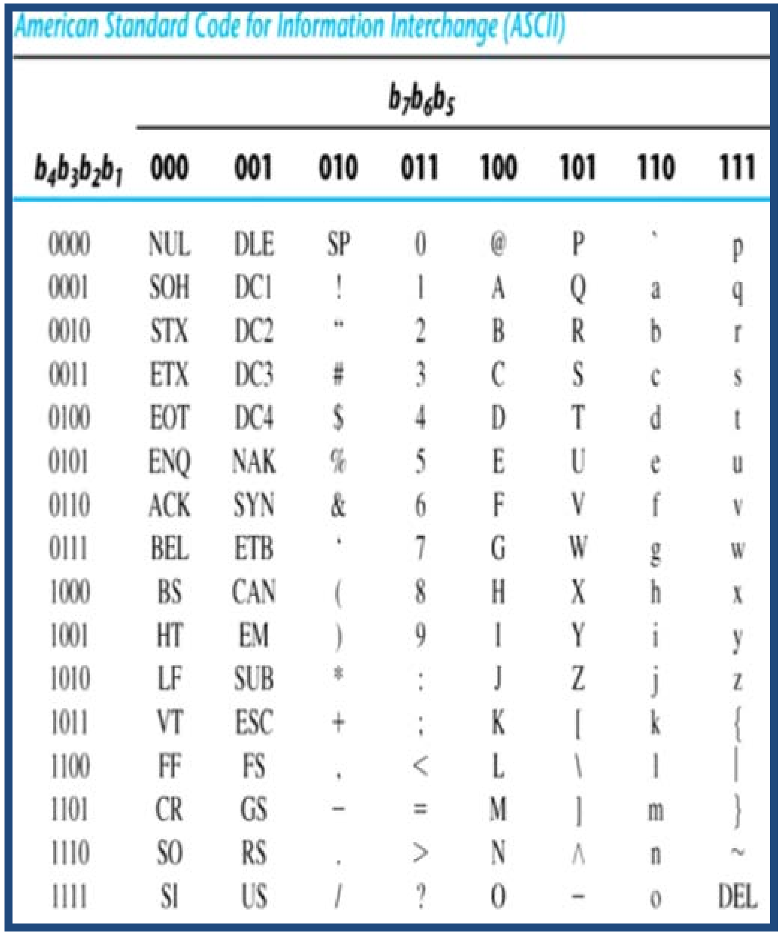
It uses 7-bits to represent, 94 Graphic printing characters and 34 Non-printing characters.

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).

ASCII has some interesting properties:

1. Digits 0 to 9 span Hexadecimal values 3016 to 3916.
2. Upper case A-Z span 4116 to 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

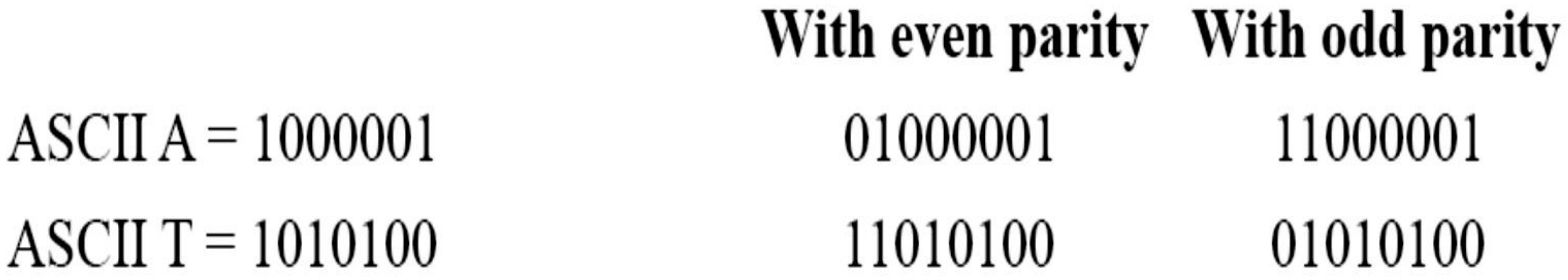
Image source : Google

Fundamentals of Digital Systems and Logic Families

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 parity**.**



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

Fundamentals of Digital Systems and Logic Families

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: 10001001*1* (even parity)

Message B: 100010010 (odd parity)

Fundamentals of Digital Systems and Logic Families

Hamming Code

Hamming code is a block-code that is capable of detecting up to two simultaneous bit errors and correcting single-bit errors. It was developed by R.W. Hamming for error correction.

- In this coding method, the source encodes the message by inserting redundant bits within the message. These redundant bits are extra bits that are generated and inserted at specific positions in the message itself to enable error detection and correction.
- When the destination receives this message, it performs recalculations to detect errors and find the bit position that has error.

Encoding a message by Hamming Code

Step 1 − Calculation of the number of redundant bits.

Step 2 − Position the redundant bits.

Step 3 − Calculate the values of each redundant bit.

Once the redundant bits are embedded within the message, this is sent to the user.

Fundamentals of Digital Systems and Logic Families

- Here, *k* parity bits are added to an *n*-bit message to form an (*n*+*k*) bit code, e.g. 3 parity bits are added to 4 bits of information to make a 7-bit hamming code.
- For this, the no. of parity bits, *k*, must satisfy the inequality: $2^k \geq n + k + 1$
- The location of each bit is assigned a decimal number (MSB:1 to LSB: *n*+*k*)
- *k*-parity checks are performed on selected bits of the code word.
- Result of the parity check is recorded as 1 if there is an error, otherwise it is 0.
- The resulting bits are $c_1\ c_2\ c_3 \ldots\ldots c_k$
- The decimal value of the binary word formed by $c_1\ c_2\ c_3 \ldots\ldots c_k$ gives the decimal value of the error location.

- Example: for 4-bit message, *n*=4, so, $2^k \geq 5 + k$. Therefore *k* must be atleast 3 (also 3 parity checks)
- So, total bits = 4 (message) + 3 (parity) = 7 bits

p_1	p_2	n_1	p_3	n_2	n_3	n_4
1	2	3	4	5	6	7

Fundamentals of Digital Systems and Logic Families

Error correction with (7, 4) Hamming code

- Parity checks: $p_1 n_1 n_2 n_4$ or bits 1, 3, 5, and 7. if error occurs then $c_3 = 1$ otherwise $c_3 = 0$
- Parity checks: $p_2 n_1 n_3 n_4$ or bits 2, 3, 6, and 7. if error occurs then $c_2 = 1$ otherwise $c_2 = 0$
- Parity checks: $p_3 n_2 n_3 n_4$ or bits 4, 5, 6 and 7. if error occurs then $c_1 = 1$ otherwise $c_1 = 0$

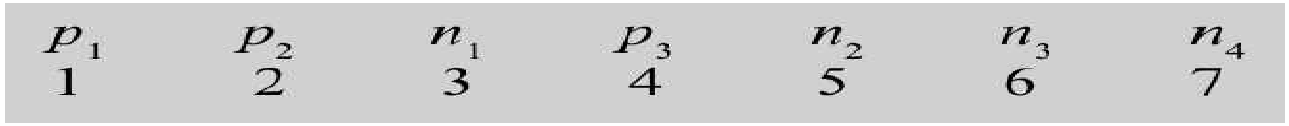
p_1	p_2	n_1	p_3	n_2	n_3	n_4
1	2	3	4	5	6	7

Error position	Position number		
	c_1	c_2	c_3
0 (no error)	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

- p_1, p_2, p_3 are selected to have either even-parity or odd-parity system.

Fundamentals of Digital Systems and Logic Families

- Suppose **0110** is to be transmitted (in even parity system)



p_1	p_2	n_1	p_3	n_2	n_3	n_4
1	2	3	4	5	6	7

- p_1, p_2, p_3 are selected to have either even-parity or odd-parity system.

p_1	p_2	n_1	p_3	n_2	n_3	n_4
1	**1**	**0**	**0**	**1**	**1**	**0**

- Now suppose 3rd bit gets in error, 1110110
- Parity result for 1,3,5,7 : odd. So $c_3 = 1$
- Parity result for 2,3,6,7 : odd. So $c_2 = 1$
- Parity result for 4,5,6,7 : even. So $c_1 = 0$

c_1	c_2	c_3	Error is in bit-3 (location: 011)
0	**1**	**1**	

Fundamentals of Digital Systems and Logic Families

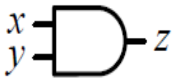
Binary logic consists of binary variables and a set of logical operations.

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.

AND

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

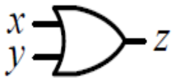
$z = x \bullet y = x\,y$



OR

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

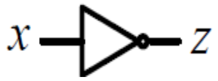
$z = x + y$

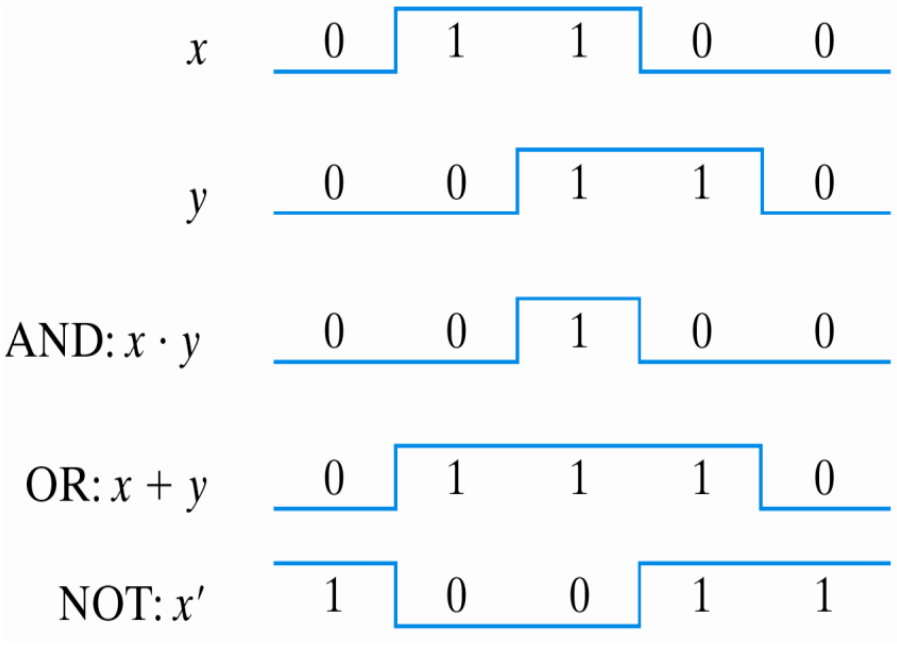


NOT

x	z
0	1
1	0

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





Fundamentals of Digital Systems and Logic Families

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

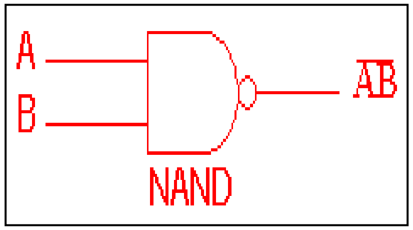
The variables are designated by letters of the alphabet, such as A, B, C, x, y, z, etc, with each variable having two and only two distinct possible values: 1 and 0.

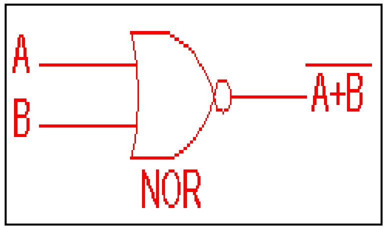
2 Input NAND gate		
A	B	$\overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0

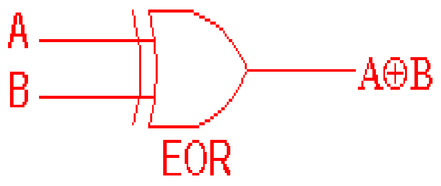
2 Input NOR gate		
A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

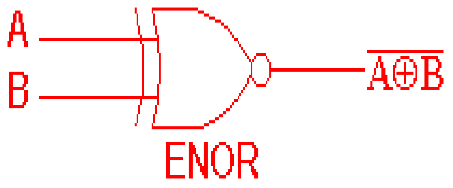
2 Input EXOR gate		
A	B	$A\oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

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



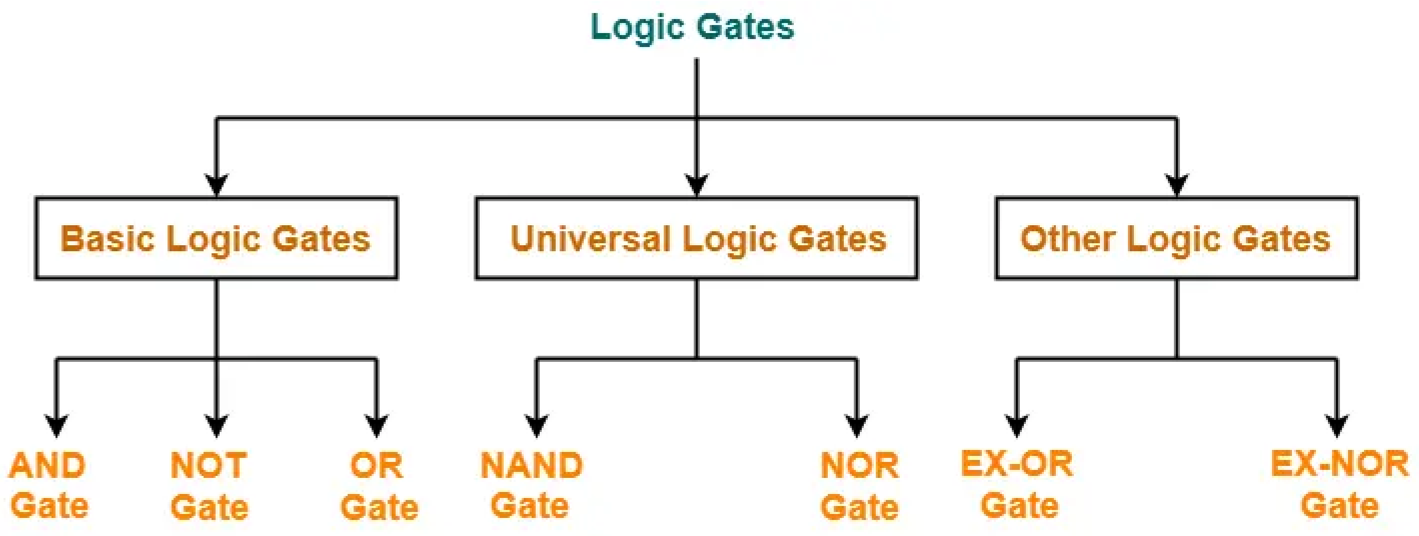




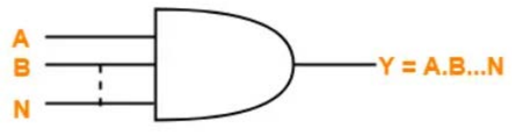


Universal Gates

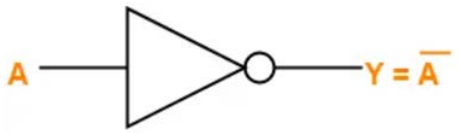
Fundamentals of Digital Systems and Logic Families



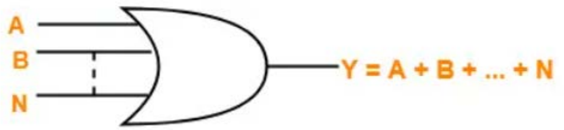
Types of Logic Gates



N-Input AND Gate

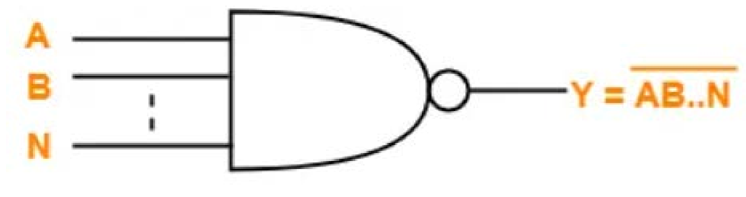


NOT Gate

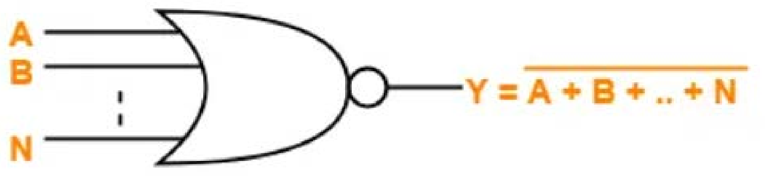


N-Input OR Gate

Fundamentals of Digital Systems and Logic Families



N-Input NAND Gate



N-Input NOR Gate

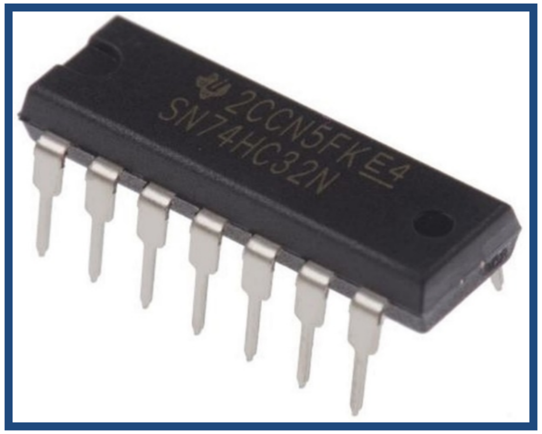
A **NAND Gate** is constructed by connecting a NOT Gate at the output terminal of the AND Gate
- The output of NAND gate is high ('1') if **at least** one of its inputs is low ('0')
- The output of NAND gate is low ('0') if **all** of its inputs are high ('1')

A **NOR Gate** is constructed by connecting a NOT Gate at the output terminal of the OR Gate
- The output of OR gate is high ('1') if **all** of its inputs are low ('0')

The output of OR gate is low ('0') if **any** of its inputs is high ('1')

Fundamentals of Digital Systems and Logic Families

Two Input logic gate IC 74xx series:



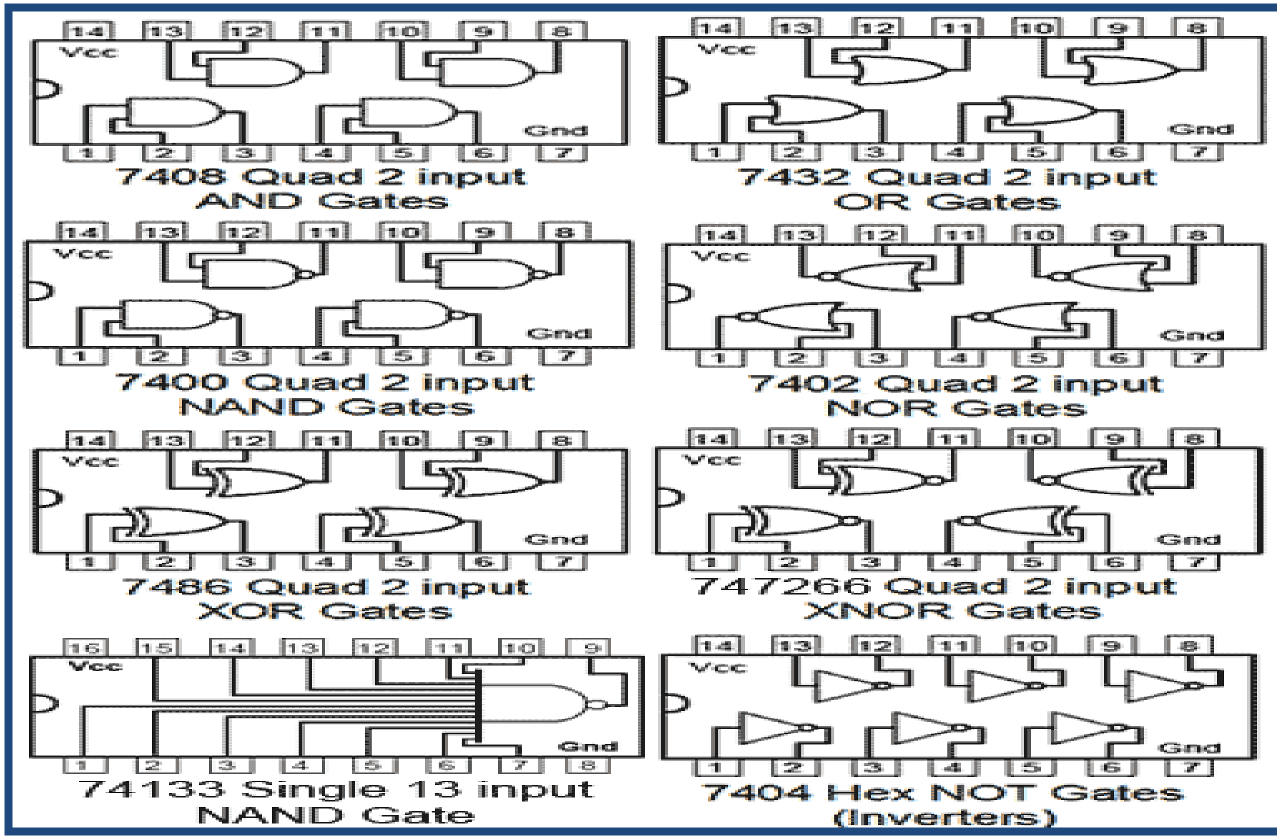
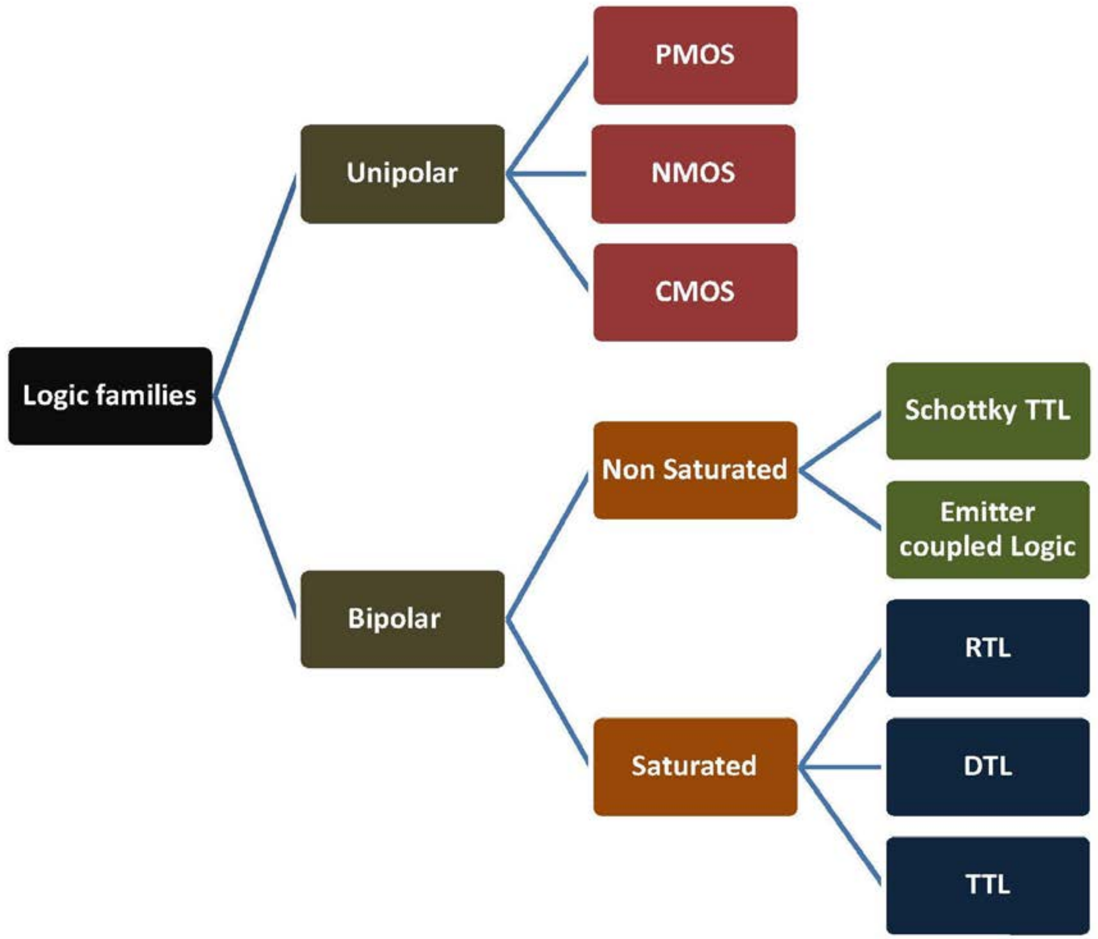

Vcc
Gnd
7408 Quad 2 input AND Gates
7432 Quad 2 input OR Gates
7400 Quad 2 input NAND Gates
7402 Quad 2 input NOR Gates
7486 Quad 2 input XOR Gates
747266 Quad 2 input XNOR Gates
74133 Single 13 input NAND Gate
7404 Hex NOT Gates (Inverters)

Image source : Google

Fundamentals of Digital Systems and Logic Families

The digital integrated circuits are designed using bipolar devices (**Bipolar logic families**) or Metal Oxide Semiconductor (MOS) (**Unipolar family**), or a combination of both.



In non-saturated logic, the transistor is switched between the off and active regions.

In saturated logic, the transistor is switched between the off and saturation regions.

Fundamentals of Digital Systems and Logic Families

Common Digital Logic Families:

1. RTL (Resistor-transistor logic)
2. DTL (Diode-transistor logic)
3. TTL (Transistor -transistor logic)
4. ECL (Emitter-coupled logic)

$\left.\begin{array}{l}\text{RTL}\\ \text{DTL}\\ \text{TTL}\end{array}\right\}$ Saturated

Un-saturated (non-saturated)

Characteristics of Logic Families:

1. Logic Assignments
2. Logic voltage levels
3. Noise margin
4. Supply voltage
5. Propagation delay
6. Fan-in/ Fan-out
7. Operating speed
8. Operating temperature
9. Power Dissipation
10. Speed-Power Product (**F**igure **of M**erit)

Fundamentals of Digital Systems and Logic Families

Logic Assignments

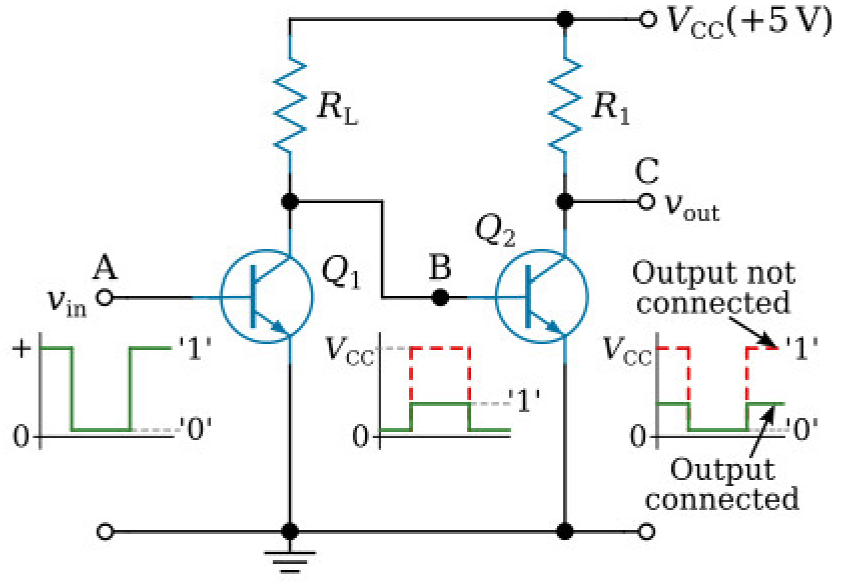
- Two possible logic assignments, positive logic or negative logic, are used to implement a Boolean function. For positive logic, a logical "one" is represented by a "high" voltage level, and a logical "zero" is represented by a "low" voltage. The reverse is used to represent negative logic.

Logic Voltage Levels

- Logic circuits are normally connected in cascade; that is, the output from one gate is connected to the input of the following gate, and so on. Thus, the switching behavior of one circuit depends not only on its own output characteristic, but may also depend on the input characteristic of the next gate.

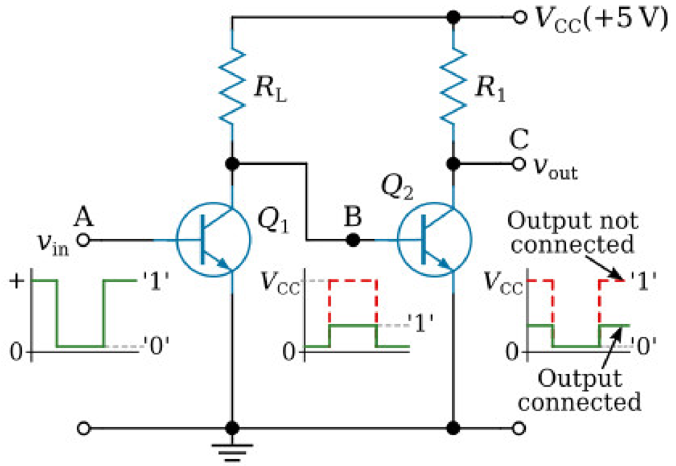
Fundamentals of Digital Systems and Logic Families

- Consider the simple case of one inverter circuit driving another as shown in the figure below. With the second transistor disconnected, the output of Q1 will swing from $V_{CE\ (saturated)}$ (about 0.1 volts), up to V_{CC} (5 V). However, with the second stage connected, the positive voltage at the collector of Q1 will be limited to that of the base-emitter voltage of transistor Q2. Thus, the voltage swing at the output of transistor Q1 will now be from 0.1 to about 0.7 V



Fundamentals of Digital Systems and Logic Families

- The output swing of transistor Q1, from 0.1 to 0.7 V, gives satisfactory operating points for the input of Q2, which in turn will give the same swing at its output when it is connected to the input of the next stage.
- The conditions for satisfactory switching between the two states are as follows: when Q1 is saturated, its collector voltage must be low enough to keep Q2 cut off; and when Q1 is cut off, the base current of Q2 (dependent on R_L) must be high enough to saturate Q2.
- For example, standard TTL voltage levels fix the logic 0 level between 0 and +0.8 V, while a logic 1 is between +2.4 and +5 V.



Fundamentals of Digital Systems and Logic Families

Noise Margin

- In logic systems, the word noise refers to any unwanted voltage (AC or DC) appearing at the input of a logic circuit. If such a noise voltage were high enough, it could cause the circuit to change state with no change in the input signal voltage.
- Both DC and AC noise should be considered in the design of digital systems. The DC noise is the steady drift in the voltage levels of the logic states, and AC noise is the narrow pulses that are created, primarily, by switching transients.

Supply Voltage

- Integrated circuit manufacturers include the supply voltage limits of their digital circuits on the data sheets. For example, the correct operation of standard TTL circuits should fall between the 4.75 V and 5.25 V limits of the supply voltage. Within these limits, the device is designed to operate compatibly with the other TTL devices.

Fundamentals of Digital Systems and Logic Families

The DC noise margin (immunity) of a digital circuit is the ability of that circuit to maintain a logic state in the presence of DC noise.
The DC noise margin is expressed by the following equations

$$N_{\mathrm{H}} = V_{\mathrm{OHmin}} - V_{\mathrm{IHmin}}$$

$$N_{\mathrm{L}} = V_{\mathrm{ILmax}} - V_{\mathrm{OLmax}}$$

Where,
N_L - noise immunity of the digital circuit input when the input signal is LOW,
N_H - noise immunity of the digital circuit input when the input signal is HIGH,
V_{ILmax} - maximum input voltage that can be read by the circuit as LOW,
V_{OLmax} - maximum output voltage that can represent LOW,
V_{OHmin} - minimum output voltage that can represent HIGH,
V_{IHmin} - minimum input voltage that can be read by the circuit as HIGH.

Fundamentals of Digital Systems and Logic Families

Integrated circuit manufacturers include the voltage level limits of their digital circuits on the data sheets. From those values, the noise immunity can be computed.

For example, a standard TTL logic gate has the following input and output level characteristics:
V_{ILmax} = 0.8 V, V_{OLmax} = 0.4 V
V_{IHmin} = 2 V, V_{OHmin} = 2.4 V

Substituting these values into the noise immunity equations:

$$N_L = V_{ILmax} - V_{OLmax} = 0.8 - 0.4 = 0.4\ \mathrm{V}$$

and

$$N_H = V_{OHmin} - V_{IHmin} = 2.4 - 2 = 0.4\ \mathrm{V}$$

These values indicate that for reliable operation, the DC noise on the signal lines should not exceed 0.4 V.

Fundamentals of Digital Systems and Logic Families

Propagation delay:

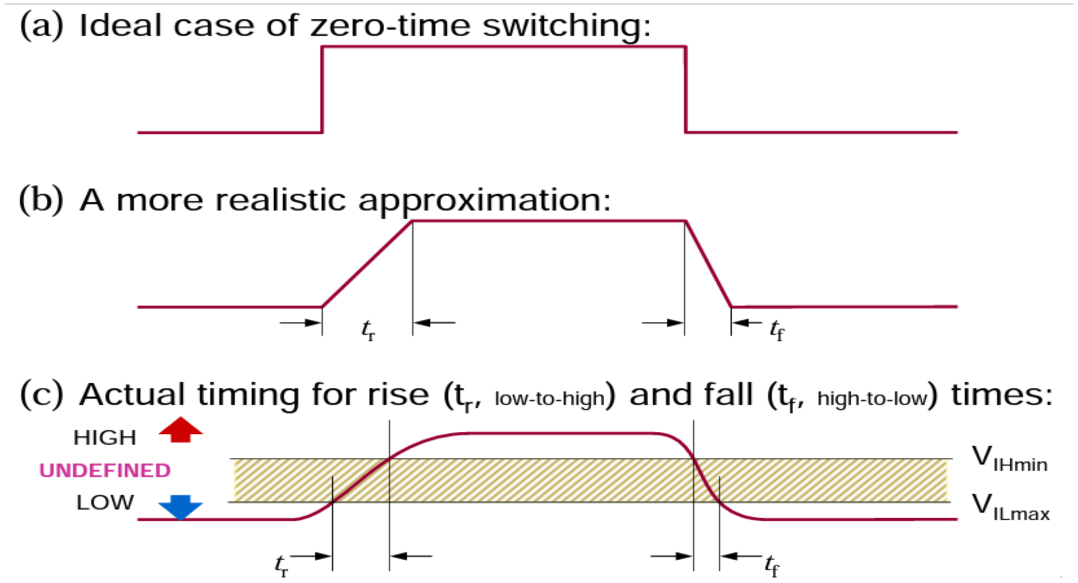
When the input to a logic gate is changed, the output will not change immediately.

•The switching elements within a gate take a finite time to react to a change (transition) in input.

•As a result, the change in the gate output is delayed w.r.t. to the input change.

•Such delay is called the **propagation delay** of the logic gate (t_p)

•The propagation delay for a 0-to-1 output change (t_{pLH}) may be different than the delay for a 1-to-0 change (t_{pHL})


(a) Ideal case of zero-time switching:
(b) A more realistic approximation:
t_r
t_f
(c) Actual timing for rise (t_r, low-to-high) and fall (t_f, high-to-low) times:
HIGH
UNDEFINED
LOW
V_{IHmin}
V_{ILmax}
t_r
t_f

Fundamentals of Digital Systems and Logic Families

Fan-in/Fan-out

Fan-out is the term used to describe the number of loads (inputs) that are driven from a single output. Sometimes it is necessary to better define the maximum fan-out capabilities of a circuit in terms of output current that can be supplied or sunk from a single output.

Fan-in is the number of inputs of a logic gate. Some devices provide increased fan-in by having provisions for the addition of expander circuits; fan-in also can be increased through combinational logic.

Operating Speed

It takes a finite time for a circuit to change from one logic state to the other. In a bipolar transistor, this time is that which is necessary for the base current to supply a charge to, or remove a charge from, the capacitive elements associated with the transistor structure, in order to produce the required voltage change at the output. In a circuit, additional time is required to charge any capacitance associated with the load. This time delay (between input and output) is called the propagation delay of the circuit. Delay values vary considerably depending on the particular circuit, but with integrated logic circuits, the delay is typically in the range of 2 to 50 ns per gate. The overall propagation delay of a complete logic system will be the delay per gate multiplied by the number of gates in series.

Fundamentals of Digital Systems and Logic Families

Operating Temperature

All semiconductor devices are temperature sensitive. This is due primarily to the characteristics of the P-N junctions changing with temperature. Silicon transistors and diodes can operate satisfactorily with junction temperatures up to about 200° C; above this temperature, the characteristics become so poor as to result in inferior performance. Resistors may also change their value with a change of temperature due to the thermal generation of carriers and the change in the mobility of the carriers.

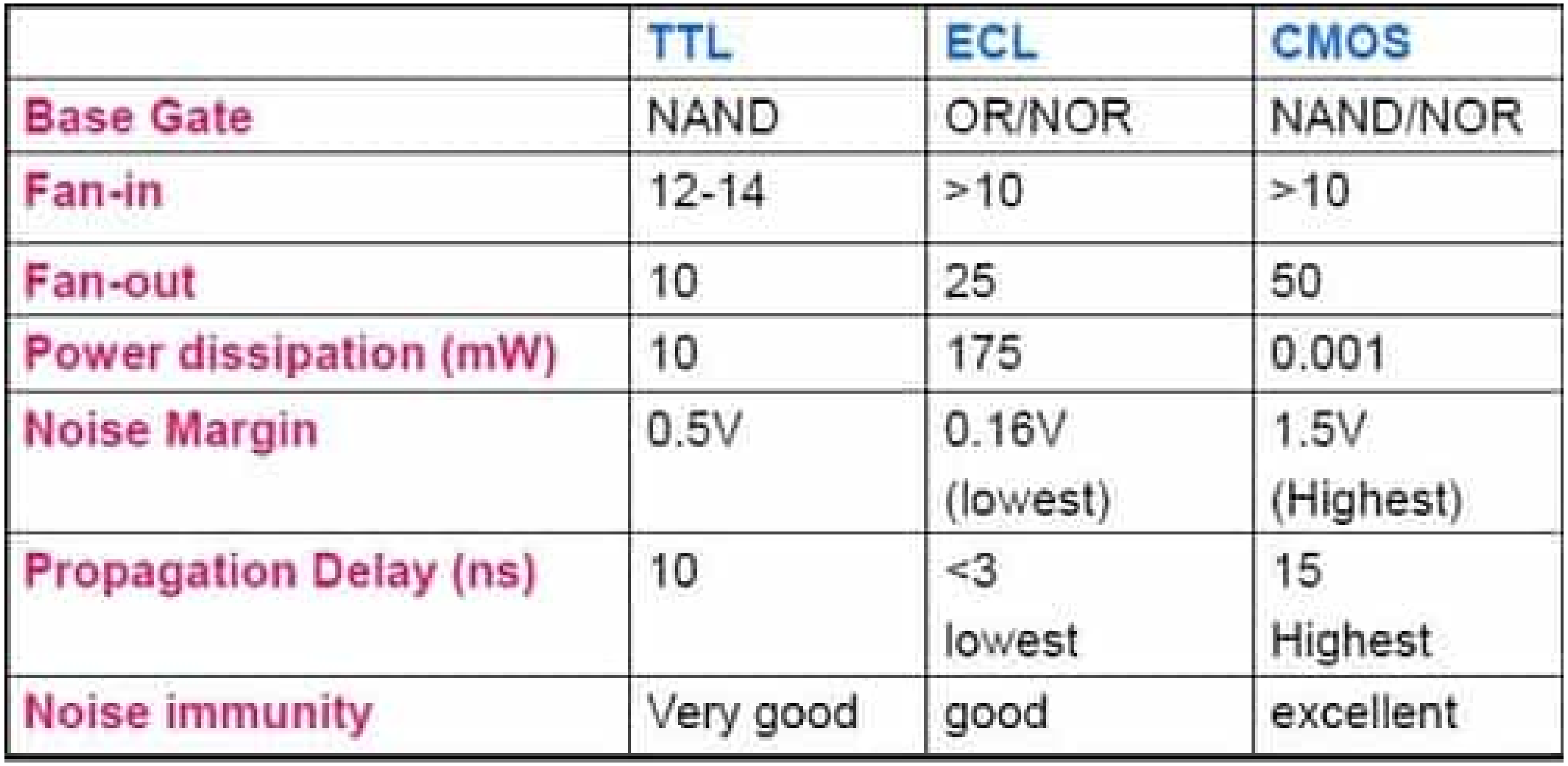
Power Dissipation

The power dissipation of a logic circuit is usually defined as the supply power required for the gate to operate with a 50 percent duty cycle, that is, equal times in the 0 and 1 logic states. The power dissipation of typical integrated logic circuits ranges from a few milliwatts to about 50 mW per gate, depending on the type of circuit. In general, high-speed circuits with short propagation delays require higher power.

Speed-Power Product (**F**igure **of M**erit)

The product of power dissipation and propagation delay for a given logic or IC family (should be as low as possible)

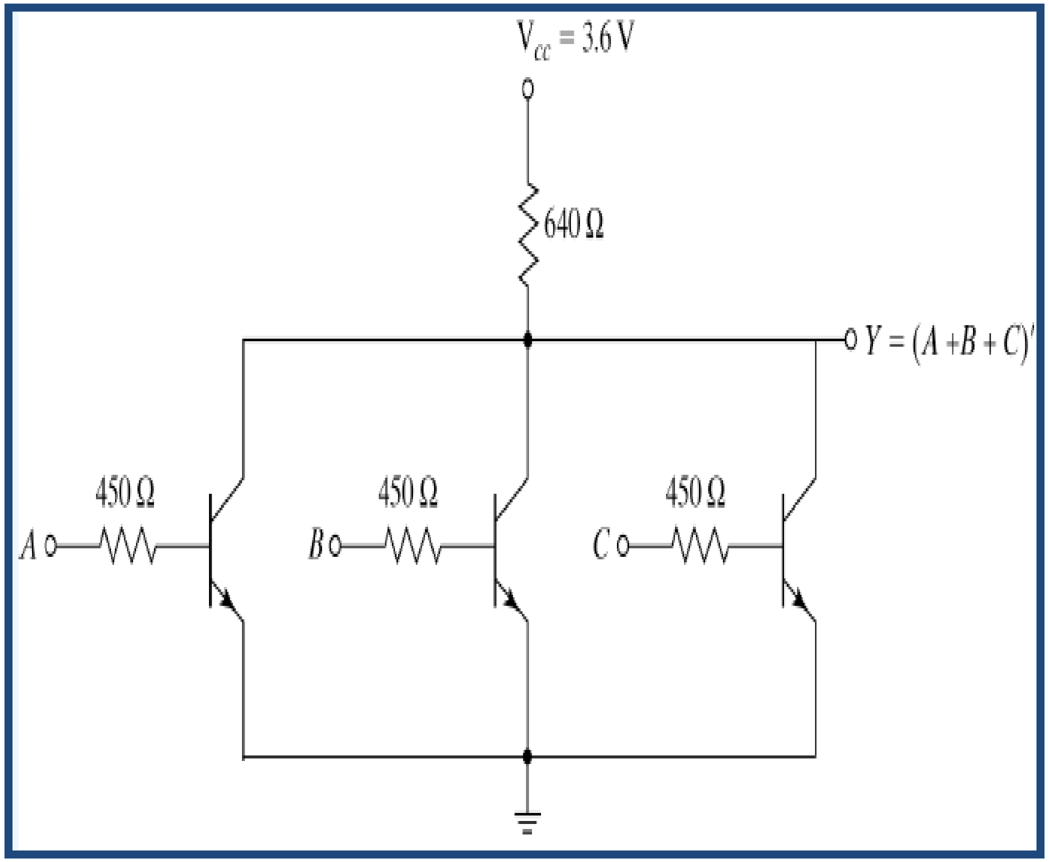
Fundamentals of Digital Systems and Logic Families



	TTL	ECL	CMOS
Base Gate	NAND	OR/NOR	NAND/NOR
Fan-in	12-14	>10	>10
Fan-out	10	25	50
Power dissipation (mW)	10	175	0.001
Noise Margin	0.5V	0.16V (lowest)	1.5V (Highest)
Propagation Delay (ns)	10	<3 lowest	15 Highest
Noise immunity	Very good	good	excellent

Fundamentals of Digital Systems and Logic Families

RTL-NOR



DTL-NAND

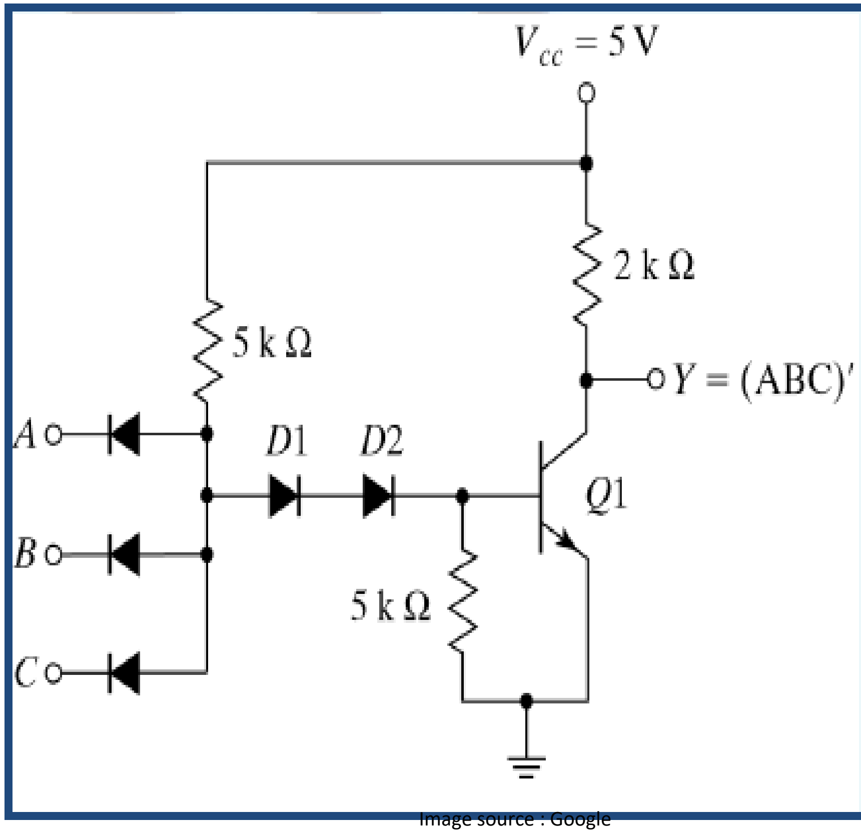
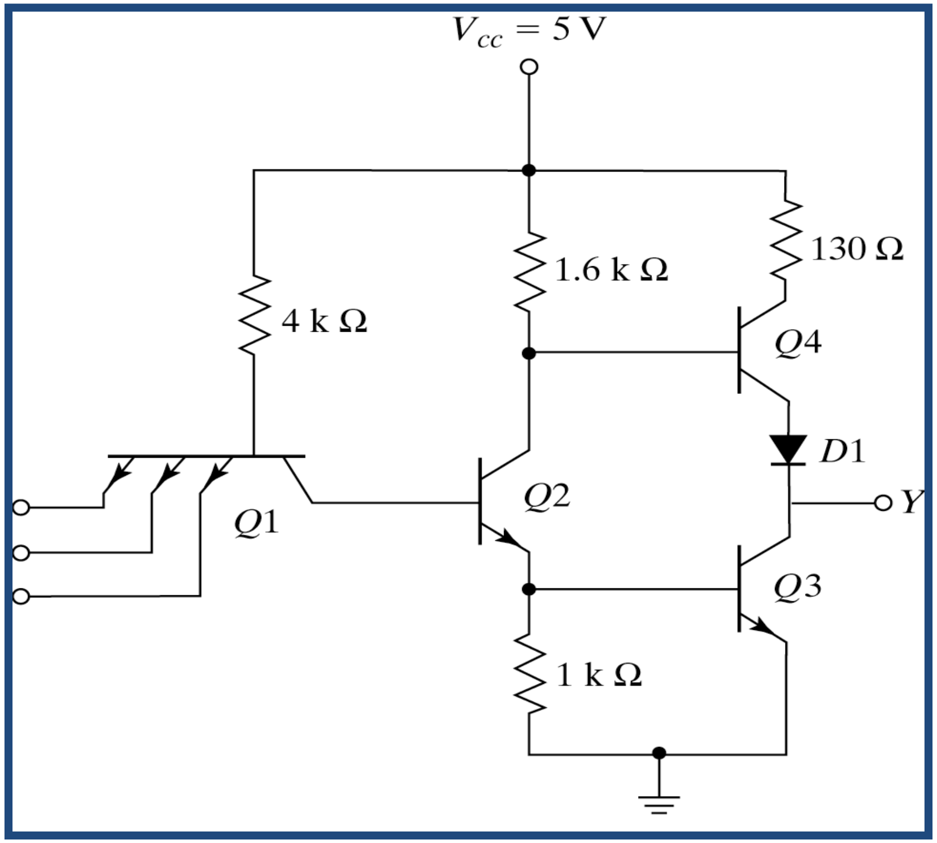


Image source : Google

Fundamentals of Digital Systems and Logic Families

Transistor-Transistor Logic (TTL):



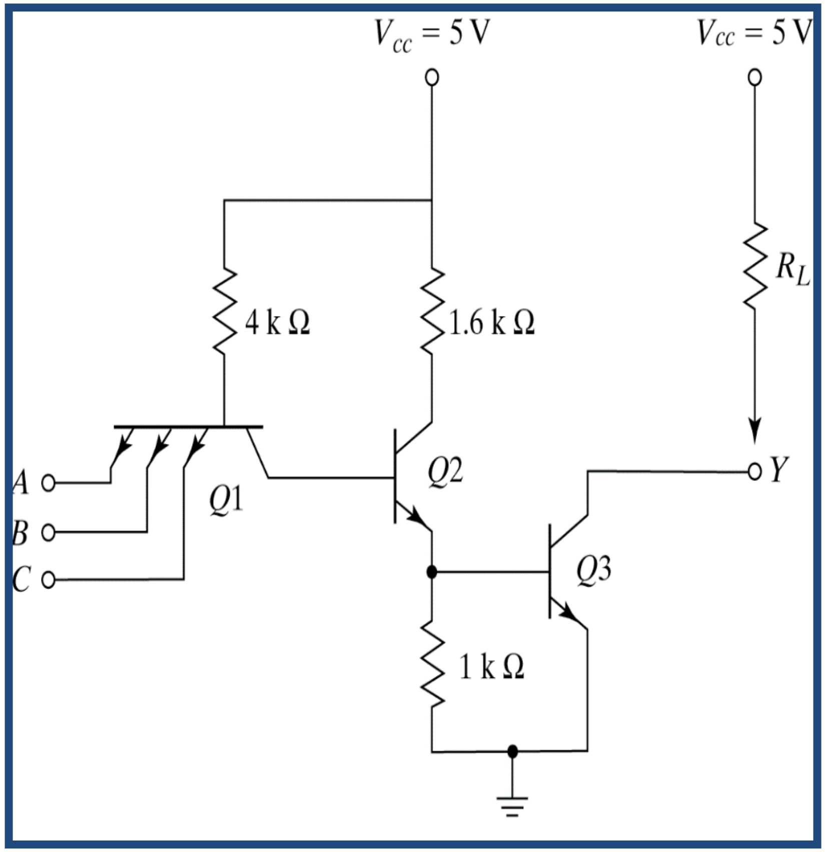


Image source : Google

Fundamentals of Digital Systems and Logic Families

EMITTER-COUPLED LOGIC (ECL):

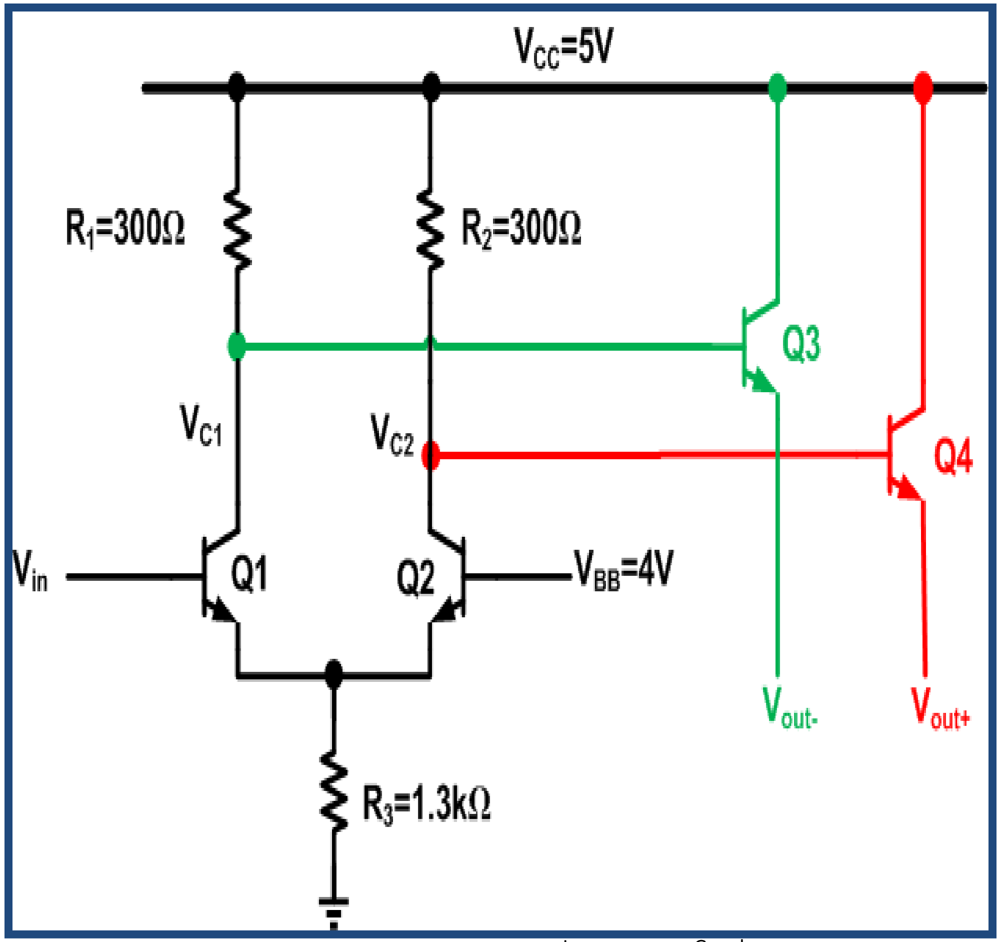


Image source : Google

Fundamentals of Digital Systems and Logic Families

- CMOS consumes very little power, has excellent noise immunity, and is used with a wide range of voltages.
- TTL can drive more current and uses more power than CMOS

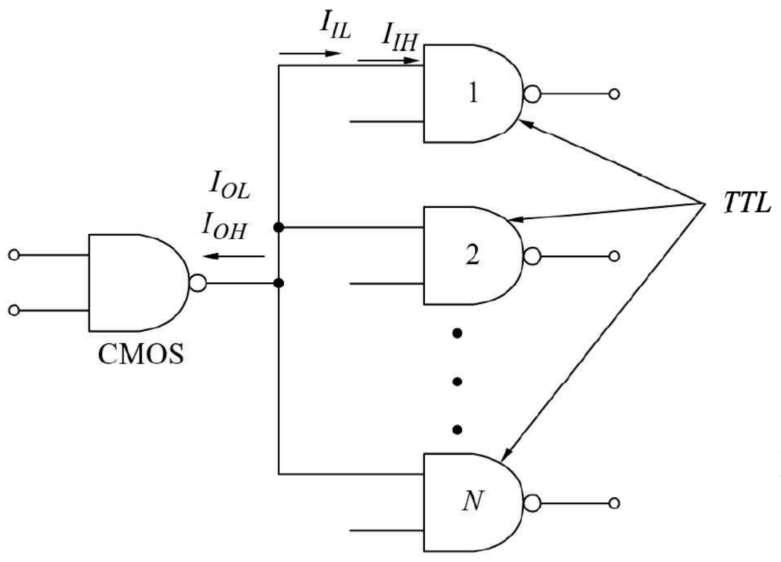


Fig. 4.40 ***A CMOS Gate Driving N TTL Gates***

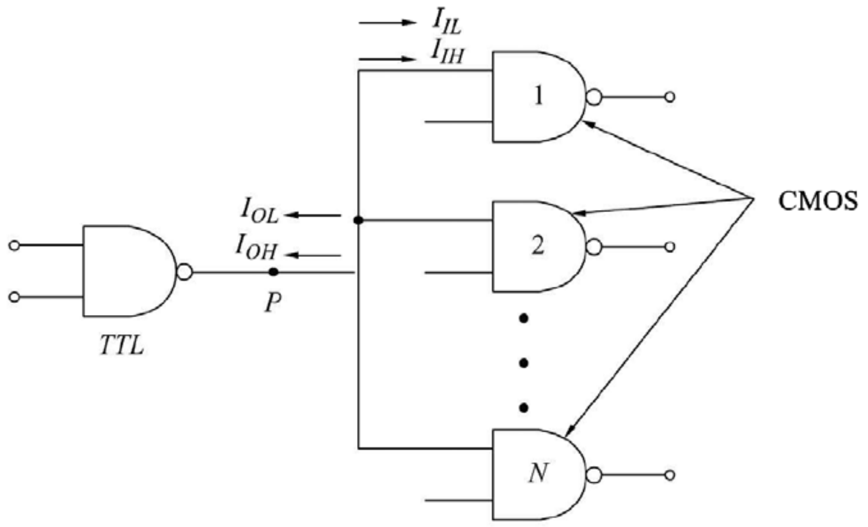


Fig. 4.41 ***A TTL Gate Driving N CMOS Gates***

Image sources: R P Jain, Modern Digital Electronics, pg. 149-150.