IMPORTANT FORMULAS ON DIGITAL ELECTRONICS

Number System and Codes

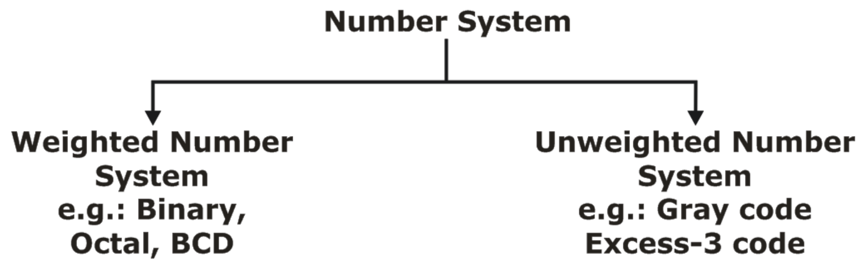

Number System
Weighted Number System e.g.: Binary, Octal, BCD
Unweighted Number System e.g.: Gray code Excess-3 code

Fig. 1

Types of Number System:

The number can be represented in various ways to show the data and process it on the processing devices.

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

Table 1: Counting in different number system

A number system with base ‘r’, contains ‘r’ different digits and they are from 0 to r –1.

Decimal to other codes conversions:

To convert decimal number into other system with base ‘r’, divide integer part by r and multiply fractional part with r.

Other codes to Decimal Conversions:

$$(x_2x_1x_0.y_1y_2)_r \to (A)_{10} \quad , \qquad A = x_2r^2 + x_1r + x_0 + y_1r^{-1} + y_2r^{-2}$$

Hexadecimal to Binary:

Convert each Hexadecimal digit into 4 bits binary.

$(5AF)_{16} \rightarrow \frac{(0101}{5}\frac{1010}{A}\frac{1111)_2}{F}$

Binary to Hexadecimal:

Grouping of 4 bits into one hex digit.

$(110101.11)_2 \rightarrow 00110101.1100 \rightarrow (35.C)_{16}$

Octal Binary and Binary to Octal:

Same procedure as discussed above but here group of 3 bits is made.

Codes:

Binary coded decimal (BCD):

- In BCD code each decimal digit is represented with 4 bit binary format.

  $\text{Eg}: (943)_{10} \left(\underset{9}{1001}\ \underbrace{0100}_{4}\ \underset{3}{0011}\right)_{BCD}$

- It is also known as 8421 code.
- Invalid BCD codes are the codes whose decimal equivalent is more than 9. i.e. Valid BCD codes ranges from 0 to 9.

  Total number of codes possible $\rightarrow 2^4 \Rightarrow 16$

  Valid BCD codes $\rightarrow 10$

  Invalid BCD codes $16 - 10 \Rightarrow 6$

  There 1010, 1011, 1100, 1110 and 1111

Excess-3 codes: (BCD + 0011)

- It can be derived from BCD by adding '3' to each coded number.
- It is unweighted and self-complementing code.

Gray Code:

It is also known as minimum change codes or unit distance code or reflected code.

Binary code to Gray code:

In order to find Gray code from Binary code, XOR Gate is applied between the present binary bit and the next binary bit starting from the MSB side(keeping MSB of Binary and Gray as same) e.g.

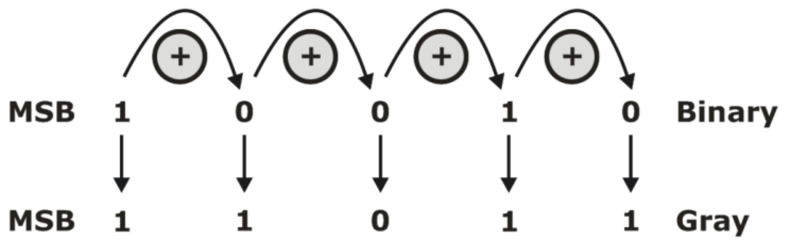


Fig. 2

Gray code to Binary code:

In order to find Binary code from Gray code, XOR Gate is applied between the present binary bit and the next gray bit starting from the MSB side (keeping MSB of Gray and Binary as same) e.g.

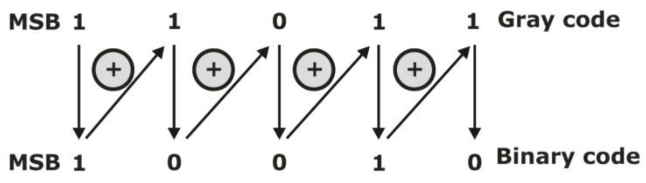


Fig. 3

Alpha Numeric code:

EBCDIC (Extended BCD interchange code)

It is an 8-bit code. It can represent 128 possible characters.

- Parity method is most widely used schemes for error detection.
- Hamming code is most useful error correcting code.
- BCD code is used in calculators, counters.

Complements:

Its base is r then we can have two complements.

(i) (r – 1)’s complement

(ii) r’s complement

To determine(r – 1)’s complement: First write maximum possible number in the given system and subtract the given number.

To determine r’s complements: (r – 1)’s complement + 1

i.e. First write (r – 1)’s complement and then add 1 to LSB

Example:

Q. Find 7’s and 8’s complement of 2456

Sol. 7’s Complement $\begin{array}{r} 7777 \\ \underline{-2456} \\ 5321 \end{array}$

8’s Complement $\begin{array}{r} 5321 \\ \underline{+1} \\ 5322 \end{array}$

Q. Find 2’s complement of 101.110

Sol. 1’s complements 010.001

For 2’s complement add 1 to the LSB

$$\begin{array}{lr} & 010.001 \\ \text{2's complement} & \underline{\quad +1} \\ & 010.010 \end{array}$$

Data Representation:

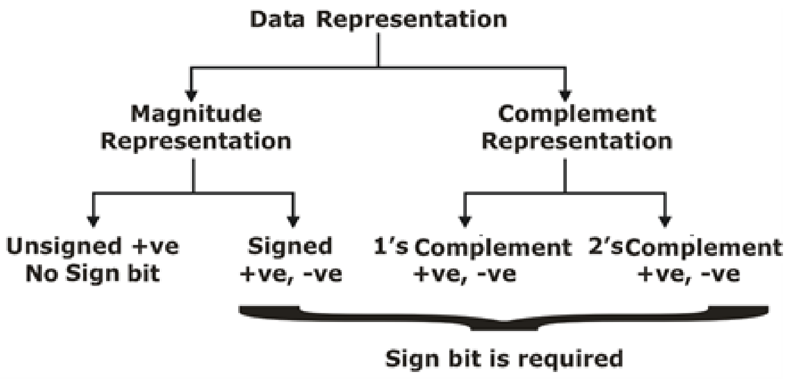


Fig. 4

Unsigned Magnitude:

Range with n bit

$\rightarrow 0$ to 2^{n-1} $\quad +5 \Rightarrow 101$

$-5 \Rightarrow$ Not possible

Signed Magnitude:

Range with n bit

$-6 \Rightarrow \underbrace{\overset{}{\underset{\text{sign bit}}{1}}\ 110}_{\text{with 4 bits}} \quad \underbrace{\underset{\text{sign bit}}{1}\ 0000110}_{\text{with 8 bits}}$

1's complements:

Range with n bit

$\rightarrow -\left(2^{n-1}-1\right)$ to $+\left(2^{n-1}-1\right)$

$+6 \Rightarrow 0110 \quad -6 \Rightarrow 1001$

2's complements:

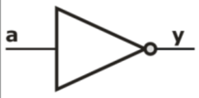
With n bit range -2^{n-1} to $(2^{n-1}-1)$

$+6 \Rightarrow 0110 \quad -6 \Rightarrow 1010$

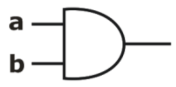
In any representation +ve numbers are represented similar to +ve number in sign magnitude.

Logic Gates

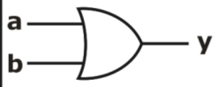
NOT Gate:

Symbol	Truth-table	Boolean
	a | y 0 | 1 1 | 0	$y = \overline{a}$

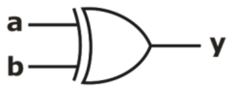
AND Gate:

Symbol	Truth-table	Boolean
	a b | y 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1	$y = ab$

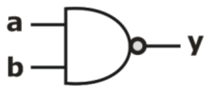
OR Gate:

Symbol	Truth-table	Boolean
	a b | y 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1	$y = a + b$

Exclusive OR Gate:

Symbol	Truth-table	Boolean
	a b | y 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0	$y = a \oplus b$

NAND Gate:

Symbol	Truth-table	Boolean
	a b | y 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0	$y = \overline{ab}$

NOR Gate:

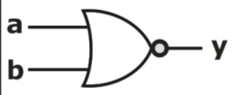
Symbol	Truth-table	Boolean
	a b | y 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0	$y = \overline{a + b}$

Fig.5

Note:- NAND and NOR are Universal Gates.

XOR gate:

Symbol of two input XOR gate

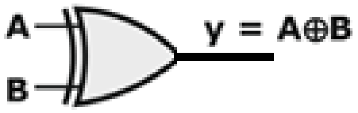


Fig.6

X-NOR gate:

Symbol for two input X-NOR gate

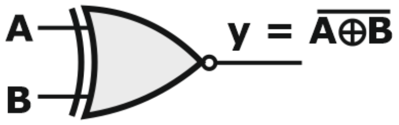


Fig.7

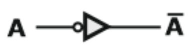
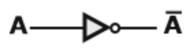
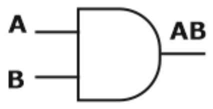
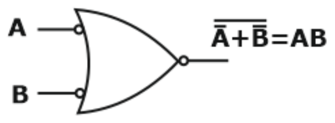
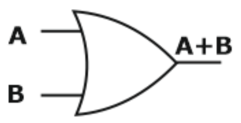
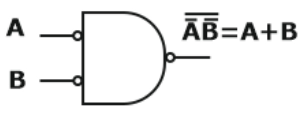
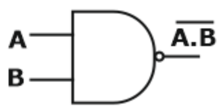
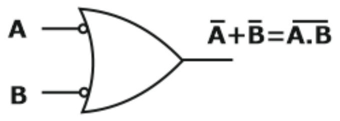
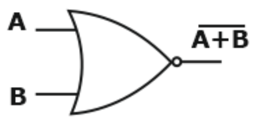
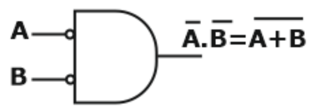
Logic	Normal symbol	Alternate symbol
NOT		
AND		
OR		
NAND		
NOR		

Table 2: Logic Gates representation

<u>Minimization</u>

Boolean Laws:

Commutative	$x \wedge y = y \wedge x$	$x \vee y = y \vee x$
Associative	$x \wedge (y \wedge z) = (x \wedge y) \wedge z$	$x \vee (y \vee z) = (x \vee y \vee z)$
Distributive	$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$	$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$
Idempotence	$x \wedge x = x$	$x \vee x = x$
Absorption	$x \wedge (x \vee y) = x$	$x \vee (x \wedge y) = x$
Consensus theorem	$(x \wedge y) \vee (\overline{x} \wedge z) \vee (y \wedge z)$ $= (x \wedge y) \vee (\overline{x} \wedge z)$	$(x \vee y) \wedge (\overline{x} \vee z) \wedge (y \vee z)$ $= (x \vee y) \wedge (\overline{x} \vee z)$
Combining	$(x \wedge y) \vee (x \wedge \overline{y}) = x$	$(x \vee y) \wedge (x \vee \overline{y}) = x$
DeMorgan's	$\overline{(x \wedge y)} = \overline{x} \vee \overline{y}$	$\overline{(x \vee y)} = \overline{x} \wedge \overline{y}$

Table 3: Boolean law expressions

Dual of a function:

The dual of logic function, f, is the function f^D derived from f by substituting a $\wedge$ for each $\vee$, and $\vee$ for each $\wedge$. 1 for each 0, and 0 for each 1.

$$f(a,b) = (a \wedge b) \vee (b \wedge c) \qquad f^D(a,b) = (a \vee b) \wedge (b \vee c)$$

Boolean Functions:

Any Boolean expression can be expressed in two forms

- Sum of Product form (SOP)
- Product of Sum form (POS)

SOP form:

The SOP expression usually takes the forms of two or more variables ORed together.

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

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

SOP forms are used to write logical expression for the output becoming logic '1'.

POS form:

The POS expression usually takes the form of two or more OR variables within parentheses, ANDed with two or more such terms.

Karnaugh Map (K-MAP):

The K-map is a graphical method which provides a systematic method for simplifying the Boolean expressions. In n variable K-map, there are 2^n cells.

Simplification Rules:

1. Construct the K-map and place 1's in those cells corresponding to the 1's in the truth table. Place the 0's in the other cells.
2. Examine the map for adjacent 1's and loop those 1's which are not adjacent to any other 1's. These are called isolated 1's.
3. Next, look for those 1's which are adjacent to only one other 1. Loop any pair containing such a 1.
4. Loop any octet even if it contains some 1's that have already been looped.
5. Loop any quad that contains one or more 1's which have not already been looped, making sure to use the minimum number of loops.
6. Form the OR sum of all the terms generated by each loop.

Implicants:

Implicants is a product term on the given function, and for that combination, the function output must be 1.

Prime Implicant (PI)

Prime implicant is a smallest possible product term of the given function.

Essential Prime Implicants (EPI)

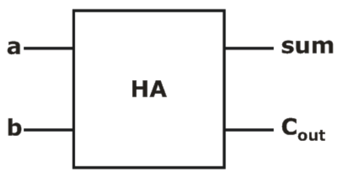
EPI is a prime implicant it and must cover at least one minterm, which is not covered by other PI.

<u>Combinational Circuits</u>

The combinational circuit has 'n' input variables and 'm' output variables. Since, the number of input variables is n, there are 2^n possible combinations of bits at the input.

Adders:

Half Adder



Logic circuit

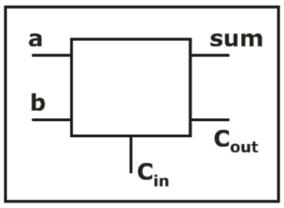
a	b	c_{out}	sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Truth table

$$Sum = \bar{a}b + a\bar{b} = a \oplus b$$
$$c_{out} = ab$$

Equations

Full Adder



Logic circuit

c_{in}	a	b	c_{out}	sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Truth table

$$Sum = \overline{c_{in}}X + c_{in}\overline{X} = c_{in} \oplus X = c_{in} \oplus a \oplus b$$
$$c_{out} = c_{in}(a \oplus b) + ab$$

Equations

Fig. 8

Binary parallel adder:

An n-bit parallel adder can be constructed using n number of full adders which are connected in parallel, and hence; it is also known as parallel adder such that the previous carry or carry input for adder 0 is set to zero. The carry output of each adder is connected to the carry input of the next higher order adder. Hence, it is also known as carry propagate adder.

Carry look ahead adder:

The look ahead carry adder speeds up the operation by eliminating this ripple carry delay. It examines all the input bits simultaneously, and also generates the carry in bits for all the stages simultaneously. The method of speeding up the addition process is based on the two additional functions of the full adder called the carry generate and carry propagate functions.

Multiplexer:

- 2^n inputs; 1 output and 'n' select lines.
- It can be used to implement Boolean function by selecting select lines as Boolean variables.
- For implementing 'n' variables Boolean function, $2^n \times 1$ MUX is enough.
- For implementing "n+1" variables Boolean function, $2^n \times 1$ MUX + NOT gate is required.
- For implementing "n+2" variables Boolean function, $2^n \times 1$ MUX + Combinational Ckt is required.
- If you want to design $2^m \times 1$ MUX using $2^n \times 1$ MUX. You need two $2^n \times 1$ MUXes.

Demultiplexer:

- 1 input; 2^n outputs and 'n' select lines.
- It can be used to implement Boolean function by selecting select lines as Boolean variables.
- For implementing 'n' variables Boolean function, 1×2^n DeMUX is enough.
- For implementing "n+1" variables Boolean function, 1×2^n DeMUX + NOT gate is required.
- For implementing "n+2" variables Boolean function 1×2^n DeMUX + Combinational Circuit is required.
- If you want to design 1×2^m DeMUX using 1×2^n DeMUX. You need two $2^n \times 1$ DeMUXes.

Comparison between Multiplexer and Demultiplexer:

S.No.	**Parameter of comparison**	**Multiplexer**	**Demultiplexer**
1.	Type of logic circuit	Combinational	Combinational
2.	Number of data inputs	M	1
3.	Number of select inputs	N	N
4.	Number of data output	1	M
5.	Relation between input/output lines and select lines	$m = 2^n$	$M = 2^N$
6.	Operation principle	Many to 1 or data selector	1 to many or data distributor

Table 4: Comparison between Multiplexer and Demultiplexer

Decoder:

- n input and 2^n output.
- Used to implement the Boolean function. It will generate required min terms at the output and those terms should be "OR" ed to get the results.
- Suppose it consists of more min terms then connect the max terms to NOR gate then it will give the same output with less no. of gates.
- For carry look ahead adder, delay = $2 \times t_{pd}$.

Encoder:

An encoder is a combinational logic circuit that performs the inverse operation of a decoder. An encoder has 2^n (or fewer) input lines and n output lines.

Octal to Binary Encoder:

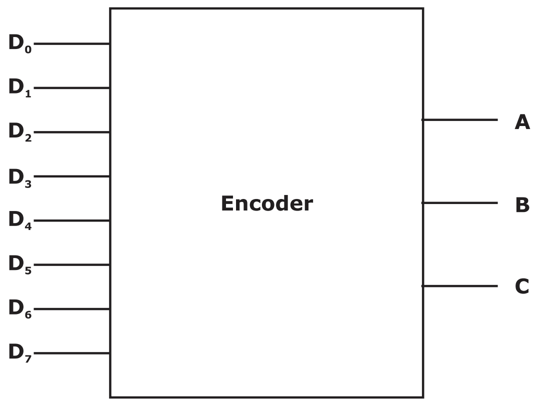


Fig. 9

The logical expression for the outputs are as follows:

$A = D_4 + D_5 + D_6 + D_7$

$B = D_2 + D_3 + D_6 + D_7$

$C = D_1 + D_3 + D_5 + D_7$

Realization of a Octal to Binary Encoder using Logic Gates:

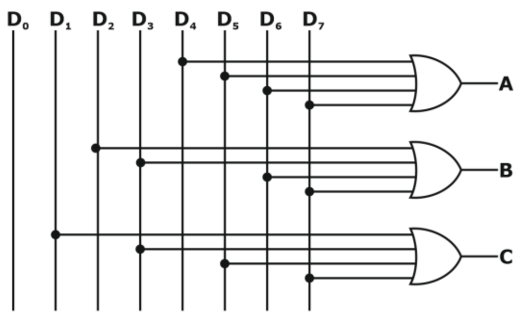


Fig. 10

Subtractors:

Half subtractor:

A half subtractor is a combinational logic circuit, which performs the subtraction of two 1-bit numbers. It subtracts one binary digit from another to produce a DIFFERENCE output & a BORROW output.

The truth table of half – subtractor, where A, B are the inputs, and difference (D) and borrow (B) are the outputs.

Inputs		**Outputs**	
A	**B**	**D**	**B_{out}**
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Table 6: Truth table of Half – Subtractor.

Difference, $D = \bar{A}B + A\bar{B} = A \oplus B$

Borrow, $B_{out} = \bar{A}B$

Logic Diagram:

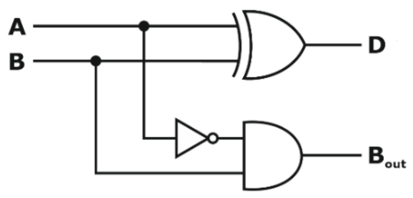


Fig. 11

Full Subtractor:

A full subtractor performs subtraction operation on two bits, a minuend and a subtrahend.

		Inputs	**Outputs**	
A	**B**	**B_{in}**	**D**	**B_{out}**
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0

1	1	1	1	1

Table 7: Truth table of Full subtractor.

Difference, $D = A \oplus B \oplus B_{in}$

Borrow, $B_{in} = \bar{A}B + \bar{A}B_{in} + BB_{in}$

Logic Diagram:

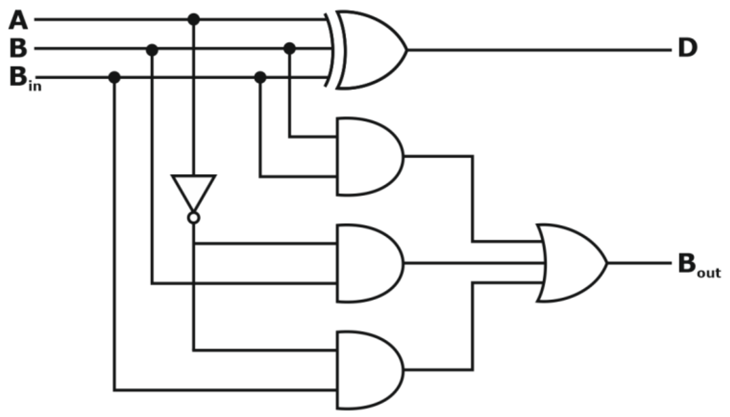


Fig. 12

Comparator:

The comparator has three outputs namely A>B, A=B and A<B. Depending upon the result of comparison, one of these outputs will go high.

2-bit Magnitude Comparator:

Let the two 2-bit numbers be $A = A_1A_0$ and $B = B_1B_0$

Inputs				Outputs		
A_1	A_0	B_1	B_0	X(A<B)	Y(A=B)	Z(A > B)
0	0	0	0	0	1	0
0	0	0	1	1	0	0
0	0	1	0	1	0	0
0	0	1	1	1	0	0
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	1	0	0
0	1	1	1	1	0	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1

1	0	1	0	0	1	0
1	0	1	1	1	0	0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	0	1	0

Table 9: Truth Table for a 2-bit Comparator.

Logic Diagram of 2-bit Comparator:

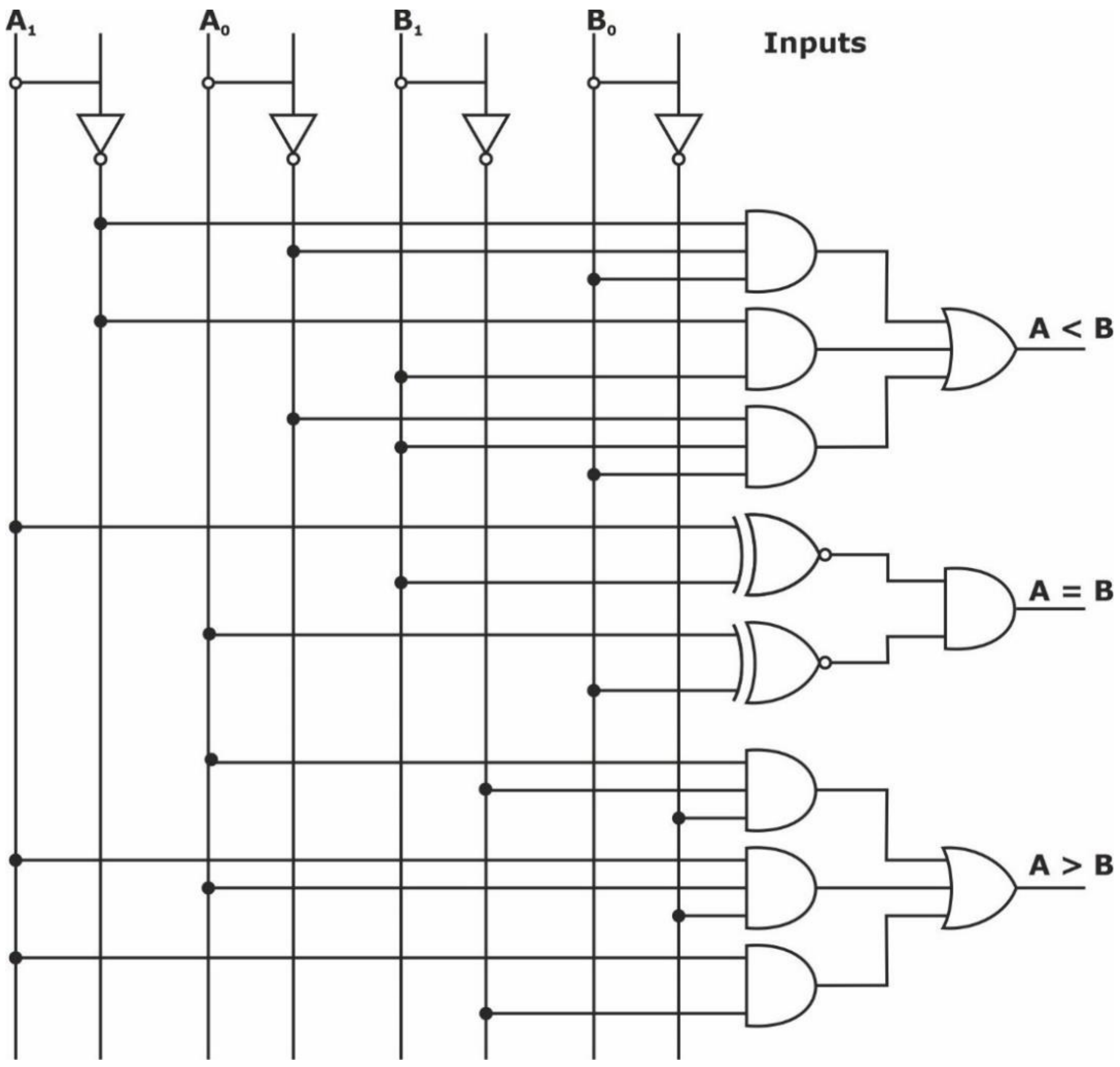


Fig. 13

CODE CONVERTERS:

A code converter is a combinational logic circuit which accepts the input information in one binary code, converts it and produces an output into another binary code.

The truth table for 4-bit Binary and its Equivalent BCD:

Decimal	Binary Input				BCD Output				
	A	B	C	D	B_4	B_3	B_2	B_1	B_0
0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	1
2	0	0	1	0	0	0	0	1	0
3	0	0	1	1	0	0	0	1	1

4	0	1	0	0	0	0	1	0	0
5	0	1	0	1	0	0	1	0	1
6	0	1	1	0	0	0	1	1	0
7	0	1	1	1	0	0	1	1	1
8	1	0	0	0	0	1	0	0	0
9	1	0	0	1	0	1	0	0	1
10	1	0	1	0	1	0	0	0	0
11	1	0	1	1	1	0	0	1	0
12	1	1	0	1	1	0	0	1	0
13	1	1	0	1	1	0	0	1	1
14	1	1	1	0	1	0	1	0	0
15	1	1	1	1	1	0	1	0	1

Table 11: Truth table for 4-bit Binary and its Equivalent BCD

The minimized expression of outputs are as follows:

$B_1 = \overline{A}C + AB\overline{C}$

$B_2 = \overline{A}B + BC$

$B_3 = A\overline{B}\overline{C}$

$B_4 = AB + AC$

$B_0 = D$

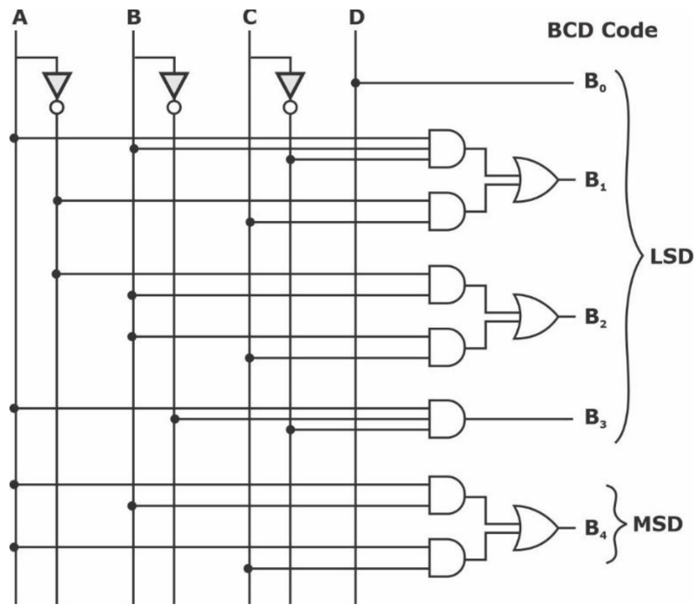


Fig. 14

Parity Generator:

Parity generators are circuits that accept an (n-1) bit data stream and generate an extra bit that is transmitted with the bit stream. This extra bit is referred to as the parity bit. The parity added in binary message is such that the total number of 1's in the message can be either odd or even according to the type of parity used.

Even Parity Generator:

The parity bit is '1' if there are odd number of 1's in the data stream and the parity bit is '0' if there are even number of 1's in the data stream.

The minimized expression for even parity generator is

$P = A \oplus B \oplus C \oplus D$

The logic diagram for the even parity generator is given as

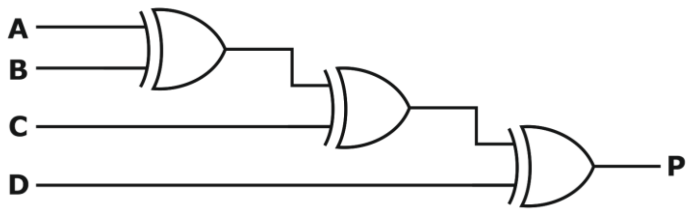


Fig. 15

Odd Parity Generator:

The parity bit is '0' for odd number of 1's and '1' for even number of 1's in the bit stream.

The minimized expression for odd parity generator is

$P = (A \oplus C) \odot (B \oplus D)$

The logic diagram of odd parity generator is given as

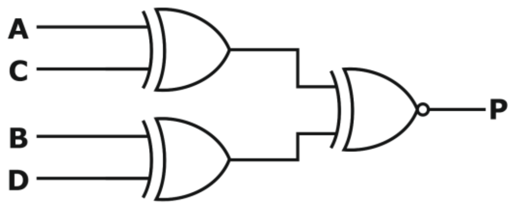


Fig. 16

Sequential Circuits

Sequential circuit include memory elements to store past data.

The flip-flop is a basic element of sequential logic circuits.

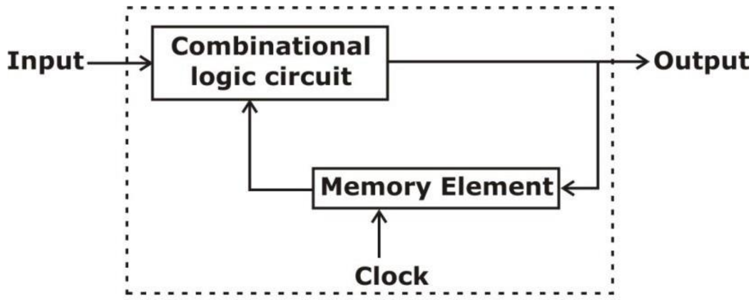


Fig. 17

There are two types of sequential circuits:

1. Synchronous Sequential Circuits
2. Asynchronous Sequential Circuits

DIFFERNCE BETWEEN SYNCHRONOUS AND ASYNCHRONUS SEQUENTIAL CIRCUITS:

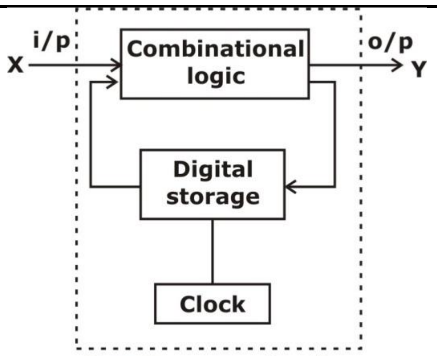
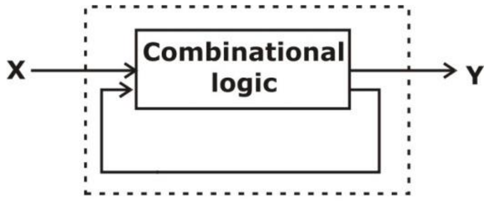
S.No.	Synchronous Sequential Circuits	Asynchronous Sequential Circuits
1.	In synchronous circuits, the change in input signals can affect memory elements upon activation of clock signal.	In asynchronous circuits, change in input signals can affect memory elements at any instant of time.
2.	In synchronous circuits, memory elements are clocked FF's.	In asynchronous circuits, memory elements are either unclocked FF's or time delay elements.
3.	The maximum operating speed of the clock depends on time delays involved.	Since the clock is not present, asynchronous circuits can operate faster than synchronous circuits.
4.	They are easier to design.	More difficult to design.
5.	 i/p X Combinational logic o/p Y Digital storage Clock Synchronous circuit	 Asynchronous circuit

Table 12: Asynchronous and Synchronous Circuit

Difference between Latches and Flip-flops:

S.No.	Latch	Flip-flop

1.	A latch is an electronic sequential logic circuit used to store information in an asynchronous arrangement.	A flip-flop is an electronic sequential logic circuit used to store information in a synchronous arrangement. It has two stable states and maintains its states for an indefinite period until a clock pulse is applied.
2.	One latch can store one-bit information, but output state changes only in response to data input.	One flip-flop can store one-bit data, but output state changes with clock pulse only.
3.	Latch is an asynchronous device and it has no clock input.	Flip-flop has clock input and its output is synchronised with clock pulse.
4.	Latch holds a bit value and it remains constant until new inputs force it to change.	Flip-flop holds a bit value and it remains constant until a clock pulse is received.
5.	Latches are level-sensitive, and the output tracks the input when the level is high. Therefore, as long as the level is logic level 1, the output can change if the input changes.	Flip-flops are edge sensitive. They can store the input only when there is either a rising or falling edge of the clock.

Table 13: Latches and Flip Flops

RS Latch:

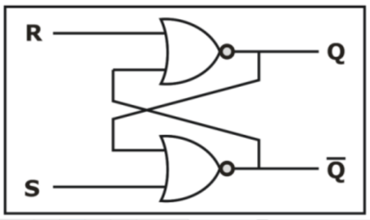


Fig. 18

S R	$Q' \overline{Q'}$	Comment
0 0	$Q \overline{Q}$	Hold
0 1	0 1	Reset
1 0	1 0	Set
1 1		Illegal

Truth Table

Where Q' is the next state and Q is the current state.

JK Flip Flop:

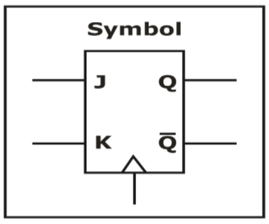


Fig. 19

J K	Q' $\overline{Q'}$	Comment
0 0	Q $\overline{Q}$	Hold
0 1	0 1	Reset
1 0	1 0	Set
1 1	$\overline{Q}$ Q	Toggle

Truth Table

T – Flip Flop:

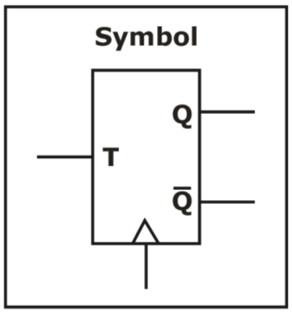
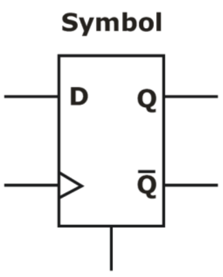


Fig. 20

T	Q' $\overline{Q'}$	Comment
0	Q $\overline{Q}$	Hold
1	$\overline{Q}$ Q	Reset

Truth Table

D – Flip Flop:



Truth Table

clk	D	Q	$\overline{Q}$
0	0	Q	$\overline{Q}$
0	1	Q	$\overline{Q}$
1	0	0	1
1	1	1	0

Fig. 21

Excitation tables:

QQ′	S R	QQ′	J K	QQ′	D	QQ′	T
0 0	0 X	0 0	0 X	0 0	0	0 0	0
0 1	1 0	0 1	1 X	0 1	1	0 1	1
1 0	0 1	1 0	X 1	1 0	0	1 0	1
1 1	X 0	X 0	X 0	1 1	1	1 1	0

- Q(n + 1) = $S + R'Q$ = D = $JQ' + K'Q = TQ' + T'Q$
- For ring counter total no. of states = n
- For twisted ring counter, total no. of states = "2n" (Johnson counter/switch tail ring counter).
- Master-slave circuit is used to counter the Race-around condition in JK flip flop. In Master slave Flip Flop, master is level triggered and slave is edge triggered.

Propagation Delay Time:

Propagation delay time is the time interval required after an input signal has been applied for the resulting output change to occur.

Set-up time (t_s):

It is the minimum time interval required for the logic levels (0 or 1) to be maintained constantly on the inputs (J, K or D) prior to the triggering edge of the clock pulse for the levels to be reliably clocked into the flip-flop.

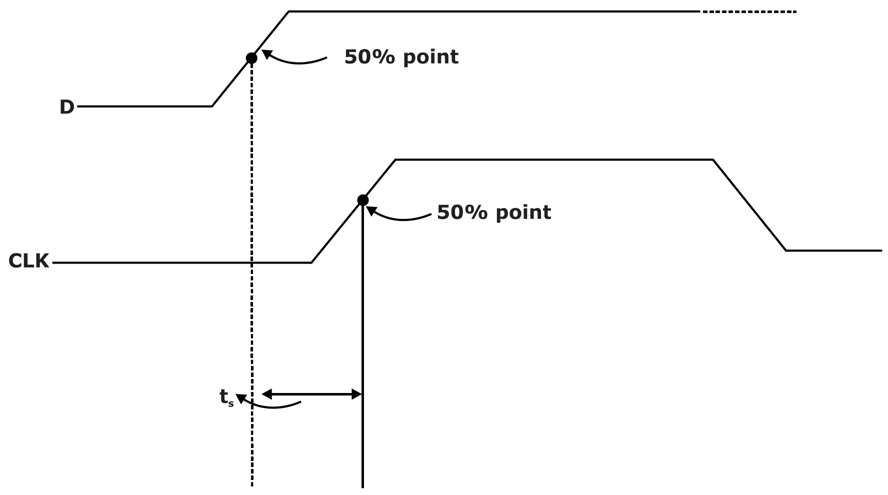


Fig. 22

Hold time (t_h):

It is the time for which the data must remain stable after the triggering edge of the clock.

Clock-pulse width:

The minimum time duration for which the clock pulse must remain HIGH and LOW which are designed by manufacturers. Failure to clock pulse width results in unreliable triggering.

Maximum clock frequency:

The maximum clock frequency (f_{max}) is the highest rate at which flip-flop can be reliably operated.

Designing of one Flip Flop by other flip flop

The steps for designing of one flip flop or new flip flop using existing or same existing flip flop.

Step 1: Write the characteristic table for the designed flip flop.

Step 2: Write the excitation table for the available flip flop.

Step 3: Write the logical expression.

Step 4: Minimize the logical expression.

Step 5: Circuit Implementation.

Shift Registers:

SISO (serial-in, serial-out) Shift Register

- It is the slowest shift register among all the shift registers.
- To store n-bits in a n-bit SISO register, then minimum "n" clock pulses are required.
- To retrieve n-bits from a n-bit SISO register, then minimum "(n-1)" clock pulses are required.

SIPO (serial-in, parallel-out) Shift Register

- To store n-bits in a n-bit SIPO register, minimum "n" clock pulses are required.
- To retrieve n-bits from a n-bit SIPO register, there is no pulse required.

PISO (parallel-in, serial out) Shift Register

- To store n-bit in a n-bit PISO register, a single clock pulse is required.
- To retrieve n-bit from n-bit PISO register, minimum "(n-1)" clock pulses are required.

PIPO (parallel in, parallel out) Shift Register

- To store n-bit in n-bit PIPO register, only a single clock pulse is required.
- To retrieve n-bits from n-bit PIPO register, no clock pulse is required.

Universal shift register

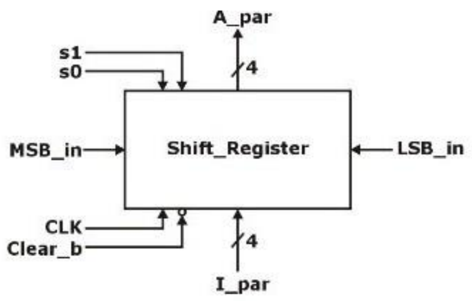


Fig. 23

The function for the Universal Shift Register is as follows:

Mode Control		Register Operation
S_1	S_0	
0	0	No change
0	1	Shift right
1	0	Shift left
1	1	Parallel load

Table 14: Truth Table of Universal Shift Register

Applications of shift registers

(a) Delay line: A shift register can be used to introduce a delay (Δt) in signals

$$\boxed{\Delta t = N \times \frac{1}{f_c}}$$

Where N is number of stages & f_c is the clock frequency.

(b) Serial-to-parallel converter

(c) Parallel-to-serial converter

(d) Ring counter

(e) Twisted ring counter

(f) Sequence counter

Counters:

S.No.	Asynchronous Counter	Synchronous Counter
1.	Clock input is applied to LSB FF. The output of first FF is connected as clock to next FF.	Clock input is common to all FF.
2.	All Flip-Flop are toggle FF.	Any FF can be used.
3.	Speed depends on number of FF used for n bit. $f_{max} = 1/(n*t_n)$	Speed is independent of number of FF used. $f_{max} = 1/t_p$
4.	No extra logic Gates are required.	Logic Gates required based on design.
5.	Cost is less.	Cost is more.

Table 15: Difference between Asynchronous(ripple) and Synchronous Counter

<u>ADC & DAC</u>

Analog to Digital converter (ADC):

- Resolution = $\frac{FSV}{2^n - 1}$ where FSV means Full Scale Value
- Quantization error = $\frac{V_R}{2^n}\%$
- Flash type ADC: $2^{n-1} \rightarrow$ comparators

  $2^n \rightarrow$ resistors

  $2^n \times n \rightarrow$ Encoder

  $$T_C = \frac{2^n}{2} \times \frac{1}{f_{clock}} = 2^{n-1} \times \frac{1}{f_{clock}}$$

  Fastest ADC:
- Successive approximation ADC: n clock pulses.
- Counter type ADC: $2^n - 1$ clock pulses
- Dual slope integrating type: 2^{n+1} clock pulses.

Digital to Analog Converter (DAC):

- $FSV = V_R\left(1 - \frac{1}{2^n}\right)$
- Resolution = $\frac{\text{Step size}}{FSV} = \frac{V_R / 2^n}{V_R\left(1 - \frac{1}{2^n}\right)} = \frac{1}{2^n - 1} \times 100\%$
- Accuracy = $\pm\frac{1}{2}LSB = \pm\frac{1}{2^{n+1}}$
- Analog output = K. digital output

<u>**FSM & Memory**</u>

Finite State Machines (FSM):

A general block diagram of clocked sequential circuit is also known as finite stable machine (FSM). Depending upon the external outputs, there are two types of models of sequential circuits.

1. Mealy Model
2. Moore Model

Mealy model:

In mealy model, the next state of the function depends on present state as well as present inputs.

Moore Model:

In moore model, the next state depends on the present state and present inputs but also on the outputs of more model.

The block diagram of a Moore model is given as

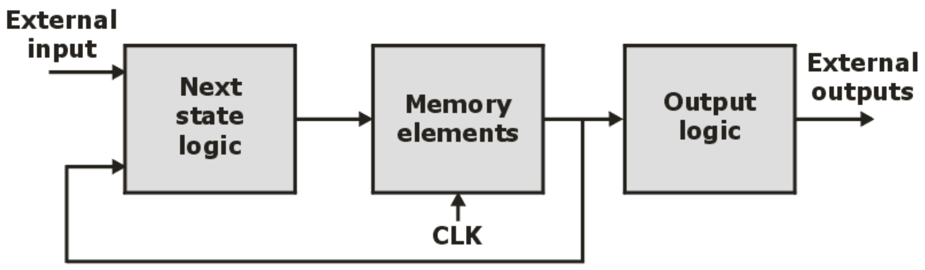


Fig. 24

MEMORY:

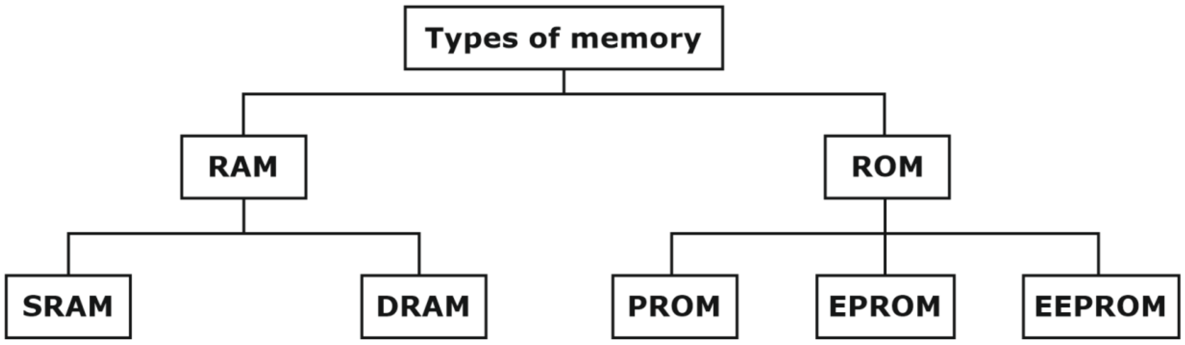


Fig. 25

RAM:

Below table lists some of the differences between SRAM and DRAM:

SRAM	DRAM

1. SRAM has lower access time, so it is faster compared to DRAM.	1. DRAM has higher access time, so it is slower than SRAM.
2. SRAM is costlier than DRAM,	2. DRAM costs less compared to SRAM.
3. SRAM requires constant power supply, which means this type of memory consumes more power.	3. DRAM offers reduced power consumption, due to the fact that the information is stored in the capacitor.
4. Due to complex internal circuitry, less storage capacity is available compared to the same physical size of DRAM memory chip.	4. Due to the small internal circuitry in the one-bit memory cell of DRAM, the large storage capacity is available.
5. SRAM has low packaging density.	5. DRAM has high packaging density.

Table 16: Difference between SRAM and DRAM

ROM:

The required paths in a ROM may be programmed in four different ways.

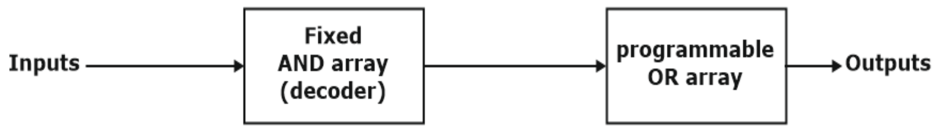
Mask programming: fabrication process

Read-only memory or ROM: blown fuse/fuse intact

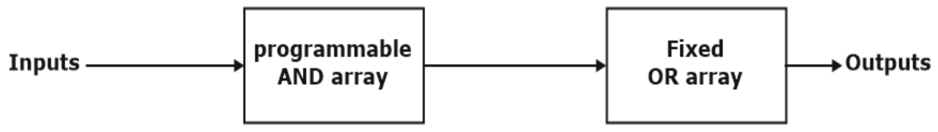
Erasable PROM or EPROM: placed under a special ultraviolet light for a given period will erase the pattern in ROM.

Electrically erasable PROM(EEPROM): erased with an electrical signal instead of ultraviolet light.

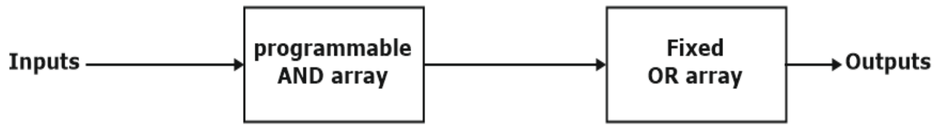
PROM, PAL & PLA:



(a) Programmable read-only memory (PROM)



(b) Programmable array logic (PAL)



(b) Programmable logic array (PLA)

Fig. 26

AND	OR	
Fixed	Programmable	PROM
Programmable	Fixed	PAL

Programmable	Programmable	PLA

Table 17: AND & OR Gates in PROM, PAL and PLA

Logic Family

- Figure of Merit, $\boxed{FOM = T_{PD} \times P_D}$

  where, T_{PD} is the average transition delay time for the signal to propagate from input to output of logic gate.

  and, P_D is the power drawn from power supply.

- Fan out of a logic gate = $\frac{I_{OH}}{I_{IH}}$ or $\frac{I_{OL}}{I_{IL}}$

- Noise margin: $NM_H = V_{OH} - V_{IH}$ or $NM_L = V_{OL} - V_{IL}$

  $\boxed{V_{OH} > V_{IH} > V_{IL} > V_{OL}}$

  $\boxed{\text{Noise-Margin} = \min(NM_H, NM_L)}$

- Logic swing: $V_{OH} - V_{OL}$
- Power Dissipation,

  $P_D = V_{CC}\, I_{CC} = V_{cc}\left[\frac{1 + \_I}{2}\right]$

  $I \to I_C$ when output low.
  $I \to I_C$ when output high.

- TTL , ECL & CMOS are used for MSI or SSI.
- I^2L has best FOM.
- RTL , DTL , TTL → saturated logic

  ECL → Unsaturated logic
- Advantages of Active pullup; increased speed of operation, less power consumption.
- For TTL floating input considered as logic "1" & for ECL it is logic "0".
- "MOS" mainly used for LSI & VLSI. Fan out is too high.
- ECL is fastest gate & consumes more power.
- CMOS is slowest gate & less power consumption.
- NMOS is faster than CMOS.
- Gates with open collector output can be used for wired AND operation (TTL).
- Gates with open emitter output can be used for wired OR operation (ECL).
- 1024 × 8 memory can be obtained by using 1024 × 2 memories.
- Number of memory ICs of capacity 1k × 4 required to construct memory of capacity 8k × 8 are "16"