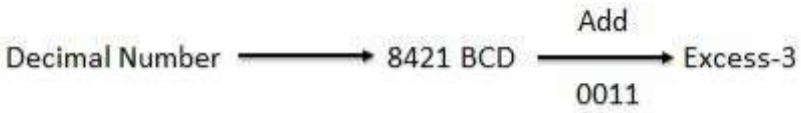
DELD

Unit-2

Combinational Logic Design

2.1 Code converter: BCD, Excess-3, Gray code, Binary Code

- **Excess-3 code**
- It is also known as the XS-3 code.
- It is a non-weighted code used to express decimal numbers.
- They are derived from the 8421 BCD code words adding (0011)2 or (3)10 to each codeword in 8421.
- The excess-3 codes are obtained as –



Example

Decimal	BCD 8 4 2 1	Excess-3 BCD + 0011
0	0 0 0 0	0 0 1 1
1	0 0 0 1	0 1 0 0
2	0 0 1 0	0 1 0 1
3	0 0 1 1	0 1 1 0
4	0 1 0 0	0 1 1 1
5	0 1 0 1	1 0 0 0
6	0 1 1 0	1 0 0 1
7	0 1 1 1	1 0 1 0
8	1 0 0 0	1 0 1 1
9	1 0 0 1	1 1 0 0

Fig.1: BCD to XS 3 conversion (Ref. 1)

Gray Code

- It is the non-weighted code and is not an arithmetic code.
- This means that there are no specific weights assigned to the bit position.

- Here only one bit will change every time the decimal number is incremented.
- The gray code is also known as a unit distance code as only one-bit changes at a time.
- The gray code is a type of cyclic code.
- It cannot be used for all arithmetic operations.

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

Fig.2: Gray codes (Ref. 1)

Application of Gray code

- They are used in the shaft position encoders.
- This encoder produces a code word which represents the angular position of the shaft.

Binary Coded Decimal (BCD) code

- Here each decimal digit is represented by a 4-bit binary number.
- It's a way to express each of the decimal digits with a binary code.
- Therefore, by four bits we can represent sixteen numbers (0000 to 1111).
- But in BCD code only the first ten of these numbers are used (0000 to 1001) and rest are invalid.

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

Fig.3: BCD codes (Ref. 1)

Advantages of BCD Codes

- It is very similar to the decimal system.
- We have to remember the binary equivalent of 0 to 9 only.

Disadvantages of BCD Codes

- The addition and subtraction of the BCD number system have different rules.
- The BCD arithmetic is more complicated.
- BCD code requires more number of bits than binary code to represent the decimal number.
- Hence is less efficient than binary.

Binary Code

- In coding, when alpha-numeric characters or words are represented by a specific group of symbols, it is said that it is being coded.
- The group of symbols is known as a code.
- The digital data can be represented, stored, and transmitted as a group of binary bits.
- This group is called a **binary code**.
- It is represented by the number as well as the alphanumeric character.

Advantages of Binary Code

- They are used in computer applications.
- They are suitable for digital communications.
- They make the analysis and designing of digital circuits easy.
- Implementation becomes very easy since only 0 & 1 are being used.

Classification of binary codes

The codes are broadly classified as:

- Weighted Codes
- Non-Weighted Codes
- Binary Coded Decimal Code
- Alphanumeric Codes
- Error Detecting Codes
- Error-Correcting Codes

Weighted Codes

- These codes obey the positional weight principle.
- Here each position represents a specific weight.
- Several systems are used to express the decimal digits 0 through 9.
- Here, each decimal digit is represented by a set of four bits.

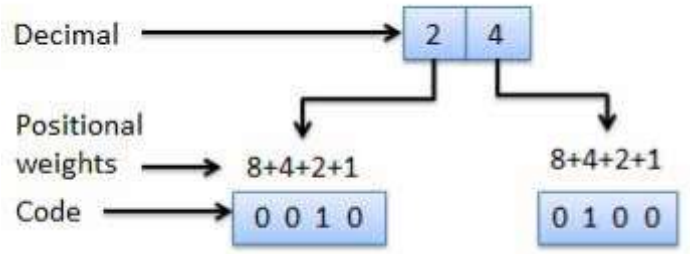


Fig.4: Weighted codes (Ref. 1)

Non-Weighted Codes

- Here, the positional weights are not assigned.
- Example: Excess-3 code and Gray code.

Alphanumeric codes

- A binary digit orbit can be represented by two symbols as '0' or '1'.
- This is not sufficient for communication between two computers as we need many more symbols for communication.
- These symbols represent 26 alphabetic characters with capital and small letters, numbers from 0 to 9, punctuation marks, and other symbols.
- These alphanumeric codes represent numbers and alphabetic characters.
- Most of them also represent other characters like symbols and various instructions necessary for conveying information.
- The code at least represents 10 digits and 26 letters of alphabet i.e. total of 36 items.
- The following three types of alphanumeric codes are commonly used for data representation.
  - American Standard Code for Information Interchange (ASCII).
  - Extended Binary Coded Decimal Interchange Code (EBCDIC).
  - Five bitBaudot Code.

- ASCII code is a 7-bit code whereas EBCDIC is an 8-bit code.
- ASCII code is used worldwide while EBCDIC is primarily used in large IBM computers.

Error Codes

- Their technique is available to detect and correct data during data transmission.

Error Code	Description
Error Detection and Correction	Error detection and correction code techniques

2.2 Half- Adder, Full Adder, Half Subtractor, Full Subtractor, Binary Adder (IC 7483), BCD adder

- **Half Adder**

- It is a combinational circuit that has two inputs and two outputs.
- It is designed to add two single bit binary numbers A and B.
- It has two outputs **carry** and **sum**.

Block diagram

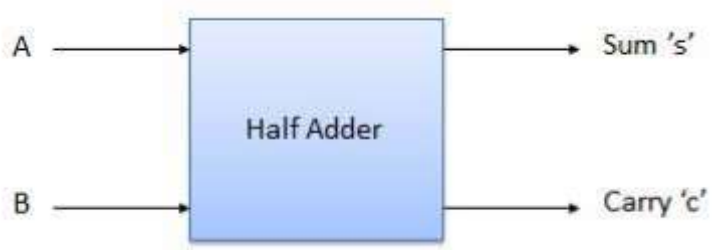


Fig. : A half adder (ref. 2)

Truth Table

Inputs		Output	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Circuit Diagram

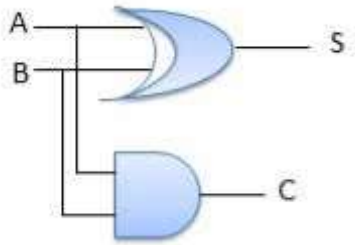


Fig.: A half adder (ref. 2)

B. Full Adder

- It is developed to overcome the drawback of the Half Adder circuit.
- It can add two one-bit numbers A and B and a carry C.
- It is a three-input and two output combinational circuit.

Block diagram

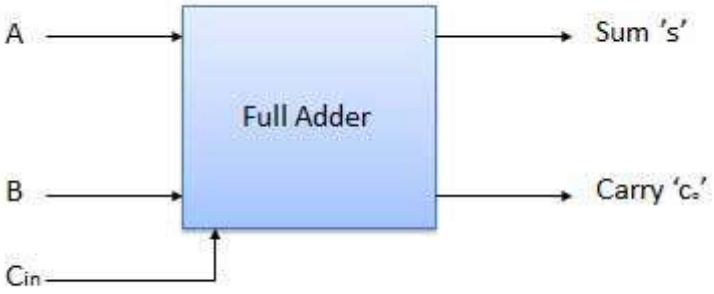


Fig. : Full adder (ref. 2)

Truth Table

Inputs			Output	
A	B	Cin	S	Co
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Circuit Diagram

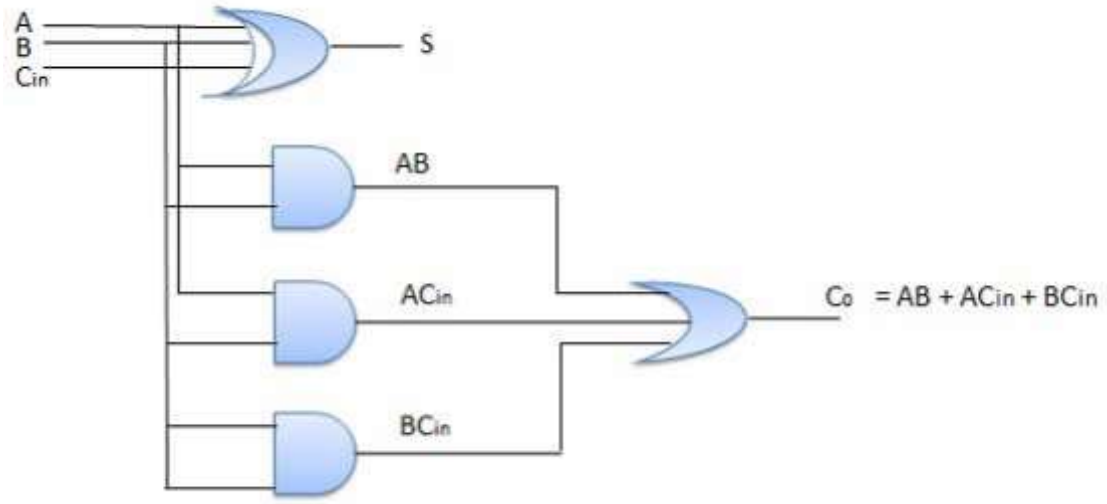


Fig.: Full adder (ref. 2)

C. Half Subtractors

- It is a combination circuit with two inputs and two outputs.
- The difference between the two binary bits is obtained at the output and an output (Borrow) indicates if a 1 has been borrowed.
- Here A is called Minuend bit and B is called a Subtrahend bit.

Truth Table

Inputs		Output	
A	B	(A – B)	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Circuit Diagram

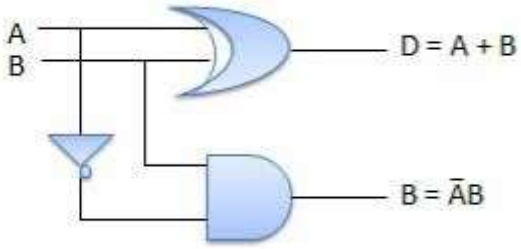


Fig.: Half subtractor (ref. 2)

D. Full Subtractors

- It is a combinational circuit which has three inputs A, B, C, and two output D and C'.
- A is the 'minuend', B is 'subtrahend', C is the 'borrow' which is produced by the previous stage, difference output D and C' is the borrow output.

Truth Table

Inputs			Output	
A	B	C	(A-B-C)	C'
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

Circuit Diagram

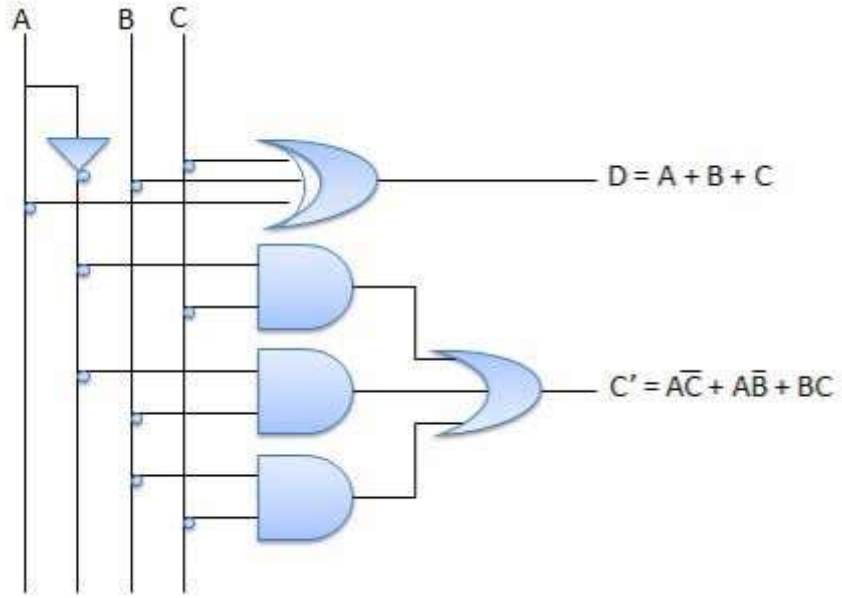


Fig.: Full subtractor (ref. 2)

Binary Adder (IC 7483)

F. BCD Adder

- BCD stands for binary coded decimal.
- Suppose, we have two 4-bit numbers A and B. The value of A and B can vary from 0(0000 in binary) to 9(1001 in binary).

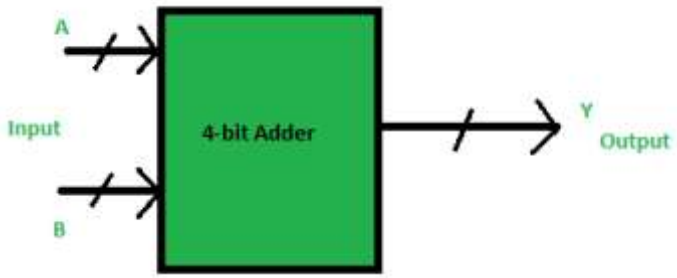


Fig. : BCD adder (ref. 2)

- The output varies from 0 to 18 if the carry from the previous sum is not considered.
- But if we consider the carry, then the maximum value of output will be 19 (i.e. 9+9+1 = 19).
- When we simply add A and B, then we get the binary sum, and to get the output in BCD form, we use BCD Adder.

Example 1:

Input :

A = 0111 B = 1000

Output :

Y = 1 0101

Explanation: We are adding A(=7) and B(=8).

The value of the binary sum will be 1111(=15).

But, the BCD sum will be 1 0101,

where 1 is 0001 in binary and 5 is 0101 in binary.

Example 2:

Input :

A = 0101 B = 1001

Output :

Y = 1 0100

Explanation: We are adding A(=5) and B(=9).

The value of the binary sum will be 1110(=14).

But, the BCD sum will be 1 0100,

where 1 is 0001 in binary and 4 is 0100 in binary.

Now, let's move to the table and find out the logic when we are going to add “0110”.

Decimal	Binary Sum					BCD Sum				
	C'	S3'	S2'	S1'	S0'	C	S3	S2	S1	S0
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	1
2	0	0	0	1	0	0	0	0	1	0
3	0	0	0	1	1	0	0	0	1	1
4	0	0	1	0	0	0	0	1	0	0
5	0	0	1	0	1	0	0	1	0	1
6	0	0	1	1	0	0	0	1	1	0
7	0	0	1	1	1	0	0	1	1	1
8	0	1	0	0	0	0	1	0	0	0
9	0	1	0	0	1	0	1	0	0	1
10	0	1	0	1	0	1	0	0	0	0
11	0	1	0	1	1	1	0	0	0	1
12	0	1	1	0	0	1	0	0	1	0
13	0	1	1	0	1	1	0	0	1	1
14	0	1	1	1	0	1	0	1	0	0
15	0	1	1	1	1	1	0	1	0	1
16	1	0	0	0	0	1	0	1	1	0
17	1	0	0	0	1	1	0	1	1	1
18	1	0	0	1	0	1	1	0	0	0
19	1	0	0	1	1	1	1	0	0	1

Fig. : Table explaining BCD addition (ref. 2)

We are adding “0110” (=6) only to the second half of the table because of the following conditions:

- If C’ = 1 (Satisfies 16-19)
- If S3′.S2′ = 1 (Satisfies 12-15)
- If S3′.S1′ = 1 (Satisfies 10 and 11)

So, our logic is

C' + S3'.S2' + S3'.S1' = 1

Implementation :

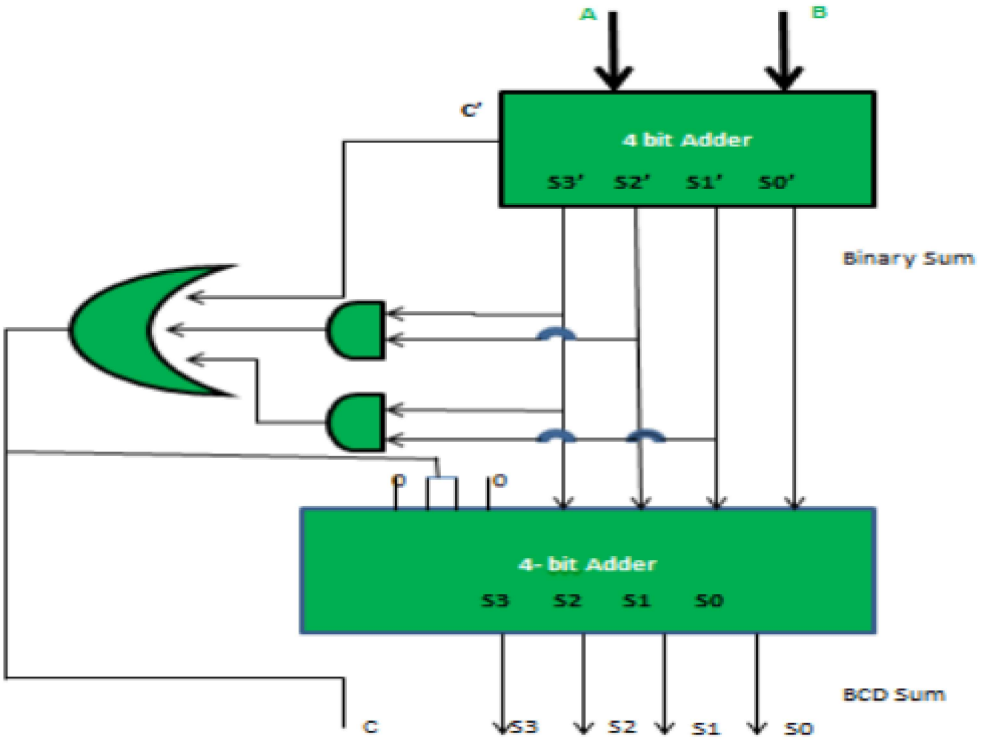
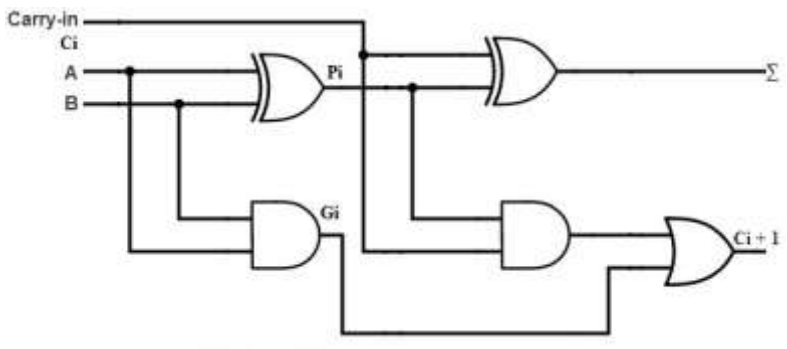


Fig. : BCD adder (ref. 2)

2.3 Look ahead carry generator

A carry-Lookahead adder is a fast parallel adder as it reduces the propagation delay by more complex hardware, hence it is costlier. In this design, the carry logic over fixed groups of bits of the adder is reduced to two-level logic, which is nothing but a transformation of the ripple carry design.

This method makes use of logic gates so as to look at the lower order bits of the augend and addend to see whether a higher order carry is to be generated or not.



A	B	Ci	C i +1	Condition
0	0	0	0	No carry generate
0	0	1	0	
0	1	0	0	
0	1	1	1	No carry propagate
1	0	0	0	
1	0	1	1	
1	1	0	1	Carry generate
1	1	1	1	

Consider the full adder circuit shown above with corresponding truth table. If we define two variables as carry generate Gi and carry propagate Pi then,

$P_i = A_i \oplus B_i$

Gi = Ai Bi

The sum output and carry output can be expressed as

Si = Pi ⊕ Ci

C i +1 = Gi + Pi Ci

Where Gi is a carry generate which produces the carry when both Ai, Bi are one regardless of the input carry. Pi is a carry propagate and it is associate with the propagation of carry from Ci to Ci +1.

The carry output Boolean function of each stage in a 4 stage carry-Lookahead adder can be expressed as

C1 = G0 + P0 Cin

C2 = G1 + P1 C1

= G1 + P1 G0 + P1 P0 Cin

C3 = G2 + P2 C2

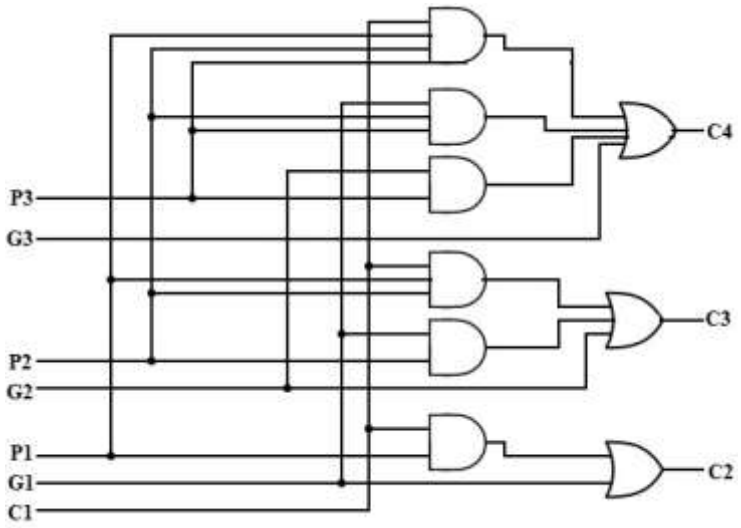
= G2 + P2 G1+ P2 P1 G0 + P2 P1 P0 Cin

C4 = G3 + P3 C3

= G3 + P3 G2+ P3 P2 G1 + P3 P2 P1 G0 + P3 P2 P1 P0 Cin

From the above Boolean equations we can observe that C4 does not have to wait for C3 and C2 to propagate but actually C4 is propagated at the same time as C3 and C2. Since the Boolean expression for each carry output is the sum of products so these can be implemented with one level of AND gates followed by an OR gate.

The implementation of three Boolean functions for each carry output (C2, C3 and C4) for a carry-Lookahead carry generator shown in below figure.



2.4 Multiplexers (MUX): MUX (IC 74153, 74151)

- It is a special type of combinational circuit.
- It has n-data inputs, one output, and m inputs select lines with $2^m = n$.
- It selects one of the n data inputs and routes it to the output.
- The selection of one of the inputs is done by the select lines.
- Depending on the code applied at the inputs, one of the n data sources is selected and transmitted to the single output Y.
- E is the enable input which is useful for cascading purposes.
- It is an active low terminal hence performs the required operation when it is low.

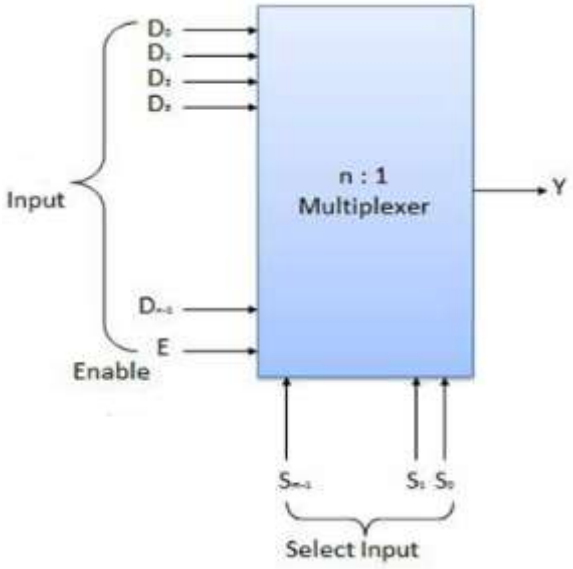
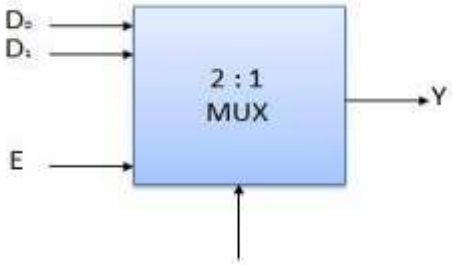


Fig. : Block diagram of the multiplexer (ref. 2)

Multiplexers come in multiple variations

- 2: 1 multiplexer
- 4: 1 multiplexer
- 16: 1 multiplexer
- 32: 1 multiplexer

Block Diagram of 2:1 MUX



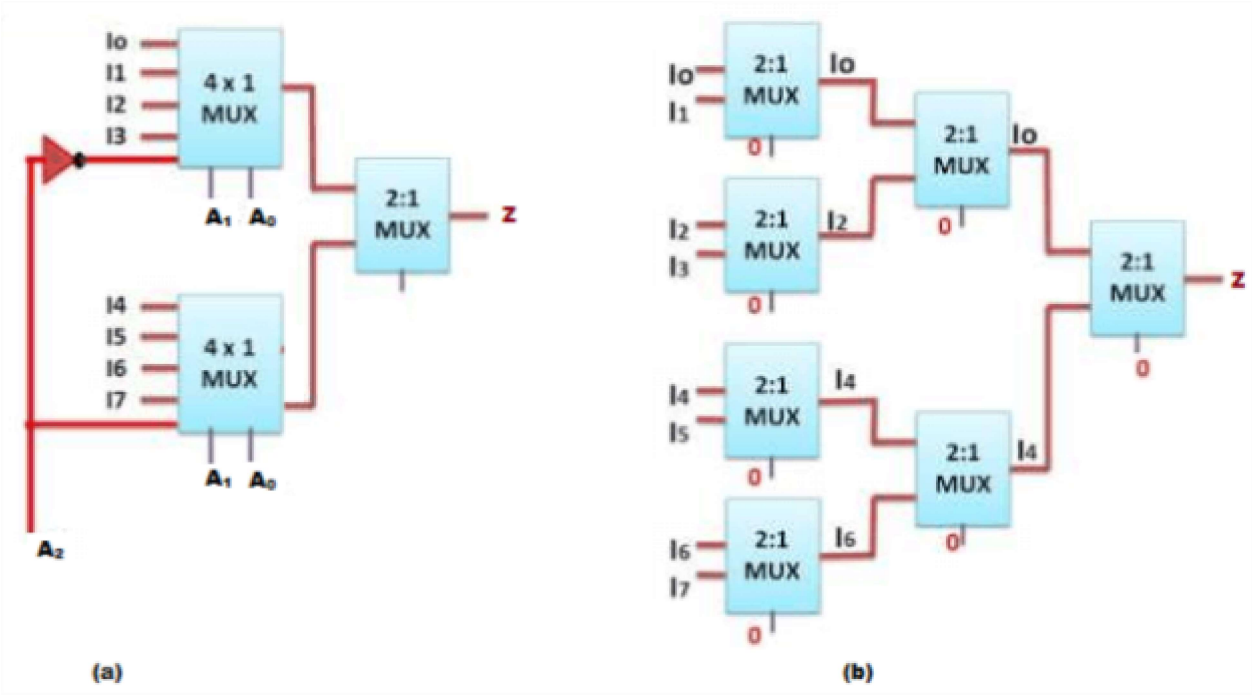
Truth Table of 2:1 MUX

Enable	Select	Output
E	S	Y
0	x	0
1	0	D_0
1	1	D_1

Where x is don’t care.

2.5 Cascading multiplexers

Cascading refers to a process where large Multiplexers can be designed and implemented using smaller Multiplexers. **Example**: 8:1 Mux can be designed using two 4:1 Multiplexers and similarly it can be designed using four 2:1 Multiplexers as shown in the below figure.



2.6 Demultiplexers (DEMUX)- Decoder (IC 74138, IC 74154)

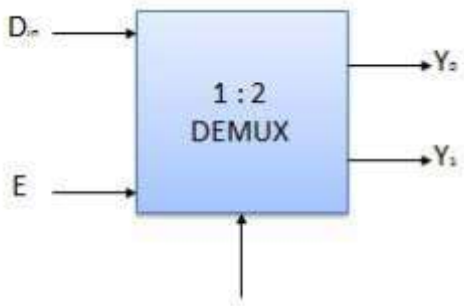
Demultiplexers

- It performs the inverse operation of a multiplexer as it receives one input and distributes it across its outputs.
- It has only one input and n outputs with m select input.
- At a time only one output line is selected by the select lines and that input is transmitted through the output line.
- It is equivalent to a single-pole multiple-way switch.

Various Demultiplexers are used as:

- 1: 2 demultiplexer
- 1: 4 demultiplexer
- 1: 16 demultiplexer
- 1: 32 demultiplexer

Block diagram



Truth Table

Enable	Select	Output
E	S	Y0 Y1
0	x	0 0
1	0	0 D_{in}
1	1	D_{in} 0

Where x is don't care.

2.7 Implementation of SOP and POS using MUX, DMUX, Comparators (2 bit)

MUX

Given SOP function f(A, B, C) = m(0, 1, 4, 6, 7) and MUX is

For 3 variable function, the truth table is

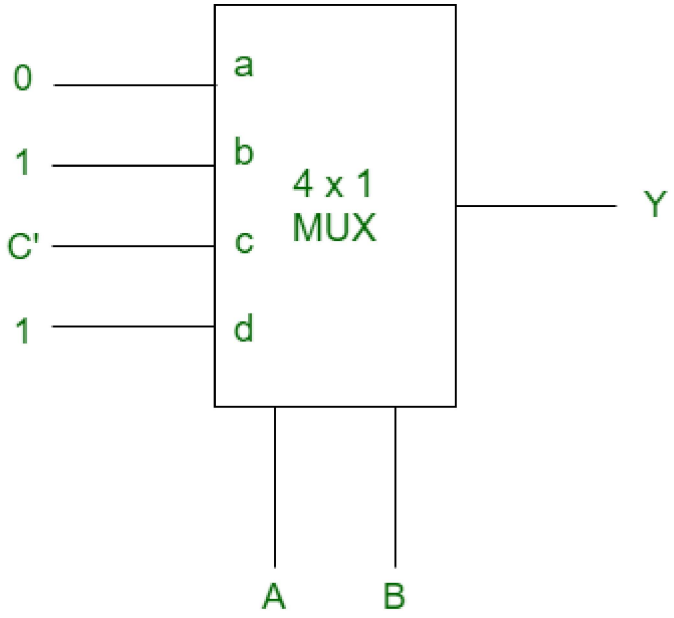
Truth Table

	A	B	C	Y
0	0	0	0	1
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1

Let A and B are the select lines and C be the input,

	a	b	c	d
C'	0	2	4	6
C	1	3	5	7
	1	0	C'	1

Thus, for the implementation of given logical function, required is one 4×1 MUX and and inverter.

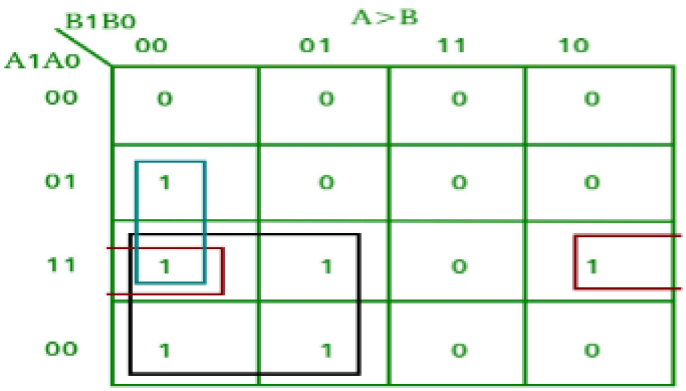


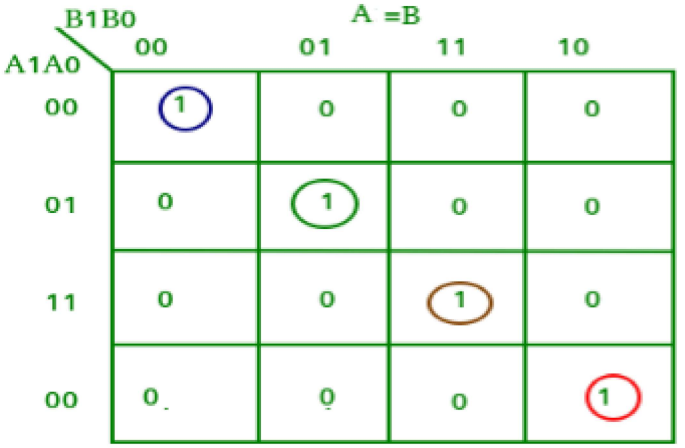
2-Bit Magnitude Comparator

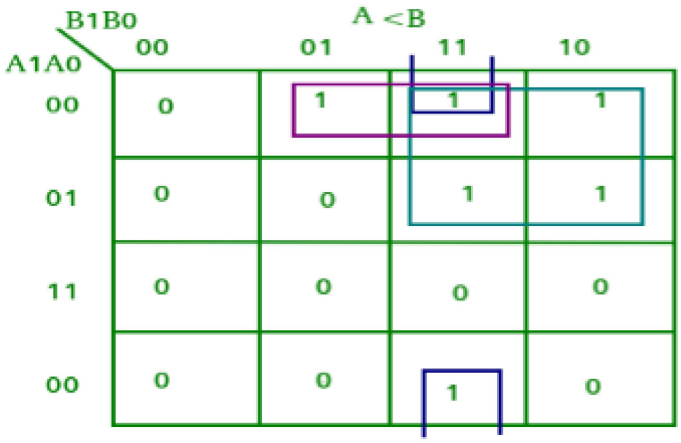
- It is used to compare two binary numbers (two bits each).
- It consists of four inputs A and B which are 2 bit each and three outputs.
- The truth table is given below:

INPUT				OUTPUT		
A1	A0	B1	B0	A<B	A=B	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

K-map for each output is as follows:







From the above K-maps output can be expressed as follows:

A>B:A1B1' + A0B1'B0' + A1A0B0'

A=B: A1'A0'B1'B0' + A1'A0B1'B0 + A1A0B1B0 + A1A0'B1B0'

: A1'B1' (A0'B0' + A0B0) + A1B1 (A0B0 + A0'B0')

: (A0B0 + A0'B0') (A1B1 + A1'B1')

: (A0 Ex-Nor B0) (A1 Ex-Nor B1)

A<B:A1'B1 + A0'B1B0 + A1'A0'B0

A logic circuit for this comparator as given below:

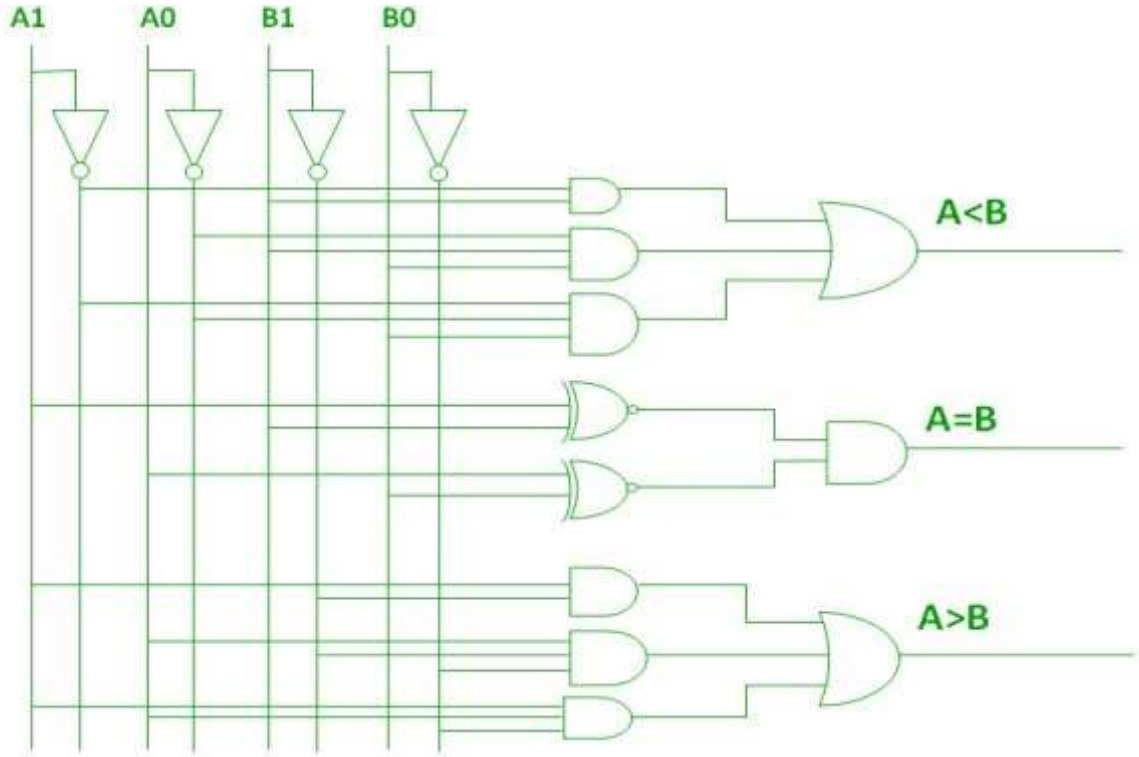


Fig:-bit comparator (ref.2)

2.8 Parity generators and Checker

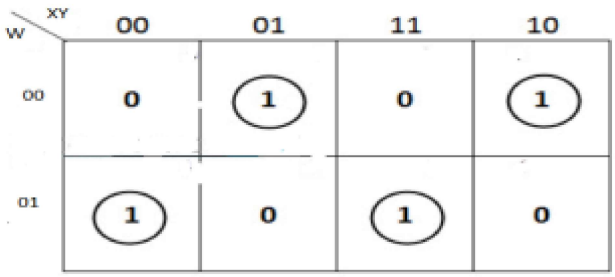
- The parity generating technique is used as error detection technique for the data transmission.
- When binary data is transmitted and processed , it is subjected to noise and that noise can alter 0s to 1s and 1s to 0s of data bits in digital systems.
- Therefore, a parity generator is a combinational logic circuit that generates the parity bit at the transmitter end.
- The basic principle is that sum of odd number of 1s is always 1 and sum of even number of 1s is always zero.
- Such error detection and correction can be implemented by using Ex-OR gates.

Even Parity Generator

A 3-bit message is transmitted with an even parity bit. Hence assuming, the three inputs W,X and Y that are applied to the circuits and output bit is the parity bit P. The total number of 1s must be even, to generate the even parity bit P.

3- bit message			Even Parity
W	X	Y	P
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

The K-map simplification for 3-bit message even parity generator is

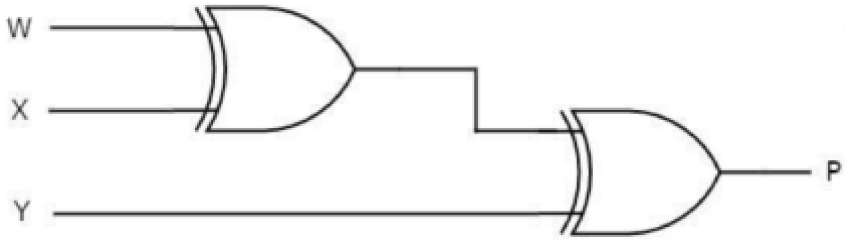


From the above K-Map, the expression is:

P=W′X′Y+W′XY′+WX′Y′+WXY

P=W′(X′Y+XY′)+W(X′Y′+XY)

P=W′(X⊕Y)+W(X⊕Y)′=W⊕X⊕Y

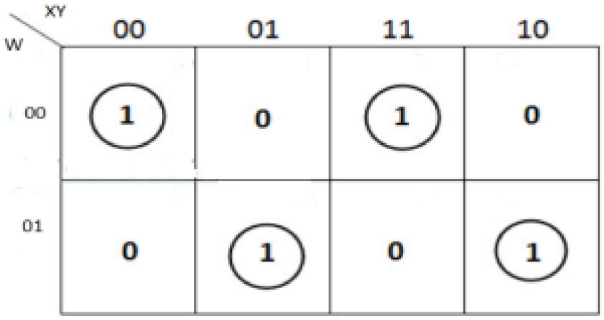


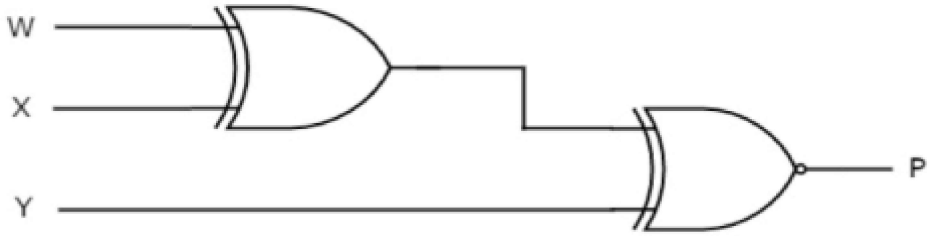
Odd Parity Generator

Considering that the 3-bit data is transmitted with an odd parity bit. The three inputs are W, X and Y and P is the output parity bit. The total number of bits must be odd in order to generate the odd parity bit.

3- bit message			Odd Parity
W	X	Y	P
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

The K-map simplification for 3-bit message even parity generator is





Parity Checker

- They are of two types :
- **Even parity checker: It** checks error in the transmitted data, which contains message bits along with even parity.
- **Odd parity checker: It** checks error in the transmitted data, which contains message bits along with odd parity.

Even parity checker

Assume a 3-bit binary input, W, X and Y is transmitted along with an even parity bit, P. So, the resultant data contains 4 bits, that is received as the input of even parity checker.

It generates an **even parity bit, E**. This bit is zero, if the received data contains an even number of ones, which indicates that there is no error in the received data and vice versa.

The **Truth table** of an even parity checker is:

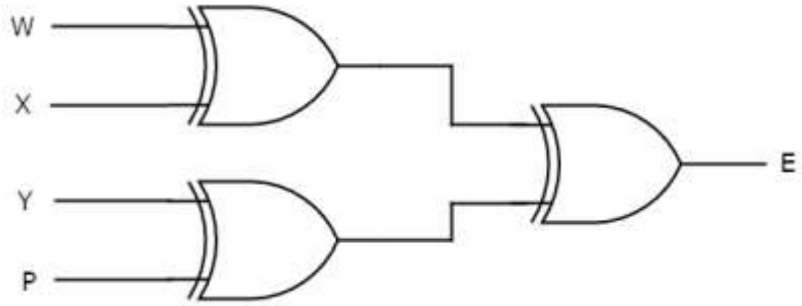
4-bit Received Data WXYP	**Even Parity Check bit E**
0000	0
0001	1
0010	1
0011	0
0100	1
0101	0
0110	0
0111	1
1000	1
1001	0
1010	0

1011	1
1100	0
1101	1
1110	1
1111	0

The **Boolean function** of even parity check bit as

E = W xor X xor Y xor P

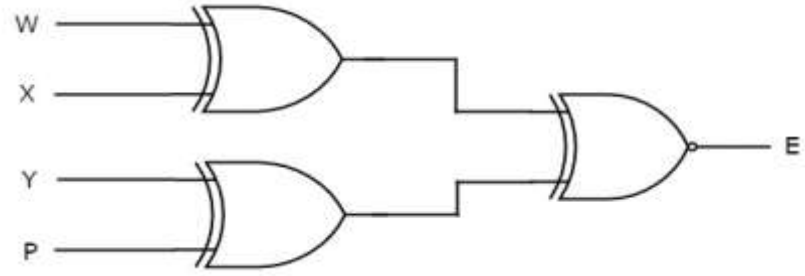
The following figure shows the **circuit diagram** of even parity checker.



Odd Parity Checker

Assume a 3-bit binary input, W, X and Y is transmitted along with an odd parity bit, P. So, the resultant data contains 4 bits, that is received as the input of odd parity checker.

It generates an **odd parity bit, E**. This bit is zero, if the received data contains an odd number of ones, which indicates there is no error in the received data.



Reference Books:

1. John Yarbrough, —Digital Logic Applications and Design, Cengage Learning, ISBN – 13: 978-81-315-0058-3

2. D. Leach, Malvino, Saha, —Digital Principles and Applications‖, Tata McGraw Hill, ISBN – 13:978-0-07-014170-4.

3. Anil Maini, ―Digital Electronics: Principles and Integrated Circuits‖, Wiley India Ltd, ISBN:978-81-265-1466-3.

4. Norman B & Bradley, ―Digital Logic Design Principles, Wiley India Ltd, ISBN:978-81-265-1258