Combinational Circuit vs Sequential Circuit

Combinational Circuit	Sequential Circuit
Present o/p is only depend on present i/p	Present o/p depends on present i/p and previous o/p
No Feedback	Feedback
No Memory	Memory
e.g. Half Adder (HA), FA, MUX, DEMUX	e.g. Flipflop , Register, Counter

Q. 1: A digital system is capable of selling rose, lotus, lily, and tulip flowers. Now, design a combinational logic system that can deliver flowers if a customer request either rose and lily or lotus and tulip flowers.

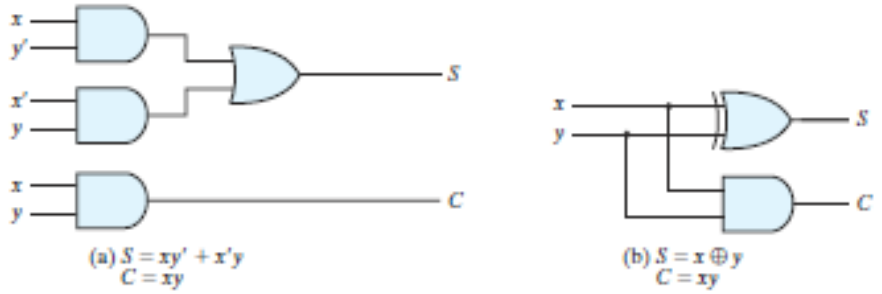
Q. 2: A digital system show weight on some condition after accepting coins in three times. It can accept1Rs, 5Rs and 10Rs coins. Design a combination logic system that deliver weight slip (i). if the sum is atleast 11Rs (ii). The sum should be perfectly divisible by 5.

Combinational Circuit

- Half Adder (HA):

Half Adder

x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



(a) $S = xy' + x'y$, $C = xy$ (b) $S = x \oplus y$, $C = xy$

- A combinational circuit that performs the addition of two bits is called a *half adder* . One that performs the addition of three bits (two significant bits and a previous carry) is a *full adder*

Full Adder

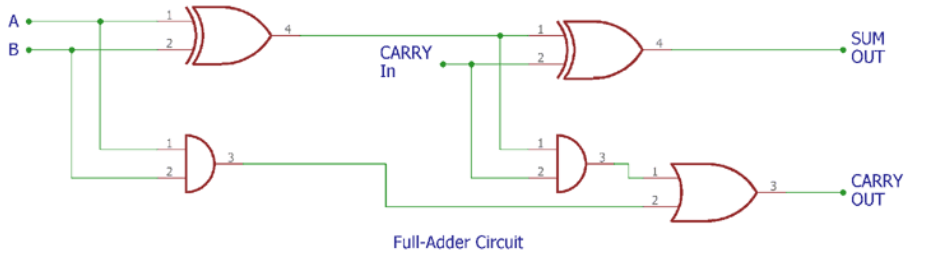
- Full Adder (FA):

- **Full adder truth table**

$S = A \oplus B \oplus Cin$

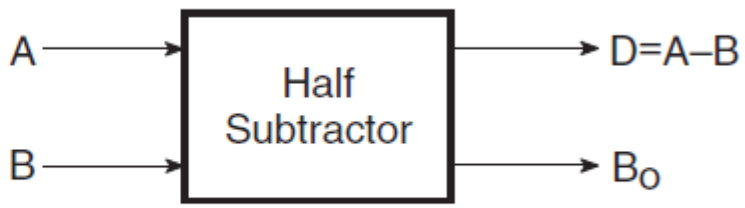
$C = AB + Cin\ (A \oplus B)$

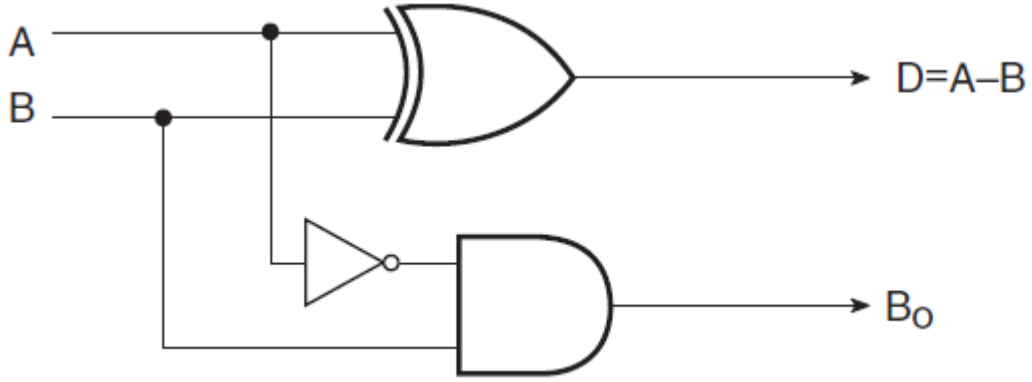
A	B	Carry-In	Sum	Carry-Out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



Half Subtractor

A	B	D	B_O
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0





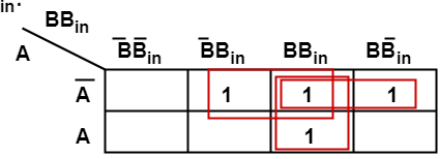
Full Subtractor

INPUT			OUTPUT	
A	B	Bin	D	Bout
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

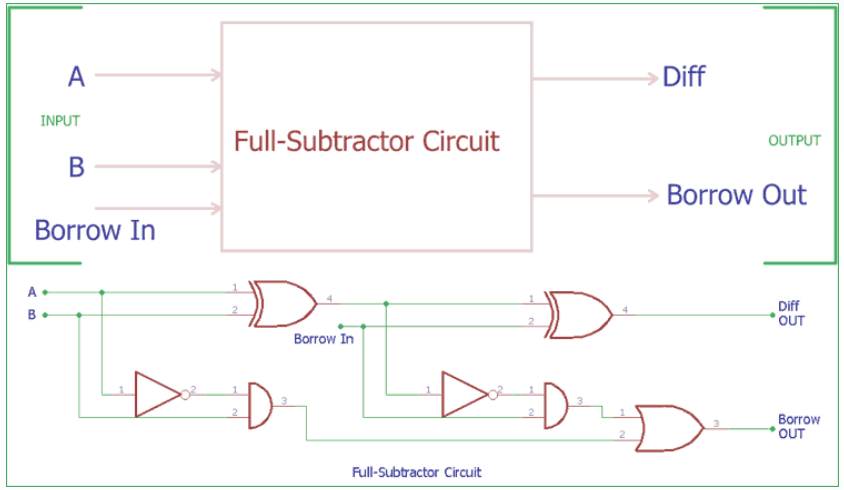
For D:

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

For B_{in}:

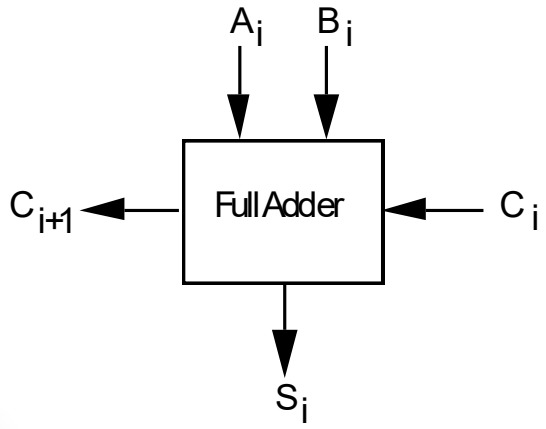


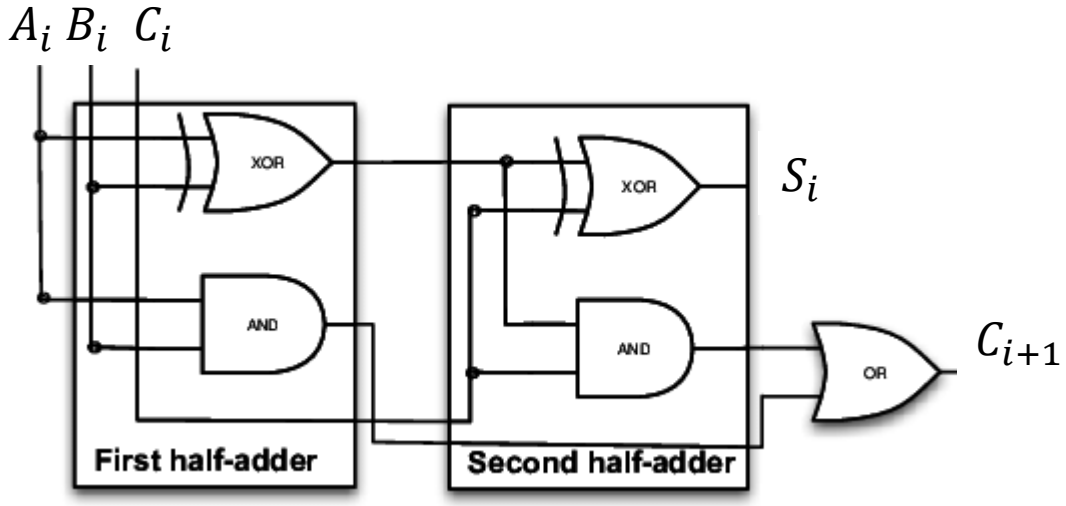
$$B_{out} = \bar{A}\,B + (\bar{A} + B)\,B_{in}$$



Multiple Bit Addition

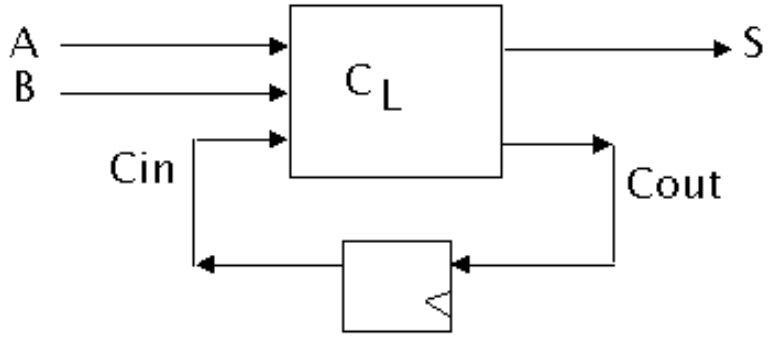
$$\begin{array}{r} C_i \\ A_i \\ +B_i \\ \hline C_{i+1} \quad S_i \end{array}$$



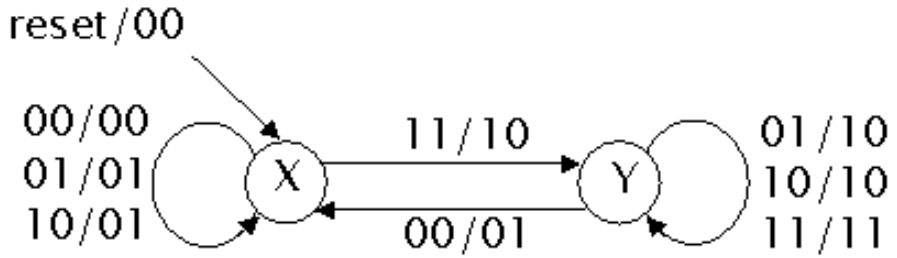


Serial Adder:

Example: serial adder

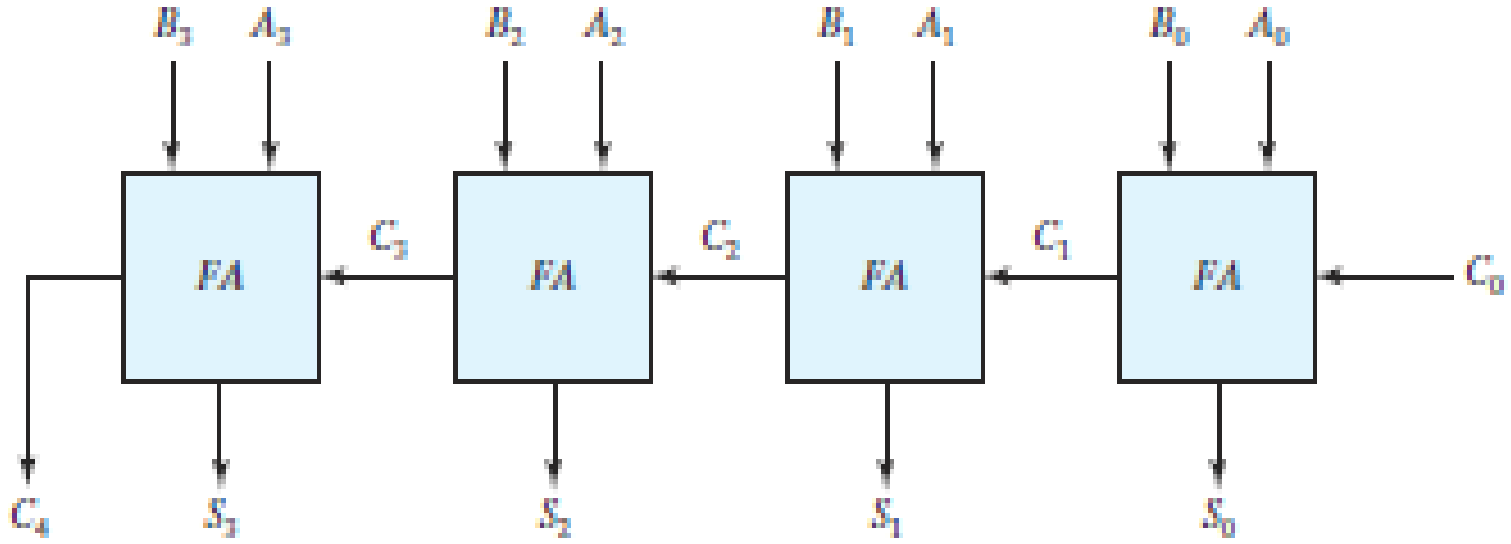


		P.S.	N.S.	
A	B	Cin	Cout	S
0	0	X:0	X:0	0
0	0	Y:1	X:0	1
0	1	X:0	X:0	1
0	1	Y:1	Y:1	0
1	0	X:0	X:0	1
1	0	Y:1	Y:1	0
1	1	X:0	Y:1	0
1	1	Y:1	Y:1	1



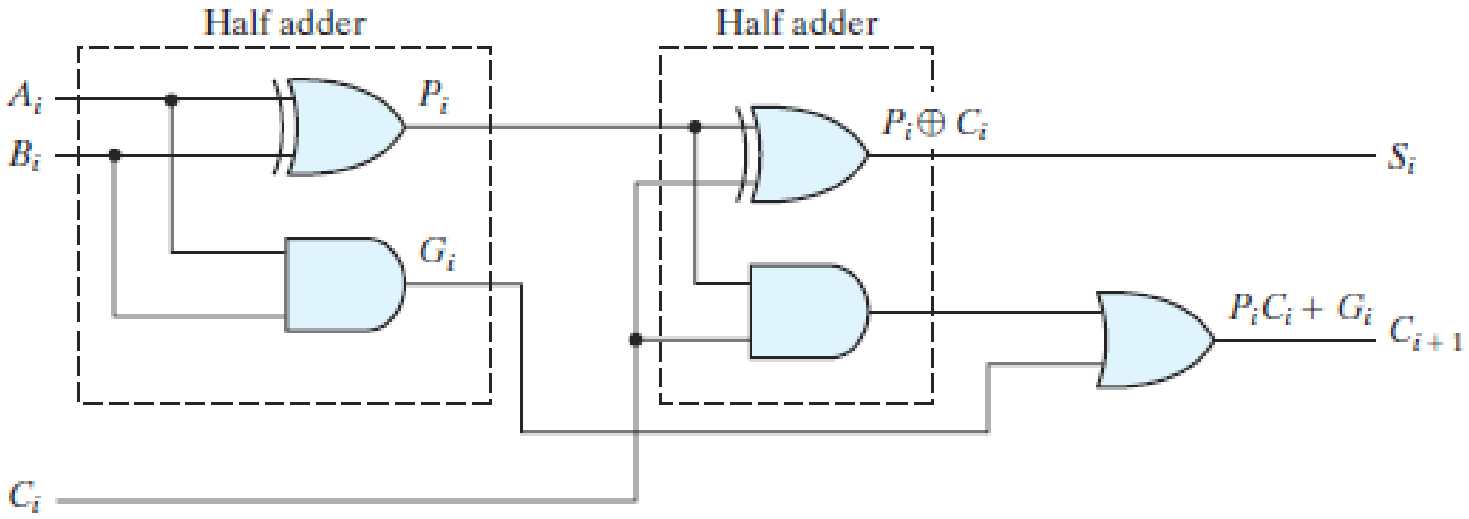
X: carry is 0
Y: carry is 1

Multiple Bit Addition (Parallel adder or Ripple adder)



C	1	1	1	0
A	0	1	0	1
B	0	1	1	1
S	1	1	0	0

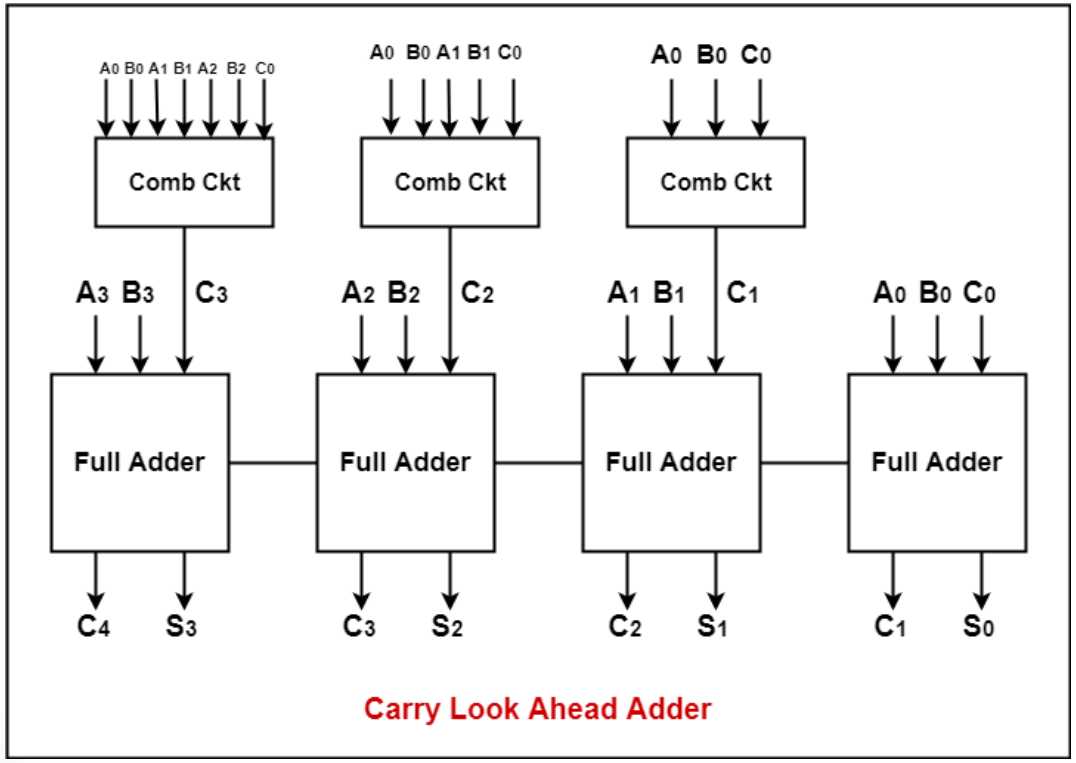
Carry Look ahead Logic



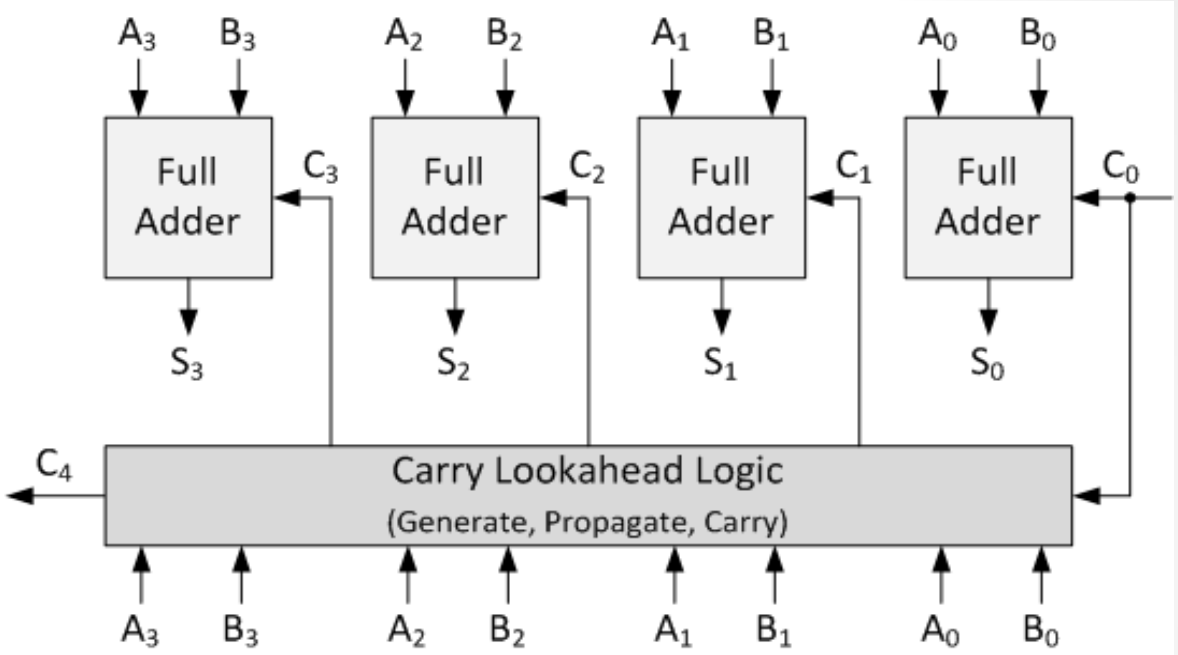
$$P_i = A_i \oplus B_i \qquad S_i = P_i \oplus C_i$$
$$G_i = A_i B_i \qquad C_{i+1} = G_i + P_i C_i$$

$$C_2 = G_1 + P_1 C_1 = G_1 + P_1 (G_0 + P_0 C_0) = G_1 + P_1 G_0 + P_1 P_0 C_0$$
$$C_3 = G_2 + P_2 C_2 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0$$

Multiple Bit Addition with Look Ahead Logic



(a)



(b)

- Each carry is generated independently.
- Reduced time than parallel and serial adders.

Difference between serial and parallel adder

- **Serial adder:**
  1) Slower
  2) It uses shift registers
  3) IT requires one full adder circuit.
  4) It is sequential circuit.
  5) Time required for addition depends on number of bits.

- **Parallel adder:**
  1) Faster
  2) It uses registers with parallel load capacity
  3) No. of full adder circuit is equal to no. of bits in binary adder.
  4)It is a combinational circuit
  5)Time required does not depend on the number of bits

Binary Coded Decimal

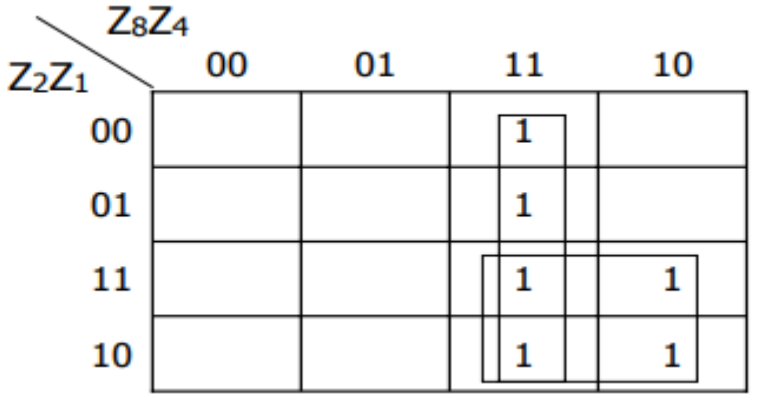
- Computers or calculators that perform arithmetic operations directly in the decimal number system represent decimal numbers in binary coded form. An adder for such a computer must employ arithmetic circuits that accept coded decimal numbers and present results in the same code.
- Eg:

- 8 4 2 = 1000 0100 0010
  8 4 2

BCD Adder

Derivation of BCD Adder

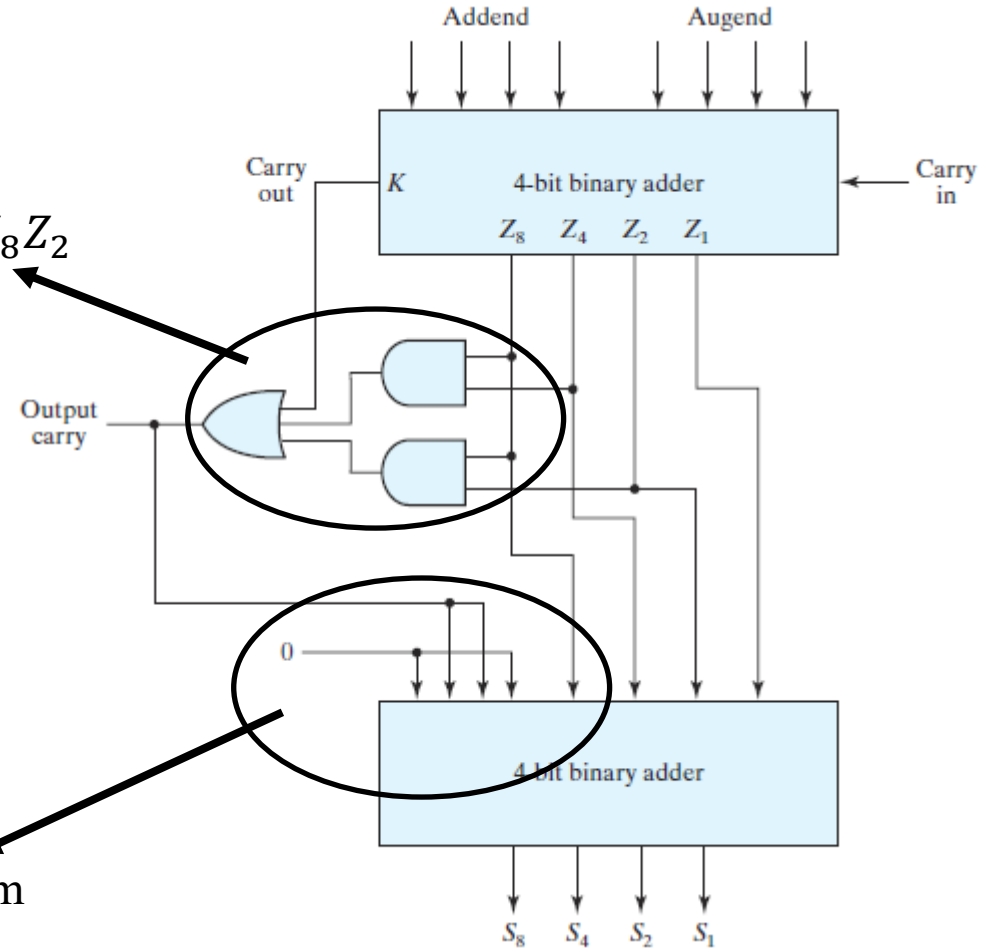
Binary Sum					BCD Sum					Decimal
K	Z_8	Z_4	Z_2	Z_1	C	S_8	S_4	S_2	S_1	
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	2
0	0	0	1	1	0	0	0	1	1	3
0	0	1	0	0	0	0	1	0	0	4
0	0	1	0	1	0	0	1	0	1	5
0	0	1	1	0	0	0	1	1	0	6
0	0	1	1	1	0	0	1	1	1	7
0	1	0	0	0	0	1	0	0	0	8
0	1	0	0	1	0	1	0	0	1	9
0	1	0	1	0	1	0	0	0	0	10
0	1	0	1	1	1	0	0	0	1	11
0	1	1	0	0	1	0	0	1	0	12
0	1	1	0	1	1	0	0	1	1	13
0	1	1	1	0	1	0	1	0	0	14
0	1	1	1	1	1	0	1	0	1	15
1	0	0	0	0	1	0	1	1	0	16
1	0	0	0	1	1	0	1	1	1	17
1	0	0	1	0	1	1	0	0	0	18
1	0	0	1	1	1	1	0	0	1	19



$$C = K + Z_8Z_4 + Z_8Z_2$$

BCD Adder Implementation

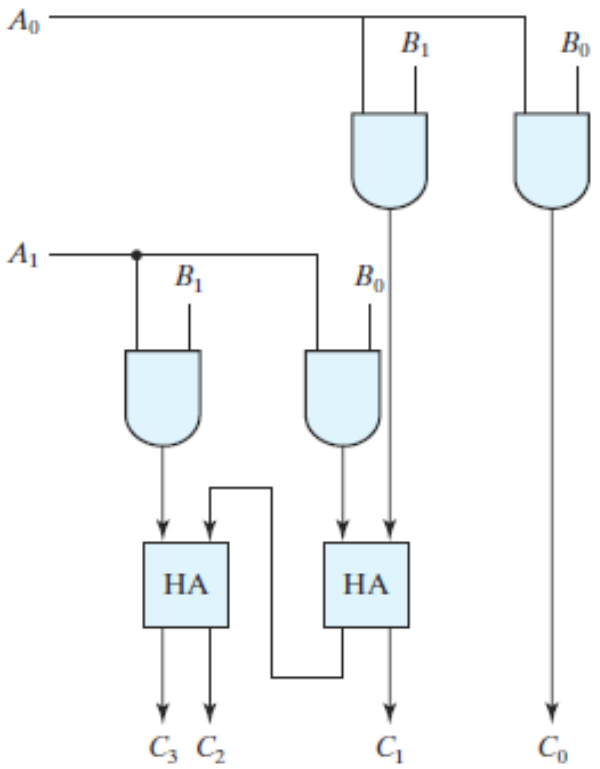
$C = K + Z_8Z_4 + Z_8Z_2$



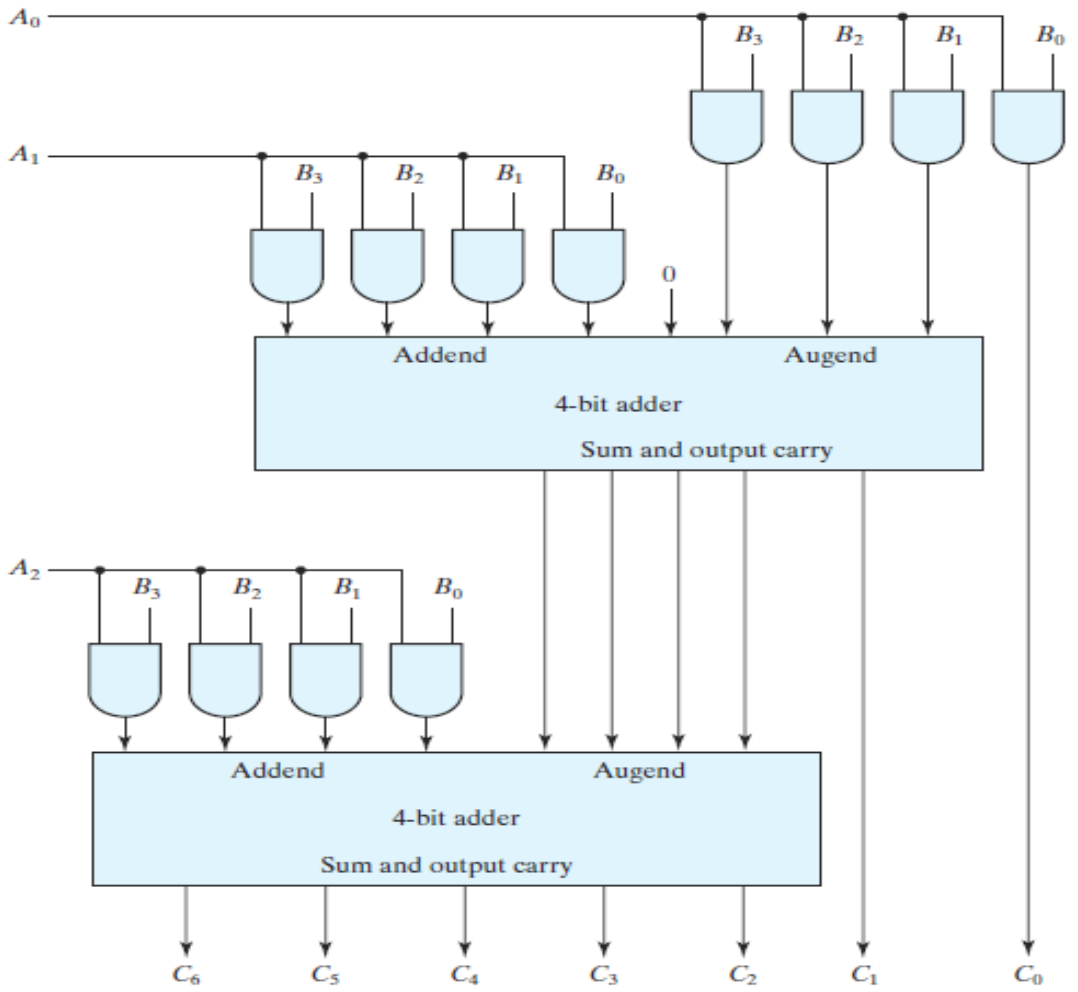
If C =1, it is necessary to add 0110 to binary sum

Binary Multiplier

$$
\begin{array}{cccc}
 & & B_1 & B_0 \\
 & & A_1 & A_0 \\
\hline
 & & A_0B_1 & A_0B_0 \\
 & A_1B_1 & A_1B_0 & \\
\hline
C_3 & C_2 & C_1 & C_0
\end{array}
$$



Four-bit by three-bit binary multiplier

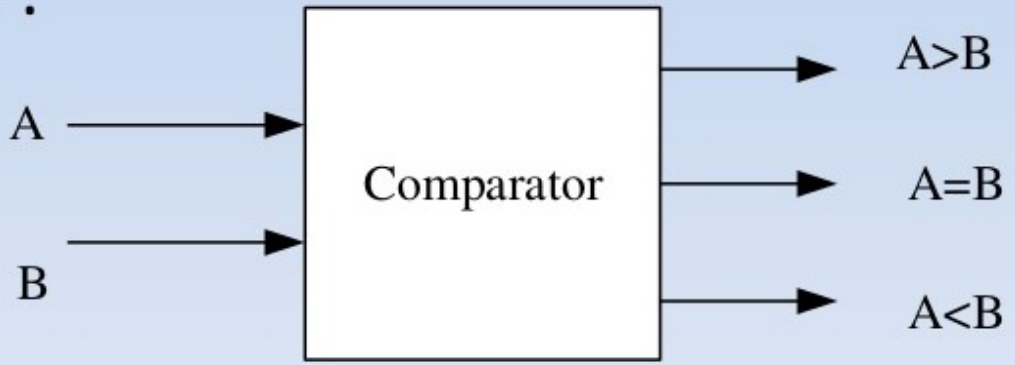


1-bit and 2-bit comparators:

1) A digital comparator is a combinational circuit that compares two digit or binary number.

2) It's determines their relative magnitudes in order to find out whether one number is equal, less than or greater than the other digital number.

3) Output would be like :
A=B , A>B ,A<B



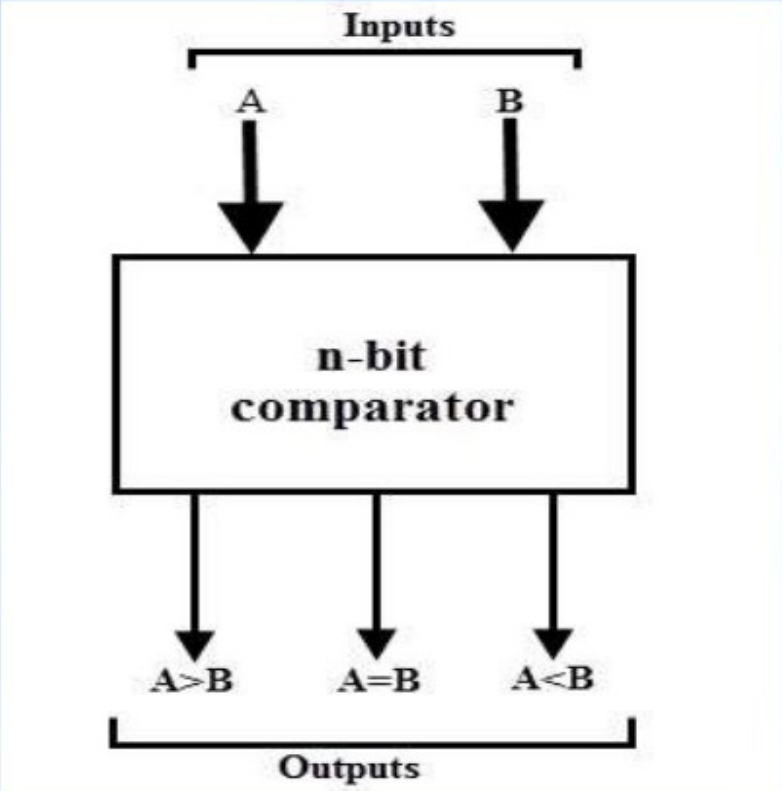
1- bit Comparator

There are 1-bit of two input X and Y

Truth Table:

Inputs		Outputs		
B	A	A > B	A = B	A < B
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

Block – diagram:



A>B

A \ B	0	1
0	0	0
1	1	0

Equation is $A>B = A.\overline{B}$

A<B

A \ B	0	1
0	0	1
1	0	0

Equation is $A<B = \overline{A}.B$

(A=B)

A \ B	0	1
0	1	0
1	0	1

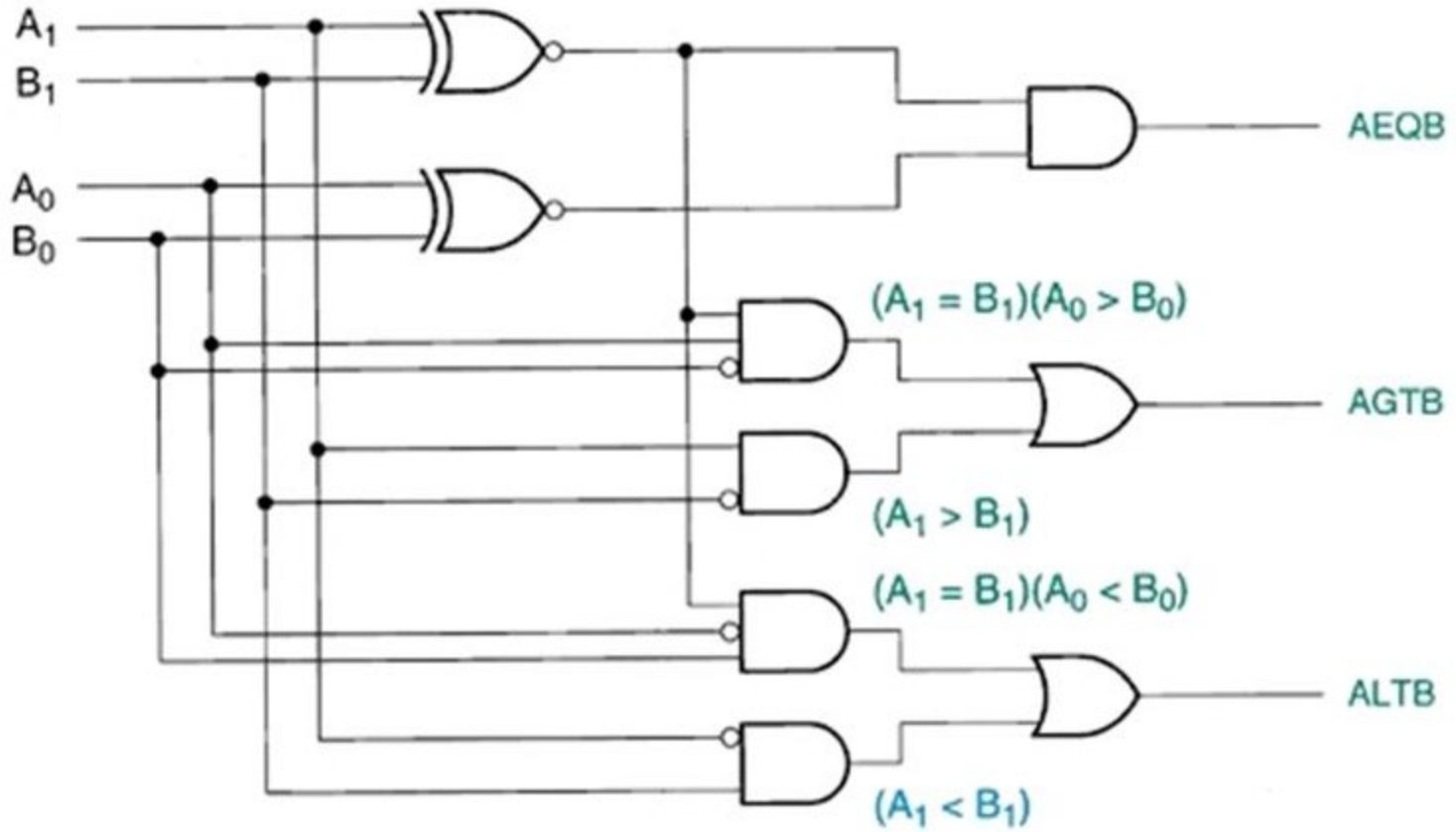
The equation is $f(A=B) = \overline{A}.\overline{B} + A.B$
$= A \text{ XNOR } B$

or we can write the equation for $f(A=B)$ as $\overline{A.\overline{B} + \overline{A}.B} = \overline{f(A>B)+f(A<B)}$

2-bit comparator:

Inputs				Outputs		
A_1	A_0	B_1	B_0	A>B	A=B	A<B
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

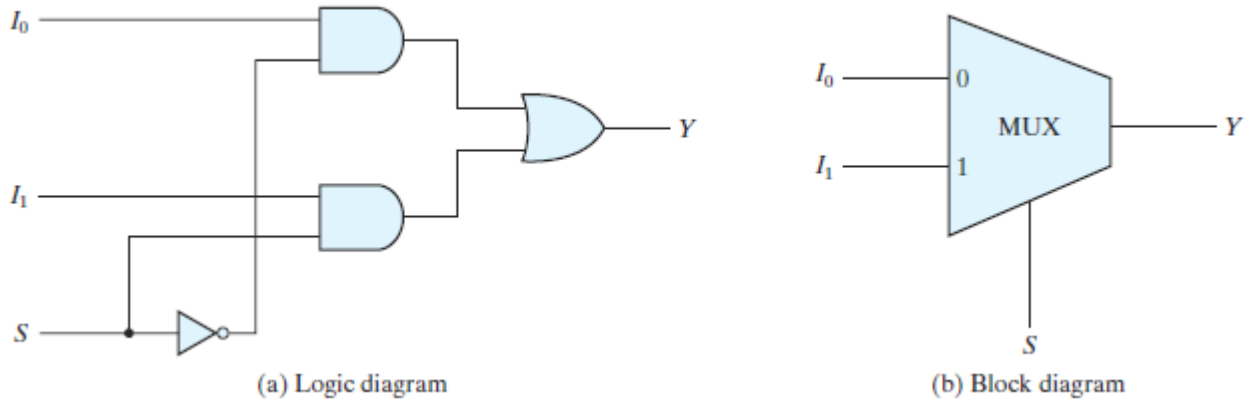
2 Bit Magnitude Comparator



Practice: 4 bit comparator

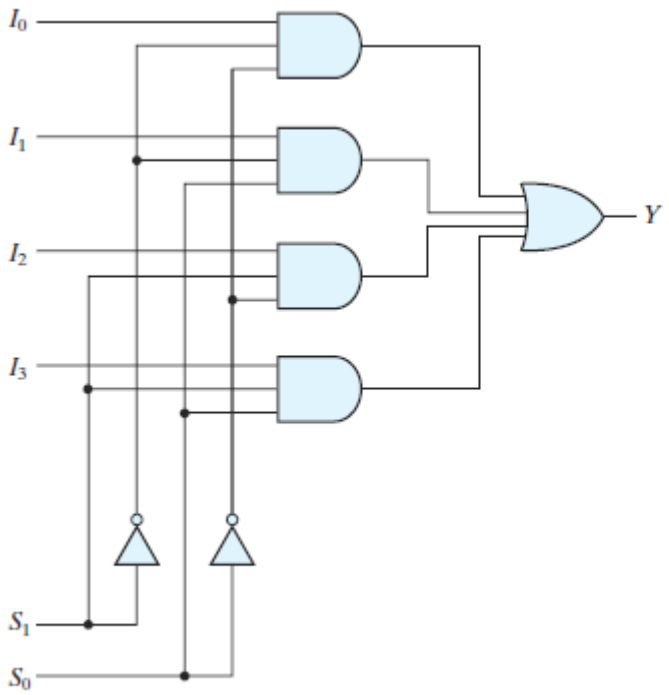
Multiplexer

- A multiplexer is a combinational circuit that selects binary information from one of many input lines and directs it to a single output line. The selection of a particular input line is controlled by a set of selection lines. Normally, there are 2^n input lines and n selection lines whose bit combinations determine which input is selected.
- A two-to-one-line multiplexer connects one of two 1-bit sources to a common destination, as shown in Figure. The circuit has two data input lines, one output line, and one selection line S . When S = 0, the upper AND gate is enabled and I_0 has a path to the output. When S = 1, the lower AND gate is enabled and I_1 has a path to the output. The multiplexer acts like an electronic switch that selects one of two sources



(a) Logic diagram (b) Block diagram

4 to 1 line Multiplexer



(a) Logic diagram

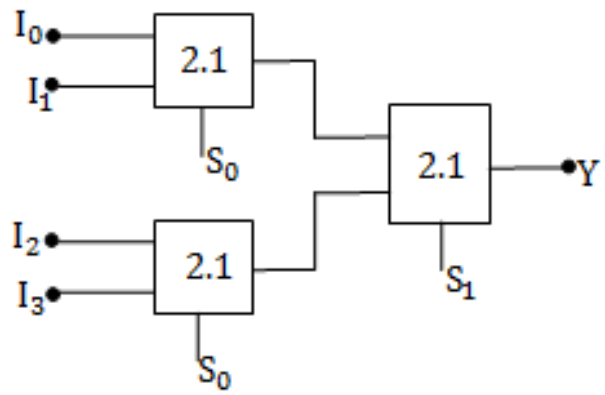
Logical Expression:

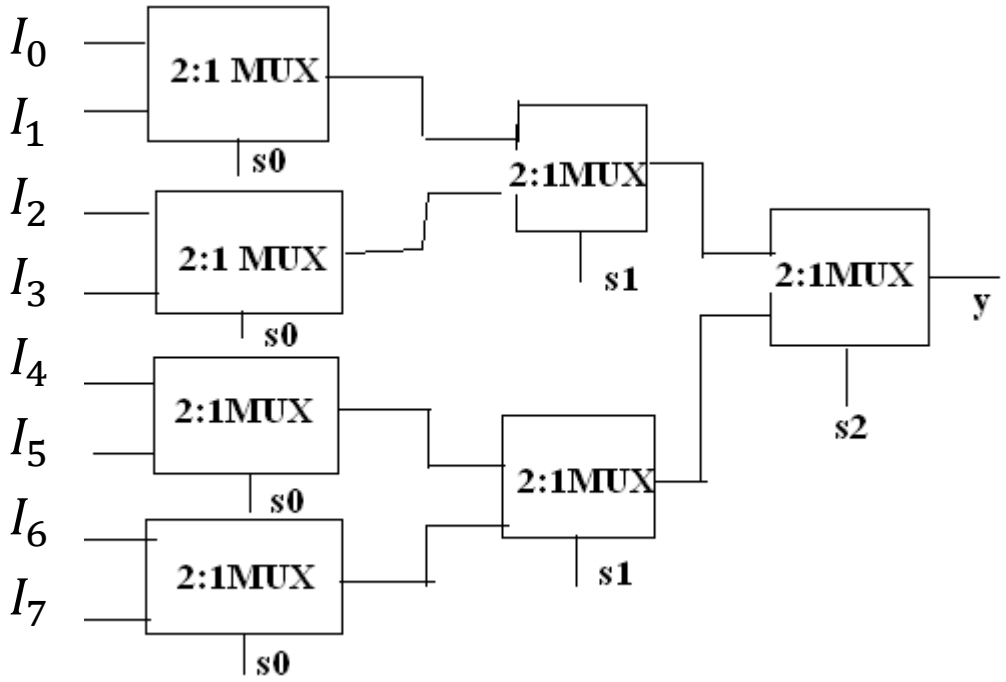
Y =$S_1'S_0'I_0$+$S_1'S_0I_1$+$S_1S_0'I_2$+$S_1S_0I_3$

S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

(b) Function table

Implement 4:1 and 8:1 MUX using 2:1 MUX



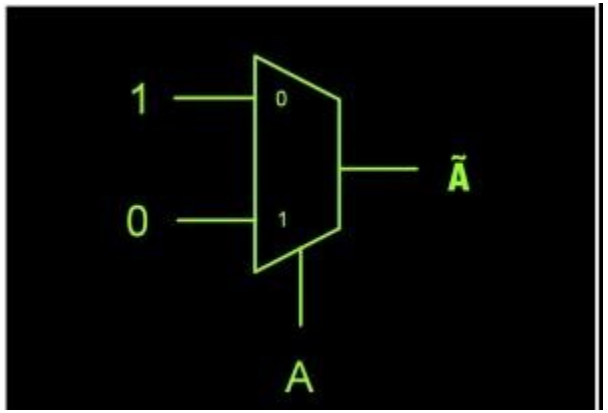


Implementation rule

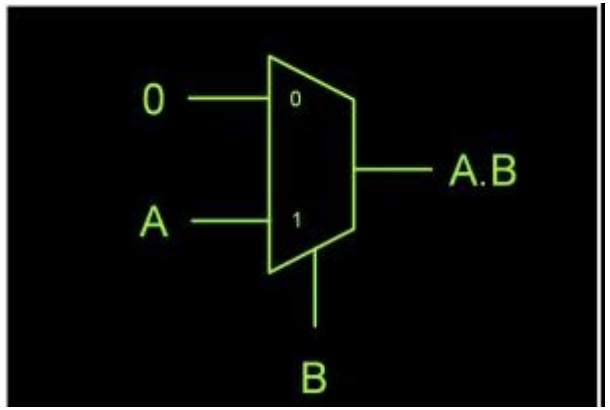
- Step 1: Calculate the number of MUX required to implement
- Step 2: Place the selection lines into different stages as per their number

- E.g. Say we have to implement 8:1 using 2:1
- First calculate the number of 2:1 MUX required: 8/2 + 4/2 + 2/2 = 4 + 2 + 1 = 7.
- Total Stage =3.
- Selection lines are : m= $\log_2$ (n). Where, n is number of inputs. So m = 3
- We have three stages and 3 selection line, So we will place it accordingly.
- S_0 in first stage S_1 in the second stage and S_2 in third stage.

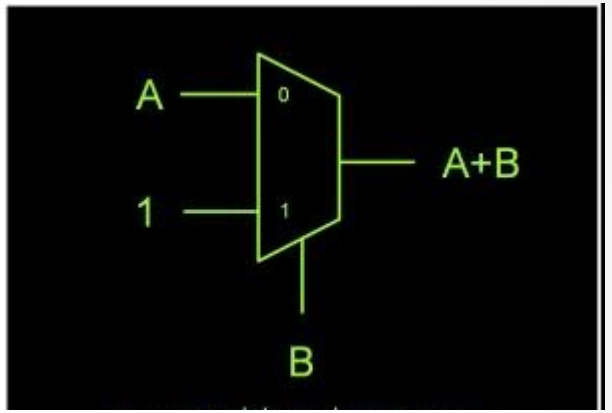
MUX as universal



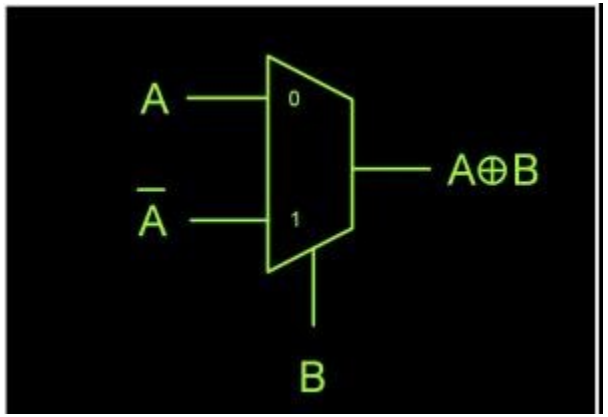
NOT gate



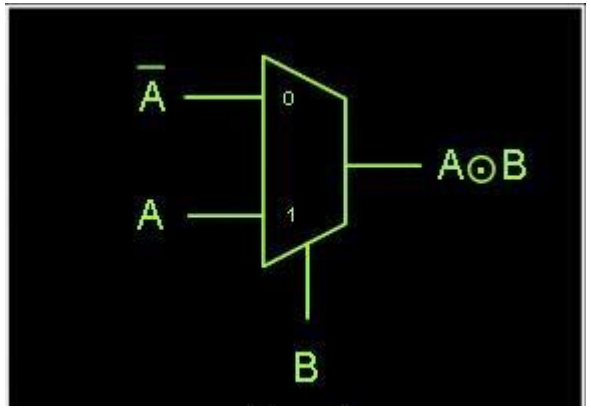
AND gate



OR gate



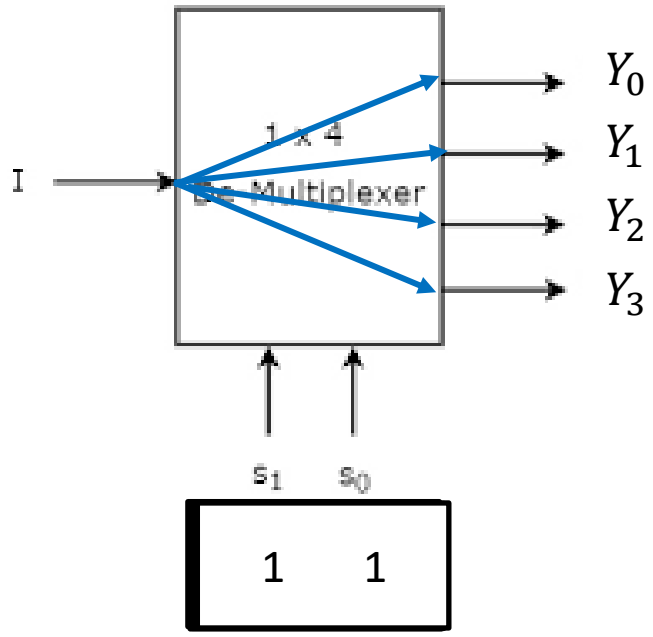
XOR gate



XNOR gate

De Multiplexer

- Single i/p and Many o/p



Selection Inputs		Outputs			
S_1	S_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	I
0	1	0	0	I	0
1	0	0	I	0	0
1	1	I	0	0	0

$$Y_3 = s_1 s_0 I$$

$$Y_2 = s_1 s_0' I$$

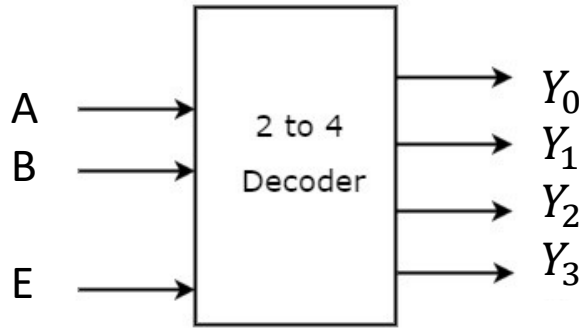
$$Y_1 = s_1' s_0 I$$

$$Y_0 = s_1' s_0' I$$

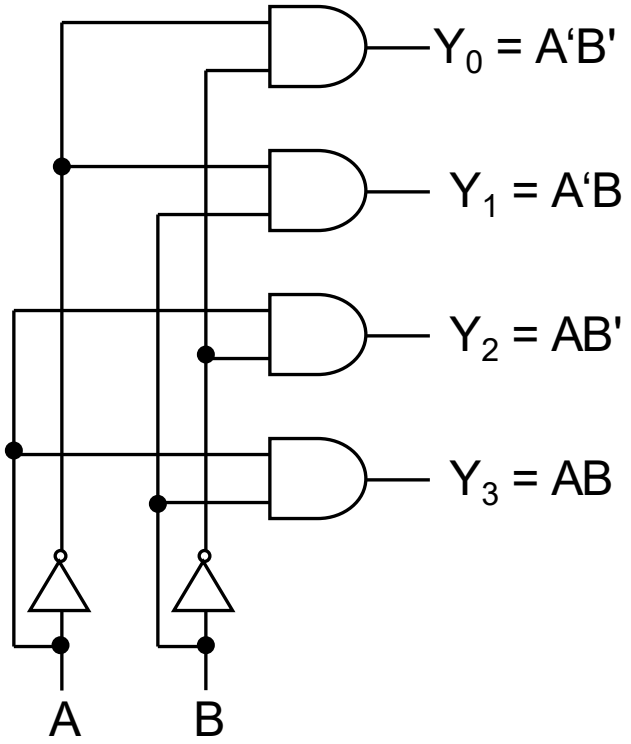
Decoder

- Decoder is a combinational circuit which have many inputs and many outputs.
- Generally decoder converts 2^n input to n- outputs
- It is used to convert binary or binary coded number to other code
- E.g. Binary to octal (3×8), BCD to Decimal (4×10), Binary to Hexadecimal, BCD to Seven segments

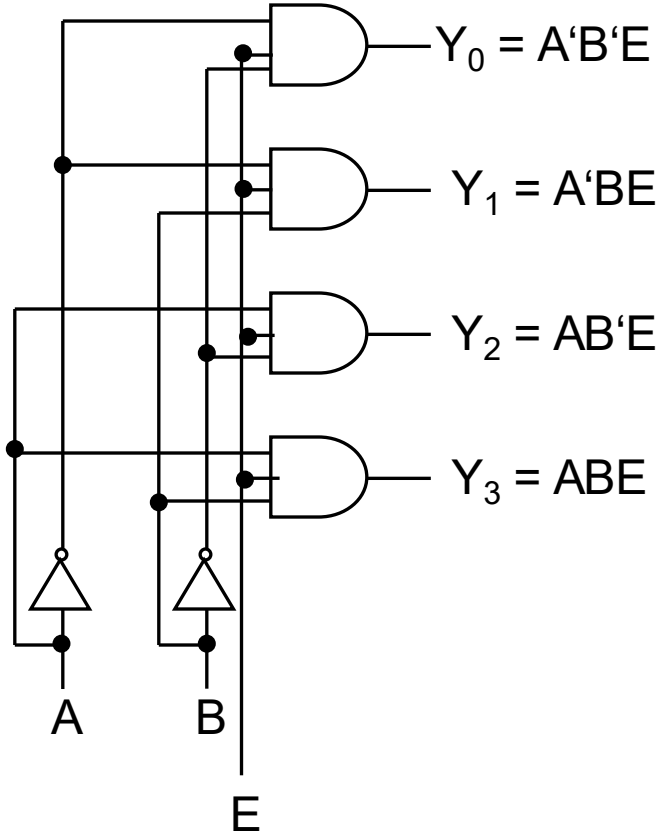
- 2 to 4 decoder is minimum possible decoder



2 to 4 Decoder

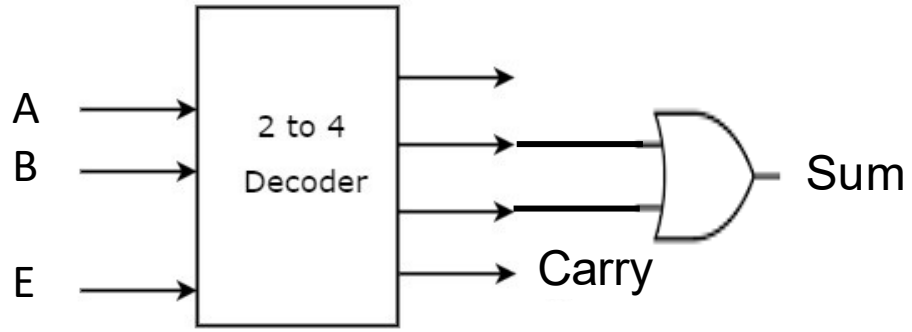


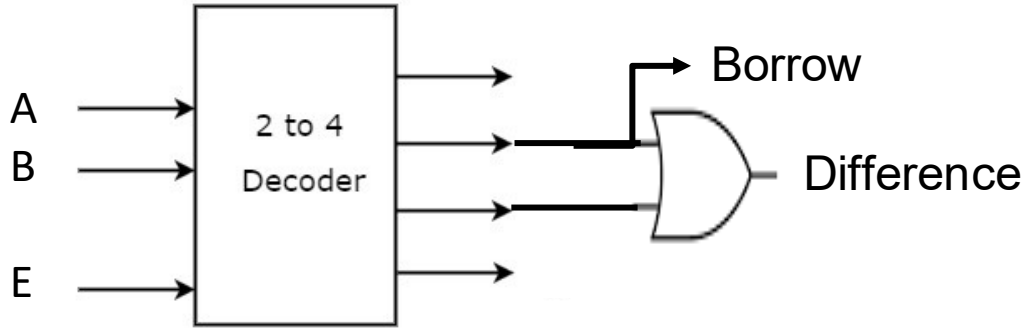
Without Enable Input



With Enable Input

Implement Half Adder/ Half Subtractor using 2×4 Decoder





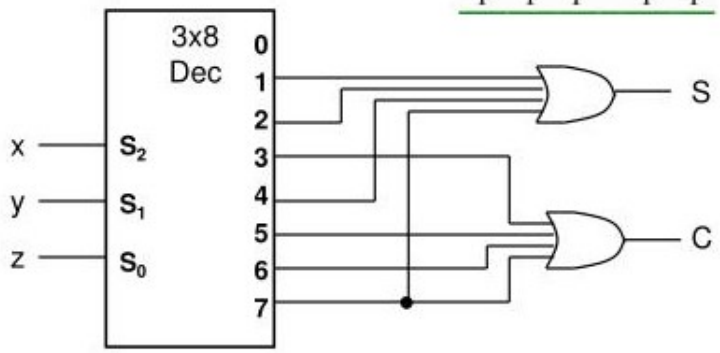
Implement FA using 3 to 8 Decoder

Example: Full adder

$S(x, y, z) = \Sigma\, m(1,2,4,7)$

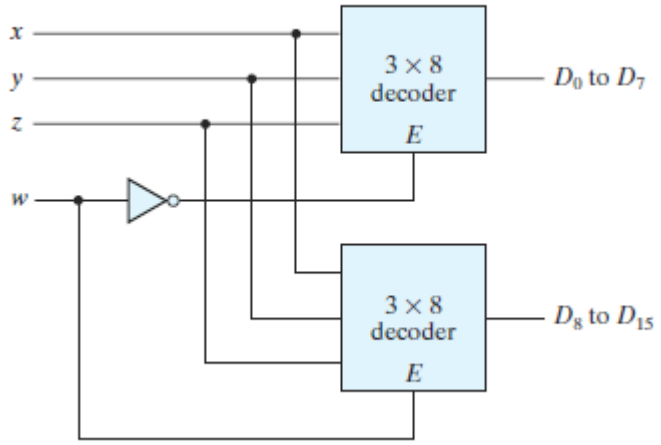
$C(x, y, z) = \Sigma\, m(3,5,6,7)$

x	y	z	C	S
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



4×16 Decoder using 3×8 Decoder

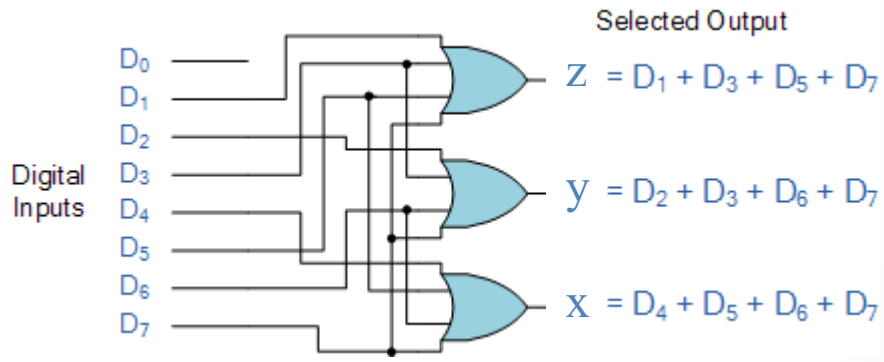
- Decoders with enable inputs can be connected together to form a larger decoder circuit. Figure shows two 3-to-8-line decoders with enable inputs connected to form a 4-to-16-line decoder. When w 0, the top decoder is enabled and the other is disabled. The bottom decoder outputs are all 0's, and the top eight outputs generate minterms 0000 to 0111. When w 1, the enable conditions are reversed



Encoder

- An encoder is a digital circuit that performs the inverse operation of a decoder.
- encoder has 2^n (or fewer) input lines and n output lines. The output lines, as an aggregate, generate the binary code corresponding to the input value.

Inputs								Outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1



$$z = D_1 + D_3 + D_5 + D_7$$
$$y = D_2 + D_3 + D_6 + D_7$$
$$x = D_4 + D_5 + D_6 + D_7$$

- binary number. It is assumed that only one input has a value of 1 at any given time. The encoder can be implemented with OR gates whose inputs are determined directly from the truth table.

BCD to gray code converter:

Truth Table relating BCD to Gray Code

Decimal	BCD input				Gray Code output			
	B3	B2	B1	B0	D3	D2	D2	D0
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

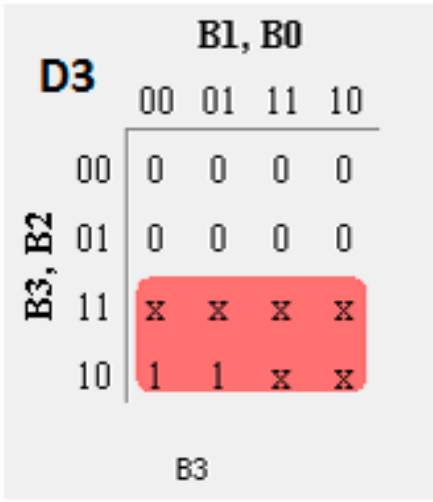
Boolean expression for each BCD bits can be written as

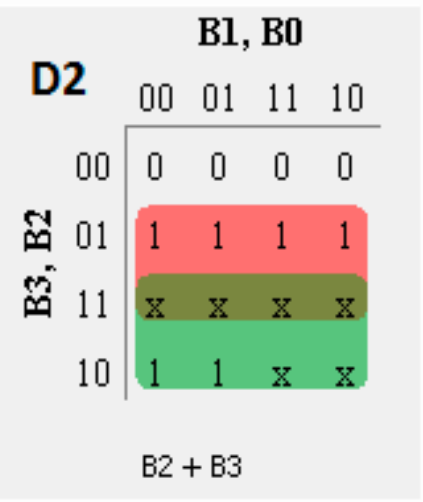
$D3 = m(8, 9) + d(10, 11, 12, 13, 14, 15)$

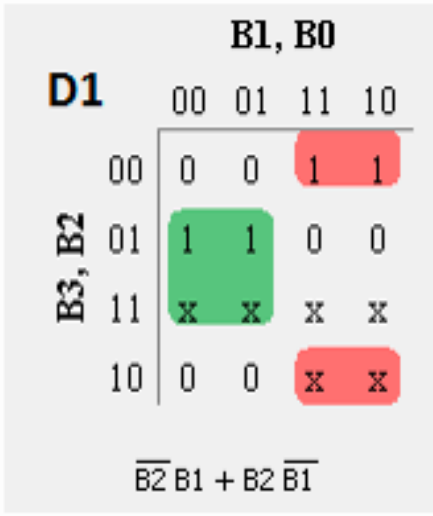
$D2 = m(4, 5, 6, 7, 8, 9) + d(10, 11, 12, 13, 14, 15)$

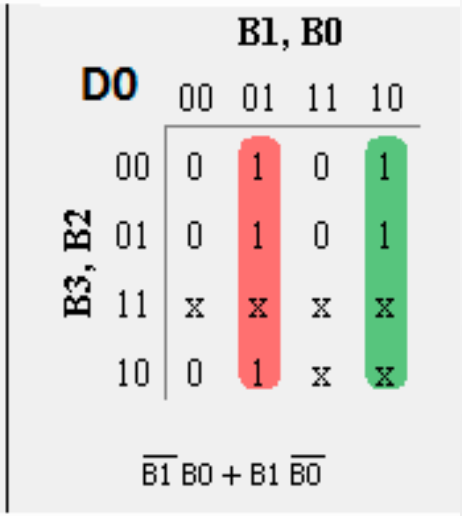
$D1 = m(2, 3, 4, 5) + d(10, 11, 12, 13, 14, 15)$

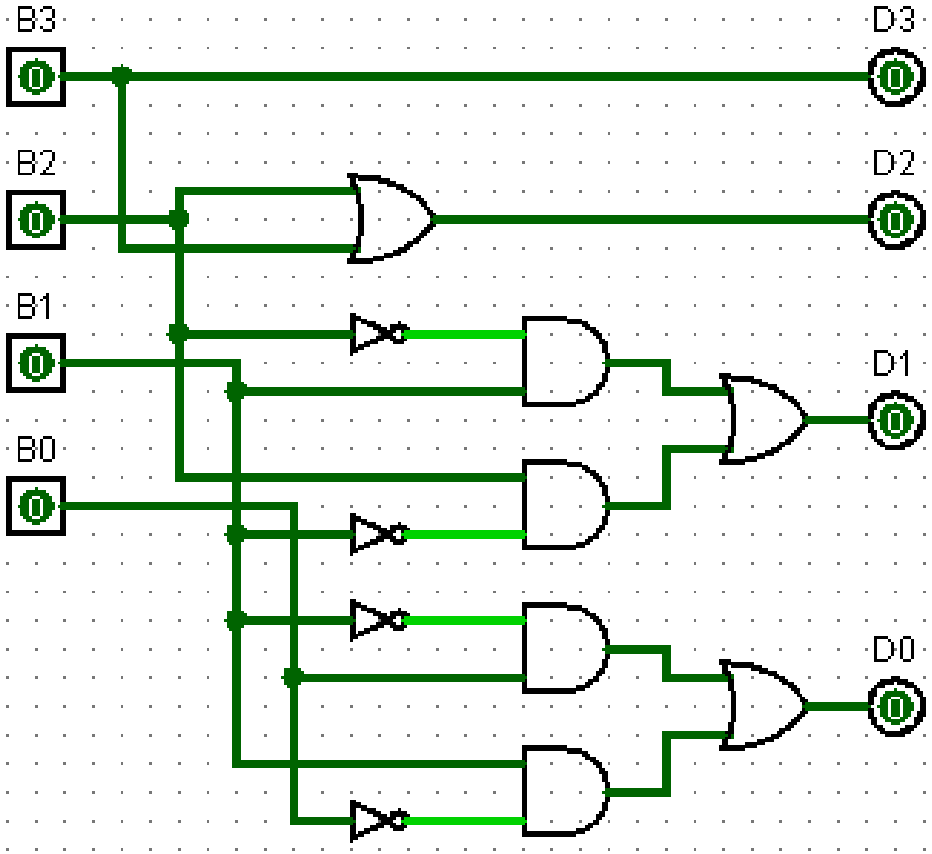
$D0 = m(1, 2, 5, 6, 9) + d(10, 11, 12, 13, 14, 15)$











BCD to gray code converter:

5421 to Natural BCD converter: https://www.youtube.com/watch?v=S-O75HI10qY

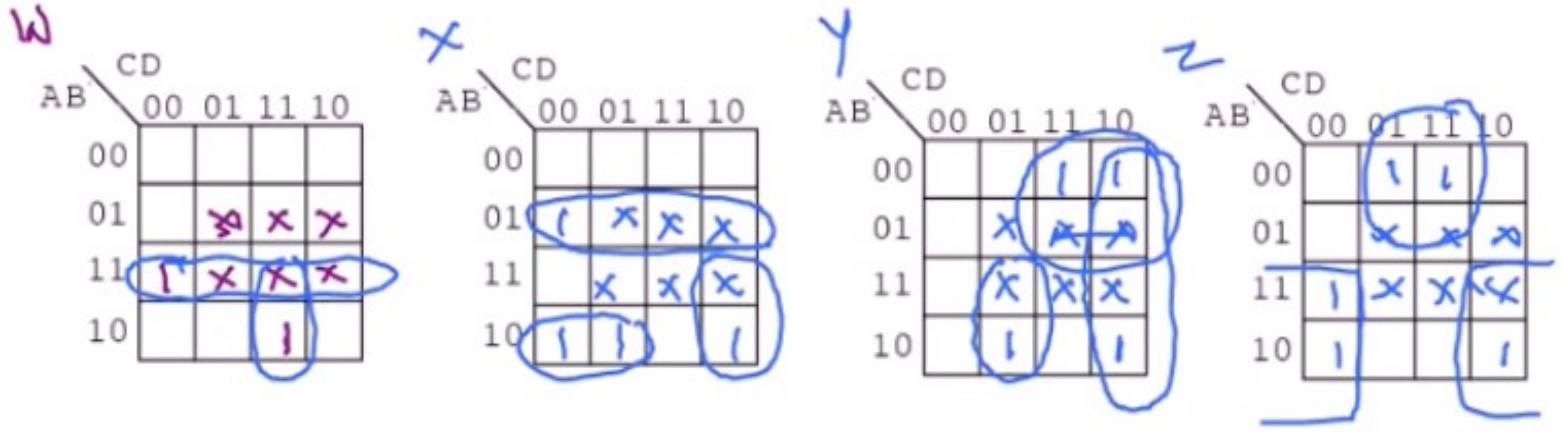
GATE PAPER
Gateway to IISc & IITs

$W(A,B,C,D) = \Sigma m(11,12) + d(5,6,7,13,14,15)$

$X = \Sigma m(4,8,9,10) + "$

$Y = \Sigma m(2,3,9,10) + "$

$Z = \Sigma m(1,3,8,10,12) + "$

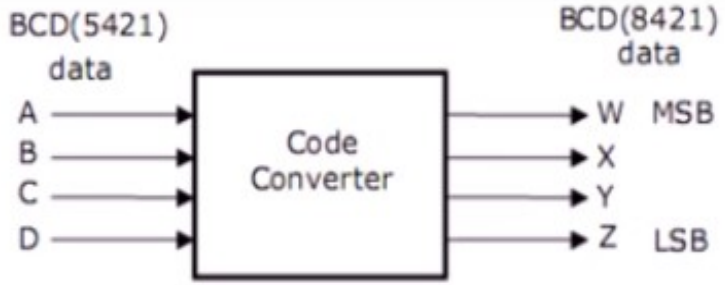


$W = AB + AC\bar{D}$

$X = \bar{A}B + A\bar{B}\bar{C} + A\bar{C}\bar{D}$

$Y = \bar{A}C + C\bar{D} + A\bar{C}D$

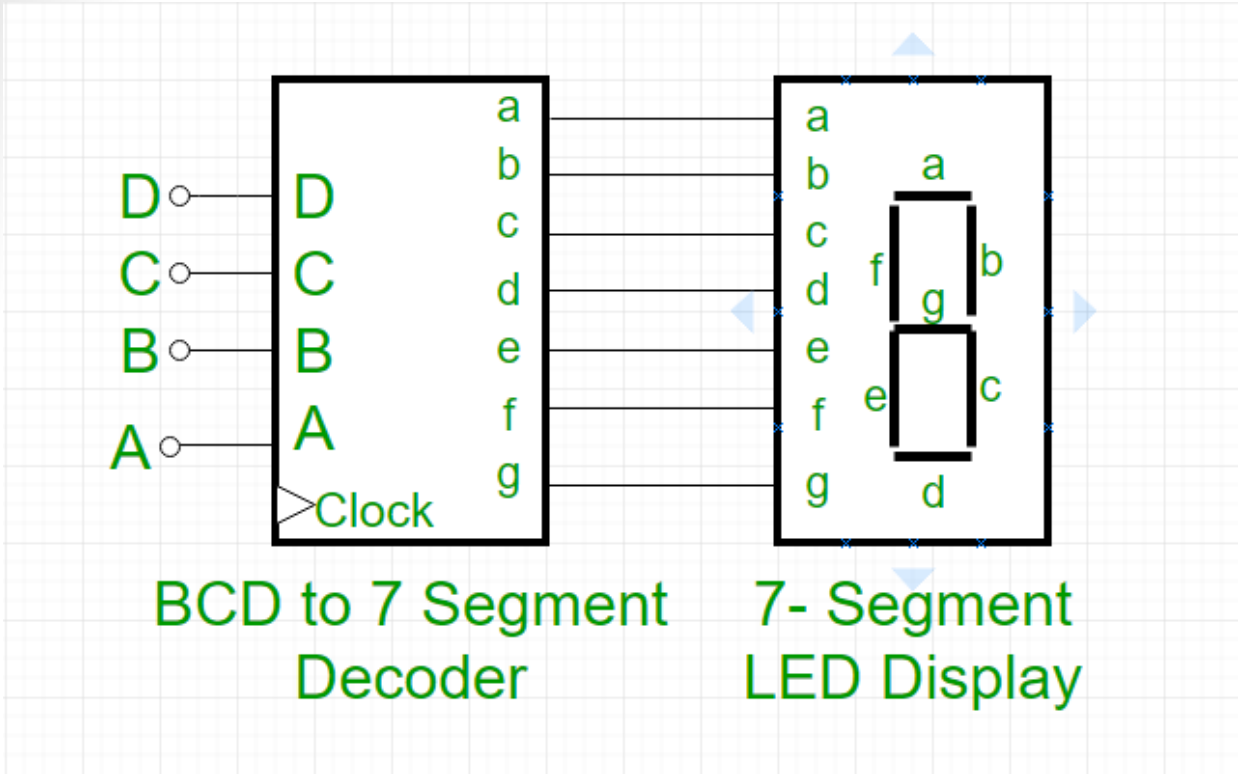
$Z = \bar{A}D + A\bar{D}$



Decimal	BCD(5421)				BCD(8421)			
	A	B	C	D	W	X	Y	Z
0 m_0	0	0	0	0	0	0	0	0
1 1	0	0	0	1	0	0	0	1
2 2	0	0	1	0	0	0	1	0
3 3	0	0	1	1	0	0	1	1
4 4	0	1	0	0	0	1	0	0
5 8	1	0	0	0	1	0	0	1
6 9	1	0	0	1	1	0	0	1
7 10	1	0	1	0	0	1	1	1
8 11	1	0	1	1	1	0	0	0
9 12	1	1	0	0	1	0	0	1

268 / 326

BCD to 7 segment display:

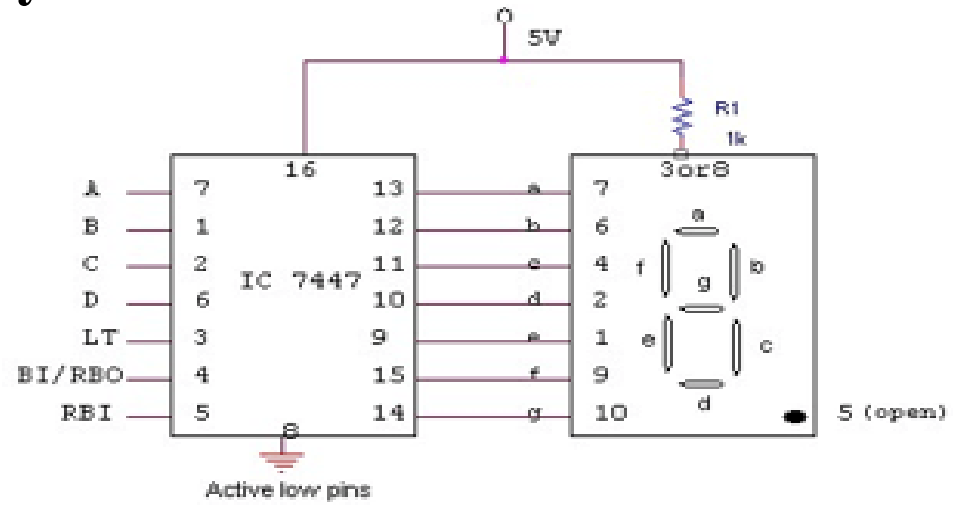


Decimal Digit	Input lines				Output lines							Display pattern
	A	B	C	D	a	b	c	d	e	f	g	
0	0	0	0	0	1	1	1	1	1	1	0	
1	0	0	0	1	0	1	1	0	0	0	0	
2	0	0	1	0	1	1	0	1	1	0	1	
3	0	0	1	1	1	1	1	1	0	0	1	
4	0	1	0	0	0	1	1	0	0	1	1	
5	0	1	0	1	1	0	1	1	0	1	1	
6	0	1	1	0	1	0	1	1	1	1	1	
7	0	1	1	1	1	1	1	0	0	0	0	
8	1	0	0	0	1	1	1	1	1	1	1	
9	1	0	0	1	1	1	1	1	0	1	1	

Examples:

1. Design BCD to gray code converter.
2. Design 2421 to Natural BCD converter.
3. Design 3-bit binary to gray code and gray code to binary converters.
4. BCD to 7 segment display converter.

BCD to 7 segment display converter:



TRUTH TABLE:

BCD Inputs				Output Logic Levels from IC 7447 to 7-segments							Decimal number display
D	C	B	A	a	b	c	d	e	f	g	
0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	1	1	0	0	1	1	1	1	1
0	0	1	0	0	0	1	0	0	1	0	2
0	0	1	1	0	0	0	0	1	1	0	3
0	1	0	0	1	0	0	1	1	0	0	4
0	1	0	1	0	1	0	0	1	0	0	5
0	1	1	0	1	1	0	0	0	0	0	6
0	1	1	1	0	0	0	1	1	1	1	7
1	0	0	0	0	0	0	0	0	0	0	8
1	0	0	1	0	0	0	1	1	0	0	9

Parity Generator and Checker

- The parity generating technique is one of the most widely used error detection techniques for the data transmission. In digital systems, when binary data is transmitted and processed , data may be subjected to noise so that such noise can alter 0s (of data bits) to 1s and 1s to 0s.

- Hence, parity bit is added to the word containing data in order to make number of 1s either even or odd.
- The message containing the data bits along with parity bit is transmitted from transmitter node to receiver node. At the receiving end, the number of 1s in the message is counted and if it doesn't match with the transmitted one, then it means there is an error in the data.

Even Parity Generator

- Let us assume that a 3-bit message is to be transmitted with an even parity bit. Let the three inputs A, B and C 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 bit generator (P)
A	B	C	Y
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

A \ BC	00	01	11	10
0		1		1
1	1		1	

$$P = \bar{A}\,\bar{B}\,C + \bar{A}\,B\,\bar{C} + A\,\bar{B}\,\bar{C} + A\,B\,C$$

$$= \bar{A}\,(\bar{B}\,C + B\,\bar{C}) + A\,(\bar{B}\,\bar{C} + B\,C)$$

$$= \bar{A}\,(B \oplus C) + A\,(\overline{B \oplus C})$$

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

Even Parity Checker

- Consider that three input message along with even parity bit is generated at the transmitting end. These 4 bits are applied as input to the parity checker circuit which checks the possibility of error on the data. Since the data is transmitted with even parity, four bits received at circuit must have an even number of 1s. If any error occurs, the received message consists of odd number of 1s. The output of the parity checker is denoted by PEC (parity error check). The below table shows the truth table for the even parity checker in which PEC = 1 if the error occurs, i.e., the four bits received have odd number of 1s and PEC = 0 if no error occurs.

4-bit received message				Parity error check C_p
A	B	C	P	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Contd:

AB \ CP	00	01	11	10
00		1		1
01	1		1	
11		1		1
10	1		1	

$$PEC = \bar{A}\,\bar{B}\,(\bar{C}\,D + C\,\bar{D}) + \bar{A}\,B\,(\bar{C}\,\bar{D} + C\,D) + A\,B\,(\bar{C}\,D + C\,\bar{D}) + A\,\bar{B}\,(\bar{C}\,\bar{D} + C\,D)$$

$$= \bar{A}\,\bar{B}\,(C \oplus D) + \bar{A}\,B\,(\overline{C \oplus D}) + A\,B\,(C \oplus D) + A\,\bar{B}\,(\overline{C \oplus D})$$

$$= (\bar{A}\,\bar{B} + A\,B)\,(C \oplus D) + (\bar{A}\,B + A\,\bar{B})\,(\overline{C \oplus D})$$

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