RGPVNOTES.IN

Program : **B.Tech**

Subject Name: **Digital Systems**

Subject Code: **CS-304**

Semester: **3rd**

Subject Name: Digital Systems **Subject Code**: CS303

Subject Notes

UNIT-II

Combinational Logic: Half adder, Half subtractor, Full adder, Full subtractor, look- ahead carry generator,BCD adder, Series and parallel addition, Multiplexer – demultiplexer, encoder-decoder, arithmetic circuits, ALU

2.1 COMBINATIONAL LOGIC:

A combinational logic circuit consists of logic gates whose outputs at any time are determined directly from the present combination of inputs without regard to previous inputs. It consists of input variables, logic gates and output variables. The design of combinational circuit start from the verbal outline of the problem and ends in a logic circuit diagram, or asset of Boolean functions from which the logic diagram can be easily obtained. The design steps are:

a) The problem is stated.
b) The number of input and required output variables is determined.
c) Truth table is derived.
d) Simplified Boolean function for each output is obtained.
e) Logic diagram is drawn.

Some examples of combinational circuits are: Half adder, full adder, half subtractor, full subtractor, BCD adder, Series and parallel adder, BCD adders, Look-ahead carry generator.

2.2 HALF ADDER:

A combinational circuit that perform the addition of two bits is called half adder. This circuit needs two binary inputs and two binary outputs. The input variables, augend (X) and addend (Y) bits; the output variables SUM (S) and CARRY (C).

Truth Table:

INPUTS		OUTPUTS	
X	Y	SUM (S)	CARRY(C)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

The simplified output Boolean function : **SUM (S) = X'.Y + X.Y' and CARRY (C) = X.Y**

The logic diagram of Half Adder :

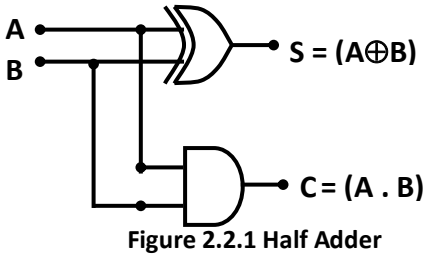

Figure 2.2.1 Half Adder

2.3 FULL ADDER:

When the augend and addend numbers contain more significant digits, the carry obtained from the addition of two bits is added to the next higher order pair of significant bits. The combinational circuit that performs the addition of three bits (two significant bits and a previous carry) is a full adder. It consists of three inputs (X and Y are actual 2-inputs and third input represents the $CARRY_{IN}$ (C_{IN}) generated from the previous lower significant bit position) and two outputs, SUM (S) and $CARRY_{OUT}$ (C_{OUT}).

Truth Table:

INPUTS			OUTPUTS	
X	Y	C_{IN}	SUM (S)	CARRY(C_{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

Boolean expression shown from the truth table which is shown:

SUM = $X'.Y'.C_{IN} + X'.Y.C_{IN}' + X.Y'.C_{IN}' + X.Y.C_{IN}$

SUM = $(X \oplus Y \oplus C_{IN})$

CARRY = $X'.Y.C_{IN} + X.Y.C_{IN}' + X.Y'.C_{IN} + X.Y.C_{IN}$

$= Y.C_{IN}(X + X') + X.Y.C_{IN}' + X.Y'.C_{IN}$

$= Y.C_{IN} + X.Y.C_{IN}' + X.Y'.C_{IN}$

$= Y(C_{IN} + C_{IN}'.X) + X.Y'.C_{IN}$

$= Y(C_{IN} + X) + X.Y'.C_{IN}$

$= Y.C_{IN} + Y.X + X.Y'.C_{IN}$

$= C_{IN}(Y + X.Y') + Y.X$

$= C_{IN}(Y + X) + Y.X$

CARRY = $C_{IN}.Y + C_{IN}.X + Y.X$

Using K-Map method, simplify the output expression for SUM(S) and CARRY (C_{OUT}):

K-Map for SUM:

	$Y'.C_{IN}'$	$Y'.C_{IN}$	$Y.C_{IN}$	$Y.C_{IN}'$
X'	0	(1)	0	(1)
X	(1)	0	(1)	0

K-Map for CARRY:

	$Y'.C_{IN}'$	$Y'.C_{IN}$	$Y.C_{IN}$	$Y.C_{IN}'$
X'	0	0	1	0
X	0	1	1	1

$S = X'.Y'.C_{IN} + X'.Y.C_{IN}' + X.Y'.C_{IN}' + X.Y.C_{IN}$

$C_{OUT} = X.Y + Y.C_{IN} + X.C_{IN}$

Logic Diagram of Full Adder using 2-half adder and OR gate:

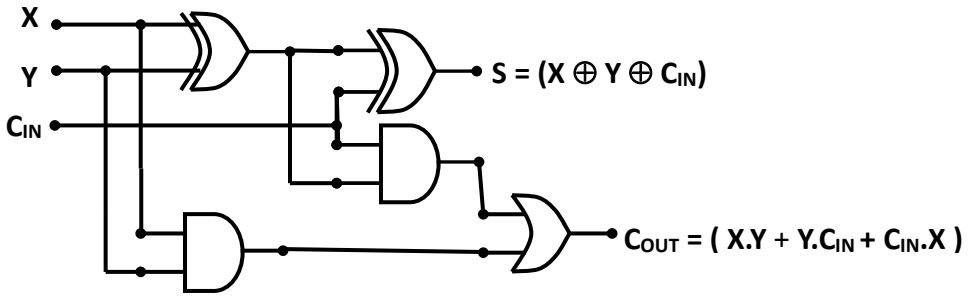

Figure 2.3.1 Full Adder

2.4 HALF SUBTRACTOR:

The half-subtractor is a combinational circuit which is used to perform subtraction of two bits. It has two inputs, X (minuend) and Y (subtrahend) and two outputs D (difference) and B (borrow). The logic symbol and truth table are shown below.

TRUTH TABLE FOR HALF SUBTRACTOR:

INPUT		OUTPUT	
A	B	DIFFERENCE(D)	BORROW (BOR_{OUT})
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Simplified Boolean function for the outputs are derived using K-map:

K-Map for DIFFERENCE(D):

	B'	B
A'	0	(1)
A	(1)	0

D = A'.B + A.B'

K-Map for BORROW(BOR_{out}):

	B'	B
A'	0	1
A	0	0

BOR_{OUT} = A'.B

LOGIC DIAGRAM OF HALF SUBTRACTOR:

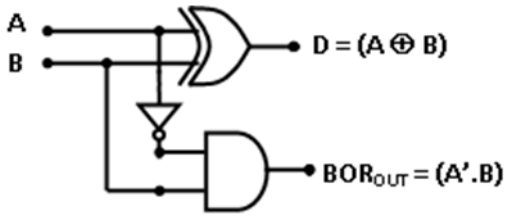

Figure 2.4.1 Half Subtractor

2.5 FULL SUBTRACTOR

The half-subtractor can be used for LSB(Least Significant Bit) subtraction. If there is a borrow during the subtraction of the LSBs, it affects the subtraction in the next higher coloumn; the subtrahend bit is subtracted from the minuend bit, considering the borrow from that coloumn used for the subtraction in the preceding column. Such a subtraction is performed by a full-subtractor.

The Full-subtractor is a combinational circuit which is used to performs subtraction between two bits, taking into account that a 1 may have been borrowed by a lower significant stage. It has three inputs, A(minuend) and B (subtrahend) and BORROW IN (BOR_{IN}) and two outputs D (difference) and BOR_{OUT} (borrow out).

TRUTH TABLE FOR FULL SUBTRACTOR:

INPUT			OUTPUT	
A	B	BOR_{IN}	DIFFERENCE(D)	BORROW (BOR_{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

Simplified Boolean function for the outputs are derived using K-map:

K-Map for DIFFERENCE(D):

	$B'.BOR_{IN}'$	$B'.BOR_{IN}$	$B.BOR_{IN}$	$B.BOR_{IN}'$
A'	0	(1)	0	(1)
A	(1)	0	(1)	0

K-Map for BORROW(BOR_{out}):

	$B'.BOR_{IN}'$	$B'.BOR_{IN}$	$B.BOR_{IN}$	$B.BOR_{IN}'$
A'	0	1	1	1
A	0	0	1	0

$D = A'.B'.BOR_{IN} + A'.B.BOR_{IN}' + A.B'.BOR_{IN}' + A.B.BOR_{IN}$

$BOR_{OUT} = A'.B + B.BOR_{IN} + A'.BOR_{IN}$

LOGIC DIAGRAM OF FULL SUBTRACTOR:

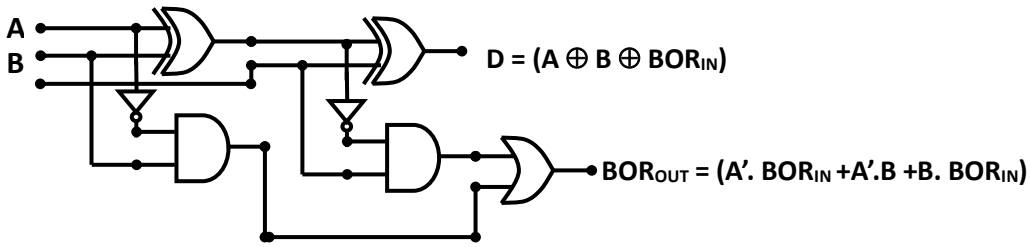

Figure 2.5.1 Full Subtractor

2.6 LOOK AHEAD CARRY GENERATOR:

Consider the full adder circuit,

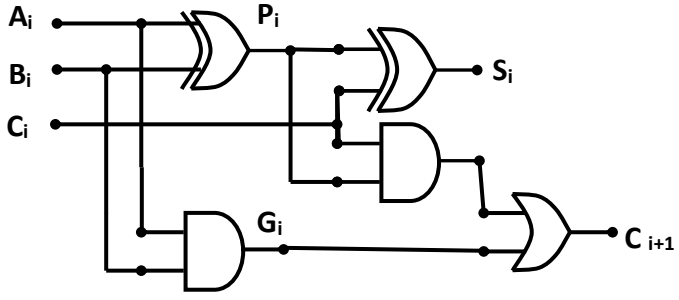

Figure 2.6.1 Full Adder

Where, G_i is carry generate and produces the carry when Ai and Bi are 1, regardless of input

carry. P_i is carry propagate, term associated with propagation of carry from C_i to C_{i+1} .
From above circuit, we define two new binary variables:

$$P_i = A_i \oplus B_i \quad \text{and} \quad G_i = A_i \,.\, B_i$$

Output sum and carry is expressed as:

$$S_i = P_i \oplus C_i \quad \textbf{and} \quad C_{i+1} = G_i + P_i \,.\, C_i$$

Now writing the boolean function for the carry output of each stage and substituting for each C_i its value from the previous equations , we get:

$$C_1 = G_0 + P_0 \,.\, C_0$$

$$C_2 = G_1 + P_1 \,.\, C_1 = G_1 + P_1 \,.\, (G_0 + P_0 \,.\, C_0) = G_1 + P_1 \,.G_0 + P_1 \,.\, P_0 \,.\, C_0$$

$$C_3 = G_2 + P_2 \,.\, C_2 = G_2 + P_2 \,.\, (G_1 + P_1 \,.G_0 + P_1 \,.\, P_0 \,.\, C_0) = G_2 + P_2 \,.\, G_1 + P_2 \,.\, P_1 \,.G_0 + P_2 \,.\, P_1 \,.\, P_0 \,.\, C_0$$

From the above equation it is noted that C_3 does not have to wait for C_2 and C_1 to propagate; in fact C_3 is propagated at the same time as C_2 and C_1 . Hence the carry's are propagated on the same time so no carry propagation delay occurs. Logic diagram of look ahead carry generator is shown below.

Logic diagram of Look Ahead Carry Generator:

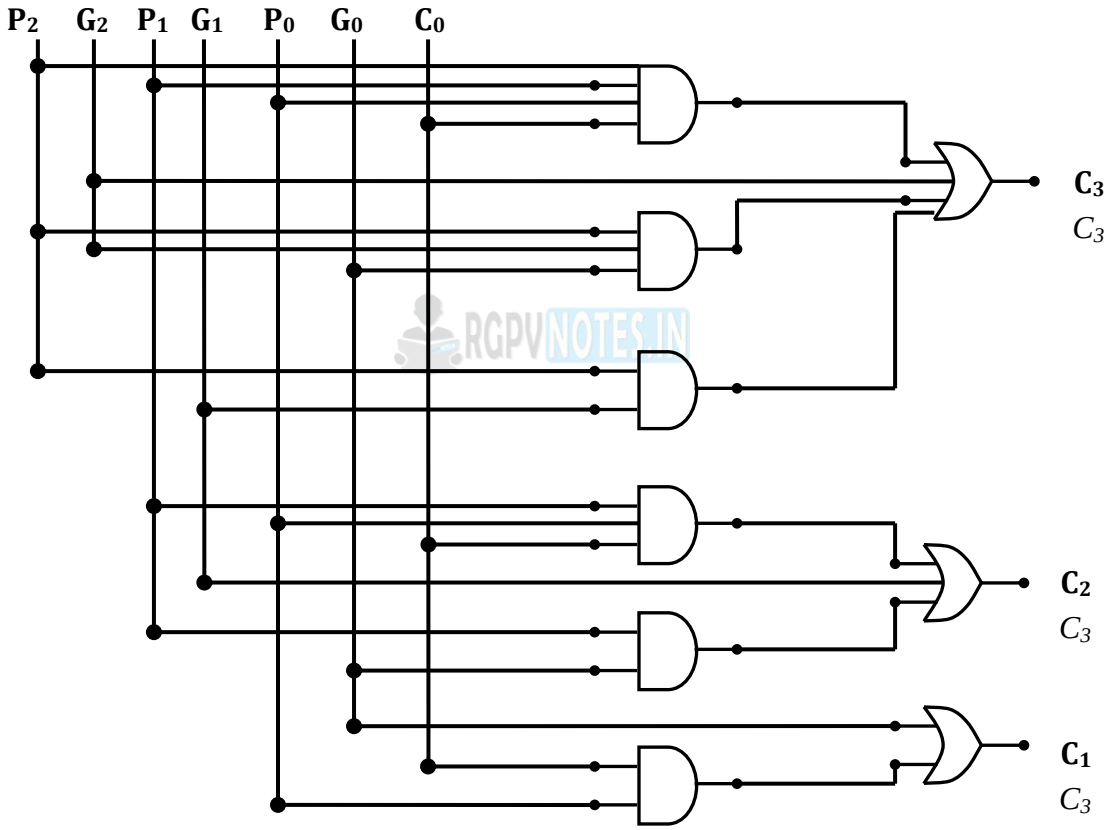

Figure 2.6.2 Look Ahead Carry Generator

2.7 BINARY CODED DECIMAL ADDER (BCD ADDER):

In BCD addition, 2-BCD digits are added in parallel, which produces a sum. If the sum is less than or equal to 9 , we get the valid BCD sum or if the sum is greater than 9 (non valid BCD sum) then we have to add 6(0110) to the sum to get the valid BCD sum. So, a logical circuit that performs the BCD addition is a BCD adder. A BCD adder is a circuit that adds two BCD digits in parallel and produces a sum digit also in BCD.

Design of BCD adder:

In BCD, each input digit does not exceeds 9, so the output sum cannot be greater than 9+9+1=19, the 1 in sum being an input carry. Suppose, we apply 2-BCD digits to a 4-bit adder. The adder form the sum in binary an produce a result that may range from 0-19. These binary

numbers are listed in table below:

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

From the table, when binary sum is less than or equal to 9(1001), corresponding BCD number is identical or valid and no conversion is needed. When binary sum is greater than 9, we obtain a non-valid BCD representation. So to get a valid BCD representation, we add 6(0110) to the binary sum and also produces an output carry as required.

The condition for correction and output carry can be expressed by a Boolean function:

$$C = K + Z_8 \cdot Z_4 + Z_8 \cdot Z_2$$

Block diagram of BCD Adder:

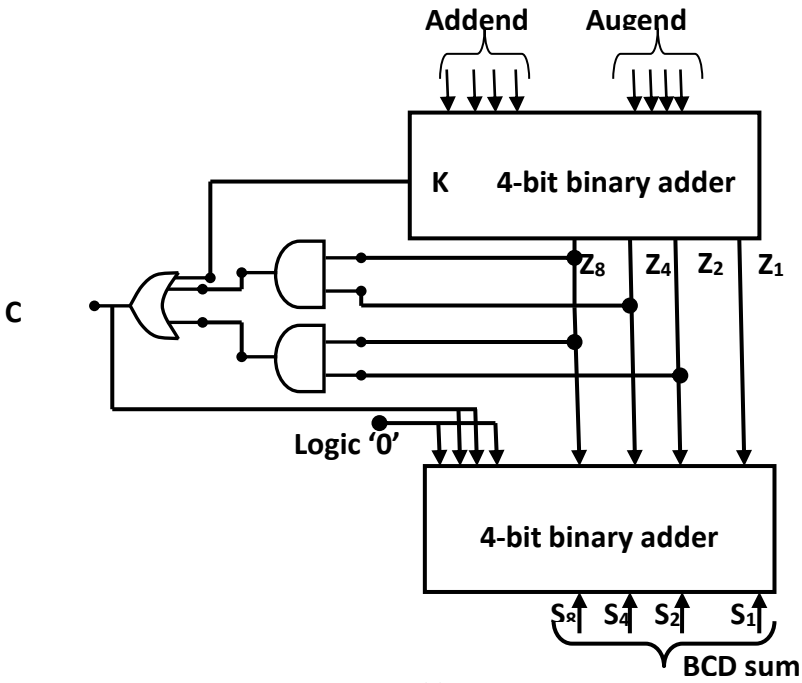

Figure 2.7.1 BCD Adder

From the block diagram of BCD adder, the binary numbers are labeled by symbols K,

Z8,Z4,Z2,Z1 where K is the carry, and the subscript under the letter Z represents the weight assigned to the bits. Letter C represent the output carry. The carry from the bottom binary adder can be ignored, since it supplies information already available at the output-carry terminal (C).

2.8 SERIES AND PARALLEL ADDER (4-Bit Binary Parallel Adder) :

A binary parallel adder produces a sum of 2-binary numbers in parallel. It consists of full adders connected in cascade, with the output carry from one full adder connected to the input carry of the next full adder. An 4-bit parallel adder requires 4-full adders. So, to design an n-bit binary parallel adder, it requires n full adders. Below figure 01 shows the interconnection of 4-full adder circuits to provide a 4-bit binary parallel adder.

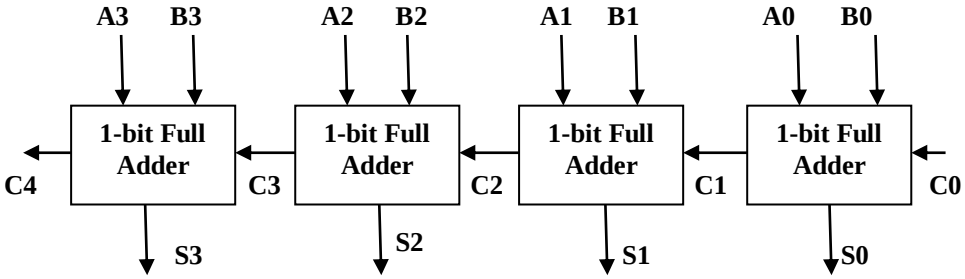

Figure 2.8.1 4-Bit Binary Parallel Adder

From figure 2.8.1, the augend bits of A (A_0,A_1,A_2,A_3) and the addend bits of B (B_0,B_1,B_2,B_3) are designated by subscript numbers from right to left, with subscript 1 denoting the lower order bit. The carries (C_0,C_1,C_2,C_3) are connected in chain through full adders. The input carry to the full adder is C_0 and output carry is C_4. The S (S_0,S_1,S_2,S_3) outputs generates the required sum bits.

Example: Consider **A = 1011** and **B = 0011**

Input carry	0 **(C_3)**	1 **(C_2)**	1 **(C_1)**	0 **(C_0)**	**C_i**
Augend	1 **(A_3)**	0 **(A_2)**	1 **(A_1)**	1 **(A_0)**	**A_i**
Addend	0 **(B_3)**	0 **(B_2)**	1 **(B_1)**	1 **(B_0)**	**B_i**
Sum	1 **(S_3)**	1 **(S_2)**	1 **(S_1)**	0 **(S_0)**	**S_i**
Output carry	0 **(C_4)**	0 **(C_3)**	1 **(C_2)**	1 **(C_1)**	**C_{i+1}**

Subscrip “i” represent i = 0,1,2,3.

In parallel adder, input carry C_0 in the least significant position must be 0.

Carry Propagation In Binary Parallel Adder: Consider the figure 2.8.1, a 4-bit binary parallel adder. The inputs A_3 and B_3 reach a steady state value as soon as inputs signals are applied to the adder. But input carry C_3 does not settle to is final steady state value until C_2 is in its steady state value. Simillarly, C_2 has to wait for C_1. Thus only after the carry propagates through all the stages will the last output S_3 and C_4 settle to its final steady state value. Thus to get the corrected output, carry has to propagate on time, if not the outputs will not be correct. So, the carry propagation time is the limiting factor.

The solution for reducing the carry propagation delay time is 1) to employ the faster gates with reduced delays. But physical circuits have a limit. 2) to increase the equipment complexity in such a way that the carry delay time is reduced. 3) the most widely used technique employs the principle of look ahead carry generator.

2.9 MULTIPLEXER - DEMULTIPLEXER:

2.9.1 MULTIPLEXERS (MUX):

Multiplexing means transforming a large number of information over a smaller number of channels or lines. A multiplexer is a combinational logic circuit that selects binary information from one of the input lines and directs it to a single output line. That's why multiplexer is also called data selector. Selection of a particular line is controlled by a set of selection input lines. A Multiplexer has 2^n input lines and "n" selection lines, whose bit combination determine which input is selected.

Block Diagram of Multiplexer:

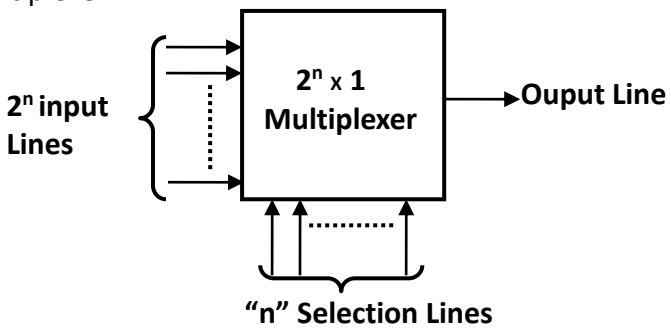

Figure 2.9.1.1 Multiplexer

4 x 1 Multiplexer:

A 4x 1 Multiplexer has 4-input lines ,1-output line and 2-selection lines.

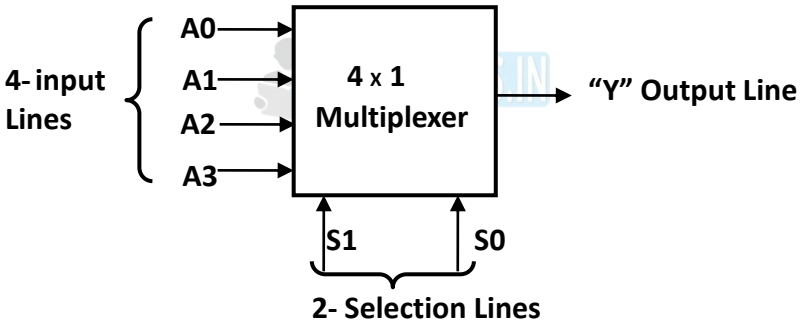

Figure 2.9.1.2 Block diagram of 4x1 Multiplexer

From the block diagram of 4x1 multiplexer, there are 4- input lines (A0,A1,A2,A3), 2-selection lines (S0 and S1) and 1-output line (Y).

Truth Table:

Input Selection Lines		Output Line
S1	**S0**	**Y**
0	0	A0
0	1	A1
1	0	A2
1	1	A3

According the truth table,

1.When selection line S0=S1=0; input line A0 is connected with output Y.

2.When selection line S0=1 and S1=0; input line A1 is connected with output Y.

3.When selection line S0= 0 and S1=1; input line A2 is connected with output Y.

4.When selection line S0=S1=1; input line A3 is connected with output Y.

Output expression: Y = A0.S0'.S1' + A1.S0.S1' + A2.S0'.S1 + A3.S0.S1

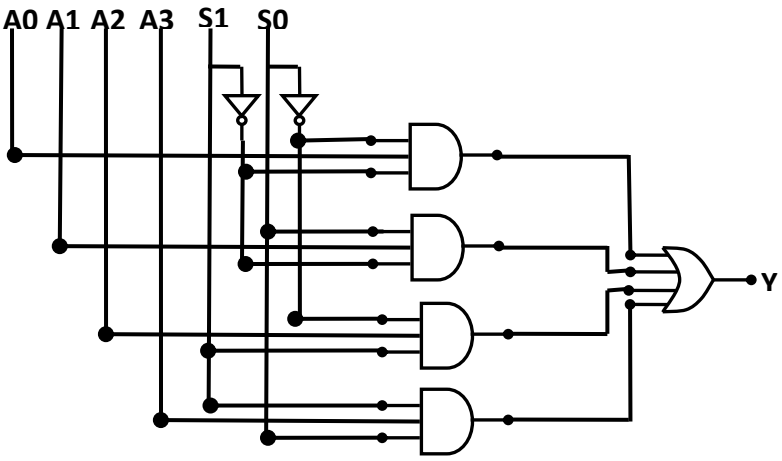

Figure 2.9.1.3 Logic diagram of 4x1 MUX

Simillarly, Multiplexer 2x1, 8x1, 16x1, 32x1 and so on can be designed.

2.9.2 DEMULTIPLEXERS (DEMUX):

A demultiplexer is a combinational logic circuit that receives information on single input line and transmits this information on one of 2^n possible output lines. The selection of a specific output line is controlled by the bit values of "n" selection lines.

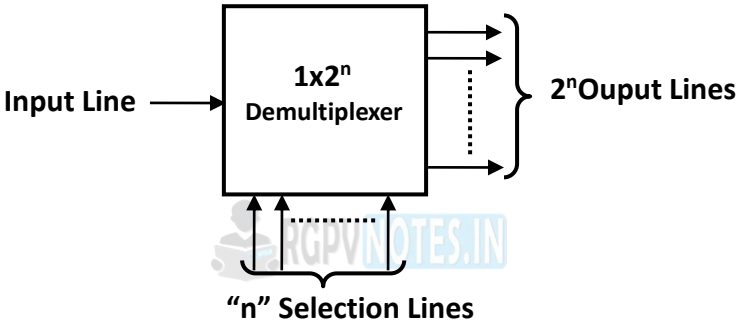

Figure 2.9.2.1 Block Diagram of Demultiplexer

1 x 4 Demultiplexer:

A 1 x 4 Demultiplexer has 1-input line ,4-output lines and 2-selection lines.

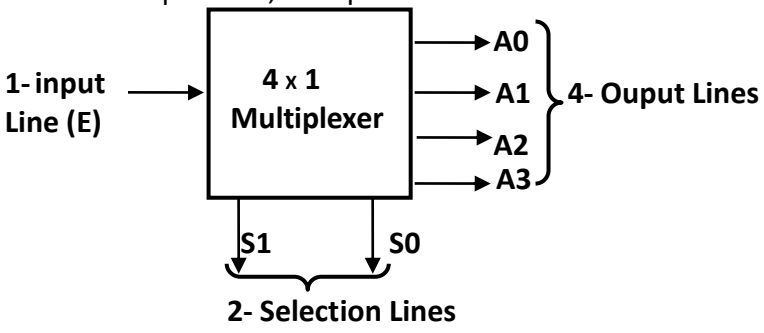

Figure 2.9.2.2 Block diagram of 1 x 4 Demultiplexer

From the block diagram of 1 x 4 Demultiplexer, there is 1- input line (E) ,2-selection lines (S0 and S1) and 4-output line (A0,A1,A2,A3).

Truth Table:

Input	Input Selection Lines		Output Line
	S1	S0	
E	0	0	A0 = E
E	0	1	A1 = E
E	1	0	A2 = E
E	1	1	A3 = E

According the truth table,

1.When selection line S0=S1=0; input line E is connected with output A0.
2.When selection line S0=1 and S1=0; input line E is connected with output A1.
3.When selection line S0= 0 and S1=1; input line E is connected with output A2.
4.When selection line S0=S1=1; input line E is connected with output A3.

Output expression: A0 =E.S0'.S1'; A1=E.S0.S1' ; A2=E.S0'.S1 ; A3=E.S0.S1

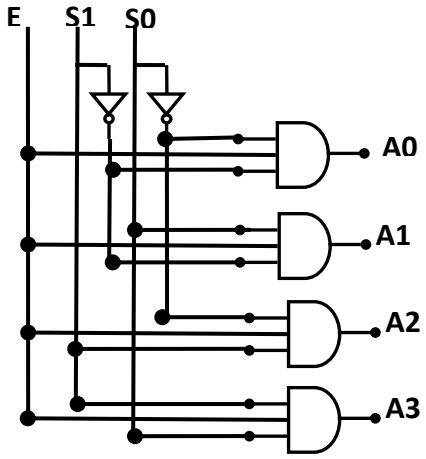

Figure 2.9.2.3 Logic diagram of 1 x 4 DEMUX

2.10 ENCODER DECODER:

2.10.1 ENCODER:

A combinational logic circuit that produces the reverse operation of a decoder. Encoder has 2^n(or less) input lines and 'n' output lines. The output lines generate the binary code for 2^n input lines.

8 to 3 Line Encoder (Octal to Binary Encoder):

Octal to binary encoder consists of eight inputs, one for each of the eight digits, and 3-outputs that generate the corresponding binary number.

Truth table-

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

Output "X"is 1 for octal digits: 4,5,6,7. So, $\mathbf{X = D_4 + D_5 + D_6 + D_7}$
Output "Y"is 1 for octal digits: 2,3,6,7. So, $\mathbf{Y = D_2 + D_3 + D_6 + D_7}$
Output "Z"is 1 for octal digits: 1,3,5,7. So, $\mathbf{Z = D_1 + D_5 + D_3 + D_7}$

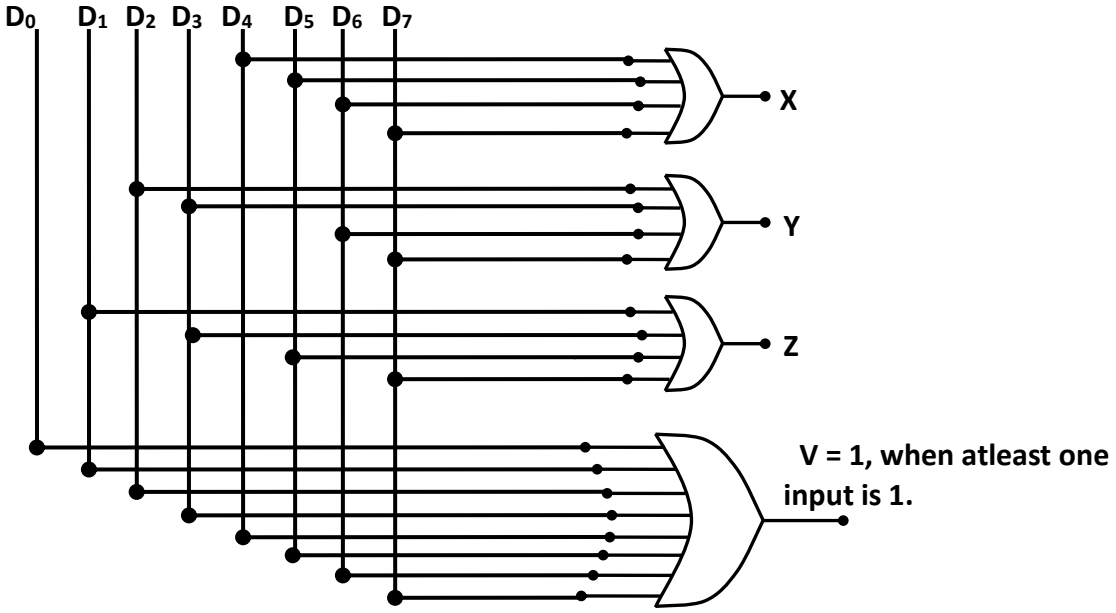

Figure 2.10.1.1:Logic diagram of 8 to 3 line encoder

The encoder in figure, assumes that only one input line can be equal to 1 at any time. Note that, circuit has 8-inputs i.e 2^8 = 256 possible input combinations and only 8 of these combinations have any meaning. The other input combinations are don't care condition. The output V in figure is to indicate the fact that all inputs are not 0's, as shown in figure.
Simillarly we can design priority encoder, decimal to BCD encoder, hexadecimal to binary encoder.

PRIORITY ENCODER:
Priority encoder establish an input priority to ensure that only the highest priority input line is encoded.Thus, if priority is given to an input with a highest subscript number over one with a lower subscript number, then the encoded output will be of highest subscript number.
Truth Table of priority encoder:

Inputs				Outputs	
D_3	D_2	D_1	D_0	A	B
1	X	X	X	1	1
0	1	X	X	1	0
0	0	1	X	0	1
0	0	0	1	0	0

" X" indicates don't care condition. i.e. "X" can be 1 or 0.
From the truth table, input D_3 is with the highest priority than inputs D_2, D_1 and Do. If D_3 = 1, and D_2, D_1 and Do are don't care , then output will be (11).
Simillarly, if D_3 = 0, D_2 = 1 and D_1 and Do are don't care , then output will be (10).
If D_3 = 0, D_2 = 0, D_1 = 1 and Do is don't care , then output will be (01).
If D_3 = 0, D_2 = 0, D_1 = 0 and Do = 1 , then output will be (00).
For example: Consider the truth table of 8 to 3 Line Encoder (Octal to Binary Encoder). Suppose, input D_5 has highest priority than D_2, then if both D_2 and D_5 are logic-1 simultaneously, the output will be 101 because D_5 has highest priorty over D_2 .

Design a 4 line to 2 line priority encoder. Include an output E to indicate that atleast one input is a 1.
Solution: Priority given to the input with highest subscript number i.e. input D_3
Truth table of 4 line to 2 line priority encoder:

Inputs				Outputs		
D_3	D_2	D_1	D_0	A	B	E
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	1	1
0	0	1	1	0	1	1
0	1	0	0	1	0	1
0	1	0	1	1	0	1
0	1	1	0	1	0	1
0	1	1	1	1	0	1
1	0	0	0	1	1	1
1	0	0	1	1	1	1
1	0	1	0	1	1	1
1	0	1	1	1	1	1
1	1	0	0	1	1	1
1	1	0	1	1	1	1
1	1	1	0	1	1	1
1	1	1	1	1	1	1

Simplification of output expression using K-map, we get:

$A = D_3 + D_2$; $B = D_3 + D_2'.D_1$; $E = D_3 + D_2 + D_1 + D_0$

Logic Diagram of 4 line to 2 line priority encoder:

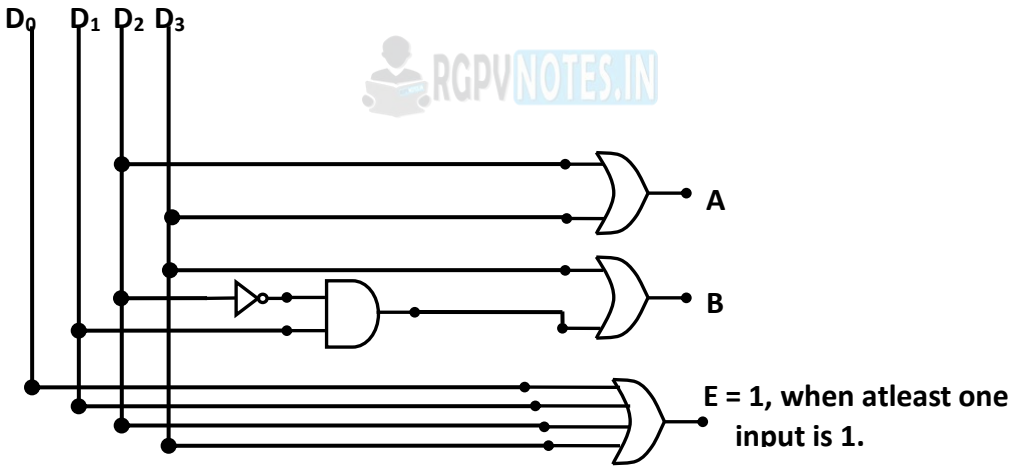

Figure 2.10.1.2: Logic diagram of 4 to 2 line priority encoder

2.10.2 DECODER:

A decoder is a combinational logic circuit that converts binary information from n-input lines to a maximum of 2^n unique output lines. If the n-bit decided information has unused or don't care combinations, the decoder output will have lass than 2^n outputs.

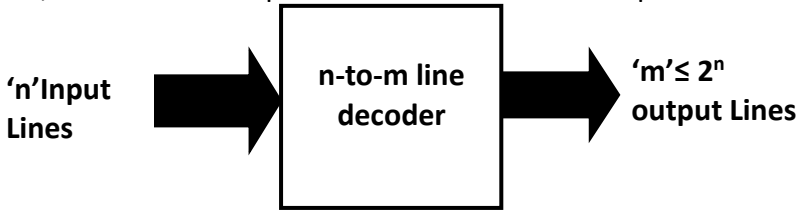

Figure 2.10.2.1 Block diagram of Decoder

3 to 8 Line Decoder:

3-inputs are decoded into eight output, each output representing one of the minterms of the 3-input variables.

Inputs			Outputs							
X	Y	Z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Truth Table of 3 to 8 line decoder

The output line whose value is equal to 1 represents the minterm equivalent of the binary number available in the input lines.

Output expressions: $D_0 = X'.Y'.Z'$; $D_1 = X'.Y'.Z$; $D_2 = X'.Y.Z'$; $D_3 = X'.Y.Z$
$D_4 = X.Y'.Z'$; $D_5 = X.Y'.Z$; $D_6 = X.Y.Z'$; $D_7 = X.Y.Z$

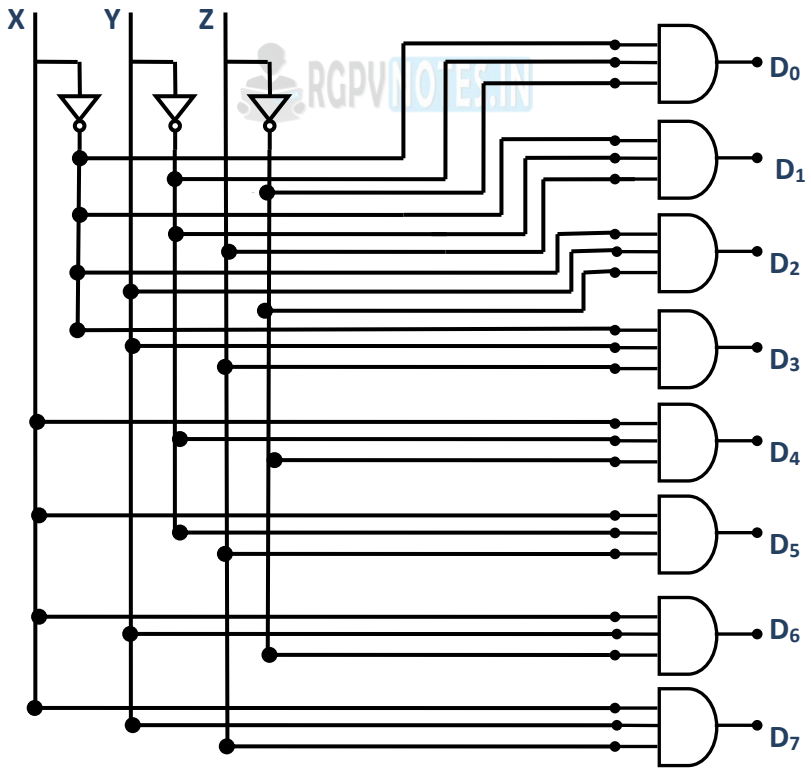

Figure 2.10.2.2: Logic diagram of 3 to 8 Line Decoder

Simillarly, 2 to 4 line decoder and 4 to 16 line decoder can be designed.

2.11 ARITHMETIC CIRCUITS:

CODE CONVERTER:

1.Binary to Gray code converter:

Truth Table:

Inputs Binary				Outputs Gray			
B3	B2	B1	B0	G3	G2	G1	G0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

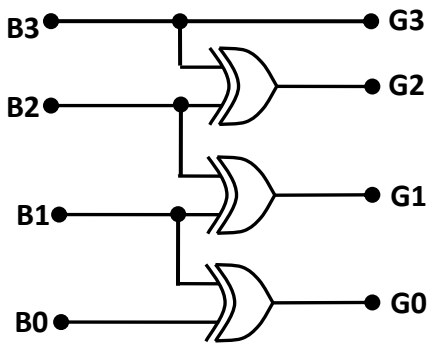

Figure 2.11.1.1: Binary to Gray Code Converter

Output expression is determine using K-map method, we get:

G3 = B3. **G1 = B2 ⊕ B1.**

G2 = B3 ⊕ B2. **G0 = B1 ⊕ B0.**

2. Gray to Binary code converter:

Truth Table:

Inputs Gray				Outputs Binary			
G3	G2	G1	G0	B3	B2	B1	B0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	1	0	0	1	0
0	0	1	0	0	0	1	1
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
0	1	0	1	0	1	1	0
0	1	0	0	0	1	1	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	1	1	0	1	0
1	1	1	0	1	0	1	1
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	0	0	1	1	1	1	0
1	0	0	0	1	1	1	1

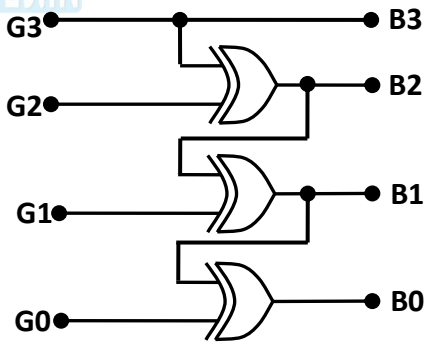

Figure 2.11.1.2 Gray to Binary Code Converter

Output expression is determine using K-map method, we get:

B3 = G3. **B1 = B2 ⊕ G1.**

B2 = G3 ⊕ G2. **B0 = B1 ⊕ G0.**

2.12 ARITHMETIC LOGIC UNIT (ALU)

ALU is a multioperation, combinational logic digital function. It can perform a set of basic arithmetic and a set of logical operations. ALU has a set of selection lines to select a particular operation. The selection lines are decoded within the ALU.

Figure 2.12.01 shows the block diagram of 4-bit ALU.

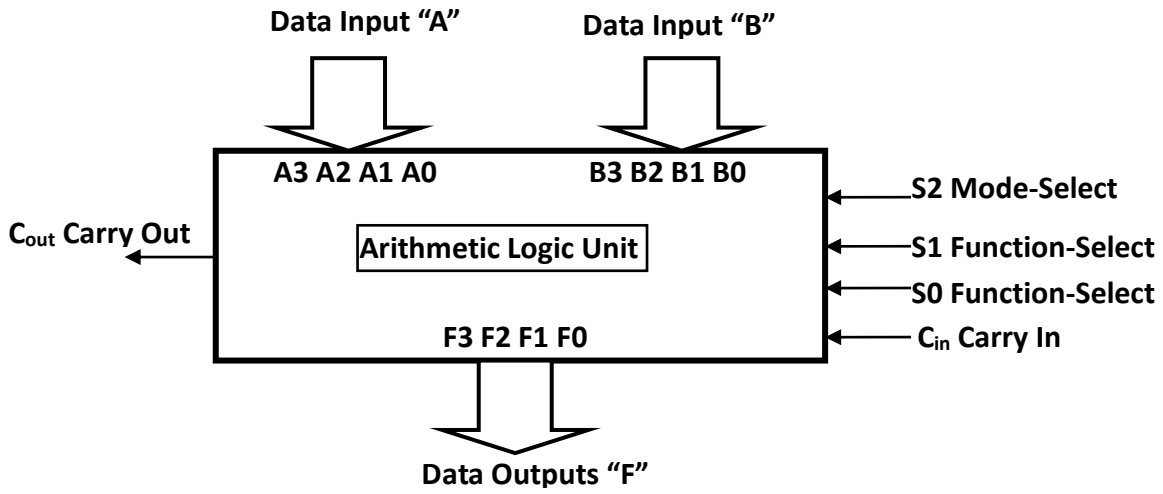

Figure 2.12.01 4 Bit Arithmatic & Logic Unit

Above block diagram, is of a 4-bit ALU. The 4-data inpus from register A are combined with 4-data inputs from register B to generate an operation at the F outputs. Mode select input S2 will specify whether the operations performed are arithmetic or logic. Two function-select S0 and S1 specify the particular arithmetic or logical operation to be generated. With 3-selection variables, it is possible to specify 4-arithmetic operations (with S2 in one state) and 4-logical operations (with S2 in another state). Input and output carries have meaning only during arithmetic operation.

DESIGN OF ALU:

Carried out in three stages: (i) Design of arithmetic section (ii) Design of logic section (iii) Finally the arithmetic section will be modified so that it can perform both arithmetic and logic operations.

DESIGN OF ARITHMETIC CIRCUIT:

The basic component of arithmetic section of ALU is parallel adder. Parallel adder is constructed with anumber of full adder circuits connected in cascade.
A 4-bit arithmetic circuit that performs eight arithmetic operations is shown in figure below.
The arithmetic operations implemented in the arithmetic circuit are listed in the below table.
The values to the Y inputs to the full adder circuits are a function of selection variables S1 and S0. Adding the value of Y to the value of A plus the C_{in} value gives the arithmetic operation.
The combinational circuit that inserted in each stage between external inputs A and B and the inputs of parallel adder X and Y, is a function of the arithmetic operations that are to be implemented. The combinational circuit will be different if the circuit generates different arithmetic operations.

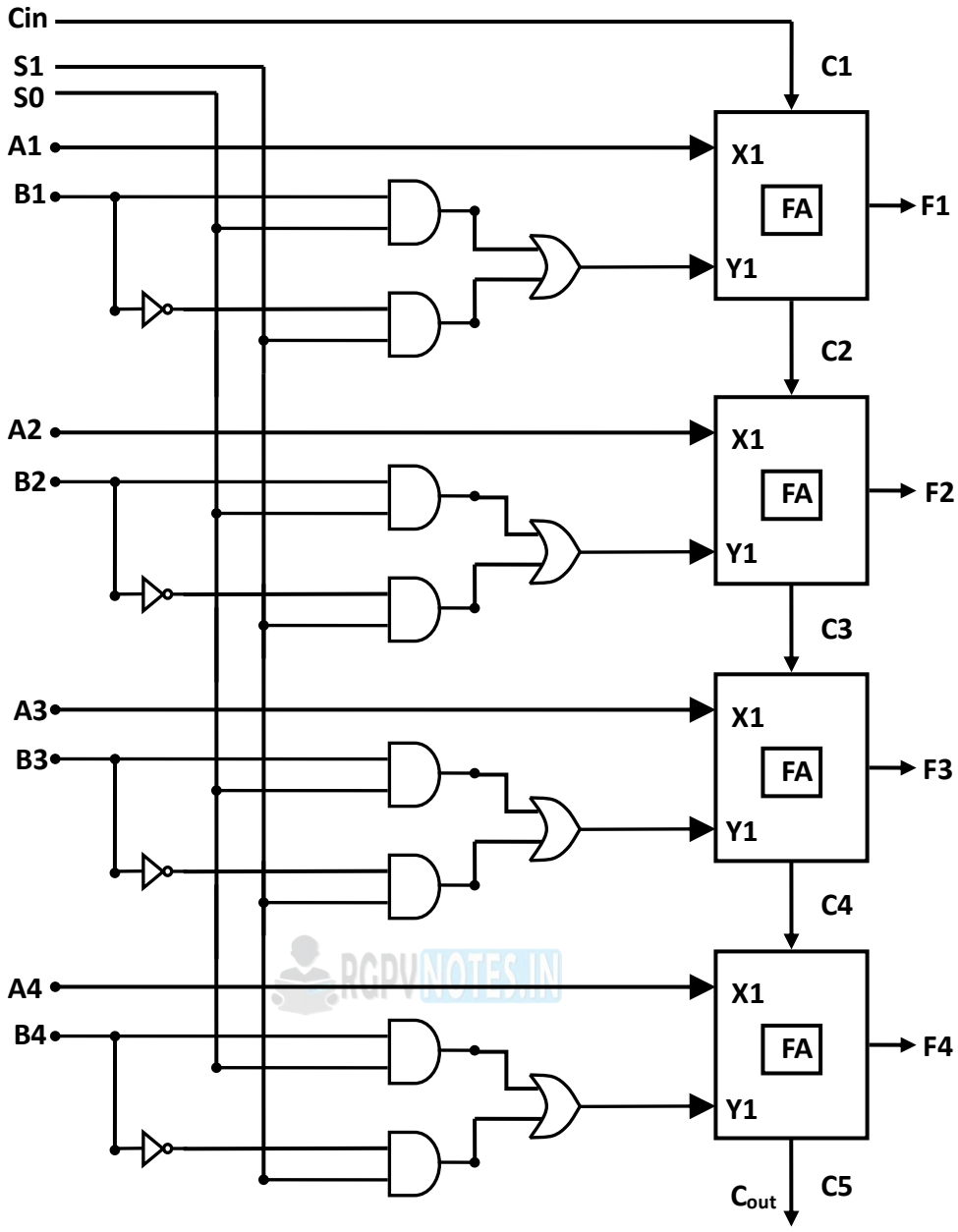

Figure 2.12.02: Logic Diagram of 4-Bit Arithmetic Circuit

Function Table for the arithmetic circuit:

Function Select			Y equals	Output equals	Function
S1	S0	C_{in}			
0	0	0	0	F=A	Transfer *A*
0	0	1	0	F= A+1	Increment *A*
0	1	0	B	F= A+B	Add *B* to *A*
0	1	1	B	F= A+B+1	Add *B* to *A* plus 1
1	0	0	B’	F= A+B’	Add 1’s complement of *B* to *A*
1	0	1	B’	F= A+B’+1	Add 2’s complement of *B* to *A*
1	1	0	All 1’s	F = A− 1	Decrement *A*
1	1	1	All 1’s	F =A	Transfer *A*

****End of Unit 2****

RGPVNOTES.IN

We hope you find these notes useful.

You can get previous year question papers at https://qp.rgpvnotes.in .

If you have any queries or you want to submit your study notes please write us at rgpvnotes.in@gmail.com

LIKE & FOLLOW US ON FACEBOOK
facebook.com/rgpvnotes.in