DIGITAL INTEGRATED CIRCUITS ANALYSIS

ELECTRONICS & COMMUNICATION ENGINEERING

unit -6

HAND NOTES

BY:
P.RAJESH M.TECH.,
ASSISTANT PROFFESOR
EMAIL: rajesh.crit@gmail.com
ph: 9989786119
9985786099

CRIT COLLEGE OF ENGG & TECHNOLOGY

ANANTAPURAMU

/ [illegible]

*) **Barrel Shifter :-**

A barrel shifter is a Combinational logic circuit with n-data i/p and n-data o/p's, and a set of Control i/p which specifies how to ~~Control~~ ~~i/p~~ ~~that~~ shift the data b/w i/p & o/p.
Barrel shifter in C.P.U will specify the direction of shift i.e., whether it is left or right shift. It also specify type of shift.
1) Circular shift
2) Arithematic shift
3) Logical shift.

1) Explain the operation of barrel shifter and write a VHDL Code for Left and right circular shift.

(or)

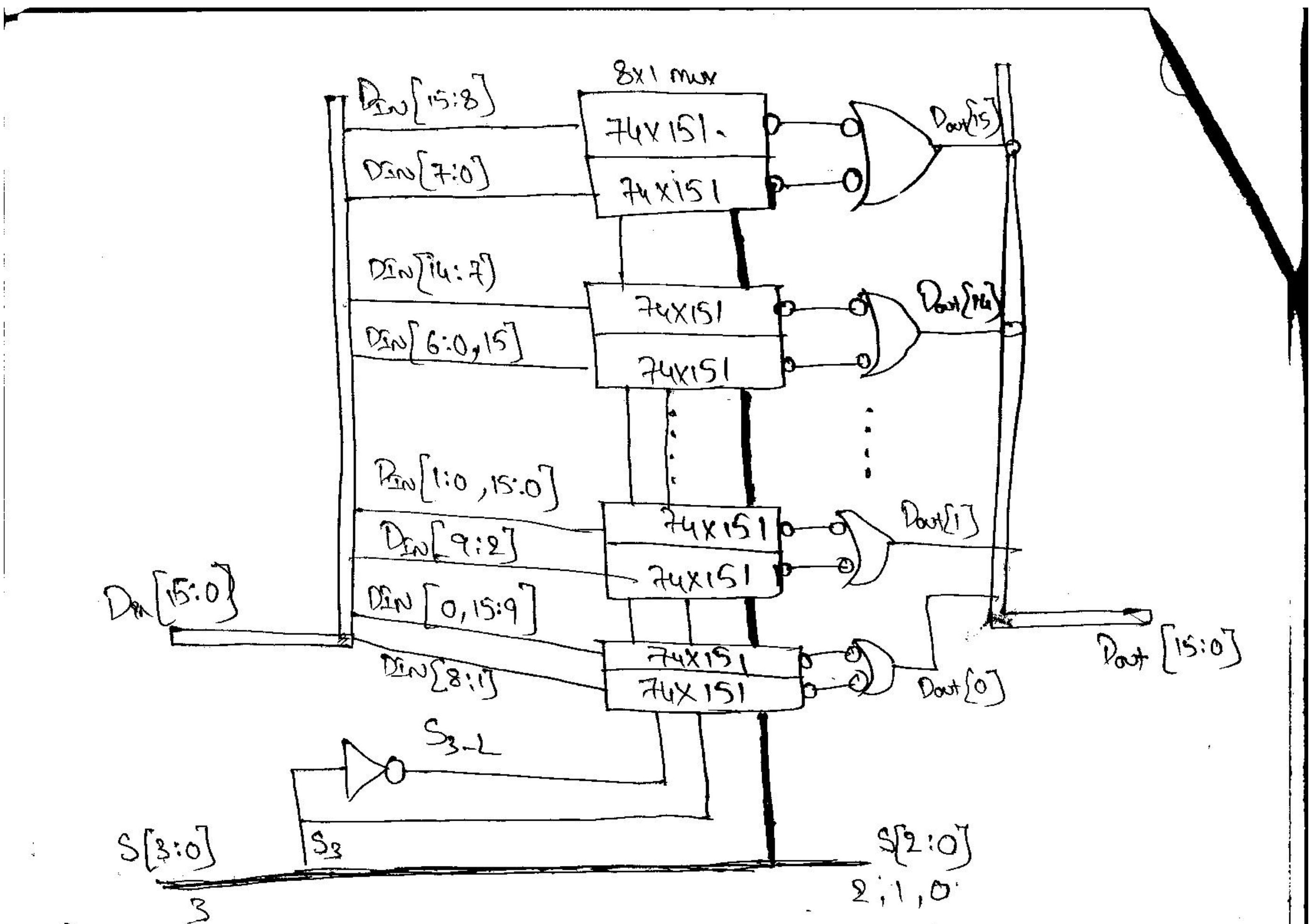
A 16-bit barrel shifter is a Combinational logic circuit with 16 data i/p, 16-data o/p and 4-Control i/p. The i/p word is rotated by a no: of 16 bit Positions specified by Control bits.

(or)

Sol:- A barrel shift is a Combinational ckt which can rotate (or) shift data word by any number of bits in a single operation.
thus the o/p of one mux is Connected to i/p of next mux
The no: of multiplexer required is $n \times \log_2(n)$
∴ for 16. barrel shifter

$$= 16 \times \log_2(16)$$
$$= 16 \times 4 = 64$$

It requires 64 multiplexers.



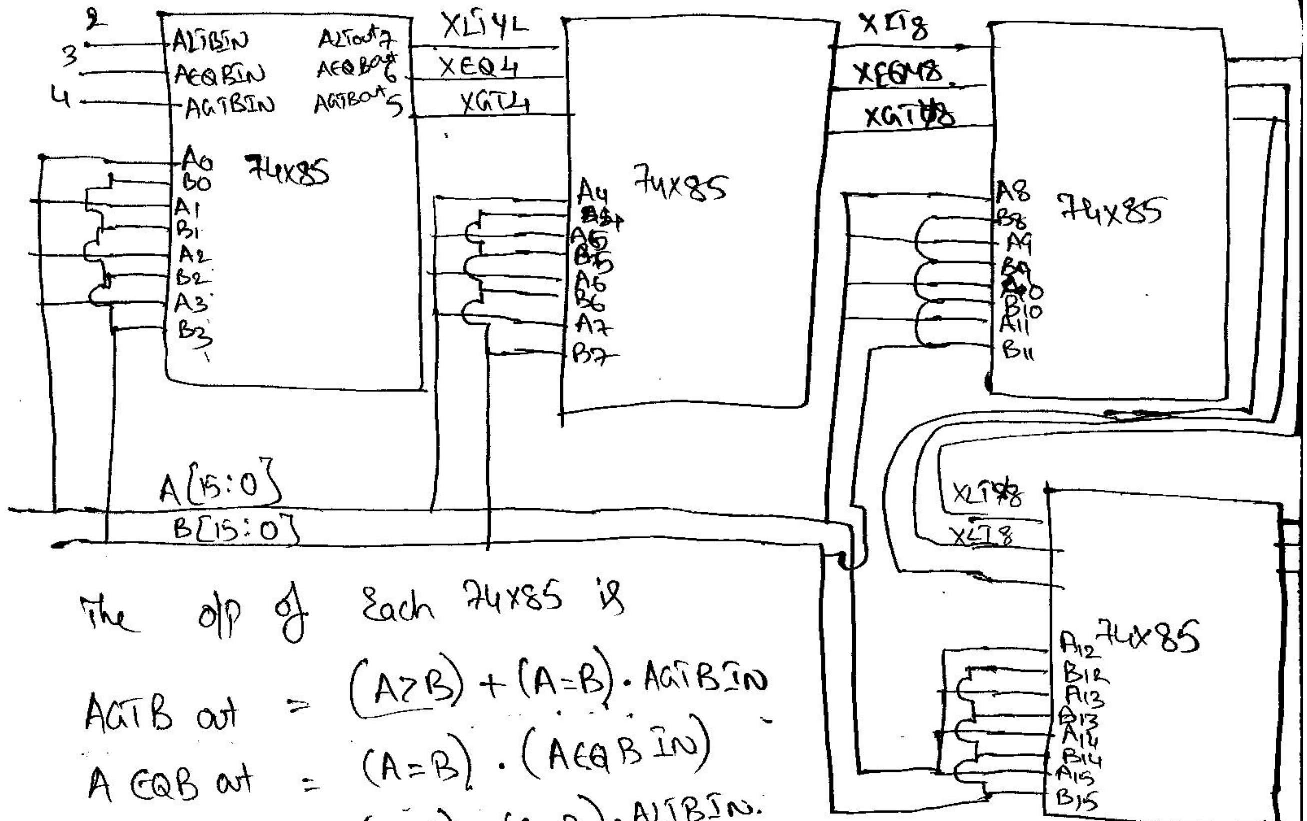
Program:-

```
library ieee;
use ieee.std-logic-1164.ALL;
Entity barrel shifter;
Port( DIN : in std-logic-vector(15 downto 0);
   S : in std-logic-vector (3:0); ---> shift amount, 0-15
  DIR : in std-logic;     ------> shift direction 0⇒L ; 1⇒R
                                                 left       Right
  Dout : out std-logic-vector(15 downto 0));
  End barrel shifter;
Architecture behavioral of barrel shifter is
 begin
 Process (DIN, S, DIR)
 variable  x, y, z : std-logic-vector (15 downto 0);
 Variable . CTRL0, CTRL1, CTRL2, CTRL3 : std-logic-vector (1 downto 0);
 begin.
```

```
  RLO : S(0) & DIR;
 TRL1 : S(1) & DIR;
 CTRL2 : S(2) & DIR;
 CTRL3 : S(3) & DIR;
Case CTRLO is
 when "00" | "01" => X : = Din;
 when "10" => X : Din (14 downto 0) & Din (15);
  when "11" => X : Din (0) & Din (15 downto 1);
  when others => null;
End Case;
 Case CTRL1 is
   when "00" | "01" => Y : = X;
   when "10" => Y : = X (13 downto 0) & X (15 downto 14);
   when "11" => Y : X (1 downto 0) & X (15 downto 2);
    when others => null;
  End Case;

 Case CTRL2 is
  when "00" | "01" => Z : Y;
 when "10" => Z : = Y (11 downto 0) & Y (15 downto 12);
 when "11" => Z : = Y (3 downto 0) & Y (15 downto 4);
  when others => null;
  End Case;
  End Process;
 End behavioral;
```

Design 16-bit Comparator using 74x85 IC's



The o/p of Each 74x85 is

$AGTB\ out = (A>B) + (A=B) \cdot AGTBIN$

$A\ EQB\ out = (A=B) \cdot (AEQB\ IN)$

$A\ LTB\ out = (A<B) + (A=B) \cdot ALTBIN.$

In above Eq 'A' stands for A_3 to A_0 and

B stands B_3 to B_0

$\underline{A>B}$; $\underline{A=B}$; $\underline{A<B}$

Floating - Point Encoder :-

The simple floating point encoder is used to represent a 11-bit no: interms of a floating value. in to 7-bit as 3-bit exponent & 4-bit mantissa
the condition for simple floating point encoder is

$$B = M \cdot 2^{E} + T$$

Where M - four bit mantissa T is truncation error
E - three bit exponent

Eg:- Let us represent 11010110110 in terms of floating point

ie $1101 \cdot 2^7 + 0110110$

Explanation:- For representing given 11-bit binary no:- in terms of floating point we have to search for first '1' and from the first '1' copy four no:- in sequence in given then it is called Mantissa.

from above Eg:- 1101 is mantissa. then remaining no:- of bits as power of 2 i.e 2^7

After that add remaining no:- that were present excluding mantissa.

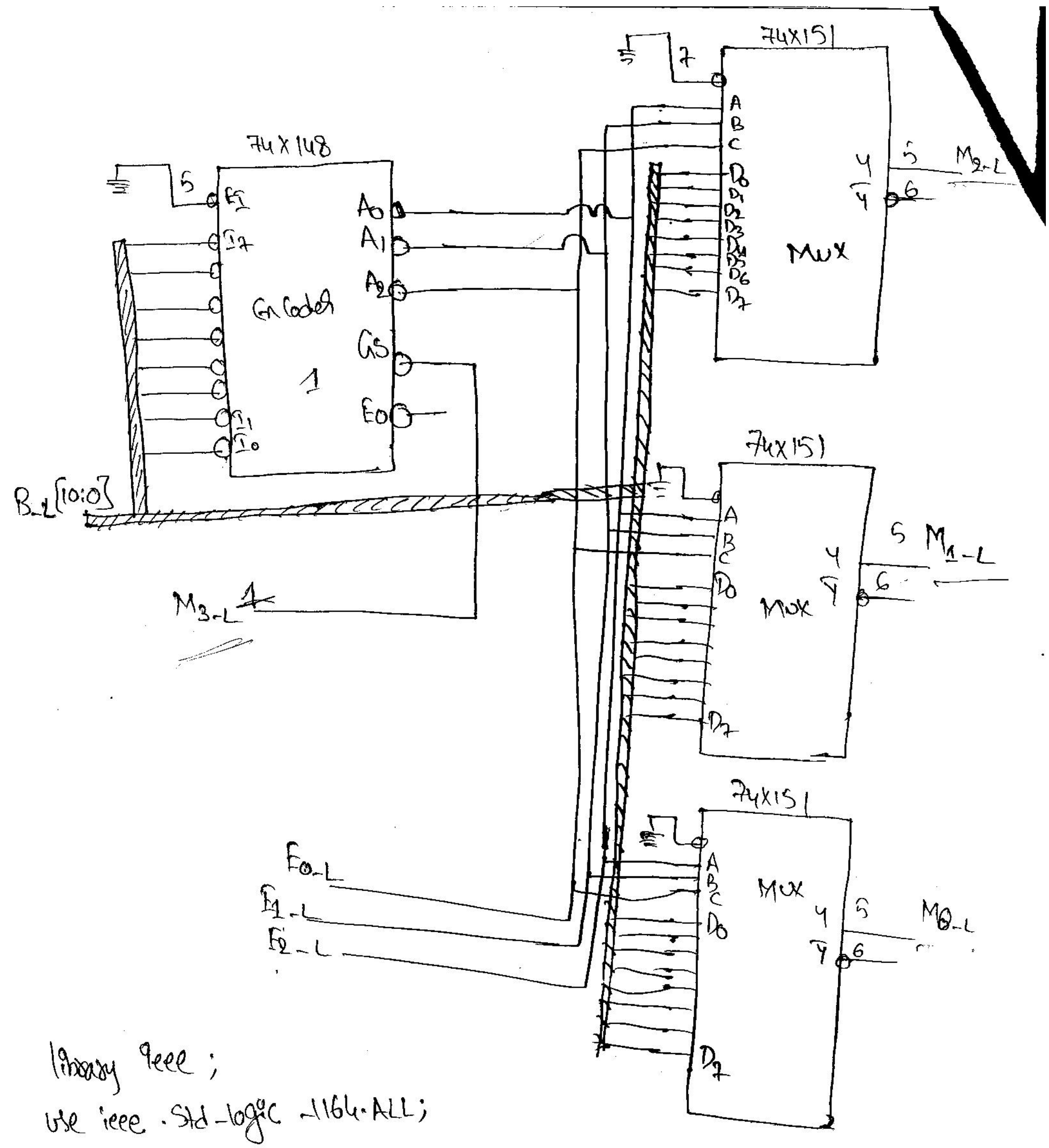
$\therefore$ Value becomes $\underset{M}{\underline{1101}} \cdot \underset{2^E}{\underline{2^7}} + \underset{T}{\underline{0110110}}$

This is simple floating point.

This process is carried out by using a circuit known as simple floating point encoder. This circuit consists of 74x151 ICs three (8x1) MUX and one (74x148) IC encoder.

$M_3 M_2 M_1 M_0$ Truncation E_2 E_1 E_0

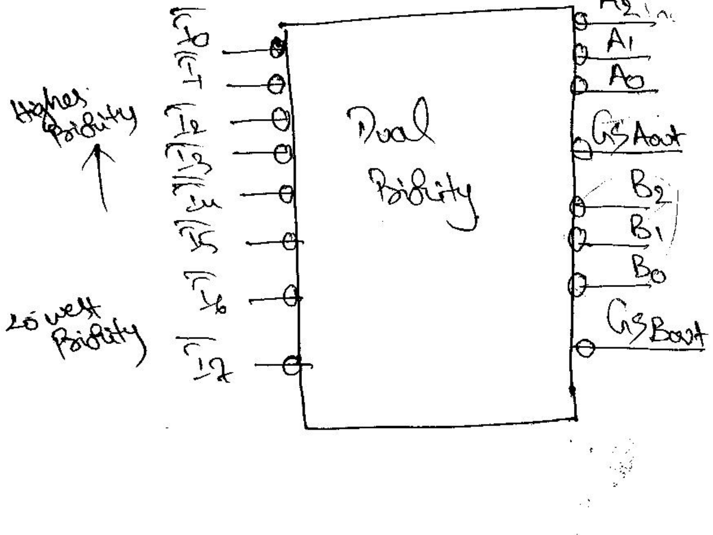
$\underset{\text{mantissa}}{\underline{1101}} \Rightarrow 1\ 0\ 1\ 1 + 0110110 \times\ 1\ 1\ 1$



```
library ieee ;
use ieee . std-logic -1164.ALL;
Entity FPrec is
  Port ( B : in std -logic-vector (10 downto 0); -> Bits 11-bits given
      M : out std -logic -vector (3 downto 0); -> floating Point mantissa
      E : out std- logic - vector (2 down to 0)); -> floating Point Exponent
  End FPrec;
Architecture behavioral of FPrec is
```

```
egin
Process (B)
begin
 If    B(10) = '1' then   M<= B(10 downto 7) ; E<= "111";
 Elsif B(9)  = '1' then   M<= B(9 downto 6) ;  E<= "110";
 Elsif B(8)  = '1' then   M<= B(8 downto 5) ;  E<= "101";
 Elsif B(7)  = '1' then   M<= B(7 downto 4) ;  E<= "100";
 Elsif B(6)  = '1' then   M<= B(6 downto 3) ;  E<= "011";
 Elsif B(5)  = '1' then   M<= B (5 downto 2) ; E<= "010";
 Elsif B(4)  = '1' then   M<= B (4 downto 1) ; E<= "001";
 Else    M<= B ( 3 downto 0) ; E <= "000";
  Else
  End if ;
 End Process ;
 End behavioral;
```

*) Dual * Priority * Encoder :- (for 8 i/p signal i/p)



$\bar{I}_0$	$\bar{I}_1$	$\bar{I}_2$	$\bar{I}_3$	$\bar{I}_4$	$\bar{I}_5$	$\bar{I}_6$	$\bar{I}_7$	A_2	A_1	A_0	A_{out}	B_2	B_1	B_0	
X	X	X	X	X	X	X	1	1	1	1	1	1	1	1	1
X	X	X	X	X	X	1	0	1	1	0	1	1	1	0	1
X	X	X	X	X	1	0	0	1	0	1	1	1	0	1	1
X	X	X	X	1	0	0	0	1	0	0	1	1	0	0	1
X	X	X	X	0	0	0	0	0	1	1	1	0	1	1	1
X	X	1	0	0	0	0	0	0	1	0	1	0	1	0	1
X	1	0	0	0	0	0	0	0	0	1	1	0	0	1	1
1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

VHDL Program:-

```
library ieee ;
use ieee. Std-logic-1164.All ;
Entity Dual Priority is
Port ( I : in std-logic-vector (0 to 7);
   A, B : out std-logic-vector (2 downto 0);
   Aout , Bout : Buffer std-logic);
 End Dual Priority ;
Architecture ~~do~~ behavioral of dual priority is
begin Process(I, A, Aout, Bout) begin.
  if   I(0) = '1' then  A <= "000" ; Aout <= '1';
 Elsif I(1) = '1' then  A <= "001" ; Aout <= '1';
 Elseif I(2) = '1' then  A <= "010" ; Aout <= '1';
 Elseif I(3) = '1' then  A <= "011" ; Aout <= '1';
 Elsif I(4) = '1' then  A <= "100" ; Aout <= '1';
```

```
    Elsif I(5) = '1' then A <= "101" ; Aout <= '1';
    Elsif I(6) = '1' then A <= "110" ; Aout <= '1';
    Elsif I(7) = '1' then A <= "111" ; Aout <= '1';
    Else  A <= "000" ; Aout <= '0' ;
    End if ;
If  I(1) = '1' And A/= "001" then B <= "001"; Bout <= '1';
Elsif I(2) = '1' And A/= "010" then B <= "010" ; Bout <= '1';
 Elsif I(3) = '1' And A/= "011" then B <= "011" ; Bout <= '1';
 Elsif I(4) = '1' And A/= "100" then B <= "100" ; Bout <= '1';
 Elsif I(5) = '1' And A/= "101" then B <= "101" ; Bout <= '1';
 Elsif I(6) = '1' And A/= "110" then B <= "110" ; Bout <= '1';
 Elsif I(7) = '1' And A/= "111" then B <= "111" ; Bout <= '1';
 Else  B <= "000" ; Bout <= '0';
 End if ;
 End Process;
 End behavioral; //.
```