CODE :

Top Module

```
--LIBRARIES TO USE
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

--ENTITY OF 8 bit Floating Point Multiplier top module
entity floating_point_mul is
  port( num1 : in unsigned(7 downto 0);
        num2 : in unsigned(7 downto 0);
      output: out unsigned(7 downto 0);
      op_flag:out std_logic);
end floating_point_mul;

--ARCHITECTURE
architecture Behavioral of floating_point_mul is


--COMPONENT OF MULTIPLIER
component mul_8bit is
    Port ( a : in  unsigned(7 downto 0);
           b : in  unsigned(7 downto 0);
           prod : out  unsigned(7 downto 0);
           sflag : out  std_logic);

end component;

--MAIN BODY
begin

--FLOATING POINT MULTIPLIER
FLT_MUL_INS:  mul_8bit port map(a=>num1,b=>num2,prod=>output,sflag=>op_flag);

--END OF ARCHITECTURE
end Behavioral;
```

Multipler

```
----------------------------------Floating Point Multiplier
    ----------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity mul_8bit is
    Port ( a : in  unsigned(7 downto 0);
           b : in  unsigned(7 downto 0);
           prod : out  unsigned(7 downto 0);
           sflag : out  std_logic);
end mul_8bit;

architecture Behavioral of mul_8bit is

```

```
16 begin
17   process(a,b)
18   --declaring variable for each part of the floating point number
19   variable sign_a,sign_b,sign_z,svar: std_logic;
20   variable exp_a,exp_b,exp_z: unsigned(3 downto 0);
21   variable man_a,man_b: unsigned(4 downto 0);
22   variable man_z: unsigned(9 downto 0);
23   begin
24   --1)checking infinite value in inputs
25   if (a(6 downto 4)="111" or b(6 downto 4)="111") then
26   sflag <='0';
27   prod<="01110000";
28   else
29
30     --2)assigning value to each variable
31       sign_a:=a(7);
32       sign_b:=b(7);
33       exp_a(2 downto 0):=a(6 downto 4);
34       exp_a(3):='0';
35       exp_b(2 downto 0):=b(6 downto 4);
36       exp_b(3):='0';
37       man_a(3 downto 0):=a(3 downto 0);
38       man_b(3 downto 0):=b(3 downto 0);
39       man_a(4):='1';
40       man_b(4):='1';
41     --3)success if nothing lowers success flag
42       svar:='1';
43
44     --4)adding exponents
45       exp_z:=exp_a+exp_b;
46     -- subtracting bias
47       exp_z:=exp_z-"0011";
48     --5)checking overflow
49       if exp_z(3)='1' then
50         svar:='0';
51       end if;
52
53     --6)multiplying mantissas
54       man_z:=resize(man_a*man_b,10);
55
56     --7)normalising and shifting exp
57       if (man_z(9)='1') then
58         man_z:=man_z srl 1;
59         exp_z:=exp_z+"0001";
60       end if;
61
62     --8)raising overflow flag and lowering sucess flag
63       if (exp_z(3)='1') then
64         svar:='0';
65       end if;
66
67     -- 9)calculating sign of product
68       sign_z:=sign_a xor sign_b;
69
70     --10)Assigning final values to prod
71       prod(3 downto 0)<=man_z(7 downto 4);
72       prod(6 downto 4)<=exp_z(2 downto 0);
73       prod(7)<=sign_z;
74       sflag<=svar;
75     end if;
76     end process;
77 end Behavioral;
```