```
 1
 2  ------------------Exponent Subtractor(Used to adjust resultant exponent in case of Normalisation)------------
 3
 4  library IEEE;
 5  use IEEE.STD_LOGIC_1164.ALL;
 6  use IEEE.NUMERIC_STD.ALL;
 7
 8  entity ExpSubtractor is
 9      Port ( X : in  INTEGER := 0;
10             TempEr : in UNSIGNED(2 downto 0);
11             Er : buffer UNSIGNED (2 downto 0);
12             ShiftDirection : in STD_LOGIC);
13
14  end ExpSubtractor;
15
16  architecture Behavioral of ExpSubtractor is
17
18  begin
19  process(X,TempEr,ShiftDirection)
20     variable BitX : UNSIGNED (2 downto 0) := "000";
21     variable sum_ans : UNSIGNED (3 downto 0) := "0000";
22     variable carry_ans : UNSIGNED (4 downto 0) := "00000";
23     variable Ernew : UNSIGNED (3 downto 0) := "0000"; --1 bit extra for knowing the sign
24     variable Xnew : UNSIGNED (3 downto 0) := "0000";
25     variable TempOperand : UNSIGNED (3 downto 0) := "0000";
26     variable TempCarry : UNSIGNED (4 downto 0) := "00001";
27     begin
28        case X is
29           when 0 => BitX := "000";
30           when 1 => BitX := "001";
31           when 2 => BitX := "010";
32           when 3 => BitX := "011";
33           when 4 => BitX := "100";
34           when 5 => BitX := "101";
35           when 6 => BitX := "110";
36           when others => null;
37        end case;
38
39
40        if ShiftDirection = '0' then -- Shift is left, so subtract
41           carry_ans(0) := '1';
42           Ernew(2 downto 0) := TempEr; --remaining bits as usual
```

```
43            Xnew(2 downto 0) := BitX xor "111"; --2's complement of X. (Refer the rule of subtraction using 2's
    complement)
44
45            for i in 0 to 3 loop
46               sum_ans(i) := (Ernew(i) xor Xnew(i)) xor carry_ans(i);
47               carry_ans(i+1) := (Ernew(i) and Xnew(i)) or (Ernew(i) and carry_ans(i)) or (Xnew(i) and carry_ans(i));
48            end loop;
49
50            if sum_ans(3) = '1' then
51               Er <= sum_ans(2 downto 0); --Ans is +ve, so no change
52            else
53               sum_ans(2 downto 0) := sum_ans(2 downto 0) xor "111"; -- Ans is -ve, so ans is 2's complement of itself
54               for i in 0 to 3 loop
55                     TempCarry(i+1) := (sum_ans(i) and TempOperand(i)) or (sum_ans(i) and TempCarry(i)) or (TempOperand
    (i) and TempCarry(i));
56                     sum_ans(i) := (sum_ans(i) xor TempOperand(i)) xor TempCarry(i);
57               end loop; -- This part is to add 1 to sum_ans in order to complete the 2's complement
58               Er <= sum_ans(2 downto 0);
59            end if;
60            --Sign <= not sum_ans(3); -- Zero is +ve, One is -ve, But in program it comes opposite
61
62         else -- shift is right so add
63            Ernew(2 downto 0) := Temper;
64            Xnew(2 downto 0) := "001";
65            for i in 0 to 3 loop
66               sum_ans(i) := (Ernew(i) xor Xnew(i)) xor carry_ans(i);
67               carry_ans(i+1) := (Ernew(i) and Xnew(i)) or (Ernew(i) and carry_ans(i)) or (Xnew(i) and carry_ans(i));
68            end loop;
69            Er <= sum_ans(2 downto 0);
70         end if;
71      end process;
72   end Behavioral;
73
74
```