Floating Point Multiplier

By:

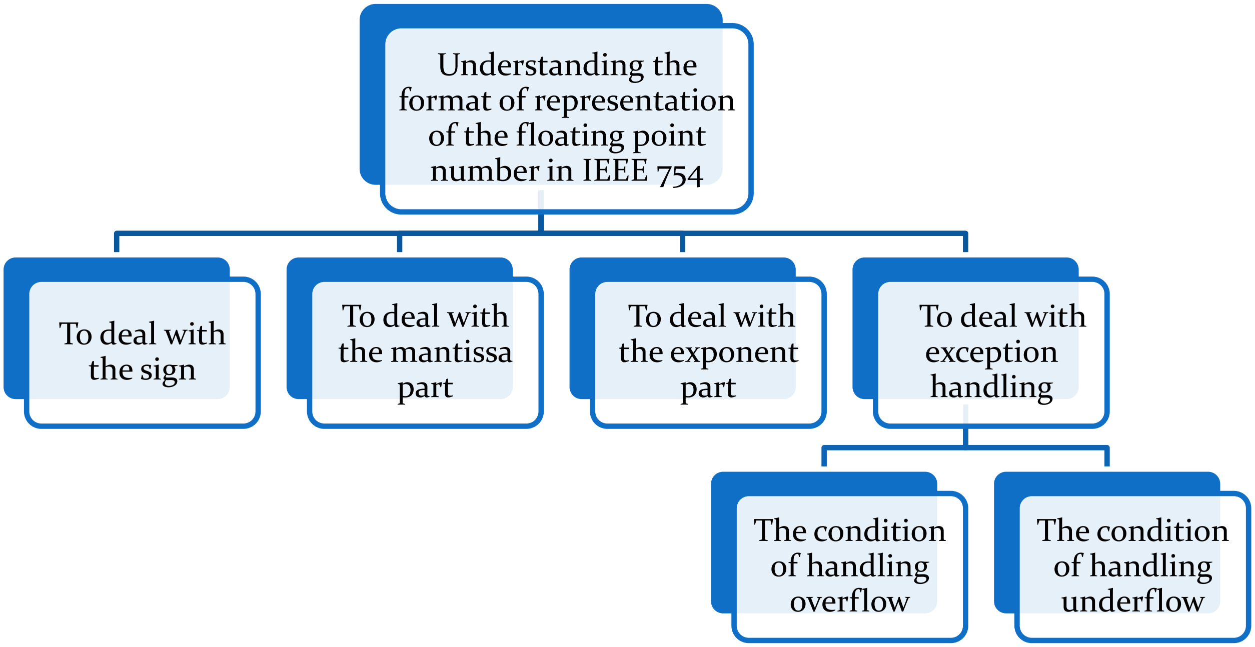
Aryaman Maheshwari (16114018)

Aviral Verma (16114019)

Hasanwala Faizal (16114030)

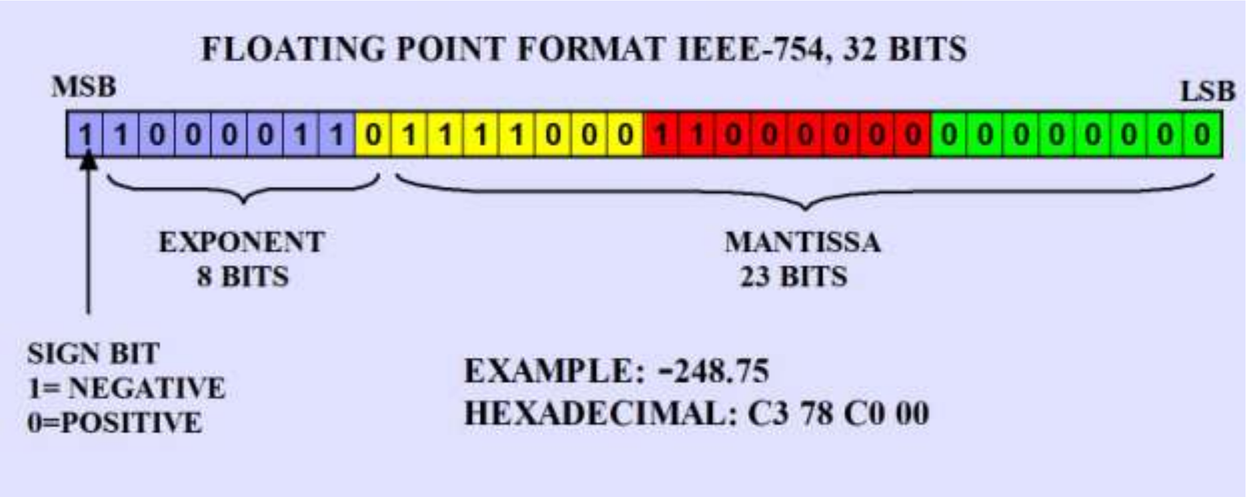
Being honest, It took us 36 hours of man hours, a lot of google and a whole lot of Xilinx, and last but not the least Microsoft office tools to build this report. ☺

Following is the flow chart of what we went learnt during the whole journey of making this project.

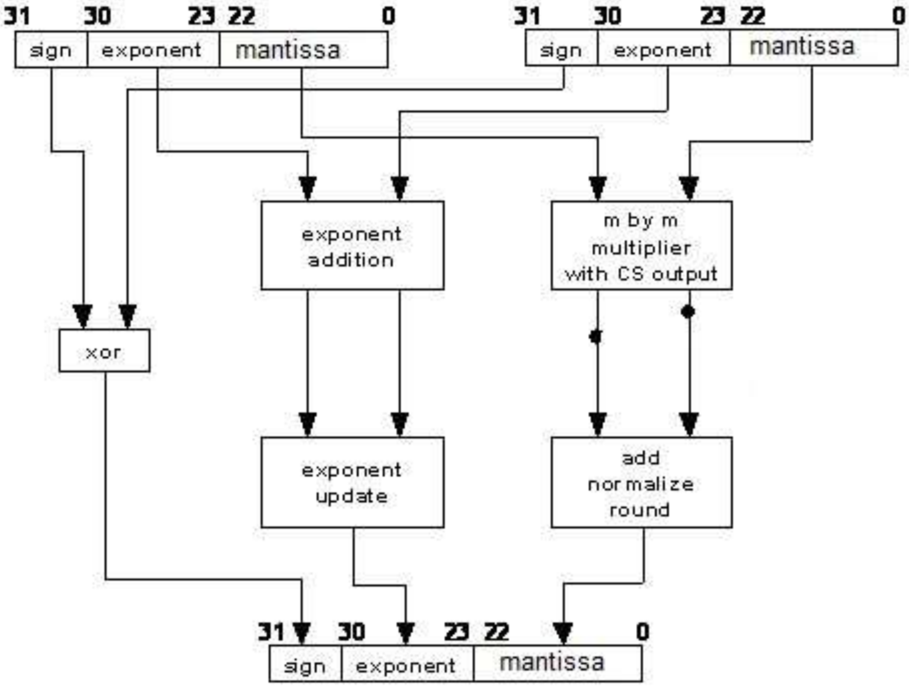


Understanding the format of representation of the floating point number in IEEE 754
To deal with the sign
To deal with the mantissa part
To deal with the exponent part
To deal with exception handling
The condition of handling overflow
The condition of handling underflow

1. Understanding the format of representation of the floating point number in IEEE 754



2.) Flow Chart



- To deal with the sign

```
z_sign := x_sign  xor  y_sign;
```

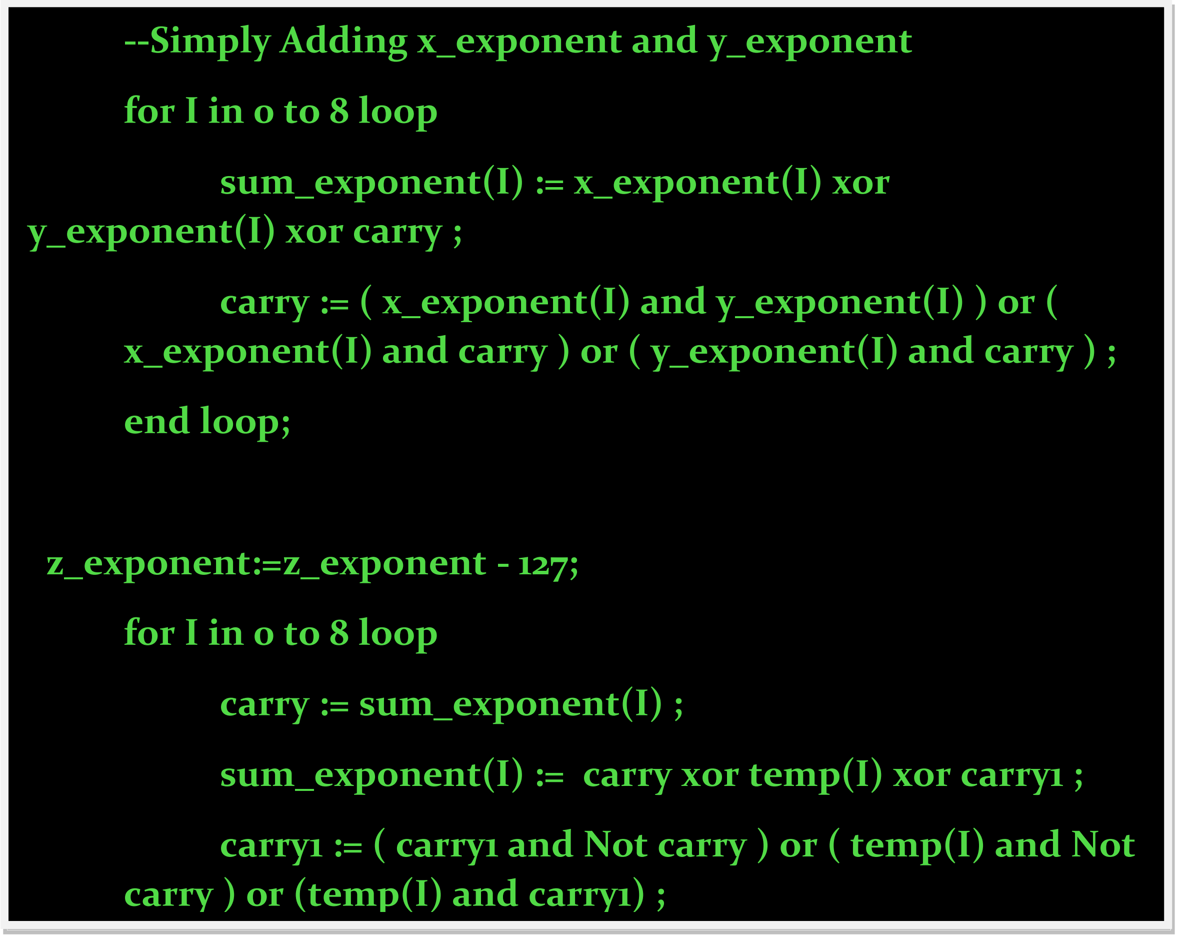
- To deal with the mantissa part

Following is the pseudo code to simplify the understanding of working at mantissa part.

```
24 bit of x * 24 bit of y = 48 bits of temp;
If( $48^{th}$ bit = 1):
    Exponent of z(“product”) ++;
    23 bits of Mantissa of z = (47 to 25)th bit
    temp;


If($48^{th}$ bit =0):
```

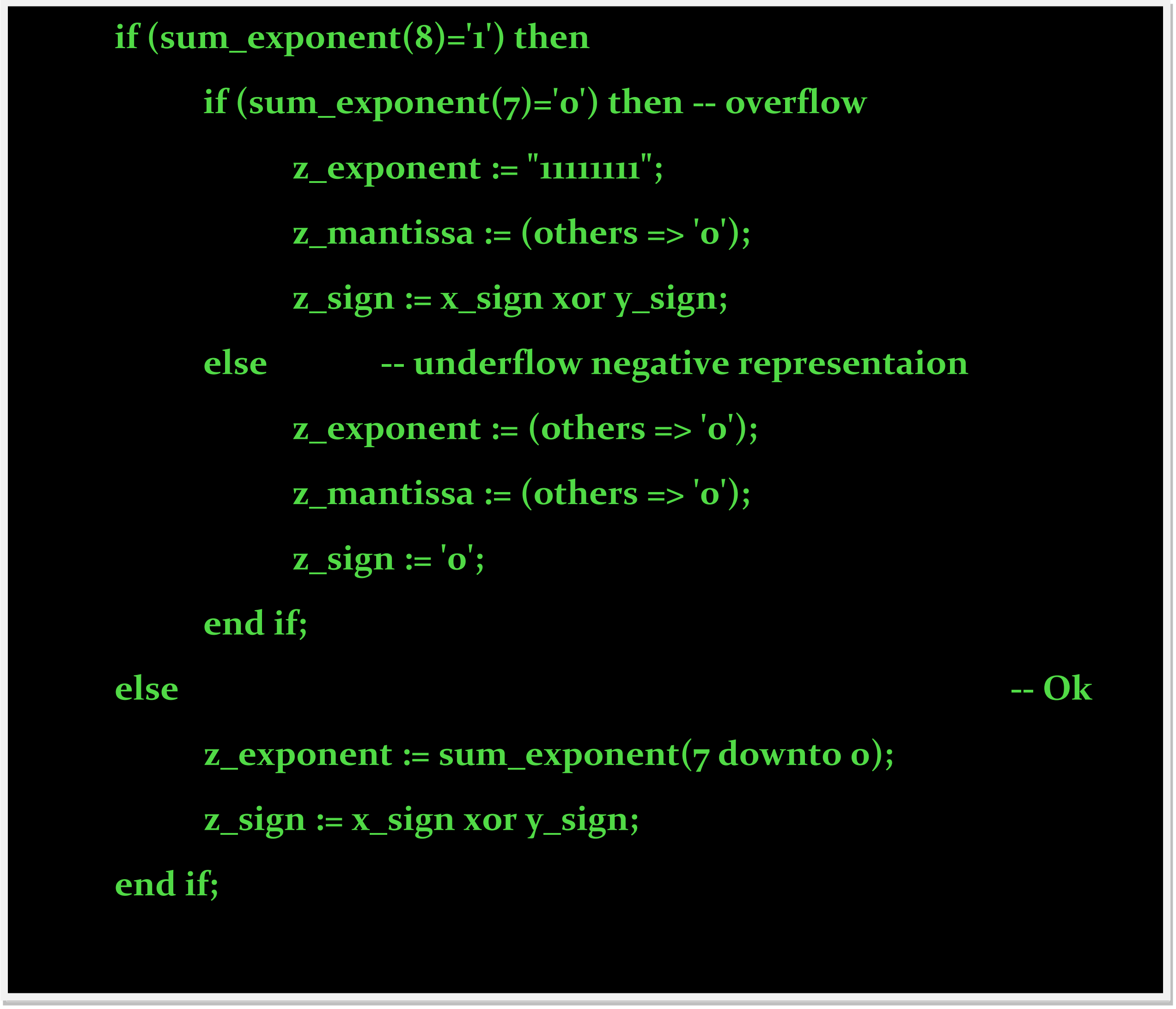
- To deal with the exponent part



```
        --Simply Adding x_exponent and y_exponent
        for I in 0 to 8 loop
                sum_exponent(I) := x_exponent(I) xor
y_exponent(I) xor carry ;
                carry := ( x_exponent(I) and y_exponent(I) ) or (
        x_exponent(I) and carry ) or ( y_exponent(I) and carry ) ;
        end loop;


  z_exponent:=z_exponent - 127;
        for I in 0 to 8 loop
                carry := sum_exponent(I) ;
                sum_exponent(I) :=  carry xor temp(I) xor carry1 ;
                carry1 := ( carry1 and Not carry ) or ( temp(I) and Not
        carry ) or (temp(I) and carry1) ;
```

- The condition of handling overflow and underflow



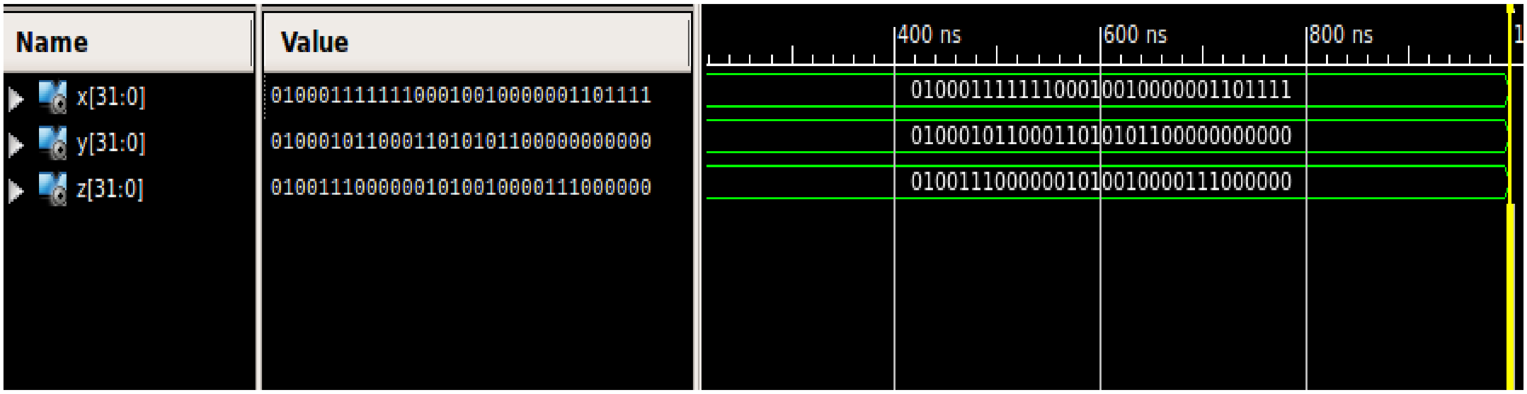
```
if (sum_exponent(8)='1') then
        if (sum_exponent(7)='0') then -- overflow
                z_exponent := "11111111";
                z_mantissa := (others => '0');
                z_sign := x_sign xor y_sign;
        else            -- underflow negative representaion
                z_exponent := (others => '0');
                z_mantissa := (others => '0');
                z_sign := '0';
        end if;
else                                                                        -- Ok
        z_exponent := sum_exponent(7 downto 0);
        z_sign := x_sign xor y_sign;
end if;
```

- Here is the test case simulated in the test bench of our project…

x<=”01000111111100010010000001101111” -- (123456.87)

y<=”01000101100011010101100000000000” -- (4523.0)

z<=”01001110000000101001000011100000” -- (558395423.01)



Name	Value
x[31:0]	01000111111100010010000001101111
y[31:0]	01000101100011010101100000000000
z[31:0]	01001110000000101001000011100000

Thanking you for giving us the opportunity to get started with VHDL!