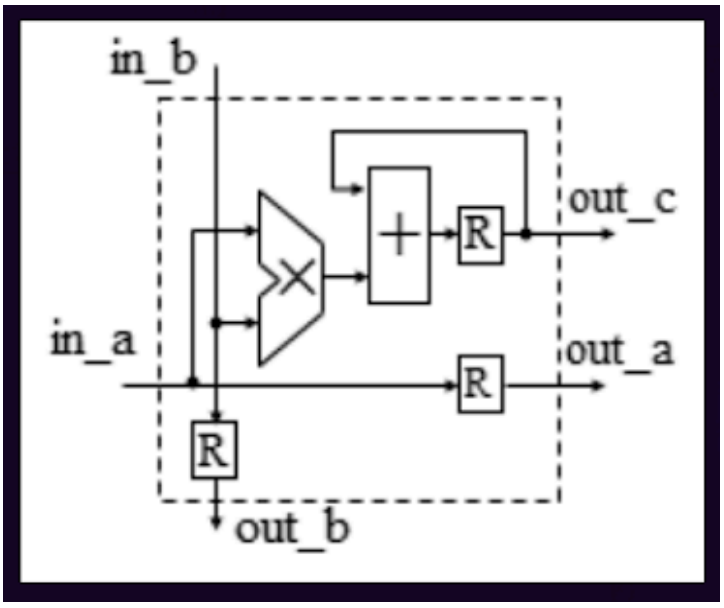
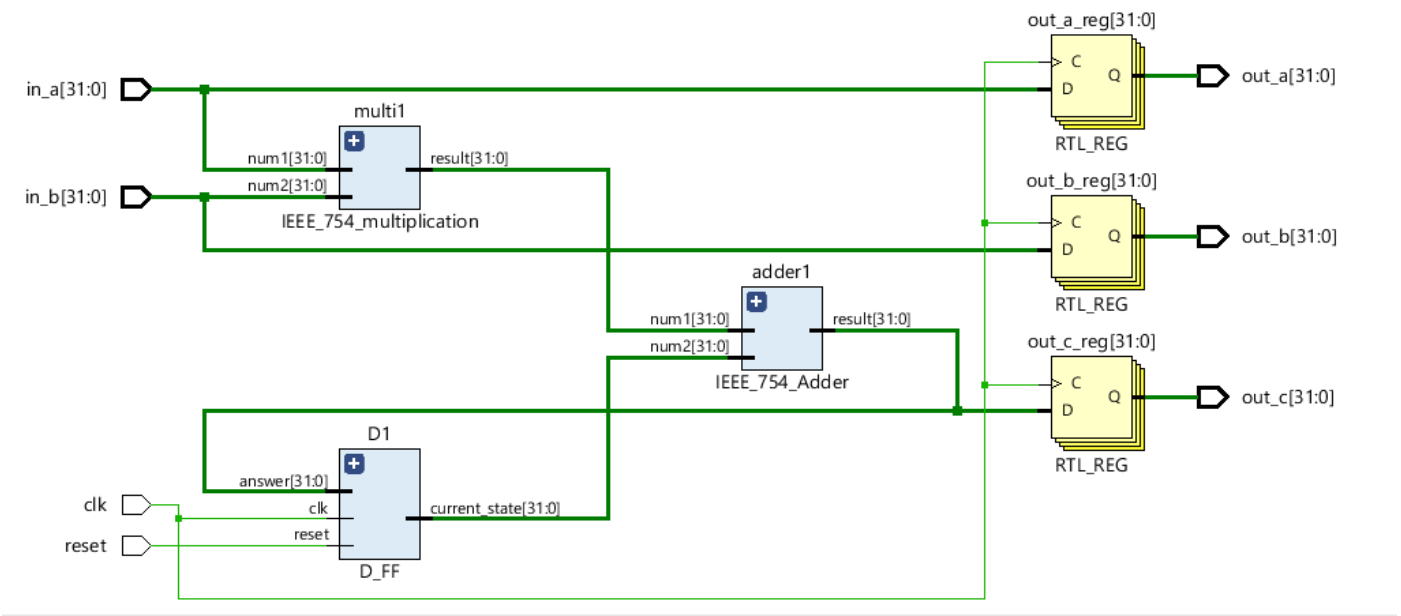
DESIGNING OF PROCESSING ELEMENT(PE)



Our Processing Element consists of Floating Point Multiplier, Floating point Adder and a D Flip Flop(register) .
The two inputs in_a and in_b are fed to the Floating Point Multiplier and their output is fed to the Floating Point Adder with the previous value of the Register.
Thus, this value is stored in register for upcoming operation.
And is outputted as out_c.

RTL Analysis of PE(elaborated design):

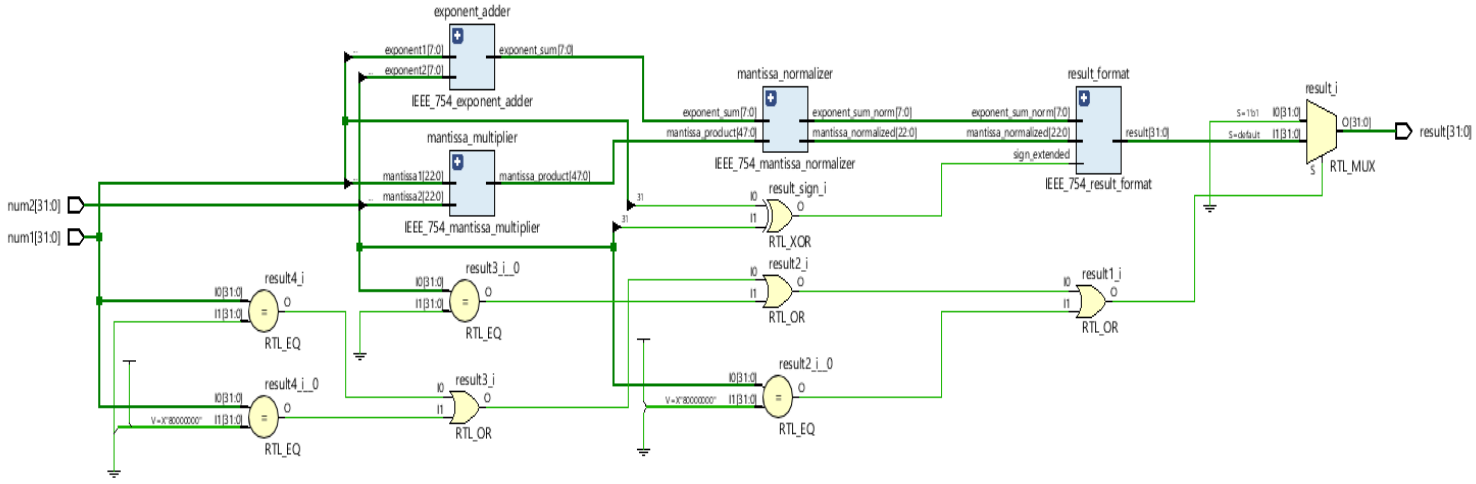


The detailed description of the Floating point adder and Floating point multiplier are as follows:

A.FLOATING POINT MULTIPLIER

This component takes two input num1 and num2 and outputs following floating point multiplication.

Schematic Diagram:



Modules of Multiplier

Exponent Adder:

Two inputs expo_num1 and expo_num2 of 8 bits each are given and result of module is binary addition of these two inputs minus bias(127).

Mantissa Multiplier:

Inputs are mantissa_num1 and mantissa_num2 of 23 bits each.
Hidden bit(1) is concanated to both mantissa.
And Result of module is binary multiplication of mantissa in 48 bits .

Mantissa Normalizer:

This module Normalizes the 48 bit mantissa_product into 23 bits.
If the leading bit is 1,then mantissa is right shifted by one bit
And 1 is added to exponent.
And result of module is Mantissa of 23 bits(mantissa_product[45:23])

Deciding the sign bit.

Sign = xor (sign_num1, sign_num2).

Result Format:

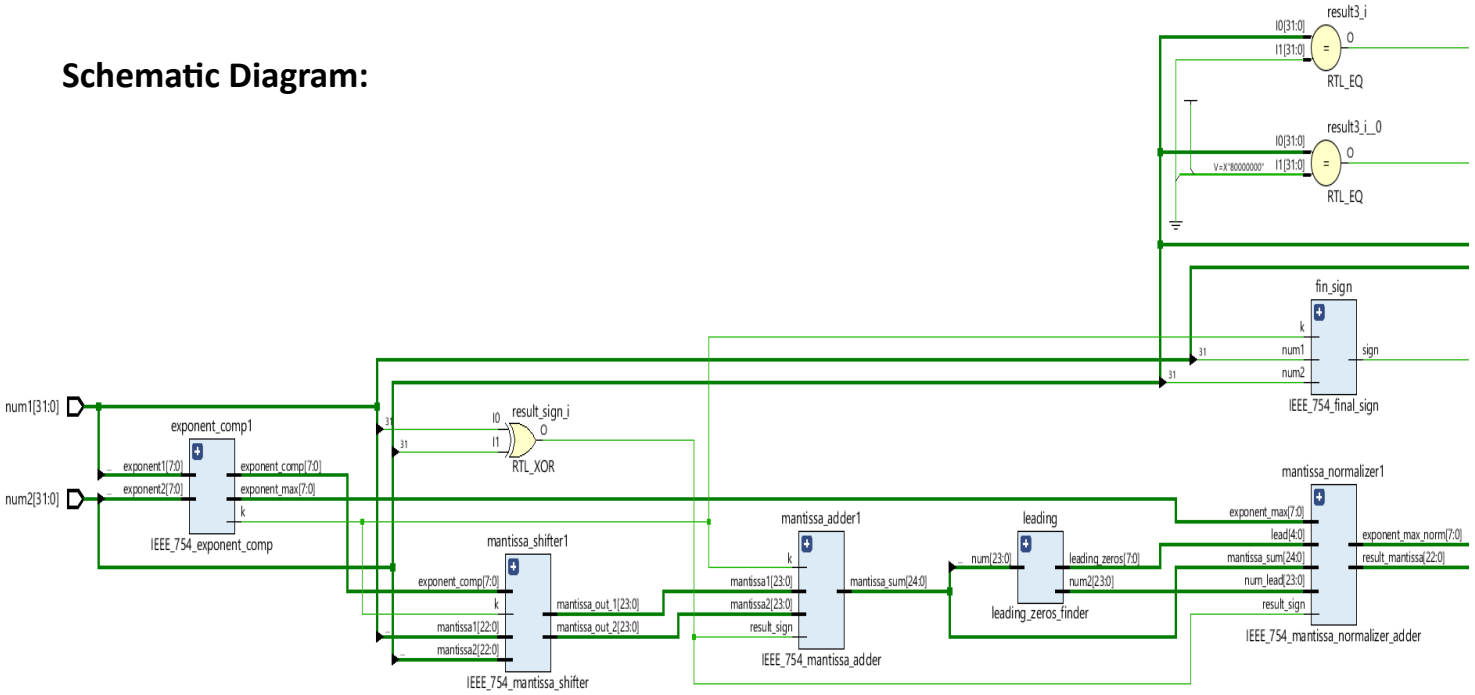
Takes sign, mantissa and exponent as input and concanates it to give our required output.

B.FLOATING POINT ADDER

This component performs addition of Two floating point number.

It takes two input **Num1** and **Num2** and gives the resulting sum.

Schematic Diagram:



Modules of Adder

Exponent comparator:

Take's two input expo_num1, expo_num2 of 8 bit and compares whose exponent is bigger.

If exponent of num1 is bigger than it returns output k=0 and max exponent as expo_num1 otherwise k=1 and expo_num2.

Mantissa Shifter:

Now according to k mantissa is shifted
if k=0 then exp_num1 is greater.
Var = exp_num1- exp_num2
mantissa of num2 is right shifted by var.

else k=1 then expo+num2 is greater.
Var = expo2-expo1
mantissa of num1 is right shifted by var.

Mantissa adder:

Whether to perform addition or subtraction it is decided by taking xor(sign_num1, sign_num2). If result is 0, we perform addition otherwise subtraction.

Leading Zero finder:

If we are performing subtraction this module finds the leading zero by iterating through mantissa bits.

Mantissa Normalizer:

In case of Addition if MSB=1 then max_expo = max_expo+1.
And mantissa is shifted right.

In case of subtraction
Leading zero module is called and mantissa is left shifted by l_zero.
max_expo = max_expo -l_zero.

Final Sign:

Result is given on the basis of k if k=0 final sign of num1 is taken else of num2.

Result_format:

Concanates sign_bit ,exponent bias and mantissa
It also deals with underflow cases; by making the numbers 0 whose exponent is less than equal to 105.