Team- *AA*

Team Members-

Ansh Chaudhary (22095133)

Aditya C Singh (22095143)

In this problem statement, our task was to design and optimize a Processing Element (PE) that will serve as a fundamental building block for the AI accelerator.

The PE consists of the following components:

1) Floating Point Multiplier: This component is dedicated to performing multiplication operations on 32-bit floating-point numbers.

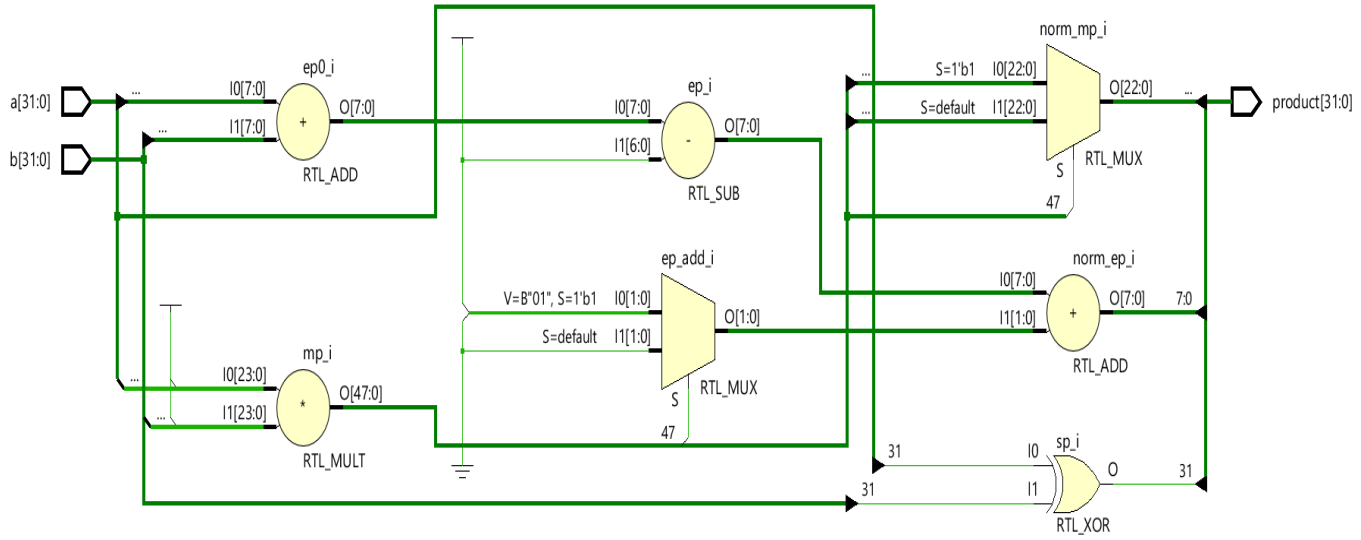
2) Floating Point Adder: Responsible for executing addition and subtraction operations on 32-bit floating-points numbers.

3) Register: The PE includes a register for storing intermediate results and operands during arithmetic operations.

<u>Floating Point Multiplier:</u>

Let us consider we have two 32-bit floating-point numbers a and b. Let us define a module fp_mult, consisting of input as 32-bit a and 32-bit b, while output is 32 bit variable product. Let sa and sb represent the sign bit of a and b. Let ea and eb represent the 8-bit exponent of a and b. Let ma, mb represent the mantissas of a and b concatenated with 1. Let sp represent the sign bit of final product which is obtained by having XOR operation between sa and sb. The 8-bit exponents ea and eb are added and a bias value of 127 in 8-bit binary is removed from their sum. Also, the multiplication operation is performed between ma and mb and assigned to a variable mp,

which is 48 bit long. Another 8-bit variable ep_add is created which is assigned with 8 -bit binary representation of decimal 1 or 0, if the MSB of mp is 1 or 0 respectively. Another 23-bit variable norm_mp is created, which is assigned with 46th to 24th bit of mp, if MSB of mp is 1, else it is assigned with 45th to 23rd bit of mp. The final product is represented using a 32-bit variable product, which is formed by concatenation of sp, norm_ep, norm_mp.



Floating Point Adder:

A module FloatingAddition is created with inputs 32-bit variable A and 32-bit variable B and output variable result which is also 32-bit long. 24-bit variables A_Mantissa, B_Mantissa, Temp_Mantissa are created. A 22-bit variable Mantissa is created. 7-bit variables A_Exponent, B_Exponent, Temp_Exponent, diff_Exponent, Exponent, exp_adjust are created. Single bit variables Sign, carry, comp, A_sign, B_sign and Temp_sign are created. Next, we compare the exponents of A and B and assign single bit 1 to comp if exponent of A is greater than or equal to the exponent of B, or else we assign single bit 0 to it.

Basically, the variables A_Mantissa, A_Exponent, A_sign represent the mantissa, exponent and the sign bit of the greater number out of A and B, while the variables B_Mantissa, B_Exponent, B_sign represent the mantissa, exponent and the sign bit of the smaller number out of A and B. The comparison between the two is decided by the variable comp. The variable diff_Exponent is assigned with the difference between the exponents A_Exponent and B_Exponent. The B_Mantissa is then right shifted by diff_Exponent bits. If the A_sign and B_sign are same, then addition is performed or else subtraction between them is performed and assigned to carry, Temp_Mantissa concatenated. The variable exp_adjust is assigned with A_Exponent. If after addition operation, carry produced is 1, then Temp_Mantissa is right shifted by 1 bit and exp_adjust incremented by 1. Else if Temp_Mantissa is 0, it means that the output is 0 and so Temp_Mantissa is assigned with 23-bit 0, exp_adjust is assigned with 8-bit 0 while A_sign is assigned with 1-bit 0. Else, if 23^{rd} bit of exponent is 0, it represents the underflow condition. Here the Temp_Mantissa is left shifted by 1 and exp_adjust is decremented by 1. The variable Mantissa is assigned with the 22^{nd} to 0^{th} bit of Temp_Mantissa. The variable exp_adjust is assigned to the variable Exponent. The variable result is obtained after the concatenation of the variables A_sign, Exponent and Mantissa. This variable result is the final output of the Floating-Point Adder.