

calculate num1\*num2+num3

Expo\_numo\_3

isZero\_3

isZero\_2

Unum1\_shift

Unum2\_shift

isZero\_1

NaN\_3

isInf\_3

NaN\_2

isInf\_2

Expo\_num1

Expo\_num2

temp1\_2

temp2\_2

isInf\_1

num2

num1

Num3

If num1 or num2 is negative, change it into 2’s complement (required by the format). Check whether they are zero (store in isZero). Check whether one of them is infinite. If one of them is infinite, set isInf to 1.

Use LZC(leading zero counter) to get the length of regime bits

isZero3\_1

isInf3\_1

Unum3\_shift

Temp3

Temp1

temp2

Leftshift three numbers to make exponent bits and fraction bits in certain position.

Calculate the exponent values (in 2’s complement) by using exponent bits and the length of regimes bits

Temp3\_2

Expo\_num3\_3

isZero3\_2

Multiple two fractions (product will be in signed magnitude), add two exponent values. Calculate the difference between expo\_num3 and exponent of product (exponent value1+ exponent value2-exponent value3). The difference can be positive or negative.

In order to make the working frequency higher, multiplication of fractions will take two clock cycles. The multiplication is done in the frac\_mult module

Calculate the sign of product

Frac\_num3\_3

Expo\_diff

Frac\_mult module

Frac\_numo\_3

If the difference of exponent value is negative, change it to positive. (2’s complement to signed magnitude)

Expo\_numo\_4

Frac\_numo\_4

Frac\_mult module

Frac\_num3\_4

Expo\_diff\_4

isInf\_4

NaN\_4

Right shift the fraction of product or fraction of num3 following the difference of exponent value

Expo\_numo\_4

isInf\_4

NaN\_4

Frac\_numo\_4

Frac\_mult module

Frac\_num3\_4

Expo\_diff\_4

NaN\_5

isInf\_5

Expo\_numo\_5

Frac\_numo\_5

Frac\_num3\_5

First step of adding two fractions (in signed magnitude),

Use LZA (leading zero anticipator) to estimate the amount of shift

The addition of fractions in adder is finished in 1 clock cycle in signed magnitude because I can know the which number (num1 or num2) has a greater absolute value in adder module. But for the number3 and product in this module, I cannot compare. So I have to change them into 1’s complement, calculate and changed back, so it will take 2 clock cycles

shift

NaN\_6

isInf\_6

Expo\_numo\_6

Frac\_numo\_6

Do the second step of the addiction. Use LZC to get the amount of shift.

Calculate the result of exponent value by using the larger value between exponent value3 and (exponent value 1+2) and the result of LZC

Shift\_7

Normalize the fraction bits

NaN\_8

isInf\_8

NaN\_7

isInf\_7

Expo\_numo\_8

Frac\_numo\_8

Frac\_numo\_7

Expo\_numo\_7

LZA(leading zero anticipator) is designed based on *Leading-zero anticipator (LZA) in the IBM RISC System/6000 floating-point execution unit* (http://ieeexplore.ieee.org/document/5389860/)

Internal multipliers of FPGA are used to do the multiplication of fraction bits.

LZC(leading zero counter) module is designed based on MODULAR DESIGN OF FAST LEADING ZEROS COUNTING CIRCUIT (http://iris.elf.stuba.sk/JEEEC/data/pdf/6\_115-05.pdf)

manipulate overflow, underflow, rounding and negative number

Right shift the exponent bits and fraction bits following the regime value (in order to adding regime bits on the left side)

Unumo\_10

round

NaN\_8

isInf\_9

Frac\_numo\_9

NaN\_8

isInf\_8

Expo\_numo\_8

Frac\_numo\_8