

num1

num2

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

temp2[31:0]

isZero\_1

isInf\_1

temp1[31:0]

unum1\_shift

Unum2\_shift

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

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

Temp2\_2

Expo\_num2

Compare\_abs\_2

Temp1\_2

isInf\_2

Expo\_num1

Store the exponent number of the larger number in expo\_numo (expo\_numo=max(expo\_num1, expo\_num2)). And store the difference between two exponent values. (if absolute value of number1 greater than absolute value of number2: expo\_num1>=expo\_num2, diff\_expo=expo\_num1-expo\_num2 . Vice versa) If the number is not zero, the fraction will be *1.fraction*, if is zero, will be 0.00000…00

Compare the absolute value of two numbers.

Compare\_abs\_3

Frac\_num2\_3

Expo\_numo

Diff\_expo

isInf\_3

Frac\_num1\_3

Right shift the fraction of the number with smaller absolute value. Shift diff\_expo bits.

Frac\_num1

isInf\_4

Frac\_num2

Expo\_numo

IsZero\_6

Expo\_numo\_6

Frac\_num2\_4

Expo\_numo

Manipulate zero, Inf and do the rounding

Right shift the number following the exponent value to add the regime bits on the left side

Normalize frac\_numo by using LZC. Check whether the result is zero

Add frac\_num1 and frac\_num2. (in 1’s complement)

Unumo

isInf\_7

Round

isInf\_6

Frac\_numo\_6

Frac\_numo\_5

Expo\_numo

isInf\_5

Runumo

Runumo

isInf\_4

Frac\_num1\_2