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.

isZero\_1

isInf\_1

temp1[31:0]

temp2[31:0]

Separate sign bit, regime bits, exponent bits, and fraction bits. Change regime bits and exponent bits into real exponent value, stored in expo\_num. store sign bit and fraction bits in frac\_num.

Compare\_abs\_2

Frac\_num2\_2

Expo\_num2

Frac\_num1\_2

isInf\_2

Expo\_num1

Store the larger exponent number in expo\_numo. Caculae whether abs(num1)>=abs(num2)

Compare\_abs\_3

Frac\_num2\_3

Expo\_numo

Diff\_expo

isInf\_3

Frac\_num1\_3

if abs(num1)>=abs(num2), move frac\_num2 following diff\_expo

Frac\_num1

isInf\_4

Frac\_num2

Expo\_numo

IsZero\_6

Expo\_numo\_6

Frac\_num2\_4

Expo\_numo

Runumo=runumo+round

Put expo\_numo, and frac\_numo into unum format. Round bit is the MSB of ignored number. If the length of the expo is 3, round is the MSB of ignored fraction bit. If length of expo is 0 or 1. Round bit is expo\_numo[1] or expo\_numo[2]

Normalize frac\_numo

Add frac\_num1 and frac\_num2

Unumo

isInf\_7

Round

isInf\_6

Frac\_numo\_6

Frac\_numo\_5

Expo\_numo

isInf\_5

Runumo

Runumo

isInf\_4

Frac\_num1\_2