****

School

of

Electronics and Communication Engineering

ADLD COURSE PROJECT

**IE-TEAM -2**

**Team members:**

1. Poornima Bendigerimath – 01FE20BEI046

2. Varun Savadatti – 01FE20BEI006

3. Abhay Deshpande– 01FE20BEI018

**Guide By:**

Dr. Saroja V. Siddamal

Semester: VI,2023

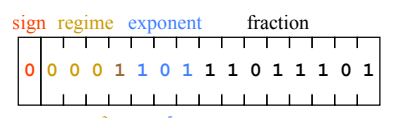
1. Problem Statement :

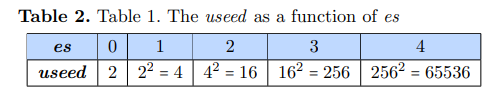
Design of power efficient posit multiplier .

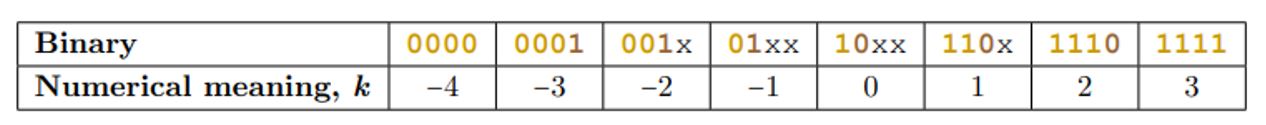
2) Architecture :

For our architecture , we are converting a floating point number into posit number .Posit number system is shown in fig1.

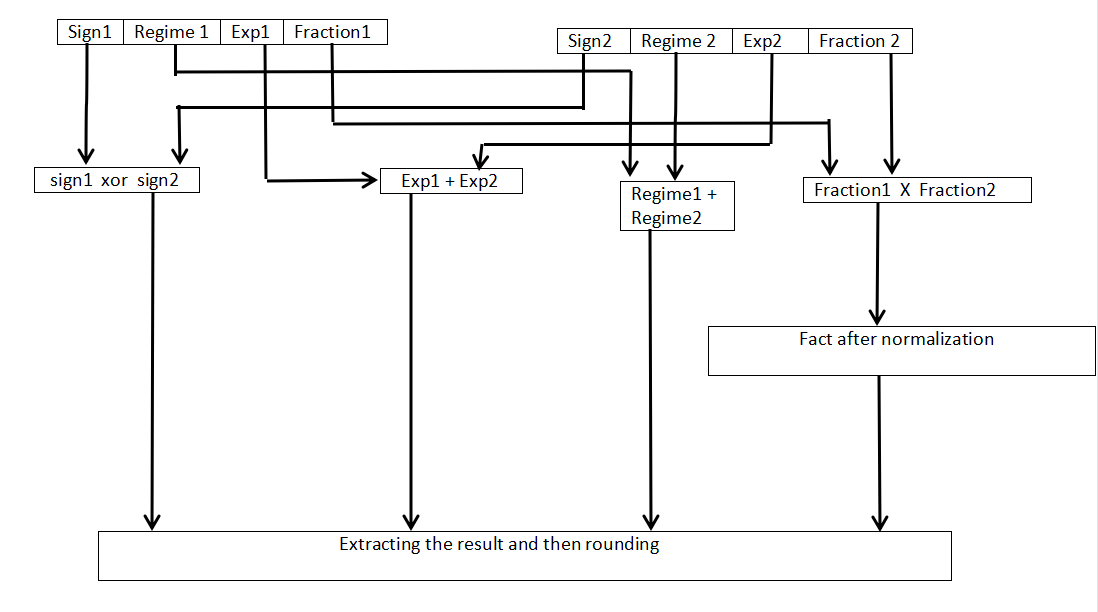
We are taking two posit numbers for multiplication. For posit conversion of a number we are referring the table shown in fig2. , based on the bit size of a posit number ‘es’ values will get assigned.And for k value , corresponding bit pattern is assigned.







PROPOSED ARCHITECTURE:



Our architecture works as follows:

1. Resultant sign bit is ex-or of both the sign bits.
2. The resultant exponent bits are the addition of both exponents.
3. Regime bits are the addition of both regimes.
4. For fraction buts, we need to multiply both the fraction bits and then normalize the resultant fraction bits.

3)Theory:

Posit number format, which is a type of numerical representation designed as an alternative to traditional floating-point number formats.

Posit numbers were developed by John Gustafson and are based on the concept of a "universal bit pattern" that can represent a wide range of numbers with high accuracy and low storage requirements. The posit format uses a fixed number of bits to represent a number, but unlike floating-point formats, the number of bits used can vary depending on the size of the number being represented.

Posit numbers are designed to overcome some of the limitations of floating-point numbers, such as rounding errors and overflow/underflow issues. They are also intended to be more efficient in terms of hardware implementation, as they require fewer operations to perform basic arithmetic operations.

Overall, posit numbers are a relatively new concept in numerical representation and are still being researched and developed. While they hold promise for certain applications, they are not yet widely used in practice.

4) Algorithm :

Step 1 : Convert the given number into it’s binary form.

Step 2: Separate the numbers according to the posit format representation like sign bit, regime bit, exponent and mantissa.

Step 3: To multiply the 2 numbers, xor the sign bits, add regime bits, add exponent bits and multiply the mantissa bits.

Step 4: Represent the multiplied number into posit format.

Step 6: Display the result.

4) Results:

Simulation

This code takes two 16 bit Posit numbers a and b as input, and outputs their product result in Posit format. The a and b numbers are first decomposed into their sign, exponent, and fraction components, which are then used to perform the multiplication. The resulting Posit product is then converted back to Posit format by adjusting the exponent and fraction as necessary. Note that this code assumes that the Posit format being used is the standard format with a 32-bit total length, a 2-bit regime, and a 30-bit fraction.

6) Conclusion:

By this, we conclude that how to convert the IEEE floating point number to posit number and multiplication of 2 posit numbers using the proposed architecture.