CSE 210
Computer Architecture Sessional

Assignment-2: Floating Point Adder Simulation

Section - B1
Group - 05

Prepared By
Name:Diganta Saha Tirtha
Id:2105081

Members of the Group:

i 2105068 - Anika Morshed

ii 2105069 - Sheikh Iftikharun Nisa

iii 2005081 - Diganta Saha Tirtha

1 Introduction

A computer representation of a real number with a fixed number of digits before and after the decimal point (or radix point in a more general sense) is called a floating point number. A floating point adder is a digital circuit that works with floating point numbers to add and subtract. Numbers that are too large or too small to be accurately represented by integer representations can be represented using this method.

Three fields make up the floating point representation of a number: the sign bit, the exponent field, and the mantissa field. The sign bit represents the sign, either positive or negative. To express a wide range of values, the exponent field permits the significand to be multiplied by a power of the base using a fixed amount of bits in a biased form. The bias needs to be deducted from the stored bits in order to get the real exponent. The fractional portion of the number, or mantissa, is made up of the digits that come after the decimal point. The sign bit, exponent field, and mantissa in this implementation use 1, 9, and 22 bits, respectively. Thus, the exponent bias for our problem would be $2^{9-1} - 1 = 255$.

In order to add two numbers, a floating point adder first aligns their decimal points before adding their mantissas. After necessary shifts and related modifications (increment/decrement), the result's exponent is fixed. Normalization and rounding are performed afterwards. The signs of the two integers being added determine the sign of the outcome.

Applications for floating point adders include financial modeling, computer graphics, and calculations in science and engineering. Because co-processors can be computationally demanding, it is utilized in them to do quick, hardware-accelerated floating point arithmetic. These calculations can be completed far more quickly by a specialized floating point adder than by the main processor. Applications requiring high-precision floating point calculations, such data analysis and scientific simulations, may find this to be especially crucial..

2 Problem Specification

In this assignment, we are required to design a floating-point adder circuit which takes two floating points as inputs and provides their sum, another floating point as output. Each floating point will be 32 bits long with following representation:

Sign	**Exponent**	**Fraction**
1 bit	9 bits	22 bits

Table 1: Problem Specification

3 Flowchart of the addition/subtraction algorithm

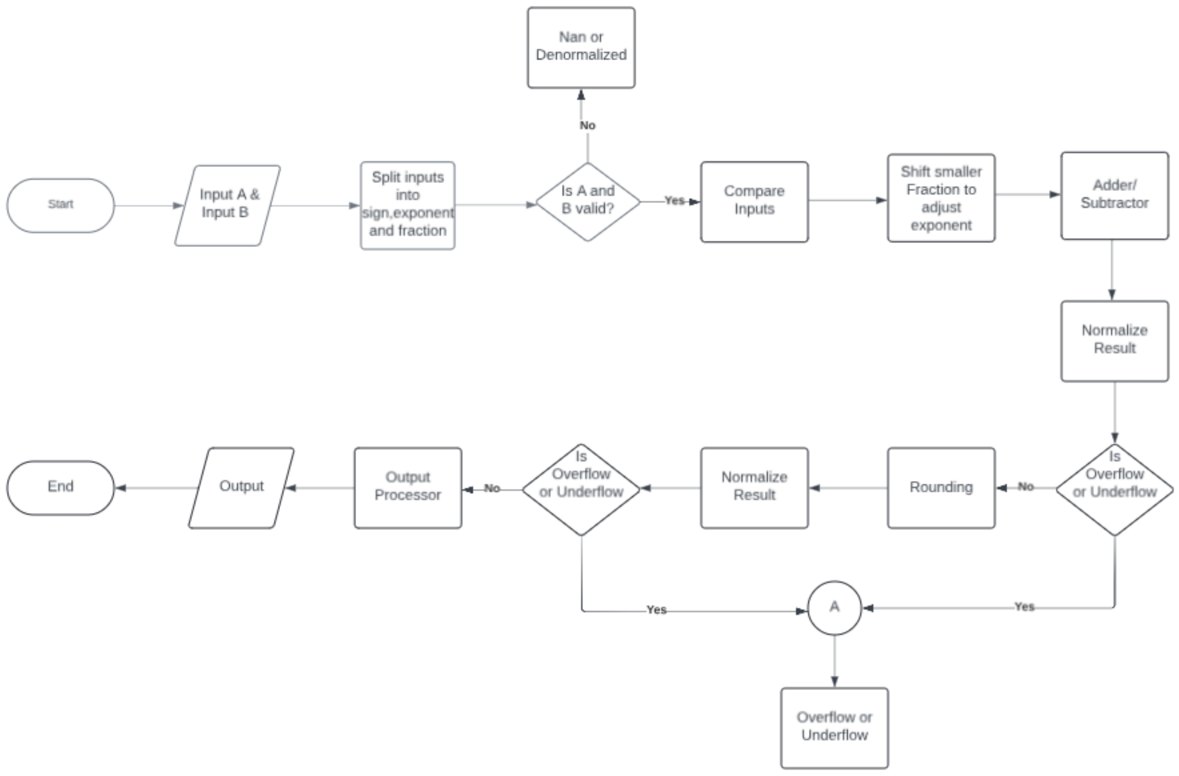


Figure 1: Flowchart of the addition/subtraction algorithm

4 High-level block diagram of the architecture

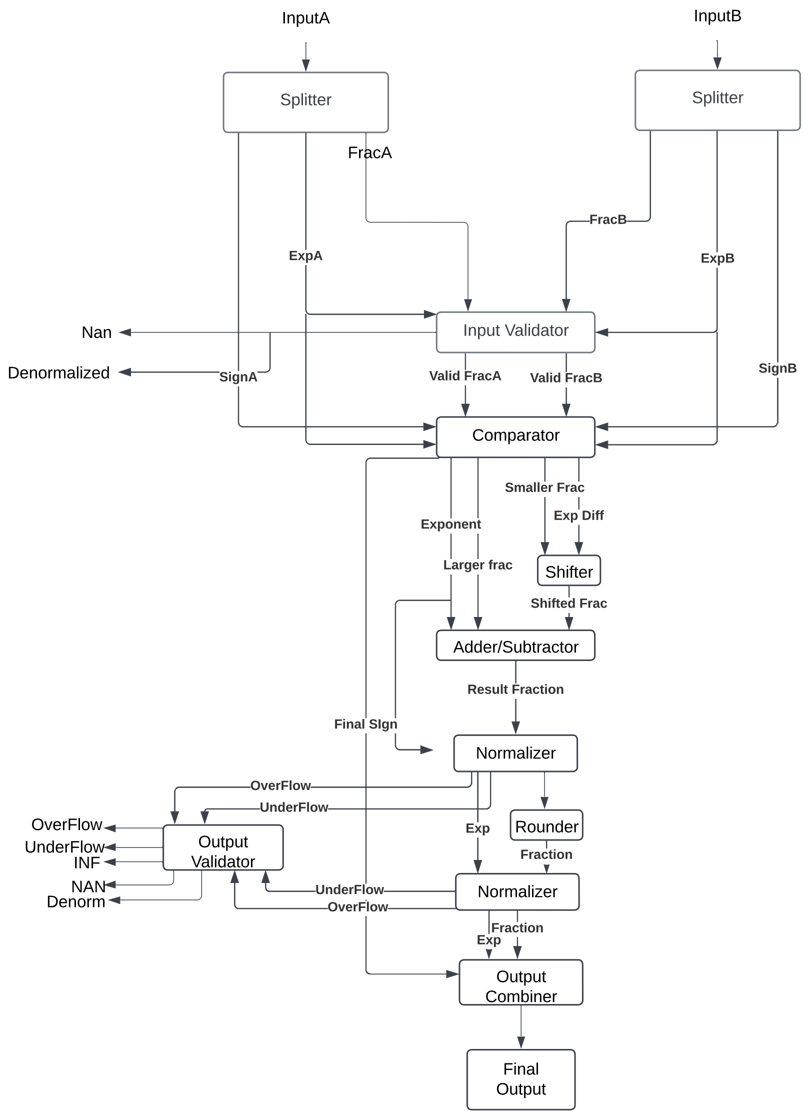
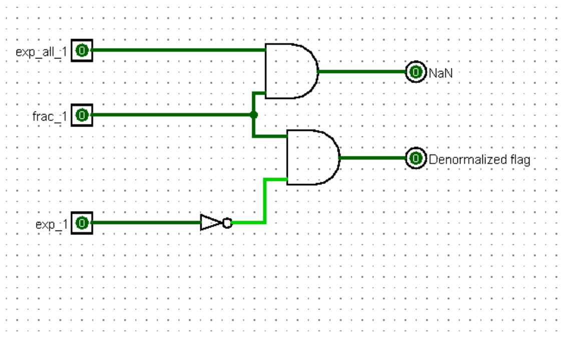


Figure 2: High-level block diagram of the architecture

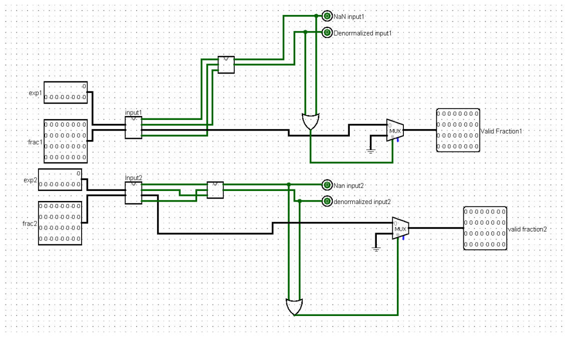
5 detailed circuit diagram of the important blocks

5.1 Input Validator

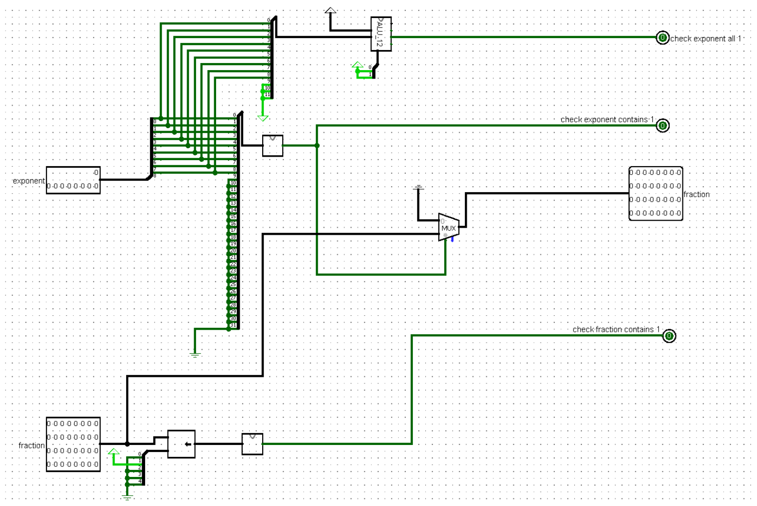
This library includes:



(a) Nan/Denormalized Checker



(b) Fraction/Exponent Checker



(c) Main Input Validator

5.2 Comparator

Main purpose of this circuit is to determine the larger and smaller fraction,larger exponent,shift the smaller fraction to adjust with the exponent,and determine the output sign

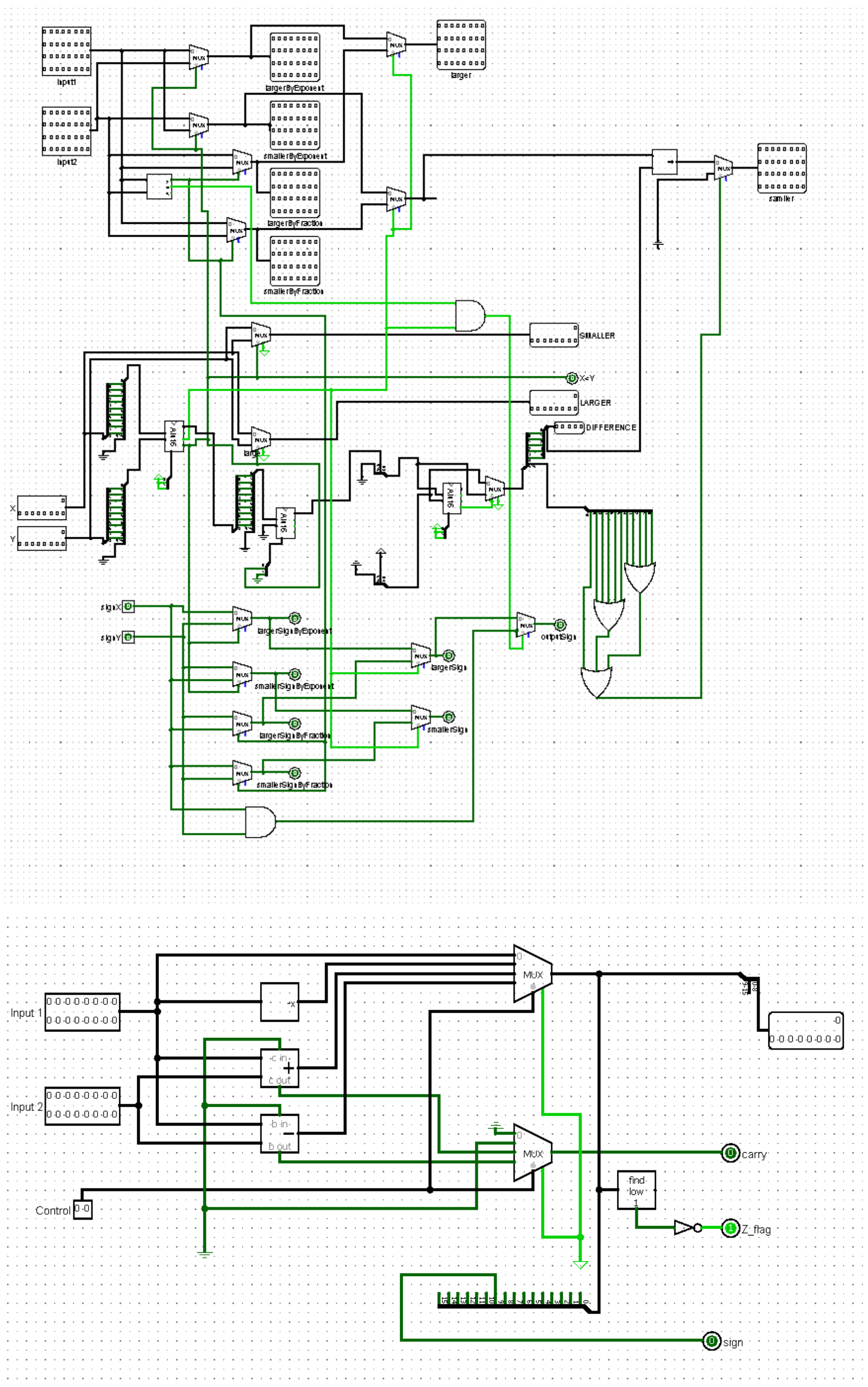
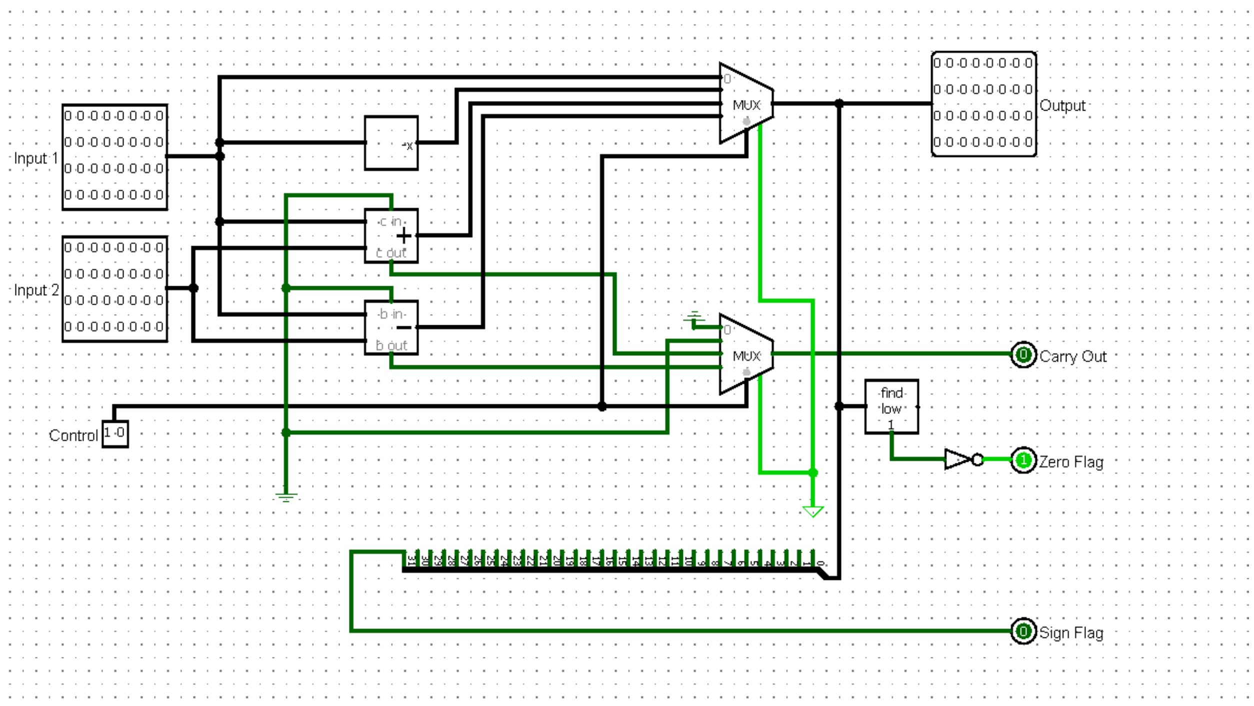


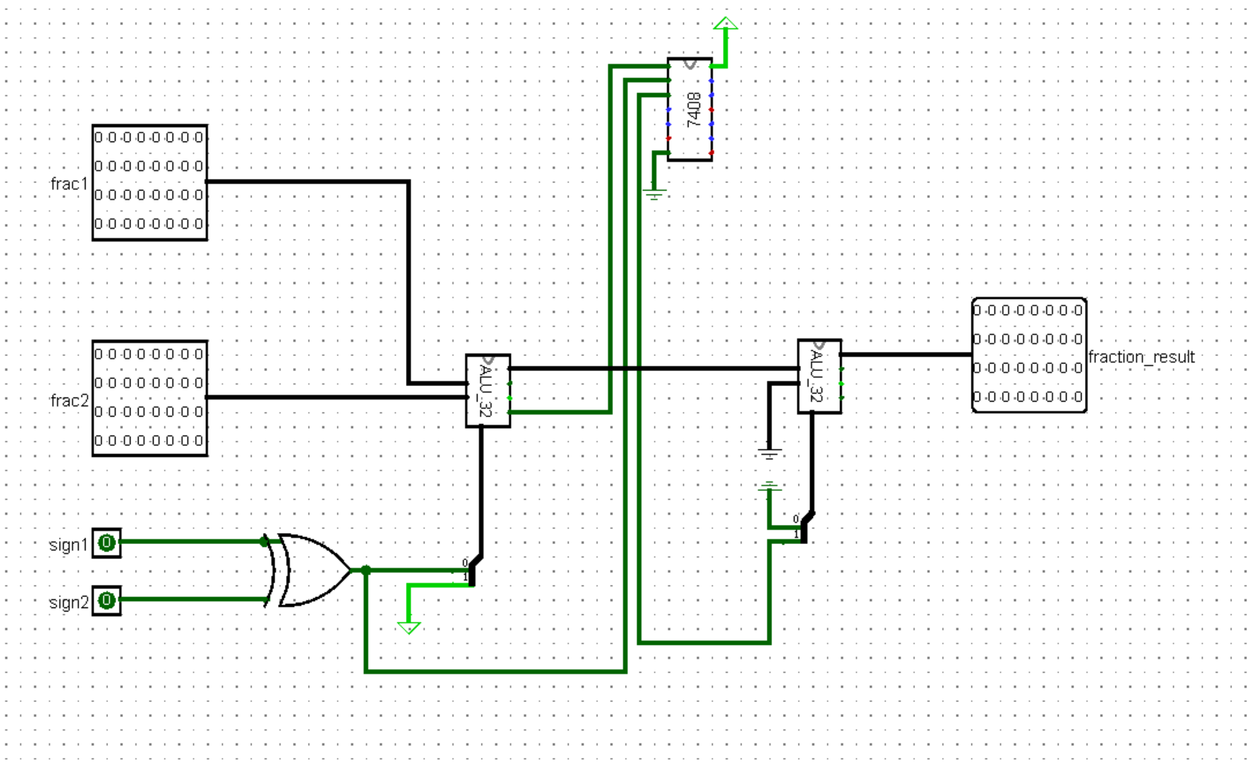
Figure 4: Main Comparator Circuit

5.3 Adder/Subtractor

This module includes the 32 bit ALU and Main Adder/Subtractor circuit and generates the output fraction.



(a) 32 Bit ALU



(b) Adder/Subtractor

Figure 5: Main Adder/Subtractor Circuit

5.4 Normalizer

This module normalizes the output fraction and adjust the exponent by shifting appropriately.This also generates the INF,NAN,OVERFLOW,UNDERFLOW,DENORM flags.

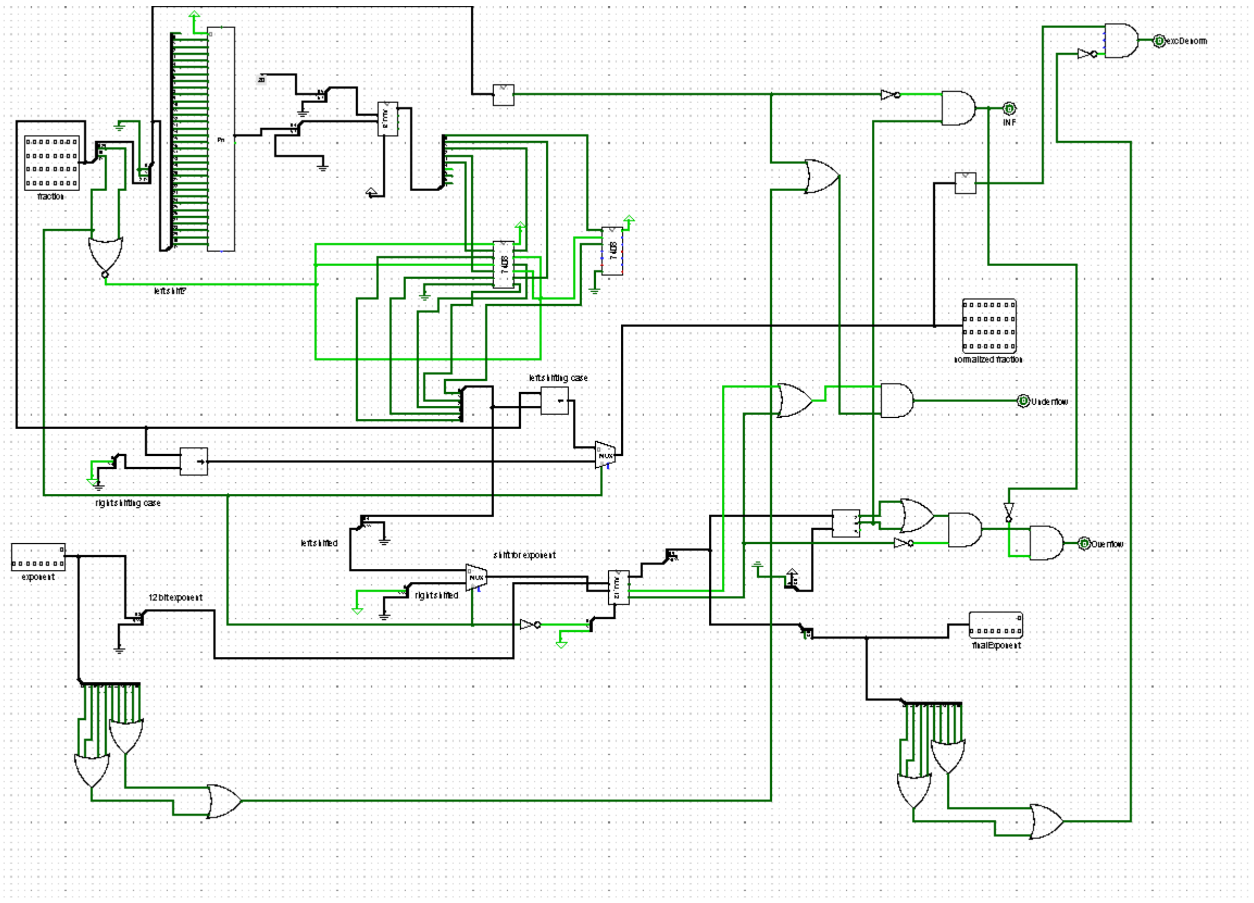


Figure 6: Normalizer Circuit

5.5 Rounding Circuit

The rounding of the result is done by this circuit.

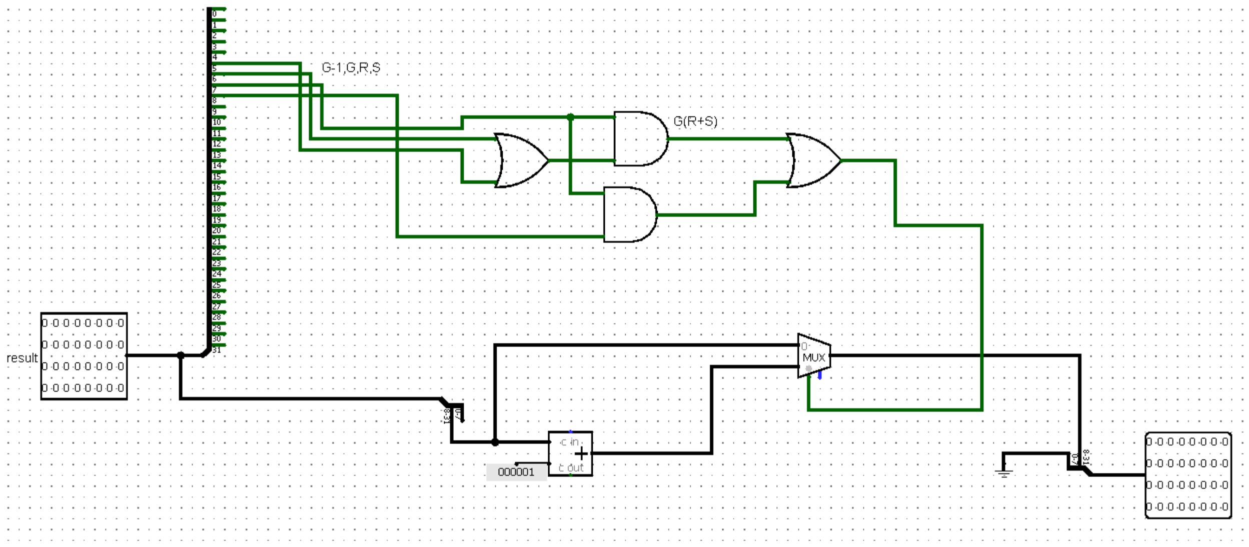


Figure 7: Rounding Circuit

5.6 Floating Point Adder

This is the main floating point adder.

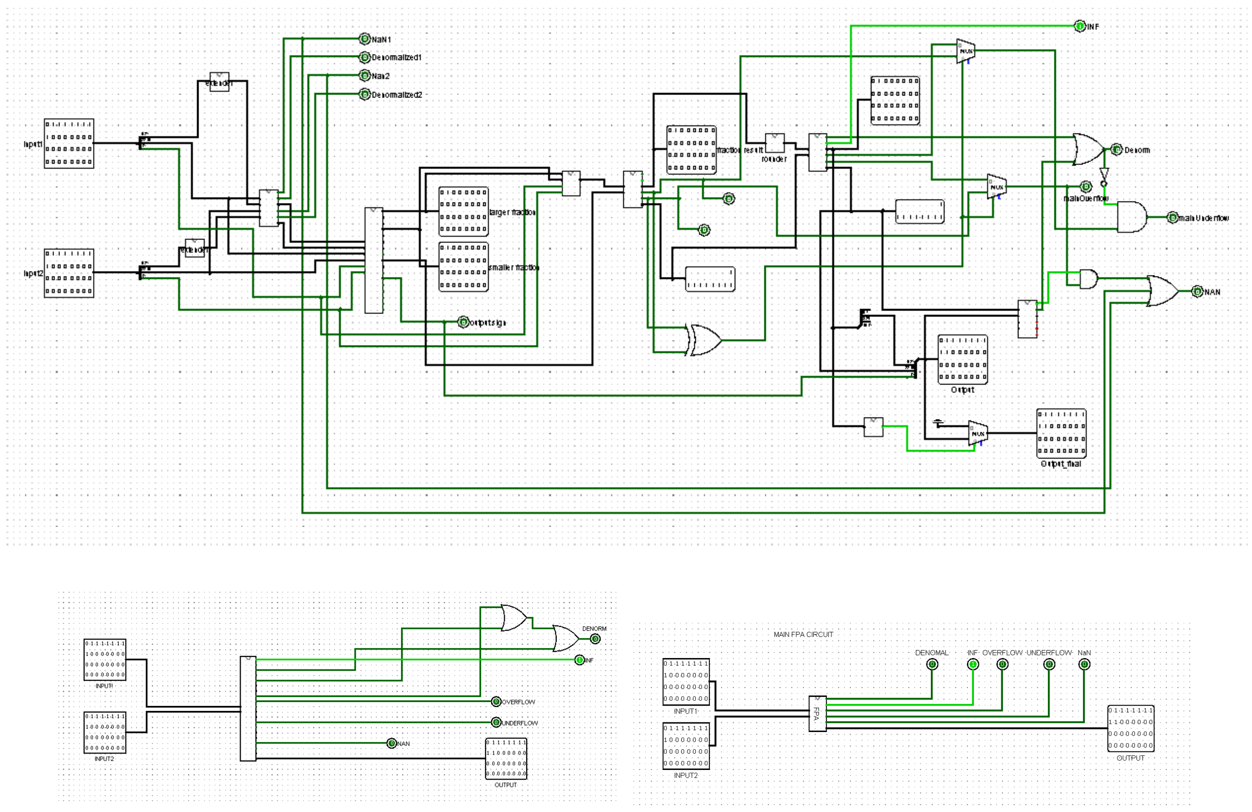


Figure 8: Floating Point Adder

6 ICs Used with Count as a Chart

IC	Quantity
IC 7404	2
IC 7408	9
IC 7432	109
IC 7486	1
IC 74157	110
Total	231

Table 2: ICs Used with Quantity
ALU(16 & 32 bit),built-in comparator,shifter and priority encoder were used seperately.

7 Simulator used Along with the Version Number

Logisim -2.7.1

8 Discussion

We put significant effort into completing our assignment. Utmost importance was given to reducing the number of ICs.

We faced multiple challenges while implementing the circuit. In the IEEE format, the implicit 1 of the fraction is not presented. However, while adding two numbers, there can be cases where 10 or 11 appears before the decimal point. To handle this, we added two extra bits, 01, to each fraction while splitting the number into sign, exponent, and fraction.

Next comes the case of shifting. In the comparator module, we compare the two numbers and determine the larger and smaller fractions. The larger exponent is also calculated here. The smaller fraction is shifted left appropriately, corresponding to the exponent difference. The 22-bit fraction was extended to 32 bits by adding 01 on the left, and the right side of the fraction was padded with zeros. Thus, the maximum shift for the fraction can be 31 bits. Any shift greater than 31 results in the smaller fraction becoming all 32-bit zeros. This scenario was implemented carefully. The sign bit was also determined in the comparator module.

In the normalization module, we normalized the resulting fraction from the adder/subtractor module. Appropriate right or left shifts were made to normalize the fraction. The exponent was also increased or decreased accordingly. The INF, Denorm, NAN, Overflow, and Underflow flags were implemented to identify any invalid output.

Appropriate rounding was performed to ensure correct output. Finally, the sign, exponent, and fraction were combined to produce the final output.

The assignment was tough yet interesting. We learned a lot from this assignment.

9 Contribution

- 2105081- Contributions in:
  - Comparator
  - Normalizer
  - Floating Point Adder
  - Debugging And Testing
- 2105068- Contributions in:
  - Adder/Subtractor
  - Normalizer
  - Rounding Circuit
  - Debugging And Testing
- 2105069- Contributions in:
  - Input Validator
  - Normalizer
  - Rounding Circuit
  - Debugging And Testing