

CSE-306

# Floating Point Adder Implementation

Submitted by:

Group: 1

Section: B-2

Members:

Nahian Salsabil - 1705091

Farhana Khan - 1705100

Rittik Basak Utsha - 1705105

Muhtasim Noor - 1705108

## **Introduction**

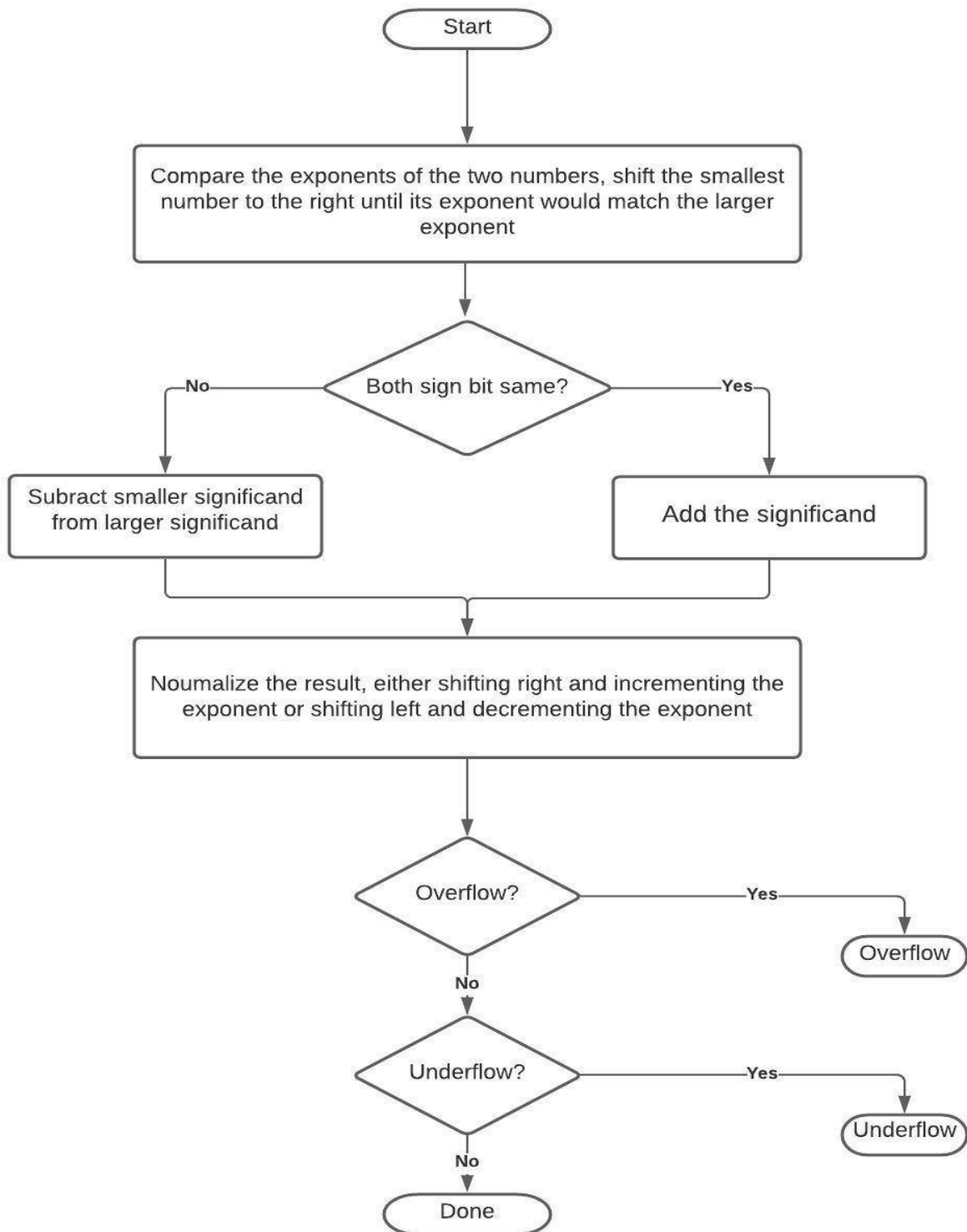
In this assignment, our goal is to design and implement a Floating-Point Adder. Floating points in computer memory are stored using binary representation following a certain convention. Adding floating points is different from adding integers.

## **Problem Specification**

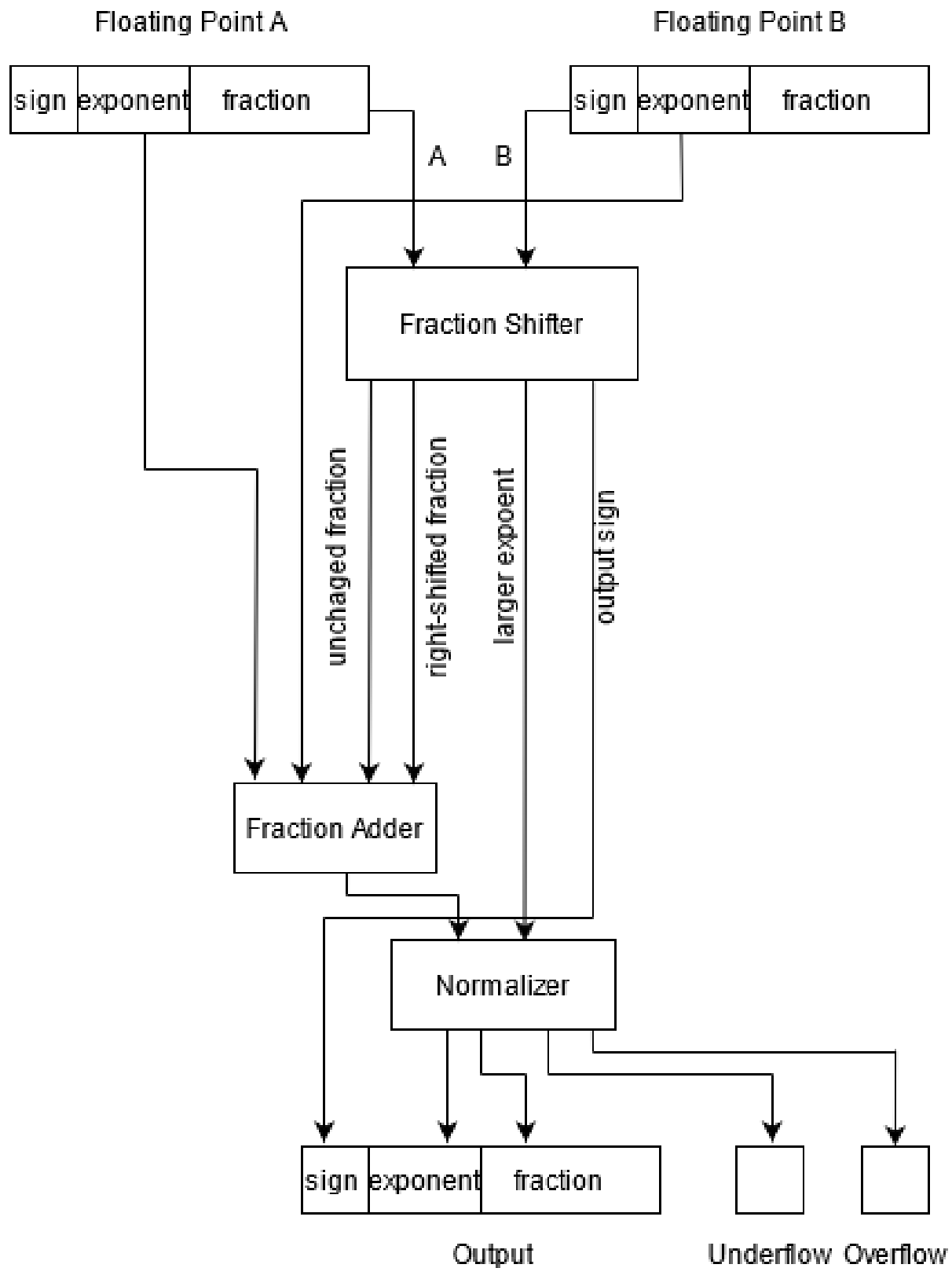
Given two floating points we have to design a circuit that adds them and gives another floating point as result. We also have to implement the circuit using simulation software. Floating points in this task will be represented by the following format:

<i>Sign</i> <b>1 bit</b>	<i>Exponent</i> <b>4 bit</b>	<i>Fraction</i> <b>11 bit</b>
-----------------------------	---------------------------------	----------------------------------

## Flow Chart

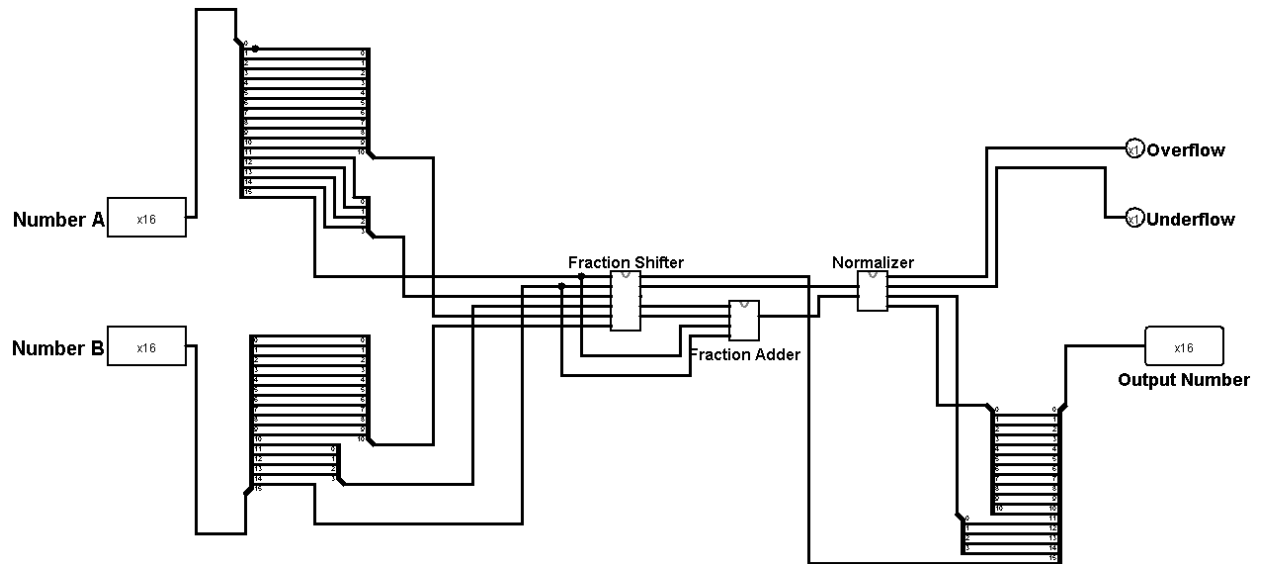


## Block Diagram

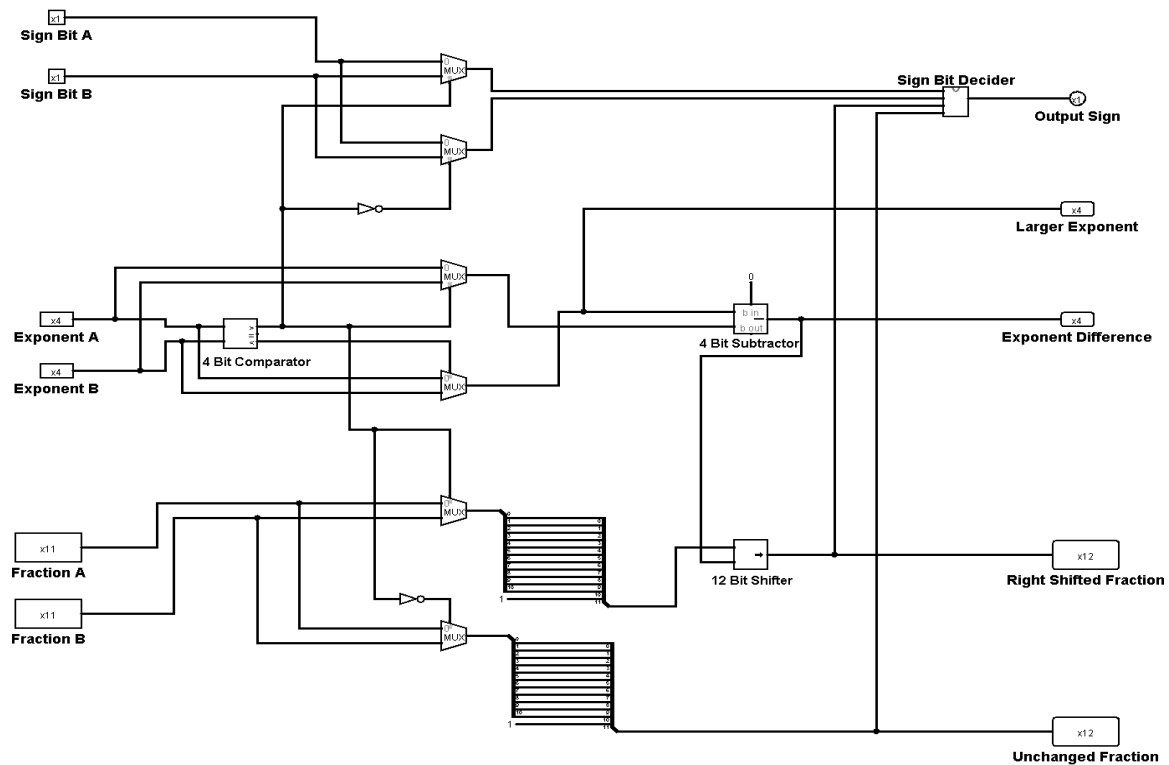


# Circuit Diagrams:

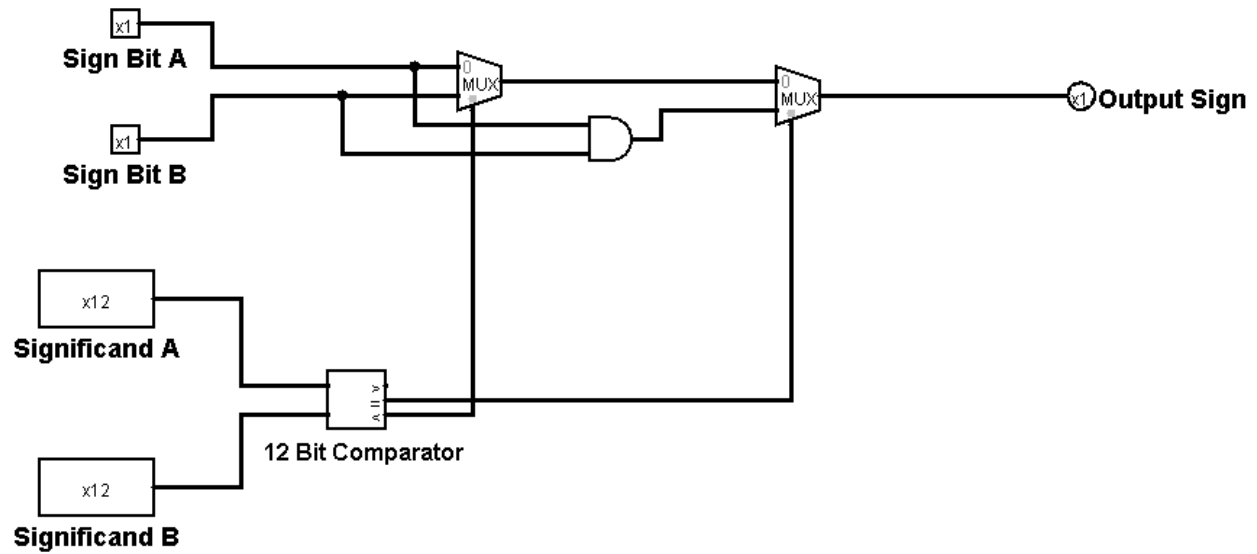
## Fp Adder:



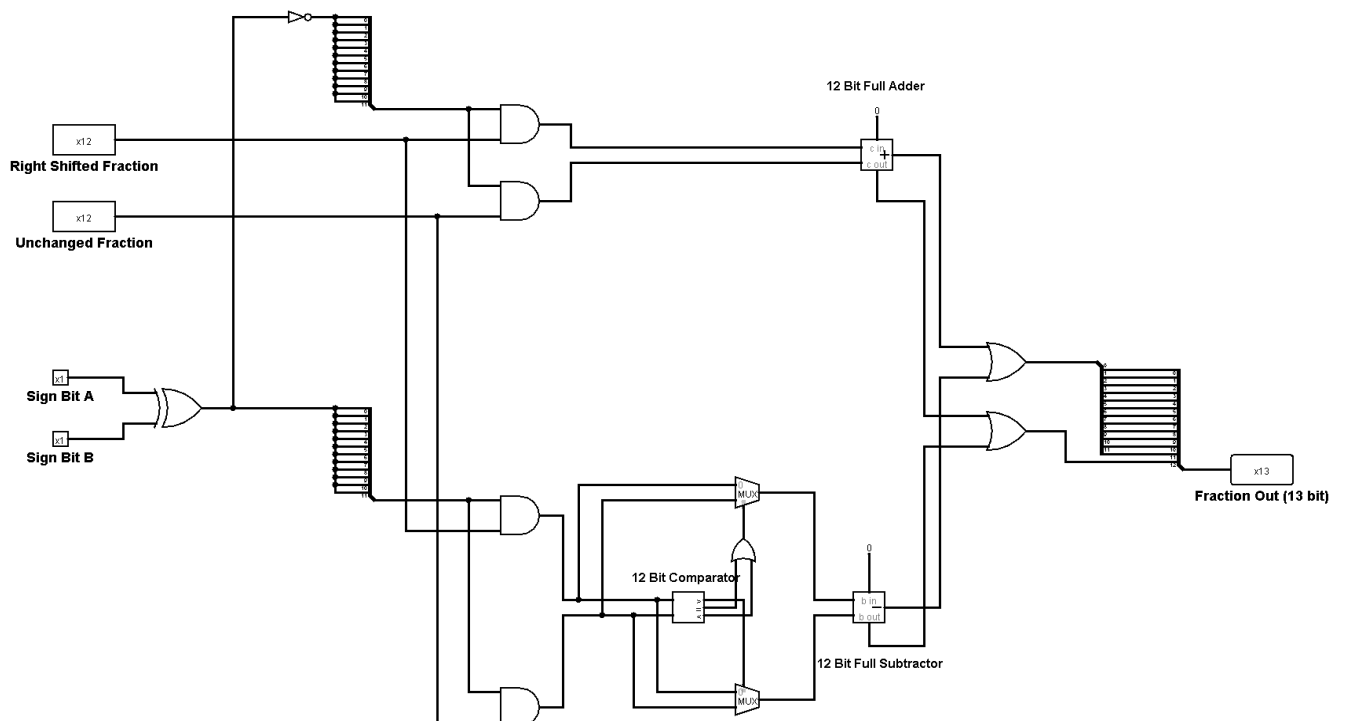
## Fraction Shifter:



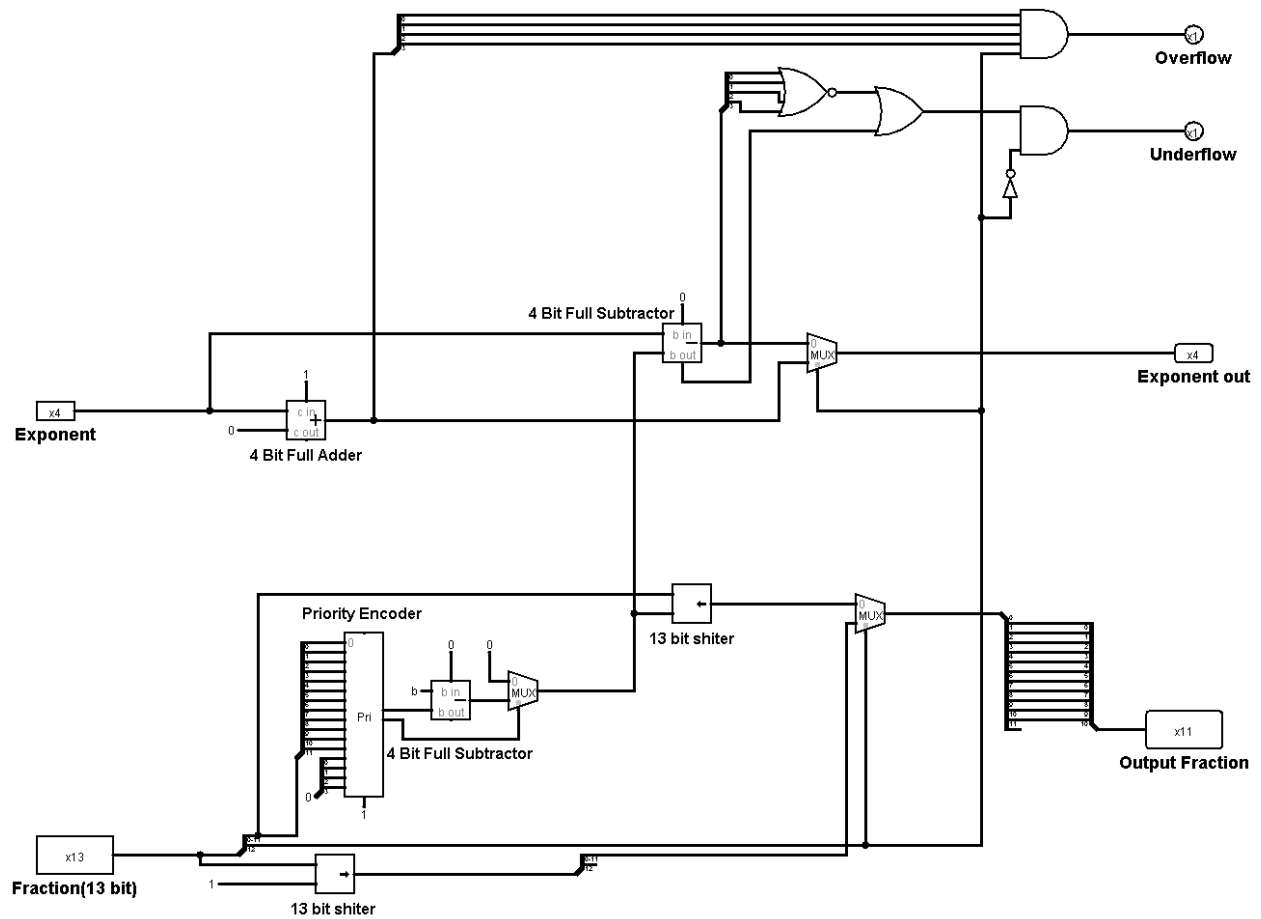
## Sign Bit Decider:



## Fraction Adder:



## Normalizer:



## **IC Count**

IC	Name	Count
7432	OR Gate	4
7408	AND Gate	7
7486	XOR Gate	1
7402	NOR Gate	1
7404	NOT Gate	3
74157	2x1 Mux	13
7485	Magnitude Comparator	3
7483	4-bit Full Adder	2
74148	Priority Encoder	1
	Shifter	3
	4-bit Full Subtractor	2

## **Simulator Used**

We have used “Logisim 2.7.1” to implement and simulate our design of Floating Point Adder

## **Discussion**

We tried to minimize the use of ICs in this design and it was designed to be as simple as we could think.

We used a priority-encoder to find the most significant 1.

The normalized result was truncated to 11 bit fraction. After normalization overflow (exponent value above 14) and underflow (exponent value below 1) was detected.