CSE306 - Computer Architecture Sessional

Assignment 2

Floating Point Adder

Submitted by:

1705062	Jawad Ul Kabir
1705063	Md. Mahfuzur Rahman Rifat
1705065	Taseen Mubassira
1705066	Ataf Fazledin Ahamed
1705067	Nishat Farhana Purbasha

30 May, 2021

1 Introduction

A **Floating Point Adder** is a combinational logic circuit which takes two floating point numbers as input and outputs their sum which is also a floating point number.

In order for computer systems to compute different arithmetic operations on floating point numbers, floating point arithmetic is developed to support the operations within a fast processing time. The design and implementation of floating point adder is different from usual full adder due to several reasons-

1. The floating point number has a sign part, an exponent part as well as a fraction part.
2. The output needs to be in normalized form. The inputs are assumed to be in normalized form as well.

The three parts represent the actual floating point number as follows:

$$(-1)^{Sign} * (1 + Fraction) * 2^{Exponent-Bias}$$

The normalized form allows the numbers to be precise as it ignores all the unnecessary leading zeroes by replacing them with real digits to the right of the binary point. It also ensures uniformity as all the numbers will always be in this form. Using biased exponent ensures that the exponent portion will always be positive and very small numbers can be stored efficiently as well as the large numbers.

The common trade-off in floating point arithmetic is range vs precision. A floating point system having a limited number of bits has to either compromise its range for precision by increasing number of bits for fraction, or it may extend its range by increasing the number of bits for exponent, which in turn reduces its precision.

2 Problem Specification

A floating point adder circuit is required to be designed and implemented in a simulator software. The circuit takes two floating points as inputs and provides their sum, another floating point as output. Each floating point will be 16 bits long with following representation:

Sign	***Exponent***	***Fraction***
1 bit	4 bit	11 bit

The overflow and underflow flags also need to be implemented so that it correctly indicates when the output is out of range for the given number of bits. Rounding is also necessary in order to fit the result in said number of bits.

3 Flowchart of the Algorithm

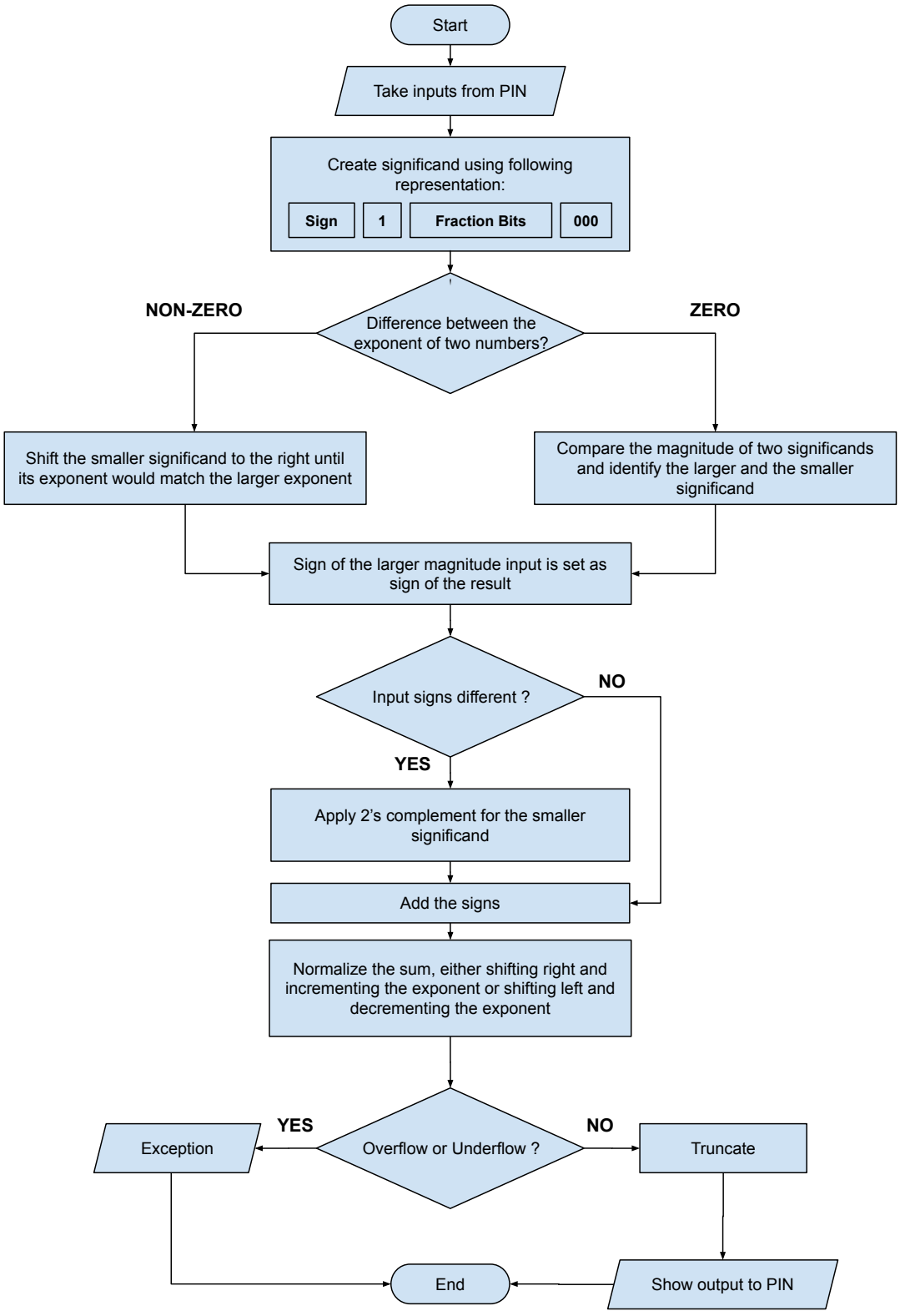

Figure 1: Flow chart

4 Block Diagram

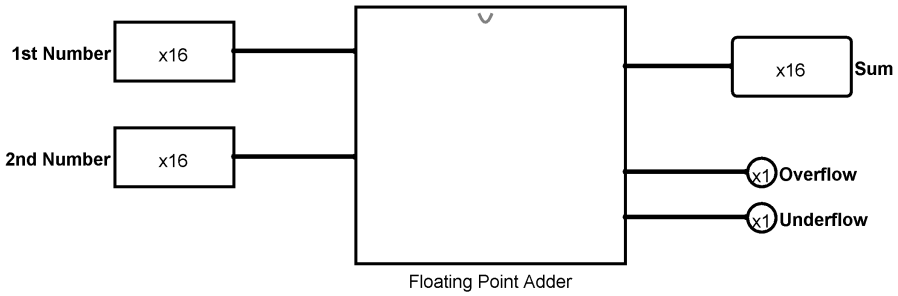

Figure 2: Floating Point Adder (Block Diagram)

5 Circuit Diagram of the Important Blocks

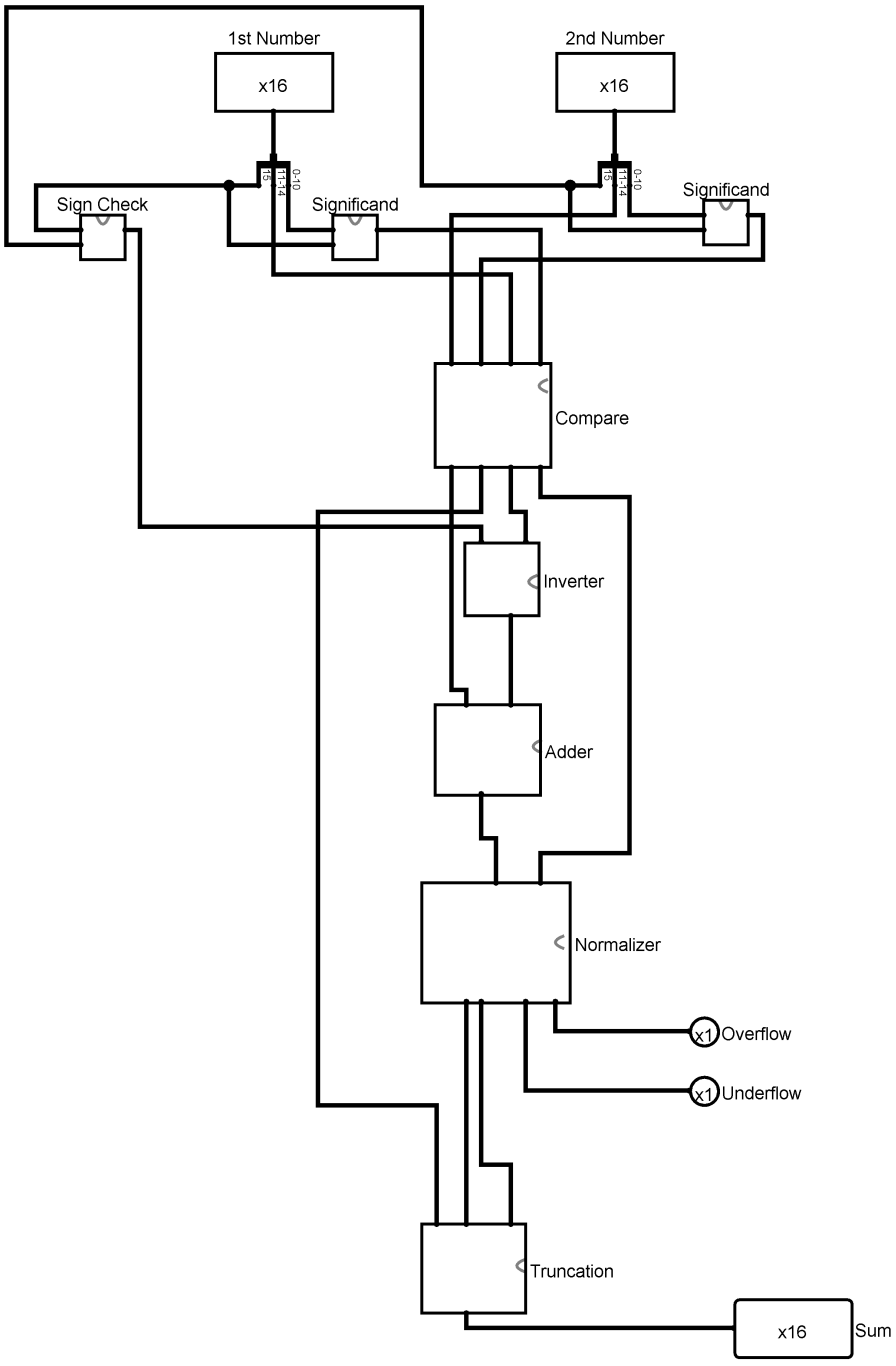

Figure 3: Floating Point Adder Circuit

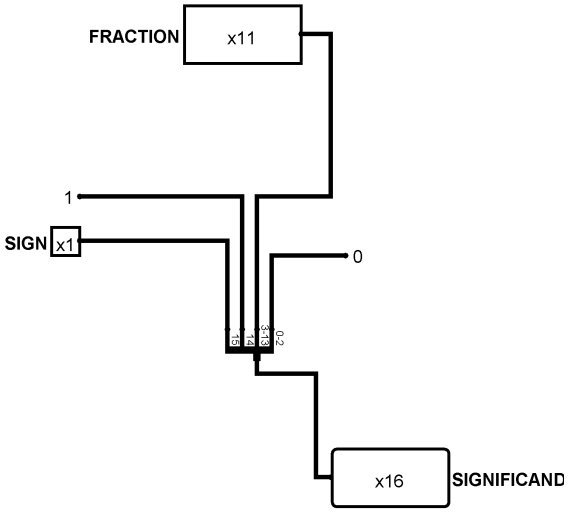

Figure 4: Significand Circuit

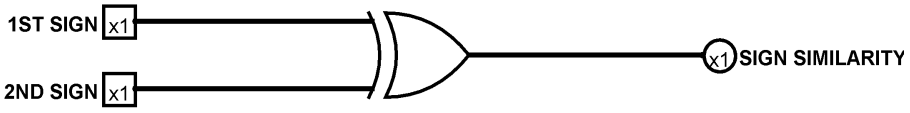

Figure 5: Sign Check Circuit

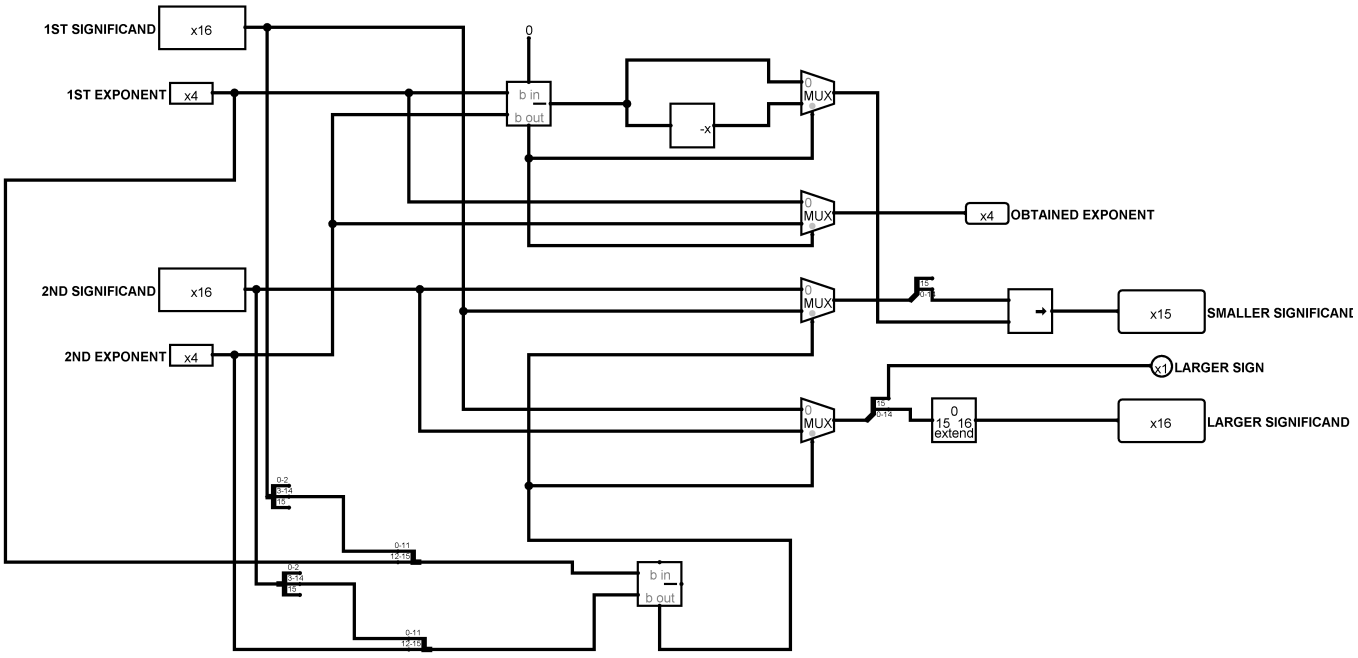

Figure 6: Compare Circuit

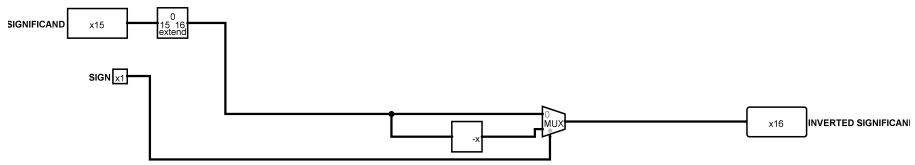

Figure 7: Inverter Circuit

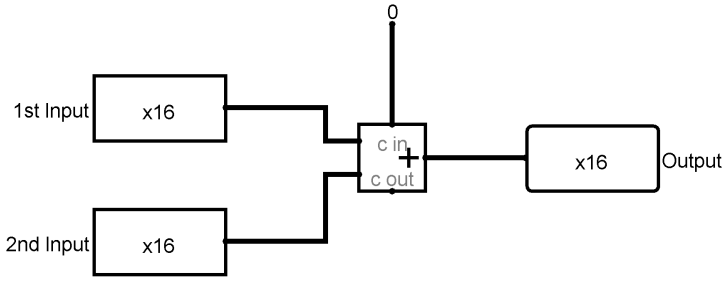

Figure 8: Adder Circuit

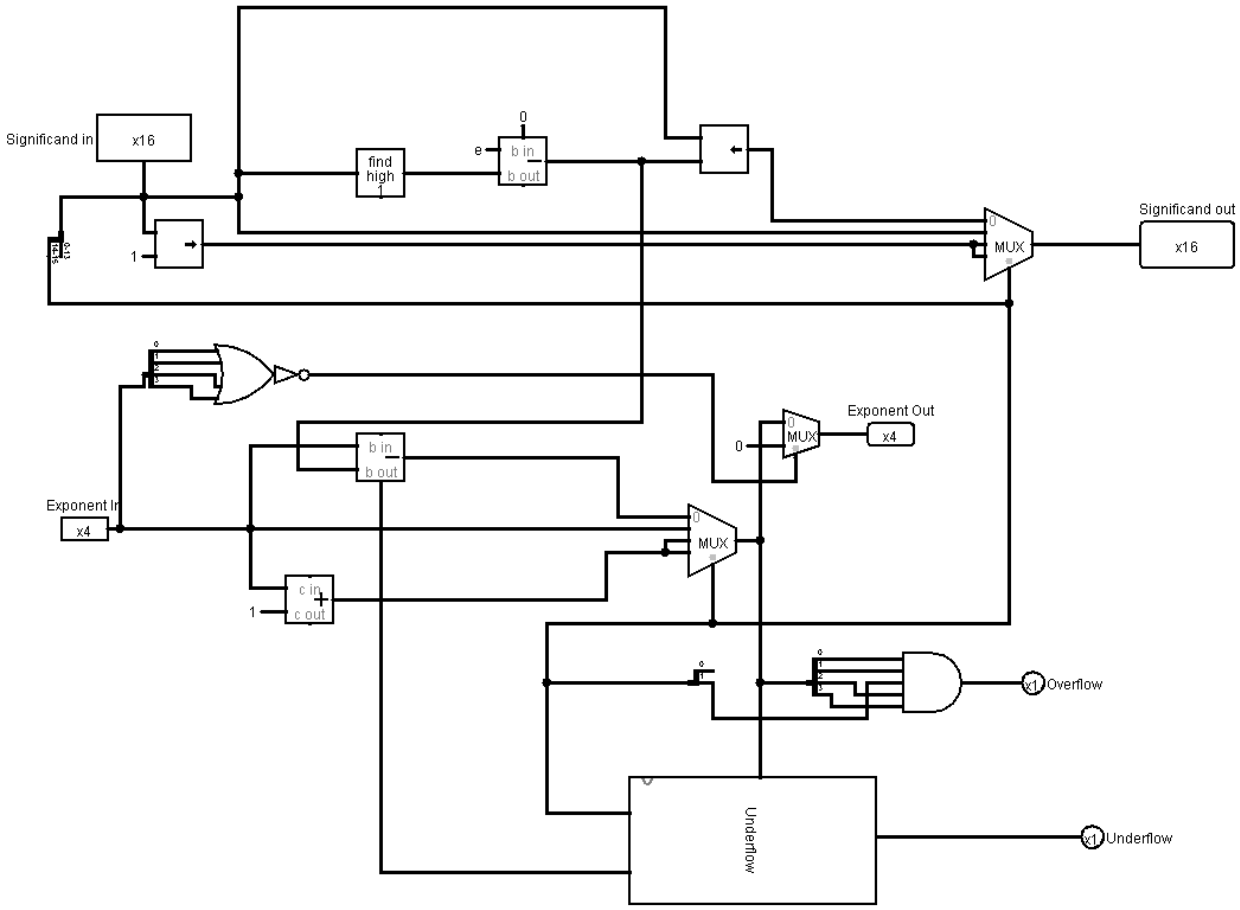

Figure 9: Normalizer Circuit

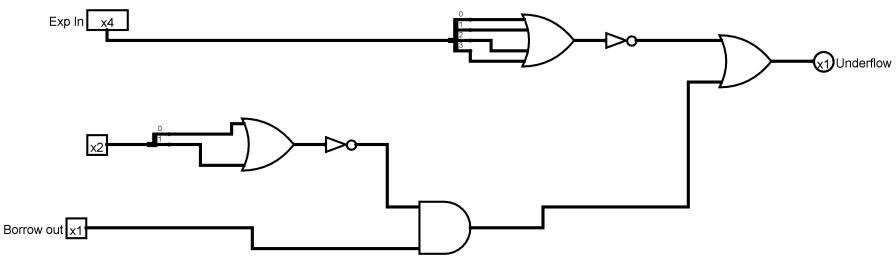

Figure 10: Underflow Circuit

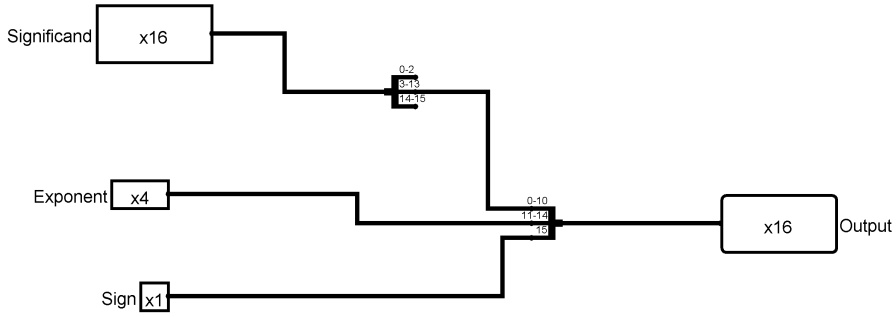

Figure 11: Truncation Circuit

6 ICs and Components Used

Name	**IC**	**Quantity**
4 bit Subtractor	-	3
16 bit Subtractor	-	1
16 bit 2:1 MUX	-	3
16 bit 4:1 MUX	-	1
4 bit 2:1 MUX	-	3
4 bit 4:1 MUX	-	1
16 bit Negator	-	1
4 bit Negator	-	1
Bit Extender (15 to 16 bit)	-	2
16 bit Bit Finder	-	1
16 bit Shifter	-	3
4 bit Adder	IC 7483	5
AND	IC 7408	2
OR	IC 7432	2
NOT	IC 7404	1
XOR	IC 7486	1

7 Simulator Info

Logisim - Java Platform (Version 2.7.1)

8 Discussion

- We designed the circuit in Logisim simulator.
- The Floating Point Adder was designed according to the previously stated addition algorithm.
- As the number of bits reserved for exponents is 4, we chose the bias $2^{4-1} - 1 = 7$. Consequently, we added 7 to the original exponents in order the determine the normalized form of the number.
- As there is a lot of shifting involved in the algorithm, the sum may not fit in the fixed number of bits. Therefore, 3 additional bits (zeroes) were added after the 11 fraction bits of the number while creating the significand. After normalizing the output, these 3 bits were truncated and the result was rounded so that the significand is 12 bits again.
- The number of bits for exponent is 4. Keeping the values 0 and 15 for reserved numbers, the biased exponent can be anything from 1 to 14. So, the bit pattern of value 1 represent the exponent 1 - 7 = -6 and the similarly, the bit pattern of value 14 represent the exponent 14 - 7 = 7. The range of floating point numbers in this assignment can be as small as $\pm 1.00000000000 \times 2^{-6}$ to as large as $\pm 1.11111111111 \times 2^{7}$ which in decimal indicates a range of ± 0.015625 to ± 255.9375.
- As the range of biased exponent is 1 to 14, the overflow flag becomes active whenever the exponent gets larger than 14 and the underflow flag becomes active if the exponents becomes smaller than 1.
- The input exponent can only be 0 when the fraction is also 0, otherwise it indicates denormalized numbers.
- The circuit was tested with diverse data sets and produces acceptably accurate results.