CSE 306 Assignment 2
A1
Group 5

Md. Zarif Ul Alam : 1705010
Jamilus Sheium : 1705012
Naeem Ahmed : 1705014
Md. Shadmim Hasan Sifat : 1705021
Solaiman Ahmed : 1705022

May 30, 2021

1 Introduction

Floating-Point Adder is a combinational circuit which takes two floating points as inputs and provides their sum which is another floating point as output. Its implementation requires some basic n bits adder, subtractor, shifter, multiplexer, comparator and some other basic gates.

Floating-Point Adder is designed to perform "Floating Point arithmetic" which is by far the most used way of approximating real number arithmetic for performing numerical calculations on modern computers.

The floating-point numbers representation is based on the scientific notation: the decimal point is not set in a fixed position in the bit sequence, but its position is indicated as a base power.

All the floating-point numbers are composed by four components:

- Sign: it indicates the sign of the number (0 positive and 1 negative)
- Significant: it sets the value of the number
- Exponent: it contains the value of the base power (biased)
- Base: the base (or radix) is implied and it is common to all the numbers (2 for binary numbers)

The steps involved in the design of a Floating-Point Adder are as follows:

1. Extracting signs, exponents and fractions of both A and B numbers.
2. Treating the special cases:
   - Operations with A or B equal to zero
   - Operations with $\pm\infty$
   - Operations with NaN

   (For simplicity of our design, we are only dealing with the first case)
3. Finding out what type of numbers are given:
   - Normalized
   - Unnormalized

   (For simplicity of our design, we are assuming that numbers are given in normalized form)
4. By comparing the exponent of the numbers, finding out their difference which is actually the required shifting amount and also the smaller & larger number.
5. Shifting the lower exponent number fraction to the right [Exp1 - Exp2] bits. Setting the output exponent as the highest exponent.

6. Working with the operation symbol and both signs to calculate the output sign and determine the operation to do.

7. Addition of the numbers and detection of overflow (carry bit)

8. Standardizing fraction shifting it to the left up the first one will be at the first position and updating the value of the exponent according with the carry bit and the shifting over the fraction.

2 Problem Specification

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 16 bits long with following representation:

Sign	Exponent	Fraction
1 Bit	4 Bits	11 Bits

3 Flowchart

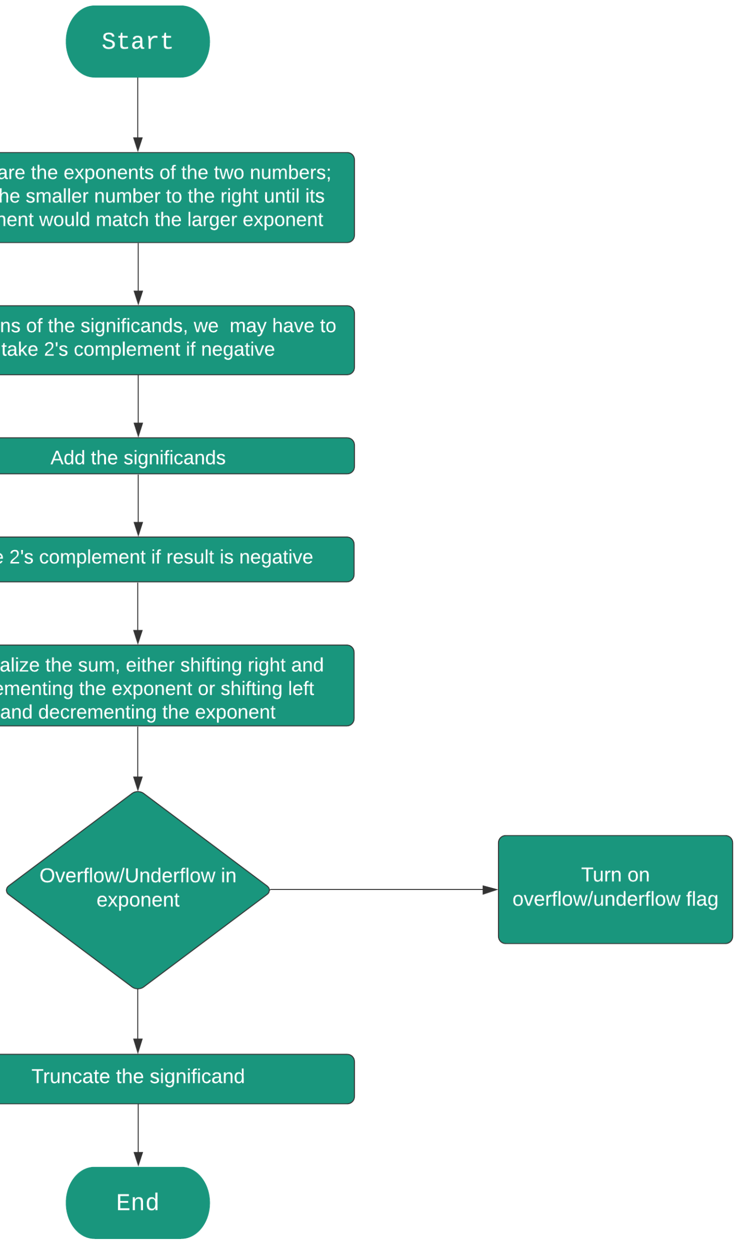
Start
Compare the exponents of the two numbers; shift the smaller number to the right until its exponent would match the larger exponent
Set signs of the significands, we may have to take 2's complement if negative
Add the significands
Take 2's complement if result is negative
Normalize the sum, either shifting right and incrementing the exponent or shifting left and decrementing the exponent
Overflow/Underflow in exponent
Turn on overflow/underflow flag
Truncate the significand
End

Figure 1: Flowchart of Floating Point Adder

4 Circuit Diagram

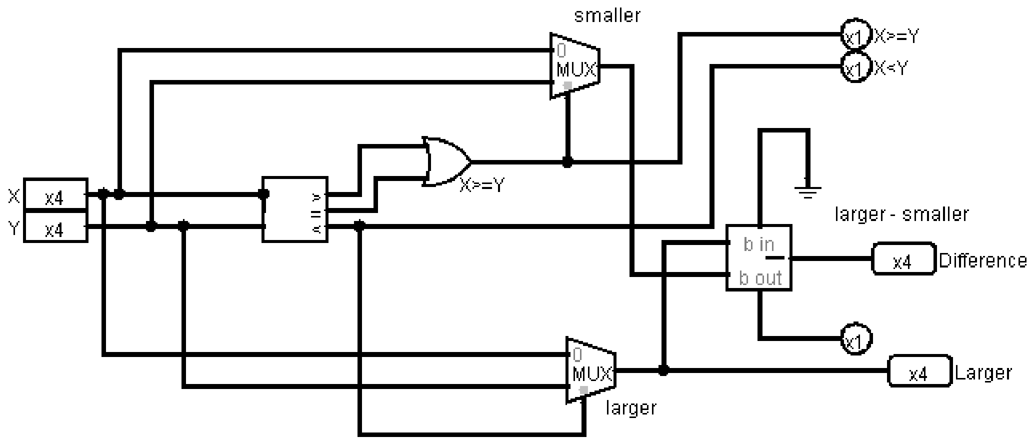

Figure 2: Circuit for comparing the exponents

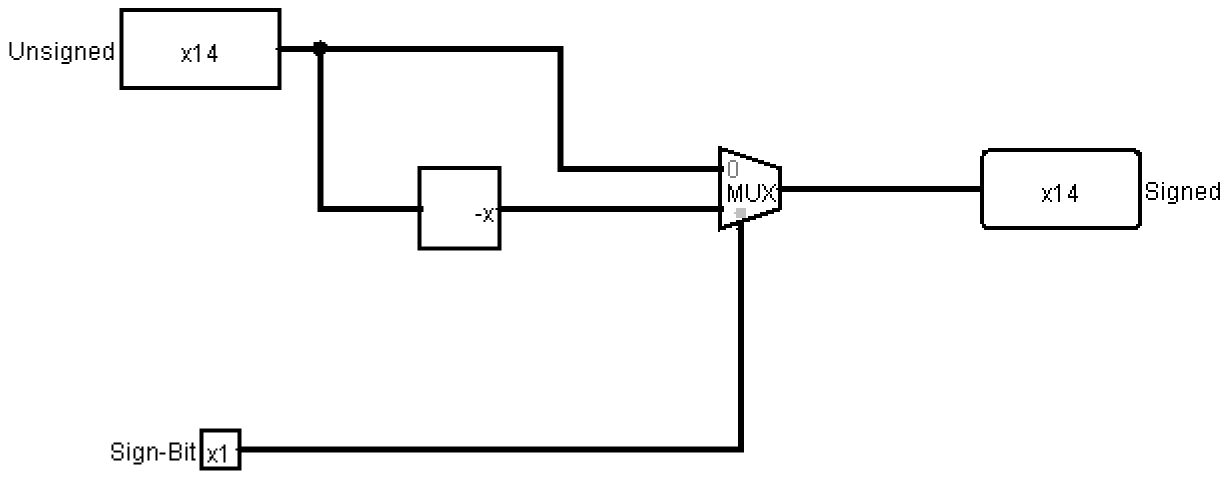

Figure 3: Circuit for adjusting sign bits of inputs

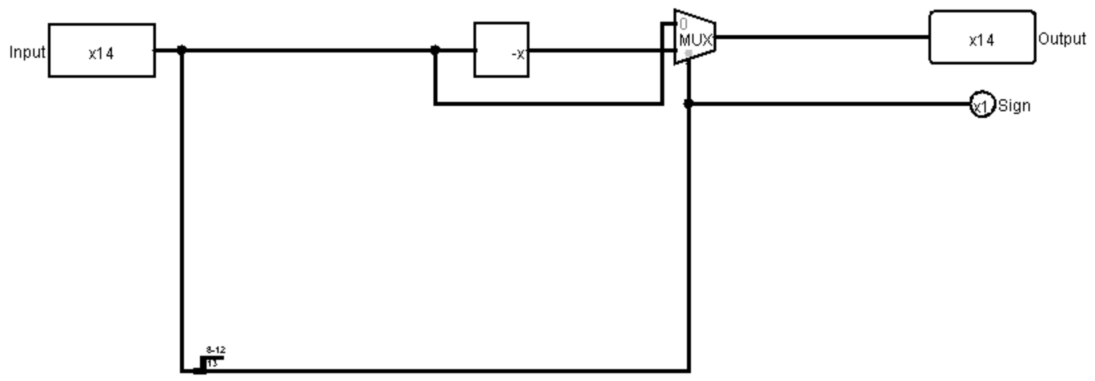

Figure 4: Circuit for complementing negative result

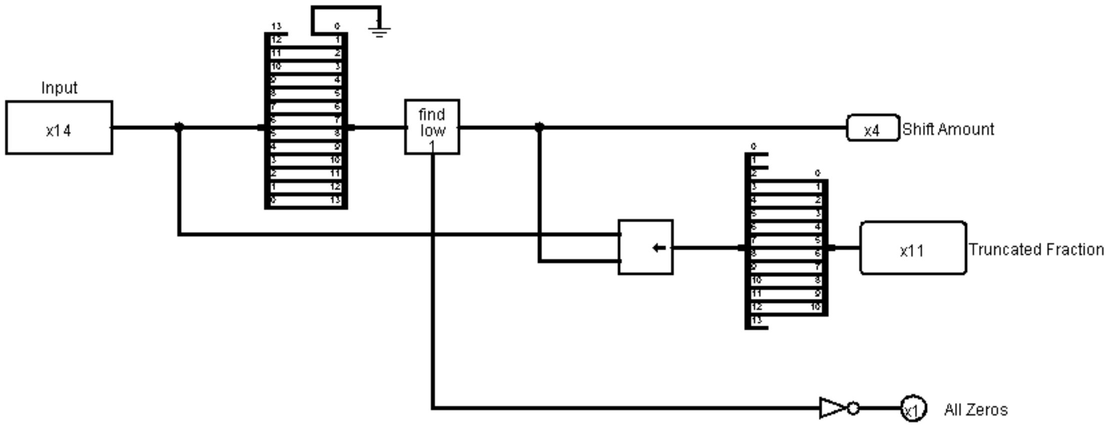

Figure 5: Circuit for normalizing and truncating result

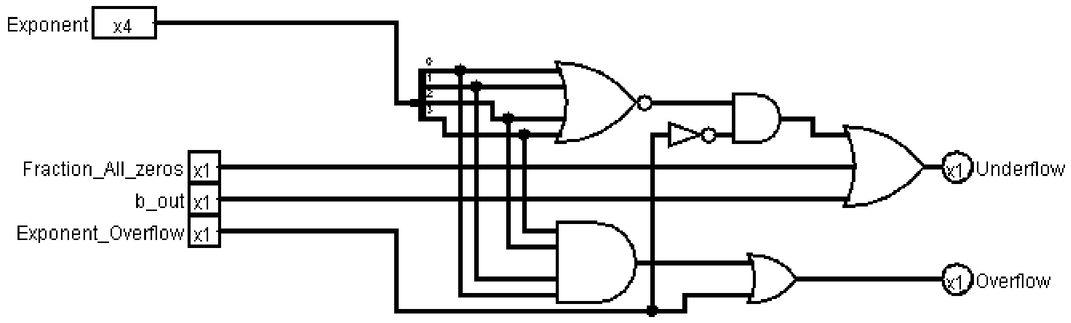

Figure 6: Circuit for checking overflow and underflow of result

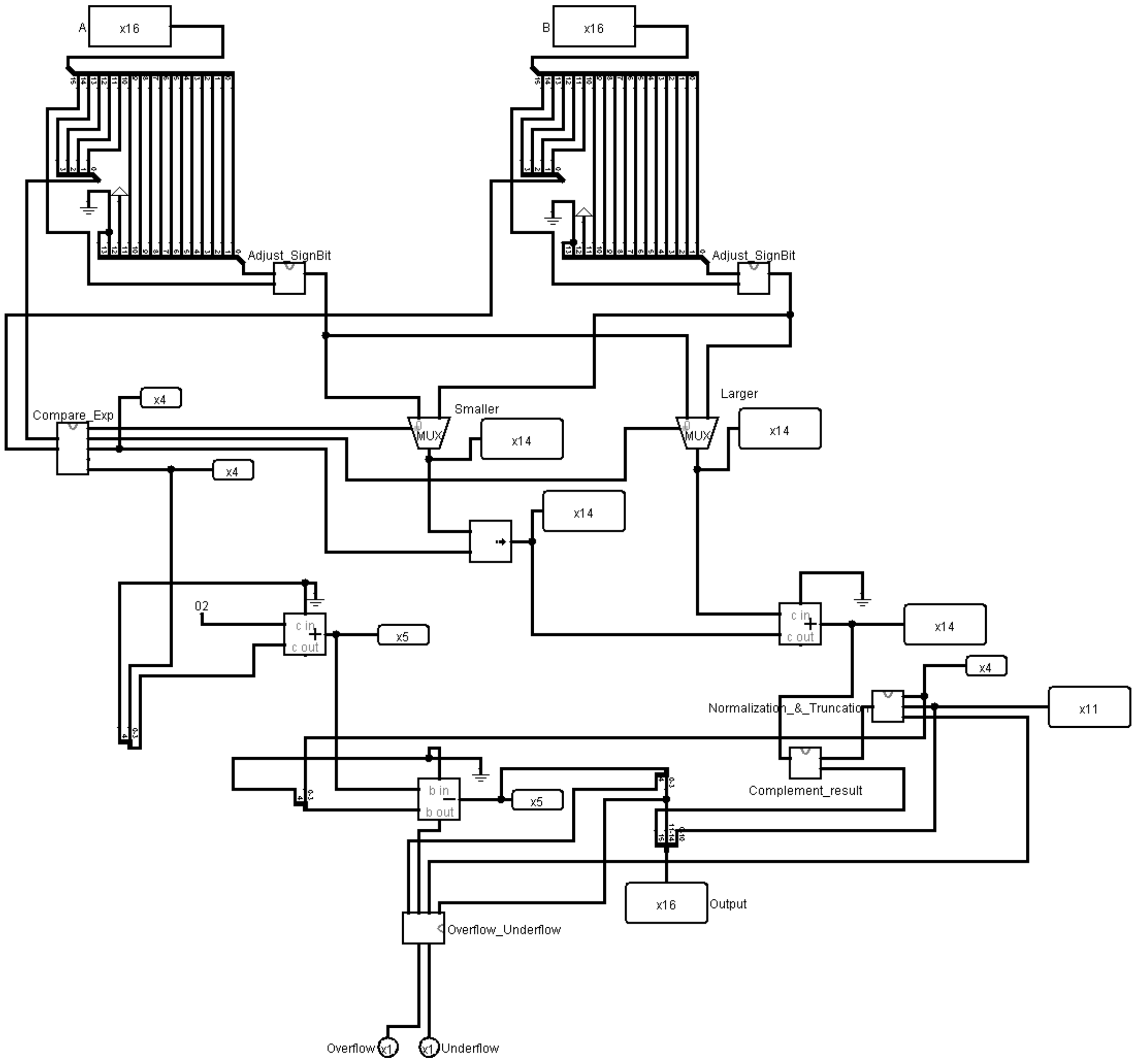

Figure 7: Diagram of complete circuit

5 Block Diagram

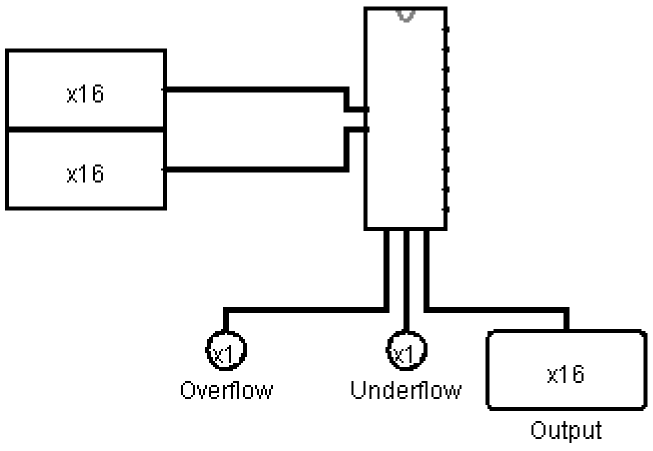

Figure 8: Block Diagram of Floating Point Adder

6 ICs used with count as a chart

Gate	Gate Count	IC	IC Count
2-bit OR	2	7432	1
3-bit OR	1	7411	1
4-bit NOR	1	7425	1
2-bit AND	1	7408	1
4-bit AND	1	7421	1
NOT	1	7404	1
4-bit 2-to-1 MUX	2	74157	2
4-bit Comparator	1	7485	1
4-bit Adder	1	7483	1

Component	Component Count
14-bit 2-to-1 MUX	5
14-bit Negator	3
14-bit Bit-Finder	1
14-bit Left-Shifter	1
14-bit Right-Shifter	1
5-bit Adder	1
14-bit Adder	1
5-bit Subtractor	1
4-bit Subtractor	1

7 Simulator

Logisim Version 2.7.1

8 Discussion

While implementing the circuit, we had to change our design several times. Some designs required a lot of ICs. To optimize number of ICs in our design, we had to discard those. Again, we invested enough of our time in cross-checking corner cases, overflow, underflow conditions for each of sample test cases cautiously. However, because of truncating and rounding up the sum, sometimes we encountered precision loss of minimum one or maximum two bits. Also, in some cases we reused some logic gate outputs to minimize the number of ICs used. Considering all these aspects, we finally implemented the most optimized design we could find.