CSE 210 - Computer Architecture Sessional

Assignment-2: Floating Point Adder

January 9, 2025

Section - C1
Group - 01

Members of the Group:

1. **2105123** - Shatabdi Dutta Chowdhury
2. **2105137** - Arpita Dhar
3. **2105141** - Md Mehedi Hasan Khan

1 Introduction

A floating-point number is a computer representation of a real number, featuring a fixed number of digits both before and after the decimal point (or radix point). A floating-point adder is a digital circuit designed to perform addition and subtraction operations on floating-point numbers. This method enables the representation of numbers that are too large or too small for standard integer representations.

The floating-point format consists of three fields: the sign bit, the exponent field, and the mantissa (or significand) field. The sign bit indicates whether the number is positive or negative. The exponent field allows the significand to be scaled by a power of the base, represented in a biased form to cover a wide range of values. To find the actual exponent, a bias is subtracted from the stored exponent bits. The mantissa field contains the fractional portion of the number, positioned after the decimal point. In this format, the sign bit, exponent field, and mantissa field occupy 1, 11, and 20 bits, respectively. Thus, for this implementation, the exponent bias is calculated as $2^{9-1} - 1 = 255$.

To add two floating-point numbers, a floating-point adder aligns the decimal points before adding the mantissas. Following necessary shifts and adjustments (such as incrementing or decrementing), the resulting exponent is finalized. Normalization and rounding are then applied, and the result's sign is determined based on the signs of the input numbers.

Floating-point adders are used in applications like financial modeling, computer graphics, and scientific or engineering computations. They are also utilized in co-processors for fast, hardware-accelerated floating-point arithmetic, performing such calculations more quickly than the main processor. This capability is especially valuable in applications that require high-precision floating-point calculations, such as data analysis and scientific simulations.

2 Problem Specification

The task involves creating a circuit for a floating point adder that accepts two floating points as inputs, adds their sum, and outputs another floating point. Every floating point will have a length of 32 bits and be represented as follows:

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

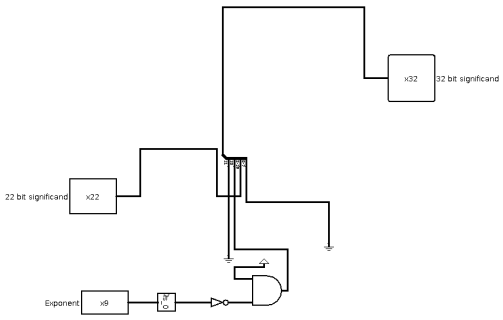
Table 1: Problem specification

3 Detailed Circuit Diagram of Modules

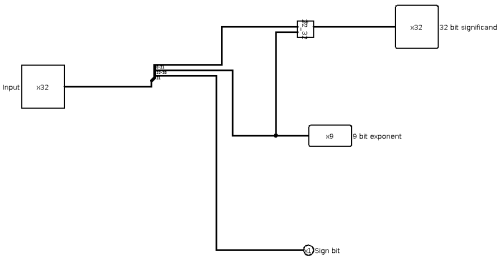
In order to maintain modularity in the floating point adder design, multiple libraries have been built and implemented. Descriptions and usages of the libraries are given below:

3.1 Processing the Input

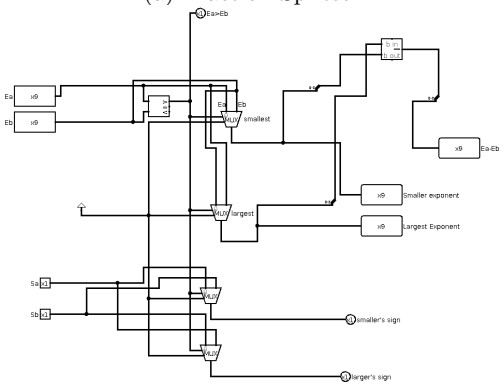
- A fraction splitter that extends the 22-bit significand to 32 bits [**Fraction Splitter**].
- An input splitter splits the 32-bit floating-point number into an 9-bit exponent and a 22-bit significand [**Input Splitter**].
- An exponent differentiator that calculates the difference between the exponents of two inputs [**Exponent Differentiator**].
- A circuit that determines which input has the greater value, compares exponents, calculates exponent difference, and makes other control decisions [**Sample Input**].



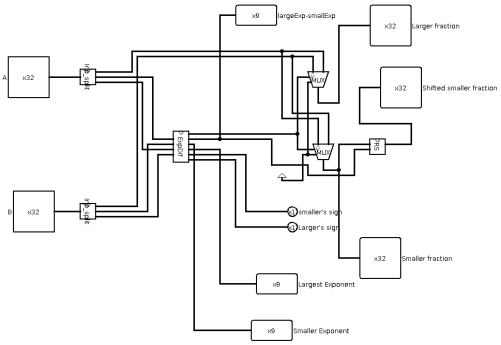
(a) Fraction Splitter



(b) Input Splitter



(c) Exponent Difference



(d) Sample Input

Figure 1: Input Processing

3.2 Adder Library

An adder circuit (adder32bit.circ) has been included in the adder32bit library which performs the most crucial part of the FPA, which is to add the significands of the two given input floating point numbers.

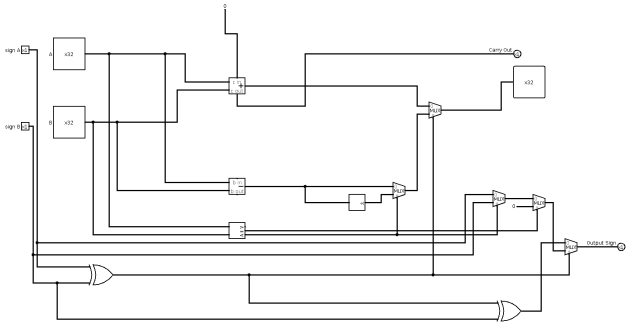
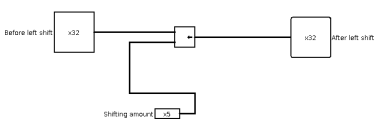


Figure 2: 32 Bit Adder Circuit

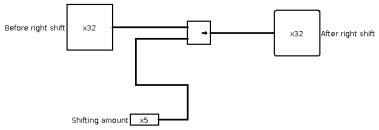
3.3 Shifter

Shifters are required in floating point adders to shift the fraction in order to normalize or balance with the other operand. We have implemented random left shift and random right shift with splitters and muxes from the Logisim built-in library. The circuits are:

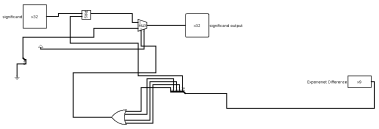
- Random left and right shifter (Can shift any number of bits up to 31 bits) [Random Left Shift, Random Right Shift]
- Right shifter that will make every bit 0 if it needs shfting more than 31 bits [Right Shifter]



(a) Random Left Shift



(b) Random Right Shift



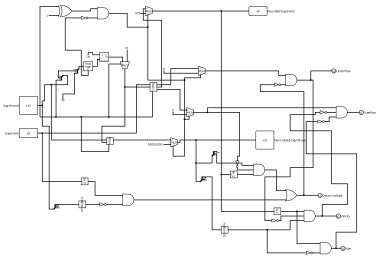
(c) Right Shifter

Figure 3: Example of Shift Operations

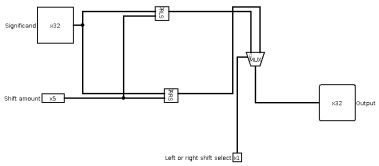
3.4 Normalization

The circuit normalizes the output by applying the necessary number of bit shifts with the aid of an encoder library. Additionally, it produces an 9-bit value that is added to the exponent to finish the normalizing process.

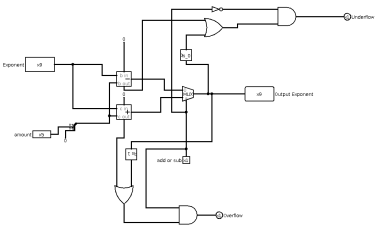
- Normalizing
- Left-Right Shift Normalizer
- Adder Subtractor
- Rounded Normalizer
- Zero Checker
- One Checker



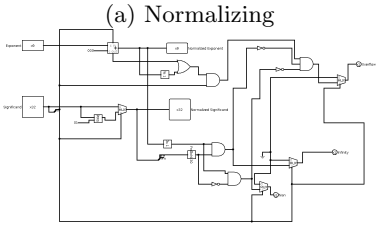
(a) Normalizing



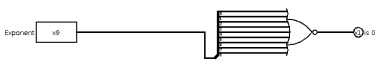
(b) Left-Right Shift Normalizer



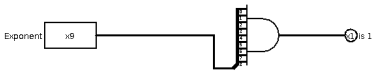
(c) Adder Subtractor



(d) Rounded Normalizer



(e) Zero Checker



(f) One Checker

Figure 4: Figure 1 - Various Components in the System

3.5 Rounding

It contains a comparator and and a full adder and does the rounding of the mantissa.

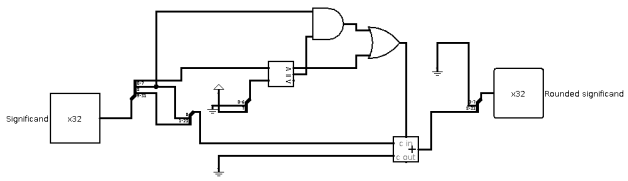
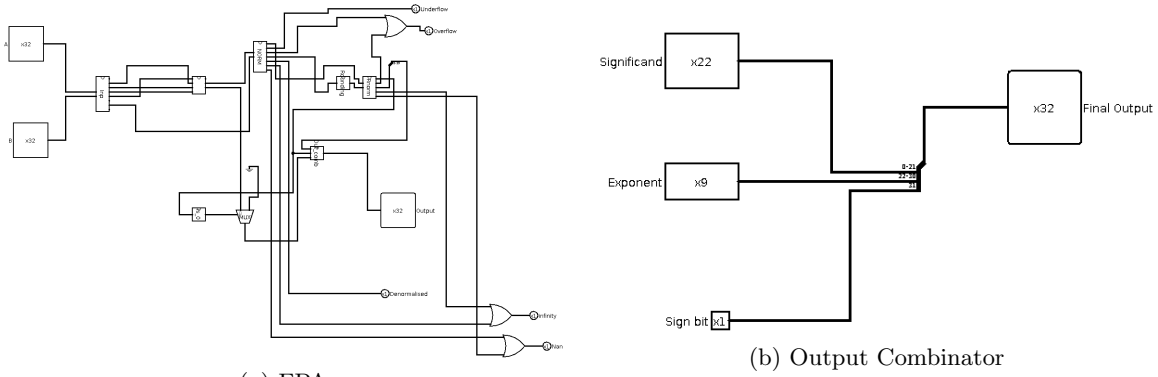


Figure 5: Rounding Circuit

3.6 Floating Point Adder

The second module, called FPA.circ, uses the adder library and other components to fully create a floating point adder. It has the real floating point adder, or circuit FPA. The output processor circuit that merges the sign, exponent and significand of the result is also included in this module.[Output Combinator]



(a) FPA

(b) Output Combinator

Figure 6: FPA and Output Combinator

3.7 Third Party Libraries

One third party library 7400-lib.circ is used to incorporate 7400 series ICs in the floating point adder implementation.

4 Flowchart of the Addition/Subtraction Algorithm

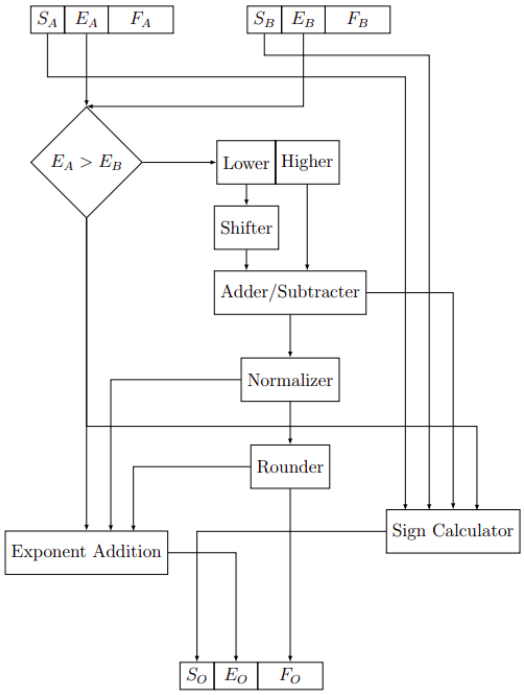


Figure 7: Flowchart of the algorithm

5 High-level Block Diagram of the Architecture

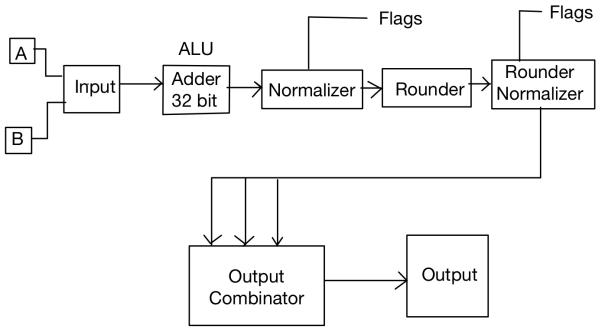


Figure 8: High level block diagram of FPA

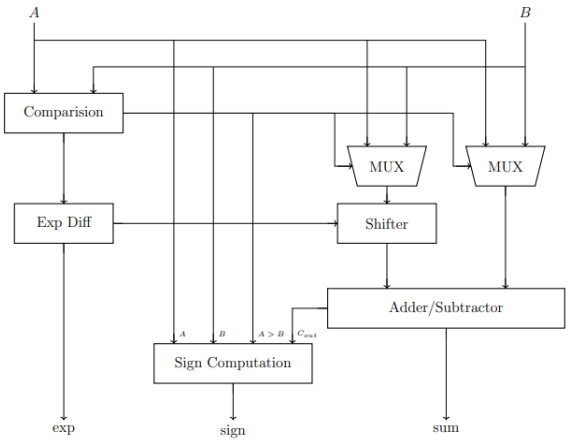


Figure 9: High level block diagram of Adder

6 ICs Used with Count as a Chart

IC	Quantity
IC 7404	2
IC 7408	6
IC 7432	5
IC 7486	3
IC 74157	18
Total	**34**

Table 2: ICs Used with Quantity

7 Simulator used Along with the Version Number

Logisim 2.7.1 has been used for simulating the floating point adder circuit.

8 Contribution of each member

2105123 - Design and Software Implementation
2105137 - Design, Software Implementation and Report
2105141 - Software Implementation and Report

9 Discussion

We focused on ensuring the novelty of our floating-point adder (FPA) implementation. Building the entire circuit from scratch would have been overly complex, so we opted to design and combine separate modules. This modular approach also facilitated an efficient division of tasks among team members.

We encountered several challenges during the process. The first was designing a shifter capable of shifting by arbitrary amounts. We achieved this by using Random Left and Right shifters.

Handling negative numbers as inputs presented another challenge. After analysis, we realized the behavior of the circuit depended on the signs and relative magnitudes of the inputs. These considerations led to four distinct cases, which we addressed using a combination of adders, subtractors, complementers, XOR gates, and multiplexers. This systematic approach allowed us to handle negative inputs efficiently.

Overflow,Underflow,Denormalize,Nan.Infinity are handled from Normalizing and Rounded Normalizer circuit.

To balance functionality with simplicity, we integrated Logisim's built-in circuits with our custom-designed components. This hybrid strategy ensured that our implementation remained both minimalist and fully operational.

Overall, designing the Floating Point Adder was a rewarding experience. It deepened our understanding of the intricate processes involved in floating-point arithmetic while challenging us to find creative solutions to complex design problems.