BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CSE 306

Computer Architecture Sessional

ASSIGNMENT NO. 02

DESIGN OF 32-BIT FLOATING POINT ADDER CIRCUIT

SECTION: B2

GROUP: 06

GROUP MEMBERS: 1905104, 1905105, 1905107, 1905108, 1905109

1. Introduction

A floating Point Adder is a combinational circuit that is used to add two floating points. As input, it takes two floating points and as output, it gives another floating point. This Floating Point Adder represents one kind of ‘Floating Point Arithmetic which is approximately real number arithmetic.

The term ‘Floating Point’ means the radix point can float to any position in the number whereas, in a fixed point system, there is a fixed position for the radix point. The number is represented like a scientific notation.

All the floating point number has four things in common. A bit to represent the sign, the value of the number (generally between 0 and 1) or fraction, the value of base power or exponent, and the radix point or called the base. In fact, the number is made of four-part:

- Sign
- Fraction
- Exponent
- Base

As the base(for binary it’s 2) is common for all numbers it need not be stored. The other three are represented in Binary format.

To implement a Floating Point Adder, adder, shifter, multiplexer, comparator, and some basic gates are needed.

Design steps:

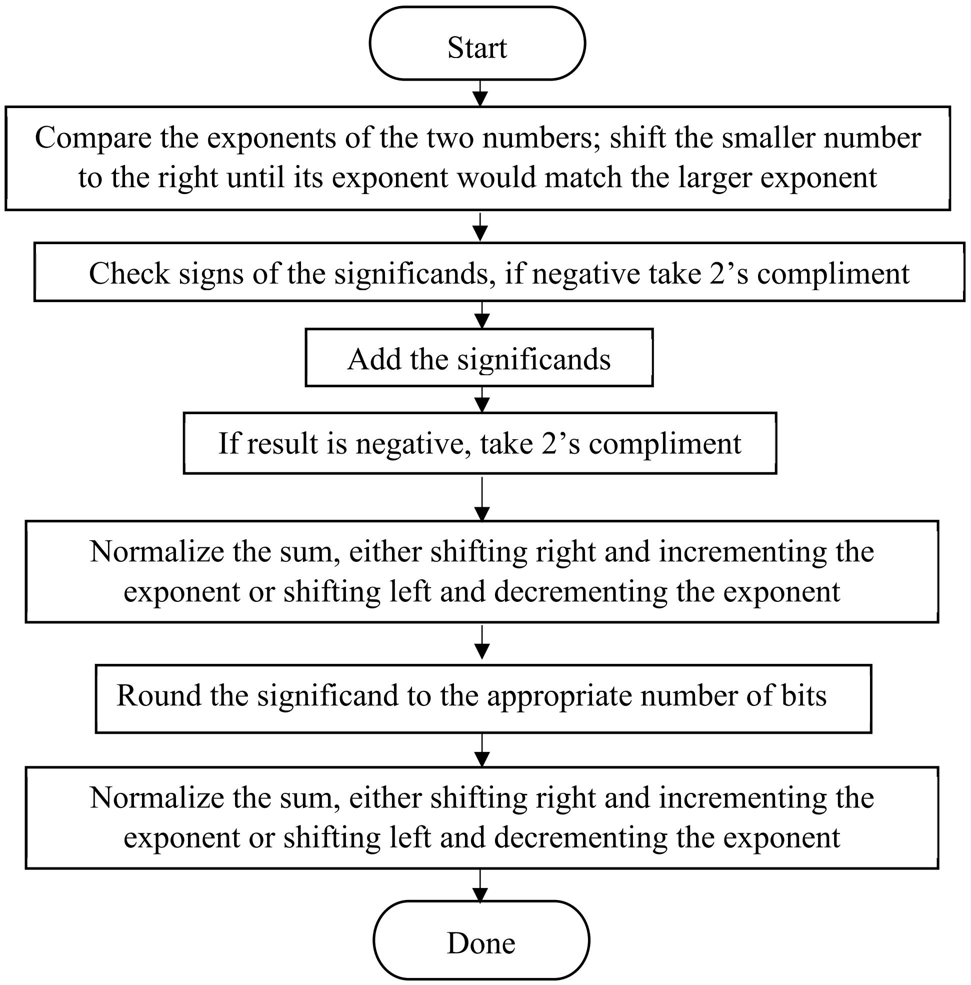
- Separating sign bit, exponents, and fractions from the two numbers
- Finding whether the number is normalized or not, valid or invalid
- Comparing two numbers
- Shifting the smaller number to make the exponent equal to both numbers
- Depending on the sign bit doing an addition
- Detection of overflow
- Setting up the number according to the format

2. Problem Specification

Designing a Floating Point Adder circuit which takes two 32-bit long floating points as inputs and provides their sum, and another 32-bit long floating point as output. The Floating Point follows the following representation:

Sign	Exponent	Fraction
1 bit	12 bits	19 bits (Lowest bits)

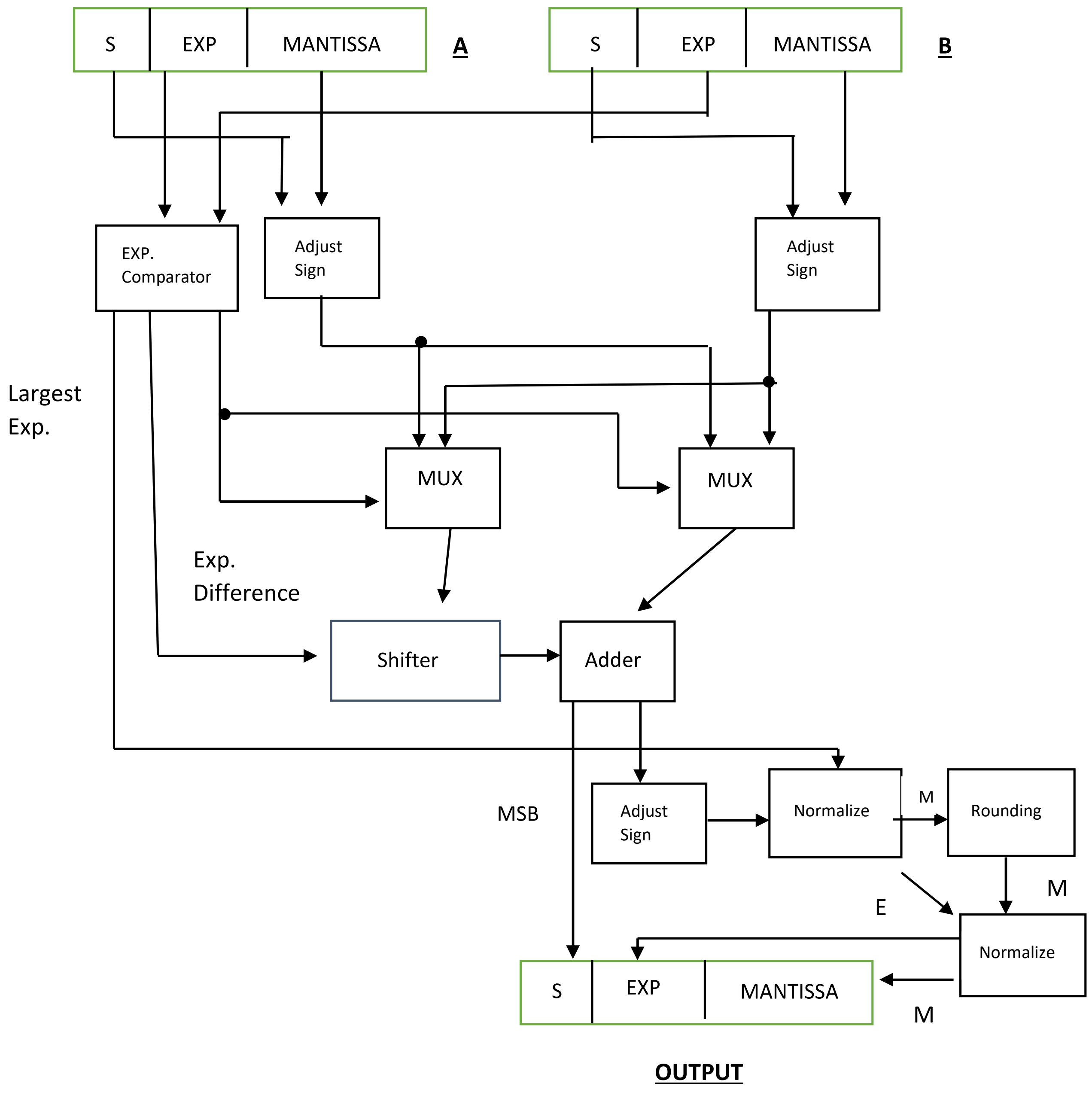
3. Flowchart



Start
Compare the exponents of the two numbers; shift the smaller number to the right until its exponent would match the larger exponent
Check signs of the significands, if negative take 2's compliment
Add the significands
If result is negative, take 2's compliment
Normalize the sum, either shifting right and incrementing the exponent or shifting left and decrementing the exponent
Round the significand to the appropriate number of bits
Normalize the sum, either shifting right and incrementing the exponent or shifting left and decrementing the exponent
Done

Figure 1: Flowchart of Floating-Point Adder

4. **High-Level Block Diagram**



5. Circuit Diagram

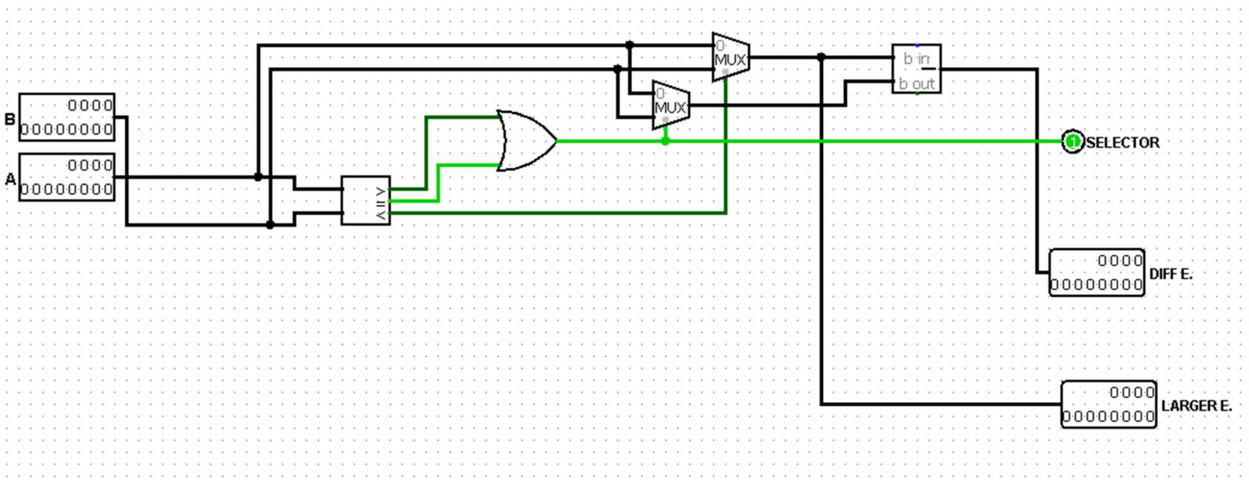


Figure 2: Circuit for comparing the exponents and extending bits

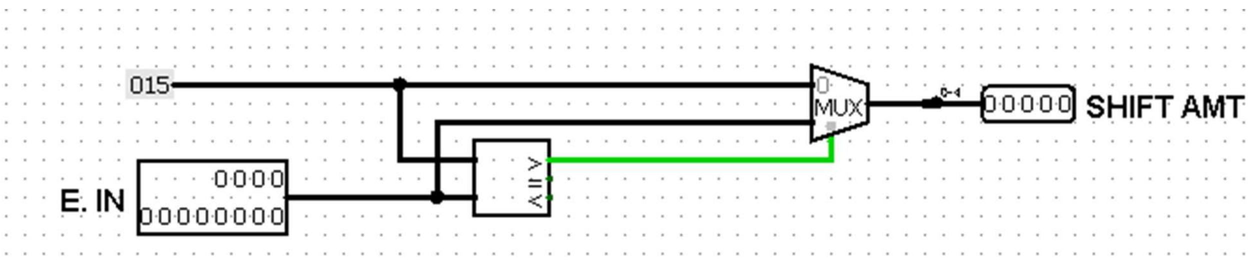


Figure 3: Circuit for shifting bits

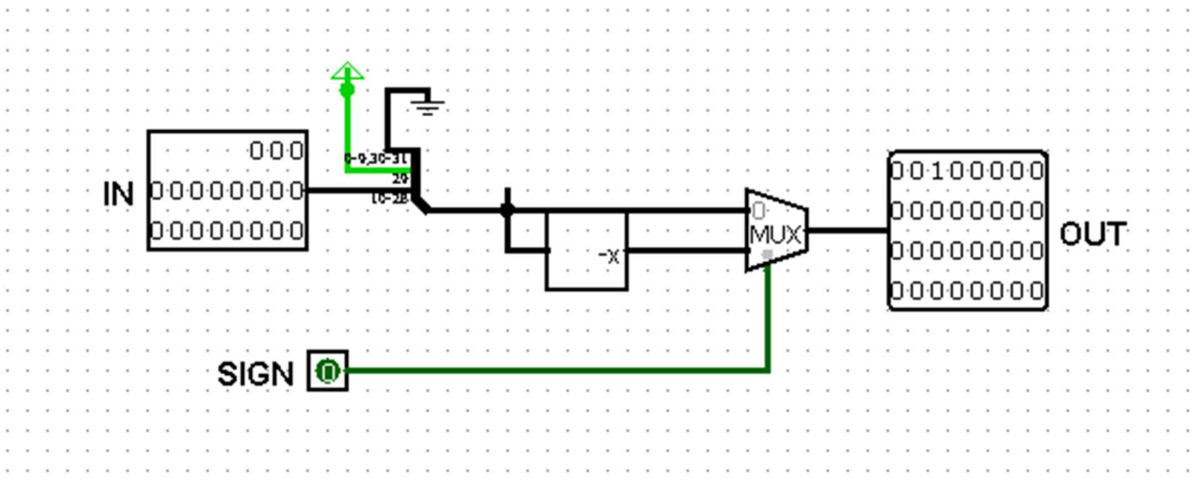


Figure 4: Circuit for complementing

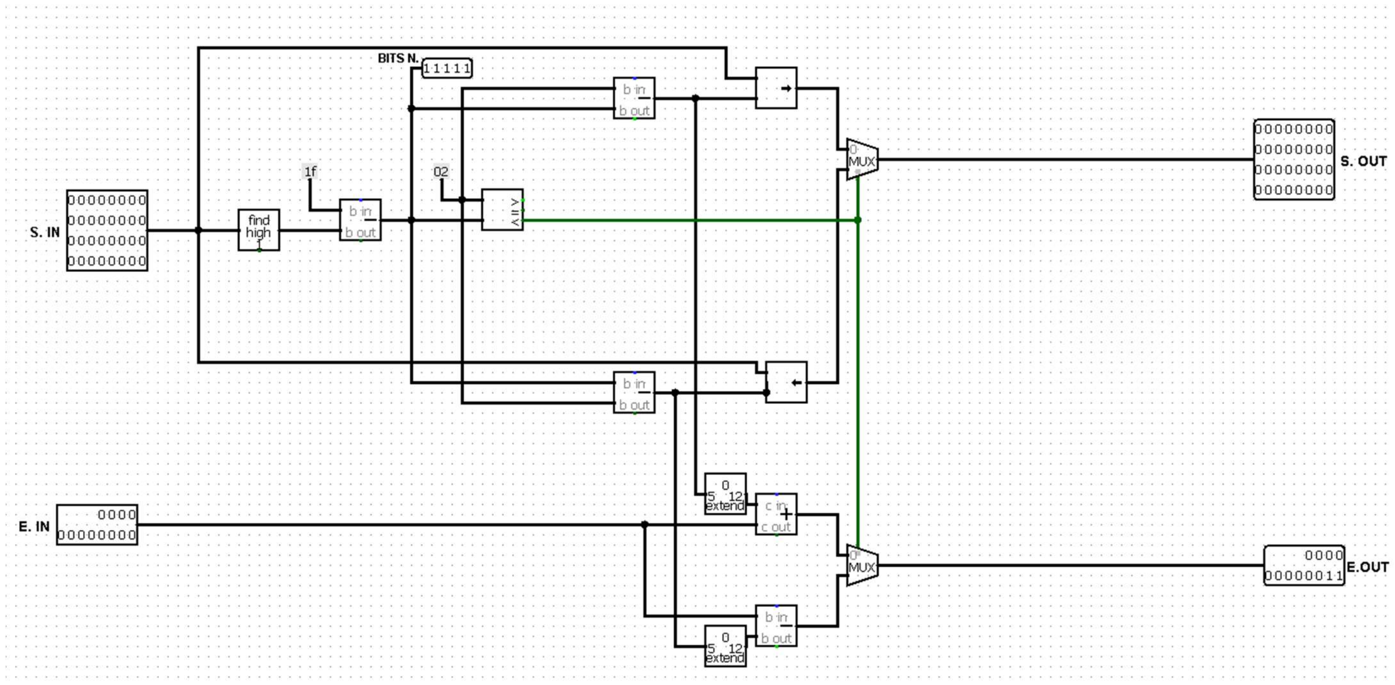


Figure 5: Circuit for normalizing the result

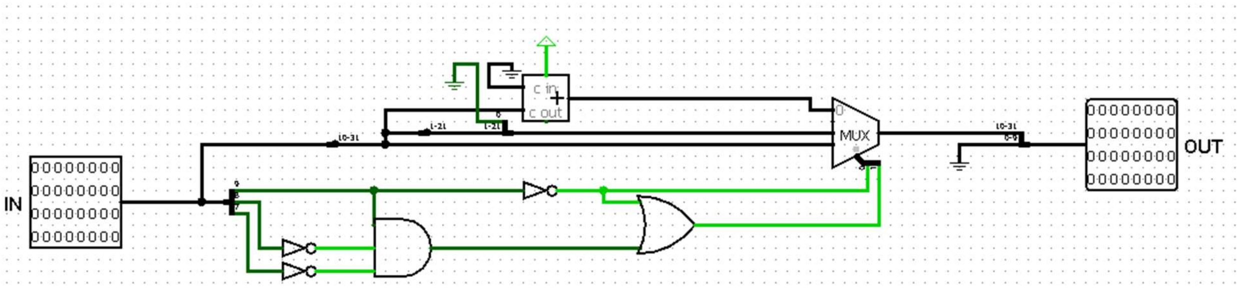


Figure 6: Circuit for rounding the result

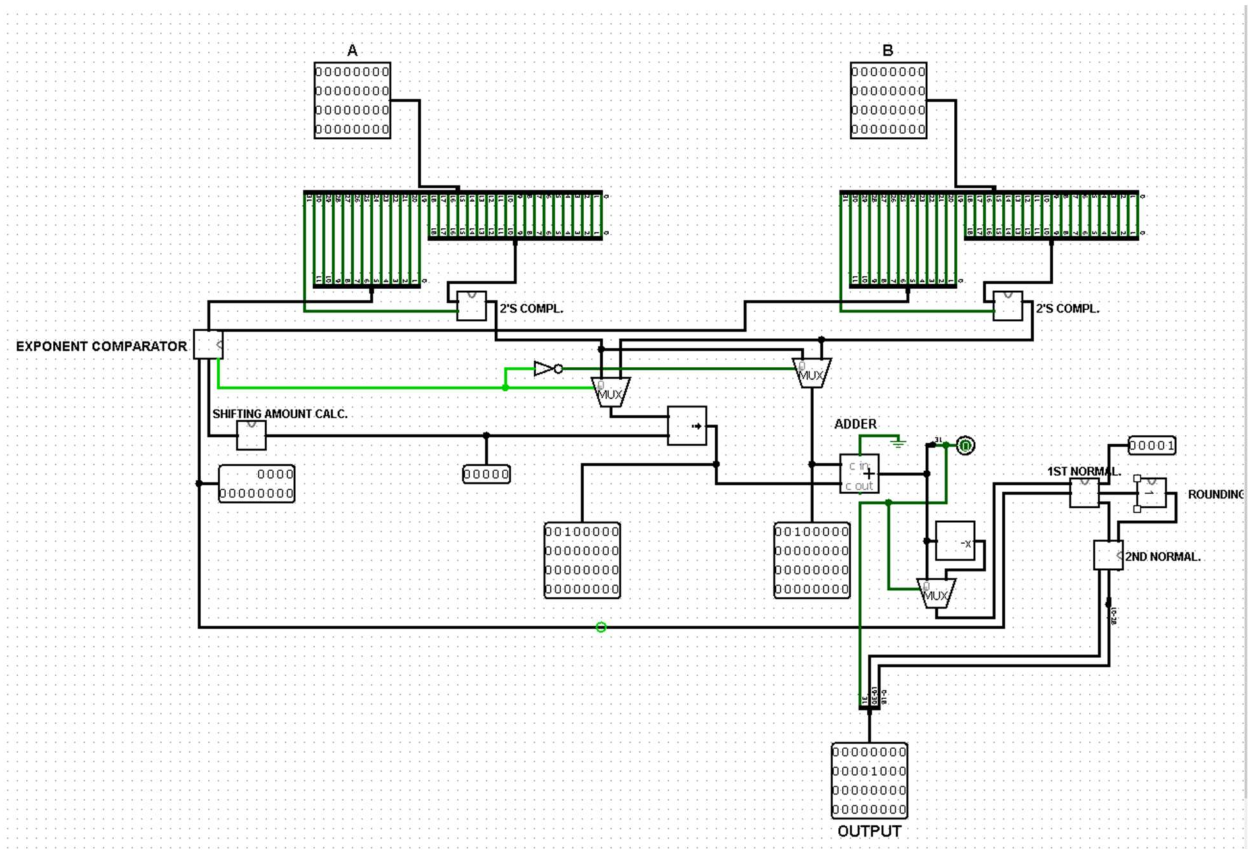


Figure 7: Diagram of the complete circuit

6. <u>Block diagram</u>

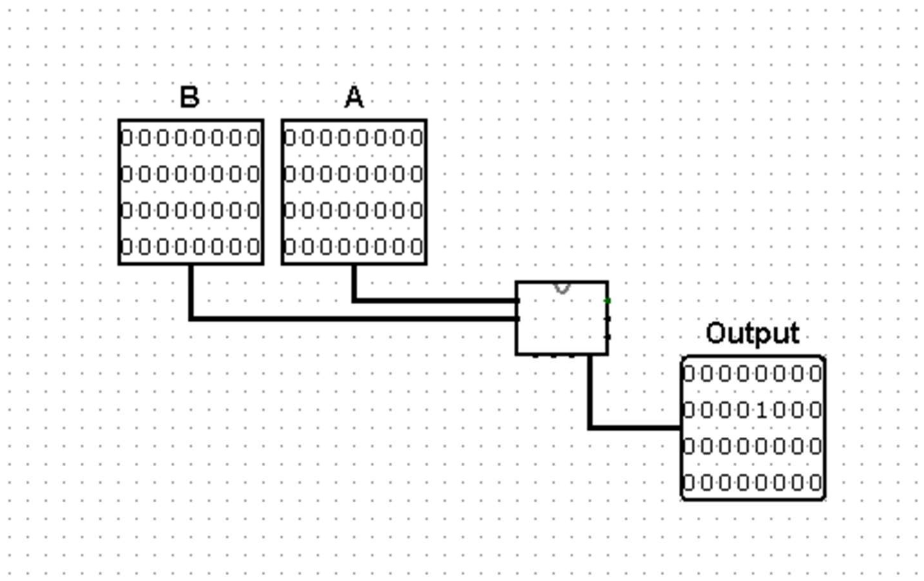


Figure 8: Block Diagram of Floating Point Adder

7. <u>ICs used with count as a chart</u>

Gate	**Gate Count**	**IC**	**IC Count**
2-Bit OR	2	7432	1
2-Bit AND	2	7408	1
1-Bit NOT	4	7404	1

Component	**Component Count**
12-Bit 2x1 MUX	5
32-Bit 2x1 MUX	7
22-Bit 4x1 MUX	1
12-Bit Adder	2
22-Bit Adder	1
32-Bit Adder	1
5-Bit Subtractor	6
12-Bit Subtractor	3
32-Bit Negator	3
Splitter	15
5-Bit Comparator	2
12-Bit Comparator	2
32-Bit Logical Left Shifter	2
32-Bit Logical Right Shifter	2
32-Bit Arithmetic Right Shifter	1
5 to 12-Bit Bit Extender	4
32-Bit Bit Finder	2

8. Simulator

Logisim Version 2.7.1

9. Discussion

We designed a 32-bit floating point adder that can add two floating point numbers with significand/mantissa of 19 bits and exponential of 12 bits long. The circuit we designed is a combinational circuit, which does not have any previous state dependency. We followed the steps described in the flowchart (figure 1). We did not use an ALU instead, we used a 32-bit adder as we did not need all the operations of an ALU. And so that we can use the adder for different signed floating-point additions, we took the mantissa in 2's complemented form. We took extra 10 bits after the 19-bit mantissa representation for more precision. And took a '1' before the 19-bit mantissa which represents the '1' before the floating point. And took two extra 0 bits for storing the carry bits. After adding the mantissa, we normalized it. And after the first normalization, we rounded the mantissa and then again normalized it. This should have gone through a loop until no more rounding was possible. But for simplicity in design, we normalized it twice and rounded it once. And as there were 19 significant bits in the mantissa if we calculate the relative precision: $(19 \times log_{10}2) = (19 \times 0.3) \approx 5$ decimal digits of precision.

Moreover, we used the built-in modules and components of Logisim to do the calculations. We were cautious at the time of designing so that we don't lose any precision.