BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY

CSE-306: COMPUTER ARCHITECHTURE SESSIONAL

32 BIT FLOATING POINT ADDER

Section: B1

Group: 6

Members: 2005074

2005084

2005085

2005088

2005090

Submitted to

Saem Hasan

19 January, 2023

1. Introduction :

A floating-point adder is a critical component in computer arithmetic designed to perform arithmetic operations on real numbers represented in the floating-point format. Floating-point arithmetic is essential for handling a wide range of numerical computations, particularly in scientific, engineering, and mathematical applications where precision and a broad range of values are crucial. Unlike fixed-point arithmetic, which has a fixed number of digits after the decimal point, floating-point arithmetic allows for dynamic scaling of the decimal point, providing a broader representation of both small and large numbers. This flexibility makes floating-point representation suitable for applications where a uniform precision across a wide magnitude range is required. 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. Some components by which floating point numbers are composed of are:

- Sign: It indicates whether the number is positive (sign bit 0) or negative (sign bit 1)
- Base: The base for binary numbers is 2 and it is applied here.
- Significant: It determines the value of the number.
- Exponent: It gives the value of power and which is of base 2 and here biased power is used.

Our design comprises of some steps. They are:

- Splitting the sign, exponent and significant part from both the binary numbers.
- Making the exponent of both numbers equal by right shifting the significant part of the number with smaller exponent.
- After we have made the exponents same, we add the significant parts of the two numbers.
- Then if the sum is not in normalized form, then we convert the sum into normalized form by right or left shifting the sum.
- At the end, rounding is done using the GFS bits (we took extra 3 LSB bits as 0 at the start as GFS bits) and on the basis of that rounding is done by either adding 1 with the number or not adding anything.
- After rounding we have to check again if it is in normalized form or not, if not then we make it normalized again.

- Moreover for checking the underflow or overflow of the exponent part, we kept two extra MSB bits as 0 and on the basis of them we determined the underflow and overflow flags.

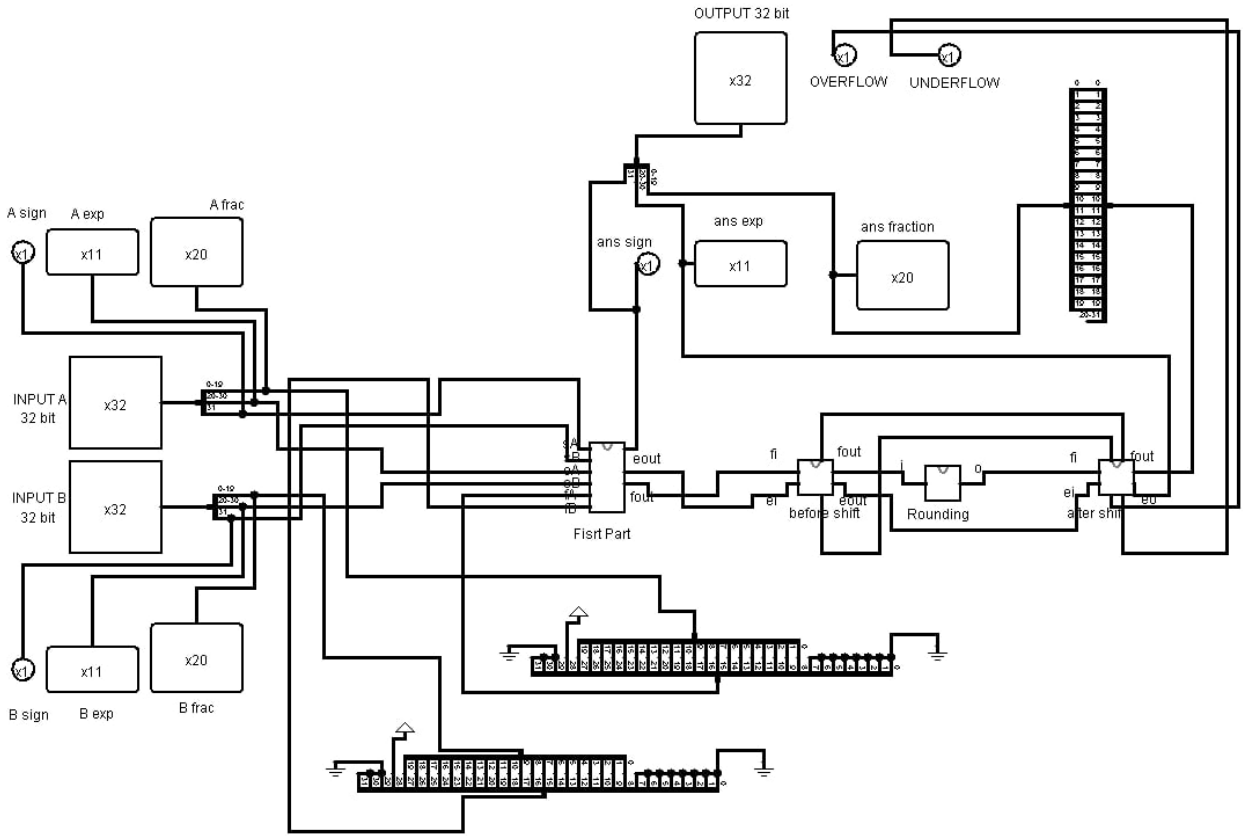
2. Problem Specification :

Design a floating-point adder that takes two floating-point number as inputs and provides their sum as floating-point number as output. Each floating-point number will consist of 32-bit binary representation. The representation will be:

Sign Bit	Exponent Bits	Fraction Bits
1 Bit	11 Bits	20 Bits

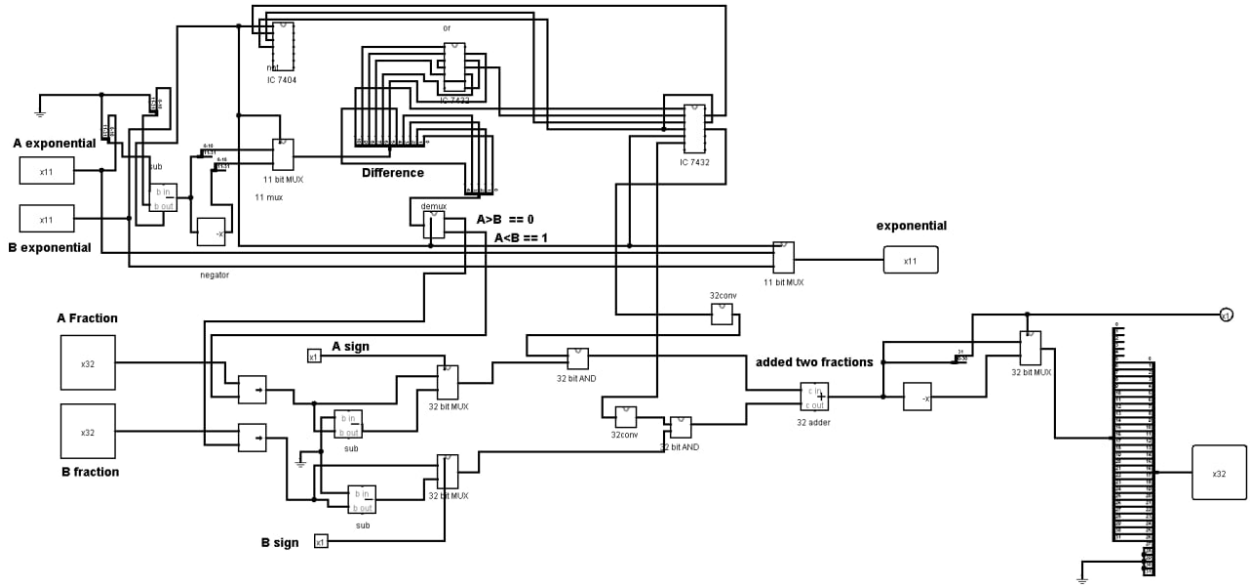
3. Flow Chart of the Addition/Subtraction Algorithm

4. High Level Block Diagram of the Architecture

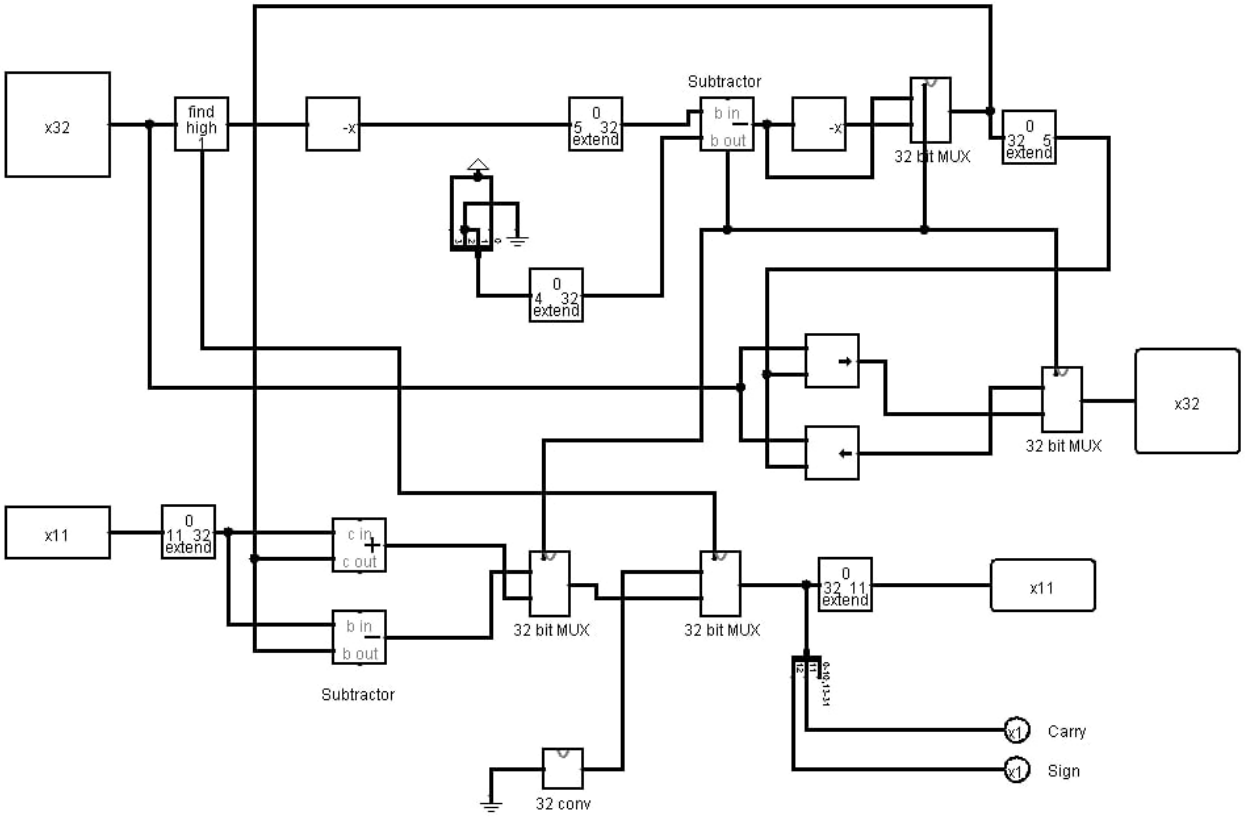


5. Detailed Circuit Diagram of the Important Blocks

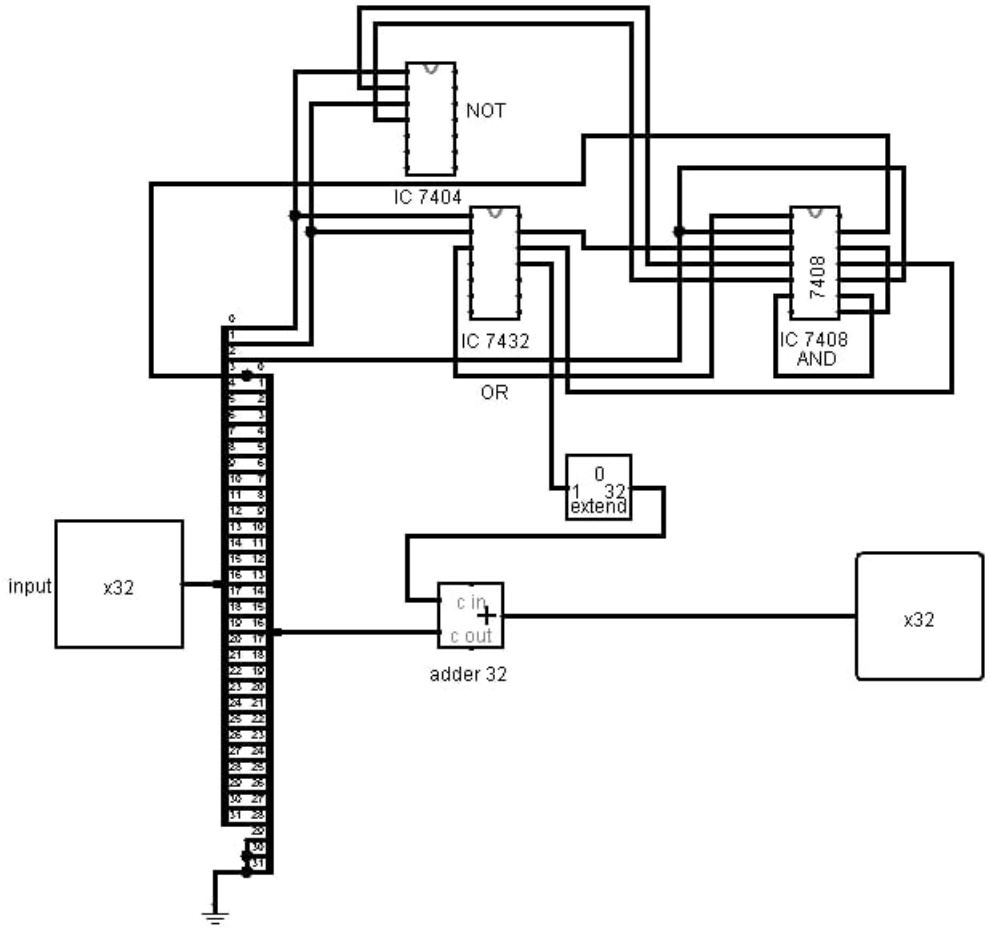
i. Block Diagram of the circuit responsible for balancing the fraction numbers and addition of them



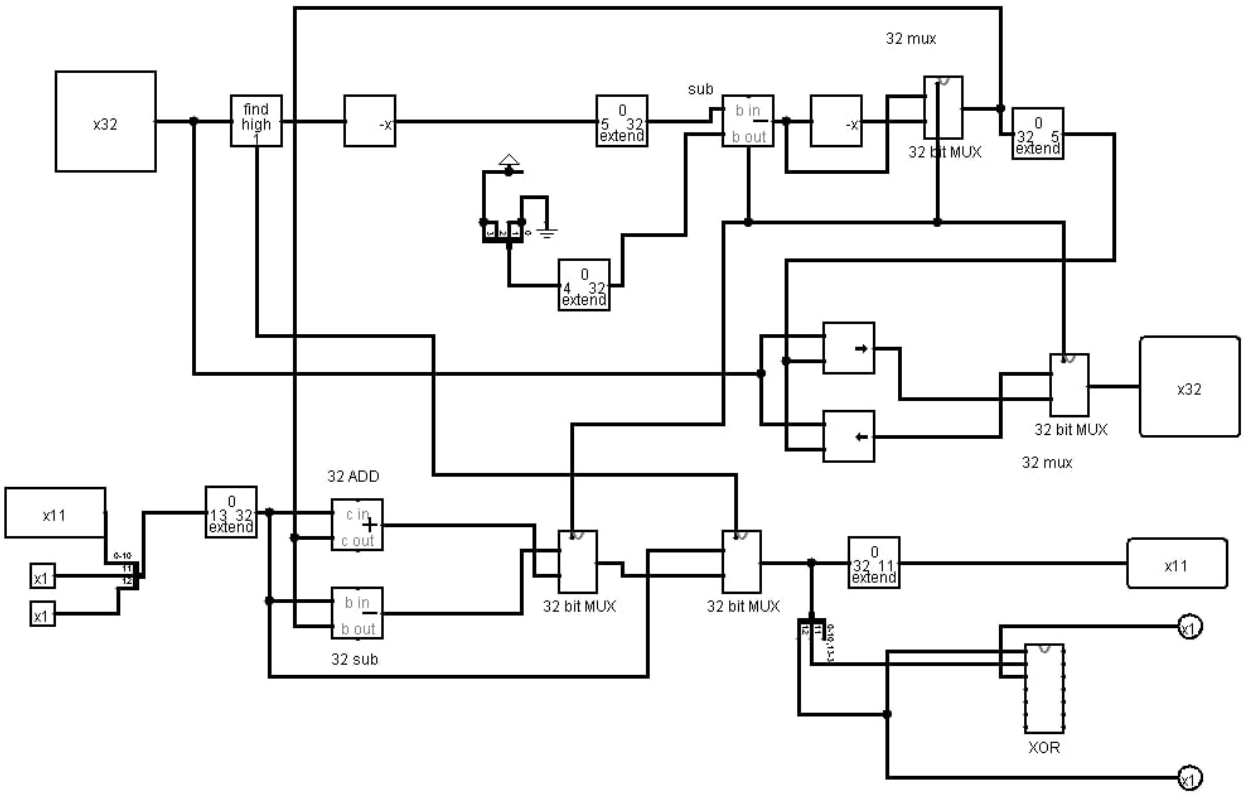
ii. Block Diagram of the circuit responsible for shifting the fraction to preferred position



iii. Block Diagram of the circuit responsible for rounding the fractional answer



iv. Block Diagram of the circuit responsible for shifting the fractional number again after rounding operation



6. ICs used with count as a chart

Gate	IC	IC Count
2-bit OR	7432	3
2-bit AND	7408	20
NOT	7404	3
2-bit XOR	7486	1
4-bit 2-to-1 MUX	74157	94
		Total IC Count: 121

Component	Component Count
32-bit Right-Shifter	4
32-bit Left-Shifter	2
32-bit Adder	4
32-bit Subtractor	7
32-bit Negator	5

5-bit Negator	1
32-bit Bit-Finder	2

7. Simulator

Logisim Version 2.7.1

8. Discussion

1. The 20-bit fraction is extended to 32 bits. The first 3 bits are reserved for sign and carry. The 4th bit is set to 1 to represent the 1 before the decimal point. The next 20 bits represent the significant part which is taken directly from the input. The last 8 bits are reserved for ensuring precision while rounding.

2. The in-built shifters can shift the input up to 32 bits. We manually set the fraction zero when more than 32 bits are needed to be shifted.

3. We ensured that every bit of the fraction and the exponent is zero when the answer is zero.

4. 8 extra bits are added to the right of the fraction to ensure accuracy. Rounding is done with 3 GRS bits. Truncating and rounding up is handled with K-map, rounding to even is handled separately.

5. For the underflow and overflow flags, 2 extra bits are added to the left of the exponent. They are used as sign and carry bit and the flags are carried out accordingly. If the sign bit is 1 in any stage, underflow flag is set to be 1. And the overflow flag is the XOR of the sign and the carry bit.