CSE 306 (Computer Architecture Sessional)

Experiment No: 02

Name of the experiment:

Floating Point Adder Software Implementation

Group No.	**06**
Section	A1
Department	CSE
Group Members	1705026 1705027 1705028 1705029 1705030
Date of Performance:	27-05-2021
Date of Submission:	30-05-2021

Introduction:

The prime objective of this assignment was to build a Floating Point Adder which performs the summation of two signed floating point numbers. The Floating Point Adder circuit was designed using the available logic gates, plexers and arithmetic units.

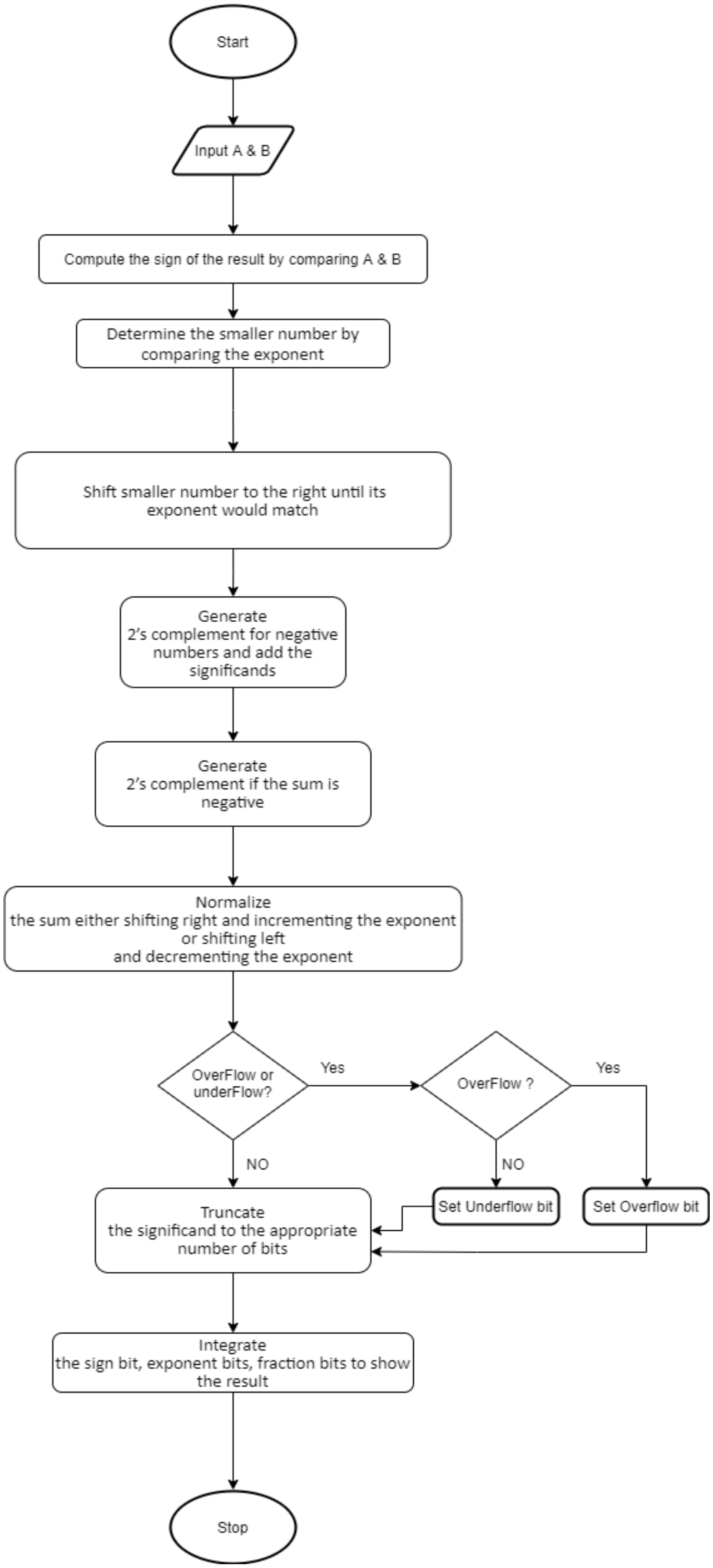
Problem Specification:

A Floating Point Adder has to be designed efficiently which takes two signed floating point numbers (Input A and Input B). Each floating point number is 16 bits long represented in this following format:

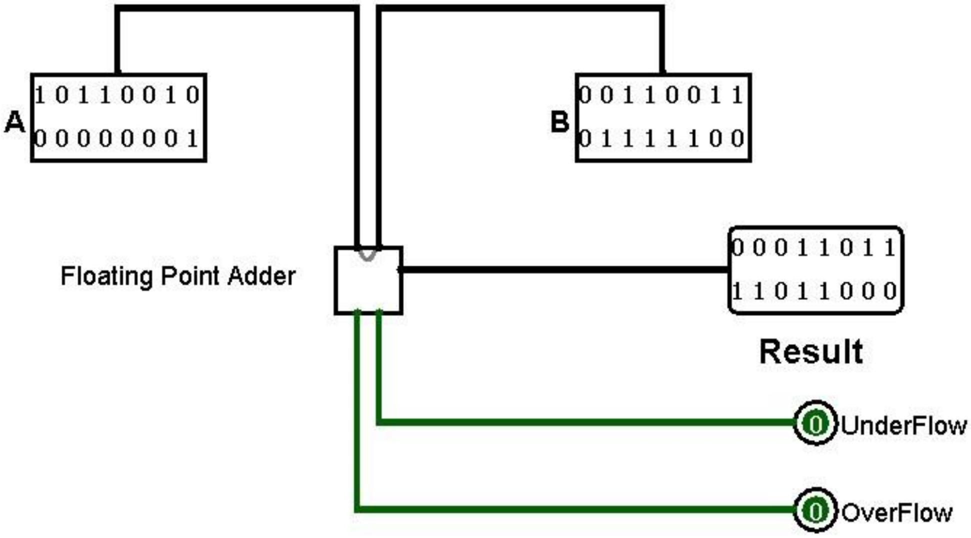
Sign 1 bit	Exponent 4 bit	Fraction 11 bit

The Floating Point Adder performs the summation operation of these two signed floating point numbers and produces a signed floating point number as the output in the same format as the inputs. The overflow flag and the underflow flag also need to be shown.

Flowchart of the Algorithm:


Start
Input A & B
Compute the sign of the result by comparing A & B
Determine the smaller number by comparing the exponent
Shift smaller number to the right until its exponent would match
Generate 2's complement for negative numbers and add the significands
Generate 2's complement if the sum is negative
Normalize the sum either shifting right and incrementing the exponent or shifting left and decrementing the exponent
OverFlow or underFlow?
Yes
OverFlow ?
Yes
NO
NO
Set Underflow bit
Set Overflow bit
Truncate the significand to the appropriate number of bits
Integrate the sign bit, exponent bits, fraction bits to show the result
Stop

Block Diagram:



Detailed Circuit Diagram of important Blocks:

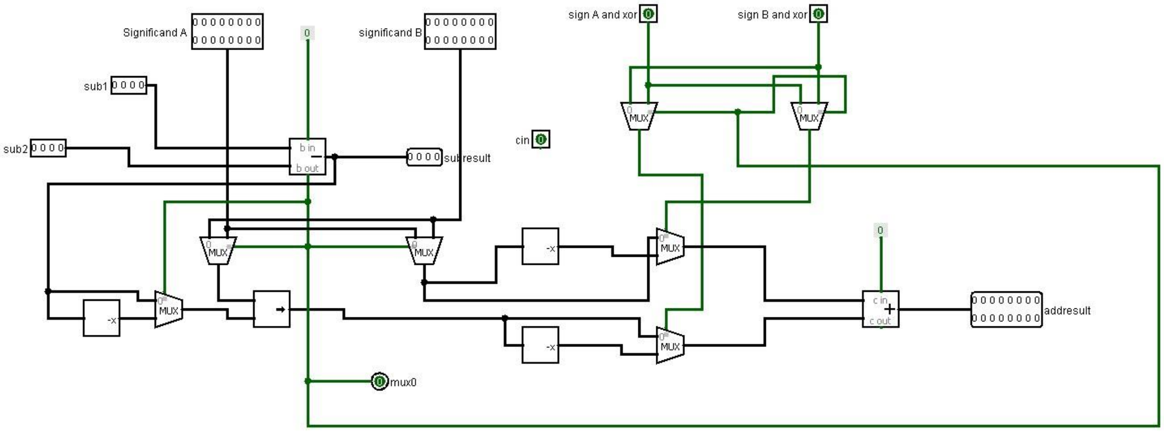


Figure 1: Adding Operation

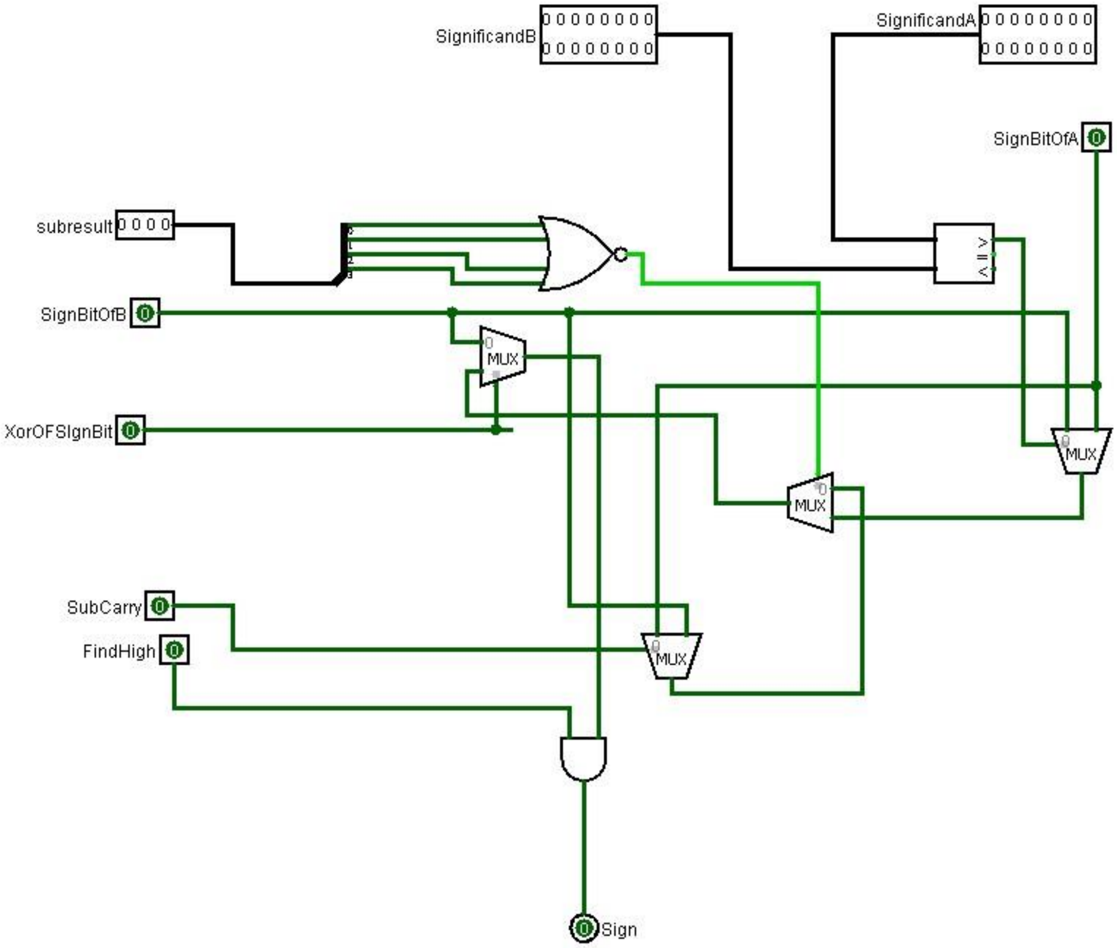


Figure 2: sign determination

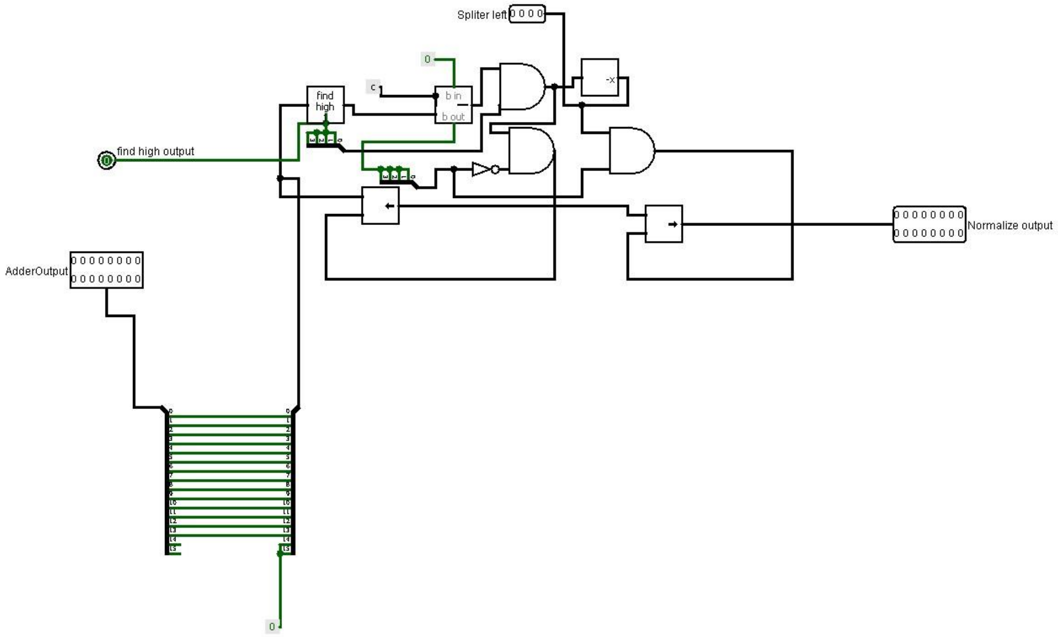


Figure 3: Normalization

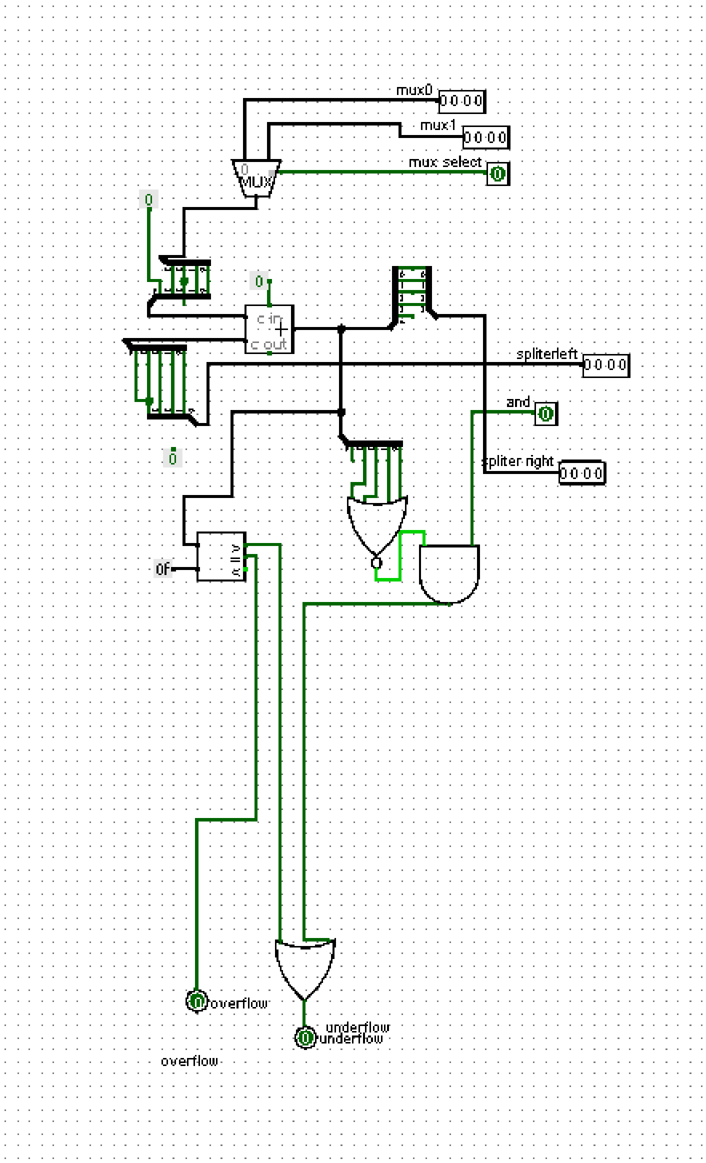


Figure 4: Overflow and Underflow

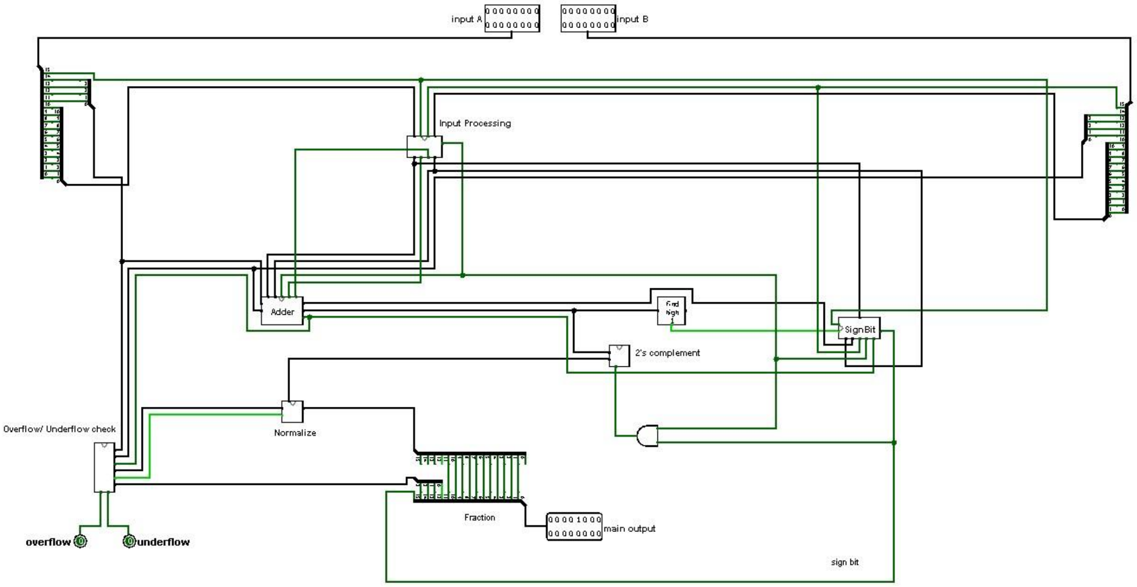


Figure 5: Full Circuit

IC Count:

Used IC	Operation	Count
IC 74LS04	NOT	1
IC 74LS08	AND	2
IC 74LS32	OR	1
IC 74LS86	XOR	1
IC 74LS25	Dual Four-Input NOR	1
NA	4 bit Full Subtractor	2
IC 74LS83	4 bit Full Adder	6
NA	2:1 Multiplexer	13
NA	16 bit Right Shift Register	2
NA	16 bit Left Shift Register	1
NA	4 bits Negator	2
NA	16 bit Negator	3
NA	16 bit Unsigned Comparator	1
NA	5 bit Unsigned Comparator	1
NA	16 bit Bit Finder	2

Total Chips Needed: 39

Simulator Used: Logisim

Version Number: 2.7.1

Discussion:

In this assignment, a floating point adder circuit was designed using Logisim. The circuit takes two floating point inputs and produces their sum as the output. The floating points used here are 16 bits long with 1 sign bit,4 exponent bits and 11 fraction bits. The larger of the two input exponents was chosen. The significands of the inputs were added using a 16 bit adder after necessary shifting. In case the generated sum was not normalized, it was normalized with either left shift or right shift. The number of shifted bits was added with or subtracted from the chosen exponent in order to get the exponent of the output. Here, the calculation of the output sign was done separately to avoid errors due to overflow. Also, two different flags indicating overflow and underflow were kept to check if the output exponent was within the allowable range(1-14). 0 in exponent was considered underflow and 15 was considered overflow. Here, when the exponent was found to be 0, it was also checked if the entire fraction part was 0. 0 in the exponent with a fraction part containing only zeros indicated that the output of the circuit was 0.On the other hand,0 in the exponent with non-zero fraction part indicated underflow.

Using a minimal number of ICs in order to reduce complexity was the priority while designing the circuit. Outputs of some of the intermediate gates and ICs were reused to minimize the circuit even more.

The connections were made carefully and the components were placed at fair distances. Messiness was avoided as much as possible to ensure higher readability. The circuit was divided into different modules to lessen congestion. Suitable labels were used at different parts of the circuit to make the functions of individual parts more understandable. Finally, the circuit was tested several times to make sure it did not have any sort of error.