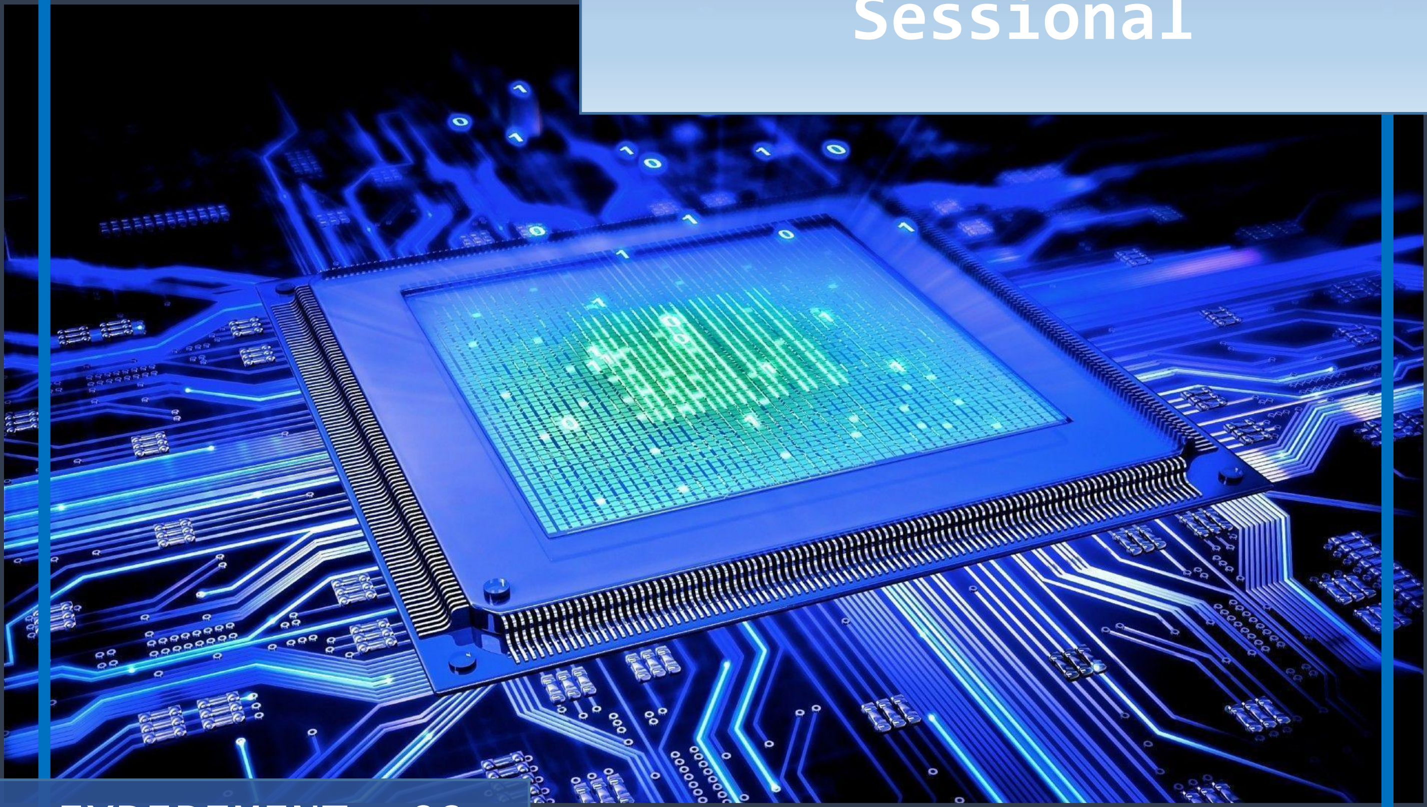
CSE 306

Computer Architecture Sessional



EXPERIMENT: 02

NAME OF EXPERIMENT:

Floating Point Adder

GROUP NO:	**04**
Section:	A2
Department:	CSE
Group Members:	1705041 1705047 1705049 1705050 1705052
Date of Submission:	30-05-2021

Introduction:

A Floating Point Adder is a digital circuit to do floating point addition or subtraction.

The goal of this experiment is to understand the basics of building complex circuit from simple logical gates and build a floating point adder from scratch, using these simple logic gates and other components. The Adder will take two 16bit numbers as input. The input is in IEEE 754 Standard format of binary representation of the floating point number (sign bit, exponent bits, fraction bits). For 16 bit input number, first bit represents sign of the number, next 4 bits are for exponent followed by 11 fraction bits.

The circuit will do addition or subtraction operation and output the result along with two status flags (overflow flag and underflow flag).

Problem Specification:

A Floating Point Adder is to be designed that takes two 16 bit fraction number of IEEE 754 Standard and output their sum along with two status bits (overflow and underflow flags).

Truth Table of ALU:

Selection				Output	Function
S2	S1	S0	Cin		
0	0	0	0	F = A	Transfer A
0	0	0	1	F = A + 1	Increment A
0	0	1	0	F = A + B	Addition
0	0	1	1	F = A + B + 1	Add with carry
0	1	0	0	F = A - B - 1	Subtract with borrow
0	1	0	1	F = A - B	Subtraction
0	1	1	0	F = A - 1	Decrement A
0	1	1	1	F = A	Transfer A
1	0	0	X	F = A ∨ B	OR
1	0	1	X	F = A ⊕ B	XOR
1	1	0	X	F = A ∧ B	AND
1	1	1	X	F = A'	Complement A

Flow Chart:

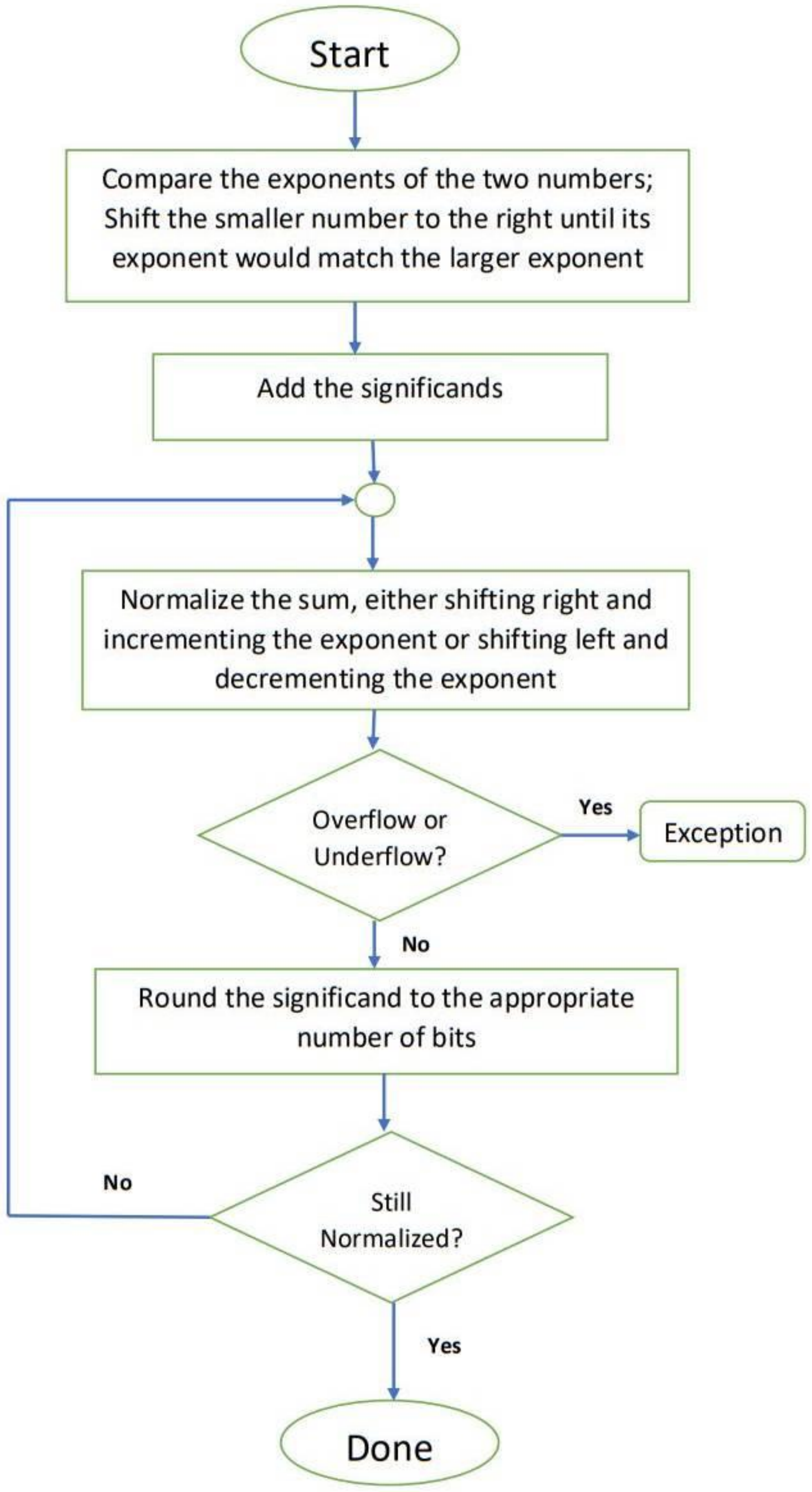

Start
Compare the exponents of the two numbers; Shift the smaller number to the right until its exponent would match the larger exponent
Add the significands
Normalize the sum, either shifting right and incrementing the exponent or shifting left and decrementing the exponent
Overflow or Underflow?
Yes
Exception
No
Round the significand to the appropriate number of bits
No
Still Normalized?
Yes
Done

Fig-1: Flow chart of floating point adder

High Level Block Diagram:

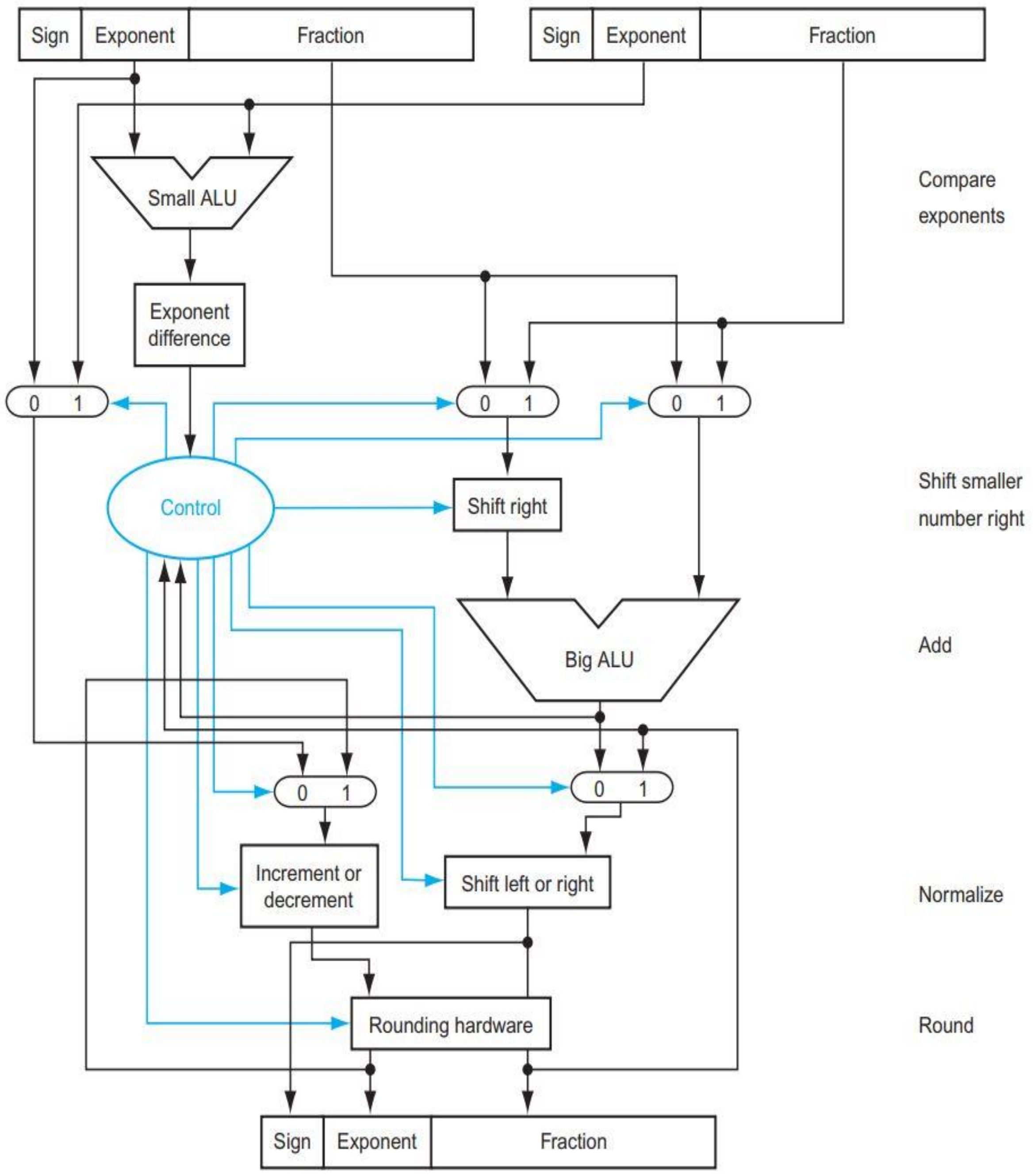


Fig-2: High level block diagram of floating point adder

Detailed Circuit Diagram:

1 bit ALU:

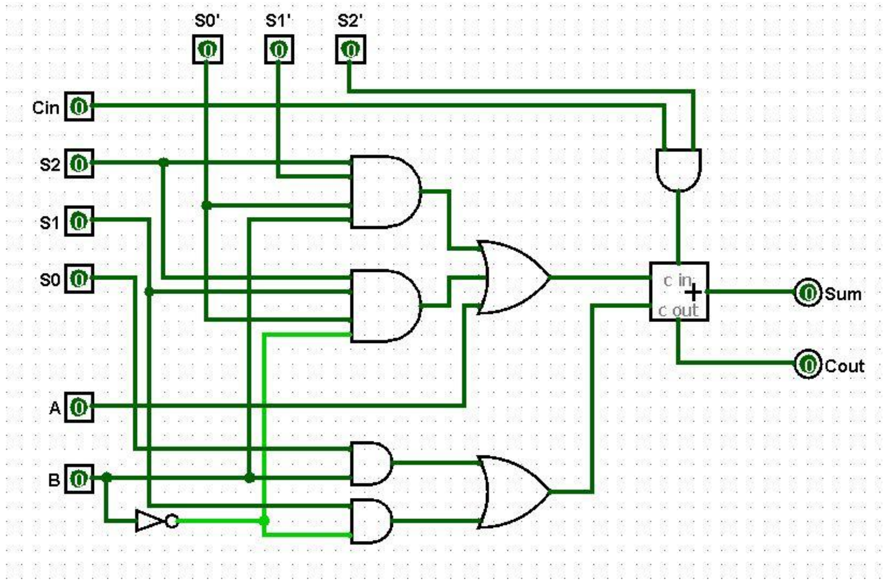


Fig-3.1: 1 bit ALU circuit

4 bit ALU:

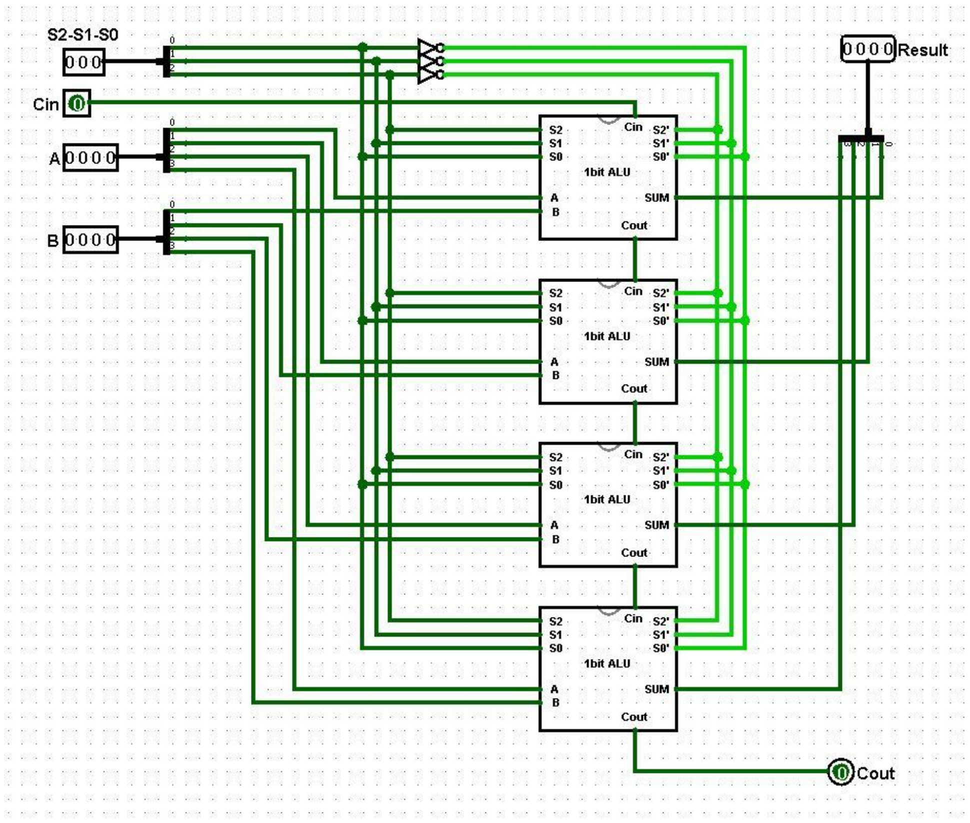


Fig-3.2: 4 bit ALU circuit

<u>8 bit ALU:</u>

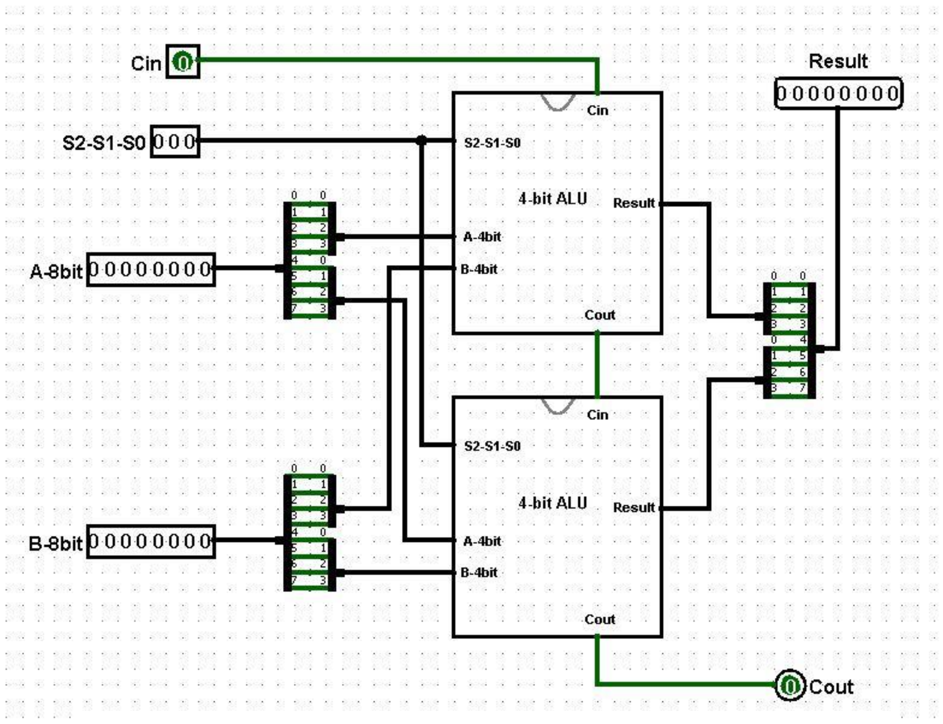


Fig-3.3: 8 bit ALU circuit

<u>16 bit ALU:</u>

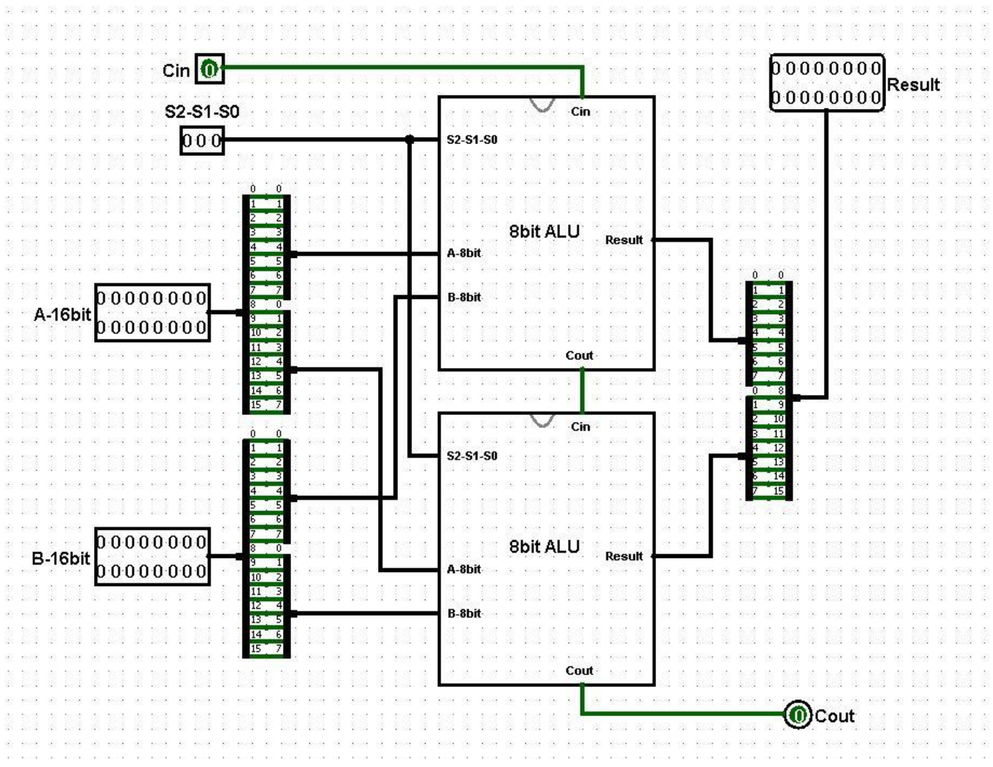


Fig-3.4: 16 bit ALU circuit

16 bit Separator:

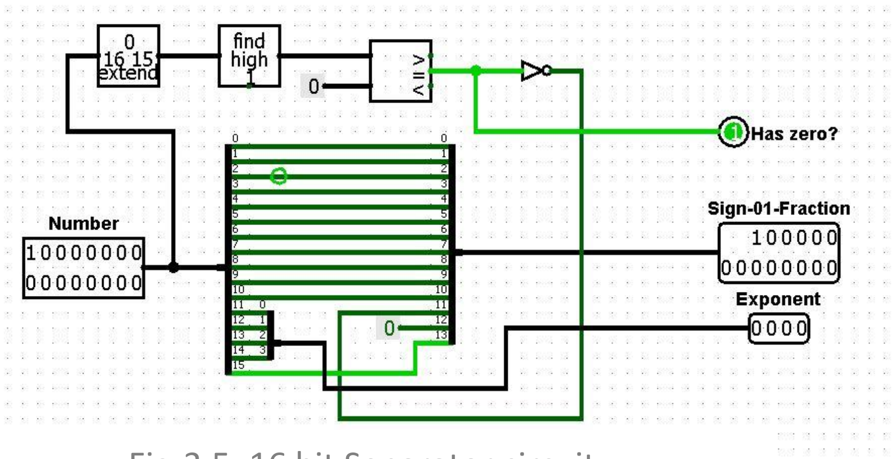


Fig-3.5: 16 bit Separator circuit

14 bit Right Shifter:

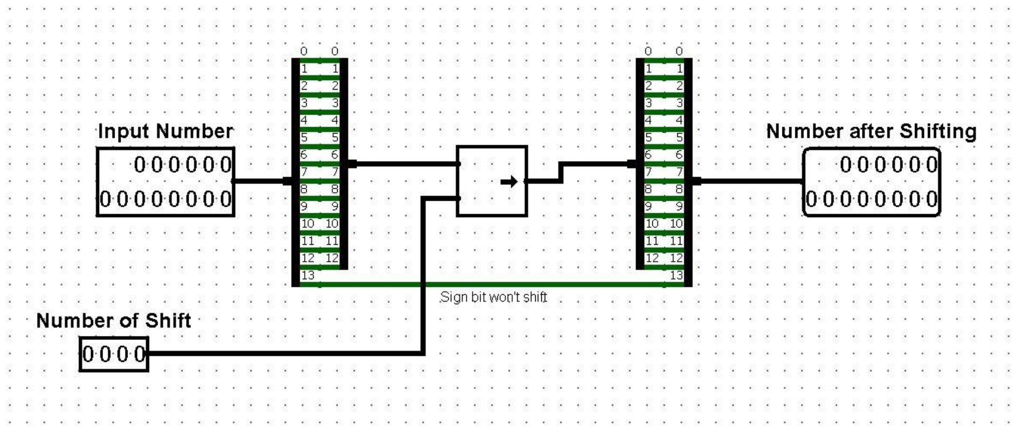


Fig-3.6: 14 bit Right shifter circuit

14 bit Addend Creator:

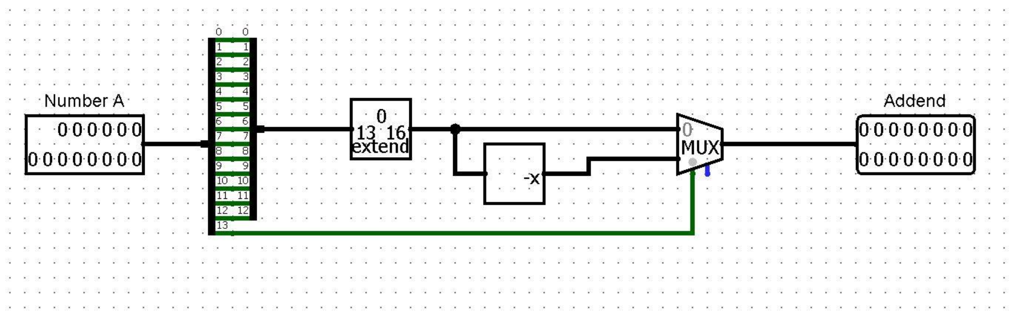


Fig-3.7: 14 bit Addend Creator circuit

16 bit Operand Selector:

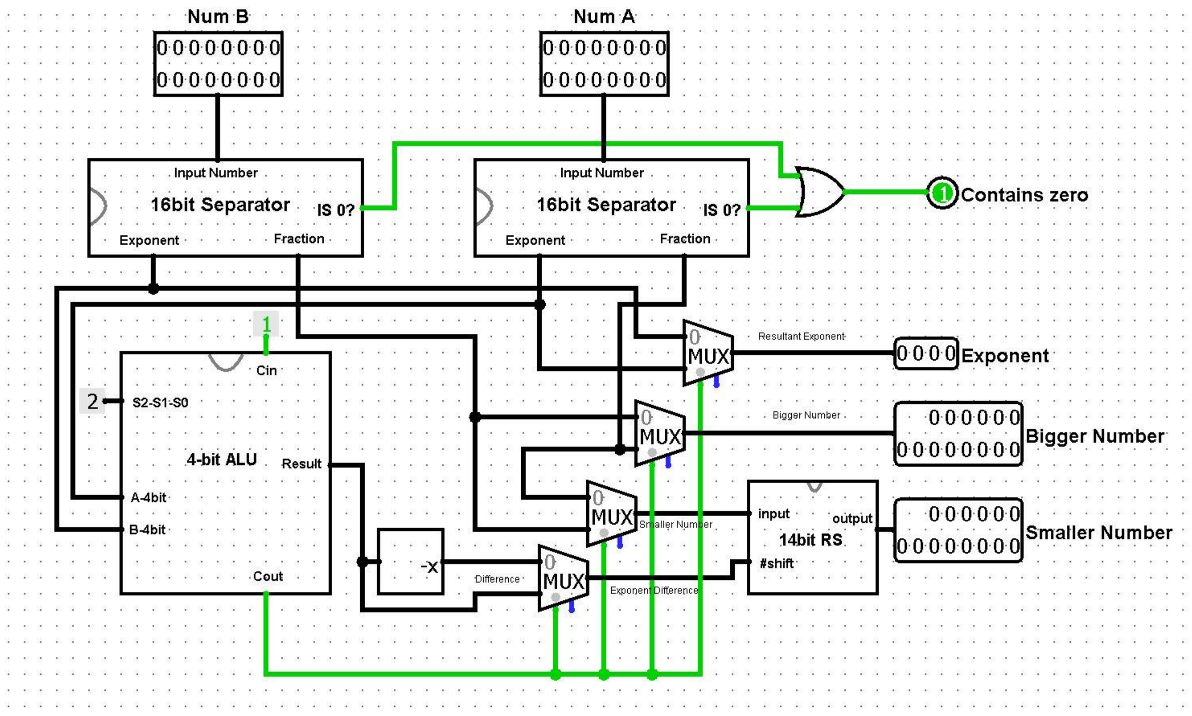


Fig-3.8: 16 bit Operand Selector circuit

16 bit - 14 bit Result Formatter:

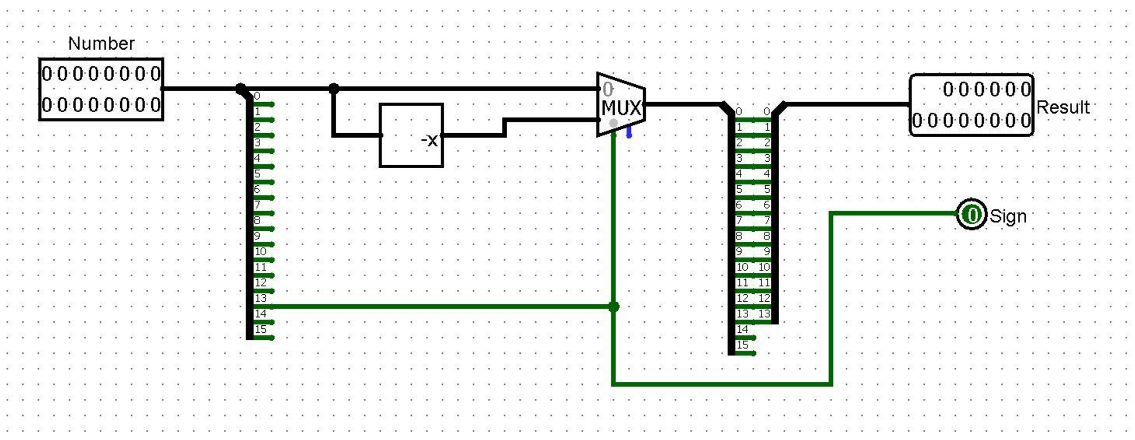


Fig-3.9: 16 bit - 14 bit Result Formatter circuit

Significand Adder:

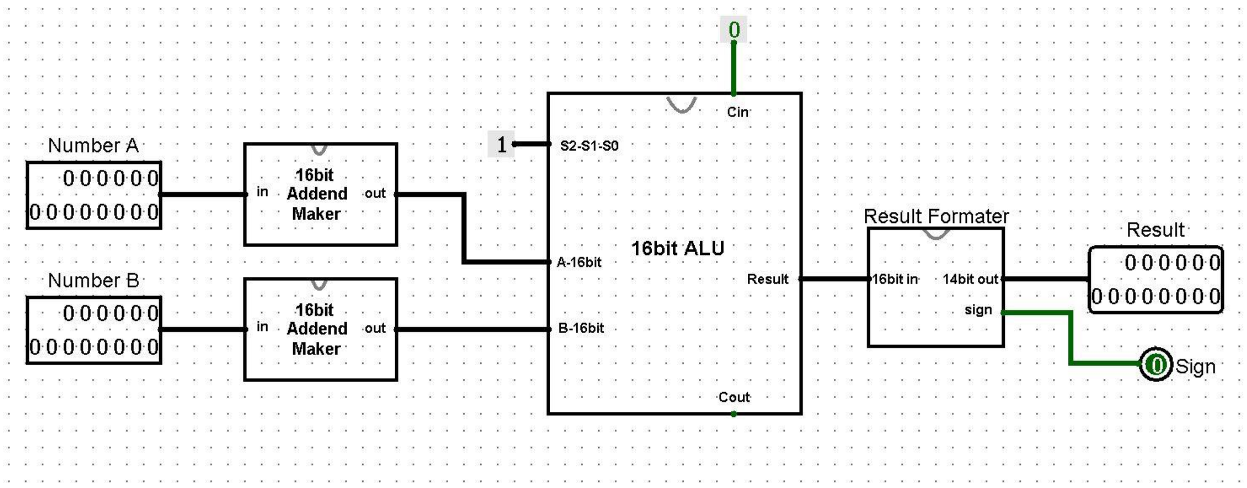


Fig-3.10: Significand Adder circuit

Normalizer Left Shift:

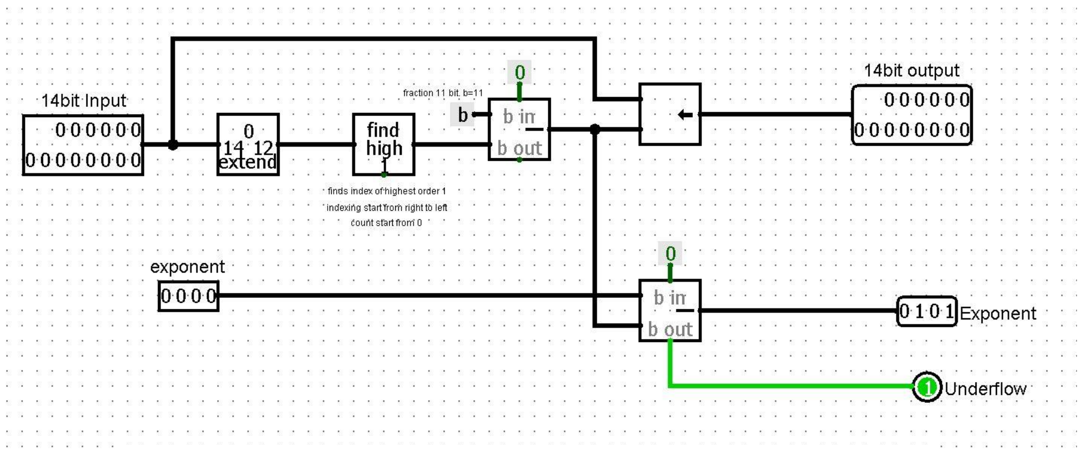


Fig-3.11: Normalizer Left Shift circuit

Normalizer Right Shift:

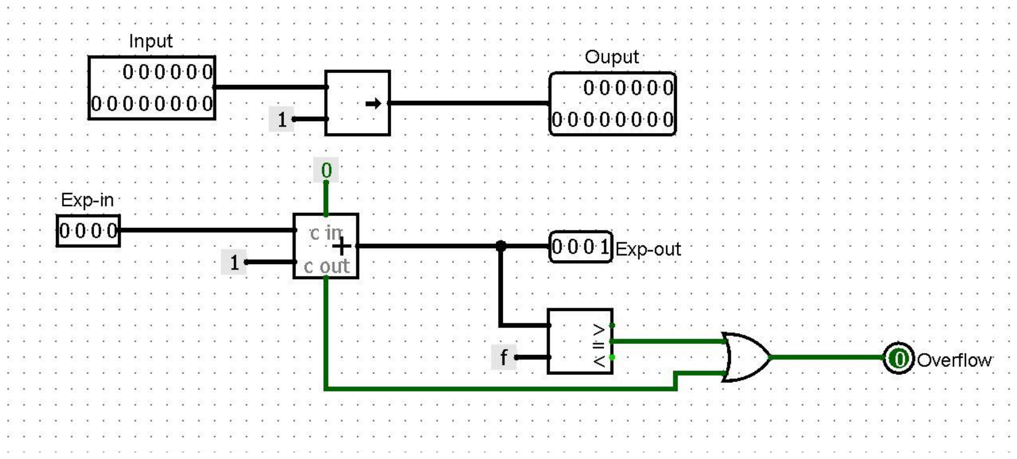


Fig-3.12: Normalizer Right Shift circuit

Normalizer:

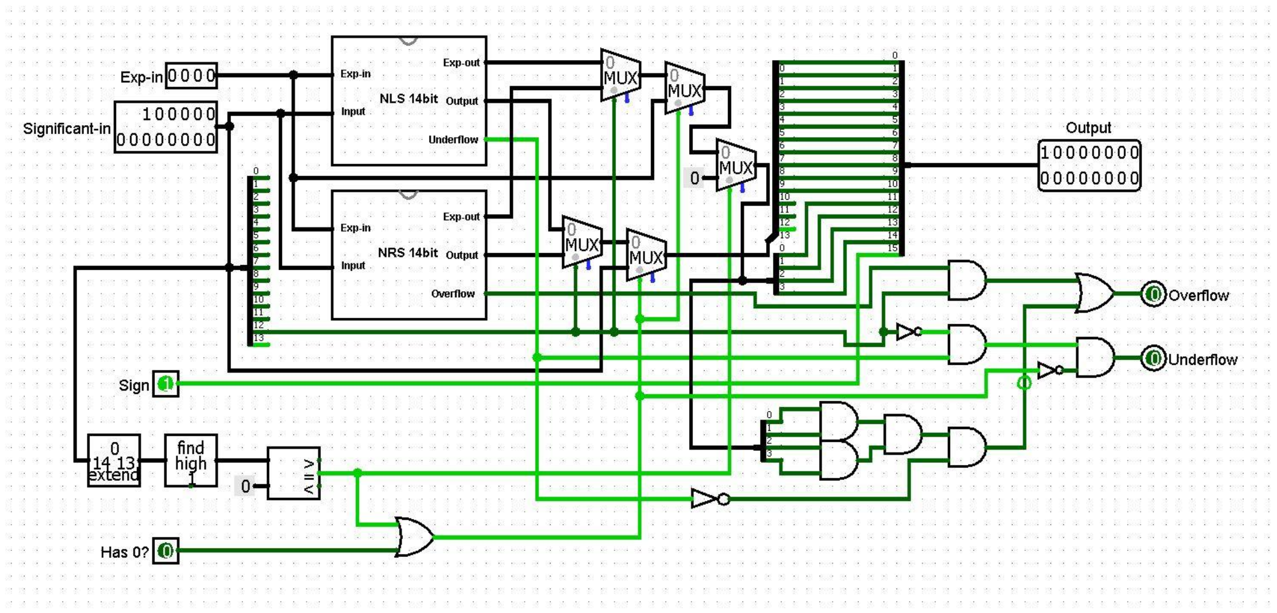


Fig-3.13: Normalizer circuit

16 bit Floating Point Adder:

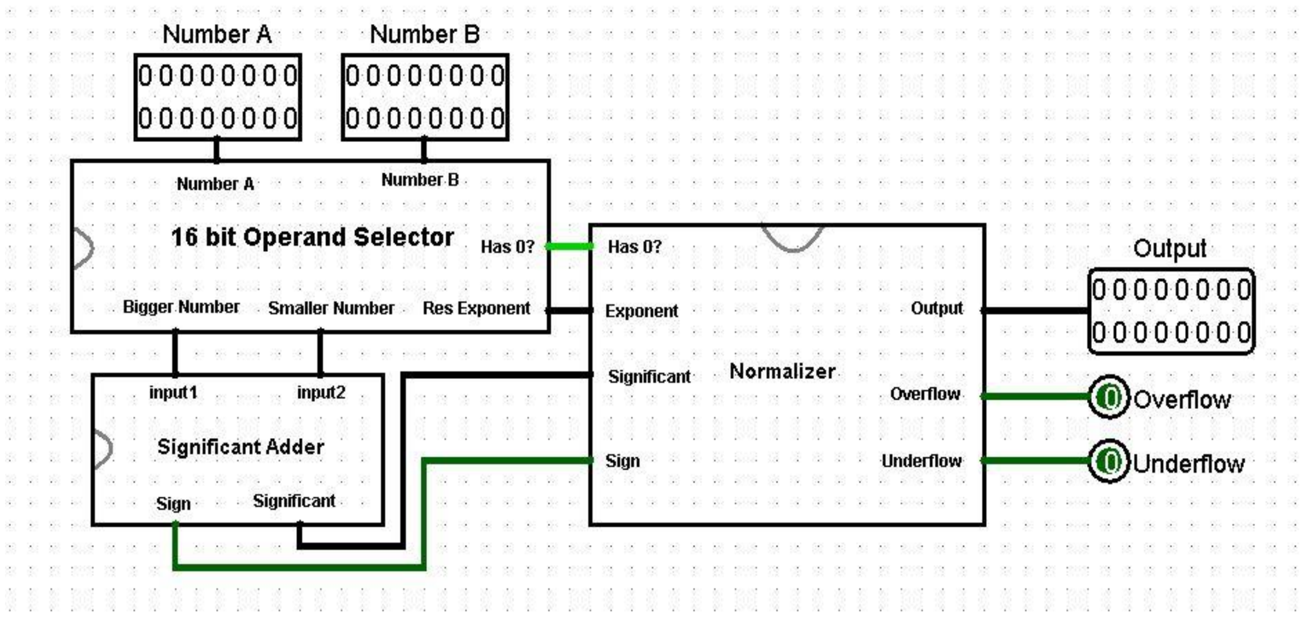


Fig-3.14: 16 bit Floating Point Adder circuit

Simulator Circuit:

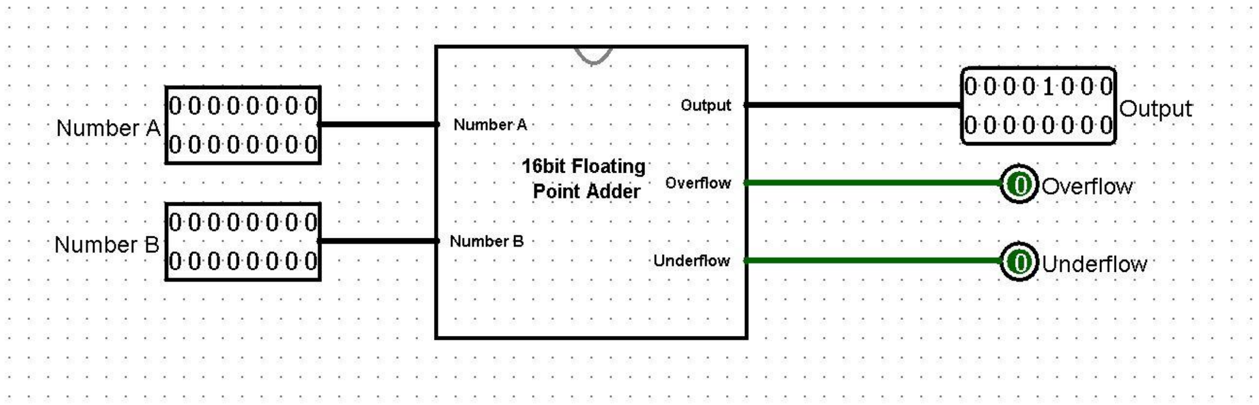


Fig-3.15: Floating Point Adder Simulator circuit

IC Count:

Used IC	Operation	Gates Used	Gates Per IC	IC Count
IC 74LS08	AND	107	4	27
IC 74LS32	OR	44	4	11
IC 74LS04	NOT	39	6	7
IC 74157	MUX (2:1)	104	4	26
IC 7480	1-bit Full Adder	21	1	21
IC 7485	4-bit Comparator	3	1	3
(Logisim)	Shifter	3		
(Logisim)	Bit Extender	4		
(Logisim)	Negator	3		
(Logisim)	Bit Finder	3		
(Logisim)	Subtractor	2		

Total IC used: 89

Simulator Used: Logisim

Version Number: 2.7.1

Discussion:

The importance and usefulness of floating point format nowadays does not allow any discussion. Any computer or electronic device which operates with real numbers implements this type of representation and operation. Floating point addition is the most frequent floating point operation and accounts for almost half of the scientific operation. Therefore, it is a fundamental component of math coprocessor, DSP processors, embedded arithmetic processors, and data processing units. These components demand high numerical stability and accuracy and hence are floating point based. Floating-point addition is a costly operation in terms of hardware and timing as it needs different types of building blocks with variable latency.

To construct the FPA (Floating Point Adder), we have implemented the various building blocks and connect them together to form the FPA. We follow the systematic way to approach this problem by building and testing each of the bottom level modules first and then proceed to working on larger and larger pieces. This requires us to verify the functionality of each module. We have tested special boundary cases like the number zero, saturation, etc. These tests allow us to pinpoint the errors at their source, instead of trying to find them after they've propagated through a large and complicated system. We ensure that input is in IEEE 754 Standard format of binary representation of the floating point number (sign bit, exponent bit and fraction bits). The shifter, negator, bit extender and bit finder circuits we used are provided by the simulator software tool. The IEEE 754 defines five types of exceptions: overflow, underflow, invalid operation, inexact result, and division-by-zero. These exceptions are signaled by setting a flag for each. In evaluating our implementation, we have implemented overflow and underflow flags in our designs as they are the most frequent exceptions that occur during addition.

For implementing and simulating these functional designs, we used **Logisim**. It is a useful software to design circuit and block diagram. During simulation it is necessary for the user to ensure simulation enabled is checked.

During designing the circuit we focus on simple and easier solution rather than using minimum number of IC's for better understanding. Because we need to make our circuit more understandable as our goal is to understand the operations of floating point numbers only.

From this assignment, the details of Floating Point Adder implementation can be understood precisely and thoroughly.