CSE 306: Computer Architecture Sessional

Assignment 2: Floating Point Adder

Submitted By:

Group No: 05

Sub-Section: A2

Group Members:

1. Kawshik Kumar Paul (ID: 1705043)
2. Iftekhar Hakim Kaowsar (ID: 1705045)
3. Rasman Mubtasim Swargo (ID: 1705051)
4. Apurba Saha (ID: 1705056)
5. Maisha Rahman (ID: 1705060)

Department: CSE

Level: 3 **Term:** 1

Date of Submission: 25 May, 2021

Problem Specification :

In this assignment, we have to design a floating point adder circuit which takes two floating point numbers as inputs and output their sum which is another floating point number.

Introduction:

A Floating Point Adder is a digital circuit used to add two floating point numbers presented in IEEE 754 Standard format of their binary representation.

Floating Point Representation : A designer of a floating-point representation must find a compromise between the size of the fraction and the size of the exponent. Floating-point numbers are usually a multiple of the size of a word.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Sign	Exponent				Fraction										
1 bit	8bits				23 bits										

In general, floating-point numbers are of the form $(-1)^s \times F \times 2^E$

Here, F involves the value in the fraction field and E involves the value in the exponent field. MIPS does something slightly more sophisticated.

According to IEEE 754 floating-point standard, the floating point numbers are of the form

$$(1)^S \times (1 + \text{Fraction}) \times 2^E$$

IEEE 754 uses a bias of 127 for single precision, so an exponent of -1 is represented by the bit pattern of the value $-1+127_{ten}$, or $126_{ten} = 0111\ 1110_{two}$, and +1 is represented by $1 + 127$, or $128_{ten} = 1000\ 0000_{two}$. The exponent bias for double precision is 1023. Biased exponent means that the value represented by a floating-point number is really

$$(-1)^s \times (1+\text{Fraction}) \times 2^{(\text{Exponent-Bias})}$$

Floating Point Addition Algorithm:

Given input A(S_1, E_1, F_1) and B(S_2, E_2, F_2). Now, F_1 and F_2 are of 11 bits.

1. Prepend “01” to both F_1 and F_2.
2. Align F_1, F_2 according to maximum of E_1, E_2.
3. Now, we need to set the sign bit S_1 and S_2 to F_1 and F_2 respectively. If the sign bit is 1, we need to take the 2’s complement of magnitude. Now, F_1 and F_2 are of 14 bits.
4. We find the sum of F_1 and F_2. We call it ‘sum’. We can hold the sum in 14 bits because in step 1 we prepended a zero bit.
5. We extract the sign of final output from the 13th bit (0-indexed) of sum.
6. Last 13 bits comprise the unnormalized fraction of output. We note that if the sign bit of final output is 1, we need to take the 2’s complement of fraction beforehand.
7. Normalize the fraction by finding the most significant set-bit and change the exponent accordingly. Now, the fraction is in the normalized scientific form of 13 bits.
   Here 12th-bit (0-indexed) must be 1. And, the 11th-1st bit comprises the output_fraction. 0th bit has been truncated.
8. After normalizing, if the exponent cannot be expressed by 4 bits or if it becomes less than zero, we turn on the overflow/underflow flag.

Flowchart :

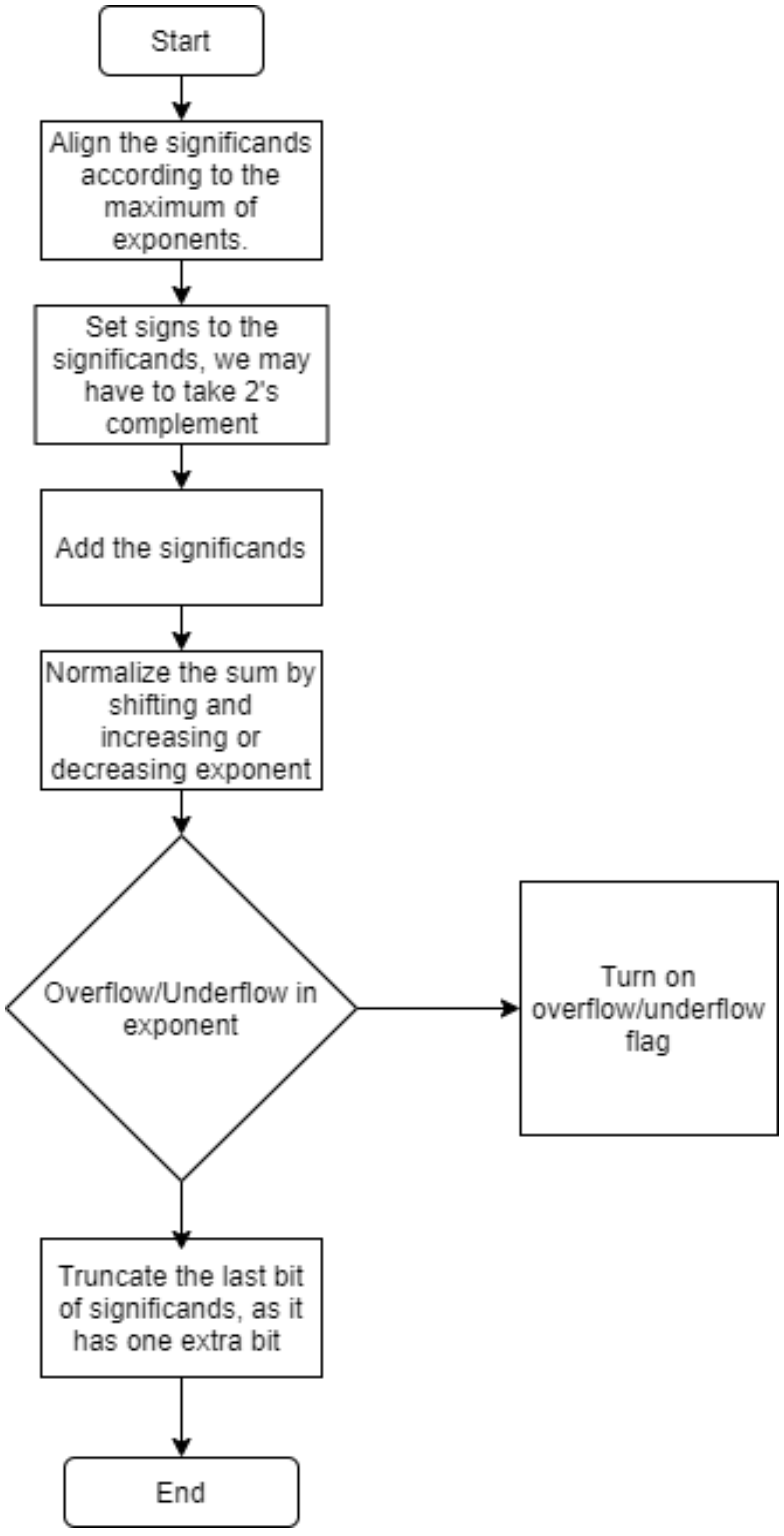

Figure 1 : FlowChart of Floating Point Adder

IC used with count:

IC	Count
74x897 (16 Bit Shifter)	3
Most Significant Set Bit Finder	1
7480 (FULL ADDER)	18
7404 (NOT)	6
7432 (OR)	1
74HC157 (2X1 MUX)	9

Block Diagram:

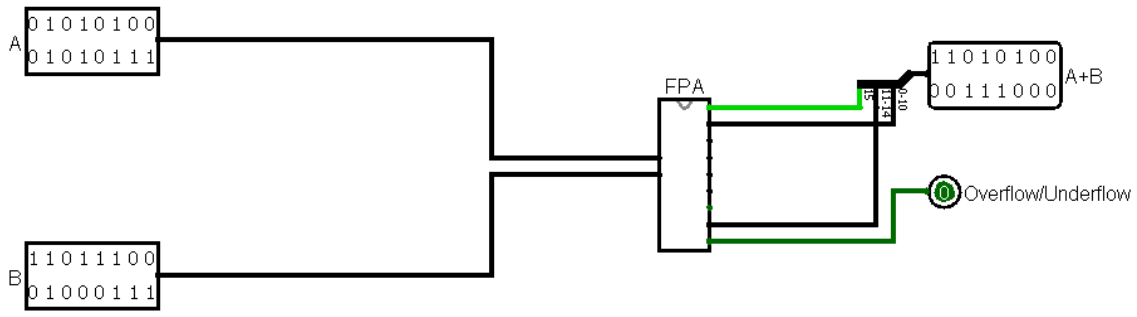

Figure 1: Block Diagram

Circuit Diagram:

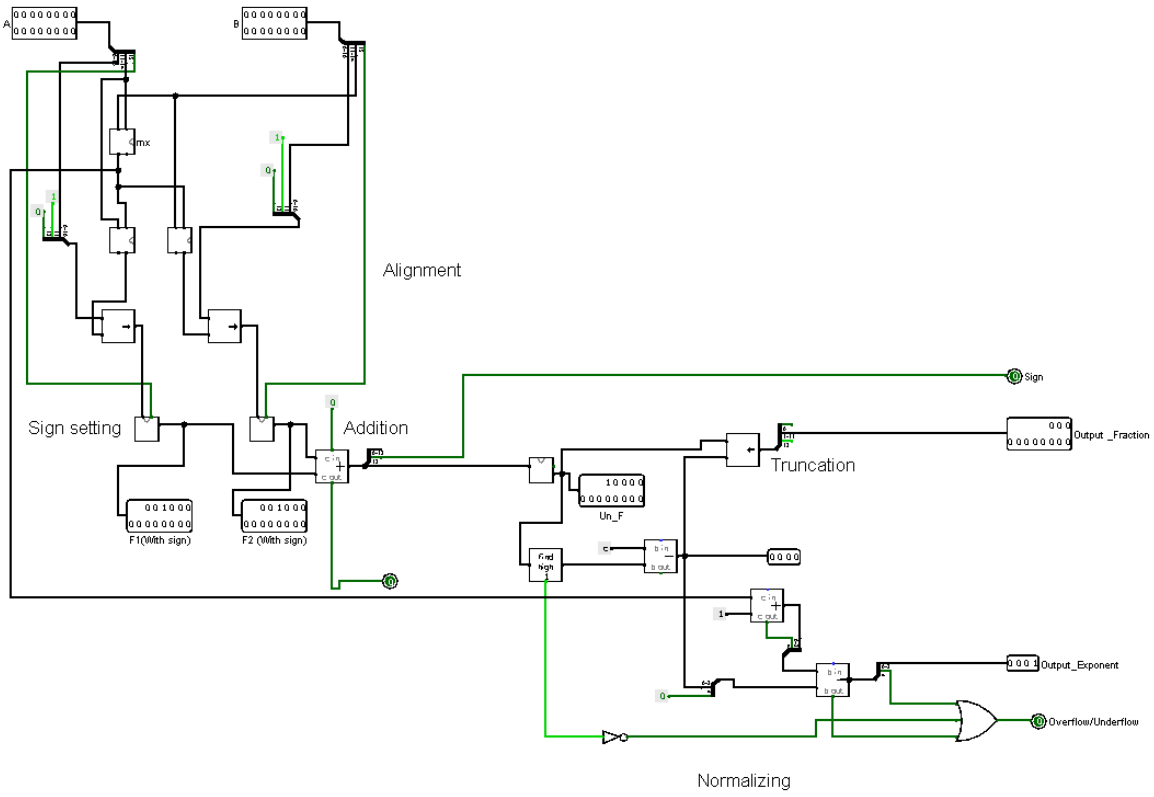

Figure 2: Circuit Diagram 1

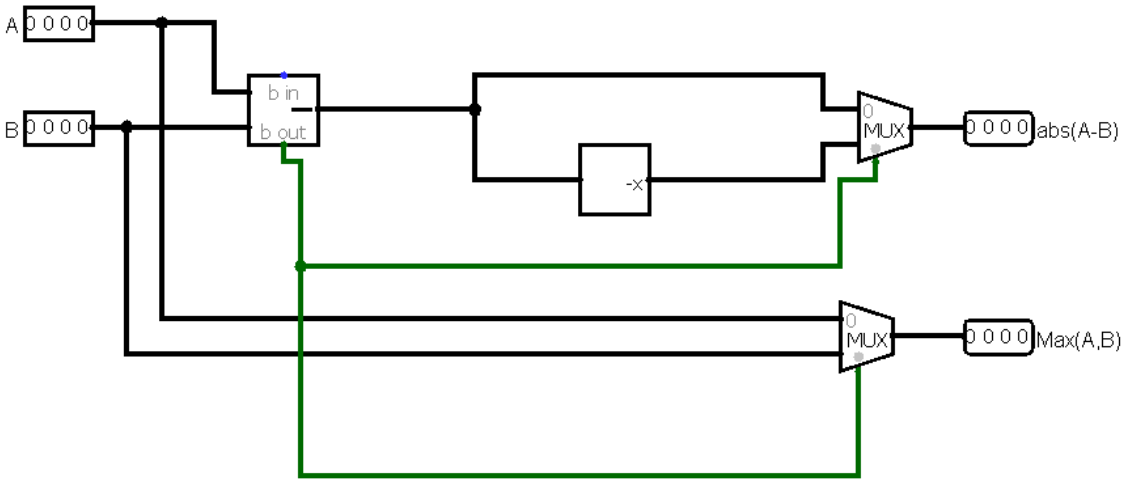

Figure 3 : Circuit Diagram 2

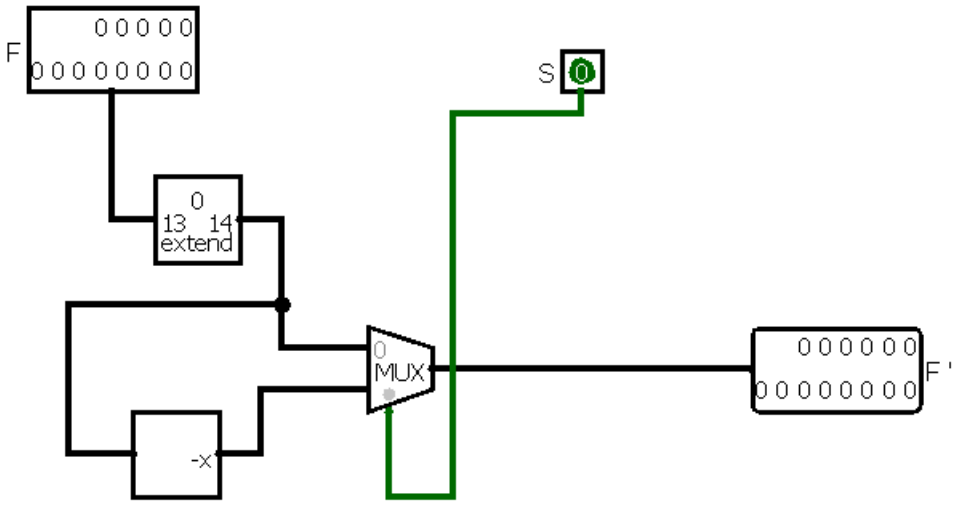

Figure 4 : Circuit Diagram 3

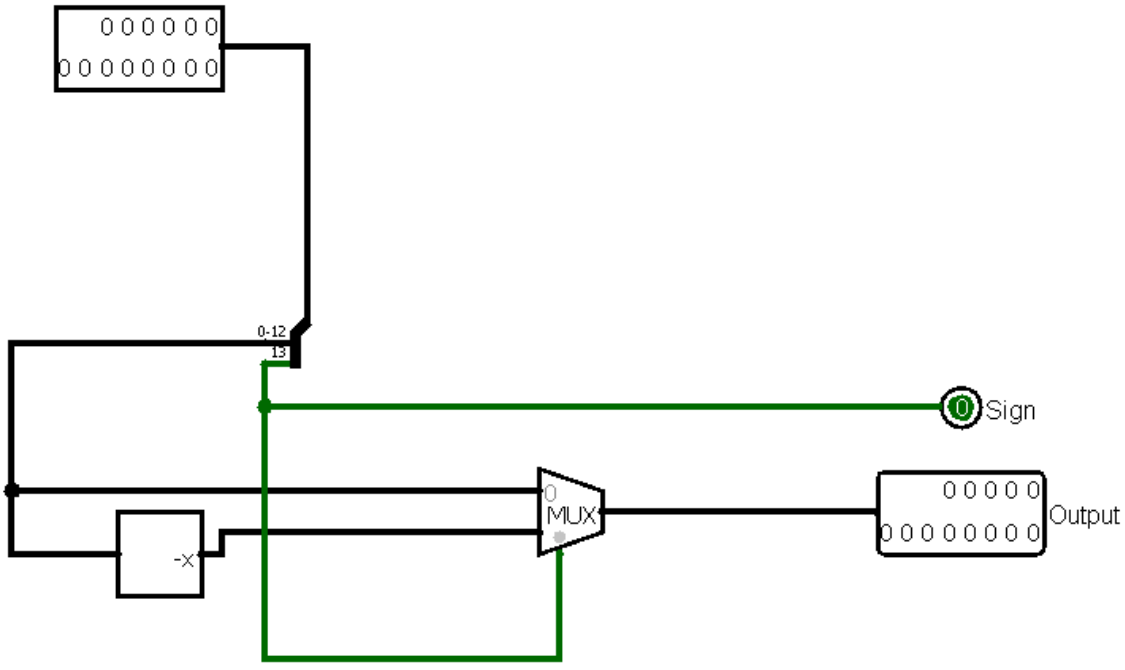

Figure 5 : Circuit Diagram 4

Simulator used along with the version number:

Logisim 2.7.1

Discussion:

In this assignment, we designed a Floating Point Adder. For this purpose, we used basic gates (AND, OR, NOT, NOR), universal gates(XOR, XNOR) and some other necessary gates(MUX, Shifter, Bit Finder, Adder, Subtractor, Bit Extender, Negator). The Floating Point Adder takes two 16 bit binary numbers A and B which are basically binary representations of a floating-point number. Our implemented circuit outputs the added value of A and B. We used the known technique of designing the floating-point adder. We followed the provided flowchart, built the circuit diagram, and implemented the circuit. While designing, we put emphasis on simplifying the circuit. We asserted our design by testing various inputs and matching the corresponding outputs. On the simulator, alongside the full circuit, we made a block circuit so that all signals can be visualized easily. In our implementation of fp-adder, we used some built-in circuits available in Logisim like bit-shifter, bit-finder, bit-extender, etc. Unfortunately, we failed to figure out the IC number of bit finder.

However, testing the outputs and completing the data table, we successfully finished our simulation.