Aditya Sangawar

MS Electrical Engineering

ID – XXXXX9637

1. Introduction

The goal of the project is to build working of a 5-stage pipelined IEEE single-precision (SP) floating-point (FP) adder.

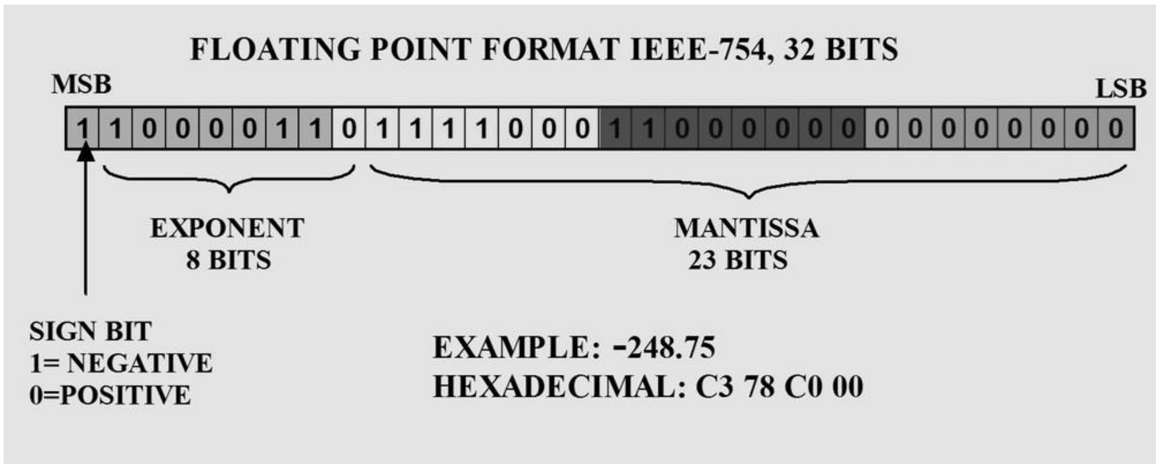
Key Learnings – There are 2 key learnings from the project –

a. It gives us the insight on how to build an adder that does arithmetic operation on the numbers in IEEE 754 format.
b. It helps us to learn and gain insight on pipeline architecture and how to implement them using Verilog HDL.

2. Description of the design

Design Summary-

Floating point format IEEE-754 is 32-bit format of representing floating point numbers. The bits representation is shown below -



Brief Description -

The IEEE Standard for Floating-Point Arithmetic (IEEE 754) is a technical standard designed for floating-point computation and storing.
It was established in 1985 by the Institute of Electrical and Electronics Engineers (IEEE).

The standard defines:

- Arithmetic formats: Sets of binary and decimal floating-point data consisting of finite numbers (including signed zeros and subnormal numbers), infinities, and special "not a number" values (NaNs)
- Interchange formats: Encodings bit strings for communication of data.
- Rounding rules: properties to be satisfied when rounding numbers.
- Operations: Arithmetic and trigonometric functions etc.
- Exception handling: indications of exceptional conditions (such as division by zero, overflow, etc.)

Source: Wikipedia.
https://en.wikipedia.org/wiki/IEEE_754

Project Description –

The design of the Adder is into two parts –

Part One -
1. Designing an un-pipelined adder and testing them using the below mentioned test cases.
2. All the values are visible at the output as soon as they are sent to the input.
3. This type of architecture may require less hardware but required more time for instruction to be executed.

Code : PipelineAdderNonPipelined.v

<u>Part Two -</u>

1. To improve the speed and thus the performance, we updated the design, by using the method of pipelining.
2. The modified design is a 5-staged pipelined FP adder.
3. In this version, it will work for the six pairs of inputs and will take 12 cycles to output all 8 cases.

PipelineAdder.v

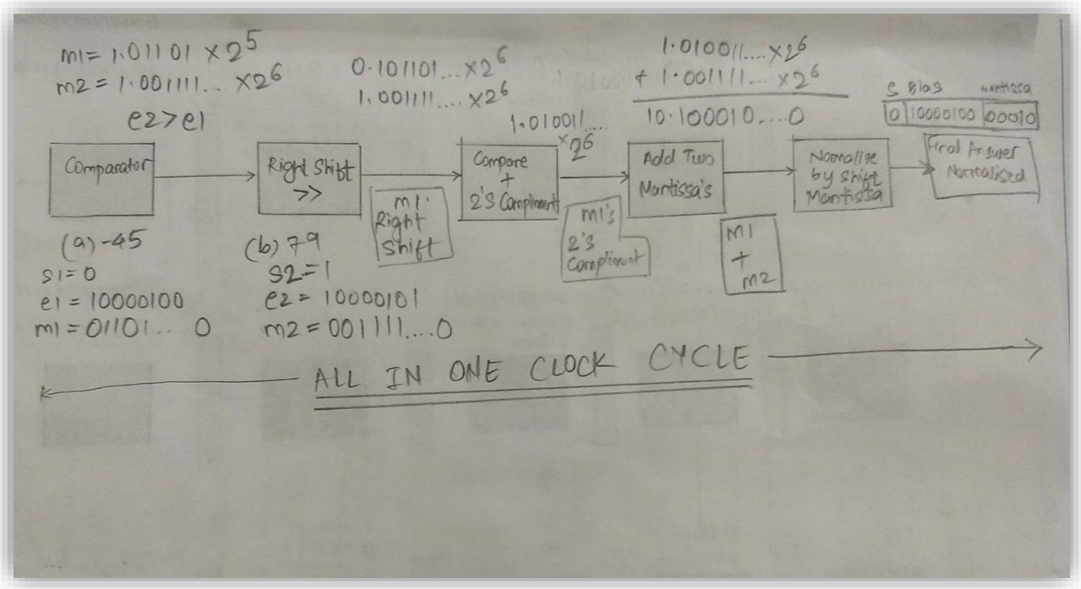
Code :

3. Simulation/Verification: (20 points including the section on "Theory of Operation" unless otherwise indicated)

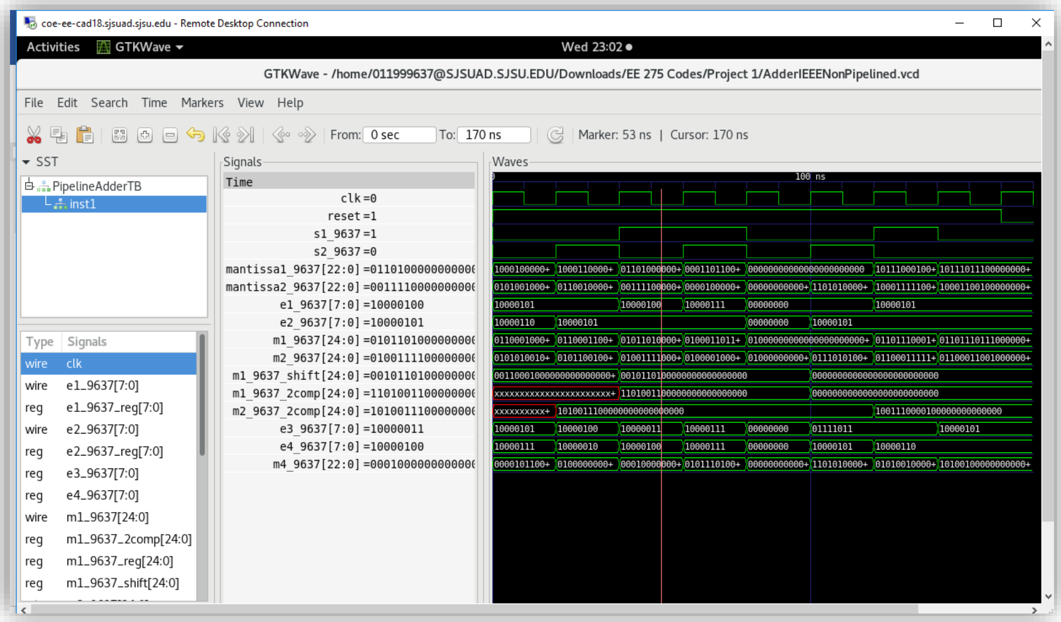
3.Theory of Operation –
Design 1
This design is a non-pipelined version that has the data output directly once the input is applied.

A : -45 & b : 79

Below is the diagram that shows the steps of the calculation in terms of each Block -
<u>Note</u> : The entire execution happened in One Clock Cycle.



Result Waveform – As seen, the final output


coe-ee-cad18.sjsuad.sjsu.edu - Remote Desktop Connection
Activities
GTKWave
Wed 23:02
File Edit Search Time Markers View Help
SST
PipelineAdderTB
inst1
Waves
100 ns

GTKWave - /home/011999637@SJSUAD.SJSU.EDU/Downloads/EE 275 Codes/Project 1/AdderIEEENonPipelined.vcd

From: 0 sec To: 170 ns Marker: 53 ns | Cursor: 170 ns

Type	Signals
wire	clk
wire	e1_9637[7:0]
reg	e1_9637_reg[7:0]
wire	e2_9637[7:0]
reg	e2_9637_reg[7:0]
reg	e3_9637[7:0]
reg	e4_9637[7:0]
wire	m1_9637[24:0]
reg	m1_9637_2comp[24:0]
reg	m1_9637_reg[24:0]
reg	m1_9637_shift[24:0]

```
Time
                    clk =0
                  reset =1
                s1_9637 =1
                s2_9637 =0
mantissa1_9637[22:0] =0110100000000000
mantissa2_9637[22:0] =0011110000000000
        e1_9637[7:0] =10000100
        e2_9637[7:0] =10000101
       m1_9637[24:0] =0101101000000000
       m2_9637[24:0] =0100111100000000
 m1_9637_shift[24:0] =0010110100000000
 m1_9637_2comp[24:0] =1101001100000000
 m2_9637_2comp[24:0] =1010011100000000
        e3_9637[7:0] =10000011
        e4_9637[7:0] =10000100
       m4_9637[22:0] =0001000000000000
```

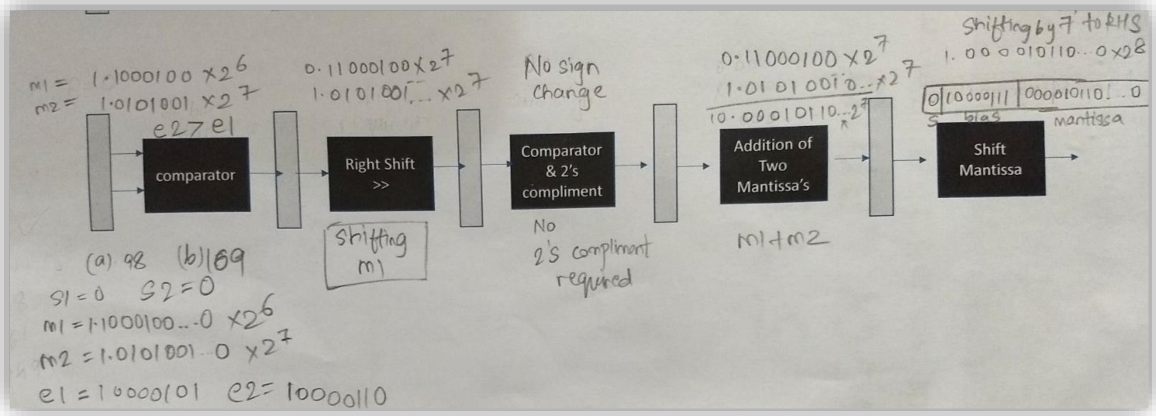
```
mantissa1: 1000100000+ | 1000110000+ | 01101000000+ | 0001101100+ | 00000000000000000000000 | 10111000100+ | 1011101110000000+
mantissa2: 0101001000+ | 0110010000+ | 00111100000+ | 0000100000+ | 00000000000+ | 1101010000+ | 10001111100+ | 1000110010000000+
e1:        10000101 | 10000100 | 10000111 | 00000000 | 10000101
e2:        10000110 | 10000101 | 00000000 | 10000101
m1:        0110001000+ | 0110001100+ | 01011010000+ | 0100011011+ | 0100000000000000000000000+ | 01101110001+ | 0110111011100000+
m2:        0101010010+ | 0101100100+ | 01001111000+ | 0100001000+ | 01000000000+ | 0111010100+ | 01100011111+ | 0110001100100000+
m1_shift:  0011000100000000000000000+ | 0010110100000000000000000 | 0000000000000000000000000
m1_2comp:  xxxxxxxxxxxxxxxxxxxxxxxxx+ | 1101001100000000000000000 | 0000000000000000000000000
m2_2comp:  xxxxxxxxxx+ | 1010011100000000000000000 | 1001110000100000000000000
e3:        10000101 | 10000100 | 10000011 | 10000111 | 00000000 | 01111011 | 10000101
e4:        10000111 | 10000010 | 10000100 | 10000111 | 00000000 | 10000101 | 10000110
m4:        0000101100+ | 0100000000+ | 00010000000+ | 0101110100+ | 00000000000+ | 1101010000+ | 01010010000+ | 1010010000000000+
```

Design 2

The design consists of a 5 stage pipelined operation with stage description given below –
Assume we have 2 numbers in IEEE 754 format –

A : 98 & B: 169

Below is the diagram that shows the steps of the calculation in terms of each Block -



<u>Stage 1</u>: Compares the exponents and determine the number of shifts required to align the mantissa to make the exponents equal. Here since e2>e1, the mantissa 'm1' will have to be realigned.

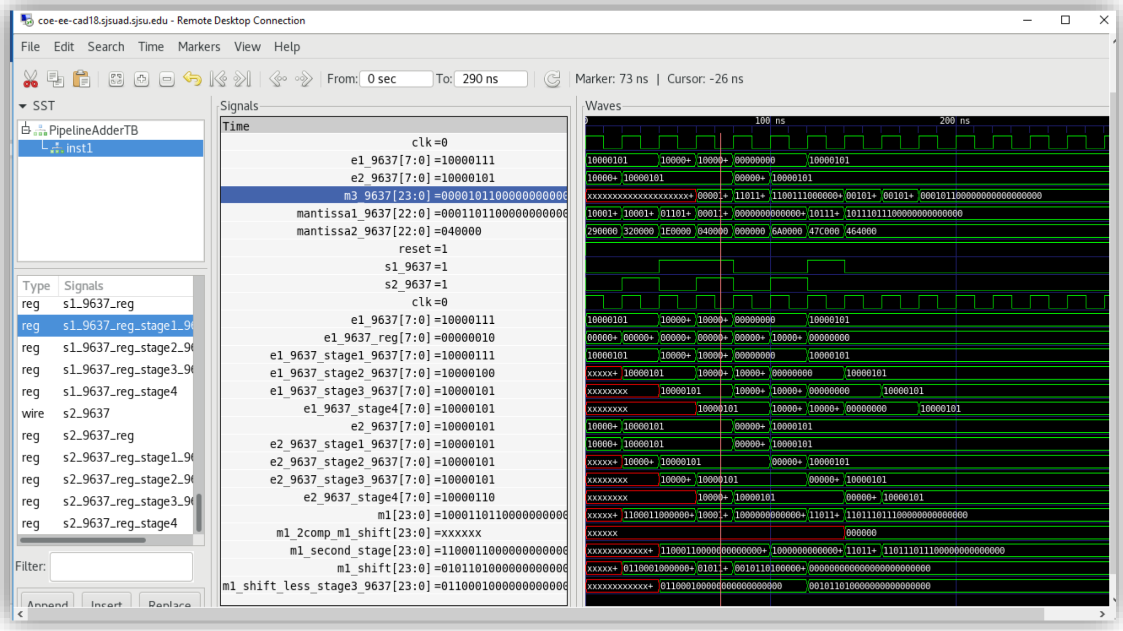
<u>Stage 2</u>: Here we Right-shift the mantissa of the smaller exponent, in this case 'm1' by the required one space on RHS.

<u>Stage 3</u>: After both mantissas have the same bias, we do a comparison of the mantissas to determine which one is smaller if the signs of both numbers are different we then take 2's complement of the smaller mantissa.
In the above example, signs are same and hence we need not require 2's complement.

<u>Stage 4:</u> The two mantissas 'm1' and 'm2' are added here. We also determine the number of shifts required and its corresponding direction to normalize the result
Here 1 shift is required to the Left-Hand Side since for the result.

<u>Stage 5</u>: The final mantissa is shifted to the required direction here it is LHS by 1 bit and the exponent is adjusted accordingly.

Below is the result of the simulation for the **above example** as show by the Flag.
This data comes for clock 5 due to pipeline. Below is shown unnormalized data of sa,e value at clock 4 as a part of the non-pipelined structure.


coe-ee-cad18.sjsuad.sjsu.edu - Remote Desktop Connection
File Edit Search Time Markers View Help
From: 0 sec To: 290 ns
Marker: 73 ns | Cursor: -26 ns
SST
PipelineAdderTB
inst1
Signals
Waves
100 ns
200 ns
Filter:
Append Insert Replace

Type	Signals
reg	s1_9637_reg
reg	s1_9637_reg_stage1_96
reg	s1_9637_reg_stage2_96
reg	s1_9637_reg_stage3_96
reg	s1_9637_reg_stage4
wire	s2_9637
reg	s2_9637_reg
reg	s2_9637_reg_stage1_96
reg	s2_9637_reg_stage2_96
reg	s2_9637_reg_stage3_96
reg	s2_9637_reg_stage4

```
Time
                              clk =0
                  e1_9637[7:0] =10000111
                  e2_9637[7:0] =10000101
                  m3_9637[23:0] =00001011000000000000
       mantissa1_9637[22:0] =00011011000000000000
       mantissa2_9637[22:0] =040000
                            reset =1
                          s1_9637 =1
                          s2_9637 =1
                              clk =0
                  e1_9637[7:0] =10000111
             e1_9637_reg[7:0] =00000010
     e1_9637_stage1_9637[7:0] =10000111
     e1_9637_stage2_9637[7:0] =10000100
     e1_9637_stage3_9637[7:0] =10000101
          e1_9637_stage4[7:0] =10000101
                  e2_9637[7:0] =10000101
     e2_9637_stage1_9637[7:0] =10000101
     e2_9637_stage2_9637[7:0] =10000101
     e2_9637_stage3_9637[7:0] =10000101
          e2_9637_stage4[7:0] =10000110
                    m1[23:0] =10001101100000000000
        m1_2comp_m1_shift[23:0] =xxxxxx
          m1_second_stage[23:0] =11000110000000000000
                m1_shift[23:0] =01011010000000000000
m1_shift_less_stage3_9637[23:0] =01100010000000000000
```

Result -
As seen in above screenshot, the simulated result match with the one we calculated theoretically as part of our learning process.
We saw that result comes same for both the designs for two examples, however the clock cycle at which it comes is different.

4. Conclusion

1. When we enter two numbers in IEEE 754 format in the circuit, we get the expected output in the same IEEE 754 format after normalization.
2. The Theory of Operation shows that both the designs (unpiped and pipelined) have the same output for two set of inputs.
3. The goals of achieving a adder circuit was successfully achieved until stage 4 as part of Verilog code, however due to some wire misconnection, stage 5 could not be achieved in given format. However, these are still represented in the graphs to make the reader understand how the desired output would look like.

Appendix:

s1_9637; //Sign bit of 1st Operator
s2_9637; //Sign bit of 2st Operator
s3_9637; //Sign bit of Output
[7: 0] e1_9637; //Bias bits of 1st Operator
[7: 0] e2_9637; //Bias bits of 2nd Operator
[7: 0] e4_9637; //Bias bits of Output
[22: 0] mantissa1_9637; //Mantissa bits of 1st Operator
[22: 0] mantissa2_9637; // Mantissa bits of 2st Operator
[22: 0] m3_9637; // Mantissa bits of Output
clk; //Clock
reset; //Reset