CS M152A Lab 1

Ryan Riahi

Introduction

The purpose of this laboratory was to design a combinational circuit that converts a 13-bit linear encoding of an analog signal into a compounded 9-bit Floating Point (FP) Representation.

Floating point representation in binary is done in the following way:

8	7	6	5	4	3	2	1	0
S	E			F				

The value represented by an 8-Bit Byte in this format is:

$$V = (-1)^S \text{ x } F \text{ x } 2^E$$

Conversion to floating point was done by first converting the two's compliment number to a sign magnitude value. We then record the most significant bit of this number and use it as the S value in the FP representation. After that we count the number of leading zero and base the value of the exponent (E) on it:

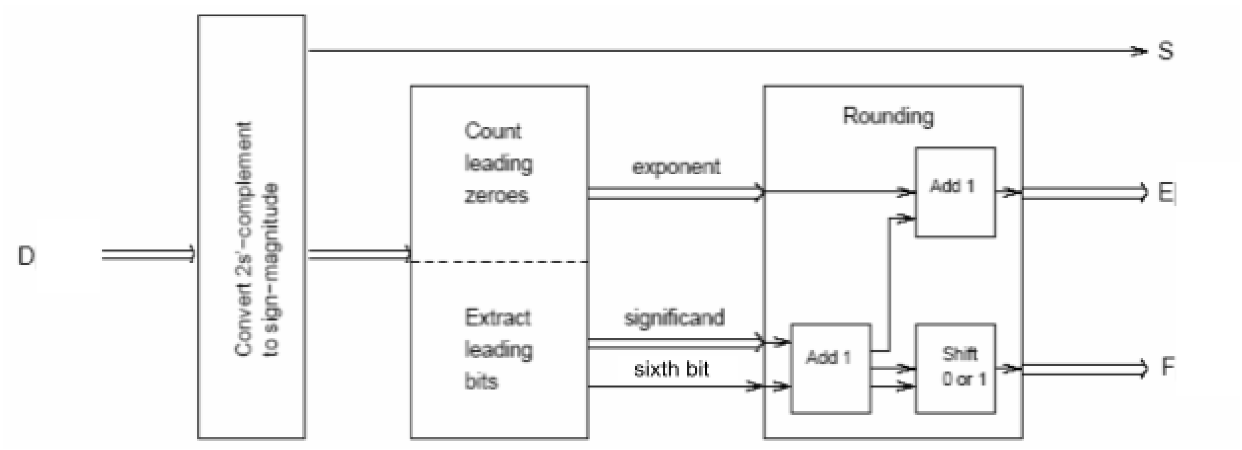
Leading Zeroes	Exponent
1	7
2	6
3	5
4	4
5	3
6	2
7	1
≥ 8	0

After that, we take the first 5 bits after the last leading zero from the sign magnitude value and use it as the significand (F) in the FP representation. Lastly, we round the FP representation since we cannot capture all 13-bit two's compliment values with only 8 bits. Along this process, we also have to handle cases where the values may overflow.

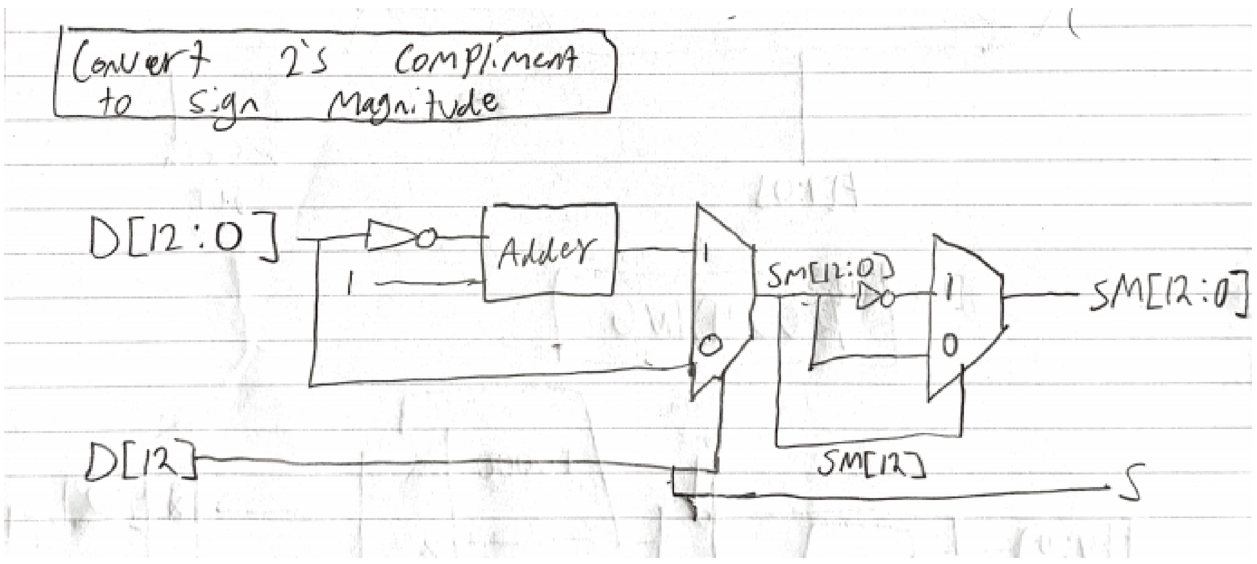
Design Description

The overall design (which is loosely described in the introduction) is show below:

FPCVT Pin Descriptions	
D [12 : 0]	Input data in Two's Complement Representation. D0 is the Least Significant Bit (LSB). D12 is the Most Significant Bit (MSB).
S	Sign bit of the Floating Point Representation.
E [2 : 0]	3-Bit Exponent of the Floating Point Representation.
F [4 : 0]	5-Bit Significand of the Floating Point Representation.

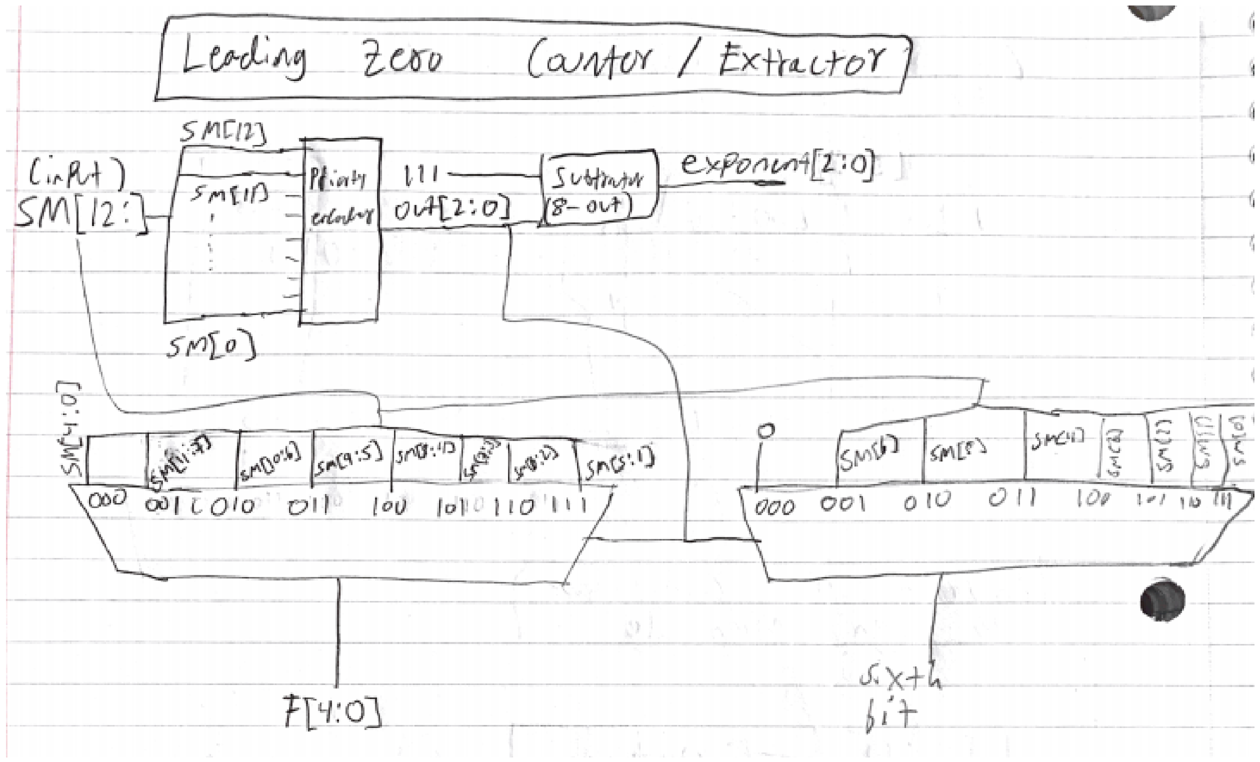


The 2's Compliment to Sign magnitude implementation is described below:



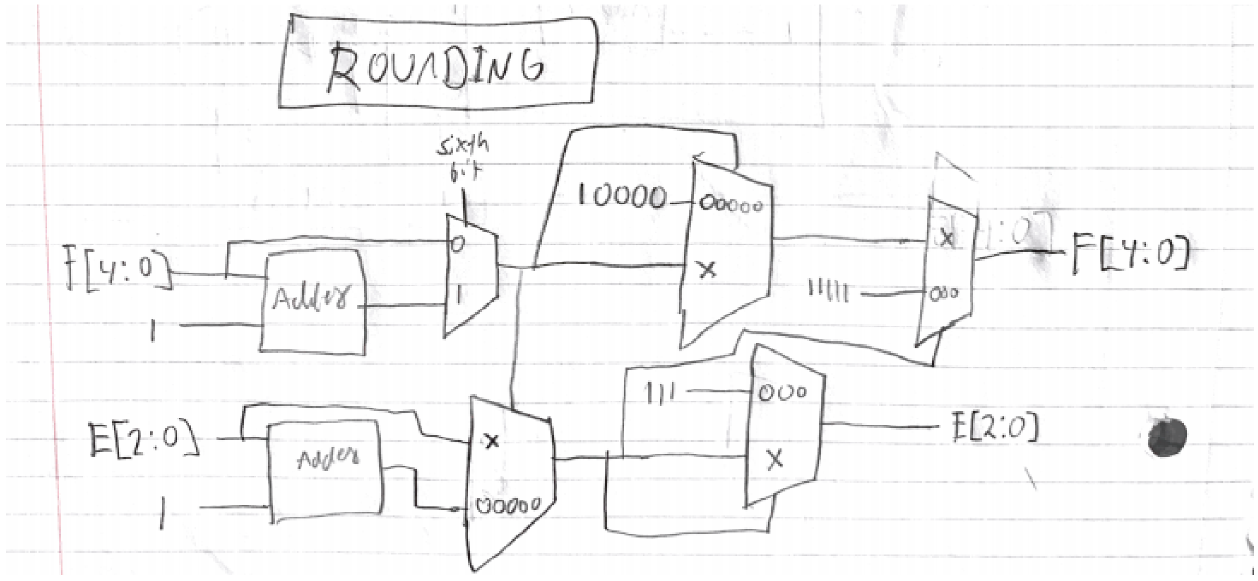
We use the sign bit (S) as the decider to the multpilexer which will output the untouched input (D) as the sign magnitude (SM) value if S is 0, otherwise we set SM to the compliment of D + 1. We then check for overflow by examining if the MSB of SM is 1. If so, we set SM to the compliment of SM.

The Count leading zeros implementation is described below:



We begin by passing the bits of SM to a priority encoder. This priority encoder will return the number of leading zeros. We take 8 subtracted by this number to be the exponent value (E). Afterwards, we pass the number of leading zeros as inputs to two multiplexers to index the Significand (F) value and the value of the sixth bit.

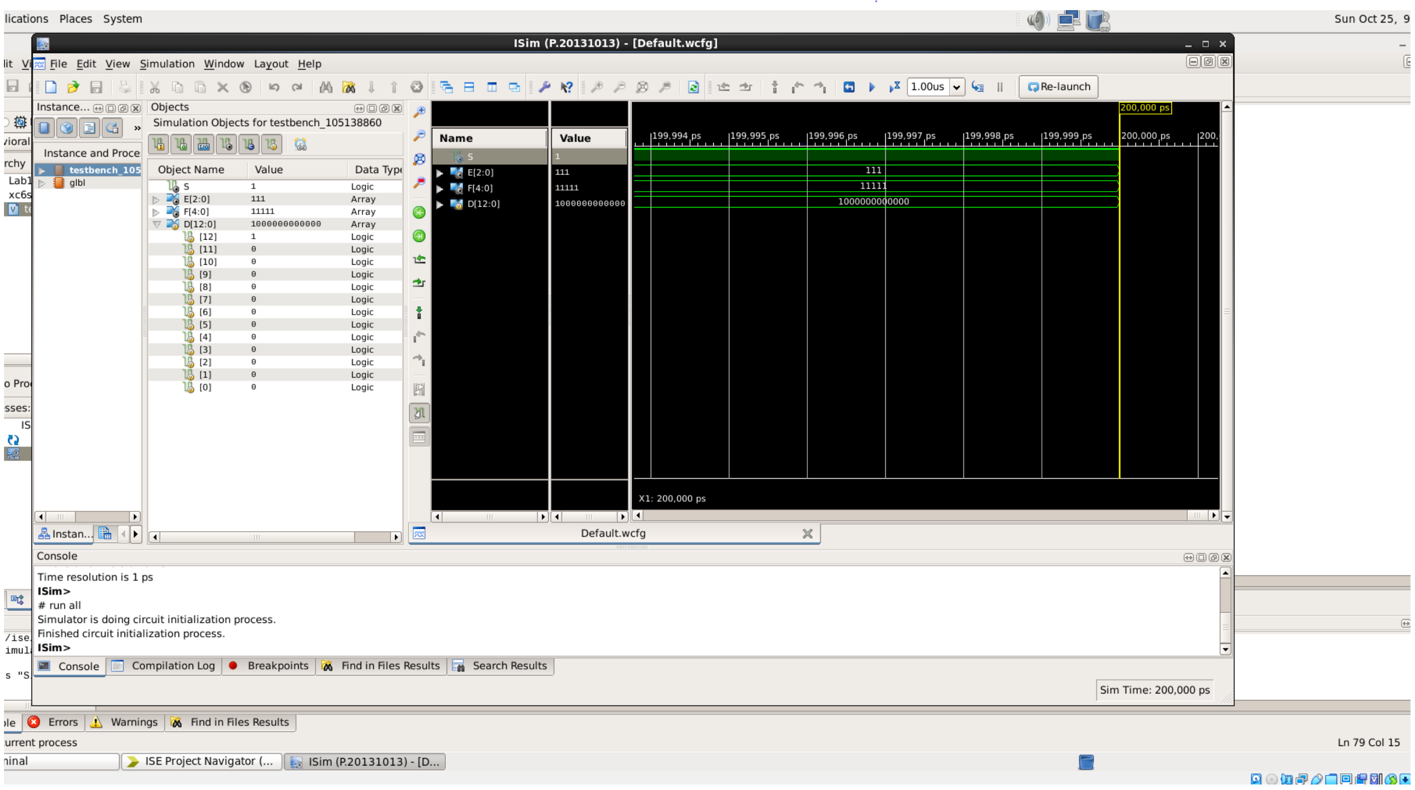
The Rounding implementation is described below:



To perform the rounding, we first check if the sixth bit is 1. If not, we simply return our inputs as outputs. However, if it is 1, we perform the rounding. In order to perform the rounding, we add 1 to the significand (F). After this, we check if F is all zeros. If so, F has overflowed and we return 5’b10000 for F and we increment E by 1. If E also overflows, we return E as 3’b111 and F as 5’b11111.

Simulation Documentation

In order to test my design, I ran a simulation using VISM. The simulation was performed with the testbench document which specified certain test cases. I tested a decent amount of random two’s compliment values along with certain edge cases (0, 1, -1, INT_MAX, INT_MIN). A screen capture of the simulation is shown below:



Simulation Objects for testbench_105138860

Object Name	Value	Data Type
S	1	Logic
E[2:0]	111	Array
F[4:0]	11111	Array
D[12:0]	1000000000000	Array
[12]	1	Logic
[11]	0	Logic
[10]	0	Logic
[9]	0	Logic
[8]	0	Logic
[7]	0	Logic
[6]	0	Logic
[5]	0	Logic
[4]	0	Logic
[3]	0	Logic
[2]	0	Logic
[1]	0	Logic
[0]	0	Logic

Name	Value
S	1
E[2:0]	111
F[4:0]	11111
D[12:0]	1000000000000

```
Time resolution is 1 ps
ISim>
# run all
Simulator is doing circuit initialization process.
Finished circuit initialization process.
ISim>
```

Sim Time: 200,000 ps

Here we can the input value of INT_MIN being passed into the converter which then outputs the maximum positive FP value.

Conclusion

In conclusion, we converted two's compliment values to FP representation in the following steps: 1. We first convert the two's compliment value to sign magnitude representation. 2. We then convert this value to FP by using the number of leading zeros and the bits following it. 3. We then round the FP value.

During this process we are also handling any possible overflow to ensure we get the most accurate representation possible.