CS M152A - Lab 1

Jair Hinojosa, Kyle Romero
904771873, 204747283
Section 5
TA: Fan Hin Hung
Date Performed: 1/16

1.0 - Introduction

The purpose of this laboratory was to demonstrate the correct Verilog implementation of a design specification. Our task was to implement a combinational circuit which converts a 12-bit analog two's complement signal to its corresponding 8-bit floating point representation. We accomplished this by converting the 12-bit input to three signals: a 1-bit sign (S), a 3-bit exponent (E), and a 4-bit mantissa (F). The relationship between the input value (V) and the outputs above is summarized with the following:

$$V = (-1)^S * 2^E * F \quad \textbf{(1.0.1)}$$

Note: This lab only consisted of Verilog implementation and simulation using the Xilinx interface; no FPGA component was required.

1.1 - Background

Computers often utilize two's complement form to represent signed integers. An n-bit two's complement integer can be used to represent numbers in the range $[(-2)^{n-1}, (2^{n-1})-1]$. The most significant bit determines the sign by having a weight of $(-2)^{n-1}$ and each subsequent bit has a weight of 2^{n-k}, where k is the index of the bit from the left plus one.

-2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
[11]	[10]	[9]	[8]	[7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]

Figure 1.1.1 - The weight of each bit in 12-bit two's complement form.

Moreover, floating point form allows computers to represent many more numbers by using a form of scientific notation. Floating point numbers are broken up into different sections representing the sign bit (S), the exponent field (E), and the mantissa (F). The value represented by a specific floating point form can be calculated using **Formula 1.0.1**.

S	**E**			**F**			
[7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]

Figure 1.1.2 - 8-bit Floating Point Representation

Conversions between these two representations involve some considerations. For example, 12-bit two's complement form can represent more numbers than 8-bit floating point form. Furthermore, multiple numbers can map to the same floating point form because we can only represent numbers which are products of a power of two and the mantissa. These nuances present challenges when implementing circuits that manipulate representations of numbers.

1.2 - Design Requirements

Our task was to implement a module, called FPCVT, which would convert a signed 12-bit two's complement integer to its corresponding floating point approximation.

Input/Output Name	Description
D[11:0]	Input: Signed two's complement representation.
S	Output: The sign bit of the floating point output
E[2:0]	Output: The 3-bit exponent of the floating point output
F[3:0]	Output: The 4-bit mantissa of the floating point output

Figure 1.2.1 - The input/output pins of the FPCVT module

The following are some important notes that the laboratory specification makes regarding the algorithm for performing this conversion.

Leading Zeros and the Exponent Bits

The specification explicitly states the number of leading zeros in the 12-bit absolute value of D provides us with enough information to determine the correct value for E.

Leading Zeros	Exponent
1	7
2	6
3	5
4	4
5	3
6	2
7	1
>= 8	0

Figure 1.2.3 - The number of leading zeros in the 12-bit absolute value of D and the corresponding value for E.

Note that we require the 12-bit absolute value of D in order utilize the chart above. We must consider that negative numbers will need to be converted to their absolute value before consulting the chart. The absolute value (A) of a negative 12-bit two's complement number (D) can be found by using the following formula:

$$A = \sim(D) + 1 \qquad (1.2.1)$$

Also note that we can also resolve the mantissa by truncating the leading zeros and taking the next four bits to be F.

Rounding

In order to obtain the closest floating point approximation for a given input, rounding may need to be taken into account. The solution to this problem is to look at the first bit to the right of the four bits taken to be F, which we called the fifth bit (FB). If FB is 0, then we need to round down the mantissa is F. If the FB is 1, then we need to round up and the mantissa is F + 1.

Edge Case Considerations

The rounding procedure outlined in the specification introduces some edge case considerations that must be made. If we round up and F overflows, then we must be sure to add one to E and shift F down by one bit. If this causes E to overflow, then we must saturate E and F to all 1s.

Likewise, if we are presented an input that is too large or too small to be represented in 8-bit floating point, then we must saturate E and F to all 1s. This case would occur if there are no leading zeros in the absolute value form of D.

2.0 - Design Documentation

The overall design of our combinational circuit is modularized into three parts. First, S is passed through by extracting it from D. Then, D is passed to the first module, the Absolute Value Converter.

After correct absolute value form of D is then passed to the Leading Zero Counter module which uses the chart in **Figure 1.2.3** to resolve E' and then truncates the leading zeros to determine F'. The fifth bit signal is taken to be the first bit to the right of the F' bits. The no leading zero edge case is resolved in this stage.

The Rounding and Edge Case Handling module uses E', F', and the fifth bit to resolve any rounding that must be done in order to produce the closest floating point approximation of D. Additionally, any additions, shifts, or saturations are resolved at the end of this stage.

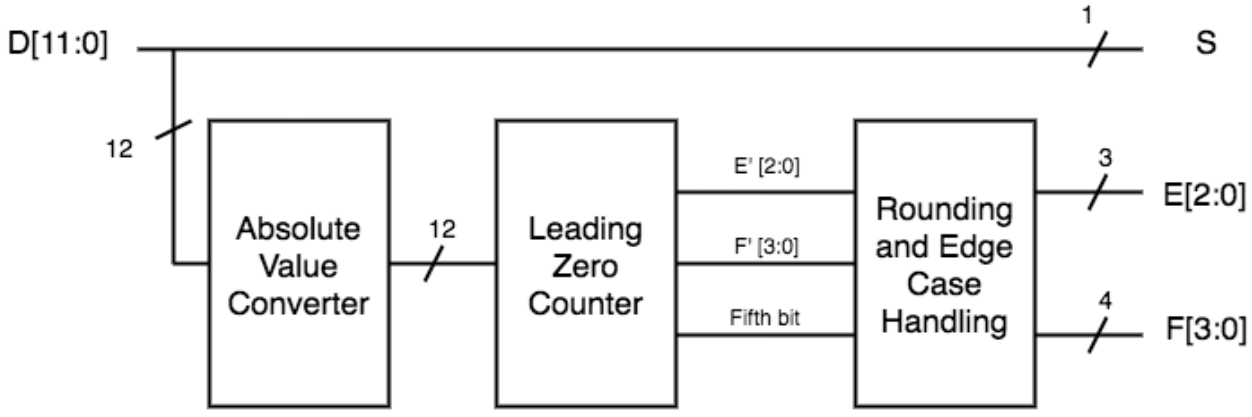

Figure 2.0.1 - A diagram showing the top level modules of our design.

2.1 - Absolute Value Converter

Our Absolute Value Converter module can be thought of as a multiplexer which uses the sign bit to select for the correct 12-bit absolute value form of D.

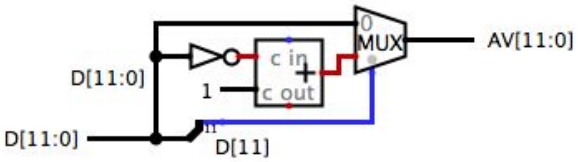

Figure 2.1.1 - The combinational circuit that outputs the 12-bit absolute value of D (AV).

2.2 - Leading Zero Counter and Selector

Our implementation of a leading zero counter and selector can be thought of as three multiplexers that either select for constant values or pass on specific bits from the input.

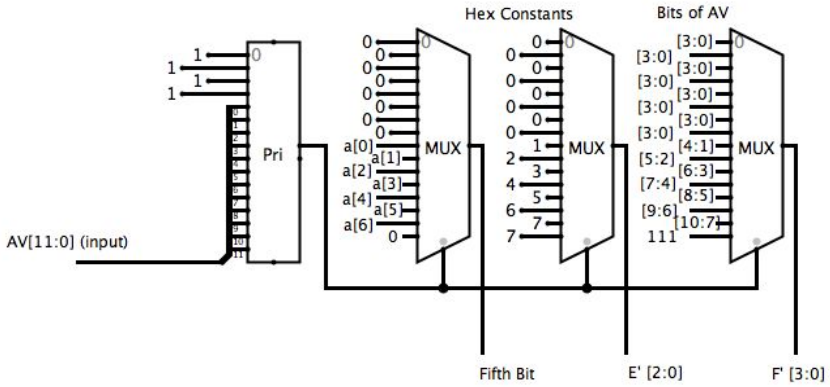

Figure 2.2.1 - The multiplexers/priority encoder that resolve F', E', and fifth bit.

2.3 - Rounding and Edge Case Resolver

Our implementation of a rounder uses the fifth bit to determine whether or not the mantissa bits need adjustment. Our combinational circuit adds 1 to the mantissa if the fifth bit indicates that we should round and then resolves any potential overflowing issues.

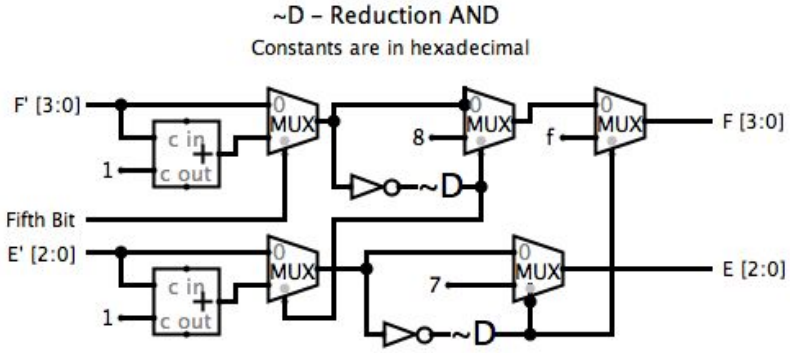

Figure 2.3.1 - The combinational circuit that implements our rounding and resolves any edge case overflows.

3.0 - Simulation Documentation

Incremental development allowed us to test our Verilog code in small chunks. After completing each module, we would confirm some basic functionality by noting the partial output of each of our test cases (S or E or F) before moving on to the next component of the system. Thus, if each component worked according to its specification, we could be confident that the system would function as intended.

As is common in computer science, there are some specific project requirements that create certain edge cases that are more likely to produce errors than the general use cases. The following are the error prone edge cases that we identified in writing test cases for our system simulation:

- Zero
- ± Max 12-bit #
- (- Max 12-bit #) + 1
- Rounding Cases
- #'s that overflow F
- #'s that overflow E

Along with some additional arbitrary values for general functionality, these signals were enough to test the module for the project requirements outlined in **Section 1.2**.

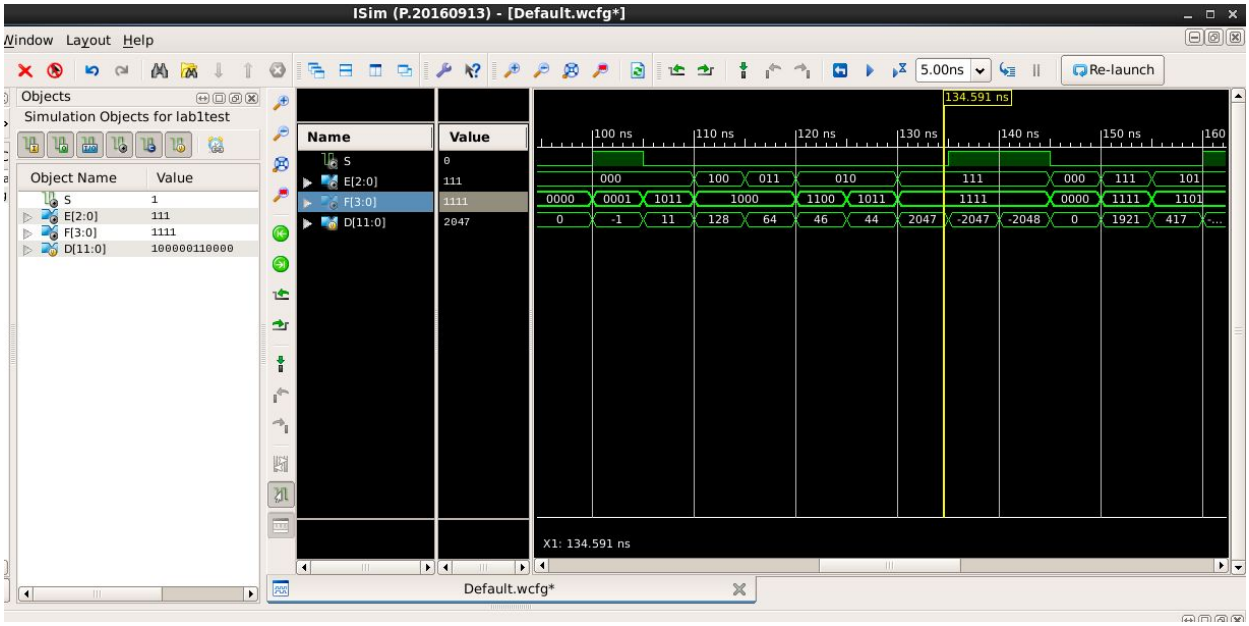

Figure 3.0.1 - Several test cases are shown in this waveform.

We compared the outputs of these test cases with either conversions that we did by hand or by comparing to values provided in the lab manual. We are not aware of any bugs that were not resolved before the project deadline.

4.0 - Conclusion

In summary, our FPCVT module is able to convert a 12-bit two's complement integer into its closest floating point approximation using combinational logic. The floating point output is represented using a 3-bit exponent (E), a 4-bit mantissa, and a sign bit (S). We were able to successfully implement this project specification by combining our Absolute Value Converter (**Section 2.1**), Leading Zero Counter and Selector (**Section 2.2**), and Rounding and Edge Case Resolver (**Section 2.3**) into a well-functioning system.

We faced some challenges in becoming familiar with the Xilinx interface and learning about some nuances in Verilog. Our TA, the provided course materials, and Google were extremely helpful in remedying these difficulties. Otherwise, we found this lab fairly easy since both of us have recently used Verilog in a related course. All in all, this lab served as a very nice Verilog refresher and introduction to the Xilinx interface and do not have any recommendations about how to improve this project.