Sriram Balachandran (UID: 805167463)

Mohit Garg

CS M152A: Introductory Digital Design Laboratory

Section 2

October 25th 2020

Laboratory 1 Report

Introduction & Background

In order to grow accustomed to Verilog programming, we were tasked with utilizing Xlinx ISE software to design and test a combinational circuit module, FPCVT, to convert a 13-bit linear encoding of an analog signal into a compounded 9-bit Floating Point representation.

Traditional linear encoding is what we're most familiar with - it involves addition of the values represented by each bit. Floating Point representations utilize a compounded representation, where the analog signal can be divided into segments, and the value of each segment is multiplied together in some manner. In the case of this assignment, we convert a 13-bit signal into 3 signals composing a Floating Point representation: a single sign bit (S), a 3-bit exponent (E), and a 5-bit significand (F). In order to calculate the value of the FP representation, we use the following formula:

$$\text{Value} = (-1)^S * 2^E * F$$

Figure 1: Formula to calculate value of Floating Point representation

Due to the multiplicative nature of the FP representation, it's easy to see that it allows us to represent a wider range of numbers than traditional linear encoding would, in terms of the smallest and largest numbers we can represent. But in the context of this assignment, it is important to note that a 13-bit two's complement signal can represent more numbers than a 9-bit floating point signal.

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

Figure 2: 9-bit Floating Point Representation Subcomponent

However, it should also be evident that multiple numbers can map to identical FP representations since we can only represent numbers as products of a power of two and the significand.

Design Requirements

The lab specification gives us the following information regarding a viable algorithm to perform the conversion.

In order to determine the correct value for E, we are given the fact that E can be determined from the number of leading zeros in the 13-bit absolute value of D.

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

Figure 3: Number of leading zeros in 13-bit absolute value and corresponding decimal value for E.

Since we need the absolute value of D in order to use the above information, we should briefly mention the method by which negative two's complement linear representations can be converted into their absolute value:

Absolute Value = ~(D) + 1

Figure 4: Negation + Incrementation formula for absolute value of a negative 2's complement representation.

The significand can be resolved by stripping the leading zeros and taking the next five bits to be F. To find the closest floating point representation for a given input, we may need to round the output value. In order to determine whether or not we need to round, we look at the first bit following the five bits we take to be the significand, which will be called the sixth bit (SB) from this point forward. Assuming the five bits to the right of the leading 0 to be F, if the SB is 0, no rounding is necessary, and if the SB is 1, then we increment F by 1.

Edge Cases

The first edge case is somewhat obvious - if we are presented with inputs that are simply too small or too large to be represented as a 9-bit floating point number, we must saturate E and F to all 1s. The other edge case is less apparent - the aforementioned rounding cases could result in overflow. If we round up and F overflows, E must be incremented and we must Logical Shift F one bit to the right. If the incrementing of E causes E to overflow, we must then saturate E and F with 1s again.

Design Description

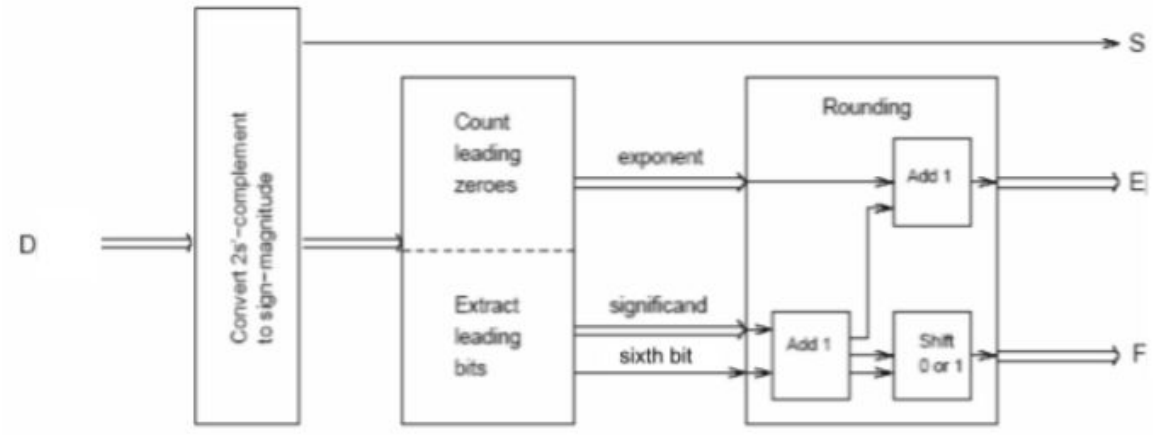

Figure 5: FPCVT Modular Design Outline (taken from lab specification)

Figure 5 shows the basic flow of the entire FPCVT module. First, we set the value of S simply by looking at the Most Significant Bit of D. After that, we apply the absolute value conversion mentioned in Figure 4. We then determine the number of leading zeros and extract temporary values for E and F, as well as the value of the SB. Those input values are used to determine whether any rounding or edge case mitigation must be performed, and determines the final values of E and F.

Simulation Documentation

My development process followed a two-step process. My first goal was to implement the bare bones logic necessary to perform the conversion on a standard input that didn't fall into a potential edge case. This involved implementing the absolute value conversion section of the code as well as the leading zero counting section of the code. When I first compiled these two sections of the code successfully and attempted to run a simulation, I found that although my code compiled, it couldn't be synthesized. This was due to my misuse of `assign` statements in my code. As such for future endeavors, when writing Verilog code, I will ensure that my code not only compiles but that it is also synthesizable. After I had this basic functionality in place, I then implemented the rounding module, also ensuring that I handled any potential overflow as

well as the edge case involved with the negation + incrementation absolute value conversion process.

I chose to test some basic arbitrary values, but he lab specification outlined some edge cases that could cause issues in our implementations, most notably the following: Zero value, +/- Max 13-bit Number, Numbers that overflow F/E, Rounding Cases, and the negative max 13-bit Number + 1. I added test cases to cover these edge cases as well.

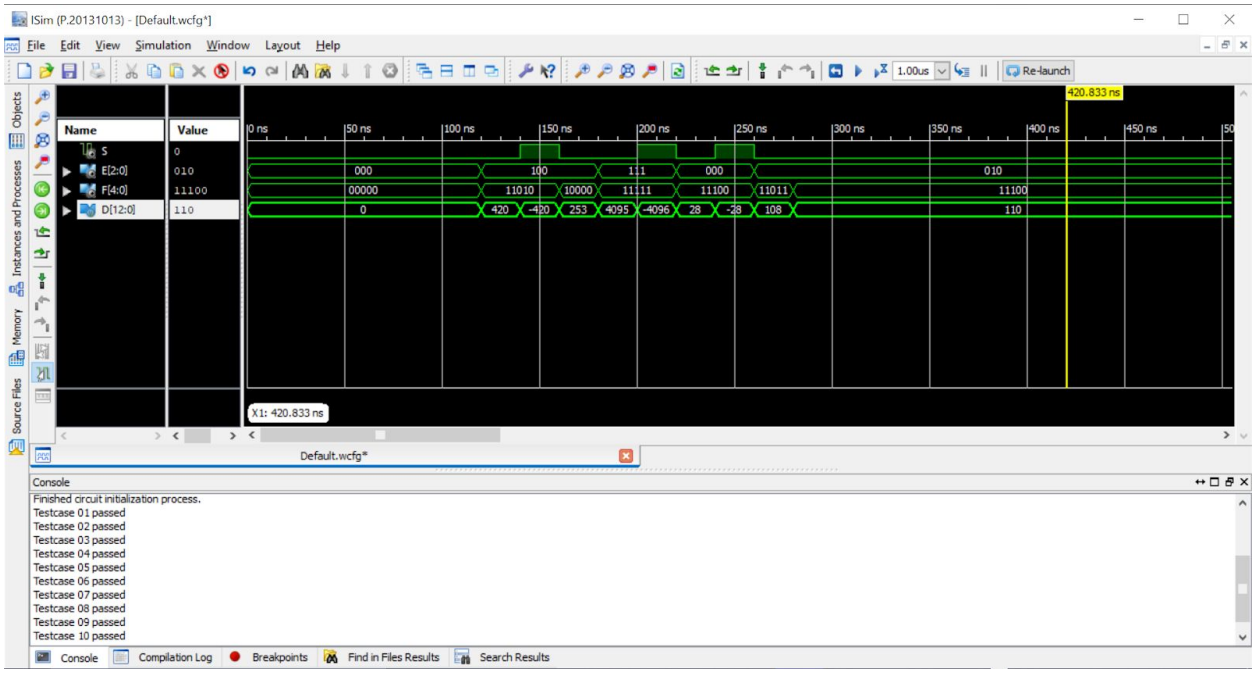

Figure 6: Waveform display of several test cases, as well as console ouptut

Conclusion

My floating point conversion module was able to successfully convert a 13-bit two's complement integer into its closest compounded FP representation using synthesizable combinational logic. The output representation can be split into 3 components - a sign bit (S), the 3-bit exponent (E), and a 5-bit significand (F). This conversion was handled by a design consisting of 3 primary components - an absolute value converter to extract sign and magnitude, a leading zero counter to determine approximate values of E, F and the value of the Sixth Bit, and a rounding module to round and handle edge cases to finalize the E and F values.

I think this assignment was a nice introduction to the Verilog language. By providing the algorithm in the specification, it allowed us to focus on learning implementation details and grow accustomed to development within the Xlinx ISE environment. At this time, I don't have any suggested improvements.