Lab 1 Report

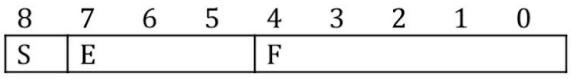
Group 7
Daniel Chris Daniil - 504636684
Junhong Wang - 504941113

1. Introduction and requirements

Background Information
We want to design a combinational circuit that converts 13-bit two's complement representation number into 9-bit floating point representation.

Design requirements

- The module is called FPCVT.
- FPCVT input: D (13 bits)
- FPCVT outputs: S (1 bit), E (3 bits), and F (5 bits).
- D denotes the two's complement representation of the number we want to convert.
- S denotes the sign of the floating point representation.
- E denotes the exponent of the floating point representation
- F denotes the significand of the floating point representation
- $D = (-1)^S * F * 2^E$
- S, E and F are formatted as the follows:



- When chopping off some significand, round up the significand by 1 if the next bit is 1. For example, when F=D[10:6], if D[5] is 1, round up. Otherwise, don't round up. If the significand is already all 1's, round up the exponent, then set the significand to be 10000. If exponent is also all 1's, then no need to round up.
- When D is -4096 (1_0000_0000_0000), complement D and add 1 results in -4096 instead of +4096. Handle this case carefully and represents it with the most negative number possible. In other words, if D is -4096, S, E, and F should be 1, 111, and 11111 respectively.

2. Design description

Design Description

We can easily find the sign bit by looking at the most significant bit of the input. Then, we convert the input to positive number. Next, we check the bits of unsigned input from most significant bits to least significant bits, and find identify the first occurance of a 1. In other words, we are counting the number of leading zeros. Once we know how many leading zeros there are, we know the lookup the exponent in the provided table that associated leading zeros with the exponent. At this point we also know the significand and rounding bit. Note exponent and significand here are not finalized yet because we still need to take care of rounding. Finally, we look at the rounding bit to see if we need to round the number. We also take care of the edge cases at this point.

Modular Architecture

1. Determine the value of unsigned D based on D
2. Determine the value of temporary exponent based on unsigned D
3. Determine the value of temporary significand based on unsigned D
4. Determine the value of rounding bit based on unsigned D
5. Determine the value of S based on D
6. Determine the value of F and E
    a. If rounding bit is 1, significand is all 1's, exponent is all 1's, then we don't have enough bit to round up. Set F=significand and E=exponent.
    b. If rounding bit is 1, significand is all 1's, then we round up the exponent instead. Set F=10000 and E=exponent+1
    c. If rounding bit is 1, then we round up the significand. Set F=significand+1 and E=exponent.
    d. If rounding bit is 0, significand is all 0's, exponent is all 0's, S is 1, then D was 1_0000_0000_0000. This is a special case. Set F=11111 and E=111.
    e. If rounding bit is 0, then set F=significand and E=exponent.

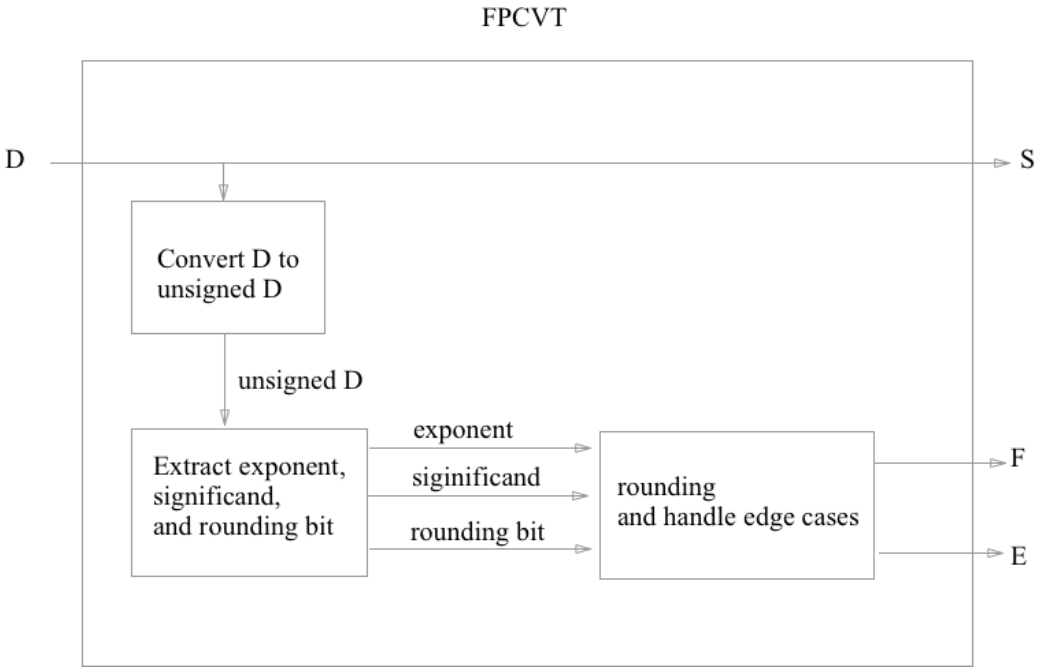
Interaction among the Modules

There is only one module.

Interface of Each Major Module

The FPCVT module uses D (13 bits) as input, then outputs S (1 bit), E (3 bits), and F (5 bits).

Schematics for System Architecture



3. Simulation documentation

Requirement Tested / Test Cases

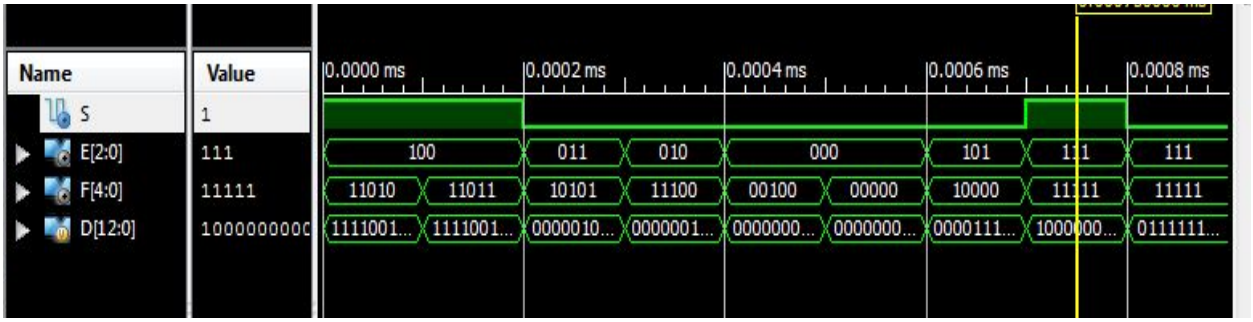
Requirement Tested	Test Case	Result F	Result E	Result S
Negative number	1_1110_0101_1010	11010	100	1
Negative and requires rounding	1_1110_0101_0010	11011	100	1
Positive number	0_0000_1010_0001	10100	011	0
Positive and Rounding	0_0000_0110_1111	11100	010	0

Significand starts from 0	0_0000_0000_0100	00100	000	0
Significand all 1s and needs rounding	0_0001_1111_1111	10000	101	0
Largest negative number	1_0000_0000_0000	11111	111	1
Significand all 1s, Exponent all 1s and rounding	0_1111_1100_0000	11111	111	0

Bugs

There were no bugs identified during the test case runs.

Simulation Waveforms



4. Conclusion

Summary of the Design

1. Determine the value of S by looking at the most significant bit of D.
2. Determine the number of leading zeros, and lookup exponent
3. Determine the significand using the 5 bits following the last leading zero
4. Check if rounding is required
5. If rounding is indeed required, check for edge cases, i.e. significand all 1s, exponent all 1s

Difficulties

We encountered several difficulties while implementing the provided design. We implemented FPCVT with four different methods.

1. Using if else

The first thing we tried is to use a long if else statements to check all the possible values of D. The maximum delay was 15.068 ns.

2. Using wire and assign

The next thing we tried is to use wires. We used assign statements and were able to reduce the number of if else statements in the always block from previous method. The maximum delay was 12.173 ns.

3. Using for loop

We also tried replacing the first method with for loop. Before we start handling edge cases, the maximum delay was already 12.720 ns, which is slower than the second method. So we stopped using this method.

4. Using case statements

Finally we tried replacing the second method with case statements. Instead of using wires, we used reg for significand, exponent, and rounding bit. Then assign these values inside the always block with case statements. The maximum was 13.498 ns.

We chose the second method as our final design as it has the least maximum delay.

Suggestions

Given that a fair level of understanding of Verilog was acquired during Lab 0, there are not any suggestions to this lab other than maybe a softer exercise before the lab to get us better acquainted with Verilog. we feel like the previous lab focused more on us learning the programming environment instead of learning how to program in Verilog.