Project 1: Floating Point Conversion

Haley Kim
405111152

Introduction and Requirement

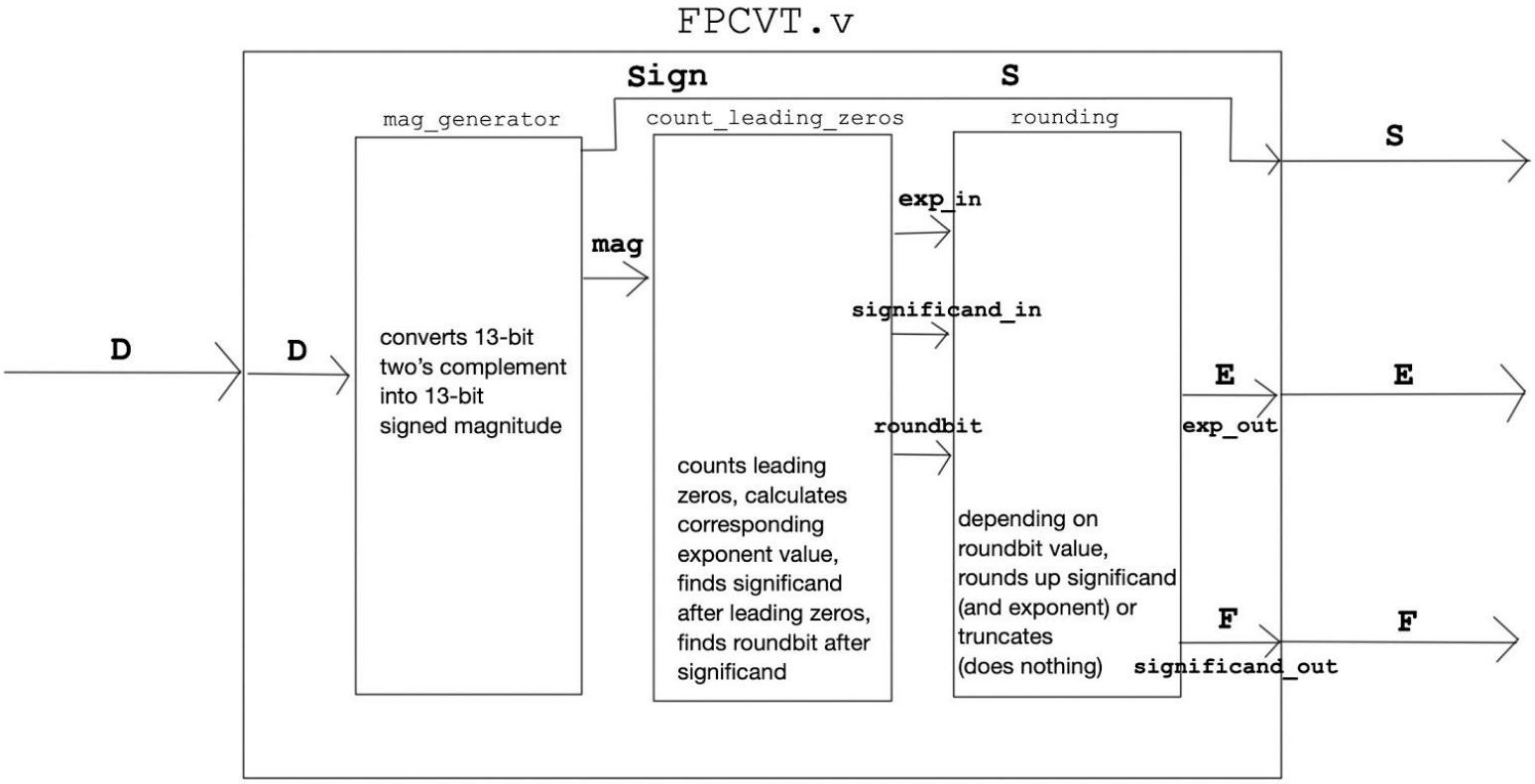
The two's complement binary number has the most significant bit as either 0 or 1, where 1 signifies the negative value of that bit position. The following bits act as regular unsigned bits, and are added to the most significant bit's value. For an example, 1_1111 in two's complement form would signify -16 + 8 + 4 + 2 + 1, or -1. Thus, two's complement numbers can represent both negative and positive integers.

The floating point binary number has a sign bit, the exponent bits, and the significand (or mantissa) bits. Together, they represent (-1)^(sign bit) * (significand) * 2^(exponent), and thus are able to represent both positive and negative decimal numbers.

This project is designed to allow students to create their own floating point converter, from a 13-bit two's complement binary number to a 9-bit floating point binary number. The difficult part of this project is that this is not a one-to-one mapping, due to its compressive nature. From the given 13-bit two's complement number (input D), students must extrapolate a sign bit (output S), 3 exponent bits (output E), and 5 significand, or mantissa, bits (output F). Given an input in the form **D_DDDD_DDDD_DDDD**, the final output will resemble **S_EEE_FFFFF**. In decimal form, this floating point value will resemble **(-1)^S * F * 2^E**.

Design Description

The main overarching module is FPCVT, with input [12:0] D, and outputs S, [2:0] E, and [4:0] F.
This module FPCVT employs modules **mag_generator**, **count_leading_zeros**, and **rounding**.



Module FPCVT is to be used as:
FPCVT [instance_name1] (.D(D), .S(S), .E(E), .F(F));
FPCVT's logic:

takes in "D" as input
outputs "S", "E", and "F"
has no real logic, uses mag_generator, count_leading_zeros, and rounding

Module mag_genereator is to be used as:
mag_generator [instance_name2] (.D(D), .sign(S), .mag([var_name1]));
mag_generator's logic:

takes in "D" as input
if sign bit of D is 0, mag is set equal to D

if sign bit of D is 1, mag is D flipped bit by bit and added by 1

outputs “mag” and “sign”

Module count_leading_zeros is to be used as:

count_leading_zeros [instance_name3] (.mag([var_name1]), .exp([var_name2]), .significand([var_name3]), .roundbit([var_name4]));

count_leading_zeros’ logic:

takes in “mag” as input

uses casex statement to parse through all possible options starting from 0_1xxx_xxxx_xxxx up to 0_0000_0000_0000 with special case 1_1111_1111_1111

with each case, assigns corresponding exponent, significand, and roundbit

outputs exp, significand, and roudbit

Module rounding is to be used as:

rounding [instance_name4] (.roundbit[var_name4]), .exp_in([var_name2]), .significand_in([var_name3]), .exp_out(E), significand_out(F));

rounding’s logic:

takes in “roundbit,” “exp_in,” and “significand_in” as input

if roundbit is 1

if significand is not 11111, then add 1 to significand

if significand is 11111 but exp is not 111, turn significand into 10000 and add 1 to exp

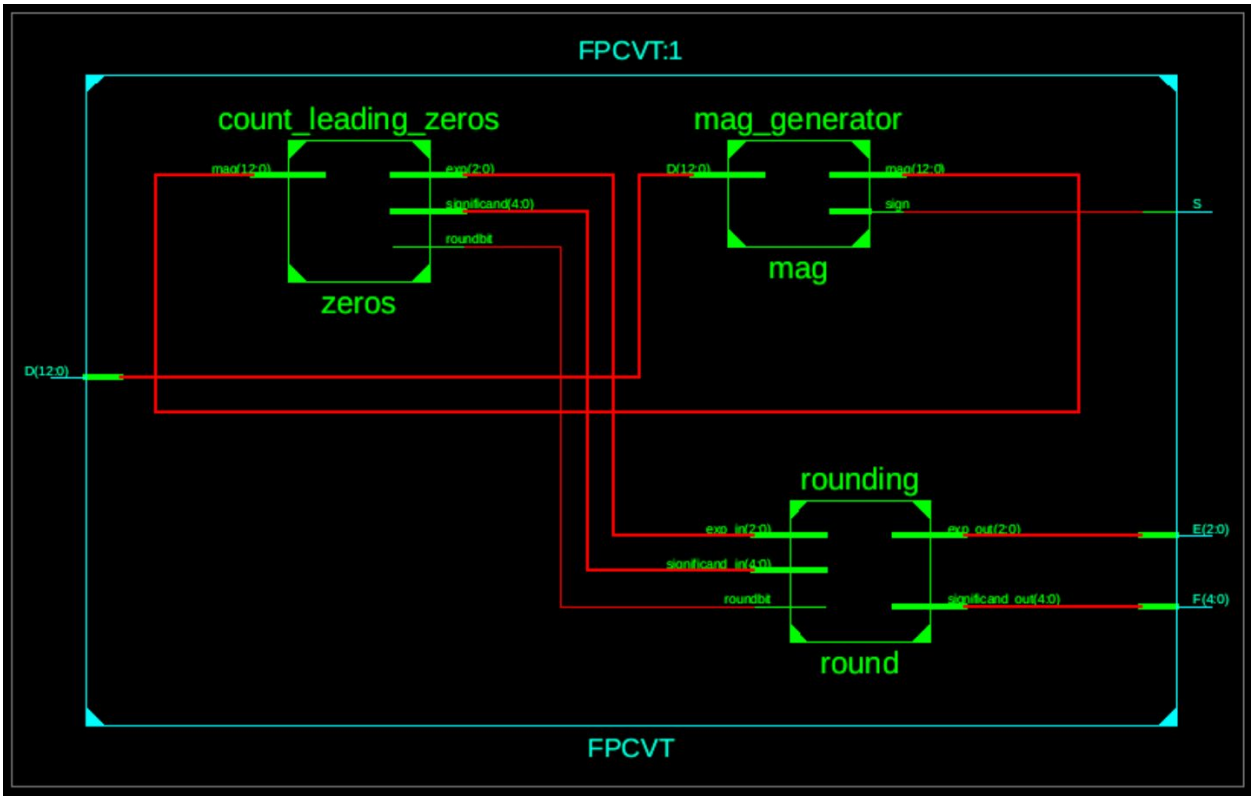
if significand is 11111 and exp is 111, leave them both be because they are at maximum

if roundbit is 0

leave both significand and exp be

outputs “exp_out” and “significand_out”

Simulation Documentation




```
=========================================================
HDL Synthesis Report

Macro Statistics
# Adders/Subtractors                                   : 3
 13-bit adder                                         : 1
 3-bit adder                                          : 1
 5-bit adder                                          : 1
# Latches                                              : 3
 1-bit latch                                          : 3
# Multiplexers                                         : 9
 1-bit 2-to-1 multiplexer                             : 4
 13-bit 2-to-1 multiplexer                            : 2
 3-bit 2-to-1 multiplexer                             : 1
 5-bit 2-to-1 multiplexer                             : 2

=========================================================
```

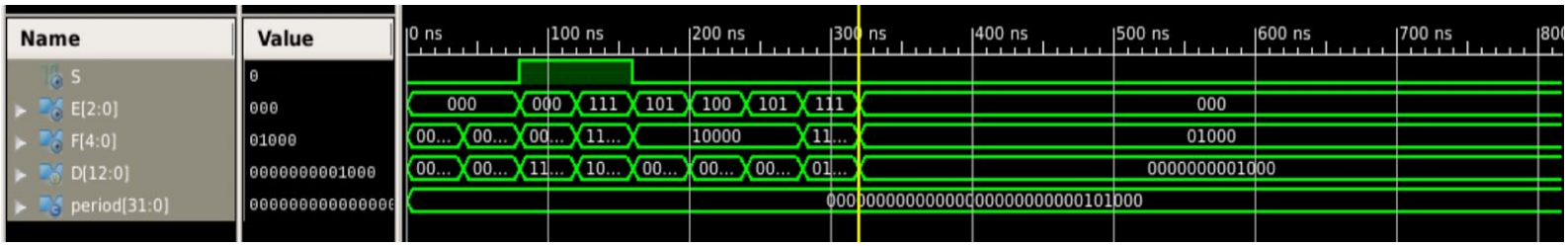
FPCVT Project Status (10/23/2020 - 23:06:19)			
Project File:	Proj1.xise	**Parser Errors:**	No Errors
Module Name:	FPCVT	**Implementation State:**	Mapped
Target Device:	xc6slx4-3tqg144	**• Errors:**	x 1 Error (0 new)
Product Version:	ISE 14.7	**• Warnings:**	3 Warnings (0 new)
Design Goal:	Balanced	**• Routing Results:**	
Design Strategy:	Xilinx Default (unlocked)	**• Timing Constraints:**	
Environment:	System Settings	**• Final Timing Score:**	

Device Utilization Summary	[-]

Detailed Reports					[-]
Report Name	**Status**	**Generated**	**Errors**	**Warnings**	**Infos**
Synthesis Report	Current	Fri Oct 23 23:05:59 2020	0	3 Warnings (0 new)	1 Info (0 new)
Translation Report	Current	Fri Oct 23 23:06:14 2020	0	0	0
Map Report	Current	Fri Oct 23 23:06:18 2020	x 1 Error (0 new)	0	0
Place and Route Report					
Power Report					
Post-PAR Static Timing Report					
Bitgen Report					

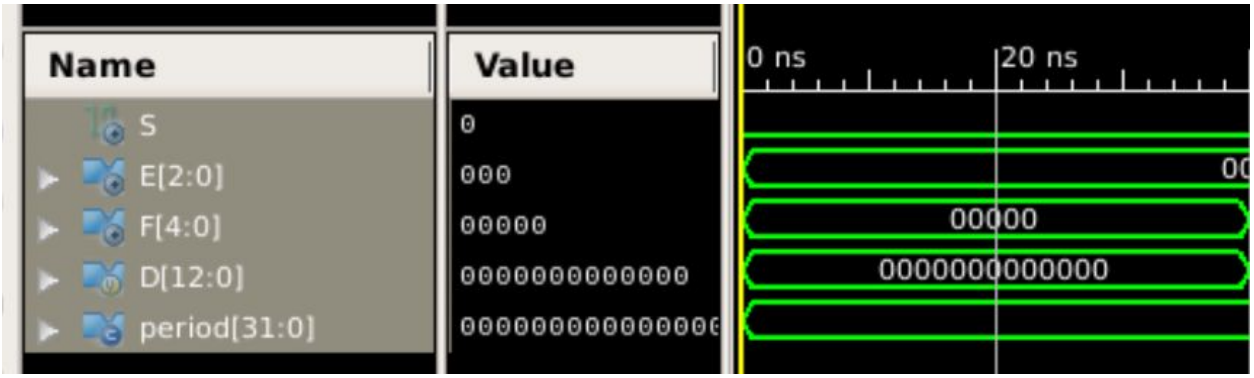
Secondary Reports			[-]
Report Name	**Status**	**Generated**	
ISIM Simulator Log	Out of Date	Fri Oct 23 23:02:29 2020	

Date Generated: 10/23/2020 - 23:11:38



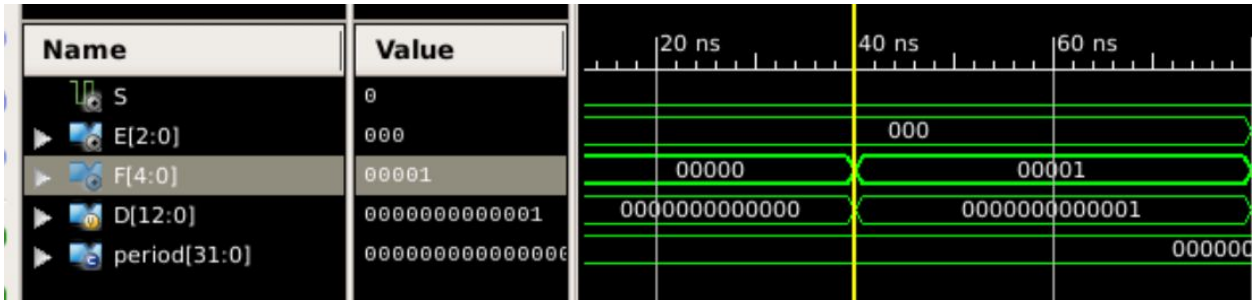
Test case 1: D = 0_0000_0000_0000

0 is a nominal case, and could behave differently. Should result in 0_000_00000.



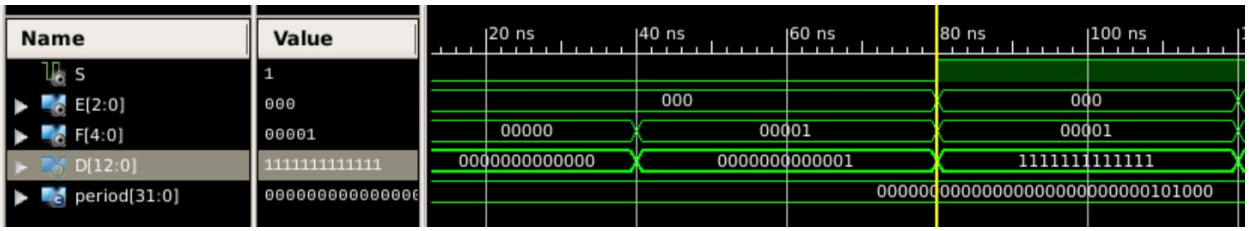
Test case 2: D = 0_0000_0000_0001

+1 is a nominal case, and could behave differently. Should result in 0_000_00001.



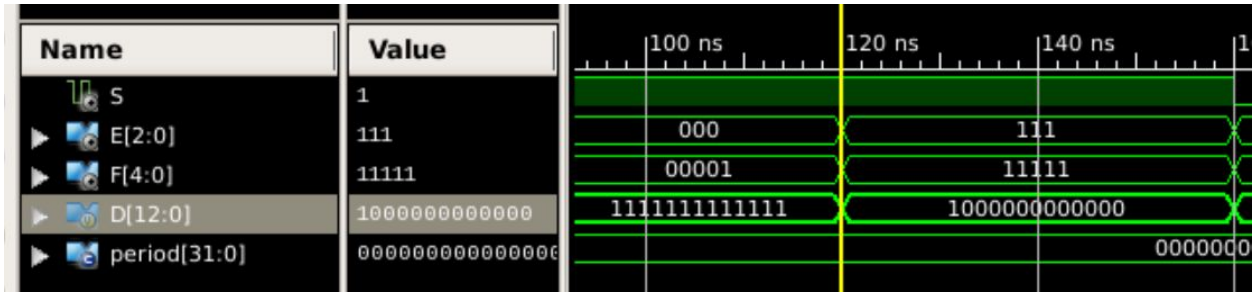
Test case 3: D = 1_1111_1111_1111

-1 is a nominal case, and could behave differently. Should result in 1_000_00001



Test case 4: D = 1_0000_0000_0000

-4096 was listed in the lab 1 specification as a unique case. Conversion to signed magnitude may pose a problem. Should result in 1_111_11111, the largest number representable by a 9-bit floating point.



Test case 5: D = 0_0001_1111_1010

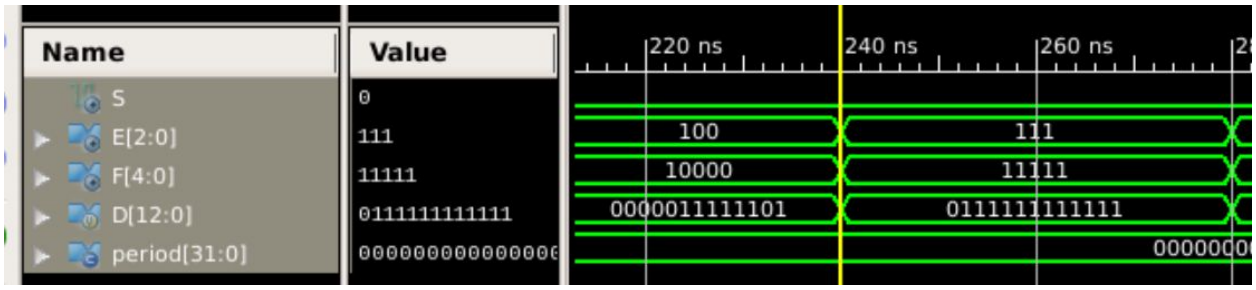
+506 is an insignificant number, but requires a regular rounding up. Should result in 0_100_11111 previous to rounding, and then 0_101_10000 after rounding.

Test case 6: D = 0_0000_1111_1101

+509 is very similar to +506, and should also result in 0_100_11111 previous to rounding, and 0_101_10000 after rounding. Checking whether mapping two distinct inputs to the same output works smoothly.

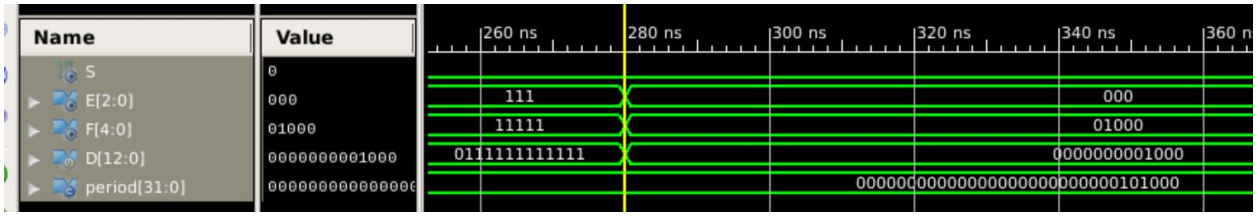
Test case 7: D = 0_1111_1111_1111

This is the largest possible positive two's complement number, and should result in the largest possible floating point number, 0_111_11111.



Test case 8: D = 0_0000_0000_0000_1000

+8 is an insignificant number, but could possibly pose problems since the significand is shorter than 5 bits, and no rounding 6th bit is present. Should result in 0_000_01000.



Map Report:

```
Design Summary
--------------
Number of errors:      0
Number of warnings:    1
Slice Logic Utilization:
  Number of Slice Registers:                    0 out of   4,800    0%
  Number of Slice LUTs:                        74 out of   2,400    3%
    Number used as logic:                      74 out of   2,400    3%
      Number using O6 output only:             59
      Number using O5 output only:             12
      Number using O5 and O6:                   3
      Number used as ROM:                       0
    Number used as Memory:                      0 out of   1,200    0%

Slice Logic Distribution:
  Number of occupied Slices:                   31 out of     600    5%
  Number of MUXCYs used:                       16 out of   1,200    1%
  Number of LUT Flip Flop pairs used:          74
    Number with an unused Flip Flop:           74 out of      74  100%
    Number with an unused LUT:                  0 out of      74    0%
    Number of fully used LUT-FF pairs:          0 out of      74    0%
    Number of slice register sites lost
      to control set restrictions:              0 out of   4,800    0%
```

```
Peak Memory Usage:  756 MB
Total REAL time to MAP completion:  5 secs
Total CPU time to MAP completion:   4 secs
```

```
Section 6 - IOB Properties
--------------------------

+------------------------------------------------------------------------------------------------------------------------------------------------+
| IOB Name                          | Type             | Direction | IO Standard          | Diff  | Drive    | Slew | Reg (s)      |
|                                   |                  |           |                      | Term  | Strength | Rate |              |
+------------------------------------------------------------------------------------------------------------------------------------------------+
| D<0>                              | IOB              | INPUT     | LVCMOS25             |       |          |      |              |
| D<1>                              | IOB              | INPUT     | LVCMOS25             |       |          |      |              |
| D<2>                              | IOB              | INPUT     | LVCMOS25             |       |          |      |              |
| D<3>                              | IOB              | INPUT     | LVCMOS25             |       |          |      |              |
| D<4>                              | IOB              | INPUT     | LVCMOS25             |       |          |      |              |
| D<5>                              | IOB              | INPUT     | LVCMOS25             |       |          |      |              |
| D<6>                              | IOB              | INPUT     | LVCMOS25             |       |          |      |              |
| D<7>                              | IOB              | INPUT     | LVCMOS25             |       |          |      |              |
| D<8>                              | IOB              | INPUT     | LVCMOS25             |       |          |      |              |
| D<9>                              | IOB              | INPUT     | LVCMOS25             |       |          |      |              |
| D<10>                             | IOB              | INPUT     | LVCMOS25             |       |          |      |              |
| D<11>                             | IOB              | INPUT     | LVCMOS25             |       |          |      |              |
| D<12>                             | IOB              | INPUT     | LVCMOS25             |       |          |      |              |
| E<0>                              | IOB              | OUTPUT    | LVCMOS25             |       | 12       | SLOW | OLATCH       |
| E<1>                              | IOB              | OUTPUT    | LVCMOS25             |       | 12       | SLOW | OLATCH       |
| E<2>                              | IOB              | OUTPUT    | LVCMOS25             |       | 12       | SLOW | OLATCH       |
| F<0>                              | IOB              | OUTPUT    | LVCMOS25             |       | 12       | SLOW |              |
| F<1>                              | IOB              | OUTPUT    | LVCMOS25             |       | 12       | SLOW |              |
| F<2>                              | IOB              | OUTPUT    | LVCMOS25             |       | 12       | SLOW |              |
| F<3>                              | IOB              | OUTPUT    | LVCMOS25             |       | 12       | SLOW |              |
| F<4>                              | IOB              | OUTPUT    | LVCMOS25             |       | 12       | SLOW |              |
| S                                 | IOB              | OUTPUT    | LVCMOS25             |       | 12       | SLOW |              |
+------------------------------------------------------------------------------------------------------------------------------------------------+
```

I was finally able to circumvent the licensing issue, and the map process completed with only one warning. No partitions were found, all constraints were met, and no registers were used. 16 MUXCYs were used, and 22 bonded IOBs were used, as well as 3 IOB latches.

Conclusion

The design of FPCVT includes the usage of 3 submodules in order to obtain the outputs S, E, and F, given D. Submodule mag_generator generates the signed magnitude number given the two's complement number, submodule count_leading_zeros generates the temporary exponent, temporary significand, and roundbit values given the signed magnitude number, and submodule rounding generates the final exponent and significand values given the roundbit value.

The most difficult part of this project was dealing with port connections, figuring out conditional statements for both wire and reg values, and learning how to use if and case statements using wire values. I learned that the easiest way to deal with these issues is to use wire values as inputs and outputs to the module, and then to use temporary reg values to fiddle with the values while inside those modules to use if and case statements. Another difficulty was counting the leading zeros in order to calculate the exponent and significand, which was solved by using casex statements.

I cannot think of ways to improve the overall lab experience. It was succinct and well-explained, and the TA's preeemptive advice and pointers were highly useful when implementing the difficult features of the lab.