Lab1: Floating Point Conversion

Introduction and Requirements

In this lab, the goal is to use the Xilinx ISE software to design and test a combinational circuit that converts a 13-bit linear encoding of an analog signal into a compounded 9-bit Floating Point (FP) Representation. What is required to do is mainly simulation. Focus on implementing a top-level module called FPCVT in verilog, and create a testbench to test the module using the simulation waveforms.

The inputs and outputs of FPCVT logic block should in the following table:

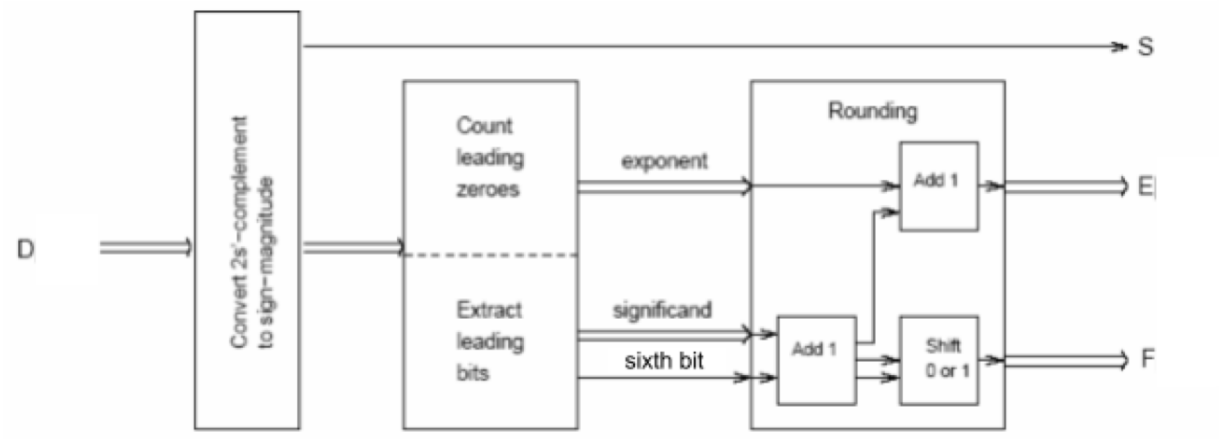
FPCVT Pin Descriptions	
D [12 : 0]	Input data in Two's Complement Representation. D0 is the Least Significant Bit (LSB). D12 is the Most Significant Bit (MSB).
S	Sign bit of the Floating Point Representation.
E [2 : 0]	3-Bit Exponent of the Floating Point Representation.
F [4 : 0]	5-Bit Significand of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

The 8-bit Byte signal value (V) in base-10 use the following formula:

$$V = (-1)^S \times F \times 2^E$$

Except for the design, testing the design is also required. Consider edge cases such as overflow, a testbench file is needed to check if the program can output the correct result. The key is to understand the feature of each blocks and design the bit manipulation stey by step.

Design Description



The overall design follows the given recommendation in the spec. `D` represents the 13-bits input net, and `S`, `E`, `F` compose the 8-bit output whose format is `[S_E_F]`

There are mainly three differnet blocks that need to design:

- Two's Complement to Sign Magnitude Representation
- Linear to Floating Point Conversion
- Rounding of the Floating Point Representation

13-bit Two's-complement to Sign-magnitude Representation

It is the first block, which comvers the 13-bit two's-complement input to sign-magnitude representation, it converts D to absolute value. The most significant number of the input is `S`, directly output is as `S` element. Note that:

- Nonnegative numbers (Sign bit 0) are unchanged.
- Negative numbers (Sign bit 1) are replaced by their absolute values. To negate a number, invert all bits and add 1 to this intermediate result.

In the module, I use bit operation `~` to invert all the bits. And I make a register `abs_value` to hold the transient absolute value to futher manipulate bits.

There is a edge case (D = -4096 - [1_0000_0000_0000]) mentioned in the spec. I set a if statement to handle it to make sure it will could convert to the correct result [1_1111_1111_1111].

Linear to Floating Point Conversion

The second block perforsm the basic linear to floating point conversion.

The exponent `E` is determine by the number of leading zeros of the linear encoding, as table below:

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

In the module, I use a serires of if statement to implemnt the above rules, by checking the number of leading zeros, assign corresponding value to Exponent `E`. Furthermore, assign `F` to store the 5 bits that following the last leading zeros. And use `sixth_bit` to store the the 6th bits following the last leadning zeros, which could determine whether round up or round down in the next block.

In the case of negtive number input, the first step is negating it, then check the number of leading zeros.

Rounding of the Floating Point Representation

The last block. Here require to round the linear encoding to the nearest floating point encoding. The significand consists of the 5 bits immeadiately following the last leading 0. The implement need to check the 6th bit (`sixth_bit` in the implement) following the last leading 0.

- `sixth_bit` is 0. Round the first 5 bits following leading zeros down, simply use it as `F` in output.
- `sixth_bit` is 1. Round the first 5 bits following leading zeros (`F`) up by adding 1.
    - If the significand overflows, i.e., `F` = [11111], shift the significant (`F`) right one bit and increase the Exponent (`E`) by 1.

Module FPCVT

The module FPCVT I created for this lab, it consists of the implementation of above three blocks. I use `//` comments to spearate them to improve readability.

Simulation Documentation

Test Case

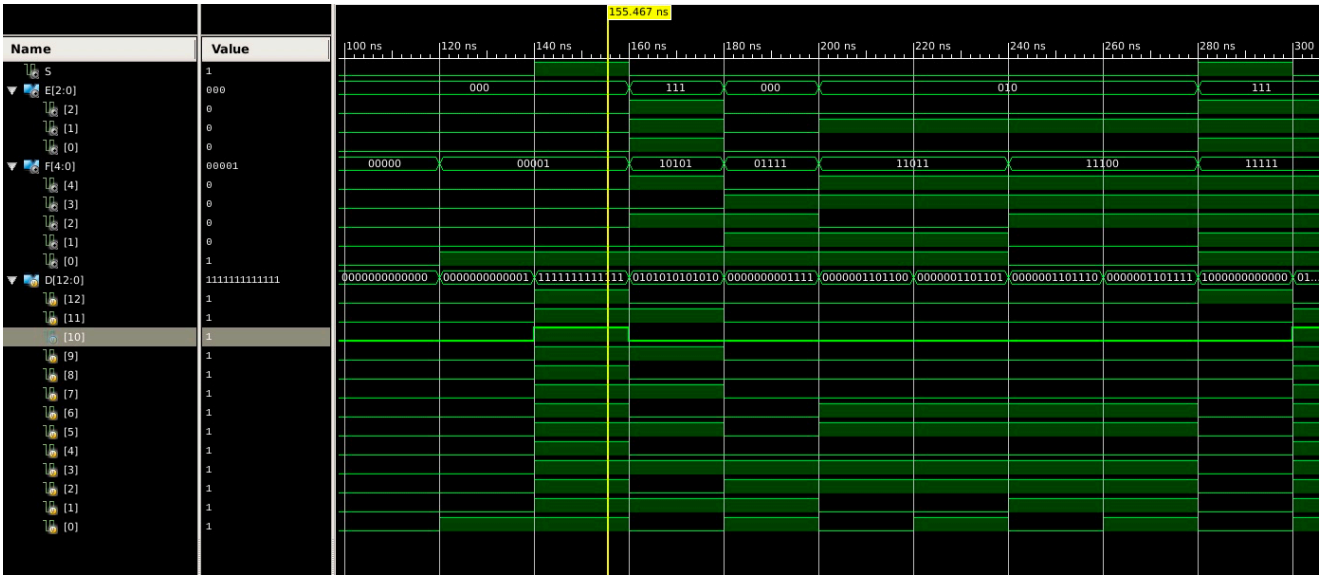
There are a few test cases used in the testbench file, some edge cases that are mentioned in the given spec are included. If the S, E, F output elements are the predicted result, the information would be shown as “passed” in the console by using `$display()` command. Otherwise, it shows “failed”. The predict results are obtained by manually calculation.

The test cases are as below:

Case #	Type of Cases	Linear Encoding - Input D[12:0]	Input Value	FP Representation - Output [S_E_F]	Output FP value
1	**General Cases**	0_0000_0000_0000	0	[0_000_00000]	0
2	-	0_0000_0000_0001	1	[0_000_00001]	1
3	-	1_1111_1111_1111	-1	[1_000_00001]	-1
4	- (leading zeros = 1 -> exp = 7)	0_1010_1010_1010	2730	[0_111_10101]	2730
5	- (leading zeros ≥ 8 -> exp = 0)	0_0000_0000_1111	15	[0_000_01111]	15
6	**Rounding Examples**	0_0000_0110_1100	108	[0_010_11011]	108
7	- (down)	0_0000_0110_1101	109	[0_010_11011]	108
8	- (up)	0_0000_0110_1110	110	[0_010_11100]	112
9	- (up)	0_0000_0110_1111	111	[0_010_11100]	112
10	Overflow from Two’s complement to sign magnitude (**most negative number case** in spec)	1_0000_0000_0000	-4096	[1_111_11111]	-3968
11	**Overflow of F & E** (rounding up)	0_1111_1111_1111	4095	[0_111_11111]	3968
12		1_0000_0000_0001	-4095	[1_111_11111]	-3968
13	Overflow of F (rounding up) - shift right	0_0000_1111_1101	253	[0_100_10000]	256
14	**Nonnegative** number in 2’s complement (spec)	0_0001_1010_0110	422	[0_100_11010]	416
15	**Negative number** in 2’s complement (spec)	1_1110_0101_1010	-422	[1_100_11010]	-416

Simulation Waveforms

A *testbench_UID.v* is created for testing. Switch to the simulation view and run “Simulation Behavioral Model”, the simulation waveforms output is display as below figure.



Test case results form console:

```
Test case#1 Passed.
Test case#2 Passed.
Test case#3 Passed.
Test case#4 Passed.
Test case#5 Passed.
Test case#6 Passed.
Test case#7 Passed.
Test case#8 Passed.
Test case#9 Passed.
Test case#10 Passed.
Test case#11 Passed.
Test case#12 Passed.
Test case#13 Passed.
Test case#14 Passed.
Test case#15 Passed.
ISim>
```

Conclusion

Design Summery

It is a great lab for getting know verliog language and FPGA design. Bit manipulation is a big part for this lab. We need to manually convert the result and test the different cases.

The below reports display the general design overview summary.

FPCVT Project Status (04/18/2021 - 11:47:26)			
Project File:	Project1.xise	**Parser Errors:**	No Errors
Module Name:	FPCVT	**Implementation State:**	Translated
Target Device:	xc6slx16-3csg324	**• Errors:**	No Errors
Product Version:	ISE 14.7	**• Warnings:**	No Warnings
Design Goal:	Balanced	**• Routing Results:**	
Design Strategy:	Xilinx Default (unlocked)	**• Timing Constraints:**	
Environment:	System Settings	**• Final Timing Score:**	

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice LUTs	65	9112	0%
Number of fully used LUT-FF pairs	0	65	0%
Number of bonded IOBs	22	232	9%

Detailed Reports					[-]
Report Name	**Status**	**Generated**	**Errors**	**Warnings**	**Infos**
Synthesis Report	Current	Sun Apr 18 11:28:57 2021	0	0	0
Translation Report	Current	Sun Apr 18 11:47:26 2021	0	0	0
Map Report	Out of Date	Sun Apr 18 11:47:14 2021	X 1 Error (0 new)	0	0
Place and Route Report	Out of Date	Sun Apr 18 09:13:05 2021	0	0	2 Infos (2 new)
Power Report					
Post-PAR Static Timing Report	Out of Date	Sun Apr 18 09:13:10 2021	0	0	4 Infos (4 new)
Bitgen Report					

Secondary Reports			[-]
Report Name	**Status**	**Generated**	
ISIM Simulator Log	Out of Date	Sun Apr 18 11:23:09 2021	

Date Generated: 04/18/2021 - 11:47:26

```
=========================================================================
*                            Design Summary                             *
=========================================================================

Top Level Output File Name         : FPCVT.ngc

Primitive and Black Box Usage:
------------------------------
# BELS                             : 93
#      GND                         : 1
#      INV                         : 12
#      LUT1                        : 1
#      LUT2                        : 4
#      LUT3                        : 5
#      LUT5                        : 13
#      LUT6                        : 30
#      MUXCY                       : 12
```

```
#      MUXF7                       : 1
#      VCC                         : 1
#      XORCY                       : 13
# IO Buffers                       : 22
#      IBUF                        : 13
#      OBUF                        : 9

Device utilization summary:
---------------------------

Selected Device : 6slx16csg324-3


Slice Logic Utilization:
 Number of Slice LUTs:                   65  out of   9112     0%
    Number used as Logic:                65  out of   9112     0%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:     65
   Number with an unused Flip Flop:      65  out of     65   100%
   Number with an unused LUT:             0  out of     65     0%
   Number of fully used LUT-FF pairs:     0  out of     65     0%
   Number of unique control sets:         0

IO Utilization:
 Number of IOs:                          22
 Number of bonded IOBs:                  22  out of    232     9%

Specific Feature Utilization:

---------------------------
Partition Resource Summary:
---------------------------

  No Partitions were found in this design.
```

Difficulties

Since unfamily with the ISE software in Virtual box, I met a difficulty that I cannot access the simulation view. It is missing from the software.

I uninstalled everything and redownload the ISE zip file and VM. Strictly follow the official tutorial in their website. Finally, I am able to do the simulation.

In the aspect of the coding, it is fairly easy but it does requires time to know the verliog language and how to design the bit manipulation. Furthermore, we need to consider some edge cases such overflow, and we need to check if the program can handle it. n

Reference

Beside the tutorials provided in the course, I also use the following website to help me understand and design the lab:

- https://www.mouser.cn/datasheet/2/903/ug973-vivado-release-notes-install-license-1596316.pdf
- https://www.chipverify.com/verilog/verilog-display-tasks