Stephanie Doan (604981556)
Rohan Surve
CS M152A Lab 6
24 January 2021

Lab 1: Floating Point Conversion

Introduction

In this first lab, we aim to build a combinational circuit to convert 13-bit numbers from two's complement representation to 9-bit floating point representation. We design and test our program in Verilog using Xilinx ISE, with which we familiarize ourselves as an important tool in circuit design.

The FPCVT module is to take as input a linearly encoded binary number in two's complement representation, therefore a 13-bit binary from -4096 to 4095. The module is to translate this value to floating-point representation that consists of a 1-bit sign S, 3-bit exponent E, and 5-bit significand F for a total of 9 bits. The final value is given by the equation:

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

However, this process of mapping linear encoding to floating point representation, called compression, suffers from some inaccuracies due to rounding. Different values in two's complement can map to the same floating point encoding when rounding the significand up or down according to the sixth bit after the last leading zero, as well as ignoring bits after it due to limited capacity with nine bits. We also handle cases of overflow accordingly, as later explained with examples in our test cases.

As explored in the Implementation section, we go about constructing our FPCVT module in three main blocks/steps:

1. Extract the sign bit and convert two's complement to signed magnitude representation.
2. Extract value of the exponent by counting leading zeros in the signed magnitude value binary and extract the significand.
3. Perform necessary rounding on the significand based on the bit following the five bits of F.

Implementation

Our module is sectioned into three main blocks of logic:

I. Two's complement to sign magnitude

The first step consists of extracting the most significant bit (MSB) of the input data, since in two's complement representation, an MSB of 0 signifies a non-negative number and an MSB of 1 means a negative number. In the code, this is done by doing a continuous assignment inside the main always block to output register *S* the value at input *D[12]*.

Afterwards, I convert two's complement to sign magnitude by doing another continuous assignment to a register I declare called *sign_magnitude[12:0]*. If the sign bit is 0, we just write the entire content of the non-negative input *D[12:0]* to *sign_magnitude*. If the sign bit is 1, we flip all the bits of *D[12:0]* and add 1 before writing to *sign_magnitude*. We also handle the special case of *D* being 1000000000000 in binary, the most negative number-4096, which should instead become 1111111111111 in sign magnitude, which is 4095, the closest positive number to the absolute value -4096, as this number's negativeness is already noted by the sign bit.

II. Linear to floating point

The goal of this block is to correctly populate the exponent output register *E[2:0]* and significand output register *F[4:0]*, as well as note the round bit that dictates whether to round the significand up or down. To extract these values, we must count the number of leading zeros in the *sign_magnitude[12:0],* as it tells us the exponent value according to the table below, as well as which indices of the *sign_magnitude* register to assign to *F[4:0]*, which are the five indices following the last leading zeroes in *sign_magnitude*.

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

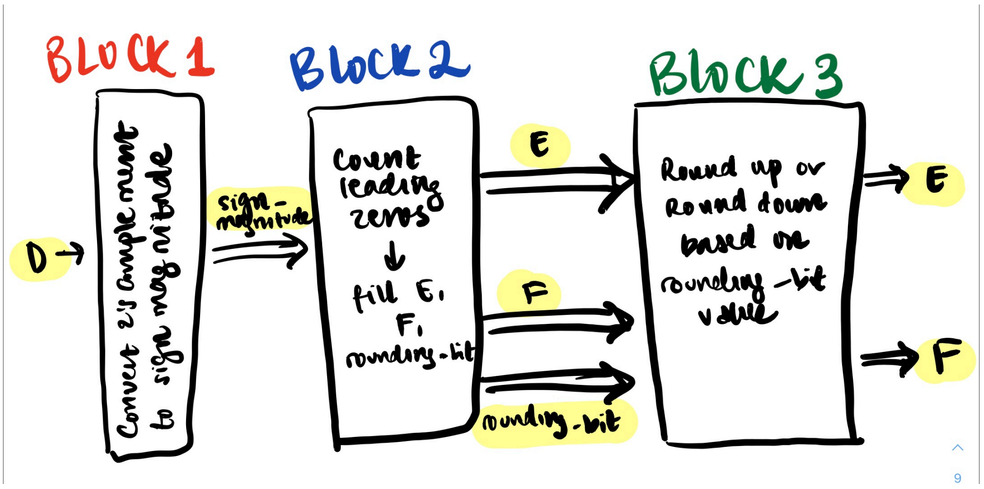
In the code, I first declared a register *round_bit* and to hold the sixth bit after the last leading zero, before creating a large if/else block (that can also be made a switch statement, but they synthesized roughly the same circuit) with an if-statement handling each number of leading zeros. In each else statement, *E* is assigned a hardcoded 3-bit binary value; *F* is read from the next 5 spots in *sign_magnitude*, and *round_bit* is read from the subsequent index.

III. Rounding of floating point

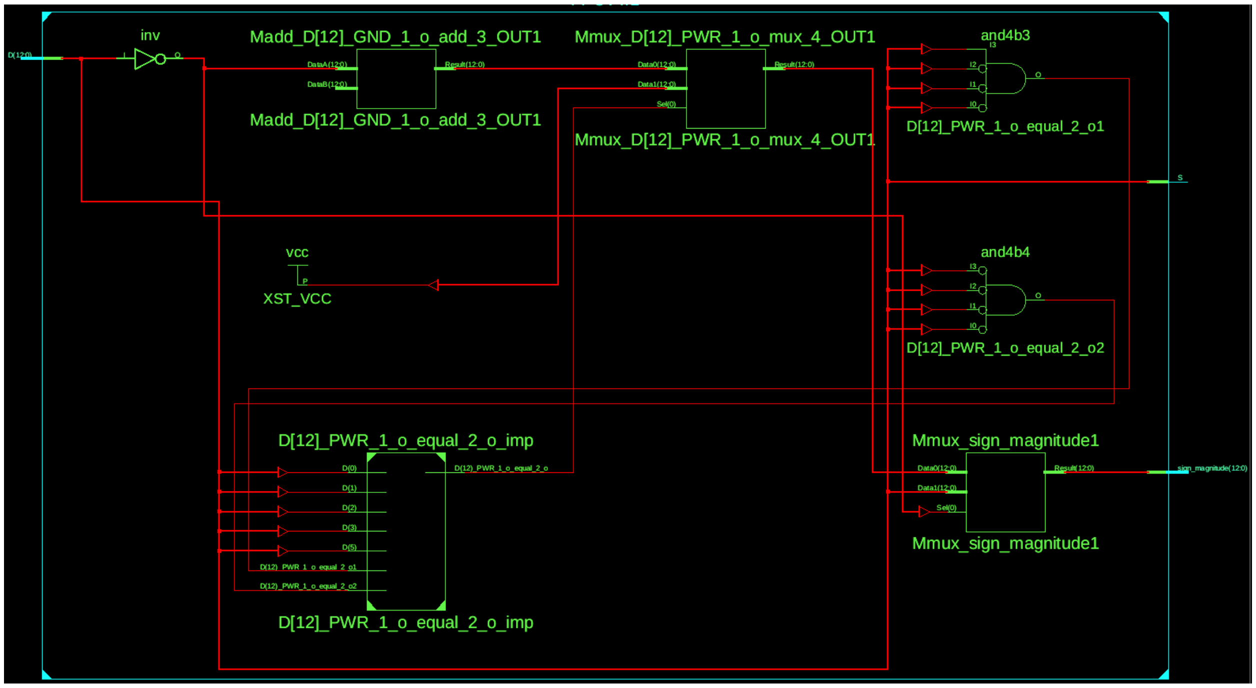
Finally the significand value held in *F* is changed according to the value held in *round_bit*. We round *F* up by adding 1 to it if *round_bit* is 1 and do keep it the same value if *round_bit* is 0. We must also handle cases of overflow in the former case, specifically when *F* holds *11111* in binary and incrementing it will lead to overflow. In this case, *F* becomes *10000* and *E* is incremented by 1. However, if *E* is already *111*, it will also overflow. In this case, *F* stays as *11111* and *E* stays as *111*.

Schematics

It is important to visualize our implementation and make sense of the schematics made in synthesis, whose full version is included at the end of the report. The three main logical blocks are outlined in the hand-drawn diagram below:

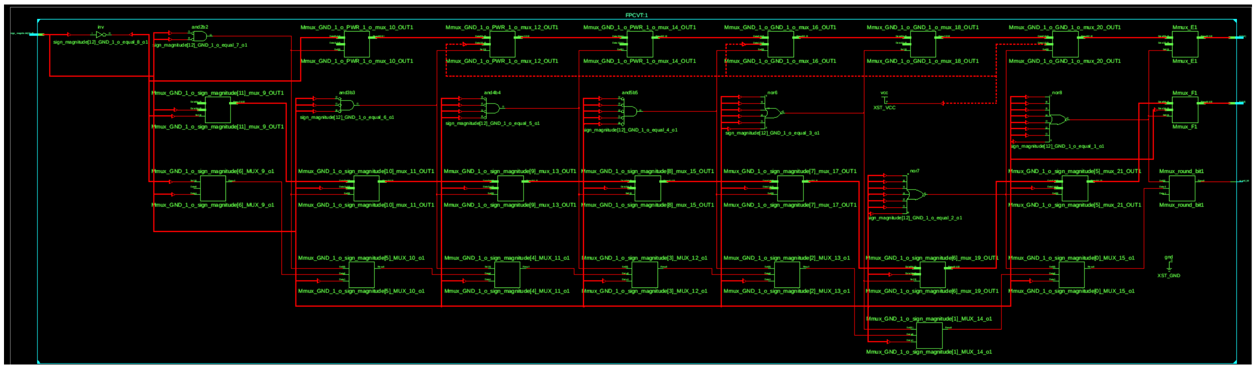


Now, to understand the workings of these individual blocks, I synthesized the RTL schematic for each block by commenting out the other blocks and focusing on the manipulated inputs and outputs. The schematic below isolates the first block, first extracting the sign bit as seen by the wire from input *D[12:0]* to the *S* output. The top left triangle inverts all the bits of *D*, while adder to its immediate right adds 1 to this inverted value, to get *~D[12:0] + 1'b1* as seen in the code. The two multiplexers named "Mmux" are to assign to *sign_magnitude* different values, one based on if *D[12]* is *1'b0* or *1'b1*, one based on if *D[12:0]* is *1'b1000000000000*, checked by the other structures in the schematic. The output of this block is *S* and *sign_magnitude[12:0]*.



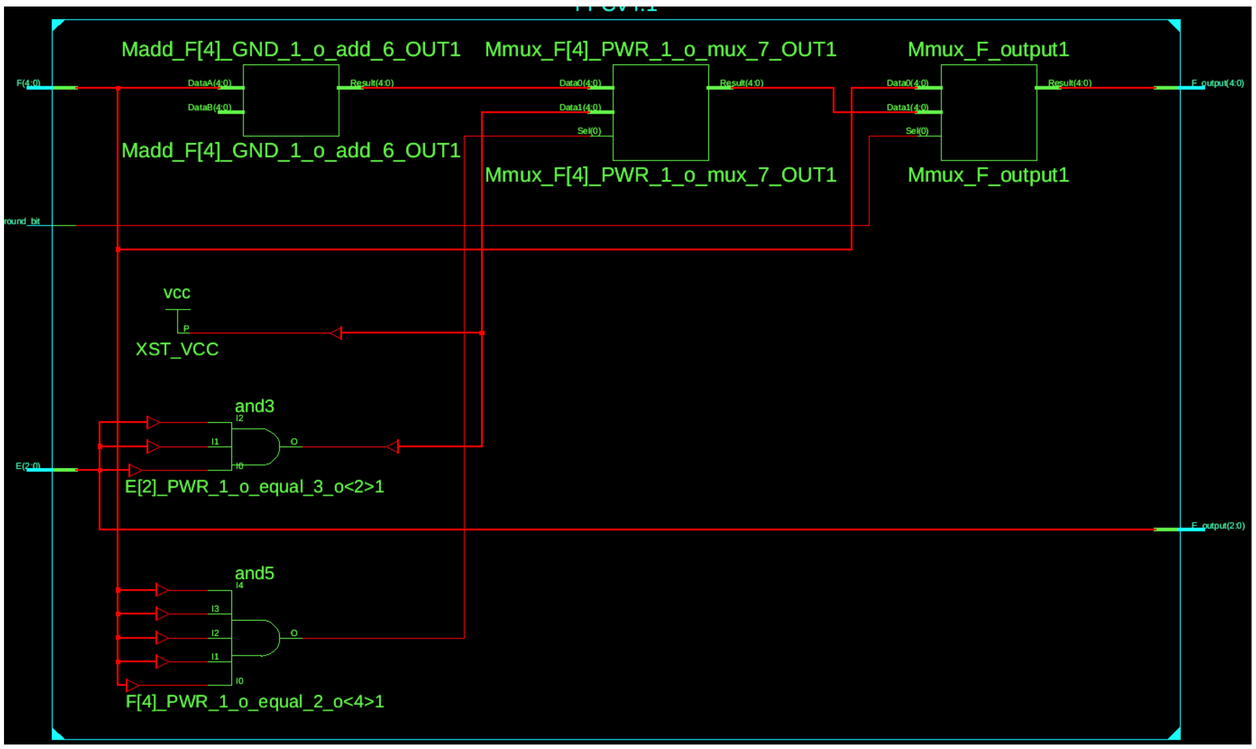
Block 1: Two's complement to sign magnitude

Next, we isolate the second block which counts leading zeroes in the *sign_magnitude* to get the values for *E[2:0]* and *F[4:0]* and *round_bit*. Since I implemented this “counting" with many if/else statements checking individual bits of *sign_magnitude*, as we can see with the many AND gates checking the bit values, and the multiplexers choosing values for each output based on the outputs of these AND gates.



Block 2: Linear to floating point

Finally, we isolate the third block by making *E[2:0]*, *F[4:0]*, and *round_bit* inputs and making *E_output[2:0]* and *F_output[4:0]* to hold the final values. As seen in the schematic below, the final rightmost multiplexer chooses based on the value of *round_bit* whether to round up or round down. In the case of round up, the edge cases are handled by manually checking the values of *F[4:0]* and *E[4:0]* with all the AND gates, whose outputs are used as selector to the other multiplexer. The adder does the *E[2:0] + 1’b1* in case of the significand overflow.



Block 3: Rounding of floating point

Tests

To test my FPCVT module, I created a testbench that changes the value of input D in 10 time unit intervals. Inputs, expected outputs, and the reasons for each test case are outlined in the table below. In general, I covered these cases in my test suite:

- General cases checking basic functionality to correctly populate the S, E, and F outputs without overflow, including D = 0 and D = -1, consisting of all 0's and all 1's respectively
- Edge cases with overflow
  - Specific case of 1000000000000 that overflows when converting to sign magnitude, covered with an if/else statement in the code
  - Overflow of F when rounding up, so F is handled accordingly and E is incremented by 1
  - Overflow of F and E when rounding up, so E stays 111 and F stays 11111
- Two round up cases provided in the spec: 108 and 109
- Two round down cases provided in the spec: 110 and 111

	Linear encoding D[12:0]	**Linear encoding value**	**Floating point encoding [S E F]**	**FP value**
Round down significand	[0000001101100]	108	[0 010 11011]	108
	[0000001101101]	109	[0 010 11011]	108
Round up significand	[0000001101110]	110	[0 010 11100]	112
	[0000001101111]	111	[0 101 11100]	112
[edge] Overflow from 2s complement to sign magnitude	[1000000000000]	-4096	[1 111 11111]	-3968
[edge] Overflow of F and E when round up	[0111111111111]	4095	[0 111 11111]	3968
	[0111111000000]	4095	[0 111 11111]	3968
[edge] Overflow of F when round up	[0000011111101]	253	[0 100 10000]	256
General cases	[0000000000000]	0	[0 000 00000]	0
	[1111111111111]	-1	[1 000 00001]	-1
	[0101010101010]	2730	[0 111 10101]	2730
	[1111110111000]	-72	[1 010 10010]	-72
	[0000001100100]	100	[1 010 11001]	100
	[1011010001011]	-2421	[1 111 10011]	-2421
	[0000000001111]	15	[0 000 01111]	15

To run the tests, I created the *testbench_UID.v* file that also logs to console if each test case passed or failed with the *$display* command. I ran the simulation with *Simulate Behavioral Model*, whose simulation output is displayed in Figure 3 with each change in D's signal marking a new test case being run.

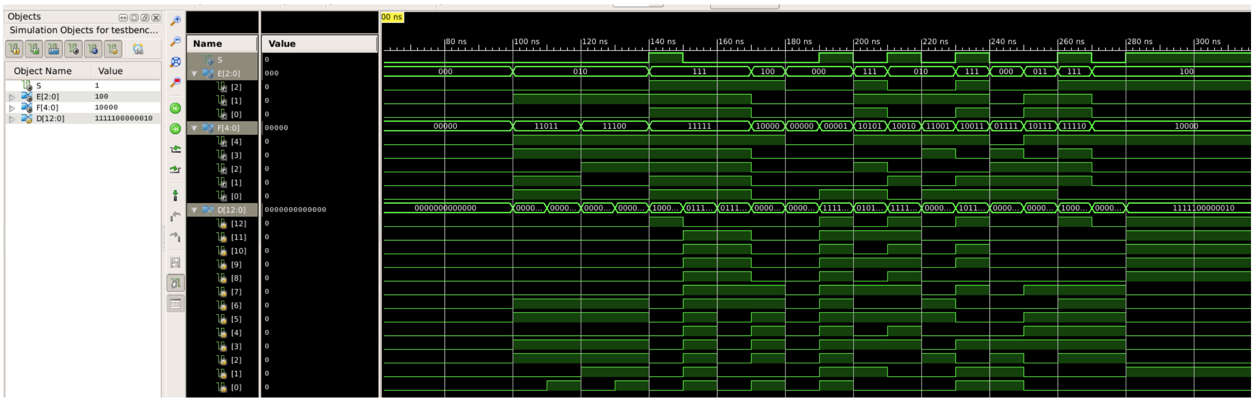
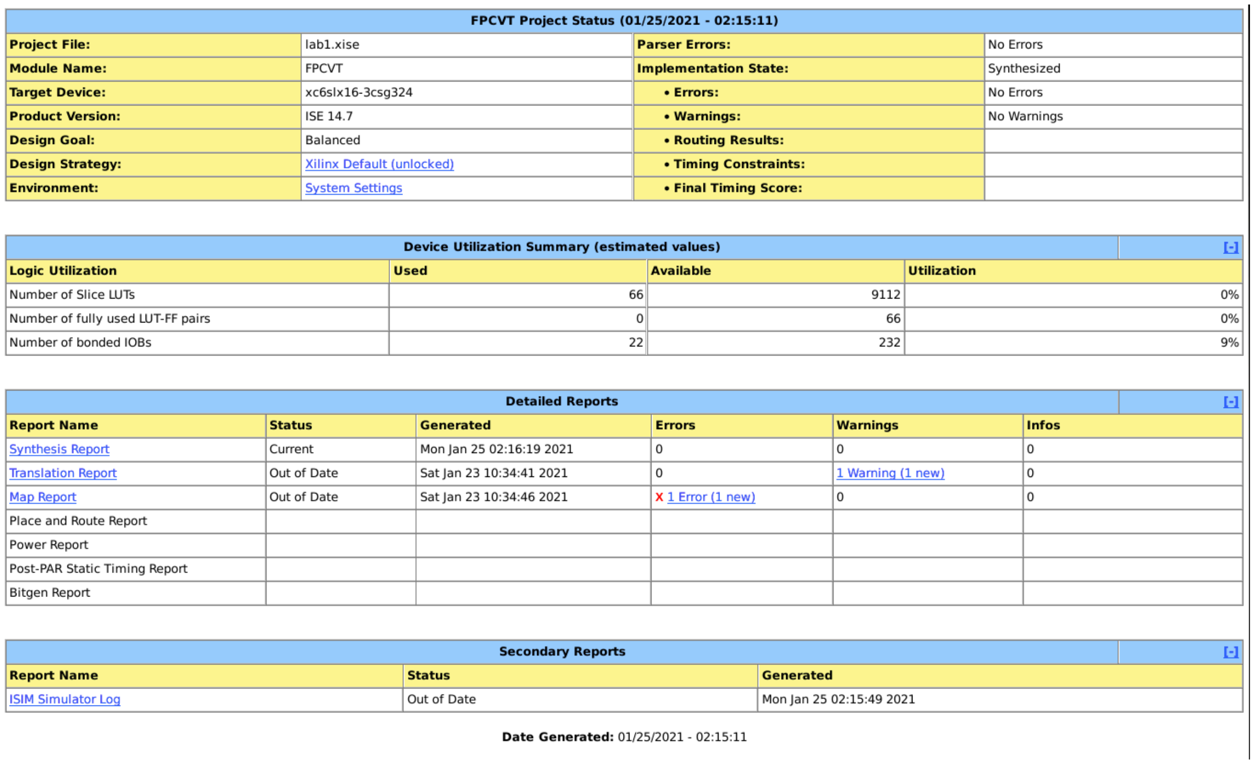


Figure 3: Simulation output on test suite, with each output register expanded into its bits

The output logged to console is:

```
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
Passed: linear input 108
Passed: linear input 109
Passed: linear input 110
Passed: linear input 111
Passed: linear input -4096
Passed: linear input 4095
Passed: linear input 4032
Passed: linear input 253
Passed: linear input 0
Passed: linear input -1
Passed: linear input 2730
Passed: linear input -72
Passed: linear input 100
Passed: linear input -2421
Passed: linear input 15
Passed: linear input 184
Passed: linear input -3844
Passed: linear input 252
Passed: linear input -254
ISim>
```

Design Overview Summary Report



FPCVT Project Status (01/25/2021 - 02:15:11)			
Project File:	lab1.xise	**Parser Errors:**	No Errors
Module Name:	FPCVT	**Implementation State:**	Synthesized
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	66	9112	0%
Number of fully used LUT-FF pairs	0	66	0%
Number of bonded IOBs	22	232	9%

Detailed Reports [-]

Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report	Current	Mon Jan 25 02:16:19 2021	0	0	0
Translation Report	Out of Date	Sat Jan 23 10:34:41 2021	0	1 Warning (1 new)	0
Map Report	Out of Date	Sat Jan 23 10:34:46 2021	X 1 Error (1 new)	0	0
Place and Route Report					
Power Report					
Post-PAR Static Timing Report					
Bitgen Report					

Secondary Reports [-]

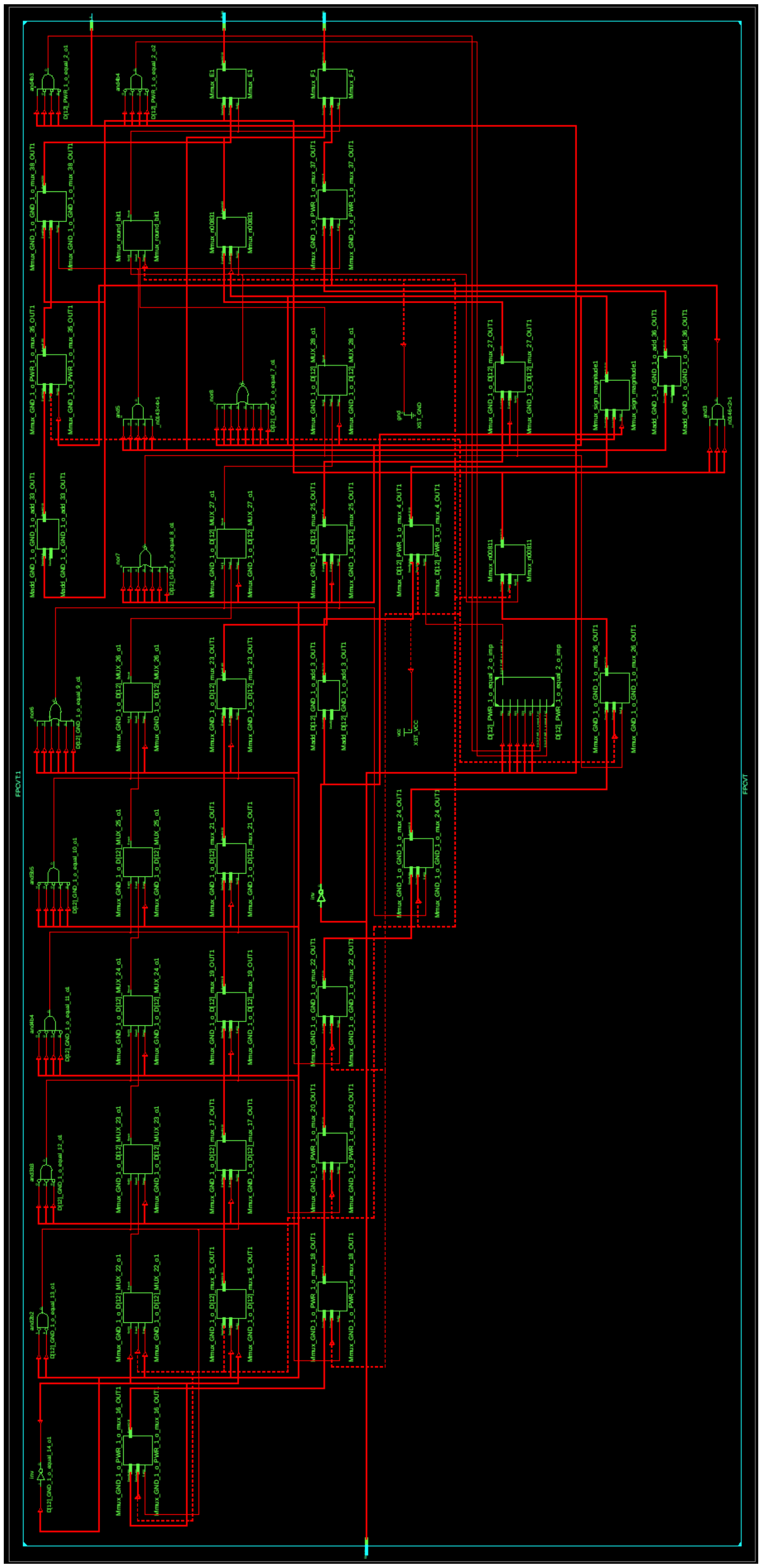
Report Name	Status	Generated
ISIM Simulator Log	Out of Date	Mon Jan 25 02:15:49 2021

Date Generated: 01/25/2021 - 02:15:11

Figure 4: Design overview summary report

Figure 4 displays the generated design overview summary. Briefly looking the report, we see that there are no errors or warnings, with 66 lookup tables, no LUT-FF pairs, and 22 input-output blocks used, which is leaves most of these resources unused. A complete version is included at the very end of this report.

RTL Schematic for full FPCVT module



FPCVT Project Status (01/25/2021 - 03:42:53)			
Project File:	lab1.xise	**Parser Errors:**	No Errors
Module Name:	FPCVT	**Implementation State:**	Programming File Generated
Target Device:	xc6slx16-3csg324	• **Errors:**	No Errors
Product Version:	ISE 14.7	• **Warnings:**	No Warnings
Design Goal:	Balanced	• **Routing Results:**	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	• **Timing Constraints:**	
Environment:	System Settings	• **Final Timing Score:**	0 (Timing Report)

Device Utilization Summary				**[-]**
Slice Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Registers	0	18,224	0%	
Number of Slice LUTs	60	9,112	1%	
Number used as logic	60	9,112	1%	
Number using O6 output only	42			
Number using O5 output only	12			
Number using O5 and O6	6			
Number used as ROM	0			
Number used as Memory	0	2,176	0%	
Number of occupied Slices	23	2,278	1%	
Number of MUXCYs used	16	4,556	1%	
Number of LUT Flip Flop pairs used	60			
Number with an unused Flip Flop	60	60	100%	

Number with an unused LUT	0	60	0%	
Number of fully used LUT-FF pairs	0	60	0%	
Number of slice register sites lost to control set restrictions	0	18,224	0%	
Number of bonded IOBs	22	232	9%	
Number of RAMB16BWERs	0	32	0%	
Number of RAMB8BWERs	0	64	0%	
Number of BUFIO2/BUFIO2_2CLKs	0	32	0%	
Number of BUFIO2FB/BUFIO2FB_2CLKs	0	32	0%	
Number of BUFG/BUFGMUXs	0	16	0%	
Number of DCM/DCM_CLKGENs	0	4	0%	
Number of ILOGIC2/ISERDES2s	0	248	0%	
Number of IODELAY2/IODRP2 /IODRP2_MCBs	0	248	0%	
Number of OLOGIC2/OSERDES2s	0	248	0%	
Number of BSCANs	0	4	0%	
Number of BUFHs	0	128	0%	
Number of BUFPLLs	0	8	0%	
Number of BUFPLL_MCBs	0	4	0%	
Number of DSP48A1s	0	32	0%	
Number of ICAPs	0	1	0%	
Number of MCBs	0	2	0%	
Number of PCILOGICSEs	0	2	0%	
Number of PLL_ADVs	0	2	0%	
Number of PMVs	0	1	0%	
Number of STARTUPs	0	1	0%	
Number of SUSPEND_SYNCs	0	1	0%	
Average Fanout of Non-Clock Nets	3.57			

Performance Summary			**[-]**
Final Timing Score:	0 (Setup: 0, Hold: 0)	**Pinout Data:**	Pinout Report
Routing Results:	All Signals Completely Routed	**Clock Data:**	Clock Report
Timing Constraints:			

Detailed Reports					**[-]**
Report Name	**Status**	**Generated**	**Errors**	**Warnings**	**Infos**
Synthesis Report	Current	Mon Jan 25 03:50:12 2021	0	0	0
Translation Report	Current	Mon Jan 25 03:50:23 2021	0	0	0
Map Report	Current	Mon Jan 25 03:50:39 2021	0	0	6 Infos (6 new)
Place and Route Report	Current	Mon Jan 25 03:51:05 2021	0	0	2 Infos (2 new)
Power Report					
Post-PAR Static Timing Report	Current	Mon Jan 25 03:51:16 2021	0	0	4 Infos (4 new)
Bitgen Report	Current	Mon Jan 25 03:51:34 2021	0	0	0

Secondary Reports		**[-]**
Report Name	**Status**	**Generated**
ISIM Simulator Log	Out of Date	Mon Jan 25 02:15:49 2021
WebTalk Report	Current	Mon Jan 25 03:51:36 2021
WebTalk Log File	Current	Mon Jan 25 03:51:37 2021

Date Generated: 01/25/2021 - 03:53:55