Senior Design Final Report

Team Overbuilt and Underpaid

Hsiang-Yi Chung

March 17, 2016

Introduction:

Our team chose the second option of the quarter 2 Project: Radar Sensor for UAV's. My portion of the project was to implement the digital signal processing modules for a FPGA.

The following are the modules involved in the design:

- Down sampling Module
- Windowing Module
- FFT Module
- Serial Communication
- PLL-FPGA Interface

Discussions:

Figure 1 shows the block diagram for the digital signal processing. The following discusses what each module does.

Figure 1: the above figure shows the block diagram for the signal processing.

ADC: the ADC used in the design has a 10 Mhz sampling rate with 12-bit resolution.

Down Sampling:

Since the signals coming out of the mixer has a maximum frequency of 229 kHz ideally,

$$f_{max} = \frac{BW}{T} * \frac{2R_{max}}{C} = \frac{160\ MHz}{0.7\ ms} * \frac{2 * 150m}{3 * 10^8 \frac{m}{s}} = 229\ kHz$$

the 10 MHz ADC is oversampling the input signals. The problem with oversampling is that it would increase the sample size of FFT, which will increase the latency of the FFT module. In order to reduce the sample size of FFT, down sampling by a factor of 8 was used to reduce the sampling rate to

$$f_s = \frac{10\ MHz}{8} = 1.25\ MHz$$

This new sampling frequency has a Nyquist frequency of 625 kHz, which is higher than the f_{max}. To avoid aliasing problem, an analog low pass filter before the ADC stage was used to filter out frequencies higher than Nyquist frequency.

Windowing:

In order to reduce the side lobes of the frequency spectrum, the input signal needs to be multiplied by a window function. Figure 2 shows the Matlab generated Hanning window used in the design. To avoid performing floating point operations on the FPGA, a scaled version of the window was used (Figure 3). The scaled version of the window was obtained by multiplying each value by 128 and rounding to the nearest integer.

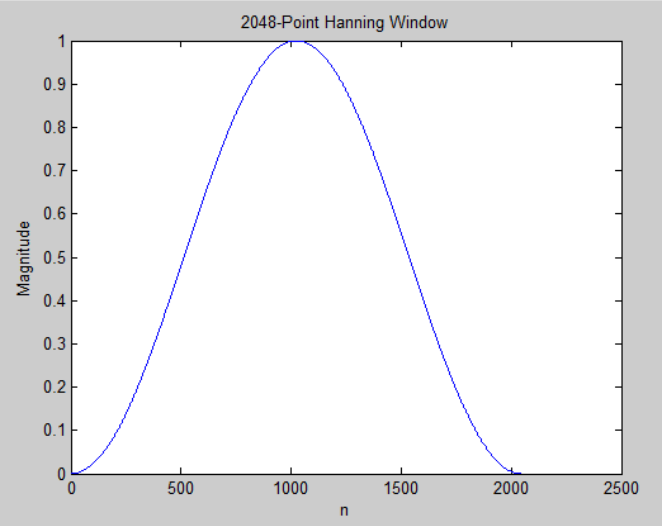

Figure 2: Matlab generated 2048-Point Hanning Window

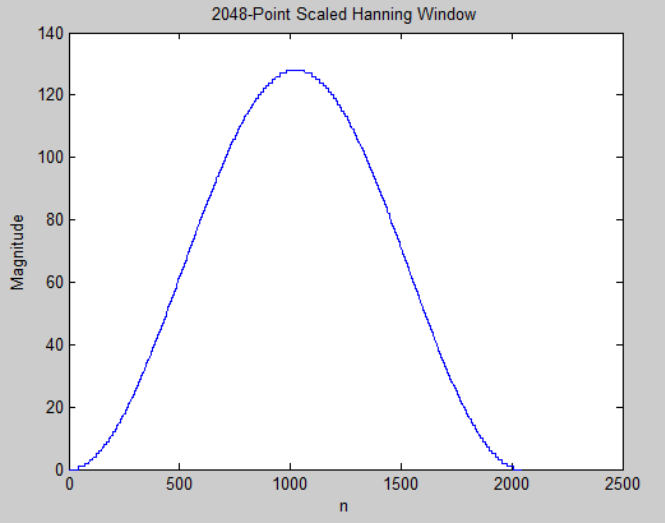

Figure 3: Matlab generated 2048-Point Scaled Hanning Window

The problem with the scaled version of the window is that when a signal passes through this filter, there will be a gain of 128. Thus, after multiplying the input signal by the window, the module will divide the product by 128.

FFT:

The FFT module was obtained from the *Spiral Software/Hardware Generation for DPS Algorithms.*

The following were the inputs to the generator:

- Transform Size = 2048
- Direction = forward
- Data Type = 12 bit fixed point, scaled
- Architecture = iterative reuse
- Radix = 2
- Streaming width = 2
- Data Ordering = natural input / natural output
- BRAM budget = 1000
- Permutation Method: JACM'09 [3]

Magnitude:

The FFT module outputs in complex number format, so this module was necessary to compute the magnitude. It is not practical to find the magnitude by using square and square root operations on a FPGA. Thus, the magnitude was estimated by using this formula:

$$\text{Magnitude} = \max(|I|,|Q|) + 1/4*\min(|I|,|Q|) - 1/16*\max(|I|,|Q|)$$

This estimation has a maximum error of 6.25%, but it only requires two adders and two arithmetic shift operations.

Serial:

The serial communication used for the design was a custom communication that has three outputs: clock, data, start_of_output. I was planning to use the Quartus SPI IP core, but it requires Avalon interface, which could make the design more complicated. Thus, this 3-wire interface was used in the design. Figure 4 shows the timing diagram for the communication.

- Clock: the clock rate was set to 625 kHz.
- Start_of_output: when it is asserted high, the data starts to output in the next clock cycle.
- Data: it outputs 2048 12-bit magnitude spectrum values. It starts from outputting the MSB to the LSB of the first magnitude value, and then the MSB to the LSB of the second magnitude value… and so on.

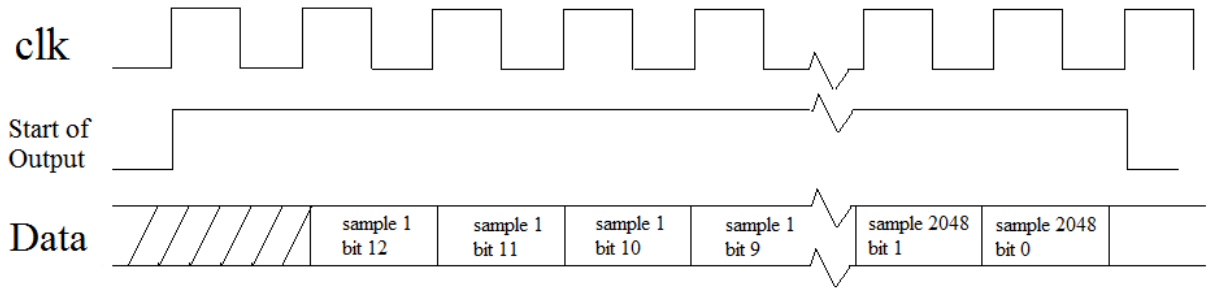

Figure 4: the above figure is the timing diagram for the serial output.

PLL-FPGA Interface:

The PLL used in the radar is ADF4158. The purpose of this module was to initialize the PLL to the desired state. It was set so that along with VCO, it will output continuous triangular waveform with frequency range from 7.92 GHz to 8.08 GHz.

Results:

In order to test the validity of the entire system without physical hardware, simulation using ModelSim was used to verify the system. The way I tested was making the testbench to read in a Matlab generated input signal, simulate the system, and write the output to a text file. Then I use Matlab to graph the spectrum values in the text file and compare it to the expected spectrum.

The input signals used to test the entire system were the following:

1. Sinusoidal signal with single frequency of 50 kHz sampled at 10 MHz
2. Sinusoidal signal with two frequencies of 54 kHz and 96 kHz sampled at 10 MHz

Sinusoidal signal with single frequency of 50 kHz:

Figure 5 and 6 show the Matlab generated input sinusoidal signal with frequency component of 50 kHz (top) and its corresponding Matlab calculated magnitude spectrum (bottom). In addition, Figure 7 and 8 show the magnitude spectrum outputted from the Verilog radar testbench. The system is fairly accurate that the peak of the spectrum is about 51 kHz, which is about 2% error. This error comes from the magnitude estimation that has a maximum error of 6.25%.

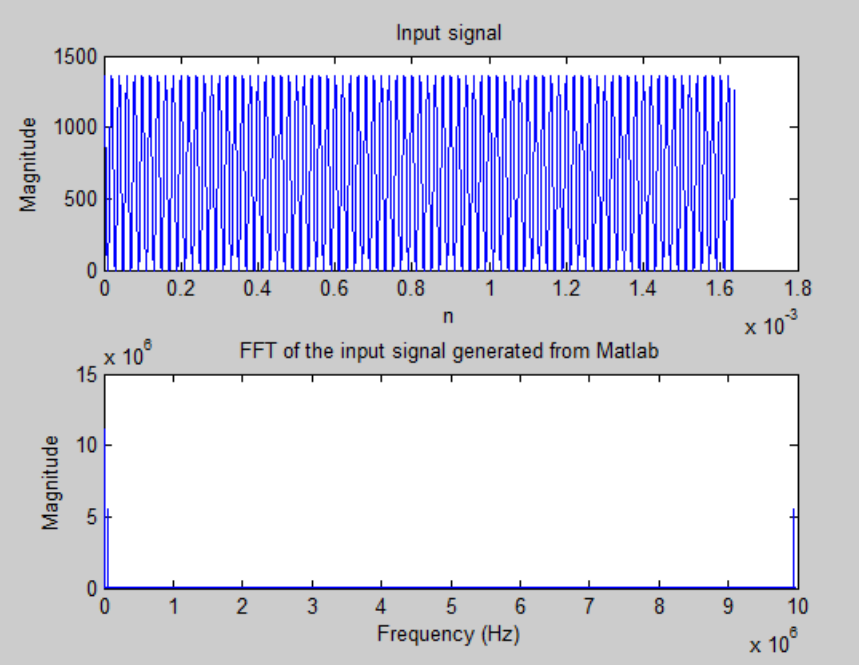

Figure 5: the above figure shows the Matlab generated input sinusoidal signal with frequency components 50 kHz and its frequency spectrum.

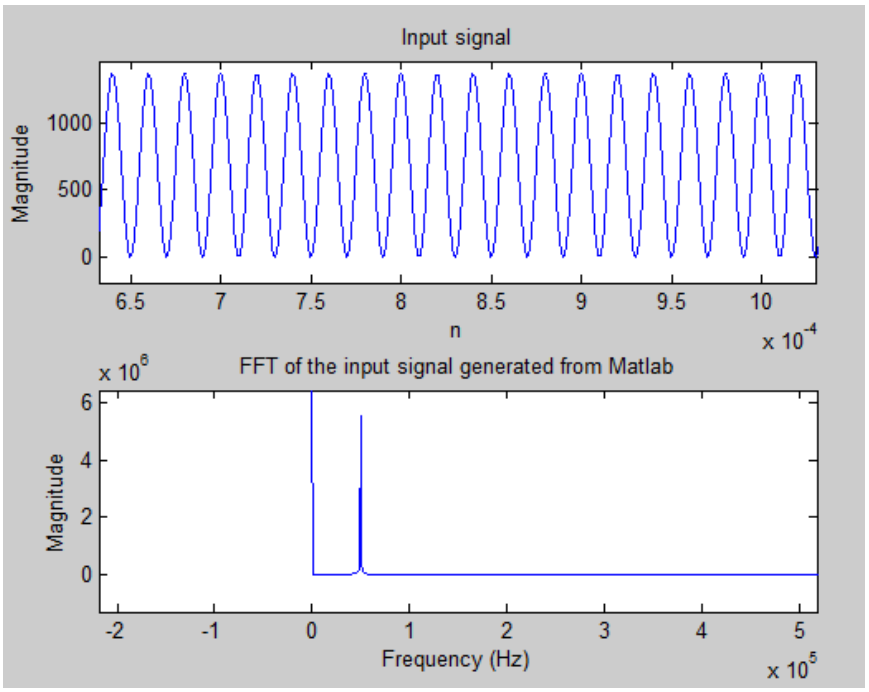

Figure 6: the above figure shows the zoomed in version of Figure 5.

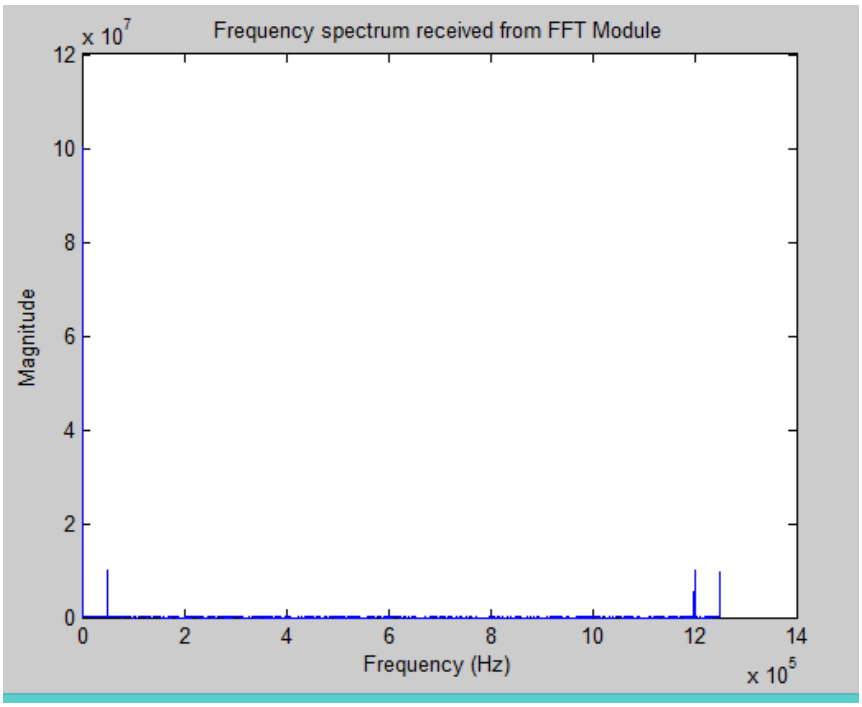

Figure 7: the above figure shows the magnitude spectrum outputted from the Verilog radar testbench (with input of 50 kHz sinusoidal).

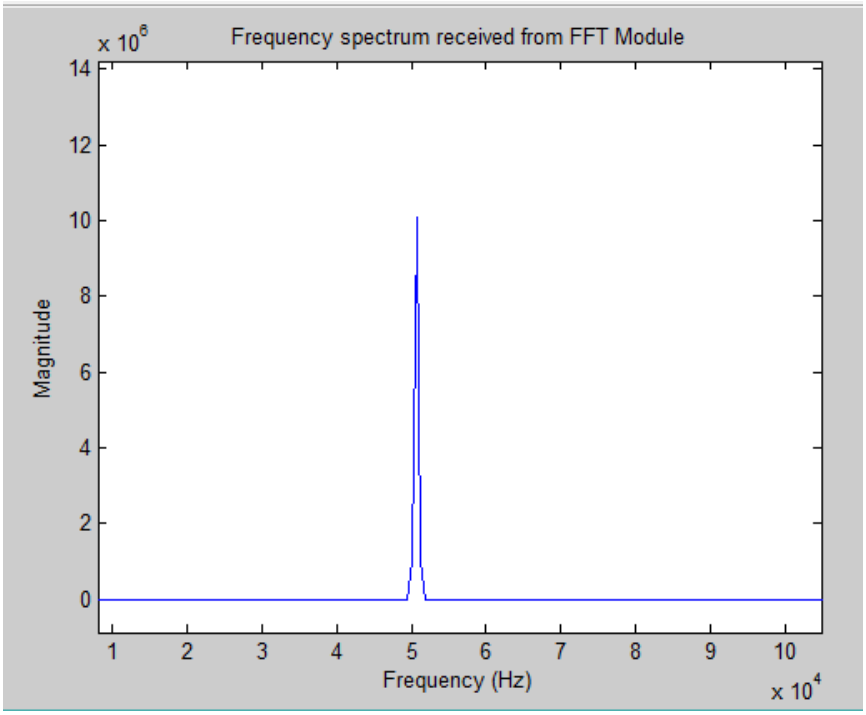

Figure 8: the above figure shows the zoomed in version of Figure 7.

Sinusoidal signal with two frequencies of 54 kHz and 96 kHz:

Figure 9 and 10 show the Matlab generated input sinusoidal signal with frequency components 54 kHz and 96 kHz (top) and its corresponding Matlab calculated magnitude spectrum (bottom). Figure 11 and 12 show the magnitude spectrum outputted from the Verilog radar testbench. The system have peaks at 54.5 kHz and 9.64 kHz (1% and 0.4% error respectively).

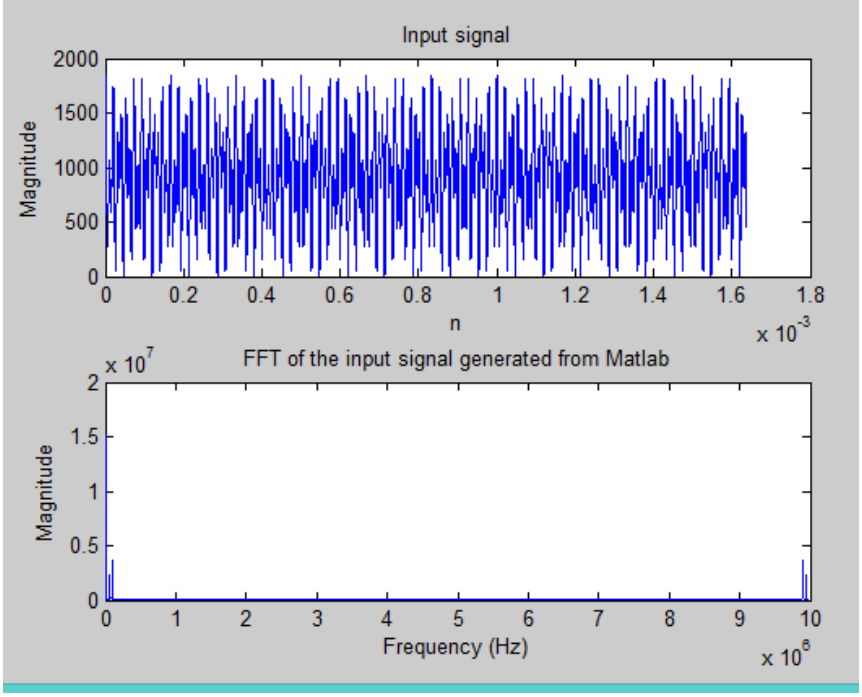

Figure 9: the above figure shows the Matlab generated input sinusoidal signal with frequency components 54 kHz and 96 kHz and its frequency spectrum.

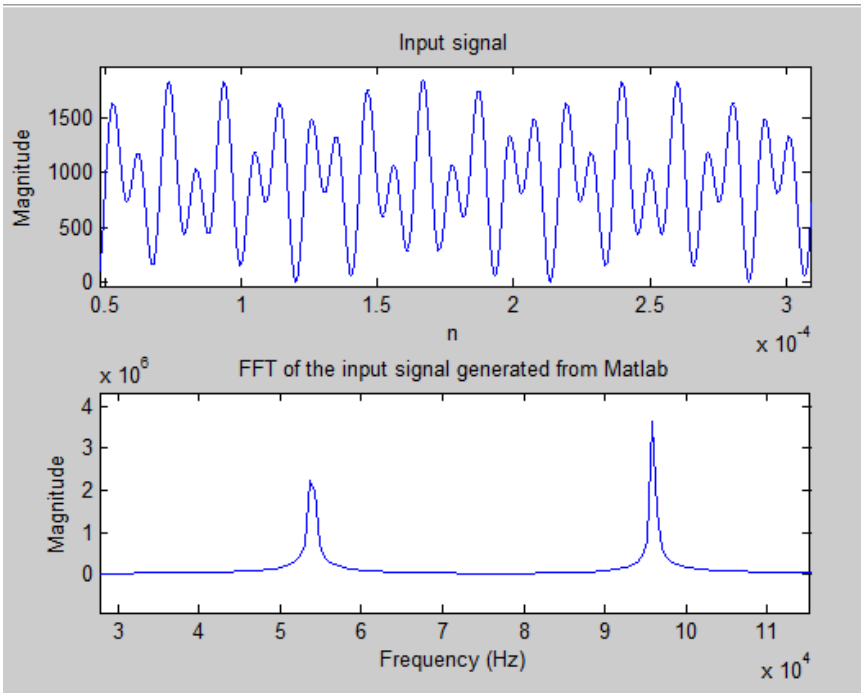

Figure 10: the above figure shows the zoomed in version of Figure 9.

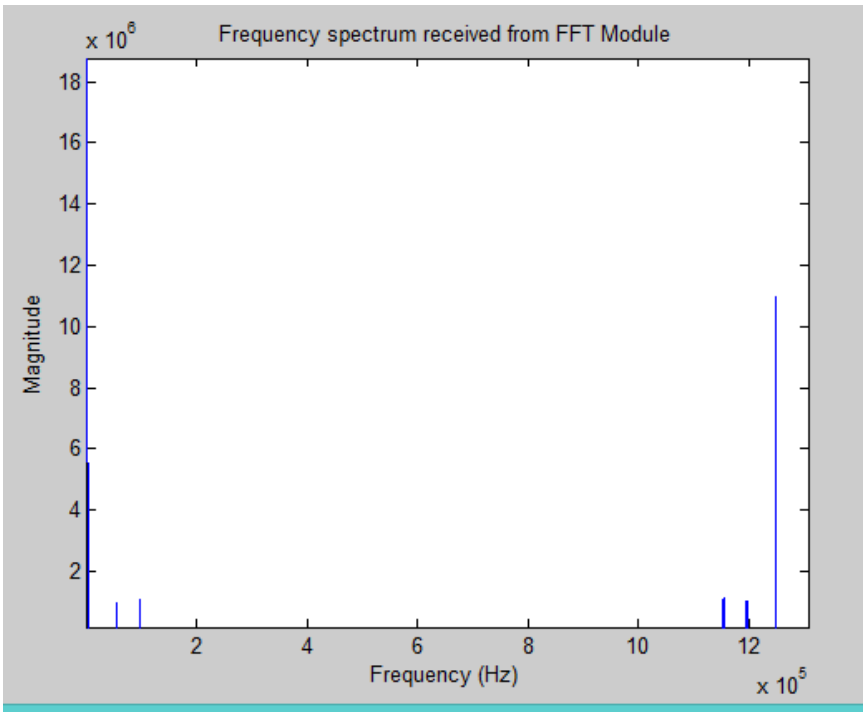

Figure 11: the above figure shows the magnitude spectrum outputted from the Verilog radar testbench (with input of 54 kHz and 96 kHz sinusoidal).

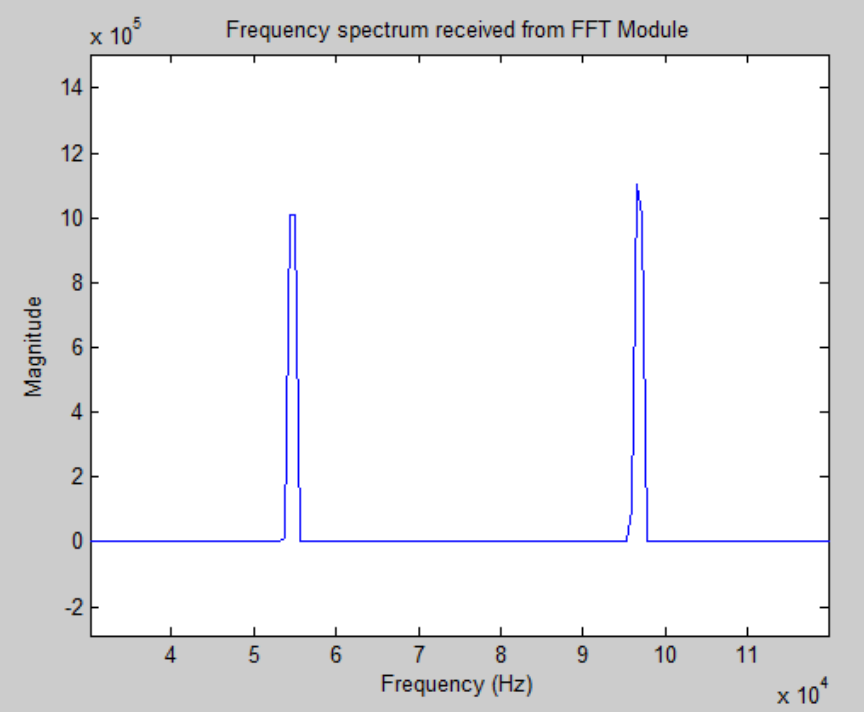

Figure 12: the above figure shows the zoomed in version of Figure 11.

Conclusion:

In conclusion, even though the system functions as expected in the simulation, it still needs to be tested with a FPGA and the PCB's designed by other team members.

Reference:

Spiral Software/Hardware Generation for DPS Algorithms.

http://www.spiral.net/hardware/dftgen.html