An Efficient & Reconfigurable FPGA and ASIC Implementation of a Spectral Doppler Ultrasound Imaging System

Adam Page
Dept. of Computer Science & Electrical Engineering
University of Maryland, Baltimore County
Baltimore City, USA

Tinoosh Mohsenin
Dept. of Computer Science & Electrical Engineering
University of Maryland, Baltimore County
Baltimore City, USA

***Abstract*—Pulsed wave (PW) Doppler ultrasound is a common technique used for making non-invasive velocity measurements of blood flow in humans. Most current PW Doppler ultrasound designs rely on fixed signal processing hardware; greatly limiting their versatility. This paper presents a highly efficient and highly versatile FPGA-based PW spectral Doppler ultrasound system. The system is implemented on a Virtex-5 FPGA using Xilinx's ISE design suite. In order to measure the accuracy of the system, a similar design was implemented in MATLAB. Furthermore, the design was also implemented in 65 nm CMOS ASIC design for performance comparisons. The Virtex-5 design requires 1,159 of 17,280 slice resources and consumes 1.089 watts of power when running at its maximum clock speed of 333 megahertz. The ASIC design has an area of .573 mm^2 and consumes 41 mW of power at a maximum clock speed of 1 GHz.**

Keywords—Doppler ultrasound; High performance; FPGA; CMOS; 65 nm;

I. Introduction

Pulsed Wave (PW) Doppler ultrasound is an important technique commonly used for making non-invasive velocity measurements of blood flow in the human body [2]. The technique makes use of what is known as the *Doppler effect*, a phenomenon in which there is a change in frequency of a wave for an observer moving relative to its source. Using the Doppler effect relationship between velocity and frequency, it is possible to determine the velocity of an object by measuring the change of the object's frequency relative to the medium in which the waves are transmitted. In order for PW Doppler ultrasound systems to measure blood velocity, they must be able to analyze the change in the observed frequency relative to the emitted frequency while filtering out noise. Therefore, these systems rely heavily on the use of digital signal processing (DSP) techniques. Most common PW Doppler ultrasound imaging systems use fixed DSP hardware to accomplish this. As a consequence, these systems have limited target frequency ranges.

In this paper, we propose a PW spectral Doppler ultrasound imaging system that is both highly efficient and versatile. The design is implemented on a Virtex-5 FPGA using Xilinx ISE design suite as well as in 65 nm CMOS ASIC design. The main components constituting the design include a finite impulse response (FIR) filter, hamming window, discrete Fourier transform (DFT), non-DC shift and magnitude, and finally a logarithmic compression. All of the frequency-specific components have been designed such that they can be tuned for a range of target frequencies by simply replacing corresponding lookup tables (LUTs). The following sections begin by discussing the overall design and its main components. Then, simulation data is presented and analyzed. From there, the FPGA implementation is compared to its equivalent ASIC implementation for performance purposes.

II. Background

There are currently only a few studies available for the implementation of an efficient, reconfigurable FPGA-based PW Doppler ultrasound system. These studies mainly discuss a few variations of the system but fail to discuss performance and details on the reconfigurability of the system [3][5]. For instance in [3], their system was implemented on a Virtex-II Pro with reported slice usage of 29%. However, without knowing the particular FPGA device, the slice *count* utilization is unknown. The report also fails to discuss power usage. Furthermore, the system is composed only of a hamming window and FFT, which have no discussion of reconfigurability. In [5], the system is designed using MATLAB Simulink tool. This tool was used to auto generate synthesizable HDL code. Unfortunately, no details are provided as to the reconfigurability or performance of the system.

III. PW Doppler Ultrasound System Design

A. Overview

As mentioned previously, the design consists of 5 major components: 107-Tap finite impulse response (FIR) filter, 128-point hamming window, 128-point discrete Fourier transform, non-DC shift & magnitude, and a base-10 log compression. For this design, we are assuming a sampling frequency of 150 kHz and 128 samples per set. This provides an adequate frequency resolution of 1.172 kHz. *Figure 1* depicts a simplified block diagram of the design. The system receives 2 16-bit inputs representing the real and imaginary portion received from the ultrasound transducers. The signals are then passed through a bandpass FIR filter. The filtered signals are then sent through a hamming window to prevent leakage/aliasing when sent through the discrete Fourier transform. After a set of 128 samples is sent through the DFT, the outputs are squared and added together. These single values are then compressed using a logarithmic compression. Only the first 64 samples of each 128-sample set are provided on the output data bus since we are interested in a single-sided spectrum. Subsections *B. – F.* discuss in detail each of the major components.

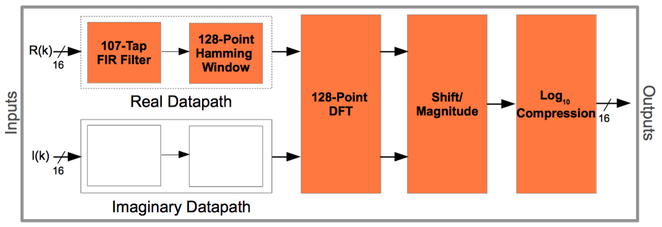

Figure 1: Proposed PW Doppler Ultrasound Imaging System Block Diagram

B. 107-Tap FIR Filter

The FIR filter is designed to meet the requirements detailed in *TABLE I*. The FIR filter is implemented using a N-stage delay design.

TABLE I. FIR FILTER DESIGN SPECIFICATIONS

FIR Requirements		
Lower Stopband	≤ 1 kHz	15 dB Attenuation
Passband	1.6 – 10 kHz	3 dB Ripple
Upper Stopband	≥ 11 kHz	25 dB Attenuation
Sampling Freq	150 kHz	

In order to meet the requirements, the design needs 107 coefficients. However, due to symmetry of the filter, only 64 coefficients need to be stored [1].

Figure 2 depicts the design of the 107-tap FIR filter where N in this case is 107. Since the sampling frequency is 150 kHz, the main clock can be easily made to be 64 times faster than the sampling clock. The design can then utilize this faster clock and perform the 64 multiplications and 63 additions using only one multiply-accumulate (MAC) unit. The design also requires 53 adders to sum each delayed value with the symmetric delayed value. These coefficients can be updated by replacing a LUT. Also, more than 107 taps can be accommodated while still maintaining the same throughput by utilizing more MAC units.

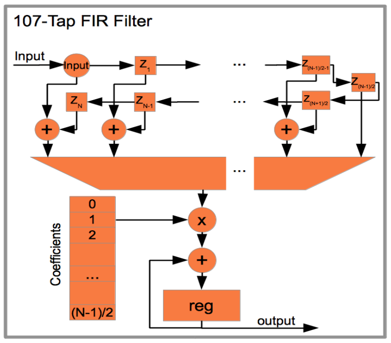

Figure 2: 107-Tap FIR Filter Design

The coefficients for the design were produced using MATLAB. The FIR filter component's accuracy was determined by comparing its outputs to outputs produced in MATLAB for the same input set. The plot in *Figure 3* shows the output produced by both MATLAB and Verilog implementations where the input stream consisted of 64 samples with a value of 127 followed by 64 samples with a value of 3192. The mean percent error in this example is 6.09%.

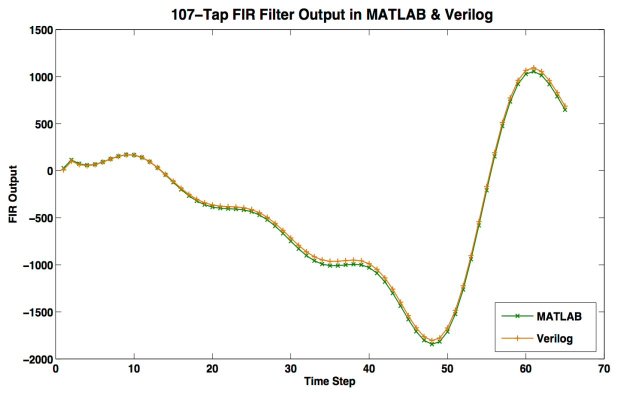

Figure 3: Plot of FIR filter Output in MATLAB & Verilog

C. Hamming Window

The hamming window is the next component the input stream is sent through. The purpose of the windowing is to prevent aliasing occurring in the following DFT stage from potentially non-periodic input sets [7]. The design required 128 coefficients since this is the size of a set. The hamming window design is depicted in *Figure 4*, below.

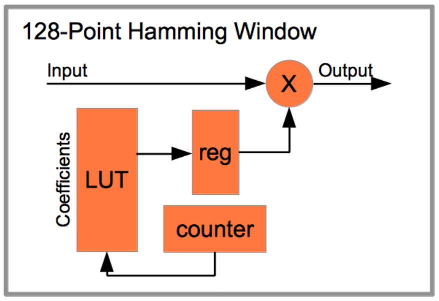

Figure 4: 128-Point Hamming Window Design

The hamming window coefficients were determined using the following equation [1].

$$w(n) = 0.54 - 0.46 \times cos\left(\frac{2\pi n}{N}\right), 0 \leq n \leq N$$

These coefficients are also located in a LUT. In this implementation, N is set to be 128. Due to symmetry, only the first 64 coefficients are needed. The coefficients were also multiplied by 2^{14} so that they can be stored as 16-bit 2's complement integers. The output for the block is then shifted by 14-bits to correct for this multiplication. Similar to the FIR filter, these coefficients can be replaced if other windowing styles are desired. The hamming window's accuracy was determined by comparing its outputs to outputs produced in MATLAB for the same 128 input set. The plot in *Figure 5* shows the output produced for fixed inputs of 2239. The maximum percent error was found to be only .509%.

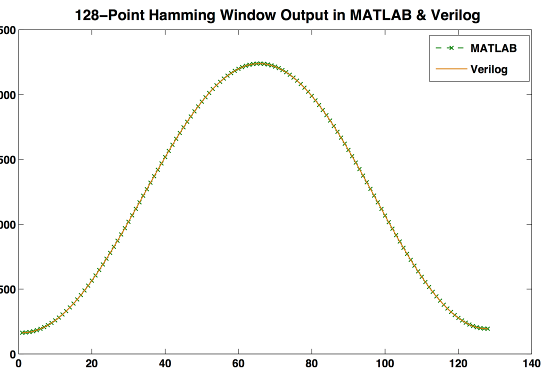

Figure 5: Plot of Hamming Window Output in MATLAB & Verilog

D. 128-Point DFT

The discrete Fourier transform is implemented using a fast Fourier transform (FFT) algorithm [6]. The Fourier transform maps each set of 128 samples from the time-domain to its corresponding frequency-domain. The following equation shows how each frequency value is determined:

$$X_K = \sum_{n=0}^{N} x_n \times e^{-\frac{2\pi kn}{N}i}, N = \#\,Points$$

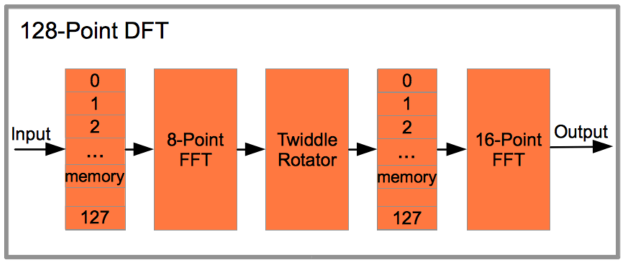

Figure 6: 128-Point DFT Design

Figure 6 shows the design of the 128-point DFT. As seen in the design, the 128-point DFT is implemented using an 8-point FFT and a 16-point FFT along with a twiddle rotator. The twiddle rotator multiplies the current output of the 8-point FFT with the appropriate twiddle factor. The twiddle factors are a predetermined set of values corresponding to the exponential in the DFT equation. A twiddle factor is typically denoted as W_N^n, where $W_N^n = e^{-\frac{2\pi n}{N}i}$. Comparing the design's output to the output produced by MATLAB for some given input set measured the accuracy of the 128-point DFT design. The plots depicted in *Figure 7* were produced by supplying an impulse for the input set, such that input samples were 1 except input sample 63 and 64, which had values of 8000. The Verilog version was able to mirror almost exactly the output produced in MATLAB with a maximum percent error of .367% for the real and .616% for the imaginary portion.

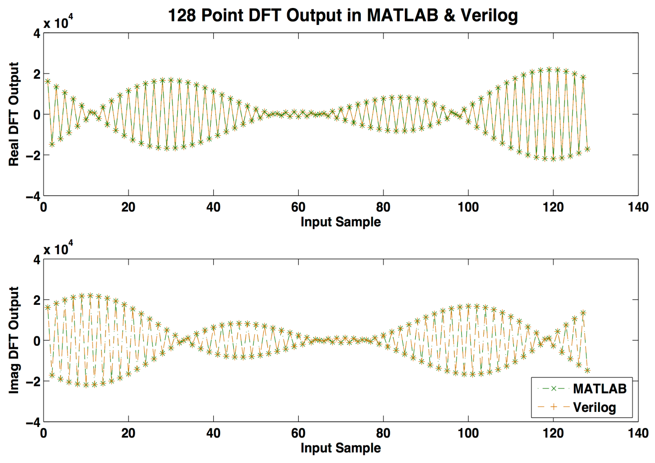

Figure 7: Plot of 128-Point DFT Real & Imaginary Output in MATLAB & Verilog

E. Non-DC Shift & Magnitude

The first 64 output pairs of the DFT are then sent through the non-DC shift & magnitude component. This component is responsible for squaring both the real and imaginary inputs and then summing these values together. Additionally, the non-DC components are multiplied by a factor of 2^2, which is done using a counter and shifter unit. Discarding the second 64 output pairs and multiplying the non-DC components by 4 is done to convert from a two-sided power spectrum to a single-sided spectrum. The design for this component can be seen in *Figure 8*.

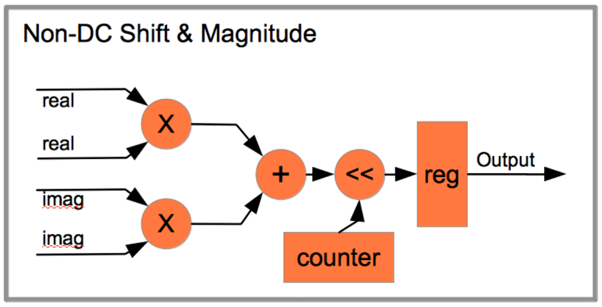

Figure 8: Non-DC Shift & Magnitude Design

The component was again tested for accuracy, but is omitted here due to the simplicity of the design.

F. LOG$_{10}$ Compression

The final component constituting to the overall design is a base-10 logarithmic compression. There are various ways of implementing such a component. The approach taken in this design uses Mitchel's Approximation Method [4]. The input data is taken as unsigned 16.0 format and the output is put in unsigned 4.12 format. The whole portion of the output is equal to the index of the most significant bit (MSB) of the input. This is done using a modified 16x4 decoder. The fractional portion of the output is equal to the input's bits to the right of the MSB and is padded on the right with zeros as needed. The design of the log compression is shown in *Figure 9*.

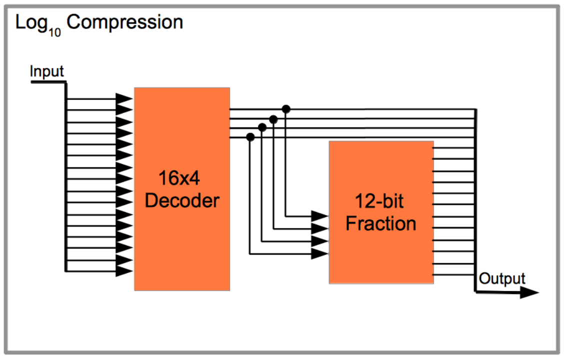

Figure 9: Log$_{10}$ Compression Design

Comparing the design's output to the output produced by MATLABs log10() function for some given input set measured the accuracy of the logarithmic compression. The plot depicted in *Figure 10* was produced by supplying an input set that swept from 2^0 to 2^{16} with increments of 3. Looking at the plot, it is clear that the approximation does an excellent job of interpolating the actual values with a maximum percent error of less than 2%.

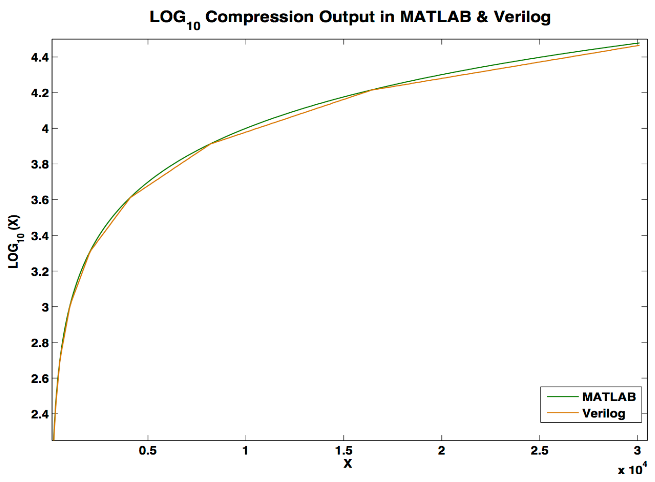

Figure 10: Plot of Log$_{10}$ Compression Output in MATLAB & Verilog

IV. Proposed System Simulation Results

After designing all of the necessary components, they were then connected together with pipeline stages between each component. Each component also included many pipeline stages internally in order to allow for even higher clock frequencies if needed. From examining each component's design, it can be seen that the overall design can be easily reconfigured for various target frequencies. In particular, the FIR filter and hamming window contain all of their frequency-dependent information in LUTs. The following plot, *Figure 11*, shows the output produced in MATLAB as well as from the design simulated in Xilinx's ISim software. The input data consists of 224,896 samples that were taken from an actual ultrasound transducer and saved in a text file. The FPGA design was passed the input data via an input file and the output of the design was similarly stored in an output file. This output file was then imported into MATLAB and displayed using the imshow() function. A MATLAB function was also written to mirror that of the FPGA design. This function used the same code used previously to test each main component. The function was passed the input data file and used imshow() to produce its image.

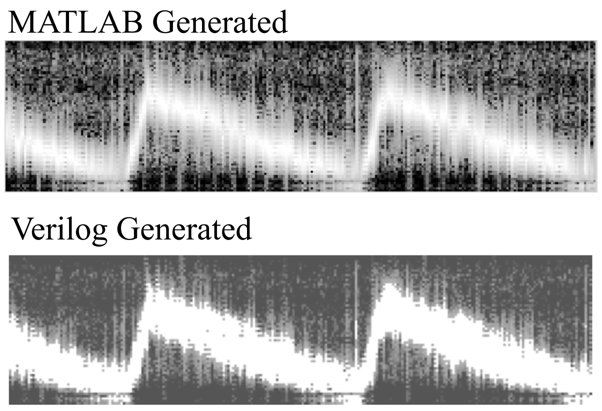

Figure 11: Ultrasound Image generated in MATLAB & Verilog

The design was also programmed on an Xilinx ML505 board housing a Virtex-5 FPGA. The same input file was sent to the board from the computer via USB-to-Serial Comm. port. The data was then processed, stored into memory, and displayed on a monitor via DVI connector. An image of the setup with results is shown in *Figure 12*.

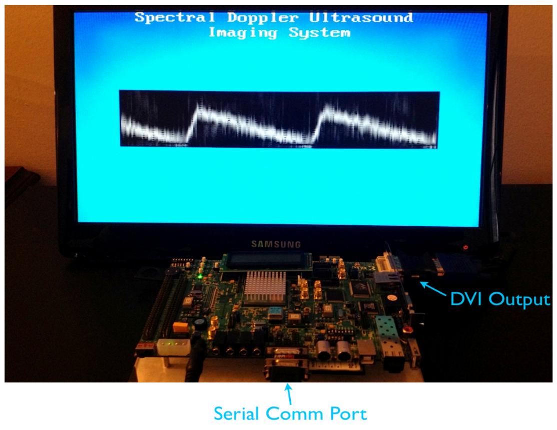

Figure 12: Design Implemented on Xilinx ML505 Board

V. ASICs Implementation and Comparison

The PW Doppler ultrasound system is also implemented in 65 nm CMOS ASIC design. The RTL version of the system was generated using RTL Compiler and then laid out in an ASIC design using standard cells in Cadence Encounter software. The following, *Figure 13*, is an image of the layout for the ASIC implementation.

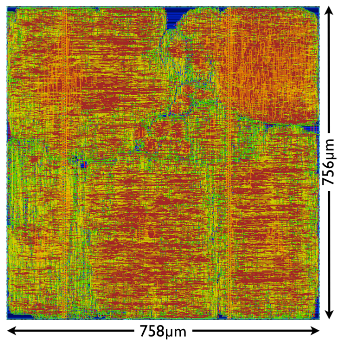

Figure 13: ASIC Implementation of PW Doppler Ultrasound System

Place and route results indicate that the ASIC layout occupies a total area of .573 mm^2 and can operate at clock speeds of up to 1 GHz with a power consumption of .0408 watts. As stated previously, the design is also implemented in an Xilinx Virtex-5 FPGA. The FPGA implementation used 3,223/69,120 slice registers, 3,199/69,120 slice LUTs, and 1,159/17,280 slices. Xilinx XPower Analyzer estimated a power consumption of 1.089 W of which 0.046 W is from dynamic and the rest is from quiescent power consumption. The maximum reported reliable clock speed using a balanced design strategy for the FPGA implementation is 333 MHz. The two design implementations' characteristics are summarized in *TABLE II* and *TABLE III*.

TABLE II. FPGA IMPLEMENTATION CHARACTERISTICS

FPGA Implementation		
Technology	65 nm, 1 V	
	Utilization	**Total Available**
Slice Register	3,223	69,120
Slice LUT	3,199	69,120
Occupied Slices	1,159	
Performance (MHz)	333	
Total Power (W)	1.089	

TABLE III. ASIC IMPLEMENTATION CHARACTERISTICS

ASIC Implementation	
Technology	65 nm, 1 V
Logic Utilization	93%
Total Area (mm^2)	.573
Performance (MHz)	1,000
Total Power (W)	0.0408

It is clear from the tables that the ASICs implementation consumes far less power and can run at a much higher clock frequency than the FPGA implementation. The dynamic power dissipation of the ASIC implementation when scaled to 333 MHz is approximately 12 mW, which is 4 times lower than the FPGA implementation's dynamic power of 46 mW. More importantly, the FPGAs leakage is very high at about 1043 mW, which is 115 times higher than ASICs leakage power of 8.995 mW. On the other hand, FPGAs provide an economically suitable reconfigurable platform whereas the cost of producing the ASIC design would only be feasible for large-scale production. Furthermore, in order to alter the ASIC design requires redoing the layout, which is not efficient. Advanced FPGAs, such as a Virtex-7, are a very good alternative as they can be easily reconfigured while still maintaining relatively high efficiency.

VI. CONCLUSION

This paper presents an efficient PW Doppler ultrasound imaging system implemented in both an FPGA and ASIC design. The current system is targeted for 150 kHz sampling frequency, but can be easily adjusted for other target frequencies. The FPGA design had comparable efficiency and performance compared to the ASIC implementation, while having the advantage of being cost-effective and reconfigurable. The ASIC implementation is capable of being driven at clock frequencies up to 1 GHz while consuming only approximately 0.0408 W. Additionally, the ASICs layout has a total area of .573 mm^2 with 93% logic utilization. The FPGA implementation is capable of being driven at clock frequencies up to 333 MHz and consuming approximately 1.089 watts of power. Programming the FPGA design on a Virtex-5 and running a sample test-bench produced excellent results when compared to results produced in MATLAB.

REFERENCES

[1] A. V. Oppenheim, A. S. Willsky, "Signals & Systems," 2nd ed., Prentice Hall, 1997.
[2] C. Huang, P. lee, P. Chen, and T. Liu, "Design and Implementation of a Smartphone- Based Portable Ultrasound Pulsed-Wave Doppler Device for Blood Flow Measurement," IEEE Trans on Ultrasonics, Ferroelectronics, & Freq Control, vol. 59, pp. 182-188, January 2012.
[3] C. Hu, Q. Zhou, & Shun, "Design and Implementation of High Frequency Ultrasound Pulsed-Wave Doppler using FPGA," Ultrasonics, Ferroelectrics and Frequency Control, IEEE Transactions, vol. 55, no.9, pp. 2109-2111, September 2008
[4] D. J. McLaren, "Improved Mitchell-Based Logarithmic Multiplier for Low-power DSP Applications," published.
[5] D. Mahmoud, A. M. Youssef, and Y. M. Kadah, "Embedded Doppler Ultrasound Signal Processing Using Field Programmable Gate Arrays," unpublished.
[6] S. Cho, K. Kang, "A low-Complexity 128-Point Mixed-Radix FFT Processor for MB-OFDM UWB Systems."
[7] "Understanding FFT Windows," Application Note AN014, LDS Ltd, 2003.