Ultrasound Research Platform Prototype

TABLE OF CONTENTS

ABSTRACT

Current ultrasound research platforms either do not allow for sophisticated, multi-element excitations or are substantially large due to the complexity of the hardware. This hinders the research of the use of coded excitation signals in ultrasound to potentially diagnose tumors. Therefore, the objective of this study is to develop a portable, cost-efficient, multi-element prototype research platform. To minimize hardware complexity, waveforms are encoded by using sigma-delta modulation and transmitted in parallel at a high frequency by using a field-programmable gate array. Correlation between this system and a simulated method of a current multi-element system is 99.3%. Several amplifier designs were created and simulated. Data processing schemes such as delay sum beamforming and a Wiener filter were implemented to convert recorded echoes into an image. These schemes were tested in two simulations, which used a pre-enhanced linear chirp and a conventional pulse as the excitation signals. The pre-enhanced chirp increased the depth of the conventional image by 20 mm.

I. INTRODUCTION

Conventional medical ultrasound uses a short duration electrical pulse excitation for imaging. The ultrasonic transducer converts the electrical energy to pressure waves which penetrate human tissue. These waves are reflected in all directions, but a small amount of energy is reflected back towards the transducer proportional to the density of the tissue [1]. The ultrasonic transducer converts these waves into a voltage waveform, which is recorded for signal processing and imaging.

Coded excitations have a longer duration than the signals used in conventional ultrasound. Previous research conducted using coded excitation indicates that this technique improves the signal-to-noise ratio (SNR) [2]. Consequently, this increases the penetration depth of ultrasonic imaging. Coded excitation has also been shown to improve the spatial resolution of ultrasonic images [3]. These improvements have the potential to enhance the diagnostic capabilities of medical ultrasound.

Current ultrasonic research platforms to test the application of coded excitations are limited. Single-channel platforms suffer from slower data acquisition and are more prone to motion artifacts. Multi-channel platforms, such as the Remotely Accessible Software configurable Multi-channel Ultrasound Sampling (RASMUS) system, are large, high-cost, and quickly become outdated [4]. Despite these flaws, the RASMUS is the most sophisticated research platform to date; consequently, researchers travel from all over the world to test their work with this system.

The objective of this project was to develop a portable, cost-efficient, multi-channel, prototype research platform for coded excitations. The motivation was to design a system that decreased the hardware size and cost as compared to the RASMUS system. By encoding excitation waveforms with sigma-delta modulation, this allowed for a 1-bit

resolution format while retaining most of the signals' information. With only two voltage levels required to transmit this waveform, the need for digital-to-analog converters was eliminated [5]. Furthermore, power amplifiers, with their complex design and sensitivity to component variations, were replaced with less expensive switching power amplifiers. The engineering tradeoff for this size reduction was a faster sampling rate to retain accuracy. To transmit the sigma-delta modulated signal at the required rate, a field-programmable gate array (FPGA) was used.

The significance of this prototype research platform is that it will allow more ultrasound researchers access to affordable, multi-channel testing devices. With this access, researchers could be able to enhance ultrasonic imaging techniques that diagnose tumors, which could remove the need for invasive biopsy diagnostics.

II. SYSTEM DESCRIPTION

A high-level description of the prototype is represented in the system block diagram shown in Fig. 1.

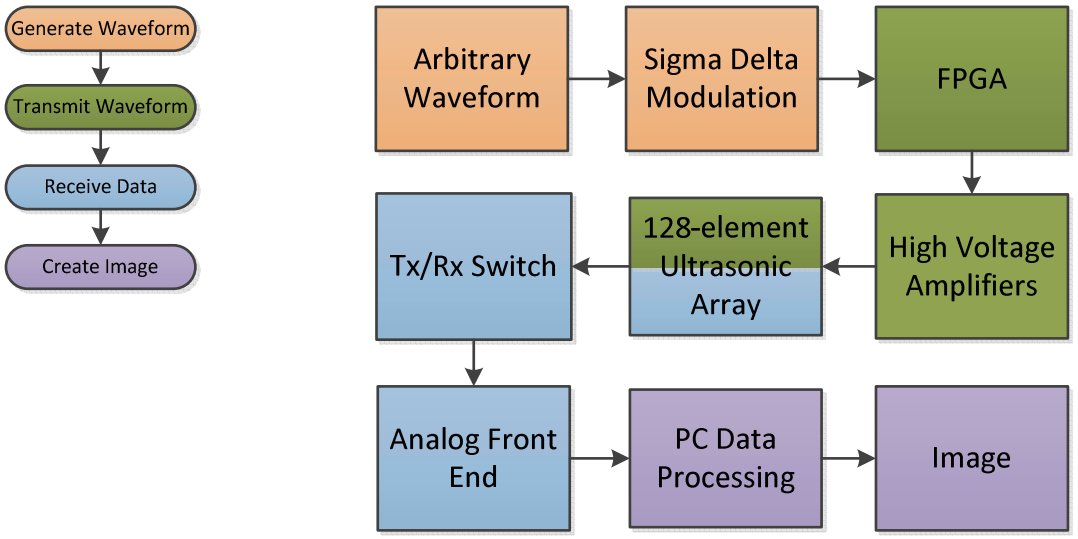

Fig. 1. High level system block diagram

The first phase of the system begins with the generation of coded excitation waveforms using MATLAB (Mathworks, Natick, MA). These waveforms are then digitally encoded using the sigma-delta modulation toolbox [6]. The user can assign converted waveforms to pins on the ultrasonic transducer as desired using a graphical user interface (GUI). To allow for beamforming to correct the returned image, delays can also be specified for the pins used. The waveforms and output delays are transferred from the personal computer (PC) to the FPGA using a universal asynchronous receiver transmitter (UART) protocol and stored in memory for later transmission. Each waveform can be at most 3 μs in duration to reduce negative ultrasound bioeffects [7].

Following the generation of the waveform, the process of transmission begins with a start command sent from the PC to the FPGA. The waveforms previously stored in memory are retrieved after the desired delay for each pin. The waveforms are then transmitted at 500 MSamples/s to the FPGA's output pins. This is performed in parallel for a maximum of four channels. Although not completed, the high voltage amplifiers would then amplify the signals to high voltage levels to increase the penetration depth. Next, the ultrasonic transducer converts the signals into a pressure wave that then traverses the object to be imaged. Due to the band-pass shape of the transducer's frequency response, a close approximation to the original analog signal is recovered from the sigma-delta modulated form.

After transmission, the reflected pressure waves are then received by the ultrasonic transducer. The electrical signals then pass through the transmit/receive (Tx/Rx) switch to the analog front-end. The transmit/receive (Tx/Rx) switch protects the analog front-end circuitry from the high voltages of transmission by clamping the voltage

to a safe level. The analog front-end device captures the voltage data and transmits it to the PC.

Finally, the PC processes the received voltage data in order to construct an image. The processing occurs in many steps described in detail in the data processing methods section of this paper. After an image is constructed, it is displayed in a GUI. Parameters such as maximum depth and dynamic range allow the user to adjust the image displayed.

III. METHODS

Sigma-delta modulated waveforms represent the quantitative properties of a signal as a stream of binary data, using only two quantization levels. This stream of data can then be transmitted from a FPGA. After capturing reflected wave data, processing is performed to construct an image. These methodologies are described in the following sections.

A. Sigma-Delta Modulation

Sigma delta modulation is an analog-to-digital conversion (ADC) technique that includes quantization error compensation [8]. This technique will be used to store, in digital form, arbitrary waveforms that will be transmitted to the ultrasonic transducer. A software-based 1-bit ADC is initially used to round the input to either 1 or -1. This value is then subtracted from the initial input creating an error value. The error is added to the next input value, providing conversion error compensation. This adjusted input is rounded to 1 or -1, producing the next output value. The overall output expresses an input of analog values as a series of 1s and -1s. This process is shown in Fig. 2.

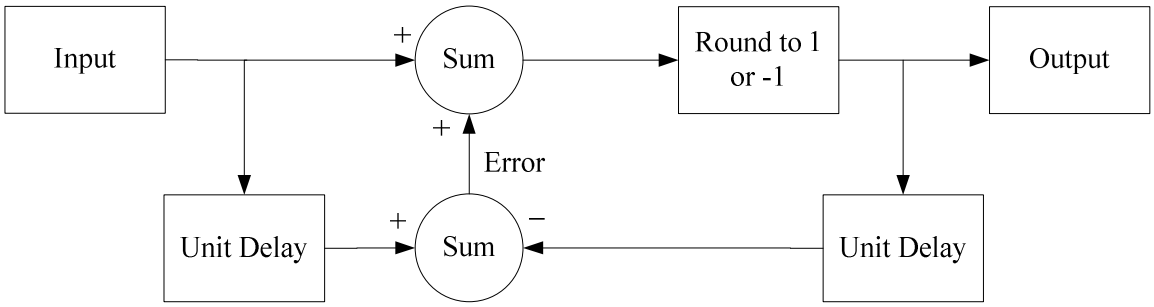

Fig. 2. Software flowchart for sigma-delta modulation

To increase accuracy, a second order sigma-delta modulator was used. This adds the error from the previous two conversions before rounding to 1 or -1. A third order sigma-delta modulator was not used as this can cause overloading and saturation if the input signal has a high frequency and amplitude. The fast change between large values can cause the error to accumulate faster than the sigma-delta modulator can compensate. This results in a consistently high error to add to each new input value, which causes the output to reflect the accumulated error rather than the input of the function. The resulting modulated signal consists of blocks of ones and negative ones instead of the desired oscillation between ones and negative ones. Therefore, a second order sigma-delta modulator was used to give the highest accuracy without concern for overloading.

B. FPGA Architecture

A Virtex 5 FPGA (Xilinx, San Jose, CA) receives waveform data from the PC, writes and reads the data from the double data rate (DDR2) memory. Once a transmit signal from the PC is sent, the FPGA transmits the waveform data in parallel to the output pins.

1) *PC to FPGA Communication*

A GUI in conjunction with a serial transmission program was designed to interface the PC to the FPGA. The GUI allows the user to select sigma-delta modulated

waveforms to be uploaded. These waveforms must have a length of 1536 bits, which is checked by the software of the GUI. This allows the waveform to be transmitted for a duration of 3 μs at a rate of 500 MSamples/s with 36 buffer bits. The buffer bits allow the input to be divisible by 256, which is the desired format for DDR2 memory. Once waveforms have been selected, the waveforms can be assigned to any number of the four output pins. Each of these output pins can also be assigned a delay. After the user has assigned all desired waveforms and delays, the user can click a "load settings" button to upload the waveform and delay data for each of the four pins to the FPGA's DDR2 memory. The functional requirement for this subsystem was to upload this data in less than 1 second. To meet this requirement, communication was performed over the UART protocol at a rate of 115,200 baud, which is the maximum baud rate for the Virtex 5 FPGA. The user can then click an output now button to start the transmission of waveforms from the DDR2 memory to the assigned output pins at 500 MHz.

2) *Arbitration System*

To integrate the DDR2 interface with the other FPGA functionalities, a system was constructed to arbitrate access to the memory. The importance of the arbitration system is to allow multiple components to connect to the memory. The waveform data, which is first stored in memory, needs to be accessible from multiple software blocks, or sockets. However, only one socket can retrieve data from the memory at a time. While constructing an arbitration system, special consideration was needed to ensure that more important sockets had priority to access the memory. For example, a socket requiring data for immediate transmission would have priority over a socket receiving waveform data from the PC. Furthermore, the arbitration system must prevent starvation caused by a large number of requests for memory access depriving a socket from its turn. A scheme

was developed to reduce the possibility for starvation by cycling access in a round robin fashion. The arbitration system follows a process outlined in Fig. 3.

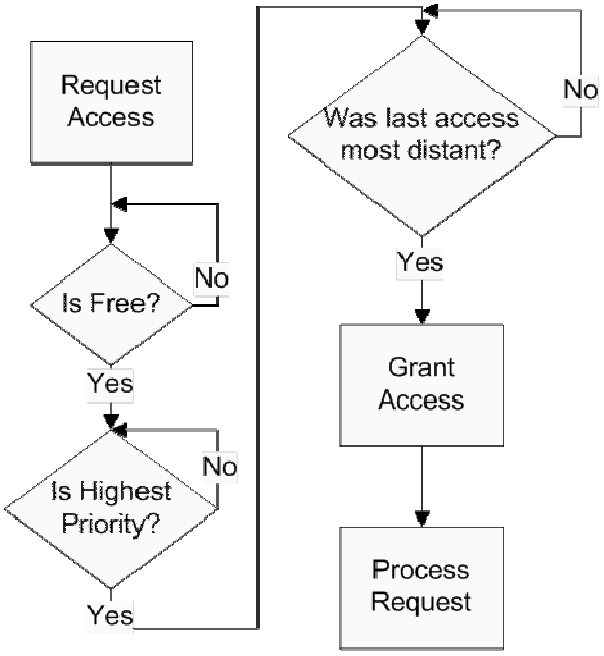

Fig. 3. Arbitration system flow chart

Multiple sockets may request access to the memory at the same time. To ensure data integrity, only after the current block completes its list of reads and writes are the remaining of the requesting sockets considered for access. Next, the remaining requesting sockets are compared to determine which socket should receive access next. This is performed in two steps: all requesting sockets of a lower priority are eliminated, and then the histories of the remaining sockets are compared. Access is then provided to the remaining socket that had its previous access furthest in the past.

3) *High Speed Transmission*

Due to the nature of the sigma-delta modulated coded excitations, a high data rate of 500 MSamples/s was needed to retain the accuracy of the signals. A clock rate of 250 MHz was used for the FPGA, as it was near the maximum speed of the DDR2 memory.

Therefore, a method to double the speed of transmitting the data from DDR2 memory to the output was developed. By using both edges of a single clock, rather than just one edge, the speed of transmission is doubled. Because the flip-flops on an FPGA can only be clocked on one edge of the clock, two separate signal streams were made. One signal path used flip-flops clocked on the rising edge, and another used flip-flops clocked on the falling edge. These signals are then combined by an exclusive or (XOR) gate to create the output. This process is shown in Fig. 4.

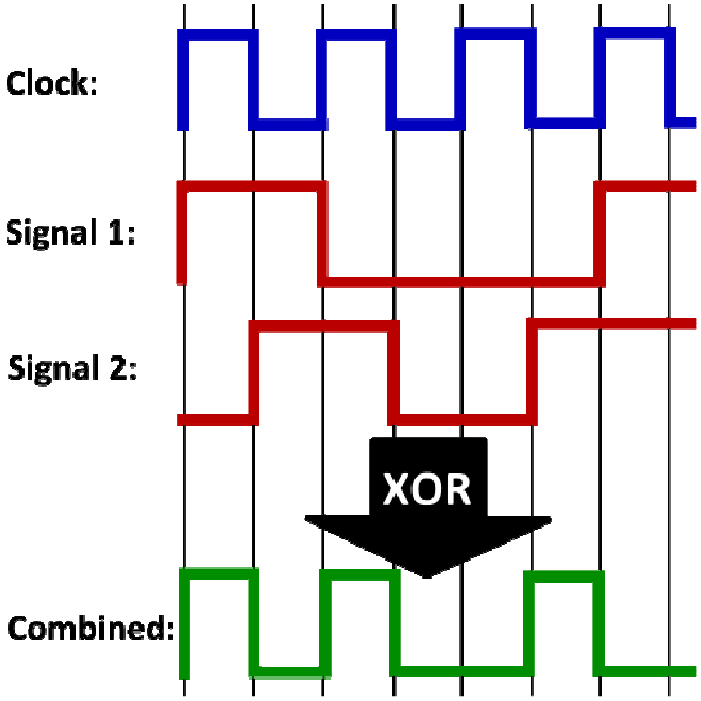

Fig. 4. High-speed transmission method

4) *Controls for Excitation*

To offer more flexibility to the user of the platform, two controls were added to the system: assignment of waveforms to output pins, and assignment of delays to output pins. To implement the assignment of waveforms to output pins, functionality was added to store the location of the waveforms in the DDR2 memory. The location is then combined with an identification number that represents the assigned output pin for that

waveform. Delay assignments were implemented by using a counter that starts when the output now button is pressed by the user. The delays are converted from seconds to counts from the beginning of transmission and stored in a table, one value for each pin. Because the counter increases every clock cycle, the smallest delay increment is 4 ns. Each pin is held at zero until the current count matches each pin's stored delay value. The transmission for each pin then begins independently.

C. High Voltage Amplifier

The amplifier is a key component of the ultrasound system. The output waveform of the amplifier will be sent to the transducer. Over the course of the project, the amplifier has evolved and changed. The first design proposed was using a metal-oxide semiconductor field-effect transistor (MOSFET) H-bridge design. This design was chosen for the low on-resistance, and because it can produce a bipolar signal. The part that was chosen was Zetex's ZXMHC10A07N8 complementary MOSFET H-bridge. This component was tested using the switching test circuit provided in the datasheet [9]. The design was simulated using a tool called *LTSpice IV*. This tool is a free program offered by Linear Technology.

The H-bridge in LTSpice was simulated using four MOSFETs that had very similar characteristics to those provided by Zetex. The H-bridge was excited using a -10V to 10V signal that was to represent an amplified output from the FPGA. The simulation was run at lower frequencies to prove the concept of the design. The amplifier simulation was set up as show in Fig. 5. The resistor values used were standard 1 kΩ resistors. These provided a low enough current to safely run the MOSFETs.

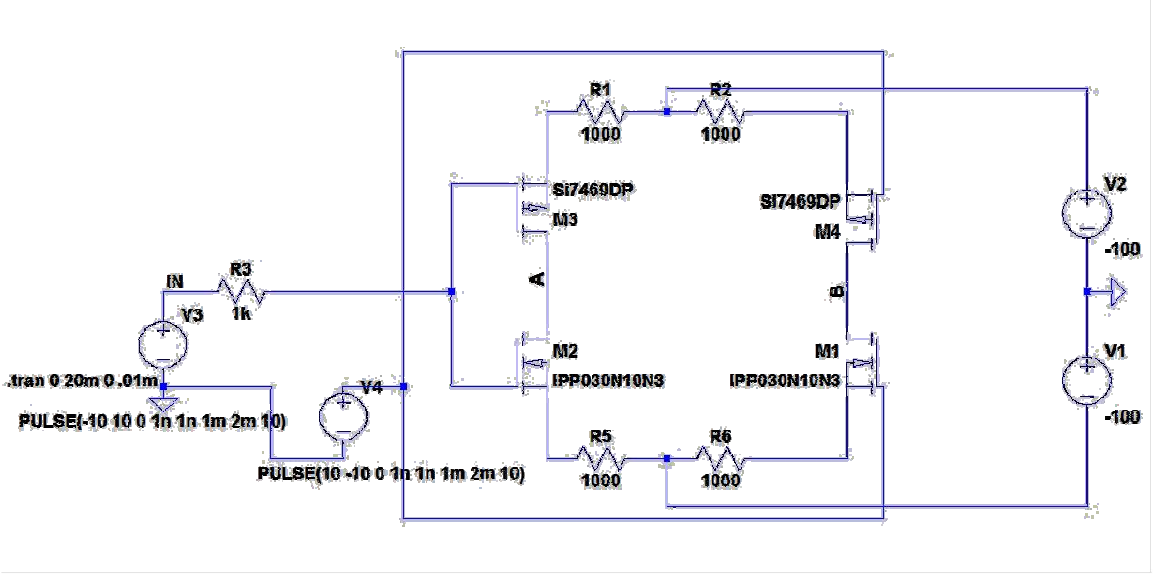

Fig. 5. H-Bridge Amplifier

The next model that was suggested was a push-pull configuration using 2 N-channel MOSFETs. This design had the advantage of being able to run at radio frequency (RF) speeds. Since the speeds were so high, the design was limited to N-channels because P-channel MOSFETs could not run that fast.

This design was also modeled in LTSpice. Since a component was not provided for this design, components that would be able to meet the specifications were picked from the library supplied by the tool. This design could be started from scratch rather than limited to previous parts received. Again, the design was excited using a -10 V to 10 V square wave. This circuit was also simulated at lower frequencies to verify plausibility. The basic design came from research performed into RF push-pull amplifiers. The 1 kΩ resistors serve the same function as they did in the H-bridge design. They limit the current to safe levels for the MOSFETs. The two other MOSFETs were simply replaced with 1 MΩ resistors to simulate an open circuit. Fig. 6 shows the LTSpice model that was simulated.

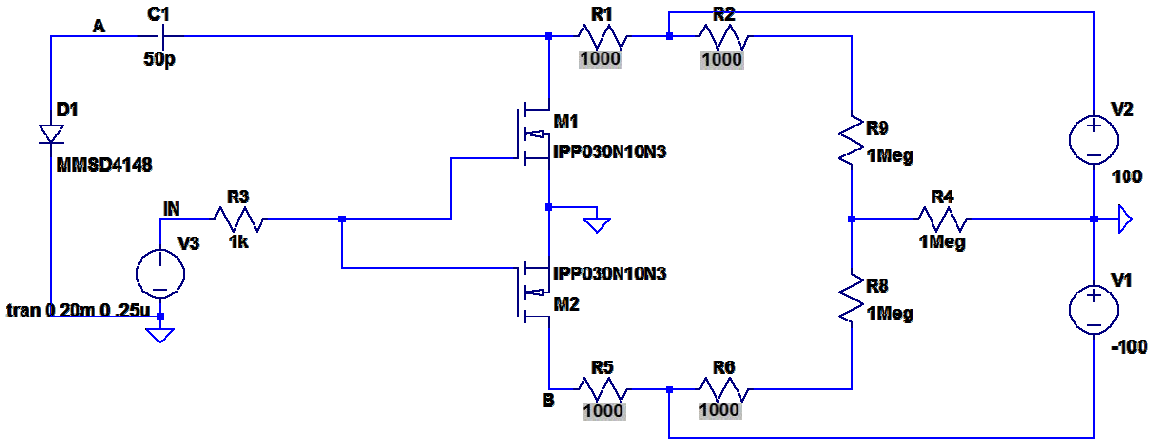

Fig. 6. Push-Pull Amplifier Configuration

The final design on the amplifier was a single ended MOSFET. This design used less material, therefore taking up less space, and could provide the signal at a lower cost. The availability of a RF MOSFET that would function properly at the previously required 1 GHz and 100 V is very low. Therefore, the specifications were dropped to -50 V to 50 V output at a frequency of 500 MHz. This would ensure that a single MOSFET would be able to handle the power and speed.

The single-ended MOSFET would provide an inverted amplified result of the input. For this design, the input was also changed. A 0 V to 5 V square wave was implemented as the excited signal. Shown in Fig. 7 is the final design.

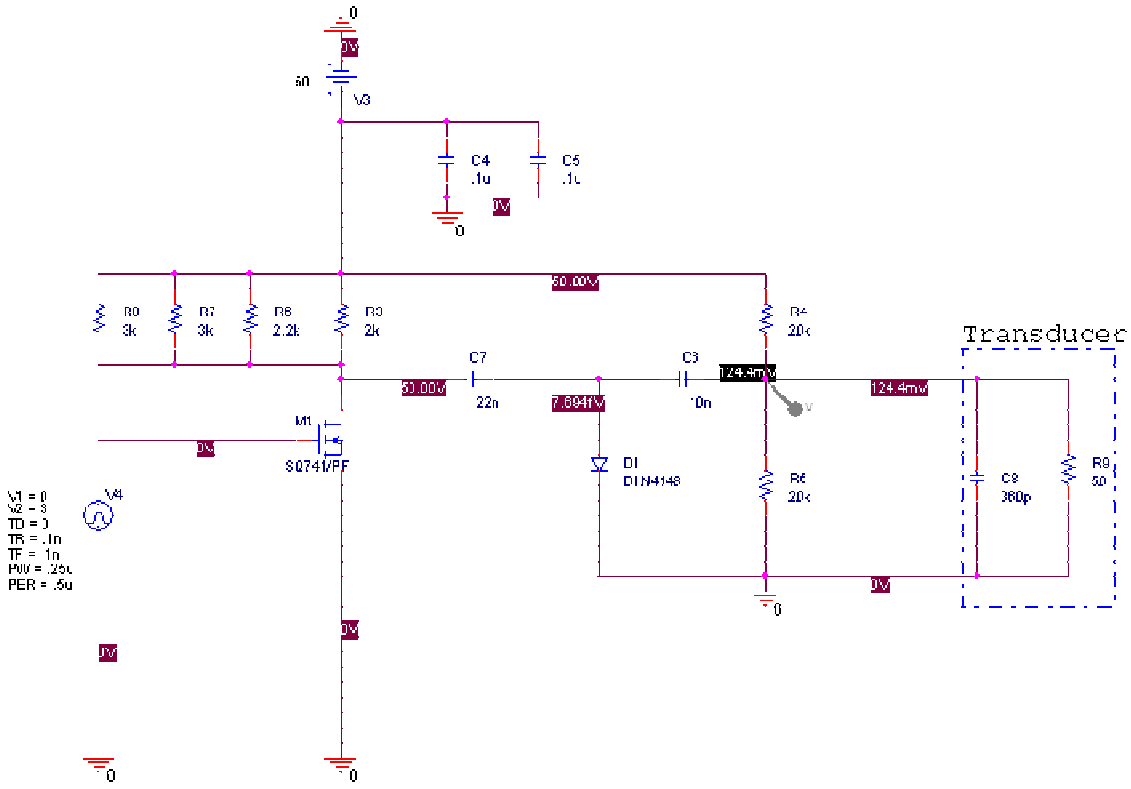

Fig. 7. Single-ended Amplifier Design

The decoupling capacitors are there to simulate a real voltage source. The 1 kΩ resistors that had previously been used were replaced by four resistors in parallel so as not to load a single resistor with too much power. Using four resistors at a R_{total} value at approximately 1 kΩ reduces the power by roughly one-fourth. The capacitors C7 and C8 eliminate the DC offset associated with the amplifier. The diode essentially clips the voltage so there is no overshoot when switching.

Fig. 7 shows the transducer model connected to the amplifier. This model was determined by using the transducer specifications such as impedance as a function of frequency and the Mason model [10]. The transducer model was calculated to be a 360 pF capacitor in parallel with a 50 Ω resistor. Loading effects that distorted the output of the amplifier were observed when the amplifier was simulated with the transducer model.

D. Data Processing

As previously stated, in ultrasound the backscattered pressure waves are converted into voltage waves by the ultrasonic transducer. An analog front-end samples and records these waves for data processing. Data processing consists of filtering and manipulating the data to produce an image. In the completed system, these functions were performed on a PC using MATLAB software. The procedure for data processing is illustrated in Fig. 8.

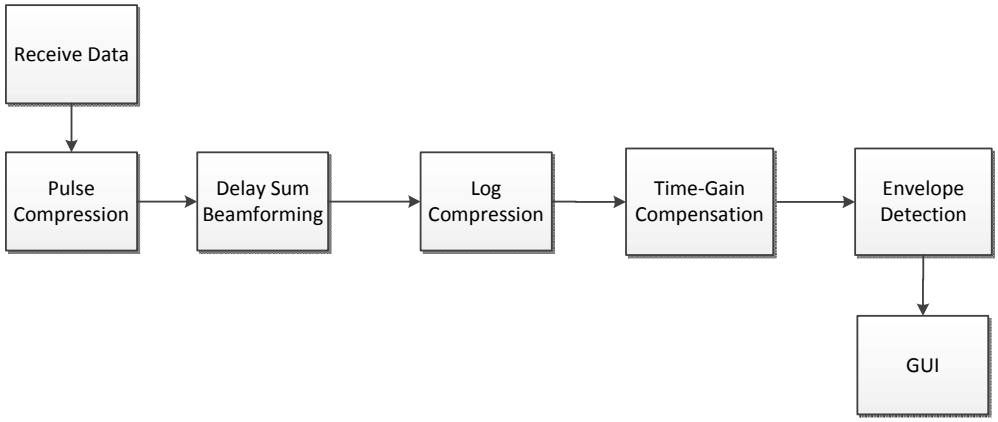

Fig. 8. Data Processing Procedure

1) *Pulse Compression*

The first step in data processing is to perform pulse compression. When ultrasonic imaging is performed using a longer excitation, the spatial resolution of the received data is degraded relative to conventional ultrasound, which uses a short pulse as an excitation. Pulse compression uses a filter to compare the received data with the original excitation, restoring much of the resolution which was lost by using a longer excitation.

The most common method used for pulse compression is the matched filter. This filter has an impulse response equal to the excitation signal with reversed time axis. Thus, the filter is equal to the complex conjugate of the excitation signal in the frequency

domain. Applying the matched filter is equivalent to performing cross correlation between the excitation signal and the received data. The matched filter is commonly used because it attenuates all noise outside the frequency range of the excitation signal; however, filtering using a matched filter results in a sinc shape, with sidelobes on either side of the main lobe. If these sidelobes are too large, they can create false echoes in the image [2].

In the absence of noise, an inverse filter is the optimal filter to reduce sidelobes while restoring the spatial resolution. This filter is equivalent to the inverse of the excitation signal in the frequency domain. Thus, the filter amplifies noise outside the frequency spectrum of the excitation. This makes the filter impractical in real systems [2].

A Wiener filter was employed to perform pulse compression on the research platform [2]. This filter is an optimization of two components, the matched filter and inverse filter. The two filters are combined in the Wiener filter equation (1), where V_{in} is the excitation signal, λ is a smoothing parameter, and S is the estimated signal-to-noise ratio of the system [11].

$$W(f) = \frac{V_{in}^*(f)}{|V_{in}(f)|^2 + \lambda/S} \qquad (1)$$

The value of λ/S is used to adjust the operating point of the filter on the continuum between the matched and inverse filters. By inspection of (1), it can be seen that as this term increases, the denominator becomes dominated by the λ/S, which is a constant. The equation then resembles the complex conjugate of the excitation signal multiplied by a constant, which is equal up to a constant to the matched filter. In the limit as λ/S approaches zero, the denominator is equal to the square of the magnitude of the

excitation signal. The square of the magnitude of the excitation signal is equal to the excitation signal multiplied by its complex conjugate. The complex conjugate terms present in the numerator and denominator cancel, and the equation resembles the inverse of the excitation signal in the frequency domain, which is equivalent to the inverse filter.

The Wiener filter is therefore asymptotically equal to the matched filter and inverse filter, and equal to a blending of those two filters for intermediate values of λ / S. The smoothing parameter, λ, can then be adjusted, based upon the type of excitation signal and the type of tissue being imaged, to optimize the Wiener filter for both noise attenuation and sidelobe reduction.

2) *Delay Sum Beamforming*

Delay sum beamforming compensates for the spherical propogation of the sound waves that are transmitted and received. Without this conversion, echoes appear in the resulting image as scatterers are represented by arcs of points rather than single points. This is due to the unique path that sound waves travels from each scatterer to each transducer which makes each scatterer appear to be differing distances from the sensor. These differing distances are equal to the Pythagorean distances from the scatterer to each transducer element. If unaccounted for, the changes in distance from a scatterer to each transducer element can create a misalignment in the overall image as seen in Fig. 9. In this figure, the single scatter point at 10mm is represented by an arc that stretches from 20mm to 10mm rather than a single point.

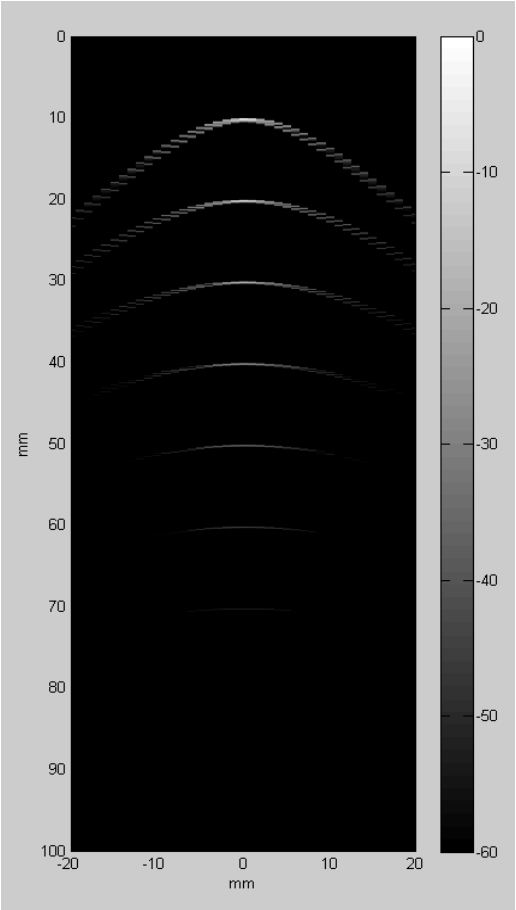

Fig. 9. Effects of Ultrasound Imaging on Single Points

Delay sum beamforming analyzes each pixel of the image individually by calculating the Pythagorean distance from the pixel to each transducer element and from the pixel to the focal point of the transducer. The focal point is located at a distance of 20 mm from the center of the transducer. (2) shows how delays are calculated, where D_p is the calculated delay for a pixel, c is equal to 1540 m/s (the average speed of sound in tissue), D_{sp} is equal to the distance from the transducer element to the pixel, and D_{fp} is equal to the distance from the transducer element to the focal point. Note that there is no delay at the focal point, as D_{sp} equals D_{fp}.

$$D_p = \frac{D_{sp} - D_{fp}}{c} \qquad (2)$$

Once the delays have been calculated they are applied to adjust the positioning of backscattered echoes. The new values at the distance from the pixel to the edge of the transducer, which have been recorded and delayed for each transducer element, are added together. The sum is iteratively calculated and stored for each pixel.

3) *Data Manipulation*

After pulse compression and delay sum beamforming have been performed, the data must be manipulated into a form which can be displayed as an image. This process is achieved in three steps: log compression, time gain compensation, and envelope detection.

The logarithm of each data point is taken prior to imaging. A logarithmic calculation is necessary to increase the magnitude of weak scatterers relative to the magnitude of stronger scatterers, so that more detail is present in the final image.

As pressure waves pass through tissue, they are attenuated. The further a wave travels through tissue, the more it is attenuated. Therefore, voltages corresponding to pressure waves which are received later in time are more attenuated. To display all features of the image equally regardless of depth, compensation must be performed for the attenuation based on the time-of-flight of the wave in tissue. The equation for compensation for a data point is given in (3), where C is the compensation, D is the depth of the point being imaged, F is the probe frequency, and A is the attenuation, which is equal to 1 dB/cm/MHz (twice the average attenuation of tissue as the signal must travel the distance to the scatterers and back). For our system, the transducer's center frequency was estimated at 8.3 MHz.

$$C = ADF \qquad (3)$$

Envelope detection is equivalent to rectification followed by low pass filtering. To calculate the envelope of each signal, the absolute value of the Hilbert transform on the data is determined. The Hilbert transform equation is described in (4), where *sgn(x)* is the sign of *x*, and *N* is the number of points in the discrete Fourier transform of the input signal [12]. The vector *H(k)* is multiplied by the discrete Fourier transform of the input signal to perform envelope detection.

$$H(k) = -i\,\mathrm{sgn}(\frac{N}{2} - k)\,\mathrm{sgn}(k) \qquad (4)$$

D. Resolution Enhancement Compression (REC)

REC is an example of a coded excitation which could be tested using the completed platform. This coded excitation was used in simulations discussed in Section IV. REC is technique which uses a pre-enhanced chirp to increase the bandwidth of the transducer. By doing so, the axial resolution of the image can be enhanced.

The pre-enhanced chirp is calculated using convolution equivalence. The process is displayed in Fig. 10. First, a model for the pulse-echo impulse response of an ultrasonic transducer is created. Then, a transducer with the same center frequency but an increased bandwidth is modeled. In this project, the improved transducer had 150% of the original transducer's bandwidth of 8.3 MHz. Next, a linear chirp is generated, containing frequencies in the range of the bandwidth of the improved transducer. Then the linear chirp convolved with the improved impulse response is equated to the pre-enhanced chirp convolved with the original impulse response. In this manner, the pre-enhanced chirp can be determined.

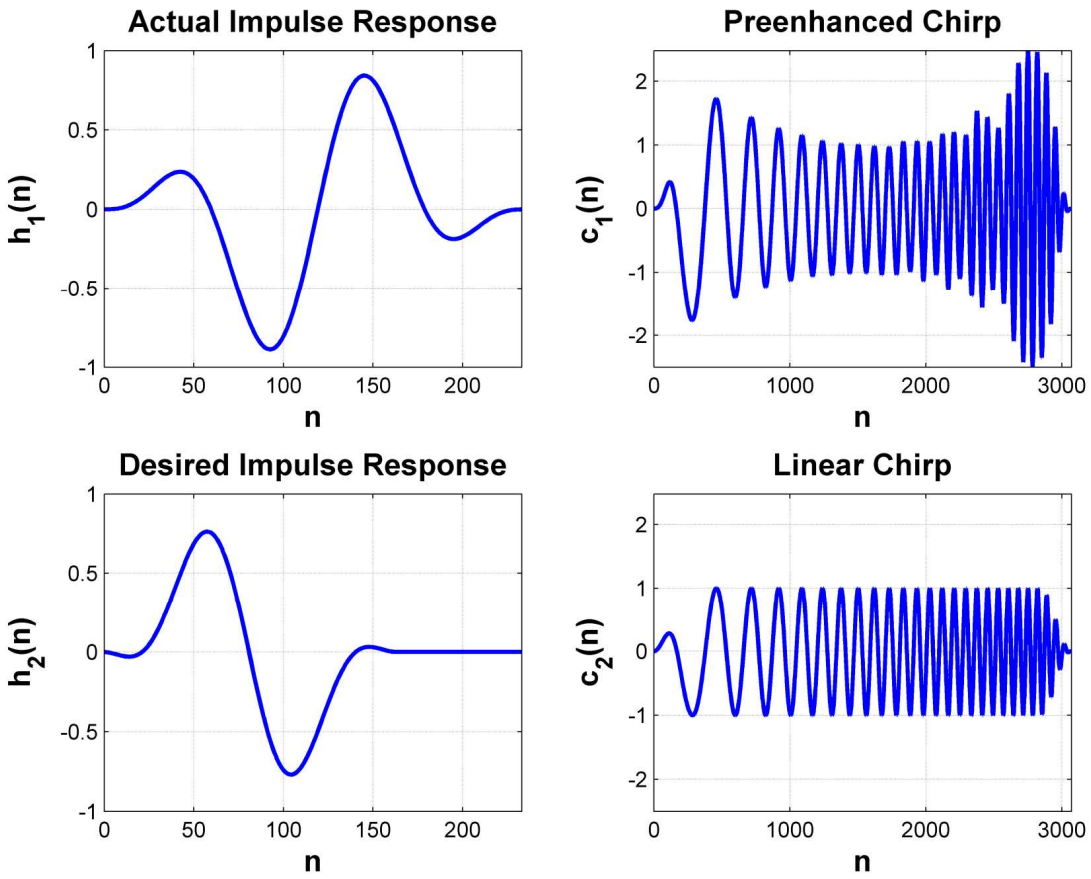

Fig. 10. Transducer pulse-echo impulse response (top left), REC chirp (top right), improved pulse-echoimpulse response (bottom left), and linear chirp (bottom right)

IV. SIMULATION RESULTS

A. Amplifier Simulations

Simulations were conducted to test the amplifier designs discussed in section III-C. The H-bridge design took the difference of two nodes to produce the amplified signal. The simulation was run as slower speeds to prove the concept. After the H-bridge amplifier was designed, it was discovered that the component supplied would not be able

to run at the specified 1GHz that was mentioned in the Fall. This specification is no longer a requirement and was replaced with 500MHz.

The push pull amplifier provided two signals that could be used to obtain a final signal that would be sent to the ultrasonic transducer. The first signal was at node A in Fig. 6. This signal was -100 V to 0 V. The second signal was received from node B in Fig. 6. This was a 0 V to 100 V. These two signals provided the full range of voltage needed, according to the original requirements, in the final output. Again, as with the H-bridge, these signals came from two different nodes. One possibility to get the needed signal was to design a passive summer. Unfortunately, this summer was not designed in time before the final design change for the amplifier was suggested.

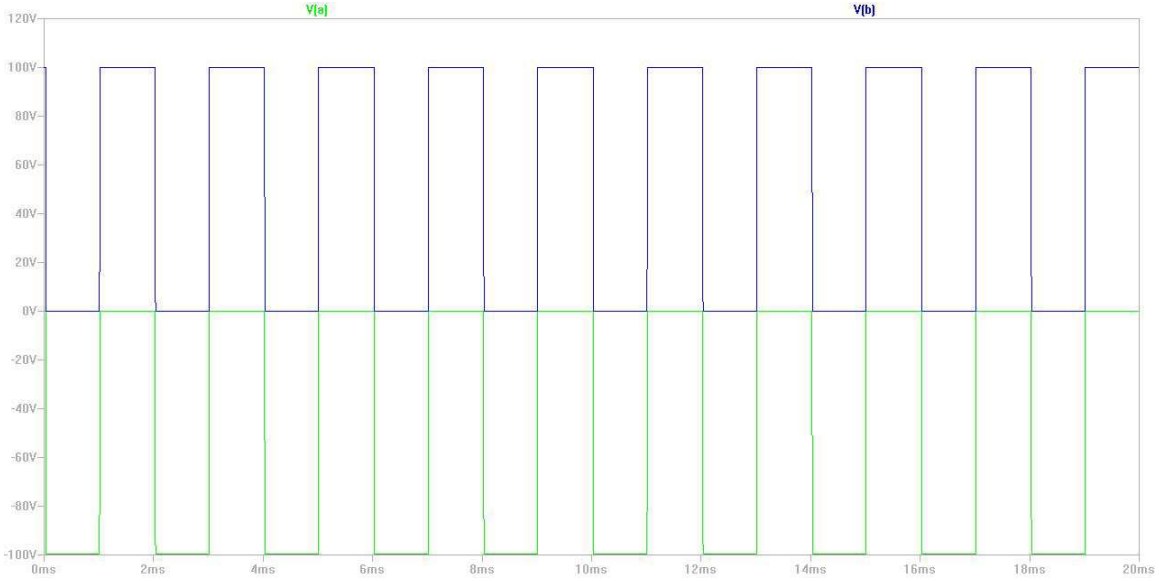

Fig. 11. Simulation of Push-Pull Amplifier Design

The single-ended amplifier output is shown in Fig. 12. The figure shows the output before the transducer model was simulated. The output is the required -50V to 50V.

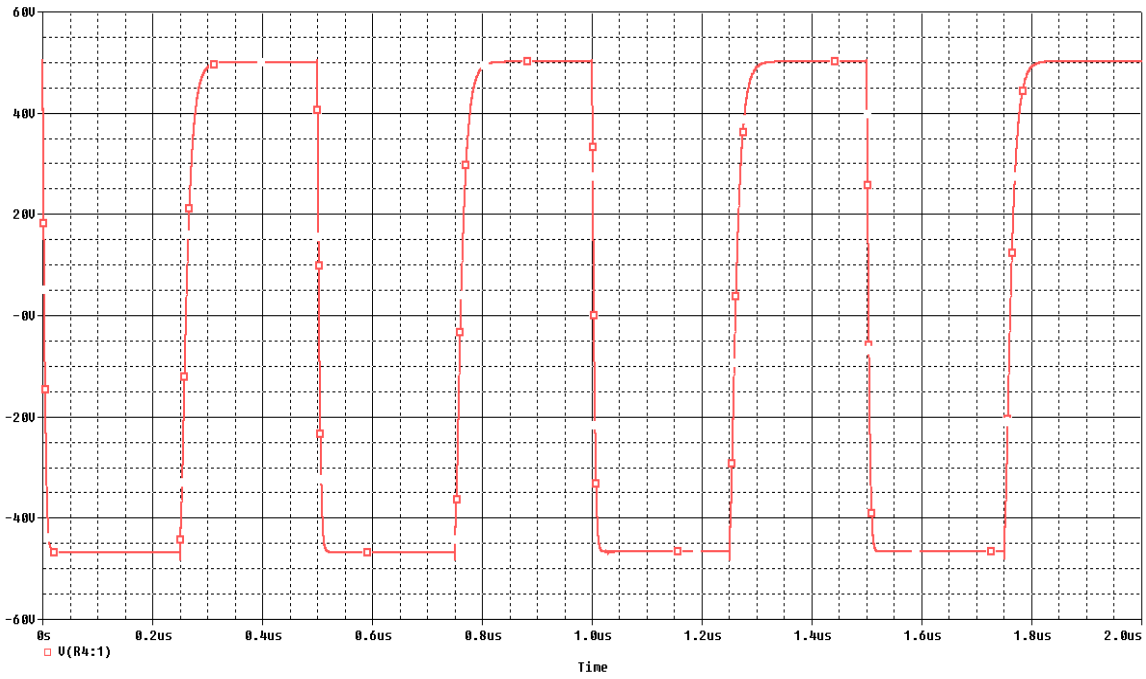

Fig. 12. Simulation of Single-Ended Amplifier w/o Transducer Model

The loading effects of the transducer model hindered the design of the amplifier. One way to counteract the loading effects of the transducer would be to do load matching. This matches the impedance of the amplifier with the impedance of the transducer. This would also ensure maximum power transfer.

B. Data Processing Simulations

Simulations were conducted to test the data processing code. These simulations were performed using Field II software in MATLAB [13]. This software models an ultrasonic transducer and point targets. The transducer was set to have a center frequency of 8.3 MHz and a fractional bandwidth of 100%, to approximate the ultrasonic transducer which will be used with the completed system. The probe consists of eight linearly-arranged channels, to simulate a prototype multi-channel system. The phantom was

simulated as having reflective points arranged in a row along a line which is perpendicular to the transducer. The points were placed every 10 mm in axial distance from the transducer. An illustration of this configuration is shown in Fig. 13. A signal-to-noise ratio of 46 decibels was used in the simulations.

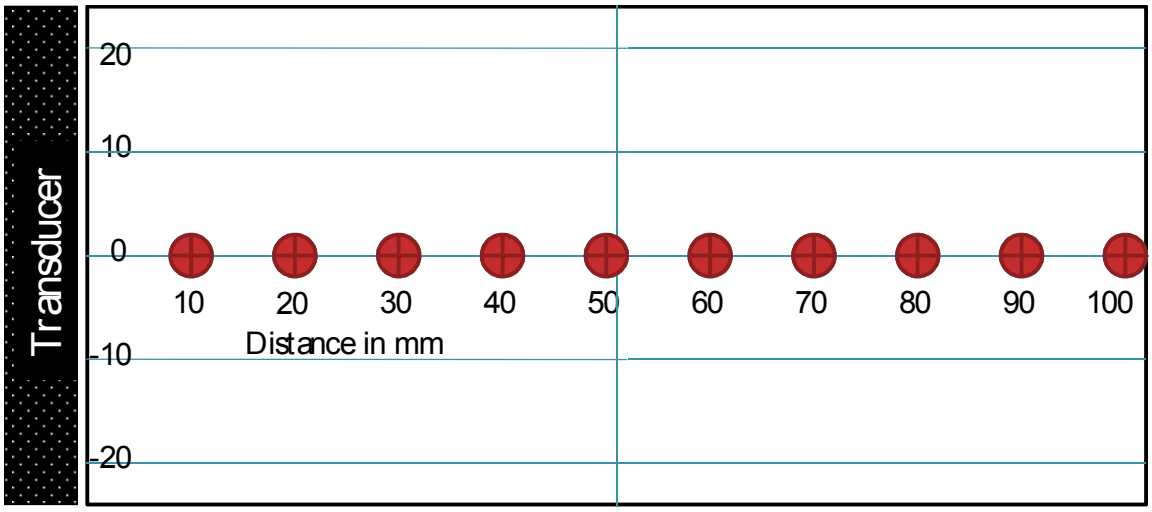

Fig. 13. Field II Configuration

Simulations were performed using two different excitations. The first was a short pulse as is conventionally used in medical ultrasound. The second was the REC pre-enhanced chirp described in Section III-D. Field II software generated the simulated captured data. This data was then processed as described in Section III-C. The resulting image created using the conventional excitation is shown on the left in Fig. 14, and the image created using the REC excitation is shown on the right in Fig. 14.

The benefits of coded excitations can be observed from Fig. 14. The REC excitation, along with pulse compression using a Wiener filter, resulted in far less noise in the final image. For depths less than 10 mm, all noise in the REC image is less than -60 dB in magnitude. In the conventional image noise is present with -50 dB magnitude for the same depth. The REC image also has less noise for greater depth, allowing points

to be visible at up to 70 mm in depth. The impulse generated image only shows points for depths up to 50 mm. At depths of 60 and 70 mm, the noise present in the image conceals the point sources.

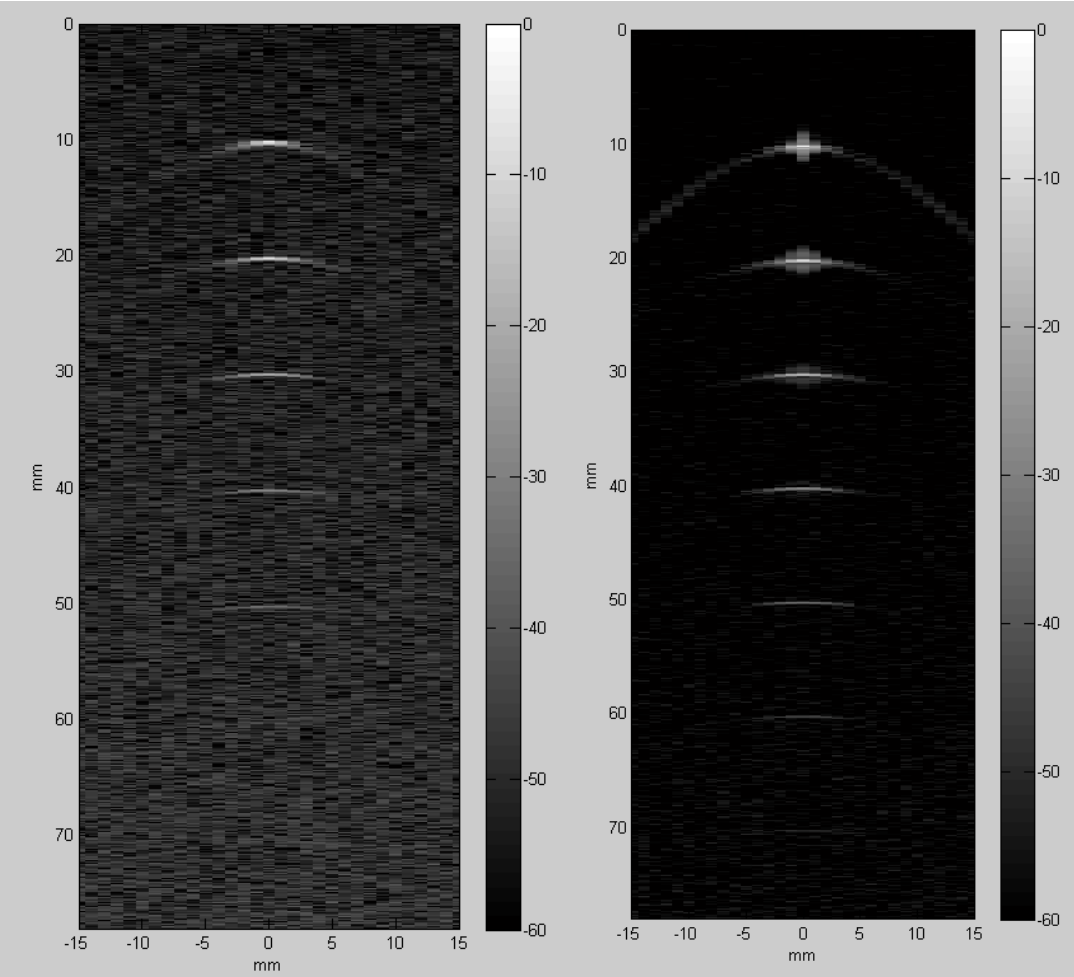

Fig. 14. Data Processing Simulation Images Created using conventional pulse excitation (left), and REC chirp coded excitation (right)

V. EXPERIMENTAL RESULTS

An experiment was conducted to validate the accuracy of the system in transmitting arbitrary waveform. Fig. 15 shows the procedure for this experiment.

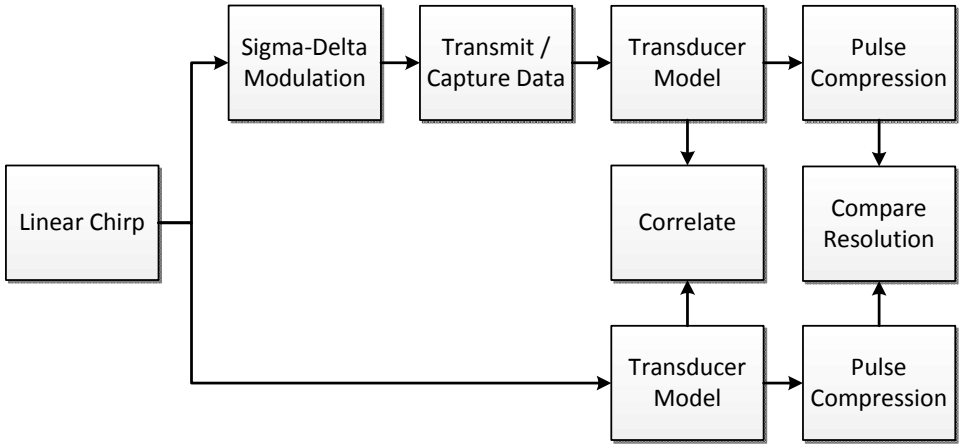

Fig. 15. Transmitting Experiment Procedure

A linear chirp that spans the frequencies of 4 MHz to 12 MHz, as shown in Fig. 16, was selected for testing as it is a complex frequency modulated coded excitation used in ultrasonic research.

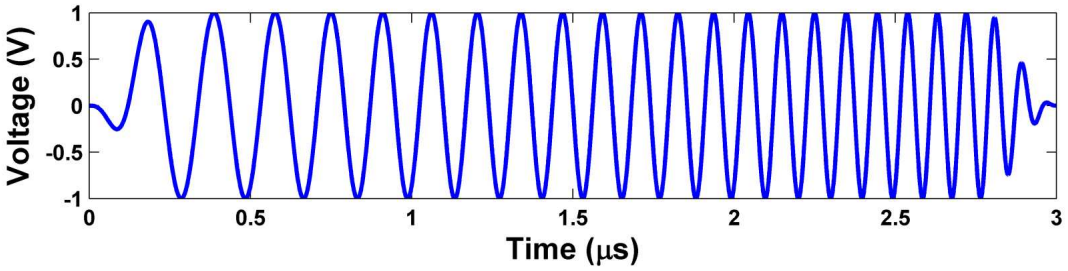

Fig. 16. Linear chirp spanning 4 MHz to 12 MHz

This waveform was encoded using sigma-delta modulated, as shown in Fig. 17, and transmitted by the FPGA.

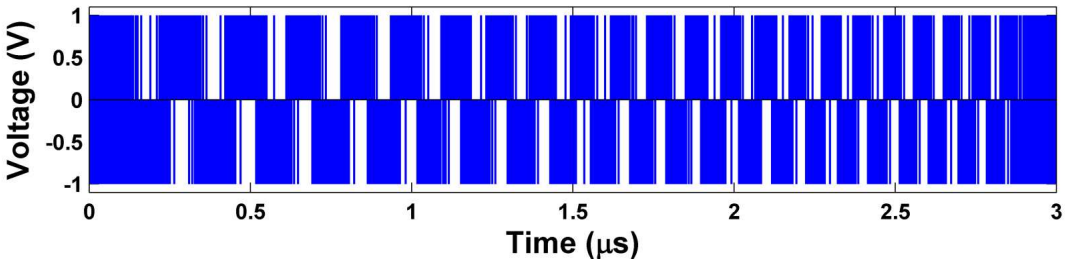

Fig. 17. Sigma-delta modulated, linear chirp

The pin output of the sigma-delta modulated, linear chirp was captured using a high-speed digital oscilloscope (Lecroy, Chestnut Ridge, NY), and is shown in Fig. 18. The FPGA characteristics cause output noise, and the bandwidth of the oscilloscope converts the delta pulses from the sigma-delta modulated form to a continuous signal.

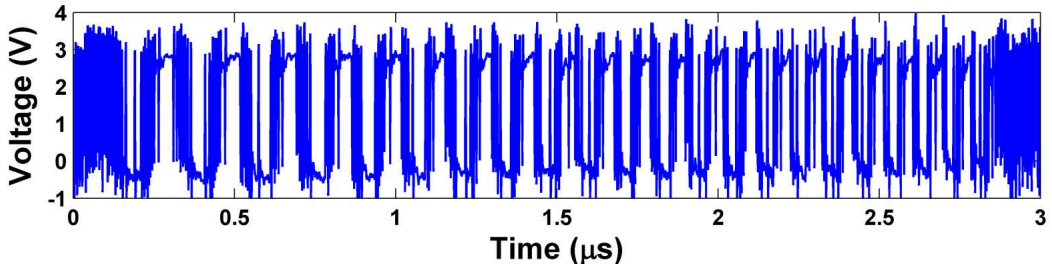

Fig. 18. Captured, sigma-delta modulated, linear chirp from an FPGA output pin

A model for the frequency response of the transducer, shown in Fig. 19, was constructed in MATLAB using information from transducer's datasheet. This model has a center frequency of approximately 8 MHz and a fractional bandwidth at -6 dB of approximately 100 %.

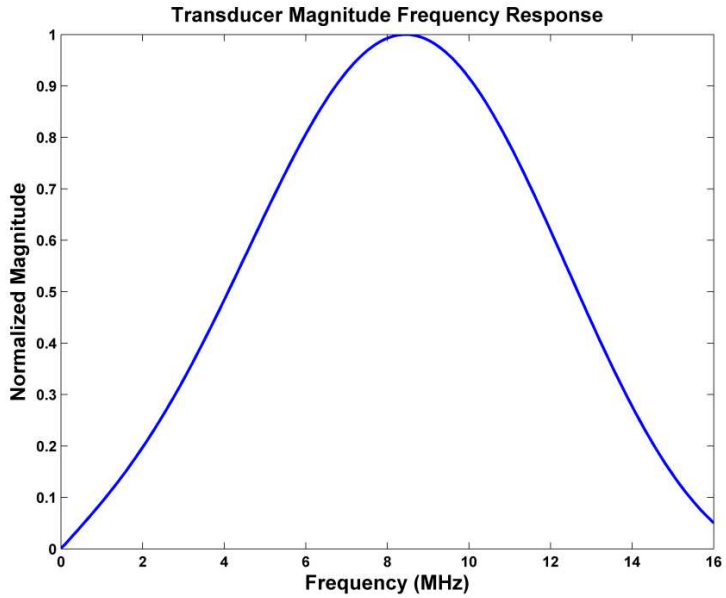

Fig. 19. Transducer frequency response model

The transducer frequency response model was used to filter the linear chirp, the sigma-delta modulated chirp, and the captured, sigma-delta modulated, linear chirp from an FPGA output pin. The resulting signals were calculated to have correlations listed in Table I. These correlations suggest that there is less error between the linear chirp and the sigma-delta modulated, linear chirp compared to the error between the stored, sigma-delta modulated chirp in the FPGA DDR2 memory and the captured, sigma-delta modulated, linear chirp from an FPGA output pin.

Table I. Cross correlation between filtered, linear chirp; filtered, sigma-delta modulated, linear chirp; and filtered, captured, sigma-delta modulated, linear chirp

Cross Correlation	Filtered, Sigma-delta Modulated, Linear Chirp	Filtered, Captured, Sigma-delta Modulated, Linear Chirp
Filtered, Linear Chirp	99.84%	99.33%
Filtered, Sigma-delta Modulated, Linear Chirp	-	99.53%

Despite the 0.67% error, the signal retains enough accuracy to prove the concept of using sigma-delta modulation to transmit the properties of the linear chirp. This can be seen in Fig. 20, which shows the comparison between the filtered, linear chirp and the filtered, captured, sigma-delta modulated, linear chirp in both the time and frequency domains.

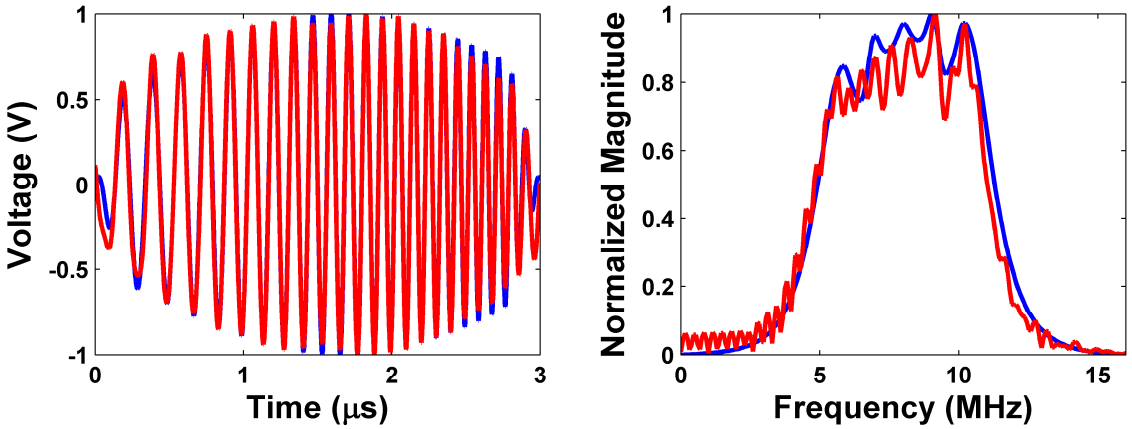

Fig. 20. Filtered, linear chirp (blue) and captured, sigma-delta modulated, linear chirp (red) in the time domain (left) and the frequency domain (right).

VI. CONCLUSION

Based upon the results of the transmitting experiment, waveforms which are transmitted by the FPGA are valid for ultrasonic imaging, and can be almost identically reconstructed to the same coded excitation as the original waveform after being passed through the ultrasonic transducer. Therefore, this system could be used to perform ultrasonic research into the use of coded excitations to improve the diagnostic capabilities of medical ultrasound.

REFERENCES

[1] J. A. Zagzebski, *Essentials of Ultrasound Physics*, St. Louis, MO: Mosby, 1996.

[2] T. Misaridis and J. A. Jensen, "Use of modulated excitation signals in medical ultrasound. Part I: basic concepts and expected benefits," *IEEE Trans. Ultrason., Ferroelect., Freq. Contr*, vol.52, no. 2, pp.177-191, Feb. 2005.

[3] M. L. Oelze, "Bandwidth and resolution enhancement through pulse compression," *IEEE Trans. Ultrason., Ferroelect., Freq. Contr*, vol. 54, no .4, pp. 768-781, Apr. 2007.

[4] J. A. Jensen et al., "Ultrasound Research Scanner for Real-time Synthetic Aperture Data Acquisition," *IEEE Trans. Ultrason., Ferroelect., Freq. Contr.*, vol. 52, no. 5, pp. 881–891, 2005.

[5] S. Ricci et al., "Multichannel FPGA-based arbitrary waveform generator for medical ultrasound," *Electronics Letters*, vol. 43, no. 24, pp. 1335-1336, Nov. 2007.

[6] R. Schreier, *The Delta-Sigma Toolbox Version 7.3*. Analog Devices, Inc, 2009.

[7] W. D. O'Brien et al., "Threshold Estimation of Ultrasound-induced Lung Hemorrhage in Adult Rabbits, and Comparison of Thresholds in Mice, Rats, Rabbits and Pigs,"*Ultrasound in Medicine and Biology*, vol. 32, pp. 1793-1804, 2006.

[8] R. Schreier and G. C. Temes, *Understanding Delta-Sigma Data Converters*, John Wiley & Sons, Inc., 2005.

[9] Zetex, "Complementary enhancement mode MOSFET H-Bridge," ZXMHC10A07N8 datasheet, March 2009.

[10] Blatek, Inc., private communication, February 2011.

[11] J. R. Sanchez et al., "A Novel Coded Excitation Scheme to Improve Spatial and Contrast Resolution of Quantitative Ultrasound Imaging," *IEEE Trans. Ultrason. Ferroelectr. Freq. Control,* vol. 56, no. 10, pp. 2111-2123, Oct. 2009.

[12] M. Johansson. *The Hilbert Transform* [Online]. Available: http://fuchs-braun.com/media/d9140c7b3d5004fbffff8007fffffff0.pdf

[13] J.A. Jensen. *Field: A Program for Simulating Ultrasound Systems, Medical & Biological Engineering & Computing*, pp. 351-353, vol. 34, 1996.

APPENDIX A: How to Operate the ULTRA Transmitter

A: Introduction

The following guide describes the steps necessary to use the developed platform from waveform creation, to transmission, to processing received data. The platform will send any correctly formatted waveform to the pins of the FPGA, and from there the captured experimental data can be passed through a simulation of the ultrasonic transducer. This guide also describes how these results can be verified for accuracy by comparison to the theoretical waveform transmission. To follow this guide, access to the Ultra site on Sakai is necessary.

B: First Time Setup

1. Obtain the Delta Sigma Toolbox from the Ultra site on Sakai under *Resources > Sigma Delta Modulation > Delta Sigma Toolbox > delsig.zip*
2. Extract the delsig.zip to a convenient location on the hard drive. Remember this location.
3. Download and save the SigmaDeltaGenerator.m code from the Ultra site under *Resources > Repository > MATLAB_files > Sigma Delta Modulation > SigmaDeltaGenerator.m*
4. Open the SigmaDeltaGenerator.m file and edit the 5th line to reflect the location of the delsig toolbox (shown in Fig. A1). This must point to the directory that directly contains the MATLAB files. Save the file.
5. Download and extract the FPGA_GUI to a convenient location from the Ultra site under *Resources > Repository > FPGA_GUI.zip*
6. Download and save the ultrasound.bit file from the Ultra site at *Resources > Repository > FPGA > ultrasound.bit*

C: Waveform Generation

1. Open the SigmaDeltaGenerator.m file and replace the code in the EXAMPLE block to the desired input analog waveform to use in the sigma delta modulation, as shown in Fig. A1. Make sure the waveform is exactly 1500 samples in length. This guide will use the example code given (note that Fig. A1 does not reflect this).
2. Change any parameters in the OTHER PARAMETERS section as necessary.

```
% change to reflect location of the delta sigma toolbox
addpath 'C:\temp\ultrademo\delsig'

% set the name of the output file here (.wve extension preferred):
WAVEFORM_NAME = 'ones.wve';

%%%%%%%%%%%% OTHER PARAMETERS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
order = 2;
osr = 16;                       % oversampling rate
freq = 500*10^6;                % frequency of fpga board
                                % sampling frequency = freq/osr
per = 3e-6;                     % period of chirp signal
N = freq*per;                   % Number of sample points - should be 1500
n = 0:N-1;                      % n points
t = per*(n / N);                % time points

fb = 4000000;                   % initial frequency ~3.526MHz
fe = 12000000;                  % final frequency ~12.874MHz
mult = fb*per;                  % multiplier for time of frequency change
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%% Set the input waveform here %%%%%%%%%%%%%%%%%%%%%%%%
% must be 1500 samples in length

x = ones(1,1500);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```


Fig. A1: Updates to the SigmaDeltaGenerator.m file

3. Change the name of the output waveform file by setting the WAVEFORM_NAME variable to the desired name. Using a .wve extension will make sending the waveform to the FPGA easier later on.

4. Click the *Run* icon to generate the waveform. The waveform is saved in a file located in the same directory as the SigmaDeltaGenerator.m code.

D: FPGA Preparation

1. Obtain the Virtex-5 XUPV5-LX110T Evaluation Platform and connect two SMA cables as shown in Fig. A2.

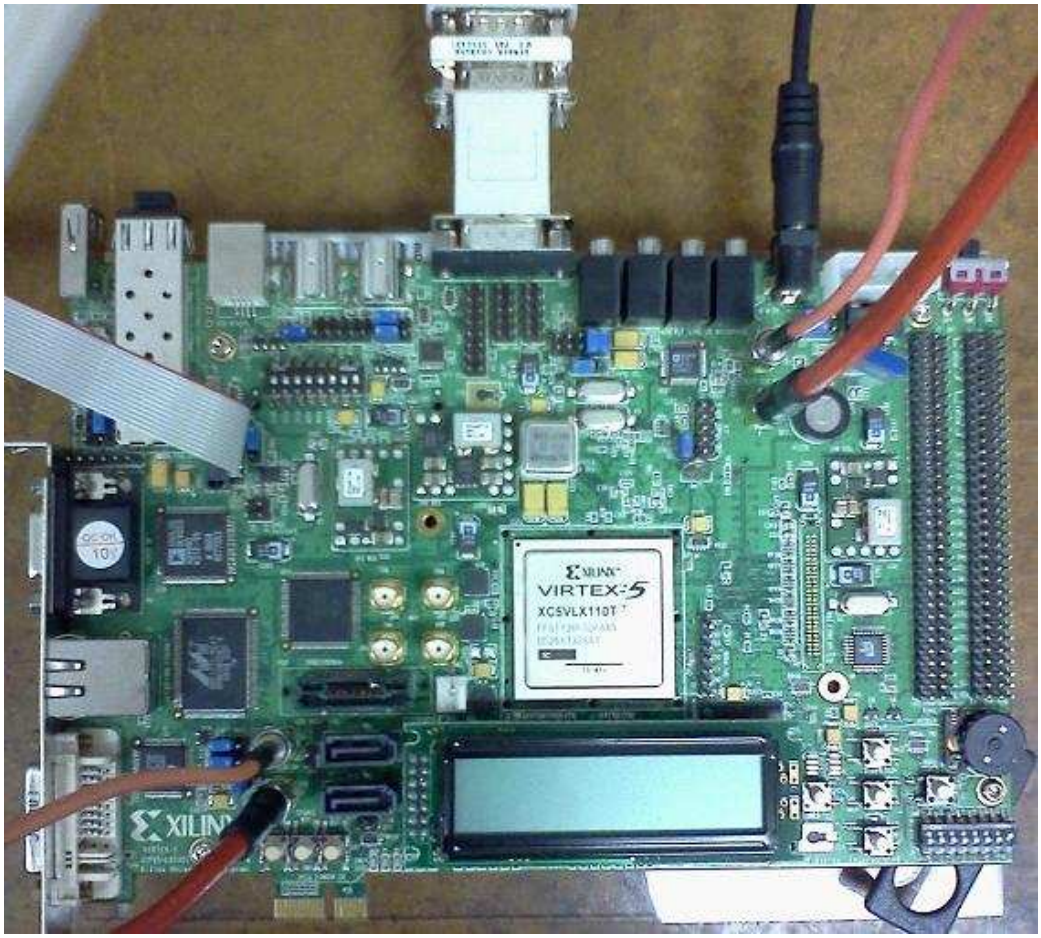

Fig. A2: FPGA Hardware Setup

2. Connect the power to the FPGA.
3. Connect the JTAG cable to the FPGA and to an available USB port on the PC as shown in Fig. A2.

4. Connect the PC's serial cable to the FPGA using a null-modem adapter. This is also shown in Fig. A2.
5. Power on the board with the switch at the top right.
6. Open up Xilinx iMPACT by following *Start > EE Applications > Xilinx ISE Design Suite 11 > ISE > Accessories > iMPACT*
7. Press OK if the iMPACT Access Permission box appears.
8. Select No for the "Automatic Project File Load" box as shown in Fig. A3.

Fig. A3: Automatic Project File Load.

9. Select No to the "Automatically create and save a project" as shown in Fig. A4.

Fig. A4: Automatically create and save a project.

10. Select Cancel for the "New iMPACT Project" box.
11. Double click the Boundary Scan button under iMPACT Flows, shown in Fig. A5.

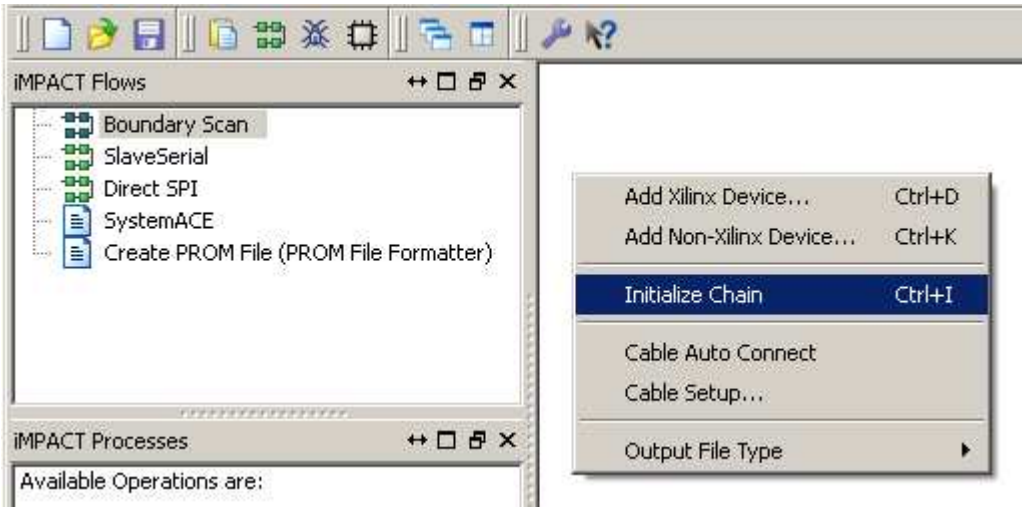

Fig. A5: Initialize Chain

12. Right click in the Boundary Scan area, then select Initialize Chain, as shown in Fig. A5.
13. Press Esc four times to close the first four "Assign New Configuration File" boxes.
14. On the fourth "Assign New Configuration File" box, navigate to the location of the ultrasound.bit file. Select it, and click *Open*. This is shown in Fig. A6.

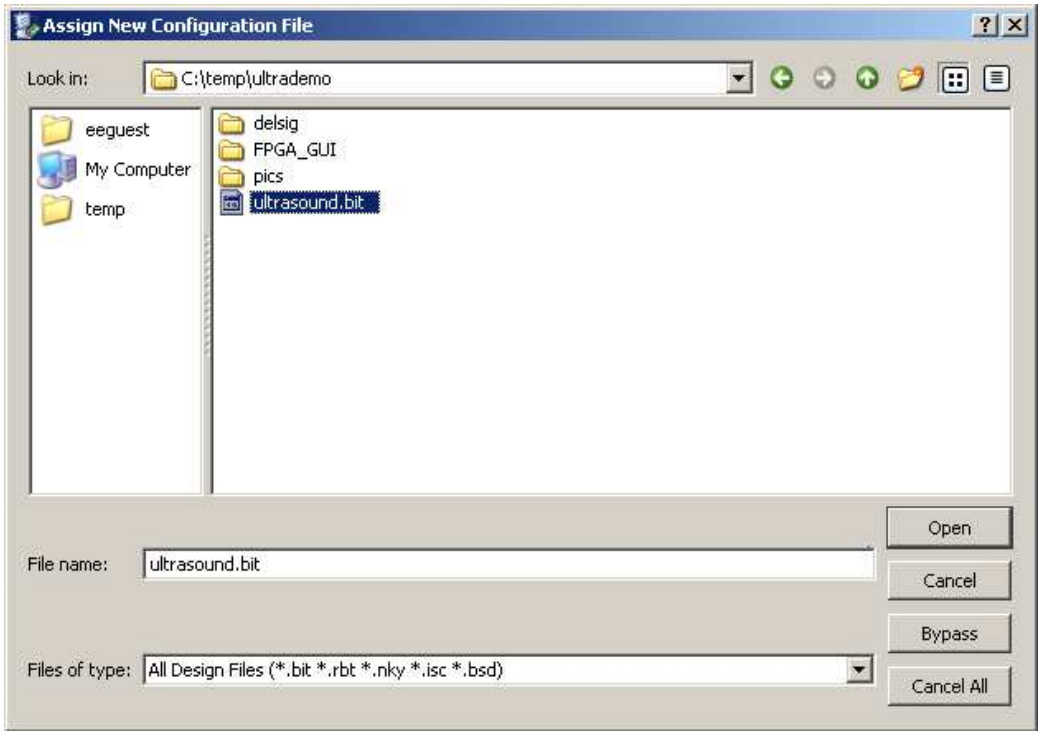

Fig. A6: Assign New Configuration File.

15. Select *No* to the "Attach SPI or BPI PROM" box.
16. Select *OK* to the "Device Programming Properties – Device 1 Programming Properties" box.
17. Right click the fifth Xilinx chip icon and select *Program*, as shown in Fig. A7.
18. Click *OK* to the "Device Programming Properties – Device 5 Programming Properties" box.
19. After the status bar reaches 100%, a blue box should appear with the words "Program Succeeded." If there is a problem, this box will be red and say "Program Failed."

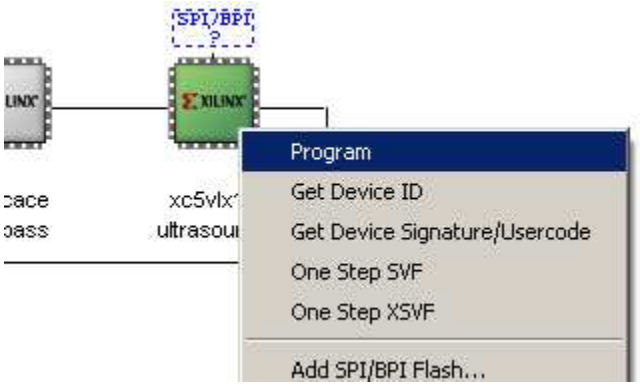

Fig. A7: Program the FPGA board

E: Transmission

1. Open the waveform.exe program located in the FPGA_GUI\waveform\bin\Debug directory.
2. Select the *Add* button shown in figure A8 and navigate to the waveform file (.wve). Open it.

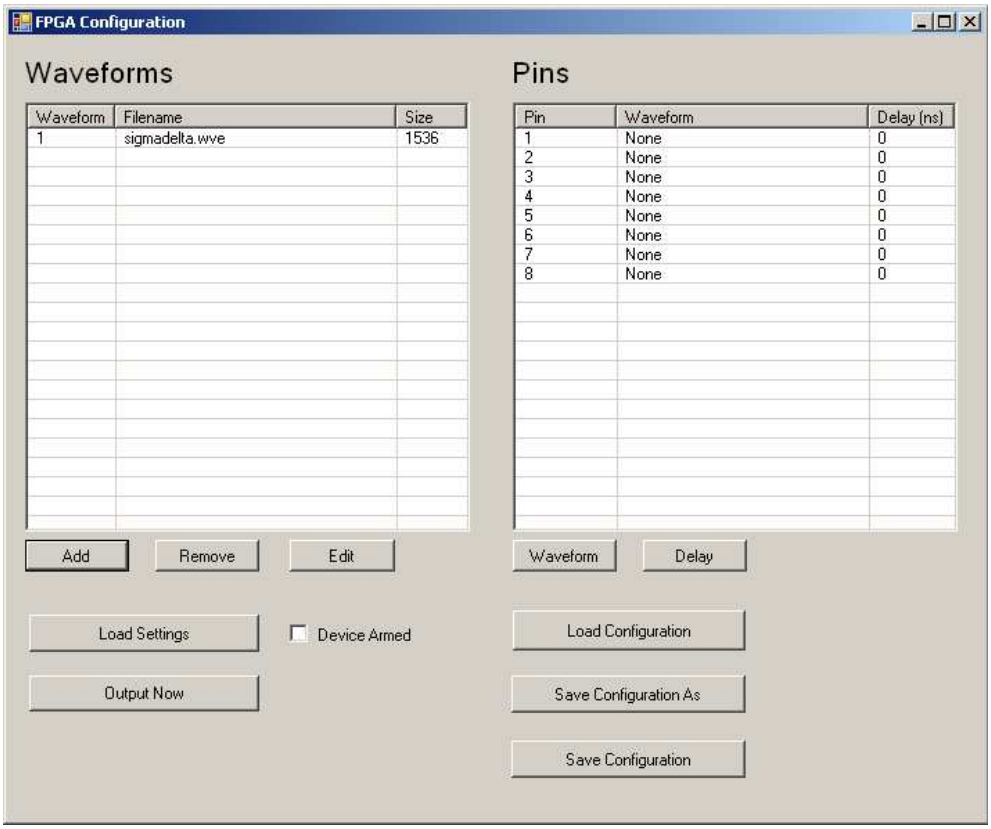

Fig. A8: FPGA GUI

3. Select the desired pins to send the waveform to (multiple pins can be selected). For this guide, select pin 2.
4. Select the Waveform button and assign the waveform index 1 to pin 2. This is shown in Fig. A9. Press OK.

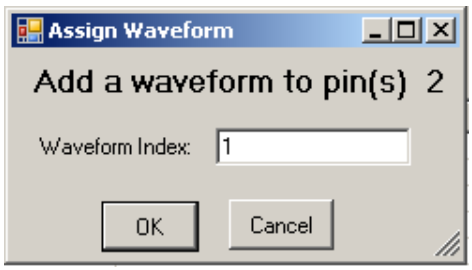

Fig. A9: Assign Waveform.

5. Optionally assign a delay by pressing the *Delay* button.

6. Press the *Load Settings* button shown in Fig. A8. The "Device Armed" checkbox should now be checked. If an error occurs, press ok to the error message and retry. Note: sometimes resetting the FPGA will help with repeated errors. Press the west button as shown on Fig. A10 to reset the FPGA.

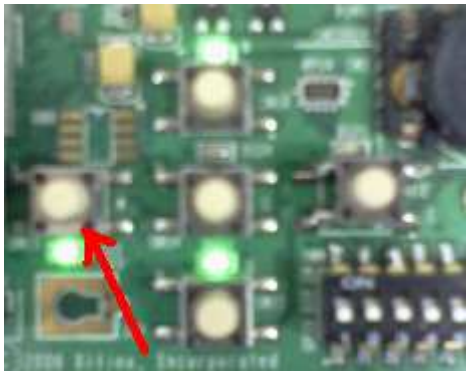

Fig. A10: Reset FPGA Button

7. Plug in the power to the LeCroy Oscilloscope; set the on/off switch on the back to on; Remove the cover and press the power button.

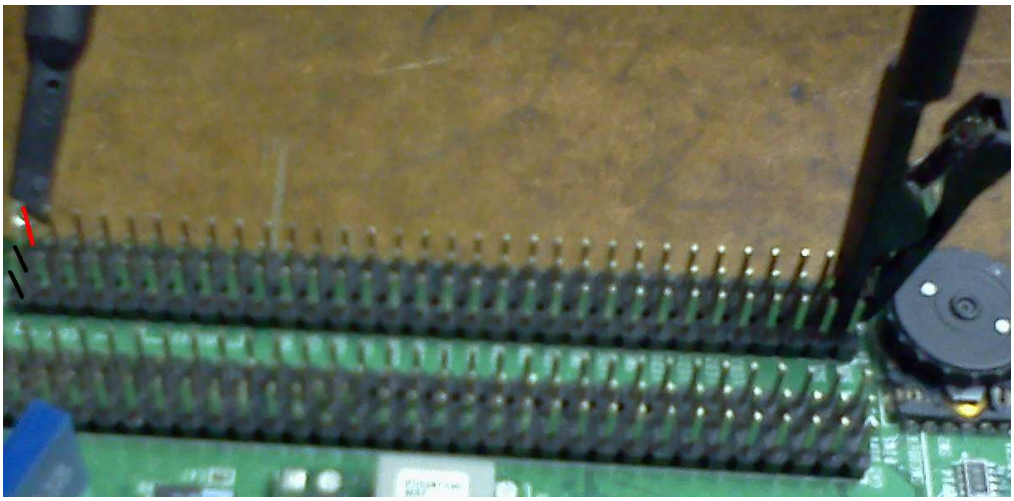

Fig. A11: Probe Connections to Pins

8. Connect the probe to channel 1 and then to the FPGA pins as shown in Figs. A11 and A12. The ground connector can connect to any of the middle column of pins, but the bottom is most convenient.

Fig. A12: Probe Connection to Pin Close Up

9. Right click the C2 Waveform box shown in Fig. A13 and select *Off* to remove it from the screen.

Fig. A13: Turning Ch. 2 Off

10. Change the horizontal scale to 500 ns/div and the vertical scale to 1.00 V/div as shown in Fig. A14.

Fig. A14: Oscilloscope Buttons and Wheels

11. Move the C1 waveform lower on the screen by scrolling the *Vertical Offset* scroll-wheel counter-clockwise. This is shown in Fig. A14. The waveform should be on the bottom quarter of the screen.
12. If needed, adjust the trigger to between one and two volts.
13. Press the Single button shown in Fig. A14.
14. Press the *Output Now* button on the PC to transmit to the oscilloscope (shown in Fig. A8). The waveform should now be visible on the screen, as shown in Fig. A15.

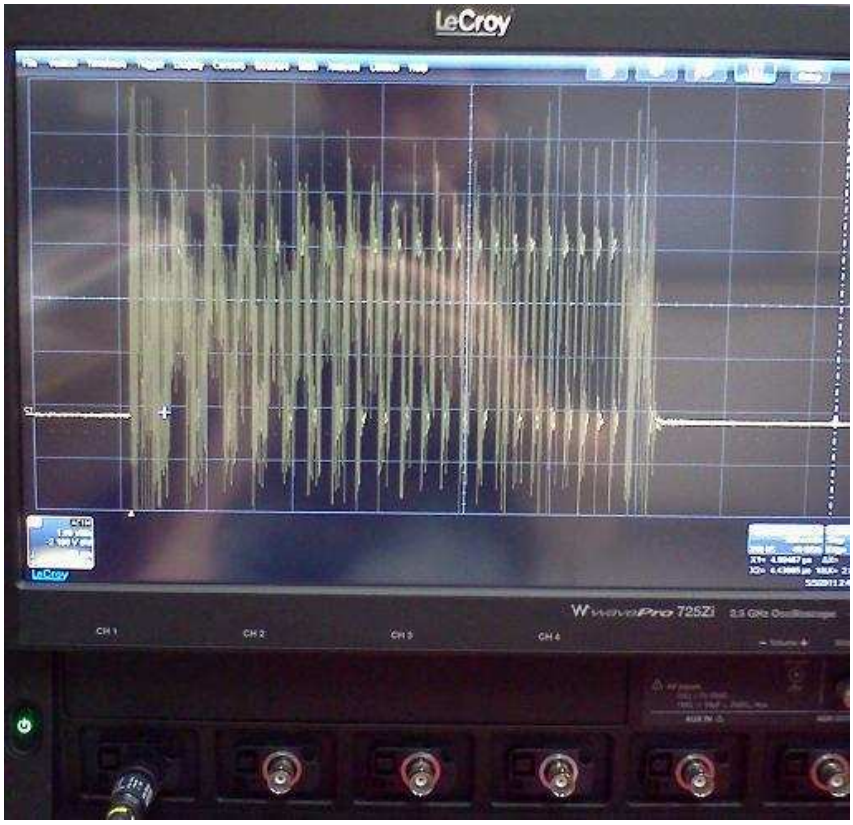

Fig. A15: Waveform Displayed On Oscilloscope

15. Insert a thumb drive to a USB port on the oscilloscope.
16. Select *File* > *Save Waveform*
17. Select *Browse* and locate a place to store the waveform. Press *OK*.
18. Press *Save Now!* Remove the thumb drive. Note: to safely remove, select *File* > *Minimize*, then click the device icon on the bottom right.

F: Data Processing

The data processing section of this tutorial works for the linear chirp example inside the EXAMPLE block of the SigmaDeltaGenerator.m file. It would need to be adjusted if any other waveform was used.

1. Download and save the chirp_filt_pc.m file from the Ultra site at *Resources* > *Repository* > *MATLAB_files* > *Chirp_Reconstruction* > *chirp_filt_pc.m*
2. Double click the chirp_filt_pc.m file to open it up with MATLAB.
3. Move the captured waveform data file to the same location chirp_filt_pc.m is stored.
4. Edit the *DATA_FILE* variable on line 14 of chirp_filt_pc.m to match the name of the waveform data file. The name will be similar to 'C1t00000.dat' or 'C1Trace00000.dat'.
5. Click the *Run* icon to process the data. The created figures will show the transducer frequency response and a comparison of the experimental and simulated chirps in the time domain, frequency domain, and after pulse compression.

APPENDIX B: How to Create a New PCB Footprint

A. Introduction

This tutorial will walk the user on how to create a footprint for the TX810 from Texas Instruments (TI). This footprint will be made using OrCAD PCB Designer. The footprint can then be used in many different PCB layout tools. This part is a special package that has no leads outside the package. Therefore, some accommodations will need to be made when designing the part. The datasheet for this part can be found at: http://focus.ti.com/lit/ds/symlink/tx810.pdf

B. Create Padstack

1. Go to Allegro Pad Designer: *Start* > *EE Applications* > *OrCAD 16.3* > *PCB Editor Utilities* > *Pad Designer*. Click *OK* for the pop up windows.
2. The following window will appear:

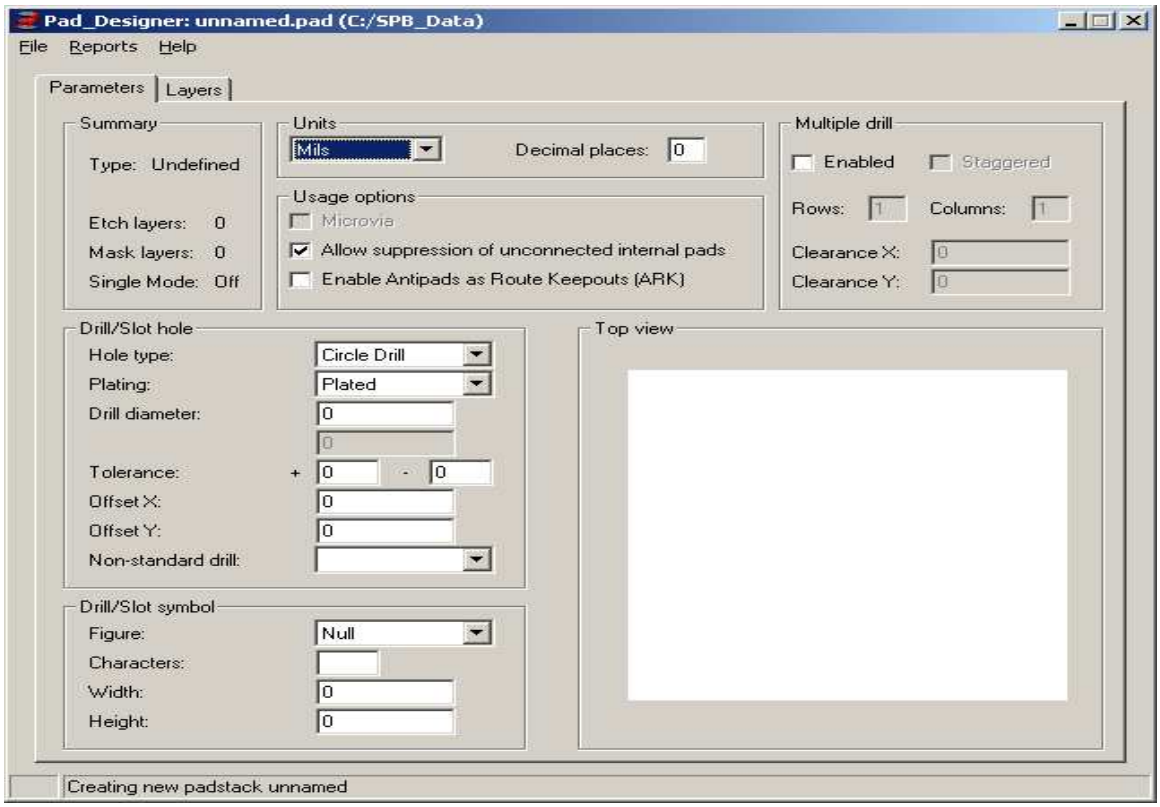

Fig. B1 Pad Designer Window, Parameters Tab

3. Change the units to *Millimeter* because the datasheet gives dimensions in millimeters. Change the decimal places to 4. Also change the Hole Type to *Oval Slot*. Do not change the slot size or any other value. Since this is not a thru-hole component, we do not need to drill.
4. Next go to the *Layers* tab.

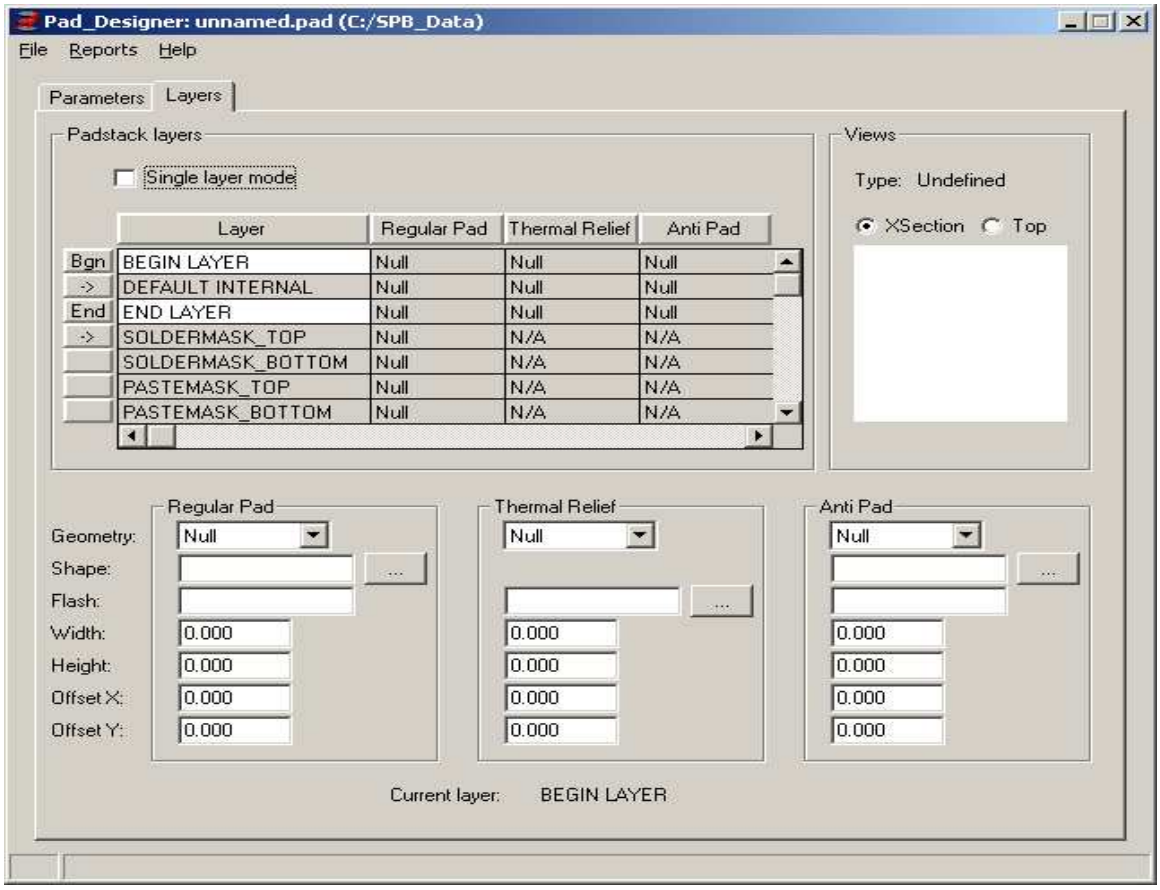

Fig. B2. Pad Designer Window, Layers Tab

5. In the *Layers* tab, the layers and dimensions need to be adjusted. First, select the box next to *Single Layer Mode.* Again, as this in a Surface Mount Device (SMD), we only need the top layer.

6. Change the Geometry to *Oblong*. Next, adjust the Width to 0.55, and Height to 0.24. These measurements were obtained from the datasheet. The height is half the tolerance that they give (0.18-0.30). Same for the width. (0.45-0.65)
7. Right click on *Bgn* to the left of BEGIN LAYER. Select *Copy*. Then right click on the "->" and click *Paste.*
8. Finally, save your padstack. *File* > *Save As...* Click the x box to exit the warnings box and select *Yes* to the pop up box that asks if you want to save with warnings. Create a new folder for your padstack. (C:\Documents and Settings\eeguest\Desktop\Footprint) Name the padstack *TX810.pad* in this folder.
9. Exit out of that window.
10. Repeat this process, only for this second padstack, change Width to 0.75 and Offset X to -0.1. Save this padstack as *TX810-1.pad*. This padstack will some into play much later.

C. New Design

1. Go to OrCAD PCB Editor: *Start* > *EE Applications* > *OrCAD 16.3* > *OrCAD PCB Editor.*
2. Select *OrCAD PCB Designer with PSpice* from the pop up box. Click *OK.*
3. After the window loads. Go to *File* > *New…* This box should pop up:

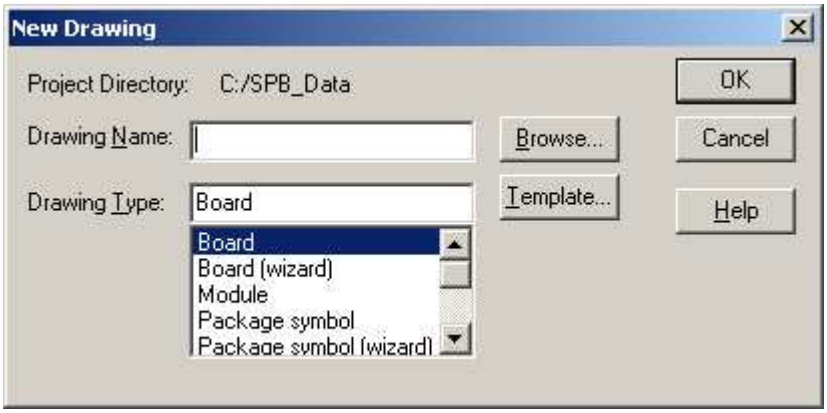

Fig. B3 New Drawing Window

4. Click *Browse* and place the drawing in the same folder the padstack is located.

   (C:\Documents and Settings\eeguest\Desktop\Footprint)

5. Give the drawing a name. *TX810.dra* will be fine. Click *OK*.

6. The next box to pop up should look like this:

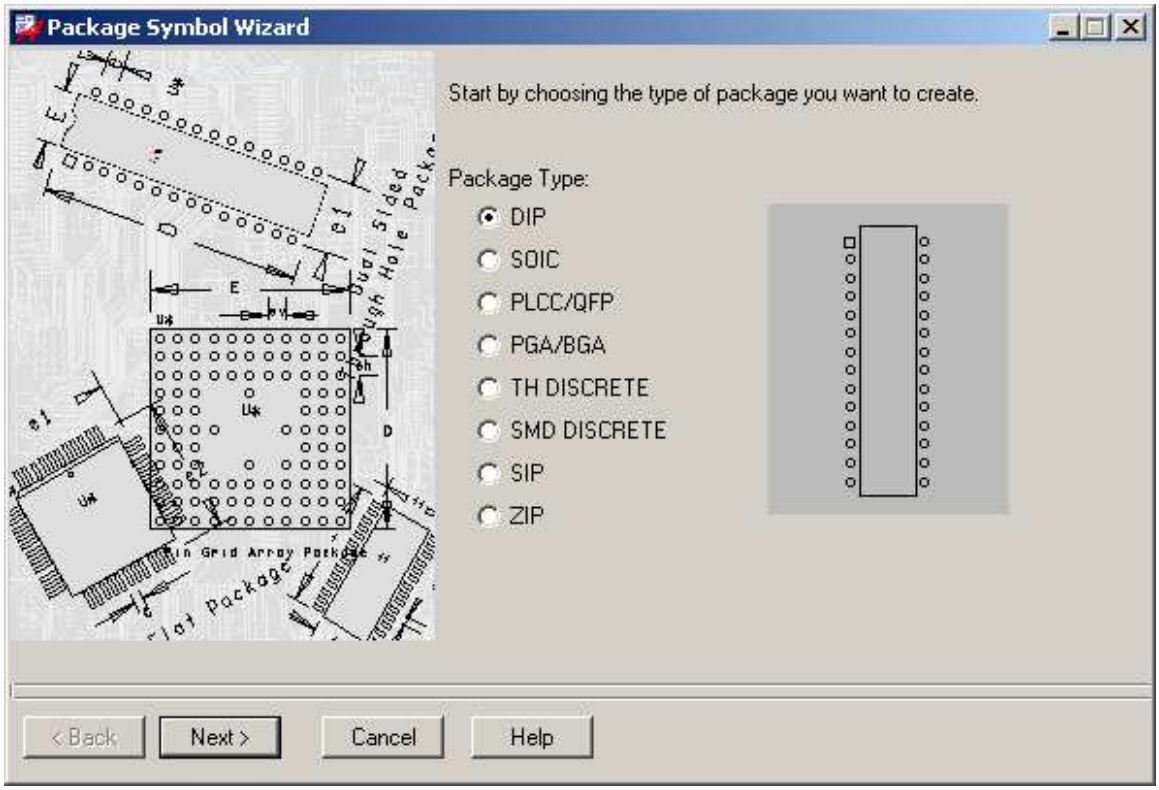

Fig. B4 Select Package Window

7. The TX810 is a special package called QFN (Quad Flat Pack- No Pins). Since this wizard does not offer this package, we will select *PLCC/QFP* this package type is very similar to the QFN. Hit *Next.*
8. On the next screen, simply click *Load Template.* Click *Next.*
9. The datasheet gives dimensions of the component in millimeters, so in the next window, select *Millimeters* from the two drop down boxes. The accuracy will automatically set to 3. This is fine. Click *Next.*
10. The next window will let the user select how many pins are on each side, and the distance between each pin (pitch). Enter the information in the next image, Click *Next* after the changes have been made.

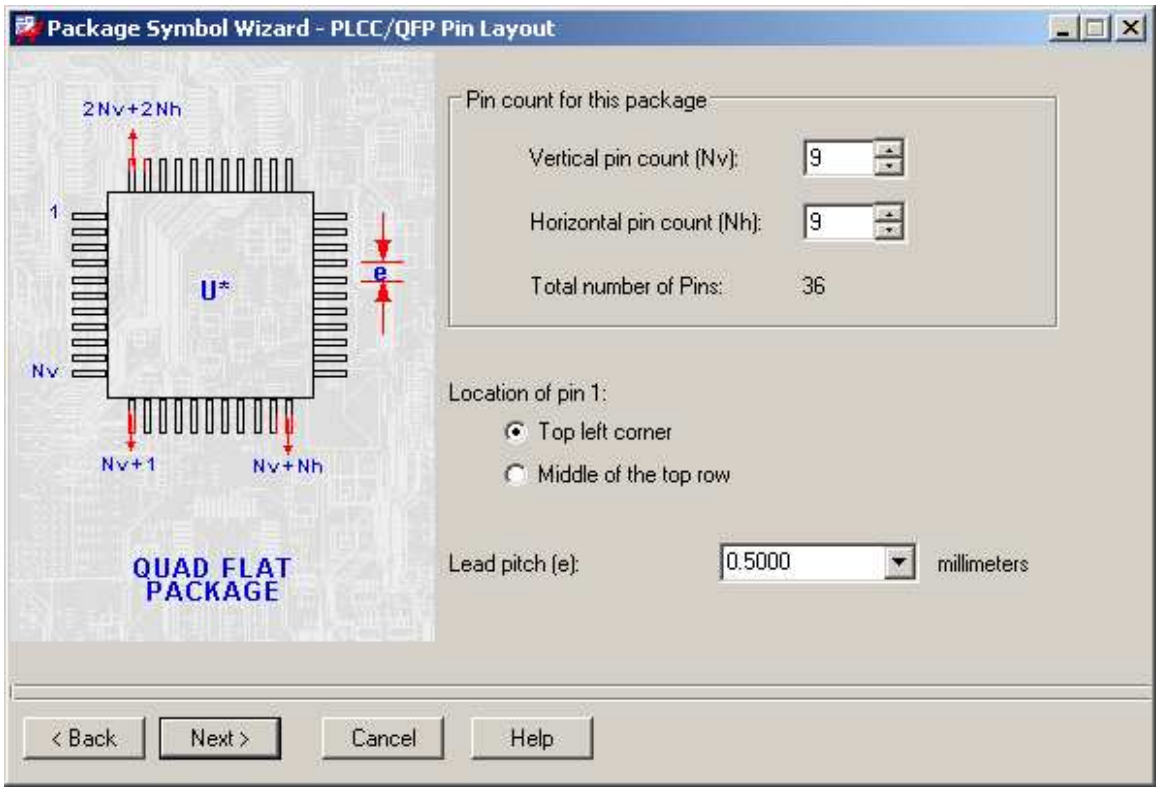

Fig. B5 Pin Count and Lead Pitch Window

11. The next window allows the user to adjust the dimensions of the package. These can be found in the datasheet, with a little simple math. Since the leads

for this package are under the package, we need to subtract the length of the pin from the width of the package (6mm-0.55mm=5.45mm). The reason for using 0.55mm is because that is width of the padstack. Keep in mind, the footprint is not the same as the part itself; this is the solder pads that are placed on the board. Therefore, e1=e2=E=D=5.45mm. Hit *Next.*

12. The next window will ask for the padstack for symbol pins and the padstack for pin 1. This is the padstack created in part B. Select the Tx810 for both boxes. Click *Next.*

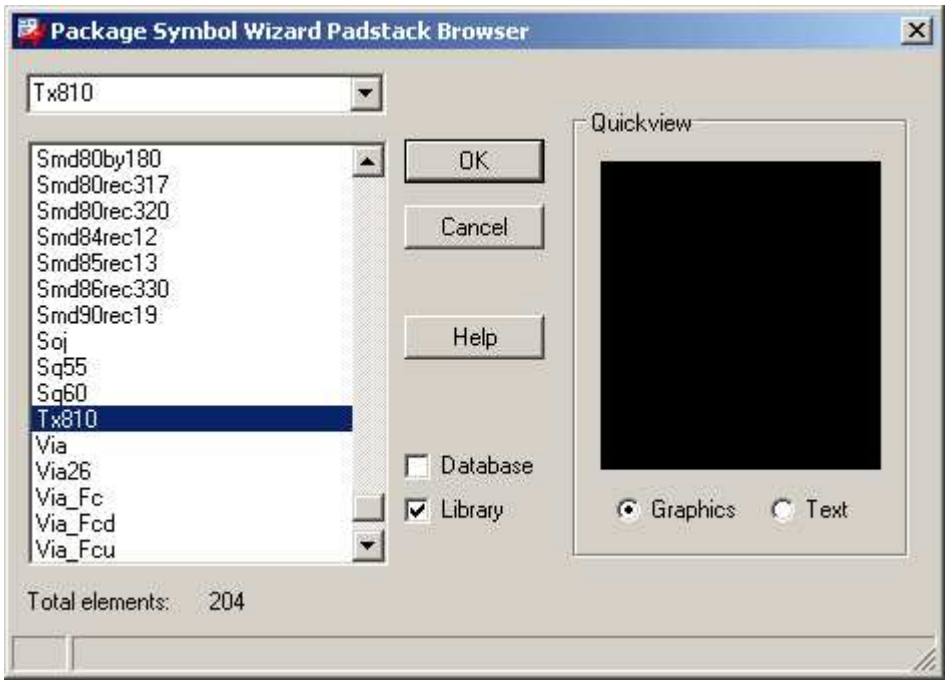

Fig. B6 Select Padstack Window

13. Next, select *Pin 1 of symbol*. Also, make sure that *Create a compiled symbol* is selected. Click *Next.* Click *Finish.*

C. Modify Pins

1. Next, we will need to adjust the padstack to accommodate for QFN specifications set by TI. They suggest overlapping the padstack .2mm

OUTSIDE the end of the package. Your part should look something like Fig. B7.

2. To do this, click *Tools* from the top menus. Select *Padstack*. Then click on *Modify Design Padstack.*
3. Next, on the right side of the screen, there should be a tab that says *Options*. Hover the mouse over this and a column will pop out. Double click on *TX810*.
4. The padstack design window will pop up. Change the Width to 0.75 and OffsetX to 0.1. Then repeat step 7 in the padstack design by right clicking on *Bgn* to the left of BEGIN LAYER. Select *Copy*. Then right click on the "->" and click *Paste.*

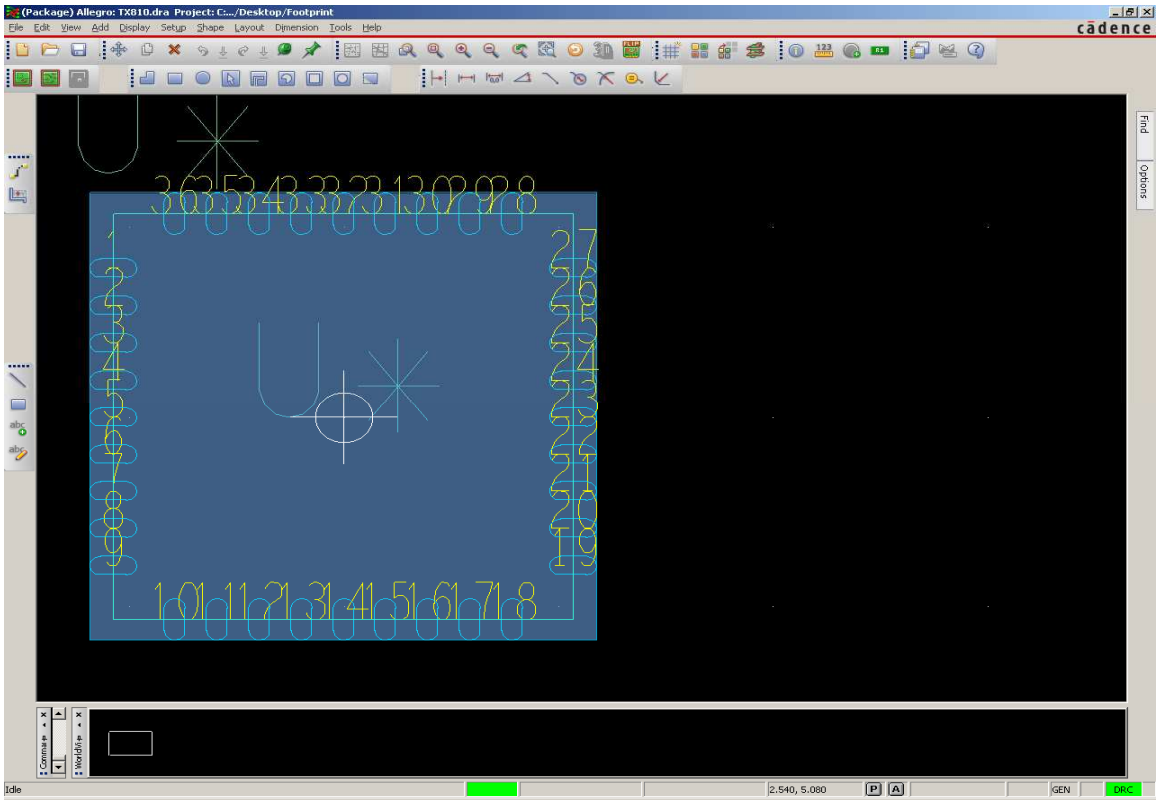

Fig. B7 TX810 Footprint

5. After you have pasted the layer, go to *File* > *Update to Design*. Exit the first pop up and then click *Yes* in the next pop up as you did before. Also, go to *File* > *Save As*…and save the padstack. Then exit out of the padstack screen.
6. It's obvious that the package is not symmetric. Right click in the black somewhere and select *Done.* Next, click on pin 1. Then click *Tools* from the top menus. Select *Padstack*. Then click on *Replace.* Hover the mouse over the *Options* tab again. This time, it should pop out with a column like Fig. B8. Change *Pin #* to 1, and select Old as TX810, and New as TX810-1 (the alternate made before). Click *Replace*. Pin 1 should shift.

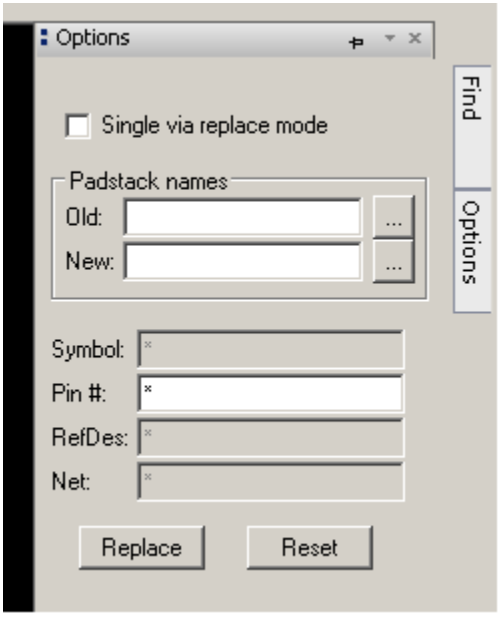

Fig. B8 Replace Options Tab

7. Repeat step 6 for pins 2-18. Make sure each time you change *Pin #* to the desired pin. Otherwise, it will replace ALL the pins.
8. Right click in the black somewhere and click *Done.* Save the footprint. The final footprint should look like this:

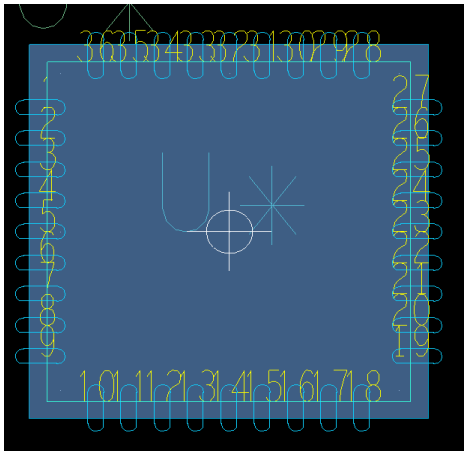

Fig. B9 Final Footprint for TX810