Ultrasound Beamforming: FPGA Implementation and Dynamic Focusing

By

Sara Shakil Qureshi
2009-NUST-MS-EE(S)-26

Supervised By

Dr. Kashif Mahmood Rajpoot
Assistant Professor

This thesis is submitted in partial fulfillment of the requirements for the degree of Masters of Science in Electrical Engineering (MS EE)

School of Electrical Engineering and Computer Science,
National University of Sciences and Technology (NUST),
Islamabad, Pakistan.

December 2012

APPROVAL

It is certified that the contents of thesis document titled, "Ultrasound Beamforming: FPGA Implementation and Dynamic Focusing" submitted by Miss. Sara Shakil Qureshi have been found satisfactory for the requirement of degree.

Advisor: Dr. Kashif Mahmood Rajpoot

Signature: _____

Date: _____

Committee Member1: Dr. Steven Freear___

Signature: _____

Date: _____

Committee Member2: Dr. Shahzad Younis_

Signature: _____

Date: _____

Committee Member3: Dr.Sohail Iqbal____

Signature: _____

Date: _____

To my parents and teachers

CERTIFICATE OF ORIGINALITY

I declare that the research work titled **"Ultrasound Beamforming: FPGA Implementation and Dynamic Focusing"** is my own work to the best of my knowledge. It contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at SEECS or any other education institute, except where due acknowledgment is made in the thesis. Any contribution made to the research by others, with whom I have worked at SEECS, University of Leeds or elsewhere, is explicitly acknowledged in the thesis.

I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic is acknowledged. I also verified the originality of contents through plagiarism software.

Author's Name: Sara Shakil Qureshi

Signature: _____

ACKNOWLEGMENTS

I am grateful to Almighty Allah who gave me courage and strength to complete this thesis. I am really thankful to Dr. Kashif Mahmood Rajpoot for his kind attention and guidance during this thesis. I specially thank Dr. Steven Freear for his continuous supervision and guidance in testing the design using Ultrasound Array Research Platform (UARP) at University of Leeds, UK. I am also thankful to my worthy committee members, Dr. Shahzad Younis and Dr. Sohail Iqbal, for their support and becoming a part of this work. I am extremely gratified to Peter Smith, Dr. David Cowell and the other Ultrasound group members in University of Leeds, UK in helping me during the design and testing of the software and hardware implementation of the system. Finally, I am thankful to all who had helped me in any form during all this period.

Table Of Contents

Table Of Tables

Table Of Figures

ABSTRACT

Beamformer is a core component in ultrasound systems, which stands as a signal processing unit forming a beam in a desired direction from interfering signals at each channel by adjusting their weights and delays. Upon reception by the transducer, the signals are sampled using high speed analog to digital converters. These samples are then beamformed using a delay profile generated for a single focal point. Implementing the beamformer in hardware for efficient frame rates, typically required in medical applications, is a crucial task involving handling of data sampled at high sampling frequency, parallel processing and inter-medium bandwidth constraints. This thesis investigates the implementation of a fixed focus delay and sum beamformer on the Ultrasound Array Research Platform (UARP) configured at University of Leeds, UK. Furthermore, fixed focus beamforming often results in poor image quality with focus fixed at one pixel location only. Dynamic focusing is an alternative approach where every pixel in the image is focused. This research also presents a study of the dynamic focus delay and sum beamformer to enhance the image quality and signal to noise (SNR) ratio.

In this work, feasibility study is performed, and architecture is proposed for performing beamforming in real time. Such architecture can be implemented on FPGAs and can be fabricated into ASICs. The proposed system can be installed with any Ultrasound system being modular; hence it removed the platform dependency.

This work resulted in implementation of a real time FPGA based delay and sum beamformer for Ultrasound system (UARP). The hardware replacement of the software based beamformer in UARP increased its frame rate to about 10 times while further optimization provided frame rate as high as 32 frames per second as compared to only 1 frame per second initially. Also, the similar design was studied for dynamic focusing instead of fixed focusing and it was observed that the signal to noise ratio (SNR) increased as compared to the fixed focus beamforming. The image resolution also increased which enhances the system output in terms of image quality.

Chapter 1:

Introduction

Ultrasound systems stand as a very important way of analysis, testing and measurement in medical procedures. This chapter highlights the overall ultrasound system structure with detailed discussion on the receive side with beamformer as the topic of emphasis. It also introduces the document breakdown in terms of its chapters.

1.1 Ultrasound system and its components

Ultrasound systems use sound waves above the normal audible range of humans (>20 kHz) for analyzing different objects. These waves are transmitted from the ultrasound system using pulse generators. These waves get reflected back from the target object or artifacts in the focused region and thus the reflection pattern gives information/details about the object. Medical science uses ultrasound-based diagnostic imaging for visualizing subcutaneous body structures like tendons, tissues, etc. Also, ultrasound systems are now being used for delivery of drugs i.e. micro-bubbles are utilized in ultrasounds for medical diagnostics and drug delivery.

Ultrasound system contains the following basic components as shown in the Figure 1.

- Ultrasound transmit system.
- Ultrasound receive system.

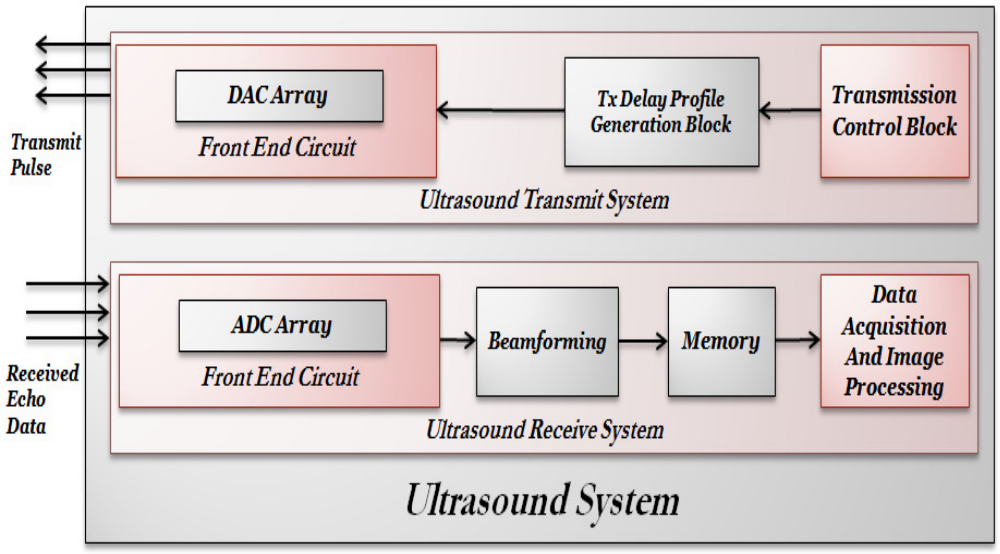

Figure 1 - Ultrasound system and its components

1.1.1 Ultrasound transmit system

Ultrasound transmit system contains a transmission control block, transmission delay profile generation block and the front end transmission circuit.

Transmission control block generates the transmission specific control signals and also inputs the data from the user for the transmission. Transmission specific parameters include setting the frequency, the aperture size, the focal depth, etc.

Transmission delay profile generation block inputs the required parameters from the transmission control block and then generates the delay profile to be applied to the elements included in the aperture. The delay profile is generated at runtime or may already be stored in a memory or lookup table.

The ***front end circuit*** contains the circuitry that converts the system level commands coming from the ultrasound transmit system data path to the waves to be transmitted. The front end circuit contains:

- Digital to analog converters.
- Pulse generators.
- Transducer elements.

Digital to analog converter converts the digital samples coming from the ultrasound transmit data path into analog signals. There are efficient digital to analog converters giving best approximation to the digital samples. The ***pulse generator*** converts the analog signals into pulses which are used to excite the transmission elements. An ultrasound system may have separate transmitters and receivers or they may have same elements being used as transmitters as well as receivers (one at a time). These are piezoelectric elements which oscillate when the electric pulses are passed through them. The elements which are used as transmitters as well as receivers are called ***transducers***. An ultrasound system may have many elements in total e.g. 32 to 512. But for a particular transmission, only a selected number of them are used. The selected number of elements, from the total number of elements in the system, used in a transmission is termed as

the ***aperture size*** of the system. A delay profile is generated to focus the elements of the aperture to the focal depth.

1.1.2 Ultrasound receive system

Ultrasound receive system contains a front end circuit, beamformer, memory and the data acquisition and image formation block.

Front end circuit in receptions contains circuitry which converts the input signals into digitized samples so that they can be processed in the hardware circuit. It contains the following as shown in Figure 2:

- Transducer/receiver elements.
- Time gain compensation.
- Analog to digital converter.

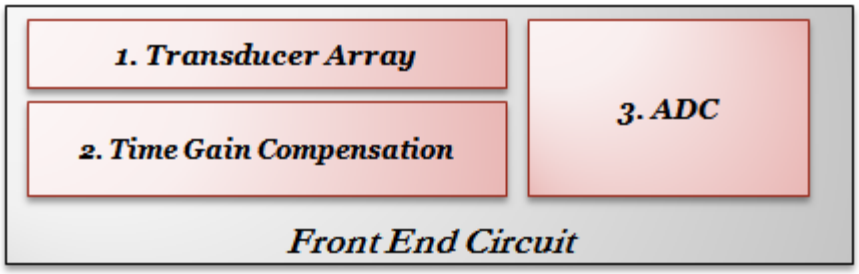

Figure 2 - Front end circuit and its components

An ultrasound system may have again elements which can be transmitters as well as receivers in the same system or there may be separate transmitters and receivers. Once the reflected waves are received by the ***transducer array of receivers***, the signals are passed through the analog to digital converters. ***Analog to digital converters (ADCs)*** convert the analog signals coming after reflections from the focal region into digital samples. These are efficient analog to digital converters giving best approximation to the analog signals. They are usually high sampling rate converters so that sufficient samples of the input analog signals are generated to form a best fit image. Sampling rate also results in improved resolution, which is the smallest possible physical quantity that can be represented in the system. Also, in the ADCs, there are low noise amplifiers (LNA) to do ***time gain compensation (TGC)***.

Waves in ultrasound attenuate more and more while moving deeper into the human body. In order to get the perfect discrimination between the shadow and deeper objects, the signals received must be amplified. As in an ultrasound system, the samples are approximation of distance in space. Thus, they represent time when they are being processed in the digital systems. So there is an analogue between distance and time in ultrasound setup. Thus, the time gain compensation circuits are present which amplify the signal greater with increasing depth in space.

Beamformer and memory

After digitizing the received signals, this data is to be aligned in the ultrasound system. This alignment is required so that they can be added coherently such that the energy reflected from a point in space gets combined correctly. The samples are to be added in the correct order to get best analysis of the imaging object or body. Thus, beamformer (as shown in Figure 3) plays its part to delay, weight and add these samples coherently to form scan lines focused at the focal depth. These scan lines which are beamformed data for the aperture size need to be stored till all the scan lines for the elements in the overall ultrasound system are created and stored.

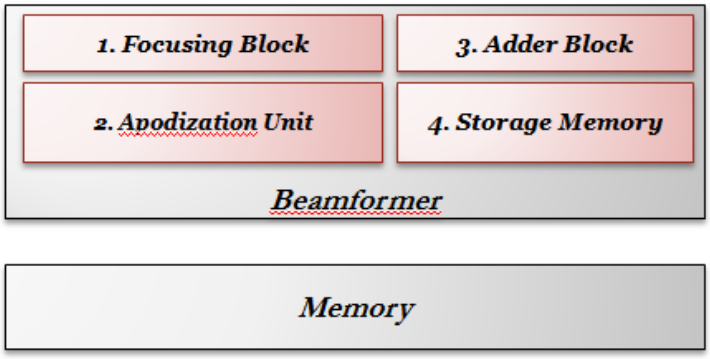

Figure 3 - Beamformer and its memory

Data acquisitions and image processing

Once the ultrasound system has all the scan lines ready, these scan lines are processed using signal processing techniques that include filtering, envelop detection, normalization and log compression in the data acquisition and image processing block as shown in Figure 4. After these

post-processing steps are applied, and the scan lines have been processed, the resultant image is plotted. The image contributes as a frame in the ultrasound system. The number of these frames that the ultrasound system can produce including the entire transmission and reception path is known as the ***frame rate***.

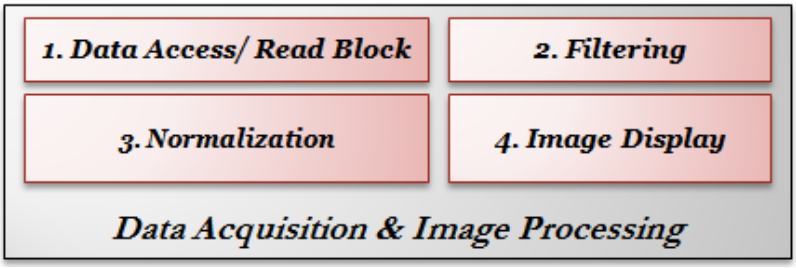

Figure 4 - Data acquisition and image processing block

The Ultrasound receive system and its basic components have been described above. This study focuses primarily on the receive system.

1.2 Beamforming

Beamforming is a signal processing technique which manipulates the signals in such a form that the noise and interference are processed to provide a valuable pattern. This pattern can be used for receiving signals at any desired location by adjusting the values of the delays and weights properly.

The conventional beamforming techniques captured data from a number of sensors/transducer elements. This data is processed in a coherent manner to form the final image [1]. Initially, these beamformers were present in analog domain. But the advent of very large scale integrated (VLSI) circuits and devices as Field Programmable Gate Array (FPGA) have resulted in systems which are low cost, small in size and high speed. Beamformers are now implemented with manipulations in digital domain, and thus utilize the FPGAs for low cost and high performance systems. In order to utilize the digital advantages, the signals are to be sampled using very fast analog to digital (ADC) converters. Till 1990's, the development of beamforming appears to be confined to the literature with little visible practical applications. This was basically due to the

lack of ADCs to sample at higher rates with fine precision. But now ADCs with very high sampling rates have been introduced which process significant number of bits [2].

Now-a-days, beamforming in digital domain has enhanced the use of it in many applications including ultrasounds and radar. Techniques have been developed to target the issues involved in enhancing the image quality and reducing the errors in approximations in digital domain. Digital implementations provide greater signal computational flexibility with better utilization of vast range of digital procedures proven much faster and cheaper with respect to resources and time.

The application discussed in this document is the use of beamforming in ultrasound systems.

Figure 5 shows an ultrasound receive system which consists of an array of transducer elements which receive the reflected signals. The signals received are amplified in the analog circuitry based on varying amplification considering more amplification for in-depth signals (Time Gain Compensation, TGC) and then finally these signals are sampled by the analog-to-digital converters at a higher sampling rate to efficiently approximate the image reconstruction.

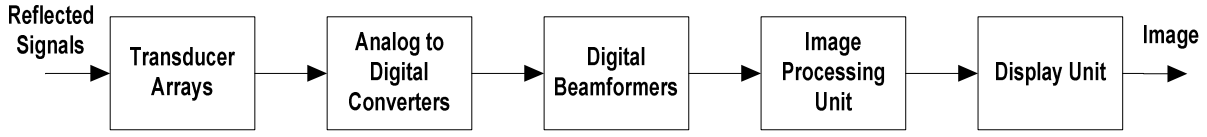

Figure 5 - Ultrasound system flow

After the ADC, digital beamformers are used. Initially analog beamformers [2] were used due to lack in development and precision of digital circuitry. Now as the digital domain have enhanced many folds and most of the signal processing techniques have been developed in digital domain demonstrating much greater performance and efficiency rates, thus the beamformers have shifted from analog to digital domain as well.

The difference in these beamformers is: ***Analog beamformer*** has the delay line in analog domain, where as in ***digital beamformer***, ADCs are used immediately after the time gain compensation/control (TGC) circuitry to ensure completely digital beamforming.

Beamforming involves the coherent addition of the received signals so that the gain can be maximized in a desired (steered) direction and the signal contributions are properly weighted/ scaled to sum up all the signals to give a better image. Following the beamformer block is the unit which manipulates the beamformed data such that it forms the image correctly with proper resolution.

Each block in the ultrasound system is a full research area in itself and requires extensive insight for proper parametric requirements and best results. In this document, the emphasis is laid down on the beamformer block.

Beamformers have different characteristics and more and more research and development of the digital domain is attracting more studies in this domain [1-5]. There are beamformers which are simple and are known as *conventional beamformers*. They have the parameters (i.e. delay, apodization coefficients, etc.) calculated by default before the actual beamforming system starts. But another kind of beamformer involves adaptive schemes and is known as *adaptive beamformer*. The beamformer adapts to the surrounding space for example it changes the sensitivity level at different directions to focus in a particular direction. This is done by analyzing the environment around the system. The structure of a simple/conventional beamformer is given in Figure 6.

These signals are sampled using high speed analog to digital converters. After the digital samples are formed, this data is delayed and weighted and then summed to form properly focused beams at the beamformer output.

From the above discussion, we gather that beamformer is an essential and integral part of an ultrasound system. It needs to be efficient in terms of its resource and time utilization as well as it has to be portable to provide ease of movement. In order to take real time measurements, the speed is critical issue with reduced memory requirements as hardware is expensive. Thus, to get images at higher frame rate with refined resolution and good signal to noise ratio, more insight

towards development of less complex and more efficient beamformers is required. Below are a number of concepts related to the beamforming.

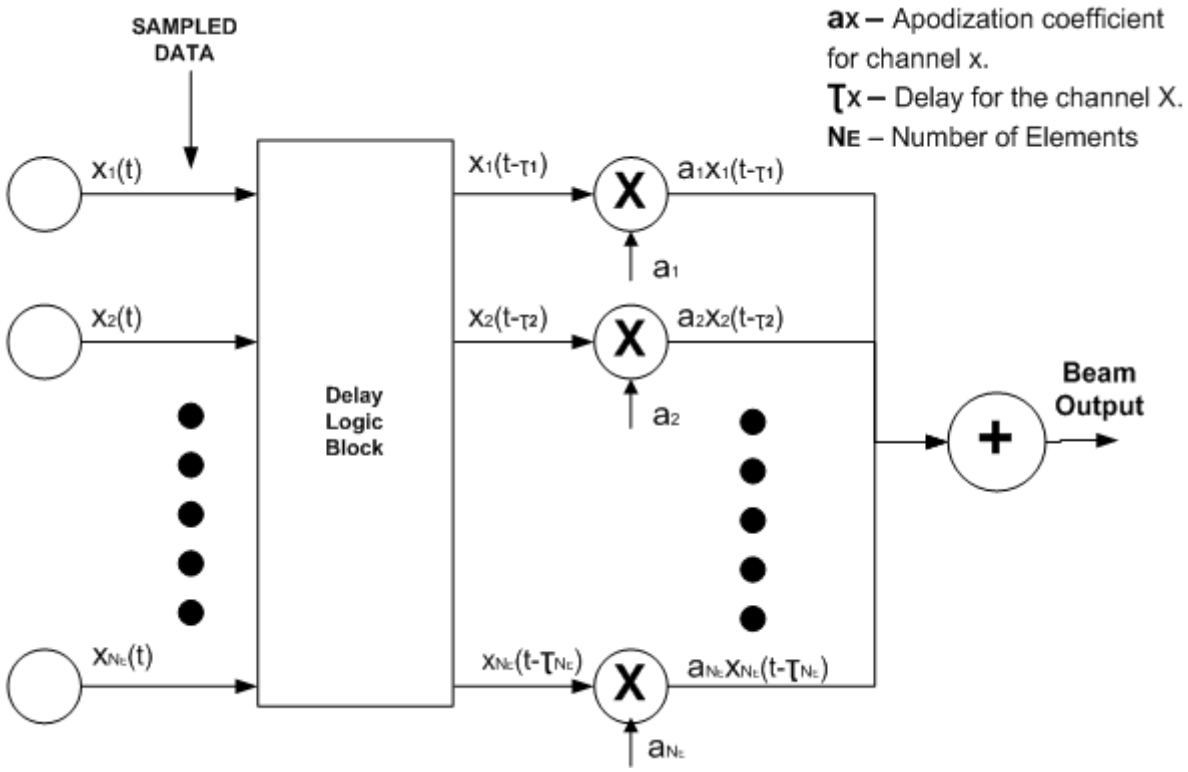

Figure 6 - Conventional delay and sum beamformer [1]. NE represents the total number of transducer elements that transmit and receive the signals. This number can vary depending upon the scheme used. xn(t) are the signals received at the nth transducer element.

Beam steering/focusing is the process required to focus a particular focal point using proper delays.

Beam apodization values are used to properly weight the samples received in order to efficiently define the contribution of every sample towards a particular beam. The basic beamforming formula is given by the following equation [1]:

$$b(t) = \sum_{n=1}^{NE} a_n x_n(t - T_n)$$

where NE are the number of transducer array elements, τ_n is the delay to be set for the n^{th} sample and $x_n(t)$ are the signals coming from the n^{th} transducer element.

1.3 Ultrasound system operation

In Ultrasound system, the waves or pulses transmitted from the transmitted elements, which are closely spaced, interfere with each other. This interference can be made directional by adjusting the delays between them so that the generated energy is incident on a desired focal depth. Focal depth is the distance from the transmitter to the point of focus in the image.

For the data acquisition, before the transmission, a focal depth is decided according to which the elements are delayed to generate the focused beams. This is to ensure that the waves of maximum energy reach the focal point.

The sound waves are to be focused at the point of measurement. This is done by adjusting the shape of the signal transmitted from the transducer, placing a lens in front of the transducer or by electronically controlling the pulses from the ultrasound machine. The electronic manipulation of these pulses is known as beamforming. It is an essential part of ultrasound systems these days.

1.4 Thesis contributions

This thesis makes two main contributions: (i) towards efficient implementation of beamformer using FPGA (Field Programmable Gate Array) systems. FPGAs are known to provide efficient solutions to implement components/systems in hardware with cost effectiveness, and (ii) towards efficient system resolution and focusing using dynamic focusing for beamforming in ultrasound systems.

Also, most of the beamformers do fixed focusing by applying a single focal depth in calculation of all the scan lines. This focuses the image in a single depth in the image and the points/regions far/near from that depth remain unfocused. Dynamic focusing is studied to examine that the

proposed delay and sum beamformer design works with dynamic focusing as well. Also, this proves that implementing dynamic focusing in hardware with the delay and sum beamformer will increase the system resolution and signal to noise ratio (SNR) in comparison to the fixed focus beamformer.

1.5 Thesis organization

The rest of this thesis is organized into six chapters. Below is a description of each of them.

Chapter 2 contains an in-depth presentation of the beamformer designs already proposed and implemented in the literature. It also explains the advantages and disadvantages of them by comparing them. Fixed and dynamic focus implementation and designs are also discussed.

Chapter 3 describes 'Ultrasound Array Research Platform (UARP)' established at University of Leeds, UK. UARP is the testing environment used for hardware implementation of the delay and sum beamformer. This chapter explains the UARP system with its features and specifications.

Chapter 4 demonstrates FPGA based implementation of delay and sum beamformer. This chapter involves the implementation details of the fixed focus, FPGA based delay and sum beamformer.

Chapter 5 proposes the dynamic focus beamforming method and compares it with the fixed focus beamforming. It explains how it has been designed to be included with the same design of the beamformer as described in Chapter 4.

Chapter 6 is divided into two sections. First section presents the results for the fixed focus, FPGA based delay and sum beamformer while the second section presents the results for the dynamic focusing using the same beamformer design.

Chapter 7 concludes the thesis document and also provides some proposals for further research areas from this study.

Chapter 2:

Literature Survey

This chapter highlights the literature related to the beamformers and different techniques used for its implementation. It also emphasizes the importance of implementing beamformers in hardware systems as Field Programmable Gate Arrays (FPGAs).

2.1 Types of beamformers

Beamformers can be classified into a number of implementation types depending upon the system requirements and the performance. Following are the three basic types of beamformers being developed over the period of digital advancements [1]. We will discuss each in detail below.

2.1.1. Delay and sum beamformer (or conventional beamformer)

The basic functionality of a delay and sum beamformer includes the data samples from the ADCs which are delayed, weighted, and summed to get the desired beam. Once the proper delays have been incorporated, the signal values are weighted using apodization values. Thus, a delay and sum beamformer has the following basic functionalities:

- Delay Unit.
- Weighting/Apodization Unit.
- Adder Unit

The block diagram of a basic delay and sum beamformer was shown in the Figure 6 in Chapter 1.

Pros and cons

The delay and sum beamformer has the advantage of being very simple in complexity but it requires very high sampling rate (much higher than the Nyquist rate) which results in increased requirement of input storage components [1].

2.1.2 Partial sum beamformer (or sum and delay beamformer)

Partial sum beamformer [1] (as shown in Figure 7), also termed as sum and delay beamformer, reduces the amount of sampling storage required in delay and sum beamformer for beam steering. In partial sum beamformer, the partial sums are calculated as soon as the data is sampled. The partial sum calculation is done in a pipelined fashion resulting in reduced memory requirements but enhanced complexity. The process follows that at a particular time stamp, the samples are taken which are required to be delayed by same value. This is controlled by a control unit which manages the pipelining as well as the addresses of the memory storage block. Thus, all the samples which require same delays from highest value towards lowest are taken and after weighting, they are summed together in a pipelined adder which takes care of the delays as well. This reduces the memory requirements. The number of partial sums required to form a beam in direction θ_i are given by [1]:

$$S_j = \frac{\max \theta_i(\tau_n)}{T_o}$$

where $\max \theta_i (\tau_n)$ denotes the maximum delay corresponding to the jth beam and T_o is the time after which a partial sum is formed. For a single beamformer cycle, the number of beams to be formed is predefined which are directly proportional to the number of scan lines formed. Thus the total number of sums to form NB beams is given by [1]:

$$S_T = \sum_{n=1}^{NB} S_j \,, \leq \frac{NB \max \theta_i(\tau_n)}{T_o}$$

where $\max \theta_i (\tau_n)$ denotes the maximum delay corresponding to all beams. The memory required for a conventional delay and sum beamformer is $(\frac{NB \max \theta_i(\tau_n)}{T_o})$. Thus, if $NE > NB$ and $T_o \gg T_i$, the memory storage saving is very high.

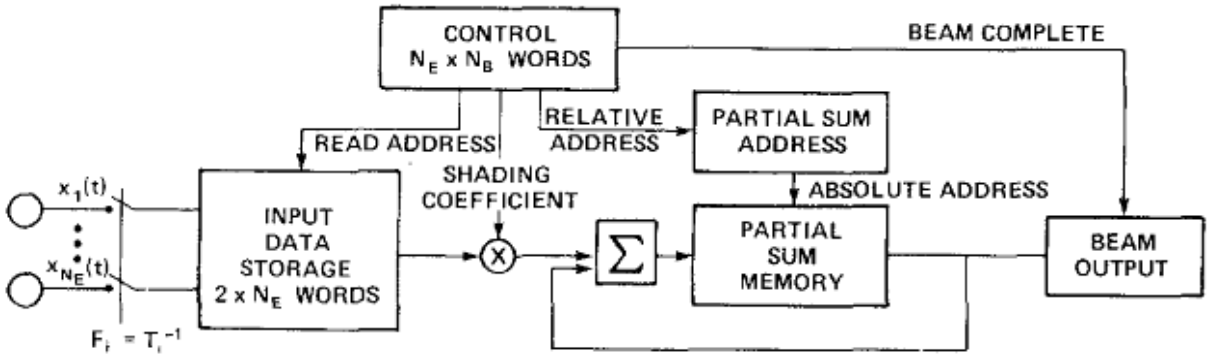

Figure 7 - Partial sum beamformer [1]

Pros and cons

The advantage of partial sum beamformer is that it reduces the amount of memory storage requirements but still it faces the issue of high sampling rate requirements like the delay and sum beamformer.

2.1.3 Interpolation beamformer

With the advancements in the beamformer architectures, the requirements of most refined images and least errors resulted in refining the system speed and resource utilization. The limitation of the sampling rates of the ADCs has resulted in concepts of interpolation [3]. As the sampling is required to be consistent with the Nyquist rate, thus this is achieved using the digital interpolation techniques [4]. Interpolation is said to be a two-step process as shown in Figure 8, that are zero padding and filtering.

The impact of this structure is significant as the interpolation system shares the burden of high sampling rate requirements with the ADCs. Also, added to this are the effects of using the beamformer with the interpolation. As beamforming and digital interpolation are both linear processes, thus interpolation can be done either before or after the beamforming. But the impact of positioning the interpolation around beamformer is different. The figures of the two approaches as well as the impact are explained below.

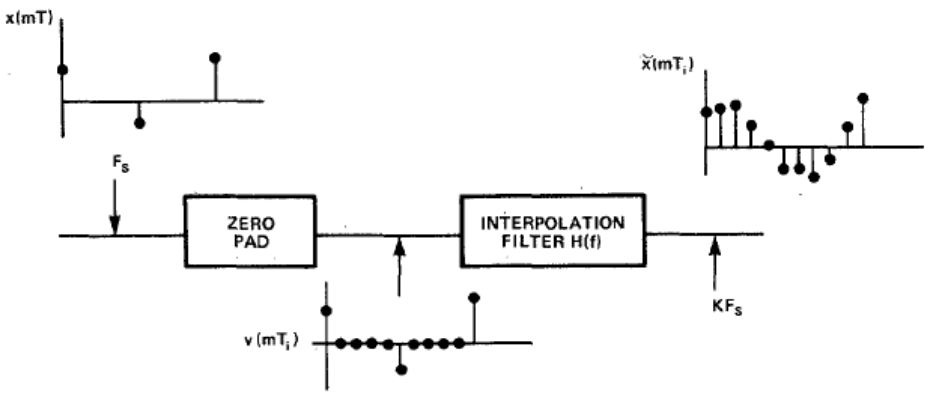

Figure 8 - Two-step digital interpolation [1]

Interpolation and beamforming

One approach is where the interpolator can be placed before the beamforming circuitry as shown in Figure 9 and Figure 10. This is also termed as pre-beamforming interpolation. When the interpolation is done at the input of the beamformer, the output sample rate is much higher than the input sample rate.

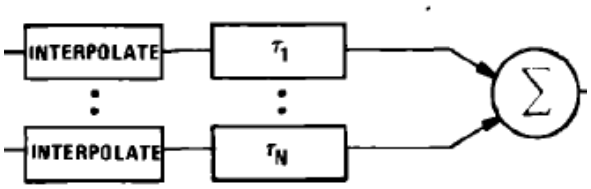

Figure 9 - Pre-beamforming interpolation [3]

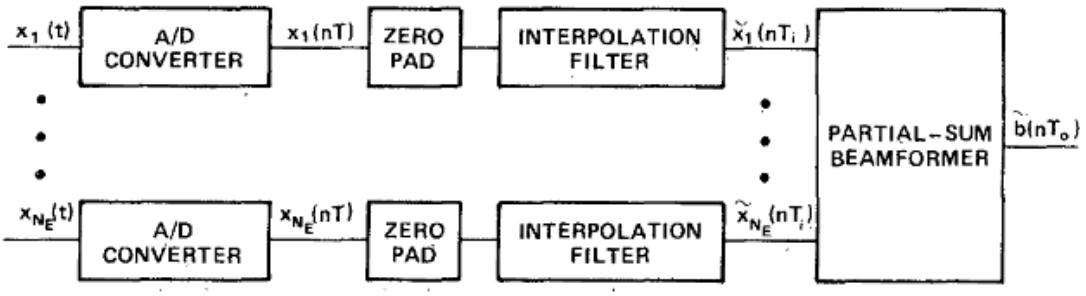

Figure 10 - Pre-beamforming interpolation block diagram [1]

Beamforming and interpolation

Other approach is where the interpolator can be placed after the beamforming circuitry as shown in Figure 11 and Figure 12. This is also termed as post-beamforming interpolation. When the interpolation is done at the output of the beamformer, the output sample rate is similar to the input sample rate. The zero padded samples are processed in the beamformer and then the interpolation filter down samples the data for the image reconstruction. This scheme is said to be more advantageous as it reduces the amount of the memory requirement as well as enhances the computational complexity if the number of beams formed is less than the number of input elements [3].

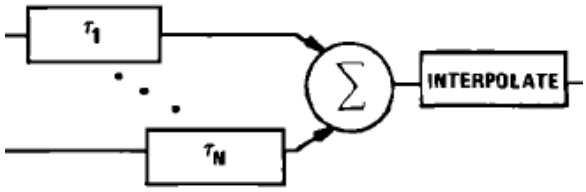

Figure 11 - Post-beamforming interpolation [3]

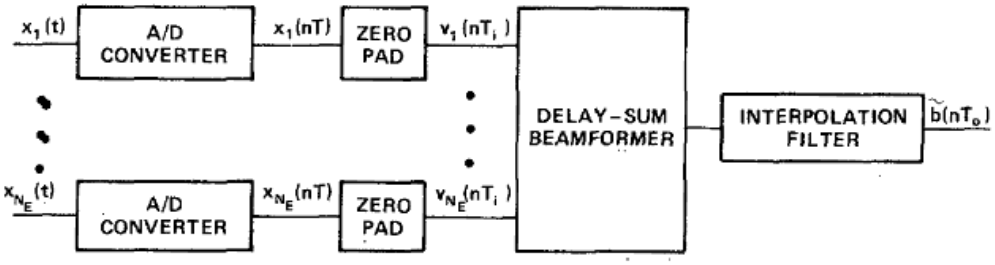

Figure 12 - Post-beamforming interpolation block diagram[1]

Interpolation filter is a critical requirement in the design of the interpolated beamformer. This filter needs to have characteristics of an ideal low-pass filter. The zero padding in interpolation adds a lot of undesired spectra which needs to be removed. This is typically done by using a filter with least pass-band and stop-band ripple using least number of filter coefficients [3].

The filter types are Finite Impulse Response (FIR) filters and Infinite Impulse Response (IIR) filters. FIR filters have the following advantages over the IIR filters [3][5]:

- FIR filters have a non-recursive implementation scheme which results in advantages of interpolation implementation at both the inputs and outputs of the beamformer. They overcome the issues caused by the recursive IIR filters which replace the zero padded values by non-zero values in the feedback path.
- The implementation in terms of storage and computational feasibility can be enhanced using efficient FIR filters.
- They add linear phase delays due to the absence of feedback paths which results in no phase distortions.

Normally, the implementation of an FIR filter is done closest to an ideal low-pass filter but still there are differences which result in errors [5]. These errors are due to pass-band and stop-band ripples which can be reduced by increased filter coefficients. Thus, filter design involves a tradeoff between the tolerance of ripples and the number of filter coefficients [3].

Pros and cons

Interpolated beamformer can approximate the fractionally delayed signal values and eliminates the requirement of very high sampling rates. But the computational complexity increases with the increased resolution required.

2.1.4 Phase rotation beamformer

Phase rotation beamformer exploits the complex sampling concepts of the band-pass signals. The signals are sampled so that both the real and the imaginary parts are extracted which are separately manipulated to result in accurate phase shifts.

As shown in Figure 13, the samples of the signals from the ADCs are down-mixed/shifted to zero frequency to pass them through the low-pass filters to remove any harmonic components at the higher frequencies. After the demodulation, the delays are incorporated for the beam

focusing and finally the phase corrections are made to eliminate the phase aberration errors. Figure 13 shows the frequency spectrum as well as the chain through which the signal samples pass after reception by the ADCs.

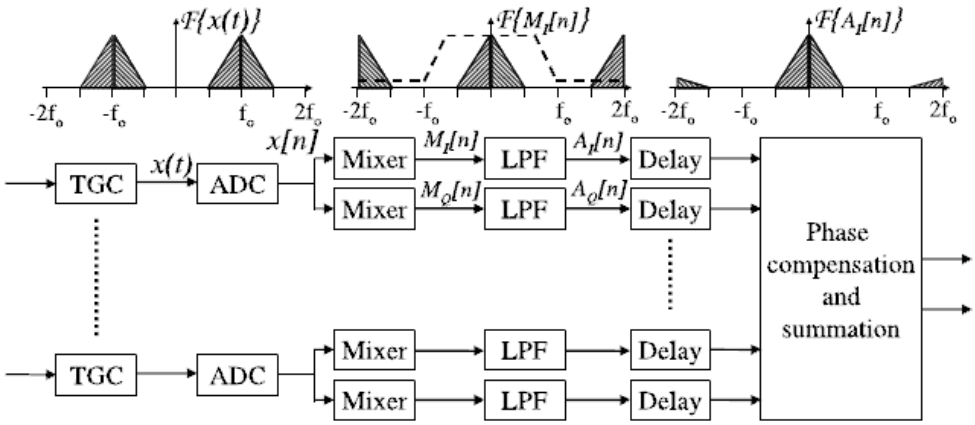

Figure 13 - Block diagram of a phase rotation beamformer [6]

Techniques have been developed to reduce the amount of hardware resources used by the phase rotation beamformers [6-7]. Two-step demodulation is developed that reduces the amount of resources used with maintenance of the image quality. The architectures exploit the uniform nature of the filter coefficients resulting in less number of multiplications. Filter implementations where multiplications are replaced by shifters and adders are used. Multiplications are always hardware expensive.

Pros and cons

Phase rotation beamformers have the advantage of correcting the phase aberration errors but they require much more hardware resources to incorporate the complex demodulation.

2.1.5 Synthetic aperture beamformer

In this beamformer, a single element is used for transmit while all the elements are used to receive the signals. Thus, each excitation results in formation of a low resolution image. All the

low resolution images are added to form a single high resolution image as shown in Figure 14. The addition involves techniques that result in better resolution and coherent addition.

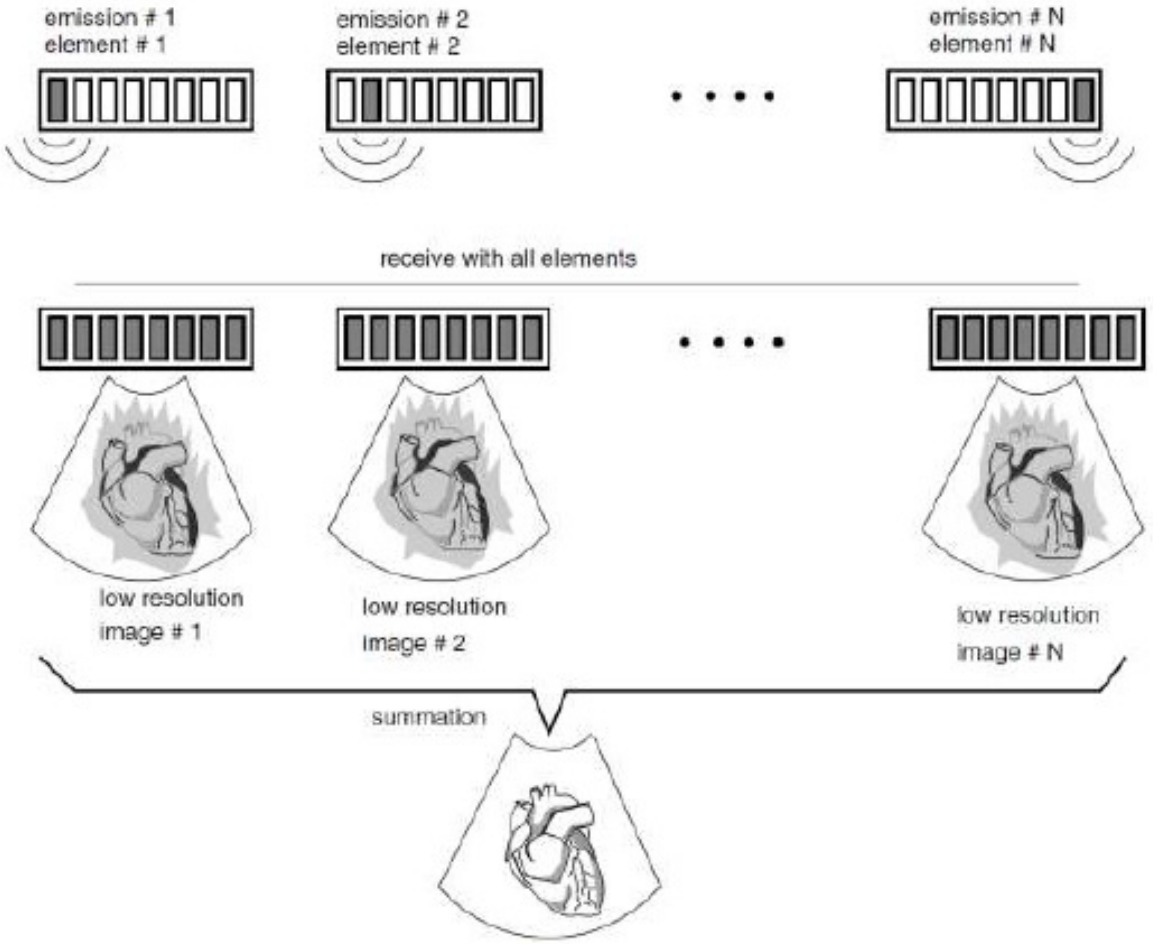

Figure 14 - Synthetic aperture beamforming [8]

Pros and cons

These are best suited for the imaging of moving objects as they record the data continuously. They are closely related to the phased-arrays as both encounter the problem of phase errors. Thus, they also require phase compensation techniques to be incorporated for coherent addition of signals.

2.1.6 Fractional delay (FD) beamformer

Fractional delay beamformers use fractional delay filters for finely estimating the delays as shown in Figure 15. These fractional delay filters are a form of interpolation filters which have the characteristic of having a flat magnitude that best filter the signal with minimal attenuation [9]. They are proved in literature to utilize less hardware resources as compared to general interpolation or phase rotation beamformers [10].

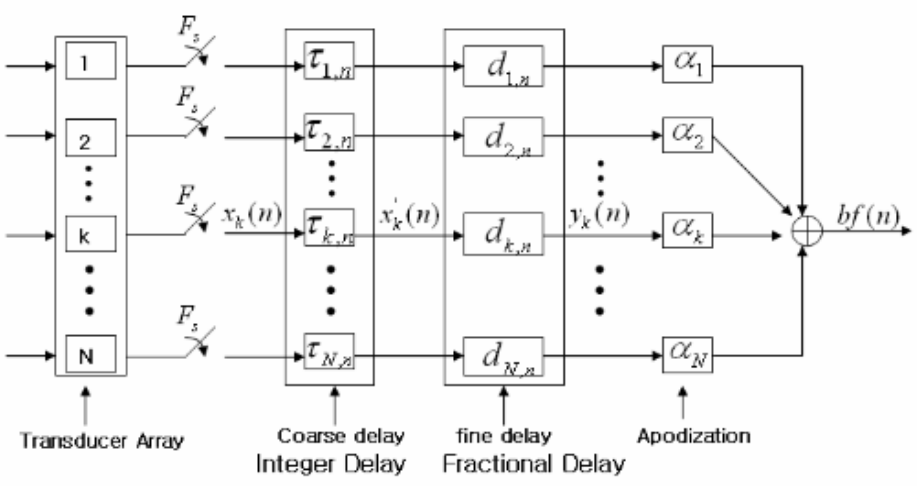

Figure 15 - Fractional delay beamformer [10]

Pros and cons

Experimental results show that the fractional delay filters are superior in performance and resource utilization to the interpolation and phase rotation beamformers, but there are limitations in the lowest delay that could be encountered.

2.2 Comparison of beamformer techniques

Table 1 illustrates comparison of the beamformers discussed in section 2.1. It summarizes the pros and cons of each of the techniques.

Table 1 - Comparison of beamforming techniques

No.	Beamformer Type	Pros	Cons
1	Delay and sum beamformer	-Simple in implementation and complexity.	-High sampling rate required. -Large memory requirements.
2	Partial sum beamformer	-Reduced memory requirements compared to (1).	-High sampling rate required. -Greater complexity than (1).
3	Beamform and interpolate	-Efficient in terms of memory resources required.	-Filter design complexities.
4	Interpolate and beamform	-Better image quality.	-Less efficient than (3) if no. of beams is less than no. of elements. -Filer design complexities.
5	Phase rotation beamformer	-Provides refined phases which helps in dynamic focusing. -Refined delayes.	-Increased hardware requirements for each in-phase and out of phase component.
6	Synthetic aperture beamformer	-Less energy requirements at the input. -High resolution obtained.	-Huge resource requirements. -Complex image reconstruction methods for refined images.
7	Fractional delay beamformer	-Better hardware utilization as compared to (5). -Better delay accuracy than (1).	-Very small delays cannot be easily achieved. -Filter selection constraints to reduce the error.

The discussions show that beamformer is an essential and critical component of an Ultrasound system. It needs to be designed, implemented and used efficiently to get enhanced Ultrasound functionality. Beamformers, formally implemented as analog components, have now been

designed to benefit from the digital domain and its advantages. Digital systems facilitate fast, accurate and efficient designs.

Field Programmable Gate Arrays (FPGAs) are efficient systems which provide flexibility to implement designs easily and in a very cost effective manner. They are reconfigurable which makes them highly a viable component for research environments where frequent testing is required. They provide an easy interface for implementing designs with reprogrammable features.

Beamformers are now being implemented using FPGAs [10-15]. Still the designs need to be efficient to occupy least space on the device. This thesis proposes a beamformer design using FPGAs for fixed focusing.

2.3 FPGA based beamformer design

Field programmable gate arrays (FPGAs) are programmable devices which are configured after they are manufactured. They contain logic cells and interconnect to connect these logic cells. This interconnect can be programmed to connect in a way required. And this flexibility of FPGAs to be programmed as many times as needed makes it highly acceptable by the research community.

Now-a-days, FPGAs in addition to being able to provide the flexibility to be used many times with a variety of settings and configurations, it is now being loaded with ARM processors which makes it even more viable solution to provide ‘programmable system on chip’. This is making FPGA custom boards a cost effective and feasible solution for implementing software connected hardware systems.

Beamformer, as already emphasized in previous discussion, is a very critical component of the Ultrasound system. Commercial Ultrasound systems cannot be changed and modified as they are tamper proofed. Also, they are very expensive machines [11-12]. For exploring the new technology and analyzing the already available procedures, the Ultrasound system needs to be

broken down to study the different components. Also, to modify these components to adopt and to test for newer concepts and studies, the Ultrasound system needs to be open for experimentation and research. But unfortunately, this is not possible with the commercially available systems as mostly inputs and outputs are defined with specific mode of operation which are supported by the machine. In addition, these systems are very expensive and nearly unaffordable by the research labs.

Thus, the research groups like in University of Denmark and University of Florence have created their own Ultrasound setups (RASMUS/ SARUS[22-23] and ULA-OP[11,12], respectively) to study its components and develop new techniques and tests these on the system.

Remotely Accessible Software configurable Multi-channel Ultrasound Sampling (RASMUS) developed in University of Denmark works in optimization of the beamformer design to explore the inability of commercial ultrasound systems to be modified and investigated. This system is working in areas like development of synthetic aperture based techniques (SARUS), dynamic focusing and compact FPGA based implementations [22-23]. This system is developed for latest research requirements and testing but it is not feasible for on the spot testing due to its huge size and circuitry.

Ultrasound advanced open platform (ULA-OP) is another group working on implementation of the ultrasound components in FPGAs in University of Florence. They are working to develop compact FPGA based beamformer designs. The basic beamformer is implemented by fixed focusing resulting in low resolution. Interpolation is planned to be exploited to resolve this [11-12, 30]. This system, as shown in Figure 17, generates the parametric information required for the platform using software and then loads these values to the platform. Analog and digital boards are used during data transmission and reception.

The systems described in Figure 16 along with others [13-15, 40-50] provide examples of hardware based Ultrasound systems developed by different groups to study the Ultrasound procedures and also experiment to explore newer ones.

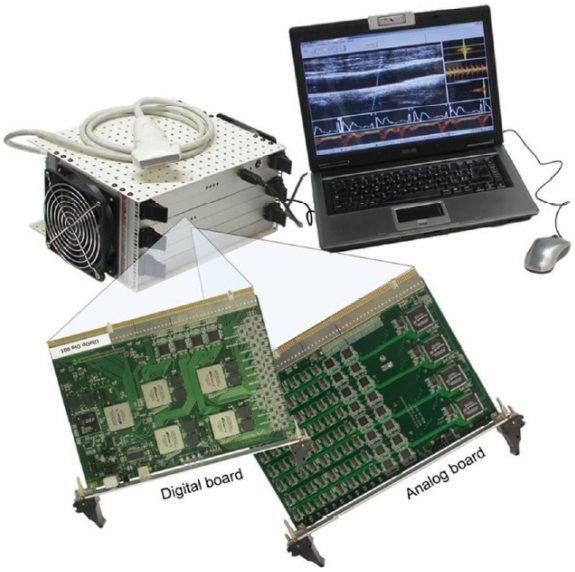

Figure 16 - ULA-OP platform [11-12]

Ultrasound array research platform (UARP) [27] is another such system developed in University of Leeds, UK. The Ultrasound group at the University is developing this platform to study the Ultrasound procedures and techniques such as contrast agent, micro-bubble detection, ultrasound imaging, non-invasive measurement, etc. UARP presently uses software based beamforming using Matlab. This inherits the software based overheads of slow processing speeds. This affects the overall system frame rate. The architecture of UARP is up to date and supports 96 independent channels with 50MHz sampling frequency with 12bit precision at input.

These systems are developed and designed to support real time testing and also to adapt to newer research ideas and techniques. Implementation of beamformer in hardware enhances the system speed. Hardware based Ultrasound operations make the system faster and feasible for easy real time testing considering movement.

The need of hour is to produce a beamforming structure which is efficient in terms of flexibility, speed and performance and occupies least space and resources. This thesis will accomplish this goal by proposing a design which will be modular and flexible for newer technologies.

2.4. Dynamic focusing

The existing beamformer design generates a single image line in real-time per transmit event. This line is beamformed to a specified single focal depth and angle [2]. As a consequence, multiple transmit events are required to form a (multi-line) image, using either linear or phased imaging.

In the initial design, the beamformer pre-loads zeros into FIFOs to delay the data sampled at the ADC sampling rate. This results in delays being quantized to integer multiples of the ADC periods. Coarse quantization can reduce the dynamic range and therefore contrast of features in an image [16]. Thus the system resolution gets affected and beamformer needs to be efficient in terms of the smallest possible object it can image.

A further advancement in the delay-and-sum beamforming is the use of multiple focal zones. This can be achieved in post processing by picking various focal points and delay-and-summing at these particular points. These points can then be 'spliced' together to form a multi-focal image [29-31].

A more advanced method alters the focal depth with time during the beamforming process [2]. This is achieved by either calculating focal delays 'on-the-fly' or by using pre-calculated values stored in look up tables [17]. Delay coefficients are then altered as data is being sampled on a per-sample basis, thus creating a single image line with multiple focal zones or continuous focus. This process is often known as 'dynamic' or 'perfect' focusing. The performance of dynamic focusing is improved if an interpolating beamformer is used [2].

Dynamic focusing is a standard imaging tool in clinical systems [18-21], and is also currently used as a fundamental process in several other imaging techniques and modalities. These modalities offer improved image quality but at a cost of increased computational intensity. An example is Synthetic Aperture imaging [2, 22-26] which uses a large number of transmit events to form a number of low resolution image frames. These low resolution images are then summed

and combined to create a high resolution image. Dynamic focusing techniques are required to form each low resolution image.

Dynamic focusing is a current research being done with the ultrasound systems to gain system resolution. Ultrasound elastography is a technique where ultrasonic imaging is being used to identify tumors in the human body. Availability of single focus beamformers limit the measurement and analysis in modern high end techniques to work with 100% output efficiency. Focusing at multiple points in an image produces high quality images [20, 33]. Also, other photo acoustic imaging requires the beamformers to perform dynamic beamforming [34].

Extending the existing beamformer designs of single/fixed focus to perform dynamic beamforming would require alteration to the delay process. In the dynamic beamformer design, the data should be saved into the memory. Delays can then be implemented using independent read pointer addressing different memory locations. Each pointer points to a different sample value as defined by a time dependent delay profile [11, 16]. The location of the pointers is updated on a per-sample basis, with the pointer locations calculated on-the-fly or stored in look up tables as discussed previously.

This chapter unfolds the literature for the details about different beamforming techniques available. Delay and sum beamformer stands as adoptable approach as it may be adjusted easily to be used for any current emerging imaging schemes in ultrasound systems as synthetic aperture. Dynamic focusing is proposed to enhance system resolution, but its hardware implementations are still a challenge.

Chapter 3:

Ultrasound Array Research Platform (UARP)

Ultrasound Array Research Platform (UARP) as a testing environment is explained in this chapter. It highlights the overall platform design and explains the basic system features.

The Ultrasound Group in University of Leeds, UK has developed Ultrasound Array Research Platform (UARP) as a system to carry out Ultrasound research to explore Ultrasound methods and applications [27]. The setup is shown in Figure 17.

Commercial ultrasound systems are well designed and suited to the medical professionals providing them support in accurate diagnosis of medical conditions. As time has advanced the technology many folds, these systems need to adapt to the latest research. This requires study of latest techniques and experimentation, which is not possible as these systems, although can alter some of the parameters like focal depth, etc., cannot be changed. However, they being well suited to the clinical setup stand inappropriate for the research environments.

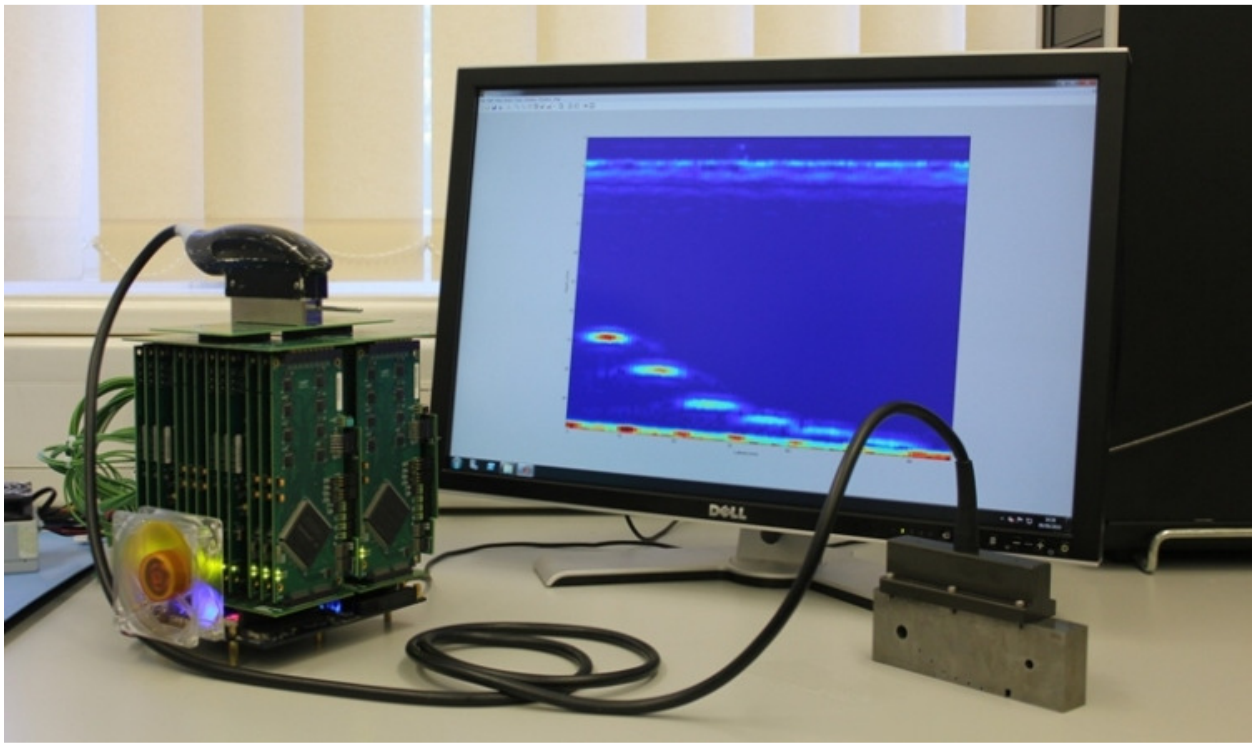

Figure 17 - Ultrasound Array research Platform (UARP) including the ultrasound probe, test block A and the Testing Software in PC.

UARP is designed and developed to provide the researcher full control over all the system parameters with the flexibility to monitor any middle-of-the-path data analysis. This enhances the system flexibility and enables the researcher to design and test both new and established algorithms.

3.1 UARP hardware/ software components

UARP mainly consists of the data acquisition circuitry and the image processing algorithms implemented in the Matlab. Figure 18 shows the overall UARP structure composed of different components which are explained below.

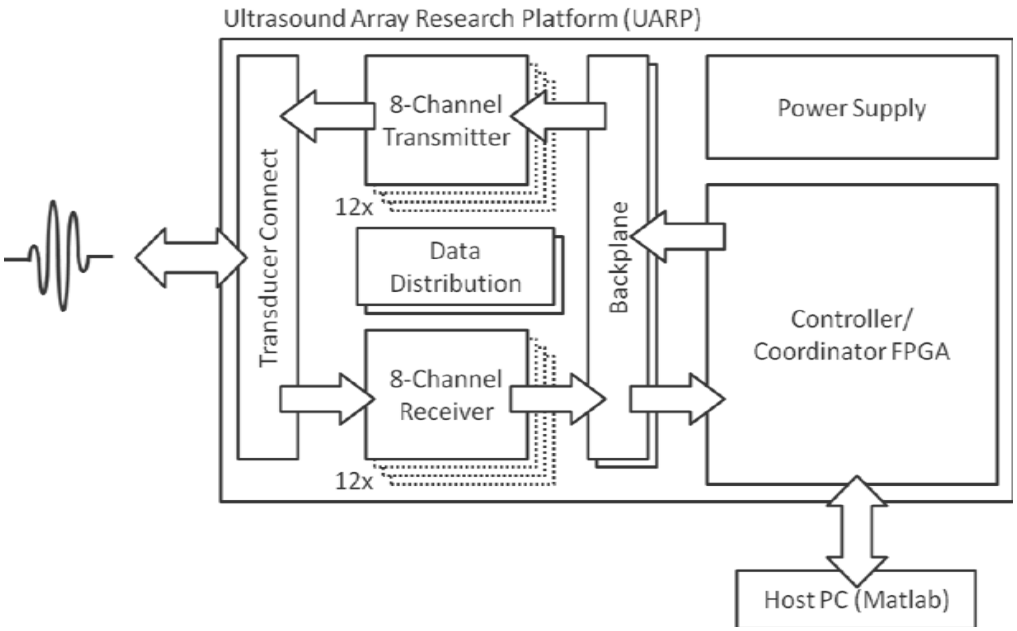

Figure 18 - UARP structure [27]

UARP system consists of the following components:

- STRATIX III custom FPGA system.
- Transmitter (Tx) and Receiver (Rx) boards mounted on the Stratix III FPGA.
- Ultrasound measurement probe.
- Test blocks.
- Testing Software.

3.1.1 STRATIX III custom FPGA system

STRATIX III custom FPGA system (as shown in Figure 19) provides the system on board to implement different algorithms in hardware to enhance the system speed in terms of frame rate. Field programmable gate array (FPGA) are devices which are 'field programmable' which means that they are configured by the designer or programmer after they are manufactured. They support designs which have parallel structures and provide faster computational speed as compared to the serial calculations in software in PC.

Figure 19 - STRATIX custom FPGA system

3.1.2 Transmitter (Tx) and receiver (Rx) boards mounted on the Stratix III FPGA

Figure 20 shows the Ultrasound Array Research Platform (UARP) circuit boards. It comprises of 12 transmitter boards and 12 receiver boards. These 24 boards are connected to the STRATIX III custom FPGA system. Each transmitter board contains 8 digital to analog converters while each receiver board contains 8 analog to digital converters. Thus, each transmitter and receiver board pair results in 8 data channels. Each of these board pairs share a common bit clock, which runs with the ADC or DAC, which samples the data and outputs the samples serially bit by bit. Having 12 such pairs in the UARP system, the total number of channels that are supported becomes 12 * 8 = 96 and the total number of bit clocks become 12 as well. The serial data from the ADCs are fed to the STRATIX III board while with the help of programming, the serial data is converted into 12-bit samples and the bit clocks are transformed to frame clocks.

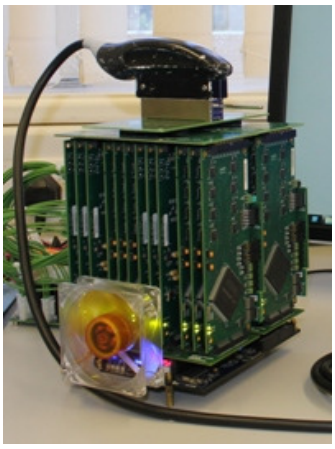

Figure 20 – UARP Tx and Rx circuit mounted on a STRATIX III FPGA board

3.1.3 Ultrasound measurement probe

Ultrasound systems use a hand-held probe known as a transducer. It is placed on the object or body to be analyzed. Figure 21(left) shows different types of probes used now-a-days as linear, bi-planar, omni-planar, etc. Transducers which can be placed inside the body have also been designed. This transducer is normally separated from the test object using a couplant or by water. This increases the efficiency of the system by reducing losses due to surface separations.

In the UARP system, the probe is designed to be linear as shown in Figure 21(right). Thus, all the transducer elements are placed in a line.

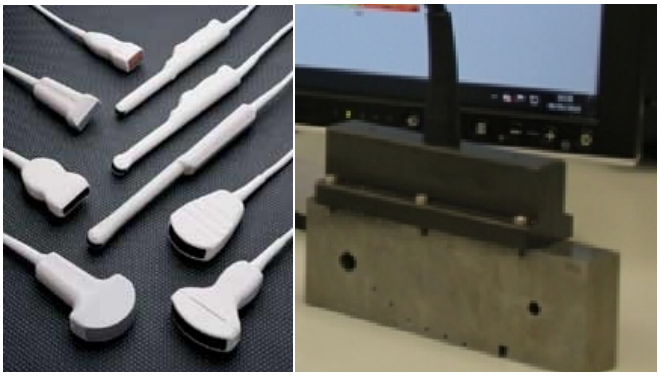

Figure 21 – (Left) Different types of ultrasound probes as linear, biplanar, omniplane, etc. (Right) UARP linear measurement probe and test block A

3.1.4 Testing blocks

Figure 22 shows the two test blocks A and B being used for testing in UARP system. Use of these blocks before using the system on human body is to test for any effects. These blocks have holes created at specific positions with specific dimensions. The holes are filled with air while the block is made of metal. Thus, the reflections are very large in amplitude due to different refractive indexes.

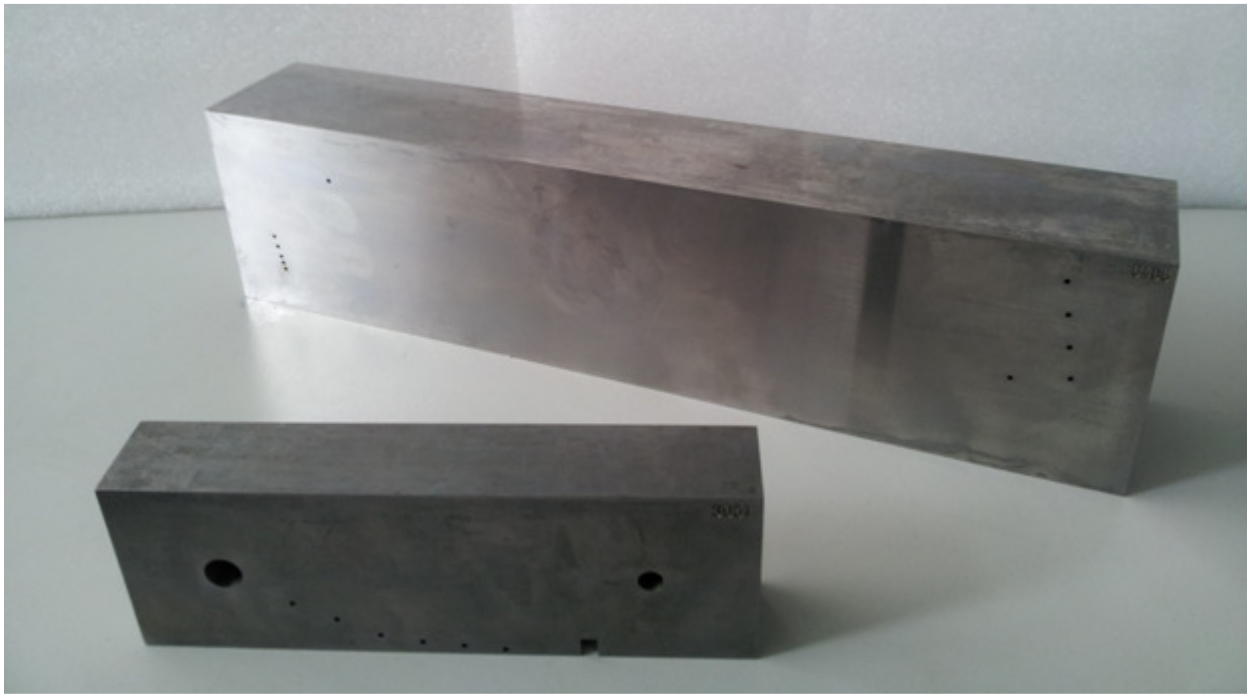

Figure 22 – UARP test blocks. (Top) Test block B and (Bottom) test block A.

3.1.5 Testing software

Matlab is software which provides greater flexibility to users for testing different systems. It provides flexibility to interact in a variety of ways, making it a viable solution for testing different systems.

Ultrasound Array Research Platform (UARP) has a Matlab based software testing paradigm which provides simple functional interface to user to interact with the UARP hardware. Simple

commands are used to communicate with the hardware for example, setting the number of transmitters/ receivers, setting the transducer frequencies, etc.

This software interacts with the STRATIX III FPGA system using a USB 2.0 link. The bandwidth of this interface limits the amount of data as well as the speed of data possible. The system can be checked by different read and write operations. Thus, this software makes the hardware fully operable with maximum flexibility of setting any parametric values.

3.2 UARP features

UARP is designed to provide maximum flexibility to the researcher to carry out variety of experiments to test established algorithms and design new techniques and algorithms. Figure 23 shows the UARP platform.

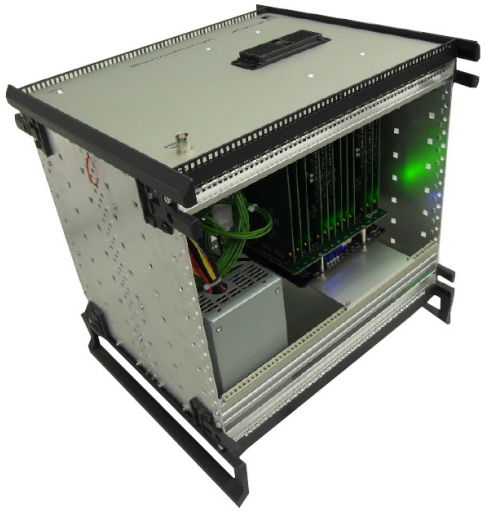

Figure 23 - Ultrasound Array Research Platform (UARP)

3.2.1 Mode of operations

An ultrasound system has two methods of receiving the ultrasound waves back.

- Reflection.
- Attenuation

In reflection mode, the sending and receiving are both done by the transducer and the reflected signal is measured by the same transducer. These reflections are from any back wall or any imperfection in the object. The reflections are displayed after processing as amplitude and distance. Amplitude represents the intensity of the reflection and the distance represents the time to the reflection point. This mode is also known as pulse-echo mode.

In attenuation mode, the transmission is done using a set of transmitters from one end of the object, while the waves are received by a set of receivers at any other end of the object. The waves received are measured and show the amount of attenuation while they moved from the transmitter to the receiver. This mode is also known as through transmission mode.

UARP uses the reflection mode where the same transmitters are used as receivers.

3.2.2 96 Independent transmit/receive channels

UARP consists of 12 transmitter and receiver boards which makes 96 data channels. These transmitters and receivers are selectable to be any of them from 1 to 96 for a particular transmit receive event. This enhances the system flexibility.

3.2.3 Frequency

Ultrasound systems normally have frequencies in the range upto 15 MHz. There are systems which support/require higher frequency as from 50 – 100 MHz.

Frequency has always been a tradeoff between the system spatial resolution and the penetration depth. Lower frequencies give higher penetration into the body but gives less resolution while the higher frequencies gives less penetration due to large attenuation coefficient while have higher resolution.

UARP supports a transmit frequency of up to 15 MHz. This gives higher penetration depth with acceptable resolution. This signal is sampled at 50 MHz by the analog to digital converters. They

sample the input analog signals and converts it into a 12 bits samples, making the receive resolution of 12 bits.

3.3.5 Pre-beamformed data access

The beamforming was done in the UARP system in Matlab, before the hardware based implementation (as explained later in Chapter 4) is done. Thus, the data samples captured from the analog to digital converters are stored in the STRATIX III memory. This memory is read by the Matlab module using the USB 2.0 link.

This data path where the samples are read from the analog to digital output and placed in the STRATIX III memory and then read by the Matlab code is known as pre-beamformed data path. The data is processed using image processing techniques to form ultrasound scan lines and then display the image. This is a useful data path as different beamforming schemes can be applied using the data samples. However, this scheme adds the processing overhead in Matlab, where every step is computationally time expensive.

3.3.6 Parameter loading setup

UARP is a system that contains Matlab based software testing code, a STRATIX III based hardware reception module and the front end circuitry. The STRATIX III system requires certain parameters using which it decides the amount of transmitter and receivers to use and their settings. This loading is done using USB 2.0 based commands from the Matlab interface into the STRATIX III registers. These registers can be varied in number and size.

The STRATIX III custom FPGA system has a NIOS II processor which handles this loading. NIOS II processor is an intermediate unit on the STRATIX III custom FPGA board which connects with the Matlab through the USB 2.0 link as well as the FPGA code. NIOS II processor interprets the commands from the Matlab and performs the tasks. NIOS II processor is also connected to the front end circuitry. Thus, Matlab communicates to the NIOS II processor in order to manipulate the hardware based components of the UARP.

3.3.7 Beamformer

Beamformer is an essential component in ultrasound machines. It is a signal processing technique which combines the interference effect of closely spaced transmission and reception elements to divert the energy in a defined direction.

Delay and sum beamformer contains three basic building blocks as:

- Focusing block.
- Apodization block.
- Summation block.

Beamforming in UARP was done in Matlab after the pre-beamformed data is accessed from the hardware using the NIOS II processor and USB 2.0 link. All the three above mentioned steps were performed in Matlab.

3.3.8 Data memory

The pre-beamformed and beamformed data samples are stored in a two port RAM (random access memory). This memory is accessed by both the Verilog code and also by the Matlab testing software using the NIOS II processor interface. This memory can store 2K 32-bit words.

This chapter demonstrated the basic functionalities and features of the UARP system with detailed discussion on the components involved. It highlights the already available system to clearly differentiate the contributions later on.

Chapter 4:

FPGA Based Implementation of Delay and Sum Beamformer

This chapter explains the hardware implementation that we did using the UARP system. It explains the overall beamformer flow, with its breakdown as macro- and micro- architectural breakdown. It also highlights our block wise implementation details with an optimization of the proposed scheme which results in appreciably high frame rates as compared to the software based beamforming using Matlab.

4.1 Flow of delay and sum beamformer

Ultrasound array research platform (UARP) first transmits the excitation pulses and then the front end circuitry switches to the receive mode. Just before the reception starts, NIOS II processor performs the calculation for the delay while the excitation pulses are being transmitted. Also, these values are then loaded into the STRATIX III board into the defined registers.

When the transmission is done, the values for the delay and other system parameters are already loaded from the NIOS II processor into the STRATIX III registers. The UARP system controls the enable signal for the beamforming. A pulse is generated by the NIOS II processor which approximates the time while the transmissions by the hardware will be done safely and the reception can be started now. This pulse enables the receive circuitry. Also, a beamformer enable signal is automatically generated from the hardware system. When the registers are loaded with the delay and the parameter values from the NIOS II processor, they are then loaded to the beamformer when the register read signal is generated. As soon as the beamformer enable signal is generated, the beamformer starts receiving the data from the analog to digital data path.

There is a memory in which the beamformed scan lines are written after they are beamformed. Every transmission results in formation of a single scan line. Beamformer keeps on beamforming till this memory is filled to full. Below is the sequence which is repeated till the memory full signal is not generated.

- Data is read from the input data path on sample by sample basis.
- This sample is then written in the First In First Out (FIFO) structure.

- A delay counter is used to incorporate the delay. As soon as the delay counter is zero, the FIFO is read.
- After the appropriate delay, the sample is passed to the apodization block, where the data sample is weighted according to the apodization coefficients loaded from the NIOS II processor.
- Once the data samples are delayed, they are passed to the pipelined adder.
- All above steps are repeated for every input sample till the beamformer memory is full.

As evident from the above flow, we designed the FPGA based UARP beamformer to allow full user control so that the system is modular and can be analyzed at any point.

4.2 Macro architecture

Macro architecture means high level description of the system. It involves high level details of what basic blocks are involved in the system design and their very basic details.

We divided the whole beamformer architecture in its basic building blocks and then separated the UARP components and our contributions. As shown in Figure 24, when we included the FPGA based beamformer with the UARP system, the design can be divided into pre-beamformed and beamformed data paths. The complete beamformed data path, as shown in Figure 24 is our contribution towards the UARP system.

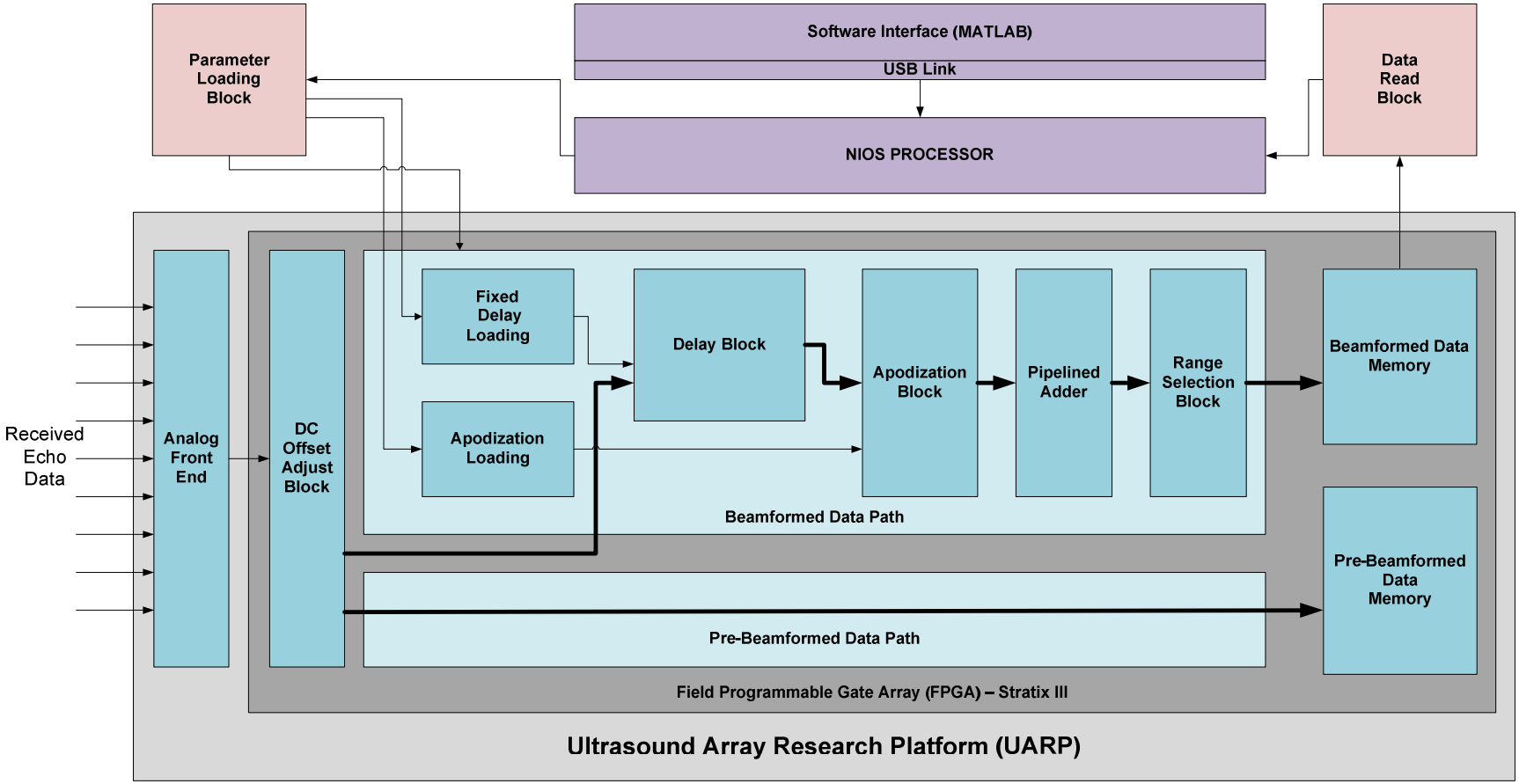

Figure 24 - Macro architecture of UARP with beamformer

Basic components of a delay and sum beamformer are defined previously, but the UARP beamformer has some additional components specific to the system design as shown in Figure 24.

DC offset adjust block

On power up, all the transducer elements get misaligned with respect to each other by a DC value. This causes an increase in the amplitude of all the samples received. This needs to be fixed. We accomplished this by subtracting a system computed value from all the samples of a particular channel. This is done with all the channels i.e. 96 in UARP.

NIOS II processor receives the value to be subtracted known as the DC offset from the Matlab. System can load any number of offset values from 1 to 96 depending upon the number of channels being used.

Fixed delay and apodization loading block

Delay block is an essential component where the samples need to wait before they get aligned properly. We calculated the delays in the NIOS II processor providing it the necessary parameters as aperture size, quantization step, etc using Matlab. These delays are then loaded using the shift registers in the fixed delay loading block.

The samples received by each channel needed to be weighted properly. For UARP beamforming, as only the delays are being tested in this thesis, the weights are kept as unity for each channel. The analysis of weights is another study [36-39] being carried out, as they result in sidelobe reduction making approximations of the objects more clear.

We enabled the flexibility in UARp system to be able to compute the weights as well in NIOS II processor and then load them to the hardware platform. This path is defined to enhance the system modularity and to keep the research aspect to implement variable weight generation techniques open. We again used the shift registers to load these weights. ***Shift registers*** are a

resource efficient way to load values. We first implemented the delay and apodization loaders by creating a separate bus for each channel's delay and apodization values. This caused exhausting the bandwidth available between NIOS II processor and the FPGA board. Thus, to create an efficient system, we shifted the design from separate registers to a single shift register. Here a pulse signal is used that loads all the delay values i.e. 96 in maximum before the actual data reception starts.

Delay block

In transmission, to focus at a particular focal depth, the delays are applied to the channels so that all the transmitted energy reaches the same target point. This is also true for the reception mechanism.

As already discussed, we used the NIOS II processor to calculate the delays for the current fixed focal depth. Then we applied these delays to the received samples from each channel. We implemented the logic for delaying in hardware using a delay counter. As soon as the delay is counted for the number of clock cycles, the values are read from the FIFO.

Apodization block

Linear parameterized modules (LPM) defined by Altera in Quartus are used to implement efficient multipliers. The data samples have signed representation and thus, signed multipliers are used.

We designed the system to load the apodization coefficients as 16 bit bus, but only lower 8 bits are valid weights. Rest of the higher 8 bits are kept zero. The loaded weights/apodization values are then multiplied using LPMs to the data samples of the corresponding channel. We used these LPMs because they are Altera modules which are confirmed to provide flexibility in addition to an efficient design.

Pipelined adder

The data samples once multiplied by the weights/apodization coefficients move to the pipelined adder stage. We made the pipelined adder as a 7 stage adder, where samples from two neighboring channels are added. Each stage reduces the number of adders required. Again the LPM adders are used and the addition is also done in consideration that the data samples are signed.

Range selection block

The data captured at the input by the front end circuitry can contain some abrupt spikes. These spikes need to be separated from the data overflow. The data width after the data samples pass through the delay, apodization and the pipelined adder block becomes 32 bits. To enhance the system speed and utilize the bandwidth properly, the data is clipped from the lower bits to preserve the maximum value. This adds an error of 2^16 to the value.

In order to avoid confusion between spikes and valid data, we calculated the maximum value that can be represented in decided 16 bits and then the value in the 32 bit register from the pipelined adder is compared. If the expected value is less than the value obtained from the pipelined adder, the data sample value is clipped to a known value. For us, this results in a defined response and division between the spike and the data overflow.

4.3 Micro architecture

Micro architecture means low level description of the system. It involves the block level details showing the interfacing signals and their use. It involves the detailed description of the system and purpose of each component and the relation to other components. Figure 25 shows the micro architecture of UARP system including the FPGA based beamformer designed by us.

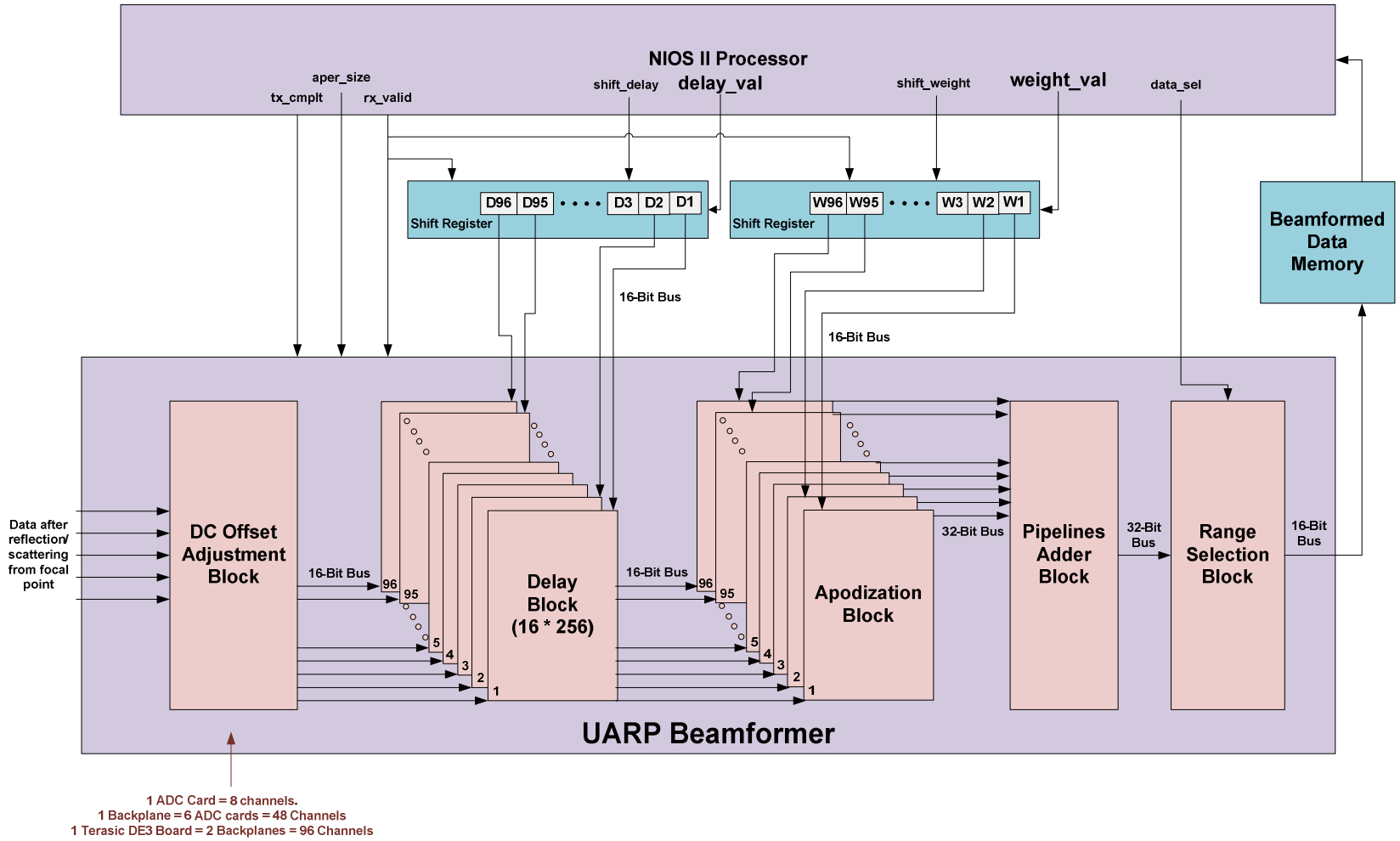

Figure 25 - Micro architecture of UARP beamformer

The beamformer design is completely our creation and has been developed to support flexibility to many diverse ultrasound modalities. The modules of the beamformer are defined and broken down to support and become part of not only the UARP system, but they make the beamformer as a standalone chip design. Following are the IOs (Inputs and outputs) of each of our coded modules for the implementation of delay and sum beamformer.

DC offset adjust block

The module (***load_dcoffset.v***) loads the offset values for each transducer element. For every power up, the transducer elements get misaligned with respect to other. This is adjusted by subtracting this offset from all the samples from a particular channel. The values for each channel are known as the offset. These offset values are loaded from the NIOS II processor into the STRATIX III FPGA system. Figure 26 shows the inputs and outputs of the delay and apodization loading block. The 'x' in the output ports is a substitution from 1 to 96 for 96 transducers.

Figure 26 - Inputs/outputs of DC offset adjust block

Table 2 shows the exact number of ports, their width, their direction and the purpose. 'x' is shown to denote the channel 1 to 96 (x = [1,2,3,…..96]).

The module (***sub_t.v***) is used by the *load_dcoffset.v* module. This is a user generated module which subtracts the offset value shown by the dc_offset_valx ('x' denotes the channels from 1 to 96. System has 96 such inputs, but for convenience only x is used as convention) from the data_in_chx ('x' denotes the channels from 1 to 96. System has 96 such inputs, but for convenience only x is used as convention). The output is shown by the data_out_chx ('x' denotes the channels from 1 to 96.). As each transmitter and receiver shares a single frame clock, thus they are used correspondingly. This is why 12 clocks are input to the *sub_t.v* module.

Table 2- Signal description of DC offset adjust block

#	Signal Name	Width	Direction	Description
1	shift_dcoffset	1	I	This signal is the shift register on basis of which the respective dcoffset registers for each channel are loaded.
2	dc_offset_val	16	I	This signal is the offset value from the NIOS II processor which is loaded to the respective dcoffset register.
3	load_dcoff_valx	16	O	This bus is the offset value for channel x.

Figure 27 shows the inputs and outputs of the delay and apodization loading block. The 'x' in the input and output ports is a substitution from 1 to 96 for 96 transducers. Table 3 shows the exact number of ports, there width, there direction and the purpose.

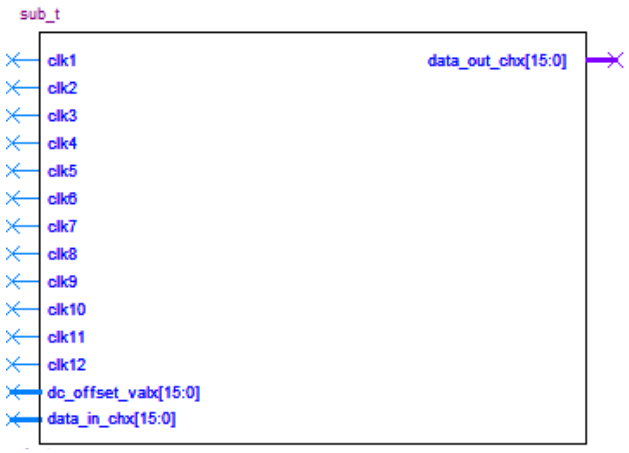

Figure 27 - Inputs/outputs of sub_t

Fixed delay and apodization loading block

The module (***initialize.v***) is used to load/ initialize the delay and apodization values for each of the channels. At every shift for the delay and the apodization, the values for each of the delay and apodization for each channel are shifted. This data is set for each channel before starting receiving the valid data. Figure 28shows the inputs and outputs of the delay and apodization

loading block. Table 4 shows the exact number of ports, their width, their direction and the purpose.

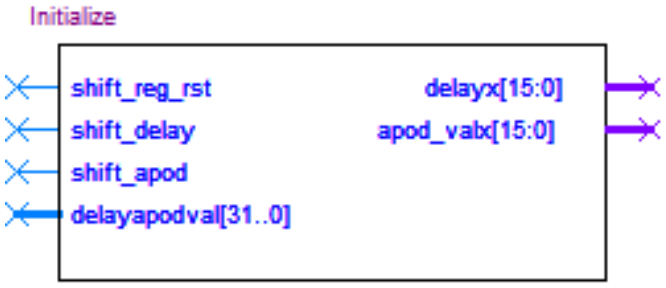

Figure 28 - Inputs/outputs of delay and apodization block

Table 3 - Signal description of sub_t block

#	Signal Name	Width	Direction	Description
1	clk1	1	I	Clock for the 1st 12 transducer elements.
2	clk2	1	I	Clock for the 2nd 12 transducer elements.
3	clk3	1	I	Clock for the 3rd 12 transducer elements.
4	clk4	1	I	Clock for the 4th 12 transducer elements.
5	clk5	1	I	Clock for the 5th 12 transducer elements.
6	clk6	1	I	Clock for the 6th 12 transducer elements.
7	clk7	1	I	Clock for the 7th 12 transducer elements.
8	clk8	1	I	Clock for the 8th 12 transducer elements.
9	clk9	1	I	Clock for the 9th 12 transducer elements.
10	clk10	1	I	Clock for the 10th 12 transducer elements.
11	clk11	1	I	Clock for the 11th 12 transducer elements.
12	clk12	1	I	Clock for the 12th 12 transducer elements.
13	dc_offset_valx	16	I	This bus is the offset value for channel x.
14	data_in_chx	16	I	This bus is the data from channel x.
15	data_out_chx	16	O	This bus is the offset adjusted data of channel x.

Table 4 - Signal description of delay and apodization load block

#	Signal Name	Width	Direction	Description
1	shift_delay	1	I	A signal that indicates the shifting of delay value across the shift registers chains for the delay values for each channel.
2	shift_reg_rst	1	I	A signal that resets the delay and apodization load registers to defaults.
3	shift_apod	1	I	A signal that indicates the shifting of apodization value across the shift registers chains for the apodization values for each channel.
4	delayapodval	32	I	A bus that contains the values for the delay and apodization. [31:16] - Delay Values. [15:0] - Apodization Values.
5	delayx	16	O	This signal contains the delay value for channel x.
6	apod_valx	16	O	This signal contains the apodization value for channel x.

Delay block

The module (***Delay_FIFO.v***) is used to generate the write_en and read_en logic for the memory for each channel. This function is called separately for each of the channel. Figure 29shows the inputs and outputs of the delay FIFO (First In First Out).

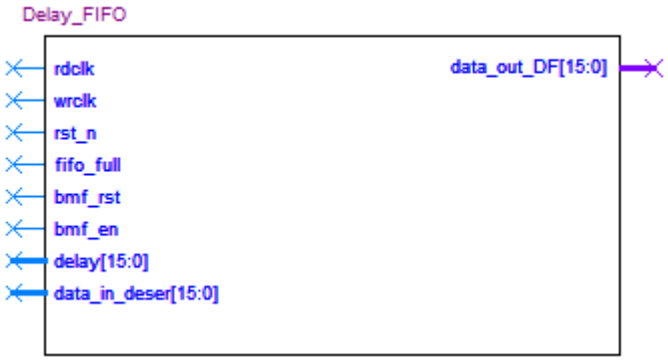

Figure 29 - Inputs/outputs of delay block

The Figure 30 shows the breakdown of the delay block into its basic components. It demonstrates that a delay counter is used to count down the delay value for a particular channel and a 2 X 1 multiplexer is used to select the data to be dropped at the output lines based on the value of the delay counter. Table 5 shows the data select values that the multiplexer must output based on the counter value.

Table 6 shows the exact number of ports, their width, their direction and the purpose.

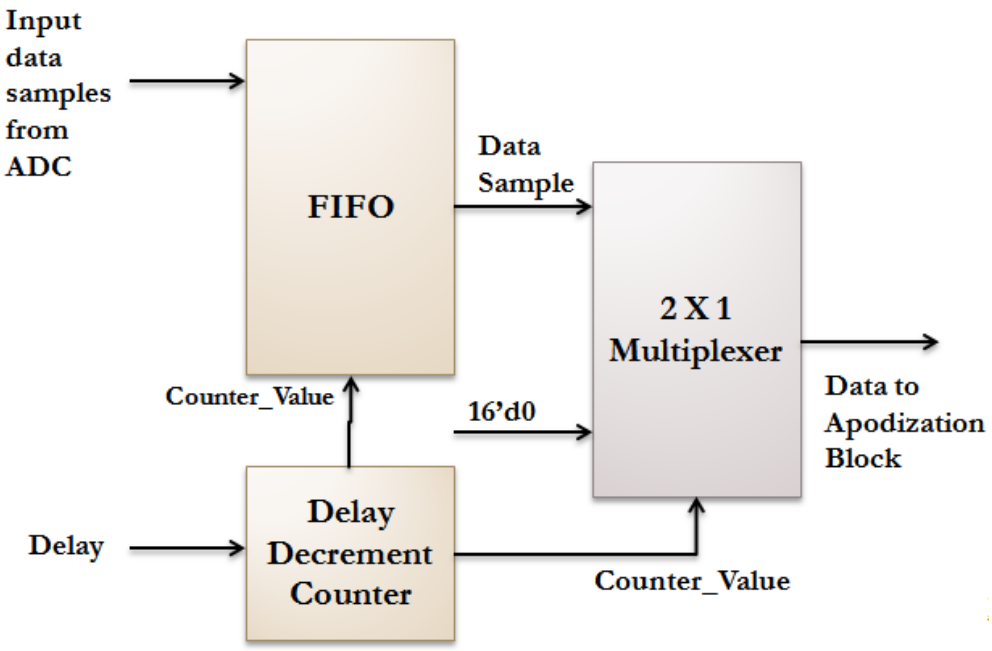

Figure 30 - Block breakdown of delay block

Table 5 - Data select values for delay block

Counter Value	Multiplexer Data Output
0	Data from FIFO
>0	16'd0

The module (FIFO_File.v) is used to generate the write and read pointer logic for the memory for each channel. This function is called by the Delay_FIFO function. Figure 31 shows the inputs and outputs of the FIFO wrapper.

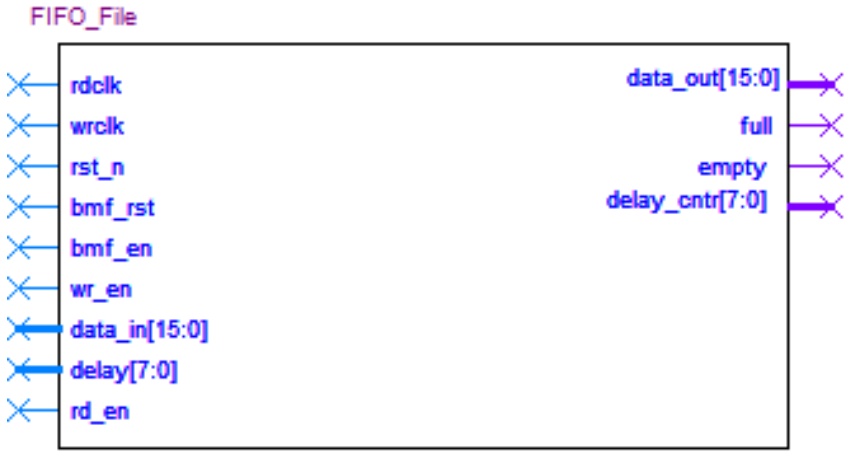

Figure 31 - Inputs/outputs of FIFO_File

Table 7 shows the exact number of ports, their width, their direction and the purpose.

Table 6 - Signal description of delay block

#	Signal Name	Width	Direction	Description
1	rdclk	1	I	This is the read clock generated by the system.
2	wrclk	1	I	This is the write clock generated by the system.
3	rst_n	1	I	This is the reset signal which acts at negative logic.
4	fifo_full	1	I	This signal shows the completion of a beamformer cycle. It indicates the memory to which the beamformed data is being written is full or not.
5	bmf_rst	1	I	This signal when asserted tells that the initialization values have been written for each channel.
6	bmf_en	1	I	This signal when asserted tells when the data being received is valid. It is the enabling signal for the beamformer.
7	delay	16	I	This signal contains the integer delay to be incorporated in the FIFO.
8	data_in_deser	16	I	This signal contains the data from the deserializer.
9	data_out_DF	16	O	This signal contains the data read from the FIFO.

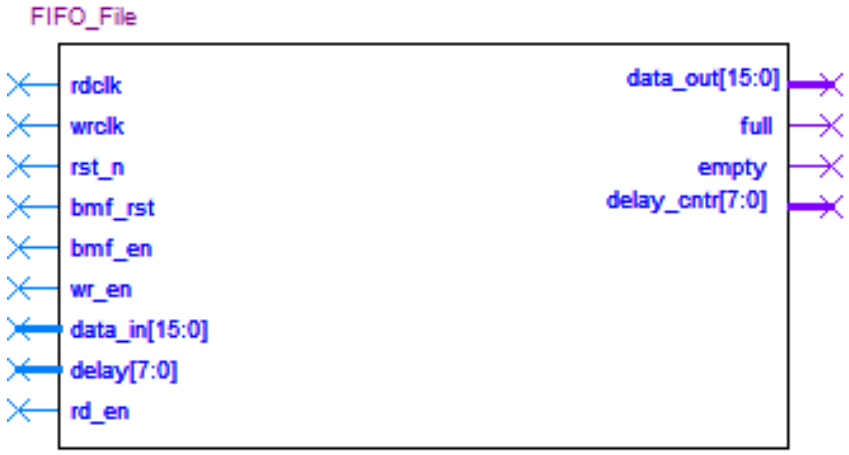

Figure 31 - Inputs/outputs of FIFO_File

Table 7 - Signal description of FIFO_File block

#	Signal Name	Width	Direction	Description
1	rdclk	1	I	This is the read clock generated by the system.
2	wrclk	1	I	This is the write clock generated by the system.
3	rst_n	1	I	This is the reset signal which acts at negative logic.
4	bmf_rst	1	I	This signal resets the memory with the delay s soon as the register values are written.
5	bmf_en	1	I	This signal when asserted tells when the data being received is valid. It is the enabling signal for the beamformer.
6	wr_en	1	I	This is the write enable signal for the memory.
7	data_in	16	I	This bus contains the data from the deserializer.
8	delay	16	I	This signal contains the integer delay to be incorporated in the FIFO.
9	rd_en	1	I	This is the read enable signal for the memory.
10	data_out	16	O	This bus carries the data out from the memory.
11	full	1	O	This signal is asserted when the memory is full.
12	empty	1	O	This signal is asserted when the memory is empty.
13	delay_cntr	8	O	This bus is a counter which counts the delay to zero, before the samples are read from the FIFO.

The module (***ram_2_port.v***) is a system generated function from the library of parameterized modules. It is a two port memory which uses separate read and write clocks. This function is called by the *FIFO_File* function. Figure 32 shows the inputs and outputs of the two ports RAM.

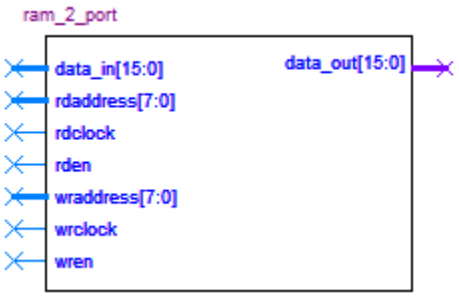

Figure 32 - Inputs/outputs of ram_2_port memory

Table 8 shows the exact number of ports, their width, their direction and the purpose.

Table 8 - Signal description of ram_2_port memory

#	Signal Name	Width	Direction	Description
1	data_in	16	I	This bus contains the data from the deserializer.
2	rdaddress	8	I	This bus shows the address to be read.
3	rd_en	1	I	This is the read enable signal for the memory.
4	rdclock	1	I	This is the read clock generated by the system.
5	wraddress	8	I	This bus shows the address to be written.
6	wrclock	1	I	This is the write clock generated by the system.
7	Wren	1	I	This is the write enable signal for the memory.
8	data_out	16	O	This bus contains the data read from the memory.

Apodization block

This module (***multiplier_t.v***) is used to implement the apodization/weighting of the data from each channel. Based on the aperture size and the aperture position, the data is multiplied with the weighting coefficients while the undesired data is weighted by coefficients having value of 0 to nullify the effect.

This block uses the Altera LPMs (Library of parameterized modules) for the implementation of multipliers. LPMs are the Altera designed most optimized modules for the specific functions. They are used for efficient design and memory utilization. Figure 33 shows the inputs and outputs of the multiplier_t (apodization block).

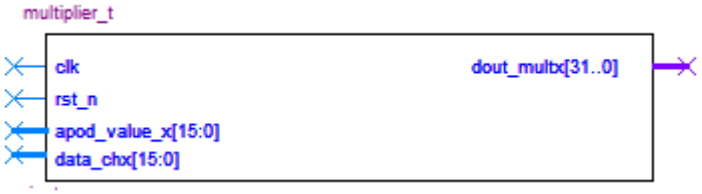

Figure 33 - Inputs/outputs of apodization block

Figure 34 shows the apodization block breakdown showing that linear parameterized modules (LPMs) are used to multiply the already loaded apodization coefficients with the incoming data value. The apodization block receives the apodization coefficient of zero from the NIOS II processor for all the channels which are not part of the current aperture of the transmission. When the zero valued coefficients are multiplied by the data sample, the contribution of that channel becomes zero towards the final data computation.

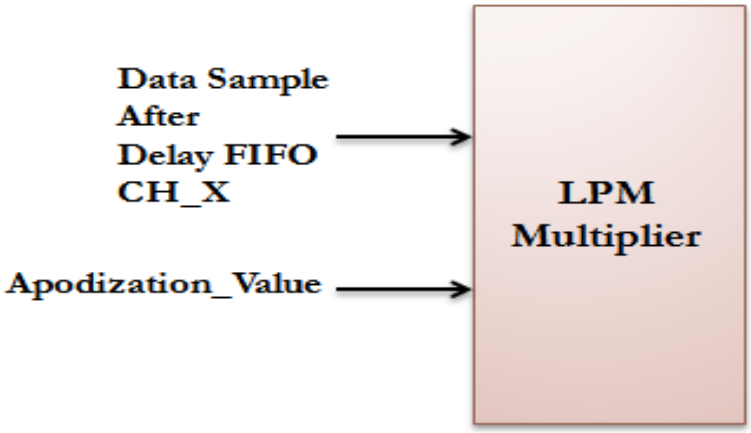

Figure 34 - Block breakdown of apodization block

Table 9 shows the values of the apodization coefficients based on the aperture size. These values are already set by the NIOS II processor and are just loaded in run time before the data reception starts.

Table 10 shows the exact number of ports, their width, their direction and the purpose.

Table 9 - Data select values for apodization block

Aperture Size	**Apodization Value From NIOS II Processor**
<X	16'd0
>=X	Apodization Coefficient

Table 10 - Signal description of multiplier block

#	Signal Name	Width	Direction	Description
1	Clk	1	I	This signal is the clock.
2	rst_n	1	I	This signal is the reset signal for the system.
3	apod_value_1	16	I	This bus shows weighting/apodization coefficient input to the apodization unit of channel 1.
4	data_chx	16	I	This bus shows the data input to the apodization unit of channel x.
5	dout_multx	32	O	This bus shows the output from apodization unit of channel x.

The module (***lpm_multiplier.v***) is the LPM instantiated which implements a signed multiplier. Figure 35 shows the inputs and outputs of the multiplier_t.

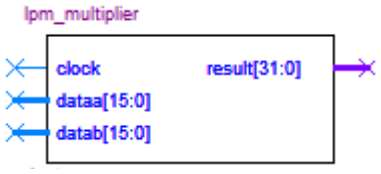

Figure 35 - Inputs/outputs of multiplier LPM

Table 11 shows the exact number of ports, their width, their direction and the purpose.

Table 11 - Signal description of mult LPM

#	Signal Name	Width	Direction	Description
1	clock	1	I	This signal is the clock.
2	data	16	I	This bus shows weighting/apodization coefficient input to the LPM.
3	datab	16	I	This bus shows data samples input to the LPM.
4	result	32	O	This bus shows the data output from the LPM to the pipelined adder.

Pipelined adder

This module (***adder_t.v***) is used to implement the pipelined adder. This block uses the Altera LPMs (Library of parameterized modules) for the implementation of adders. LPMs are the Altera designed most optimized modules for the specific functions. They are used for efficient design and memory utilization. Figure 36 shows the inputs and outputs of the adder_t.

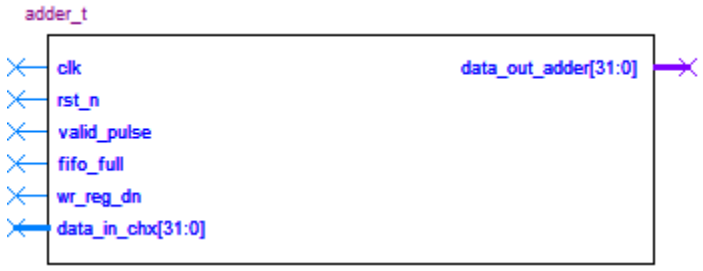

Figure 36 - Inputs/Outputs of adder block

Figure 37 shows the structure how the pipelined adder is designed. Here a 16 channel pipelined adder is demonstrated which shows 15 adders to be used and four pipelined stages. For the UARP having 96 channels, the maximum length of this adder for an aperture size of 96 becomes 7 stages with 95 adders.

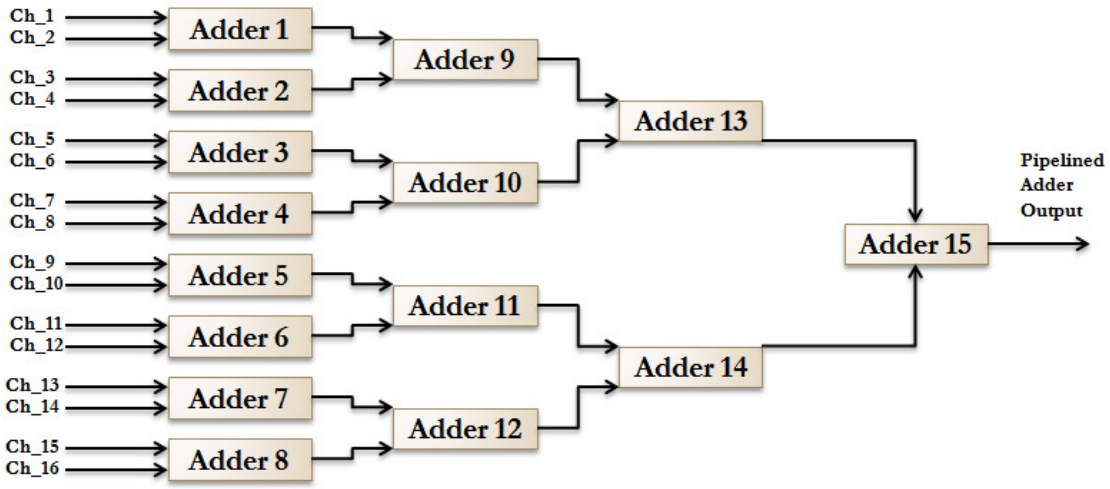

Figure 37 - Pipelined adder structure

Table 12 shows the exact number of ports, their width, their direction and the purpose.

Table 12 - Signal description of adder block

#	Signal Name	Width	Direction	Description
1	Clk	1	I	This signal is the clock.
2	rst_n	1	I	This signal is the reset signal for the system.
3	valid_pulse	1	I	This signal when asserted indicates that valid data has started to arrive at the receiver.
4	fifo_full	1	I	This signal when asserted indicates that the memory to which the beamformed data is written is full.
5	wr_reg_dn	1	I	This signal when asserted indicates that the register values for each channel has been initiated.
6	data_in_chx	32	I	This bus shows data from channel x to the adder unit.
7	data_out_adder	32	O	This bus carries the output of the adder.

The module (lpm_adder.v) uses the LPMs to implement the adder. It adds two 32-bit numbers and gives a 32-bit result. 32-bit addition is made in consideration to avoid any data overflow in the beamformer. Figure 38 shows the inputs and outputs of the lpm_adder.

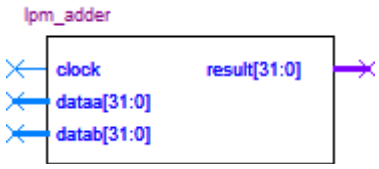

Figure 38 - Inputs/outputs of add LPM

Table 13 shows the exact number of ports, their width, their direction and the purpose.

Table 13 - Signal description of add LPM

#	Signal Name	Width	Direction	Description
1	clock	1	I	This signal is the clock.
2	Data	32	I	This bus shows data input to adder LPM.
3	Datab	32	I	This bus shows data input to adder LPM.
4	Result	32	O	This bus shows data output from adder LPM.

Range selection block

This module (*compare_t.v*) is used to select the 16 bits out of the 32 bits of data coming from the adder unit based on the value of the data_sel signal. This module avoids mixing of data spike or any abrupt signal changes between samples from the samples where the data exceeds the limit.

Figure 39 shows the inputs and outputs of the range selection block.

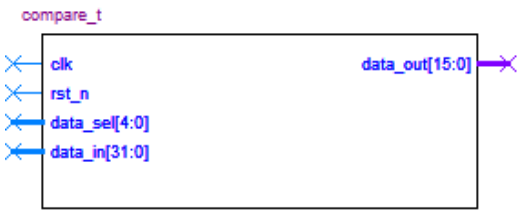

Figure 39 - Inputs/outputs of range selection block

Figure 40 shows how the 16 bit value is selected from the 32 bit output from the pipelined adder. A data select bus loads the start address from the NIOS II processor. The signed bit is appended

to reserve the sign of the data value. Table 14 shows the exact number of ports, their width, their direction and the purpose.

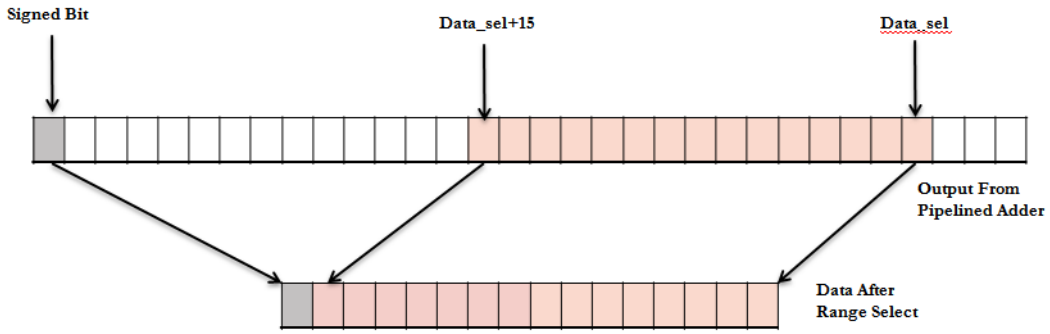

Figure 40 - Range selection by bits

This module (***lpm_comp.v***) is a compare LPM which compares the beamformes sample value to the greatest positive value that can be represented in valid 16 bits. It tells if the beamformed sample value is greater than the maximum value or not. Figure 41 shows the inputs and outputs of the lpm_comp block.

Table 14 - Signal description of range selection block

#	Signal Name	Width	Direction	Description
1	Clk	1	I	This signal is the clock.
2	rst_n	1	I	This signal is the reset signal for the system.
3	data_sel	5	I	This bus shows the reference after which the 16 bits are to be selected.
4	data_in	32	I	This bus shows the 32-bit data from the adder.
5	data_out	16	O	This bus shows the 16 bit output selected based on the data_sel value.

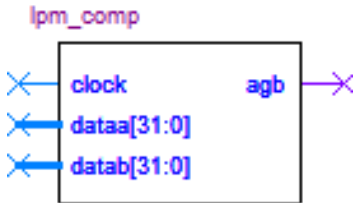

Figure 41 - Inputs/Outputs of =ive compare LPM

Table 15 shows the exact number of ports, their width, their direction and the purpose.

Table 15 - Signal description of +ive compare LPM

#	Signal Name	Width	Direction	Description
1	clock	1	I	This signal is the clock.
2	data	32	I	This bus shows data input to compare LPM.
3	datab	32	I	This bus shows data input to compare LPM.
4	result	32	O	This bus shows data output from compare LPM.

This module (***lpm_comp_neg.v***) is a compare LPM which compares the beamformed sample value to the most negative value that can be represented in valid 16 bits. It tells if the beamformed sample value is less than the minimum negative value or not. Figure 42 shows the inputs and outputs of the lpm_comp_neg block. Table 16 shows the exact number of ports, their width, their direction and the purpose.

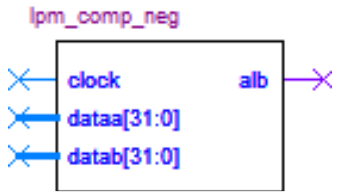

Figure 42 - Inputs/outputs of -ive compare LPM

Table 16 - Signal description of -ive compare LPM

#	Signal Name	Width	Direction	Description
1	clock	1	I	This signal is the clock.
2	dataa	32	I	This bus shows data input to compare LPM.
3	datab	32	I	This bus shows data input to compare LPM.
4	result	32	O	This bus shows data output from compare LPM.

As the range selection block truncates the lower 16 bits in the worst case, thus an error of 2^16 can be added in worst case. This is shown in Figure 43.

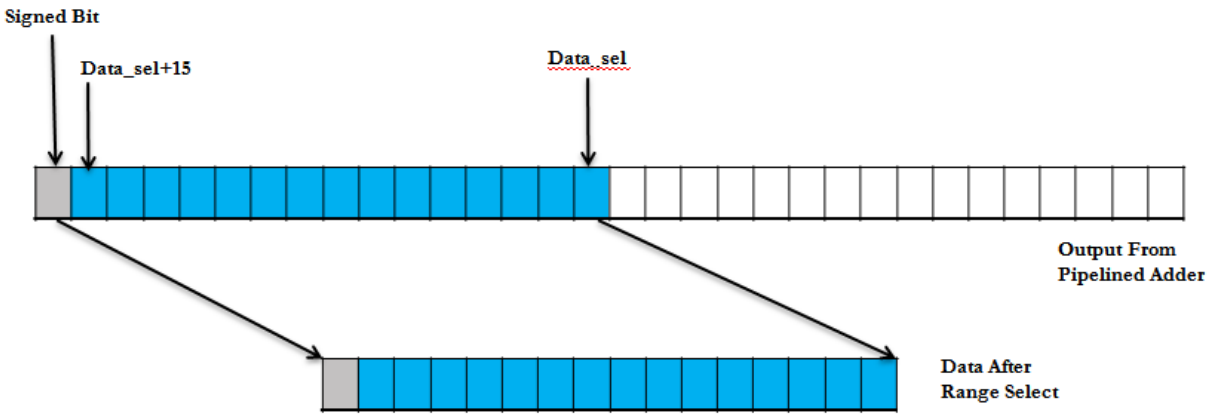

Figure 43 - Range selection error

Beamformed data memory

This module (***RAM_2Port.v***) is a RAM LPM which stores the values from the beamformed data path till a scan line is complete. This is the interface between the Matlab and the STRATIX III FPGA system. NIOS II processor is used by Matlab to access this memory using the USB 2.0 commands. Its depth is 2048 and width is 32 bits. Two samples are concatenated to increase the bandwidth efficiency while reading using the USB 2.0 link. Figure 44 shows the inputs and outputs of the lpm_comp_neg block.

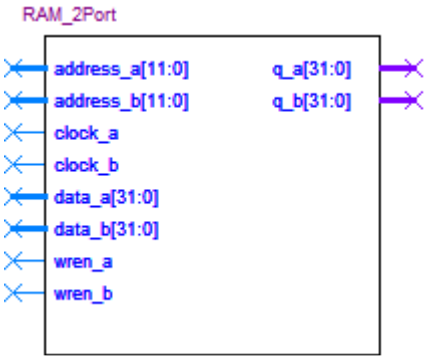

Figure 44 - Inputs/Outputs of beamformer memory

The above block division and implementation is our own design which comfirms flexibility and modularity to be adapted to any ultrasound system. Table 17 shows the exact number of ports, their width, their direction and the purpose.

Table 17 - Signal description of beamformer memory

#	Signal Name	Width	Direction	Description
1	clock_a	1	I	This signal is the port A clock.
2	clock_b	1	I	This signal is the port B clock.
3	address_a	12	I	This bus shows the address for port A.
4	address_b	12	I	This bus shows the address for port B..
5	data_a	32	I	This bus shows the 32 bit input data for port A.
6	data_b	32	I	This bus shows the 32 bit input data for port B.
7	wren_a	1	I	This signal is the write enable for port A.
8	wren_b	1	I	This signal is the write enable for port B.
9	q_a	32	O	This bus shows the 32 bit output data for port A.
10	q_b	32	O	This bus shows the 32 bit output data for port B.

4.4 Features of UARP delay and sum beamformer

UARP delay and sum beamformer is designed to incorporate maximum flexibility and modularity rather than being fixed for a particular implementation. It is designed in consideration to be easily altered without affecting any system around as shown in Table 18.

Table 18 - UARP beamformer features

S. No	UARP delay and sum beamformer features
1	DC Offset Removal
2	Real Time Delay Loading
3	Real Time Apodization Coefficient Loading
4	Real Time Data Range Selection
5	Pre-beamformed and Beamformed Data Path
6	Flexible Aperture Size
7	Flexible Number of Channels

4.5 Optimization for enhancing the UARP beamformer frame rate

The delay and sum beamformer system is further optimized to enhance the system frame rate as shown in Figure 45. The UARP FPGA based beamformer design with a single scan line memory results in reading a scan line in Matlab after every transmission. While the optimized structure results in placing number of memories. This number is according to the number of scan lines needed.

Memory access in hardware is an expensive operation. Every time the memory is read, there are a number of commands that are exchanged between the Matlab and the NIOS II processor interface on the FPGA based custom board. These commands are to be completed before the actual data can be read. This optimization is done to reduce this communication to save the time occupied by them. Now, the data is stored after every transmission on the FPGA memory and is read only once after all the scan lines are formed.
UARP being limited by the number of hardware resources in terms of number of memory bits restricted to test the system with an aperture size of 48 forming 49 scan lines. Any smaller aperture size could not be tested. The former beamformer design using a scan line memory could support any aperture size from 1 to 96 considering 96 transducer elements in UARP platform.

The optimization results in increased frame rate which is demonstrated in Chapter 6.

In this chapter, we have explained the proposed design of the FPGA based delay and sum beamformer. It also shows the motivation of using a specific design method over the others.

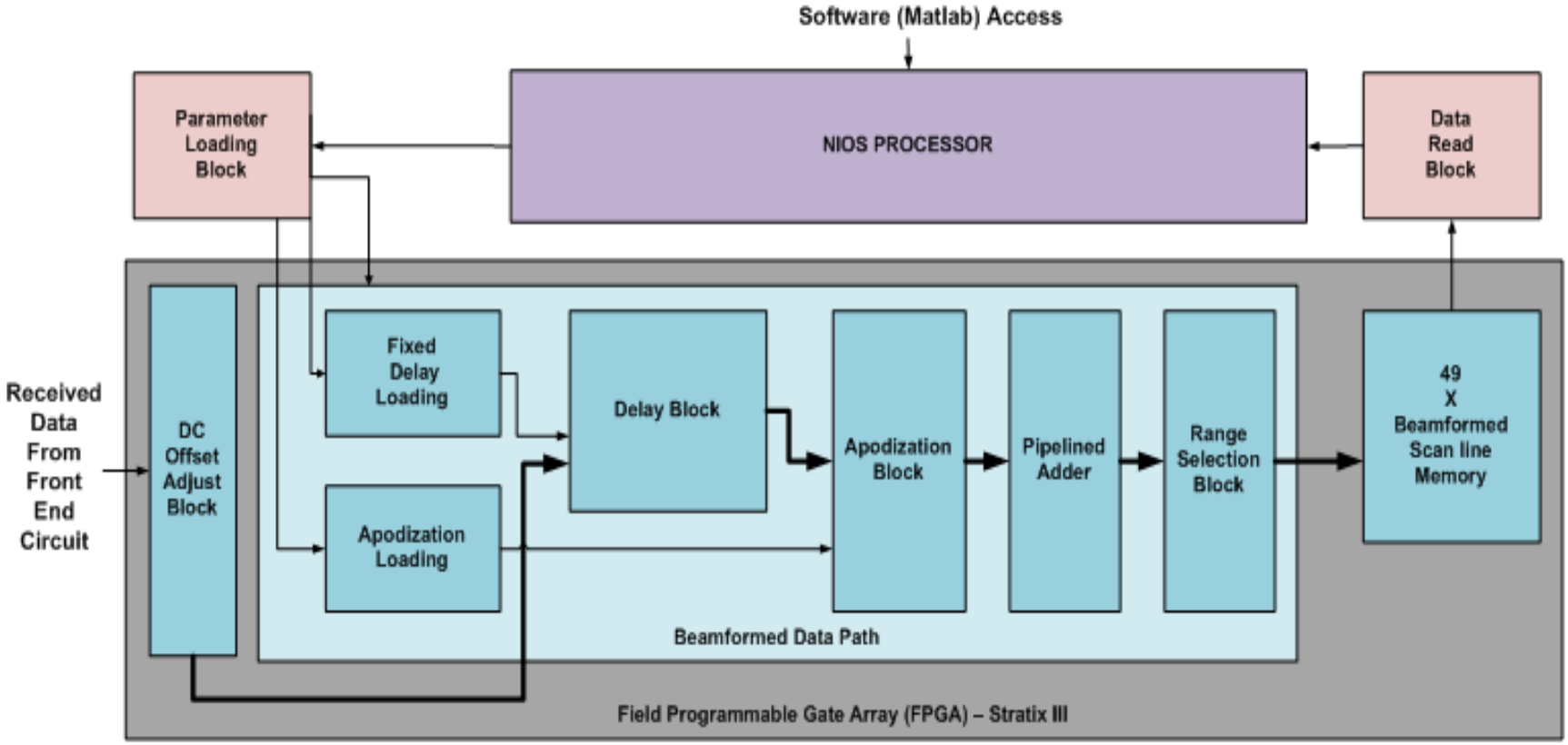

Figure 45 - Optimization to UARP beamformer

Chapter 5:

Dynamic Focusing

We designed, implemented and tested the UARP delay and sum hardware beamformer which supports fixed focus beamforming. It focuses at a single point in receive. Analysis and literature study [20-25] unfolded the use of a dynamic focusing technique for focusing in beamforming to enhance system resolution. Dynamic focusing increases the system focusing capacity equal to the number of samples involved.

5.1. Dynamic focus, multiple focus and fixed focus

Focusing can be divided into different types as described below.

Fixed focus beamforming means that the numbers of channels in the reception which are used according to the aperture size are combined together using a delay profile that focused all the samples in the channel at one point along the axial directions.

Multiple focus beamforming means that the full axial range in front of the transducer is divided into a pre-decided number of regions. And the samples which reflect a particular region are focused according to the consequent focal depth.

Dynamic focus beamforming stands different from the fixed and multiple focus cases in respect that the samples are focused for every point in the scan line. Every sample in the image corresponds to a depth in the focal region. Focusing at every sample focuses every pixel in the image. This results in fully dynamic focusing.

We carried out tests for fixed, multiple and dynamic focus using five wires connected at equal distance between two boards and then imaged then using the UARP system. Figure 46 shows the image formed by application of fixed focus delay and sum beamforming using Matlab. From left to right, the image is focused axially at 10mm, 20mm, 30mm, 40mm and 50mm respectively. We focused the wires perfectly by selecting the focal depths carefully. The focal points are carefully selected at the location of the wire. It is evident that only one point/wire is focused in each of the images and rest of the points/wires remain unfocused making them appear blurry and less likely as a perfect hole.

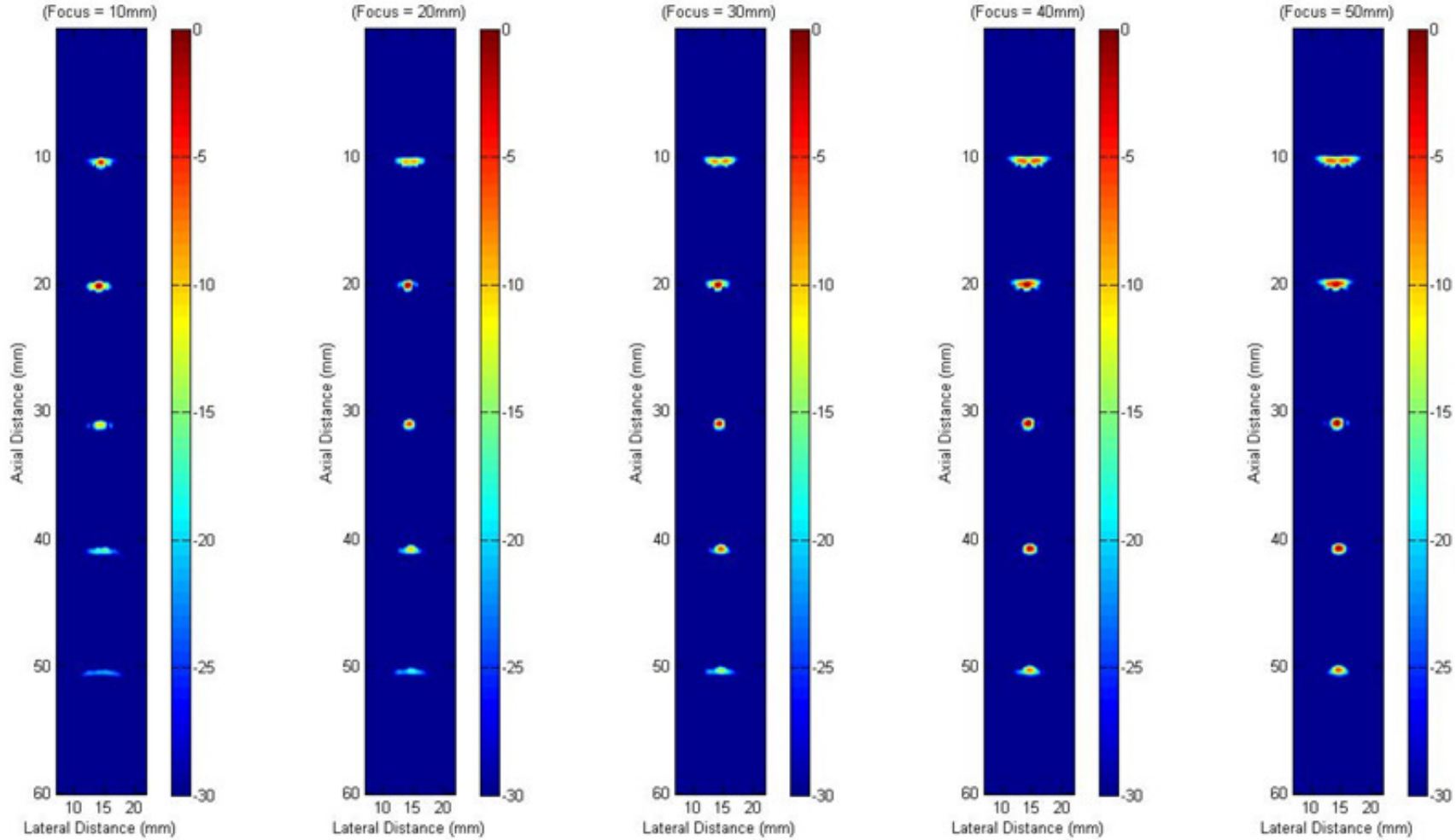

Figure 46 - Single focus delay and sum beamforming. (Left to right) The focal region is axially focused at 10mm, 20mm, 30mm, 40mm, and 50mm, respectively.

Figure 47 shows a multiple focused image. This image is formed by dividing the whole image depth into five regions known as focal regions as:

- Region 1 : 0 – 15mm
- Region 2 : 16-25mm
- Region 3 : 26-35mm
- Region 4 : 36-45mm
- Region 5 : 46-60mm

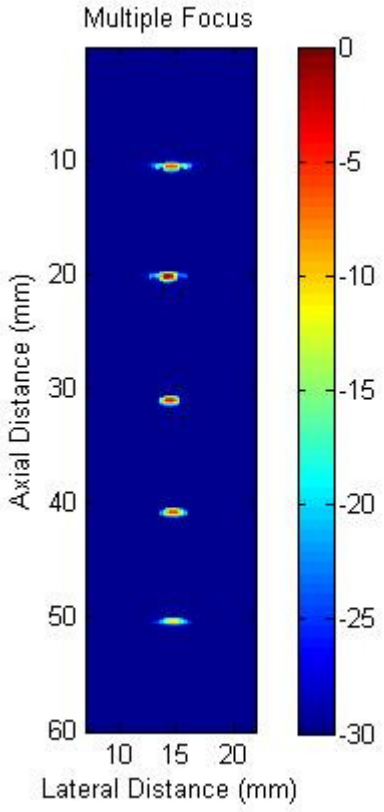

Figure 47 - Multiple focus image

We calculated the total number of samples for the maximum focal depth and then divided them in the above five regions. A focal point was decided in each region at the same location as of the wire/hole. The focal points for the regions as mentioned above are:

- Region 1 has its focal point at 10mm.
- Region 2 has its focal point at 20mm.
- Region 3 has its focal point at 30mm.

- Region 4 has its focal point at 40mm.
- Region 5 has its focal point at 50mm.

As all the valid imperfections lie at the five points where the wires actually are, thus the multiple focus image is focused at each and every wire which is represented as a hole in the image.

Figure 48 represents a dynamically focused image that we created using Matlab and the five wired test phantom. In comparison to multiple focus beamforming, we noticed that the results for the dynamically focused beamforming appeared same (Figure 49 and 50) but the difference lied in their operation. Multiple focused beamforming is a subset of dynamically focused beamforming where all the pixels in the image are focused. While a number of these pixels were decided and then focused in multiple focus. Later experiments will focus only on the fixed focus and the dynamic focus cases.

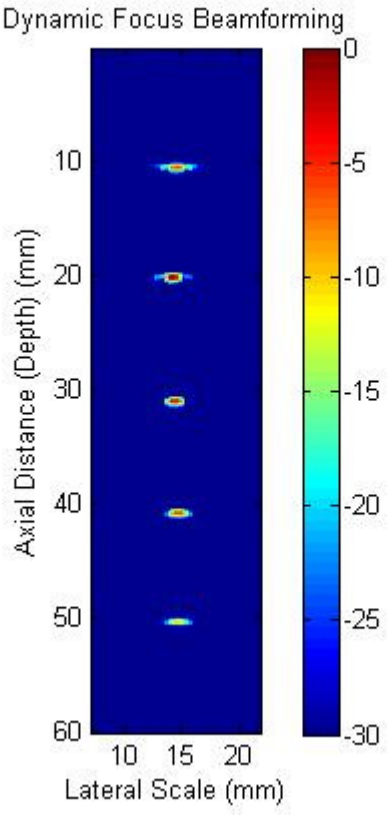

Figure 48 - Dynamic focus image

5.2. Dynamic focusing using UARP beamformer

The UARP fixed focus beamformer design is highly flexible as it supports variable aperture size, channel count, etc. In addition, we designed it to be highly modular to be adapted to newer technologies easily and without affecting the overall system infrastructure. Thus, newer imaging paradigms as dynamic focusing which results in increase in system resolution and focal depth can be easily adapted. For the software (Matlab) based testing, we modified the existing architecture to change its focusing technique from fixed to dynamic. The overall system block diagram with the fixed focus beamformer is shown in Figure 49.

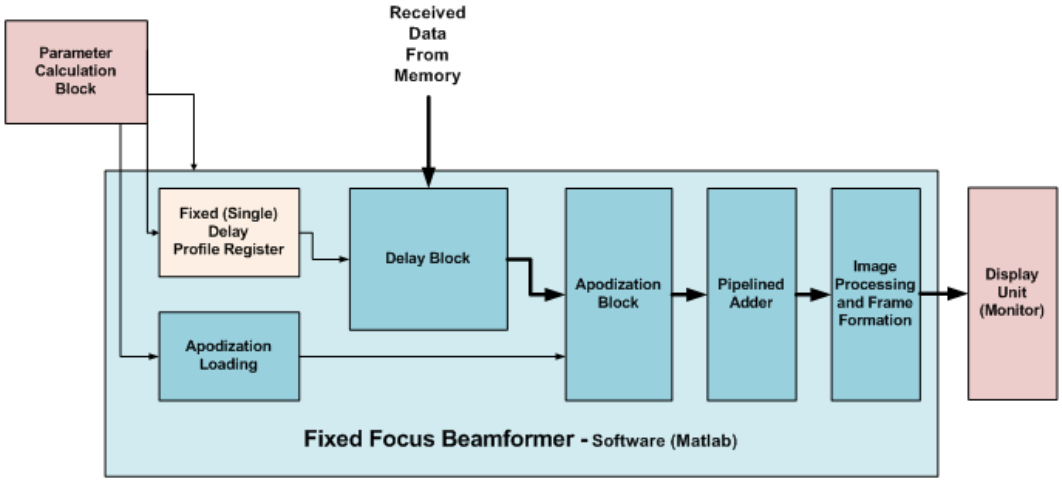

Figure 49 - Fixed focus delay and sum beamformer

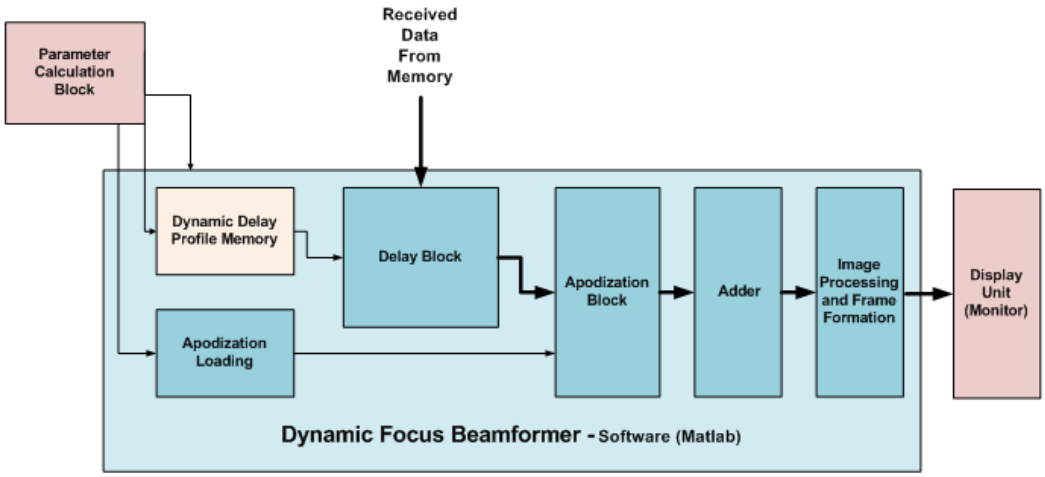

Figure 50 - Proposed dynamic focus delay and sum beamformer design

The literature [21-22] shows that dynamic focusing results in better system resolution and provides flexibility of focusing at every pixel. Thus, we tested the UARP delay and sum beamformer design with dynamic focusing. Figure 50 shows the beamformer design which we modified from fixed focus to dynamic focus. Studying the dynamic focusing technique, we identified that the only change is that in the fixed focus design the delay is same for all the samples for an aperture size but in dynamic focus design, the delay profile changes for every sample in the delay profile. The delay profile for all the samples in an aperture is different while the same delay profile set is used for every aperture. Thus, all the delay profiles need to be stored in a memory which we termed as 'Dynamic delay profile memory'as shown in Figure 50.

We ran a couple of tests in Matlab. Considering the aperture size of 24, if the focal depth requires 1000 samples to be manipulated, 1000 delay profiles need to be generated. While moving along the transducer, the same 1000 delay profiles are used for every aperture.

The same UARP design can be used with just modification to the delay generation and loading. An additional change we noticed was that now instead of FIFOs, memory will be required because the samples are not need to be added coherently in sequence as they are received. We had to design an address generationg technique to perform testing using dynamically focused beamforming. Although the memory demands for the delay profiles storage have increased, but the overall system remains same. This small increase in space requirement results in much enhanced image quality and signal to noise ratio.

Our tests and their results showed that the same UARP beamformer design is flexible enough that small changes can be made for it to be used as a dynamically focused beamformer.

5.2.1 Delay profile generation

We generated the delay profiles in dynamic focusing by the similar formula as used in fixed focus beamforming from [35]. Instead of generating a single delay profile for a fixed focal depth, the region in front of the transducer array is divided into X number of samples. Here, X stands for the number of samples required getting sufficient data to image at the maximum focal depth.

Thus, each sample needs to be focused. So equal number of delay profiles as the number of samples are generated and then applied to the incoming samples one by one.

Delay generation is done using the following formula [35]:

$$Delay_{Profile} = \frac{1}{\mathrm{c}} * \left(sqrt\left(\mathrm{R}^2 + \left(\frac{(\mathrm{N}-1)*\mathrm{d}}{2} \right)^2 + (\mathrm{N}-1)*\mathrm{R}*\mathrm{d}*\sin\left(\mathrm{abs}(\mathrm{thetas})\right) \right) - sqrt\left(\mathrm{R}^2 + \left(\frac{\mathrm{n}-(\mathrm{N}+1)*\mathrm{d}}{2} \right)^2 - 2*n*\mathrm{R}*\mathrm{d}*\sin\left(\mathrm{abs}(\mathrm{thetas})\right) \right) \right)$$

Where:

c	Speed of light
d	Element to element spacing
R	Radial distance to focal point
thetas	Steering angle
tmin	Minimum step for quantization

The above equation is used to generate the delay profile. This delay profile is converted according to the 20ns time step used. Thus, the resolution of the system is 20ns. After rounding, these values are utilized by the beamformer. In the fixed focus case, R represents the distance from the center element of the aperture to the focal depth, while in the dynamic case, it is the focal step that represents the distance between the two successive samples. The delay calculation is done using Pythagoras theorem as shown in Figure 51.

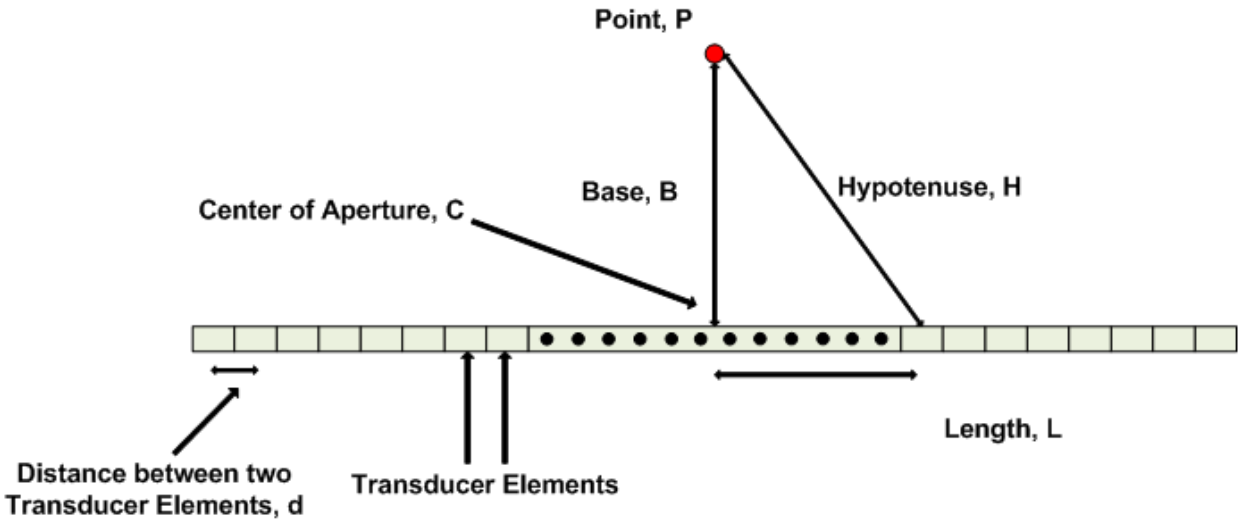

Figure 51 - Delay calculation procedure

In Figure 51:

Point P Focal point

B Distance from center of transducer array to focal point

H Distance from the transmission element to the focal point

L Length is the distance from the center of the transducer array to the center of the transmission element.

This chapter explained the importance and need of dynamic focusing and its implementation. We emphasized on the transitional differences from the fixed focused beamforming to the dynamically focused beamforming with demonstration using comparisons of how dynamically focused beamforming yields better resolution/results.

Chapter 6:

Results and Discussions

This chapter explains the results for the methodologies/ procedures explained in Chapter 4 and chapter 5. It is divided into two basic sections:

- Results for FPGA based delay and sum beamformer.
- Results for dynamic focusing.

6.1 UARP test blocks

Ultrasound array research platform (UARP) is a test environment simulating the ultrasound system and its surroundings to create a system to test and do studies of different ultrasound procedures and techniques. Test block A and test block B are shown in Figure 52.

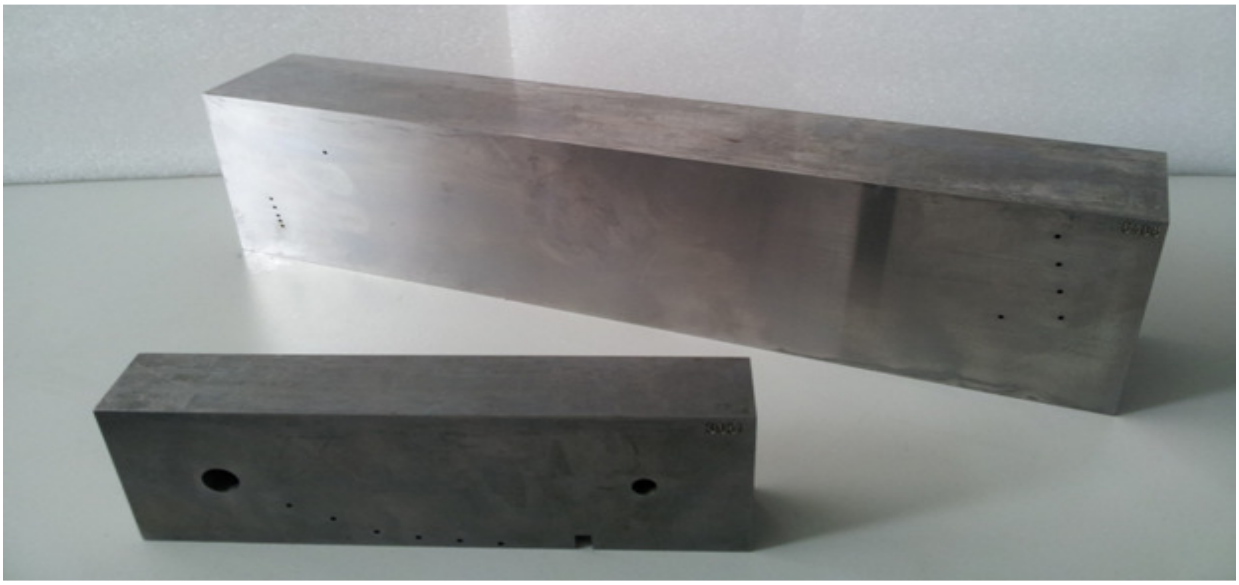

Figure 52 - UARP test blocks. Block A (below) and block B (above).

6.2 Software (Matlab) Results –Delay and sum beamformer

We implemented a flexible and easy to use modular beamformer in Matlab to test the functionality of the beamformer before its testing on UARP. We designed it exactly similar to the hardware based beamformer for a fair comparison and results. We generated the delay

profiles initially and stored them in the Matlab memory. These profiles are accessed by the Matlab files during operation just as the profiles already loaded in case of hardware beamformer.

We did testing in software (Matlab) by:

- Capturing real time data from the UARP platform.
- Collecting dummy data sets from UARP and then beamform in Matlab.

We noticed that both the real time and the dummy data sets give the same results. It is always a good practice to initiate software tests before testing the design on hardware. Figure 53 shows the image that we generated with an aperture size of 24 using UARP real time data samples.

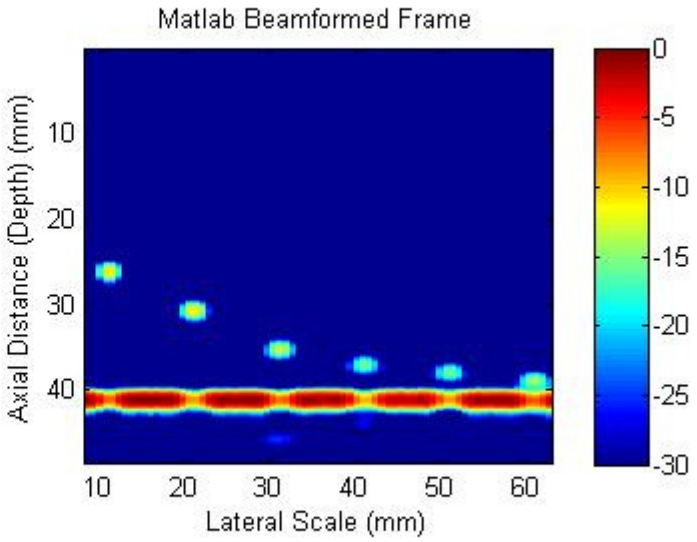

Figure 53 - Matlab generated beamformed image. 24 aperture size and 73 scan lines. The scale is converted to equivalent millimeters.

6.3 Hardware results - FPGA based delay and sum beamformer

The hardware results are divided into two sections:

- FPGA based beamformer with a single scan line and data access after every transmission.
- FPGA based beamformer with multiple scan line memories in hardware and data access at frame generation.

6.3.1 - FPGA based beamformer with a single scan line memory

The design which we have explained in the above sections generated a single scan line after every transmission and this scan line is accessed and stored in Matlab before the next transmission is done. In this section, we will demonstrate the results for this design.

Resource Utilization

Table 19 shows the our resources analysis of the UARP FPGA based beamformer as compared to the Matlab based beamformer in STRATIX III FPGA custom system. We designed the testing structure to capture the data from UARP front end, digitize it, store it in the memory and then make it available in Matlab for beamforming. Whereas, in UARP hardware based beamformer we capture the data, digitize it, and then perform beamforming in FPGA. These beamformed scan lines are stored to memory, from where Matlab accesses it to produce images/frames.

We noticed that the logic cell utilization is 15% more in case of the UARP beamformer as compared to the Matlab based beamformer, while the memory requirement of UARP beamformer is just 3% more than that of the Matlab based beamformer. Although, more and more resources are exhausted in UARP based beamformer, still the advantages of having hardware based beamformer results in an increase in system speed.

Table 19 - Resource utilization summary of Matlab beamformer compared to UARP beamformer

Resource Name	Total Resources	Matlab Beamformer	UARP Beamformer
Logic utilization	100%	20%	35%
Combinational ALUTs	270,400	36,743 (14%)	69,502 (26%)
Dedicated Logic Regs	270,400	27,775 (10%)	50,422 (19%)
Block Memory Bits	16,662,528	7,984,032 (48%)	8,442,784 (51%)
DSP Block	576	4 (<1%)	4 (<1%)

Image quality

We tested the performance of Matlab and FPGA based beamformer to analyze the image quality and found approximately no difference as shown in Figure 54. Figure 56 shows that the two images/frames, created one (Figure 54-A) by the UARP hardware based beamformer and the other (Figure 54-B) by the Matlab based beamformer.

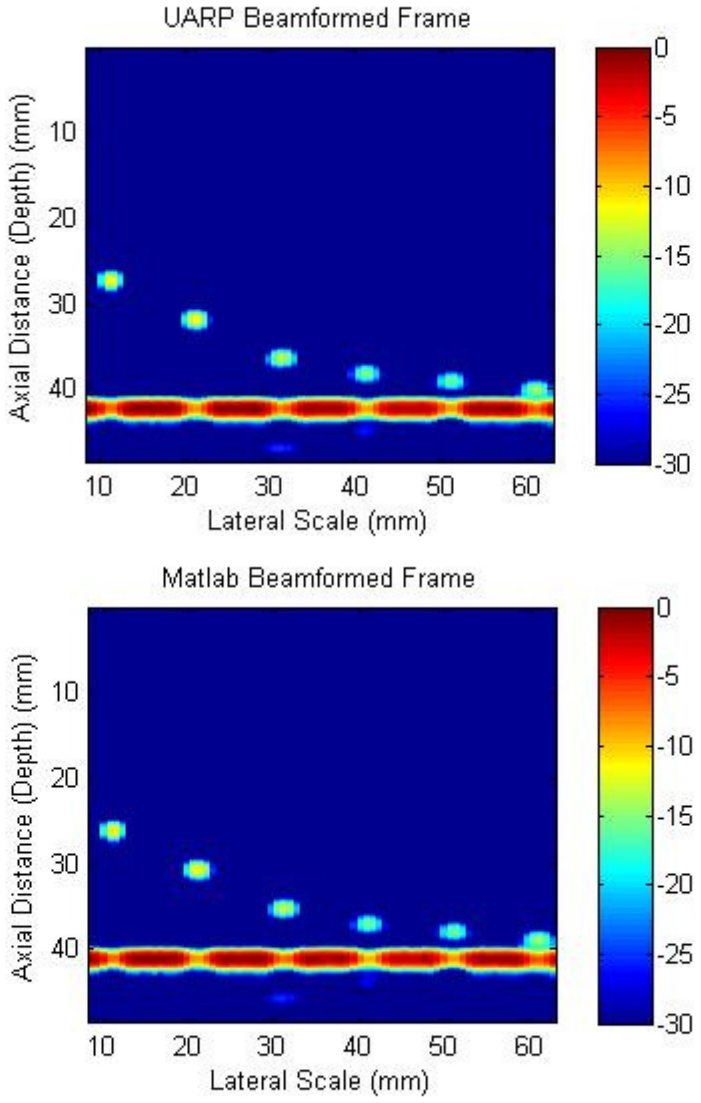

Figure 54 - UARP beamformed frame (above (A)) and Matlab beamformed frame (below(B))

In testing the image quality (as in Figure 54), we processed the frame using image processing techniques to create good quality images.

The image processing steps are listed below:

- Filtering : This removes any unwanted component from the signal. This unwanted signal is removed which lie out of the signal frequencies.
- Envelop detection: it uses an envelop detector which takes the signal and produces its envelop.
- Normalization: This is limiting the signal strength from 0 to 1 to enhance system flexibility for plotting and analysis.
- Log compression: It is taken to ease out calculations as all the numbers are then relative to each other.

The statistics of the ultrasound system during beamforming are given in Table 20.

Table 20 - Test statistics

No.	Property	Value
1	Number of Elements	96
2	Speed of Sound	5400
3	Element Spacing	7.5mm
4	Aperture Size	24
5	Number of Scan lines	73
6	Minimum Quantization Step	20nm
7	Steering Angle	0
8	Focal Depth	45e-3m
9	Number of Samples	829
10	Z_Offset	10mm

In order to compare the images created using both the Matlab (Software) and UARP (Hardware) beamforming, we took the absolute difference between the images created. The images are the normalized images after filtering the scan lines as shown in Figure 55. We kept the testing conditions for both the images same as in Table 20.

Figure 56 shows the difference of the normalized images shown in Figure 55. This is the difference image of the normalized Matlab and the UARP based images above. The original data

range is from 0 to 4500 approximately as also obvious from the image. The difference in the two frames, one created from the Matlab based beamforming and other created from the UARP based beamforming, is less than 10% of the total data range which is negligible.

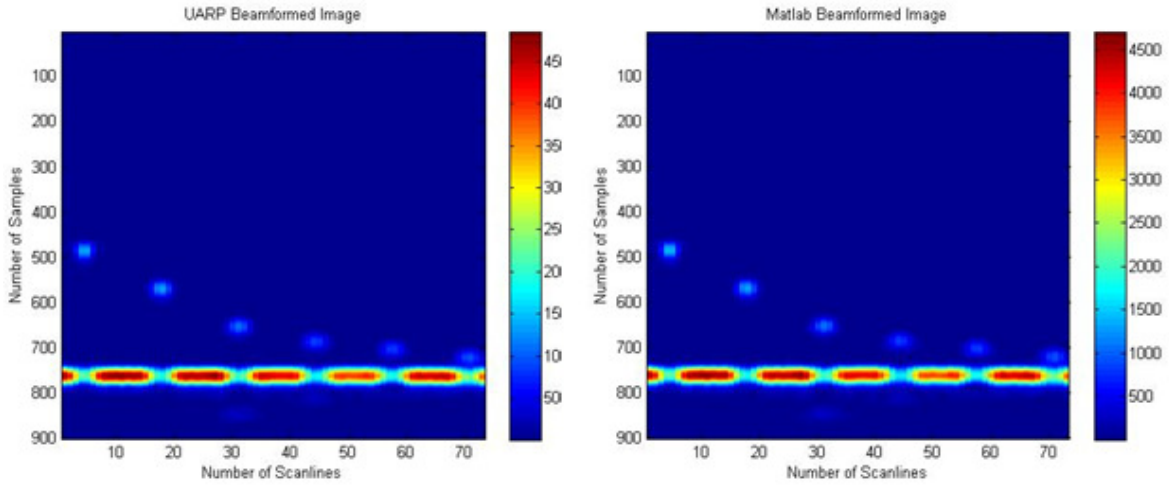

Figure 55- UARP beamformed frame (Left) and Matlab beamformed frame (Right). These are normalized images without log compression

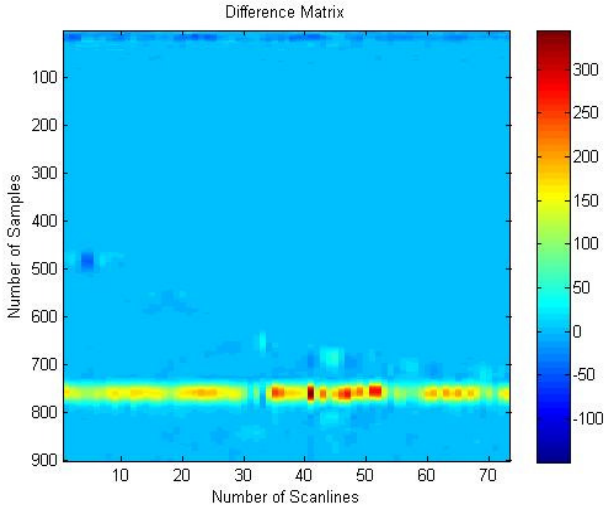

Figure 56 - Difference image between UARP beamformed frame and Matlab beamformed frame

Frame rate

For analysis, in addition to the image quality, we took frame rate as another parameter. Frame rate is a very valuable parameter for real time systems.

We noticed that the frame rate generated by the UARP FPGA based beamformer is approximately 3 times that of the Matlab based beamformer as shown in Table 22. This increase makes the UARP system more viable for real time measurements involving the measurement and analysis of moving objects.

We also identified that the effect of increasing the aperture size in Matlab based beamformer adversely effects the frame rate while it is again a positive gain in case of the UARP FPGA based beamformer as evident from the measurements shown in Table 21. The decrease in the frame rate in case of the Matlab based beamformer even after increasing the aperture size is because although the number of scan lines have decreased but the overall number of calculations per scan line have increased, which enhances the overhead in software. Whereas in the UARP beamformer, as hardware deals signals/samples in parallel, thus the increase in aperture size does not affect the system speed and thus increases the frame rate. This is what we noticed after repeated tests and measurements.

Table 21- Frame rate comparison between UARP beamformer (Single scan line memory) and Matlab beamformer

Aperture Size	**Matlab Beamformer (Frames per second)**	**UARP Beamformer (Frames per second)**
24	0.81	2.41
48	0.66	3

6.3.2 FPGA based beamformer with multiple scan line memories

FPGA based beamformer with multiple scan line memories is the beamformer where the scan line is produced after every transmission and is stored in the FPGA memory before the next

transmission. All the transmissions are done and the scan lines are stored in the memory on FPGA. After all the scan lines are produce for the corresponding aperture size, Matlab reads the FPGA memory once by using the bulk data access instead of reading one scan line per transmission in the case of single scan line memory.

Reading a hardware memory space from software required many initial commands to setup the connections and then access the data. The setup of connections in every transmission results in increased bandwidth absorption and time. This is reduced by optimizing the beamformer design by accessing the memory for all scan lines once after all the transmission increasing the system speed and frame rate. Using UARP, this concept has been tested with an aperture size of 48, and a total of 49 scan lines were stored and read by Matlab. Other smaller aperture sizes could not be tested as the memory resources were almost exhausted for implementation of more memories.

Resource utilization

Table 22 shows the comparison of resources between the Matlab based beamformer and the UARP beamformer with multiple scan line memory with 48 aperture size and 49 scan line using UARP. Statistics shows that UARP beamformer with multiple scan line memories uses about 23% more logic cells and 21% more memory bits as compared to the Matlab based beamformer using UARP platform.

Table 22 - Frame rate comparison between UARP Beamformer (Multiple scan line memories) and Matlab beamformer

Resource Name	Total Resources	Matlab Beamformer	UARP Beamformer (Multiple Scan line)
Logic utilization	100%	20%	43%
Combinational ALUTs	270,400	36,743 (14%)	86,725 (32%)
Dedicated Logic Regs	270,400	27,775 (10%)	58,047 (21%)
Block Memory Bits	16,662,528	7,984,032 (48%)	11,580,320 (69%)
DSP Block	576	4 (<1%)	4 (<1%)

Table 23 shows the comparison of resources between the UARP based beamformer with single scan line memory (48 aperture size and 49 scan line in UARP) and the UARP beamformer with multiple scan line memory (48 aperture size and 49 scan line in UARP). Statistics shows that UARP beamformer with multiple scan line memories uses about 8% more logic cells and 18% more memory bits as compared to the UARP based beamformer with single scan line memory using UARP platform.

Table 23 - Frame rate comparison between UARP Beamformer (Single scan line memory) and UARP Beamformer (Multiple scan line memories)

Resource Name	**Total Resources**	**UARP Beamformer (Single Scan line)**	**UARP Beamformer (Multiple Scan line)**
Logic utilization	100%	35%	43%
Combinational ALUTs	270,400	69,502 (26%)	86,725 (32%)
Dedicated Logic Regs	270,400	50,422 (19%)	58,047 (21%)
Block Memory Bits	16,662,528	8,442,784 (51%)	11,580,320 (69%)
DSP Block	576	4 (<1%)	4 (<1%)

Image quality

The image quality is not affected by the way the scan lines are saved and accessed from Matlab. Thus, there is no difference in terms of image quality in the implementations of UARP beamformer with single or multiple scan line memories.

Frame rate

UARP based beamformer with multiple scan line memories (48 aperture size and 49 scan line in UARP) uses bulk data access method for reading the scan lines from Matlab instead of reading a single scan line after every transmission, saving the system bandwidth and thus increases the system speed. This increase in speed directly affects the frame rate of the Ultrasound system (UARP).

Table 24 shows that the UARP beased beamformer as compared to the Matlab based beamformer shows an overall increase in frame rate. The UARP beamformer results in a frame rate of approximately 11 in case of single scan line memory and 32 in case of multiple scan line memories. Thus, the proposed optimization will enhance the system frame rate making the UARP platform feasible for fast data acquisition and motion estimation.

Table 24 – Frame comparison between Matlab beamformer, UARP Beamformer (Single scan line memory) and UARP Beamformer (Multiple scan line memories)

Aperture size	**Matlab beamformer (Frames per second)**	**UARP beamformer (Frames per second)**
48 (Single scan line beamformer)	~1	10.61
48 (Multiple scan line beamformer)	~1	32

6.3.3 Discussion – FPGA based UARP delay and sum beamformer

The results and analysis in the previous sections show that implementing the delay and sum beamformer in the hardware results in enhancing the overall system speed. This increases the frame rate.

Optimizations to the system design can result in huge frame rates which comply to latest measurements of moving particles.

6.4 Results – Dynamic focusing

A couple of tests were carried out in the Ultrasound laboratory in University of Leeds, UK using the test blocks A and B. Three tests show the qualitative difference between the fixed focus delay and sum beamforming and the dynamic focus delay and sum beamforming. For the quantitative analysis, the results for the signal to noise (SNR) ratio of both the techniques have been compared. All the experimental setups have specific statistics as focal depth, focal point in case

of the fixed focus delay and sum beamforming, number of samples, speed of sound, etc. All these parameters are well stated to clarify the experimental setup and environment.

6.4.1 Tests – Dynamic Focusing

Table 25 shows the statistics of the test experients 1 and 2. Each experient has the data statistics for fixed focus delay and sum beamforming as well as dynamic focus beamforming. In Fixed focus case, there is a single focal depth and in dynamic focus, the focal depth is divided into focal steps e.g. 5.43e-5m in these tests.

Table 25 – Statistics for Test_1 and Test_2

S. No	Property	Values for Test_1		Values for Test_2	
1	Number of Elements	96		96	
2	Speed of Sound	5431		5431	
3	Element Spacing	7.5mm		7.5mm	
4	Aperture Size	24		24	
5	Number of Scan lines	73		73	
6	Minimum Quantization Step	20nm		20nm	
7	Steering Angle	0		0	
8	Radial distance to focal point	Fixed	Dynamic	Fixed	Dynamic
		25mm	5.43e-5m	50m	5.43e-5m
9	Number of Samples	1105		1105	
10	Z_Offset	10mm		10mm	
11	Total Focal Depth	60mm		60mm	

Figure 57 shows the delay profiles generated for the Tests according to the data statistics of Table 25. In fixed focus case, a single delay profile is generated according to the single focal depth (Figure 57 (A and C)), but for the dynamic focus, a new delay profile at every focal step in the axial direction is created (Figure 57 (B and D)).

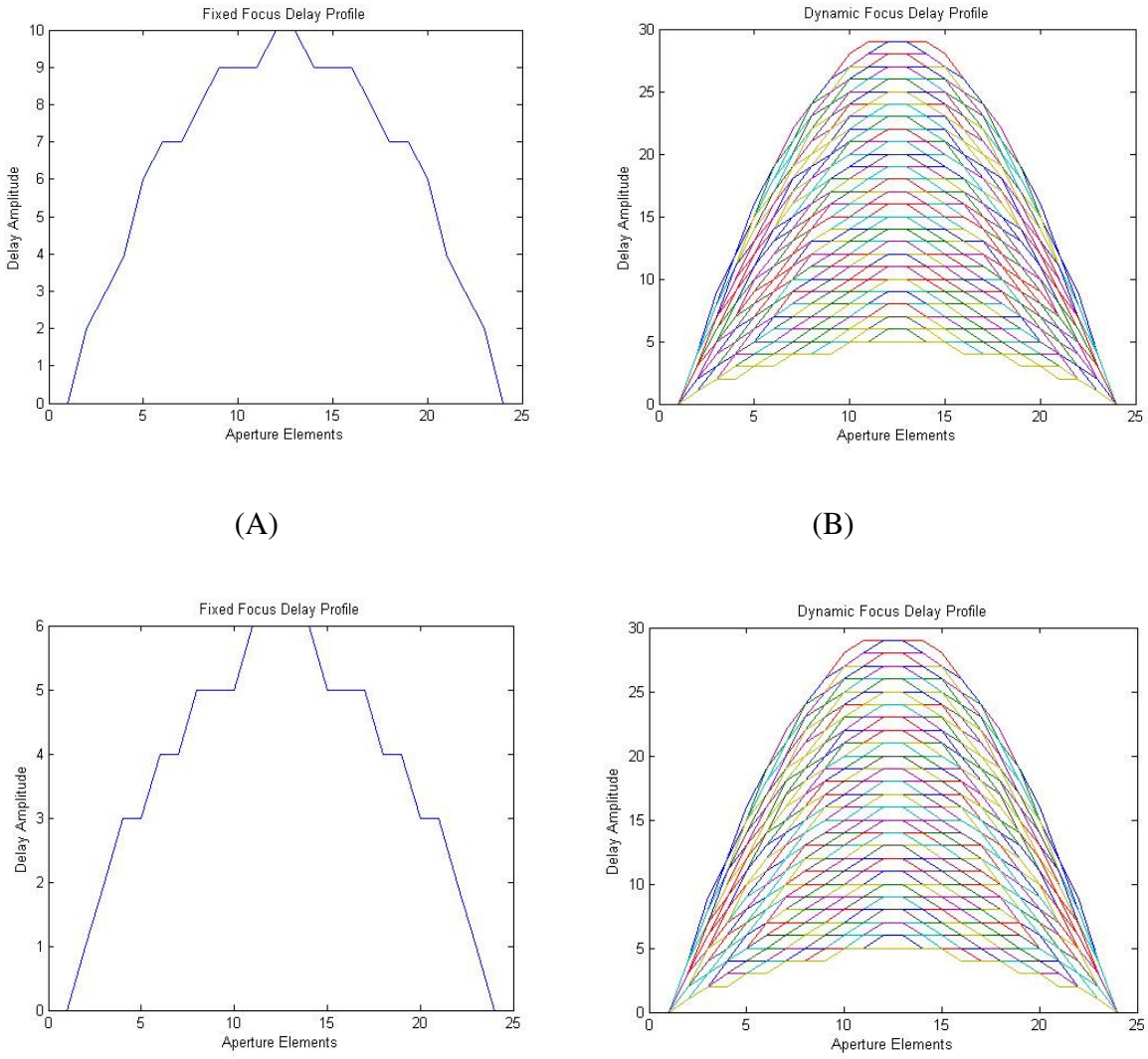

Figure 57 - Delay profile for Test_1 and Test_2 – Test_1 (Fixed focus profile (A) and Dynamic focus profile (B)) and Test_2 (Fixed focus profile (C) and Dynamic focus profile (D))

Figure 58 shows the images generated after beamforming using the fixed (Figure 58 (A and C)) and dynamic focus beamforming (Figure 58 (B and D)). We observed that the system resolution has increased by application of dynamic focusing as compared to the fixed focus beamforming.

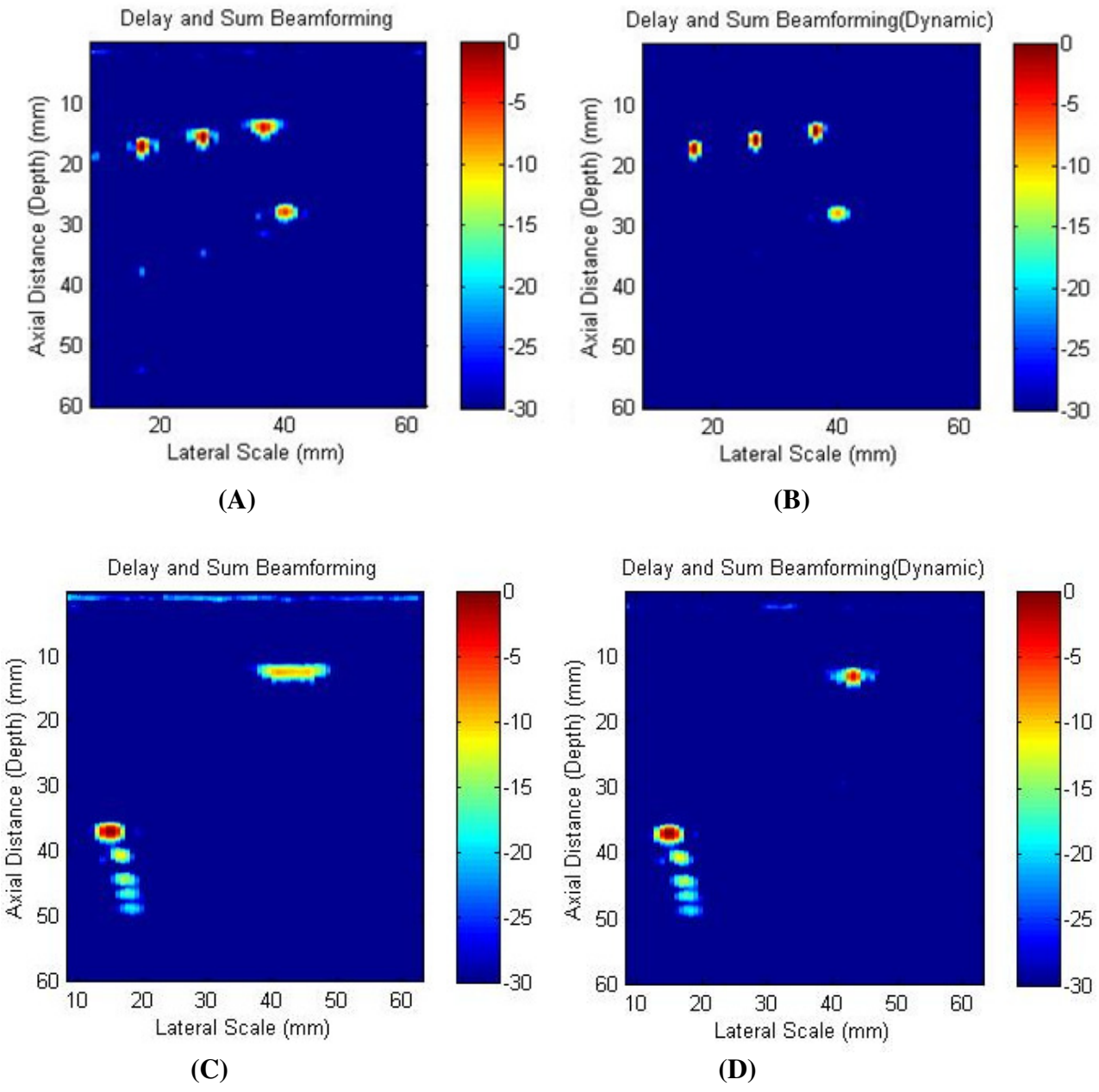

Figure 58 - Images for Test_1 and Test_2 – Test_1 (Fixed focus (A) and Dynamic focus (B)) and Test_2 (Fixed focus (C) and Dynamic focus (D))

Signal to noise (SNR) ratio analysis

We noticed that the advantage of dynamic focus over the fixed focus is that the signal to noise ratio (SNR) of the dynamic focus is increased as compared to fixed focus beamforming as shown in Table 26. This gave better image quality and good resolution. Resolution defines the smallest possible object that can be correctly seen by the system.

$$\mathbf{SNR\ (dB) = 20 * \log_{10}\left(\frac{\mu_{Region}}{\sigma_{Region}}\right)}$$

Table 26 - Signal to noise ratio (SNR) for Test 1 and Test_2

Fixed Focus Beamformer SNR (dB)		**Dynamic Focus Beamformer SNR (dB)**	
Test_1	**Test_2**	**Test_1**	**Test_2**
16.02	15.68	19.96	21.39

Table 26 shows that the mean signal to noise ratio for the dynamic focus is higher than fixed focus beamforming.

6.4.2 Discussion – Dynamic focusing

The tests and results in the section 6.4 show that the dynamic focus delay and sum beamformer has better image quality as compared to the fixed focus delay and sum beamforming.

Fixed focus delay and sum beamforming image quality is dependent on the focal point and some area may be completely unfocused and it may be even impossible to identify a particular shape. The results shown for the fixed focus are carried using the focal points which best show all the holes in the image.

Dynamic focus delay and sum beamforming shows that all the samples/pixels are equally focused not requiring the need for any focal point decision and thus every point remains focused in the image if the focal depth remains same.

Dynamic focus delay and sum beamforming with improved signal to noise ratio and improved image quality is better than the fixed focus delay and sum beamforming with decreased signal to noise ratio and unfocused regions.

6.4.3 Hardware implementation constraints of dynamic focusing

The hardware implementation of the dynamically focused delay and sum beamformer involves:

- Delay calculations per transmission.
- Delay profile storage space.
- Changes in FIFO Structure.
- Memory generator modification.

Delay calculations per transmission

In the dynamic focusing, the delays need to be generated for every sample depth. These delays are generated using the same delay profile as in the fixed focus delay and sum beamformer. For the fixed case, the focal point in a single point, while in the dynamic case, the full focal depth is divided into the number of focal regions. Full dynamic focusing is possible if every sample or pixel is focused. In the fixed focus beamforming, the delay needs to be calculated for each and every transmission. But in the dynamic focus beamforming, the same delay profile can be reused if the number of samples and the size of aperture is not varied. Thus, there is a tradeoff between storage and the number of delay profile calculation per transmission.

Delay profile storage

The delay profile for the dynamic focus case needs to be stored in memory for all the samples, which was not the case in the fixed focus case. The delay profile is loaded for every transmission

for a specific aperture size in the fixed focus case. No extra memory is required in fixed focus beamforming, but a memory of size Aperture_Size * Maximum_No_Samples is required for the dynamic focus beamforming.

Changes in the FIFO structure

In the fixed focus delay and sum beamforming, the delay was included using the First In First Out (FIFO) data structure. The delays are incorporated using a delay down counter and when the delays have been counted, the FIFO is read to access the samples in sequence.

In the dynamic focus delay and sum beamformer, the UARP beamformer needs to be modified to access any sample from the sample stream of a data channel. This results in the replacement of the FIFOs with a memory where any address can be accessed at anytime, without the restriction that the data sample first input to the FIFO memory will be read out first.

Memory generator modification

The memory generator for the fixed focus delay and sum beamformer was designed to increment the FIFO write address as soon as the beamformer is enabled and the read address as soon as the delay counter is count down to zero.

The dynamic focus delay and sum beamformer needs to generate the address according to the delay profile. This might not be the address accessed in the previous read operation, converting the FIFO to a memory where any location can be read anytime.

In this chapter, we showed the results and discussed them to support the need for implementation of a hardware based beamformer. Also, the advantages of dynamic focusing as compared to the fixed focusing have been highlighted. The beamformer optimization results in a significant difference in frame rate, making the system more viable for real time testing and measurements.

Chapter 7:

Conclusion

And

Future Work

7.1 Conclusion

This document proposes a resource efficient, modular, FPGA based delay and sum beamformer design for the UARP system. This design has been tested and after a number of cycles of optimization, it was seen that the frame rate increased to about 30 times as compared to the software based beamforming. Also, dynamic focus has been proposed as a future work to be included in hardware for the design. This will enhance the system resolution nd signal to noise ratio considerably.

Hardware implementation of delay and sum beamformer using STRATIX III FPGA custom board and Ultrasound Array Research Platform (UARP) shows that frame rate has increased. Shifting the beamformer from software to hardware creates no significant difference in the image quality but the gain is attractive in terms of increased beamforming speed.

The hardware UARP beamformer is implemented using fixed focusing. Dynamic focusing was explored for the same design in Matlab. The results demonstrate that the signal to noise ratio (SNR) of dynamically focused images is enhanced as compared to the fixed focus beamformed images. Also, the resolution is also increased resulting in high quality images.

7.2 Future work

The implementation of the hardware based beamformer has not only enhanced the system speed but has also opened new ways to test and design more algorithms. It will welcome more studies to be done which were not possible with the slow processing speed of software.

There are two possible direct modifications of this beamformer:

- Hardware implementation of the proposed dynamic focus delay and sum beamformer.
- Modification to the delay and sum beamformer to be utilized for the synthetic aperture imaging.

7.1.1 Hardware implementation of dynamic focus delay and sum beamformer

The dynamic focus advantages have been demonstrated with software (Matlab) results in Chapter 5. The resolution as well as the quality of the image has increased. Analysis of Chapter 4, where delay and sum beamformer implementation in hardware has been highlighted, shows that the frame rate enhances to many folds as compared to the software based beamforming. The implementation of dynamic focus, which has been shown to give better quality as well as improved signal to noise ratio as compared to the fixed focus beamforming, in hardware will enhance the system speed as well as the resolution.

The dynamic focus implementation in hardware is greatly restrained by the number of delay profiles to be generated and then loaded. This will consume lots of bandwidth and time in loading the delay profiles and also a huge amount of memory would be required in proportion to the number of samples to store these delay profiles. One of the solutions could be to implement the delay profile generator in the hardware. Also the scheme involved for the coarse delay implementation would require changes. Still this architecture supports not changing the whole design, but just changing the address generation scheme for the memory read operations.

7.1.2 Synthetic aperture imaging using delay and sum beamformer

The proposed design of the beamformer is flexible to be adapted to the new technique for imaging known as synthetic aperture imaging.

Synthetic aperture imaging is a technique where one or a number of elements are used in transmission, while all the elements receive at reception. This gives the flexibility to use any aperture size as required and any focal depth can be focused. The system becomes highly flexible to select any focal depth at runtime and any number of elements as aperture size. This scheme involves the focusing at the receive side first and then focusing for all the transmit locations. This results in a system which can focus at receive as well as transmit side at any desired focal depth even after transmission. This is different from the linear imaging where the transmit focus is fixed. The proposed design is flexible to support the above mentioned technique by being

replicated once more. The scan lines can be created and then they can be focused once more with respect to the transmitter position [40-42].

The implementation constraints involve huge storage space required. In this technique, the image is formed after every transmission as compared to a single scan line formed in the linear imaging. In synthetic aperture imaging, the image formed per transmission is of low resolution while the image formed in the linear imaging is of high resolution. All these low resolution images need to be added together to form a high resolution image. Thus, all the low resolution images need to be stored so that they can be added together to form a high resolution image. One other modification could be that after every transmission, the scan lines are made and then adjusted to transmit focus and then added to the already stored scan lines. This will require only one image space while in contrast this will increase the amount of manipulations per transmission. The actual tradeoffs need to be analyzed after hardware implementations of both the variations.


NATIONAL UNIVERSITY OF SCIENCES & TECHNOLOGY
PAKISTAN
Defining futures

References

References

[1] Ronald A. Mucci, “A Comparison of Efficient Beamforming Algorithm”, IEEE Transactions on Acoustics, Speech, and Signal Processing, VOL. ASSP-32, No.3, June 1984.

[2] Kai E. Thomenius, “Evolution of Ultrasound Beamformers”, IEEE Ultrasound Symposium, 1996.

[3] Roger G. Pridham and Ronald A. Mucci, “A Novel Approach to Digital Beamforming”, Acoustical Society of America, 1978.

[4] Nkwachukwu Chukwuchekwa, “Interpolation Technique in Multirate Digital Signal Processing for Efficient Communication Systems”, International Journal of Academic Research, Vol. 3, No. 2, March 2011, Part III.

[5] Ronald W. Schafer and Lawrence R. Rabiner, “A Digital Signal Processing Approach to Interpolation”, Proceedings of the IEEE, VOL. 61, No. 6, June 1973.

[6] Fabio Kurt Schnieder, Yang Mo Yoo, Anup Agarwal, Liang Mong Koh and Yongmin Kim, “New Modulation Filter in Digital Phase Rotation Beamforming”, Ultrasonics 44, 2006, 265-271.

[7] Anup Agarwal, Fabio Kurt Schnieder, Yang Mo Yoo and Yongmin Kim, “Image Quality Evaluation with a New Phase Rotation Beamformer”, IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control, VOL. 55, No. 9, September 2008.

[8] Jorgen Arendt Jensen, Svetoslav Ivanov Nikolov, Kim Lokke Gammelmark and Morten Hogholm Pedersen, “Synthetic Aperture Ultrasound Imaging”, Ultrasonics 44 (2006) e5-e15.

[9] V. Valimaki and T. I. Laakso, “Principles of Fractional Delay Filters”, IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP’00), June 2000.

[10] Jun- Young Lee, Hyeong-Seok Kim, Jae-Hee Song, Jeong Cho and Tai-Kyoung Song, “A Hardware Efficient Beamformer for Small Ultrasound Scanners”, IEEE Ultrasonics Symposium, 2005.

[11] S. Ricci, L. Bassi, A. Cellai, A. Ramalli, F. Guidi and P. Tortoli, “ULA-OP: A fully open Ultrasound Imaging/ Doppler System”, Acoustic Imaging, Vol. 30, ISBN: 978-90-481-3254-6, 2011.

[12] E. Boni, L. Bassi, A. Dallai, F. Guidi, A. Ramalli, S. Ricci, J. Housden and P. Tortoli, “A reconfigurable and programmable FPGA-based system for nonstandard ultrasound methods”,

IEEE transactions on Ultrasonics, Ferroelectrics and Frequency Control, Issue 7, Pg. 1378-1385, 2012.

[13] A. A. Assef, J. M. Maria, F.K. Schnieder, E.T. Costa and V. L. S. N. Button, "FPGA-based real-time digital beamformer for Ultrasonic elastography", Health Care Exchanges (PAHCE), 2011.

[14] Gi-Duck Kim, Changhan Yoon, Sang-Bun Kye, Youngbae Lee, Jeoun Kang, Yangmo Yoo and Tai-Kyong Song, "A single FPGA-based portable ultrasound imaging system for point-of-care applications",IEEE transactions on Ultrasonics, Ferroelectrics and Frequency Control, Vol. 59, Issue 7, Pg. 1386-1394, 2012.

[15] Weibao Qiu, Yangyan Yu, Fu Keung Tsang and Lei Sun, "An FPGA-based open platform for Ultrasound biomicroscopy",IEEE transactions on Ultrasonics, Ferroelectrics and Frequency Control, Vol. 59, Issue 7, Pg. 1378-1385, 2012.

[16] Freeman et. al., "Delta-Sigma Oversampled Ultrasound Beamformer with Dynamic Delays", IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control, vol. 46, no. 2, march 1999.

[17] Yang et. al., "A Comparison of the Lookup Table and On-The-Fly Calculation Methods for the Beamforming Control Unit", 23rd Technical Conference on Circuits/Systems, Computers and Communications. 2008.

[18] Camacho et. al., "A Strict-Time Distributed Architecture for Digital Beamforming of Ultrasound Signals", IEEE Transactions on Instrumentation and Measurement Vol 59. No. 10. Oct 2010.

[19] Vogel et. al., " Transducer Design Considerations in Dynamic Focusing", Ultrasound in Med. & Biol. Vol 5. Pp 187-193. 1979.

[20] Sohn et. al., "Time-sharing bilinear delay interpolation for ultrasound dynamic receive beamformer", Electronics Letters 20th January 2011, Vol. 47, No. 2.

[21] Harrison & Zemp, "The Applicability of Ultrasound Dynamic Receive Beamformers to Photoacoustic Imaging", IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control, vol. 58, no. 10, Oct 2011.

[22] Jensen et. al., "Ultrasound Research Scanner for Real-time Synthetic Aperture Data Acquisition", IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control, vol. 52, no. 5, may 2005.

[23] Jensen et. al., “Performance of SARUS:A Synthetic Aperture Real-time Ultrasound System”, pp. 305-309, Proceedings Ultrasonics Symposium 2010.

[24] Nikolov et. al., “Fast Parametric Beamformer for Synthetic Aperture Imaging”, pp. 1755-1767. IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control, Vol. 55, No. 8, Aug 2008.

[25] Park et. al., “A Real-time Synthetic Aperture Beamformer for Medical Ultrasound Imaging”, pp. 1992- 1995, Proceedings Ultrasonics Symposium 2010.

[26] Ihor Trots, Andrzej Nowicki and Marcin Lewandowski, “Synthetic Transmit Aperture Method in Medical Ultrasonic Imaging”, World Academy of Science, Engineering and Technology 64 2010.

[27] Ultrasound Array Research Platform (UARP), University of Leeds, UK.

[28] Harry L. Van Trees, “Optimum Array Theory”, Part IV of Detection, Estimation and Modulation Theory.

[29] Mawia A. Hassan, Abou-Bakr M. Youssef, Yasser M. Kadah, “Modular FPGA-Based Digital Ultrasound Beamforming”, IEEE 2011.

[30] Piero Tortoli, Luca Bassi, Enrico Boni, Alessandro Dallai, Francesco Guidi and Stefano Ricci, Member, “ULA-OP: An Advanced Open Platform for Ultrasound Research”, IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control, vol. 56, no. 10, October 2009.

[31] Chang-Hong Hu, Fan Zheng, Yi Huang, Jonathan M. Cannata, K. Kirk Shung and Ping Sun ,“Design of a 64-channel Digital High Frequency Linear Array Ultrasound Imaging Beamformer on a Massively Parallel Processor Array”, 2008 IEEE International Ultrasonics Symposium Proceedings.

[32] ChangHong Hu, Lequan Zhang, Jonathan M. Cannata, Jesse Yen, K. Kirk Shung, “Development of a 64 channel ultrasonic high frequency linear array imaging system”, Ultrasonics 2011.

[33] D. Lertsilp, S. Umchid, U. Techavipoo and P. Thajchayapong, “Improvements in Ultrasound elastography using dynamic focusing”, Biomedical Engineering International Conference (BMEiCON), 2011.

[34] T. Harrison and R. J. Zemp, “The applicability of Ultrasound dynamic receive beamformers to photo acoustic imaging”, IEEE transactions on Ultrasonics, Ferroelectrics and Frequency Control, 2011.

[35] Cobbold, R.S.C., “Foundations of Biomedical Ultrasound”, Oxford Univ. Press, New York 2007.

[36] D.A. Guenther and W.F. Walker, “Optimal apodization design for medical ultrasound constrained least squares part I: theory”, IEEE transactions on Ultrasonics, Ferroelectrics and Frequency Control, 2007, Vol. 54, Issue: 2, Pg. 332-342.

[36] D.A. Guenther and W.F. Walker, “Optimal apodization design for medical ultrasound constrained least squares part II: simulation results”, IEEE transactions on Ultrasonics, Ferroelectrics and Frequency Control, 2007, Vol. 54, Issue: 2, Pg. 343-358.

[38] Chi Seo and Jesse Yen, “Sidelobe suppression in ultrasound imaging using dual apodization with cross-correlation”, IEEE transactions on Ultrasonics, Ferroelectrics and Frequency Control, 2008, Vol. 55, Issue: 10, Pg. 2198-2210.

[39] Minghui Li and Gordon Hayward, “Ultrasound nondestructive evaluation (NDE) imaging with transducer arrays and adaptive processing”, Sensors 2012, doi:10.3390/s120100042.

[40] M. A. Hassan, A. M. Youssef and Y. M. Kadah, “Embedded digital signal processing for digital ultrasound imaging”, 28th National Radio Science Conference (NRSC), 2011.

[41] B. Y. S. Yiu, I.K.H. Tsang and A. C. H. Yu, “Real-time GPU-based software beamformer designed for advance imaging methods research”, IEEE Ultrasonics Symposium (IUS), 2010.

[42] J. Kortbek, J.A. Jensen and K. L. Gammelmark, “Synthetic Aperture Sequential Beamforming”, IEEE Ultrasonics Symposium (IUS), 2008.

[43] J. A. Jensen, J. Kortbek, S. I. Nikolov, M. Hemmsen and B. Tomov, “Implementation of Synthetic Aperture Imaging in Medical Ultrasound: The Dual Stage Beamformer Approach”, 8th European Conference on Synthetic Aperture Radar (EUSAR), 2010.

[45] F. K. Schneider, A. Agarwal, M. Y. Yang, T. Fukuoka and K. Yongmin, “A Fully Programmable Computing Architecture For Medical Ultrasound Machines”, IEEE Transactions on Information Technology in Biomedicine, 2010.

[46] F. Stuart Foster, James Mehi, Marc Lukacs, Desmond Hirson, Chris White, Chris Chaggares and Andrew Needles, “A New 15–50 MHz Array-Based Micro-Ultrasound Scanner for

Preclinical Imaging", Ultrasound in Medicine and Biology, Vol. 35, Pg. (1700-1708), October 2009.

[47] J. A. Jensen, O. Holm, L. J. Jensen, H. Bendsen, H. M. Pedersen, K. Salomonsen, J. Hansen and S. Nikolov, "Experimental ultrasound system for real-time synthetic imaging", IEEE Ultrasonics Symposium (IUS), 1999.

[48] I. K. H. Tsang, B. Y. S. Yiu, D.K. H. Cheung, H. C. T. Chiu, C. C. P. Cheung and A. C. H. Yu, "Design of a multi-channel pre-beamform data acquisition system for an ultrasound research scanner", IEEE International Ultrasonics Symposium (IUS), 2009.

[49] C. Fritsch, M. Parrilla, A. Ibanez, R. C. Giacchetta and O. Martinez, "The progressive focusing correction technique for ultrasound beamforming", IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control, 2006.

[50] A. B. Abche, A. Maalouf, R. Ayoubi, E. Karam and A. M. Alameddine, "An FPGA Implementation of a High Resolution Phase Shift Beamformer", IEEE Signal Processing and Communications (ICSPC), 2007.